www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 4 Issue 4 April 2015, Page No. 11704-11717

Design And Implementation Of A Robot ARM Control System

Prof. Hycinth C. Inyiama[1], Prof Christiana .C. Okezie[2], Maxwell C. Okeke[3], Tochukwu C. Akubue[4]

[1]*Electronic & Computer Engineering Department, NnamdiAzikiweUniversity,Awka,Anambra State, Nigeria*

[2]*Electronic & Computer Engineering Department,NnamdiAzikiweUniversity,Awka, Anambra State, Nigeria*

[3]*World University of Information and Communication Technology, Enugu State, Nigeria*

[4]*General Studies Department, Enugu State Polytechnic, Iwollo, Enugu State, Nigeria.*

Abstract - **The Design and Implementation of Robot Arm Control System is the subject of this thesis. To achieve the prototype, an appropriate payload was chosen and its torque requirements were carefully calculated. This formed the basis of selecting a motor of appropriate torque. A permanent magnet Stepper Motor was chosen for use in constructing the prototype because it delivers a known incremental step for each pattern sent to the motor. The prototype used four stepper motors, one at the ankle, one at the base, one at the wrist and one at the shoulder. This made possible, four axis of movement for the Robot Arm. Although a stepper motor can be used to carry out different process control tasks, the particular task assigned to the prototype is a pick and drop process control. The actions required for the Robot Arm during a pick and drop process was subdivided into twelve sub tasks. Each sub task led to a flowchart which guided the development of the corresponding microcontroller software that helps to achieve the sub tasks. A main program was designed and developed which calls each of the twelve sub-routines for pick and drop activities in sequence, at appropriate times in order to achieve the pick and drop process. The 8051 Microcontroller was selected because of its ready availability as well as its features that match the pick and drop operation. A fully expanded state transition table and a content addressable memory were used to achieve smooth transitions of the Robot control stepper motors. User friendliness was achieved by providing a Liquid Crystal Display and a Keypad. The system can also be reprogrammed to perform other tasks such as painting, welding, pounding, among others, provided the end effectors are modified to match the operation envisaged. The prototype testing and evaluation show that it met the criteria listed in the design objectives.**

Keywords - **Robot arm, Control system, Prototype, Payload, Torque, Microcontroller, Stepper motor**

INTRODUCTION

In the modern world, robotics has become popular and has achieved great successes in several fields of human endeavour. Robotics has become very useful in medicine, education, military, research and mostly, in the world of manufacturing. It is a term that has since been used to refer to a machine that work to assist people or do work that humans find difficult or undesirable. Robots, which could be destructive or non-destructive, perform tasks that would have been very tedious for human beings to do. They are capable of carrying out repetitive tasks more quickly, cheaply, and accurately than humans (Desmond Jeffery Will, January 2004).

Robotics involves the integration of many different disciplines; among them are kinematics, signal analysis, and artificial intelligence. These disciplines when applied suitably, lead to the design of a very successful robot (Desmond Jeffery Will, January 2004). With the pressing need for increased productivity and delivery of end products of uniform quality, industry is turning more and more to computer-based automation (RobotShop Distribution Inc., 2008, [www.robotshop.ca]). At the present time, most of industrial automated manufacturing is carried out by special purpose machines - robots, designed to perform specific functions in a manufacturing process. The inflexibility and generally high cost of these machines often referred to as hard automation systems, have led to a broad-based interest in the use of robots capable of performing a variety of manufacturing functions in a more flexible working environment and at lower production costs.

A robot can be in the form of reprogrammable general-purpose manipulator that can perform various assembly tasks. A robot may possess intelligence, which is normally due to computer algorithms associated with its controls and sensing systems. Industrial robots are general-purpose, computer-controlled manipulators consisting of several rigid links connected in series by revolute or prismatic joints.

Most of today’s industrial robots, though controlled by mini and microcomputers are basically simple positional machines. They execute a given task by playing back a pre-recorded or pre-programmed sequence of motions that has been previously guided or taught by the hand-held control teach box. Moreover, these robots are equipped with little or no external sensors for obtaining the information vital to its working environment.

As a result robots are used mainly for relatively simple, repetitive tasks. More research effort has been directed in sensory feedback systems, which has resulted in improving the overall performance of the manipulator system (RobotShop Distribution Inc., 2008, [www.robotshop.ca]).

I. OVERVIEW OF THE ENVISAGED SYSTEM

The block diagram overview of the envisaged system is shown in figure 1. The envisaged system (robot arm control system) is made up of a pick and drop control system (A), four stepper motor control systems (B, C, D, and E), a keypad (F), and four stepper motor drives (G, H, I, and J).The robot arm to be controlled has four joints. Stepper motors, one at each joint are used to aid the robot arm joint movement. Each of the stepper motor at the robot arm joint has a stepper motor drive connected to it and the stepper motor drives are connected to a corresponding stepper motor control system. The robot-arm control system has four ROM-Based stepper motor control systems, all of which are connected to the pick and drop control system.The pick and drop control system is made up of a microcontroller (K) and four switches (H/Fs). The switches, one for each stepper motor control system are used to set the stepper motor control in a half step or full step mode. The half step mode would cause a smoother movement of the stepper motor and makes twice the number of steps that the full step mode would make in order to complete one complete revolution. Control software embedded in the microcontroller is used to direct the controller to select the stepper motor control systems (one at a time). The microcontroller operates the stepper control systems one at a time by sending clock pulses and clockwise/counter clockwise (CW/CCW) signal to the required stepper control system. The sequence of stepper motor control selection and the extent of the stepper motor movement are controlled via software sub routines. The actions of the microcontroller are initiated using the keypad.

