## LCD1602 Display Experiment

## Introduction

**LCD1602** is a character **LCD** module specially designed for displaying letters, Numbers and symbols. It is extensively used in industry, such as electronic clocks, temperature displays. The character **LCD** on the market is mostly according to the **HD44780** character **LCD** chip, the control principle is exactly the same. "**1602**" means **2** lines and **16** characters per line. The **LCD1602** display with the adapter board use the **IIC** interface, which saves a lot of **I/O** ports. The **1602Liquid Crystal Display** (hereinafter referred to as the **1602 LCD**) is a common character **LIQUID Crystal Display**, so named because it may be able to display **16\*2** characters. Usually the **1602LCD** we use is integrated with the word library chip, through the **API** provided by **LiquidCrystal** class, we may be able to easily use the **1602LCD** to display English letters and some symbols. Before using the **1602 LCD**, we need to connect it to **Raspberry Pi**.

In the kit, we may be able to make the use of **LCD1602** easier by using **IIC LCD1602** module to integrate the **IIC I/O** extension chip **PCA8574**. **Raspberry Pi** may be used to control **LCD 1602** display via two-wire **IIC** bus (serial clock line **SCL**, serial data line **SDA**). It not only simplifies the circuit, but also saves **I/O** port, enabling **Raspberry Pi** to achieve more functions. The contrast of the **LCD** display may be able to also be adjusted via the potentiometer on the module. You may be able to also set the address: **0x20-0x27** by setting the jumper, which enables **Raspberry Pi** to control multiple **LCD 1602**.

A blue potentiometer may be seen on the back of the module. It may be rotated to adjust the contrast of the **1602 LCD**. **GND, VCC, SDA** and **SCL** (**DATA** line and clock line of **IIC** communication for **SDA** and **SCL** respectively) are the physical diagram of **LCD1602**:

**Note**: If the LCD light is too dim, you may be able to adjust the blue variable resistor on the back of the LCD (note: connect the wire jumper on the back)



## Experimental Principle

Connect the **Raspberry Pi** main control board with the serial **LCD1602** screen and use **I2C** communication to control the **LCD1602** display characters.

## Experimental Purpose

The **LCD1602** display is controlled by the **Raspberry Pi** master board.

