

Part 3

Lesson

9

Creating a Timer

Summary:

In this course, you will learn how to make timers for all kinds of occasions. In the process of learning how to make it, you will learn about MCU interrupts, key debouncing, dynamic scanning of digital displays and other related knowledge.

Component Required:

- (1) x Elegoo Uno R3
- (1) x 830 breadboard
- (1) X Active Buzzer
- (1) x NPN transistor 8050
- (3) x key
- (1) x LED lamp
- (5) x 220_resistance
- (1) x 10k_resistance
- (1) x 74HC595 IC
- (1) x 4 Digit 7-Segment Display
- (2) x M-F wires (Male to Female jumper wires)
- (32) x M-M wires (Male to Male jumper wires)



Component Introduction

There are three 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 quickly.

"second" button:

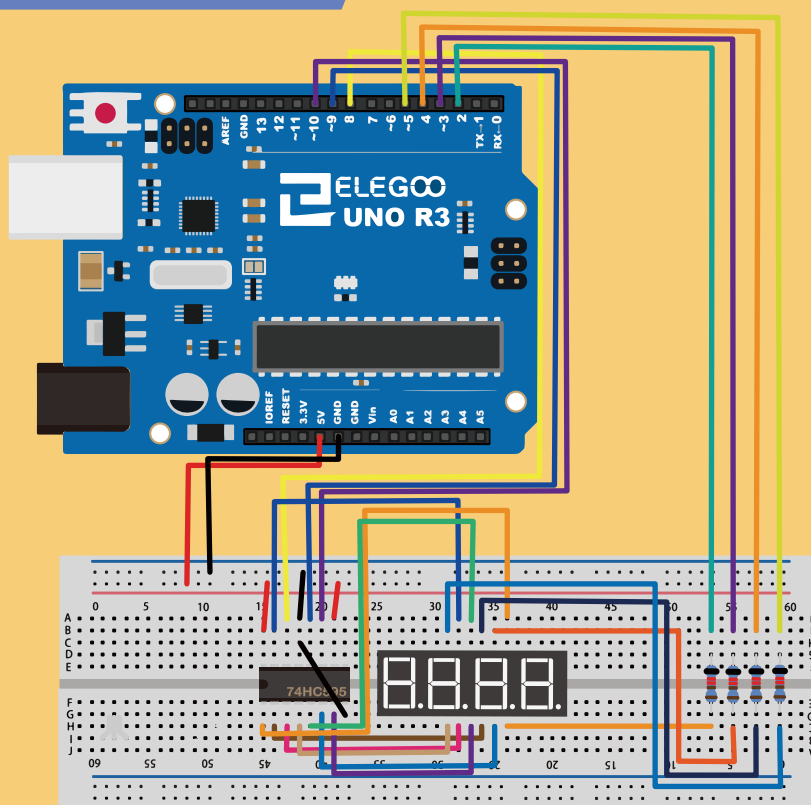
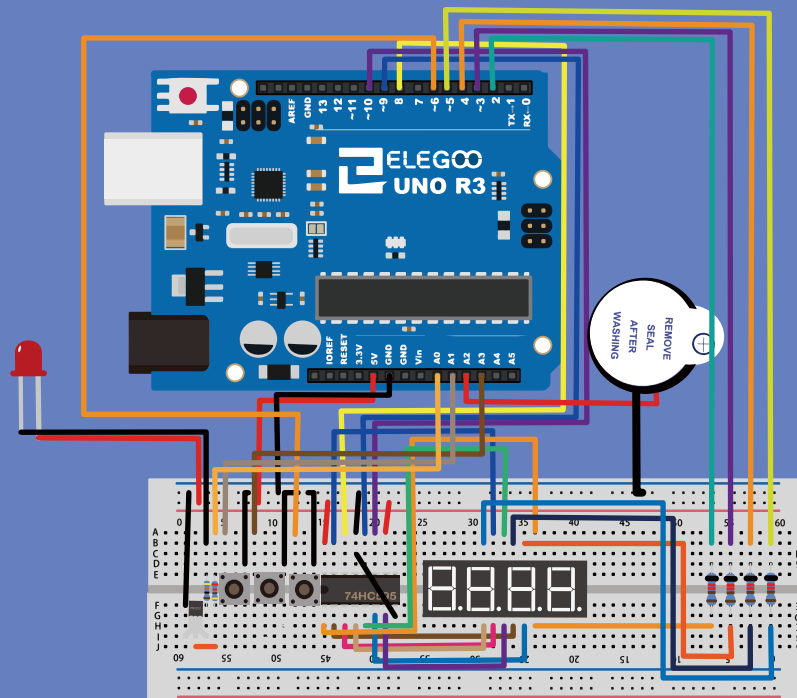
Every times you press it, the time you set will add one second. Once you press the key and hold it down, the time will add quickly.

