# A MULTITASKING PC BASED ROBOTIC ARM MANIPULATOR CONTROL SYSTEM IN RT-LINUX ENVIRONMENT


Md.Adnan Al Moshi
ID: 09221102

Md. Eftakhairul Islam
ID: 09221175

Salwa Salam Cynthia
ID: 09221211

**Department of Electrical and Electronic Engineering**


Summer 2010



**BRAC University, Dhaka, Bangladesh**

DECLARATION

We hereby declare that this thesis report is based on the results found by ourselves. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of Supervisor

Signature of Authors

ACKNOWLEDGEMENT

We would like to take this opportunity to express our gratitude and sincere thanks to our supervisor DR. AKM Abdul Malek Azad. We would like to thank him for giving us the opportunity to work on this project under his supervision. We are indebt to him for his support and guidance throughout the period of pre-thesis and thesis semesters.

We would also like to convey our thanks to our co-supervisor Rumana Rahman for her support and guidance in our project.

Lastly we would like to thank Dr.Khalilur Rahman for helping us a lot and for sharing different ideas with us. We would also like to thank Md.Asiful Alam, Md. Asifur Rahman,Raquib Ahmed, Ahsanul Haque, Raied Hassan and Tomal vai for their support and appriciation.

## ABSTRACT

Automation in industries helps in achieving the target of quality standardization and process visibility. In overseas the industries are automated using different robots which work efficiently. But in a developing country like ours, we can not afford to import robots from overseas. Again our country is not technically sound enough to produce robots commercially in large scale for industrial usage. Considering these aspects we have prepared a robotic arm manipulator which work under RT-Linux environment. This step will introduce automation in our country. In the RT-Linux environment two or more robotic arms can be operated simultaneously. This environment is established in such a way that multiple tasks can be done at the same time. This is efficient as well as cost effective. In our project the robotic arms are able to pick and place objects controlled by two bipolar stepper motors and a DC motor (for the gripper) for each robotic arm. The motor will be operated in an open loop (without feedback) system. The operation of the Stepper Motor is controlled using RT-Linux to minimize the real-time error (jitter). From one computer we can operate all the manipulators. Stepper motors are used because it gives the steps so precisely. The theoretical results are confirmed with practical application.

TABLE OF CONTENTS

List of Figure:

Page no.

# CHAPTER I. INTRODUCTION & MOTIVATION

## 1.1 What is Robot and Robotics?

An automatically guided machine which is able to do tasks on its own is known as a ‘ROBOT’. A robot is a machine designed to execute one or more tasks repeatedly, with speed and precision. A robot is a machine that gathers information about its environment (senses) and uses that information (thinks) to follow instructions to do work (acts).

Robotics is the engineering science and technology of robots, and their design, manufacture, application, and structural disposition. Robotics is related to electronics, mechanics, and software. To perform high-precision jobs such as welding and riveting, robots are now widely used in factories. They are also used in special situations that would be dangerous for humans -- for example in cleaning toxic wastes or defusing bombs. The field of robotics is more practically defined as the study, design and use of robot systems for manufacturing a top-level definition relying on the prior definition of robot. Robotics is the art, knowledge base, and designing, applying, and using robots in human endeavors. Robotic system consists of not only robots but also some other devices and systems are used together with the robots to perform necessary tasks. Robotics is an inter-disciplinary subject that benefits from mechanical, electronic and electrical engineering.

## 1.2 Introduction to Real time Linux:

RT-Linux is an operating system in which a small real-time kernel coexists with the Posix-like Linux kernel. The intention is to make use of the sophisticated services and highly optimized average case behavior of a standard time-shared computer system while still permitting real-time functions to operate in a predictable and low-latency environment. In RT-Linux, all interrupts are initially handled by the Real-Time kernel and are passed to the Linux task only when there are no real-time tasks to run. In practice, the RT-Linux approach has proven to be very successful. Many applications appear to benefit from a synergy between the real-time system and the average case optimized standard operating system. For example, data-acquisition applications are usually composed a simple polling or interrupt driven real-time task that pipes data through a queue to a Linux process that takes care of logging and display. In such cases, the I/O buffering and aggregation performed by Linux provides a high level of average case performance while the real-time task meets strict worst-case limited deadlines. The operating system allows for great flexibility in such things as the characteristics of real-time tasks, communication, and synchronization.

# CHAPTER I I BASICS OF ROBOTIC ARM

## 2.1 Basics of Robotic Arm

The robotic arm manipulators (*Figure 1.1*) will be able to work in any environment which is basically independent on time, emotions, and place. The whole operation is controlled by Real-Time Linux operating system. It will be able to rotate to any commanded direction. After that it will execute the picking and placing

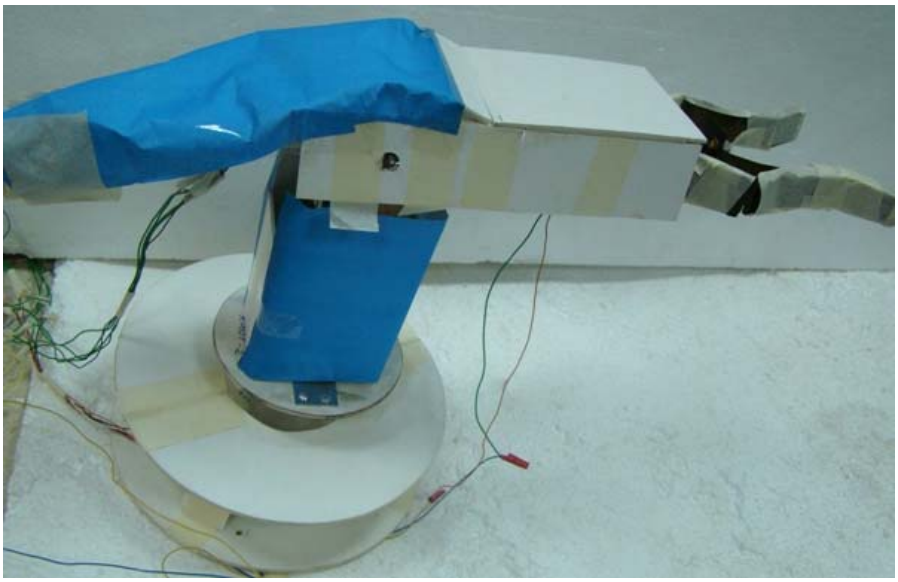

Fig:1.1 Body of the Robotic Arm

and will rotate in the direction of destination. The motion of the robots is controlled using stepper motors at the arm and DC motor at the gripper.

2.2 Industrial Purpose of a Robotic Arm:

Industrial Robots performs or assists to perform hazardous tasks in multiple industrial applications. Welding, Soldering, Material handling, Thermal spraying Painting, Drilling these are some tasks that can be done by a robot.


Fig:1.2 Industrial Robotic Arm

# CHAPTER III COMPONENTS of ROBOTIC ARM

## 3.1 Plastic board

The weight of Plastic board is less compared to iron or any metal that's why we choose PVC board to make the robotic arm. This is also inexpensive. For industrial use in our country we were looking for a material by which a robotic arm can be made easily. This board doesn't require any maintenance. It ensures inherent fire safety, excellent durability and long-life expectancy.

## 3.2 Stepper Motor

A stepper motor (Fig:3.1) is an electromechanical device which converts electrical pulses into discrete mechanical movements. Steppers can be moved to any desired position reliably by sending them the proper number of step pulses.

Fig 3.1: Stepper Motor

The motor will be operated in an open loop (without feedback) system to reduce error. More details about stepper motor are discussed later.

## 3.3 DC Motor

A DC motor is an electric motor that runs on direct current (DC) electricity.More details about Dc Motor is discussed later in this report.

**Fig 3.2: DC Motor**

3.4 Data AcQuisition Card

A Data AcQuisition card (DAQ, PCL-812PG) (Fig:3.3) will be used as a hardware interface. Data acquisition is the sampling of the real world to generate data that can be manipulated by a computer. DAQ is what usually interfaces between the signal and a PC. Driver software for DAQ card (PCL-812PG) will be used to register data coming from hardware. The pulses send from the PC goes to the driver circuit through DAQ Card. There are 16 input pin and 16 output pin in the card. In our case we used 16 output pins,each 8 (eight) pins are giving the output of one robot i.e. two stepper motors.