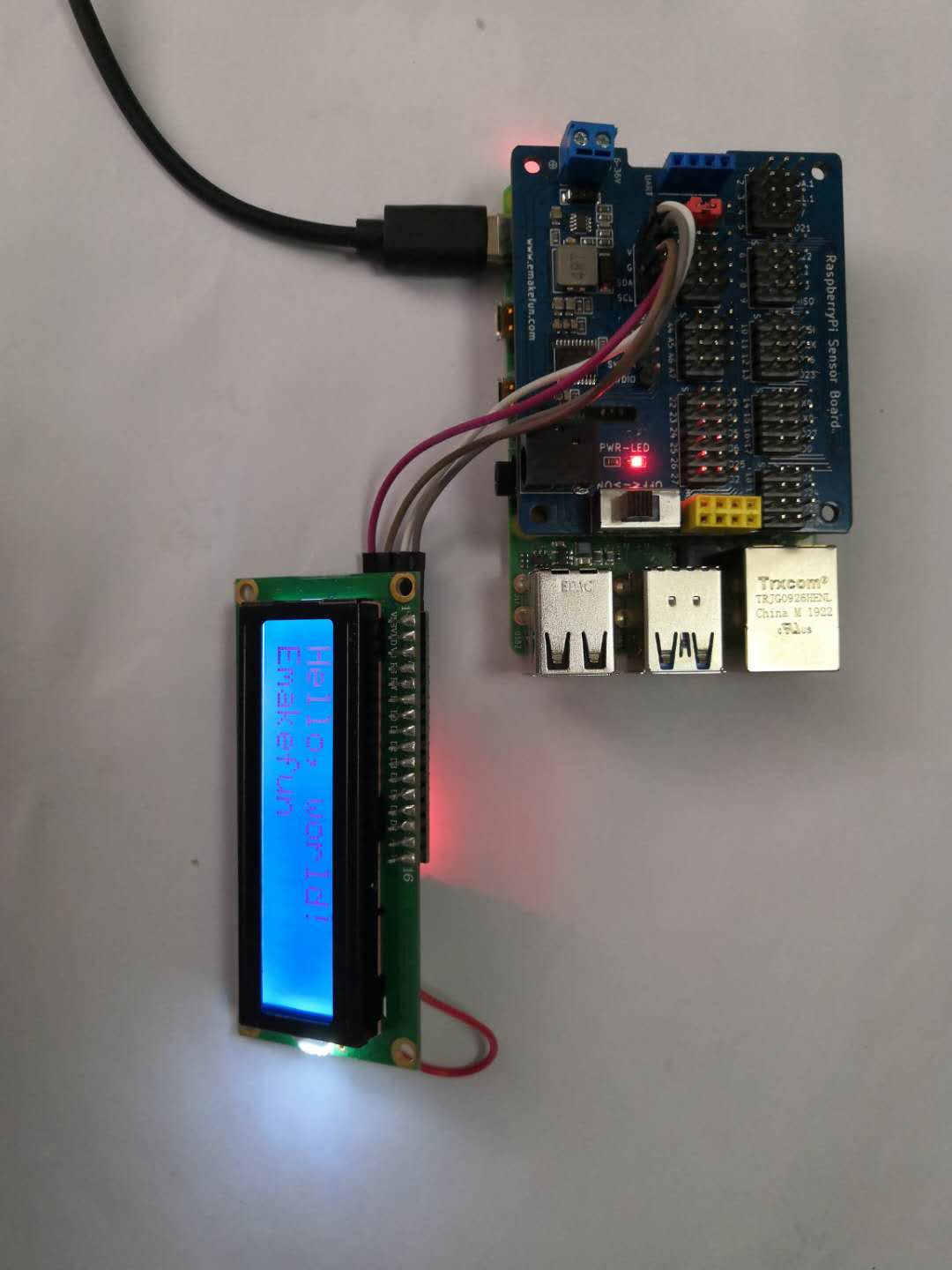
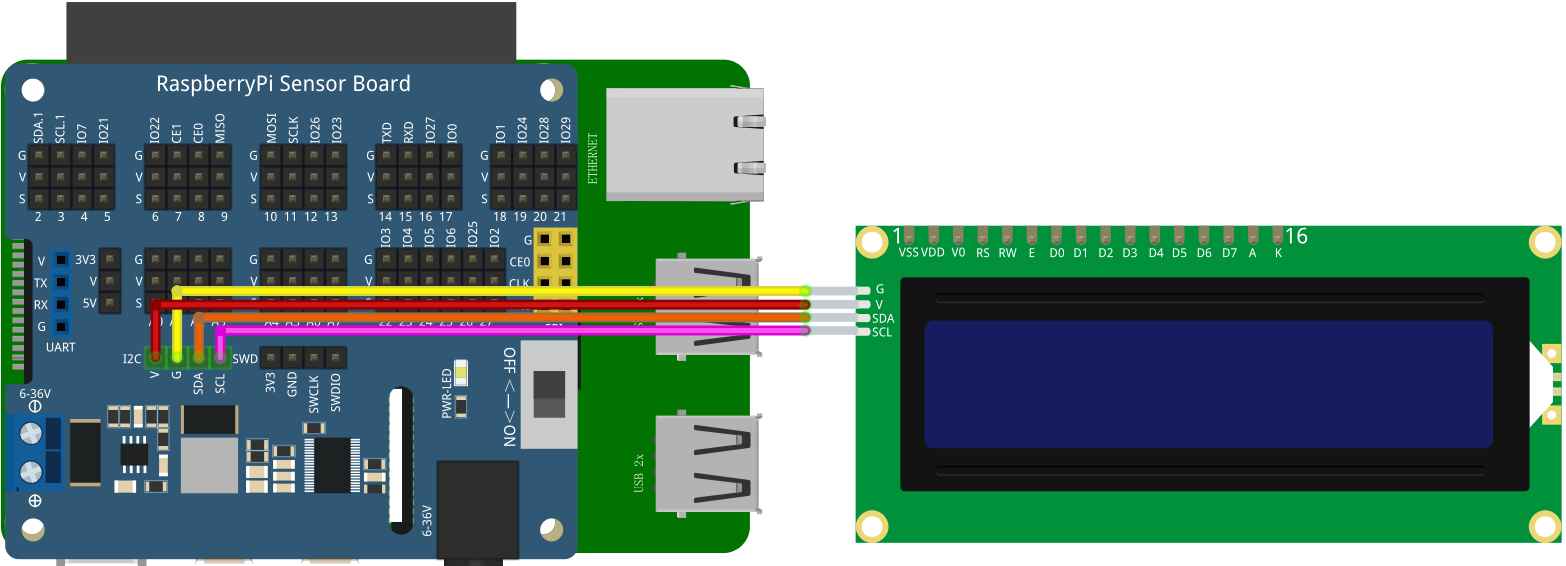
## Component List

