© 2020 IJSRST | Volume 5 | Issue 7 | Print ISSN : 2395-6011 | Online ISSN: 2395-602X

4th National Conference on Advances in Engineering and Applied Science
Organized by : Anjuman College of Engineering and Technology (ACET) Nagpur, Maharashtra, India, In association with
International Journal of Scientific Research in Science and Technology

Linear Gantry Robot Control System

Siddhi Date[1], Shreya Makwana[2], Divya Mewada[3], Dr. Sheshang D. Degadwala[4]

[1-3]U.G. Scholar, Computer Engineering, Sigma Institute of Engineering Vadodara, Gujarat, India

[4]Associate Professor Computer Engineering, Sigma Institute of Engineering Vadodara, Gujarat, India

ABSTRACT

A Linear Robot comprises of a controller mounted onto an overhead framework that permits development over a flat plane. Gantry are likewise called Cartesian or straight robots. Direct Gantry Robot will be made which will pick and place the things in ideal spot. Initially, a robot with two hub primarily x-hub and y-hub will be made utilizing different electronic instruments after that one programming will be made will provide guidance to the robot to move according to the necessity. The product will be configuration utilizing python programming and will be constrained by Linux working framework and furthermore by Raspberry pi's GPIO header. By making this framework the effectiveness will expand, takes less floor space, will be lower in expense and it can accomplish substantial payloads.

Keywords : Pick and place movement, Cycle Time, Estimation, Axes, Control system, Path planning, Driver, Linear motion.

I. INTRODUCTION

With its two tomahawks, the straight gantry robot LGR is prefect for stacking and emptying machines and installations, just as taking care of parts between various stations. The framework, made up of aluminium and steel tomahawks can ship heaps of up to 1500 kg and has stroke of up to 40m per carriage. The all out length of the robot is up to 100m.

II. INTERCONNECTIONS IN MACHINE

1) Input and output

Raspberry Pi is a sort jack of all trades when it comes to being a single board computer based on the Arm processor. The general purpose input output(GPIO) pins on the Raspberry pi speak and listen to the outside world and can be controlled or programmed. Each pin has a specific role. Its hardware has a limited number of digital I/O pins.

A discrete signal (digital signal) supplied to Raspberry Pi is known as digital input. This signal can be generated manually using a push button switch.

Push button switch is a switch which provides connectivity between its terminals when pressed. When the button is released terminals get disconnected.

Python coding with Raspberry Pi connects your project to the real world.[5]

2) Coordinates of the input

A coordinate system with axes or dimensions that are intersecting and perpendicular(orthogonal). The origin is the intersection of the three coordinates-x,y

and z axes that locate a point in space and measure its distance from any of three intersecting coordinate planes. The coordinates are used to identify points for the positioning of an end-effector.

A python code is used to find the coordinates of the input. These coordinates are further segmented to get a precise and smooth curves and lines. The coordinates found in such a way that it can be easily implemented by the stepper motor.

Sometimes it is impossibles to get the precise curve, in such cases the point nearest to the original path of the curve is used to get a curve like structure. More the precision and less the degree of the motor used, smother the curve generated.

This method is quite similar to how pixels are used to get better resolution.[5]

III. CONTROLLED PATH

This robot is thought its motions according to capabilities inherent in point-to-point and continuous path systems: robot axes need not be specified, while the desired contour, acceleration, and deceleration are automatically generated. Special features of this kind of robot are path computations, programmable velocities, coordinated axis motions, abilityto make changes in end-effector length.

