Technology Journal

Full Paper

DESIGN AND DEVELOPMENT OF LEAD-THROUGH PROGRAMMING METHOD USING LOW COST INCREMENTAL ENCODER FEEDBACK

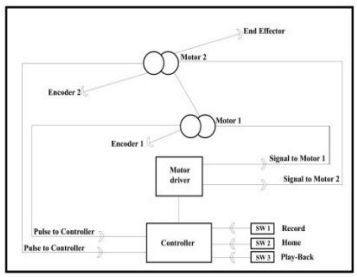
Muhammad Fahmi Miskon*, Sameh Mohsen Omer Kanzal, Muhammad Herman Jamaluddin, Ahmad Zaki Shukor, Fariz Ali

Universiti Teknikal Malaysia Melaka, Malaysia
Center of Excellence in Robotics and Industrial Automation

Article history
Received
15 May 2015
Received in revised form
1 July 2015
Accepted
11 August 2015

*Corresponding author
fahmimiskon@utem.edu.my

Graphical abstract



Abstract

Recently robots are widely used in a various field particularly in the industry. Despite this fact robot still requires an undeniable amount of knowledge from the operators or workers who deal with them. As a result, robots cannot be easily programmed if the operator or the worker is not experienced in robotics field. One of the programming methods that has been introduced to make programming task user friendly is lead-through robot programming. However, the existing lead-through programming methods still requires an amount of knowledge that is not available for most of the operators and workers. The main objective of this project is to design a lead through method for point to point robot programming using incremental encoder feedback, which can record, save and playback the robot motion while considering the accuracy and precision of the robot. To validate the method, experiments were conducted in this project, where an operator manually moves a two DOF (degree of freedom) robotic arm on a white board while the encoder feedback was recorded and later played back by the robot. Then both recorded and playback trajectories were compared and analyzed. The result shows that the played back accuracy is 96.17% for motor 1 and 97.86% for motor 2 with standard deviation of 0.9593 for motor 1 and 2.33583 for motor 2.

Keywords: Lead-through programming, incremental encoder

1.0 INTRODUCTION

1.1 Motivation

The advent of robotics technology allows manufacturers to increase yield and reduce dependency on human labors. However, small medium enterprise (SME) does not benefit much from robotic technologies due to its high capital, operation and maintenance cost. In Europe, an integrated project funded under the European Union's Sixth Framework Programme (FP6) called SMErobot™ was conducted from March 2005 to May 2009 to create a new family of SME-suitable robots and to exploit its potentials for competitive SME manufacturing.

Research and development in SMErobot™ is geared towards creating the following technical innovations: 1. Robot capable of understanding human-like instructions (by voice, gesture, graphics), 2. Safe and productive human-aware space-sharing robot (cooperative, no fences) and 3. Three-day-deployable integrated robot system (modular plug-and-produce components). Even though their objective is to lower up to a third of the current life cycle cost of automation, it is still a luxury for Malaysian SMEs.

Hence, in our effort to make robotic technology affordable to SMEs in Malaysia, a new type of user-friendly robot is developed. The robot is designed especially with the intention to reduce the capital, operation and maintenance cost. Capital cost of the

robot is reduced by using low specification sensors and actuators, such as using low cost incremental encoders rather than high cost absolute encoders. The operational cost of the robot is reduced by simplifying robot programming processes. The benefit not only limited to industrial robots application but also to other type of robots with complex motion such as in autonomous guided vehicle[1], humanoid motion [2][3][4] and others.

1.2 Background

The purpose of programming a robot is to set trajectory profile to dictate robot motion. There are three approaches to do this which are 1. Teach method, 2. Off-line method and 3. Lead through programming. Out of the three options, lead through is the most intuitive method since it involves moving robot physically while recording its position. For this reason, the robot can be programmed by any non-technical person.

Lead through programming is typically a physical movement of the robot itself by the operator, during that movement the robot records the movement of its joint and then plays it back. Since many of the industrial processes require the utilization of big and heavy robots, it is not practical to program the robots by moving them by hands. Hence, lead through programming is less favored when compared to programming using teach pendant, jogging devices or any other human machine interface (HMI) devices. However, lead through method becomes useful in programming applications that require delicate tool movements such as when teaching multiple via points along arc welding line.

