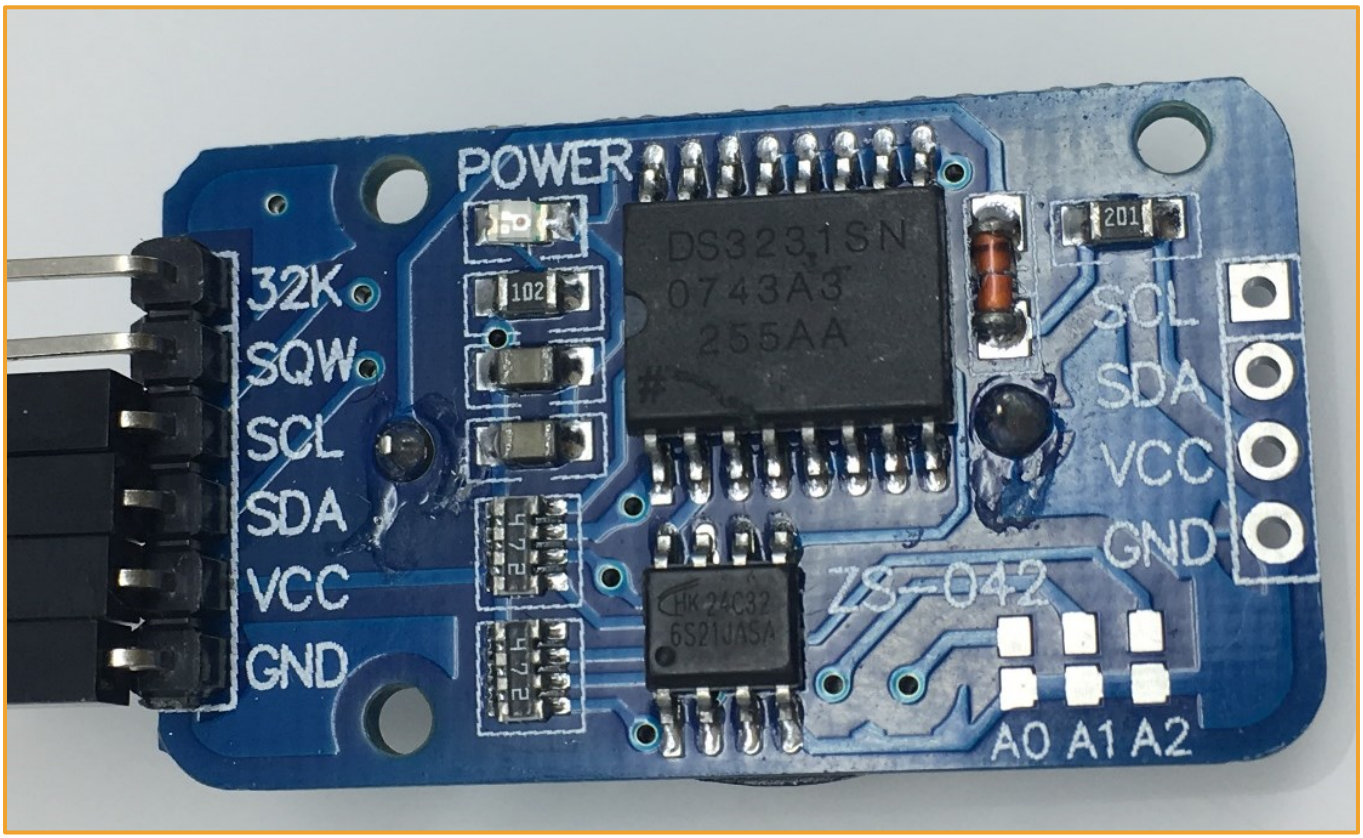
DS3231 Clock Module

Introduction

There are many popular serial clock circuits today, such as DS3231, DS1302, DS1307, PCF8485, etc. They are widely used due to their simple interfaces, low cost, and ease of use. In this lesson, we will use DS3201 real-time clock module as shown below:



1, The I2C interface;

