BIRZEIT UNIVERSITY

Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

ENGINEERING SIMULATION LAB

(ENEE4104)

Experiment 10

"Keypads and DC Motors"

Prepared by:

Name: Mohammad Shehadeh Number:1201458

Section:1 Date:10/24/2024

Contents

Table of Figures

Abstract

The aim of this experiment was to implement a DC motor speed control system using keypad interface and PWM techniques. The project utilized a PIC Microcontroller emulator using software such as "Proteus" and MicroC Pro for programming. The focus was on understanding keypad operation, implementing PWM control, and achieving variable speed control of the DC motor through user input.

The aim of this experiment was first to become familiar with MicroC Program to build codes and create hex files for PIC operations, and to simulate the PIC microcontroller in Proteus to test codes before implementation. Second, to become familiar with keypads and implement a practical application using them for DC motor speed control using Proteus and MicroC programs.

Theory

PIC Microcontrollers

PIC microcontrollers (Programmable Interface Controllers) are versatile electronic circuits that can be programmed to perform a wide variety of tasks. These tasks can range from simple functions like acting as timers to more complex operations such as controlling automated production lines. Due to their flexibility and reliability, PIC microcontrollers are widely used in many electronic devices, including alarm systems, home appliances, robotics, embedded systems, and even mobile phones. In fact, they are found in nearly all modern electronics, making them an essential component in the world of technology and automation.

Figure 1 PIC Microcontroller

Proteus Software

Proteus is a comprehensive design tool that combines user-friendly functionality with advanced features, making it ideal for designing, testing, and laying out professional printed circuit boards (PCBs). It supports a wide range of microcontroller variants available for simulation directly from the schematic, providing a robust platform for testing embedded system designs. Proteus is especially valuable for engineers and developers working with microcontroller-based projects, as it allows for virtual testing and debugging before hardware implementation. The software supports both analog and digital circuit simulation, and its integrated environment includes tools for PCB layout, circuit analysis, and 3D visualization, streamlining the design process from concept to prototype. This combination of features makes Proteus a go-to choice for professionals looking to optimize their embedded system development workflow.

MicroC Program

MikroC Pro is a powerful and user-friendly integrated development environment (IDE) designed for creating applications for Microchip PIC microcontrollers. The software allows developers to write code in the C programming language, compile it, and generate the corresponding hexadecimal (.hex) files needed for microcontroller programming. These hex files can then be uploaded to PIC microcontrollers for execution or integrated into circuit simulations in Proteus for virtual testing. MikroC Pro provides a range of features to streamline embedded system development, including a comprehensive code editor, built-in libraries for various peripherals, and debugging tools. It is widely used for developing firmware across different industries, from simple control systems to complex embedded solutions, due to its ease of use, extensive documentation, and support for multiple PIC variants. Additionally, the software supports code optimization and efficient memory management, making it suitable for both beginner developers and seasoned engineers working on resource-constrained embedded systems.

Keypad Interface

Keypads are widely used input devices in various electronics and embedded projects. They provide a reliable method for numerical input and system control. This keypad features 16 buttons, organized in a matrix of 4 rows and 4 columns. When a button is pressed, it completes a circuit between a specific row and column, allowing current to flow. This current is detected by the microcontroller, which then identifies the exact button pressed based on the row and column intersection. The versatility and simplicity of keypads make them ideal for various applications, including security systems, user interfaces, and control panels for electronic devices. The integration of a keypad into a project enhances user interaction, allowing for easy data entry and commands to be issued quickly. Whether used in home automation or robotics, keypads remain a fundamental component in the design of intuitive and user-friendly systems.