1.3 State of the Art of Lead Through Method

Lead through programming has several technical challenges that are 1. Affordability, 2. Intuitiveness and teaching accuracy of the teach-pendant interface as a human machine interface (HMI), 3. Feasibility of the on-line programming due to the great number of the teaching points. The first problem can be described in terms of changing the robotic arms' location and orientation. It is desirable to move the robotic arm's tool frame rather than moving the space frame itself, for such changes in locations and orientations maneuvering robots using a keypad of joystick on the pendant is not easy and affordable to all operators, as it requires a non-deniable amount of skills and experiences [5], and they come at a cost. The second problem is regarding the teach-pendant which is one of the most common ways for programming a robot as well as a common human machine interface (HMI) [6]. Yet to program a robot using a teach pendant, the operator should set up the robot's jogging conditions, frame and motion mode, only then he can use the joystick of the teach pendant to move the robot [7]. This makes the teach-pendant programming not intuitive. The third problem is due to robotics manipulation in machining that are governed by complex work-piece [8] such as cleaning and deburring. Cleaning and deburring machines have a very complex 3D curved surface path, crucial cycle time requirements and relatively low surface accuracy. Most of the deburring operations are done manually in extremely noisy, dusty and unhealthy environmental conditions, therefore an automation for these operations is highly desirable [4].

Existing methods for generating a robots' trajectory are i. Mouse jog which raises high demand to robot motion control, as it is attached to the robot arm rigidly, furthermore it a bit complicated for unskilled operators to calibrate a 6-DOF mouse and the robot coordinate system [5], ii. Robot Puppet which needs minimum robotics understanding and it is matched to low accuracy applications such as painting and spraying [6], iii. Programming by demonstration which must satisfy the requirements for potential robot operators who have knowledge about machining, basic robotic operations such as jogging and writing a simple robot program [8] and iv. Path learning through a GUI and teach pendant which requires the operator to setup the jogging condition, frame, motion mode and steps. Additionally, it involves experimental results and simulations on dummy doll before applying it on a virtual world application. Moreover, it requires the operator's knowledge to understand, analyze and interpret the obtained information on the Graphical user interface.

Since the advantage of lead through is in simplifying the positioning of end tool in its working space, position feedback from potentiometer and absolute encoder is often used. In this project, incremental encoders' feedback is proposed for a lead-through programming method. Incremental encoder is much simpler to fabricate hence cost less when compare to absolute encoder and is note easily wear and tear like potentiometer. However, the accuracy and precision of the programming when using incremental encoders as the only feedback is expected to be less since there is a possibility that loss count occurs. Nevertheless, in this project, we would like to see the feasibility of using incremental encoders as the only feedback to the lead through programming system.

The remainder of this paper is divided as follows. In Section 2, we present the technical details of the methods. In Section 3, we validate the method by the real robot by using the developed prototype. In section 4 we discuss the obtained results. Concluding remarks are made in Section 5. Lastly, recommendation and future work is presented in section 6.

2.0 LEAD-THROUGH PROGRAMMING METHOD USING LOW COST INCREMENTAL ENCODER FEEDBACK

In this project, only small robot is developed with a selected light material (Aluminum), hence moving the robot links is not an issue. Nevertheless, accuracy and

repeatability problems as mentioned earlier still need to be solved to improve the performance of the trajectory generated.

In this paper, we proposed a new framework that enables a robot to be trained by using lead through programming using feedback from low cost incremental encoder. We proposed a new approach for the non-technical operators to deal with the robotic systems physically without any sophisticated devices that requires engineering experience and skills.

A lead-through programming method is a term used to indicate the ability of the robot to physically learn its designed trajectory path. The system used in this experiment uses a controller (Arduino DUE) and two motors to be controlled alternatively, both of the motors used are attached with rotary incremental encoders. The system compares the number of pulses given out by the encoder in both stages, record and play-back. To elaborate, during the record stage the number of pulses given by the encoder is saved and then once the play-back stage is initiated the microcontroller starts to compare the number of the pulses given during the play-back and compare it with the saved number from the record stage. The controller stops supplying voltage to either of the motors as soon as the number of pulses for both stages are equal. Figure 1 explains, in details, the system designed to program a robotic arm using the lead-through programming method.

