周三 9、10 节

浙江大学

本科实验报告

课程名称： 电子电路实验 II

姓　　名： 李坤林

学　　院： 信息与电子工程学院

　　系：

专　　业： 信息工程

学　　号： 3200101135

指导教师： 李锡华、施红军、叶险峰

2022 年 04 月 18 日

专业：信息工程
姓名：李坤林
学号：3200101135
日期：2022 年 4 月 18
地点：东 4- 216

浙江大学实验报告

课程名称：电子电路设计实验II　　指导老师：李锡华、施红军、叶险峰　成绩：

实验名称：Arduino 项目 6 中期设计报告　实验类型：设计实验　　同组学生姓名：孙扬赫

一、实验目的

1. 学习掌握 Arduino UNO 开发板的使用并设计倒计时定时器；
2. 学习掌握 PCB 电路板的设计和制作；
3. 了解并学习 Arduino UNO 扩展板的设计；
4. 学习使用旋转编码器、数码管等电子元器件。

二、实验任务

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；
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. Arduino UNO 开发板

Arduino Uno 是一款基于 ATmega328P 的微控制器板。它有 14 个数字输入/输出引脚(其中 6 个可用作 PWM 输出)，6 个模拟输入，16MHz 晶振时钟，USB 连接，电源插孔，ICSP 接头和复位按钮。只需要通过 USB 数据线连接电脑就能供电、程序下载和数据通讯。

Power 引脚: 开发板可提供 3.3V 和 5V 电压输出，Vin 引脚可用于从外部电源为开发板供电。

Analog In 引脚： 模拟输入引脚，开发板可读取外部模拟信号，A0~A5 为模拟输入引脚。

Digital 引脚：UNO R3 拥有 14 个数字 I/O 引脚，其中 6 个可用于 PWM(脉宽调制)输出。数字引脚用于读取逻辑值(0 或 1)，或者作为数字输出引脚来驱动外部模块。标有"~"的引脚可

产生 PWM。

TX 和 RX 引脚： 标有 TX(发送)和 RX(接收)的两个引脚用于串口通讯。其中标有 TX 和 RX 的 LED 灯连接相应引脚，在串口通讯时会以不同速度闪烁。

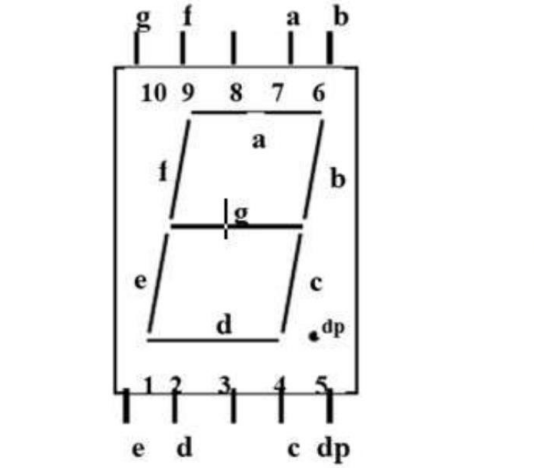
13 引脚：开发板标记第 13 引脚，连接板载 LED 灯，可通过控制 13 引脚来控制 LED 灯亮灭。一般拿到开发板上电板载灯都会闪烁，可辅助检测开发板是否正常。

2. LED 时间显示模块

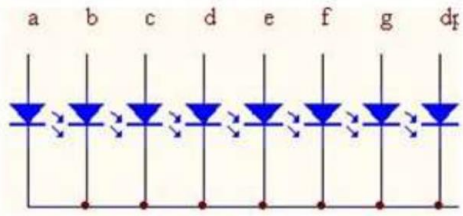
(1) 7 段数码管结构

7 段数码管一般由 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。

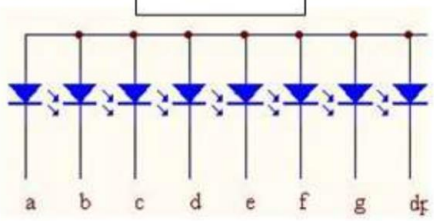
当发光二极管导通时，相应的一个点或一个笔画发光。控制相应的二极管导通，就能显示出各种字符，尽管显示的字符形状有些失真，能显示的字符数量也有限，但其控制简单，使有也方便。发光二极管的阳极连在一起的称为共阳极数码管，阴极连在一起的称为共阴极数码管，如图所示。