Optimization Method by Dynamic Programming:

Suppose the time taken by the Cartesian robot in one cycle time is T. Then our is to minimize cycle time Hence the Optimization Function is as follow.

$$Tmin = \min \left[\sum_{i=1}^{17} ti \right]$$

Where reduced Robot Cartesian Cycle time is depends only on distance between patch1 to patch2. Because we cannot make any anywhere else Hence:

$$Tmin = \min \left[\sum_{i=2}^{6} ti \right]$$

Where t2, t4, t6 are constant, because these are the necessary distance which have to travel by Cartesian robot.[5]

Now our goal reduce the is only to reduce t3 and t5. Only t3 and t5 are varying. Hence the optimization function is only depends upon t3 and t5.Means our goal is to minimize the distance travel by the Cartesian robot in time t3 and t5. So the final optimization of function is:

Tmin =min(t3+t5)

IV. CONSTRAINTS

We have to minimize the distance travel during time t3 and t5 such that the distance travel in these time should be greater then the height of stopper placed there.

Suppose the height of stopper is hs and distance travel in time t3 or t5 is d.Then our constraints for minimization function is:d>hs

(d=Distance Travel in Time t3 or t5,hs=Height of Stopper)

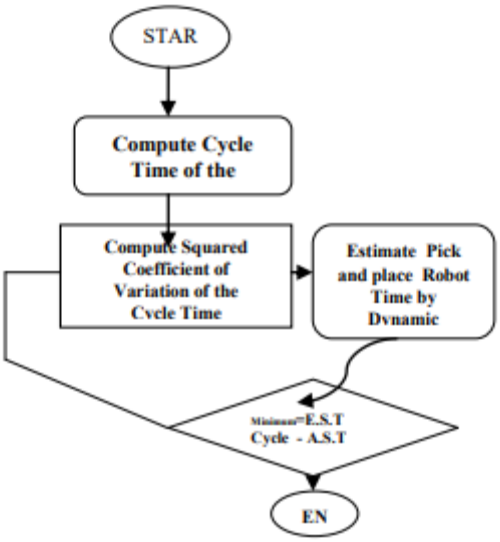


Figure 1. Estimating Parameters of Cycle Time[5]

V. SIMULATION OF ACTUAL & ESTIMATED TIME AND DISTANCE

Parameter of robot movement to pick and place crown gear to evaluate of reducing time parameter of Cycle in comparison actual robot working cycle by calculating Matlab Software.[5]

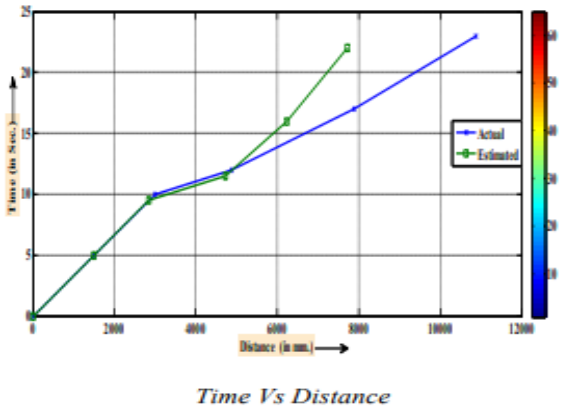


Figure 2. Actual Vs Estimate(1)

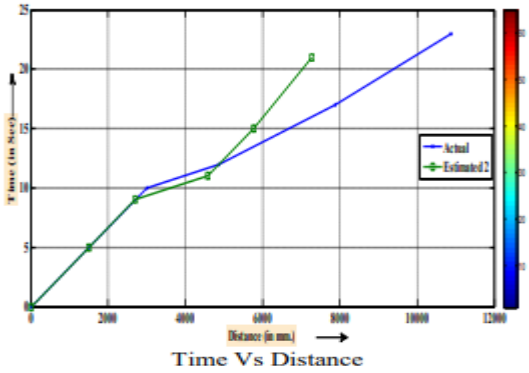


Figure 3. Actual Vs Estimate(2)

