浙江大学实验报告

专业：信息工程
姓名：卢泽熙
学号：3220102478
日期：2024.6.1

课程名称：电子电路设计实验Ⅱ　　指导老师：施红军、叶险峰、邓靖靖　　成绩：
实验名称：倒计时定时器的设计、制作与调试　实验类型：设计实验　同组学生姓名：张笑瑜

目录

一、 实验目的

1、学习掌握用 Arduino UNO 设计倒计时定时器；

2、学习掌握 PCB 电路板的设计和制作；

3、学习掌握 Arduino UNO 扩展板的设计与制作；

4、学习掌握旋转编码器的使用。

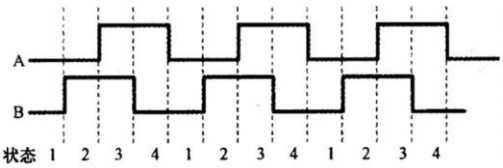
二、 实验任务与要求

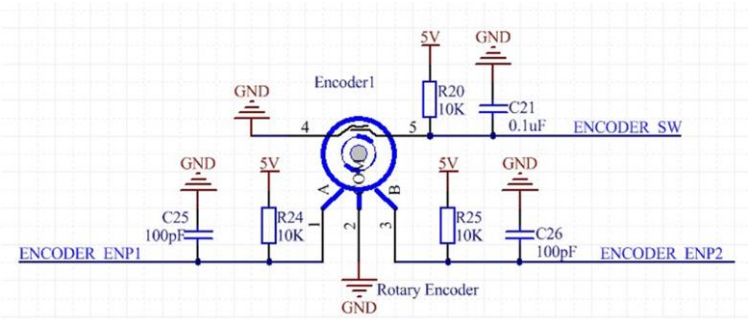
1. 用 Arduino UNO 设计倒计时定时器，要求如下：
   - 设定倒计时时间若干（设定标准时间数组），通过旋转编码器选择；
   - 时间到时报警（要求按数码管是共阴的情况设计）。
2. 设计电路，完成相应器件的选择，制作 Arduino UNO 扩展板。
3. 编制与调试倒计时定时器程序。
4. 将制作的扩展板与 Arduino UNO 板组装后，进行系统联调。

三、 实验原理

1. 旋转编码器

旋转编码器是一种数字器件，这种器件有两个输入。如左图所示，当顺时针旋转旋钮时，脉冲会变化，并且会在图中从左向右移动； 当逆时针旋转旋钮时，脉冲会在图中从右向左移动。旋转编码器接线图如右图所示。实验中可根据两个脉冲波形变化情况判断旋钮旋转方向，从而更新时间设定。

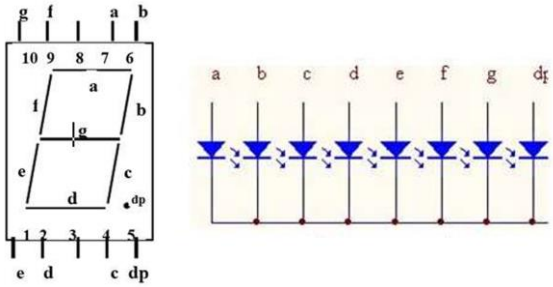




2. 数码管

（1） 7 段数码管结构

数码管一般由 8 个发光二极管组成，其中由 7 个细长的发光二极管组成数字显示，另外一个圆形的发光二极管显示小数点。当发光二极管导通时，相应的一个点或一个笔画发光。控制相应的二极管导通，就能显示出各种字符。共阴极数码管内部接法如下：



（2）驱动方式

本实验采用动态显示驱动方式。动态驱动是将所有数码管的 8 个显示笔划"a,b,c,d,e,f,g,dp"的同名端连在一起，同时为每个数码管的公共极 COM 增加位元选通控制电路，计时过程中将需要显示的数码管的选通控制打开，该位元就显示出字形，没有选通的数码管就不会亮。在轮流显示过程中，每位元数码管的点亮时间为 1～2ms；由于人的视觉暂留现象及发光二极管的余辉效应，只要扫描的速度足够快就不会产生闪烁感。动态显示的效能够节省大量的 I/O 接口，而且功耗更低。

