A4: PWM motor control

Group members:
Priya Wilkhu
Elizabeth Sotomayor
Matthew Sharp
Bekhzodbek Moydinboyev

Project description: In this project, we will design and build a system to control the speed of a 12V DC motor using a matrix keypad and display the speed percentage on a 2x16 LCD connected via I2C. We will start by connecting the matrix keypad to a microcontroller to allow user input for speed settings. The microcontroller will read the inputs from the keypad, generate corresponding Pulse Width Modulation (PWM) signals, and send these signals to an H-bridge motor driver to control the motor's speed. We will also interface a 2x16 LCD with the microcontroller via I2C to display the current speed percentage in real-time. Finally, we will ensure the motor and the microcontroller are powered properly with dedicated power supplies and integrate all components into a cohesive system for precise motor speed control.

need to include these libraries:

- **Keypad Library** for reading matrix keypad inputs.
- **I2C Library** for communicating with the LCD.
- **PWM Configuration** using Timer_B.

Components Needed:

1. **MSP430FR2355 Microcontroller**: To control the system.
2. **Matrix Keypad (4x4 or 4x3)**: For user input.
3. **Motor Driver (L298N or L293D)**: To control the 12V DC motor.
4. **12V DC Motor**: The motor to be controlled.
5. **2x16 LCD Display with I2C Interface**: To display the motor speed percentage.
6. **Power Supply**: 12V power supply for the motor and a regulated power supply for the MSP430 (e.g., 3.3V).
7. **Breadboard, cables, resistors**

Wiring

1. **Power Supply:**
    - 12V Power Rail: Connect to L298N VCC and motor.
    - 3.3V Power Rail: Connect to MSP430FR2355 and LCD VCC.
2. **MSP430FR2355 Connections:**

    - **VCC (3.3V):** To 3.3V power rail.
    - **GND:** To ground rail.
    - **Keypad Rows:**
        - P3.0 to Row 1
        - P3.1 to Row 2
        - P3.2 to Row 3
        - P3.3 to Row 4
    - **Keypad Columns:**
        - P3.4 to Column 1 with 10kΩ pull-up resistor(Resistor between pin and + 5V is called pull-up resistor,) to 3.3V
        - P3.5 to Column 2 with 10kΩ pull-up resistor to 3.3V
        - P3.6 to Column 3 with 10kΩ pull-up resistor to 3.3V
        - P3.7 to Column 4 with 10kΩ pull-up resistor to 3.3V
    - **L298N Control:**
        - IN3 (3.3v powerrail), IN4 (GRND) to control direction of motor
        - ENB (P6.1 on MSP430) for PWM
        - **VCC:** To 12V battery
        - **GND:** To ground rail.
    - **LCD I2C:**
        - SDA (P1.6 on MSP430) with 10kΩ pull-up resistor to 3.3V
        - SCL (P1.7 on MSP430) with 10kΩ pull-up resistor to 3.3V
        - **VCC:** To 5V on MSP430.
        - **GND:** To MSP4

- **Pseudocode**

```
1. Stop the Watchdog Timer and Set up GPIO and timer configurations, initialize an I2C connection and set up the LCD.
2. Define the port and bits that the keypad will use for input, as well as an array Define an array to map the key presses to their actual values.
3. Set port 3 as input. Enable pull-up registers and set the initial output of all pins to 1.
4. Set Port 6 as output and the initial value of all pins to 0.
5. Display "Motor Speed:" on the LCD and use a loop to take key presses.
6. Loop:

    Call get_key() function and wait until a key is pressed on the keypad.

    Check the character of the pressed key.
```

If the pressed key is invalid, the entered number has more than three digits, or the entered speed is greater than 100, ignore this input and continue with the next iteration of the loop.

Once the 'A' key (which works as an Enter key) is pressed, then convert the

entered string into an integer (speed) and display the entered speed on the LCD.

Adjust the duty cycle of the timer to change the speed of the motor.

Reset the string index to start accepting a new input.

If a digit key is pressed, add it to the input string and wait for the next key press.

