重慶理工大學

智能硬件综合实践课程设计

题目：基于 ESP32 的多功能电子日历

所在学院	电气与电子工程学院
专　　业	电子信息工程
班　　级	电子信息工程四班 121070204
学生姓名	
学　　号	
同 组 人	
指导教师	黄丽雯、王培容、全晓莉、丛超、彭醇陵

注意事项：

1、以上各项由本科生认真填写；

2、课程设计论文应符合一般学术规范，具有一定应用价值，严禁网上下载或抄袭；凡检查或抽查不合格者，一律取消该门课程成绩和学分，并按有关规定追究相关人员责任；

3、论文得分由批阅教师填写（见封底），并签字确认；批阅教师应根据作业质量客观、公正的在文后签写批阅意见；

4、原则上要求所有课程论文均须用 A4 纸打印，加装本封面封底，左侧装订；

基于 ESP32 的多功能电子日历

班级级号：121070204 **作者 姓名：姚鑫 学号：**12107980106

摘要

本项目是基于 ESP32 的多功能电子日历是一款结合了现代科技与实用性的创新设备。该设备的核心是 ESP32 模块，这是一个高性能、低功耗的微控制器，适合用于物联网(IoT)应用和各种无线通信需求。这款电子日历采用了 4.2 寸黑白红三色墨水屏作为显示界面，这种屏幕具有超低功耗、高对比度的特点，能够在阳光直射下清晰可见，同时避免了传统 LCD 屏幕对眼睛的疲劳影响。墨水屏的颜色丰富性使得信息展示更具吸引力和可读性。在功能上，这款多功能电子日历集成了热点新闻推送、温湿度监测、未来天气预报、实时时间显示以及每日一句励志语录等多种实用功能。用户可以通过 Wi-Fi 连接互联网，获取最新的新闻资讯和天气预报，确保用户时刻掌握最新动态。此外，该电子日历还具备有线与无线充电的功能，大大增强了其便利性和实用性。无论是通过传统的 USB 接口进行充电，还是利用 Qi 无线充电标准实现无接触式充电，都能轻松满足用户的充电需求。总之，基于 ESP32 的多功能电子日历凭借其丰富的功能、友好的用户界面以及便捷的充电方式，成为了一款理想的个人生活助手，为用户提供了全方位的信息服务，提升了生活的品质和效率。

关键词：ESP32；电子墨水屏；MQTT；Wi-Fi 通信；电子日历；

目录

图目录

表目录

一、 绪论

随着各国环保意识的增强，对资源浪费的行为逐步引起人们的注意，尤其是在日常需要信息标识的场合下，常见使用以纸质或者其他材料作为载体进行信息标识的现象产生环境污染，当此类以实体材料为载体的信息需要修改或者撤换，又容易造成资源的二次浪费[1]。对于此问题我们设计了基于 ESP32 的多功能电子日历，以电子墨水屏来展示需要的信息，进而可以节约资源，使人们生活水平得到提高。

人类文明的载体从以纸张、书籍为主体，逐渐向网络、电子文档的形式转变，越来越多的印刷制品被电子刊物所代替，给人类社会带来了巨大的便利与进步，人类文明的传播与发展也获得了跨越式的提高。但是伴随着电子显示技术的飞速发展，其伴生问题也接踵而至。传统显示技术通常通过显像管等电子显示屏进行显示，信息由显示屏直射到人们眼中从而实现成像。虽然这种显示方式方便快捷，但是长时间使用直射式显示屏带来的是长期的辐射伤害、视力疲劳甚至神经衰弱等问题，越来越多的人们患上了各式各样的眼科疾病。

电子墨水屏便得到了大家的认可。

电子墨水屏无需像 LCD 屏背光发射，它是利用自然环境光打在显示屏上，再折射到眼睛的原理。这种方式模拟了墨水与纸张的特点，环境光越强，其显示效果越清晰。由于没有了闪烁与蓝光，所以在长时间阅读时，眼睛不容易感到疲累。

电子信息技术不断推陈出新，人们对于电子产品性能的要求不断提高的同时，也对电子产品的便捷性、产品的功耗以及持续工作性能提出了更高的要求。伴随着这些多样性的需求，如何对产品进行更加有效的能量供给同样成为了越来越重要的议题。

传统的供电方案主要有有线供电和内置干电池供电，这两种传统供电方式技术相对成熟，供电效率高，且成本低，但这两种传统供电方式的明显缺点有：

1. 由于干电池自身寿命问题，需要定时更换，并且对环境也有一定影响。
2. 不同设备不同厂家生产预留的充电接口可能不同，对其做适配浪费人力物力。

伴随着传统供电方式的种种缺点，无线充电方式应运而生，相较于传统供电方式的各种缺点，利用无线供电方式进行供电具有多种优点，具体有：

1.便捷性：无线充能方式避免了更换电池，且不存在适配充电接口的问题，在极大程度上简化了设备的供电过程。

2.安全性：具有更好的密封性，无需频繁拆机，避免更换电池所带来的附带安全问题。

二、 系统主要技术及开发工具

2.1 电子墨水屏

本次用到的电子墨水屏基于电泳式电子墨水技术，简称电子墨水技术。电子墨水屏一般含有一层电子墨水薄膜，薄膜中间充斥着电子墨水。电子墨水内部包含有几百万个细小的球状的微胶囊，每个微胶囊内部充满了透明染液以及白色与黑色的微胶粒[2-3]。电子墨水微胶囊如图所示。

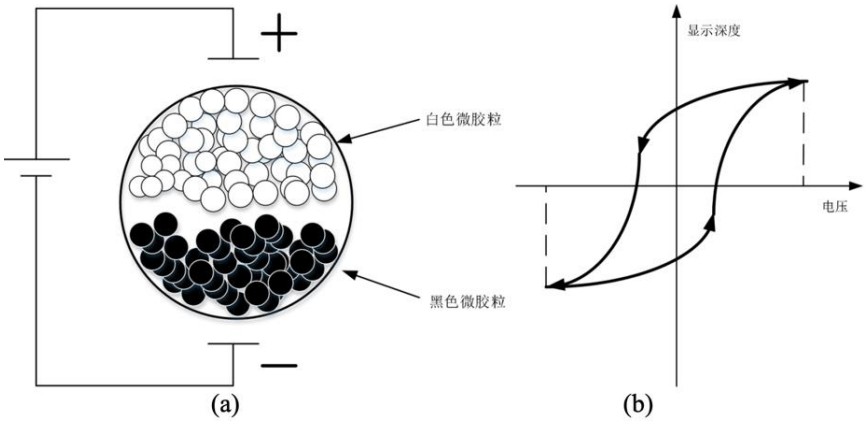

图 二-1 电子墨水 （a）电子墨水微胶囊（b）双稳态磁滞效应

图 2-1 中(a)图展示了微胶囊的具体结构。微胶囊中的黑白微胶粒分别带有正负电，屏幕通电时，胶囊上下两边被附加一个电场，胶囊中的带电微胶粒在电场的作用下开始向两侧移动，电子墨水具有双稳态磁滞效应，使得画面形成后，粒子可以停止运动，即使断电也不会影响屏幕的显示，这也是电子书等产品待机功耗极低的原因。双稳态磁滞效应如图 2-1 中图(b)所示。电压升高和降低所带来的电子墨水显示效果变化是不同的，这意味着当屏幕从零开始升压时，可以快速改变电子墨水显示到目标效果，如图中下侧曲线所示;而之后断电后，电压下降，由于双稳态效应的原因，电子墨水显示深度将首先缓慢降低，当两侧施加负压时才会快速降低显示深度，如图中上侧曲线所示。这意味着断电时，电子墨水的改变将被保留下来。也正是因为这个原因，电子墨水屏只有在屏幕刷新时才会耗电，屏幕断电不会影响屏幕的显示，因而电子水墨屏显示技术非常省电。而由于电子墨水显示效果基于光线反射，这也使得电子墨水产品易于阅读，显示效果与纸张无异。

这个原理也可以方便我们理解在电子墨水屏进行全局刷新时，整个屏幕会黑白交替闪烁一下的视觉效果的原因了。系统将所有的电压加到最大电压或者减小到最小电压，这样可以同步所有电子墨水的实际状态，之后再进行新的刷新显示，就可以使屏幕正常完成工作了。若是不进行这样的全局刷新，在前一次显示结果上直接附加电压进行更新工作时，由于不同位置的电子墨水处于图中的不同位置，从而导致该像素点之后的颜色变化轨迹可能与预期不同，甚至相反，最终的结果可能导致新旧图片显示重叠，即产生了我们肉眼所视的残影。

本次选择的是 4.2 寸黑白红电子墨水屏，拥有 400*300 像素，以显示黑白为例，定义像素点黑色为 0，则白色定义为 1。

像素	1	2	3	4	5	6	7	8
bit	7	6	5	4	3	2	1	0
数据	0	0	0	0	1	1	1	1
颜色	黑	黑	黑	黑	白	白	白	白
byte	0x0F							

