Proceedings of the 6th WSEAS Int. Conf. on Systems Theory & Scientific Computation, Elounda, Greece, August 21-23, 2006 (pp167-173)

Design and Fabrication of a Programmable 5-DOF Autonomous Robotic Arm

SAJID GHUFFAR, JAVAID IQBAL, USMAN MEHMOOD and MUHAMMAD ZUBAIR
Department of Mechatronics
College of Electrical and Mechanical Engineering,
National University of Sciences and Technology,
Peshawar Road, Rawalpindi,
PAKISTAN.
sghuffar@yahoo.com , jiqbal-eme@nust.edu.pk, mani_de24@yahoo.com , zubairaw@yahoo.com

Abstract – This paper presents a software architecture required to build a user-friendly interface for a robot manipulator. For this purpose, during research a prototype 5-DOF robotic arm was built. The end effector of this robotic arm is a two-fingered gripper. The interface consists of two components generic to most industrial robot systems present today, first is the teach-pendant and second one is the computer based robot programming language. The main focus of the project is to develop a Language that provides robot-specific commands along with the framework common to all high level languages. A user interface has been developed that provides powerful tools for efficient control of the robot manipulator. The aspiration of this work is to present a system which is powerful, cost effective and at the same time intelligible to an average user.

Key-Words: - robotic arm, robot programming language, teach pendant, user interface, AVR microcontroller, robot kinematics

1 Introduction

Since the dawn of industrial robotics, robot software architectures and programming languages have been a topic of great interest [1]. It is imperative that a user friendly interface is built for an industrial robot system in order to gain universal recognition. Hence a lot of work had been done to develop user friendly interfaces that will at the same time provide powerful programming environment [2]. Our research focuses on developing such an interface, which is easy to use without compromising on the control that has been given to the user.

Biggs and MacDonald in [3] have divided the robot programming systems into three categories: automatic programming (learning and programming by demonstration), manual programming (text and graphics based programming) and software architectures (control methodology). Our paper involves all three categories, the text based programming, teaching and control methodology.

A robot control language defines the actions that a robot manipulator needs to perform in order to complete some task [4]. Languages such as VAL [5] and AML [6] which were developed in seventies are the earliest examples of structured robot programming languages. Since that time languages improved and evolved and as a result a wide variety of languages were developed. The problem of too many languages opened the way for using standard languages such as C for building robot programming languages. As a result language developers used the approach of adding libraries to the standard languages, for instance RCCL [7] and ARCL [4] used C, while others have used Pascal [8] for this approach in the past.

The languages mentioned above might be difficult for an average user to use. So we have used the approach of creating a new language that contains the fundamental elements of a high level language along with the robot specific commands. The syntax is made simpler and self explanatory.

The Graphical User Interface (GUI) on the computer features an Integrated Development Environment or IDE (also known as the programmers platform [9]) which allows user to work both offline and online. While offline, user can write program routines that will enable robotic am to perform wide range of tasks, varying from a simple pick and place job to long term plan oriented tasks. When online, user can either execute the programs that have been written previously or teach points to the robotic arm with the help of a set of buttons present in the GUI.

Teach pendant is part of most industrial robot systems and is used in teaching points and

programming of the robot manipulator [10]. The advantage of teach-pendant over computer is that it is a small portable box which can be used in places where setting up a computer might not be feasible. Our robotic arm is equipped with a teach pendant. It provides manual control for teaching and a set of robot commands for the programming of the robot manipulator.

The host controller of the system is the Atmel AVR ATmega8535 [11] controller which directly controls every motion of the robotic arm. The electrically erasable, programmable, read-only memory (EEPROM) of the microcontrollers (teach pendant's microcontroller and the ATmega8535) is used to save all the dynamic data such as the programs written in the teach pendant and points that are taught to the robotic arm. This made the robot more cost and power effective and increased the autonomy of the system.

The robot for our system is a 5-DOF robotic arm whose end effector is two finger hybrid gripper. This robotic arm is a prototype and low powered version of an industrial robotic arm. The goal of the project is to implement a programming system and prove it using this model.

Fig.1 shows the overview of the system. It contains four components robotic arm, host microcontroller, teach pendant and the PC. These components are discussed in detail in the next sections.