* Raspberry Pi main board
* Raspberry Pi expansion board
* Breadboard
* Cable
* LCD1602 display with adapter board
* Several jumper wires

## Wiring

First of, we need to solder the adapter board on the LCD display screen.

|  |  |
| --- | --- |
| **LCD1602 Module** | Raspberry Pi |
| GND | GND |
| VCC | VCC |
| SDA | SDA |
| SCL | SCL |



### C++ program

|  |
| --- |
| #include <stdio.h>  #include <wiringPi.h>  #include <wiringPiI2C.h>  #include <string.h>  #include <stdlib.h>  #include "LiquidCrystal\_I2C.h"  int main()  {  init();  delay(100);  wiringPiSetup();  while(1)  {  write(0, 0,"Hello, world!");  write(0, 1,"Emakefun");  }  } |

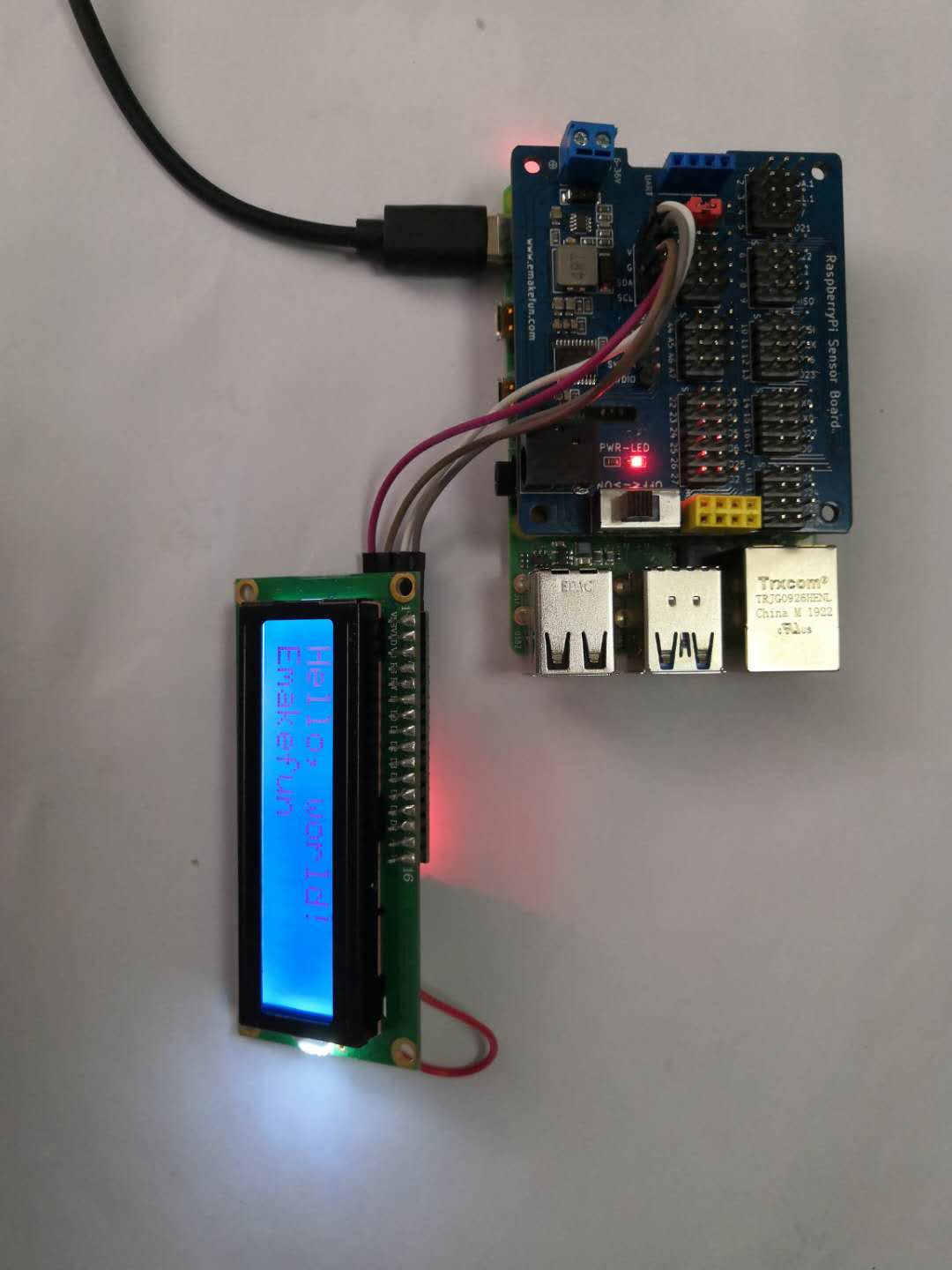
## Python partial program

|  |
| --- |
| import time  import smbus  import logx  import logging  BUS **=** smbus**.**SMBus**(**1**)**  LCD\_ADDR **=** 0x27  BLEN **=** 1 #turn on**/**off background light  def turn\_light**(**key**):**  global BLEN  BLEN **=** key  **if** key **==**1 **:**  BUS**.**write\_byte**(**LCD\_ADDR**,** 0x08**)**  logging**.**info**(**'LCD executed turn on BLight'**)**  **else:**  BUS**.**write\_byte**(**LCD\_ADDR**,** 0x00**)**  logging**.**info**(**'LCD executed turn off BLight'**)**  def write\_word**(**addr**,** data**):**  global BLEN  temp **=** data  **if** BLEN **==** 1**:**  temp **|=** 0x08  **else:**  temp **&=** 0xF7  BUS**.**write\_byte**(**addr**,** temp**)**  def send\_command**(**comm**):**  # Send bit7-4 firstly  buf **=** comm **&** 0xF0  buf **|=** 0x04 # RS **=** 0**,** RW **=** 0**,** EN **=** 1  write\_word**(**LCD\_ADDR**,** buf**)**  time**.**sleep**(**0.002**)**  buf **&=** 0xFB # Make EN **=** 0  write\_word**(**LCD\_ADDR**,** buf**)**    # Send bit3-0 secondly  buf **=** **(**comm **&** 0x0F**)** **<<** 4  buf **|=** 0x04 # RS **=** 0**,** RW **=** 0**,** EN **=** 1  write\_word**(**LCD\_ADDR**,** buf**)**  time**.