四、 实验方案设计与实验参数计算

1. 硬件电路设计

1） 电路总体设计

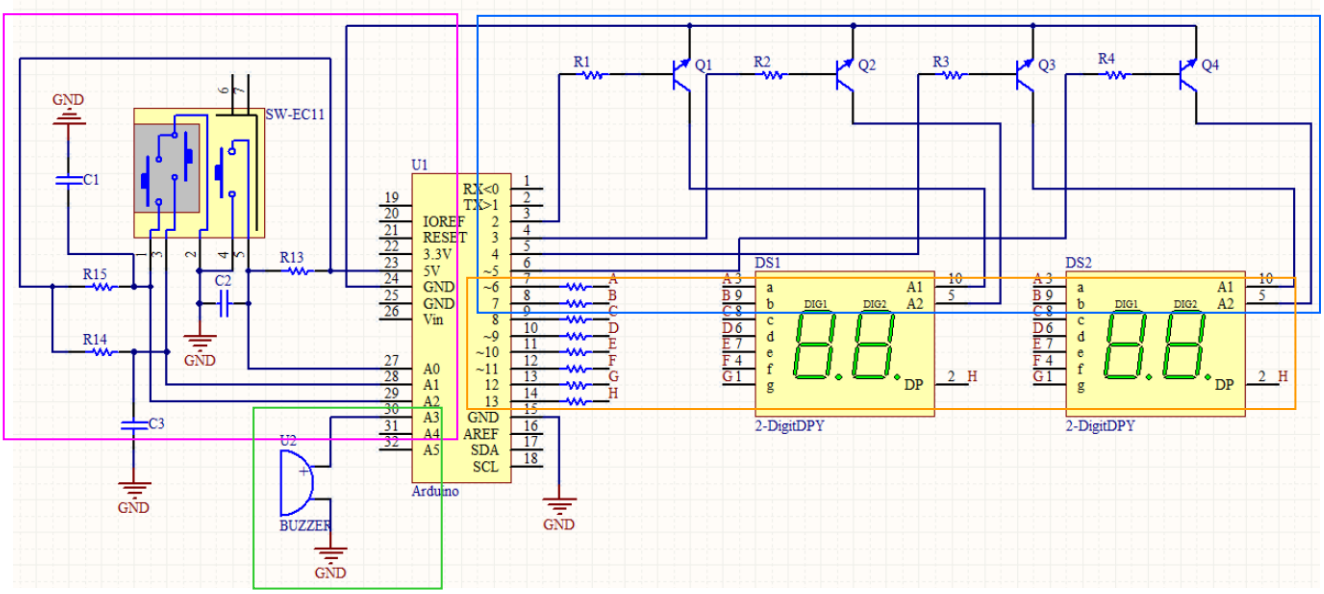
定时器总是处于三种状态之一：分钟设定、秒钟设定与定时运行。在分钟设定状态，转动旋转编码器就可以改变定时分钟；在秒钟设定状态，转动旋转编码器就可以改变定时秒钟；在定时运行状态，定时器倒计时。按下旋转编码器上的按钮可以对这三种状态进行切换。编写 Arduino 代码时引入 EEPROM 库用于保存最后使用的时间。由于只显示分和秒，共需 4 个数码管。

硬件设计方面总体可分为四个模块：

- **数码管选通控制模块：**控制四个数码管的选通情况，选通的数码管才能正常工作；
- **数码管显示模块：**控制选通的数码管显示的数字；
- **蜂鸣器模块：**控制蜂鸣器在合适的时间工作；
- **旋转编码器模块：**根据电平变化正确识别旋转编码器旋转方向。