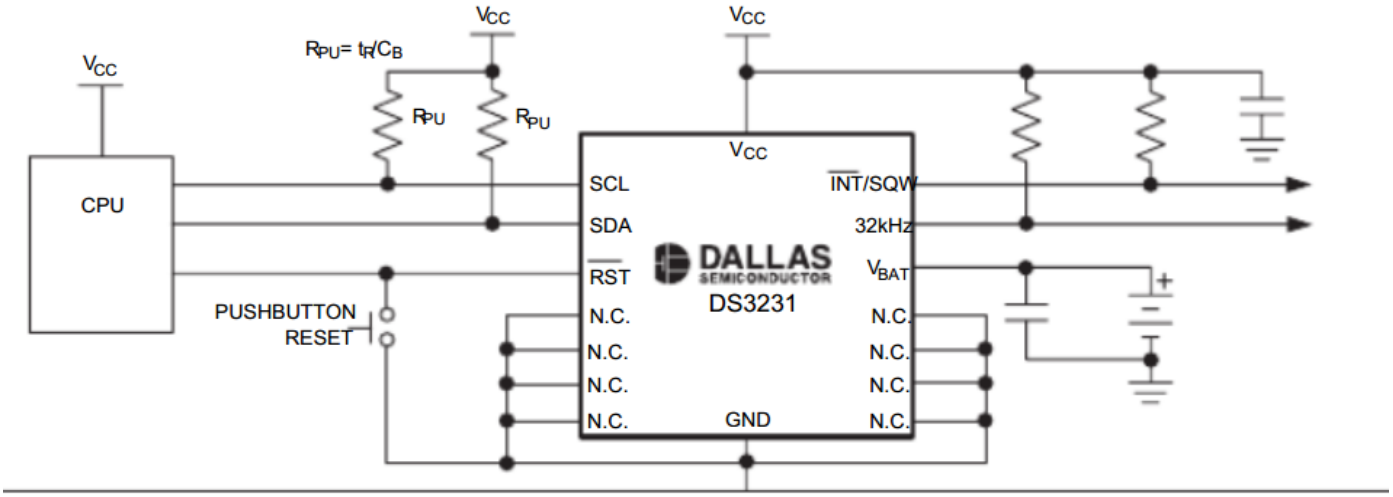
2, Real-time clock generates second, minute, hour, week, date, month, and year timer and provides the leap year compensation, expiration date is 2100;

3, Two calendar clock;

4, A digital temperature sensor is integrated internally (update every 64 seconds)

5, Programmable square wave output.

Below is DS3231 typical applied circuit. We can see that, in the figure, DS3231 hardly requires external components.



DS3231 Clock Chip Structure

The main components of DS3231 contains 8 modules, divided into 4 functional groups: TCXO, power control, button reset and RTC.

1 32 kHz TCXO

TCXO includes temperature sensor, oscillator and logic controller. The controller reads output from temperature sensor, using look-up table method to determine the required capacitance and the AGE register aging correction, then setting the capacitance and selecting register. Only loading the new changed value of AGE register when the temperature changes or the temperature conversion has been completed. the temperature value will be read every 64s when VCC powers up for the first time.

2 DS3231 internal registers and function

DS3231 register address is 00h ~ 12h which respectively store second, minutes, hour, week, date and alarm information. In multi-byte access period, if the address reaches the end of the RAM space 12h, winding will happen, at this point positioning to the starting position 00h unit. The time and calendar information were set up and initialized by reading the corresponding registers. The user assistance buffer is used to prevent potential errors when the internal registers update. While reading time and calendar registers, the user buffer synchronizes with the internal registers under the condition of any START or registers' pointers backing to zero. Time information is read from auxiliary registers and the clock still keeps running, so, during the read operation, rereading registers can be avoided when the main registers update. Taking the control register (address is 0EH) as an example, it can control real-time clock, alarm clock and the square wave output. The various bits are defined as follows in the table.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
$\overline{\text{EOSC}}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE
0	0	0	1	1	1	0	0

3 DS3231 power control

The power control function is provided by a temperature compensated voltage reference (VPF) and a comparator circuit that monitors the VCC level. When VCC is higher than VPF, the DS3231 is powered by VCC, when VCC is lower than VPF but higher than VBAT, the DS3231 is powered by VCC; when VCC is lower than VPF and VBAT, the DS3231 is powered by VBAT. To protect the battery, the oscillator does not start when the VBAT is first applied to the device, unless the VCC is loaded, or a valid I2C address is written to the device. Typical oscillator start-up time is less than 1 s. After the VCC is powered up or a valid I2C address is written about 2s, the device will measure the primary temperature and calibrate the oscillator using the calculated correction values. Once the oscillator is running, it will keep working state as long as the power supply (VCC or VBAT) is valid.The device performs a temperature measurement every 64s and calibrates the oscillator frequency.