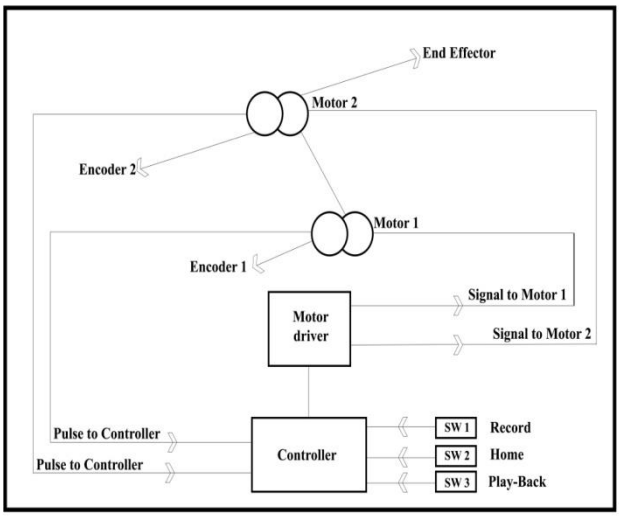


Figure 1 Lead-Through Schematic System Diagram

3.0 EXPERIMENTS

In order to test the accuracy of the system, pulses detection speed, error and precision, several types of data were gathered from the system and analyzed. X-Y coordinates (cm) of both recorded and played-back trajectories were taken, by taking the final position values on the gridded white-board as shown in Figure 2 below. Additionally, the pulses (pulse/time) given by both encoders of both motors were taken using the controller interrupts functions as shown in the schematic diagram in Figure 1. The position of both links (in degree) was obtained from the number of pulses and tabulated as well by placing a compass at the center of the joints shown in Figure 2.

As shown in Figure 2 below a 91cmX62cm white-board was used to fix the arm on and draw the final position. Additionally, two links of 30cm length each were used to link the two motors and form the two degree of freedom robotic arm. The points (22cm, 50cm) and (77cm, 50cm) on the white-board were chosen as the initial and final positions respectively. Moreover, the links were fixed to the motor's rear-shaft with two screws, one for each link, to reduce the amount of the mechanical loose encountered, despite the fact that there was still a mechanical loose due to the loose in the gearing system of the geared DC motor itself.

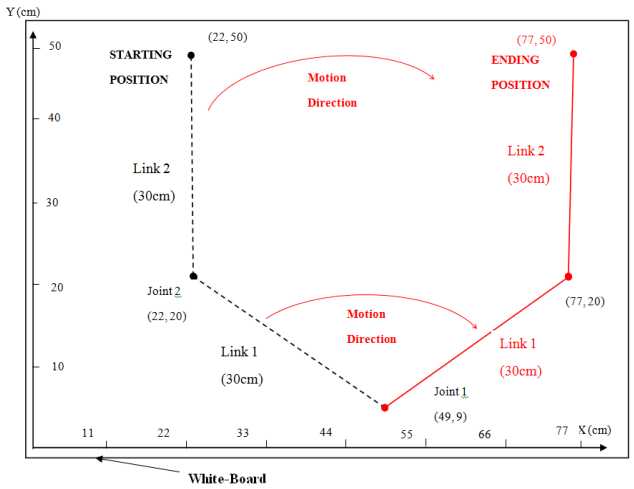


Figure 2 Planned Experimental Setup

Figure 3 below shows the experimental setup conducted as per the setup planned on Figure 2 above.

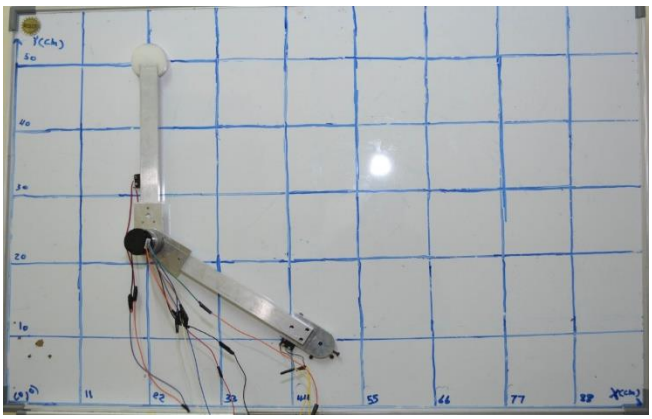


Figure 3 Real Experimental Setup

Figure 4 below illustrates the constructed circuit for the lead-through robots' programming method indicating the three buttons used for the three different trajectory generation stages, which are record, play-back and home buttons.

The controller used in this experiment is Arduino DUE board, and the two motors were used are geared DC motors (SPG30-300) with a 10 Amperes motor driver (MDD10A). The three stages of the experiment are presented in Figure 5.

Record Stage

This stage is initiated by a push button, shown as SW1 in Figure 4. The first stage is where the operator switches on the SW1 and manually moves the end-effector of the robotic arm. On the Cartesian space drawn on the white-board the final position was indicated by (22cm, 50cm) and (77cm, 50cm) was indicated as the final position.

Before starting this stage, the experiment was setup as shown in Figure 1 and 2, including the position of the arm and the power connection to both Arduino board and the motor driver, additionally the Arduino was connected to a laptop for the pulses of the encoder to be monitored.

The operator moves the end-effector of the robotic arm from the initial to the final position and the processor automatically records the movement data during the manual generation of the trajectory. When the final position is reached the operator should switch off the SW1 and move to stage two.

