Table of Contents

[Objective 3](#_Toc389885666)

[Hardware 4](#_Toc389885667)

[Microcontroller PIC18F25k22 4](#_Toc389885668)

[Keypad 4](#_Toc389885669)

[ULN2003A 4](#_Toc389885670)

[LCD 4](#_Toc389885671)

[Stepper motor 4](#_Toc389885672)

[Software 6](#_Toc389885673)

[Flow chart 6](#_Toc389885674)

[Software description 7](#_Toc389885675)

[Operation of Keypad 7](#_Toc389885676)

[LCD Display 7](#_Toc389885677)

[Flow rate calculation 8](#_Toc389885678)

[Project Code 9](#_Toc389885679)

[Verification of circuit operation 14](#_Toc389885680)

[Verification of motor control 20](#_Toc389885681)

[Simulation 20](#_Toc389885682)

[Calculated output 22](#_Toc389885683)

[Conclusion 23](#_Toc389885684)

# 

# Objective

The objective of designing the medical infusion pump is to develop the circuit designing and programming skills. It also helped us to learn the utilization of our skills for the well being of mankind. Keypad, LCD, interrupts and microcontrollers are very important part of digital circuits. So in this project we utilized these items to get complete grasp on them. The idea of the project is very simple. The user selects the infusion rate with the help of buttons on a key pad. The selected rate is displayed on a connected LCD Display. Only one button is pressed at a time and the corresponding displayed digit is incremented on each button press. If the button is held down the corresponding displayed digit will increment after each half second. When a “START” button is pressed, the system generates drive impulses to a uni-polar stepper motor .When the “STOP” button is pressed the drive and hence actuator stops movement.

# Hardware

Hardware design of medical infusion pump consists of LCD, ULN2003A, Pushbuttons, Stepper motor and the microcontroller PIC18F25k22. The apparatus of the project is shown in figure 1.

## Microcontroller PIC18F25k22

Microcontroller is the main component of this project. It is the brain of the circuit and controls the operation of all components of the circuit. In this project it takes input through the keypad. Interprets the input, performs the calculations and produce the desired output. PIC18F25k22 is an 8-bit microcontroller. It has 28 pins of which 24 are I/O pins. It has external hardware interrupts on PORTB and many features like ADC, UART, ECCP, SPI and I2C. The range of operating voltage is 2.3V to 5.5V.

## Keypad

Keypad used in this project consists of five buttons. These buttons are labeled as “TENS”, “UNITS”, “TENTHS”, “START” and “STOP”. These buttons are connected to RC4, RC5, RC6, RC7 and RB0 respectively.

## ULN2003A

ULN2003A is a high voltage, high current Darlington transistor array. It consists of seven Darlington pairs transistors with clamp diodes. It has high voltage output up to 50V. It is suitable for relay driver applications. It is used in the project to meet the power requirements for driving motor.

## LCD

LCD used in this project has 2 rows and 16 columns. It has 14 pins. Three pins are for power supply, three pins are to control the operation of LCD and 8 pins are for the data or commands. In order to optimize the circuit design we can use the LCD in 4-bit mode. Its operating voltage ranges from 2.7V to 5V.

## Stepper motor

Uni-polar stepper motor is connected to the microcontroller through ULN2003A IC. In this project, we have assumed that the stepper motor is connected via a 64:1 gear reduction to a drive shaft with a 1mm pitch. This means that 64 rotations of the stepper motor produces 1 complete gearbox shaft rotation which causes a captive nut mechanism attached to the drive shaft to advance by 1mm.