4 DS3231 clock and calendar RTC

Clock and calendar information can be obtained by reading the appropriate register bytes. Setting or initializing clock and calendar data by writing appropriate register bytes. The contents of the clock and calendar registers are in two-decimal (BCD) format. DS3231 runs in 12 hour or 24 hour mode. The sixth bit of the hour register is defined as a mode select bit of either 12 or 24 hours. When this bit is high, selecting the 12 hour mode. Under 12 hour mode, the 5 bit is AM / PM indicating bit, shows PM when at high level.

5 DS3231 reset button

DS3231 has a push button switch connected to the RST output pin. If the DS3231 is not in the reset cycle, it will continue to monitor the falling edge of the RST signal. If an edge conversion is detected, the DS3231 complete the switch debounce by pulling RST low. After the timer ends timing, DS3231 continues to monitor RST signal. If the signal remains low, the DS3231 continuously monitors the signal line to detect the rising edge. Once the button is released, DS3231 forces RST to be at low and hold tRST. RST can also be used to indicate power failure alarm. When the VCC is lower than VPF, an internal power failure alarm signal is generated and the RST pin is forced to pull low. When VCC returns to higher than

VPF. The RST maintains a low level about 250ms (tREC) so that the power supply is stable. If the oscillator does not work when the VCC is loading, the tREC will be skipped and RST immediately becomes high.

6 DS3231 alarm clock and alarm function

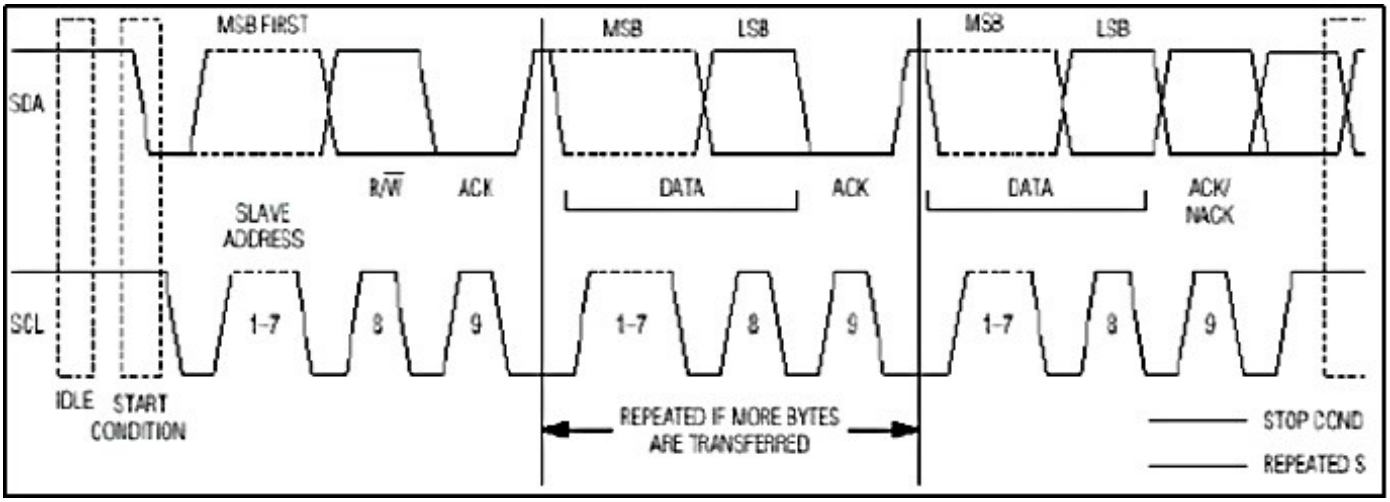
DS3231 contains 2 timer / date alarm clock. Alarm clock 1 can be set by writing to register 07h ~ 0Ah. Alarm clock 2 can be set by writing to register 0Bh ~ 0Dh. Alarm clock can be programmed (through the controlling of register alarm clock enable bit and INTCN bit), so as to trigger INT / SQW output. The seventh bit of each timer / date alarm clock register is the mask bit. When the mask bit of each alarm clock is 0, the alarm is given only when the value of the timer register is matched with the corresponding value stored in the time / date alarm clock register. The alarm clock can also be programmed as a second, minute, hour, week or date repeating alarm. When the RTC register value is matched with the set value of the alarm clock register, the corresponding alarm clock flag bit A1F or A2F is set to logic 1. If the corresponding alarm clock interrupt enable bit A1IE or A2IE is also set to logic 1, and the INTCN position is logic 1, the alarm clock condition will trigger the INT / SQW signal. RTC matches the time and date registers when they are updated every second.