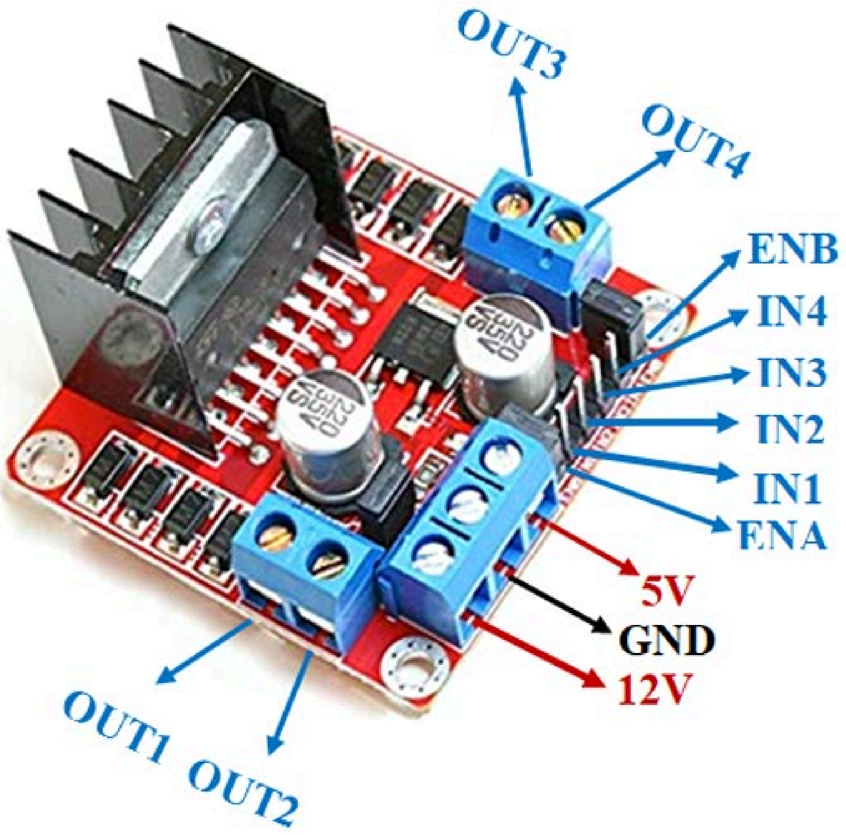
4. The get_key function:

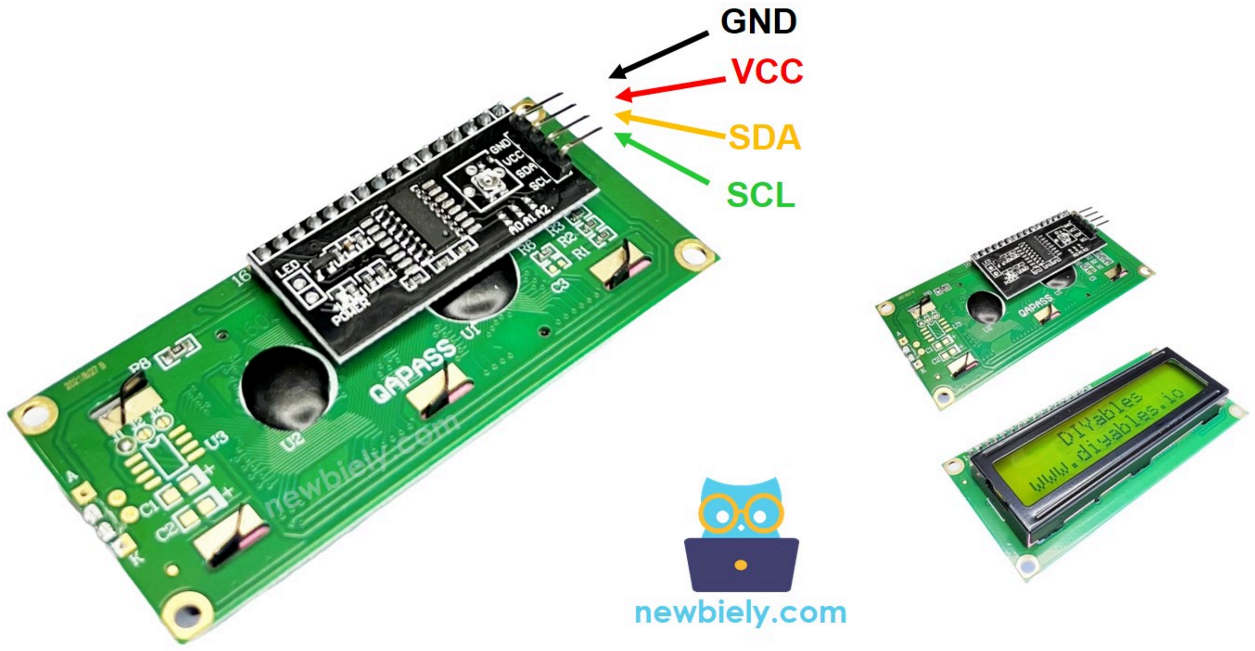
● It scans each row of the keypad for a key press.

