Birzeit University
Faculty of Engineering & Techonology
Department of Electrical & Computer Engineering
ENEE4104

"Robot Manipulator Project"

Student :Mohamad Bornat

Instructor:Dr.Jamal Siam

Date: 28-4-2017

Abstract

This project aims to design, implement and control a three degree of freedom (DoF) robotic manipulator using stepper motors. The control of robotic manipulator is achieved by a PIC 16F877A microcontroller. The purpose of the microcontroller is to generate pulse width modulation (PWM) signals and apply it to the stepper motors to achieve the desired rotation. The main advantage of controlling the stepper motors with PWM signals is that they can be programmed to have an initial position and to rotate with an exact degree with respect to the requirements. In this study, three stepper motors are employed to realize the robotic manipulator.

Contents

Acronyms and Abbreviations

DC	Direct Current
AC	Alternating Current
PWM	Pulse Width Modulation
DoF	Degree of Freedom

List of Figures

Chapter 1

Introduction

The importance of robots increases yearly as the human needs increase and the computer programs develop and get more complicated and there are situations where a robot is a replacement for human because the human does not have the capability to work under the specific conditions, or to ease the actions done by the human or when the human is handicapped.

Generally robots are designed, built and controlled via a computer or a controlling device which uses a specific program or algorithm and there are several types of robots which are :Linear Robots ,Cylindrical Robots ,Parallel Robots Spherical Robots, SCARA Robots and Articulated robots. In our project we are going to use the last type.

Articulated robots (also known as revolute robots) have three fixed axis connected to two revolute base

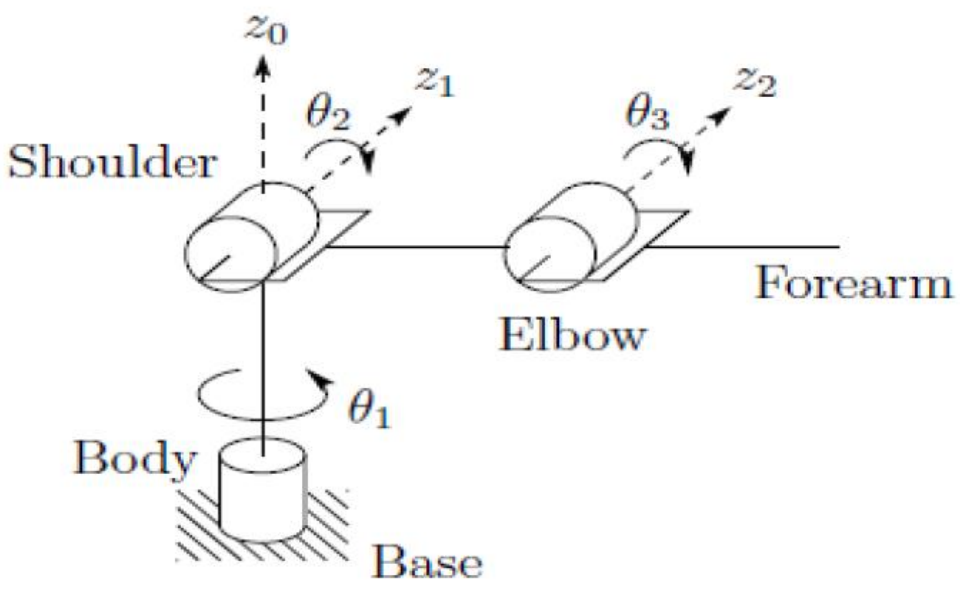

Fig. 1.1: Schematic Diagram of the articulated robot

Each joint of the robot defines the relative motion of the other two object it links which determines the subset of the whole configuration space. Each configuration subset is a different position for each link.

The articulated robot has many advantaes such as Superb structural flexibility, compatible with other robots operating in common workspace and high rotation speed .On the other hand, it has many disadvantages such as low accuracy and resolution because of rotary joints and positional errors, counter balancing difficulties due to the large and variable torque, high chance of collision and dynamic instability due to higher moment of inertia and gravity.

