项目概述

•目标：DIY 一个智能插座的工程原型机。
•实施场景：在智能插座的研发过程中。
•时间安排：
（1）电路设计（当日）——认知设计方案，了解电路结构。
（2）手工焊接 PCB（当日和第 2 次课）——结合自身的焊接技能，关注并选择适应于研发过程的电装（即电子装配）工艺，注重硬件纠错测试中对元器件的保护。
（3）电路模块调试（第 3 次课）——了解系统开发中的调试方法，关注对软、硬件设计相结合。
的系统调测方法的认知。
（4）系统测试（第 3 和第 4 次课）——了解系统测试方法，注重对整系统的功能验证与纠错。

第一部分：智能插座 DIY 电装

结合开发调试的电装过程
•首先，从结合调试的电装角度来理解电路结构——按功能分块；其次，按块分步安装并分块调试。
•分模块电装调试的具体过程：
（1）分清供电模块；分清输入与输出通道模块；
（2）遵循先供电模块，后用电模块的安装步骤。在供电模块安装完成后必须调试供电电路的各项参数，如电压、纹波等，避免因供电电路的故障，造成对后续安装电路的影响，甚至损坏！
（3）较便宜的器件先装，较贵重的器件后装。
（4）各模块的焊接从其输入部分开始，直至其输出
部分，逐级安装，逐级调试；
（5）为稳妥起见，强调采用安装一部分，调试一部分的谨慎法则。

小结内容为在完成了全部电装任务的基础上，总结自己的电装经验，具体如下：
（1）在报告中列出你的电装元器件顺序。
（2）列出你在上述电装期间遇到的各类问题。
（3）分析问题并给出纠错解决方案与解决后的效果。
（4）之前的各测试数据记录表以及一些问题解答等等

测试一、供电电路测试

完成供电电路模块后，可进行一次分块测试，以便确定供电电路的设计与电装是否正确

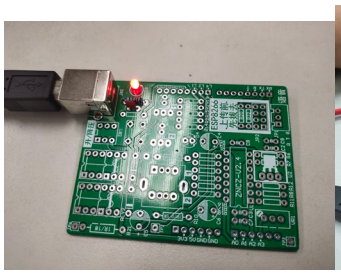
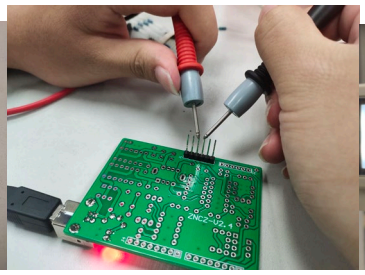
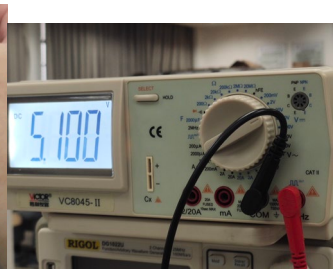
（1）插上外接 5V 电源，“D3”会亮。

（2）用万用表的直流电压档，检查 J1 排针上的第 3 脚（标注 5V）与 4 或 5 脚（注 GND）之间的 5V 电压是否正确（注意是哪个排针的哪两个引脚！）

（3）用万用表的直流电压档测量 USB 电压是否与外接 5V 电源电压一致，即约 5V。测量时注意找准电路系统的地线端 GND（TP3）接电压表的负极表棒。

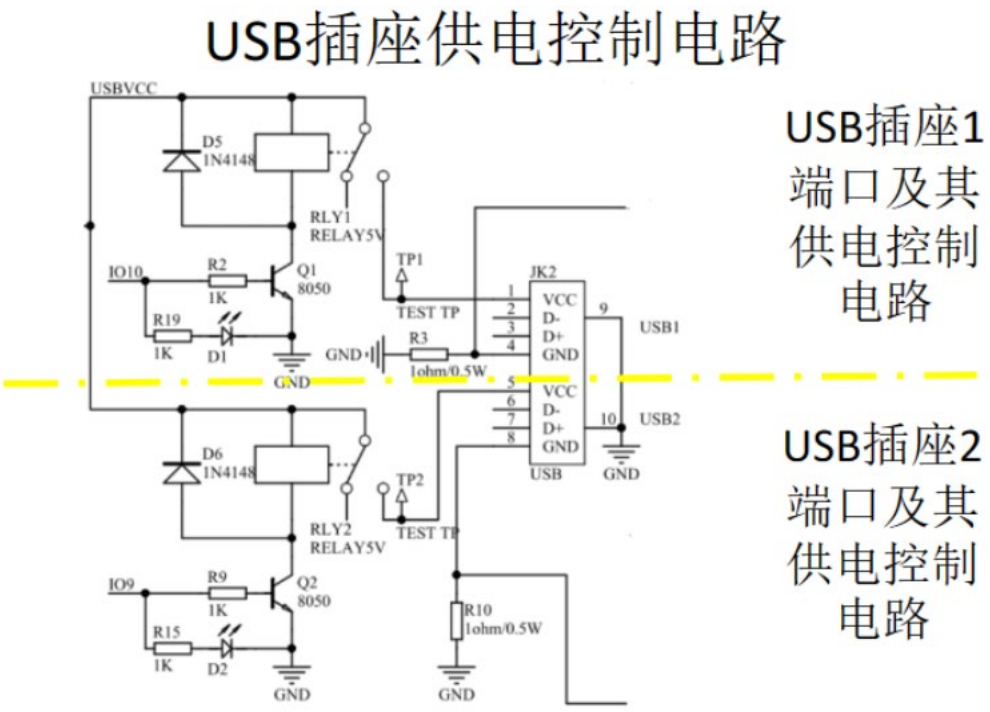
检测项目	检测结果
电源空载时的输出电压：	5.050V
插上电源后 LED3 状态（亮/灭）：	亮
插上电源后，标注 5V 处（J1 的 3 脚）的电压（即以万用表直流压档测量标注 5V 处对地线 GND 的电压）：	5.100V

测试过程顺利进行，测试结果正常，符合预期。说明供电电路部分功能正常。

分析一、USB 供电电路信号分析

自行分析该供电通路的信号路径，并写入总结报告中



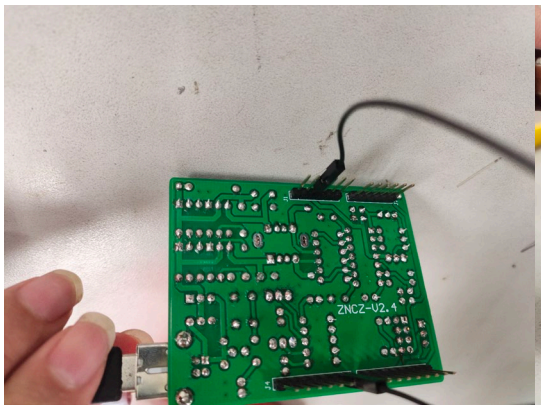
分析：
USB2 的电路通道：引脚 IO9 通过高/低电平（即+5V/0V）的输入，可控制三极管 Q2 的导通/截至，同时控制指示灯 D2 的亮灭来表示高低电平的输入。从而 Q2 的导通/截止决定继电器 RLY2 的通断，最终决定 USB 供电插座 JK2 上的 USB2 口是否有 5V（即 USBVCC）电源输出。供电电流将从 USB2 口的 VCC 引脚流出，经过外接用电器，从 USB2 口的 GND 流回。电阻 R10 和后续的测量功能有关。电阻 R9 和 R15 起保护功能