表 二-1 墨水屏显示对应表

计算机中数据储存时高位在前，低位在后，最高位对应第一个像素点。如上表所示，四个黑像素点和四个白像素点对应的十六进制数据即为 0x0F。红白的显示与黑白显示类似，只是红白显示写入的是 0x26 寄存器，而黑白显示写入 0x24 寄存器。图片显示数据由处理芯片经由 SPI 总线传送给屏幕芯片，而具体每个像素点的刷新电压需要靠驱动电路来提供。

屏幕驱动电路层利用行电极与列电极来为每个电子墨水胶囊提供外部电场。屏幕在每个像素点上附加开关控制并且连接有对应电容。当屏幕需要刷新时，依据 SPI 端传来的信息，电子墨水屏将控制对应开关，在某一行电极开始，利用源极驱动扫描列电极，并为对应电容充电，完成此行工作后再利用栅极驱动扫描下一行电极。重复这项工作直到完成所有像素点的扫描，并最终实现整个屏幕的全局刷新。

2.2 无线供电技术

无线能量传输技术(Wireless Power Transfer，WPT)，即无线电力传输技术，最早起源于物理学家兼电气工程师尼古拉·特斯拉于 1890 年的无线输电实验，特斯拉已经成功证明无线能量传输是可行的。2008 年，无线充电联盟 WPC 成立，并推出 Qi 标准，Qi 取自汉语“气”的读音，意指无线的能量，下面是当前主流无线供电技术。

供电方式	电磁感应式	磁场共振式	无线电波式	电场耦合式
传输功率	5-15W	数 KW	≥100mW	1-10W
传输距离	≤1cm	≤10m	≥10m	≤1cm，但横向允许最大 4cm 偏移
供电效率	≥80%	≥50%	≥30%	70-80%
优点	能量传输效率高，布置简单	可以进行中距离能量传输	可以远距离能量传输 供电范围要求宽松	无发热问题 能量传输效率高
不足	能量传输距离较近，且需要精确对准	电磁辐射较大	能量转换效率较低，容易受到多径效应等影响接收功率	体积较大

表 二-2 无线供电技术对比

本文基于电子日历的使用场景，考虑到电子日历本身就放在桌面上，对于无线供电距离没有要求，结合供电效率的问题，磁场共振式与无线电波式无线供电不适合作为此场景供能技术。经过对比，结合成本与入手难度等考量，本文最终选择市面上最为常见的电磁感应式无线供电技术为整个系统提供能量供应。

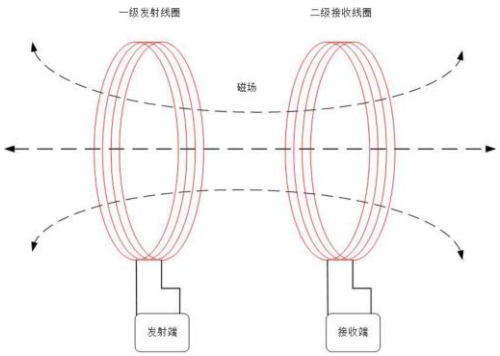

图 二-2 电磁感应式无线供电示意图

2.3 MQTT 协议

MQTT 全称为 Message Queuing Telemetry Transport，即消息队列遥测传输协议，是一种基于发布/订阅（Publish/Subscribe）模式的轻量级通讯协议，该协议基于 TCP/IP 协议，由 IBM 于 1999 年发布。该协议最大的优点是能够以较少的代码和有限的带宽为远程设备提供实时可靠的消息服务。MQTT 作为一种低开销、低带宽的即时消息协议，广泛应用于物联网、小型设备、移动应用等领域。MQTT 的核心部分为作为中间件的 Broker，其承担了数据接受和分发功能，本项目中我将选用巴法云的 MQTT 服务。其工作原理如图 2-3 所示。

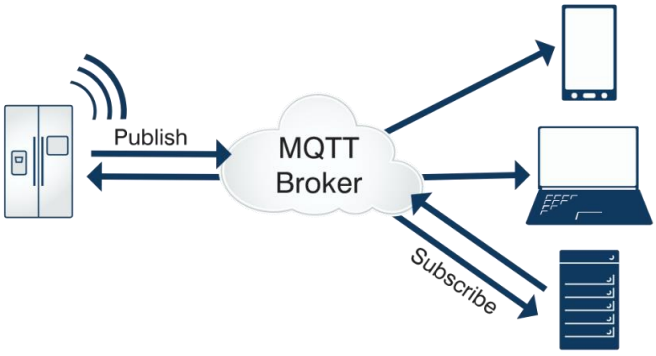

图 二-3 MQTT 工作原理图

2.4 ESP32-WROOM-32 开发板

ESP32 开发板是专为移动设备、可穿戴电子产品和物联网应用而设计的物联网开发板。ESP32 只需极少的外围器件，即可实现强大的处理性能、可靠的安全性能，拥有极高的集成度。ESP32 双核 CPU 工作频率为 80 至 240MHz，并且待机功率低，并且稳定，同时该开发板上集成了 3.3V 稳压模块，可驱动外设，ESP32 外观如图所示。

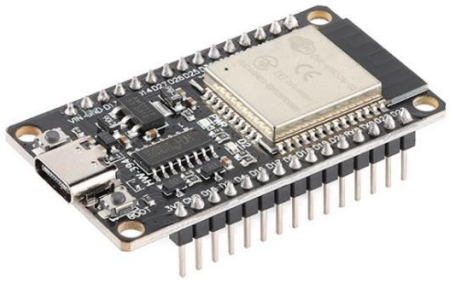

图 二-4 ESP32-WROOM-32 开发板

2.5 Arduino 平台

Arduino 是一个开源嵌入式硬件平台，用来供用户制作可交互式的嵌入式项目。此外 Arduino 作为一个开源硬件和开源软件的公司，同时兼有项目和用户社区。该公司负责设计和制造 Arduino 电路板及相关附件。这些产品按照 GNU 宽通用公共许可证（LGPL）或 GNU 通用公共许可证（GPL）许可的开源硬件和软件分发的，Arduino 允许任何人制造 Arduino 板和软件分发。

三、 系统需求分析与系统架构设计

3.1 需求分析

本系统是放置在桌面上的 E-link 设备——电子日历，用于实现信息展示与时间获取，同时使用手机 App 设置日历上显示的日程然后自动显示倒计天数，并且能方便的进行充电。

3.1.1 设备端

系统的设备端主要由 ESP32 开发板与 4.2 寸电子墨水屏组成，搭载了 Wi-Fi 模块用于无线传输和数据更新，具有墨水屏省电节能，成本较低，且功耗小的优势，仅在数据更新时需要供电，突发的断电等状况不影响其使用的特点，并且墨水屏在强光环境下显示依旧清晰，对使用者双眼刺激性较小，拥有在室内环境的良好适配性。主要功能如下：

1、信息展示：通过 Wi-Fi 模块接收 API 或者用户传送的 JSON 数据信息；以及通过 DHT11 获取的温湿度信息，将数据处理存放与指定的结构体中，再驱动墨水屏将数据进行刷新显示。