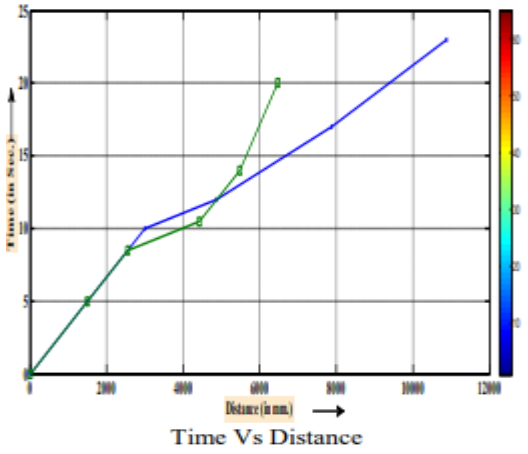


Figure 4. Actual Vs Estimate(3)

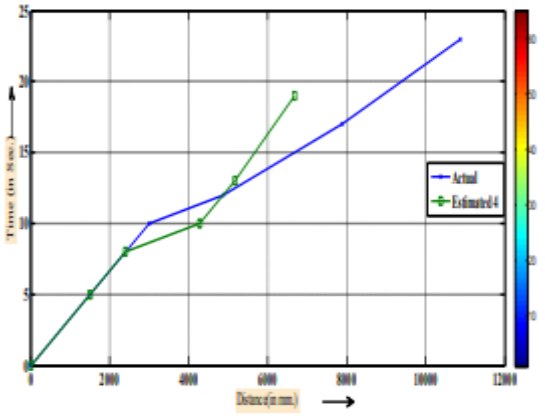


Figure 5. Actual Vs Estimate(4)

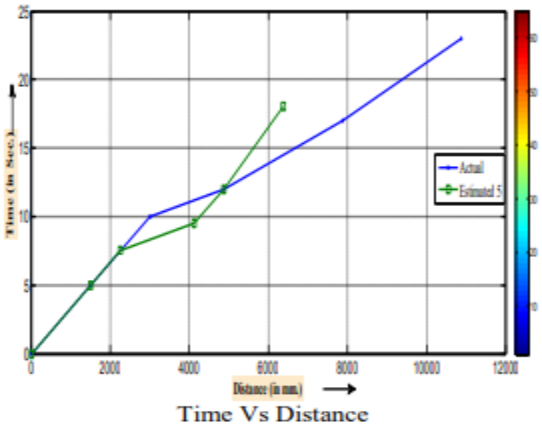


Figure 6. Actual Vs Estimate(5)

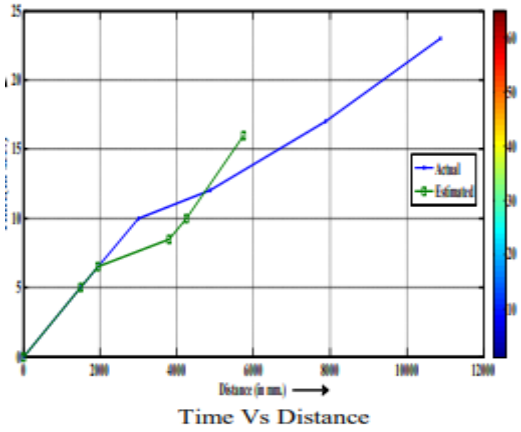


Figure 7. Actual Vs Estimate(6)

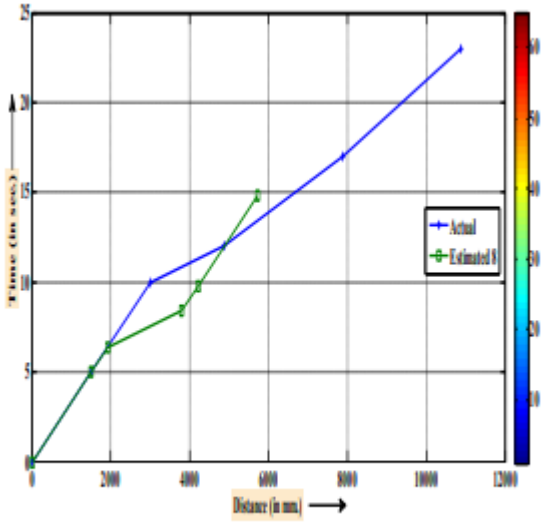


Figure 8. Actual Vs Estimate(7)