7 段数码管内部字段 LED 和引脚



共阴极



共阳极

(2) 7 段数码管驱动方法

发光二极管（LED 是一种由磷化镓（GaP）等半导体材料制成的，能直接将电能转变成光能的发光显示器件。当其内部有一一电流通过时，它就会发光。

7 段数码管每段的驱动电流和其他单个 LED 发光二极管一样，一般为 5～10mA；正向电压随发光材料不同表现为 1.8~2.5V 不等。

7 段数码管的显示方法可分为静态显示与动态显示，下面分别介绍。

静态显示驱动：也称直流驱动。静态驱动是指每个数码管的每一个段码都由一个单片机的 I/O 脚进行驱动，或者使用如 BCD 码二-十进位计数器进行驱动。

静态驱动的优点是编程简单，显示亮度高，缺点是占用 I/O 脚多，如驱动 5 个数码管静态显示则需要 5×8=40 根 I/O 脚来驱动。

动态显示驱动：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，另外为每个数码管的公共极 COM 增加位元选通控制电路，位元选通由各自独立的 I/O 线控制，当单片机输出字形码时，所有数码管都接收到相同的字形码，但究竟是那个数码管会显示出字形，取决于单片机对位元选通 COM 端电路的控制，所以我们只要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。

透过分时轮流控制各个 LED 数码管的 COM 端，就使各个数码管轮流受控显示，这就是动态驱动。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，尽管实际上各位数码管并非同时点亮，但只要扫描的速度足够快，给人的印象就是一组稳定的显示资料，不会有闪烁感，动态显示的效果和静态显示是一样的，能够节省大量的 I/O 埠，而且功耗更低。

由于时间显示分和秒，只需 4 个数码管，可考虑两个 2 位 7 段发光 LED,封装和引脚图如下图所示。

3. 蜂鸣器模块

报警电路可采用蜂鸣器。蜂鸣器按工作原理可分为压电式及电磁式的二大类：压电式蜂鸣器主要由多谐振荡器、压电蜂鸣片、阻抗匹配器及共鸣箱、外壳等组成。它是以压电陶瓷的压电效应，来带动金属片的振动而发声；电磁式的蜂鸣器，则是用电磁的原理，通电时将金属振动膜吸下，不通电时依振动膜的弹力弹回。

蜂鸣器按是否含有驱动，又分为有源蜂鸣器和无源蜂鸣器两种。“有源”是指蜂鸣器本身内含驱动了，直接给它一定的电压就可以响；“无源”是需要靠外部的驱动才可以响的。注意，这里的“源”不是指电源。而是指振荡源。也就是说，有源蜂鸣器内部带振荡源，所以只要一通电就会叫。而无源内部不带振荡源，所以如果用直流信号无法令其鸣叫，要用 2k~5k 的方

波去驱动它。无源蜂鸣器的优点是便宜、声音频率可控（可以做出“多来米发索拉西”的效果）以及在一些特例中，可以和 LED 复用一个控制口。而有源蜂鸣器的优点在于程序控制方便。