● If a key is pressed, it returns the corresponding key value, else it returns 0.

https://youtube.com/shorts/h6xr7qya_7I?si=EIHCT3Yqx9XI4txp

Wiring Diagram:





lcd i2c

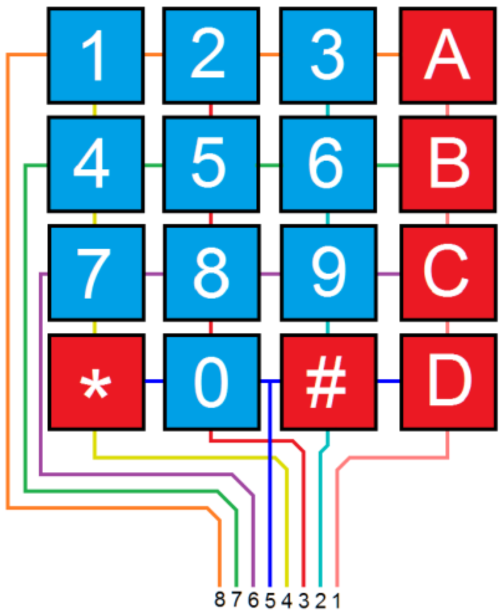
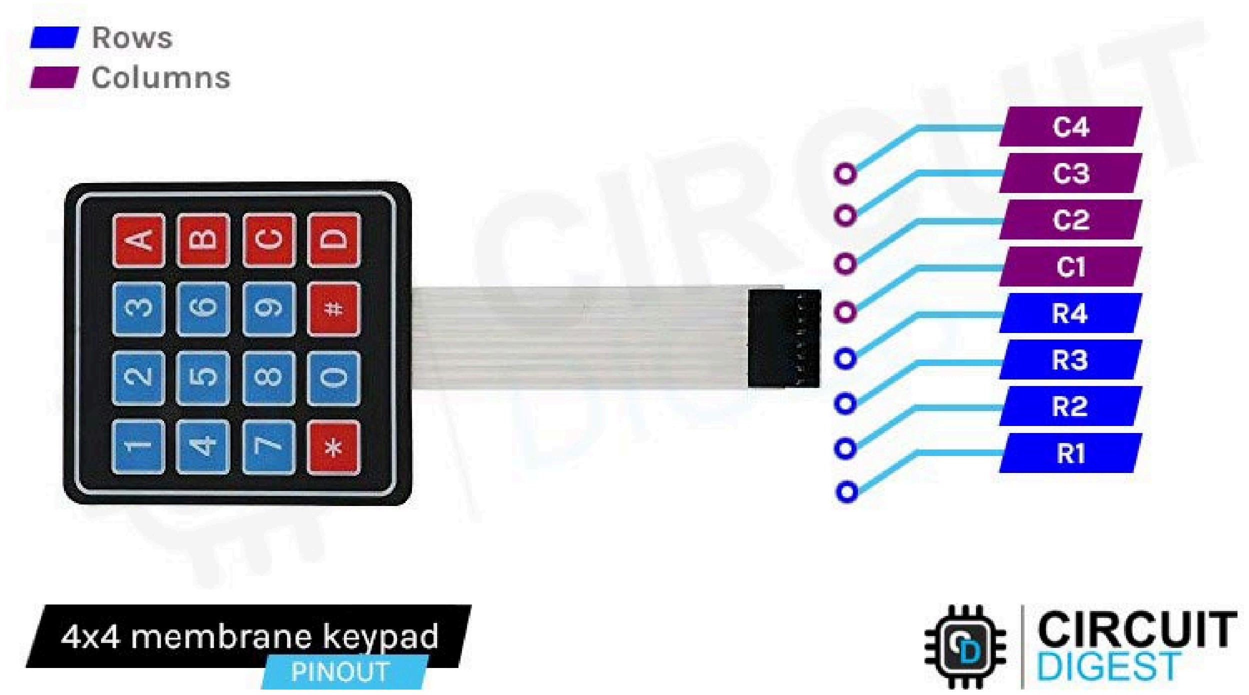