实验选择共阴数码管，故采用 npn 型三极管并使发射极共地连接；同时因为 Arduino 板数字接口数量有限，实验中选用部分模拟接口作为数字接口使用。

2） 电路原理图



其中 $R_1 \sim R_4 = 1k\Omega$，$R_5 \sim R_{12} = 100\Omega$，$R_{13} \sim R_{15} = 10k\Omega$，$C_1 \sim C_4 = 100nF$

3） 各模块设计说明

数码管控制模块

蓝色框内为电路数码管控制模块部分，由四个三极管组成，每个三极管基极通过电阻与 Arduino UNO 板的一个数字输出接口相连，同时发射极共同接地。$Q_1 \sim Q_4$ 依次控制两个数码管从左向右四个数字。例如，Q_1 集电极接数码管 DS1 的 A1 端口，基极接 Arduino UNO 的 2 端口。当 2 端口置为高电平，A1 为低电平，此时数码管 DS1 左位数字进入显示状态，可通过控制 6-12 端口的电平来控制显示的数字；当端口 2 置为低电平，A1 为高电平，此时数码管对于任何数字位的控制输入都无法工作。其他三个数字位工作模式同理。

数码管显示模块

黄色框内为数码管显示模块，选用 8 个 Arduino 数字输出接口控制每个数码管的 8 个发光二极管；当数码管进入工作状态时，可将待点亮数字位对应连接的输入端口置于高电平，即可显示点亮的数字。如想显示 0，则 a b c d e f 需点亮，控制此时 Arduino UNO 6 ~ 11 端口输出高电平，12 端口输出低电平。

