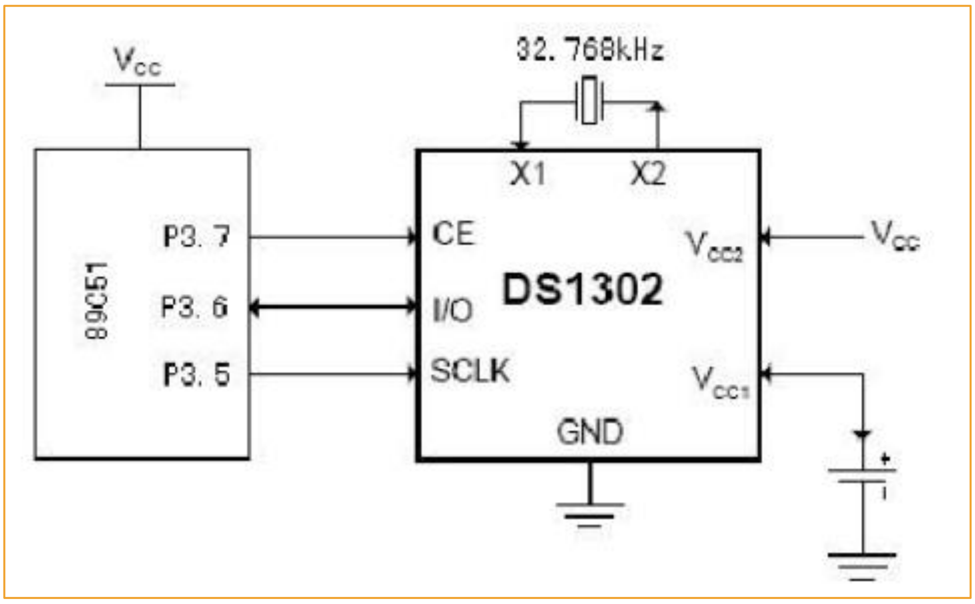
DS1302 Electronic Clock module Experiment

Introduction

There are many popular serial clock circuits, such as DS1302, DS3231, DS1307, PCF8485, etc. They are extensively used due to their simple interface, low cost and ease of use. In this lesson, we will use the DS1302 real-time clock module. As shown below, DS1302 is a high performance, low power consumption, RAM-equipped real-time clock circuit introduced by DALLAS, USA. It can be used for year, month, day and week. Day, hour, minute and second are timed, with leap year compensation function, working voltage is 2.5V~5.5V. Three-wire interface is adopted to synchronous communication with the CPU, and multiple bytes of clock signal or RAM data can be transmitted at one time in a burst mode. There is a 31×8 RAM register for temporarily storing data inside DS1302. DS1302 is an upgraded product of DS1202 and it is compatible with DS1202, but it adds dual power supply pins for main/backup power supply and it provides the ability to trickle current charge the backup power supply.

DS1302 Picture

The following is a typical application circuit for DS1302. We can see that in the diagram, the DS1302 requires almost no external components.



The pin arrangement of DS1302 wherein VCC1 is the backup power supply and VCC2 is the main power supply. The continuous operation of the clock can be maintained even while the main power is off. DS1302 is powered by the larger of VCC1 or VCC2. When VCC2 is greater than VCC1+0.2V, VCC2 supplies power to DS1302. When VCC2 is less than VCC1, DS1302 is powered by VCC1. X1 and X2 are the oscillation sources with external 32.768kHz crystal oscillators. RST is the reset/chip select line. All data transfers are started by driving the RST input to high. RST input has two functions: First, RST switches on the control logic, allowing the address/command sequence to be sent into the shift register; Second, RST provides a method to terminate single-byte or multi-byte data transfer. When RST is high level voltage, all data transfers are initialized and allowing operations on DS1302. If RST is set to low level voltage during transmission, the data transfer will be terminated and the I/O pin will become high impedance. During power-on operation, RST must remain low level voltage until VCC>2.0V. Only when SCLK is low level voltage can RST be set high level voltage. I/O is the serial data input and output terminal (bidirectional).