2、状态同步：在墨水屏启动或者受到请求时，将自身的状态通过串口输出，并进行墨水屏状态信息和 API 返回的信息进行同步。

3、智能切换有线与无线充电：默认打开无线充电关闭有线充电，当检测到有线充电时，自动关闭无线充电启用有线充电。

3.1.2 App 端

能适配大部分安卓手机，支持用户输入日程名称，并选择日期，随后封装为 JSON 格式最后发送给 MQTT 服务器，然后转发给设备端。

3.2 系统架构设计

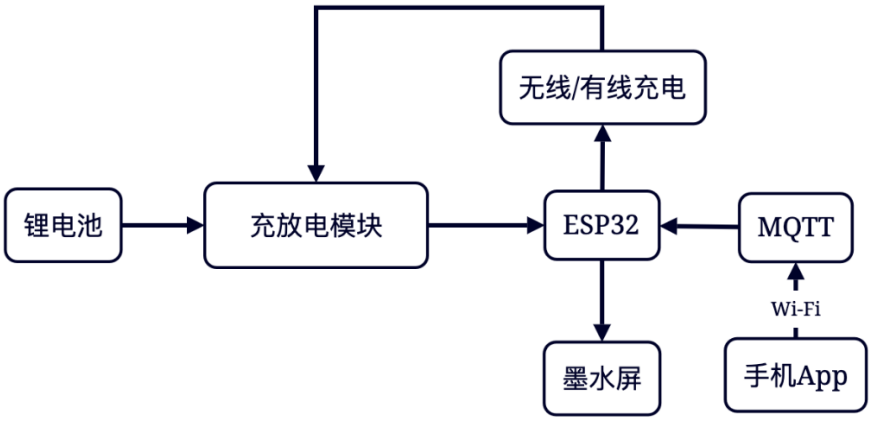

图 三-1 系统框架

四、 系统功能模块实现

4.1 系统软件设计

4.1.1 设备端

软件设计分为两个部分：Wi-Fi 连接与数据抓取部分；有线与无线充电控制部分。

Wi-Fi 连接与数据抓取：

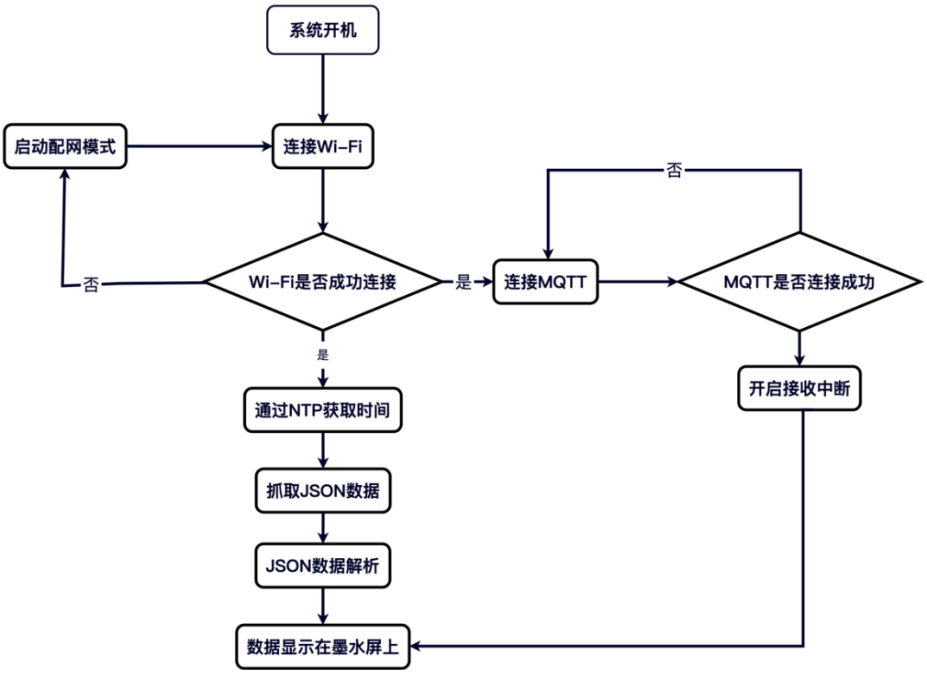

图 四-1 Wi-Fi 连接与数据抓取

解析 JSON 数据，以解析日程为例（完整代码请见附件）。

首先为 JSON 对象分配一定到空间，然后对其进行反序列化得到一个对象，若返回值 error≠ 0，则按需取出 JSON 中的值放入结构体中带回；若 error=0，则串口输出错误，并返回。

```
void ParseMindDay(String content, struct MindDay* data)
{
  DynamicJsonDocument json(128);//分配内存
  DeserializationError error = deserializeJson(json, content); //解析 json
  //serializeJson(json, Serial);//构造序列化 json,将内容从串口输出
  if (error){
    Serial.print("倒计时时间错误");
    return;
  }else{
    int Year = json["Year"];
    int Month = json["Month"];
    int Day = json["Day"];
    data->year = Year;//t 通过结构体指针带出
    data->month = Month;
    data->day = Day;
    strcpy(data->festival,json["JieRi"]);
  }  }
```

天数差计算核心算法

通过 Unix 时间戳来进行计算，通过 Unix Time 库函数将日程日期转换为 Unix 时间戳：unix，然后再与 NTP 服务器获得的时间戳：epochTime，二者相减得到秒数差，最后除以一天的秒数 86400，得到天数差。这个算法的优势在于，在计算时间差时可以完全避开闰年，大小月等复杂的时间规则，直接转换到秒进行计算，简单直观。

```
void displayday(unsigned long epochTime,struct MindDay jieri,int hour,int minute)
{//传入当前时间戳与当前时间
  stamp.setDateTime(jieri.year,jieri.month,jieri.day,hour,minute,0);
  uint32_t unix = stamp.getUnix();//将日程日期转换为 Unix 时间戳
  int difference = (unix - epochTime) / secondsPerDay;
  /*日程日期的时间戳-当前日期的时间戳，得到之间的秒数差
    然后除以一天的秒数 86400，得到天数差*/
  //显示到墨水屏上……
}
```

有线与无线充电控制部分：

由于充放电优先级较高，便使用外部中断来对充放电进行检测；首先将 D33 配置为下拉输入，用于检测有线充电；然后将 D32 与 D26 配置为输出模式，用于控制有线充电与无线充电的 MOS 管。当 D33 引脚的电压发送变化时，以插入有线充电为例，D33 由 0 变为 3.3V，触发中断，调用中断函数，更新中断标志位为 FLAG_CHARGE = 1，然后在主循环中将 D32 设置为低电平，

关闭控制无线充电的 MOS，D26 设置为高电平，打开控制有线充电的 MOS，如图 4-2 所示。

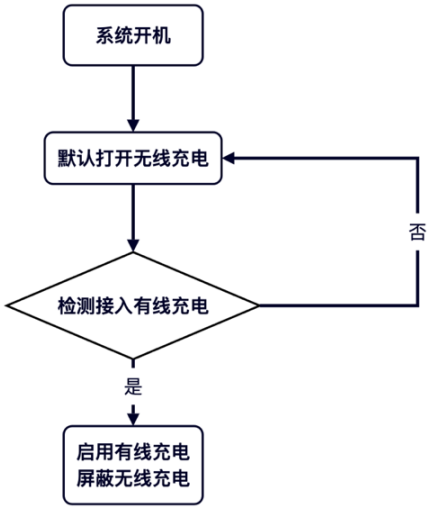

图 四-2 有线与无线充电控制

中断设置：

```
1.   pinMode(interruptPin_0, INPUT_PULLUP);  //先把引脚设置为上拉输入
2.   pinMode(inputPin, INPUT_PULLDOWN); // 设置 D33 为输入并启用内部下拉
3.   pinMode(outputPin1, OUTPUT); // 设置 D32 为输出
4.   pinMode(outputPin2, OUTPUT); // 设置 D26 为输出
5.   digitalWrite(outputPin1, HIGH);//D32 高电平启用无线充电
6.   digitalWrite(outputPin2, LOW);//D26 低电平屏蔽有线充电 默认启用无线
7.  attachInterrupt(digitalPinToInterrupt(inputPin),handleInput,CHANGE);
```

中断函数：

```
1. //充电检测中断
2. void IRAM_ATTR handleInput(){
3.   if(digitalRead(inputPin) == 1)
4.   {FLAG_CHARGE = 1;//启用有线 }
5.   else
6.   {FLAG_CHARGE = 2;//启用无线}}
```

主循环中的控制函数：

```
1. //充电选择判断
2. if (FLAG_CHARGE == 1){
3.   digitalWrite(outputPin1, LOW);
4.   digitalWrite(outputPin2, HIGH);
5.   Serial.println("启用有线充电");
6.   FLAG_CHARGE = 0;//清除中断标志
7. }else if(FLAG_CHARGE == 2){
8.   digitalWrite(outputPin1, HIGH);
9.   digitalWrite(outputPin2, LOW);
10.  Serial.println("启用无线充电");
11.  FLAG_CHARGE = 0;//清除中断标志
12. }
```

4.1.2 App 端

使用 MIT App Inventor 通过可视化的方式快速开发了日程设置 App。图 4-3 左侧为配置 MQTT 使其连接上巴法云的 MQTT 服务器，右侧为用户交互，接收用户输入的文本并封装为 JSON 发送。

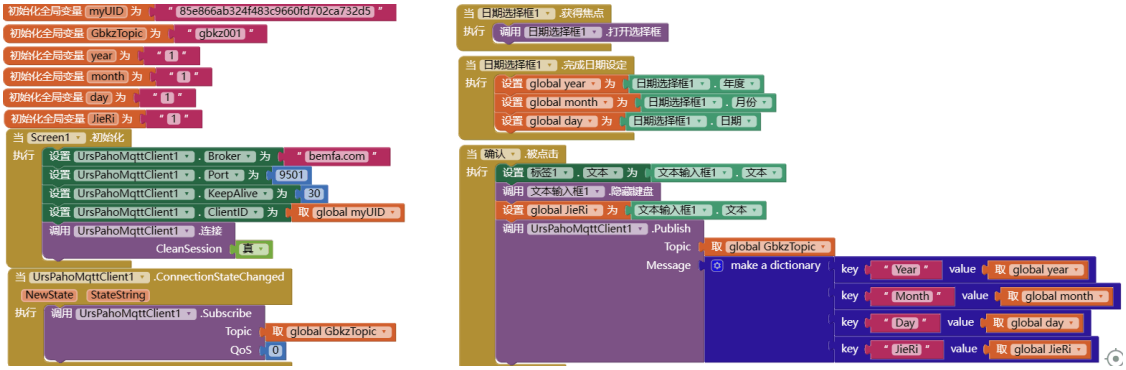

图 四-3 App Inventor

图 四-4 App 启动流程

4.2 电子墨水屏驱动电路

电子墨水屏要实现信息显示，除了由 ESP32 通过 SPI 总线控制显示信息外，还需要专门的驱动电路来为显示屏内部电路提供必需的电压控制与信息反馈。电子墨水屏电路主要包括以下几个部分:

1.振荡器，负责为内部 MCU 提供时钟。

2.时序逻辑电路，为控制刷新提供时序顺序控制。

3.外设接口，提供了有限的外设接口，允许为电子墨水屏添加诸如温度传感器之类的外设，来为特殊情况下的使用提供更精确的控制。

4.静态随机存取存储器 SRAM(Static Random-Access Memory)，提供了图片查找表 LUT(Lookup Table)，可以对传入的图片显示信息进行加工，已提供更优质的显示。

5.VCOM，电子墨水屏内部电路具有升压模块可以产生 VCOM 电压，利用 VCOM 驱动屏幕中的电子墨水进行显示，实际图片的显示灰阶由粒子在微胶囊中的相对空间位置决定，黑色与白色微胶粒在 VCOM 电压作用下分别向不同方向移动，最终停留的相对位置形成不同灰度的显示效果。

了解电子墨水屏内部电路后，需要设计实现驱动电路为其提供所需的各种电压与反馈。驱动电路部分电路图如图所示。

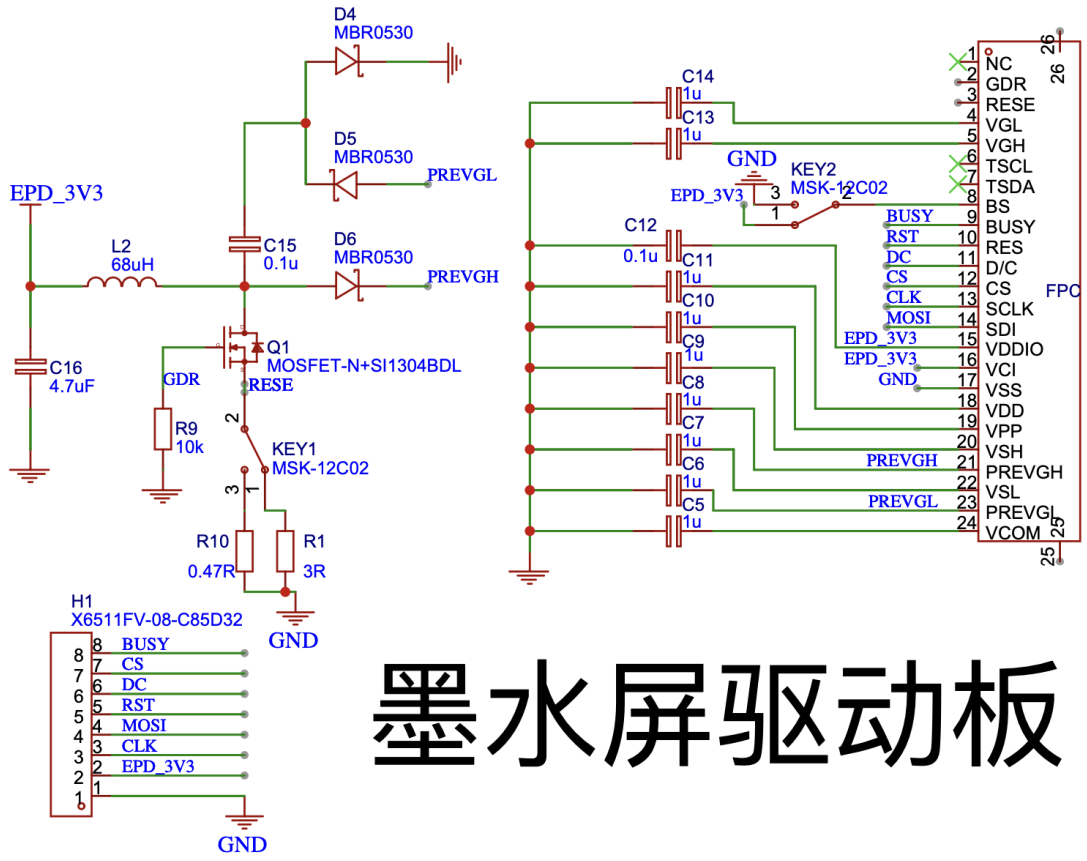

图 四-5 墨水屏驱动电路

其中 EPD_3V3 为整个驱动电路提供电源。EPD_3V3 由 ESP32 开发板提供，用 EPD_3V3 为驱动电路供能。EPD_3V3 并联电容 C16 后串联电感 L2，电感与电容不会对直流供电产生影响，但是可以防止电路电压瞬间上升损坏电路器件，可以起到保护电路元件的作用。还具有高频滤波以及抗干扰的效果。

图 4-1 右侧，由 FPC 连接器自上而下来看。首先 GDR 用来驱动控制 N 沟道 MOSFET 栅极，RESE 则为控制环路的电流检测提供输入电流。如图 4-1 左侧所示，GDR 线和 RESE 线分别连接在 MOSFETQ1 的栅极与源极上;同时源极下方使用了 KEY1 用于选择电阻阻值（0.47R/3R）；VGH 与 VGL 脚，为电子墨水屏内部电路提供栅极驱动电压，其中 VGH 提供正电压而 VGL 提供负电压。下侧的 VSH、VSL 脚和 VGH、VGL 脚对应，为内部电路提供源极驱动电压，其中 VSH 为正压，VSL 提供负压。这四个控制脚控制内部电路 MOS 管的开关，其中 VGH 与 VSH 电源由管脚 PREVGH 提供，VGL、VSL 电源由 PREVGL 提供，线路连接如图右侧所示。

TSCL 与 TSDA 脚为内部电路为外设温度传感器预留的 I^2C 接口，由于温度传感器的使用仅限于一些较为极端的使用环境，我们未添加温控外设，因此对这两个管脚不做处理。

BS 脚为总线选择引脚，负责选择 SPI 总线类型，如果置为高则表明接入为三线 SPI，置为低表明为 4 线 SPI 总线，故设计了 KEY2 用于 BS 总线选择。此处的四线 SPI 包括 D/C、CS、SCLK(serial clock)，SDI(serial data)四条线，相比三线 SPI 多了 D/C 总线，可以对发送数据与命令进行控制。BUSY 脚用来表明内部电路状态，如果处于忙碌状态则应停止操作等待该引脚空闲。VCOM 脚由 PREVGL 脚提供电源，负责驱动电子墨水进行显示。

4.3 电子墨水屏通信底座电路设计

通信底座电路分为两个部分，一个是充放电控制电路，一个是外设与 ESP32 通信电路。

4.3.1 充放电控制电路

我们在保留了传统的有线充电的同时，加入了无线充电，便于用户通过多种方式进行设备充电，如图 4-6 所示，使用 U10 与 U11 两个 MOS 管触发开关控制有线与无线的充电，同时使用 TP4056 模块管理电池的充放电，然后使用一个 5V-3.3V 电平转换模块将有线充电的 5V 转换为 3.3V 供 ESP32 检测。