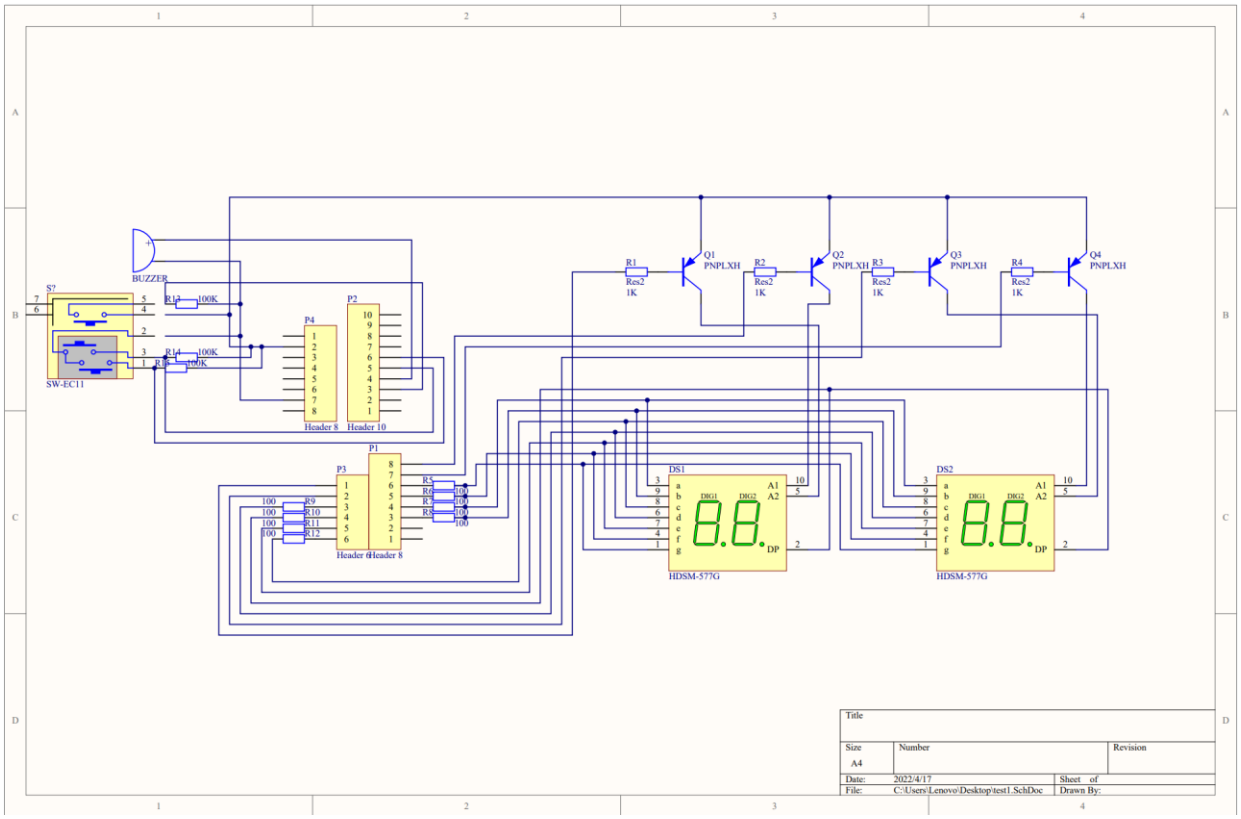
4. 旋转编码器

旋转编码器是一种数字器件，它有着两个输出 A 和 B，当旋转时，输出 A、B 会有变化，通过变化的不同我们可以判断出旋转编码器是顺时针旋转还是逆时针旋转，如此便可利用旋转编码器控制倒计时的时间。

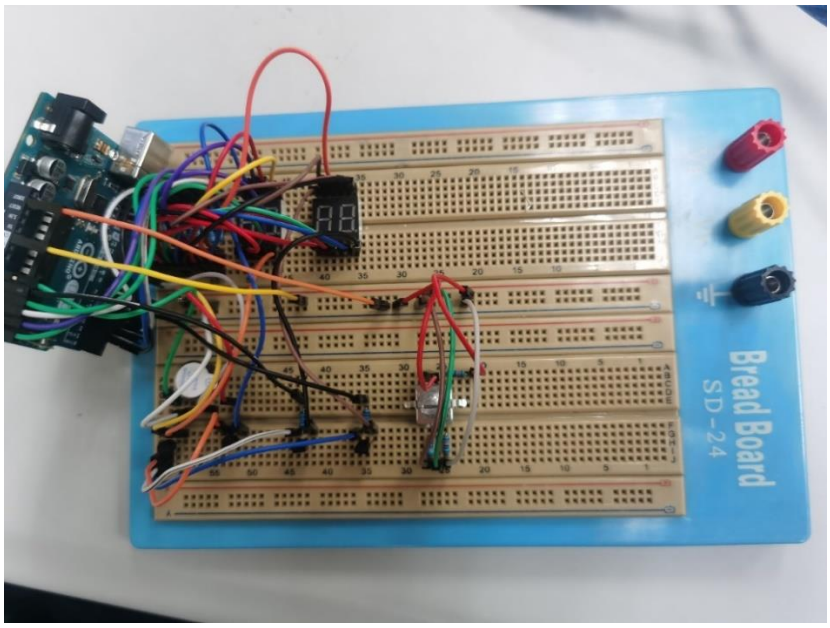
四、电路原理图设计

Arduino UNO R3，EC11 型旋转编码器，两个两位七段数码管，四个三极管，电阻若干。基本全部按照原设计进行操作，并进行了小的调整：改变三极管接法、调整了接口顺序等。