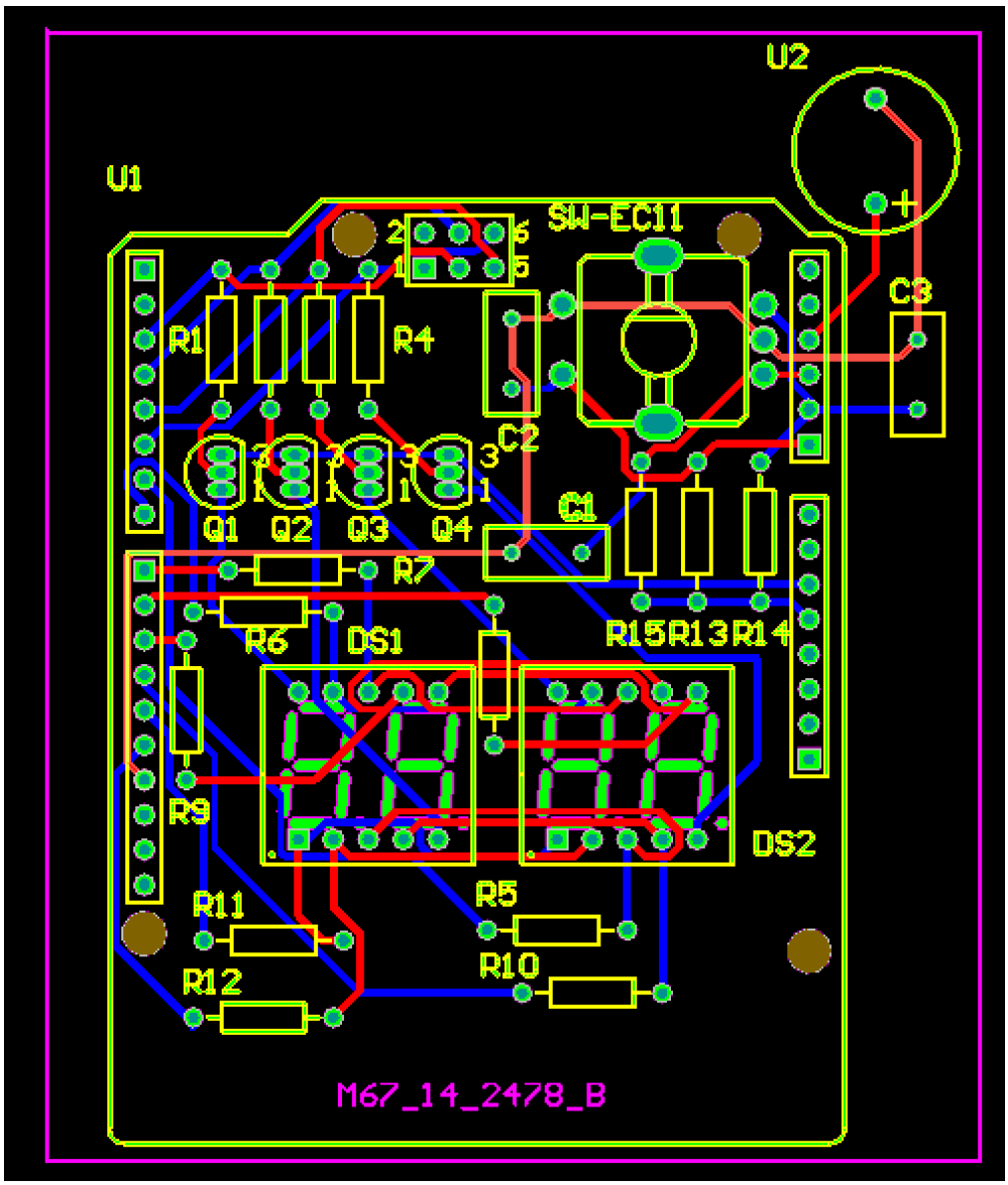
蜂鸣器控制模块

绿色框内为蜂鸣器控制模块。蜂鸣器正极连接于模拟 A3 端口，当计时结束后该端口输出合适工作电压即可控制蜂鸣器响应。

旋转编码器模块

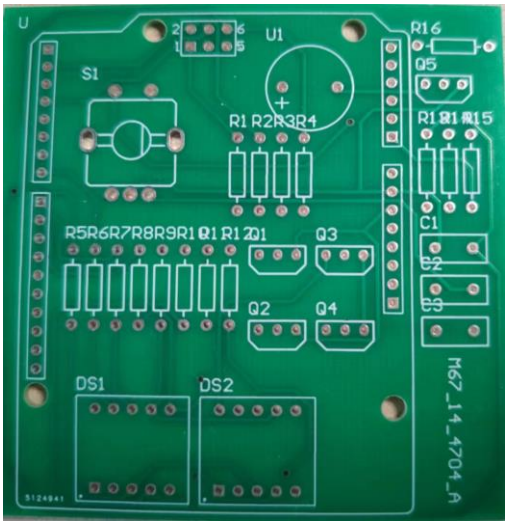
粉色框内为电路旋转编码器模块。A0 为旋转编码器按钮接口，当按下按钮时，改变定时器的状态；A1 与 A2 为旋转编码器的 a、b 信号，用于检测旋转编码器是否被旋转；Arduino 代码可根据电平的改变情况判断旋转编码器旋转方向，从而控制时间设定。