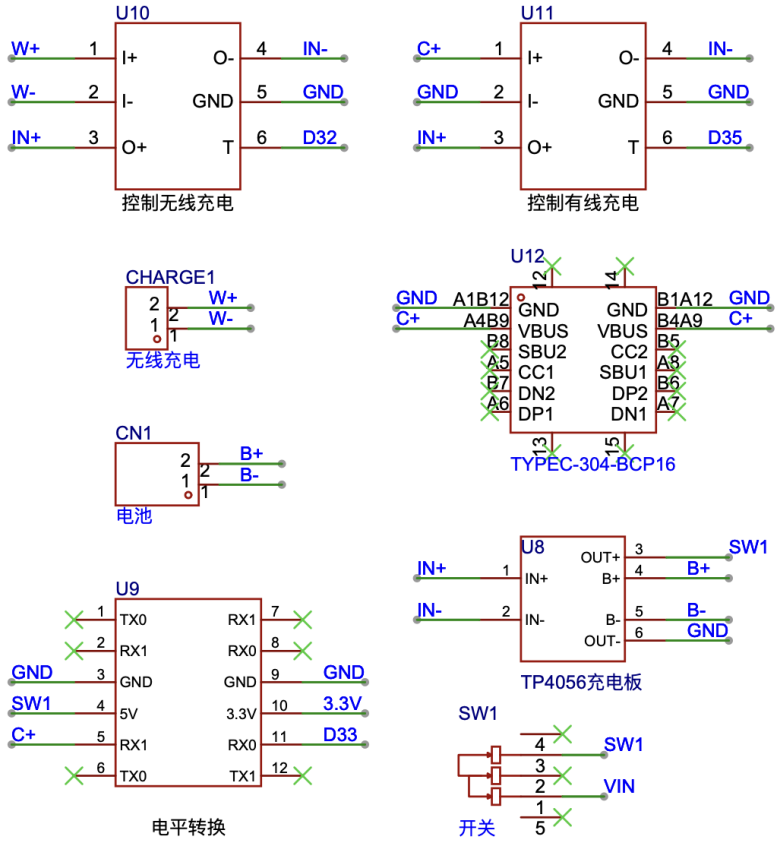

图 四-6 充放电控制电路

4.3.2 外设与 ESP32 通信电路

如图四-7 所示，将充放电控制电路，温湿度传感器，按键，墨水屏接口组合起来，同时按键搭配了 0.1uf 的消抖电容。

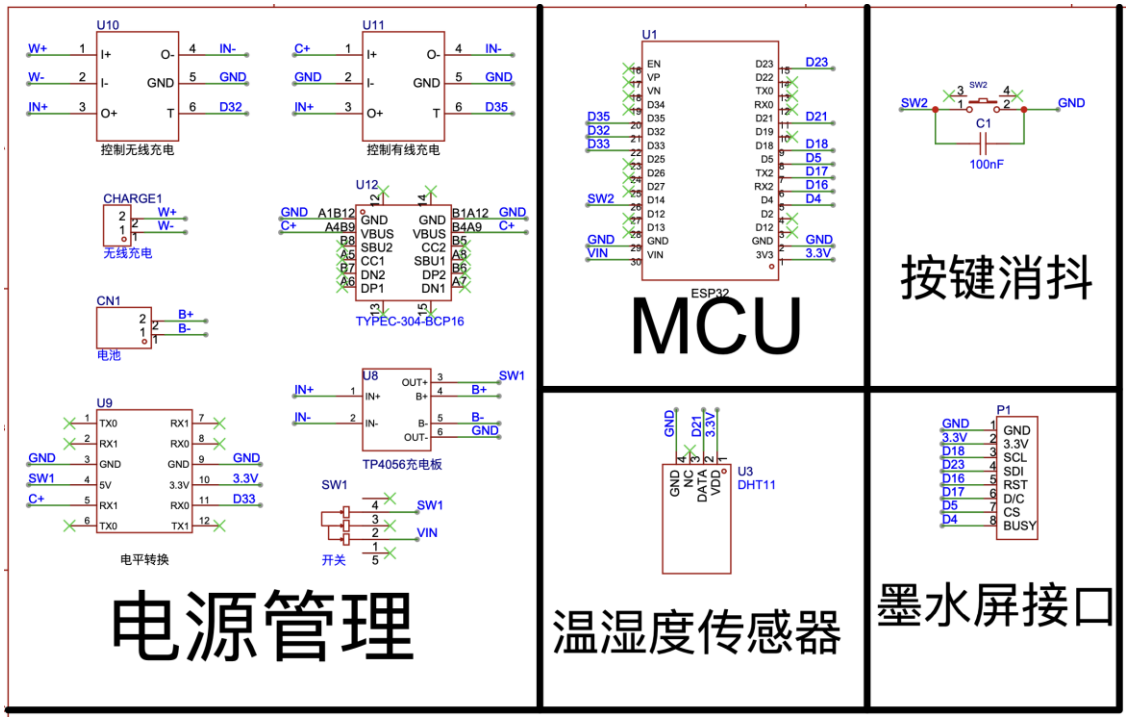

图 四-7 底座整体设计原理图

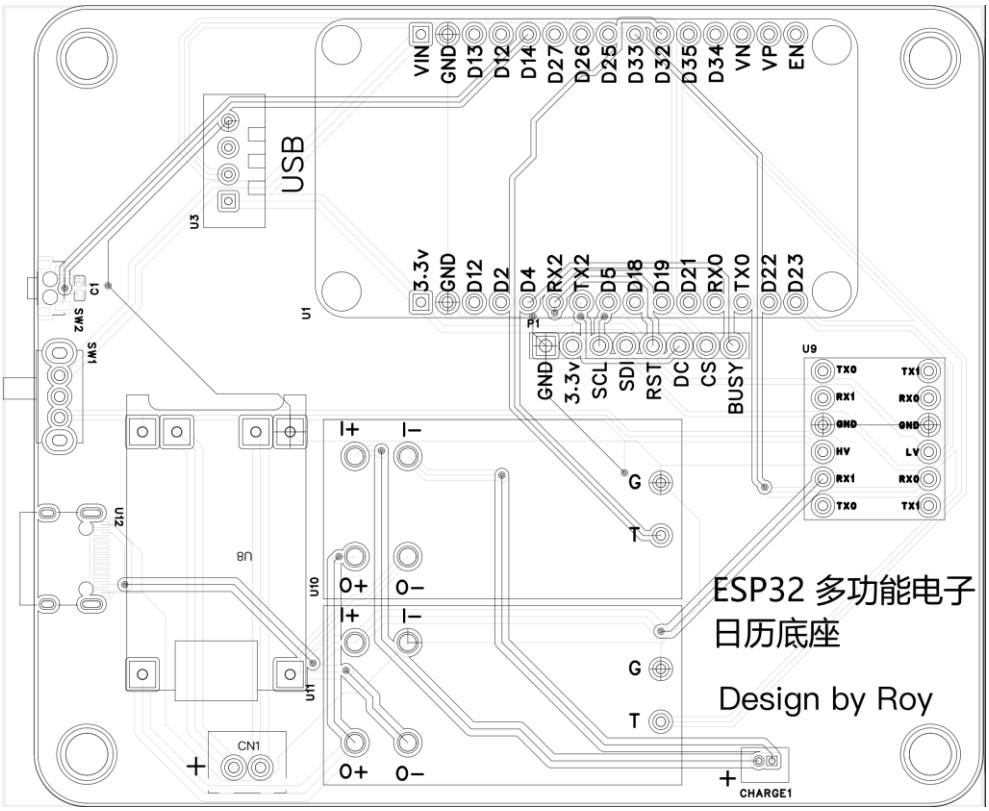

图 四-8 底座 PCB 设计

4.4 外壳建模与 3D 打印

如图所示，我们对其的外壳进行了 3D 建模，图 4-3 可以看出，左侧有接口与控制按键和温湿度传感器的开孔，四周为 PCB 的定位孔便于装配，中间的凹槽为无线充电线圈的定位槽，用于线圈的固定，外侧做了凹槽使得上盖与底座更加容易贴合，并且做了圆角设计使得外壳更加美观。图 4-4 为按钮模型，用于延长按钮，使得用户更加容易操作按键。