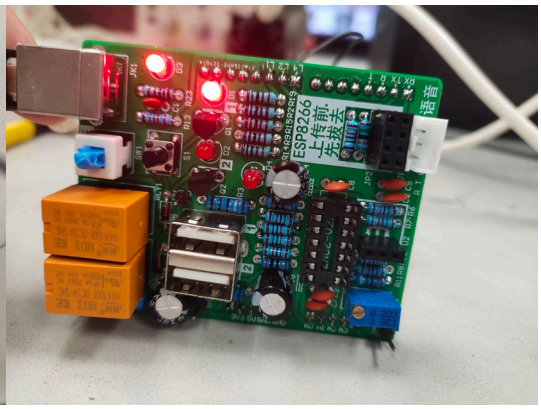
测试二、USB 插座供电控制电路测试

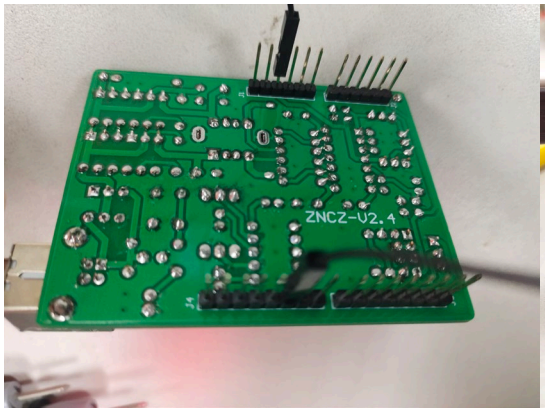
•测试插座供电输出控制功能是否完备：首先连接一根杜邦线接到 J1 排针的第 3 脚，我们的目的就是将杜邦线的一端连上 5V。
•测试 USB 供电插座 JK2 的 USB2 口的输出控制：将原先连接 J4 排针第 3 脚的杜邦线一端连接到 J4 排针的第 2 脚（即 L2）。我们的目的就是把该引脚接上 5V，以便所控制的另一路继电器导通。接通电源后，如果 D2 会被点亮，且听到继电器 RLY2 发出“卡塔”一声触点吸合的声音，则表明 USB 供电插座 2 的开关控制电路无误。
•测试 USB 供电插座 JK2 的 USB2 口的输出控制：将原先连接 J4 排针第 3 脚的杜邦线一端连接到 J4 排针的第 2 脚（即 L2）。我们的目的就是接上 5V，以便所控制的另一路继电器导通。接通电源后，如果 D2 会被点亮，且听到继电器 RLY2 发出“卡塔”一声触点吸合的声音，则表明 USB 供电插座 2 的开关控制电路无误。

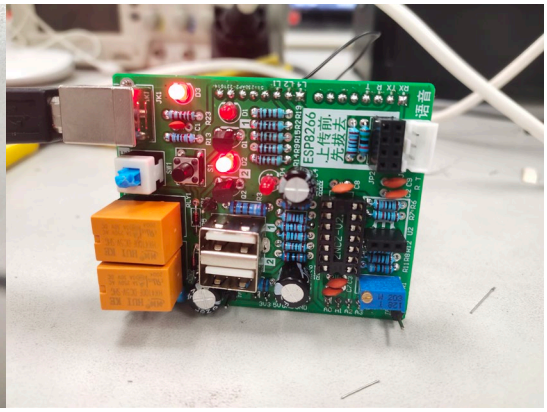
检测项目	检测结果
以杜邦线连接标注 L1 处（J4 的脚至标注 5V（J1 的 3 脚），观察到的现象：	LED1（亮/灭）__亮__，继电器 RLY1（吸合/断开）____吸合____，USB 供电插座 1 的供电电压（即 TP1）___4.957_____V。
以杜邦线连接标注 L1 处（J4 的 3 脚）至标注 GND（J1 的 4 或 5 脚），观察到的现象：	LED1（亮/灭）_灭___，继电器 RLY1（吸合/断开）__断开_____，USB 供电插座 1 的供电电压（即 TP1）__0_____V。
以杜邦线连接标注 L2 处（J4 的 2 脚）至标注 5V（J1 的 3 脚），观察到的现象：	LED2（亮/灭）__亮__，继电器 RLY2（吸合/断开）___吸合_____，USB 供电插座 2 的供电电压（即 TP2）____4.976____V。
以杜邦线连接标注 L2 处（J4 的 2 脚）至标注 GND（J1 的 4 或 5 脚），观察到的现象：	LED2（亮/灭）__灭__，继电器 RLY2（吸合/断开）____断开____，USB 供电插座 2 的供电电压（即 TP2）____0____V。

测试过程顺利，过程中未发生问题。测试结果正常，符合预期。







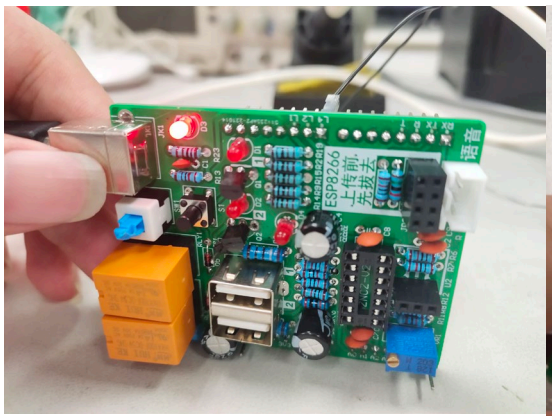
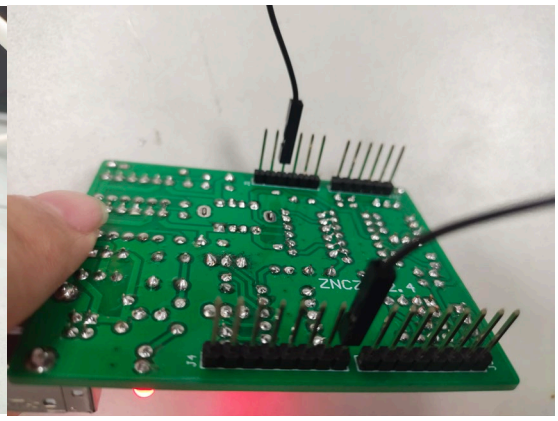
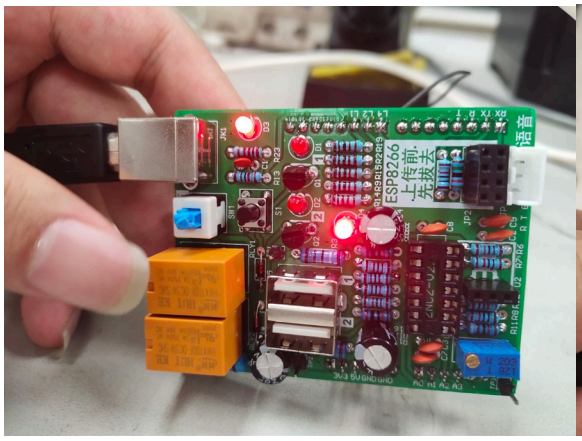
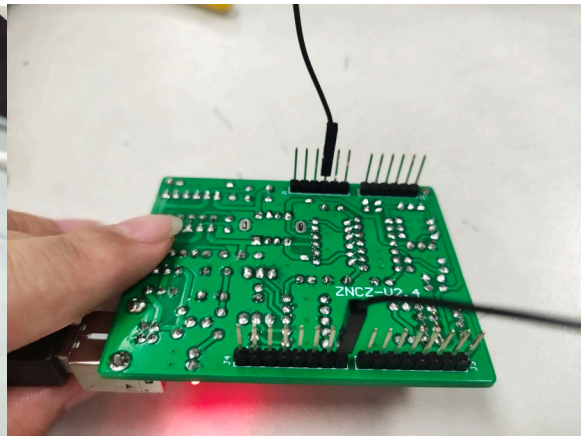
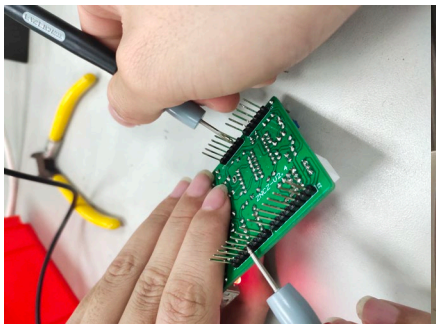
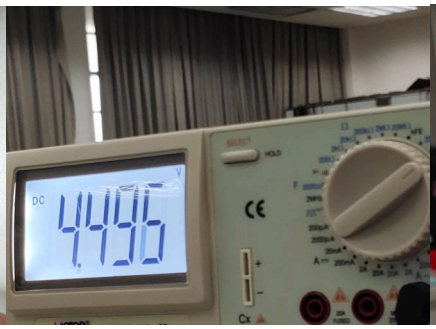
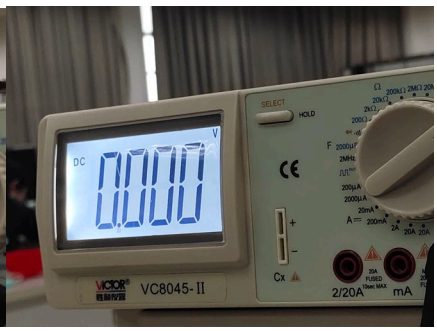