Chapter 2

Building the Robot Manipulator

2.1 Mathematical Model

The angles were calculated using an ainternet website using the following rules :

- Sine rule—the ratio of the length of a side to the sine of its opposite angle is constant

  $x/\sin(b) = y/\sin(a) = z/\sin(c)$

- The angle of a triangle can be calculated from its sides

  $a = \arccos((x^2 + z^2 - y^2)/2xz)$
  $b = \arccos((y^2 + z^2 - x^2)/2yz)$
  $c = \arccos((x^2 + y^2 - z^2)/2xy$

2.2 Hardware Implementation

In this project, PIC 16F877A was chosen to be the main processor of the system, because it has the required analogue and digital pins, its good range interfaces and ability to generate PWM signals, it can generate PWM signals on ten pins. [1]

The other parts of the systems were chosen depending on the required tasks in each part of the project.

2.2.1 PIC 16F877A Microcontroller

PIC 16F877A microcontroller is a well-known microcontroller for complex tasks, it has has 40 pins, and also has specifications such as PWM generator, 3 timers, analog capture and comparator circuit, universal synchronous receiver transmitter (USART), internal and external interrupt capabilities.

2.2.2 Stepper Motors Principles

The stepper motors which was used in this project operate with DC and are suitable for PWM control. A typical servo motor's shaft is limited to rotate from 0 to 180° but it is possible to modify the motor for a continuous rotation, but the stepper motor is essentially a servo motor that uses a different method for motorisation, where the servo motor uses continues rotation DC motor, the stepper motor utilise multiple toothed electromagnets arranged around a central gear to choose the position.

2.2.3 Implementation

The electronic circuit of the robot manipulator is simulated and tested on Proteus simulation software as in the following figure:

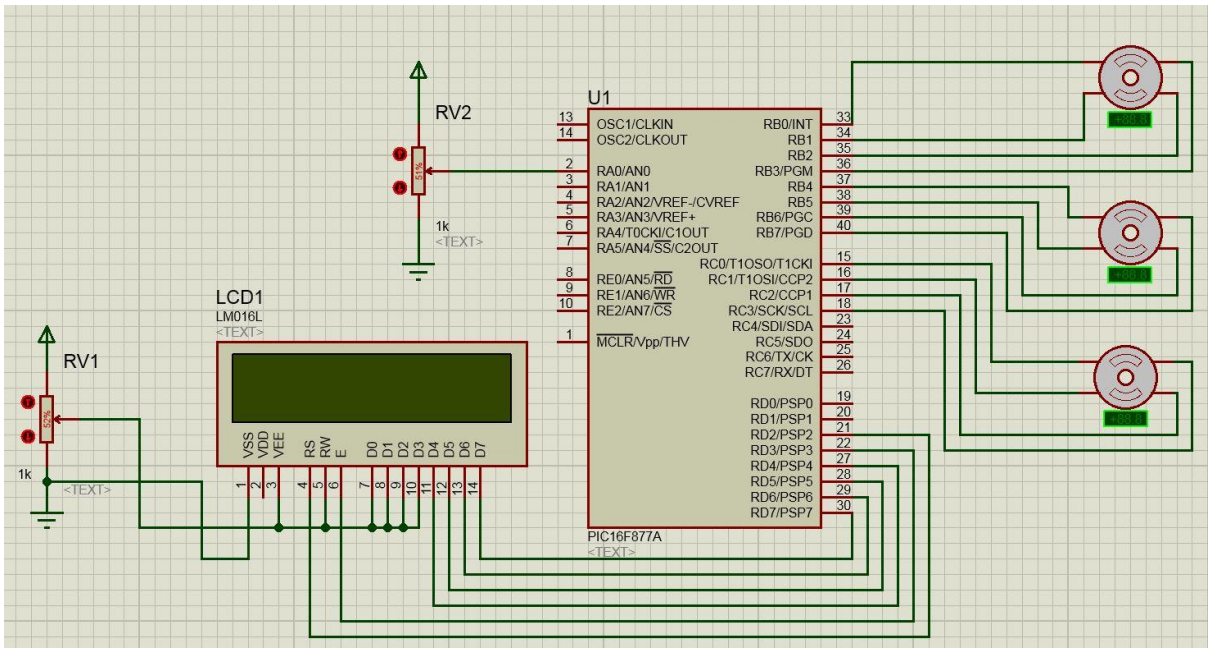

Fig. 2.1: Schematic Diagram of the controlled robot manipulator circuit

The system starts with the initialization of the robot, a potentiometer was connected to the ADC input with voltage reference Vcc. And it was read at the start of the motion action to set the relation between the shoulder and elbow angles as follow:
If input voltage < 0.2 Vcc, then θ2 > θ1
If input voltage < 0.8 Vcc, then θ1 > θ2

Chapter 3

Simulation Results & Discussion

3.1.1 Part 1

to acquire a video to test the material density on a line with slope 45o on the (x, y) horizontal plane, from the maximum possible distance to half of the maximum distance, a MikroC code (A.1) was written then generated as a HEX file and downloaded on the pic . The following results were obtained:

When RV1 Pot = 11% and RV2 Pot = 19% then :

We can notice that the first two motors returned to zero degree while the third gone to 180. X=0,Y=0,Z=45.