1. 电路图

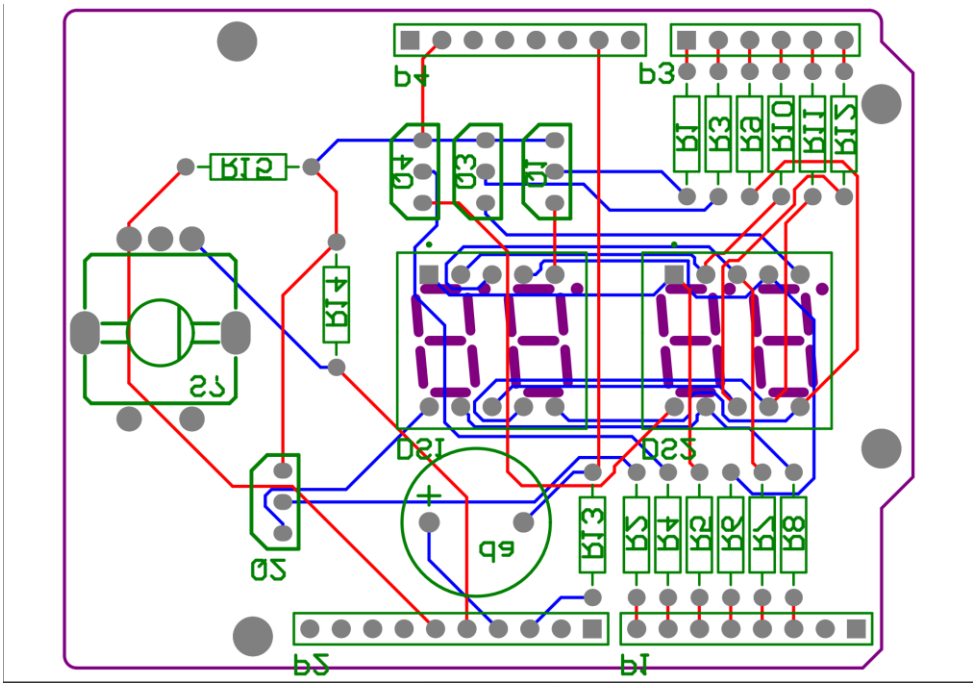


2. 面包板调试



面包板调试时，出了很多岔子。实验给出的参考例子使用的是共阳接法，在第一次搭建面包板的时候，我们依旧按照原来的方式进行组装，但出现的数字与预期大大相反，甚至难以找到规律。在仔细检查电路结构的时候最终才确定是三极管部分出的问题。在更换三极管并按照正确方式接入后，虽然数字显示正确，但顺序完全是混乱的。所以又经过一次检查后，应该是后面的电阻和三极管的接线顺序不对，在更换了三极管之后，问题得以解决。遂按照原理图绘制 PCB 板图。