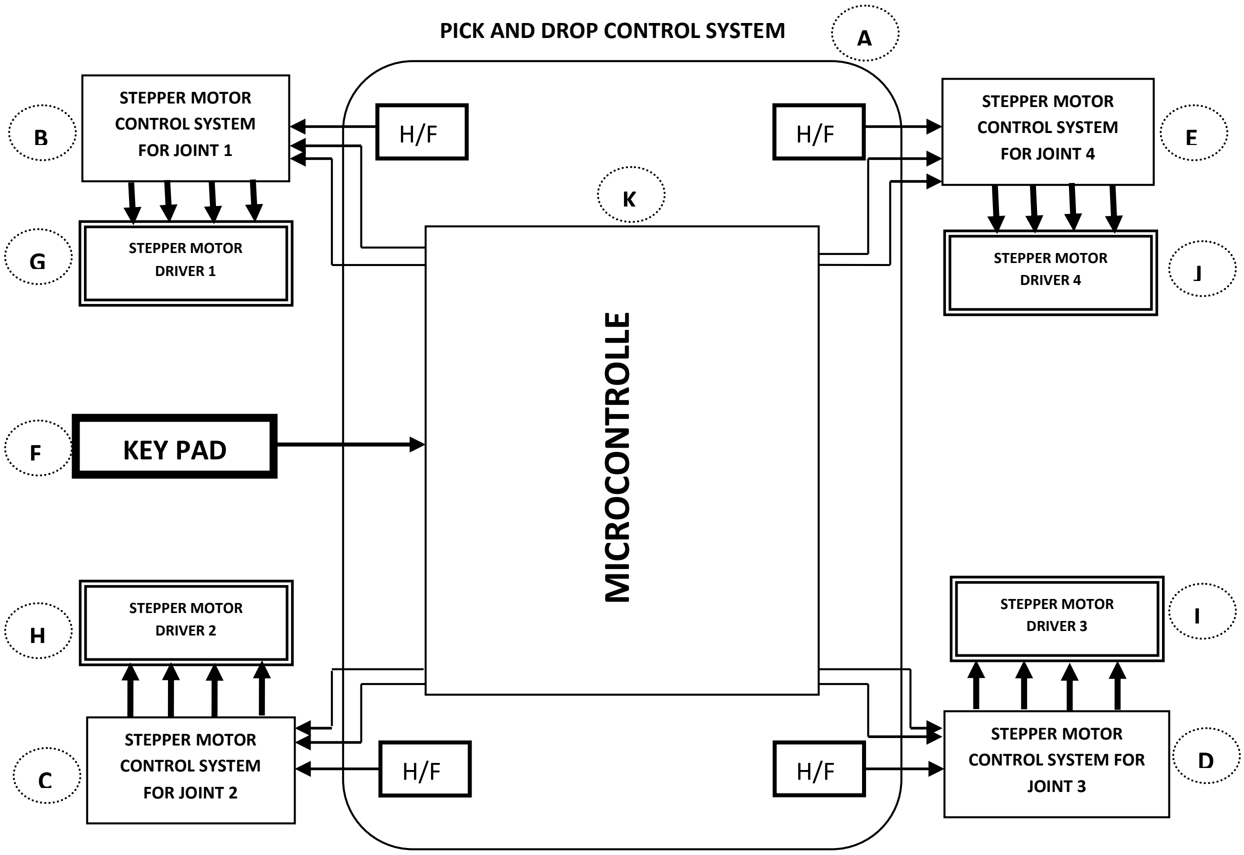

PICK AND DROP CONTROL SYSTEM
A
B
STEPPER MOTOR CONTROL SYSTEM FOR JOINT 1
H/F
H/F
STEPPER MOTOR CONTROL SYSTEM FOR JOINT 4
E
K
G
STEPPER MOTOR DRIVER 1
STEPPER MOTOR DRIVER 4
J
MICROCONTROLLE
F
KEY PAD
H
STEPPER MOTOR DRIVER 2
STEPPER MOTOR DRIVER 3
I
C
STEPPER MOTOR CONTROL SYSTEM FOR JOINT 2
H/F
H/F
STEPPER MOTOR CONTROL SYSTEM FOR JOINT 3
D

Figure 1: Block Diagram Overview of the Envisaged System

III. THE ROBOT ARM DESIGN

The design process is sectioned into the following:

- *Determination of the Payload***:** Payload is the carrying capacity of the robot arm, usually measured in terms of weight. Determining the payload is the first aspect of the system design. In this work, the payload includes the end of arm tooling and the object the robot can carry.
- *Selection of Appropriate Stepper Motor to Match the Payload:* In order to select an appropriate stepper motor to match the payload, the torque of the stepper motor needs to be considered. Torque is the tendency of the

stepper motor force to rotate an object about its axis, fulcrum or pivot. The value of the torque is determined by the product of weight value of the payload, weight of the robot link, and the length value of the payload.

From figure 2, if the robot link weight is not negligible, the weight value of the link and its distance of the stepper motor from the centre of gravity will be considered.

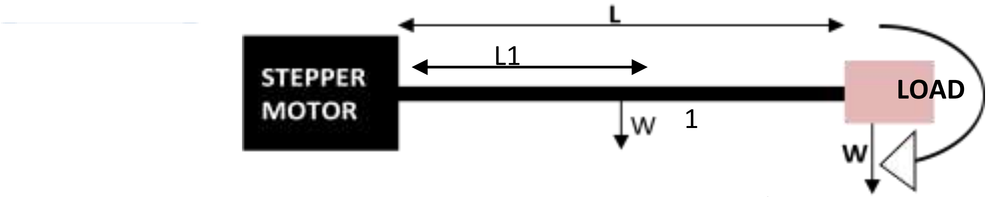


Figure 2: A Stepper Motor, the Linking Component and the Weight of the Link and Payload

The force to be considered will be the sum of the weight of the component (link) and the weight of the load. i.e.

$F = W1 + W$ 3

Thus the torque

$T = T1 + T2$..4

$T1 = L1 * W1$ 5

$T2 = L * W$ 6

$T = T1 + T2$ 7

$T = (L1 * W1) + (L * W)$ 8

Where
T is the total torque
T1 is the torque required to move the robot arm component
T2 is the torque required to move the payload
W1 is the robot arm component weight and
W2 is the payload weight

- *Design of Stepper Motor Drives*: The stepper motor drive is an IC that coverts the control bit pattern into a corresponding motion. The stepper drive input is connected to the input of the stepper motor. The stepper motor drive is made up of transistors. Single transistors switches are used when a 4 phase stepper motor whose coil current demand is low enough to allow direct switching with logic inputs as shown in Figure 3. Darlington pair is used when the coils current demand is high before there will be switching with the logic with logic inputs as shown in Figure 4.

