Lesson 16 LCD1602

Introduction

In this lesson, we will learn how to use LCD1602 to display characters and strings.

Components

- 1 * Raspberry Pi
- 1 * Breadboard
- 1 * LCD1602
- 1 * Potentiometer
- 1 * T-Extension Board
- 1 * 40-Pin GPIO Cable
- Jumper wires

Principle

LCD1602

Generally, LCD1602 has parallel ports, that is, it would control several pins at the same time. LCD1602 can be categorized into eight-port and four-port connections. If the eight-port connection is used, then all the digital ports of the SunFounder Uno board are almost completely occupied. If you want to connect more sensors, there will be no ports available. Therefore, the four-port connection is used here for better application.

LCD1602 uses the standard 16-pin port, among which:

Pin 1 (GND): connected to Ground

Pin 2 (Vcc): connected to 5V power supply

Pin 3 (Vo): used to adjust the contrast of LCD1602; the level is lowest when it's connected to a positive power supply, and highest when connected to ground (you can connect a 10K potentiometer to adjust its contrast when using LCD1602)

Pin 4 (RS): register select pin, controlling where in the LCD's memory you are writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, where the LCD's controller looks for instructions on what to do next.

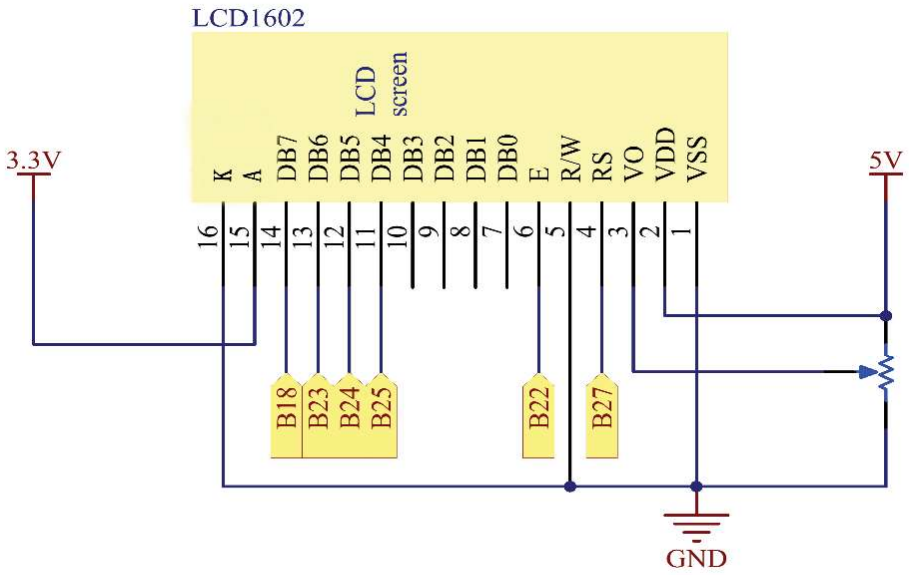
Pin 5 (R/W): to read/write signals; it reads signals when supplied with high level (1), and writes when low level (0) (in this experiment, you only need to write data to LCD1602, so just connect this pin to ground)

Pin 6 (E): An enable pin that, when low-level energy is supplied, causes the LCD module to execute relevant instructions

Pin 7 (D0-D7): pins that read and write data

A and K: controlling LCD backlight; K connects to GND, and A to 3.3V. Turn the backlight on and you can see the characters displayed clear in a dim environment

LCD1602 has two operation modes: 4-bit and 8-bit. When the IOs of the MCU are insufficient, you can choose the 4-bit mode, under which only pins D4~D7 are used. After connecting the circuit, you can operate LCD1602 by the Raspberry Pi.

