INDEX

Experiment no: 01

Name of the Experiment: Blinking LED using PIC microcontroll

Objectives:

(i) Designing a LED blinking circuit.
(ii) Understanding the circuit diagram of PIC microcontroller.

Theory:

LED's are small, powerful lights that are used in many different applications. It is as simple as light turning on and off. It is an electronic device, which emits light when the current passes through its terminals.

An LED is a two-terminal device. The two terminals are called as cathode and Anode.

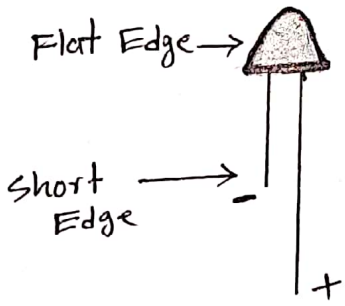


Figure-1.1: Structure of LED.

Apparatus Required:

LED, PIC16F877A, power supply, resistor, crystal oscillator, capacitor.

Circuit Diagram:

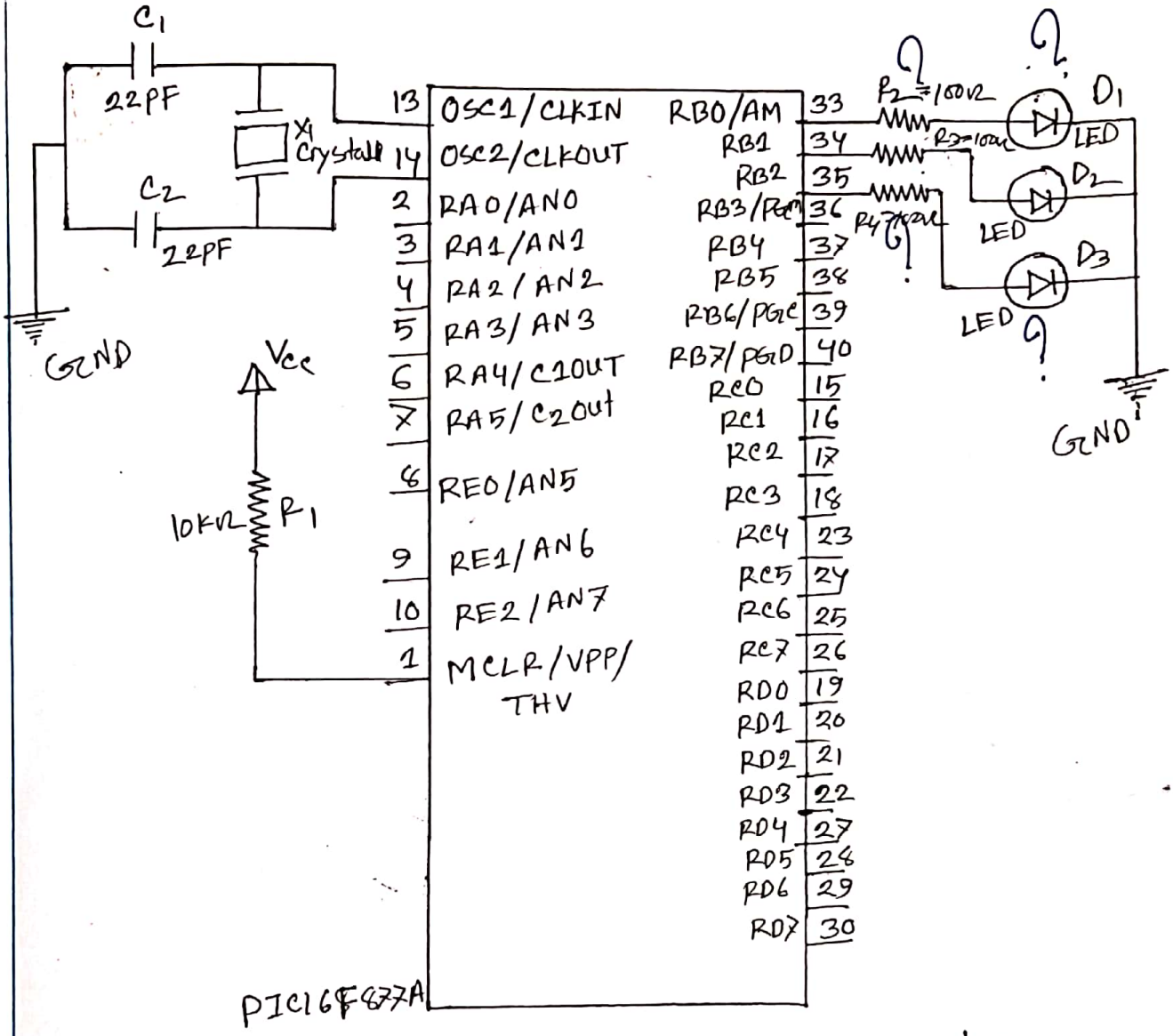
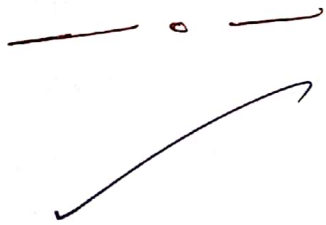


Figure-12 Blinking LED using PIC microcontroller.

Program:

```
void main()
  {
   TRISB = 0x00;    // PORTB as output
   PORTB = 0x00;    // initialize portb as off state
   while(1)
     {
      portb.f0 = 0xff;
      delay_ms(400);
      portb.f0 = 0x00;
```

```
        portb.f1 = 0xff;
        delay_ms(400);
        portb.f1 = 0x00;
        portb.f2 = 0xff;
        delay_ms(400);
        portb.f2 = 0x00;
    }
}
```



Experiment no : 02

Name of the Experiment : Write a program to count 0 to 9 in 7 segment display using PIC Microcontroller.

Objectives:

(i) Learning how to design a 7 segment display using PIC micro controller.

(ii) Understanding 7 segment displays principle.

Theory:

Seven segment displays are the output display device that provides a way to display information in the form of images or text or decimal numbers which is an alternative to the more complex dot matrix display. It is widely used in digital clocks, basic calculators, electronic meters and other electronic devices that display numerical information. It consists of seven segment of light-emiting diodes (LED's) which are assembled like numerical 8.

According to the type of application, there are two types of seven segment display.

(i) Common anode display.

(ii) Common Cathode display.

(i) Common anode display: In common anode connection of the LED segments are connected together to logic 1 in a common anode seven segment display. We use logic 0 through a current limiting resister to the cathode of a particular segment a to g.

Common anode 7 segment displays are more popular than cathode seven segment displays, because logic circuits can sink more current than they can source and it is the same as connecting LED's in reverse.

Seven segment displays are used in digital clocks.

(ii) Common cathode display: In common cathode seven segment displays all the cathode connections of LED segments are connected together to logic 0 or ground. We use logic 1 through a current limiting resistor to forward bias the individual anode terminals a to g.

Seven segment displays are used in Digital clocks, clock radios, calculator, wristwatches, speedmeters etc.

Apparatus required:

PIC16F877A, Capacitor, Resistor, Crystal oscillator, Common anode seven segment.

Circuit Diagram:

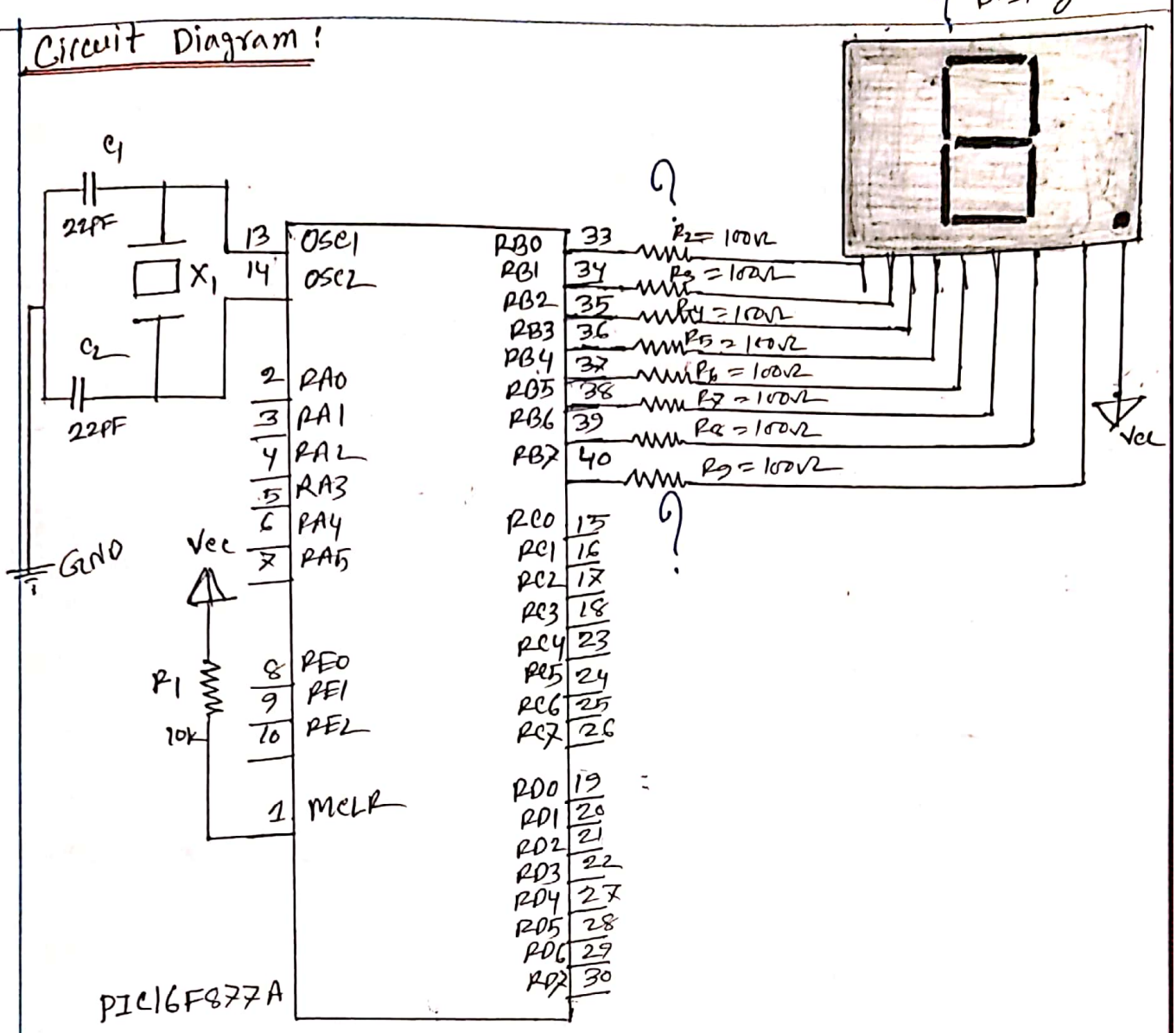


Figure 2.1: 7 segment display using PIC Microcontroller.