Rb is a base bias resistor chosen such that:

$$\frac{Vin - Vbe}{Rb} = Ib \quad ....9$$

$$hFE = \frac{Ic}{Ib} \quad ....10$$

$$Rb = \frac{(Vs - 0.7)}{Ib} \quad ....11$$

Where
Vbe is the inherent internal (reverse) base emitter voltage of a transistor (typically 0.7 volts for silicon transistor)
hFE is the d.c. Current gain of the transistor switch.
Vs is the load voltage
Ib is the base current and
Ic is the collector current

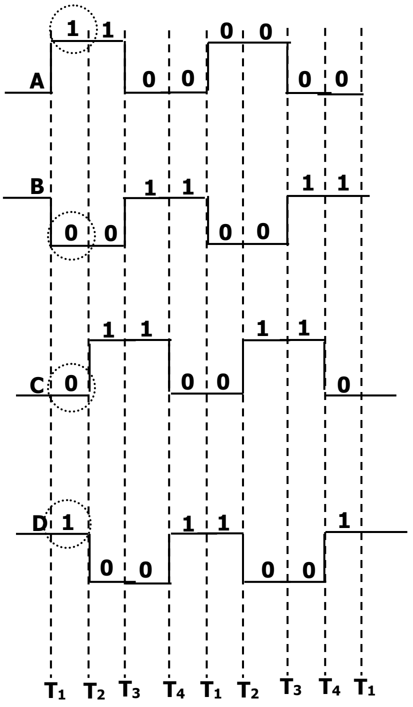


Figure 3: Wave Form Diagram for a four Phase

- *Full Step Control Bit Pattern:*The control bit pattern refers to the digital signals used to ON or OFF the stepper motor winding current. The wave form diagrams usually contain sufficient information for the coil bit pattern sequence to be built up. From the four phase stepper motor waveform in Figure 5, the columns demarcated with dotted lines in the wave form diagrams indicate the discrete time intervals within which one full step is taken by the stepper motor (1 = ON and 0 = OFF).

During the time interval starting from T_1 (column 1), ABCD = 1001. During T_2 (column 2), ABCD = 1010. During T_3 (column 3), ABCD = 0110. During T_4 (column 4), ABCD = 0101.

After the time interval T_4, the bit patterns start again from the time interval T_1 (ABCD = 1001 – encircled with dotted lines). These sequences continue to abet the stepper motor movement.

For normal full step operation of the Stepper Motor (SM), the energization bit patterns used to control coils ABCD must correspond to the ones derived from the waveform diagram (Figure 3) of the coils during each of the step intervals. It is the responsibility of the stepper motor controller to ensure that this is so. The stepper motor takes one full angular step if rotary or one full linear step if linear in a clockwise direction or forward direction for every translation in a full step sequence.

- *Half Step Control Bit Patterns:* Half-Step Control Bit Patterns replaces the double change at T_2 in the waveform diagram of Figure 3 with a two-stage single change sequence. This yield a new stepping sequence called the half step sequence shown in table 1.

Table 1: Half-Step Control Bit Patterns for a Four Phase Stepper Motor

S/N	**STEP**	**A**	**B**	**C**	**D**
0	**1**	**1**	**0**	**0**	**1**
1	**2**	**1**	**0**	**0**	**0**
2	**3**	**1**	**0**	**1**	**0**
3	**4**	**0**	**0**	**1**	**0**
4	**5**	**1**	**1**	**1**	**0**
5	**6**	**0**	**1**	**0**	**0**
6	**7**	**0**	**1**	**0**	**1**
7	**8**	**0**	**0**	**0**	**1**

There are twice as many rows as possible in a half step sequence as compared to the corresponding full step sequence. This is because each full step entry has been broken down into two discrete half-step entries in order to obtain the half step sequence. The stepper motor using the half step control bit pattern covers half of the full step distance for every transition provided that the consecutive half step transition occurs at half the normal full step time intervals. Then speed remains unchanged, but its movement is smoother and the torque level develops higher than that for the full step sequence.

- *Fully Energized Control Bit Patterns:* The fully energized control bit pattern is shown in the INDEX. It contains four set of bit patterns which are -
  - Half bit and Counter Clockwise direction bit patterns.
  - Half bit and Clockwise direction bit patterns.
  - Full bit and Counter Clockwise direction bit patterns.
  - Full bit and Clockwise direction bit patterns.

The Fully Energized Control Bit Patterns shown in INDEX is stored in the four 2K Read Only Memory which is accessible by the microcontroller. The controller uses the hexadecimal addresses of the patterns stored in the ROMs.

- *Design of Stepper Motor Control System:* The stepper motor control system is shown in Figure 4. It is made up of a Read Only Memory (ROM) and a quad D Flip-Flop. The ROM is used to store the Fully Expanded Control Bit Pattern. Though a 32 byte ROM is sufficient to store the control bit pattern, a 2 kilobyte ROM was used in the course of this work because it is the smallest size of ROM that can be found in the local market. A 2 kilobyte ROM is readily available and inexpensive (about one Dollar) in the local market. The ROM output is connected to the D-input of the quad D type flip-flops. The ROM receives the Half/Full (H/F) step signal and the Clockwise/Counter Clockwise (CW/CCW) signal from the pick and drop system. The ROM uses these signals and the stepper motor present control bit pattern to form the address of the next control bit pattern to output to the stepper motor drive.

When a clock pulse from the microcontroller is sent to the clock input of the quad D Flip-Flop, the present state of the stepper motor controller is sent through the Q output of the Flip-Flop to the stepper motor system and also feedback to the ROM address inputs. The present state, the H/F and CW/CCW signals are used to form the address of the next bit pattern (Figure 4).

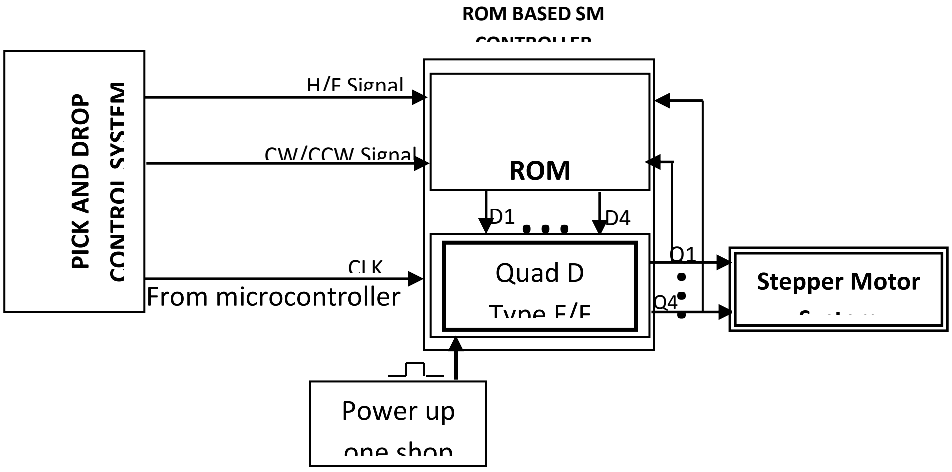


Figure 4: Stepper Motor Control System

IV. THE STEPPER MOTOR SYSTEM

The stepper motor system shown in Figure 3.9 comprises of the stepper motor drive and the stepper motor. The stepper motor drive is made up of four Darlington pair transistors because of the 4 phase stepper motor that was used in the work. The stepper motor drive converts the bit pattern from the ROM-Based controller into a corresponding motion, and causes the stepper motor to move towards the required point, one step at a time. The number of clock pulses from the microcontroller determines the number of steps taken by the stepper motor, one step per clock pulse. Figure 3.9 shows the stepper motor system with the coil connections.

