華南農業大學

本科毕业论文

基于 ARM 的纯水 TOC 检测系统设计

肖傲

202034410225

指导教师　　严炳辉　讲师

学院名称	电子工程学院（人工智能学院）	专业名称	电子信息科学与技术
论文提交日期	2024 年 4 月 15 日	论文答辩日期	2024 年 5 月 12 日

摘　　要

随着电子信息技术的不断发展，传感器技术日益强大。传感器精度的提升引发了人们对生物化学行业的关注，将电子信息与生物化学相结合成为了新的研究方向。在纯水TOC(Total Organic Carbon)检测系统中，过去的 TOC 检测方法存在体积庞大、不易移植的问题。因此，人们开始转向直接电导法。然而，该方法在国内仍相对较新颖，且受限于高昂的设备价格和缺乏二次开发接口与相关资料。因此，迫切需要设计出精简、成本合理的纯水检测系统。

本文提供了一种基于 ARM(Advanced RISC Machine)架构的纯水 TOC 检测系统的设计思路，本系统是基于实验室的应用环境，选用 Cortex-M4 内核的 STM32F407ZGT6 开发板进行开发设计。这是一种采用紫外氧化方法对水中有机物进行处理，通过电导率传感器测量水中的电导率，并根据测量结果计算出 TOC 值的解决方案，用户可通过触摸屏切换用户模式，还可以实时观测微流控芯片中水的温度、始末电导率、TOC 值。

本系统主要包含了两种用户模式：氧化模式与清洗模式。氧化模式开启后，程序将根据预设自动完成一次氧化流程，并通过无线串口将检测出来的数据自动远程传输给PC(personal computer)以 TXT 格式按时间命名保存。清洗模式开启后，将自动打开蠕动泵与电磁阀进行清洗。预设可根据需求在代码内修改，预设为泵、灯与模式的开启时间。

本检测系统的设计有助于提高纯水检测系统的精简度，优化设备的大小，同时可视化操作极大简便了用户的使用，具有广阔的应用前景和推广潜力。

关键词　　纯水　　TOC　　直接电导法　　可视化

Design of a Pure Water TOC Detection System Based on ARM

Xiao Ao

(College of electronic engineering (College of Artificial Intelligence), South China Agricultural University, Guangzhou 510642, China)

Abstract: With the continuous development of electronic information technology, sensor technology is becoming increasingly powerful. The improvement in sensor accuracy has led to growing interest in the biochemical industry, and the integration of electronic information with biochemistry has become a new research direction. In pure water TOC(Total Organic Carbon) detection systems, previous TOC detection methods suffered from bulky size and lack of portability. Therefore, attention has shifted towards direct conductivity methods. However, this method is relatively novel in China and is constrained by high equipment costs and a lack of secondary development interfaces and related documentation. Therefore, there is an urgent need to design a streamlined and cost-effective pure water detection system.

This paper presents a design idea for a pure water TOC detection system based on ARM(Advanced RISC Machine). The system is designed for laboratory application environments and uses the STM32F407ZGT6 development board with Cortex-M4 core for development. It is an advanced laboratory solution that processes organic matter in water using the ultraviolet oxidation method, measures the conductivity of water through a conductivity sensor, and calculates the TOC value based on the measurement results. Users can switch between user modes via the touch screen and can also observe in real-time the temperature, initial and final conductivity, and TOC value of water in the microfluidic chip.

The system mainly includes two user modes: oxidation mode and cleaning mode. In the oxidation mode, the program will automatically complete an oxidation process according to the preset and transmit the detected data to a PC(personal computer) via wireless serial port in TXT format named by time. In the cleaning mode, the peristaltic pump and solenoid valve will be automatically activated for cleaning. The preset can be modified within the code according to requirements, presetting the opening time of the pump, lamp, and mode.