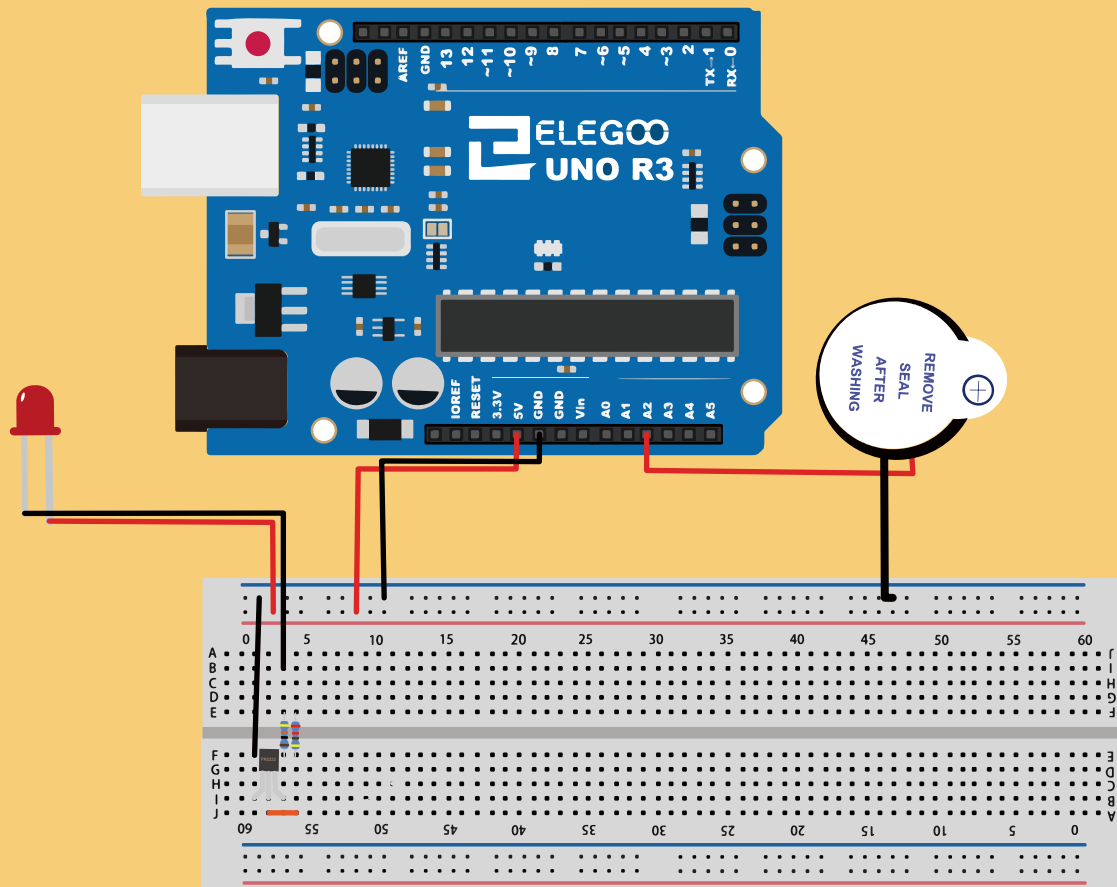
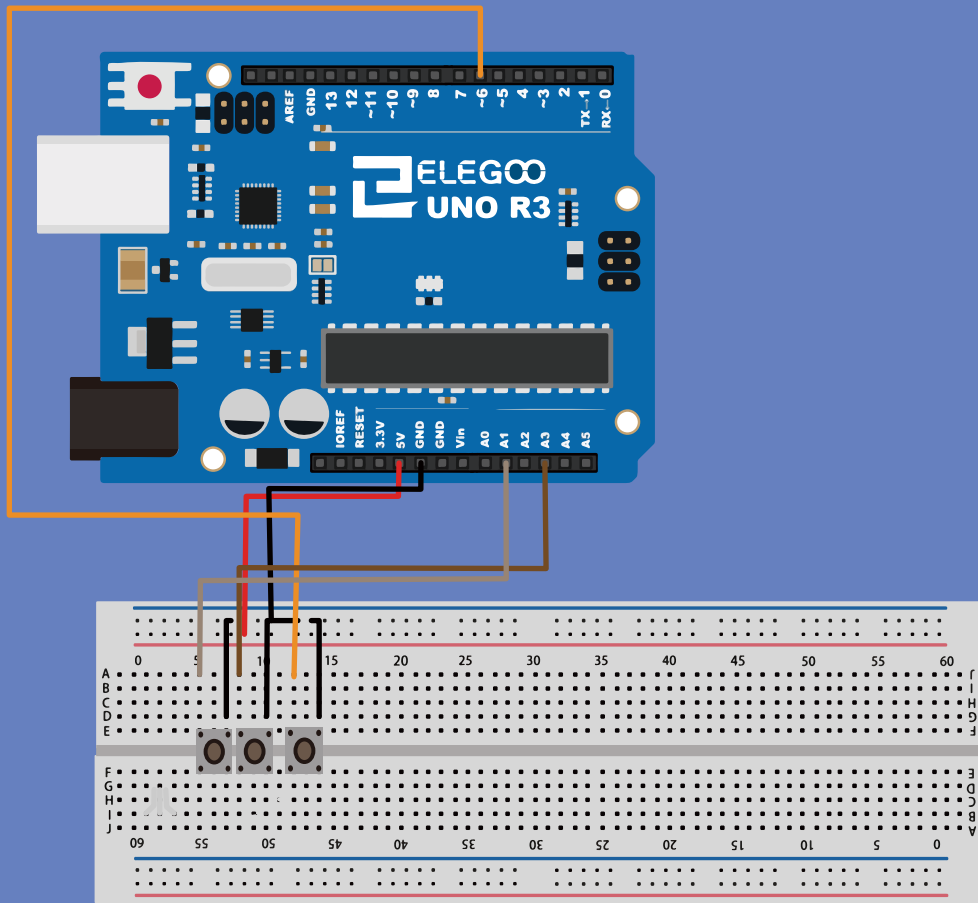
"function" button:

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 ends, the buzzer sounds and the LED lights flicker. At this time, if you press the function key again, the sound will turn off and the LED will stop flickering and the time you set will be cleared too.

Wiring diagram

Because there are too many devices connected, the driving ability of the IO port is not enough. So we use high-level driving mode for the A0 port connected with the led, and add 10K pull-up resistance to help the IO port output current. Normally, we use NPN 8050 transistor to drive the load of about 5V.





Watchdog interrupt:

Although no watchdog interruption is used in this course, we also need to understand the watchdog interruption.

The watchdog timer is mainly used to avoid equipment failure and crash, such as goes into the endless loop. If the code goes into an endless loop, the watchdog will triggers the interruption and resets the microcontroller.

The watchdog timer is set to one second and the setup contains a delay of two seconds therefore the watchdog will trigger.

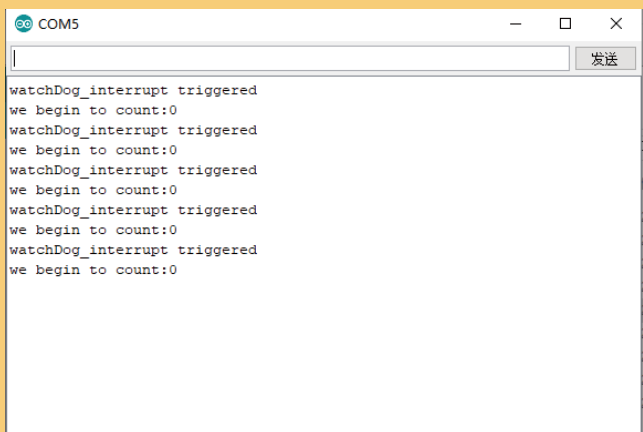
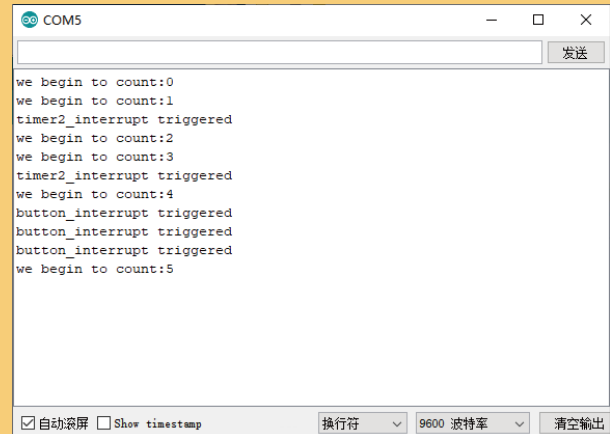
In this case, you need to delete it before running the program.

`Wdt_enable (WDTO_1S);`

`Serial.println ("watchDog_interrupt triggered");`preceding `"/"`

After uploading the program, you will see:

```
#include "avr/wdt.h"
void setup()
{
  wdt_enable(WDTO_1S);
  Serial.println("watchDog_interrupt triggered");
}
void button_interrupt()
{
  Serial.print("we begin to count:");
  Serial.println(number++);
  delay(2000);
  wdt_reset(); //we should feed the dog avoiding it reseted.
}
```



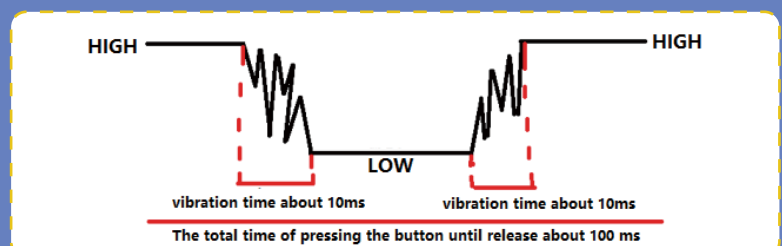
Part 2 : Keyboard Bounce Elimination

Mechanical elastic switch:

When the mechanical contact is opened and closed, because of the elastic effect of the contact, the switch will not close stably, it will bounce for a short time. The same thing happens when the switch is opened. This bouncing is eliminated in this lesson.

The problem of key bouncing is that it will cause multiple interrupts (3) instead of the one desired by the single press of the key.

You can see the bouncing depicted in the diagram.



Part 1: Interrupt

Interrupt is an indispensable mechanism for our design. Normally, our program will be executed sequentially in the `loop()`. Once something urgent happens, we need to use interruption to complete the urgent matter, and then come back and continue to execute the original program sequentially.

After wiring, please open the program in the code folder- **demo1** and click **UPLOAD** to upload the program.

Before running this program, make sure that the `<PinChangeInt>` library and `<MsTimer2>` library is installed or reinstalled if necessary. Otherwise, your code will not be compiled. For more information about loading library files, see Lesson 5 in part 1.

Timer2 interrupt:

Run the code you will see:

This means that the code that is executed sequentially in `loop()` will be interrupted by timer interrupt. The contents of interrupt function will be executed first. Only after the interrupt function has been executed, will the code run return to `loop()` function.

```
#include "MsTimer2.h"
```

```
void setup()
```

```
{
```

```
  MsTimer2::set(4000, timer2_interrupt);
```

```
  MsTimer2::start();
```

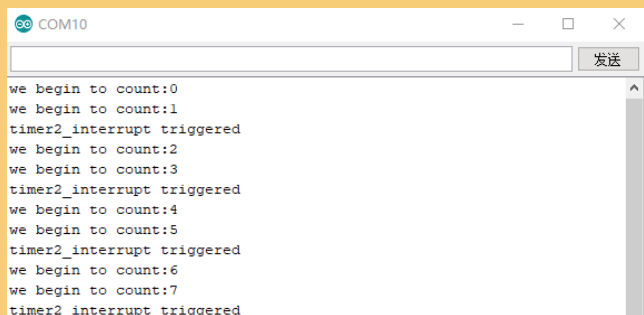
```
}
```

```
void timer2_interrupt()
```

```
{
```

```
  Serial.println("timer2_interrupt triggered");
```

```
}
```



Pin Change Interrupt

Both pin change interrupt and timer interrupt belong to interrupt, but trigger mode changes from timing trigger to level change trigger interrupt.