7 DS3231 I2C bus timing, data exchange and its format and programming considerations

DS3231 as a slave device on the I2C bus. You can access after executing the START command and verifying the device address. The registers can then be accessed until a STOP command is executed. All transmission on the I2C bus address packet length is 9, which includes 7 address bits, 1 R / W control bit and 1 acknowledge bit ACK, if R / W is 1, while performing a read operation; if R / W is 0, then performing a write operation. After addressing from the machine, a response must be made in ninth SCL (ACK) cycle by pulling the SDA, if the machine is busy or unable to respond to the host, SDA should be maintained high in the ACK period. The host then sends a STOP state or REP START state to start sending again. The address packet includes the slave address and the READ or WRITE bits called the SLA+R or SLA+W. Address byte MSB is sent first. All 1111xxxx addresses are reserved for future use. All packets transmitted on the I2C bus are 9 bits long, including 8 data bits and 1 acknowledge bits. In data transmission, the host generates a clock and START with the STOP state, while the receiver receives the response. The SCL response is achieved by pulling the ACK low in ninth SDA cycles. If the receiver pulls high SDA, then

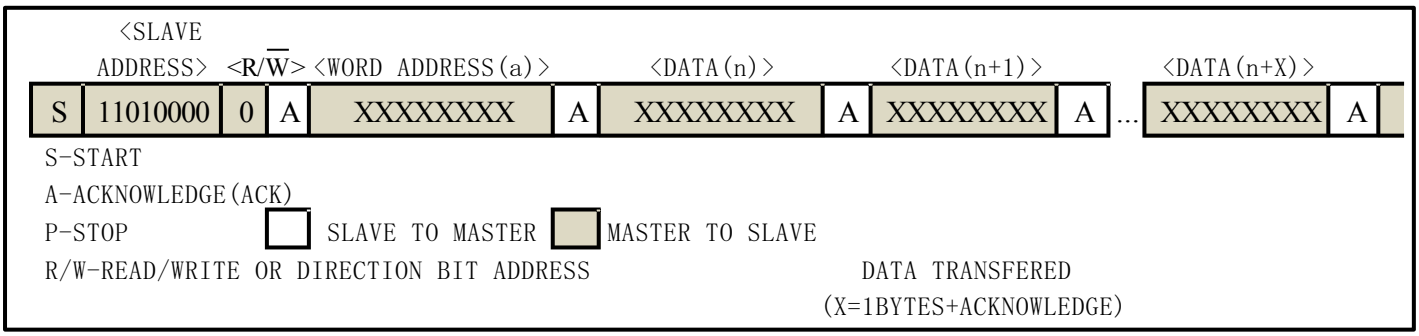
sends the NACK signal. If the receiver is unable to receive more data for some reason, the NACK signal should be sent out after the last data byte to tell the sender to stop sending the MSB.

The following figure are DS3231 and MCUI 2C bus data switching the timing sequences:

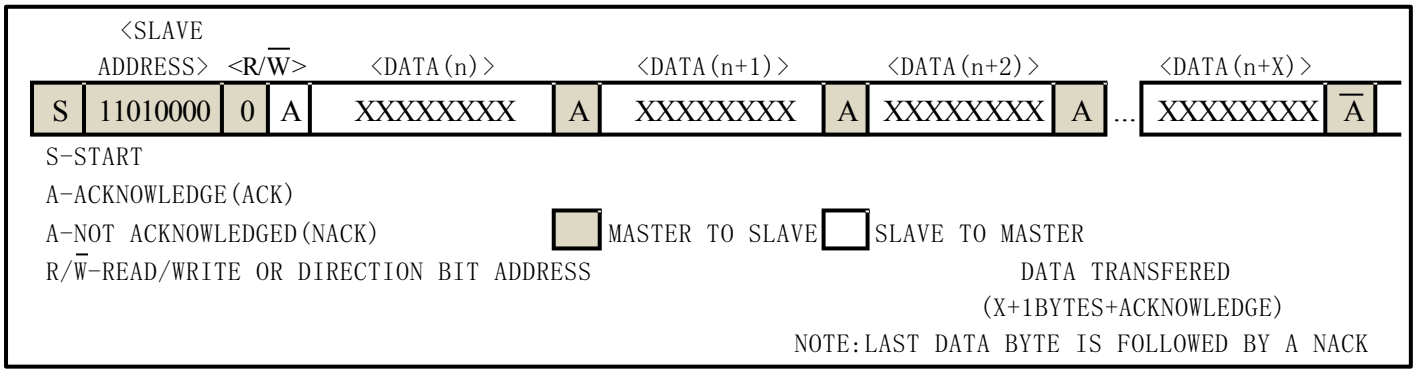


DS3231 exchanges data with the outside through two-way data line SDA and clock line SCL, from its timing relationship we can be see, DS3231 has two modes of operation:

Write operation: the data on SDA data line write N bytes of data in turn according to the specified RAM address (Word Address). The master device first transmits the address byte from the slave device, followed by a series of data bytes. The slave device returns an acknowledge bit ACK from the device whenever a byte is received . The format is shown below.



Read operation: Reading N bytes of data in turn according to the specified RAM addresses, the master device first transmits the address byte from the slave device. The slave device returns an acknowledge bit, followed by a series of data bytes. The master device returns an acknowledge bit after all bytes except the last one. After receiving the last byte, returning a "non acknowledge bit" NACK. The format is shown below.