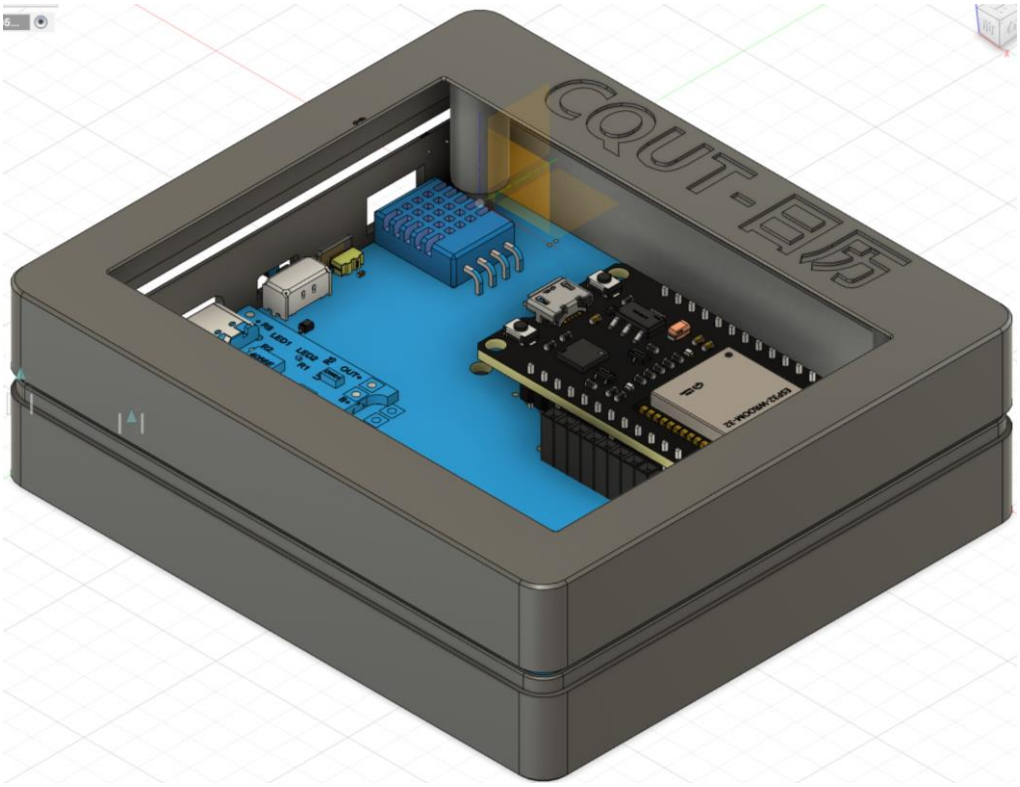

图 四-9 整体外壳模型

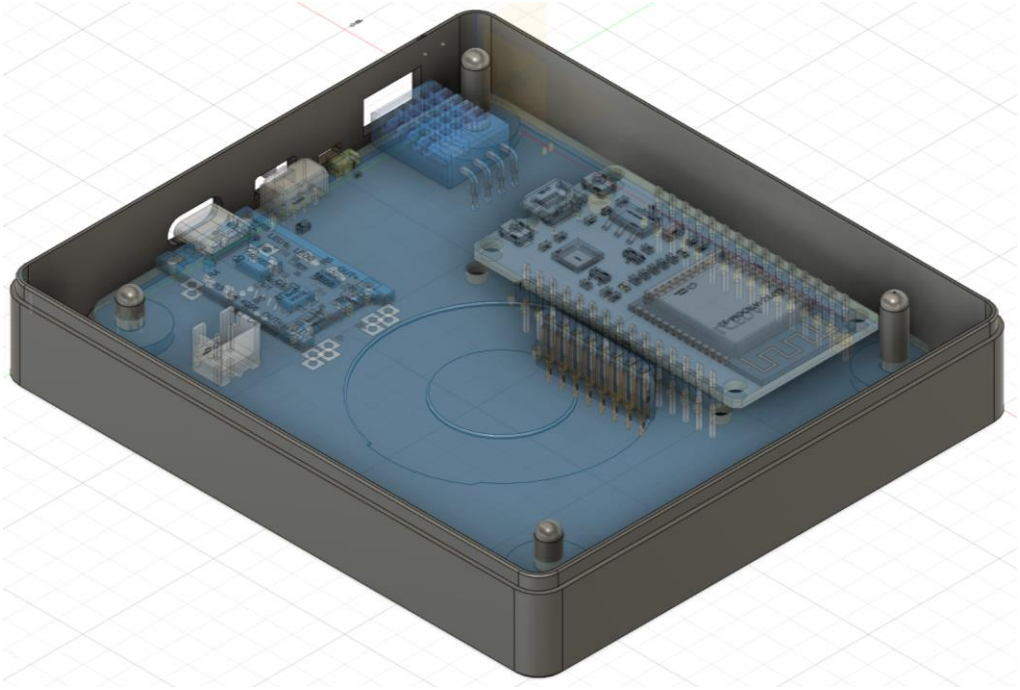

图 四-10 底座模型

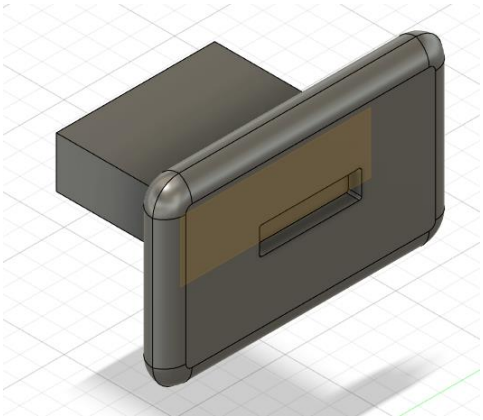

图 四-11 按键模型

五、 调试与演示结果

5.1 调试：

PCB 经历了两个大版本迭代，最后各个模块工作正常。

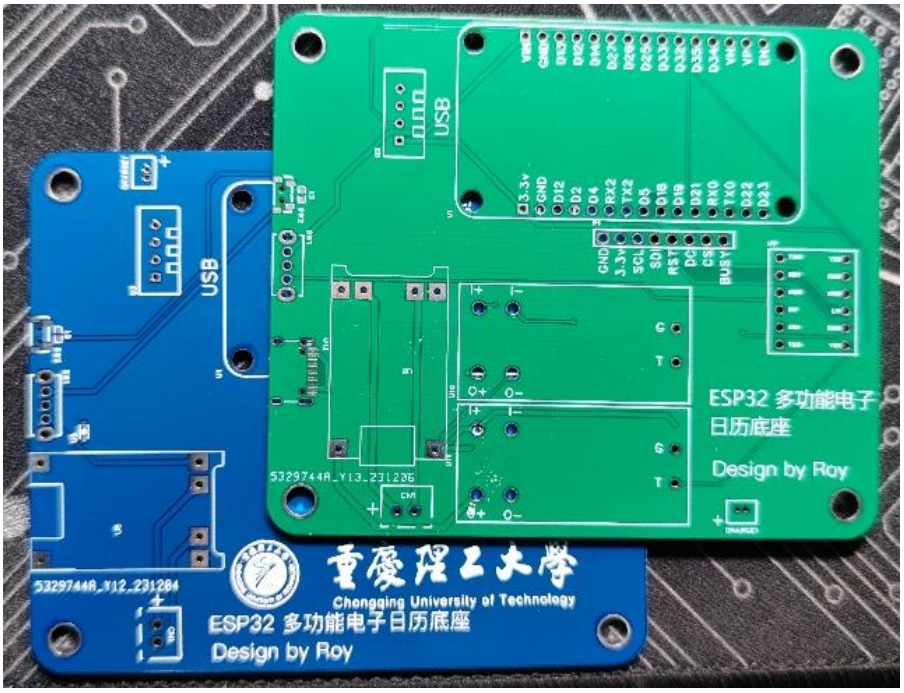

图 五-1 成品 PCB 样板

3D 模型经历了三个大版本迭代，由于对屏幕的机械尺寸考虑欠佳，以及 3D 打印机的误差，导致前两个版本屏幕无法放入，在第三个版本中成功容纳各个零部件。

图 五-2 3D 打印模型

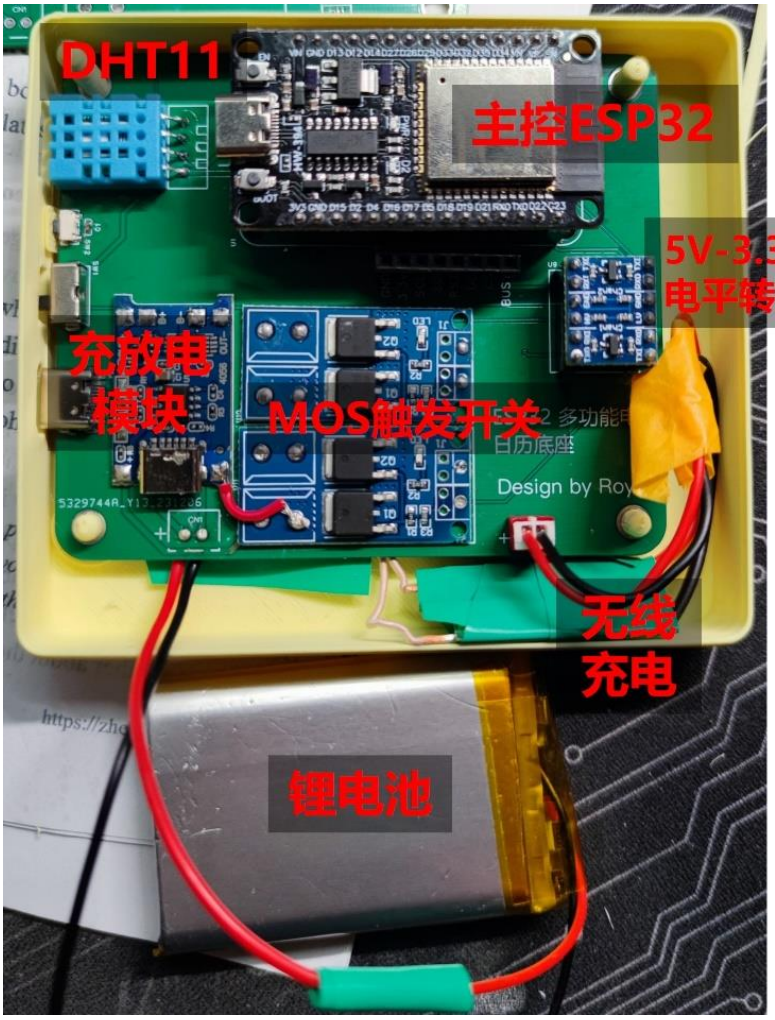

图 五-3 组装完成

5.2 作品最终效果演示：

图 五-4 最终效果

六、 总结与反思

6.1 全文总结

鉴于目前传统的日历存在的不足，我们设计了一个基于 ESP32 的多功能电子日历，该系统使用有线充电与电磁感应技术的无线充电作为充电，利用 ESP32 开发板完成各种数据的抓取，并可以驱动电子墨水屏幕进行信息显示。我们主要完成的工作有：

