

**Part 4**

**Lesson**

**5**

**Multifunctional  
Alarm Clock**

## Summary:

In this course, you will learn how to make a multifunctional alarm clock by expanding the timer in Lesson 9 of Part 3. In this course, you will use 4-bit 7-segment digital tube, RTC real-time clock, DHT11 temperature and humidity sensor, RGB LED and other modules, and combine them to achieve your multifunctional alarm clock.

### ■ Component Required:

- (1) x Elegoo Uno R3
- (1) X Active Buzzer
- (1) x DHT11 Temperature and Humidity module
- (1) x DS1307 RTC module
- (1) x 4 Digit 7-Segment Display
- (1) x ALL IN ONE Sensor Shield



## Introduction

**There are four buttons in this experiment, "minute" button:**

- **Every** times you press it, the time you set will add one minute. Once you press the key and hold it down, the time will add faster.
- **In** the mode of setting alarm clock, press once, add one hour to the timing time, and when holding the button down, turn on the continuous pressing mode.
- **"second" button:**  
Every times you press it, the time you set will add one second. Once you press the key and hold it down, the time will add faster.  
In the mode of setting alarm clock, press once, add one minute to the timing time, and when holding the button down, turn on the continuous pressing mode.
- **"function" button:**  
After finishing setting the time, you should press this button. When you press it, you will see the 4 Digit 7-Segment Display begin to count down. If you press it again during counting down, the time you set will be cleared. When the countdown time comes, the buzzer sounds and the RGB LED lights flicker. At this time, if you press the function key again, the sound will be turned off, the RGB led will stop flickering and the time you set will be cleared too.

## "Display switch" button:

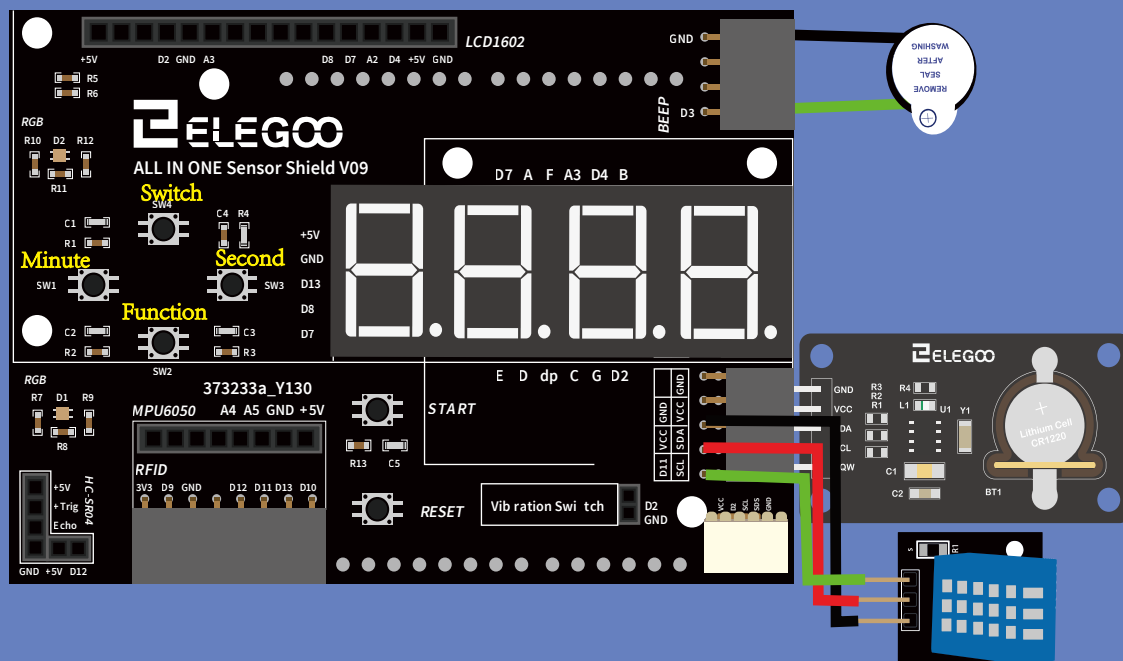
Press once to display year, press twice to display month, press three times to display time, press four times to display temperature, press five times to display humidity, and circle in turns.

## Compared to Lesson 9 of Part3, add the following functions:

1. RTC real-time clock module and DHT11 temperature and humidity module are added to collect data for displaying date, time, temperature and humidity.
2. Static 5S switch back to display time interface.
3. Set the alarm clock, store the data in EEPROM, and it won't affect the use after the resumption of power outage. and after the resumption of power failure does not affect the use.
4. Replace LED lamp with RGB color lamp.