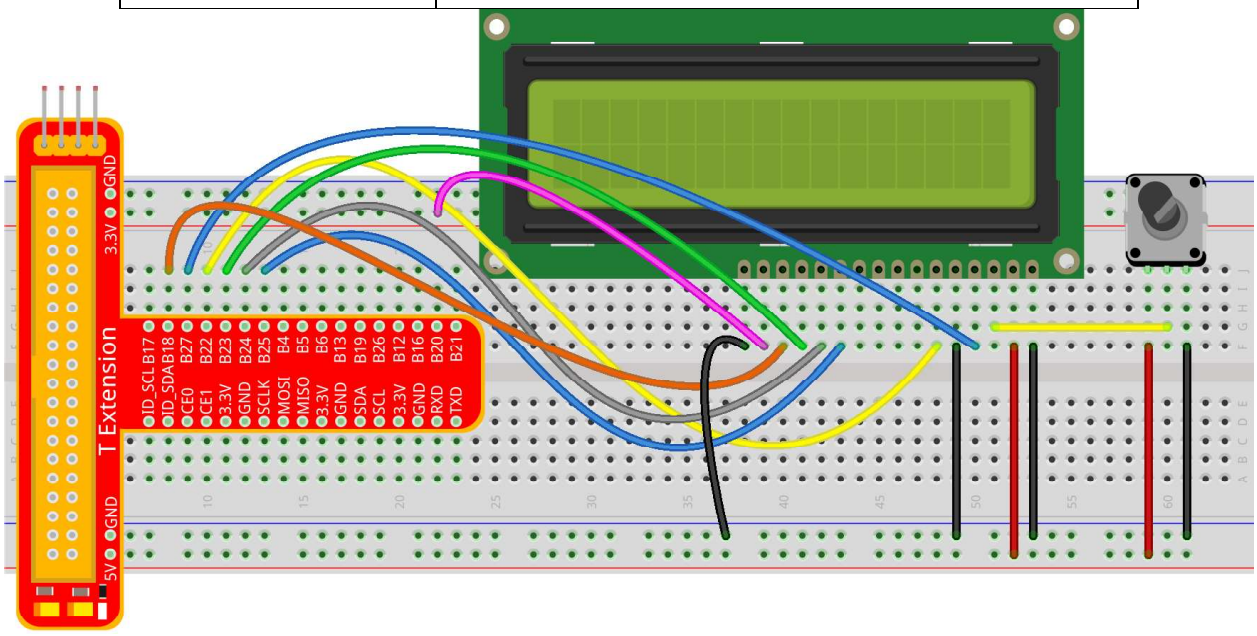


Experimental Procedures

Step 1: Build the circuit (please be sure the pins are connected correctly. Otherwise, characters will not be displayed properly):

LCD1602	T-Extension Board
VDD	5V
VSS	GND
OV	Connect to the middle pin of potentiometer
RS	B27
R/W	GND
E	B22
D0-D3	Not connected
D4	B25
D5	B24
D6	B23

D7	B18
A	3.3V
K	GND



Note: After you run the code, characters may not appear on the LCD1602. You need to adjust the contrast of the screen (the gradual change from black to white) by spinning the potentiometer clockwise or anticlockwise, until the screen displays characters clearly.

For C language users:

Step 2: Get into the folder of code

```
cd /home/pi/SunFounder_Super_Kit_V3.0_for_Raspberry_Pi/C
```