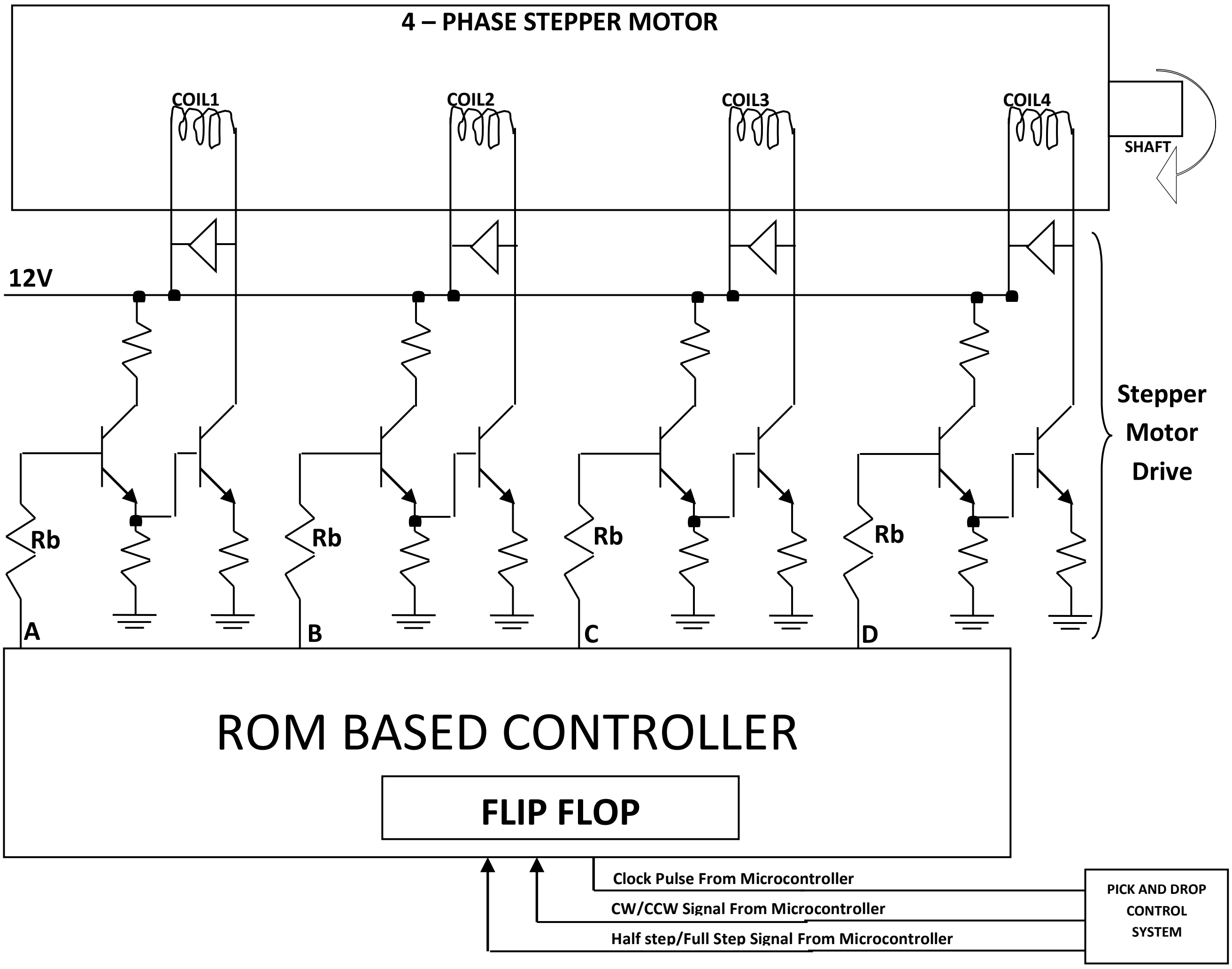

4 – PHASE STEPPER MOTOR
COIL1
COIL2
COIL3
COIL4
SHAFT
12V
Stepper Motor Drive
Rb
Rb
Rb
Rb
A
B
C
D
ROM BASED CONTROLLER
FLIP FLOP
Clock Pulse From Microcontroller
CW/CCW Signal From Microcontroller
Half step/Full Step Signal From Microcontroller
PICK AND DROP CONTROL SYSTEM

Figure 5: Stepper Motor System

When the stepper motor controller sends a control bit pattern to the stepper motor drive, the stepper motor drive energizes the coils of the stepper motors as appropriate for the bit pattern received. The energized coils cause the stepper motor shaft to move in accordance with the bit pattern sent to it. The stepper motor used in this work makes 1.8^o rotary motion per step command received (Figure 5).

Because the robot arm has four joints that need to be controlled for the pick and drop operation, and each joint has its own stepper motor control system, the entire Robot Arm Control System is therefore as shown in Figure 6.

