

Teil 4

Lektion

5

Multifunktionale Alarm Uhr

Zusammenfassung:

In diesem Kurs lernen Sie, wie Sie einen multifunktionalen Wecker herstellen, indem Sie den Timer in Lektion 9 von Teil 3 erweitern. In diesem Kurs verwenden Sie eine 4-Bit-Digitalröhre mit 7 Segmenten, eine RTC-Echtzeituhr, eine DHT11-Temperatur und Feuchtigkeitssensor, RGB-LED und andere Module und kombinieren diese, um Ihren multifunktionalen Wecker zu erhalten.

■ Component Required:

- (1) x Elegoo Uno R3
- (1) X Active Buzzer
- (1) x DHT11 Temperature and Humidity module
- (1)xDS1307 RTC module
- (1) x 4 Digit 7-Segment Display
- (1) x ALL IN ONE Sensor Shield



Introduction

In diesem Experiment gibt es vier Schaltflächen:

■ "Minute" -Taste:

Jedes Mal, wenn Sie darauf drücken, wird die von Ihnen eingestellte Zeit um eine Minute verlängert. Sobald Sie die Taste drücken und gedrückt halten, wird die Zeit schneller. Drücken Sie im Modus zum Einstellen des Weckers einmal, verlängern Sie die Zeit um eine Stunde und schalten Sie bei gedrückter Taste den kontinuierlichen Druckmodus ein.

■ "zweite" Taste:

Jedes Mal, wenn Sie darauf drücken, wird die eingestellte Zeit um eine Sekunde verlängert. Sobald Sie die Taste drücken und gedrückt halten, wird die Zeit schneller. Drücken Sie im Modus zum Einstellen des Weckers einmal, verlängern Sie die Zeit um eine Minute, und schalten Sie bei gedrückter Taste den kontinuierlichen Druckmodus ein.

■ "Funktionstaste":

Nachdem Sie die Uhrzeit eingestellt haben, sollten Sie diese Taste drücken. Wenn Sie darauf drücken, beginnt die 4-stellige 7-Segment-Anzeige mit dem Countdown. Wenn Sie es während des Countdowns erneut drücken, wird die von Ihnen eingestellte Zeit gelöscht. Wenn die Countdown-Zeit kommt, ertönt der Summer und die RGB-LED flackert. Wenn Sie zu diesem Zeitpunkt die Funktionstaste erneut drücken, wird der Ton ausgeschaltet, die RGB-LED hört auf zu flackern und die von Ihnen eingestellte Zeit wird ebenfalls gelöscht.

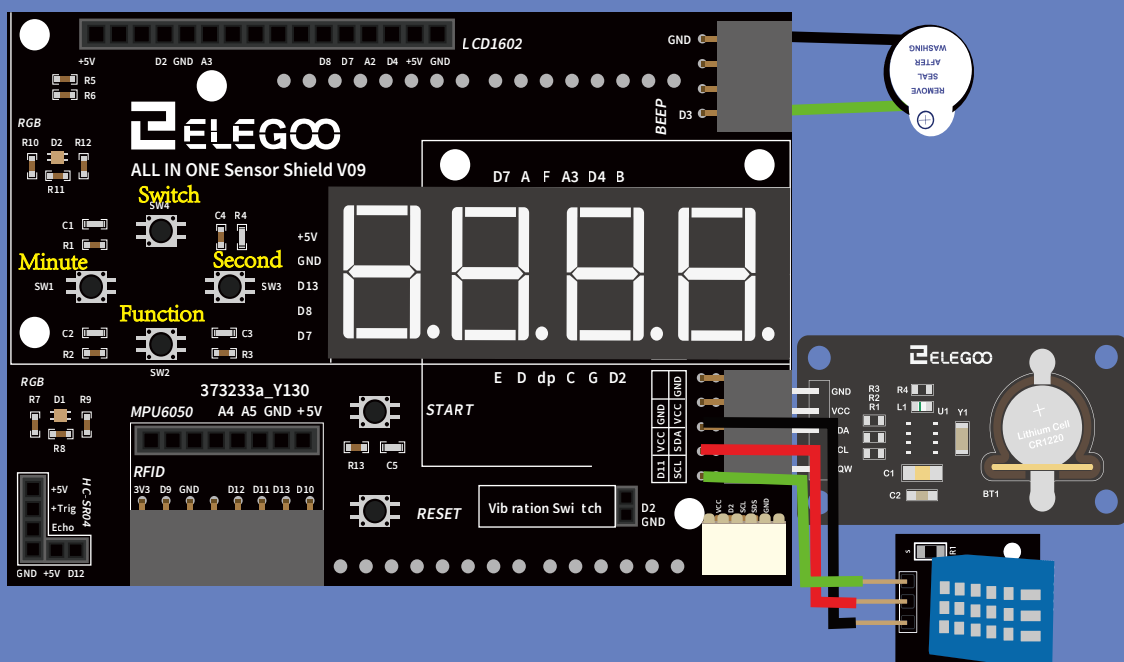
Schaltfläche "Anzeigeschalter":