Step 3: Compile

```
make 16_lcd1602
```

Step 4: Run

```
sudo ./16_lcd1602
```

Code Explanation

```
#include <lcd.h>  // includes the lcd library, containing some functions for the LCD1602 display for convenient use
const unsigned char Buf[] = "---SUNFOUNDER---";  // An array to store the characters to be displayed on the LCD1602
const unsigned char myBuf[] = "  sunfounder.com"; // Another array to store the characters
fd = lcdInit(2,16,4, 2,3, 6,5,4,1,0,0,0,0); // Initialize the LCD display, see /usr/local/include/lcd.h
```

```
 // lcdInit(rows, cols, bits, rs, strb, d0, d1, d2, d3, d4, d5, d6, d7) - LCD1602 shows 2 rows and 16 columns. If the initialization succeeds, it will return True.
lcdClear(fd); // Clear the screen
lcdPosition(fd, 0, 0); // Locate the position of the cursor at Row 0 and Col 0 (in fact it's the first line and first column)
lcdPuts(fd, "Welcom To--->"); // Display the character "Welcom To--->"on the LCD1602
lcdPosition(fd, 0, 1);  // Place the cursor at Col 0, Row 0.
lcdPuts(fd, "  sunfounder.com");
while(1){
    lcdClear(fd);
    for(i=0; i<16; i++){  // i adds one in the loop. i means the number of columns, so i adds to 16 at most.
        lcdPosition(fd, i, 0);  // Place the cursor at the first row, and moves left to right from the first character
        lcdPutchar(fd, *(myBuf+i)); // *(myBuf+i) is a pointer that points to contents in the myBuf[] array, and output the pointed data to lcd
        delay(100);
    }
    for(i=0;i<sizeof(Buf)-1;i++){
        lcdPosition(fd, i, 1);  // Place the cursor at the second row, moves from the first character
        lcdPutchar(fd, *(Buf+i));  // A pointer that points to data in the Buf[] array; output it to lcd
        delay(200);
    }
    sleep(0.5);
}
```

For Python users

Step 2: Get into the folder of code

```
cd /home/pi/SunFounder_Super_Kit_V3.0_for_Raspberry_Pi/Python
```