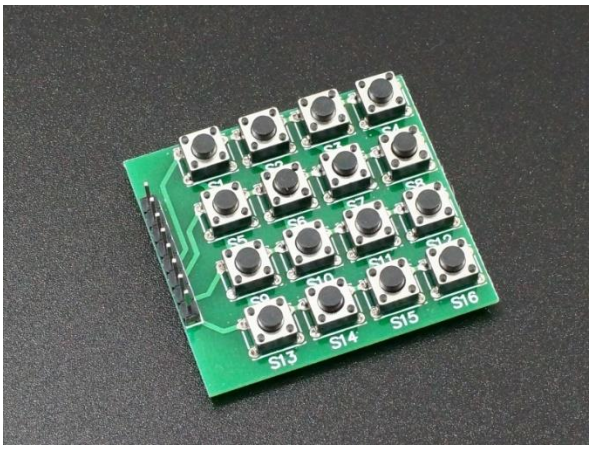

Figure 2 keypad structure

MikroC Keypad Library Functions:

Keypad_Init(): Initializes port for keypad operation

Keypad_Key_Press(): Non-blocking key read function

Keypad_Key_Click(): Blocking function for key press and release detection

PWM Control

Pulse Width Modulation (PWM) provides efficient motor speed control by varying the duty cycle of a square wave.

DC Motor Control

A DC motor controller is a system that manages the operation of a DC motor through PWM signals. The main purpose is to achieve precise speed control while maintaining system stability. The controller uses a microcontroller to generate PWM signals and process user inputs.

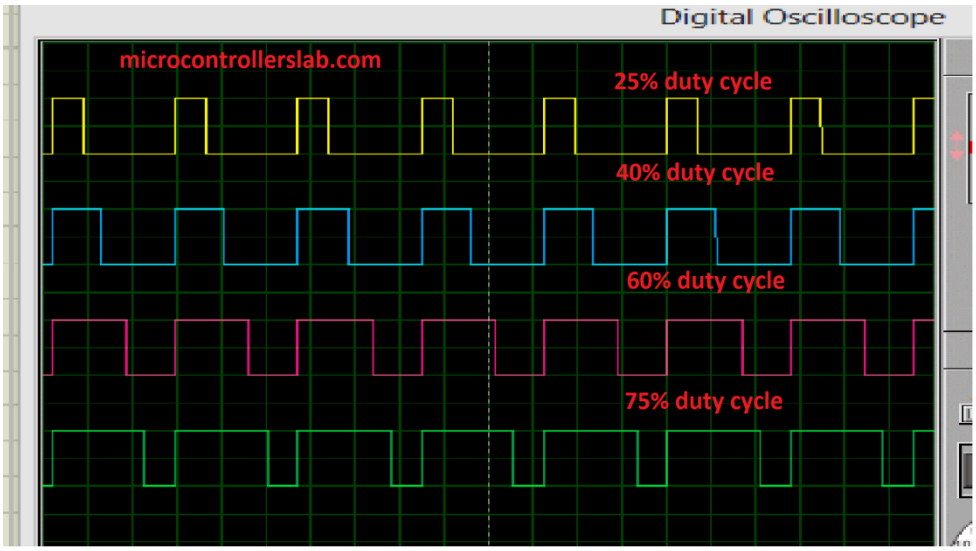

Figure 3 duty cycle change

L293D:

It’s a driver module intended for controlling two DC motors, equipped with 14 pins. The IN1 and IN2 pins dictate the rotation direction: a signal of 01 corresponds to clockwise rotation, while 10 corresponds to counterclockwise. The VS pin connects to a DC power supply with a voltage range of 12 to 36 volts.

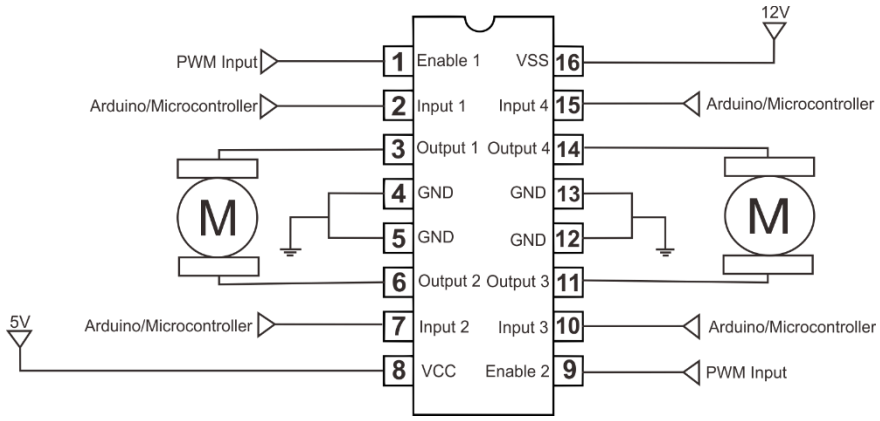

Figure 4 L293D

Procedure:

First the circuit in Figure 5 was built using Protus Software.

