DEHRADUN INSTITUTE OF TECHNOLOGY

DEVELOPMENT OF A 5-AXIS ROBOT ARM

Prepared for the partial fulfillment of the requirement for the Degree of Bachelor of Technology in Applied Electronics and Instrumentation Engineering

Submitted by:
Akanksha Bahuguna (07070106005)
Ayush Dewan (07070106012)
Saurav Gusain (07070106051)
Souri Guha (07070106055)
Branch:-AEI (final year)

Submitted to:
Mrs. Sonika Singh
Assistant Professor
AEI and ECE department

Acknowledgement

We are highly obliged to Mr. Anubhav Seth (Lecturer, Deptt. of AEI and ECE) for the effective guidance in preparation of seminar-its presentation and report on the project Development of a Five-Axis Robot Arm Implemented using PIC Microcontroller and LabVIEW.

His kind support encouragement and timely advice helped me getting acquaintance in the field of robotics and its applications in in particular, considering its rich and varied application prospects.

Contents

Objective

The aim of the project is to develop a *5-axis Robot Arm*. The motion of the Arm will be controlled by a PIC Microcontroller that will communicate on a real time basis with LabVIEW.

LabVIEW will take care of the forward/inverse kinematics calculations that are required to manipulate the Arm's motion and generate specific commands for the required process at each level by taking in data.

The PIC controller is responsible for providing feedback to LabVIEW regarding the position of each joint of the Arm as well as to receive commands from LabVIEW and ensure its proper execution.

Each of the joint of the Arm will be powered by a DC Servo Motor. The End Effector tool to be use will be a Gripper Tool. The basic function of the device will be to pick and place small objects.

Robotic Arm

A **Robotic Arm** is a robot manipulator, usually programmable, with similar functions to a human arm. The links of such a manipulator are connected by joints allowing either rotational motion translational (linear) displacement.

The links of the manipulator can be considered to form a kinematic chain. The business end of the kinematic chain of the manipulator is called the end effector and it is analogous to the human hand. The end effector can be designed to perform any desired task such as welding, gripping, spinning etc., depending on the application. For example robot arms in automotive assembly lines perform a variety of tasks such as welding and parts rotation and placement during assembly.

Types:

Cartesian robot / Gantry robot: Used for pick and place work, application of sealant, assembly operations, handling machine tools and arc welding. It's a robot whose arm has three prismatic joints, whose axes are coincident with a Cartesian coordinator.

Spherical robot / Polar robot (such as the Unimate):Used for handling at machine tools, spot welding, die-casting, fettling machines, gas welding and arc welding. It's a robot whose axes form a polar coordinate system.

- **SCARA robot:** Used for pick and place work, application of sealant, assembly operations and handling machine tools. It's a robot which has two parallel rotary joints to provide compliance in a plane.

Articulated robot: It is used for assembly operations, die-casting, fettling machines, gas welding, arc welding and spray painting. It's a robot whose arm has at least three rotary joints.

Parallel robot:One use is a mobile platform handling cockpit flight simulators. It's a robot whose arms have concurrent prismatic or rotary joints.

Defining Parameters

- **Number of axes**: Two axes are required to reach any point in a plane; three axes are required to reach any point in space. To fully control the orientation of the end of the arm (i.e. the wrist) three more axes (yaw, pitch, and roll) are required. Some designs (e.g. the SCARA robot) trade limitations in motion possibilities for cost, speed, and accuracy.

- **Degrees of freedom:** This is usually the same as the number of axes.

- **Working envelope**: The region of space a robot can reach.

- **Kinematics:** The actual arrangement of rigid members and joints in the robot, which determines the robot's possible motions. Classes of robot kinematics include articulated, Cartesian, parallel and SCARA.

- **Carrying capacity or payload**: How much weight a robot can lift.

- **Speed**: How fast the robot can position the end of its arm. This may be defined in terms of the angular or linear speed of each axis or as a compound speed i.e. the speed of the end of the arm when all axes are moving.

- **Acceleration**- How quickly an axis can accelerate. Since this is a limiting factor a robot may not be able to reach its specified maximum speed for movements over a short distance or a complex path requiring frequent changes of direction.

- **Accuracy**– How closely a robot can reach a commanded position. When the absolute position of the robot is measured and compared to the commanded position the error is a measure of accuracy. Accuracy can be improved with external sensing for example a vision system or IR. Accuracy can vary with speed and position within the working envelope and with payload.