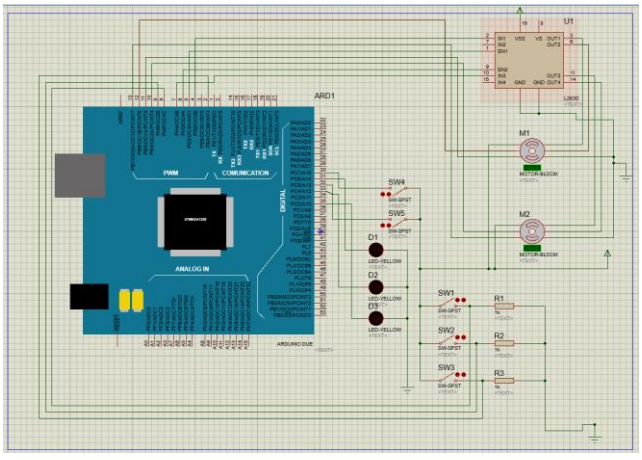


Figure 4 Constructed Circuit

Home Stage, Initial Position

In this stage the Home button is switched ON, labeled as SW2 on Figure 4, and the robotic arm automatically goes back to its initial position, the movement is ceased whenever the limit switches are turned ON by having the arm's links hitting them. In this stage the operator's only required action is to press the home button, and the arm goes back by itself to the initial position. After the robotic arm reaches its initial position the operator should depress the home button and start the next stage.

Play-Back Stage

In this stage, the operator switches on the play-back button, labeled as SW3 on Figure 4, then the robotic arm repeats the same motion made by the operator. In this stage the operator should only press on the play-back button and everything recorded will be repeated by the arm automatically, i.e. the operator's action is not required on the robotic arm to repeat the motion, it is all done automatically.

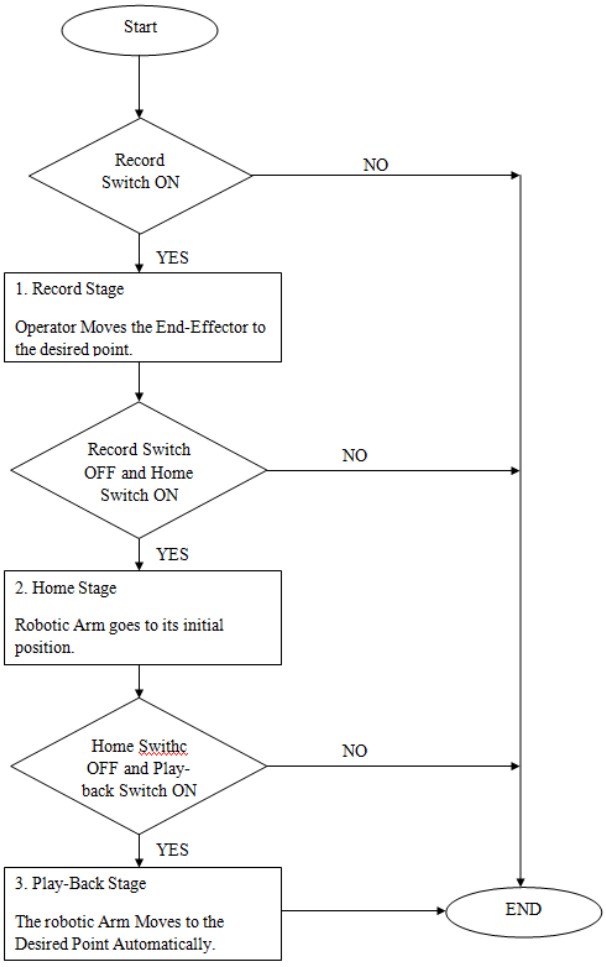


Figure 5 Stages of experiment

Method of Analysis

In order to evaluate the performance parameters from the obtained data several methods were followed to analyze the data and compare them. First of all, error was calculated using equation 3.1, where

Q_{rec} is the recorded position and Q_{played} is the played-back position.

$$\%Error = (Q_{rec} - Q_{played})/Q_{rec} \times 100\% \quad (3.1)$$

Moreover, Accuracy was considered and tabulated for each motor's trajectory using equation 3.2

$$\%Accuracy = (1 - Error) \times 100\% \quad (3.2)$$

The position of the links (in degrees) was obtained from equation 3.3

$$Q = (360 \times No.of\ Pulses/3240) \quad (3.3)$$

Precision of the system was examined by repeating the same recorded position (119.3^0) for fifty times and check the consistency of the system throughout the fifty trials.

4.0 RESULTS AND DISCUSSION

4.1 Record and Play-Back Stages Comparison

After the experimental setup was prepared the experiment was conducted and the pulses given by both encoders for the two different stages of the trajectory generation, record and play-back, along with their respective positions in respect with time were recorded and tabulated for plotting the graph shown below. Figure 6 below shows the plotted graph, which compares the trajectory generated during the record stage and the play-back stage in motor 1.