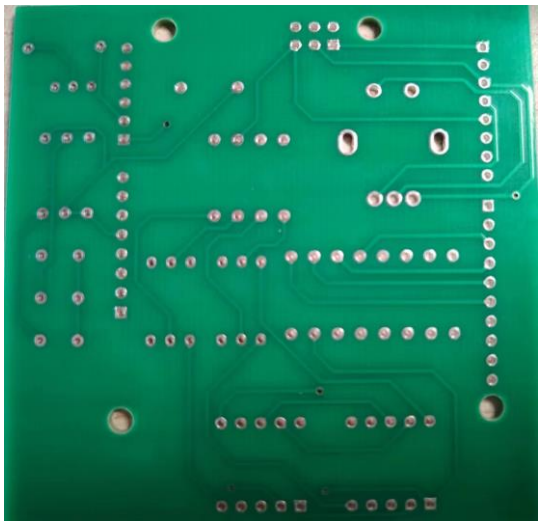
4） 电路 PCB 版图

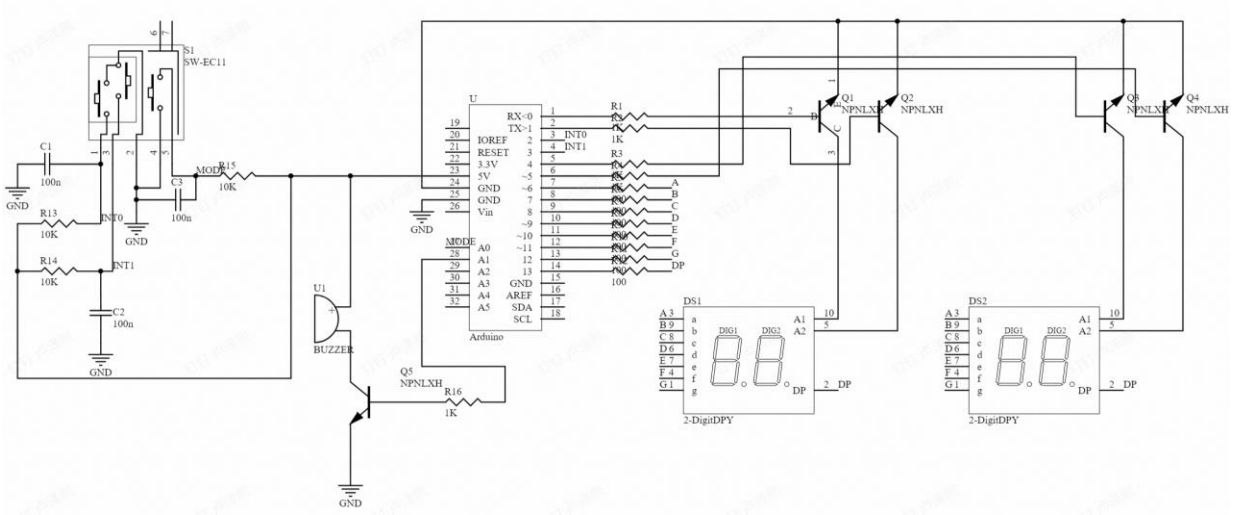


5） 设计声明

本次实验加工的 PCB 是我队友的，而上述介绍的设计是我个人的；代码部分则按照队友加工完成的电路板进行编写，与我介绍的原理图与 PCB 版图无关。队友的原理图及实际电路板如下：







2. 软件设计

1） 完整代码

本次实验我与队友两人独立设计了不同的代码，均能符合实验要求且通过验收。该部分展示的 Arduino 代码是由我设计的。**完整代码见 pdf 文件。**

2） 功能说明及创新点

1. 完成了倒计时计时功能，计时结束后蜂鸣器会报警；
2. 参考代码倒计时设定只能包含标准数组，而该设计可设定 0-100 分钟内的任意时间；
3. 时间可按分钟为步长进行调节也可按秒为步长进行调节，更符合实际需求；
4. 调节至上下界 99：59 或 00：00 后继续同方向旋转可自动进位或借位；
5. 消除了计时过程中的秒钟滴答声；
6. 计时过程中不会产生余晖现象。