Drücken Sie einmal, um das Jahr anzuzeigen, zweimal, um den Monat anzuzeigen, drücken Sie dreimal, um die Zeit anzuzeigen, drücken Sie viermal, um die Temperatur anzuzeigen, drücken Sie fünfmal, um die Luftfeuchtigkeit anzuzeigen. Sie können rotierend abwechseln.

Fügen Sie im Vergleich zu Lektion 9 von Teil 3 die folgenden Funktionen hinzu:

1. Das RTC-Echtzeituhrmodul und das DHT11-Temperatur- und Feuchtigkeitsmodul werden hinzugefügt, um Daten für die Anzeige von Datum, Uhrzeit, Temperatur und Luftfeuchtigkeit zu erfassen.
 2. Statischer 5S-Schalter: zurück zur Anzeigezeitschnittstelle.
 3. Stellen Sie den Wecker ein, speichern Sie die Daten im EEPROM und die Verwendung nach Wiederaufnahme des Stromausfalls wird nicht beeinträchtigt. Nach der Wiederaufnahme von Strom wird die Verwendung nicht beeinflusst.
 4. Ersetzen Sie die LED-Lampe durch eine RGB-Farb-LED.
- Einstellen des Weckers: Drücken Sie die Funktionstaste an der Schnittstelle zur Anzeige der aktuellen Uhrzeit, die RGB-Farblampe wird gelb und die Digitalröhre zeigt '0000' an, um anzuzeigen, dass der Wecker eingestellt ist. Drücken Sie nach dem Einstellen erneut die Funktionstaste. Bevor der Wecker klingelt, leuchtet die RGB-Farblampe gelb, was den Erfolg der Weckereinstellung darstellt. Wenn das Farblicht grün ist, bedeutet dies, dass kein Wecker eingestellt ist.

Wiring diagram:



Code

- Die Funktionen jedes Moduls werden nun vorgestellt:
- `interface_display()`
`display_num(uint16_t num)`
- `temperature_display(uint16_t num)`
`humidity_display(uint16_t num)`

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

Digitale Röhre:

- Schnittstellen-Anzeige: `interface_display()`
-> digitale Anzeige: `display_num(uint16_t num)`
 - ① Countdown für das Jahr, den Monat und den Tag anzeigen.
 - ② Zeitteilungs-Divisor zeigen, dass es eine '0000'-Option gibt.
- -> Temperatur-Anzeige: `temperature_display(uint16_t num)`
Unterschied: Die Zahl am Ende ist C.
- -> Anzeige der Feuchtigkeit: `humidity_display(uint16_t num)`
Unterschied: Die Zahl am Ende ist H.