DS1302 Clock chip structure

The main component of the DS130 consist of 8 modules and it divided into 4 function groups: TCXO, power control, reset and RTC.

Experimental Purpose

- Learning the working principle and characteristics of DS1302 clock module;
- Making a clock with DS1302 clock module

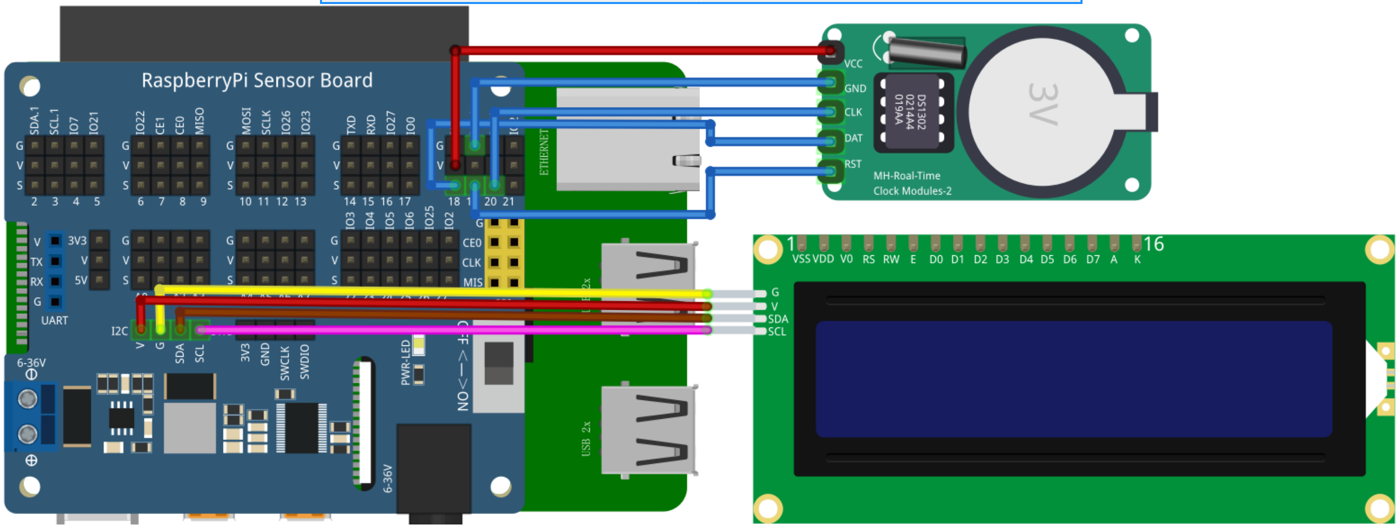
Component List

- Raspberry Pi main board
- Raspberry Pi expansion board
- USB Data Cable

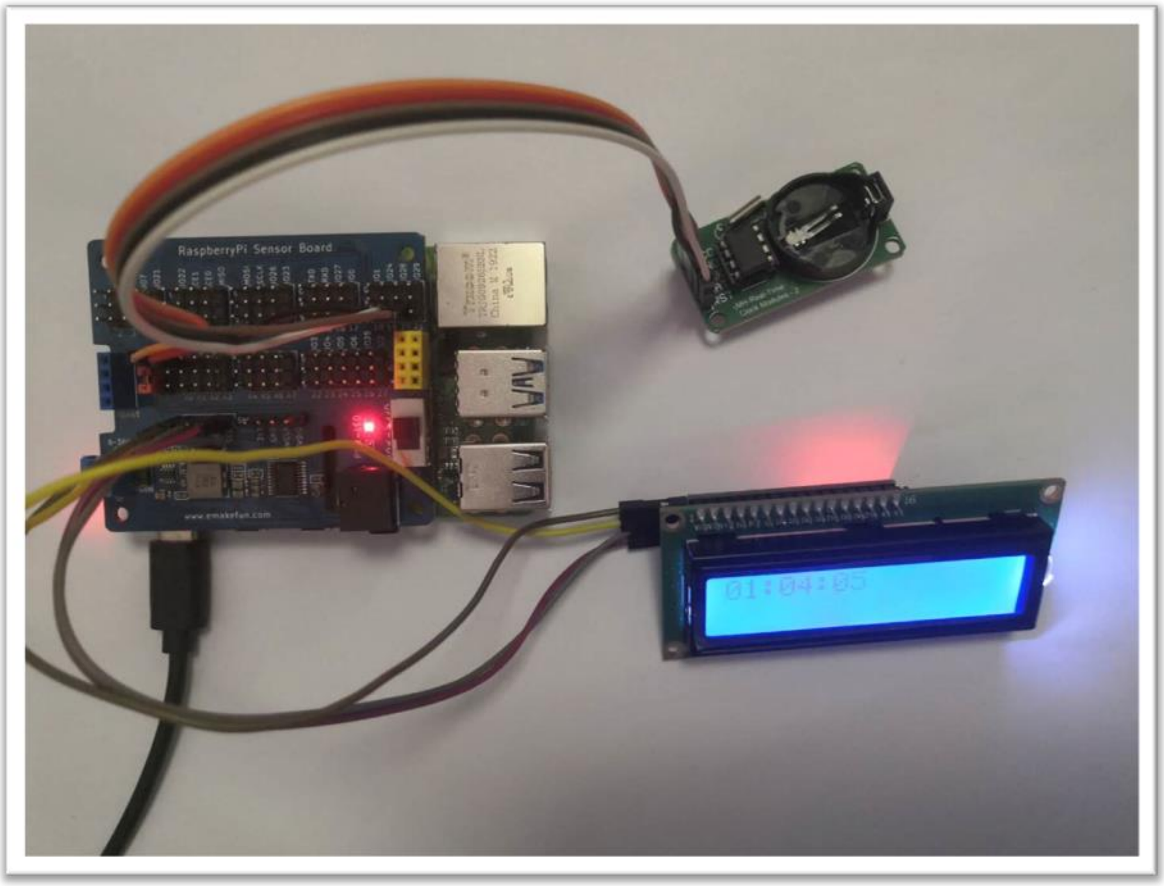
- 1602LCD *1
- DS1302 Clock Module *1
- Several jumper wires

Wiring

Raspberry Pi	LCD
GND	GND
VCC	VCC
SDA	SDA
SCL	SCL
Raspberry Pi	**DS1302**
IO24(wiringPi)/19(BCM)	RST
IO1(wiringPi)/18(BCM)	DAT
IO28(wiringPi)/20(BCM)	CLK
GND	GND
5V	VCC



RaspberryPi Sensor Board
VCC
GND
CLK
DAT
RST
MH-Real-Time Clock Modules-2
3V
VSS VDD V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
G
V
SDA
SCL



C++ partial program

```
#include "ds1302.h"
#include "LiquidCrystal_I2C.h"
#include<stdio.h>
#include<wiringPi.h>

typedef unsigned int u16;     //对数据类型进行声明定义
typedef unsigned char u8;

char num=0;
int main()
{
    init();
    Ds1302Init();
    while(1)
    {
        char DisplayData[8];
        Ds1302ReadTime();
        DisplayData[0] = TIME[2]/16;            //时
        DisplayData[1] = TIME[2]&0x0f;
        DisplayData[2] = 10;
        DisplayData[3] = TIME[1]/16;            //分
        DisplayData[4] = TIME[1]&0x0f;
        DisplayData[5] = 10;
        DisplayData[6] = TIME[0]/16;            //秒
        DisplayData[7] = TIME[0]&0x0f;
        for (int i = 0; i<8; i++)
        {
            printf("%d", DisplayData[i]);
        }
        printf("\n");
        for (int i = 0; i<8; i++)
        {
            write(i+8, 0, DisplayData[i]+48, 0);
        }
    delay(1000);
    clear();
    }
}
```