Step 3: Run

```
sudo python 16_lcd1602.py
```

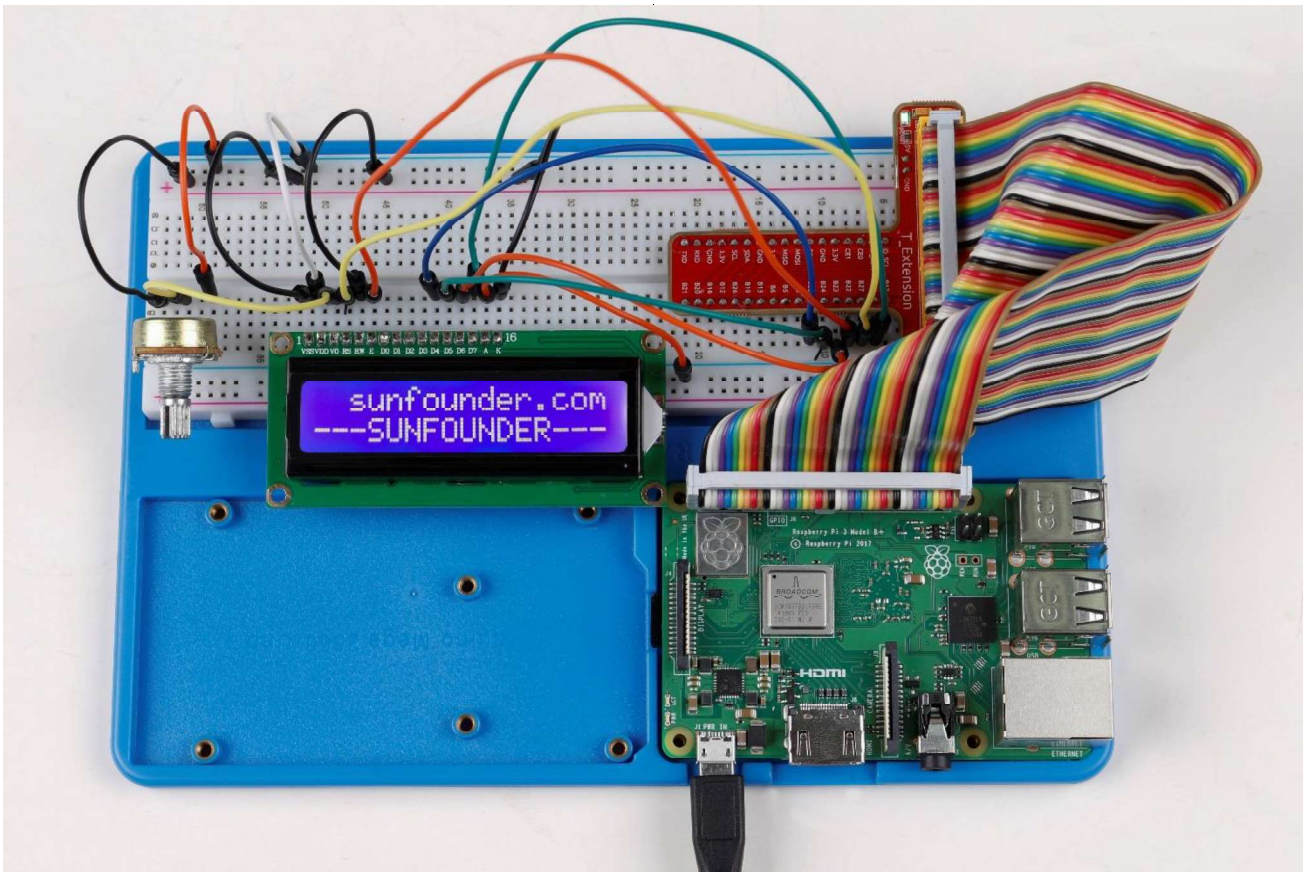
Code Explanation

```
    class LCD:  # Write an LCD class
        def __init__(self, pin_rs=27, pin_e=22, pins_db=[25, 24, 23, 18], GPIO = None); # Initialization function for the class, run when an object is created of the class. A
```

```
parameter needs to be transferred to the object when it's created; otherwise, the
default value in __init__ will be assigned.
            self.used_gpio = self.pins_db[:] # Note down the used gpio to easily clear
IO setting after the stop. pins_db[:] writes all in the pins_db list to the used_gpio
list; if here use used_gpio = self.pins_db, it means used_gpio call pins_db, in other
words, any change of pins_db will affect used_gpio.
            self.used_gpio.append(pin_e)
            self.used_gpio.append(pin_rs)
            self.write4bits(0x33) # initialization
            self.write4bits(0x32) # initialization
            self.write4bits(0x28) # 2 line 5x7 matrix
            self.write4bits(0x0C) # turn cursor off 0x0E to enable cursor
            self.write4bits(0x06) # shift cursor right
            """ Initialize to default text direction (for romance languages) """
            self.displaymode =  self.LCD_ENTRYLEFT | self.LCD_ENTRYSHIFTDECREMENT
            self.write4bits(self.LCD_ENTRYMODESET | self.displaymode) #  Set the entry
mode
        def begin(self, cols, lines):   # Start the LCD
        def setCursor(self, col, row):  # Set the cursor location
        def message(self, text):  # Send strings to the LCD. The new line wraps to the
second line
        def destroy(self):  # Clean up the used gpio

    lcd = LCD(0, 2)  # Create an lcd object
    lcd.clear()      # Clear the LCD display
    for i in range(0, len(line0)):  # i adds 1 each time within the length of the
character line0
        lcd.setCursor(i, 0)    # Locate the cursor at character No. i, Row 0
        lcd.message(line0[i])  # Display the character on the screen
        sleep(0.1)
    for i in range(0, len(line1)): # i adds 1 each time within the length of the
character line0
        lcd.setCursor(i, 1)    # Locate the cursor at character No. i, Row 1
        lcd.message(line1[i])  # Display the character on the LCD
```

You should see two lines of characters displayed on the LCD1602: " **Welcome to --->** " ," **sunfounder.com** " and "**---SUNFOUNDER---** ".



Further Exploration

In this experiment, the LCD1602 is driven in the 4-bit mode. You can try programming by yourself to drive it in the 8-bit mode.