五、PCB 板图绘制

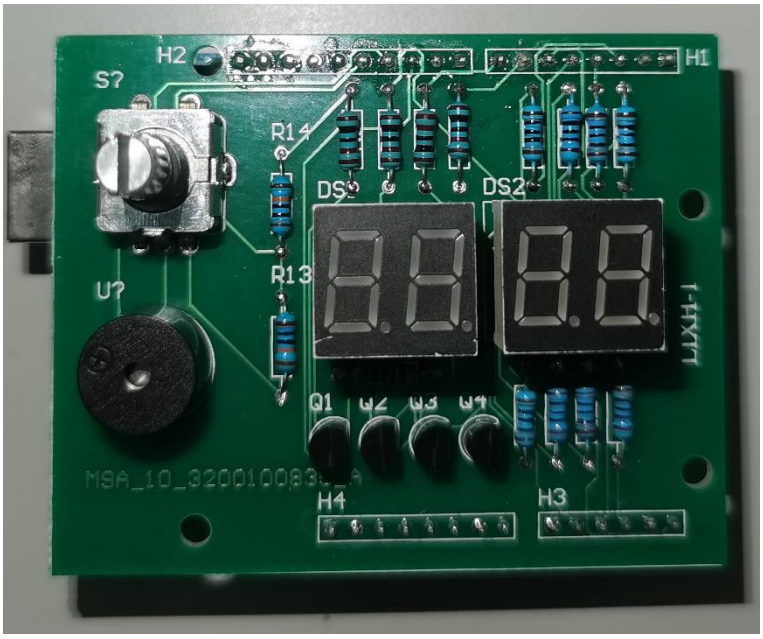


按照原理图对 PCB 板进行绘制。将 Arduino 板上的四个引脚端口与原理图中的排针端口一一对应，修改编号后自动进行覆盖连接。在手动布置好飞线位置和元件位置后，选择自动布线生成板图。再在自动布线的基础上手动进行修改，得到如上的 PCB 图。

六、焊接调试

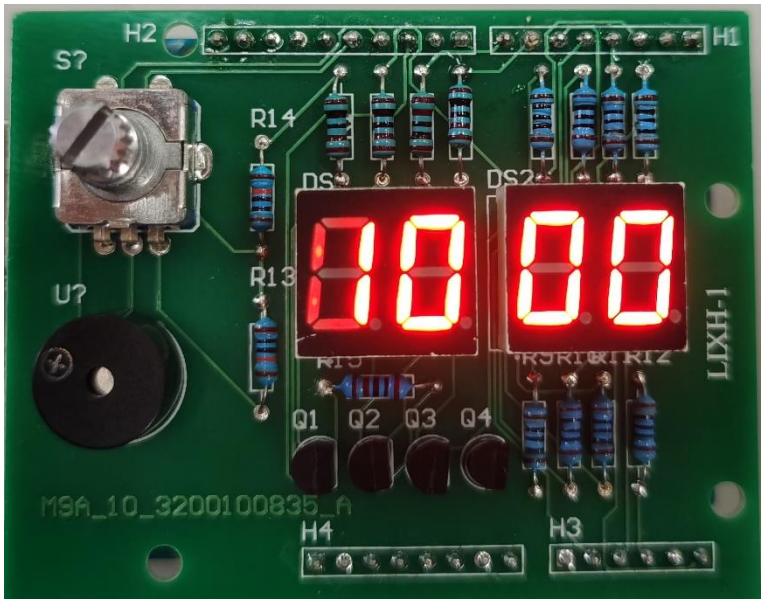
1. 焊接

按照电路设计选取元件，按高度从低到高的原则将其焊接到 PCB 板上。



2. 测试

将附录中的代码烧录入 Arduino 板中，效果如下：



- 尝试拧动旋转编码器，测试是否能正常改变设定时间；
- 按下旋转编码器，测试是否能正常倒计时，倒计时结束后能否正常发出报警；
- 设定时间后，将 Arduino 断电，再次上电测试是否能记住上次设定时间；

功能正常，但有不足之处

3. 调试问题

数码管显示有抖动，旋转编码器反转不能减小数字显示。

解决方案：

将代码中 *process* 函数中的循环次数由大调小，寻找何时次数，大约在 50-60 之间比较合适。

但旋转编码器的反向旋转在经过检查电路与连接时，并没有发现问题。在询问老师后也没有明确的答案。后来在查询 Arduino 文档时，怀疑是旋转编码器所连接的 13 号引脚因为连接了自带的 Led，可能会对旋转编码器的供能造成影响。但在重新连接电路并将引脚修改为 8 号引脚后，问题并没有得到解决。但此问题不影响正常使用。

七、思考题

1. 将图 4.15 中的数码管换成共阴的，分别说明电路和程序要做怎样的修改？

电路图可以不做修改，保持原有线路不变。

程序需要改动两个地方：

第一处将 byte digits[10][8]中的 1 和 0 互换；二则是将 *setDigit* 函数中 digitalWrite 第二个参数中的不等号换为等号。