Python partial program

```
# !/usr/bin/env python
import time
import RPi.GPIO as GPIO
import smbus
import logx
import logging
import LCD1602

DSIO   =    18
RST    =    19
SCLK   =    20
READ_RTC_ADDR = [0x81, 0x83, 0x85, 0x87, 0x89, 0x8b, 0x8d]
WRITE_RTC_ADDR = [0x80, 0x82, 0x84, 0x86, 0x88, 0x8a, 0x8c]
TIME = [0, 0, 0x12, 0x07, 0x05, 0x06, 0x16]  #//--- DS1302 clock initialization Saturday, 
May 7, 2016, 12:00:00。---//
data = [0,0,0,0,0,0,0,0]
GPIO.setmode(GPIO.BCM)
GPIO.setup(DSIO, GPIO.OUT)
GPIO.setup(RST, GPIO.OUT)
GPIO.setup(SCLK,GPIO.OUT)
```

```
def Ds1302Write(addr, dat):
    GPIO.output(RST, False)
    GPIO.output(SCLK, False)
    GPIO.output(RST, True)
    for n in range(8):
        GPIO.output(DSIO,(addr & 0x01))
        addr= addr >> 1
        GPIO.output(SCLK, True)
        GPIO.output(SCLK, False)
    for n in range(8):
        GPIO.output(DSIO,(dat & 0x01))
        addr= addr >> 1
        GPIO.output(SCLK, True)
        GPIO.output(SCLK, False)
    GPIO.output(RST, False)

def Ds1302Read(addr):
    dat = 0
    GPIO.output(RST, False)
    GPIO.output(SCLK, False)
    GPIO.output(RST, True)
    for n in range(8):
        GPIO.output(DSIO,(addr & 0x01))
        addr= addr >> 1
        GPIO.output(SCLK, True)
        GPIO.output(SCLK, False)
    GPIO.setup(DSIO, GPIO.IN)
    for n in range(0,8):
        dat = dat | (GPIO.input(DSIO)<<n)
        GPIO.output(SCLK, True)
        GPIO.output(SCLK, False)
    GPIO.setup(DSIO, GPIO.OUT)
    GPIO.output(RST, False)
    GPIO.output(SCLK, True)
    GPIO.output(DSIO, False)
    GPIO.output(DSIO, True)
    return dat

def Ds1302ReadTime():
    for i in range(0,6):
        TIME[i] = Ds1302Read(READ_RTC_ADDR[i])
```

```
print(int(TIME[2]/16),(TIME[2]&0x0f),':',int(TIME[1]/16),(TIME[1]&0x0f),':',int(TIME[0]
/16),(TIME[0]&0x0f))
    data[0] = int(TIME[2]/16)
    data[1] = TIME[2]&0x0f
    data[2] = 10
    data[3] = int(TIME[1]/16)
    data[4] = TIME[1]&0x0f
    data[5] = 10
    data[6] = int(TIME[0]/16)
    data[7] = TIME[0]&0x0f

def Ds1302Init():
    Ds1302Write(0x8E, 0X00)
    for i in range(0,7):
        Ds1302Write(WRITE_RTC_ADDR[i], TIME[i])
    Ds1302Write(0x8E, 0x80)

Ds1302Init()
LCD1602.init_lcd()
while True:
    Ds1302ReadTime()
    for i in range(8):
       LCD1602.print_lcd_char(i, 0, data[i] + 48)
        #LCD1602.print_lcd(2, 0, "tt")
    time.sleep(1)
```