The level changes include: HIGH, LOW, RISING (low level to high level conversion), FALLING (high level to low level conversion), CHANGE.

In this case, we set the "FALLING" interrupt when the key is pressed.

When you press the button, you will see the following:

But why do we only press a button once and have three interrupts?

Please think carefully and we will answer in the second part.

```
#include "PinChangeInt.h"
```

```
void setup()
```

```
{
```

```
  pinMode(A3, INPUT_PULLUP);
```

```
  attachPinChangeInterrupt(A3, button_interrupt, FALLING);
```

```
}
```

```
void button_interrupt()
```

```
{
```

```
  Serial.println("button_interrupt triggered");
```

```
}
```

- After wiring, please open the program in the code folder- **demo2** and click UPLOAD to upload the program.
- Because of bouncing, we need to add a short delay to eliminate the bouncing.
- Using a long delay is not efficient as the MCU is doing nothing.
- So milliseconds are used to determine if the key has been pressed twice or not.
- The time we use to discriminate between bouncing or 2 genuine presses is 200 milliseconds. Greater than 200 ms determines that it is not switch bounce.

```
if (200 < (abs(time_buttonApin - last_time_buttonApin)))
{
  Serial.println("buttonApin have been pressed");
  last_time_buttonApin = time_buttonApin; //update the time you press
}
```

Part 3 : Dynamic Scanning of Digital Display

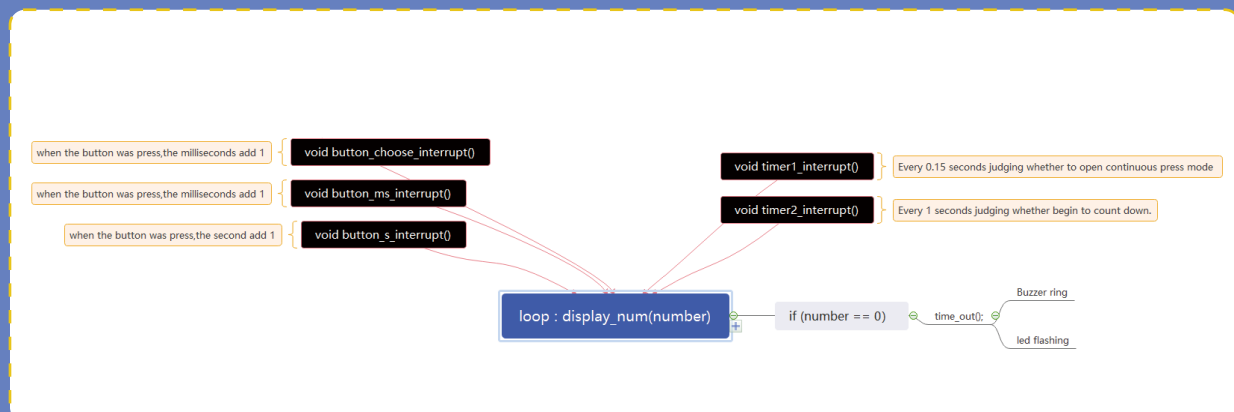
- Dynamic Scanning Display: Send font codes and corresponding bit selection to each digital display in turn, even though there is only one digit working. The principle of Persistence of Vision enables you to see all numbers displayed because each the scanning speed is so fast that you hardly notice the intervals.

Demo3:

- After wiring, please open the program in the code folder- **demo3** and click UPLOAD to upload the program.

Part 4 Creating Multifunctional Timer

Thoughts:



- After wiring, please open the program in the code folder- **MyTimer** and click UPLOAD to upload the program.