3） 设计说明

（1）变量定义

```
int time = 0;
int segmentPins[] = { 6, 7, 8, 9, 10, 11, 12, 13 };  // 数码管的段pin
int displayPins[] = { 0, 1, 4, 5 };                  // 数码管的位pin

int timerMinute;
int timerSecond;
int buzzerPin = 15;
int aPin = 2;
int bPin = 3;
int buttonPin = 14;
int state = 0;   //0 代表调整分钟；1代表调整秒钟；2代表计时

byte digits[11][8] = {
  { 1, 1, 1, 1, 1, 1, 0, 0 },  //0
  { 0, 0, 0, 0, 1, 1, 0, 0 },  //1
  { 1, 1, 0, 1, 1, 0, 1, 0 },  //2
  { 1, 0, 0, 1, 1, 1, 1, 0 },  //3
  { 0, 0, 1, 0, 1, 1, 1, 0 },  //4
  { 1, 0, 1, 1, 0, 1, 1, 0 },  //5
  { 1, 1, 1, 1, 0, 1, 1, 0 },  //6
  { 0, 0, 0, 1, 1, 1, 0, 0 },  //7
  { 1, 1, 1, 1, 1, 1, 1, 0 },  //8
  { 1, 0, 1, 1, 1, 1, 1, 0 },  //9
  { 0, 0, 0, 0, 0, 0, 0, 0 }
};
```

- time 用于记录当前的时间，timeMinte 与 timeSecond
- segmentPins 和 displayPins 确定控制数码管的选通及显示情况的数字接口；
- 依次确定旋转编码器、蜂鸣器的对应接口；
- digits 数组确定不同数字显示时各 LED 亮灭情况；
- state 用于记录当前状态；

（2）setup 函数

```
void setup() {
  for (int i = 0; i < 8; i++)
    pinMode(segmentPins[i], OUTPUT);
  for (int i = 0; i < 4; i++)
    pinMode(displayPins[i], OUTPUT);

  pinMode(buzzerPin, OUTPUT);
  pinMode(buttonPin, INPUT);
  pinMode(aPin, INPUT);
  pinMode(bPin, INPUT);

  time = EEPROM.read(0);
  timerMinute = time / 60;
  timerSecond = time % 60;
}
```

该函数包括各管脚模式确定操作与时间的初始化。

（3）loop 函数

检测按钮的状态来切换定时器的运行或停止，以及调用更新显示的函数来更新数码管上显示的时间。

```
void loop() {
  if (!digitalRead(buttonPin)) {
    state++;
    state = (state > 2 ? 0:2);
    digitalWrite(buzzerPin, LOW);
    while (!digitalRead(buttonPin)) {};
    EEPROM.write(0, time);
  }
  updateDisplay();
}
```

（4）updateDisplay 函数

```
void updateDisplay()  //mmss
{
  int minsecs = timerMinute * 100 + timerSecond;
  int v = minsecs;
  for (int i = 0; i < 4; i++) {
    int digit = v % 10;
    setDigit(i);
    setSegments(digit);
    v = v / 10;
    process();
    setSegments(10);
  }
  setDigit(5);
}
```

更新数码管上显示的时间，。它计算当前剩余的分钟和秒数，将它们转换为单独的数字，然后依次激活每个数码管的显示，并设置 相应的段落以显示当前数字。显示完一个数字后，进行短暂的延迟来保持显示，然后切换到下一个数字，这样循环确保了时间的准确显示；同时，在每个 for 循环中运行 setSegments(10)可短暂使各数码管停止显示，从而在视觉上起到消除余辉的效果。

（5）process 函数

```
void process() {
  for (int i = 0; i < 100; i++)  //tweak this number between flicker and blur
  {
    int change = getEncoderTurn();
    if (state == 0)
      changeSetMinte(change);
    else if (state == 1)
      changeSetSecond(change);
    else if (state == 2)
      updateCountingTime();
  }
  if (timerMinute == 0 && timerSecond == 0)
    digitalWrite(buzzerPin, HIGH);
}
```

这是一个辅助函数，用来在定时器运行时更新时间（即倒数秒数），以及在定时器被暂停时根据编码器的输入调整倒计时的设置。它先检测编码器的状态变化，然后根据定时器当前是运行状态还是时间设定状态，以执行相应的逻辑：