测试三、指示灯与按钮电路测试

将表三测试结果记录于总结报告中。如测试中发现问题，请分析原因，并描述解决过程（方法）。

检测项目	检测结果
电路上电，以杜邦线连接标注 L4 处（J4 的 1 脚）至标注 5V（J1 的 3 脚），观察到的现象：	D4（亮/灭）_亮___。这里通过标注 4 所施加的 5V 电压，就是后续系统能通过软件输出的二进制控制信号“1”。
电路上电，以杜邦线连接标注 L4 处（J4 的 1 脚）至标注 GND（J1 的 4 或 5 脚），观察到的现象：	D4（亮/灭）__灭__。这里通过标注 4 所施加的 0V 电压，就是后续系统能通过软件输出的二进制控制信号“0”。
电路上电，按下 S1 按钮并保持，测量 IO11（J4 的 4 脚）电平（即该点的对地电压）：	IO11 的电平__0__V。这个就是你按下按钮状态下，系统能通过软件检测到的信号。
电路上电，松开 S1 按钮，测量 IO11（J4 的 4 脚）电平（即该点的对地电压）：	IO11 的电平_4.495___V。这个就是你松开按钮状态下，系统能通过软件检测到的信号。

测试过程顺利，过程中未发生问题。测试结果正常，符合预期。

测试四、电装总体完成后的初步通电测试

•先不安装芯片 LM324AN 和 LM35DZ，通过接入外接 5V 直流稳压电源，进行初步的通电测试。

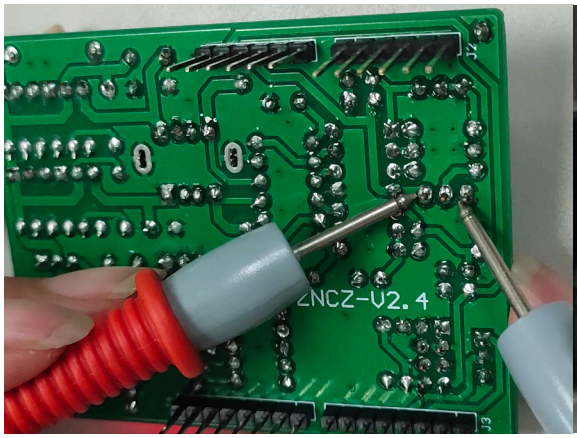
•检查 U1（即 LM324AN）芯片插座上的电源电压（即该插座上 4 脚和 11 脚之间的电压）是否为正常的 5V，并将检测结果记录于表四上。

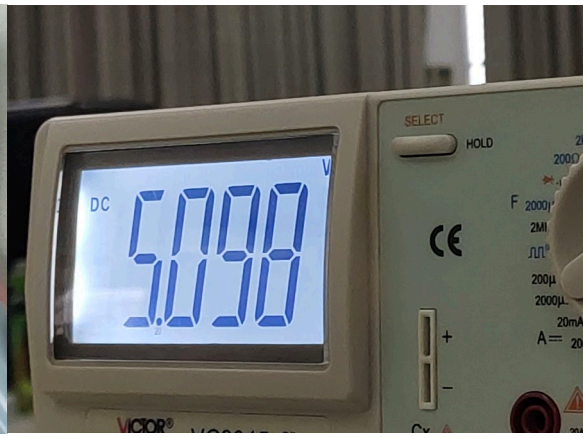
•检查 U2（即 LM35DZ）芯片插座上的电源电压是否为正常的 5V（即该插座上 1 脚和 3 脚之间的电压），并将检测结果记录于表四上。断开外接 5V 直流稳压电源。

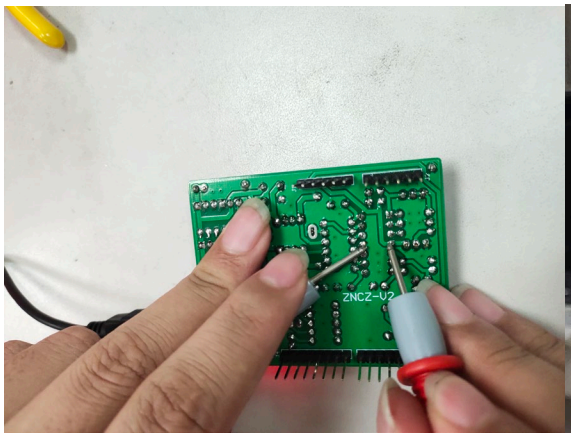
•如果上述电压均检测无误，则可以将芯片 LM324AN 和芯片 LM35DZ 安插到其相应的插座上。

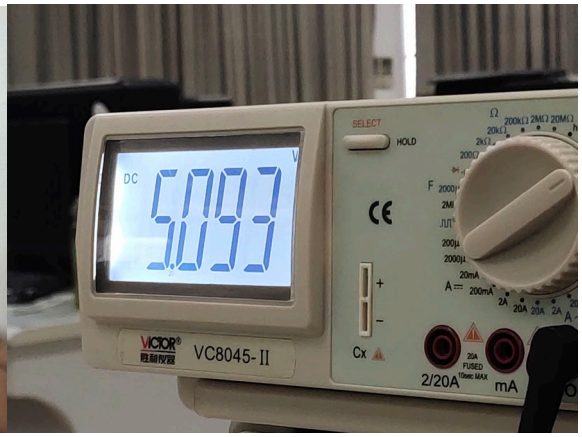
检测项目	检测结果
LM324AN 的电源电压（V）：	5.098V
LM35DZ 的电源电压（V）：	5.093V

测试过程顺利，过程中未发生问题。测试结果正常，符合预期。









电装过程小结

1)元器件安装顺序以及分析：

装配顺序：先安装供电电路模块部分，便于接下来提前测试供电电路功能，首先确保供电电路部分功能正常。之后按照从低到高，从小到大的，分类安装的顺序，分别依次安装：瓷片电容，AXIAL0.3 分装的电阻元件，AXIAL0.4、AXIAL0.5 分装的大电阻，发光二极管，二极管，三极管，电位器 VR1，按键开关，自锁开关，测试排针 TP1-TP4，各模块插座，USB 和继电器插座，J1-J4 排针。

装配分析和经验总结：

1. 低高度元件优先：首先焊接较低高度的元件，如小电容和小电阻。这样做可以确保在焊接较高元件时，低元件不会干扰到焊接过程。
2. 元件分类：元件按照类型进行分类焊接，这有助于提高工作效率，并减少焊接错误的可能性。在装配过程中，按照元件的封装类型和相似性进行分组。
3. 元件大小排序：在焊接电阻和电容等较小的元件之后，焊接较大的元件（如继电器、插座等）。这样可以确保焊接较大元件时有足够的空间和稳定性，避免阻碍和干扰其他元件的焊接过程。
4. 元件功能依赖：按照电路板功能和信号流向的顺序，将具有相关功能的元件进行相邻焊接。这有助于简化电路板的布线和连接过程，并确保信号流畅和正确。

5．稳定结构优先：焊接较大或较重的元件（如插座、继电器等）时，确保它们稳定固定在电路板上。这样可以避免后续元件焊接时的不稳定性和变形。

2）电装过程遇到的问题：

1.在给出的元器件中混入了多余的一个不同型号的三极管，导致三极管装配时误用了错误的三极管。

2.由于分模块装配，在最后装配大型元件和焊接排针时，PCB 板放置不稳定，器件引脚难以固定，影响焊接操作。

3）问题分析和纠错方案。

问题 1 分析：问题原因一方面在于在组装过程中，可能有人错误地将错误型号的三极管混入到装配的元器件中。另一方面在于装配时没有仔细检查三极管型号，导致焊接了错误器件。