Mikro C Code:

```
Void main(){
  int i =0;
  char arra[] = {0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x10};
  TRISB = 0x00;
  portb = 0xff;
  while(1){
     portb = arra[i];
     delay_ms(500);
     i++;
     if (i==10)
} }    i=0;
```

Experiment no: 03

Name of the Experiment: Analog signal input in the microcontroller or display ADC value in the virtual terminal.

Objectives:

(i) To design a circuit that display ADC value in the virtual terminal.

(ii) To learn about the display of ADC value in the vitual terminal.

Theory:

The role of ADC converter is to convert analog voltage value to digital values. The A/D converter converts analog voltage to binary number. These binary numbers can be in different length 2,4,8,16 bit. The more bits the binary number has the higher the resolution of the A/D.

With two bits, we can only display 4 different options.

00	01	10	11

We can show the changes from 0 to 5 volt with 4 levels.

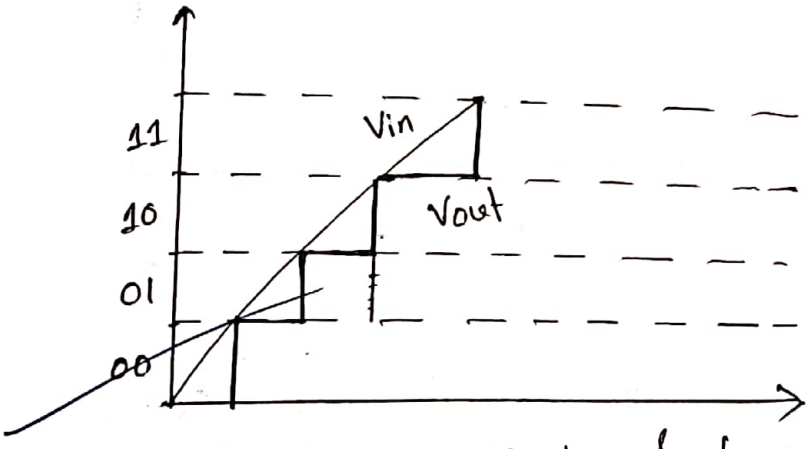


Figure-3.1: Display of four level.

We can see from figure-3.1 Vout is not close enough to the original analog input voltage values. Thus we can say that A/D with the binary number of two-bits has a low resulation and there is a large gap between the real value of the analog input voltage and the values represented by the A/D.

Now, let us consider that the voltage that supplied to the A/D converter is still varies from 0 to 5 volt, However, the A/D conVerter converts the input voltage to a binary number of three bits. With three bits we can get a different options.

000	001	010	011	100	101	110	111

We can now see the eight levels in the following illustration.

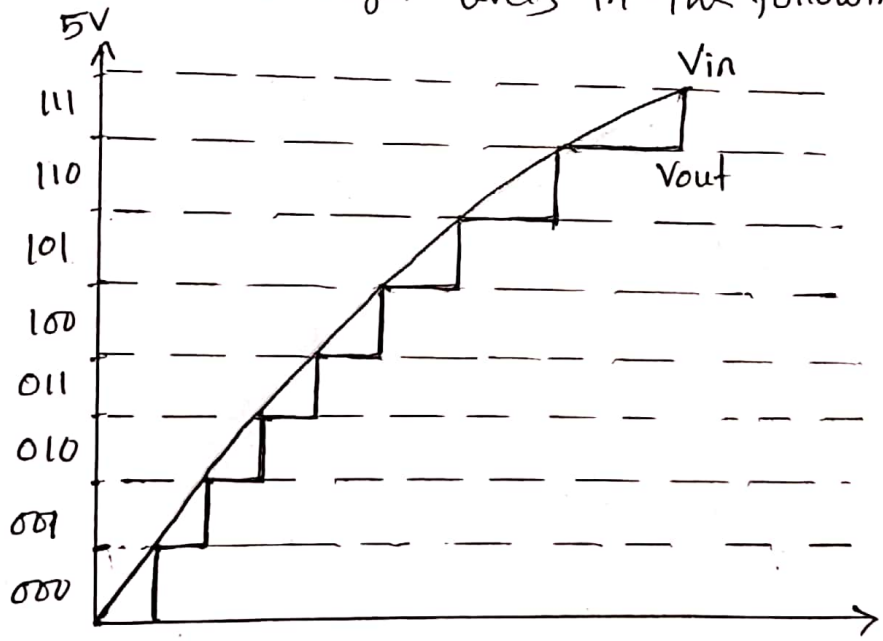


Figure-3.2: Display of eight level.

From figure 3.2, we can see that the gap between the analog signal and the digital signal is smaller compared to the previous graph.

Based on the "good" results that we received, we can say that current A/D convert has a high resolution compare to previous case.

Therefore we can say that the A/D of the micro-controller with a larger amount of bits has a higher resolution A/D takes less time than the conversion time of the high resolution A/D.

The ADCON module located within the PIC microcontroller has a resolution of ten-bit length. Therefore, the converter can devide the analog input voltage between 0V to 5V to 2^{10} levels. which are 1024 levels. We can say that the resolution of this component is very high.

We can use the triangle method to calculate the binary representation of an analog input voltage.

For example, let us calculate binary value representation on the analog input voltage of 3.65 volt.

$$\left\{ \begin{array}{l} 5V \longrightarrow 1024 \\ 3.65V \longrightarrow X \end{array} \right\} \longrightarrow X = \frac{1024 * 3.65}{5}$$

$$= 747.52$$

$$\approx 748$$

The analog input voltage of 3.65 V will be represented by decimal number 748 or by binary number 1011101100. Using similar way we can find a binary representation for any desired level of the analog input voltage.

Apparatus Required: PIC16F877A, Crystal, capacitor, Resistor,

Circuit Diagram:

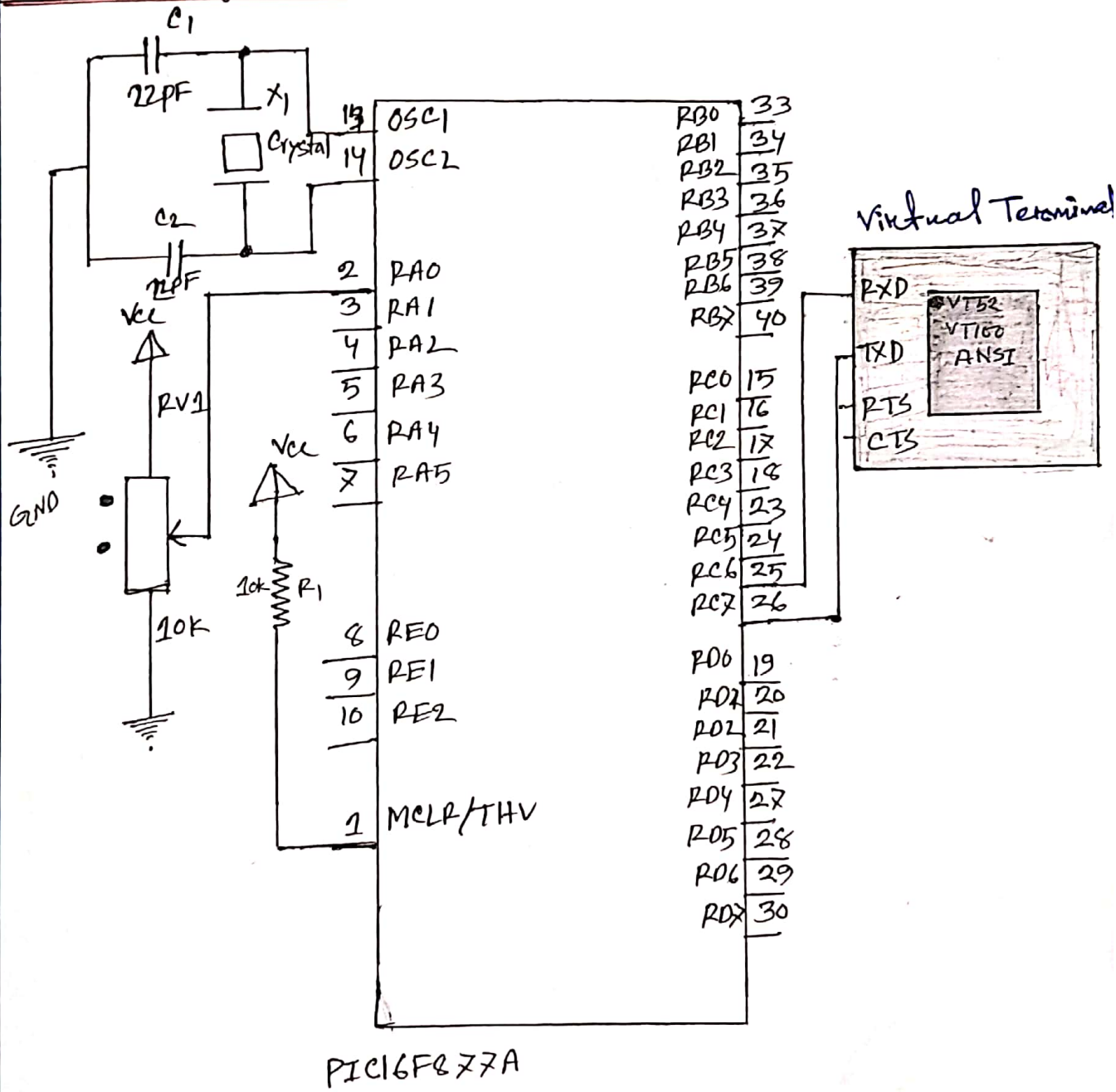


Figure3.3 Displaying ADC value in virtual terminal.