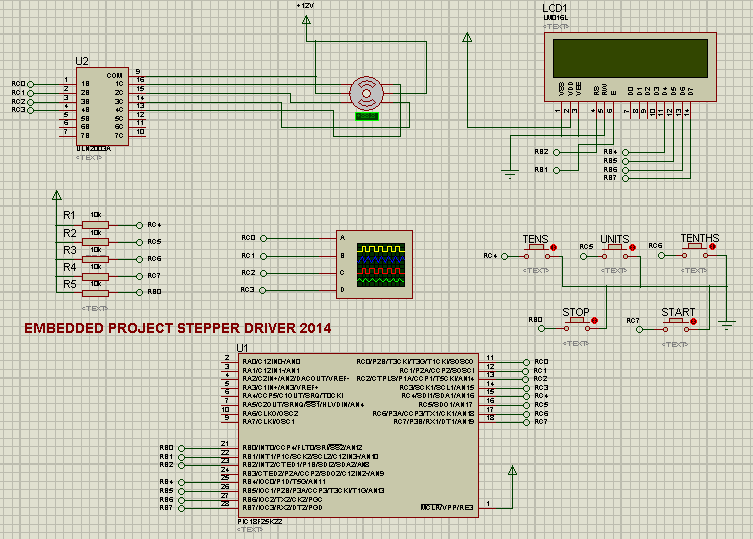


Fig 1: Hardware Design

# Software

## Flow chart

The flow chart of the code is shown in Fig 2.



Fig 2: Flow chart

## Software description

The project code is written in Mplab IDE v8.92 with compiler XC8. It consists of many built in and user defined functions. In the “main( )” these functions are called in a logical order to achieve the desired output. The operation of each hardware component is discussed below.

### Operation of Keypad

There are five pushbuttons, labelled “TENS”, “UNITS” and “TENTHS”, “START” and “STOP”. In the code status of four pushbuttons is checked continuously every half second using if statements. Only one button is active at a time. The “STOP” button is connected to external interrupt RB0.

When any of the buttons, “TENS”, “UNITS” and “TENTHS”, is pressed, the corresponding variable value gets incremented. The value of each variable continues to circulate between 0-9. This helps the user to set the flow rate of the infusion pump.

When the “START” button is pressed, microcontroller performs calculations on the selected flow rate and calculates the delay that needs to be placed between each stepper motor step.

The pushbutton “STOP” is used to stop the motor instantly.

### LCD Display

In the code, built in library of the LCD is used. The LCD library is added in the project code and its functions are called in the main program file to display data on the LCD.

In order to use LCD, first we need to initialize it. For this purpose a function, **init\_display( ),** from the LCD header file is called in the program file “main.c”.

A local function “display” is defined in the program file “main.c”. It contains some built in functions of the library LCD. The purpose of the built in functions for the LCD are discussed below.

**set\_address\_line1( char ):** It is used to set the cursor position on the first row of the LCD.

**set\_address\_line2( char ):** It is used to set the cursor position on the second row of the LCD.

**write\_data( char ):** It is used to display single character on the LCD.

**send\_msg( char[] ):** It is used to display a string on the LCD.

#### Decimal to ASCII conversion

In order to display characters on the LCD, first we need to convert them in to ASCII code. Therefore before sending the data to LCD, a local function “dec\_ASCII( )” is called in the program file “main.c”.

To generate an ASCII code for the respective character, a hexadecimal value of 0x30 is added in each character. For example 5+0x30 = 0x35, this is an ASCII code for character 5.

### Flow rate calculation

Whenever the “START” button is pressed, the microcontroller performs calculations for the selected flow rate and produces the calculated time delay after every step. The calculations for the time delay are given below.

**Given Data:**

The step angle of the stepper motor = 7.5º

Stepper motor shaft is connected to a 64:1 gear box

360º rotation of gear shaft moves the drive shaft by 1mm.

1.8mm of plunger travel will cause 1ml of fluid to be exited from the syringe

**Solution:**

Number of steps to complete one rotation of stepper motor shaft = = 48

Number of steps to complete one rotation of gear shaft = =

Number of steps to move drive shaft by 1mm = =

Number of steps to exit 1ml of fluid =

Number of steps per second to exit 1ml of fluid in an hour =

Where is a constant value

Let the selected flow rate =

Number of steps per second for selected flow rate =

We know that 1 s = 1000 ms

So time delay in milliseconds for selected flow rate =

This is the time delay between each step of the stepper motor.