- **Repeatability**- how well the robot will return to a programmed position. This is not the same as accuracy. It may be that when told to go to a certain X-Y-Z position that it gets only to within 1 mm of that position. This would be its accuracy which may be improved by calibration. But if that position is taught into controller memory and each time it is sent there it returns to within 0.1mm of the taught position then the repeatability will be within 0.1mm.

Things needed to build our robot

- Decide the Mechanical System
- Design the System
- Do calculations regarding the motion of the Robot
- Design electronic hardware to interface robot with PC for easy manipulation
- Optimization

Mechanical Hardware:

Steps in building Mechanical System:

- Freeze the DOF Required
- Find the Required Arm lengths
- Find position of Joints
- Freeze the Working Envelope required
- Freeze the Payload weight
- Calculate torque required at each joint to move the Robot
- Calculation of D-H Parameters
- Find and fix Motors that match the required parameters

Electronic Hardware:

Steps in building the Electronic System:

- Decide the method of operation of Robot(Manual/Teaching/Automatic)
- Decide the method of communication to robot for Teaching/Manual Control purposes
- Decide the method of Calculation for the Inverse Kinematics
- Find and Interface hardware that can do the decided tasks

Mechanical Parameters of our Robot

All the parameters mentioned below has been verified by defining and simulating them first in *Robotics Toolbox for MATLAB* and then by LabVIEW. We have also simulated the mechanical design in Solidworks by integrating it with LabVIEW.

Degree of freedom (DOF)

The Robot will have 5 axes and therefore 5 DOF. For movement of any Arm 3 axes are necessary but this compromises the precision and accuracy. A 6 axis arm will provide maximum accuracy and precision but we still intend to build a 5 axis arm so as to strike a balance between mechanical complexity and efficiency of the robot.

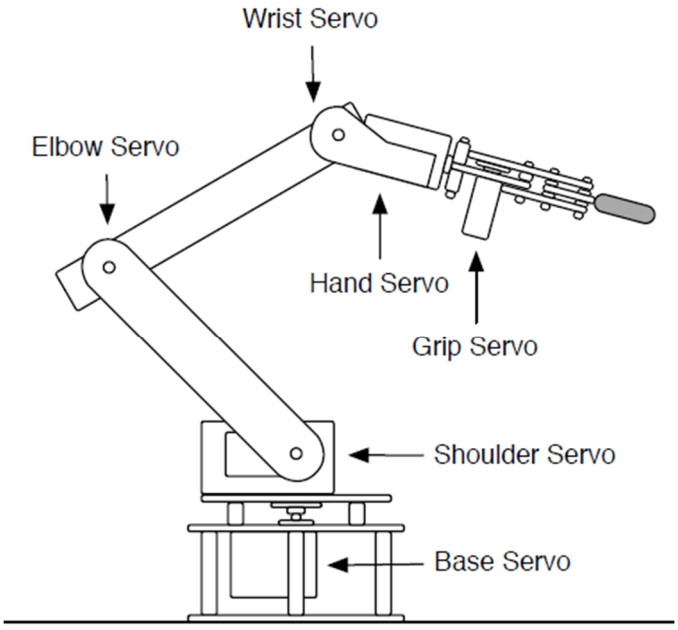
Arm Length

- The base is located at coordinate 0, 0, 0.
- The height of the shoulder joint is 3.0 inches above coordinate 0, 0, 0.
- The length of the upper arm (shoulder joint to elbow joint) is 4.75 inches.
- The length of the lower arm (elbow joint to wrist joint) is 4.75 inches.
- The hand is held at 90° (i.e. the hand is horizontal when the lower arm is parallel to the *x, y* plane)
- The grip length (wrist joint to grip point) varies from 5.2 inches to 5.9 inches, depending on the grip width

Position of joints

Assuming arm is placed in 2-D space joint coordinates is:

- Shoulder joint : 0,3
- Elbow Joint: 4.75,3
- Wrist Joint: 9.5,3
- Wrist Rotation Joint: 11,3



Side View

Working Envelope:

If we assume that all joints rotate a **maximum of 180 degrees**, because most servo motors cannot exceed that amount. To determine the workspace, the loci of all locations that the end effector can reach are traced and this will be determined by simulating the mechanical structure on Solidworks and will be verified after building the mechanical hardware.