As can be seen from Figure 6 the time taken for the record stage was 3000 milliseconds depending on the speed of the operator's hand moving the robotic arm. In this experiment the speed of the operator's hand applied to motor one can be calculated as in equation 4.1 below;

$$Speed_{rec,\ motor1} = 117.9^0/3sec = 39.3^0/sec \quad (4.1)$$

On the other hand the time taken for the play-back stage was only 2000 millisecond as the motor was given only 150 pulse width modulation analog input as the full speed would give 12 rpm which an angular gives a speed as in equation 4.2 below;

$$12\ rpm \times 360^0 = 4320^0/min \times 60 = 72^0/sec \quad (4.2)$$

for this reason only 150 PWM was given to the motor which would produce the following speed shown in equation 4.3 below;

$$Speed_{played,\ motor1} = (72^0/sec \times 150)/255 = 42.4^0/sec \quad (4.3)$$

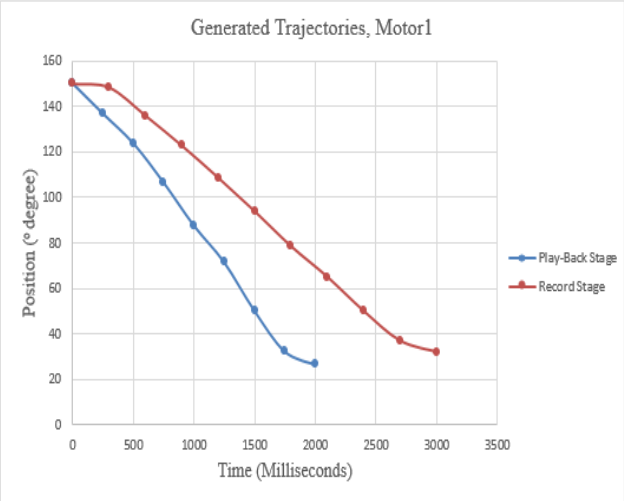


Figure 6 Generated Trajectories Comparison, Motor 1

Based on equations 4.1 and 4.3 the time difference causes such a deviation in the record and play-back graphs drawn with respect to time.

What can be seen from the graph drawn in Figure 6 rather than the time gap is the slight fluctuations in the trajectory generated during the play-back stage. These fluctuations were caused as a result of the mechanical loose resulted from a slip in the teeth of the geared DC motor. This loose caused the link to miss some of the rotation applied by the main shaft of the motor itself.

Figure 7 below shows the plotted graph from the data taken from the experiment, which compares the trajectory generated during the record stage and the play-back stage in motor 2.

As can be seen from Figure 7 the time taken for the record stage was 4050 milliseconds depending on the speed of the operator's hand moving the robotic arm. In this experiment the speed of the operator's hand applied on motor two which can be calculated as in equation 4.4 below;

$$Speed_{rec,\ motor2} = Degree\ traveled/Time\ taken = 115.8^0/4sec = 28.6^0/sec \quad (4.4)$$

On the other hand the time taken for the play-back stage was only 2600 millisecond as the motor was given only 150 pulse width modulation analog input as the full speed gives 12 rpm which gives an angular speed as in equation 4.5 below;

$$12\ rpm \times 360^0 = 4320^0/min \times 60 = 72^0/sec \quad (4.5)$$

For this reason only 150 PWM was given to the motor, which produces the following speed shown in equation 4.6 below;

$$Speed_{played,\ motor1} = (72^0/sec \times 150)/255 = 42.4^0/sec \quad (4.6)$$

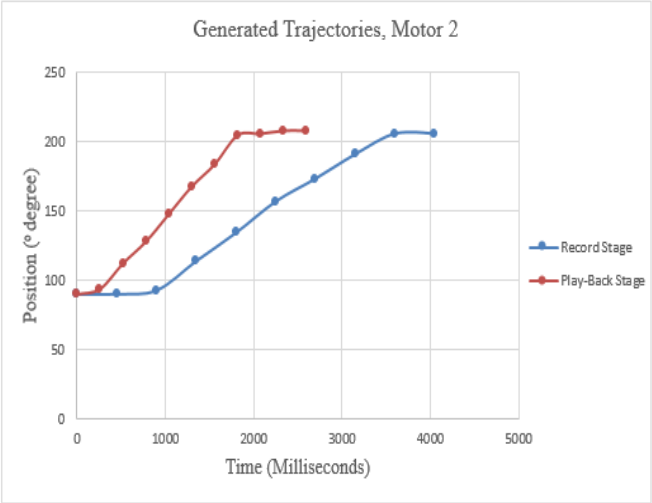


Figure 7 Generated Trajectories Comparison, Motor 2

Based on equations 4.4 and 4.6 the time difference causes such a deviation in the record and play-back graphs drawn with respect to time.