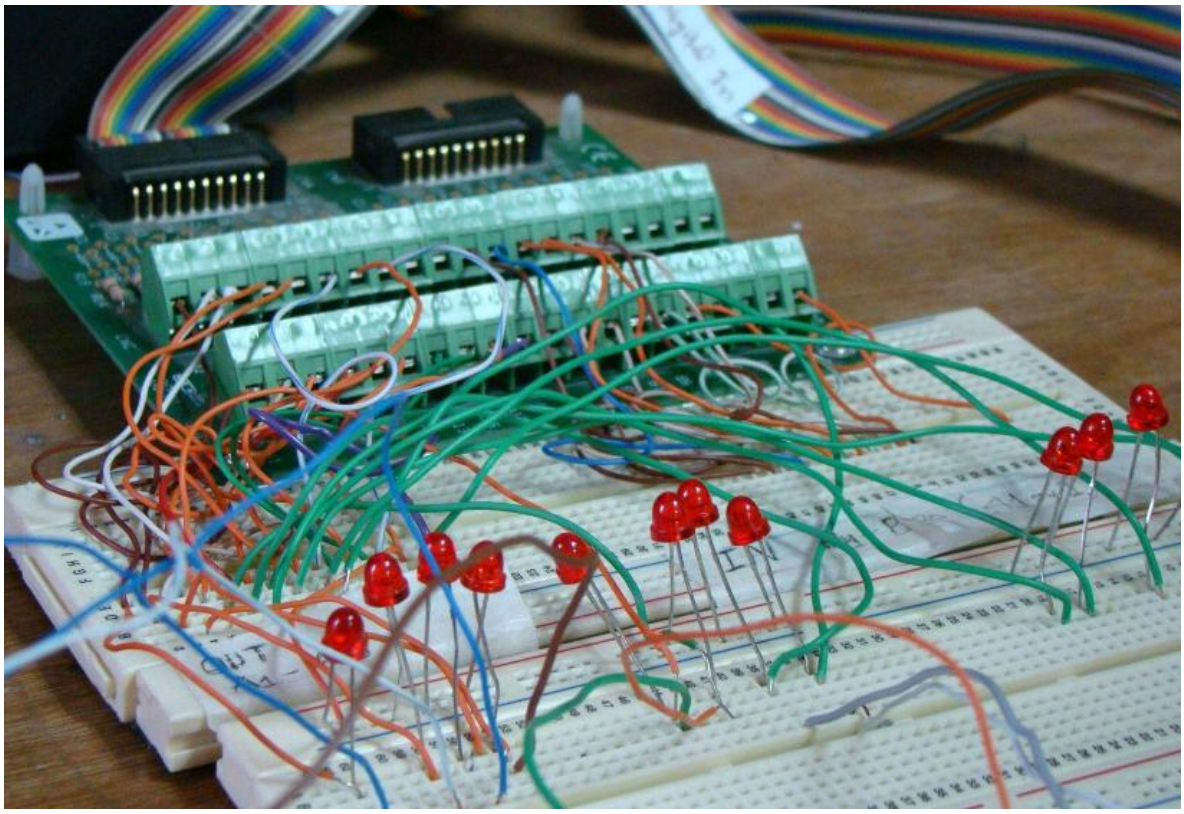

Fig:3.3: Data AcQuisition card (DAQ, PCL-812PG)

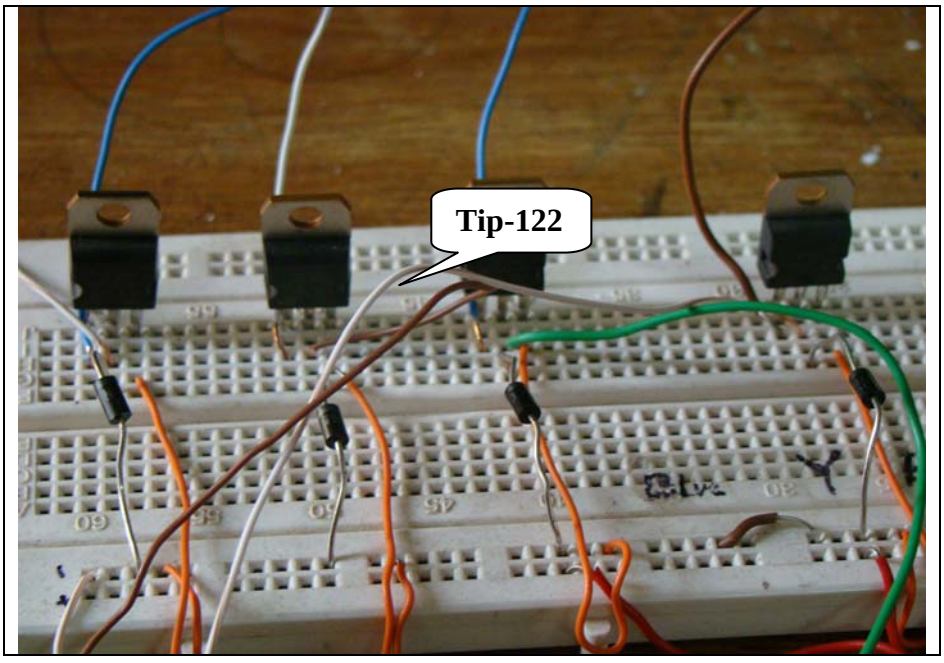


Fig.3.4 Tip122 Darlington pair transistor used in bread board

3.5 tip-122

The stepper motors required high current but the DAQ card gives small current as output. To achieve current amplification we used TIP 122 [].it is also known as Darlington Pair Transistors.

Figure 3.5 shows how a TIP 122 Darlington pair transistor looks like. Like normal transistors it also consists of Base, Emitter and Collector. The TIP 122 comes with an extra metal that is used to connect external heat sink. Since TIP 122 can handle huge amount of current so it gets very hot and hence external heat sink is sometimes required. The diagram shows the configuration of how the base, collector and emitter are arranged in this transistor.

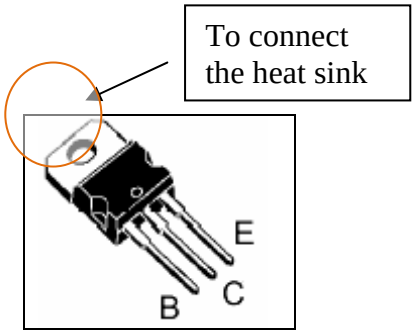


Fig.:3.5 Tip122 Darlington pair transistor

Figure 3.6 shows the internal configuration of the TIP 122 Darlington pair transistors. Two transistors are connected in the configuration show with respect to each other. Pulse from Data AcQuisition Card comes the base of the 1st transistor (Q1) and turns it on. The base of the 2nd transistor (Q2) is connected to the emitter of Q1 so that when Q1 is on, Q2 is also on. So when both the transistors are on, current can flow from collector to emitter for both the transistors and the emitter current for both the transistors are added up in the end. This adding up of current, amplifies the current that is given as input to the base of Q1, hence TIP 122 Darlington pair transistors work as current amplifiers.

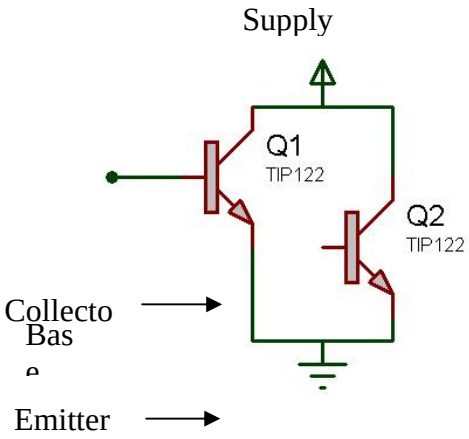


Fig.3.6: Internal diagram of TIP122

3.6 IC-7408

It is the logical AND gate (Fig:3.6) .we used this IC for the selection pulse.as we gave the same four input to first 8 (eight) tip-122 transistors so there are two selection pulses for each two motor.the outputs were driven to the Darlington pair through logical AND gates. The first four AND gates got the first four pulses along with selection pulse M1 as input. Similarly second four AND gates got the second four pulses along with selection pulse M2 as input. Similar setup is also done for another robot.This setup is shown im the block diagram of Fig.:3.7 & Fig.:3.8

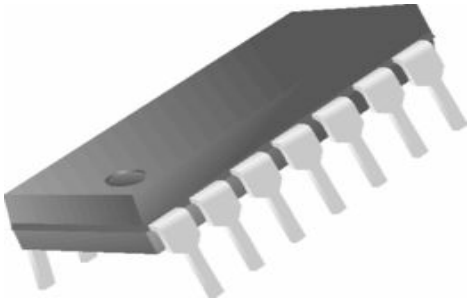

Fig:3.7 IC 7408

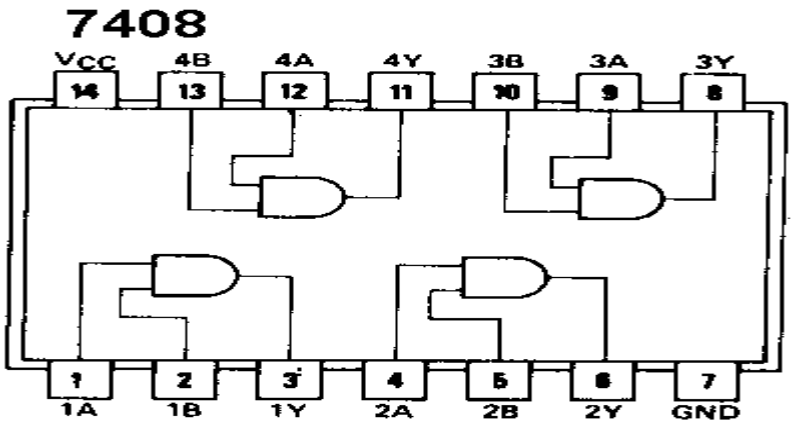


Fig:3.8 Pin Configuration of IC 7408

# CHAPTER VI .THE MOTORS

4.1 Stepper motor

Stepper motors are versatile, long lasting and very simple to use. Unlike regular DC/ AC motors, these motors do not rotate when connected to the power supply instead they rotate only when the magnetic field is rotated through different windings.