2. 考虑倒计时定时器的应用，例如如何控制定时加热，简要介绍实现的电路。

原理与本设计中使用蜂鸣器的原理相似，我们需要另外找一个数字输出端口，寻找与此输出端口相匹配的保险丝或空气开关并连接，另一端进行接地处理。时间倒数结束后，数字输出端口使其持续输出高电平即可。

3. 如何通过修改程序，消除秒钟滴答声。

删除 *updateCountingTime* 函数中的 delay(10)：

八、讨论总结与体会

心得体会：

在前期电路设计过程中，其实也遇到了不少的问题，因为以前没有设计过 Arduino 拓展板，所以在最开始拿到题目的时候陷入了迷茫之中，不知道从何而起。在反复读了参考资料之后，才渐渐有了眉目。因为老师的资料中给了电路图，我们设计电路的时候困难较少，但是当进入面包板调试过程后，我们发现事情并不像自己想象中的那么顺利，出现了以下几个问题：首先，七段数码管和 A4 那个 pdf 中的数码管封装不同。拿到器件之后，我们不知道怎么去接线，但通过互联网查找到了该型号数码管的器件手册（后面发现老师给了器件手册截图），事情变得顺利起来了。其次，旋转编码器给我留下了深刻的印象，因为不懂旋转编码器的工作原理，我们就照猫画虎地按照图上接线，最后在测试功能的时候发现拧动编码器无法改变计时的多少。

在设计电路并进行面包板调试时，因为电子电路设计实验二是开放实验，因为以前没有做过类似的实验，出现了各种各样的问题。但是在解决问题的过程中，我也很明显地感受到了自己的成长。实践是最能够检验自己所学的途径，在实践的过程中，每解决一个问题都能够获得一股巨大的成就感，这种成就感也能激励着我继续学习下去。同时，在实践中才能够更加透彻地理解各个器件地使用方法，就如数码管的驱动方式，不进行实验始终觉得动态驱动很虚无飘渺，但实验后才会发现这的确能够正确地运行。

调试过程中，除了焊接三极管时需要注意不要将三个电极焊在一起了，焊接总体而言问题不大。在调试过程中，数码管出现了闪烁的情况，极其影响观感，在研究了代码之后，发现是 *process* 函数中的循环次数过大，但是也不能改的过小，改得太小也会导致数码显示管出现模糊的情况，最终在反复尝试之后将循环次数确定在了 50 次，使得模糊和闪烁之间取得了一个平衡。

在这次的实验设计工作中，我学到的更多的是电路设计的方法，跟着老师给的大致方案，一步一步地将一个能够实现具体功能的电路构建好，并在面包板上实现。虽然其中设计的部分偏少，更多地是跟着老师的方案把东西做出来，但是这个过程也让我管中窥豹，知道了电路设计的大致流程，也在一定程度上培养了对于电路设计的兴趣，以及自己的动手能力。其次，在这个过程中也学会了 AD 软件的基本操作，懂得了利用 AD 软件绘制 PCB 版图的过程。

总的来说，正是在一次又一次的纠错、一次又一次的实践过程当中，我们的专业素养才会不断地培养和提高。通过这次设计实验，我们很好地提高了自己的自学能力和规划能力，既学会了 AltiumDesigner 软件的使用，也掌握了 PCB 和 Arduino 扩展板的设计方法，更了解

了如何利用 ArduinoUNO 设计数字时钟，以及各种功能模块的使用方法。“万事开头难”，但永远不要被开头给难住。俗话说，“师傅领进门，修行靠个人”，在实践的过程中去探索、理解，边做边学、边学边做，渐渐就能够领悟要领，而这也是对我们学习能力的一种锻炼和考验。面对夏学期的电路装配、调试与验收实验，我现在充满着期待。相信，这一系列的设计与装配实验，能够让我们在专业核心素养的道路上，不断提高。