纠错方案:仔细检查元器件清单和规格，确认正确的三极管型号和封装类型。将错误的三极管从电路板上拆下，用吸焊工具和焊锡除焊吸取焊锡，确保引脚清洁。使用正确的三极管进行装配，确保正确的极性和引脚连接。在装配过程中，仔细检查元器件的标识，以避免混淆不同型号的元器件。

解决效果：拆穿错误的三极管，装配正确的三极管，之后测试过程中电路功能一切正常。

问题 2 分析：是焊接装配过程中自然会遇见的问题，应考虑各类解决方案。

纠错方案:在装配大型元件之前，考虑使用支架或夹具来稳定 PCB 板，确保它不会晃动或倾斜。考虑在大型元件的安装位置添加临时支撑物，如胶垫或支撑块，以增加稳定性。使用适当的夹持工具或第三手工具来固定器件的引脚，确保焊接时它们保持在正确的位置。对于焊接排针，可以在 PCB 板的另一侧放置支撑物，以防止它们在焊接过程中移动。

解决效果：通过支撑物和固定等方式，顺利的进行了之后的焊接过程，焊接后效果较好，引脚位置稳固，焊接点稳定。

第二部分：智能插座 DIY 调试

调试工序

•硬件参数标定对一些需要量化测量或控制的电路硬件，进行计量调整，并与相关软件相配合，以实现计量等方面的标准化处理。如：通过对软硬件的调整，使系统所采集到的温度值与实际当前室温对应等。

•调试中软件与硬件相结合，以便使软件与硬件获得性能最佳的匹配。

测试条件

•用电设备：可调光台灯、小台灯、风扇

•测试仪器：直流稳压电源、万用表、示波器

调试环境及其配置

•Arduino IDE（Integrated Development Environment，集成开发环境）——内含编程、调试、项目管理等

•编程语言——Sketch

•调试硬件——串口

调试一、测温电路标定调试

•电路中的可变电阻 VR1 是用于调整温度测量值输入的转换比例的，所以你需要根据你在测试程序中显示的温度值，同时参考当前室温（可通过我们教室中安放的温度计来读取），并通过调整可变电阻 VR1 的螺丝旋钮来调整显示值，并使之与实际室温的尽量保持一致（正常测试环境下，如果存在少许的误差可能是元器件的参数离散性导致，请你在提交的报告中加以备注）。

项目	测量值
当前实际室温（摄氏度）	26℃
经你完成调试后测试程序显示温度（摄氏度）	26℃
是存在严重的元器件离散性问题？（是/否）	否

测试任务一

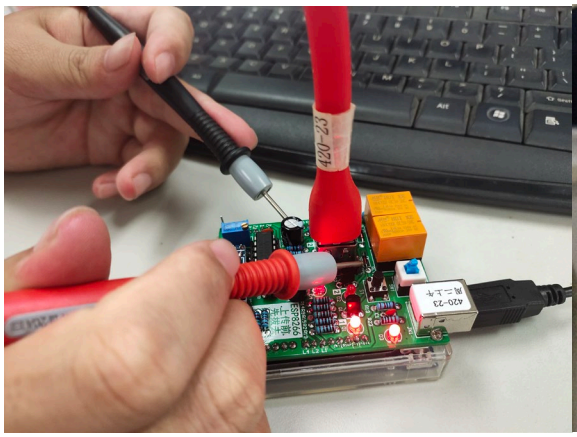
•利用小台灯和小程序 SmtScktTest_USB，确认程序控制 USB 插座中 1#和 2#插座的分别由哪个控制引脚控制，并完成下列各测试任务表。

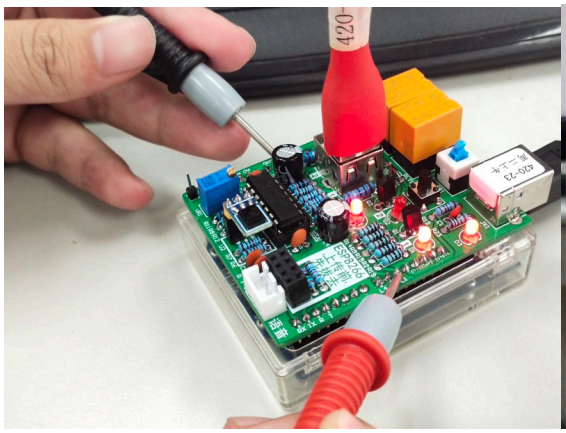
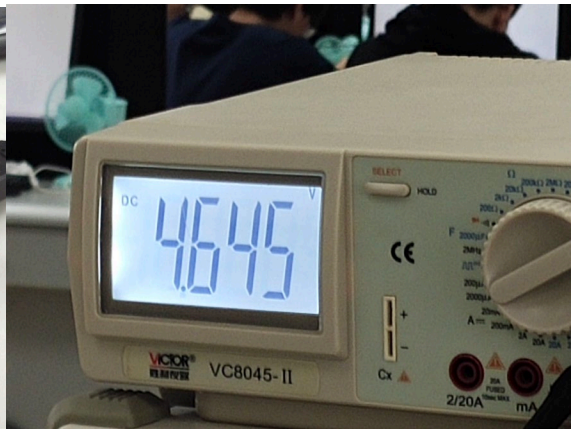
输出命令	1#和 2#插座的状态（通/断）	控制电压（用电压表测量 L1 或 L2）
发送字符 A	智能插座 1#插座的输出电压，TP1 的电压：_4.788_____V	J4 的_ L1__（选 L1 或 L2，即你程序中输出控制信号的那个引脚）脚电压：__4.645_____V
发送字符 a	智能插座 1#插座的输出电压，TP1 的电压：___0____ V	J4 的_ L1__（选 L1 或 L2，即你程序中输出控制信号的那个引脚）脚电压：_____0____V
发送字符 B	智能插座 2#插座的输出电压，TP2 的电压：	J4 的_ L2__（选 L1 或 L2，即你程序中输

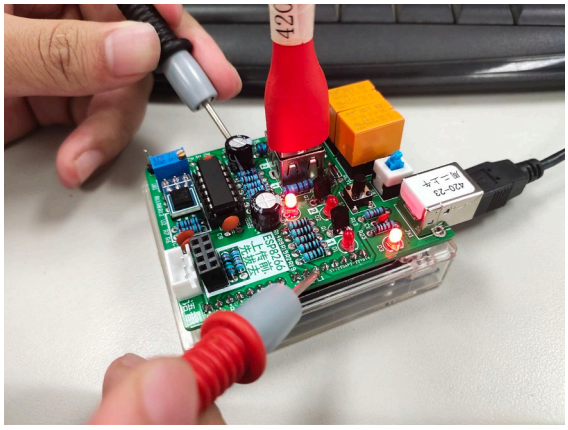
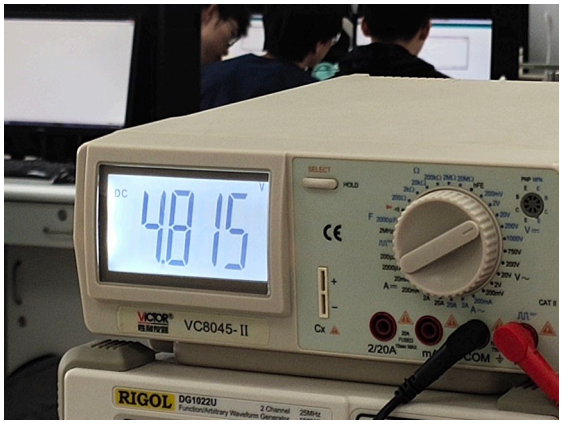
	___4.815____ V	出控制信号的那个引脚）脚电压：___4.673_____V
发送字符 b	智能插座 2#插座的输出电压，TP2 的电压：____0___ V	J4 的_ L2__（选 L1 或 L2，即你程序中输出控制信号的那个引脚）脚电压：_____0____V