## Project Code

#include <xc.h>

#include "LCD.h"

#pragma config WDTEN=OFF, FOSC=INTIO67,MCLRE=EXTMCLR,IESO=OFF, LVP=OFF

#define \_XTAL\_FREQ 4000000

unsigned char TENS1=0,UNITS1=0,TENTHS1=0, control=1;

unsigned char tens,units,tenths;

unsigned int rate=0;

unsigned char i;

double value,N,t;

double value,Syringe\_Calibration,t,Step\_Number;

void interrupt chk\_isr(void)

{ if(INTCONbits.INT0IF==1)

INT0\_isr();

}

void caluclation (void)

{ N=TENTHS1;

N=N/10;

value=(TENS1\*10)+UNITS1+N;

Step\_Number=48\*64;

Syringe\_Calibration=Step\_Number\*1.8;

Syringe\_Calibration= Syringe\_Calibration/3600;

value\*=Syringe\_Calibration;

t=1000/value;

t=t+0.5;

rate=t;

control=1; }

void Flow\_Rate (void)

{

for(unsigned int i=0;i<rate;i++){

\_\_delay\_ms(1);

}}

void INT0\_isr(void)

{

PORTC=0x00;

control=0;

INTCONbits.INT0IF=0;

}

void init(void)

{

OSCCON=0b01010010;//this to select internal osc at 4MHz

ANSELC=0x00;//make PORTC digital I/O

TRISC=0xF0;//set appropriate data direction bits

PORTC=0x00;//clear any PORTC outputs

ANSELB=0x00;//make PORTB digital I/O

TRISB=0x01;//set appropriate data direction bits (RB0 is input)

PORTB=0x00;//clear PORTB outputs

TRISA=0x0F; //all outputs except RA0 analogue IP

ANSELA=0x00; //make PORTA digital I/O except RA0 analogue

PORTA=0x00; //clear all PORTA digital pins

TRISBbits.TRISB0=1;

INTCONbits.INT0IF=0;

INTCONbits.INT0IE=1;

INTCONbits.GIE=1; }

void dec\_ASCII(void)

{

tens=(TENS1/1)+0x30;

units=(UNITS1/1)+0x30;

tenths=(TENTHS1/1)+0x30;

}

void display(void)

{

set\_address\_line1(1); //Set cursor position at 1 on line 1

send\_msg("FLOW= ");

set\_address\_line2(1);

write\_data(tens);

write\_data(units);

write\_data('.');

write\_data(tenths);

set\_address\_line2(6); //Set cursor position at 6 on line 2

send\_msg("ml/h ");

}

void main(void)

{

init();

init\_display();

TENS1=0;

UNITS1=0;

TENTHS1=0;

while(1)

{

if(!PORTCbits.RC4)

{

TENS1++;

if(TENS1 > 9)

TENS1 = 0;

for(i=0;i<4;i++)

\_\_delay\_ms(10);//make a delay

}

if(!PORTCbits.RC5)

{

UNITS1++;

if(UNITS1 > 9)

UNITS1=0;

for(i=0;i<4;i++)

\_\_delay\_ms(10);//make a delay

}

if(!PORTCbits.RC6)

{

TENTHS1++;

if(TENTHS1 > 9)

TENTHS1=0;

for(i=0;i<4;i++)

\_\_delay\_ms(10);//make a delay

}

if((!PORTCbits.RC7) && (TENS1 || UNITS1 || TENTHS1))

{

caluclation ();

while (control==1)

{

PORTC=0x09; //the sequence drives the stepper phases two at a time 1001

Flow\_Rate ();

PORTC=0x0C; // the sequence drives the stepper phases two at a time 1100

Flow\_Rate ();

PORTC=0x06; //the sequence drives the stepper phases two at a time 0110

Flow\_Rate ();

PORTC=0x03; //the sequence drives the stepper phases two at a time 0011

Flow\_Rate ();

}}

dec\_ASCII();

display();

\_\_delay\_ms(150);

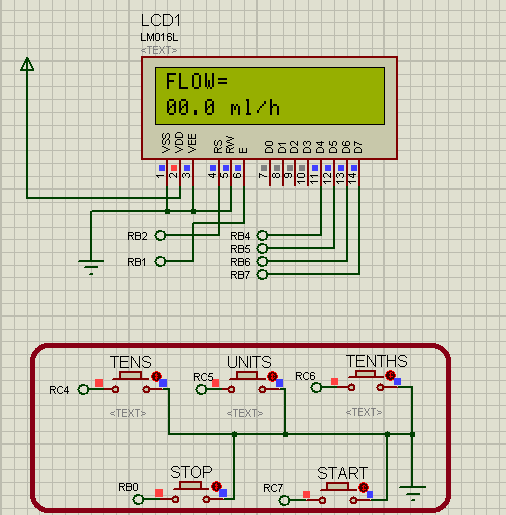
for(i=0;i<6;i++)