Stepper motors is a kind of DC motor that is brushless and has discrete rotation unlike DC motors [13]. This ability to rotate in discrete steps allows them to be very precise which makes it suitable for our project. The precision movement also has a very big advantage and that is no feedback system is required. Stepper motors are quite available as they are widely used commercially which makes them less expensive. They are easy to implement and also has longer life. Stepper motors works on the principle of energizing respective electromagnet hence they require additional circuitry in order to make them work.

The figure 4.1 show the working principle for a stepper motor. As it can be seen that a command is given to the stepper motor and it works accordingly. No feedback system is required hence making the system less complex. More details about the working principle of the motor will be discussed later.

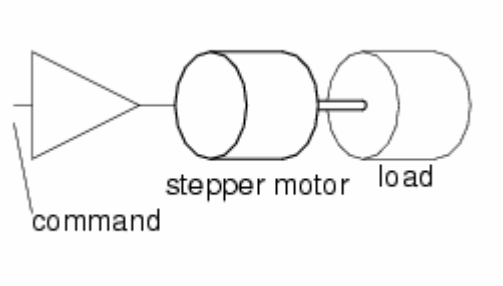


Fig. 4.1. Working principle of stepper motor

Stepper motors are a different kind of motors and they have a unique Torque vs. Speed characteristic (figure 4.2). In general stepper motors have very high torque compared to the other type of motors but this torque decreases rapidly as the speed of the shaft in the motor increases. The torque of the stepper motor remains fairly constant as the speed starts to increase but after a certain “cutoff speed” is reached, the the torque starts to decrease rapidly until it becomes zero as the speeds keeps increasing.

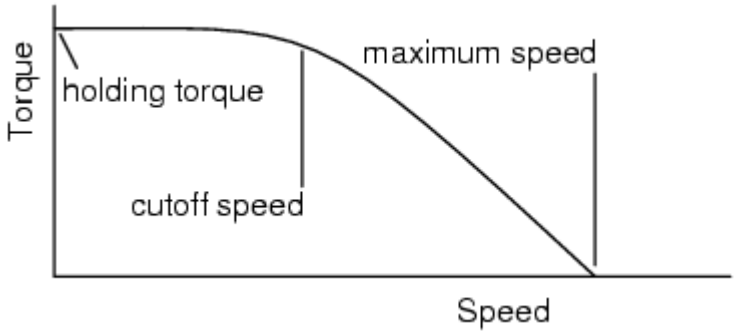


Fig. 4.2. Torque vs. speed characteristics curve

There are many types of Stepper motors available. The three main types are:

1. Permanent magnet (PM) stepper motor
2. Variable reluctance (VR) stepper motor
3. Hybrid synchronous (HS) stepper motor

For our project we have used the Permanent Magnet (PM) type stepper motor. This type of motor is easily available in the market and the working principle is very simple. Permanent Magnet (PM) type stepper motors usually have 4 electromagnets on 4 sides and a rotor/shaft sitting in the middle of these electromagnets. The shaft itself is magnetized with different polarity that is distributed evenly throughout the circular shaft. This unique design of the shaft/gear will enable it to move precisely when the electromagnets are energized.

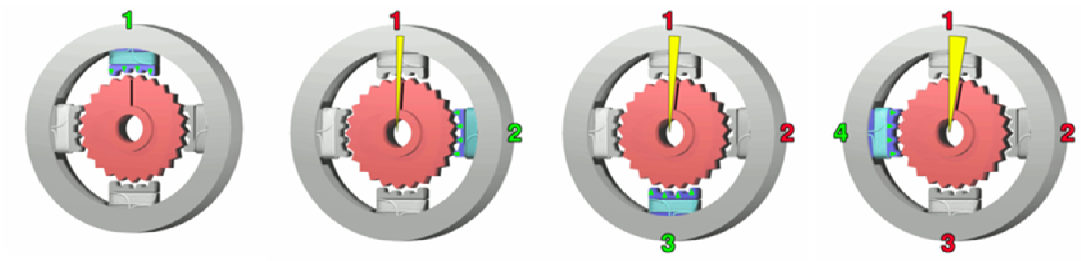


Fig. 4.3. Working principle of PM stepper motor

The figure 4.3 shows the energizing sequence of the electromagnets. The electromagnets are energized by an external control circuit, such as a

microcontroller or even using a computer’s parallel port. In our case we have energized the

electromagnets with computer’s parallel port. To make the motor shaft turn, first one electromagnet is given power, which makes the gear's teeth magnetically attracted to the electromagnet's teeth. When the gear's teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet. So when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one, and from there the process is repeated [14]. Each of those slight rotations is called a "step". This way the movement of the stepper motor is precise and can be used for high accuracy movement.

We choose stepper motors so that we can control the movement of the system more precisely. According to the requirement of the project, both the motors are different in ratings from each other. One of the motors which have a rating of 12V and 0.4A is responsible for the rotation of the base of the robot. Another motor have a rating of 5V and 1.3A which is responsible for the movement of the upper arm. For both robots the motors are of same rating. Both the stepper motors used for this thesis project has a resolution of 1.8 degree/step. This means that when a pulse is applied to the stepper motor, the shaft will rotate by 1.8 degrees.

There are two basic winding arrangements for the electromagnetic coils in a two phase stepper motors: 1) Bipolar and 2) Unipolar.

### 4.1.1 Bipolar Stepper Motor:

Bipolar motors have a single winding per phase. The current in a winding needs to be reversed in order to reverse a magnetic pole, so the driving circuit must be more complicated, typically with an H-bridge arrangement. There are two leads per phase, none are common. Static friction effects have been observed with certain drive topologies. These types of motors have no center taps. The advantage of not having center taps is that current flows through the entire winding instead of just half of the windings at a time. As a result, more torque is produced. But more complex control circuitry is required.

### 4.1.2 Unipolar Stepper Motor

A unipoar stepper motor has two windings per phase. Since in this arrangement a magnetic pole can be reversed without switching the direction of current, the commutation circuit is very simple for each winding. Typically, given a phase, one end of each winding is made common: giving three leads per phase and six leads for a typical two phase motor. Often, these two phase commons are internally joined, so the motor has only five leads. Regardless the number of wires, the center tap is connected to the power supply and the ends of the wires are alternately connected to ground.

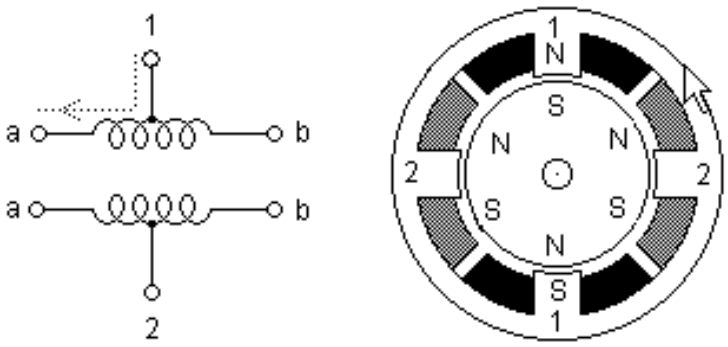


Fig 4.4: Internal Winding of Unipolar Stepper Motor

### 4.1.3 Why Using Unipolar Stepper Motor

The main difference between Uni-Polar and Bi-Polar stepper motor is that for Uni-Polar stepper change of current is not required to alter the direction of the magnet. In Bi-Polar the scene is completely the opposite. To reverse the direction of the magnet, change of direction of current is required and hence leads to a more complicated circuit and hence harder to implement because it is not always easy to reverse the direction of the current. To avoid complicated driver circuit we used Unipolar Stepper Motor in our project.

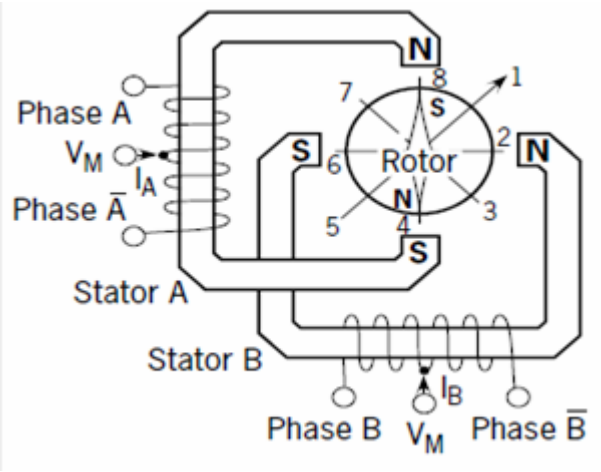


Fig 4.5(a) . Uni-polar stepper motor

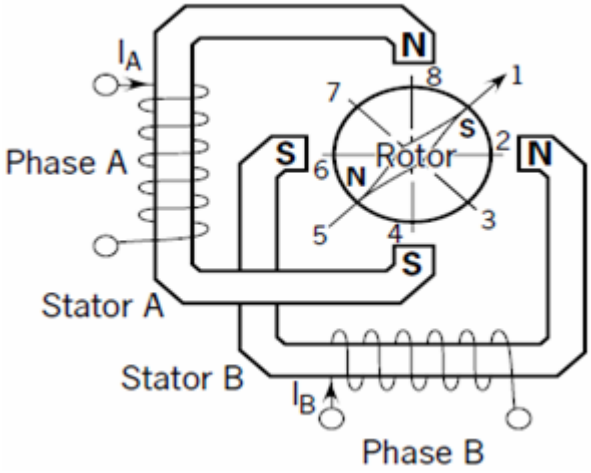


Fig. 4.5(b) Bi-polar stepper motor

Specifications of Base stepper motor: 1) 23LM-K307-PL
2) 1.8 Degree per Step.
3) 1.3 A.
4) 5 V.