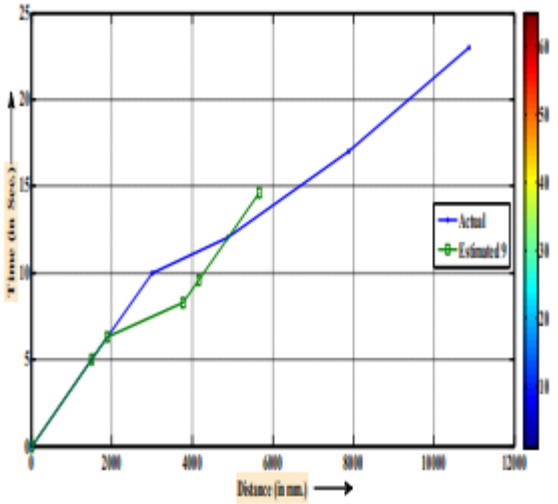


Figure 9. Actual Vs Estimate(8)

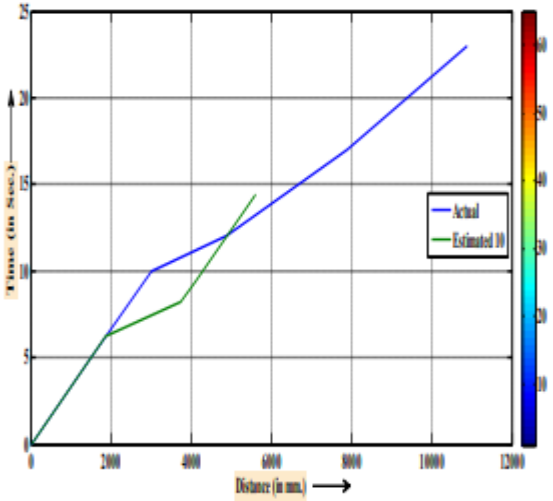
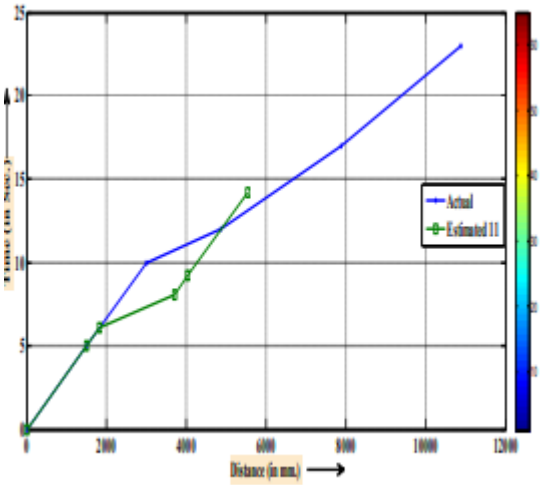


Figure 10. Actual Vs Estimate(9)



Time Vs Distance

Figure 11. Actual Vs Estimate(10)

TABLE 1. Comparison data Actual Vs Estimate Time

Actual time (in second)	Estimated time (in second)	Reduced time(in second)
15	15	0
5	5	0
5	1	4
2	2	0
5	1	4
6	6	0
6	6	0
9	9	0
5	5	0
3	3	0
2	2	0
3	3	0
3	3	0
1	1	0
2	2	0
1	1	0
7	7	0
80	72	8