（1） 完成了对无线充电技术，MQTT 协议，电子墨水屏显示技术的了解，并通过对比不同的技术，选择了电磁感应技术与 MQTT 协议与电子墨水屏和 ESP32 来完成系统功能。

（2） 基于电子墨水屏的显示原理，设计实现了电子墨水屏的驱动电路。

（3） 使用 App inventor 完成了手机 App 的设计，成功实现了向服务端发送日程信息

（4） 使用 Arduino 框架完成了 ESP32 设备端的开发，完成了以下功能：

a) 网络连接：设备能够连上 Wi-Fi 网络，这为获取实时数据和在线更新提供了基础。

b) 日历显示：电子墨水屏幕上清晰地展示了当前日期和月份，方便用户随时查看。

c) 天气预报：设备能够获取并显示未来三天的天气预报，包括温度、天气状况等信息，为用户的出行和活动安排提供参考。

d) 每日一言：为了增添生活趣味和启发思考，日历每次刷新都会更新一条引人深思或者鼓舞人心的语句。

e) 日程倒计时：用户可以设定个人日程，并在屏幕上看到这些事件的倒计时天数，有助于提醒和管理时间。

（5） 智能充电切换：支持有线和无线两种充电方式，并能够智能切换，既方便用户在不同环境下使用，又提高了设备的续航能力。

6.2 反思：

尽管这款基于 ESP32 的电子墨水屏幕日历具有诸多优点，但在以下几个方面仍有改进和优化的空间：

（1） 用户体验：虽然电子墨水屏适合长时间显示固定信息，但其刷新速度较慢。可以研究如何优化屏幕刷新算法，提高用户体验。

（2） 多语言支持：为了满足不同地区和文化背景的用户需求，可以考虑增加多语言支持功能，同时需要解决字库文件过大，ESP32 自带 FLASH 过小的问题，可以考虑使用字库芯片，或者外挂 FLASH 实现。

（3） 自定义选项：允许用户根据个人喜好定制显示的内容和样式，如选择不同的字体、颜色或者自定义每日一言的来源。

（4） 能耗优化：虽然电子墨水屏本身具有低功耗特性，但仍可进一步优化设备的电源管理，例如在无操作或特定时间段自动进入休眠模式，以延长电池寿命。

（5） 手机 App 功能进一步优化：UI 设计优化由于时间的原因，本次开发出来的 App 界面粗糙，虽然符合用户的操作便的要求，但是整体上来说 UI 设计需要进一步改进。

参考文献

[1] 刘美健.无线网络电子价签系统的设计与实现[J].自动化与仪表, 2014, 29(4):4.DOI:10.3969/j.issn.1001-9944.2014.04.012.
[2] 徐辉波,牛晓伟,路新成,等.微胶囊化电子墨水研究进展[J].化工时刊, 2008, 22(5):5.DOI:10.3969/j.issn.1002-154X.2008.05.017.
[3] 周国富,易子川,王利,等.电泳电子纸驱动波形研究现状与前景[J].华南师范大学学报：自然科学版, 2013, 45(6):6.DOI:10.6054/j.jscnun.2013.09.007.
[4] 杨雷,余芳,童立.超低功耗电子标签系统[J].物联网技术, 2019, 9(5):4.DOI:10.16667/j.issn.2095-1302.2019.05.005.
[5] 浩钿,王猛,陈聪,等.基于无线通信和墨水屏的电子标识技术研究[J].无线互联科技, 2019, 16(9):3.DOI:10.3969/j.issn.1672-6944.2019.09.002.
[6] 贺轶烈,许晓荣,楼丁溧,等.基于电子墨水屏的无线电子标签设计[J].电子设计工程, 2019, 27(6):4.DOI:CNKI:SUN:GWDZ.0.2019-06-032.
[7] 连丽红,蔡剑文.基于树莓派的智能万年历[J].物联网技术, 2021.DOI:10.16667/j.issn.2095-1302.2021.03.029.

附录

主程序：

```
#include <U8g2_for_Adafruit_GFX.h>
#include <GxEPD2_3C.h>
//#include <GxEPD2_BW.h>//适用于中景园电子的屏幕，黑白
#include <ArduinoJson.h>
#include <Timezone.h>
//GxEPD2_BW<GxEPD2_420_GDEY042T81, GxEPD2_420_GDEY042T81::HEIGHT> display(GxEPD2_420_GDEY042T81(/*CS=5*/ SS, /*DC=*/ 17, /*RST=*/ 16, /*BUSY=*/ 4)); GDEY042T81, 400x300, SSD1683 (no inking)
GxEPD2_3C<GxEPD2_420c, GxEPD2_420c::HEIGHT> display(GxEPD2_420c(/*CS=5*/ SS, /*DC=*/17, /*RST=*/16, /*BUSY=*/4));  // GDEW042Z15
#include "gb2312.c"
#include "u8g2_mflansong_36_gb2312a.h"
#include "u8g2_mflansong_24_gb2312a.h"
#include "imagedata.h"
#include "Webserver.h"
#include "PubSubClient.h"  //MQTT
#include <dht11.h>
#include <NTPClient.h>
#include <WiFi.h>  // for WiFi shield
#include <WiFiUdp.h>
#include <TimeLib.h>
#include <UnixTime.h>
extern const uint8_t chinese_city_gb2312[239032] U8G2_FONT_SECTION("chinese_city_gb2312");
#define secondsPerDay 86400;
U8G2_FOR_ADAFRUIT_GFX u8g2Fonts;
//中断配置
const byte interruptPin_0 = 14;      //这就是我们设置中断的目标对应的那个引脚
const int inputPin = 33;             //输入引脚用于检测 5v
const int outputPin1 = 32;           //控制无线充电
const int outputPin2 = 26;           //控制有线充电
//portMUX_TYPE mux = portMUX_INITIALIZER_UNLOCKED;  //声明一个portMUX_TYPE 类型的变量，利用其对主代码和中断之间的同步进行处理
//******MQTT 服务配置**********
const char* topic = "esp32calender";                //主题名字，可在巴法云控制台自行创建，名称随意
```

```
#define ID_MQTT "e2a9e2c5559e48afa9c7eb4145002640"  //修改，你的 Client ID
const char* mqtt_server = "bemfa.com";              //默认，MQTT 服务器
const int mqtt_server_port = 9501;                  //默认，MQTT 服务器
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;  //计时
//****** HTTP 服务器配置 ******
String language = "zh-Hans";                    // 请求语言
String url_yiyan = "https://v1.hitokoto.cn/";  //一言获取地址
String url_ActualWeather;                       //实况天气地址
String url_FutureWeather;                       //未来天气地址
//****** 天气数据
//我们要从此网页中提取的数据的类型
struct ActualWeather {
  char status_code[64];   //错误代码
  char city[16];          //城市名称
  char weather_name[16];  //天气现象名称
  char weather_code[4];   //天气现象代码
  char temp[5];           //温度
  char last_update[25];   //最后更新时间
};
ActualWeather actual;  //创建结构体变量 目前的
struct FutureWeather {
  char status_code[64];  //错误代码
  char date0[14];              //今天日期
  char date0_text_day[20];     //白天天气现象名称
  char date0_code_day[4];      //白天天气现象代码
  char date0_text_night[16];   //晚上天气现象名称
  char date0_code_night[4];    //晚上天气现象代码
  char date0_high[5];          //最高温度
  char date0_low[5];           //最低温度
  char date0_humidity[5];      //相对湿度
  char date0_wind_scale[5];    //风力等级
  char date1[14];              //明天日期
  char date1_text_day[20];     //白天天气现象名称
  char date1_code_day[4];      //白天天气现象代码
  char date1_text_night[16];   //晚上天气现象名称
  char date1_code_night[4];    //晚上天气现象代码
  char date1_high[5];          //最高温度
  char date1_low[5];           //最低温度