Payload Weight:

Robot will be used for pick and place and the objects will be small boxes and payload will vary from **100- 250 grams.**

Motors:

We are going to use Digital Servo Motors, one at each joint of the Robot Arm.

After the torque calculation for each joint, appropriate motors are to be chosen which not only meets the torque requirement but also supports the weight of robot.

We are yet to complete this process, since this involves calculation of torque at each joint for the given payload weight. Although it is possible to do this at the moment with all other parameters at our hand, we prefer to do this once the mechanical structure has been built because the huge cost of the motors, so we just can’t afford the slightest miscalculation/error.

Electronic Hardware of our Robot

The Electronic Hardware is essentially the brain of the Robot. So, its design is *strictly* dependent on the way we need our Robot to function.

So we'll explain the working before we put forward the Electronic Design.

Method of operation of Robot:

For a complete control over motion of robotic arm following 3 Modes are used:

- **Manual Mode:** In this mode each axis is rotated individually with the help of toggle switches.
- **Teaching Mode:** In this mode path of the robot is planned and coordinates of various points are calculated. Speed of operation is kept low.
- **Automatic Mode:** In this mode robot moves on a pre-planned path continuously with precision and accuracy with high velocity with the help of coordinates stored in memory.

Method of communication to robot for Teaching/Manual Control purposes:

- For manual control toggle switches will be used for each joint to control the rotation of motor in both clockwise and anti-clockwise direction.
- For teaching the robot a LabVIEW application will be used, this application will communicate with controller controlling the servo motors through Serial\Ethernet communication.

Manipulation of the Robot Arm

Manipulation of a Robot Arm is to control the servo motors in a specific order so as to make it work in the way we want it to work. The primary goal is to move the end effector tool to the desired positions in the 3-d working envelope

Now there will be various phases to this objective:

- **Phase1:** Move the Motors of the Robot by their individual axes i.e. one at a time
- **Phase2:** Move the End effector tool from point 1 to point 2

Now in Phase2, we have two methods to attain our objective:

- **Forward Kinematics:**
  - ***I know:*** position/velocity of each robot joint
  - ***I need to find out:*** end position/velocity of the robot
- **Inverse Kinematics:**
  - ***I know:*** desired position/ velocity of the end point where the robot must reach
  - ***I need to find out:*** the position/ velocity of each joint of the robot

The primary functioning of the Robot will be done by means of Inverse Kinematics. This is due to the fact that in maximum cases we will know the desired position of the end effector tool, but however, the rotation required by each servo will be unknown. This is where Inverse Kinematics would come in.

Method of Calculation for the Inverse Kinematics

The inverse kinematics calculations required has been verified by using *Robotics Toolbox for Matlab* as well as LabVIEW.

We intended to verify the integrity of our calculations in MATLAB since it provided a much more robust, flexible and yet a clear and transparent platform to compute the parameters. In MATLAB it was possible to look into every step of the calculation process and verify our data.

A similar thing is not possible in advanced softwares like LabVIEW, which although provides a better interface and a smart design flow of the entire system.