**sleep**(**0.002**)**  buf **&=** 0xFB # Make EN **=** 0  write\_word**(**LCD\_ADDR**,** buf**)**  def send\_data**(**data**):**  # Send bit7-4 firstly  buf **=** data **&** 0xF0  buf **|=** 0x05 # RS **=** 1**,** RW **=** 0**,** EN **=** 1  write\_word**(**LCD\_ADDR**,** buf**)**  time**.**sleep**(**0.002**)**  buf **&=** 0xFB # Make EN **=** 0  write\_word**(**LCD\_ADDR**,** buf**)**    # Send bit3-0 secondly  buf **=** **(**data **&** 0x0F**)** **<<** 4  buf **|=** 0x05 # RS **=** 1**,** RW **=** 0**,** EN **=** 1  write\_word**(**LCD\_ADDR**,** buf**)**  time**.**sleep**(**0.002**)**  buf **&=** 0xFB # Make EN **=** 0  write\_word**(**LCD\_ADDR**,** buf**)**  def init\_lcd**():**  try**:**  send\_command**(**0x33**)** # Must initialize to 8**-**line mode at first  time**.**sleep**(**0.005**)**  send\_command**(**0x32**)** # Then initialize to 4**-**line mode  time**.**sleep**(**0.005**)**  send\_command**(**0x28**)** # 2 Lines **&** 5**\***7 dots  time**.**sleep**(**0.005**)**  send\_command**(**0x0C**)** # Enable display without cursor  time**.**sleep**(**0.005**)**  send\_command**(**0x01**)** # Clear Screen  logging**.**info**(**'LCD init over'**)**  BUS**.**write\_byte**(**LCD\_ADDR**,** 0x08**)**  logging**.**info**(**'LCD turning on BLight'**)**  except**:**  **return** False  **else:**  **return** True  def clear\_lcd**():**  send\_command**(**0x01**)** # Clear Screen  def print\_lcd**(**x**,** y**,** str**):**  **if** x **<** 0**:**  x **=** 0  **if** x **>** 15**:**  x **=** 15  **if** y **<**0**:**  y **=** 0  **if** y **>** 1**:**  y **=** 1  # Move cursor  addr **=** 0x80 **+** 0x40 **\*** y **+** x  send\_command**(**addr**)**    **for** chr in str**:**  send\_data**(**ord**(**chr**))**  **if** \_\_name\_\_ **==** '\_\_main\_\_'**:**  init\_lcd**()**  print\_lcd**(**0**,** 0**,** 'Hello, world!'**)**  print\_lcd**(**0**,** 1**,** 'Emakefun'**)** |