\_\_delay\_ms(10);//make a delay

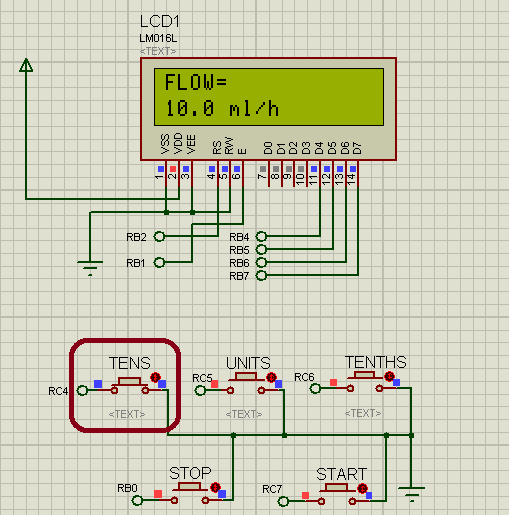
}}

# Verification of circuit operation

When “TENS” button is pressed, the digit at the tens position gets incremented by one. If it is kept pressed, the digit at the tens position continues to increase after every half second. This is depicted in the Fig 3.



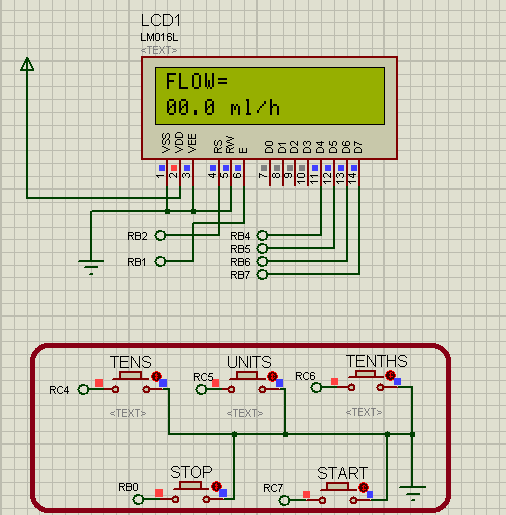
(a) No button pressed



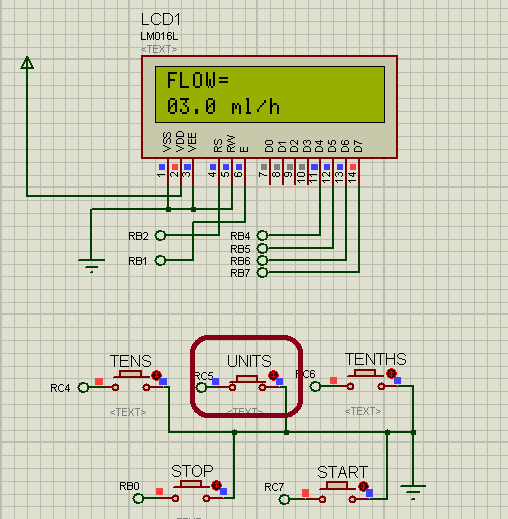
(b) When “TENS” button is pressed

Fig 3: Demonstration of “TENS” button

When “UNITS” button is pressed, the digit at the unit position gets incremented by one. If it is kept pressed, the digit at the unit position continues to increase after every half second. This is depicted in the Fig 4.