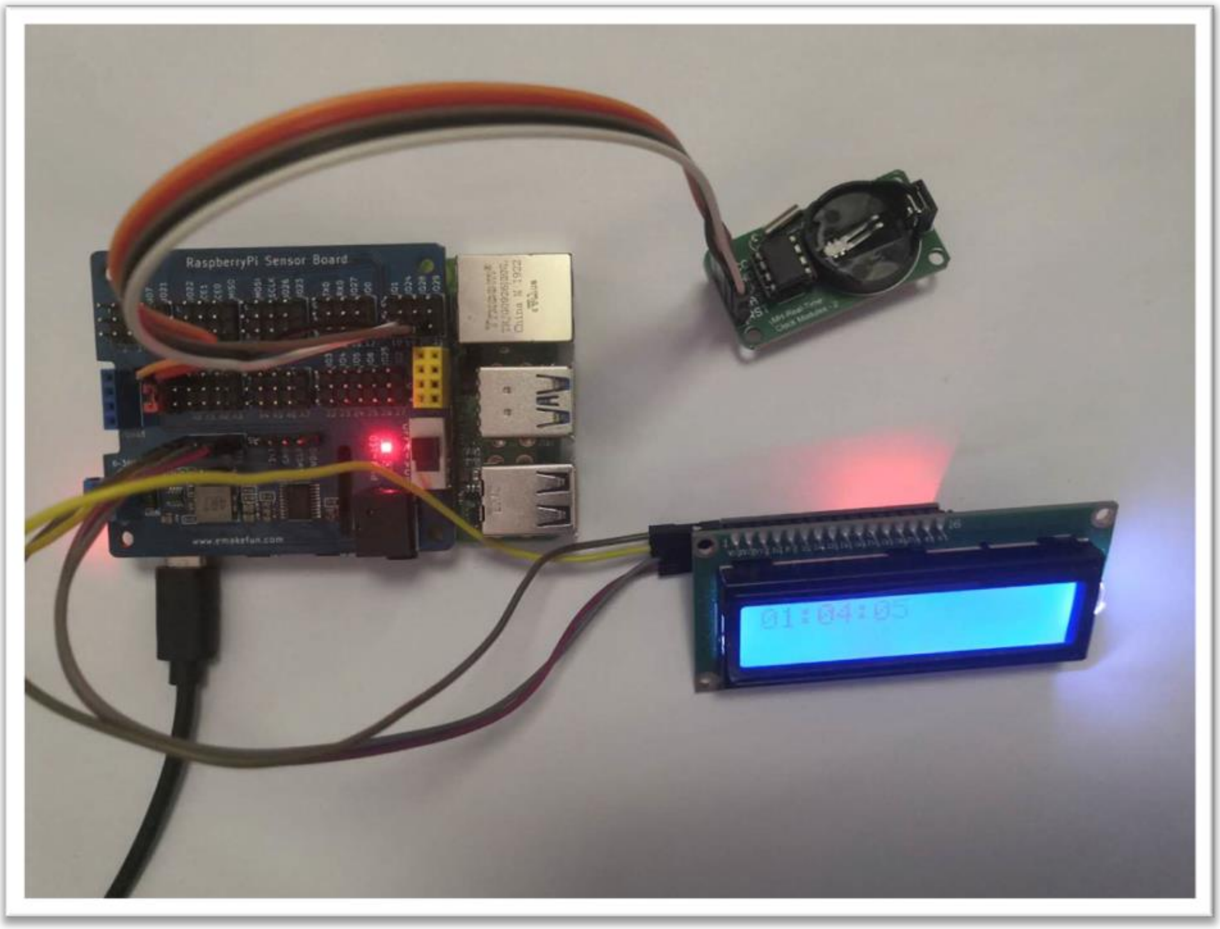
Java program

```
import com.pi4j.wiringpi.Gpio;
import com.pi4j.io.i2c.I2CBus;
import com.pi4j.io.i2c.I2CDevice;
import com.pi4j.io.i2c.I2CFactory;

public class DS1302 {
    int DSIO = 1, RST = 24, SCLK = 28;
    int READ_RTC_ADDR[]= {0x81, 0x83, 0x85, 0x87, 0x89, 0x8b, 0x8d};
    int WRITE_RTC_ADDR[] = {0x80, 0x82, 0x84, 0x86, 0x88, 0x8a, 0x8c};
    int TIME[] = {0, 0, 0x12, 0x07, 0x05, 0x06, 0x16};
    int data[] = {0, 0, 0, 0, 0, 0, 0, 0};
    String buf;

    void Ds1302Init() {
        Gpio.wiringPiSetup();
        Gpio.pinMode(DSIO, Gpio.OUTPUT);
        Gpio.pinMode(RST, Gpio.OUTPUT);
        Gpio.pinMode(SCLK, Gpio.OUTPUT);
        Ds1302Write(0x8E, 0x00);
        for (int i = 0; i < 7; i++) {
            Ds1302Write(WRITE_RTC_ADDR[i], TIME[i]);
        }
        Ds1302Write(0x8E, 0x80);
    }

    void Ds1302Write(int addr, int dat){
        Gpio.digitalWrite(RST, Gpio.LOW);
        Gpio.digitalWrite(SCLK, Gpio.LOW);
        Gpio.digitalWrite(RST, Gpio.HIGH);
        for (int i = 0; i < 8; i++) {
            Gpio.digitalWrite(DSIO, (addr & 0x01));
            addr= addr >> 1;
            Gpio.digitalWrite(SCLK, Gpio.HIGH);
            Gpio.digitalWrite(SCLK, Gpio.LOW);
        }
        for (int i = 0; i < 8; i++) {
            Gpio.digitalWrite(DSIO, (dat & 0x01));
            dat= dat >> 1;
            Gpio.digitalWrite(SCLK, Gpio.HIGH);
            Gpio.digitalWrite(SCLK, Gpio.LOW);
        }
        Gpio.digitalWrite(RST, Gpio.LOW);
    }

    int Ds1302Read(int addr){
        int dat = 0;
        Gpio.digitalWrite(RST, Gpio.LOW);
        Gpio.digitalWrite(SCLK, Gpio.LOW);
        Gpio.digitalWrite(RST, Gpio.HIGH);
        for (int i = 0; i < 8; i++) {