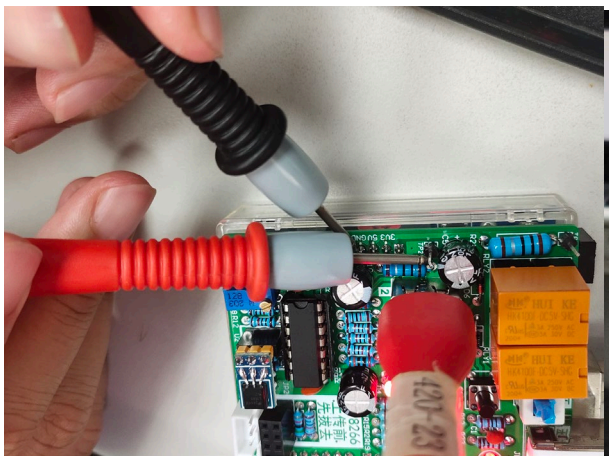
测试过程遇到的问题：在测量 J4 上的 L1 和 L2 引脚电平时，万用表表笔过大，测量不便，难以稳定接触测量点，难以稳定准确的测量电平。且在测量过程中容易导致误差和短路问题，导致单片机需要重启，影响测试过程。

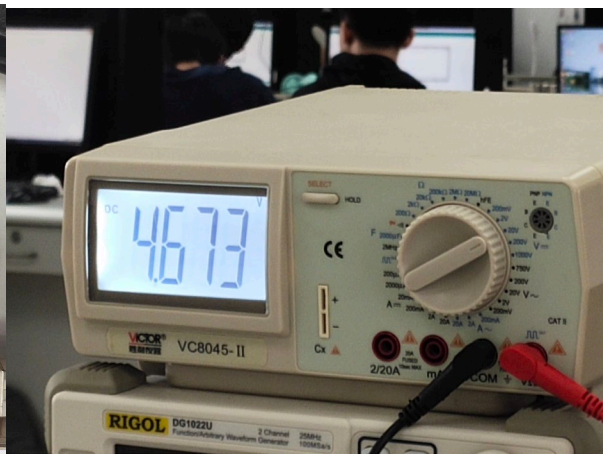
解决方案：直接从电路板上测量 L1 和 L2 引脚所在处的电平，相比测量 J4 上的针脚更加方便和容易，能较方便的测量出稳定的电压值。







测试任务二

利用各类用电器，测量其在智能插座使用中的各类测量值（注意：是通过我们的测试小程序来读取下述测量值）：

项目	测量值	单位
1#插座外接的用电器	小台灯	
1#插座电流值	286.44	mA
2#插座外接的用电器	可调光台灯	
2#插座电流值（最亮时）	341.88	mA
2#插座电流值（最暗时）	72.07	mA
插座电压值	4.92	V
温度测量值（环境温度）	26	℃
温度测量值（手指触碰温度）	33	℃

测试任务三

将 2#插座上的可调光台灯换成风扇，并接通风扇的插座电源，测量风扇的各档工作电流值。（注意：是通过我们的测试小程序来读取下述测量值）：

项目	测量值	单位
风扇慢速档电流值	260	mA
风扇中速档电流值	293	mA
风扇快速档电流值	333	mA

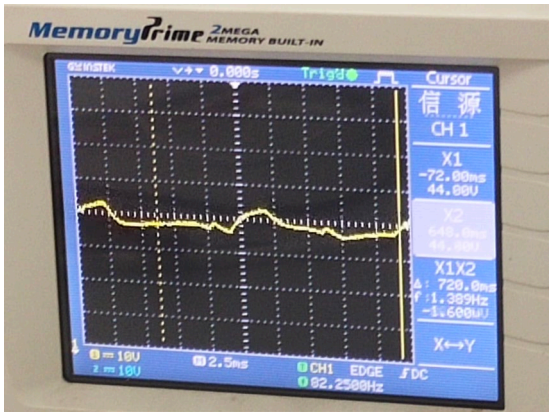
异常值分析：

上述测试任务二、三中有哪些测量值有异常？利用示波器观察并拍照记录出现这些异常时输出供电电压的波形，说明其产生的原因。

在测试任务二、三中电流值的测过程中，从串口监视器上可以发现，电流值存在异常，始终在跳变，并不稳定，难以纪录合适的测量值。利用示波器观察可以发现，电压和电流值呈波动状，并不稳定，其工作时的电流值并不稳定，最终我们选取了一段时间内的平均值作为测量结果。

可能的原因分析：

1. 电器内部负载变化：电器的工作原理可能导致其内部负载在不同时间点上升或下降。例如，风扇电机使用 PWM 电压调制，工作特性可能导致电流持续波动。这种内部负载变化会导致测量的电流波动。
2. 瞬态事件：某些电器在工作期间可能会产生瞬态事件，例如电容器充电或放电、开关电源的开关瞬变等。这些瞬态事件会导致电流波动，。
3. 电源噪声：电器供电的电源可能存在噪声，如电源干扰、谐振等。这些噪声会对测量的电流产生微小影响，导致电流呈现小范围波动。



软硬件联调任务二

•设计要求：

程序代码在读取按键输入时会遇到按键信号抖动的干扰。你需要：

（1）通过设计测试试验，展示并记录这种干扰出现的场景和干扰信号的表现；

（2）通过 Arduino 代码的编程设计，去除这种干扰对按键信号读取的影响；

（3）（选做）尽可能提高你所设计的 Arduino 代码对按键动作的响应灵敏度。

软硬件联调任务二的实现方案

•步骤 1：你需要首先弄清按下按钮/松开按钮时，Arduino 程序分别读取到的按键状态值（二进制的 0/1）。并记录在下表中。

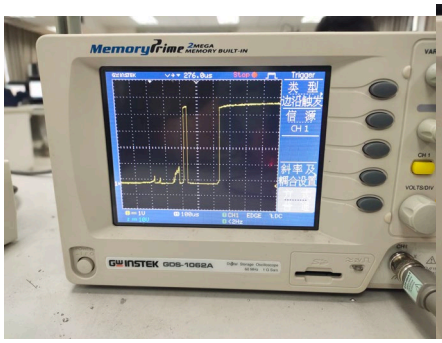
s1 按钮的程序读取值(0/1)	
按下 s1 按钮时	0
松开 s1 按钮时	1

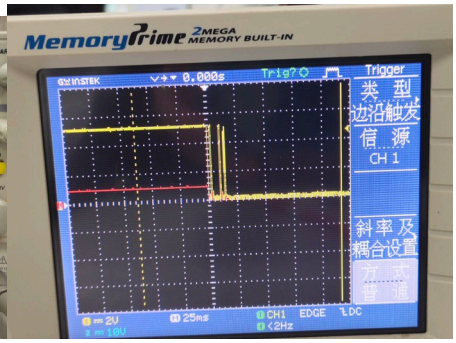
步骤 2：需要你采用各种按键速度和连续按键次数，测试程序是否能每次都正确记录按钮次数？如果不是（即存在误读次数），可通过示波器查看并记录按钮按动时的波形，并分析误读产生的原因，写入报告中。

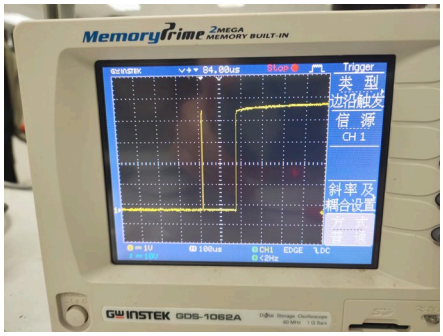
原因分析：

结合示波器的波形分析，可知。按键误读的主要原因可能是由于按钮按下时产生的扰动和 glitch。扰动是指在按钮按下或释放的瞬间，按钮引脚的电压或电流可能会短暂地发生突变或震荡。而 glitch 则是指在按钮状态发生变化时，按钮引脚的信号可能会出现短暂的非预期或错误的脉冲。