思政思考：

在整个电设实验流程当中，最让我印象深刻的就是 EDA 软件的使用了。但目前我们使用的软件大多数都是国外产品，我们国家自己的 EDA 产业虽有发展，但仍未达到国外同期水准。而 EDA 贯穿于整个集成电路产业链的工业软件系统，它处在产业链最上游、最基础的位置，在以上所提及的集成电路产业链各环节中，EDA 均起到重要的战略支撑作用。此外，EDA 工具的发展完善，非常需要和产业链上下游进行充分配合、迭代并进，共同打造全面的应用生态环境。因此，在近年来一系列贸易摩擦事件中，EDA 工具均被列入禁运名单，对我国的信息安全和集成电路产业安全造成严重负面影响。可以说，大力解决 EDA“卡脖子”问题已成为我国 IC 产业持续发展完善的一项重大任务。

虽然这几年我国国产 EDA 获得了长足进步，但与发展成熟、技术领先的美国 EDA 行业相比，依然有不小差距，具体表现在：

产业生态：国产 EDA 对先进工艺的支撑不足，除少数点工具外，大多数国产 EDA 目前仅能覆盖到 28nm 工艺，而目前国际上已商用的最先进工艺已达 5nm。这是产业链支撑存在障碍，需要扭转被动局面。

产品应用：国产 EDA 面临的应用环境较为恶劣。一方面大部分设计公司长期使用国外软件，已经形成使用习惯，不愿意切换；另一方面国内对知识产权保护意识较为淡薄，很多 IC 企业和从业者使用盗版进口 EDA，从而导致国产 EDA 减少了实际应用中迭代改进的机会。

资金和人才：据统计，过去十年，中国国产 EDA 企业研发投入总额仅相当于某国际龙头 EDA 企业一个季度的研发费用，国产 EDA 开发者总人数不足 1500 人，也不及某国际龙头 EDA 企业的总人数的十分之一。

当前，在国际局势波谲云诡、加剧动荡、中美贸易摩擦、战略脱钩的态势愈演愈烈的大背景下，快速行动起来、保障我国网络信息自主安全刻不容缓。曾经作为 IC 产业幕后英雄的 EDA 软件正逐步走向台前，引发了集成电路产业乃至全社会的极大关注和积极投入。我们看到，虽然由于一些历史原因，国产 EDA 相比美国 EDA 整体仍有一定差距，但随着政策、资

本、人才等各种资源要素的迅速投入，国产 EDA 正在迅速发展壮大，各环节都出现了一批能够替代进口软件的国产方案。希望主管机构能够给与政策牵引和支持，推进国产 EDA 在市场端的应用。

在 20 年初，美国商务部宣布新计划，通过修改出口管制规定，试图联合全球供应链厂商向华为展开新一轮“绞杀”，以使华为的各种芯片相关设计和生产工作全部受控于美国管制。这一举可谓“釜底抽薪”，势必迫使 EDA 国产替代步伐再度加快，以抵抗此类“技术霸权”带来的消极影响，让本土领先企业在技术自信、市场公平、合作共赢的生态下获得更快发展。

展望未来，随着全球 IC 产业进入“后摩尔时代”，传统的 IC 设计方法的发展陷入平缓期，而恰逢我国加快“新基建”的重要时期，5G、人工智能、大数据、新能源汽车等新兴产业对 IC 设计方法提出了全新的技术要求，这给国产 EDA 追赶提供了绝佳机遇。因此，我们需要大力发展国产 EDA， 彻底解决我国网信产业的这一安全隐患。

而我们作为青年学生，学习专业恰好是与 EDA 产业最贴近的 EE 学科，所以我们应该相应国家号召，迎难而上，学号基础知识，努力为解决这一卡脖子问题做出自己的贡献。