Combinations of Pins and Colors
Of Base stepper Motor: 1) Red
2) Green-White
3) Red-White
4) Green
5) VCC: White and Black

Specifications of Elbow Joint Stepper Motor: 1) PK244-03A
2) 1.8 Degree per Step
3) 0.4 A.
4) 12 V.

Combinations of Pins and Colors
Of Elbow Joint stepper Motor: 1) Black
2) Red

3) Green
4) Blue
5) VCC: Yellow and White.

4.2 Stepper Motor Driver Circuit:

The stepper motor driver circuit consists of two parts. First part consists of 16 AND gates (four 7408 IC). Pulses form the DAQ card are the input of the AND gates. There are two selection pulses which comes from the DAQ card. Input of the 1st AND gates are selection pulse M1 and the pulses of 1st four pin of the DAQ Card. Similarly next 8 gates have the 2nd selection pulse M2 as input and the pulses of next four pin of the DAQ Card. This configuration is shown in Fig:4.6

In second part of the driver circuit there are for Darlington pair transistors for each stepper motor. The out put of from the AND gates are given in the emmiter of the transistor through a resistor. Base of the transistor are grounded through a diode. And output is taken from the collector directly to the stepper motor. With this setup we can drive four stepper motors simultaneously.

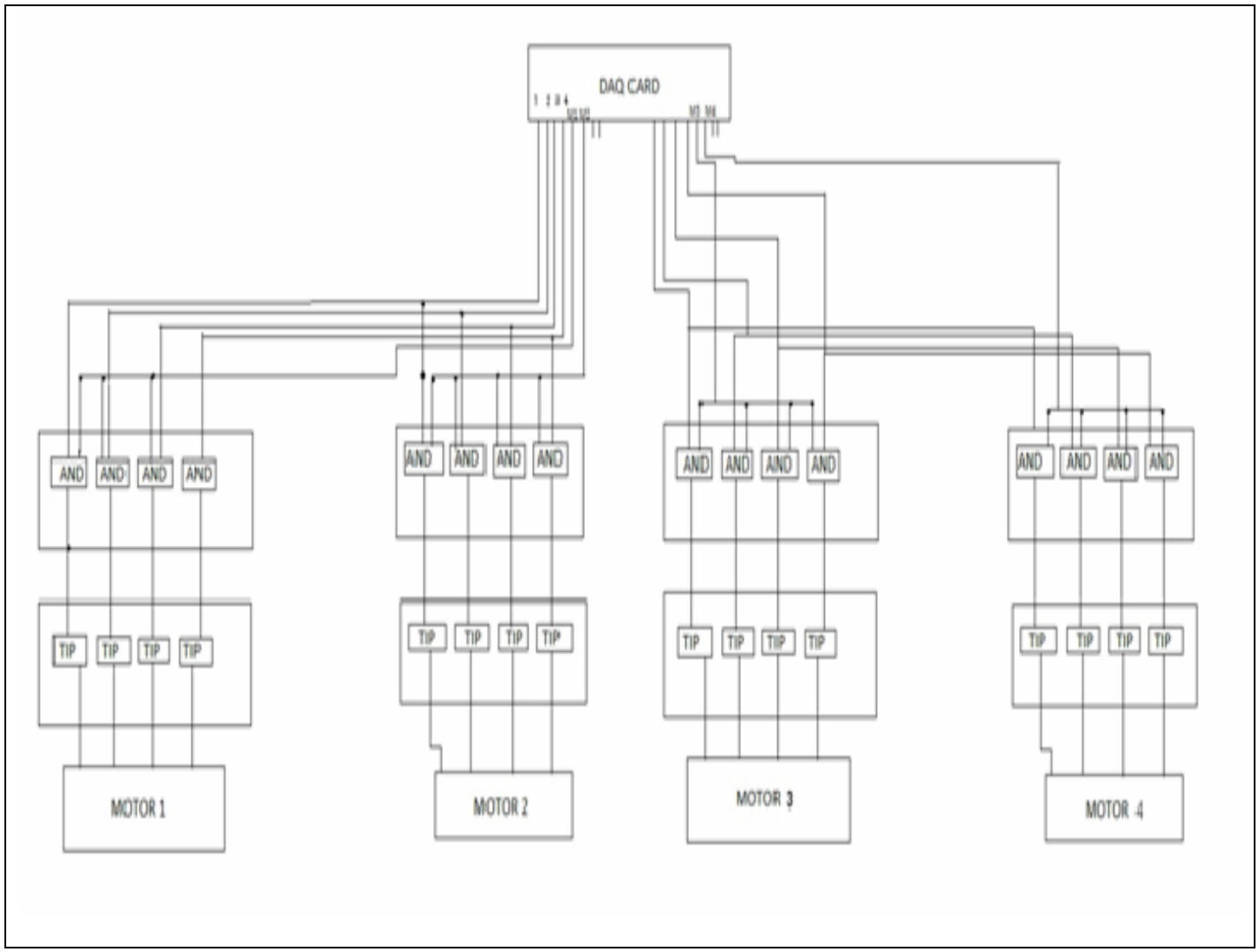


Fig 4.6: Block Diagram Stepper Motor Driver Circuit

## 4.3 DC Motor

DC motors are reliable, sturdy, and relatively powerful. It is used very commonly in industries.

The stator of DC motors is a set of fixed permanent magnets, which creates a fixed magnetic field, and the rotor carries the current. The direction of current is changed continuously through brushes and commentators, causing the rotor to rotate. Conversely, if the rotor is rotated within the magnetic field, a DC current will

develop and the motor will act as a generator though the output is DC, but not constant.

In our project we used DC motors in two places. One DC motor is used in the base of the mobile robotic arm with the wheels to control forward and backward movements. Another DC motor is used to control the gripper for both the arms.

### 4.3.1 DC Motor for the Gripper:

For the gripper of the robotic arms we used DC motor. Using a relay circuit, four combinations have been made to manipulate the direction of rotation of the DC motor and the gripper works accordingly. Details about Relay and the Relay circuit is discussed later.

### 4.3.2 Dc Motor for the movement of Robotic Arm :

For the movement of the mobile robotic arm we used DC motor.The same relay circuit is also used here. The movement is controlled in such a way so that the robot is able to move to a certain point and can also come back at the starting point.Two DC gear motors are connected with wheels to control the movement of Robot 1. To balance the movement of the base a supporting dummy wheel is used

The wheels moves on XY-plane, having one degree of freedom.

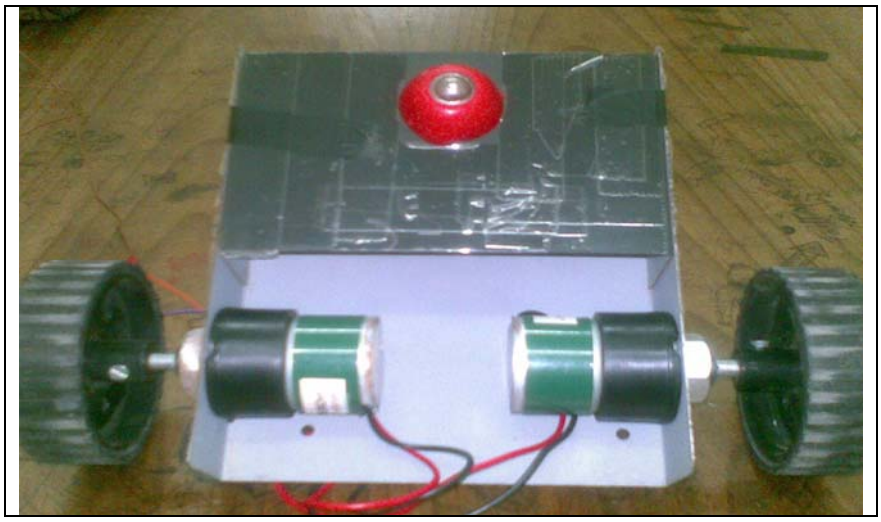

Fig 4.7: Wheel of the Mobile Robot

4.3.2.1 Relay and Relay Circuit

Relay:

Relay is an electrically operated switch. Current flowing through the coil of the relay creates a magnetic field which attracts a lever and changes the switch contacts. Relays have two switch positions. Where it is necessary to control a circuit by a low-power signal, or where several circuits or combinations must be controlled by one signal relays are used. A type of relay that can handle the high power required to directly drive an electric motor is called a Contactor (an electrically controlled switch used to switch power or control circuit). Solid-state relays control power circuits with no moving parts, instead using a semiconductor device to perform switching.