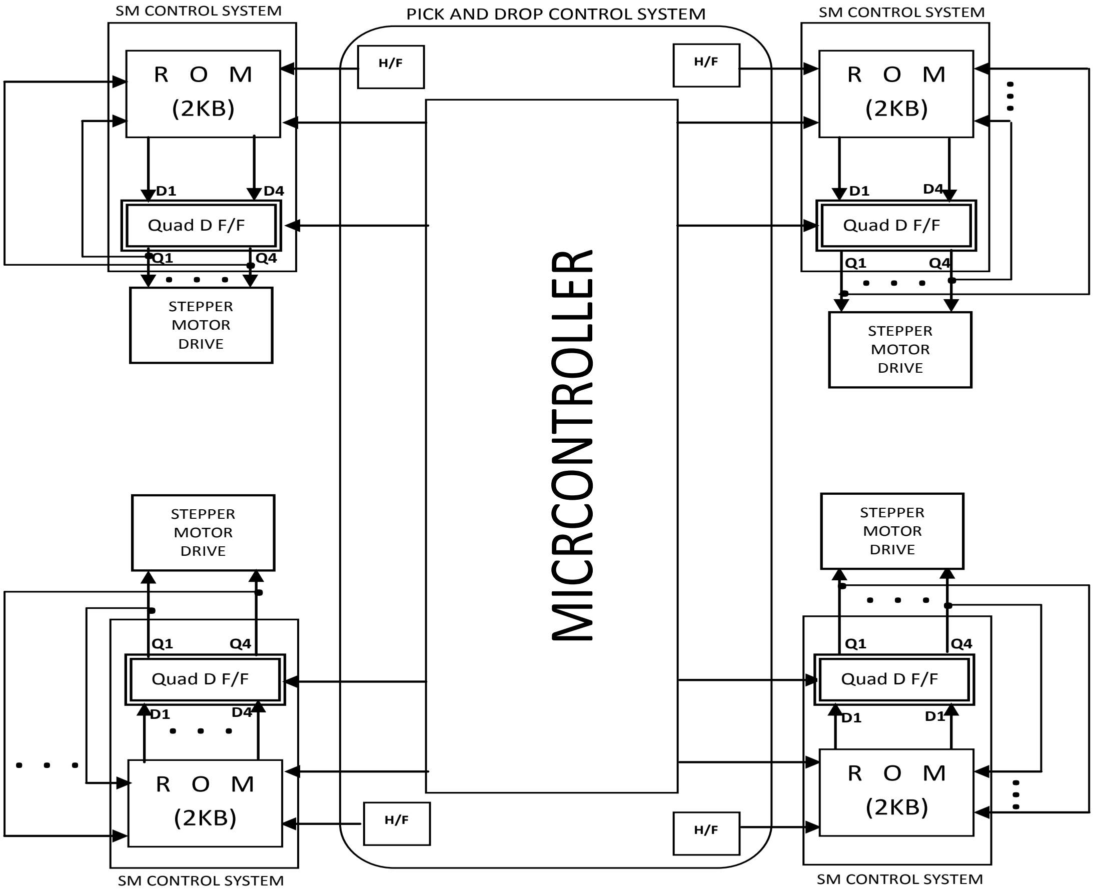


Figure 6: Complete Diagram of Robot Arm Control System

- *Control Software Design for the Pick and Drop Activities:*Control software is designed to direct the movement of the robot arm. Top down design approach is used to design the control software of this work. The robot arm movement is divided discrete motion path. An algorithm is developed for each motion path, and a corresponding flowchart also was developed for the algorithms. The robot arm control program code was developed from the flowcharts as shown in Appendix A. thus, the software design effort include the development of an algorithm, development flowcharts from the algorithm and writing a corresponding program from the flowcharts and algorithm.
- *Development of Pick and Drop Process Algorithm and Flowchart:*The robot arm is designed to move in a manner specified by the algorithm using one stepper motor (one joint movement) at a time. The stepper motors move one at a time in a specific sequence to achieve a “TO” movement and in a likewise manner, they move in a reverse sequence to achieve a “FRO” movement. This specific sequence is determined by the algorithm.

The algorithm used for the robot arm movement sequence is as follows –

a) Move Robot’s for arm up.
b) Move the robot arm to face the objects position.
c) Move the robot to stretch out its fore arm to reach the object’s position.
d) The robot lowers the fore arm so that the object is now in between the robots gripper.
e) The gripper grips the object that is now in between its two fingers.
f) With the objects still gripped in between the fingers, the robot raises the fore arm.
g) The robot now takes back the forearm still gripping the object.
h) It then rotates to the objects destination.
i) The fore arm comes forward again to drop the object.
j) The fore arm comes down to the plain.
k) The gripper then opens to leave the object on the plain surface in its destination.
l) The entire system then goes back to its origin.

Each of the stepper motors makes two distinct movement in a one complete pick and place activities .i.e. one forward movement in the “TO” and one in the “FRO” movement.

A corresponding flowchart, Figure 7 through Figure 16 is drawn for the aforementioned algorithms to show a graphic representation of the discrete movement of the robot arm.

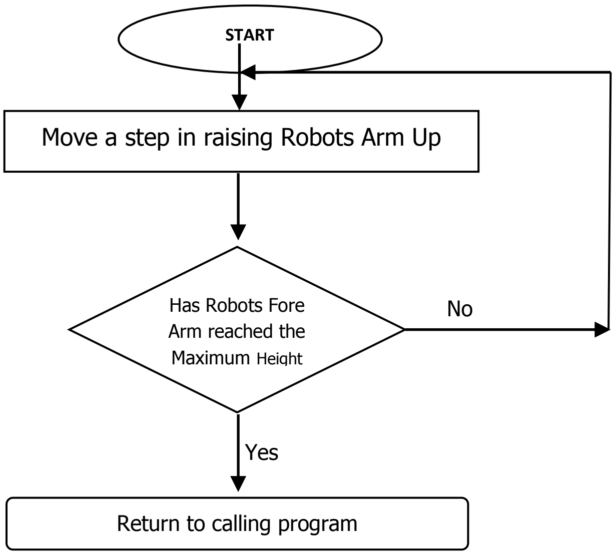
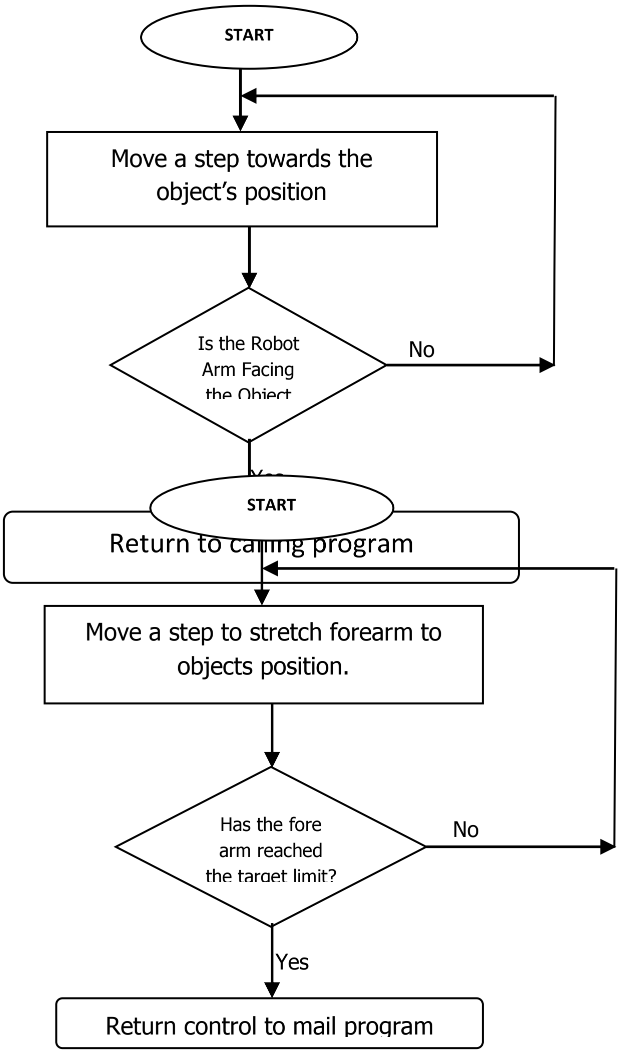