Program :

```
int valADC;  // Create a variable that will hold the ADC value
char x[4];   // create a char array
void main ()
{
  URAT1_Init (9600);   // initialize UART
  ADC_Init ();         // initialize ADC
  while (1)
  {
    valADC = ADC_Read(0);   //Read ADC value in RA0
    IntToStr (valADC,x);    //Convert into string/char array
    UART1_write_Text ("Analog value = "); //print
    UART1_write_Text (x);
    strcpy (x," ");         // Clear char array
    UART1_write(13);
    Delay_ms(1000);
  }
}
```

Experiment no: 04

Name of the Experiment: Write a program to display a digit number using 2 segment multiplexing technique.

Objectives:

(i) ~~learning how~~ to design a display of 2 digit number using seven segment multiplexing technique.

(ii) To learning and understanding about multiplexing.

Theory:

A seven segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix display.

Multiplexing technique is used for driving multiple seven segment display. The theory of 2 segment display is simple. All the similar segments of multiple LED displays are connected together and driven through a single IO pin.

Multiplexing is necessary two or more seven segment displays to a microcontroller. Software program can control these multiplexed seven segment to ON/OFF in a cyclic fashion.

This also helps to reduce power in battery operated system.

Apparatus Required:

PIC16F677A, Capacitor, Resistor, Crystal Oscylator, Button, 2 digit 7 segment.

2-digit 7 segment Display

Circuit Diagram:

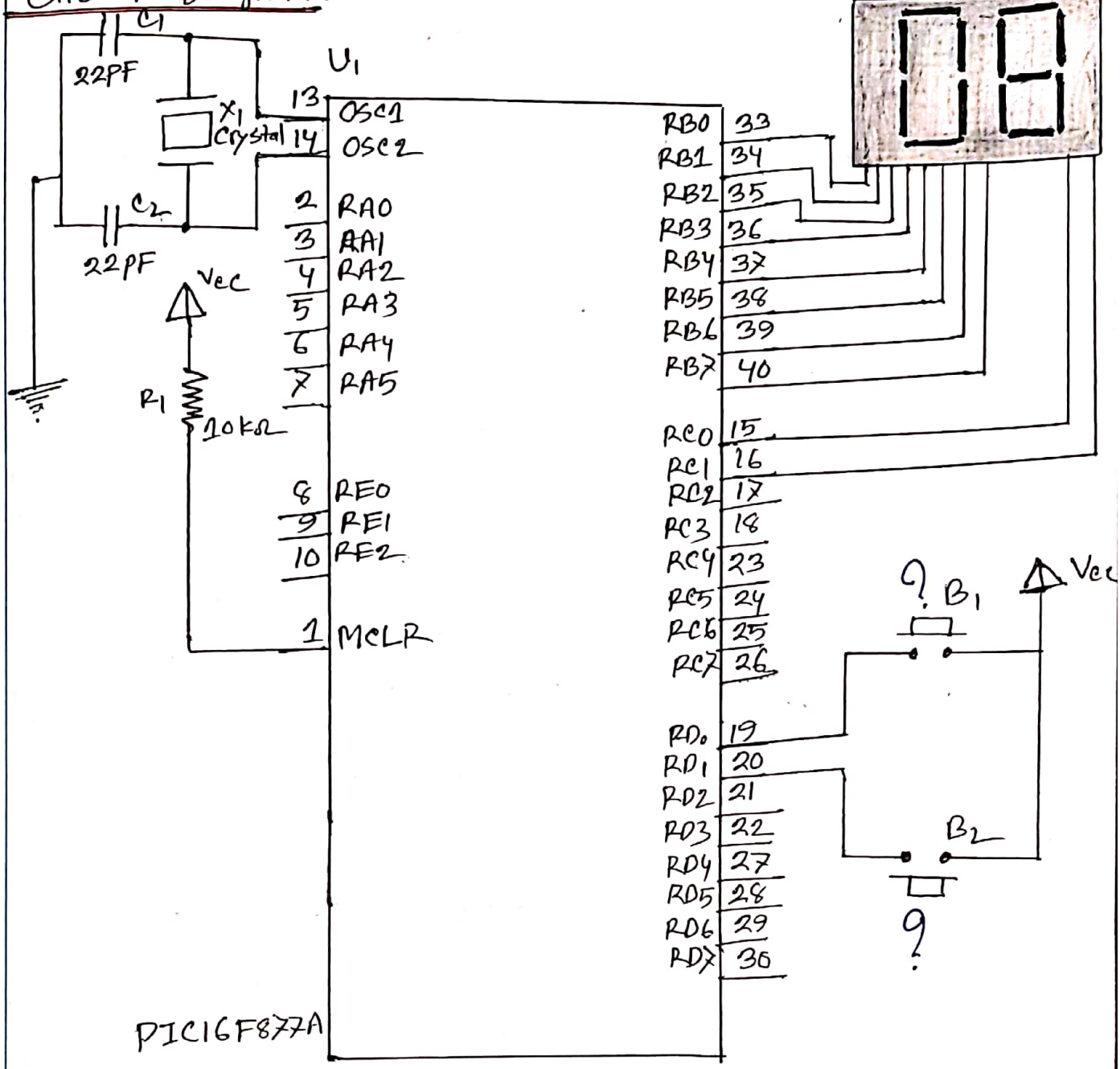


Figure4.1: Displaing two digit number using 7 segment multiplexing technique.

Program code:

```
char segment[] = { 0x3F, 0x06, 0x5B, 0x4F };
int i=0;
Void main()
{
    trisb = 0x00;
    trisc = 0x80;
```

```
portb = 0x00;
portc = 0x00;
trisd = 0xff;
portd = 0;
while(1)
{  portc.f0 = 0;
   portb = segment[i/10];
   delay_ms(10);
   portc.f0 = 1
   portc.f1 = 0;
   portb = segment[i/10];
   delay_ms(10);
   portc.f1 = 1;
   if (portd.f0 = 1)
      {  i++;
         while (portd.f0 == 1);
      }
   if (portd.f1 == 1)
      {  i--;
         while (portd.f1 == 1);
      }
      if (i< 0 || i> 99)
         i = 0;
   }
}
```

Experiment no: 05

Name of the Experiment: Write a program to interface LED display with push button using pic-microcontroller.

Objectives:

(i) To learn push button interfacing with pic microcontroller
(ii) To lean how to control a LED using a push button.
(iii) To design and understand the circuit diagram.

Theory:

The push button is a basic input device in the embedded system. It is used to control the operation of any output device using the microcontroller or control unit. It basically breaks the electrical circuit and interrupts the flow of current.

The push-button is basic mechanical ON/OFF buttons that act as control devices. It short circuit the line when it is pressed and opens when it is not pressed.

In circuit pull-up and pull-down resister use to convert infinite or zero resistance into the digital signal. On the basis of the pull up and pull down resister, we can interface the switch in two way. The value of pull-up and pull-down resistor depends on the microcontroller.

Positive logic:

In this connection, we used pull-down resistor connected to the ground. When we pressed the switch then logic asserts high and when we pressed not then disconnect the switch logic assert low.

Negative logic:

In this connection, we used pull-up resistor connects to Vcc. When we pressed the switch then logic asserts low and when we disconnect the switch logic assert high.

Apparatus Required:

PIC16F877A, Crystal oscillator, Capacitor, Resistor, LED, push button.

Circuit Diagram :

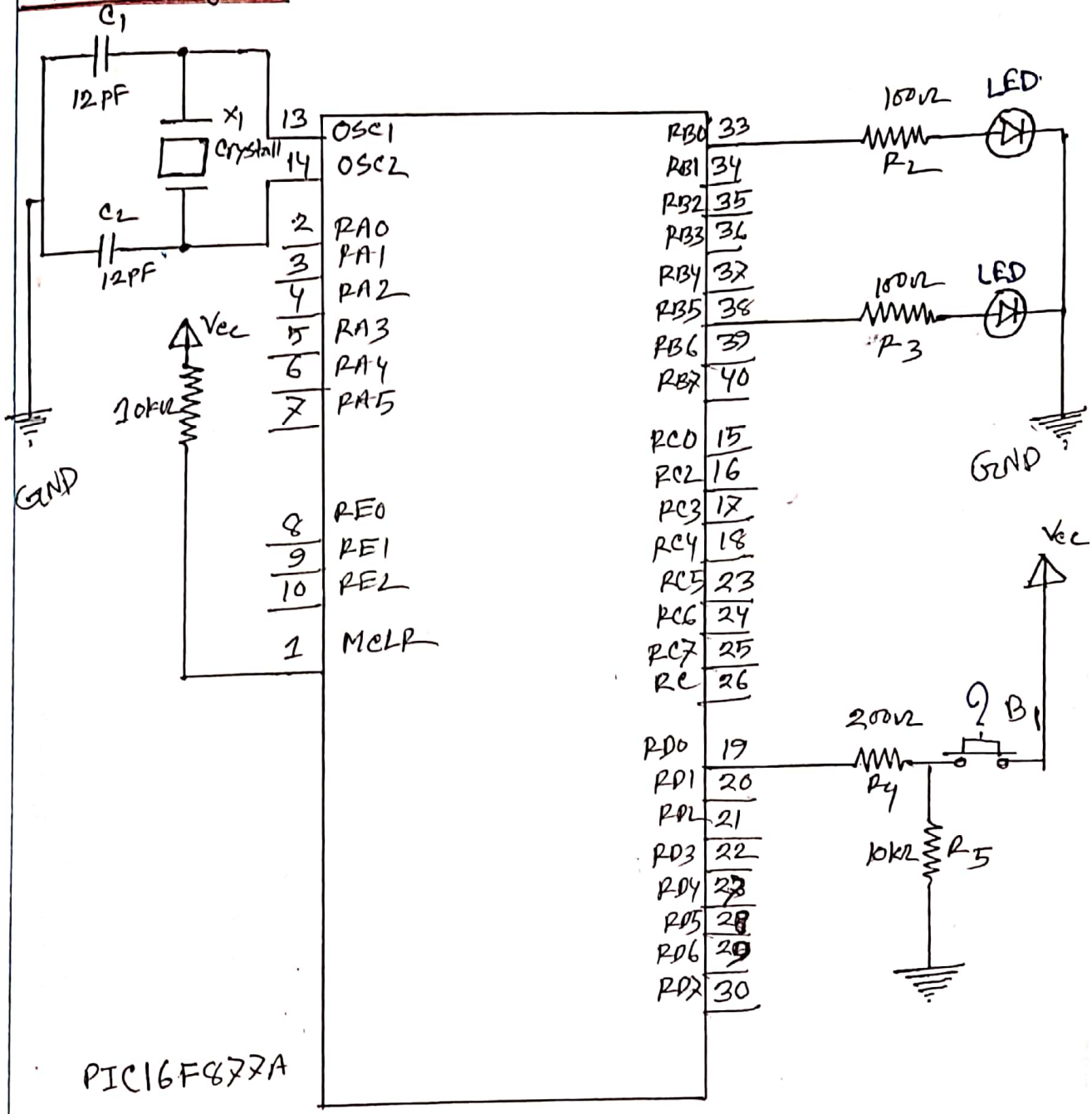


Figure5.1: Button interfacing using pic-microcontroller.