这种扰动和 glitch 可能是由于按钮机械结构的不完美、按钮引脚的电气特性、电路布线的问题、电源干扰、信号传输线路的串扰等因素导致的。







•步骤 3：修改该程序 key_in.ino，使修改后的程序能准确地读取并记录下以各按键速度连续按键的次数（注：每"按下-放开"只能算一次按键）。

在总结报告中附上你的这些测试屏显内容截图和必要的测试过程说明。

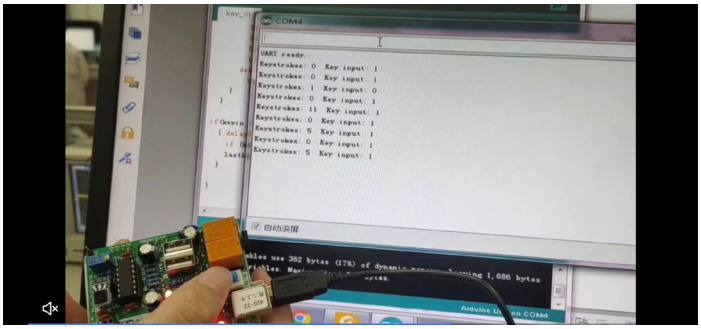
测试过程说明：

下图分别为修改的代码片段和测试屏内容截图：

通过增加一个延迟语句，消除按键按下后短时间内的抖动和毛刺，从而避免误读的情况发生，且消抖的时间较短，也不会影响连续按键的记录准确度。

测试方式为，快速、慢速、连续按键、持续按键等方式测试按键，对比人工计数的按键次数和串口监视器上打印的按键次数来测试按键程序。

```
95   if(keyin != lastkey)
96     {
97       t1=micros();         //记录按下时间
98       delay(20);           //延迟20ms，消除glitch
99       if (keyin !=1) {
100      t2=micros();         //记录响应时间
101      t=t2-t1;             //按键响应所需时间
102      if(min=0 || min>t) min=t;      //最快响应
103      if(max=0 || max<t)  max=t;     //最慢响应
104      aver+=t;                       //平均响应
105      acc++;
106      }
107      lastkey = keyin;
108    }
```



•针对慢速按键：先用'0'命令清除计数，缓慢地按下 s1，接着缓慢地松开 s1，这样多做几次，同时自己计数按动-松开的总次数。然后用'r'命令显示软件对这次操作记录的按键次数值。该软件计数正确吗？分析原因。

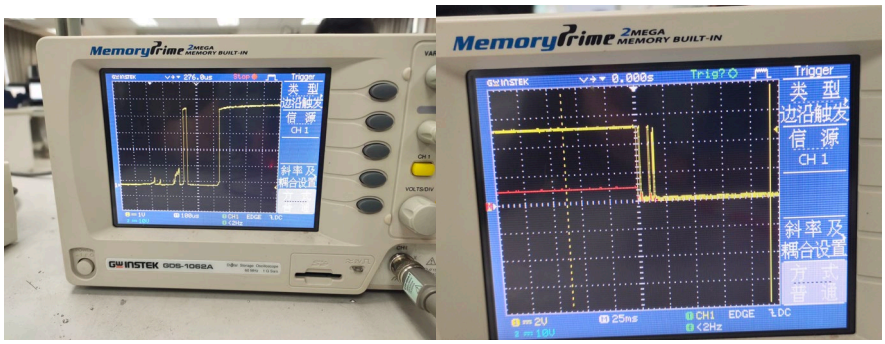
•针对快速按键：先用'0'命令清除计数，快速地连续按下-松开 s1

多次后停止，并自己计数按动-松开的总次数。然后用'r'命令显示软件对这些按键操作的所记录的按键次数值。该软件计数正确吗？分析原因。

分析原因（包括示波器显示的波形）须写在总结报告中。

原因分析：

在修改程序前，快速和慢速按测试时，软件计数不正确，会出现多于实际按键的次数，通过示波器查看按键信号波形可见，该误读产生的原因是由于按钮按下时会出现抖动和毛刺的现象，导致一次按键会被软件记录多次。
修改程序后，通过延迟避开了毛刺出现的时间段，再记录按键按下情况。再经过测试，发现软件计数基本正确。



软硬件联调任务二（选做部分）的实现方案
•可在程序中增加对各次按键中最快按键响应时间的记录与显示，以及平均按键响应时间的检测记录与显示。以便评估你所修改的程序“在能避免误计按键次数的前提下，能对多快的按键速度正确响应并计数”。
在总结报告中可附上你的这些测试屏显内容截图和必要的测试过程说明。

测试过程说明：

代码片段如下所示：各代码功能如文中注释所示。
测试时，先用’0’命令清除计数，快速地连续按下-松开 s1 多次后停止，并自己计数按动-松开的总次数。然后用’r’命令显示软件对这次操作记录的按键次数值，与人工计数比对。在确保计数准确的情况下，用’t’命令显示软件对这些按键操作的所记录的按键最快、最慢、平均响应次数，评估按钮的响应速度。修改 delay 函数中的延迟时间，重复测试，寻找在计数准确的情况下，按钮能实现的最快按键响应速度。

```
95   if(keyin != lastkey)
96     {
97     t1=micros();        //记录按下时间
98     delay(20);          //延迟20ms，消除glitch
99     if (keyin !=1) {
100    t2=micros();        //记录响应时间
101    t=t2-t1;            //按键响应所需时间
102    if(min=0 || min>t) min=t;      //最快响应
103    if(max=0 || max<t)  max=t;     //最慢响应
104    aver+=t;                       //平均响应
105    acc++;
106    }
107    lastkey = keyin;
108  }
```

```
    case 't':                   //打印按键最快、最慢、平均响应时间
      Serial.print("max: ");
      Serial.print(max);
      Serial.print("min: ");
      Serial.print(min);
      Serial.print("average: ");
      Serial.print(average/acc);
    default:
      break;
  }
```

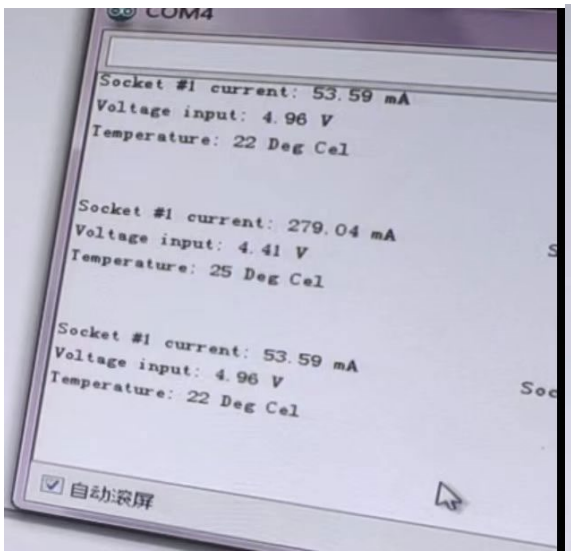
测控任务（选做题）

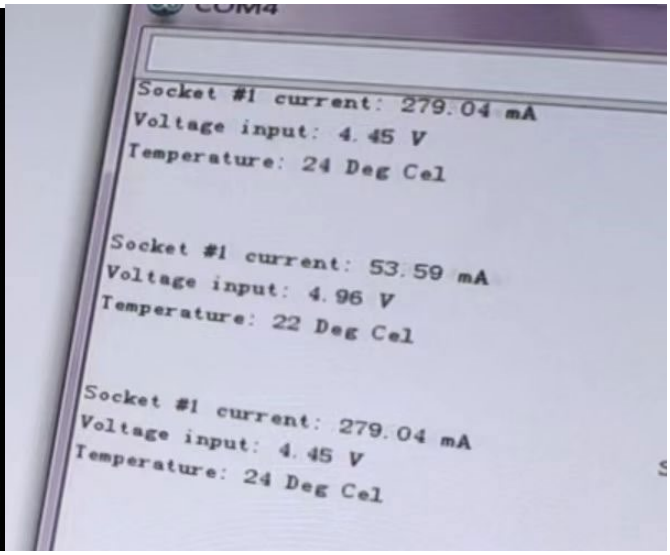
•利用风扇和你智能插座上的板载温度传感器，在课程提供的实例程序 SmtSckt Test_USB.ino 的基础上改写代码，使你的风扇能根据你设置的温度期望值自动启停，以便控制温度传感器周围的温度。
•完成代码改写，并根据当前的室温，设置你的测控温度期望值，以便展示你风扇在你设计的程序代码控制下多次自动启停以控制恒温的性能效果。

记录测试过程中的测控数据，一般包括你设置的期望温度，测控过程中的最高温度、最低温度、风扇从启动时刻至达到设定期望值所用时间、或者风扇从关闭时刻至重新启动时刻间的时间...等等（不一定要求上述全部）你认为有展示系统性能的测试数据指标。