START
Move a step in raising Robots Arm Up
Has Robots Fore Arm reached the Maximum Height
No
Yes
Return to calling program

Figure 7: Flowchart for the Moving Up of the Robot’s Arm

Figure 8: Flowchart for Moving Robot Arm to Face Object

Pseudocode for Moving Up of the Robot’s Arm

```
1. Move a step in raising Robots Arm Up
2. Has Robots Fore Arm reached the Maximum Height?(yes/no)
3. if yes, Stop the Robots Fore Arm Movement
4. if No, GoTo 1.
```


START
Move a step towards the object’s position
Is the Robot Arm Facing the Object
No
Yes
Return to calling program
START
Move a step to stretch forearm to objects position.
Has the fore arm reached the target limit?
No
Yes
Return control to mail program

Pseudocode for Moving Up of the Robot’s Arm

```
1. Move a step towards the Object Position
2. Is Robot Arm Facing the Object (yes/No)?
3. if yes, Stop the Robots Arm Movement
4. if No, GoTo 1
```

Pseudocode for Moving forward the Robot’s Arm

```
1. Move a step towards the Object Position
2. Has the forearm reached the target limit (yes/No)?
3. if yes, Stop the forearm Movement
4. if No, GoTo 1.
```

Figure 9: Flowchart for Moving Robot's Forearm to Stretch to the Objects Position

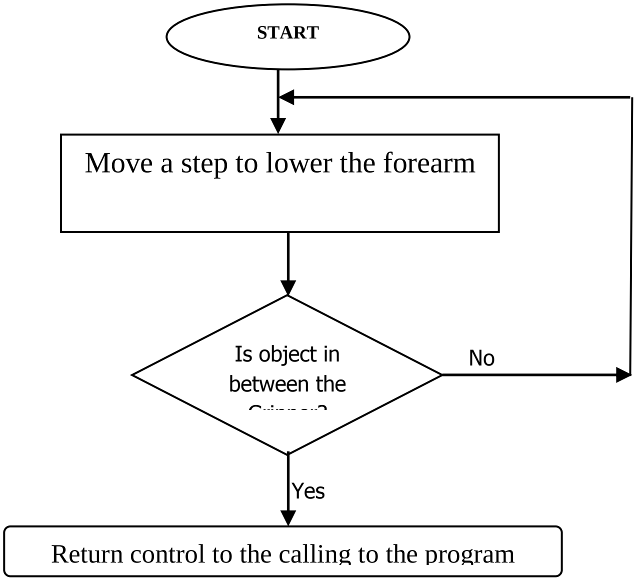

START
Move a step to lower the forearm
Is object in between the
No
Yes
Return control to the calling to the program

Figure 10: Flowchart that Lowers the Forearm

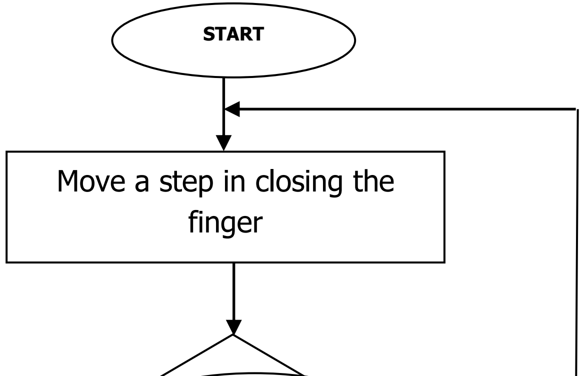

START
Move a step in closing the finger
No
gripped?
Return control to calling program

Figure 11: Flowchart that Closes the Finger of the Gripper

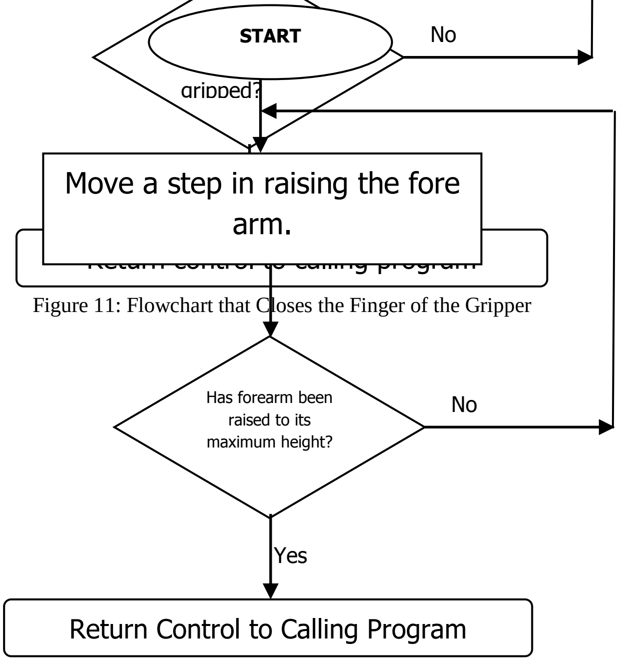

START
Move a step in raising the fore arm.
Has forearm been raised to its maximum height?
No
Yes
Return Control to Calling Program

Figure 12: Flowchart that raises the Robot's Forearm.

Pseudocode for Lowering the Robot's Forearm

```
1. Move a step towards the Object Position
2. Has the forearm reached the target limit (yes/No)?
3. if yes, Stop the forearm Movement
4. if No, GoTo 1.
```

Pseudocode for Closing the Robot's Finger

```
1. Move a step in closing the finger.
2. Has the object been gripped (yes/No)?
3. if yes, Stop closing of the finger.
4. If No, GoTo 1.
```

Pseudocode for Raising the Robot's Forearm

```
1. Move a step in closing the finger.
2. Has the object been gripped (yes/No)?
3. if yes, Stop closing of the finger.
4. If No, GoTo 1.
```