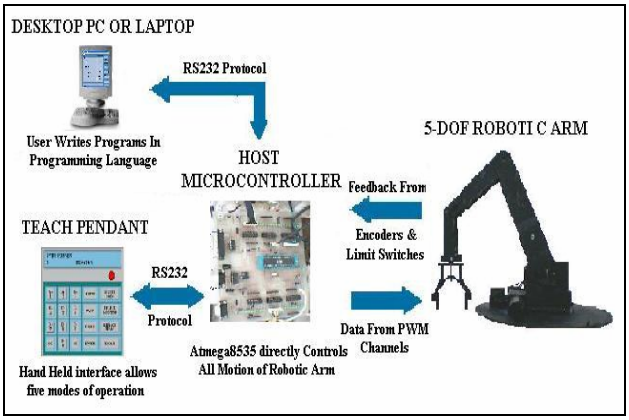

Fig.1 - System Architecture

2 The Robotic Manipulator

The robotic arm is 5-DOF which is designed and analyzed in the related software. Fig.2 shows the view of 5-DOF robotic arm. The links are made up of mild steel because of its high strength. The end effector is two-finger hybrid gripper and can grasp objects of regular shape with convenience [12]. Link lengths and workspace for all the joints are given in table 1.

Table 1: Link lengths and joint workspaces

i	L_i (cm)	θ_i^o
1	10	-90 to +90
2	40	-10 to +90
3	25	-120 to +120
4	15	-120 to +120
5		Continuous

All the joints in the structure are revolute because they are easy to implement and the structure is not bulky [13]. Three of joints form a planar manipulator, fourth is the waist for the planar arm and fifth is the rotation of the gripper. The global workspace is the set of points (x, y, z, Φ_3, Φ_4) that can be reached by the end effector for all specified orientations Φ_3 and Φ_4 of the last link and the gripper respectively.

24V servomotors have been used to drive each of the five joints. A gearbox is added to each motor to reduce the speed and increase the torque. Limit switches are also mounted on each link to prevent any motion that goes out of the limits. Limit switches also provide the hard home position of the robotic arm.

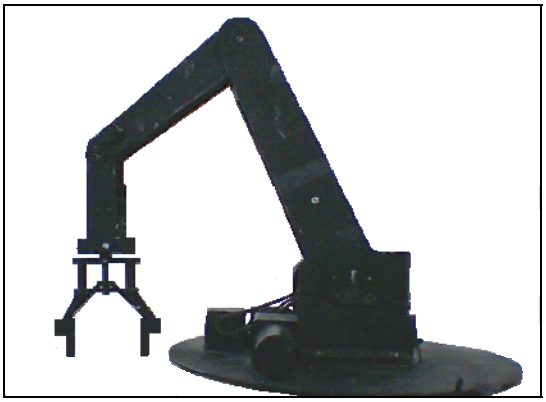

Fig.2 - View of 5-DOF Robot manipulator

3 The Host Microcontroller

The ATmega8535 controller integrates teach pendant and the computer with robotic arm. It acts as the host microcontroller and directly controls all the motion of the robotic arm.

3.1 Motor Control

ATmega8535 microcontroller is used to implement PI control algorithms for all the motors. PWM channels of the microcontroller have been used to vary the speed and direction of the motors.

Quadrature encoders have been used for the feedback, which have two channels output. One channel is used to measure angle of rotation while the other is used to measure the direction of rotation. First channel is connected to the external interrupt of the controller while the second channel is connected to the IO port. Limit switches are also directly connected to the ATmega8535 controller's IO port.

We have used single pwm channel for each motor. Hence to move a motor in both directions a multiplexer circuit is added between the microcontroller and the power amplifier circuit. The component interconnections from the host to the robotic arm as described above are shown in fig.3.

3.2 Data Storage

When a point is taught to the robotic arm the encoder counts for all the motors except the gripper motor are saved in the EEPROM of the Atmega8535 microcontroller. These encoder counts determine the position of each motor relative to the home position.

3.3 Communication

Data transmission between ATmega8535 and teach pendant or computer takes place through RS232 serial communication protocol.

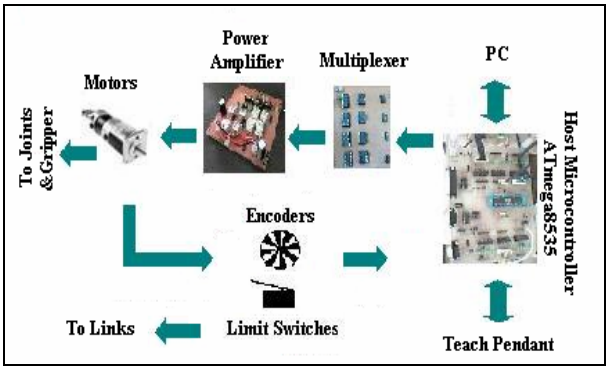

Fig.3 - Interconnection between host and the robotic arm

4 Teach Pendant Module

The teach pendant is a small portable hand held device present in all industrial robot systems. The function of the teach pendant is teaching points through manual control and programming of the robot manipulator that includes taught points and language commands [10].