VI. PATH PLANNING USING BEZIERCURVES

Bezier curve is a space curve, which is credited to Pierre Bezier of the French car firm Renault. Unlike other type of curves like polynomials or cubic spines, Bezier curve does not pass through all the data points used to define it. The points that are used to define a Bezier curve are called control points. A polygon that can be drawn through these control points is known as Bezier polygon. Bezier curves is contained within convex hull of the defining polygon. The turning points are the points where the slope of the curve changes its sign. Bezier curves have fewer turning points so that it is smoother that cubic spines. The first first and the last points on the curve are coincident with the first and last control points. The tangent vectors at the end points of the Bezier curve are directed along the first and last span of the polygon. The radius of curvature of the Bezier curve varies smoothly from the starting point to the end point because of its continuous higher order derivatives.[1]

The three points P0, P1, P2 are the control points of the quadratic Bezier segment. On the images these points are connected with straight lines. P0 and P2 are the endpoints of the curve, P1 (marked with ×) usually is not on the curve. The formula

$$B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2, \quad t \in [0,1]$$

is parametric, that is there are two expressions in terms of parameter t, that define x(t)and y(t):

Example for a=3:

$$B_x(t) = (1-t)^2 P_{0x} + 2(1-t)tP_{1x} + t^2 P_{2x},$$
$$B_y(t) = (1-t)^2 P_{0y} + 2(1-t)tP_{1y} + t^2 P_{2y}.$$

So for the first image we can assume that the control points looks like

$$P_0 = (0, a), \quad P_1 = (0, 0), \quad P_2 = (a, 0),$$

for some constant a.

Note, that coordinates of the points are completely independent of the parametric range [0,1].

The two images demonstrate how the curve changes, when just one endpoint is moved.

Example for a=3:

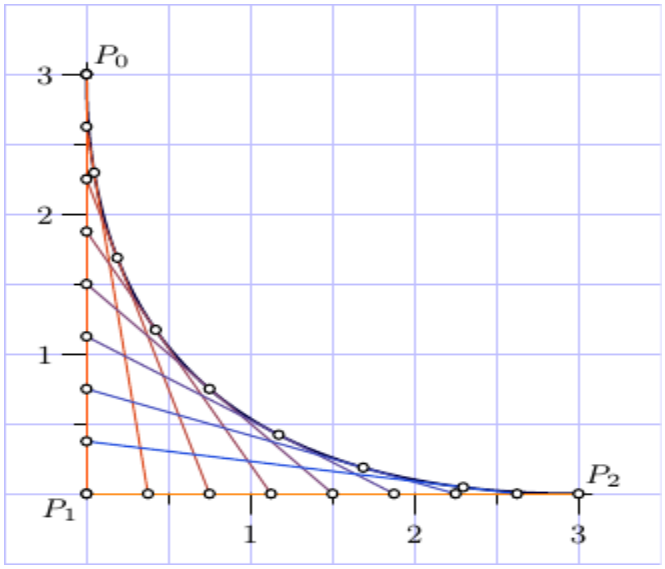


Figure 12. Bezier curve shown by points through tangents

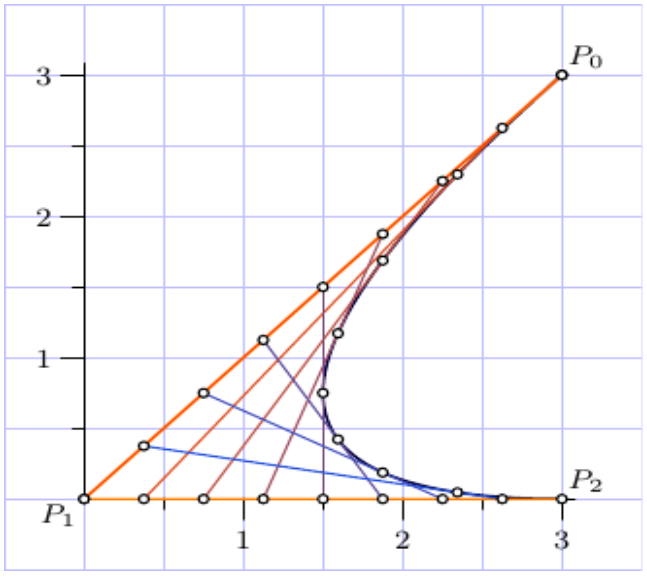


Figure 13. Bezier curve shown by points through tangents

The tangent lines on the images illustrate the process of construction of the points on a quadratic Bezier curve by means of the points on linear Bezier segments:

1. Connecting the control points P0–P1–P2, we create two linear Bezier segments: P0P1P0P1 with control points P0 and P1, and P1P2 with control points P1 and P0,
2. For any t∈[0,1], find points ut,vton the line segments P0P1and P1P2

$$u_t = P_0(1-t) + P_1(t),$$
$$v_t = P_1(1-t) + P_2(t).$$

Next, find the point wtwt on the linear Bezier segment is utvt with control points ut and vt.

$$w_t = u_t(1-t) + v_t t,$$

and the point wt is the point on the quadratic Bezier curve.

In the image, the curve seems to be the envelope of the lines whose x and y intercepts add to 11. These lines are given by

$$y = y_0 \left(1 - \frac{x}{1-y_0}\right) ,$$

where y0 is the y intercept.

The envelope maximizes y with respect to y0 for given x. Setting the derivative of y with respect to y0 to zero yields

$$1 - \frac{x}{1-y_0} - \frac{y_0 x}{(1-y_0)^2} = 0$$

and thus y0=1–√x. Substituting this into the equation for the lines yields y=(1–√x)^2, or in more manifestly symmetric form,

√x+√y=1

This can also be written parametrically as:

(x,y)=(sin^4t,cos^4t).

DRIVER A4988

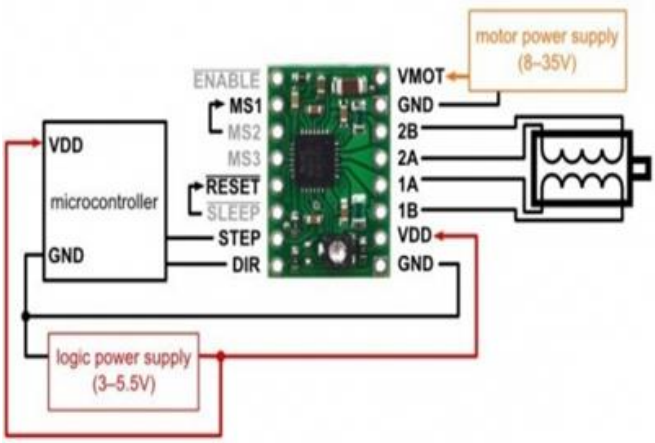


Figure 14. Driving A4988 driver with NEMA-17 stepper motor

The A4988 is a complete Micro stepping Motor Diver with built-in translator for easy operation. The driver has a maximum output capacity of 35V and ±2A. It can operate bipolar stepper motors in full-, half-, quarter-,eight-, and sixteenth-step modes.

The driver requires a logic supply voltage(3-5.5V) to be connected across the VDD and GND pins and a motor supply voltage(8-35V) to be connected across VMOT and GND. These supplies should have appropriate decoupling capacitors close to the board, and they should be capable of delivering the expected currents (peaks up to 4A for the motor supply).

Stepper motors typically have a step size specification (e.g. 1.8 degree or 200 steps per revolution), which applies to full steps. A microstepping driver such as the A4988 allows higher resolutions by allowing intermediate step locations, which are achieve by energizing the coils with intermediate current levels. For instance, driving a motor in quarter-step mode will give the 200-step-per-revolution motor 800 microsteps per revolution by using four different current levels.

The resolution (step size) selector inputs (MS1,MS2, and MS3) enable selection from the five step resolutions according to the below. MS1 and MS3 have internal 100kΩ pull-down resistors and MS2 has an internal 50kΩ pull-down resistor, so leaving these three microstep selection pins disconnected results in full-step mode.For the microstep modes to function correctly, the current limit must be set low enough (see below) so that current limiting gets engaged. Otherwise, the motor will skip microsteps.