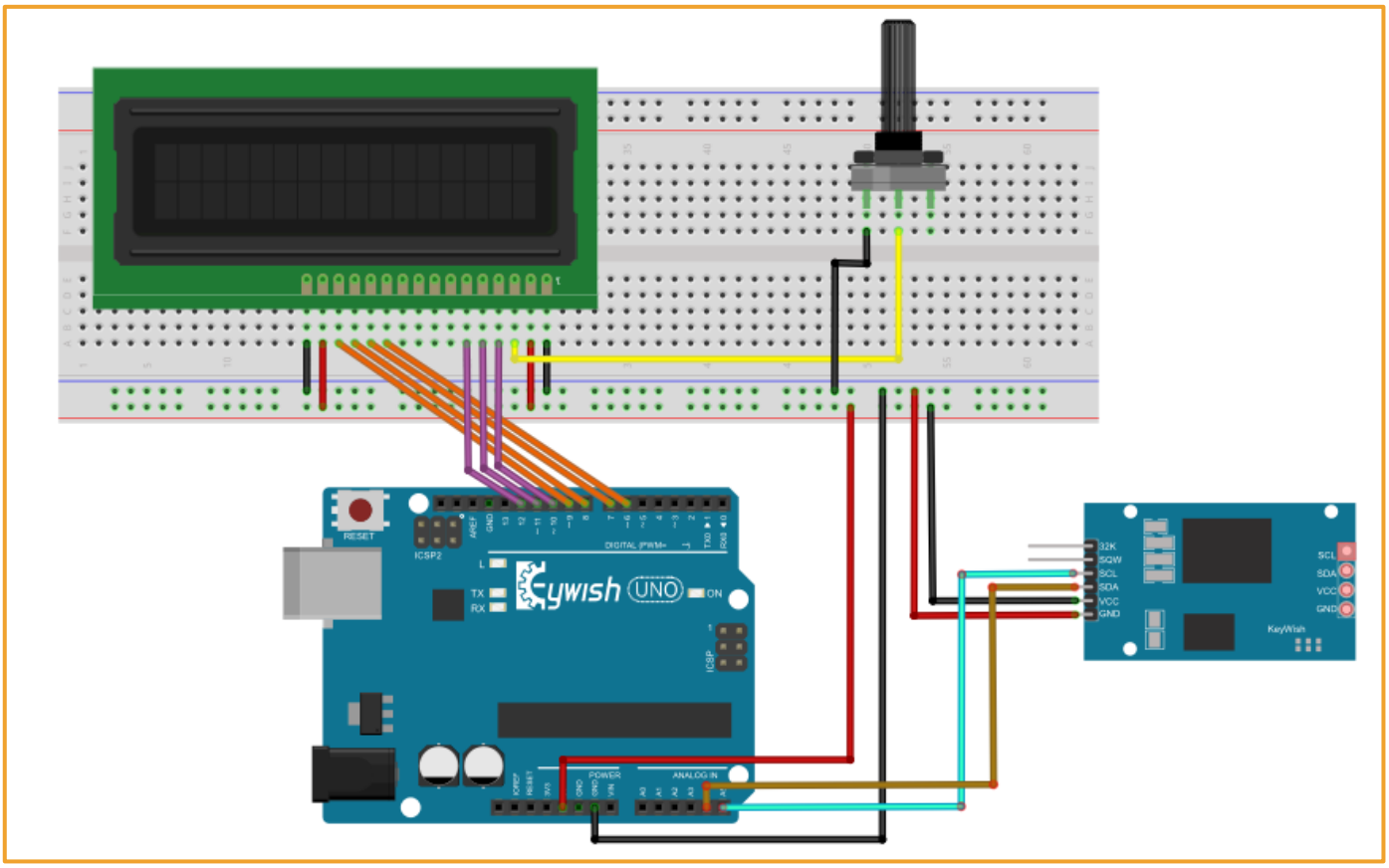
Component List

- Keywish Arduino UNO R3 mainboard
- Breadboard
- USB cable
- 1602LCD*1
- DS3231 clock module *1
- Sliding rheostat *1
- Some wires

Wiring of Circuit

arduino Uno R3	lcd1602
5	11(DB4)
4	12(DB5)
3	13(DB6)
2	14(DB7)
10	4(RS)
11	5(RW)
12	6(E)

arduino Uno R3	DS3231
A4	3(SDA)
A5	4(SCA)
5V	5(VCC)
GND	6(GND)


RESET
ICSP2
TX
RX
ON
ICSP
POWER
ANALOG IN
32K
SQW
SCL
SDA
VCC
GND

Code

```
#include "DS3231.h"
#include <Wire.h>
#include<LiquidCrystal.h>

#define    DB4    6    // lcd1602 DB4
#define    DB5    7    // lcd1602 DB5
#define    DB6    8    // lcd1602 DB6
#define    DB7    9    // lcd1602 DB7

#define    LCD1602_RS    10
#define    LCD1602_RW    11
#define    LCD1602_E     12

DS3231 Clock;

bool Century=false;
bool h12;
bool PM;

LiquidCrystal lcd(LCD1602_RS,LCD1602_RW,LCD1602_E,DB4,DB5,DB6,DB7);
// lcd init
char lcd_dis_str[2][16];

void setup() {
    lcd.begin(16,2);
    Wire.begin();
    Clock.setSecond(50);//Set the second
    Clock.setMinute(34);//Set the minute
    Clock.setHour(14);  //Set the hour
    Clock.setDoW(6);    //Set the day of the week
    Clock.setDate(11);  //Set the date of the month
```

```
    Clock.setMonth(3);  //Set the month of the year
    Clock.setYear(17);  //Set the year (Last two digits of the year)
     // Start the serial interface
    Serial.begin(115200);
}
void ReadDS3231()
{
    int second,minute,hour,date,month,year,temperature;
    second=Clock.getSecond();
    minute=Clock.getMinute();
    hour=Clock.getHour(h12, PM);
    date=Clock.getDate();
    month=Clock.getMonth(Century);
    year=Clock.getYear();

  temperature=Clock.getTemperature();

  sprintf(lcd_dis_str[0],"   20%02d-%02d-%02d",year,month,date);

sprintf(lcd_dis_str[1],"  %02d:%02d:%02d   %2d'",hour,minute,second,te
mperature);
}
void loop()
{
    ReadDS3231();
    lcd.clear();                                  // clean
    lcd.setCursor(0,0);                           // line0 display
    lcd.print(lcd_dis_str[0]);
    delay(10);
    lcd.setCursor(0,1);
    lcd.print(lcd_dis_str[1]);
    delay(1000);
}
```

Experiment Result