## Java program

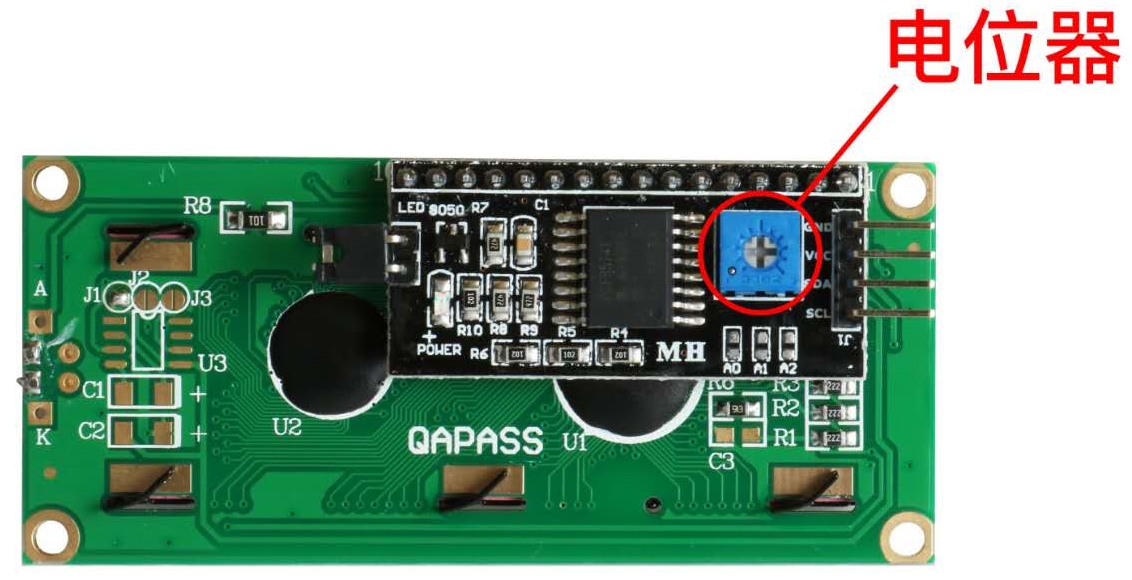
|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  **import** com**.**pi4j**.**io**.**i2c**.**I2CBus**;**  **import** com**.**pi4j**.**io**.**i2c**.**I2CDevice**;**  **import** com**.**pi4j**.**io**.**i2c**.**I2CFactory**;**  /\*\*  \*  \* **@author** user  \*/  public class LCD1602 **{**  /\*\*  \* **@param** args the command line arguments  \*/  public static void main**(**String**[]** args**)** **{**  I2CDevice \_device **=** **null;**  I2CLCD \_lcd **=** **null;**  **try** **{**  I2CBus bus **=** I2CFactory**.**getInstance**(**I2CBus**.**BUS\_1**);**  \_device **=** bus**.**getDevice**(**0x27**);**  \_lcd **=** **new** I2CLCD**(**\_device**);**  \_lcd**.**init**();**  \_lcd**.**backlight**(true);**  \_lcd**.**display\_string\_pos**(**"Hello, world!"**,** 1**,** 2**);**  **}** **catch** **(**Exception ex**)** **{**  System**.**out**.**println**(**ex**.**toString**());**  **}**  **}**  **}** |

## Experimental Effect



## LCD1602 shows common problems and solutions

1) If the backlight is on after uploading the program while the characters are not displayed, try to adjust the potentiometer behind the adapter board to adjust the brightness and display.



2) If the backlight lights up after uploading the program while only part of the characters are displayed, this is caused by the different versions of the chips used. Some chips are **PCF8574** chips, while others are **PCF8574AT** chips, so the interface address is different. The default address of **PCF8574** is **0x27**.