Setting the alarm clock: Press the function key on the interface of displaying the current time, the RGB color lamp turns yellow, and the digital tube shows '0000', indicating that the alarm clock is set. After setting, press the function key again. Before the alarm clock rings, the RGB color lamp will turn yellow, which represents the success of the alarm clock setting. If the color light is green, it means no alarm clock is set.

## Wiring diagram:



## Code

- Functions of each module are introduced as follows:
- `interface_display()`  
`display_num(uint16_t num)`
- `temperature_display(uint16_t num)`  
`humidity_display(uint16_t num)`

```
if(flag_alarm_set || function_clock)
{
    .....
}
```

## Digital tube:

- Interface Display:`interface_display()`  
-> Digital display:`display_num(uint16_t num)`
  - ① Display countdown number, year, month and day.
  - ② Time-division figures show that there is a '0000' situation.
- -> Temperature display:`temperature_display(uint16_t num)`  
Difference: The number at the end is C.
- -> Humidity display:`humidity_display(uint16_t num)`  
Difference: The number at the end is H.

```
if(flag_alarm_set || function_clock)
{
    .....
}
```

## DHT11

- `measure_environment( float *temperature, float *humidity )`  
`update_dht11_value()`
- DHT11 Temperature and Humidity Module:  
-> Measuring ambient temperature and humidity:  
`measure_environment( float *temperature, float *humidity )`  
-> Update temperature and humidity data:`update_dht11_value()`

## RGB

- >Display yellow for prompting to set the alarm clock: `rgb_yellow()`
- >Display green, indicating normal power on, for no alarm clock set: `rgb_green()`
- >Choose yellow light or green light: `rgb_light()`
- >Turn off the light: `close_rgb()`
- >Color gradient, used to prompt the timing to: `rgb_color()`

## Buzzer

- >Determine whether the timing is up: `time_out()`
- >When the time comes, the buzzer sounds and RGB color changes: `rgb_color()`

## Eeprom:

We use EEPROM to store the set alarm clock time and determine which color lights are on.

- >Put int data into eeprom:  
`eeprom_write_int(unsigned int address, unsigned int data)`
- >Read int data from eeprom  
`eeprom_read_int(unsigned int address)`
- >Clear the contents of eeprom  
`eeprom_clear_all(unsigned int eeprom_size)`

## Clock

Because RTC real-time clock module uses IIC communication, and timer interruption is used in this course, which frequently interrupts in a very short period of time. It may lead to interruption of communication between module and UNO, but cannot produce IIC events, and thus stuck in the while loop. So we transplant the DS3231 library files and add timeout processing to the while loop where errors may occur.



```
RTCDateTime clock_getDateTime(void)
{
    int values[7];

    Wire.beginTransaction(DS3231_ADDRESS);
    #if ARDUINO >= 100
        Wire.write(DS3231_REG_TIME);
    #else
        Wire.send(DS3231_REG_TIME);
    #endif
    Wire.endTransmission();

    Wire.requestFrom(DS3231_ADDRESS, 7);

    while(!Wire.available())
    {
        delay(1);
        error_cnt++;
        if(error_cnt>15)
        {
            error_cnt=0;
            return dt;}};
    .....
```

In addition, in case the program gets stuck, we need to configure a watchdog timer, feed the dog regularly and reset it when it gets stuck.

```
void setup()
{
  ....
  wdt_enable(WDTO_120MS);
  ....}
```

```
void loop()
{
  ....
  wdt_reset();
  ....}
```

```
void rgb_color()
{
  ....
  for(int i = 0; i < 20; i += 1)
  {
    wdt_reset();
    .....
  }
}
```

```
uint8_t clock_readRegister8(uint8_t reg)
{
  uint8_t value;
  Wire.beginTransmission(DS3231_ADDRESS);
  #if ARDUINO >= 100
    Wire.write(reg);
  #else
    Wire.send(reg);
  #endif
  Wire.endTransmission();

  Wire.requestFrom(DS3231_ADDRESS, 1);
  while(!Wire.available()) {
    delay(1);
    error_cnt++;
    if(error_cnt > 15)
    {
      error_cnt = 0;
      return value;
    }
  }
  return value;
}
```

Key: `button_s_interrupt()`

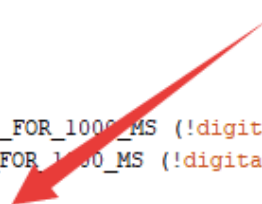
In the use of keys, the level change interruption is used, and interruption occurs when the drop edge is detected.