Moreover, it is obvious that in motor two the same fluctuations occurred for the mechanical loose occurred in motor one.

Figures 8-12 illustrate the robotic arm undergoing both record and play-back stages. These instantaneous positions were taken and captured with an interval of one second as shown in the figures.

As can be seen from the above shown figures, 8-12, both recorded and played-back trajectories were nearly following the same trajectory paths from second 1-4. On the other hand, at the fifth second the played-back trajectory was already at its final position unlike the recorded trajectory that entirely depends on the operator's hand-speed. However, the speed of the played-back trajectory can be manipulated using the pulse width modulation given by the controller to the motors. In this experiment 150 PWM was applied to both motors.

4.2 Errors and Accuracy

Errors and accuracy are determined based on equations 3.1 and 3.2 respectively. Each hall-effect sensor of each encoder gives three pulses per rear shaft revolution. Gear ratio of each motor is 270:1, so 810 pulses are given per one main shaft revolution.

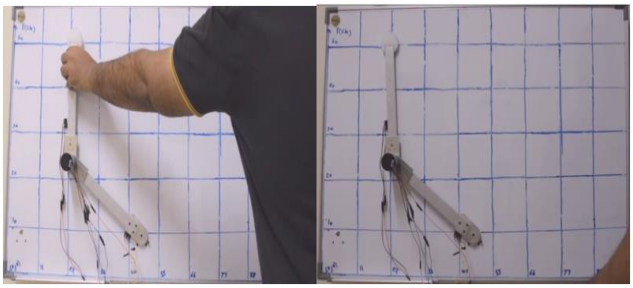


Figure 8 Record Stage, to the left, and play-back stage, to the right, at T=1 Sec

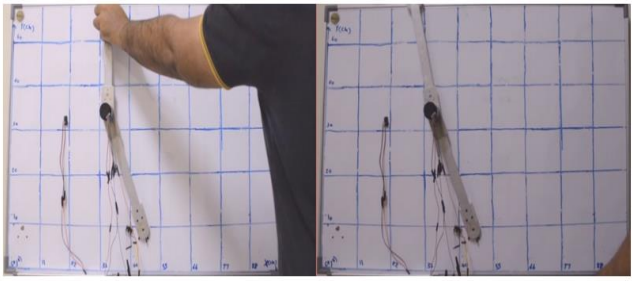


Figure 9 Record Stage, to the left, and play-back stage, to the right, at T=2 Sec

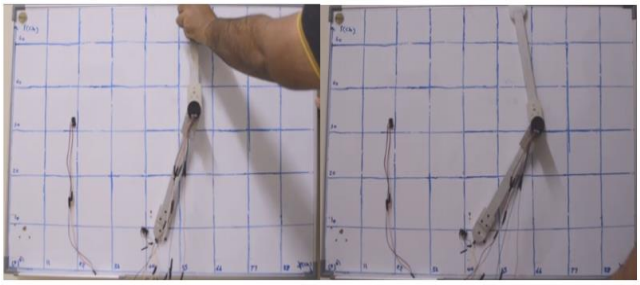


Figure 10 Record Stage, to the left, and play-back stage, to the right, at T=3 Sec

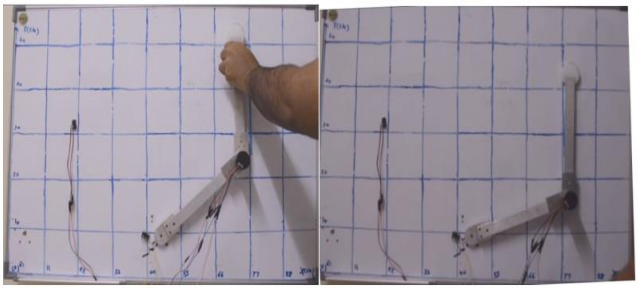


Figure 11 Record Stage, to the left, and play-back stage, to the right, at T=4 Sec

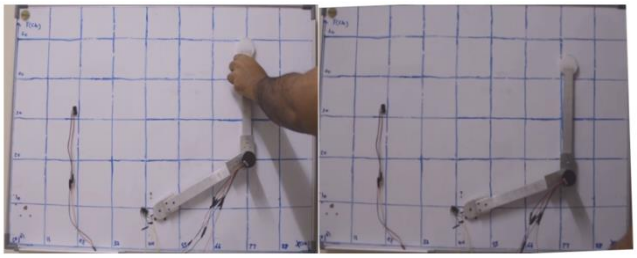


Figure 12 Record Stage at T=5 Sec

Both of the sensors output states, positive and negative, are considered, so

$$810 pulses \times 2 states = 1620 pulses/main\ shaft\ rev.$$

Since two Hall Effect sensors exist for each motor then the total number of pulses given is

$$1620 \times 2 = 3240\ pulses/main\ shaft\ rev.$$

Below Tables, 1 and 2 show the errors and accuracy of both motors.