Figure 1: Matrix Keypad Connections



Video:

Code:

```
#include <msp430.h>
#include "LiquidCrystal_I2C.h"
#

#define keyport P3OUT //Keypad Port
#define COL1 (0x10 & P1IN) //P1.4
#define COL2 (0x20 & P1IN) //P1.5
#define COL3 (0x40 & P1IN)//P1.6
#define COL4 (0x80 & P1IN)//P1.7


unsigned char key_press;
unsigned char i,k,key=0;
unsigned char Key_Val[] = {'
','1','2','3','A','4','5','6','B','7','8','9','C','*','0','#','D'};


int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;   // Stop WDT

    // Configure GPIO
    P2DIR |= BIT0;              // P2.0 output from ENB
    P2SEL0 |= BIT0;             // P2.0  options select (TB3.2)

      P2OUT |= BIT1;               //set IN2 on p2.1

    // Setup Timer3_B CCR
    TB3CCR0 = 655;                  // PWM Period
    TB3CCTL2 = OUTMOD_7;            // CCR2 reset/set
    TB3CCR2 = 0;                    // CCR2 PWM duty cycle

    //Setup Timer3_B
    TB3CTL = TBSSEL_1 | MC_1 | TBCLR;   // ACLK, up mode, clear TBR

    unsigned int count=0;
    unsigned char Value=0;
    P1DIR = 0x0F; // Set P1 to input
    P1REN = 0xFF; // Set P1.0 to 1.7 Pull up Register enable
    P1OUT = 0xF0; // Set P1.0 to 1.7 Out Register.
    //P2DIR = 0xFF; // Set P2 outputs
    //P2OUT = 0x00; // Set P2 as 0's values

    // Disable the GPIO =power-on default high-impedance mode
    PM5CTL0 &= ~LOCKLPM5;

    // Initialize I2C connection and LCD
    I2C_Init(0x27);
    LCD_Setup();

    LCD_SetCursor(0, 0);
    LCD_Write ("Motor Speed:");

    char str[4]; // The input string can represent 1-3 digit integers plus
the terminating character
    int i = 0;
    while(1)
    {
        while((count = get_key())==0); //Wait until a key Pressed
        Value = Key_Val[count];

        LCD_ClearDisplay(); // Clear display from previous input
        LCD_SetCursor(0, 0);
        LCD_Write ("Motor Speed:");

        // Check if keypad input is invalid
        if(Value == '#' || Value == '*' || Value == 'B' || Value == 'C' ||
Value == 'D' || i == 4)
        {
            if(i == 4)
                i = 3; // Set string index to last character
            continue;
        }

        if(Value == 'A') // A is the enter key
        {
            str[i] = '\0'; // Set terminal character to the current index
which is the
            int speed = atoi(str);

            if(speed > 100)
            {
                i = 0;
                continue;
            }
            // Write speed value to LCD
            LCD_SetCursor(12, 0);
            LCD_Write (str);
            if(speed == 100)
                LCD_SetCursor(15, 0);
            else if(speed > 9)
                LCD_SetCursor(14, 0);
            else
                LCD_SetCursor(13, 0);

            LCD_Write ("%");

            // Set Duty Cycle
            TB3CCR2 = speed * 6;

            i = 0;
            continue;
        }

        // Insert keypad input into string to keep track of the input
        str[i] = Value;

        i++;
    }

    return 0;
}

unsigned char get_key(void)
{
    k=1;
    for(i=0;i<4;i++)
    {
        keyport = ((0x01<<i) ^ 0xff);  //Scan for a Key by sending '0' on
ROWS
        if(!COL1){  //when a key pressed numbers 1--16 will be returned
            key = k+0;
            while(!COL1);
            return key;
        }
        if(!COL2){
            key = k+1;
            while(!COL2);
            return key;
        }
        if(!COL3){
            key = k+2;
            while(!COL3);
            return key;
        }
        if(!COL4){
            key = k+3;
            while(!COL4);
            return key;
        }
        k+=4;
        keyport |= (0x01<<i);
        }
        return 0;
}
```