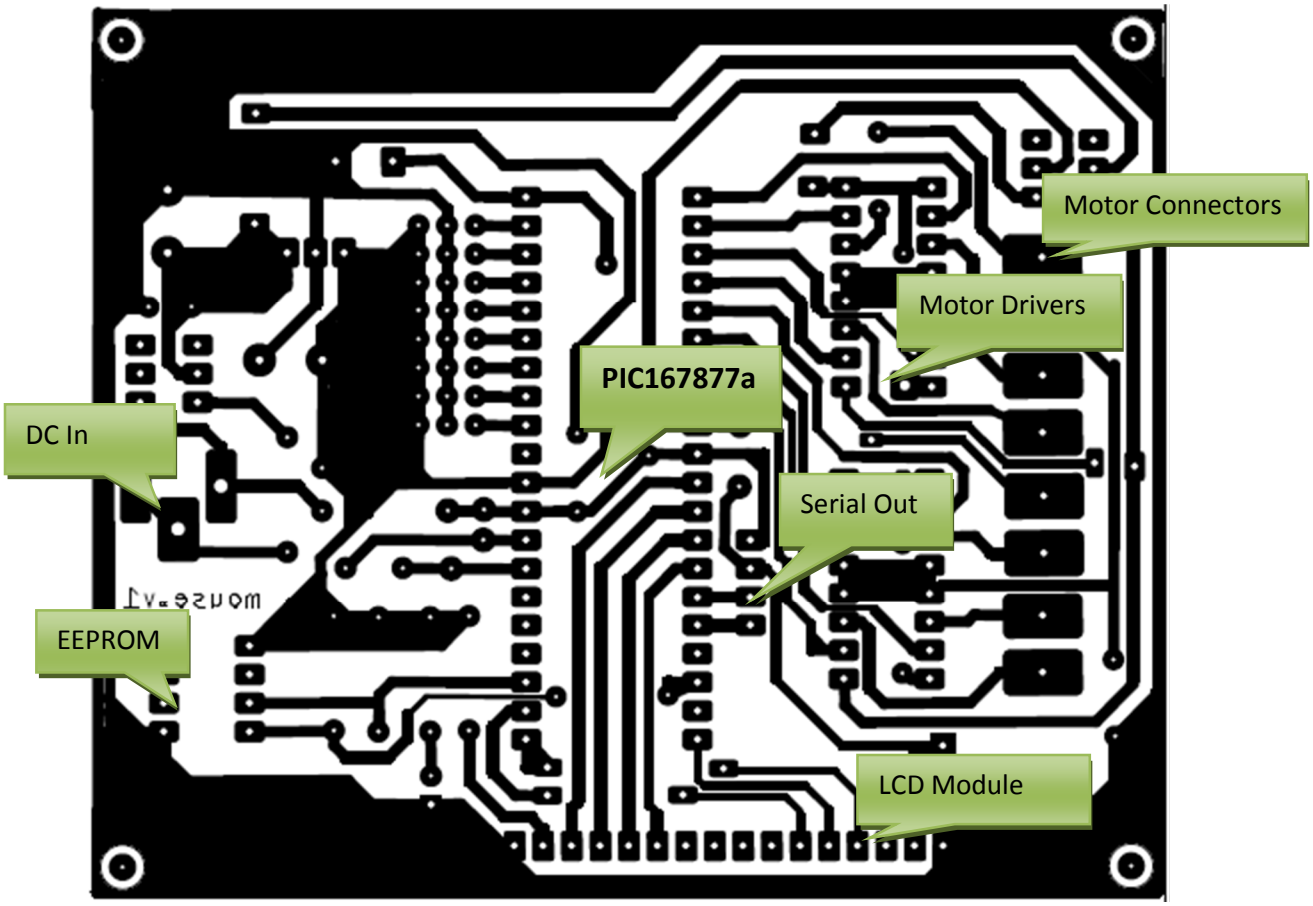
So once the data has been judged by MATLAB, we moved on to LabVIEW for a fast paced design process to implement the control system.

Interface hardware that can do the decided tasks:

Various calculations required for smooth functioning of robotic arm are:

Task	Solution
Inverse Kinematics	Labview Robotics Module
PID algorithm for servo mechanism	Labview PID and Fuzzy Logic Module
Servo Control	PIC Microcontroller
Power Supply	AC to DC power Source (12-24V)

Layout of the PCB



Technical Specifications of the Robot

Microcontroller used: PIC
Interfaced Components:

- Quadrature Encoders
- EEPROM
- Serial and/or Ethernet Interface

Control System:

- LabVIEW 2010 Base System
- LabVIEW 2010 Robotics Module
- LabVIEW 2010 PID and Fuzzy Logic Module
- A x86 Computer with >=1GB of RAM and Dual Core Intel Processor

Control Mechanism of our Robot

The device will be programmed such as to meet the following requirements:

- It can communicate with a PC by using a Serial/Ethernet connection
- It has the ability to obtain motion instructions from a PC and generate required PWM signals to control the Servos at the Linkages
- The device will have various modes of operation:
    - **Manual Control Mode :** Signals will be generated manually by using control sliders in the LabVIEW GUI individually for all the motors
    - **Teaching Mode:** We will use the end effector of the Robot and move it along the required path from the start to end point. The Control Mechanism will trace and record the feedbacks from the Servos at different instances of time so that it will be later able to trace the same path using the same recorded coordinates.
    - **Playback Mode:** The Robot will recall the table of values recorded at the time of teaching and would repeat its action. The Only thing it does automatically at the end is to pick and place the object by properly using the gripper.