Table 1 Motor one Error and Accuracy

Trial	Motor 1					
	Pulse record	Pulse played	Q1 record degree	Q1 played degree	Error	Accuracy
1	1064	1118	118.2	124.2	5%	95%
2	1064	1097	118.2	121.9	3.1%	96.9%
3	1064	1097	118.2	121.9	3.1%	96.9%
4	1064	1099	118.2	122.1	3.3%	96.7%
5	1064	1105	118.2	122.8	3.9%	96.1%
6	1064	1110	118.2	123.3	4.3%	95.7%
7	1064	1099	118.2	122.1	3.3%	96.7%
8	1064	1103	118.2	122.6	3.7%	96.3%
9	1064	1105	118.2	122.8	3.9%	96.1%
10	1064	1113	118.2	123.7	4.7%	95.3%
Mean	-	-	-	-	3.83%	96.17%

As can be seen from the Table 1 there is a small deviation between the recorded position $Q1_{rec}$ and the played-back position $Q1_{played}$ that gave an average error of 96.17% with an accuracy of 3.83%. As can be seen from the Table 2 there is a small deviation between the recorded position $Q2_{rec}$ and the played-back position $Q2_{played}$ that gave an average error of 97.86% with an accuracy of 2.14%.

Based on Tables 1 and 2 it was noticeable that the error occurred in motor one was a bit higher than motor two due to two main reasons. The first reason was the higher torque applied to motor one as the distance between the centre of motor one and the end-effector is 60cm, 30cm length of both links, which is double the distance between the center of the second motor and the end-effector, 30cm of the second link only. Regardless of the amount of force applied to the end-effector the error will still be higher on motor one as the torque is calculated by equation 4.7 below;

$$Torque = force\ applied \times perpendicular\ distance \quad (4.7)$$

Table 2 Motor Two Errors and Accuracy

Trial	Motor 2					
	Pulse record	Pulses played	Q2 record degree	Q2 played degree	Error	Accuracy
1	1087	1120	120.8	124.4	2.9%	97.1%
2	1087	1129	120.8	125.4	3.8%	96.2%
3	1087	1102	120.8	122.4	1.3%	98.7%
4	1087	1106	120.8	122.9	1.7%	98.3%
5	1087	1111	120.8	123.4	2.6%	97.4%
6	1087	1105	120.8	122.8	1.7%	98.3%
7	1087	1106	120.8	122.9	1.7%	98.3%
8	1087	1105	120.8	122.8	1.7%	98.3%
9	1087	1110	120.8	123.3	2.1%	97.9%
10	1087	1108	120.8	123.1	1.9%	98.1%
Mean	-	-	-	-	2.14%	97.86%

The second reason is that the weight carried by motor one is higher than the weight carried by motor two. As joint one, the first motor, carries both links of the robotic arm along with the second motor. The extra mechanical burden result in gear slips in the DC geared motor of the first joint.

4.3 Precision and Repeatability

After the consistency test of fifty trials was conducted a normal distribution curve, normal bell curve, was drawn for both motor one and motor two as shown in Figure 13 and 14 respectively.