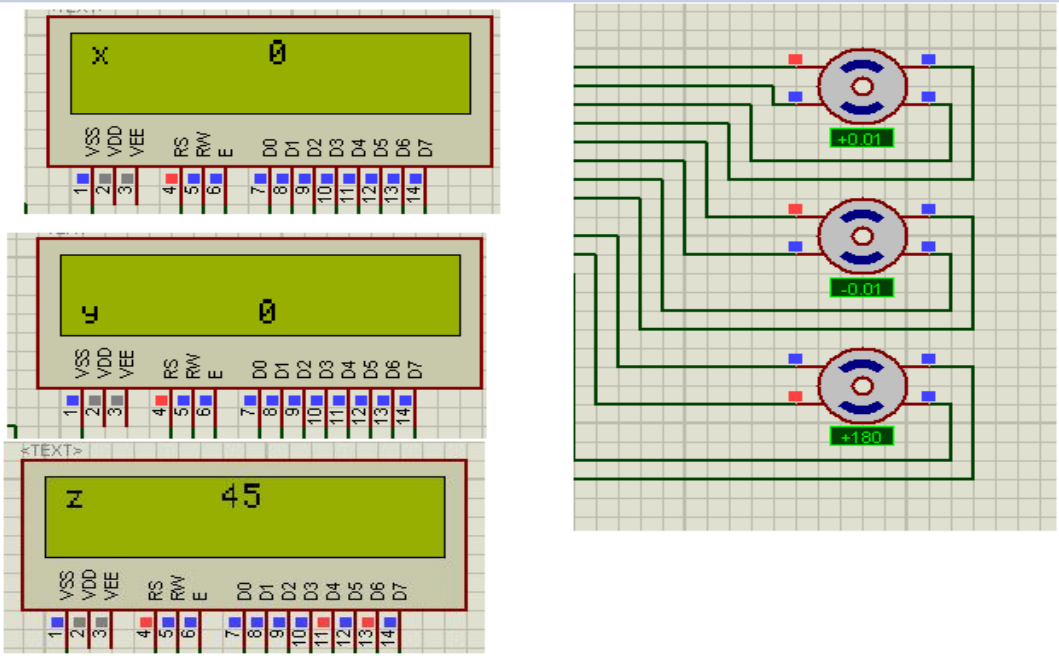

Fig. 3.1: first position and degrees to 45 slope

Then the first motor goeas to 360 and x =32,y = 32, z=0

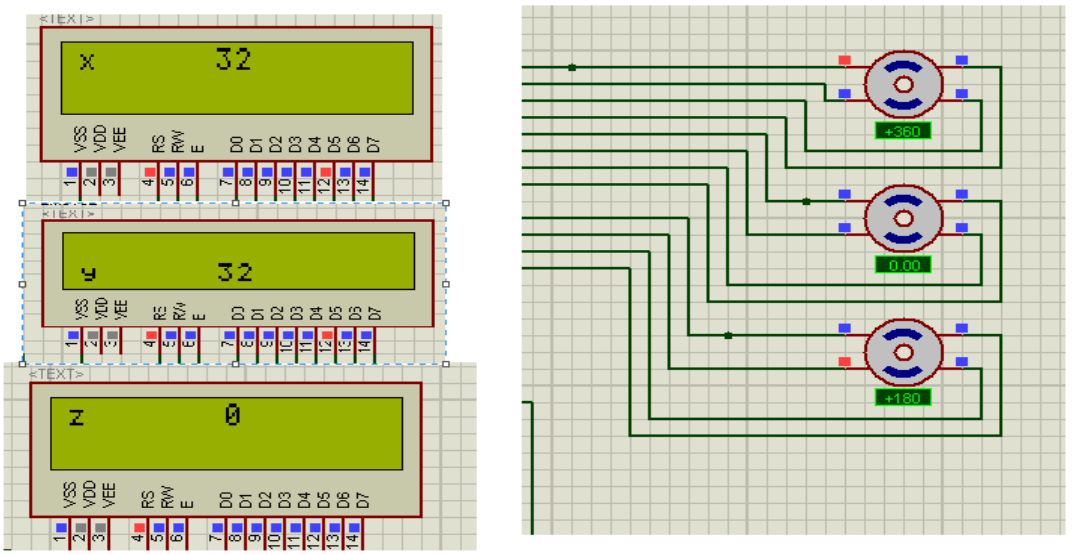

Fig. 3.2: second position and degrees to 45 slope

Then it goes back again to its initial position.