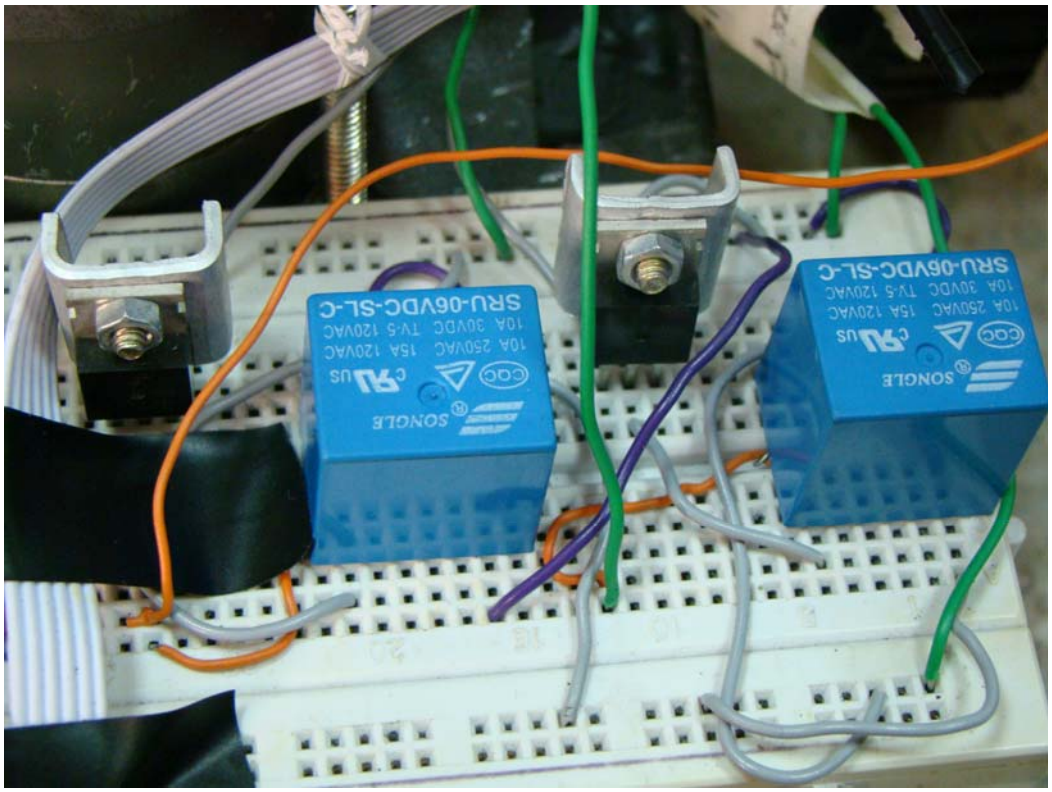


Fig 4.8: Relay Circuit

Relay Circuit:

A relay circuit is a circuit that uses a small mechanical switch or a semiconductor device to energize a relay, which will then close a contact set to complete another circuit. In this project, relay circuit is used to change the direction of rotation of the DC motor which is placed at the gripper and at the wheels of the mobile robotic arm. To make the relay circuit, we have used two 6 volt relays (SRU-06VDC-SL-C), two transistors (TIP 41). Input is coming from microcontroller's port D0 and D1 and connected to the base of each transistor. Each emitter is grounded and each collector is connected to the VCC through an electromagnet coil. Output is being collected from the 5$^{th}$ pin of each relay and these outputs are connected with the DC motor which is maneuvering the gripper. Four combinations are used to maneuver the gripper action. The combinations are-

1) 00 = gripper closes

2) 11 = gripper opens

3) 01 = no actions

4) 10 = no actions

When polarity changes, at different combinations, relay gives desired outputs and DC motor rotates clockwise and anti-clockwise and gripper performs desired actions accordingly as well.

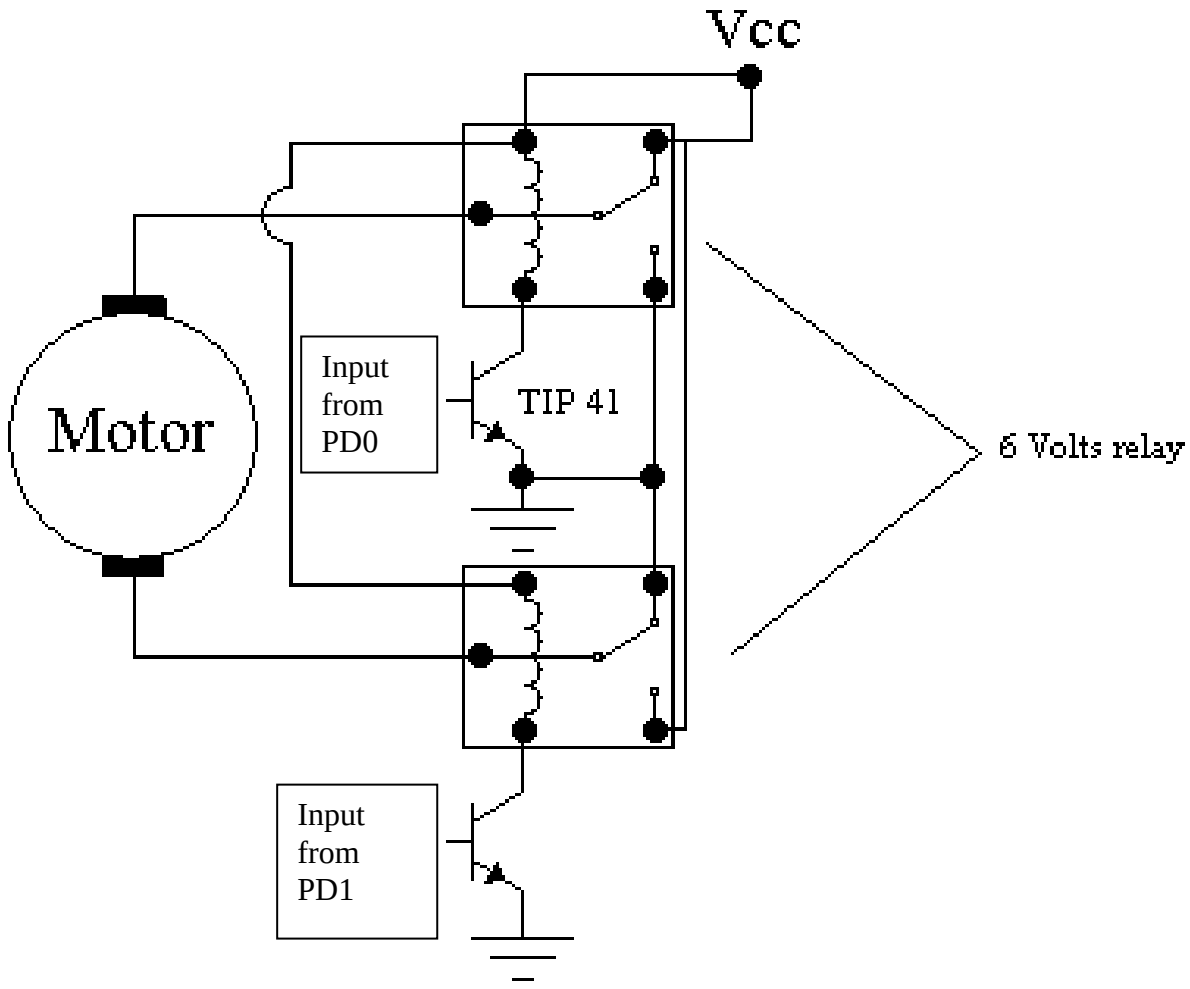


Fig .Relay Circuit Diagram

# CHAPTER V. MICROCONTROLLER & SENSOR

CHAPTER V. MICROCONTROLLER & SENSOR

5.1 Microcontroller:

In our project, we needed a device that can make decisions which way to rotate the DC motors in order to move the mobile robotic arm and also for the opening and closingof the gripper. This was achieved by using a microcontroller which is ATMEGA 32 (Fig:5.1 & Fig:5.2). This microcontroller is from the AVR ATMEL family with built in 32 kilobytes of memory. The ATMEGA 32 is a 40 pin microcontroller with 4 ports for inputs or outputs.