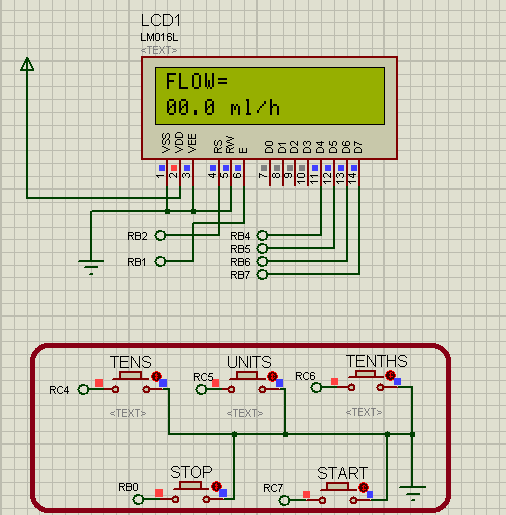
(a) No button pressed



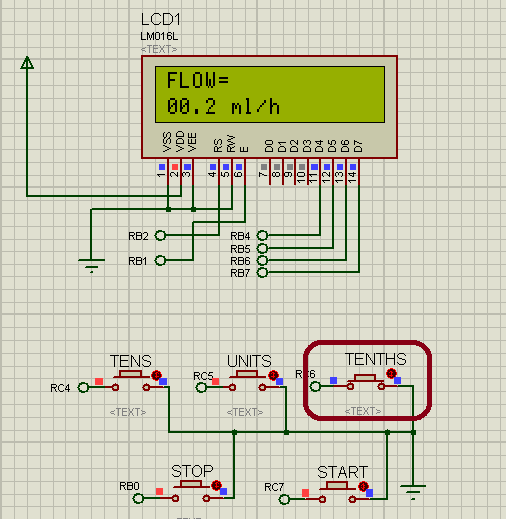
(b) When “UNITS” button is pressed

Fig 4: Demonstration of “UNITS” button

When “TENTHS” button is pressed, the digit at the tenths position gets incremented by one. If it is kept pressed, the digit at the tenths position continues to increase after every half second. This is depicted in the Fig 5.



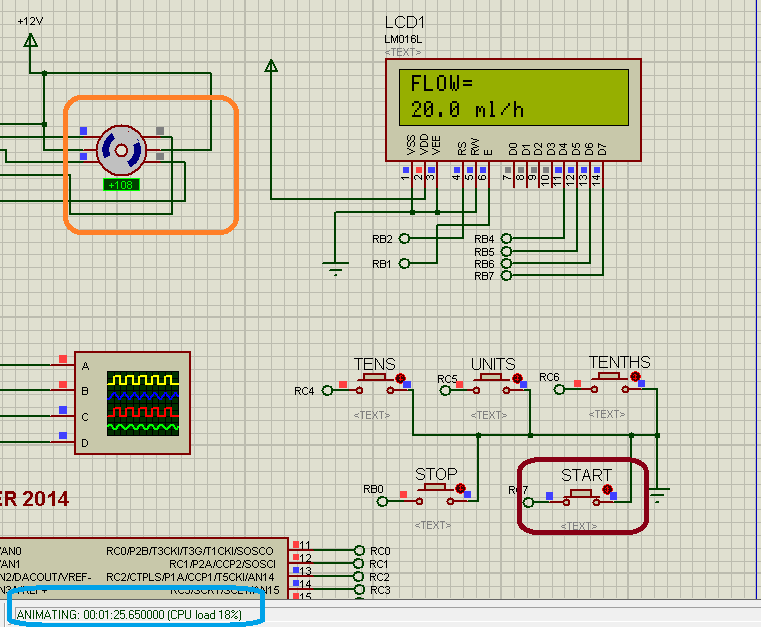
(a) No button pressed



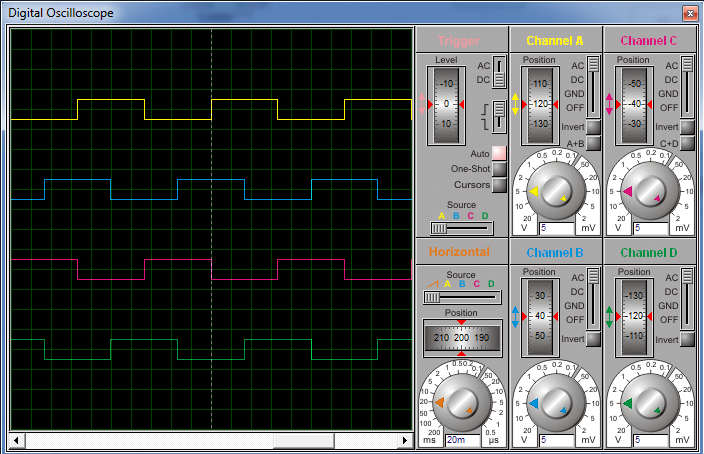
(b) When “TENTHS” button is pressed

Fig 5: Demonstration of “TENTHS” button

When user presses the “”START” button, stepper motor starts to rotate with the calculated step rate. This is demonstrated in Fig 6. It can be observed that on pressing start button motor starts to rotate. This phenomenon can also be observed on the oscilloscope.