- state = 0 时更新设定的分；
- state = 1 时更新设定的秒；
- state = 2 时为运行状态，每秒更新数字显示。

最后一行为蜂鸣器工作状态判断函数，当计时器归零后蜂鸣器开始报警。

（6）changeSetSecond/changeSetMinte 函数

```
void changeSetSecond(int change) {
  time += change;
 if(time < 0 && state == 1)
    time += 6040;
  timerMinute = time / 60;
  timerSecond = time % 60;
}

void changeSetMinte(int change) {
  time += change * 60;
  if(time < 0 && state == 0)
    time += 6040;
  timerMinute = (time / 60) % 100;
  timerSecond = time % 60;
}
```

这两个函数用于时间设定状态下计时的设置，分别对应分钟设置和秒钟设置。当旋转编码器顺时针旋转可能出现设定时间为负的情况，此时时间自动进位至 99：59 以符合实际设计需求。

（7）updateCountingTime 函数

```
void updateCountingTime() {
  static unsigned long lastMillis;
  unsigned long m = millis();
  if (m > (lastMillis + 1000) && (timerSecond > 0 || timerMinute > 0)) {
    digitalWrite(buzzerPin, LOW);
    delay(10);
    digitalWrite(buzzerPin, LOW);

    if (timerSecond == 0) {
      timerSecond = 59;
      timerMinute--;
    } else
      timerSecond--;
    lastMillis = m;
  }
  if (timerMinute == 0 && timerSecond == 0){
    digitalWrite(buzzerPin, HIGH);
    timerMinute = 0;
    timerSecond = 0;
  }
```

当定时器处于运行状态时，该函数用于按秒减少时间。它会记录上一次减少时间时的系统时刻，当达到 1 秒时减少秒数（delay 为 10ms 且该函数在 process 函数中执行 100 次）；如果秒数降低到 0，则相应减少分钟数并将秒数重置为 59；同时，每次更新时将蜂鸣器控制电平置为低电平以确保不会产生秒钟滴答声。

（8）setDigit/setSegments 函数

```
void setDigit(int digit) {
  for (int i = 0; i < 4; i++) {
    int on = (digit == i) ? HIGH : LOW;
    digitalWrite(displayPins[i], on);
  }
}

void setSegments(int n) {
  for (int i = 0; i < 8; i++) {
    digitalWrite(segmentPins[i], digits[n][i]);
  }
}
```

setDigit 函数用于控制数码管哪一位的显示。由于数码管是多位共享的，需要通过控制对应的脚来确定激活哪一个显示位；setSegments 函数控制数码管上的每一个段是否点亮，用于显示 0 到 9 的数字，它根据 digits 数组定义的数字模式，将正确的段设置为高电平或低电平，以完成正确数字的显示。

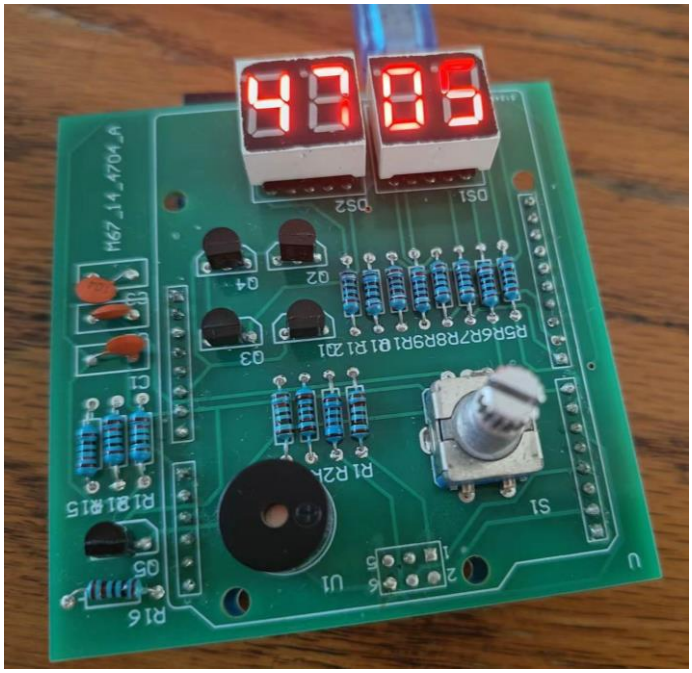
（9）getEncoderTurn 函数

```
int getEncoderTurn() {
  // return -1, 0, or +1
  static int oldA = LOW;
  static int oldB = LOW;
  int result = 0;
  int newA = digitalRead(aPin);
  int newB = digitalRead(bPin);
  if (newA != oldA || newB != oldB) {
    // something has changed
    if (oldA == LOW && newA == HIGH)
      result = -(oldB * 2 - 1);
  }
  oldA = newA;
  oldB = newB;
  return result;
}
```