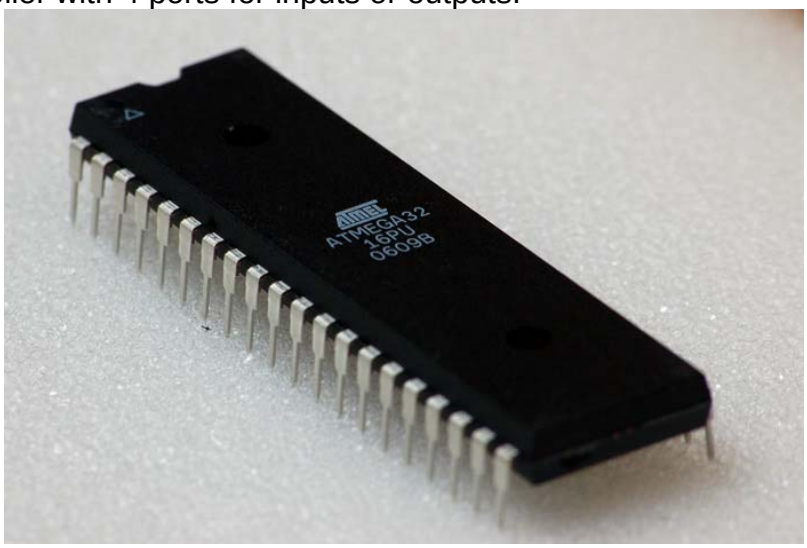


Fig:5.1 Atmega 32

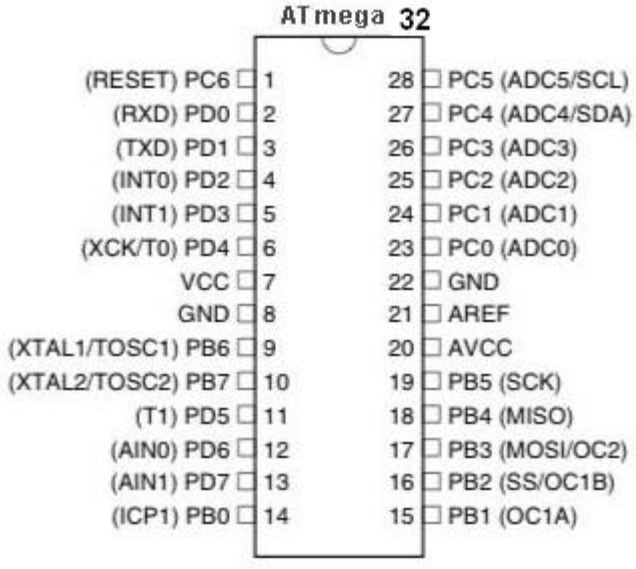


Fig:5.2 Pin Configuration of ATMEGA 32

From the figure it can be seen that Pin 10 is reserved for +VCC which is +5v and Pin 11 is reserved from ground. For ATMEGA 32 to work properly a supply of 5v is required. For our purpose we used Pin 15, 16, &17 and from PORT D as input.

These 4 ports take inputs from the inverters and makes proper decisions which pulse to give and how to rotate the DC motors. The output from the microcontroller is taken from the Pin 1 – 6. Outputs are connected to DC motors. The other two ports (PORT A and PORT C) are unused because they were not required. This ATMEGA 32 received input from the inverters and gives proper pulses which is responsible for the movement of the DC motors.

We used microcontroller for the grippers of the robots and for the wheel of the mobile
robot. The block diagram of the working principle of gripper is shown below.

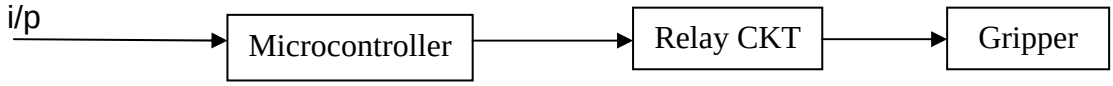


Fig: 5.3 Block Diagram of working principal of the gripper

For parallel processing it was recommended that the two robots should work synchronously. That's why we used a LDR as sensor.

### 5.2 Sensor circuit

In order to make the system completely automatic it is necessary for the system to track the exact time when robot 2 should start to work. So that it won't miss any object to pick. This has been achieved by using a special design consisting of a LDR where the intensity of light falling on each LDR is compared by a comparator and the result fed to the microcontroller that makes further decisions. Before feeding the result from the comparator to the microcontroller, we have used a inverter to amplify the output of the comparator for the microcontroller. The output of the comparator depends on the input from the sensors and since the microcontroller can only detect digital 0 or 1, so we had to use a inverter that will convert the output from the comparator into zero or 1, which can be understood by the microcontroller. Each of these associated hardware are discussed in the proceeding sections.

#### 5.2.1 LDR

LDR stands for Light Dependent Resistors. These are special kind of resistors whose resistance decreases with increasing incident light intensity [15]. LDR is also sometimes known as photo resistors. A photo resistor is made of a high resistance

semiconductor. If light falling on the device has high frequency, photons absorbed by the semiconductor give electrons enough energy to jump into the conduction band. The resulting free electron (and its hole partner) conduct electricity, thereby lowering resistance with the increase in light intensity.

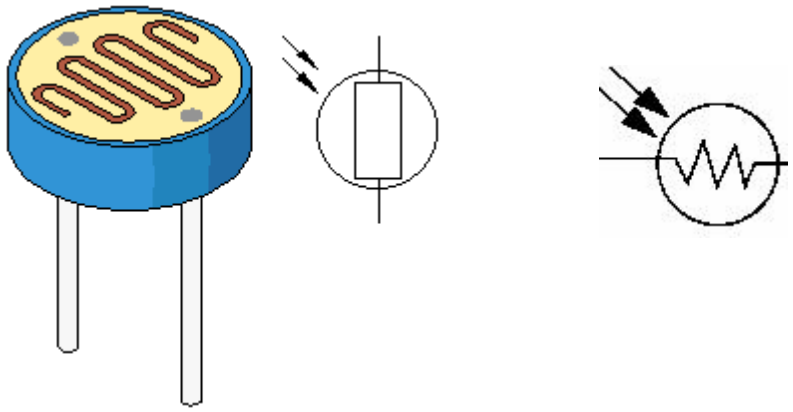

Fig. 5.4: Light Dependent Resistor (LDR) symbol

LDR are available in many sizes and shapes. For our project we decided to use a small sized LDR with acceptable response to the change of intensity of light since our prototype is comparatively small.

A constant LASER is given to the LDR. And thus microcontroller gets constant 1 as input. As soon as any object passes the LASER it gives 0 to microcontroller. A program is loaded in microcontroller so that the DC motor of the gripper of the mobile robotic arm will open when microcontroller will receive a 0 as input. The circuit is given below.

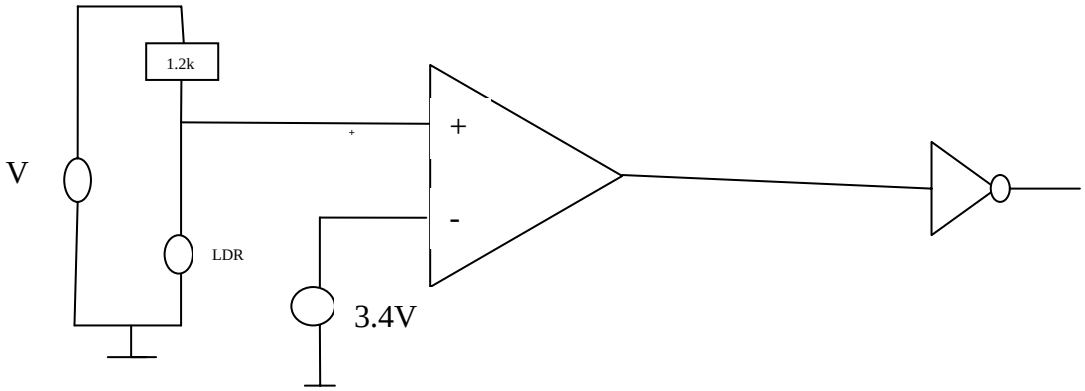


Fig. 5.5: Light Dependent Resistor (LDR) symbol

# CHAPTER VI. PARRELEL PROCESSING & MULTITASKING

## 6.1 What is Parallel Processing?

**Parallel Processing** refers to the concept of speeding-up the execution of a program by dividing the program into multiple fragments that can execute simultaneously, each on its own processor. In computers, parallel processing is the processing of program instructions by dividing them among multiple processors with the objective of running a program in less time. In the earliest computers, only one program ran at a time. A computation-intensive program that took one hour to run and a tape copying program that took one hour to run would take a total of two hours to run. An early form of parallel processing allowed the interleaved execution of both programs together. The computer would start an I/O operation, and while it was waiting for the operation to complete, it would execute the processor-intensive program. With the help of parallel processing, a number of computations can be performed at once, bringing down the time required to complete a project. Parallel processing is particularly useful in projects that require complex computations, such as weather modeling and digital special effects.

## 6.2 Parallel Processing in RT-Linux

RT-Linux supports systems in which multiple processors share a single memory and bus interface within a single computer. It is possible for a group of machines each running RT-Linux to be interconnected by a network to form a parallel-processing cluster. RT-Linux supports **SMP** (symmetric multiprocessing system) Pentium systems in which multiple processors share a single memory and bus interface within a single computer. It is also possible for a group of machines each running Linux to be interconnected by a network to form a parallel-processing **cluster**. Another alternative uses this system as a "host" for a specialized **attached**

**parallel processor**. A "new" fourth alternative is **SIMD parallelism within a register,** which is facilitated by the MMX (Multimedia extensions), which is also done in RT-Linux.