4.1 Teach Pendant Architecture

Teach pendant consists of a 2 line by 16 character liquid crystal display (LCD) and a 20 key keypad in which most of the keys carry multiple functions. The microcontroller used in teach-pendant is Atmel AVR AT90S8515 microcontroller and it communicates with the host via RS232 protocol.

4.2 Modes of Operation

Teach pendant allows the user to operate in five modes:

4.2.1 Teach Mode

Teach mode provides the manual control of the manipulator, which is used to teach points. Twelve keys are provided in the keypad for motion of all motors in both the directions. During the teach mode save and delete keys are used to save and delete points respectively.

To teach a point, first the robotic arm is physically moved to the desired point with the help of the keys, then the position of each motor is stored in the host microcontroller in the form of encoder counts, when the save key is pressed.

4.2.2 Program Mode

Program mode allows the user to write programs which will be saved in the EEPROM data memory of the teach pendant's microcontroller which is Atmel AVR AT90S8515. All the commands in the instruction set of the teach pendant have been given a unique key in the keypad which is pressed to enter the command.

.

4.2.3 Run Mode

In this mode the programs that are written in the program mode are executed. LCD displays the instruction that is currently executing and the line number of the instruction in the program. The program is run online so the robotic arm performs the tasks as written in the code.

4.2.4 Step Mode

Step mode is quite similar to run mode except that after each instruction it waits for the user to press a key in order to execute the next instruction so that the user can see the effect of each instruction in the program.

4.2.5 Edit Mode

In this mode programs that have already been written can be edited. Insert, delete and replace keys are used to edit the programs.

5 Software

The robot programming language developed provides robot-specific commands along with the framework of a high level language, allowing robot programmers to use frequent features present in a common high level programming language rather than a limited set of robot commands [4]. All the commands have easy to understand and simple syntax which reduces the overall complexity of the system. It allows user to declare variables and to perform mathematical operations like in any high level language. The grammar for this robot programming language has the following notable features:

1. Main program
2. Multiple Variable Declarations
3. Dynamic Variable Declarations at any point in the program
4. Loops
5. Nested loops
6. The If Else Statement
7. Nested If, Else
8. Arithmetic statements

This robot programming language provides sufficient tools to the robot programmer for efficient control of the robotic arm. The program is fed to an interpreter, which converts the source code in to an intermediate language, which can be executed online. The robot control functions of our programming language are shown in table below.

Table 2: Robot Control Functions

Functions	Description
Home	Go to the Home position
Go()	Move to the point in x-y-z space
Motor()	Move the specified motor in the direction specified
Gripper()	Open or Close the Gripper
Point()	Go to point taught through the manual control

5.1 Graphical User Interface

The graphical user interface is designed in the C# as shown in fig.4. The GUI features two portions, first is the IDE and second portion contains a set of buttons for the manual control of the robotic arm which is used to teach points. Help and sample programs are added in the IDE to facilitate the user. User can develop programs in the IDE without connecting to the robotic arm i.e. from any remote place. These programs can be executed when the PC is connected to the robotic arm. Delete and save buttons are provided in the GUI to save and delete the points.

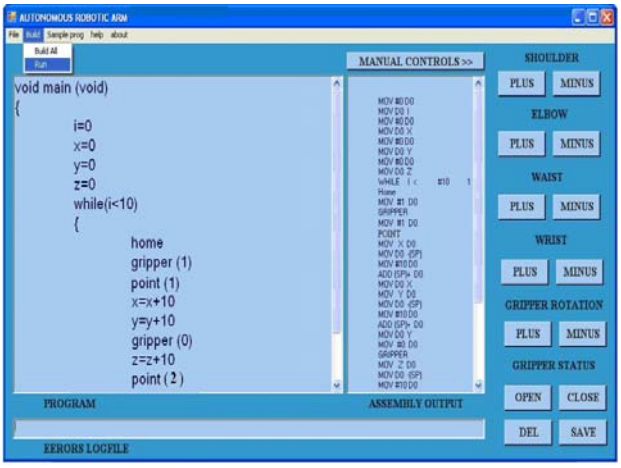

Fig.4 - View of Graphical User Interface

5.2 Synchronization

Since the computer program runs ahead of the real-time destination of the robot, the computer and the host must be synchronized [4]. To synchronize the systems, the computer program waits for the host to verify that the destination point has been reached, before moving ahead. So after execution of every robot command ATmega8535 controller sends a verification number to tell the teach pendant or the computer that the destination has been reached.