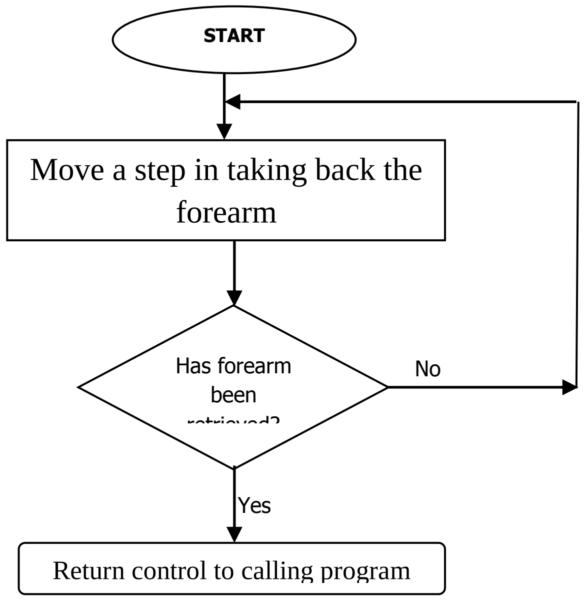

START
Move a step in taking back the forearm
Has forearm been retrieved?
No
Yes
Return control to calling program

Figure 13: Flowchart that takes back the Forearm

Pseudocode for taking back the Robot's Forearm

```
1. Move a step in taking back the forearm.
2. Has the object been taken back completely (yes/No)?
3. if yes, Stop retrieving the forearm.
4. If No, GoTo 1.
```

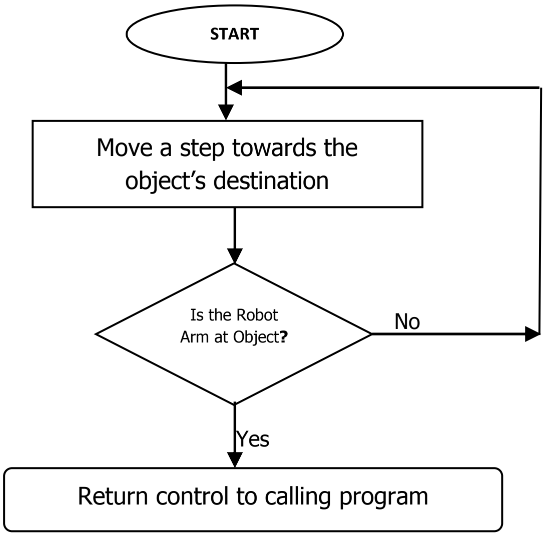

START
Move a step towards the object's destination
Is the Robot Arm at Object?
No
Yes
Return control to calling program

Figure 14: Flowchart for Moving Robot Arm to
tion

Pseudocode for Moving Up of the Robot's Arm

```
1. Move a step towards the Object destination.
2. Is Robot Arm Facing the  Object's position (yes/No)?
3. if yes, Stop the Robots Arm Movement
4. if No, GoTo 1.
```

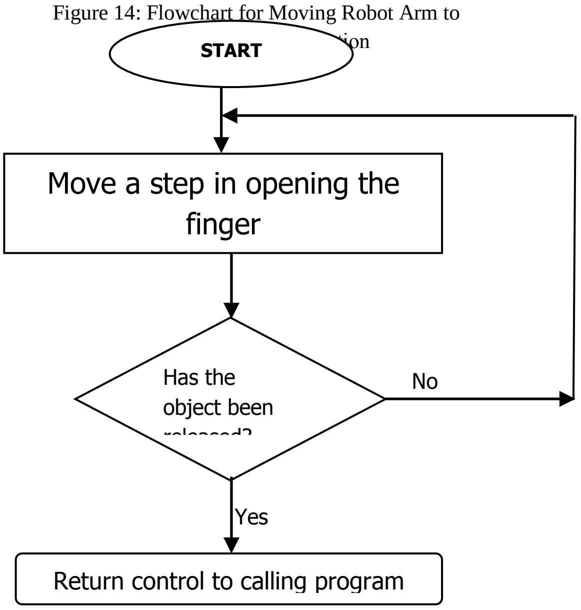

START
Move a step in opening the finger
Has the object been released?
No
Yes
Return control to calling program

Figure 15: Flowchart that opens the Gripper

Pseudocode for Open the Robot's Finger

```
1. Move a step in closing the finger.
2. Has the object been gripped (yes/No)?
3. if yes, Stop closing of the finger.
4. If No, GoTo 1.
```

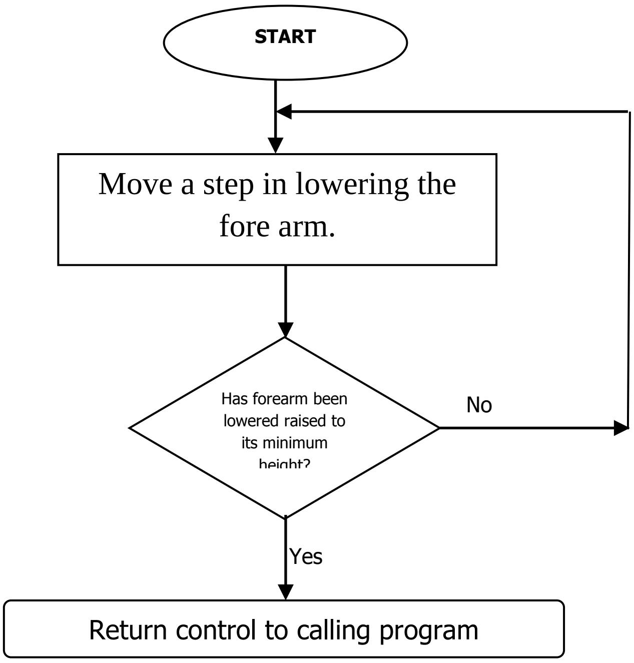

START
Move a step in lowering the fore arm.
Has forearm been lowered raised to its minimum height?
No
Yes
Return control to calling program

Figure 16: Flowchart that Lowers the Forearm after operation.

Pseudocode for lowering the Robot's Forearm

```
1. Move a step in lowering the forearm.
2. Has the object been lowered to the minimum height (yes/No)?
3. if yes, Stop lowering of fore arm.
4. If No, GoTo 1.
```

It is the duty of the control program to call the individual sub-program represented by flowcharts as at when they are need. Figure 17 shows the flowchart for the control program that calls the sub programs.