Program code:

```
Void main()
{ int count=0;
  TRISD  = 0xff;
  TRISB  = 0x00;
  portb.f0 = 0;
```

```
while (1)
{
  if (portd.f0 == 1)
  {
    delay_ms (200);
    if (count == 0)
        Count = 1;

    else
        count = 0;
  }
  if (count == 0)
  {
    portb.f0 = 0;
    portb.f5 = 0;
    delay_ms (200);
  }
  else
  {
    portb.f0 = 1;
    portb.f5 = 1;
    delay_ms (200);
  }
}
}
```

Experiment no: 06

Name of the Experiment: Dot matrix display interfacing with PIC16F877A microcontroller.

Objective(s):

(i) To design and understand the circuit diagram of dot matrix display.

(ii) To learn how dot matrix display works and learn the interfacing of dot matrix display with PIC16F877A microcontroller.

Theory: A Dot matrix display is a display device which contains light emitting diodes aligned in the form of matrix. Dot matrix display are used in applications where symbol, graphic, characteristics, alphabets, numbers are needed to be displayed in static as well as scrolling motion.

A typical dot matrix display is shown below.

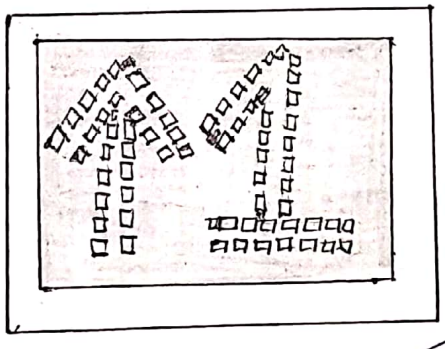


Figure 6.1: A typical dot matrix display is using in lift.

Types of dot matrix:

Dot matrix display is manufactured in various dimensions like 5x7, 8x8, 16x8, 32x8, 64x64, and 128x64 where the numbers represent LED's in rows and columns.

Construction of matrix display:

In dot matrix display, multiple LED's are wired together in rows and columns. The matrix pattern is made either in row anode and column cathode or row cathode and column anode pattern. In row anode, column cathode pattern, the entire row is anode while all column serve as cathode.

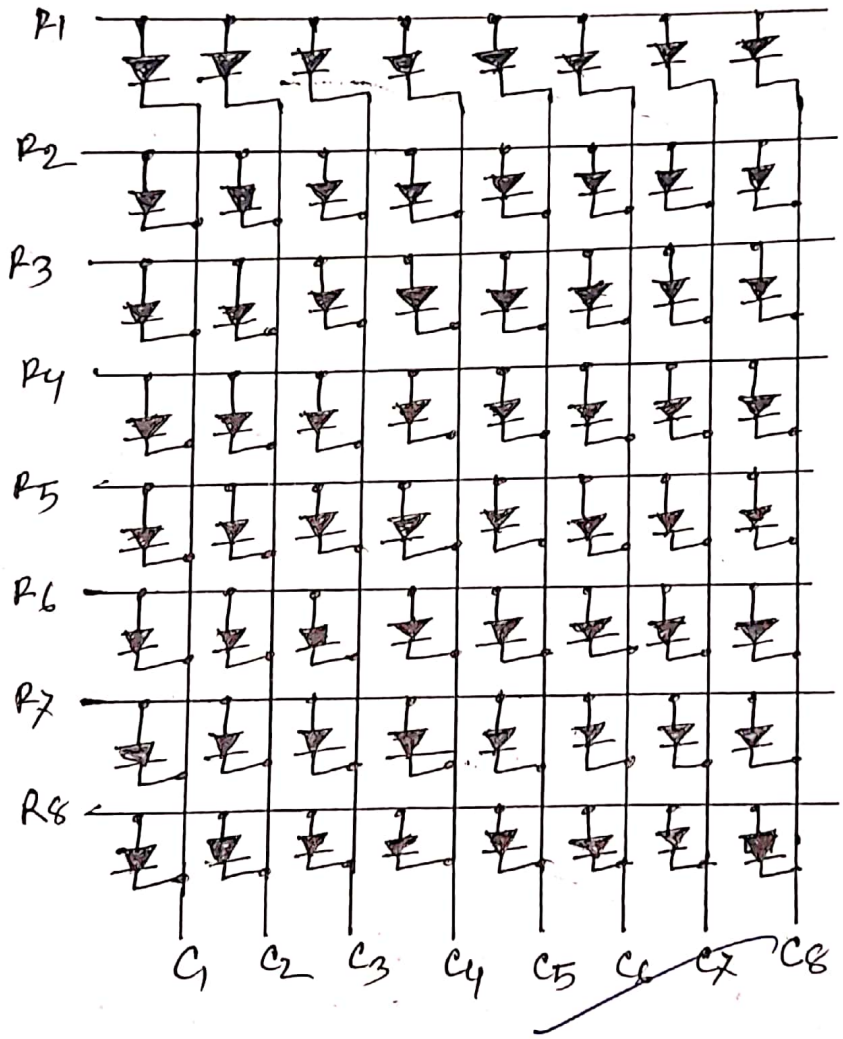


Figure 6.2: Construction of dot matrix display.

In dot matrix display, each LED can be control individually by controlling the current through each pair of column and row.

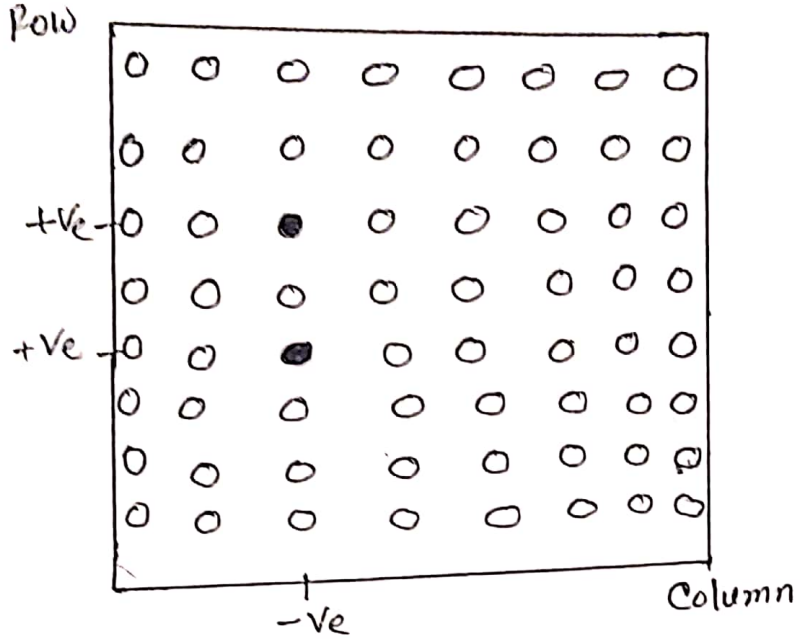


Figure-6.3: Working mechanism of dot matrix display.

How to generate code for Dot matrix Display:

	1	2	3	4	5	6	7	8	BIN	HEX
1	0	1	1	0	0	1	1	0	0b00000000	0x00
2	0	1	1	0	0	1	1	0	0b11111111	0xff
3	0	1	1	0	0	1	1	0	0b11111111	0xff
4	0	1	1	1	1	1	1	0	0b00011000	0x18
5	0	1	1	1	1	1	1	0	0b00011000	0x18
6	0	1	1	0	0	1	1	0	0b11111111	0xff
7	0	1	1	0	0	1	1	0	0b11111111	0xff
8	0	1	1	0	0	1	1	0	0b00000000	0x00

Apparatus Required: pic16F877A, crystal, capacitor, resistor, 8x8 dot matrix display, Darlington transistor array,

Circuit Diagram:

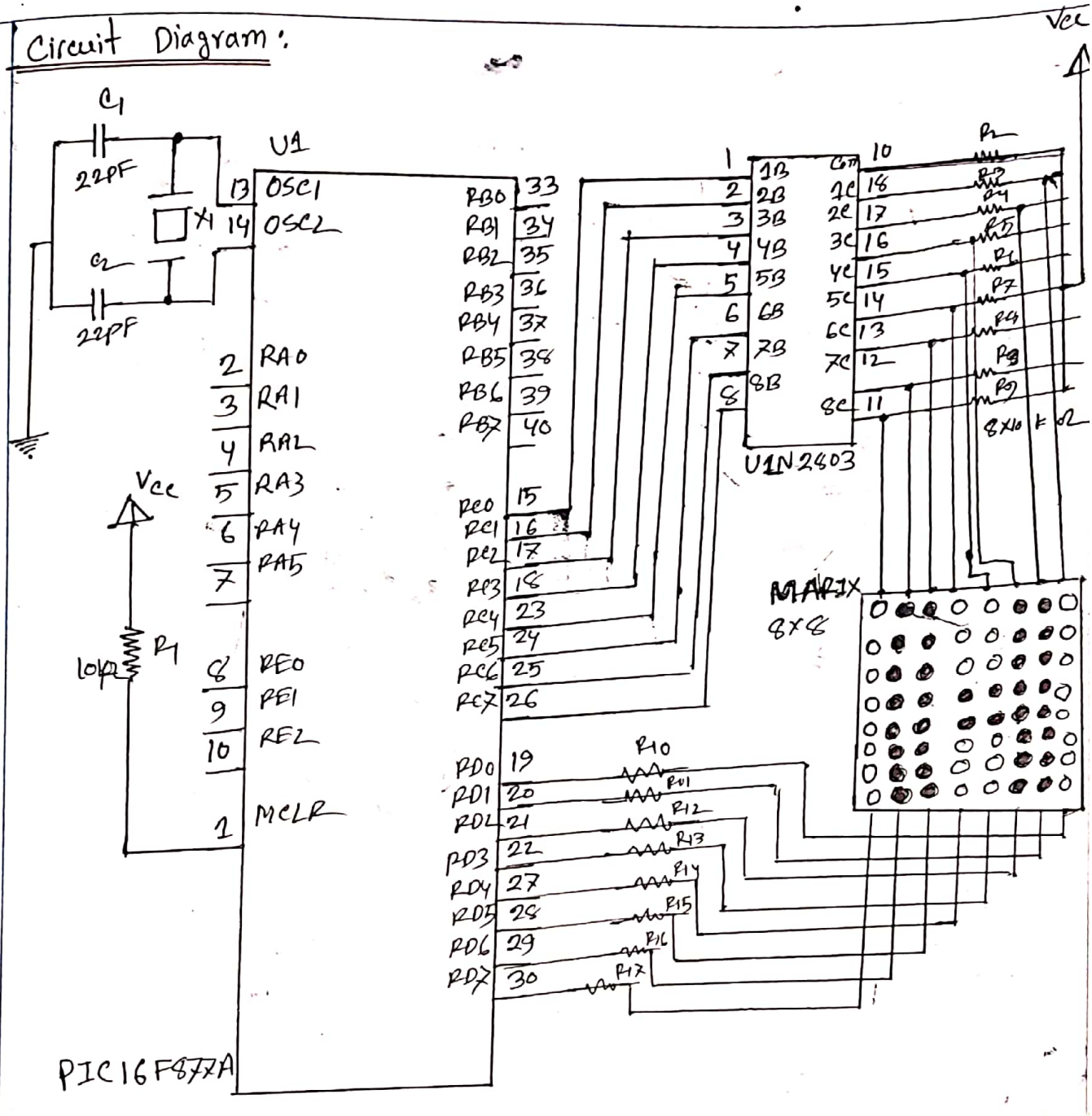


Figure 6.4: Proteus simulation of Dot matrix LED.

Programming Code:

```
void MsDelay (unsigned char time)
{
  unsigned char y,z;
  for (y=0; y<time; y++) {
      for (z=0; z<120; z++);
} }
```

```
Void main()
{ TRISB = 0x00;
  TRISC = 0x00;
  while(1)
  { PORTD = 0x80;
    PORTC = 0x00;
    MSDelay(10);
    PORTD = 0x40;
    PORTC = 0xff;
    MSDelay (10);
    PORTD = 0x20;
    PORTC = 0xff;
    MSDelay (10);
    PORTD = 0x10;
    PORTC = 0x18;
    MSDelay (10);
    PORTD = 0x08;
    PORTC = 0x18;
    MSDelay (10);
    PORTD = 0x04;
    PORTC = 0xff;
    MSDelay(10);
    PORTD = 0x02;
    PORTC = 0xff;
    MSDelay (10);
    PORTD = 0x01;
    PORTC = 0x00;
    MSDelay (10);
  }
}
```

Experiment no: 07

Name of the Experiment: LM35 Temperature sensor Data Read and Display using LCD ~~module~~ module.

Objectives:

(i) To learn about Temperature sensor using LCD module.
(ii) To learn and understand the circuit diagram of LCD module.
(iii) To learn interfacing LCD with PIC microcontroller.

Theory: To interface LCD with a PIC microcontroller, we use general purpose input-output pins (GCPIO pins). Because we send control and data signals to LCD through those I/O pins.

16×2 LCD pinout:

It consists of 14 pins. There are 4 data pin from D_0–D_7 and these controls pins such as RS, RW and E. LED + and LED - pins are used to control the backlight LED.

LM35 Temperature sensor:

(i) LM35 is a temperature measuring device having an analog output voltage proportional to the temperature.

(ii) It provides output voltage in centrigrade (celcius). It does not required any external calibration circuitry.

(iii) The sensitivity of LM35 is 10mV/ degree celcius. As temperature increases, output voltage also increases.

(iv) It is a 3-terminal sensor used to measure surrounding temperature ranging from -55°C to 150°C

(v) LM35 given temperature output which is more precise than thermistor output.

Apparatus Required : PIC16F677A, Capacitor, Crystal, LCD display, potentiometer, Temperature Sensor.

Circuit Diagram :

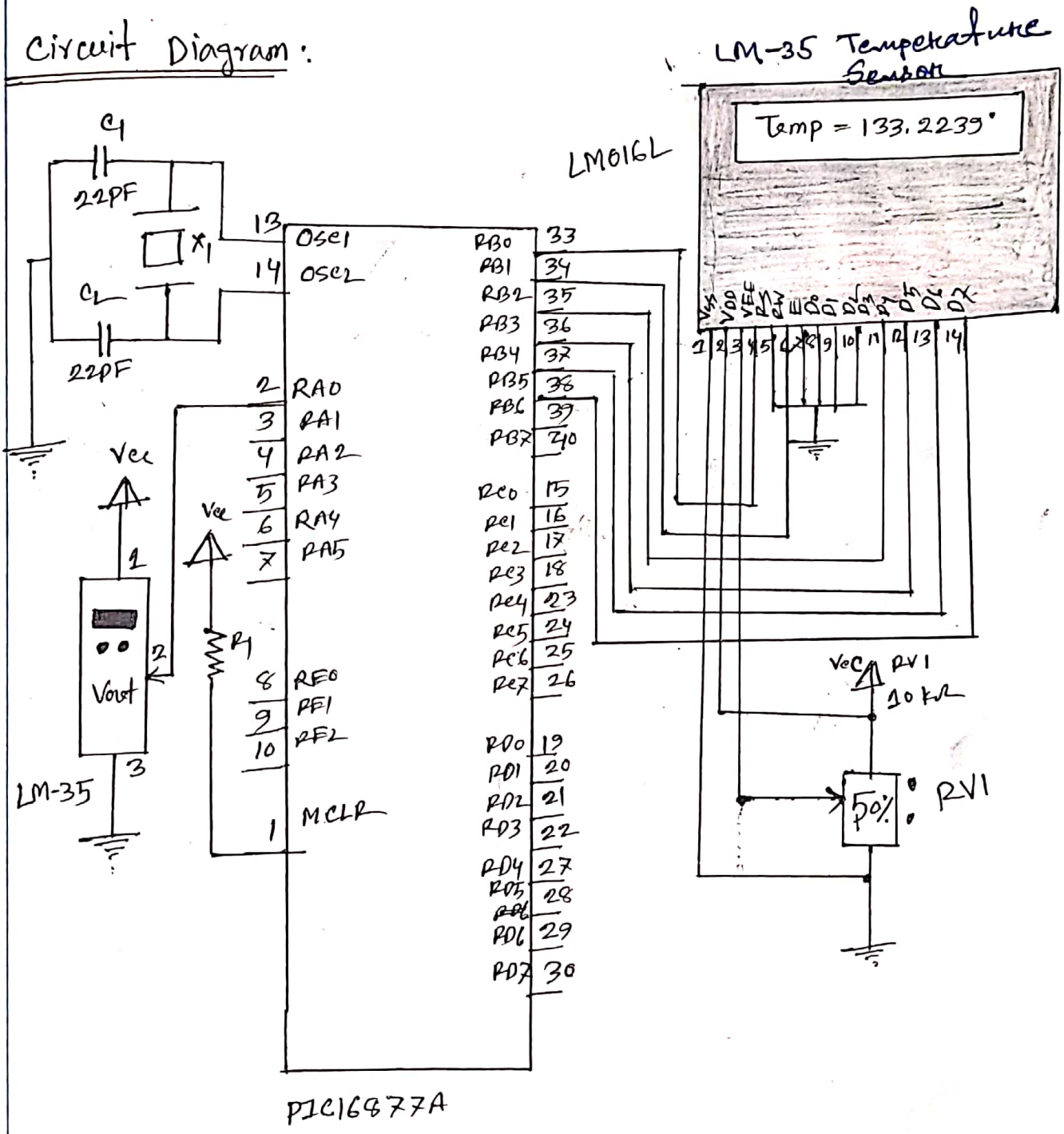


Figure-7.1 : Displaying LM35 temperature sensor data using LCD display.

Mikro C code:

```
sbit LCD_RS at RB0_bit;
sbit LCD_EN at RB1_bit;
sbit LCD_D4 at RB2_bit;
sbit LCD_D5 at RB3_bit;
sbit LCD_D6 at RB4_bit;
sbit LCD_D7 at RB5_bit;

sbit LCD_RS_Direction at TRISB0_bit;
sbit LCD_EN_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB4_bit;
sbit LCD_D7_Direction at TRISB5_bit

char display[16] = " ";
void main()
{
   unsigned int result;
   float volt, temp;
   trisb = 0x00;
   trisa = 0xff;
   adcon1 = 0x80;
   Lcd_init();
   Lcd_cmd(_lcd_clear);
   lcd_cmd(_lcd_CURSOR_OFF);
   while(1){
   result = adc_read(0);
   volt = result * 4.88;
   temp = volt/10;
   lcd_out(1,1,"Temp");
   floattostr(temp, display);
   Lcd_out_cp(display);
   lcd_chr(1,16,223);
   lcd_out_cp("C");
}
}
```

Experiment no : 08

Name of the experiment: Write a program to control high voltage load using mechanical relay.

Objectives:

(i) To learn how to interface ac load trough relay with pic microcontroller.

(ii) To design and oparate the circuit of relay interfacing.

Theory: A relay is electromagnetic switch which is used to switch high voltage/current using low power circuits. Relay isolates low power circuits from high power circuits. It is activated by engineering a coil rounded on a soft iron core. A relay should not be directly connected to a micro-controller. Because, A microcontroller is not able to supply current required for the working of relay. Maximum current that pic microcontroller handle is 25 mA while a relay needs about 50-100 mA current.

Also, a relay is activated by enginering its coil. Microcontroller may stop working by the negative voltages produced in the relay due to its back emf.

A relay can be eassily interfaced with microcontroller using a transistor, which carries the current required for operation of the relay.

Apparatus Required: PIC16F877A, Crystal, Capacitor, Resistor, Transistor (BC547BD), Diode, Bulb, Relay, ac voltage.