测控数据：

如下图所示：在代码中，设定的期望温度为 24℃，测控过程最高温度 25℃，最低温度 22℃，风扇工作电流约 280mA，风扇从最初启动时刻至达到设定期望值所用时间约 5s，风扇从关闭时刻至重新启动时刻间的时间仅需 1s 左右，系统可持续的保持温度在 24℃范围附近，测试效果良好。






```
 if(Sensor3>=24)  digitalWrite(relayPin10, HIGH);       //温度测控代码
else digitalWrite(relayPin10, LOW);
```

第三部分：智能插座 DIY 系统测试

系统测试的目的

•通过对系统每项功能在预设的典型场景中的使用测试，评价设计对前期需求分析的完成度。
•在测试中发现系统设计或实现中存在的 Bugs。
•系统测试后一般会回溯之前的研发过程，以便修复测试中存在的 Bugs。

通用系统测试方法介绍

•我们在进行一个系统测试时需要让你的测试工作尽可能覆盖到被测系统的各方面，这就是评价测试工作完备性的——测试覆盖率。
•我们在开始动手测试一个系统之前，会提前针对每项测试准备一个或多个测试步骤，准备好一套或多套的输入数据，并预测系统正常情况下的执行状态或输出结果，以便实际测试时与实际情况相对照，判断系统执行正确与否。这种测试前就准备好的测试步骤或输入数据被称为：测试用例。
•从系统测试效率来看，我们期望以最少的测试用例达到最全面的测试覆盖率。这就要求对你的测试用例精心筛选，以达到较高的系统测试效率。
•测试用例的选择设计，反应了一个产品测试的完善程度和测试效率，同时也体现了一个测试工程师的实际工作经验与测试水平。

系统测试过程

系统有如下功能：
1.我们的智能插座可连一个手机。
2.智能插座具备定时开/关、延时开/关。
3.针对超/欠电压、超电流、超功率告警并自动断开插座供电。
4.插座开关手动控制，以及温度、电压、电流和功率显示。

关于系统测试报告

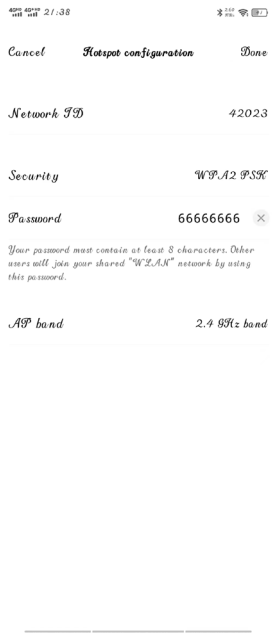
•根据上述的系统测试过程，撰写系统功能测试报告。报告包括针对 APP 中所有功能的描述，以及你的测试过程和测试用例（有了这部分描述，别人就能轻松地事后重现你做发现的 Bug），给出测试结果。如发现有 Bug 存在，请描述之。
•选做，新增一项功能：开机时按下 s1 按钮则将 ssid 和密码恢复到一个默认配置状态。要求修改代码实现之；并设计该项修改后的测试过程与测试用例，记录展示测试结果。报告中附上所修改的代码片段。
•选做，是否有 Bug（如果你发现有的话）？你能修改吗？如果你能修改的，试修改之。并在报告中附上相应的修改代码段。

系统测试报告

测试用例 1

a)测试用例描述：
装有 APP 的安卓手机（vivo z5）
b)测试目的：

检验智能插座连接手机的功能
c)操作过程及其结果：
1.设置手机热点
2.设置代码中的热点名称、密码



smtsckt-v5.ino

```
#define ssid → "42023"    //热点名称
#define password "66666666"    //热点密码

#define pinLEDob  13

#define pinTemp A3
#define pinVol A2
#define pinCurrent0 A0
#define pinCurrent1 A1

#define MAX_FAN_CURRENT  473      //unit = ma
#define MAX_LED_CURRENT  473    //unit = ma

//#define __DEBUG
```

3.打开手机热点，APP 成功连接智能插座

测试用例 2

a)测试用例描述：
插座 2 定时开、关时间设置为 11.16、11.17
插座 1 延时开、关时间设置为 1min、1min 后
b)测试目的：
检验智能插座定时开/关、延时开/关的功能
c)操作过程及其结果：
定时开/关：
1.插座 2 连接小台灯
2.设置定时开关
3.等待时间，观察到小台灯在对应时间亮、灭
延时开/关：
1.插座 1 连接小台灯
2.设置延时开
3.等待时间，观察到小台灯在 1min 后亮
4 设置延时关
5.等待时间，观察到小台灯在 1min 后灭

测试用例 3

a)测试用例描述：
安全设置电压初始值为 4.5、5.5，测试值为 4.8、4.9（单位 V）
电流初始值为 0.5，测试值为 0.3（单位 A）
功率初始值为 3.5，测试值为 1.5（单位 W）
b)测试目的：
检验智能插座针对超/欠电压、超电流、超功率告警并自动断开插座供电的功能
c)操作过程及其结果：
超、欠电压：
1.设置最大电压
2.插座二接小台灯并打开，小台灯亮
3.按下 SW1，插座二断开，小台灯灭，警报灯亮
4.恢复最大电压设置，设置最小电压
5.打开插座二，小台灯亮
6.再次按下 SW1，插座二断开，小台灯灭警报灯亮
超电流：
1.设置最大电流
2.插座一接风扇并打开，打开中速档，风扇正常工作
3.切换风扇至快速档，插座一断开，警报灯亮

超功率：
1.设置最大功率
2.插座一接风扇并打开，打开中速档，风扇正常工作
3.切换风扇至快速档，插座二断开，警报灯亮

测试用例 4

a)测试用例描述：
插座一接小台灯，插座二接风扇
b)测试目的：
检验插座开关手动控制，以及温度、电压、电流和功率显示功能
c)操作过程及其结果：
1.手动开、关插座一，观察到小台灯亮、灭
2.手动打开插座二，并依次打开风扇慢、中、快速档，观察到风扇正常工作，电压、电流、功率、温度均正常显示
3.手动关闭插座二，并用手指触碰温度传感器，观察到温度上升


插座1
插座2
温度
电流1
电流2
电压
总功率

二、新增功能：

新增一项功能：开机时按下 s1 按钮则将 ssid 和密码恢复到一个默认配置状态。
代码片段如下图所示，将宏定义改为字符串，在 set up 函数中新增语句用于判断按键响应，从而将 ssid 和密码更改为默认配置模式。
测试过程：1.在开机时，分别按下和不按下 s1 按钮，观察蓝牙模块的连接状况。2.将默认配置更改为另一部手机的热点和密码。同时开启两部手机热点，分别按下和不按下 s1 按钮，观察蓝牙模块的连接状况。
测试结果：1.不按下按键，智能插座正常连接至原先的手机，按下按钮后，无法连接。
2.不按下按键，智能插座正常连接原先手机，按下按钮后，连接至另一部手机。

```
1   //#define ssid    "42023"      //热点名称
2   //#define password  "66666666"          //热点密码
3   char *ssid= "42023";             //热点名称
4   char *password="66666666";       //热点密码
5   const int keyPin11 = 11;         //按钮引脚
6   bool keyin, lastkey;
```

```
1588  void setup()
1589  {
1590      keyin = digitalRead(keyPin11);
1591          if(!keyin){          //判断按钮按下
1592            ssid = "default";                //设置默认状态
1593            password = "00000000";
1594          }
```