The other end of the control side will reside on a Computer. This is due to the major limitations in the processing power of a Microcontroller. The details of the Server side of the Control Mechanism are as follows:

- The Server will have **LabVIEW 2010** installed in it.
- The LabVIEW core will have the following modules interfaced with it:
    - **LabVIEW Robotics Module:** This is done for analysis and simulation of the Robot Arm's motion
    - **LabVIEW PID Control Module:** This module will be used in generation of the PID Controller that we use to control the Robot to accurately follow the desired path and reach its source and destination with minimum error.

Analysis

The motion of the Robot needs to be analyzed and perfected at various stages of the build process. This is very much essential to test and debug our algorithms.

This will be done at various stages which are given below:

- **Software based analysis prior to Robot Build:** Done using Solidworks interfaced with LabVIEW. This is required to prevent any design flaw that might be detected after building the Robot.

- **Motor Analysis on 1st Robot Build:** This is done by issuing separate commands to all servos to ensure that they are functioning perfectly individually.

- **Motor Feedback analysis:** This is done by moving the Robot linkages and recording data in the PC and analyzing whether their motions match the actual feedback.

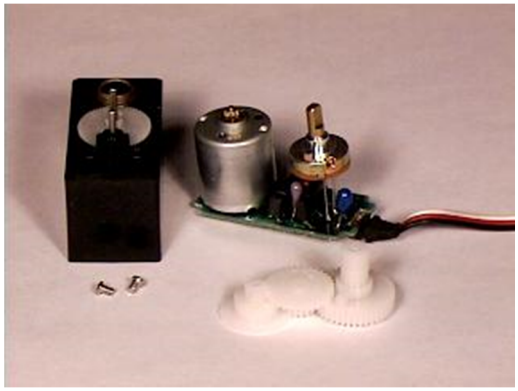
- **Analysis of error when all the Motors are used together:** This is done to test, optimize and debug the PID Control System implemented in LabVIEW.

- **Final integration analysis**

Description of used Hardware

Servo Motor

A Servo is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes.

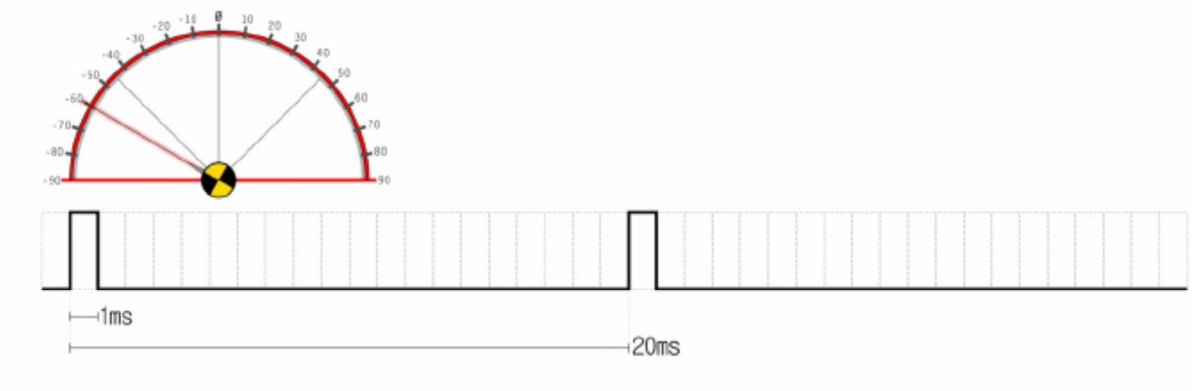


The servo motor has some control circuits and a potentiometer (a variable resistor, aka pot) that is connected to the output shaft. In the picture above, the pot can be seen on the right side of the circuit board. This pot allows the control circuitry to monitor the current angle of the servo motor. If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will turn the motor the correct direction until the angle is correct.