Circuit Diagram:

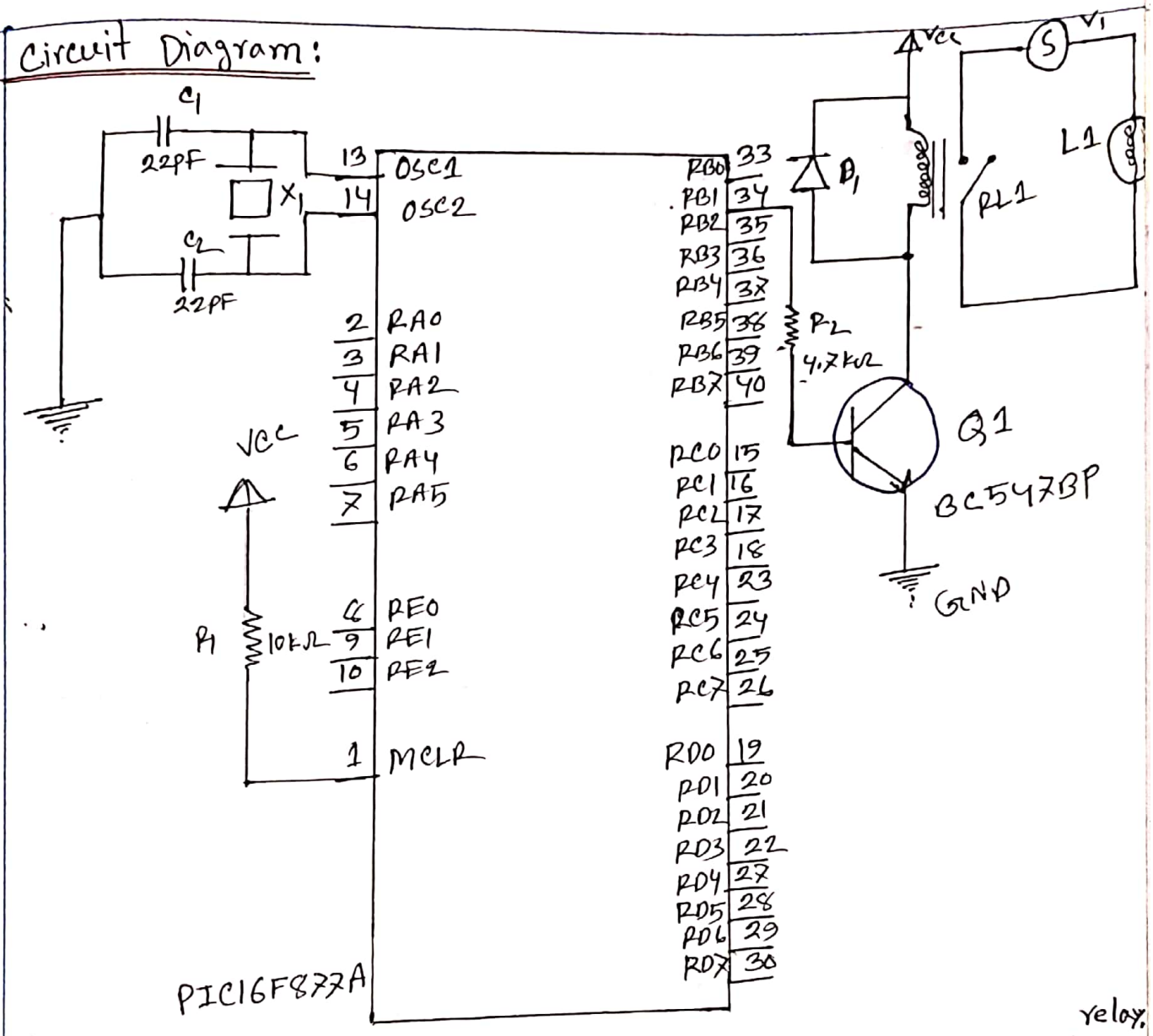


Figure 8: Controlling a high voltage load using mechanical relay.

Program Code:

```
Void main()
{ TRISB = 0x00;
  portb = 0x00;
  while(1)
  { port.f1 = 1;
    delay_ms(2000);
    portb.f1 = 0;
    delay_ms(2000);
  }
```

Result and Discussion:

For interfacing relay, we use a mechanical relay and we set an AC bulb to observe the result. We are here controlling high voltage load (bulb) using 5 volt relay. Here the relay is isolating the circuit into two parts. When we start the circuit the simulation process we can observe that the bulb is glowing then for a while it stop glowing. After that, it again start glowing. And this process will go on. This way, we can control high voltage load with low voltage relay.

Experiment no : 09

Name of the Experiment: Write a program for DC motor speed control using PWM and microcontroller.

Objectives:

(i) To learn how to control the average power delivered to a load using PWM and how to control the speed of DC motor.

(ii) To understand the circuit diagram.

Theory: PWM stands for Pulse Width Modulation. A modulation technique that generates variable width pulses to represent the amplitude of an analog input signal.

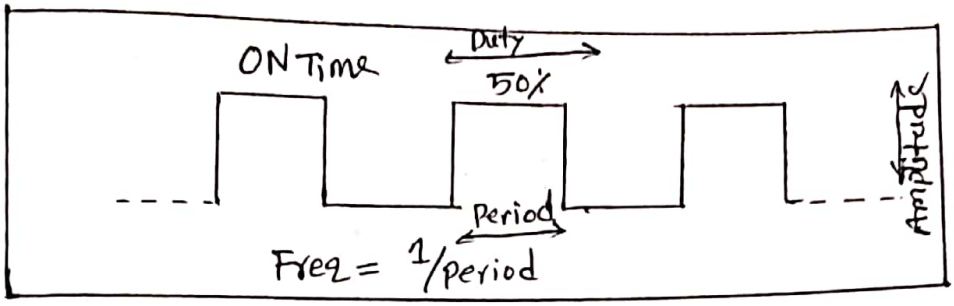
PWM is a type of signal which can be produced from a digital ic such as microcontroller or 555 timer. The signal produced will have a train of pulses and these pulses will have be in form of a square wave. That is, at any given instance of time the wave either be high or will be low. The duration at which the signal stays high is called "on time" and the duration at which the signal stays low is called "off time".

Duty cycle of PWM:

A PWM signal stays on for a particular time and then stays off the rest of the period. The percentage of time in which the PWM signals remains high is called as duty cycle. If the signal is always ON it is on 100% duty cycle and it is always off then 0% duty cycle.

Formula

$$\text{Duty cycle} = \frac{\text{Turn ON Time}}{\text{Turn ON Time} + \text{Turn OFF Time}}$$



By controlling the duty cycle from 0% to 100% we can control "on time" of PWM signal and thus the width of signal.

Frequency of PWM:

The frequency of PWM signal determines how fast a PWM completes are period,

$$\text{Frequency} = \frac{1}{\text{Time period}}$$

$\therefore$ Time period = ON Time + off time

Normally the PWM signals generates by microcontroller will be around 500 Hz which is used in high speed switching devices like inverters or converters. To control a servo motor we used to produce PWM signal with 50 Hz frequency.

Features of L293D:

(i) Width supply voltage range 4.5V to 36V and separate input logic supply.
(ii) Internal ESD protection and high-noise Immunity inputs.
(iii) Output current 1A per channel (600mA for L293D).
(iv) Peak output current 2A per channel. (1.2A for L293D).
(v) Output clamp Diodes for Inductive Transient suppression.

Apparatus Required:

Pic microcontroller, Crystal, Resistor, Capacitor, 1293D IC, DC motor, switch.

Circuit Diagram:

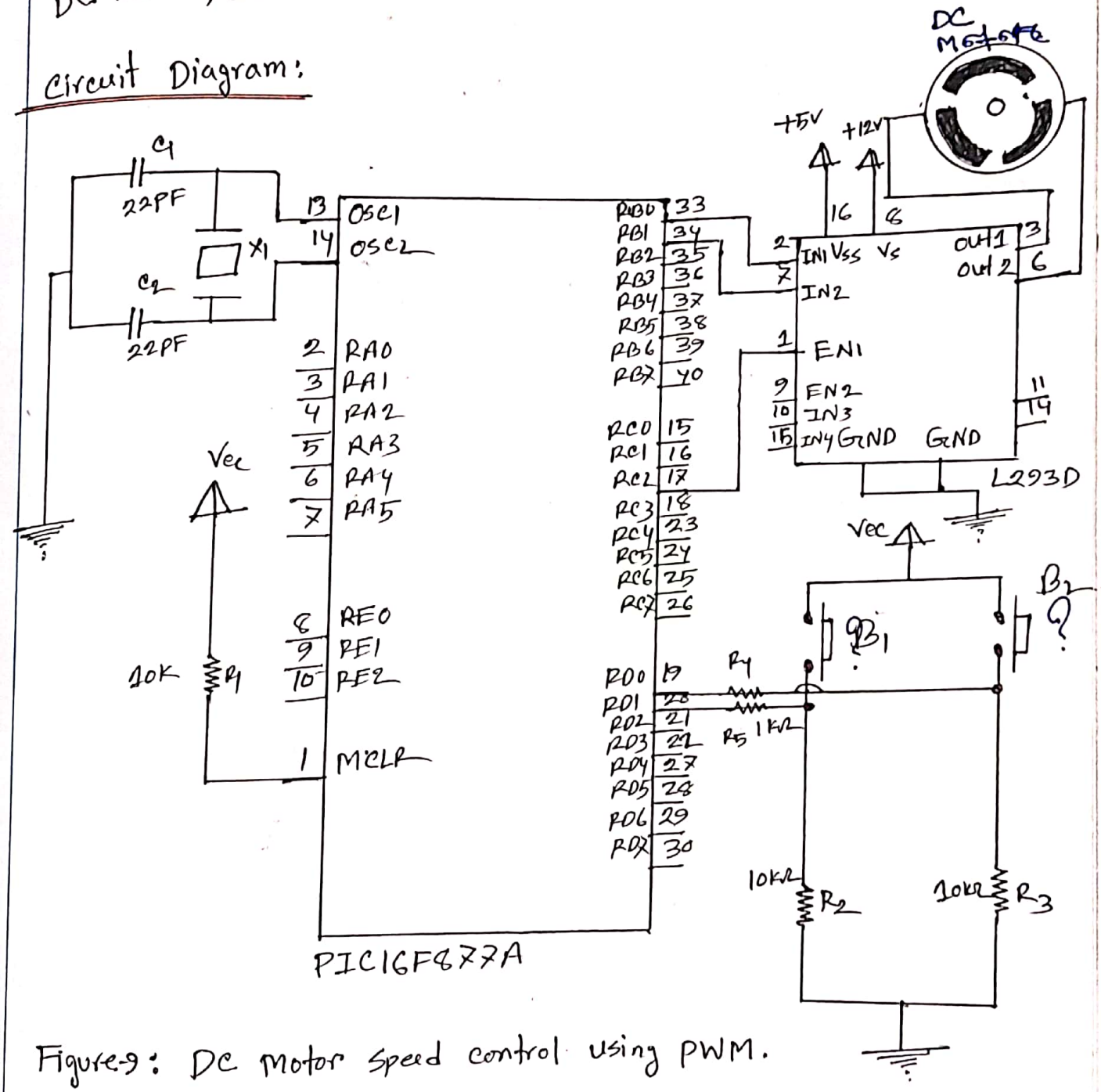


Figure-9: DC motor speed control using PWM.