```

```
  char date2[14];              //后天日期
  char date2_text_day[20];     //白天天气现象名称
  char date2_code_day[4];      //白天天气现象代码
  char date2_text_night[16];   //晚上天气现象名称
  char date2_code_night[4];    //晚上天气现象代码
  char date2_high[5];          //最高温度
  char date2_low[5];           //最低温度
};
FutureWeather future;  //创建结构体变量 未来
struct LifeIndex       //生活指数
{
  char status_code[64];  //错误代码
  char uvi[10];          //紫外线指数
};
LifeIndex life_index;  //创建结构体变量 未来
struct News            //新闻 API
{
  char status_code[64];  //错误代码
  char title[11][64];
};
struct Hitokoto  //一言 API
{
  char status_code[64];  //错误代码
  char hitokoto[64];
};
struct MindDay  //倒计时未来时间
{
  char festival[64];  //节日
    uint16_t year;//年
    uint8_t month;//月
    uint8_t day;//日
  };
  //****** 一些变量 ******
  String webServer_news = " ";
  boolean night_updated = 1;  //夜间更新 1-不更新 0-更新 为了省电还是不更新吧
  WiFiUDP ntpUDP;
  NTPClient timeClient(ntpUDP, "ntp.aliyun.com");  //NTP 服务器地址
  int k = 0;           //用于判断页面
  int FLAG_KEYIT = 0; //按键
  int FLAG_CHARGE = 0;//充电
  //RTC 临时数据
  #define RTC_hour_dz 0             //小时地址
  #define RTC_night_count_dz 1      //夜间计数地址
```

```
#define RTC_peiwang_state_dz 2    //配网状态地址
uint32_t RTC_hour = 100;          //小时
uint32_t RTC_night_count = 0;     //24-6 点，夜间不更新计数
int32_t night_count_max = 0;      //需要跳过几次
uint32_t RTC_peiwang_state = 0;   //配网状态 1-需要
Hitokoto yiyan;                           //创建结构体变量 一言
News xinwen;                              //创建结构体变量 新闻
MindDay jieri = { "元旦", 2024, 1, 1 };   //创建结构体变量 节日
//测试
int udc = 0;                //更新次数记录
unsigned long epochTime;    //当前时间戳
UnixTime stamp(8);          //设置为北京时间 GMT 8

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect(ID_MQTT)) {
      Serial.println("connected");
      Serial.print("subscribe:");
      Serial.println("esp32calender");
      //订阅主题，如果需要订阅多个主题，可发送多条订阅指令client.subscribe(topic2);client.subscribe(topic3);
      client.subscribe("esp32calender");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
/*中断函数*/
//MQTT 接收中断
String msg = "";
void callback(char* topic, byte* payload, unsigned int length) {
  // Serial.print("Topic:");//topic 消息接收
  // Serial.println(topic);
  msg = "";
  for (int i = 0; i < length; i++) {
    msg += (char)payload[i];
  }
```

```
158.     // Serial.print("Msg:");
159.     // Serial.println(msg);
160.     FLAG_KEYIT++;
161.   }
162.   void IRAM_ATTR handleInput() {//充电检测中断
163.     if (digitalRead(inputPin) == 1) {
164.       FLAG_CHARGE = 1;  //启用有线
165.     } else if (digitalRead(inputPin == 0)){
166.       FLAG_CHARGE = 2;  //启用无线
167.     }
168.   }
169.   void handleInterrupt() {//按键中断回调函数
170.     if (digitalRead(interruptPin_0) == 0)  //因为是下拉触发，所以在消抖
    时间完后读取引脚高低电平，如果还是为低那么就代表出现了一次稳定的中断
171.     {
172.       FLAG_KEYIT++;
173.     }
174.   }
175.   void setup() {
176.     Serial.begin(115200);
177.     EEPROM.begin(4096);
178.     display.init();
179.     u8g2Fonts.begin(display);  //将 u8g2 连接到 display
180.     display.firstPage();
181.     display.display(1);
182.     pinMode(interruptPin_0, INPUT_PULLUP);
                        //先把引脚设置为上拉输入
183.     pinMode(inputPin, INPUT_PULLDOWN);
                        // 设置 D33 为输入并启用内部下拉
184.     pinMode(outputPin1, OUTPUT);
                        // 设置 D32 为输出
185.     pinMode(outputPin2, OUTPUT);
                        // 设置 D26 为输出
186.     digitalWrite(outputPin1, HIGH);
                        //D32 高电平启用无线充电
187.     digitalWrite(outputPin2, LOW);
                        //D26 低电平屏蔽有线充电 默认启用无线
188.     attachInterrupt(digitalPinToInterrupt(interruptPin_0), handleInt
    errupt, FALLING);  //按键中断切换页面
189.     attachInterrupt(digitalPinToInterrupt(inputPin), handleInput, CH
    ANGE);              // 在 D33 上设置中断处理函数
190.     wifi_init();//NTP 服务 启动!
191.     timeClient.begin();
```

```
192.     timeClient.setTimeOffset(28800);  // + 1 区 偏移 3600，  +8 区 ：
    3600×8 = 28800
193.     client.setServer(mqtt_server, mqtt_server_port);
194.     client.setCallback(callback);//MQTT  启动!
195.   }
196.
197.   void loop() {
198.     if (settingMode)  //联网判断
199.     {
200.       dnsServer.processNextRequest();
201.     }
202.     webServer.handleClient();
203.     if (!settingMode) {
204.       if (!client.connected()) {
205.         reconnect();
206.       }
207.       client.loop();
208.       timeClient.update();//获取时间
209.       epochTime = timeClient.getEpochTime();
210.       // Serial.print("Epoch Time: ");
211.       // Serial.println(epochTime);//串口输出当时的 Unix 时间戳
212.       int currentHour = timeClient.getHours();
213.       int currentMinute = timeClient.getMinutes();
214.       int weekDay = timeClient.getDay();
215.       struct tm* ptm = gmtime((time_t*)&epochTime);
216.       int monthDay = ptm->tm_mday;
217.       int currentMonth = ptm->tm_mon + 1;
218.       long now = millis();
219.       if (k == 0 ||(now - lastMsg > 600000))  //10 分钟 刷新一次
220.       {
221.         lastMsg = now;
222.         k = 1;
223.         FLAG_KEYIT = 0;
224.         GetData();// 开始显示
225.         display.fillScreen(GxEPD_WHITE);
226.         get_weather();
227.         timedisplay(currentMonth, monthDay, weekDay, currentHour, cu
    rrentMinute);
228.         hitokoto();
229.         newsdisplay();
230.         gettem(); //特殊图标测试
231.         u8g2Fonts.setFont(u8g2_font_siji_t_6x10);
232.         u8g2Fonts.setForegroundColor(GxEPD_BLACK);  // 设置前景色
233.         u8g2Fonts.setBackgroundColor(GxEPD_WHITE);  // 设置背景色
```

```
234.         u8g2Fonts.drawGlyph(382, 30, 0x0e21a);      //显示 Wi-Fi 图标
235.         display.nextPage();
236.         udc++;//用于新闻内容的切换，10 分钟++，一天 24*60/10=144 次，用个
    一两年问题不大不重启
237.       }
238.       if (k == 1&&FLAG_KEYIT == 1) {
239.         k = 2;
240.         display.fillScreen(GxEPD_WHITE);
241.         get_weather();
242.         timedisplay(currentMonth, monthDay, weekDay, currentHour, cu
    rrentMinute);
243.         hitokoto();
244.         ParseMindDay(msg, &jieri);  //json 解析
245.         displayday(epochTime, jieri, currentHour, currentMinute);
246.         gettem();
247.         u8g2Fonts.setFont(u8g2_font_siji_t_6x10);
248.         u8g2Fonts.setForegroundColor(GxEPD_BLACK);  // 设置前景色
249.         u8g2Fonts.setBackgroundColor(GxEPD_WHITE);  // 设置背景色
250.         u8g2Fonts.drawGlyph(382, 30, 0x0e21a);      //显示 Wi-Fi 图标
251.         display.nextPage();
252.         Serial.println(FLAG_KEYIT);
253.       }else if (FLAG_KEYIT == 2){
254.         k = 0;
255.       }
256.       if (FLAG_CHARGE == 1) {
257.         digitalWrite(outputPin1, LOW);
258.         digitalWrite(outputPin2, HIGH);
259.         FLAG_CHARGE = 0;
260.       } else if (FLAG_CHARGE == 2) {
261.         digitalWrite(outputPin1, HIGH);
262.         digitalWrite(outputPin2, LOW);
263.         FLAG_CHARGE = 0;
264.       }
265.     }
266.     delay(1000);  //延时 1s
267.   };
```

<table>
<tr><td>评阅意见</td><td colspan="7"></td></tr>
<tr><td rowspan="2">考核项目</td><td colspan="4">实做成绩(含答辩）60%</td><td colspan="3">课程论文（含开题报告） 40%</td></tr>
<tr><td>硬件系统 15 分</td><td>软件系统 15 分</td><td>成果展示 10 分</td><td>设计答辩 20 分</td><td>方案与设计 20 分</td><td>结论与分析 10 分</td><td>团队合作 10 分</td></tr>
<tr><td>得分</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>总分</td><td></td><td>考核等级</td><td></td><td>教师签名</td><td colspan="3"></td></tr>
</table>