How we will control the servos

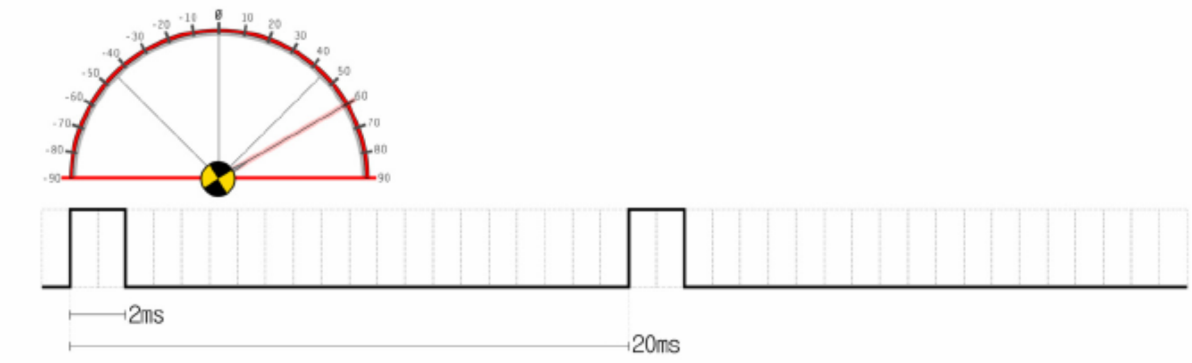
Servos have a very simple electrical interface; they usually have 3 wires, one for power, one for ground and the other for the pulse train. Once power (usually in the 4.8V-6.0V range) and ground is supplied to the servo, the data wire is prepared to receive encoded signal in the form of pulse width modulation (PWM). The duty-cycle is of 20ms, and the pulse width contained within those 20ms varies from 1ms to 2ms.

Actually, it is this variation that controls the servo. As a rule of thumb, if a 1ms pulse is constantly fed to the servo, it will position itself around -60° angle, 1.5ms and it will go to the center (0°), and 2 ms will position the servo at +60° angle.



(a)

(b)



Shaft Encoder

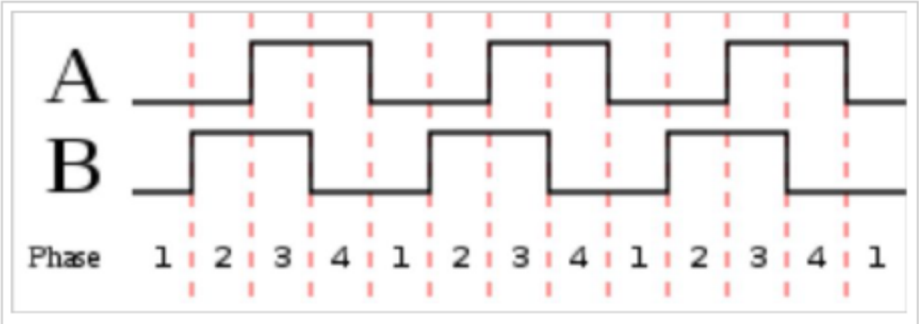
A **rotary encoder**, also called a **shaft encoder**, is an electro-mechanical device that converts the angular position of a shaft or axle to an analog or digital code, making it an angle transducer. Rotary encoders are used in many applications that require precise shaft unlimited rotation.

An incremental rotary encoder, also known as a quadrature encoder or a relative rotary encoder, has two outputs called quadrature outputs. They can be either mechanical or optical. In the optical type, there are two gray coded tracks, while the mechanical type has two contacts that are actuated by cams on the rotating shaft. The mechanical type requires debouncing and is typically used as digital potentiometers on equipment including consumer devices. Most modern home and car stereos use mechanical rotary encoders for volume. Due to the fact the mechanical switches require debouncing, the mechanical type are limited in the rotational speeds they can handle. The incremental rotary encoder is the most widely used of all rotary encoders due to its low cost: only two sensors are required.

The fact that incremental encoders use only two sensors does not compromise their accuracy. One can find in the market incremental encoders with up to 10,000 counts per revolution, or more.

There can be an optional third output: reference, which happens once every turn. This is used when there is the need of an absolute reference, such as positioning systems.

The optical type is used when higher RPMs are encountered or a higher degree of precision is required.



Incremental encoders are used to track motion and can be used to determine position and velocity. This can be either linear or rotary motion. Because the direction can be determined, very accurate measurements can be made.

They employ two outputs called A & B, which are called quadrature outputs, as they are 90 degrees out of phase.

These signals are decoded to produce a count up pulse or a countdown pulse. For decoding in software, the A & B outputs are read by software, either via an interrupt on any edge or polling, and the above table is used to decode the direction. For example, if the last value was 00 and the current value is 01, the device has moved one half step in the clockwise direction.