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

DHT11

- DHT 11 Temperatur und Luftfeuchte-Sensor:
-> Messung der Umgebungstemperatur und der Luftfeuchtigkeit
`measure_environment(float *temperature, float *humidity)`
-> aktualisiert die Werte: `update_dht11_value()`

RGB

- >gelb anzeigen, um die Einstellung des Weckers aufzurufen: `rgb_yellow()`
- >grün anzeigen, um das normale Einschalten anzuzeigen, wenn kein Wecker aktiv ist: `rgb_green()`
- > Farbwähler zwischen gelb oder grün: `rgb_light()`
- >Licht ausschalten: `close_rgb()`
- >Farbverlauf, welcher für das Timing verwendet wird: `rgb_color()`

Summer

- >Bestimmen Sie, ob das Timing abgelaufen ist: `time_out()`
- >Wenn es soweit ist, ertönt der Summer und die RGB-Farbe ändert sich: `rgb_color()`

Eeprom:

Wir verwenden das EEPROM, um die eingestellte Weckerzeit zu speichern und festzustellen, welche Farbe leuchtet.

- >Speichere Int-Daten im EEPROM:
`eeprom_write_int(unsigned int address, unsigned int data)`
- >Lese Int-Daten aus dem EEPROM:
`eeprom_read_int(unsigned int address)`
- >Löschen Sie den Inhalt des EEPROM:
`eeprom_clear_all(unsigned int eeprom_size)`

Clock

Da das RTC-Echtzeituhrmodul die IIC-Kommunikation verwendet und die Timer-Unterbrechung in sehr kurzer Zeit verwendet wird, kann dies zu einer Unterbrechung der Kommunikation zwischen RTC-Modul und UNO führen. Es könnte sein, dass keine IIC-Ereignisse erzeugt werden und der Programmablauf in der while-Schleife stecken bleibt. Daher fügen wir die DS3231-Bibliotheksdateien ein und fügen der while-Schleife, in der Fehler auftreten könnten, eine Timeout-Verarbeitung hinzu.



```
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;}};
    .....
```

Falls der Sketch hängen bleibt, müssen wir außerdem einen Watchdog-Timer konfigurieren und ihn zurücksetzen, wenn er stecken bleibt.

```
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.beginTransaction(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;
}
```

Schlüssel: `button_s_interrupt()`

Bei der Verwendung von Schlüsseln wird die Unterbrechung der Pegeländerung verwendet, und eine Unterbrechung tritt auf, wenn die Fallflanke erkannt wird.

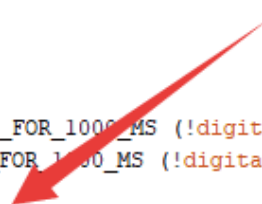
Minuten-Schlüssel:

Die für den Schlüssel erforderliche Operation:

① Jitter beseitigen

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_button_ms - last_time_button_ms)))
6 #define BUTTON_S_WAS_PRESSED_AND_LASTS_FOR_1000_MS (!digitalRead(BUTTON_S) && 1000 < (abs(time_button_s - last_time_button_s)))
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 Erkennt, ob eine Taste gedrückt wurde: wechselt in den Countdown-Modus oder ob der Countdown-Modus bereits aufgerufen wurde.
-> Wenn in den Countdown-Modus gesprungen wird, so wird dieser zuerst gelöscht, andernfalls 1 Minute hinzugefügt.

```
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 Der Wecker sollte 2359 nicht überschreiten.

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

- ④ Die maximale Countdown-Zeit kann nur 9999 betragen.

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

- Aus dem gleichen Grund auf die zweite Schaltfläche.
- Umschalten des Anzeigeinhalts in der switch-Anweisung entsprechend der Anzahl der gedrückten Tastenanschläge

```
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 ()

Funktionstaste: button_choose_interrupt()

- ①① Schalten Sie den Wecker aus, wenn er klingelt

```
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 Stellen Sie den Wecker ein

```
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 Bestätigen Sie die Wecker-Einstellung

```
( ((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 Ein-Aus-Timer-Funktion

```
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;
}
```

Timer 2: timer2 oder interrupt.

① Prüfen Sie, ob alle 0 Millisekunden eine Modus-Zeile geöffnet wird

`task1_press_continuously()`

② Bestimmen Sie, ob die Countdown-Einheit jede Sekunde gestartet wurde.

`task2_cout_down()`

③ Aktualisieren Sie die Daten der Uhr alle 0 Millisekunden.

`task3_update_RTCTime()`

④ Keine Operation. Die Zeitoberfläche wird in fünf Sekunden wieder angezeigt.

`task_reset()`