            Gpio.digitalWrite(DSIO, (addr & 0x01));
            addr= addr >> 1;
            Gpio.digitalWrite(SCLK, Gpio.HIGH);
            Gpio.digitalWrite(SCLK, Gpio.LOW);
        }
        Gpio.pinMode(DSIO,Gpio.INPUT);
        for (int i = 0; i < 8; i++) {
            int tt = Gpio.digitalRead(DSIO);
            dat = dat | (Gpio.digitalRead(DSIO) << i);
            Gpio.digitalWrite(SCLK, Gpio.HIGH);
            Gpio.digitalWrite(SCLK, Gpio.LOW);
        }
        Gpio.pinMode(DSIO,Gpio.OUTPUT);
        Gpio.digitalWrite(RST, Gpio.LOW);
        Gpio.digitalWrite(SCLK, Gpio.HIGH);
        Gpio.digitalWrite(DSIO, Gpio.LOW);
        Gpio.digitalWrite(DSIO, Gpio.HIGH);

        return dat;
    }

    void Ds1302ReadTime() {
        for (int i = 0; i < 7; i++) {
            TIME[i] = Ds1302Read(READ_RTC_ADDR[i]);
        }
        data[0] = (TIME[0] / 16 * 10) + (TIME[0] & 0x0f);
        data[1] = TIME[1] / 16 * 10 + TIME[1] & 0x0f;
        data[2] = TIME[2] / 16 * 10 + TIME[2] & 0x0f;
    }

    void get_time() {
        String s0 = String.valueOf(data[0]);
        String s1 = String.valueOf(data[1]);
        String s2 = String.valueOf(data[2]);
        StringBuffer buff = new StringBuffer();
        buff.append(s2);
        buff.append(":");
        buff.append(s1);
        buff.append(":");
        buff.append(s0);
        buf = buff.toString();
    }


    public static void main(String[] args) {
        DS1302 ds1302 = new DS1302();
        ds1302.Ds1302Init();
        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);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
        while (true) {
            ds1302.Ds1302ReadTime();
            ds1302.get_time();
            _lcd.display_string_pos(ds1302.buf, 1, 2);
        }
    }
}
```