The design of this detection system helps to streamline the pure water detection

Key words: Pure water TOC conductivity method visualization

目　　录

1 绪论

1.1 课题研究背景

超纯水也被称为“高度精制水”，是通过吸附、离子交换、膜过滤和紫外线氧化等各种工艺对天然水进行精炼而产生的(Jeongyun，2019)。随着生物医疗技术与食品技术的发展，超纯水的应用领域在逐渐扩展，根据《分析实验室用水规格和实验方法》(GB6682-2008)，实验室中使用的超纯水拥有极其严格的水质标准，分为三个等级：一级纯水、二级纯水和三级纯水（王思华，2022）。超纯水的质量在制药和半导体行业尤为重要,由于其独特的性能，它可用作制备注射液的溶剂或起始产品，也用于半导体制造中高精度元件生产中的表面清洁(Schäfer et al,2022)。三级纯水可以用在实验室里清洗实验仪器，供水给需求超纯水的仪器，其高纯度确保了实验结果的准确性和可重复性。二级纯水可以在医药制造行业参与制造药品，还能用于微生物培养、生物分析仪、pH测试等。一级纯水是实验室、医药、电子工业和化工等领域的重要资源，电子工业里需要使用极高纯度的水来清洗和制造电子产品，化工生产中需要使用超纯水用作反应介质，又或者高端应用场景，如高压液相色谱(High Performance Liquid Chromatography,HPLC)、液相色谱质谱联用(Liquid Chromatography-Mass Spectrometry,LC-MS)、电感耦合等离子体质谱(Inductively Coupled Plasma-Mass Spectrometry,ICP-MS)等痕量理化分析实验（马军等，2007）。

为了评估水质污染程度，传统的水质检测方法包括化学需氧量(COD)、生化需氧量(BOD)和水质五参数等（范浩明等，2023），但这些方法存在测量不准确的缺陷。总有机碳（TOC）作为一种更准确反映水中有机物含量的指标，逐渐受到关注。但近年来，随着工业和农业的快速发展，水体污染问题日益严重。化工产业的迅速发展导致了有毒有害物质的排放，进而加剧了水体中有机物的增加，严重影响了水质，不可避免的导致了纯水 TOC 检测任务的需求增长。因此，纯水检测系统的设计变得至关重要。

而电导率是电解质溶液的一个重要性质，电导率测量在环境监测、工业生产流程控制、电分析化学、临床医学、海洋、水文、轻工、冶金等领域中有广泛应用。电导率测量技术的重要性不仅体现在传统领域中，也在新兴领域中得到了广泛应用，比如海洋研究。近年来，电导率温度和深度测量技术在海洋研究和应用中发挥着重要作用，中国在“九五”、“863”计划的支持下取得了巨大进步。总之，研究电导率测量技术以及 TOC 等指标的测量方法对于水质评估、环境监测等领域具有重要意义。

1.2 国内外研究现状

国外关于 TOC 分析仪的研究较早，虽然我国在 1979 年有过燃烧-电导法 TOC 测定方法的研究，但此后我国有关 TOC 分析仪的研究一直停滞不前。而早在 2002 年我国的全国学术会议上，就已经确认了当时世界上的 TOC 分析仪的氧化技术以高温氧化为主，美国也拥有明文法规规定了加热的过硫酸氧化法、紫外线加过硫酸氧化法等化学试剂氧化方法的法定地位，紫外线氧化法也早在日本就得到了广泛应用（徐涛等，2002）。但我国对 TOC 分析仪的研究在那时才刚起步。在二氧化碳的探测技术方面，薄膜电导率探测虽然已经获得世界范围的认可，但是探测技术的主流仍然是非散射红外探测。

经过十几年的发展，无论国内还是国外，氧化技术都拥有了很大的变化。由于高温燃烧和化学燃烧导致分析仪体积庞大且复杂，如今世界先进的 TOC 分析仪的氧化技术已改为紫外线氧化法为主。我国也紧随世界的步伐，但是与世界平均水平相比较，我国 TOC 分析仪普遍存在系统繁琐、可移植性差等问题，技术水平落后也导致价格昂贵。

至于二氧化碳的探测技术，由于红外吸收法检测数据优良的精确性，世界主流依旧还是红外探测为主。在电导探测法上，我国学者在 2021 年初步进行了 TOC 电导法检测标准的研究，确立了温度、重复性引入等的不确定度标准（洪钏，2021），在 2022 年详细研究了基于紫外氧化法的超纯水分析仪的应用（王思华，2022），在 2023 年使用直接电导法的分析仪来研究锅炉给水中 TOC 含量的不确定度，确定了该系统测定过程中的各类不确定因素（梅玉东，2023）。因此，紫外氧化-直接电导率测定法分析仪系统的研究在我国是前沿领域，较为新颖。