In our project we have done parallel processing in RT-Linux environment. One single kernel of RT-Linux has been used with single thread. Two separate robotic arms is able to work simultaneously. They perform ‘pick and place’ depending on the command from the PC under Real time Linux. (Fig.6.1)

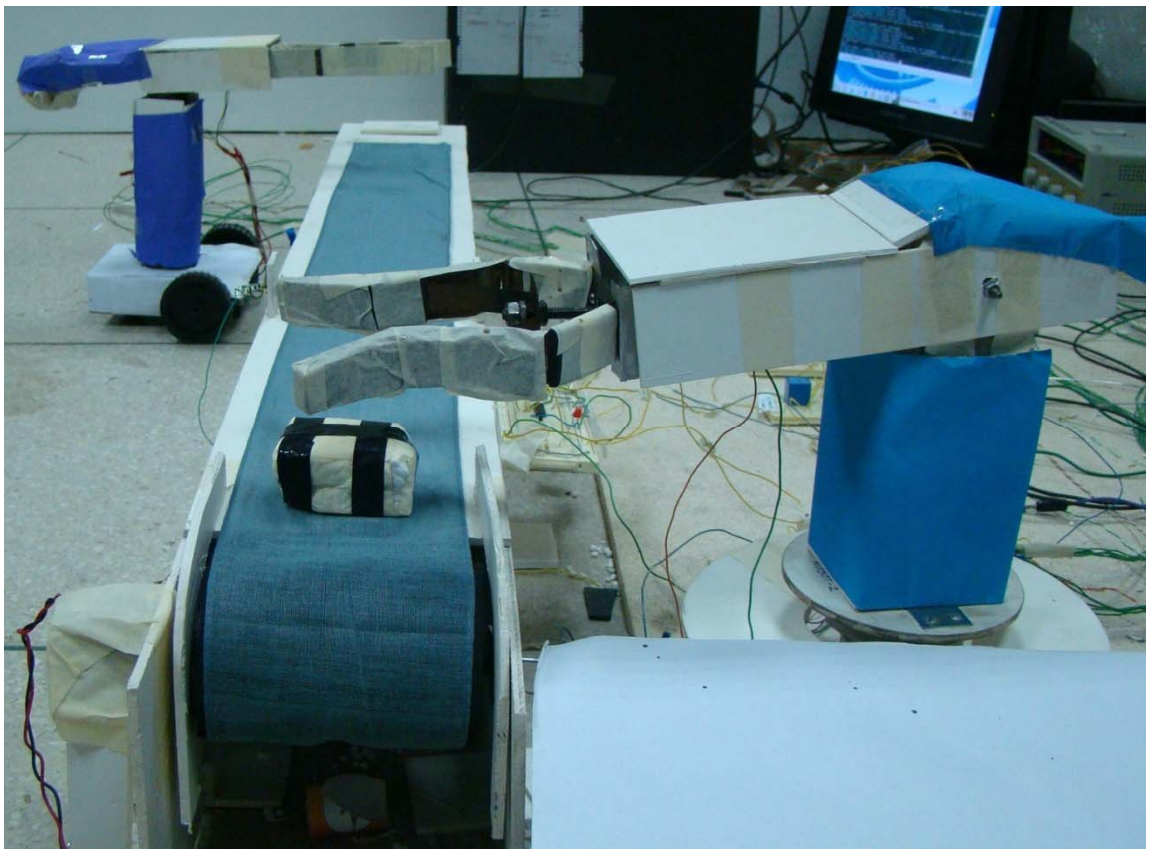

Fig: 6.1 Parallel Processing of two Robotic Arms

## 6.3 Multitasking

Multitasking is a method by which multiple tasks or processes are done by sharing common processing resources such as a CPU. It gives the ability to execute more than one task at the same time. Only one CPU is involved in the process and it switches from one program to another according to the program. In Robotics multi-tasking robots will save time.

Under RT-Linux environment we did the parallel processing with and without multitasking. As RT-Linux is real time and it can set priority so the multitasking applications did not make any change in the output waveforms.

### 6.1 What is Parrelel Processing?

Parallel Processing refers to the concept of speeding-up the execution of a program by dividing the program into multiple fragments that can execute simultaneously.Traditionally, multiple processors were provided within a specially designed "parallel computer".But RT-Linux supports systems in which multiple processors share a single memory and bus interface within a single computer.It is possible for a group of machines each running RT-Linux to be interconnected by a network to form a parallel-processing

# CHAPTER VII. PROJECT OVERVIEW

### 7.1 Working Principle

The robotic arm is able to rotate to multiple points. The first point of rotation occurs at the base of the device. This base contains a stepper motor. The base is mounted to the end of the arm. (Figure: 7.1) Temperature also plays a critical role in how any mechanical device operates. If the manipulator works in two (Hot & Cold) extreme climates the device should still function without any discrepancies. The Robotic arms are able to operate between -20° and 60° Celsius. At the base, the arm is capable of rotating 360 degrees about the z-axis.

Once the joint is rotated to the desired position about the z-axis, the attachment between the base and upper arm will have the capability of rotating about either the X or Y axes depending upon the orientation of the base. The shoulder joint connection and will mimic the motion of a hinge joint. The upper arm and lower arm will be connected by the wrist joint. This joint will be capable of rotating about the z-axis at a full range of 360 degrees. This joint will contain another Stepper Motor (Fig.7.2).The connection between the gripper and the lower arm of the device is stationary.

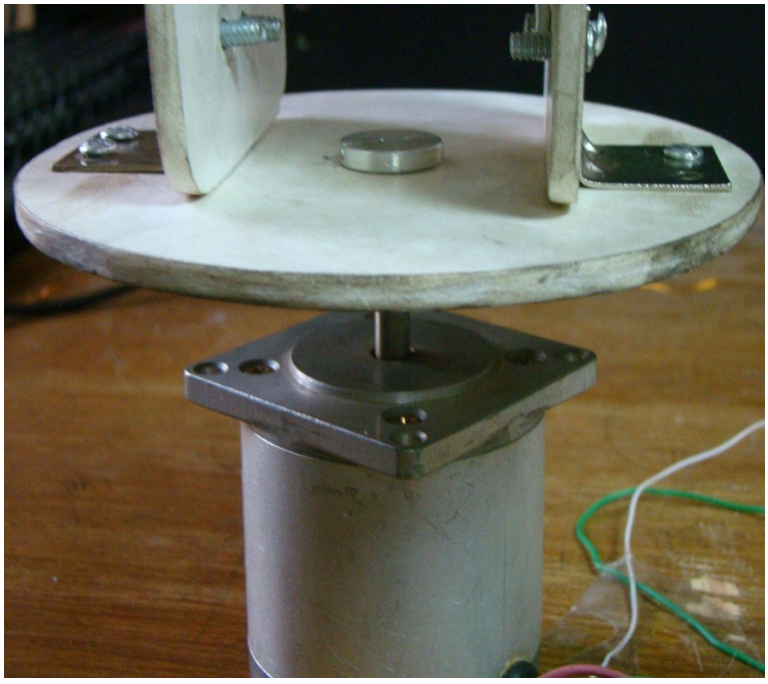

Fig 7.1: Base Motor of the Robotic Arm

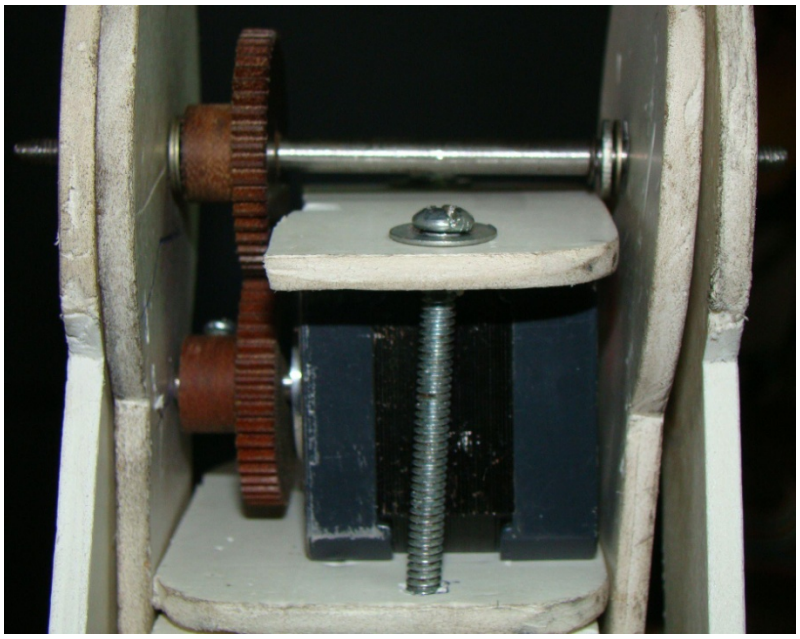

Fig 7.2: Middle Motor of the Robotic Arm

## 7.2 Industrial Application

The two robotic arms (mobile and static) can be used in industrial purpose. A conveyer belt is used in between the two robotic arms. The static robot will first pick an object and will place it on the conveyer belt. The belt is moving in a constant speed. There is a sensor on the belt. As soon as the object crosses the sensor it will send a signal to the gripper of the mobile robot and it will open and will grip the object and then it will go some distance and place the object in a fixed place.