TABLE 2

Step Resolution in A4988

MS1	MS2	MS3	Microstep Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	Quarter step
High	High	Low	Eighth step
High	High	High	Sixteenth step

Each pulse to the STEP input corresponds to one microstep of the steppermotor in the direction selected by the DIR pin. Note the the STEP ans DIR pins are not puled to any particular voltage internally, so you should not leave either of these pins floating in your application. If you just want rotation in a single direction, you can tie DIR directly to VCC and GND. The chip has three different inputs for controlling its many power states:RST,SLP, and EN.

To achieve high step rates, the motor supply is typically much higher than would be permissible, without active current limiting. For instance, a typical stepper motor might have a maximum current

rating 1A with a 50 coil resistance, which would be indicate a maximum motor supply of 5V.Using such a motor with 12V would allow higher step rate, but the current must actively be limited to under 1A to prevent damage to the motor.

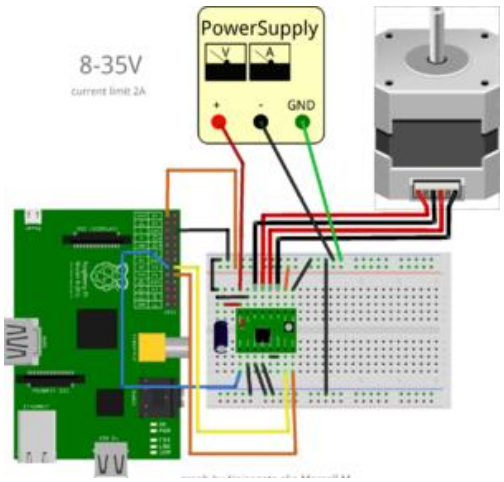


Figure 15. Driving A4988 driver with NEMA-17 stepper motor using Raspberry pi 3 b+

The above diagram represents the connection of stepper motor to A4988 driver using Raspberry Pi. Raspberry Pi GPIO Pin 7 is connected to STEP pin of A4988 Driver, GPIO Pin 11 is connected to DIR pin of A4988 Driver. Then after connecting the 4 pins of A4988 Driver with 4 wired cable Stepper Motor. The power supply to motor will be given by SMPS (Switch Mode Power Supply) of which VMOT and GND pins of A4988 Driver is connected to 12V pin and Common of SMPS.

VII. RESULT

Our research work are to be the estimated new cycle time of a robot movement is 72 sec per process, estimated no. Of Cycle increases of a robot movement is 5 cycle per hour, automatically saving a time is 8 sec by Shortest Travelling problem to reducing the travel path of robot movement.

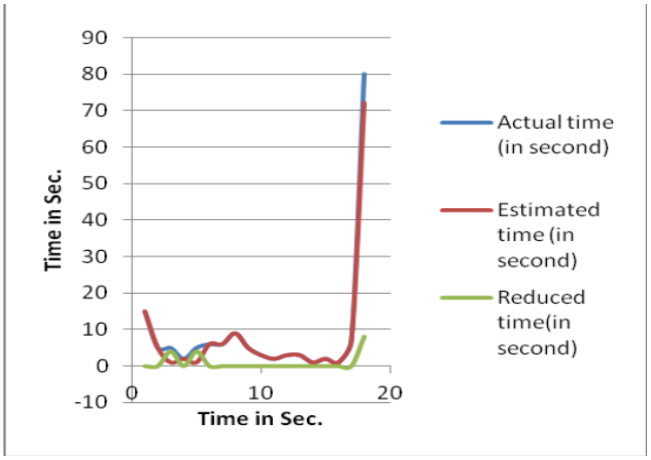


Figure 16. Comparison of Actual time Vs Estimate time

VIII. CONCLUSION

The Gantry robot scheduling problem considered in this paper can be formulated as type of dynamic programming problem. An effective path planing technique for the robots based on Bezier curves have been found compatible for the robot path planning.