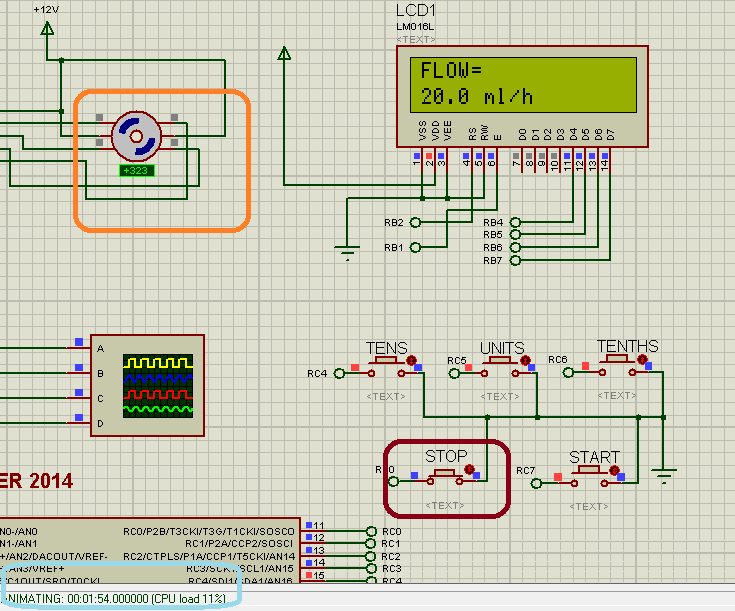
(a) Motor starts



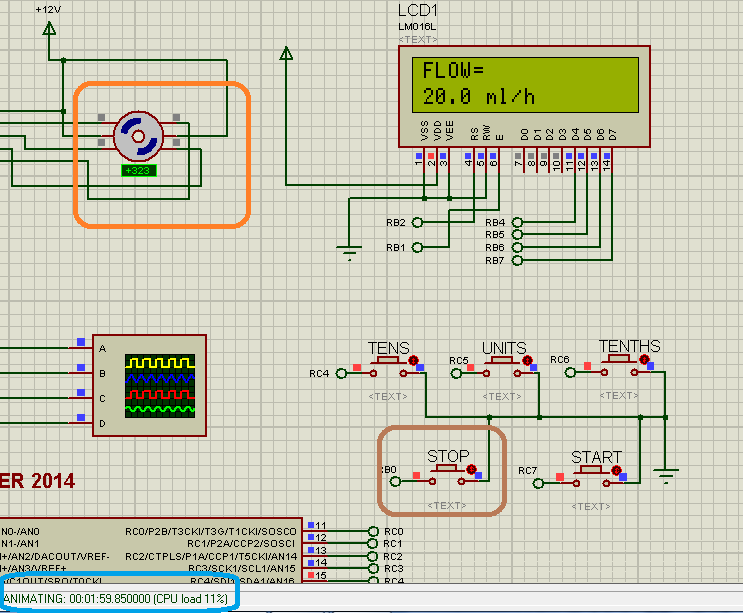
(b) Impulses sent by the microcontroller

Fig 6: Demonstration of “START” button

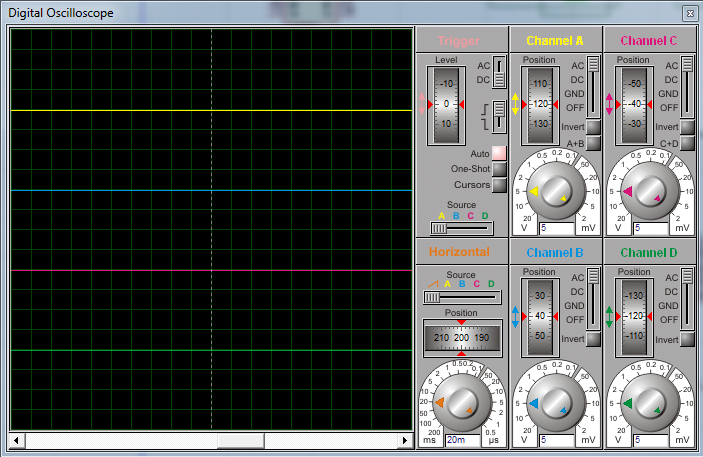
When user presses the “STOP” button, motor stops to rotate. This is demonstrated in Fig 7. It can be observed that motor angle has not changed after 29 seconds.