# CHAPTER VIII. ALGORITHM

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <rtl_time.h>
#include <rtl_sched.h>
#include <asm/io.h>
#include <time.h>


#define base 0x220
//Lower 8 bit (DOL)
#define   DOL       base+13//
//Upper 8 bit (DOH)
#define   DOH       base+14

int period = 100000000;
int period2= 100000000;
int periodic_mode=0;

int i = 0;
int j = 0;
int k = 0;
int l = 0;

pthread_t thread;
pthread_t thread2;

//Thread 1
void *bit_toggle(void *t)
{
      outb(0x00,DOH);

           while(i<6){
           outb(0x00,DOH);
           reverseMotorThree();
           i++;
           }

       i = 0;
           while(i<6){
           outb(0x00,DOH);
           delay();
           i++;
           }

           while(k<5){
           outb(0x00,DOH);
           specialMotorThree();
       k++;
           }

           k =0;

           while(k<8){
           outb(0x00,DOH);
           specialMotorFour();
       k++;
           }

        i = 0;

           while(i<5){
           outb(0x00,DOH);
```

```
            reverseMotorThree();
            i++;
            }

        i = 0;
            while(i<2){
            outb(0x00,DOH);
            delay();
            i++;
            }

        k = 0;
            while(k<4){
            outb(0x00,DOH);
            specialMotorThree();
        k++;
            }
            k = 0;

            while(k<8){
            outb(0x00,DOH);
            reverseMotorFour();
        k++;
            }

// dely for convinent belt

            i = 0;
            while(i<15){
            outb(0x00,DOH);
            delay();
            i++;
            }

        i = 0;
            while(i<5){
            outb(0x00,DOL);
            specialMotorOne();
            i++;
            }

            i = 0;
            while(i<4){
            outb(0x00,DOH);
            delay();
            i++;
            }

            i = 0;
            while(i<6){
            outb(0x00,DOL);
            reverseMotorOne();
            i++;
            }


            i = 0;
            while(i<15){
            outb(0x00,DOL);
            specialMotorTwo();
            i++;
            }
```

```
        i = 0;
            while(i<9){
            outb(0x00,DOH);
            delay();
            i++;
            }

            i = 0;
            while(i<5){
            outb(0x00,DOL);
            specialMotorOne();
            i++;
            }

            i = 0;
            while(i<5){
            outb(0x00,DOL);
            reverseMotorOne();
            i++;
            }

            i  = 0;

            while(i<15){
            outb(0x00,DOL);
            reverseMotorTwo();
            i++;
            }
}

//thread 2 for parallet processing
void *bit_toggle2(void *t)
{

}


int
init_module(void)
{
            pthread_attr_t attr;
            struct sched_param sched_param;



            pthread_attr_init (&attr);
            sched_param.sched_priority = 1;
            pthread_attr_setschedparam (&attr, &sched_param);

            pthread_create (&thread,  &attr, bit_toggle, (void *)0);

            pthread_create (&thread2,  &attr, bit_toggle2, (void *)0);



            pthread_make_periodic_np(thread, gethrtime(), period);

            pthread_make_periodic_np(thread2, gethrtime(), period2);

            return 0;
}
```

```
void
cleanup_module(void)
{
        pthread_delete_np (thread);
        pthread_delete_np (thread2);

}

//Motor 1
void motorOne()
{
    outb(0x01,DOL);
        pthread_wait_np();

        outb(0x02,DOL);
        pthread_wait_np();

    outb(0x04,DOL);
        pthread_wait_np();

        outb(0x08,DOL);
        pthread_wait_np();
}

//Motor 2
void motorTwo()
{
        outb(0x10,DOL);
        pthread_wait_np();

        outb(0x20,DOL);
        pthread_wait_np();


        outb(0x40,DOL);
        pthread_wait_np();

    outb(0x80,DOL);
        pthread_wait_np();

}

//Motor 3
void motorThree()
{
        outb(0x01,DOH);
        pthread_wait_np();

        outb(0x02,DOH);
        pthread_wait_np();

    outb(0x04,DOH);
        pthread_wait_np();

        outb(0x08,DOH);
        pthread_wait_np();
}

//Motor 4
void motorFour()
{
```

```
            outb(0x10,DOH);
            pthread_wait_np();

            outb(0x20,DOH);
            pthread_wait_np();

        outb(0x40,DOH);
            pthread_wait_np();

            outb(0x80,DOH);
            pthread_wait_np();
}


//Special Motor 1
void specialMotorOne()
{
        outb(0x11,DOL);
            pthread_wait_np();

            outb(0x12,DOL);
            pthread_wait_np();

        outb(0x14,DOL);
            pthread_wait_np();

            outb(0x18,DOL);
            pthread_wait_np();
}

//Special Motor 2
void specialMotorTwo()
{
            outb(0x21,DOL);
            pthread_wait_np();

            outb(0x22,DOL);
            pthread_wait_np();


            outb(0x24,DOL);
            pthread_wait_np();

        outb(0x28,DOL);
            pthread_wait_np();
}


//Special Motor 3
void specialMotorThree()
{
        outb(0x11,DOH);
            pthread_wait_np();

            outb(0x12,DOH);
            pthread_wait_np();

        outb(0x14,DOH);
            pthread_wait_np();

            outb(0x18,DOH);
            pthread_wait_np();
```

```
}

//Special Motor 4
void specialMotorFour()
{
        outb(0x21,DOH);
        pthread_wait_np();

        outb(0x22,DOH);
        pthread_wait_np();

        outb(0x24,DOH);
        pthread_wait_np();

    outb(0x28,DOH);
        pthread_wait_np();
}


//Special motor 1 Reverse direction
void reverseMotorOne()
{
        outb(0x18,DOL);
        pthread_wait_np();

        outb(0x14,DOL);
        pthread_wait_np();


        outb(0x12,DOL);
        pthread_wait_np();

        outb(0x11,DOL);
        pthread_wait_np();
}

//Special motor 2 Reverse direction
void reverseMotorTwo()
{
        outb(0x28,DOL);
        pthread_wait_np();


        outb(0x24,DOL);
        pthread_wait_np();


        outb(0x22,DOL);
        pthread_wait_np();

        outb(0x21,DOL);
        pthread_wait_np();
}

//Special motor 3 Reverse direction
void reverseMotorThree()
{
        outb(0x18,DOH);
        pthread_wait_np();

    outb(0x14,DOH);
        pthread_wait_np();
```

```
            outb(0x12,DOH);
            pthread_wait_np();

        outb(0x11,DOH);
            pthread_wait_np();
}

//Special motor 4 Reverse direction
void reverseMotorFour()
{
            outb(0x28,DOH);
            pthread_wait_np();

            outb(0x24,DOH);
            pthread_wait_np();

            outb(0x22,DOH);
            pthread_wait_np();

            outb(0x21,DOH);
            pthread_wait_np();
}

//manual delay
void delay()
{
            outb(0x00,DOH);
            pthread_wait_np();

            outb(0x00,DOH);
            pthread_wait_np();

            outb(0x00,DOH);
            pthread_wait_np();

            outb(0x00,DOH);
            pthread_wait_np();

            outb(0x00,DOH);
}
```

# CHAPTER IX.  CONCLUTION

We have some limitations which are mainly in  mechanical. We are very much happy to implement successfully the desired system in the lab. Now our main aim is  to implement the entire system for industrial purpose.

**Reference:**


1. http://sachin-robotics.blogspot.com/2008/01/how-to-build-microcontroller-based.html
2. csited.org/2007/34OlawCSITEd.pdf
3. eia.udg.es/~forest/VLSI/lect.10.pdf
4. www.thomasnet.com/catalognavigator.html
5. Flexible, Low-mass Robotic Arm Actuated by Electroactive Polymers and Operated Equivalently to Human Arm and Hand by Y. Bar-Cohen1, T. Xue, M. Shahinpoor, J. Simpson, and J. Smith
6. Exploiting A Real-Time Linux platform in controlling robotic manipulators by C.Bellini, F. Panepinto, S.Panzieri ,G.Ulivi, 15th Triennial World Congress, Barcelona, Spain
7. RePLiCS: An Environment for Open Real-Time Control of a Dual-Arm Industrial Robotic Cell Based on RTAI-Linux, Fabrizio Caccavale, Vincenzo Lippiello, Bruno Siciliano, Luigi Villani
8. www.robot.lth.se
9. www.rtlinux.org
10. www.edinformatics.com/math.../**robotics**/**robotics**1.htm
11. http://www.nasa.gov/audience/foreducators/robotics/home/what_is_robotics_58.html
12. http://www.cs.cmu.edu/~chuck/robotpg/robofaq/1.html
13. http://cobweb.ecn.purdue.edu/~pplinux/http://www.microcontroller.com/
14. http://www.microcontroller.com/
15. "Photoresistor - Wikipedia, the Free Encyclopedia." *Main Page - Wikipedia, the Free Encyclopedia*. Web. 12 Apr. 2010.