Minute key:

The operation required by the key:

① Eliminating jitter

MyTimer	DHT11	MCU_timer	MySegDispaly	MyTimer.h	MyTimerDetails.h	RGB	button	buzzer
1	#ifndef MYTIMERDETAILS_H							
2	#define MYTIMERDETAILS_H							
3								
4								
5	#define BUTTON_MS_WAS_PRESSED_AND_LASTS_FOR_1000_MS (!digitalRead(BUTTON_MS) && 1000 < (abs(time							
6	#define BUTTON_S_WAS_PRESSED_AND_LASTS_FOR_1000_MS (!digitalRead(BUTTON_S) && 1000 < (abs(time							
7								
8	#define ELIMINAT_THE_JITTER_OF_BUTTON_S (200 < (abs(time_button_s - last_time_button_s)))							



```
void button_s_interrupt()
{
    time_button_s=millis();

    if (ELIMINAT_THE_JITTER_OF_BUTTON_S)
    {.....}
```

- ② Determine whether the button is pressed to enter the countdown mode, or already entered the countdown mode.

-> If you enter the countdown mode, clear it first, otherwise add 1 minutes.

```
void button_s_interrupt()
{
    time_button_s=millis();
    if (ELIMINAT_THE_JITTER_OF_BUTTON_S)
    {
        .....
        if(function_clock || flag_year || flag_month || flag_temperature || flag_humidity )
        {
            display_clear();
            number = 0;
        }
    }
```

- ③ The alarm clock should not exceed 2359.

```
else if(flag_alarm_set && number >=2359)
{ number = 0;}
```

- ④ The maximum countdown time can only be 9999.

```
if (number >= 9900)
number = 9999;
```

For the same reason to the second button.

Display switch button: button\_display\_change\_interrupt()  
Switching display content in switch statement according to the number of keystrokes pressed

```
switch(display_mode[press_cnt++])
{
    case DISPLAY_YEAR:
        display_year();
        break;
    case DISPLAY_MONTH:
        display_month();
        break;
    case DISPLAY_TIME:
        display_time();
        break;
    case DISPLAY_TEMPERATURE:
        display_temperature();
        break;
    case DISPLAY_HUMIDITY:
        display_humidity();
        break;
}
```

button\_choose\_interrupt ()

Function button: button\_choose\_interrupt ()

① Turn off the alarm clock when it rings

```
if(flag_alarm_ring)
{
    flag_alarm_ring=0;
    function_clock=1;
    alarm_value = 0;
    eeprom_write_int(0,number);
    waiting_time_cnt=104;
    task2_cnt = 20;
    last_time_button_choose = time_button_choose;
}
```

② Start setting alarm clocks

```
else if( ( flag_year == 0 )  &&
        ( flag_month == 0 )  &&
        ( flag_temperature == 0 )&&
        ( flag_humidity == 0 ) &&
        ( function_clock )  &&
        ( flag_alarm == 1 )  && (flag_begin == 0) )
{
    function_clock = 0;
    number = 0;
    alarm_value = 0;
    rgb_yellow();
    flag_alarm_set=1;
    last_time_button_choose = time_button_choose;
}
```



### ③ Confirm the alarm clock setting

```
( ((flag_alarm_set && flag_alarm == 0) || (number == 0 && function_clock == 0)) && (flag_begin != 1))
{
    flag_alarm = 1;
    flag_alarm_set = 0;
    eeprom_write_int(0,number);
    alarm_value = eeprom_read_int(0);
    flag_alarm_set_ok = 1;
    EEPROM.write(3,flag_alarm_set_ok);
    function_clock = 1;
    waiting_time_cnt=104;
    last_time_button_choose = time_button_choose;
}
```

### ④ On-off timer function

```
else if( flag_alarm_set != 1)
{
    clock_to_timer();
    rgb_light();

    if(flag_begin)
    {
        waiting_time_cnt=104;
    }
    flag_begin=!flag_begin;
    last_time_button_choose = time_button_choose;
}
```

### Timer2:timer2\_interrupt()

① Judge whether to turn on the continuous pressing mode every 0.1 millisecond:

`task1_press_continuously()`

② Determine whether to turn on the countdown mode. Decrement per second.

`task2_cout_down()`

③ Update clock data every 0.2 Ms.

`task3_update_RTctime()`

④ No operation 5 seconds switch back to display time interface

`task_reset()`