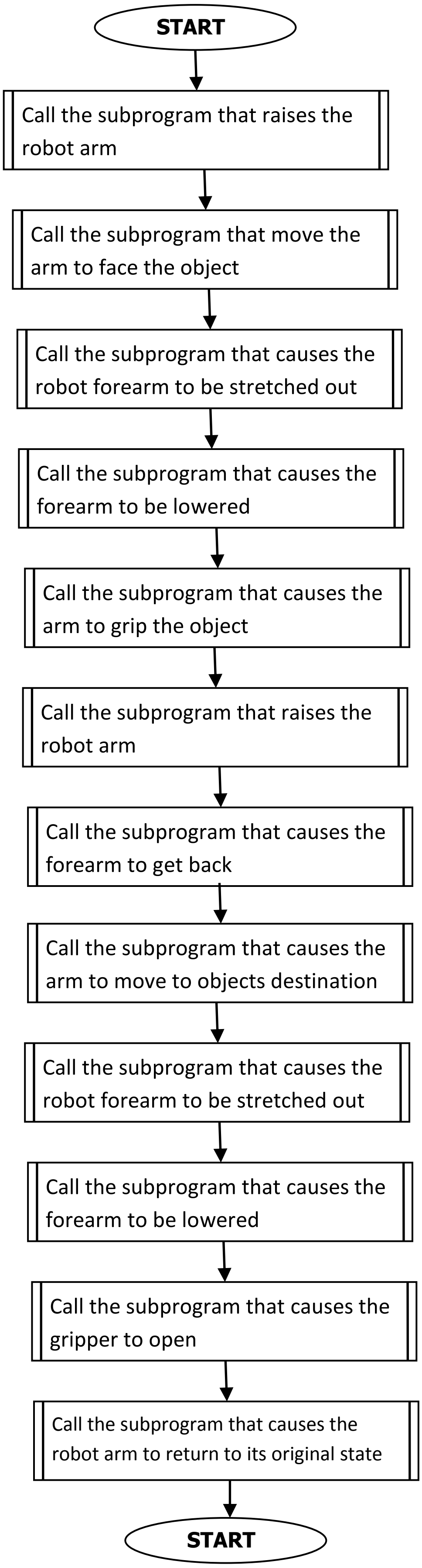

START
Call the subprogram that raises the robot arm
Call the subprogram that move the arm to face the object
Call the subprogram that causes the robot forearm to be stretched out
Call the subprogram that causes the forearm to be lowered
Call the subprogram that causes the arm to grip the object
Call the subprogram that raises the robot arm
Call the subprogram that causes the forearm to get back
Call the subprogram that causes the arm to move to objects destination
Call the subprogram that causes the robot forearm to be stretched out
Call the subprogram that causes the forearm to be lowered
Call the subprogram that causes the gripper to open
Call the subprogram that causes the robot arm to return to its original state
START

Figure 17: Flowchart for the Control Program that calls the Sub-Programs

V. SYSTEM TESTING

System testing involves exercising the system and its operation to ensure that it functions as expected. The testing is carried out to confirm the expected results and features of the system. The system, Robot Arm Control System was tested for accuracy. The system comprises of two parts, the hardware control system and the control software system.

The entire system was tested as shown in table 2

Table 2: System's Component Testing

S/N	TEST PLAN	EXPECTED RESULT	ACTUAL RESULT
1	Key recognition	Okay	Okay
2	Subroutines for **To and Fro movement Stepper motor 1**	Stepper motor 1 moves to expected position when system is initiated using the keypad.	Few millimetres from expected position due to frictional force in the gear system. This was corrected by oiling the gear system.
3	Subroutines for **To and Fro movement Stepper motor 2**	Stepper motor 2 moves to expected position when system is initiated using the keypad.	Few millimetres from expected position due to frictional force in the gear system. This was corrected by oiling the gear system.
4	Subroutines **for To and Fro movement Stepper motor 3**	Stepper motor 3 moves to expected position when system is initiated using the keypad.	Few millimetres from expected position due to frictional force in the gear system. This was corrected by oiling the gear system.
5	**Subroutines for To and Fro movement Stepper motor 4**	Stepper motor 4 moves to expected position when system is initiated using the keypad.	Few millimetres from expected position due to frictional force in the gear system. This was corrected by oiling the gear system.
6	Response on load	Reliable	Reliable

VI. CONCLUSION

VII. In this paper, a robot arm control system is proposed which will aid the automation of manual process in the industry.

REFERECES

[1] Design and Implementation of Robotic Control for Industrial Applications by Desmond Jeffery Will, January 2004

[2] RobotShop Distribution Inc., 2008, [www.robotshop.ca]

[3] Mohamed Naufal Bin Omar , April 2007

[4] Robert S. Laforge , April 2011

VIII. INDEX

PRESENT STATE								NEXT STATE				
S/N	Address(Hex)	F/H	CW/CCW	A	B	C	D	A1	B1	C1	D1	Address(Hex)
0	09	0	0	1	0	0	1	0	0	0	1	01
1	08	0	0	1	0	0	0	1	0	0	1	09
2	0A	0	0	1	0	1	0	1	0	0	0	08
3	02	0	0	0	0	1	0	1	0	1	0	0A
4	0E	0	0	1	1	1	0	0	0	1	0	02
5	04	0	0	0	1	0	0	1	1	1	0	0E
6	05	0	0	0	1	0	1	0	1	0	0	04
7	01	0	0	0	0	0	1	0	1	0	1	05
8	19	0	1	1	0	0	1	1	0	0	0	08
9	18	0	1	1	0	0	0	1	0	1	0	0A
10	1A	0	1	1	0	1	0	0	0	1	0	02
11	12	0	1	0	0	1	0	1	1	1	0	0E
12	1E	0	1	1	1	1	0	0	1	0	0	04
13	14	0	1	0	1	0	0	0	1	0	1	05
14	15	0	1	0	1	0	1	0	0	0	1	01
15	11	0	1	0	0	0	1	1	0	0	1	09
16	29	1	0	1	0	0	1	0	1	0	1	05
17	28	1	0	1	0	0	0	0	0	0	1	01
18	2A	1	0	1	0	1	0	1	0	0	1	09
19	22	1	0	0	0	1	0	1	0	0	0	08
20	2E	1	0	1	1	1	0	1	0	1	0	0A
21	24	1	0	0	1	0	0	0	0	1	0	02
22	25	1	0	0	1	0	1	1	1	1	0	0E
23	21	1	0	0	0	0	1	0	1	0	0	04
24	39	1	1	1	0	0	1	1	0	1	0	0A
25	38	1	1	1	0	0	0	0	0	1	0	02
26	3A	1	1	1	0	1	0	1	1	1	0	0E
27	32	1	1	0	0	1	0	0	1	0	0	04
28	3E	1	1	1	1	1	0	0	1	0	1	01
29	34	1	1	0	1	0	0	0	0	0	1	09
30	35	1	1	0	1	0	1	1	0	0	1	09
31	31	1	1	0	0	0	1	1	0	0	0	08

Table 3: Fully Energized Control Bit Pattern for a Four Phase Stepper Motor