Program code:

```
Void main()
{ short duty = 0;   // initial value for duty
  TRISD = 0xff;     // PORTD as input
  TRISB = 0x00;     // PORTB as output
  PORTB.F0 = 0xff;  // Run motor in antielockwise
  PORTB.F1 = 0x00;
  PWM1_Init(1000);  // Initialize PWM
  PWM_Start();      // start PWM
  PWM1_Set_Duty(duty);  // set current duty for PWM1
  While(1)
  { if (RD0_bit && duty <250)   // if button on RD0 pressed
      { Delay_ms(100);
        if(RD0_bit && duty <250)
        { duty = duty +10;
          PWM1_Set_Duty(duty); // change the duty cycle
        }
      }
  }
  if(RD1_bit && duty >0)
  {  Delay_ms(100);
     if(RD1_bit && duty>0)
     { PWM1_Set_Duty(duty);
     }
  }
  Delay_ms(100);
}
```

Result and Discussion:

In this experiment, we wanted to learn how we can control any motor depending on pwm. For this experiment, we set a dc motor to see the output. We set portb as output and portd as input. We when start the simulation we see that at the begining motor was stopped. When we pressed the speed up button the speed of motor started increasing. We can observe that motor was roating. At the same time when we press the speed down button the speed of motor is decreasing and once a time the motor is stopped. This is how we control the DC motor using PWM.

Experiment no: 10

Name of the experiment: Write a program to control servo motor using the pic microcontroller.

Objectives:

(i) To design and understand the circuit diagram of servo motor.

(ii) To learn how servo motor works and learn the interfacing of a servo motor with PIC16F877A microcontroller.

(iii) Programming to control servo motor and hardware connections of servo motor with PIC16F877A microcontroller.

Theory: A servo motor is a special kind of motor that operants upons the instructions. It provides singular precision, which means unlike other electrical motors that keep on roating until power is applied to them and stopped when power switched off, the servo motor roates only to a certain degree or until it is required to and then the motor stops and wait for the next instruction to carry out further action. Servo motor are connected with the help of servomechanism. Servo motors are used for precise positioning in robotic arms, legs etc, Aeroplanes, Helicopters etc. Hobby servo motors have three wires, two of them (RED and BLACK) are used to given power. and third one is used to give control signals. Servo motor can easily controlled using microcontrollers. Using PWM signals on the control wire. Here we are using a servo motor whose angular roation is limited to 0–180°. The input to its control line determines the position demanded for the output shaft.

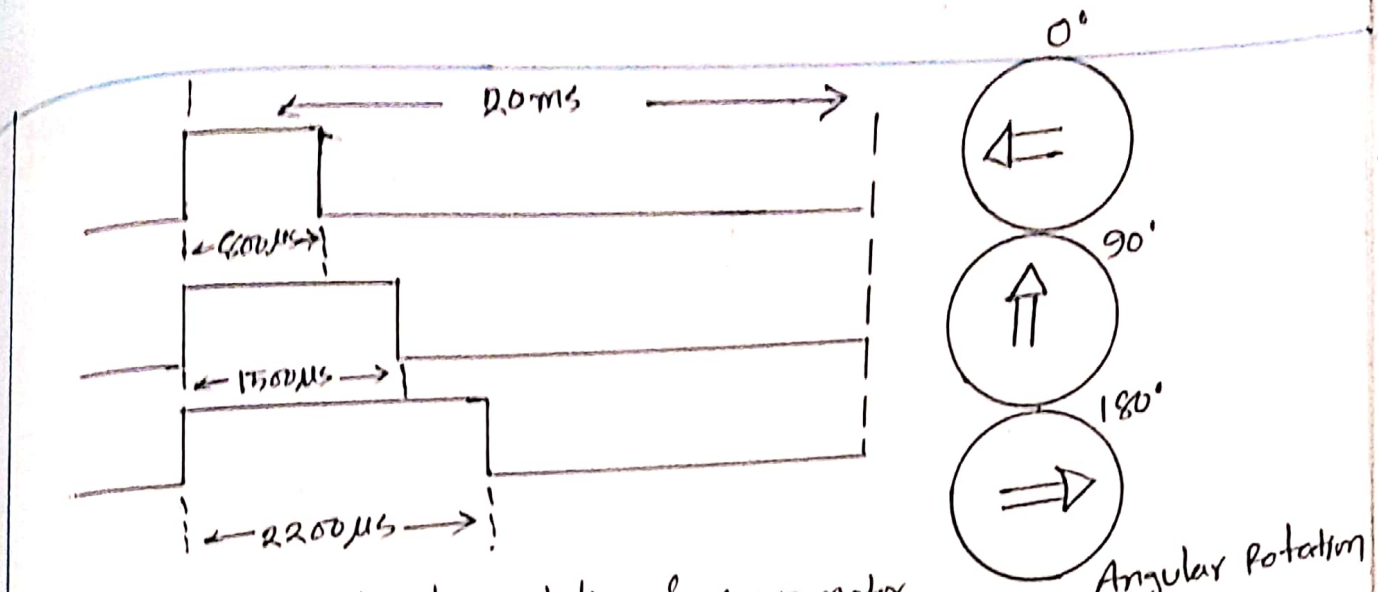


Figure-10.1 : Angular rotation of servo motor.

Apparatus Required: PIC16F877A, crystal, capacitor, servo motor.

Circuit Diagram:

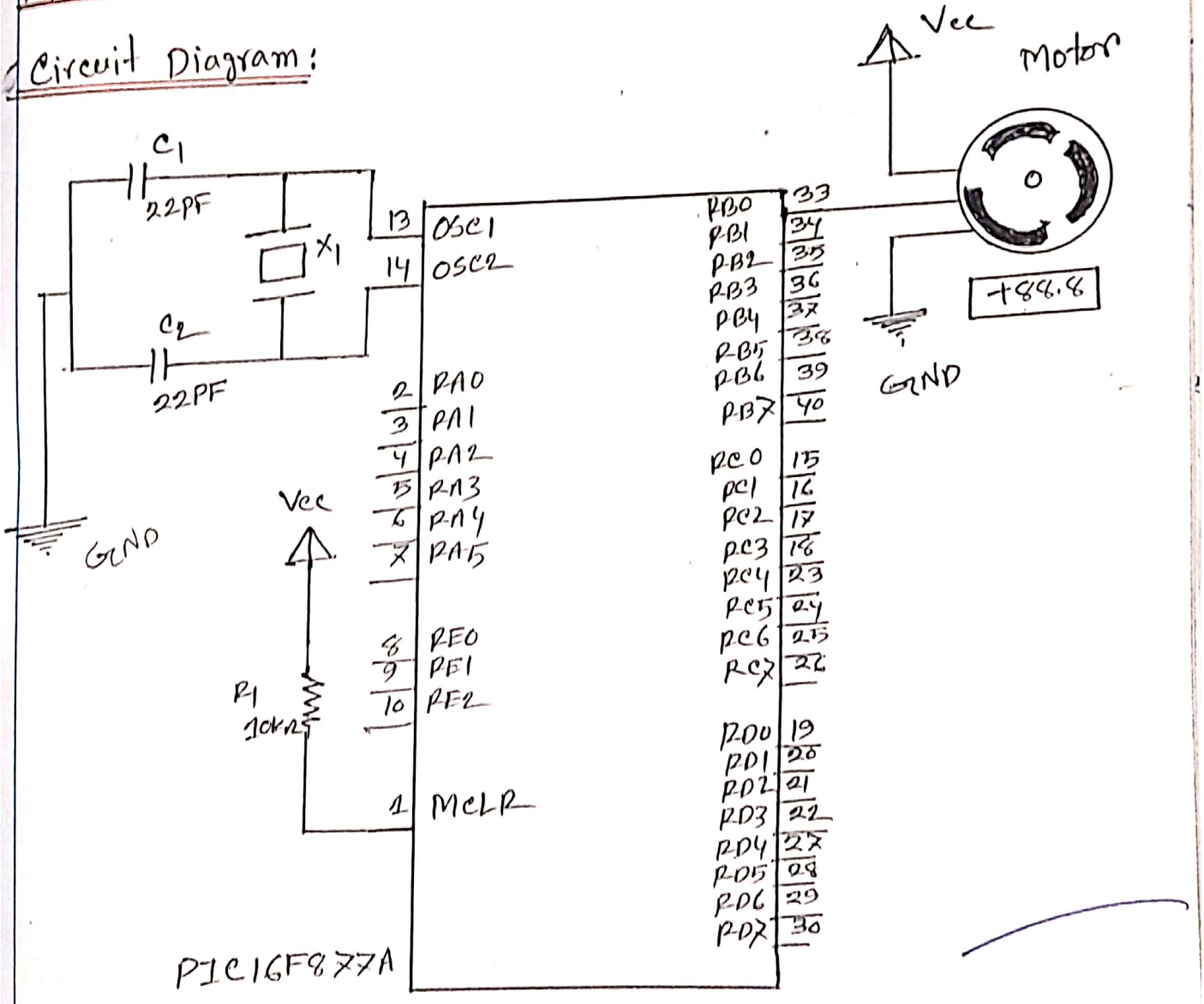


Figure 10.2 : Servo motor using PIC microcontroller.

Micro C code:

```
void servoRotate()
{ unsigned int i;
  for (i=0; i<50; i++)
  { portb.f0 = 1;
    Delay_ms (800);
    portc.f0 = 0;
    Delay_us (9200);
  }

void servoRotate90()
{ unsigned int i;
  for (i=0; i<50; i++)
  { portb.f0 = 1;
    Delay_us(1500);
    portb.f0 = 0;
    Delay_us (8500);
  }
}
Void servoRotate180()
{ unsigned int i;
  for (i=0; i<50; i++)
  { portb.f0 = 1
    Delay_us(2200);
    Portb.f0 = 0;
    Delay_us (17800);
  }
}
```

```
Void main()
{ trisb=0;
  do {
     servoRotate();
     Delay_ms(2000);
     servoRotate90();
     Delay_ms(2000);
     servoRotate180();
     } while(1);
}
```

Result and Discussion:

In this experiment we observed how we can interface servo motor with pic microcontroller. We set portb of output and we set a servo motor there. When we start the simulation we can see that first the motor is in -90 degree position. After that, the servo motor is in +0.09 position. Finally the motor is in +90 degree position And the motor repeat this process. This is how we can control servo motor to rotate only to a certain degree.