三、BUG 发现和代码修改与解决：

1.BUG 发现：在测试智能插座定时开关的过程中，我们发现：在时间到达整分钟时，定时开关的功能正常，但在随后准备手动开关智能插座时，发现智能插座的手动开关失效，插座会持续处于开启或关闭状态（取决于定时设置时选择的是开启还是关闭）。一段时间过后，手动开关又恢复正常。经过测试，我们发现插座失效时间约为半分钟。
2.BUG 分析：通过查阅 Arduino 源代码，我们发现 BUG 原因在于用于纪录系统时间的变量 lTicks 为 long 类型，在描述定时开关的代码中，lTicks 自动向下取整，导致了半分钟的开关延时，从而产生了这个 BUG。
具体代码片段如下：(下列代码均截取自 smtsckt-v.ino)
延时开关函数主体片段代码如下

```
1402
1403  void delayTimeControl()
1404  {
1405    //延时和定时控制需要结合实时钟
1406    if (toInfo1.enabled && (lTicks/60 == (toInfo1.minute +toInfo1.hour * 60) * 2) )
1407    {
1408      //toInfo1.enabled = false;
1409      state1 = true;
1410      DBUG_PRINTLN("to1");
1411    }
1412    if (tcInfo1.enabled && (lTicks/60 == (tcInfo1.minute +tcInfo1.hour * 60) * 2) )
1413    {
1414      //tcInfo1.enabled = false;
1415      state1 = false;
1416      DBUG_PRINTLN("tc1");
1417    }
1418
1419    if (toInfo2.enabled && (lTicks/60 == (toInfo2.minute +toInfo2.hour * 60) * 2) )
1420    {
1421      //toInfo2.enabled = false;
1422      state2 = true;
1423      DBUG_PRINTLN("to2");
1424    }
1425    if (tcInfo2.enabled && (lTicks/60 == (tcInfo2.minute +tcInfo2.hour * 60) * 2) )
1426    {
1427      //tcInfo2.enabled = false;
1428      state2 = false;
1429      DBUG_PRINTLN("tc2");
1430    }
1431
1432
```

变量的定义如下

```
148   long lTicks = 0;//午夜至今的半秒数
```

```
106   bool state1, state2;  //用于纪录定时开关的状态
```

```
170   //time to open & time to close
171   void initTOC()
172   {
173     toInfo1.enabled = false;
174     toInfo1.hour = 4;
175     toInfo1.minute = 5;
176     tcInfo1.enabled = false;
177     tcInfo1.hour = 6;
178     tcInfo1.minute = 7;
179     toInfo2.enabled = false;
180     toInfo2.hour = 12;
181     toInfo2.minute = 13;
182     tcInfo2.enabled = false;
183     tcInfo2.hour = 14;
184     tcInfo2.minute = 15;
185   }
```

代码分析：从代码中可见 toInfo1.enabled 用于纪录是否开启了定时功能，toInfo1.hour 和 toInfo1.minute 记录定时的小时和分钟。表达式(toInfo1.enabled && (lTicks/60 == (toInfo1.minute +toInfo1.hour * 60) * 2)用于判断是否到达设定时间。BUG 主要原因在于表达式 lTicks/60。由于 lTicks 为 long 类型，lTicks/60 会向下自动取整。导致在到达设定时间的 60

个半秒（即半分钟）内，lTicks/60 没有及时改变，if 语句的判断语句值始终为真，开关一直处于开启或关闭状态。
3.代码修改和 BUG 修复：
修改后的代码如下：（其他判别语句也同理修改即可）

```
1406    if (toInfo1.enabled && (lTicks/2 == (toInfo1.minute +toInfo1.hour * 60) * 2*30) )
```

将 lTicks/60 改为 lTicks/2，相应的表达式右侧*30，将延迟降低至一秒，不影响插座的正常使用，也保证了在一秒的时间内，在单片机的程序执行周期内，定时语句能被单片机检测到。不影响定时功能。
经过上板检测发现，BUG 基本修复，定时开关的功能正常，在定时开启后，手动开关智能插座也能正常执行。

课程总结

这门电子工程训练课程对我来说是一次极富挑战性和收获的学习经历。通过完成一个智能插座的 DIY 项目，我深入了解了电子工程的设计和实施过程，掌握了电路设计、焊接、调试和测试等关键技能。

课程总结：

1） 在项目的第一部分，智能插座的 DIY 电装，我结合了调试的过程来理解电路结构。我将电路按功能进行分块，并逐步进行安装和调试。在安装过程中，我首先明确了供电模块和输入输出通道模块，然后按照正确的顺序进行安装。我特别注重供电电路的调试，确保电压和纹波等参数符合要求，以避免对后续电路安装和功能运行造成不良影响。我遵循了先安装较便宜的器件，后安装较贵重的器件的原则，并在逐级安装和调试的过程中谨慎对待，一部分一部分地进行安装和调试。

在完成全部电装任务的基础上，我总结了自己的电装经验。首先，我列出了我的电装元器件的顺序，以便于今后的参考和复盘。其次，我列出了在电装过程中遇到的各类问题，如焊接错误、元件损坏等。针对这些问题，我进行了分析并提出了纠错解决方案，确保问题得到解决并取得了良好的效果。我还记录了之前的各种测试数据和问题解答，以备今后

查阅和参考

2） 在项目的第二部分，智能插座的 DIY 调试，我进行了硬件参数标定和软硬件配合调试。我对需要进行量化测量或控制的电路硬件进行了计量调整，并与相关软件相配合，以实现计量等方面的标准化处理。通过调试软件和硬件的配合，我使系统所采集到的温度值与实际当前室温对应，达到了预期的效果。

在调试过程中，我使用了可调光台灯、小台灯和风扇等设备，并借助直流稳压电源、万用表和示波器等测试仪器进行测试。我使用 Arduino IDE 和编程语言 Sketch 进行调试，通过串口进行硬件调试。这些工具和设备的运用，使得调试过程更加高效和准确。

3） 最后，在项目的第三部分，智能插座的 DIY 系统测试，我通过对系统每项功能在预设的典型场景中的使用测试，评价了设计对前期需求分析的完成度，并发现了系统设计或实现中存在的 Bugs。

在系统测试过程中，我学会了使用通用的系统测试方法。为了确保测试工作覆盖到被测系统的各方面，我尽可能准备了多个测试用例，并提前准备了一套或多套的输入数据。我对每个测试用例进行了测试步骤的规划，并预测了系统正常情况下的执行状态或输出结果。通过与实际情况相对照，我能够判断系统是否执行正确。

为了提高系统测试效率，我精心筛选了测试用例，力求以最少的测试用例达到最全面的测试覆盖率。我将测试用例的选择设计作为一种衡量产品测试完善程度和测试效率的标准，也是体现我实际工作经验和测试水平的重要方面。

在整个系统测试过程中，我发现了一些问题，并及时进行了修复。例如，我发现定时开/关功能在某些情况下无法正常工作，经过分析发现是程序代码逻辑错误导致的。我及时修改了代码，并进行了重新测试，最终解决了该问题。

通过完成整个智能插座 DIY 项目的过程，我不仅掌握了电子工程设计和实施的基本技能，还学会了分析和解决问题的能力。我意识到在电子工程领域，严谨的思维和谨慎的操作是非常重要的。同时，通过系统测试，我深刻体会到了测试工作的重要性，只有经过全面而有效的测试，我们才能确保产品的质量和性能达到预期。

总之，这门电子工程训练课程为我提供了一个宝贵的学习机会，让我在实践中不断成长和提升。我相信通过对所学知识的巩固和应用，我将能够在电子工程领域取得更大的成就。同时，我也非常感谢导师和教练的指导和支持，他们的专业知识和经验对我的学习和进步起到了重要的推动作用。

课程收获：

在完成这门电子工程训练课程的过程中，我获得了许多宝贵的收获/

1. 知识和技能的提升：通过课程学习，我深入了解了电子工程的基本原理、电路设计和调试方法。我学会了电装和调试电路的步骤，掌握了硬件参数标定和系统测试的技巧。这些知识和技能的提升将对我的未来学习和职业发展产生积极的影响。

2. 实践项目经验：课程中的智能插座项目让我有机会将所学的理论知识应用到实际项目中。通过手工焊接 PCB、电路模块调试和系统测试，我深入了解了项目开发的全过程。这种实践经验对我的技能提升和职业发展至关重要。

3. 团队合作与沟通能力：课程中的小组项目要求我们在团队中合作完成任务。这锻炼了我的团队合作和沟通能力，我学会了与他人合作、协调分工、解决冲突并有效沟通。这对我未来

的职场发展和团队合作非常有帮助。

这门电子工程训练课程也让我更加自信和有信心地面对未来的挑战。我相信我所学到的知识和技能将对我的学术和职业生涯产生积极的影响。在未来，我希望能继续深入学习和探索电子工程领域。我计划参与更多的实践项目，不断提升自己的技术能力和创新思维。同时，我也会继续关注新兴技术和趋势，并努力将其应用到实际项目中。