3.1.2 Part 2

to acquire a video to test the material density on a circle on the (x, y) plane with radius 30 cm. , a MikroC code (A.2) was written then generated as a HEX file and downloaded on the pic . The following results were obtained:

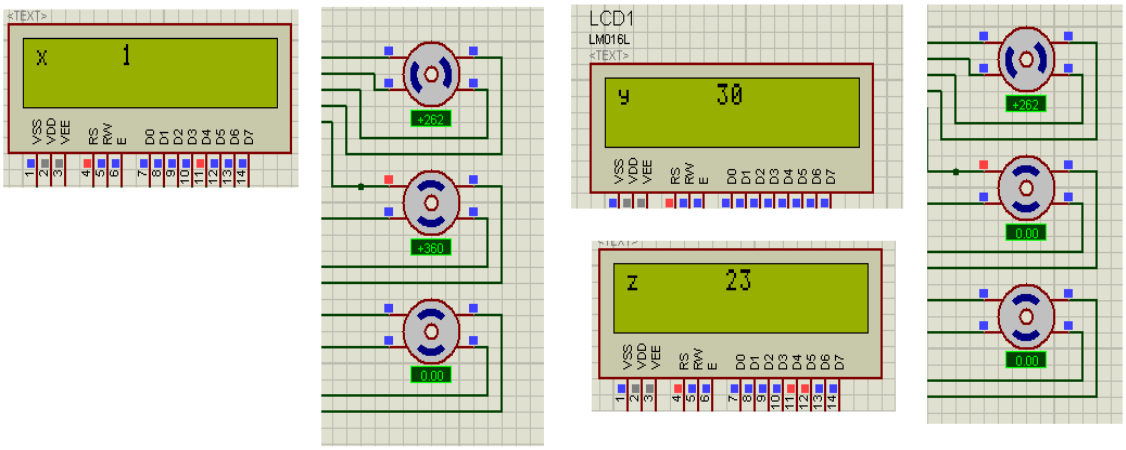

Fig. 3.3: first position and degrees with radius 30 cm

Then it goes as follows :

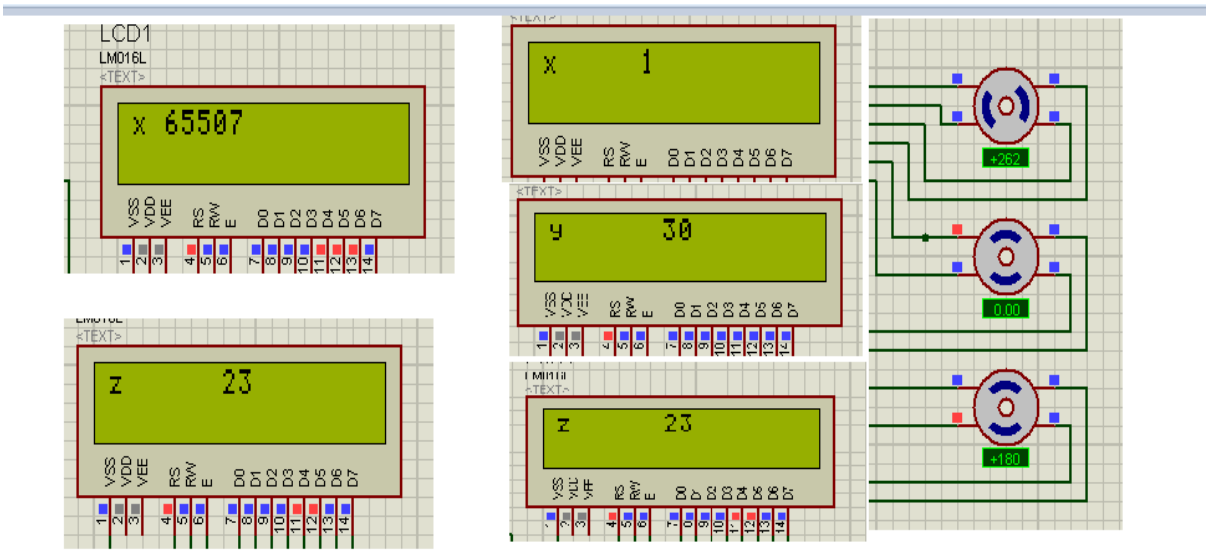

Fig. 3.4: second position and degrees with radius 30 cm

3.1.3 Part 3

To acquire a video of 40 seconds a MikroC code (A.3) was written then generated as a HEX file and downloaded on the pic. The following results were obtained:

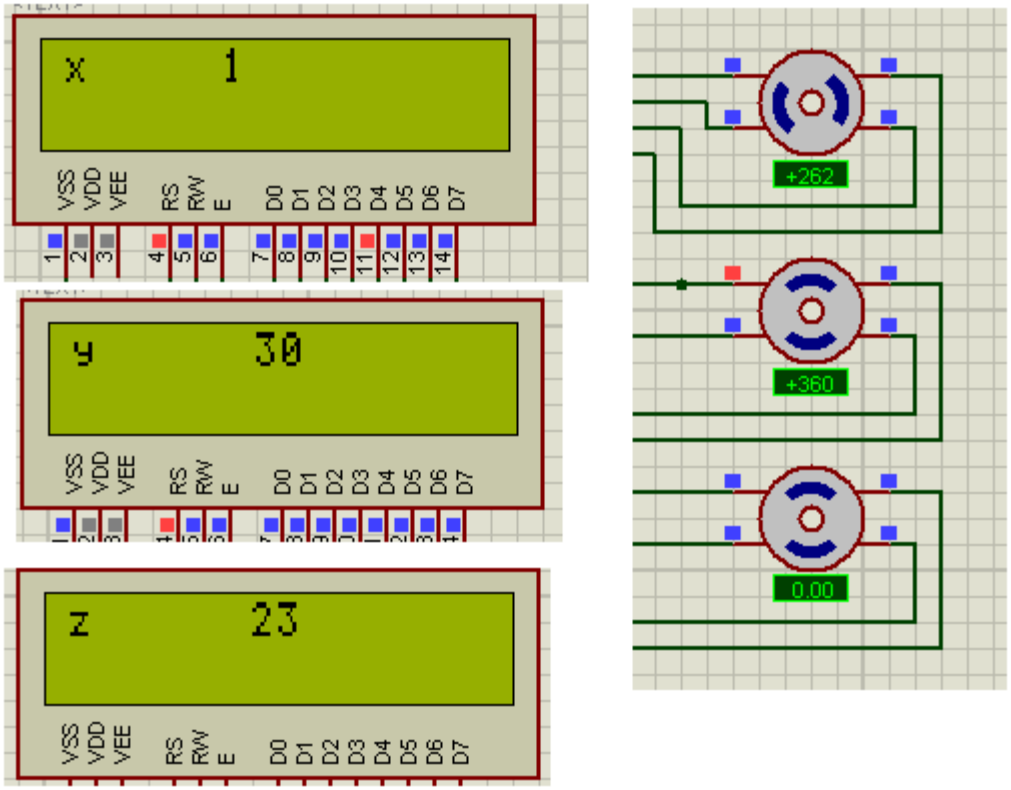

Fig. 3.5: first position and degrees for 40 seconds recording

Conclusion

In this Project, a manipulator robot was built using a microcontroller and stepper motors. The building procedure consists of hardware design and implementation, software design and implementation and microcontroller programming.

The motion of the robot was controlled by PWM signals which were generated using the microcontroller.

As the future work of the developed robot, a voice-controlled robot can be considered where the user can simply control the robot by giving voice commands such as giving directions or actions.