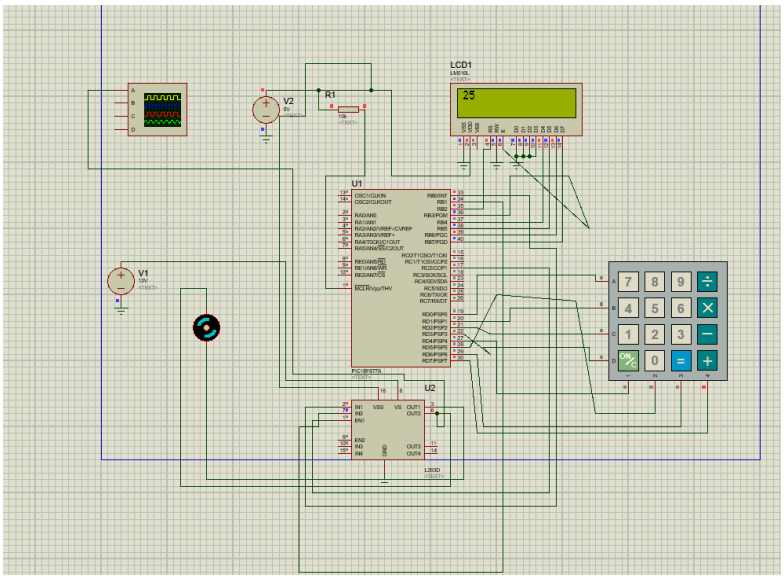

Figure 5 motor-Keypad circuit

The initialization process for the LCD and keypad involves several sequential steps:

1. **LCD Pinout Configuration**:
    - The pinout settings for the 16x2 LCD are defined in the program.
    - The pin direction for the LCD is specified in the configuration.
2. **Keypad Setup**:
    - PORTD is designated as the keypad port.
    - Four integers are declared for subsequent keypad button definitions: kpi, sum, index, and ascii.
3. **Port Configuration**:
    - Ports D and B are set as output ports using the initialization sequence.
    - PORTC is configured for the keypad.

4. **LCD Initialization**:
    - The LCD is initialized with the Lcd_Init() command.
    - The display is cleared using the Lcd_Cmd() function with the LCD_CLEAR parameter.
    - The cursor is disabled with the LCD_CURSOR_OFF command.
5. **Keypad Initialization**:
    - The keypad is initialized using the keypad_Init() function.
6. **Button Mapping**:
    - Different cases are defined for each number using ASCII codes.
7. **Final Configuration**:
    - PORTD is set to input, while PORTB and PORTC are set as output ports (PORTB for the LCD and PORTC for the PWM motor control).
    - PORTB is set to 1, which makes the motor run in a clockwise direction.
    - The PWM is started with PWM_Start(), and the duty cycle is set using PWM_Set_Duty().

Then the oscilloscope was set and obtained Figure 6 from the motor driver.

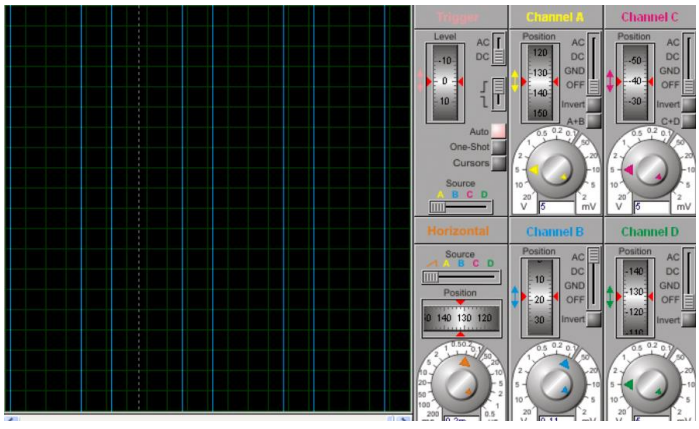

Figure 6 oscilloscope output

Conclusion

In conclusion, the implemented system showcases a successful integration of keypad input for precise motor control, coupled with reliable PWM-based speed regulation. Furthermore, the practical application of microcontroller programming underscores the project's technical depth. Overall, the experiment demonstrates the use of keypad, LCD and DC motor driver.