(a) When “STOP” button is pressed



(b) After releasing “STOP” button



(c) Wave form after pressing stop button

Fig 7: Demonstration of “STOP” button

# Verification of motor control

## Simulation

Observed data from the Fig 8 is given below,

Flow rate = 20.0 ml/h

Initial simulation time = t1 = 14.5 s

Initial stepper motor angle = A1 = 23.8º

# 

Fig 8: Initial values

Observed data from Fig 9, is given below,

Final simulation time = t2 = 15.55 s

Final stepper motor angle = A2 = 260º

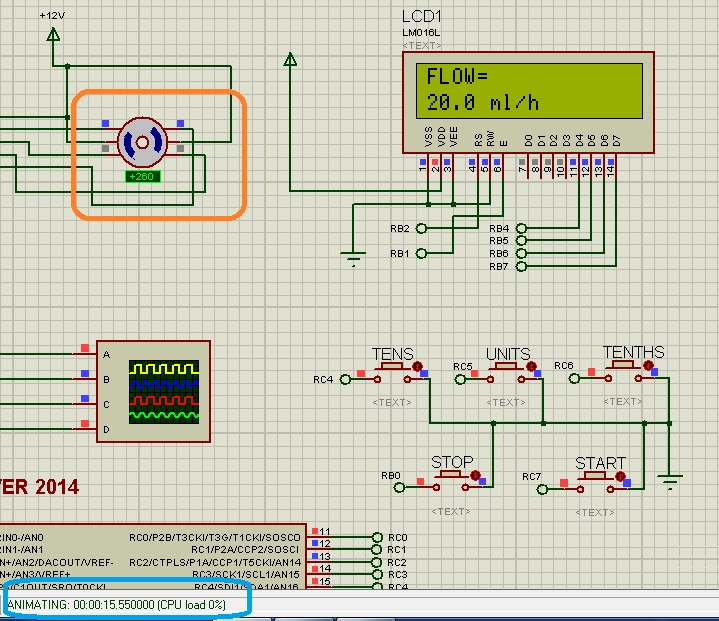


Fig 9: Final values

So

Time = t = t2 - t1 = 15.55 – 14.5 = 1.05 second

Angle of rotation in time t = As = A2 – A1

As = 260 – 23.8 = 236.2º

The simulation shows that in time T= 1s, the shaft of stepper motor covers an angle A = 236.2º

## Calculated output

As

The step angle of stepper motor = 7.5º

Flow rate = = 20.0 ml/h

Number of steps per second for selected flow rate =

As the time duration = = 1.05 s

So the shaft of stepper motor rotates in 1.05 seconds = Ac=

Ac =

Ac = 241.92º

The calculation shows that in time 1.05 s, the shaft of stepper motor rotates with angle = 241.92º

The simulation shows that in time 1.05 s, the shaft of stepper motor rotates with angle = 236.2º

Error = Ac – As = 241.92 – 236.2 = 5.72

Error = 5.72º

Error is very small and negligible

Hence

The calculated and simulation results are almost equal.

# Conclusion

The circuit design of medical infusion pump consists of keypad, LCD, ULN2003A and stepper motor. The stepper motor is connected to a 64:1 gear box. The shaft of the gear box is further connected to a drive shaft of the syringe. The user selects the infusion rate by the help of a keypad. The keypad consists of five buttons. These buttons are labelled as “TENS”, “UNITS”, “TENTHS”, “START”, “STOP”. Only one button is active at a time except the “STOP” button. “STOP” button is connected to an external hardware interrupt RB0. “TENS”, “UNITS” and “TENTHS” are used to select the flow rate. The “START” button is used to start the motor at the calculated speed. The “STOP” button is used to stop the motor at any time. The medical infusion pump works fine with very small error for the flow rate range 0.0 to 25.0 ml/h. For high flow rates the simulation results does not match the calculated results. It is due to very small time delay between the stepper motor steps.