该函数检测旋转编码器的旋转状态。当旋转编码器旋转时，它会返回一个值：没有旋转时返回 0，顺时针方向旋转时返回 -1，逆时针方向旋转时返回 1。这是通过读取编码器的两个输出引脚 A 和 B 的状态变化来确定的，同时每次返回结果后更新 A、B 电平的值。

五、 实验结果

实物展示如下图：



将代码上传至 Arduino 板后设定任意时间，成功实现倒计时功能，且计时过程中无滴答声出现，显示屏也为出现余辉；计时结束后蜂鸣器报警，与预期吻合。因此设计是成功的。**演示结果可见视频文件**。

六、 讨论、心得

在本次“倒计时定时器的设计、制作与调试”实验中，我学到了很多宝贵的知识和技能。这次实验使我在理论与实践之间架起了一座桥梁，也让我更加深刻地理解了硬件设计的核心概念。

首先我深刻体会到理论知识在实践中的重要性。例如，在设计倒计时定时器电路时，理解旋转编码器的工作原理和数码管的驱动方式是非常关键的；这需要我认真回顾模拟电路、数字电路理论课上学到的知识分析问题。通过此次实验，在课堂上学习的理论知识在实际操作中得到了充分的验证和应用，使我对这些概念有了更加深刻的理解。

在硬件设计部分，我遇到了不少挑战。例如，数码管的控制模块和显示模块的设计需要非常细致地考虑各个元件之间的连接和电平控制，此部分的 Arduino 代码调试也花费了我相当多的时间。通过亲身经历，我意识到在实际的电路设计中，每一个细节都可能影响到整个系统的功能和稳定性。因此，严谨和细心是电路设计中不可或缺的。Arduino 的代码编写也是本次实验的重要组成部分。编写和调试倒计时定时器程序让我感受到了软件设计的乐趣。通过编程，我可以灵活地控制硬件的运行，使整个系统更加合理。例如，本次实验中我增加了按键切换的状态，使得时间设置更加符合实际应用要求。这种将硬件和软件相结合的过程让我非常有成就感。

在实验过程中，我与同组的张笑瑜同学进行了密切的合作。我们分工明确，互相帮助，共同解决遇到的问题。通过这次合作，我深刻认识到团队合作的重要性。在团队中，每个人的贡献都是不可或缺的，只有通过默契的配合和共同努力，才能顺利完成实验任务。

当然，本次实验虽然顺利完成，但在过程中我也发现了自己的不足之处。例如，对于一些复杂电路的设计，我还需要进一步提高自己的技能和知识储备；同时，我的电路调试能力还有待加强。总的来说，本次实验不仅使我学习了倒计时定时器的设计与制作方法，还培养了我的团队合作精神和创新意识。这些宝贵的经验和心得将成为我未来学习和发展的重要基石。

七、 思考题

1. 控制定时加热

其它模块不变，引入继电器模块，用于控制加热设备的开关。使用时设定加热时间，继电器连接到 Arduino 的数字输出端口，倒计时过程中输出高电平控制继电器闭合，启动加热设备；倒计时结束后蜂鸣器报警，同时继电器对应的数字接口输出低电平，继电器断开，停止加热。

2. 消除秒钟滴答声

该实验中设计的代码不会产生秒钟滴答声，具体见软件设计部分。