IX.REFERENCES

[1] F. Zhou, B. Song, and G. Tian, "Bézier curve based smooth path planning for mobile robot," J. Inf. Comput. Sci., vol. 8, no. 12, pp. 2441–2450, 2011.

[2] P. P. Gandhi and A. K. Jain, "Optimization Performance of a Robot to Reduce Cycle Time Estimate," Int. J. Sci. Eng. Res., vol. 1, no. 1–3, pp. 357–363, 2013.

[3] N. Jovanovski and J. Kjosev, "Synchronized control of four or more stepper motors for computer numerical controled machines and 3D printers," Proc. Int. Conf. Appl. Innov. IT, vol. 6, no. 1, pp. 45–50, 2018.

[4] M. A. Khan, A. Shafi, S. A. Ahmad, S. F. H. Shah, and M. M. A. Bhutta, "Design , Manufacturing ,

Evaluation , and Analysis of Cnc Carving Machine," vol. XI, no. Ii, pp. 111–135, 2015.

[5] G. Yalçin and H. Terzioglu, "Computer Control of Z-axis of Drilling Machine that making Hole at Micron Accuracy," no. 12, pp. 61–66, 2016.

[6] M. A. E.-H. H.M.N. Fiyad, H.M.B. Metwally, "Application of Micro-Controllers for Stepper Motors Position and Speed Control: A Review," Int. J. Sci. Eng. Res., vol. Volume 10, no. 5, 2019.

[7] F. Alidoust Aghdam and S. Saeidi Haghi, "Implementation of High Performance Microstepping Driver Using FPGA with the Aim of Realizing Accurate Control on a Linear Motion System," Chinese J. Eng., vol. 2013, pp. 1–8, 2013.

[8] M. R. Mikhov and P. S. Nakov, "Stepping Motor Drive for Precise Positioning Applications," no. October, 2018.

[9] K. G. Jolly, R. Sreerama Kumar, and R. Vijayakumar, "A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits," Rob. Auton. Syst., vol. 57, no. 1, pp. 23–33, 2009.

[10] M. Elhoseny, A. Tharwat, and A. E. Hassanien, "Bezier Curve Based Path Planning in a Dynamic Field using Modified Genetic Algorithm," J. Comput. Sci., vol. 25, pp. 339–350, 2018.

[11] Y. J. Ho and J. S. Liu, "Collision-free curvature-bounded smooth path planning using composite bezier curve based on voronoi diagram," Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom. CIRA, pp. 463–468, 2009.

[12] P. Andhare and S. Rawat, "Pick and place industrial robot controller with computer vision," Proc. - 2nd Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2016, 2017.

[13] G. Kato, D. Onchi, and M. Abarca, "Low cost flexible robot manipulator for pick and place tasks," 2013 10th Int. Conf. Ubiquitous Robot. Ambient Intell. URAI 2013, pp. 677–680, 2013.

[14] H. I. Lin, Y. Y. Chen, and Y. Y. Chen, "Robot vision to recognize both object and rotation for robot pick-and-place operation," 2015 Int. Conf. Adv. Robot. Intell. Syst. ARIS 2015, pp. 1–6, 2015.

[15] P. Siagian and K. Shinoda, "Web based monitoring and control of robotic arm using Raspberry Pi," Proc. - 2015 Int. Conf. Sci. Inf. Technol. Big Data Spectr. Futur. Inf. Econ. ICSITech 2015, pp. 192–196, 2016.

[16] K. Nagai, "Cooperative Control of Dual- A m Robots for Reasonable Motion Dist rilxition," pp. 54–61, 1995.

[17] T. Bojko, "Educational Cartesian robot based on linear drives," Proc. Fourth Int. Work. Robot Motion Control. RoMoCo'04, pp. 203–208, 2004.