References

[1]"Remote Control of Robot Arm with five DOF", MIPRO Electronics

Appendices

(A.1)

```
#include <stdio.h>
#include <stdlib.h>

void Print(int x_axis,int y_axis,int z_axis);
void initialize(){
    LCD_INITIALIZE();
    LCD_CMD(_LCD_CLEAR);
    LCD_CMD(_LCD_CURSOR_OFF);
    TRISB = 0;
 }

/* connections of LCD module  */

sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
```

```
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;



 void main() {

int i;
unsigned int x_axis=0,y_axis=0,z_axis=45;
char x_axis_txt[4],y_axis_txt[4],z_axis_txt[4] ;

initialize();

TRISD = 0x00; // to set all pins of port A as output
TRISB = 0x00; // to set all pins of port B as output
TRISA = 0xFF; // PORTA is input
TRISC = 0;    // PORTC is output
PORTC= 0;
PORTB= 0;



/* for parking */

// Delay in ms

for (i=0;i<3;i++) {
 PORTB = 0x48;
 delay_ms(300);
 PORTB = 0x22;
 delay_ms(300);
```

```
PORTB = 0x84;
delay_ms(300);
PORTB = 0x11;
delay_ms(300);
}

/* theta 1 from  1 to 45 degree */

PORTC = 0x88;
delay_ms(300);
PORTC = 0x22;
delay_ms(300);
PORTC = 0x44;
delay_ms(300);
PORTC = 0x11;
delay_ms(300);
PORTC = 0x88;
delay_ms(300);
PORTC = 0x22;
delay_ms(240);

 for (i=0;i<3;i++) {

  PORTB = 0x84;
  delay_ms(300);
  PORTB = 0x22;
  delay_ms(300);
  PORTB = 0x48;
 delay_ms(300);
  PORTB = 0x11;
  delay_ms(300);
```

```
 }

/* theta 2 from 0 to 90 */

 Print( x_axis,y_axis,z_axis);
     for (i=0;i<3;i++) {
       PORTB = 0x08;
       delay_ms(300);
       PORTB = 0x02;
       delay_ms(300);
       PORTB = 0x04;
       delay_ms(300);
       PORTB = 0x01;
       delay_ms(300);
    }
     z_axis=z_axis-45;
     x_axis=x_axis+32;
     y_axis=y_axis+32;

     Print( x_axis,y_axis,z_axis);
}

  void Print(int x_axis ,int y_axis, int z_axis){

    LCD_OUT(1,1,"x");
    Word_to_String(x_axis,x_axis_txt);
    LCD_OUT(1,5,x_axis_txt);
    delay_ms(900) ;
    LCD_CMD(_LCD_CLEAR);
    LCD_CMD(_LCD_CLEAR);
```

```
    LCD_CMD(_LCD_CURSOR_OFF);

    initialize();
    LCD_OUT(1,1,"y");
    Word_to_String(y_axis,y_axis_txt);
    LCD_OUT(1,5,y_axis_txt);
    delay_ms(900);
    LCD_CMD(_LCD_CLEAR);
    LCD_CMD(_LCD_CLEAR);
    LCD_CMD(_LCD_CURSOR_OFF);

    LCD_OUT(1,1,"z");
    Word_to_String(z_axis,z_axis_txt);
    LCD_OUT(1,5,z_axis_txt);
    Delay(900);
    LCD_CMD(_LCD_CLEAR);
    LCD_CMD(_LCD_CLEAR);
    LCD_CMD(_LCD_CURSOR_OFF);
  }
```