Java program

```
import com.pi4j.wiringpi.Gpio;
import com.pi4j.io.i2c.I2CBus;
import com.pi4j.io.i2c.I2CDevice;
import com.pi4j.io.i2c.I2CFactory;

public class DS1302 {
    int DSIO = 1, RST = 24, SCLK = 28;
    int READ_RTC_ADDR[]= {0x81, 0x83, 0x85, 0x87, 0x89, 0x8b, 0x8d};
    int WRITE_RTC_ADDR[] = {0x80, 0x82, 0x84, 0x86, 0x88, 0x8a, 0x8c};
    int TIME[] = {0, 0, 0x12, 0x07, 0x05, 0x06, 0x16};
    int data[] = {0, 0, 0, 0, 0, 0, 0, 0};
    String buf;

    void Ds1302Init() {
        Gpio.wiringPiSetup();
        Gpio.pinMode(DSIO, Gpio.OUTPUT);
        Gpio.pinMode(RST, Gpio.OUTPUT);
        Gpio.pinMode(SCLK, Gpio.OUTPUT);
        Ds1302Write(0x8E, 0x00);
        for (int i = 0; i < 7; i++) {
            Ds1302Write(WRITE_RTC_ADDR[i], TIME[i]);
        }
        Ds1302Write(0x8E, 0x80);
    }

    void Ds1302Write(int addr, int dat){
        Gpio.digitalWrite(RST, Gpio.LOW);
        Gpio.digitalWrite(SCLK, Gpio.LOW);
        Gpio.digitalWrite(RST, Gpio.HIGH);
        for (int i = 0; i < 8; i++) {
            Gpio.digitalWrite(DSIO, (addr & 0x01));
            addr= addr >> 1;
            Gpio.digitalWrite(SCLK, Gpio.HIGH);
            Gpio.digitalWrite(SCLK, Gpio.LOW);
        }
        for (int i = 0; i < 8; i++) {
            Gpio.digitalWrite(DSIO, (dat & 0x01));
            dat= dat >> 1;
            Gpio.digitalWrite(SCLK, Gpio.HIGH);
            Gpio.digitalWrite(SCLK, Gpio.LOW);
        }
        Gpio.digitalWrite(RST, Gpio.LOW);
    }

    int Ds1302Read(int addr){
        int dat = 0;
        Gpio.digitalWrite(RST, Gpio.LOW);
        Gpio.digitalWrite(SCLK, Gpio.LOW);
        Gpio.digitalWrite(RST, Gpio.HIGH);
        for (int i = 0; i < 8; i++) {
            Gpio.digitalWrite(DSIO, (addr & 0x01));
            addr= addr >> 1;
            Gpio.digitalWrite(SCLK, Gpio.HIGH);
            Gpio.digitalWrite(SCLK, Gpio.LOW);
        }
        Gpio.pinMode(DSIO,Gpio.INPUT);
        for (int i = 0; i < 8; i++) {
            int tt = Gpio.digitalRead(DSIO);
            dat = dat | (Gpio.digitalRead(DSIO) << i);
            Gpio.digitalWrite(SCLK, Gpio.HIGH);
            Gpio.digitalWrite(SCLK, Gpio.LOW);
        }
        Gpio.pinMode(DSIO,Gpio.OUTPUT);
        Gpio.digitalWrite(RST, Gpio.LOW);
        Gpio.digitalWrite(SCLK, Gpio.HIGH);
        Gpio.digitalWrite(DSIO, Gpio.LOW);
        Gpio.digitalWrite(DSIO, Gpio.HIGH);

        return dat;
    }

    void Ds1302ReadTime() {
        for (int i = 0; i < 7; i++) {
            TIME[i] = Ds1302Read(READ_RTC_ADDR[i]);
        }
        data[0] = (TIME[0] / 16 * 10) + (TIME[0] & 0x0f);
        data[1] = TIME[1] / 16 * 10 + TIME[1] & 0x0f;
        data[2] = TIME[2] / 16 * 10 + TIME[2] & 0x0f;
    }

    void get_time() {
        String s0 = String.valueOf(data[0]);
        String s1 = String.valueOf(data[1]);
        String s2 = String.valueOf(data[2]);
        StringBuffer buff = new StringBuffer();
        buff.append(s2);
        buff.append(":");
        buff.append(s1);
        buff.append(":");
        buff.append(s0);
        buf = buff.toString();
    }


    public static void main(String[] args) {
        DS1302 ds1302 = new DS1302();
        ds1302.Ds1302Init();
        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);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
        while (true) {
            ds1302.Ds1302ReadTime();
            ds1302.get_time();
            _lcd.display_string_pos(ds1302.buf, 1, 2);
        }
    }
}
```

Experimental Effect



DS1302 is controlled by Raspberry Pi to read the time information and display the data information on LCD1602.