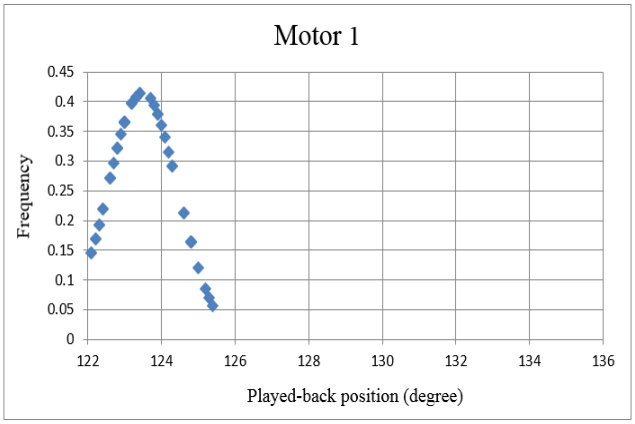


Figure 13 Normal Curve for Motor 1

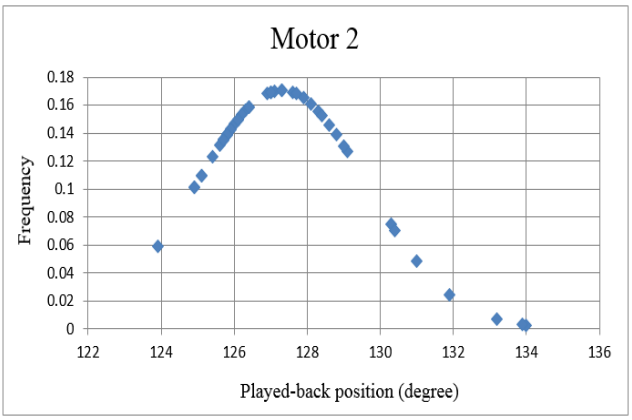


Figure 14 Normal Curve for Motor2

Based on Figure 13, the values obtained from the fifty trials were distributed evenly from 122^0 to 125.5^0 standard deviation of 0.9593 and a mean of 123.4880^0. Due to the error indicated above in Table 1 the values obtained were not distributed in a high portion in one part of the graph, which indicated that every time the experiment was conducted a slightly different value was obtained. On the other hand in Figure 14 the values obtained were intensively distributed within the range of 125^0 to 129.5^0 with a standard deviation of 2.33583 and a mean of 127.2940^0. The values of the lowest occurrence frequency were fallen to the far right and left areas of the graph in both 4.8 and 4.9.

5.0 CONCLUSION

In this project a lead-through programming method was developed using feedback from incremental encoder. Error between the recorded and played-back trajectory, accuracy, the controller speed of the encoders' pulses detection and precision were tested and results were tabulated and compared for both recorded and played-back trajectories. Based on the obtained results, it is clear that it is feasible to implement a lead-through programming method using only feedback from incremental encoder. Although the accuracy less than 98% and the standard deviation is high at more than 0.9, it still can fulfill many of the SMEs processes.

In the future, the system will be built with a wider range of applications in industrial applications by increasing the degree of freedom to up to four DOF. Moreover, optical sensors will be implemented to fulfill a full rotation of the installed geared DC motors. which will give the system the ability to fully rotate a 360^0 for each joint, as the limit switch limits the rotation of the motor to a specified angle as soon as the link hits the limit switch. Lastly, the system will be designed in fully 3D motion instead of fixing it on a white board which produces only a 2D motion.

Acknowledgement

The authors would like to thank the Ministry of Education Malaysia for funding this project under the Fundamental Research Grant Scheme and for the moral and technical support from the Universiti Teknikal Malaysia Melaka (UTeM) especially from the Robotics and Industrial Automation (RIA) research group in the Centre of Excellence in Robotics and Industrial Automation (CeRIA). Order of authorship was determined by the amount of overall contribution towards making the project successful.

References

[1] S. Sabikan, M. Sulaiman, S. N. S. Salim, and M. F. Miskon. 2010. Vision Based Automated Guided Vehicle for Navigation and Obstacle Avoidance. *Proceeding 2nd Int. Conf. Eng. ICT, Malacca, Malaysia*. 18-20.

[2] B. Bahar, M. F. Miskon, N. Abu Bakar, A. Z. Shukor, and F. Ali. 2014. Path Generation of Sit to Stand Motion using Humanoid Robot. *Aust. J. Basic Appl. Sci*. 168-182.

[3] M. B. Bahar, M. F. Miskon, N. A. Bakar, A. Z. Shukor, and F. Ali. 2014. Horizontal Distance Identification Algorithm for Sit to Stand Joint Angle Determination for Various Chair Height Using NAO Robot. In *The 8th International Conference on Robotic, Vision, Signal Processing & Power Applications*. Springer Singapore. 57-63.

[4] F. Ali, A. Z. H. Shukor, M. F. Miskon, M. K. M. Nor, and S. I. M. Salim. 2013. 3-D Biped Robot Walking along Slope with Dual Length Linear Inverted Pendulum Method (DLLIPM). *Int. J. Adv. Robot. Syst.* 10.

[5] W. Eakins, G. Rossano, and T. Fuhlbrigge. 2013. Lead-through Robot Teaching. *IEEE Conference Technology Practice Robot Application*. 1-4.

[6] L. Qi, D. Zhang, J. Zhang, and J. Li. 2009. A Lead-Through Robot Programming Approach Using a 6-DOF Wire-based Motion Tracking Device. *IEEE International Conference Robot. Biomimetics*.1773-1777.

[7] Bara.org.uk, Robot Programming Methods | BARA. [Online].From:http://www.bara.org.uk/robots/robot-programming-methods.html. [Accessed on 11- Nov- 2014].

[8] Z. Pan. 2008. Robotic Machining from Programming to Process Control. *7th World Congr. Intell. Control Autom.* 553-558.