References

[1] https://www.microchip.com/en-us/product/pic16f887 accessed at(10/24/2024 7:12 PM)

[2] https://www.labcenter.com/ accessed at(10/24/2024 8:06 PM)

[3]https://www.mikroe.com/mikroc-pic?srsltid=AfmBOooCe4ycknf8bt7eOahMbu2qjeeG4u367_RUtSJqXzMVmV9FXL2e

accessed at(10/24/2024 8:09 PM)

[4] ENEE4104 Lab Manual.

[5] DC Motor Speed Control using Pic microcontroller - PWM method (microcontrollerslab.com) accessed at (10/24/2024 8:17 PM)

[6] Interfacing DC Motor with PIC Microcontroller using L293D - MikroC (electrosome.com)

Accessed at (10/24/2024 8:35 PM)

Appendix

Code used

```
//-------------------------------------------------- Code Starts ----------------------------------------------
---

// Lcd pinout settings

sbit LCD_RS at RB2_bit;

sbit LCD_EN at RB3_bit;

sbit LCD_D7 at RB7_bit;

sbit LCD_D6 at RB6_bit;

sbit LCD_D5 at RB5_bit;

sbit LCD_D4 at RB4_bit;

// Pin direction

sbit LCD_RS_Direction at TRISB2_bit;

sbit LCD_EN_Direction at TRISB3_bit;

sbit LCD_D7_Direction at TRISB7_bit;

sbit LCD_D6_Direction at TRISB6_bit;

sbit LCD_D5_Direction at TRISB5_bit;

sbit LCD_D4_Direction at TRISB4_bit;

char keypadPort at PORTD;

int kpi;

int sum = 0;

int index = 0;

int ascii = 0;
```

```
void main() {
TRISD = 0x00; // set all pins of port D as output
TRISB = 0x00; // set all pins of port B as output
Lcd_Init(); //Initializing LCD
Lcd_Cmd(_LCD_CLEAR);//Clear Display
Lcd_Cmd(_LCD_CURSOR_OFF);//CURSUR OFF
Keypad_Init ();//Initializing keypad
do {
 do {
 kpi =0; //reset key code variable
 do{
 kpi =Keypad_Key_Click(); //store key code in kpi variable
 }while(!kpi);
 if (sum==0) {
 Lcd_Cmd(_LCD_CLEAR); //Clear display
 Lcd_Cmd(_LCD_CURSOR_OFF); //Cursor off
 }
 switch (kpi) {
 case 1: kpi =7; ascii =55; break ; //7
 case 2: kpi =4; ascii =52; break ; //4
 case 3: kpi =1; ascii =49; break ; //1
 case 4: kpi = -1; break ; //space
 case 5: kpi =8; ascii =56; break ; //8
```

```
case 6: kpi =5; ascii =53; break ; //5
case 7: kpi =2; ascii =50; break ; //2
case 8: kpi =0; ascii =48; break ; //0
case 9: kpi =9; ascii =57; break ; //9
case 10: kpi =6; ascii =54; break ; //6
case 11: kpi =3; ascii =51; break ; //3
case 12: kpi =-1; break ; }
Lcd_Chr_CP(ascii);
if(kpi != -1 )
sum = sum*10+kpi;
}while (kpi != -1);
Lcd_Cmd(_LCD_CLEAR);//CLEAR DISPLAY
Lcd_Cmd(_LCD_CURSOR_OFF);//Cursor OFF
Lcd_Out(1,2,"working ..." );
if(sum >100)
sum =100;
index = ((sum*255)/100);
sum=0;
TRISD =0xFF; //portd as input
TRISC =0x00; //portc as output
TRISB=0x00; //portb as output
PORTB.b0=1; //runmotor in anti clock wise
PORTB.b1=0;
```

```
 PWM1_Init(1000); //Initialaize PWM1
 PWM1_Start(); //start PWM1
 PWM1_Set_Duty(index); //set current duty for PWM1
 }
while(1); }
```