5.3 Gripper Control

A robotic arm must be able to interact with the objects in the surroundings to become useful [4]. For pick and place jobs gripper is used. The language command for the control of gripper is gripper ().

*gripper (*GRIPPER_OPEN*)*
*gripper (*GRIPPER_CLOSE*)*

5.4 Robot Kinematics

The programming language has following two types of robot kinematics functions.

5.4.1 Forward kinematics

Forward kinematics involves computing the position of the end-effector when inputs are the joint angles and geometric link parameters [4, 14]. Following equations are used to find the position of the end-effector when joint angles are known.

$$x = \cos\theta_1\left(l_2\cos\theta_2 + l_3\cos(\theta_2+\theta_3)\right) + l_4\cos\phi\cos\theta_1 \quad (1)$$

$$y = \sin\theta_1\left(l_2\cos\theta_2 + l_3\cos(\theta_2+\theta_3)\right) + l_4\cos\phi\sin\theta_1 \quad (2)$$

$$z = -l_2\sin\theta_2 - l_3\sin(\theta_2+\theta_3) + l_4\sin\phi \quad \ldots\ldots\ldots\ldots (3)$$

Where θ_1, θ_2, θ_3, θ_4, are the joint angles as shown in the fig.5 and x, y and z are the spatial coordinates of the end-effector relative to the base of the robot [14]. The length of the first link is assumed to be zero in both the forward kinematics and inverse kinematics equations. The angle PHI(ϕ) gives the orientation of the last link according to horizontal axis. Link lengths l_2, l_3 and l_4 are given in table 1. The language function that relates to forward kinematics is:

*motor(*MOTOR_NAME, ANGLE*)*

This function does not necessarily need to apply the forward kinematics equations to find the position of end effector because the joint angles can be directly converted to the encoder counts and transmitted to the host. The forward kinematics equations can be used to keep track of the position of the end effector in form of spatial coordinates.

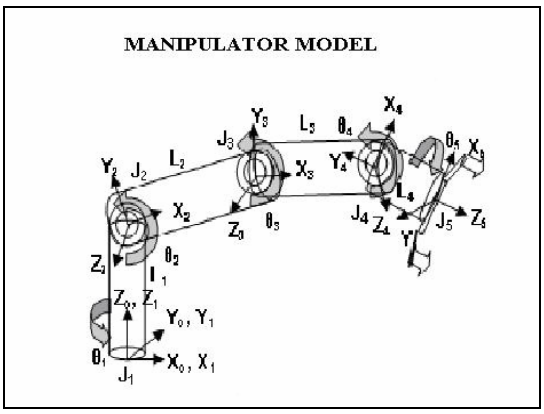

Fig. 5 - Kinematics of Robotic Arm

5.4.2 Inverse Kinematics

Given the geometric link parameters and the desired end-effector position and orientation relative to the base, computing the joint angles of the manipulator is known as inverse kinematics [4, 14]. The inverse kinematics equations depend on the link configuration of the manipulator, for our type of manipulator as shown in fig.5 these equations are as follows:

$$\theta_1 = \tan^{-1}\left(C_y, C_x\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

$$\theta_2 = -\tan^{-1}\left(C_z, \sqrt{C_x^2 + C_y^2}\right) - \tan^{-1}\left(l_3\sin\theta_3, l_2 + l_3\cos\theta_3\right) \quad \ldots\ldots\ldots (5)$$

Where, $\cos\theta_3 = \dfrac{C_x^2 + C_y^2 + C_z^2 - l_2^2 - l_3^2}{2 l_2 l_3}$

$$\theta_3 = \tan^{-1}\left(\sqrt{1-\cos^2\theta_3}, \cos\theta_3\right) \quad \ldots\ldots\ldots (6)$$

$$\theta_4 = PHI - \theta_2 - \theta_3 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (7)$$

Where C_x, C_y and C_z are the x, y and z coordinates of the end point of third link relative to the base of the robot [14]. The language command that involves inverse kinematics is:

*go(*POINT_X, POINT_Y, POINT_Z, PHI*)*

The joint angles found by applying the inverse kinematics equations are converted to encoder counts and then transmitted to the host.

5.5 Teaching

One of the most important feature of industrial robots is the teaching of points through manual control. Usually the manual control is only provided in the teach pendant. The software developed on the computer gives the capability of teaching points from the computer as well. All taught points are saved in the EEPROM of the microcontroller Atmega8535 that controls the robotic arm. In Fig. 4 the right portion of the GUI shows the buttons for the manual control and for saving and deleting points from the computer. The language command that performs the function of reaching a taught points is:

*point(*POINT_NAME*)*

6 Results

Teach pendant interface and the computer based language had been successfully tested on the robotic arm. To test the intelligibility of the system, robotic arm was operated by some novice users who found it very easy to learn and use.