(A.2)

```
#include <stdio.h>
#include <stdlib.h>

void Print(int x_axis,int y_axis,int z_axis);
void initialize(){
    LCD_initialize();
    LCD_CMD(LCD_Clear);
    LCD_CMD(LCD_Cursor_off);
    TRISB = 0;
}
```

```
/* connections OF LCD module  */

sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;



void main() {

   int i,j=0;
   unsigned int x_axis=0,y_axis=30,z_axis=23;
   char x_axis_txt[4],y_axis_txt[4],z_axis_txt[4] ;

   initialize();
   TRISD = 0x00; // to set all pins of port A as output
   TRISB = 0x00; // to set all pins of port B as output
   TRISA = 0xFF; // PORTA is input
   TRISC = 0;    // PORTC is output
   PORTC=0;
   PORTB=0;
```

```
/* for parking */
// Delay in ms

    for (i=0;i<3;i++) {
      PORTB = 0x48;
      delay_ms(240);
      PORTB = 0x22;
      Delay(300);
      PORTB = 0x84;
      Delay(300);
      PORTB = 0x11;
      Delay(300);
 }

    for (i=0;i<3;i++) {
      PORTB = 0x04;
      Delay(300);
      PORTB = 0x02;
      Delay(300);
      PORTB = 0x08;
      Delay(300);
      PORTB = 0x01;
      Delay(300);
 }

      PORTB = 0x08;
      Delay(300);
      PORTB = 0x02;
      Delay(240);
      PORTB = 0x04;
```

```
    Delay(300);

   for (i=0;i<6;i++) {
    PORTB = 0x80;
    Delay(300);
    PORTB = 0x20;
    Delay(300);
    PORTB = 0x40;
    Delay(300);
    PORTB = 0x10;
    Delay(300);
}
    x_axis=0;

  Print(x_axis,y_axis,z_axis);

  for (i=0;i<188;i++)
   {
     j=j+1;

     if (j==47)
      {

     x_axis=0;
     y_axis=30;

     Print( x_axis,y_axis,z_axis);

     }

    PORTC = 0x88;
```

```
 Delay(300);
 PORTC = 0x22;
 Delay(300);
 PORTC = 0x44;
 Delay(300);
 PORTC = 0x11;
 Delay(300);
 PORTC = 0x88;
 Delay(300);
 PORTC = 0x22;
 Delay(300);

if (j==47)
{

x_axis=-30;
y_axis=0;

Print( x_axis,y_axis,z_axis);

 }

if (j==94)

 {

 y_axis=-30;

 Print( x_axis,y_axis,z_axis);

 }
```

```
      if (j==141)
       {

       y_axis=0;
       x_axis=30;

       Print( x_axis,y_axis,z_axis);

       }

      if (j==188)

       {

       y_axis=30;
       x_axis=0;

       Print( x_axis,y_axis,z_axis);

       }

     }




}

  void Print(int x_axis ,int y_axis, int z_axis){
```

```
LCD_OUT(1,1,"x");
Word_to_String(x_axis,x_axis_txt);
LCD_OUT(1,5,x_axis_txt);
Delay(1000);
LCD_CMD(LCD_Clear);
LCD_CMD(LCD_Clear);
LCD_CMD(LCD_Cursor_off);

initialize();
LCD_OUT(1,1,"y");
Word_to_String(y_axis,y_axis_txt);
LCD_OUT(1,5,y_axis_txt);
Delay(1000);
LCD_CMD(LCD_Clear);
LCD_CMD(LCD_Clear);
LCD_CMD(LCD_Cursor_off);

LCD_OUT(1,1,"z");
Word_to_String(z_axis,z_axis_txt);
LCD_OUT(1,5,z_axis_txt);
Delay(1000);
LCD_CMD(LCD_Clear);
LCD_CMD(LCD_Clear);
LCD_CMD(LCD_Cursor_off);

}
```

(A.3)

```
#include <stdio.h>
#include <stdlib.h>

 void print(int x_axis,int y_axis,int z_axis);
 void Initialize(){
    LCD_Initialize();
    LCD_CMD(LCD_Clear);
    LCD_CMD(LCD_Cursor_Off);
    TRISB = 0;
  }

/* connections of LCD module  */

sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;


void main() {

 int i;
 unsigned int x_axis=0,y_axis=0,z_axis=45;
```

```
char x_axis_txt[4],y_axis_txt[4],z_axis_txt[4] ;

Initialize();
TRISD = 0x00; // to set all pins of port A as output
TRISB = 0x00; // to set all pins of port B as output
TRISA = 0xFF; // PORTA is input
TRISC = 0;    // PORTC is output
PORTC=0;
PORTB=0;

/* for parking */

// Delay in ms

    for (i=0;i<3;i++) {

     PORTB = 0x48;
     Delay(240);
     PORTB = 0x22;
     Delay(240);
     PORTB = 0x84;
     Delay(240);
     PORTB = 0x11;
     Delay(240);
 }

     PORTC = 0x08;
     Delay(240);
     PORTC = 0x02;
     Delay(240);
     PORTC = 0x04;
```

```
  Delay(240);
  PORTC = 0x01;
  Delay(240);

  PORTB = 0x04;
  Delay(240);
  PORTB = 0x02;
  Delay(240);
  PORTB = 0x08;
  Delay(240);
  PORTB = 0x01;
  Delay(240);
  PORTB = 0x04;
  Delay(240);
  PORTB = 0x02;
  Delay(240);
  PORTB = 0x08;
  Delay(240);

for (i=0;i<4;i++) {
  PORTB = 0x80;
  Delay(240);
  PORTB = 0x20;
  Delay(240);
  PORTB = 0x40;
  Delay(240);
  PORTB = 0x10;
  Delay(240);
 }

  PORTB = 0x80;
```

```
        Delay(240);
        PORTB = 0x20;
        Delay(240);
        PORTB = 0x40;
        Delay(240);


}
```