Experiment No: 11

Name of the Experiment: Write a program for interfacing stepper motor with PIC microcontroller.

Objectives:

(i) To describe and explain the operation of a stepper motor.

(ii) To design a stepper motor with PIC16F877A and understand it's circuit diagram.

Theory: A stepper motor is a brushless, synchronous, DC electric motor which devides the full rotation into a number of equal steps. It finds great application in field of microcontroller such as robotics. Unipolar motor is the most popular stepper motor using analog electronics hobbyist because of its ease of operation and availability.

Stepper motor can easily interfaced with pic microcontroller by using readymade ICs such as L293D, ULN2003. We have three different types of stepping modes for unipolars stepper motors.

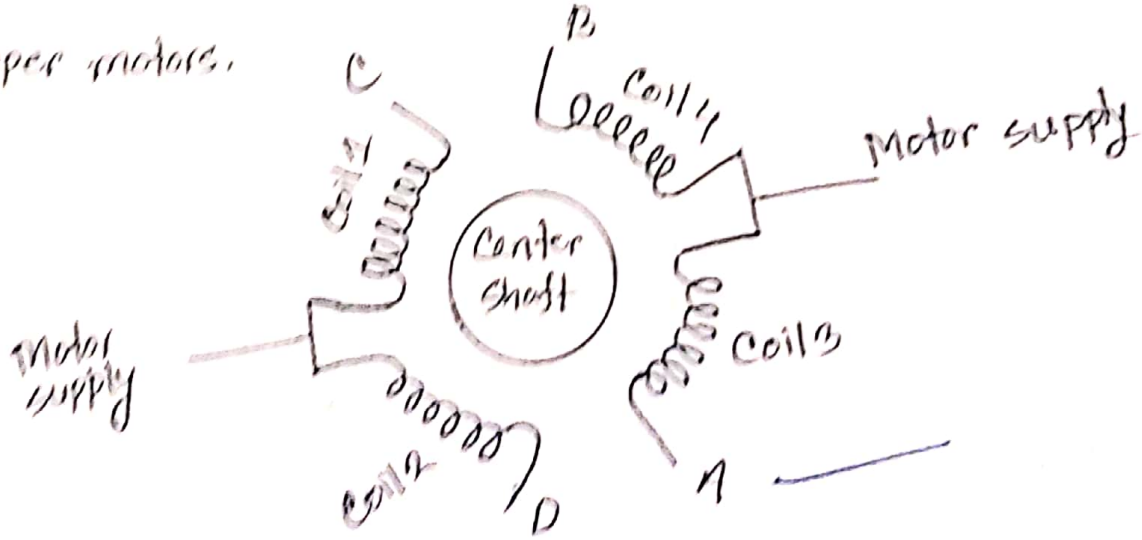


Figure-11.1: Unipolar stepper motor winding.

Wave drive:

In this mode only one stator electro magnet is energised at a time. It has the same number of steps as the full step drive but the torque is significantly less. It is rarely used. It can be used where power consumption is more important than torque.

Wave drive stepping sequence				
step	A	B	C	D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Full drive: In this mode two stator electromagnets are energised at a time. It is the usual method used for driving and the motor will run its full torque in this mode of driving.

Full drive stepping sequence				
step	A	B	C	D
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Table-11.2: Full drive stepping sequence

Half drive: In this stepping mode, alternatively one or two phases are energised, This mode is commonly used to increase the angular resolution of the motor but the torque is less approximately 70% at this half step position. We can see that the angular resolution doubles in half drive mode.

Half Drive Stepping Sequence				
Step	A	B	C	D
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

Table-11.3: Half drive stepping sequence

Driving Bipolar Motor:

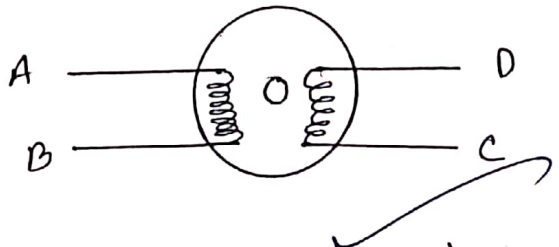


Figure-11.2: Bipolar stepper motor windings.

Bipolar motors are simpler in construction as it contains two coils and no centre tap. Being simple, driving is little complex compared to unipolar motors. To reverse the magnetic polarity of stators windings, current through it must be reversed. For this we should use H-Bridge. Here we are using 1293D, H-Bridge Motor Driver for that, we can distinguish bipolar motors from unipolar motors by measuring the coil resistance. In bipolar motors we can find two wires with equal resistance.

Bipolar Motor stepping sequence				
Step	A	B	C	D
1	1	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	0	0	1

Table 11.4: Bipolar motor stepping sequence.

Apparatus Required:

PIC16F877A, Crystal, Capacitor, Resistor, Stepper motor, Battery, L293D.

Circuit Diagram:

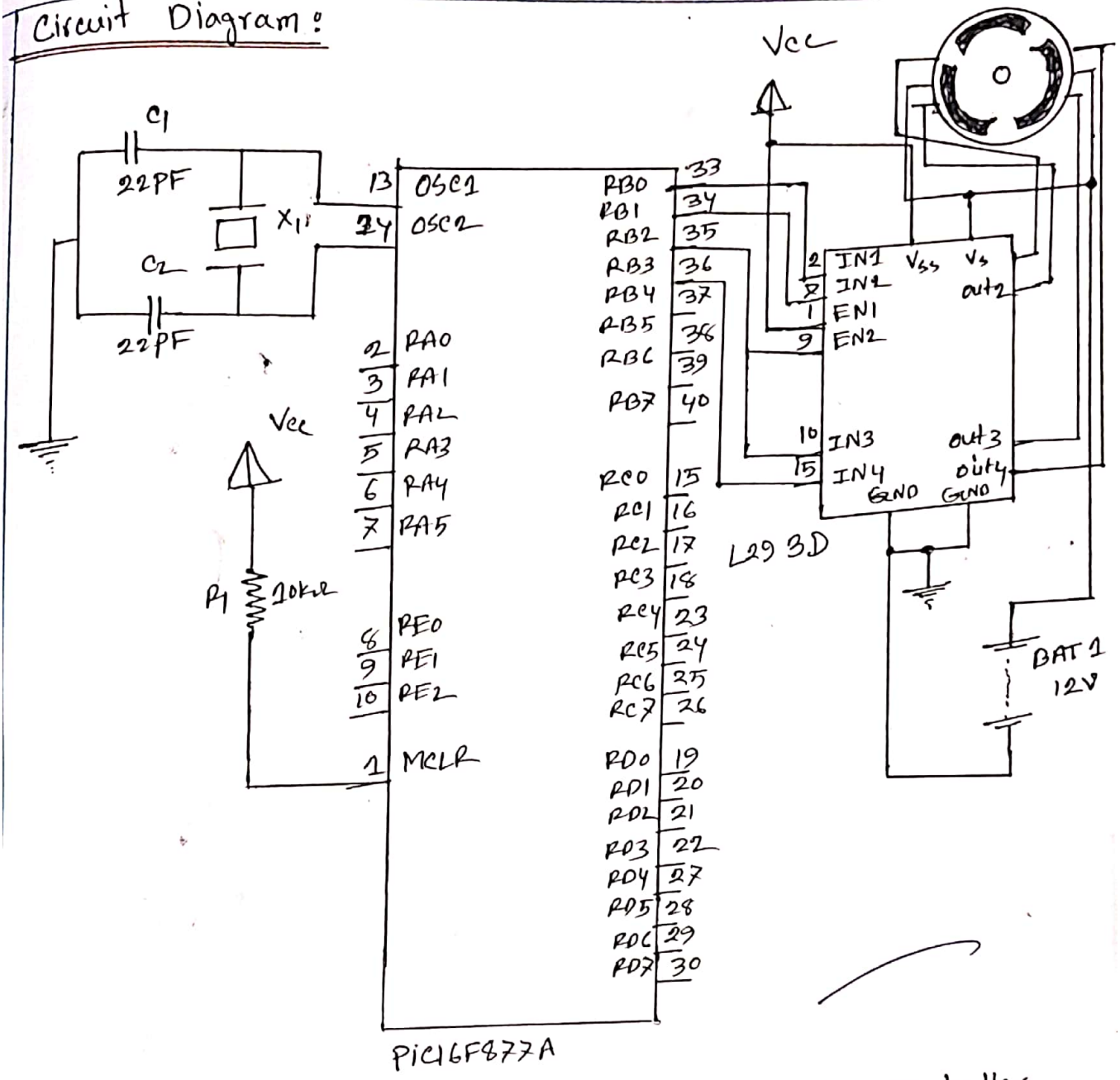


Figure-11.3: Stepper motor using PIC Microcontroller.

Mikro C code:

```
void main()
{ CMCON = 0x07;   // To turn off comparator
  ADCON1 = 0x06;  // To turn off analog to digital Conve
  TRISB = 0;
  PORTB = 0x0F;
```

```
do{
 PORTB = 0b00000001;
 Delay_ms (500);
 PORTB = 0b00000011;
 Delay_ms (500);
 PORTB = 0b00000010;
 Delay_ms (500);
 PORTB = 0b00000110;
 Delay_ms (500);
 PORTB = 0b00000100;
 Delay_ms (500);
 PORTB = 0b00001100;
 Delay_ms (500);
 PORTB = 0b00001000;
 Delay_ms (500);
 PORTB = 0b00001001;
 Delay_ms (500);
 } while(1);
}
```

Result and Discussion: In this experiment we observed how to interface a stepper motor using PIC microcontroller. We set portb as output. Here we connected a stepper motor. When we sta the simulation we can see that first the motor rotate at +90°. Then it rotate to 0 degree. After that stepper motor shows u value -90 degree. Then stepper motor rotate to -180, -270, and -3 degrees, respectively. stepper motor is continuouly rotating between -90 to -360 valve.