Very high encoder resolution of the motors and good mechanical structure resulted in a high degree of accuracy and repeatability. The EEPROM of the microcontrollers was used for storing dynamic data such as programs written in the teach

pendant and the points saved in the Atmega8535 controller this eliminated the need for a battery backup and made the robotic arm more power and cost effective.

The arm was successfully programmed to reach and carry loads up to 3kg with a high degree of repeatability. Handshaking was implemented on the serial interfaces between Atmega8535 microcontroller, the host computer and the teach pendant that made the communication secure and extremely reliable at all times.

7 Conclusions and Future Work

The interface developed is powerful tool for controlling the robotic arm. Software has been designed in such a way to make it more user-friendly and understandable. The language has very easy to understand concise, clear and self explanatory syntax. Industrial robot systems are vital for the advancement of the industries, the techniques and the ideas mentioned in the paper can help to further increase the quality of these robot systems.

Modern robot systems provide graphical simulation and virtual environment [15,16] for programming of robots. Our system can be enhanced to include these facilities. Vision is one of the most important feature of the industrial robot systems present today. For this purpose a pair of cameras can be attached to the robotic arm, which will allow robot to automatically identify and grasp the objects.

Imitation based learning capability can be added to the robotic arm, which will allow path tracking by a different technique. The instruction set for the language and the teach pendant can be enhanced to include vision, forces, torques, imitation etc. The communication from the host can be made wireless this will allow programming and teaching from a remote location and would create a lot of other applications for this robotic arm.

References:

[1] I. Pembeci and G. Hager, A Comparative Review of Robot Programming Languages, *Technical Report CIRL,* Johns Hopkins University, august 14, 2001

[2] F. M. Wahl and U. Thomas, Robot Programming – From simple Moves to Complex Robot Tasks, *Proceedings of the First International Colloquium 'Collaborative Research Centre 562 - Robotic Systems for Modeling and Assembly'*, Braunschweig, Germany, May 2002, pp. 245-259.

[3] G. Biggs and B. MacDonald, A Survey of Robot Programming Systems, *Proceedings of the Australasian Conference on Robotics and Automation*, Brisbane, Australia, 2003.

[4] P. I. Corke and R. J. Kirkham, The ARCL Robot Programming System, *Proceedings of International Conference of Australian Robot Association*, Brisbane, July 1993, pp. 484-493.

[5] B. E. Shimano, VAL: A Versatile Robot Programming and Control System, *COMPSAC 79*, 1979.

[6] R. H. Taylor, P. D. Summers, and J. M. Meyer, AML: A Manufacturing Language, *International Journal Robotics Research,* Vol. 1, No. 3, 1982, pp 3-18.

[7] V. Hayward and R. P. Paul, Robot Manipulator Control under UNIX - RCCL: a Robot Control C Library, *International Journal of Robotics Research*, Vol.6, No.4, 1986, pp 94–111.

[8] C. Blume and W. Jakob, *PasRo-PASCAL for Robot,* Springer-Verlag, 1985.

[9] Robert Lafore, *Turbo C Programming for the PC*, revised edition, Waite Group Inc., 1999.

[10] RHINO ROBOTS INC. *Introduction to Robotics Mark 1V controller and XR-3 or XR-4 Robot Arm*, 1995.

[11] www.Atmel.com last accessed on April 2006

[12] J. Iqbal, N. Hassan, A. Waqar, S. Bano and R. Ilyas, Fabrication and Control of 4-DOF, Autonomous Robotic Arm Using Low Cost AVR Controller, *Proceedings of the International DARH conference*, Yverdon-les-Bains, Switzerland, 2005.

[13] M. Adeel, J. Iqbal, M. Ali, R. Asif, S. Mumtaz, Design, Control and Implementation of a Light Weight 5-DOF Robotic Arm using a Low-Cost Microcontroller via Radio LAN, *Proceedings of the International ACIAR conference*, Bangkok, Thailand, 2005.

[14] John. J. Craig, *Introduction to Robotics*, Mechanics and Control, 2nd Edition, Addison-Wesley, 1999.

[15] Murphy R.R. and Rogers E., Introduction to the Special Issue on Human Robot Interaction, *IEEE Trans. Syst. Man, Cybern,* Vol. 34, No. 2, May 2004, pp. 101-102.

[16] J. Aleotti, S. Caselli, and M. Reggiani, Evaluation of virtual fixtures for a robot programming by demonstration interface, *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, Vol. 35, No. 4, Jul. 2005, pp. 536–545.