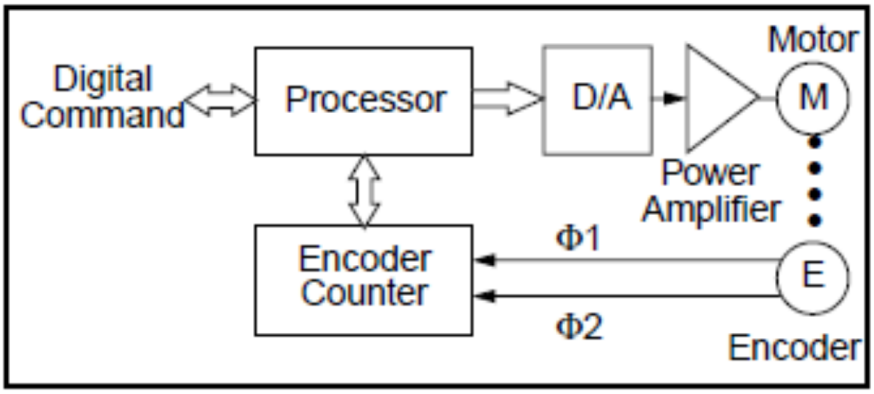
The Servo Mechanism

It is a mechanism developed for precise controlling of servo motors. It is based on close loop process control system.

Basic Principle:

These systems control a motor with an incremental feedback device known as a sequential encoder. They consist of an encoder counter, a processor, some form of D/A (Digital-to-Analog) converter, and a power amplifier which delivers current or Voltage to the motor. The PIC microcontroller implements both the servo compensator algorithm and the trajectory profile (trapezoidal) generation. A trajectory generation algorithm is necessary for optimum motion and its implementation is as important as the servo compensator itself.

The servo compensator can be implemented as a traditional digital filter, a fuzzy logic algorithm, or a simple PID algorithm (as implemented in this application note). The combination of servo compensator and trajectory calculations can place significant demands on the processor. The D/A conversion can be handled by a conventional DAC or by using the controller's pulse-width modulation (PWM).



In either case the output signal is fed to a power stage which translates the analog signal(s) into usable voltages and currents to drive the motor. PWM output can be a duty-cycle signal in combination with a direction signal or a single signal which carries both pieces of information. In the latter case a 50% duty cycle commands a null output; a 0% duty cycle commands maximum negative output, and 100% maximum positive outputs. The amplifier can be configured to supply a controlled voltage or current to the motor.

Most embedded systems use voltage output because it's simpler and cheaper. Sequential encoders produce quadrature pulse trains from which position, speed, and direction of the motor rotation can be derived. The frequency is proportional to speed and each transition of F1 and F2 represents an increment of position. Phase of the signals is used to determine direction of rotation. These encoder signals are usually decoded into Count Up and Count Down pulses, using a small state machine. These pulses are then routed to an N-bit, up/down counter whose value corresponds to the position of the motor shaft. The decoder/counter may be implemented in hardware, software, or a combination of the two.

Conclusion

Right now we have the design of the Robot Arm at hand with all the parameters simulated and verified. We took a bit of time to get the verification process done so as to ensure that we do not encounter some ridiculous error in the mid of the project.

The immediate next task at hand is to get the manufacturing work done and then the implementation and optimization of the controller work will be started.

These types of Robot Arms today are being widely used in the industry and we are making sure that this project can cope up with the specifications that are accepted at industrial level.

We also plan to build some extensions to this robot once the current target has been achieved to put ourselves towards some more innovative challenges and we are sure that we'll enjoy and learn a lot in this project.

References

- *Software Development for the Kinematic Analysis of a Lynx 6 Robot Arm* - Baki Koyuncu, and Mehmet Güzel (World Academy of Science, Engineering and Technology, 2007)

- *Kinematic Modeling and Simulation of a SCARA Robot by Using Solid Dynamics and Verification by MATLAB/Simulink* - Mahdi Salman Alshamasin, Florin Ionescu, Riad Taha Al-Kasasbeh (European Journal of Scientific Research, 2009)

- *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods*: Samuel R. Buss, Department of Mathematics, University of California, San Diego, October 7, 2009

- *Jacobian Solutions to the Inverse Kinematics Problem* - Mike Tabaczynski, Tufts University, January 17th, 2006

- *How to Interface a Microchip PIC MCU with a hobby R/C Servo* - Paulo E. Merloti

- *A Robust Inexpensive Multi-Purpose Robotic Arm* - Alana Lafferty, UAP Report (May 20, 2005)

- *Kinematic Analysis for Robotic Arm* - Sırma C. Yavuz, Yildiz Technical University, İstanbul, 2009

- *Application on D.C. Servo Control* Developed by Stephen Bowling, Microchip Technology Inc. Chandler, AZ (AN696)