九、Arduino 代码

根据实际情况，我们基于原代码，针对共阴接法和反装数码管的设计进行了小的修改。

```
#include<EEPROM.h>
int segmentPins[] = {3, 2, 19, 16, 18, 4, 5, 17};
int displayPins[] = {15, 6, 14, 7};
int times[] = {5, 10, 15, 20, 30, 45, 100, 130, 200, 230, 300, 400, 500, 600,
 700, 800, 900, 1000, 1500, 2000, 3000};
int numTimes = 19;
byte selectedTimeIndex;
int timerMinute;
int timerSecond;
int buzzerPin = 11;
    int aPin = 12;
    int bPin = 13;
    int buttonPin = 10;
    boolean stopped = true;
    byte digits[10][8] = {
      //a b c d e f g .
      {1, 1, 1, 1, 1, 1, 0, 0}, //0
      {0, 1, 1, 0, 0, 0, 0, 0}, //1
```

```
    {1, 1, 0, 1, 1, 0, 1, 0}, //2
    {1, 1, 1, 1, 0, 0, 1, 0}, //3
    {0, 1, 1, 0, 0, 1, 1, 0}, //4
    {1, 0, 1, 1, 0, 1, 1, 0}, //5
    {1, 0, 1, 1, 1, 1, 1, 0}, //6
    {1, 1, 1, 0, 0, 0, 0, 0}, //7
    {1, 1, 1, 1, 1, 1, 1, 0}, //8
    {1, 1, 1, 1, 0, 1, 1, 0} //9
};
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);
  selectedTimeIndex = EEPROM.read(0);
  timerMinute = times[selectedTimeIndex] / 100;
  timerSecond = times[selectedTimeIndex] % 100;
}
void loop()
{
  if (!digitalRead(buttonPin))
  {
    stopped = !stopped;
    digitalWrite(buzzerPin, LOW);
    while (!digitalRead(buttonPin)) {};
    EEPROM.write(0, selectedTimeIndex);
  }
  updateDisplay();
}
void updateDisplay() //mmss
{
  int minsecs = timerMinute * 100 + timerSecond;
  int v = minsecs;
  for (int i = 0; i < 4; i++)
  {
```

```
62.         int digit = v % 10;
63.         setDigit(i);
64.         setSegments(digit);
65.         v = v / 10;
66.         process();
67.       }
68.       setDigit(5);// all digits off to prevent uneven illumination
69.     }
70.     void process()
71.     {
72.       for (int i = 0; i < 100; i++) //tweak this number between flicker and b
    lur
73.       {
74.         int change = getEncoderTurn();
75.         if (stopped)
76.         {
77.           changeSetTime(change);
78.         }
79.         else
80.         {
81.           updateCountingTime();
82.         }
83.       }
84.       if (timerMinute == 0 && timerSecond == 0)
85.       {
86.         digitalWrite(buzzerPin, HIGH);
87.       }
88.     }
89.     void changeSetTime(int change)
90.     {
91.       selectedTimeIndex += change;
92.       if (selectedTimeIndex < 0)
93.       {
94.         selectedTimeIndex = numTimes;
95.       }
96.       else if (selectedTimeIndex > numTimes)
97.       {
98.         selectedTimeIndex = 0;
99.       }
100.      timerMinute = times[selectedTimeIndex] / 100;
101.      timerSecond = times[selectedTimeIndex] % 100;
102.    }
103.    void updateCountingTime()
104.    {
```

```
105.     static unsigned long lastMillis;
106.     unsigned long m = millis();
107.     if (m > (lastMillis + 1000) && (timerSecond > 0 || timerMinute > 0))
108.     {
109.       digitalWrite(buzzerPin, HIGH);
110.       delay(10);
111.       digitalWrite(buzzerPin, LOW);
112.       if (timerSecond == 0)
113.       {
114.         timerSecond = 59;
115.         timerMinute--;
116.       }
117.       else
118.       {
119.         timerSecond--;
120.       }
121.       lastMillis = m;
122.     }
123.   }
124.   void setDigit(int digit)
125.   {
126.     for (int i = 0; i < 4; i++)
127.     {
128.       digitalWrite(displayPins[i], (digit == i));
129.     }
130.   }
131.   void setSegments(int n)
132.   {
133.     for (int i = 0; i < 8; i++)
134.     {
135.       digitalWrite (segmentPins[i], digits[n][i]);
136.     }
137.   }
138.   int getEncoderTurn()
139.   {
140.     // return -1, 0, or +1
141.     static int oldA = LOW;
142.     static int oldB = LOW;
143.     int result = 0;
144.     int newA = digitalRead(aPin);
145.     int newB = digitalRead(bPin);
146.     if (newA != oldA || newB != oldB)
147.     {
148.       // something has changed
```

```
149.       if (oldA == LOW && newA == HIGH)
150.       {
151.         result = -(oldB * 2 - 1);
152.       }
153.     }
154.     oldA = newA;
155.     oldB = newB;
156.     return result;
157.   }
```