1.3 课题研究的目的与意义

在未来，精简化和可移植化是主流趋势，将占据主导地位，各国也一直在投入研究 TOC 分析仪。本文研究的基于 ARM 的纯水 TOC 检测系统目的也是为了推进其精简性，使 TOC 分析仪更容易参与实验室的研究使用，减少人力物力的开支。

该系统的氧化方法采用国际主流紫外氧化法，紫外氧化法具备广泛验证、简化操作等优势，只需要配备使用者所需规格的紫外灯，并控制灯开启与关闭的时间即可实现纯水的氧化。检测方法采用小巧精密的电导率传感器，一个铂电导率电极即可测取电导率值。该系统基于 ARM 架构，易于移植，并支持根据需求更换不同精度的电导率电极。

本文的 TOC 检测系统基于嵌入式系统，实现了温度和电导率的自动探测，并能根据需求对电导率值进行温度补偿，提高 TOC 计算值的精确度。该系统不仅可以在触摸屏上显示数据，还可以通过电脑远程获取测量的相关参数，实现了系统的智能化和可视

化。

2　系统总体设计

2.1　STM32F407ZGT6 核心板介绍

本系统基于 ZG 核心版进行二次开发。该开发板搭载以 Cortex-M4 为内核的处理器 stm32f407zgt6，它具有充足的外设资源，STM32F407ZGT6 核心板的外设资源如图 1 所示：

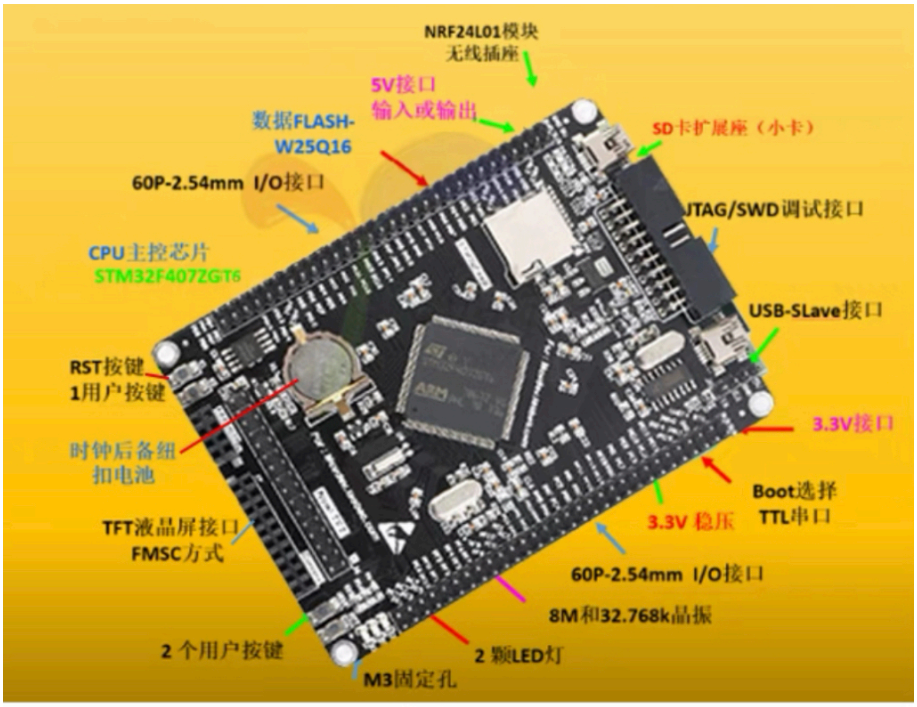


图 1 STM32F407ZGT6 核心板外设资源

该款开发板有以下特点：

（1）高性能：采用了 ARM Cortex-M4 内核，主频可达 168MHz，具备丰富的处理能力和高速运算能力。

（2）大容量存储：具备 1MB 的 Flash 存储器和 192KB 的 SRAM，可以存储大量的程序代码和数据。

（3）多种外设接口：支持多种外设接口，包括 USB、CAN、SPI、I2C、UART 等，可以与其他设备进行通信。

（4）多通道 DMA 控制器：内置多通道 DMA 控制器，可实现高效的数据传输和处理，

减轻主处理器的负担。

（5）丰富的时钟和定时器：提供多个时钟源和定时器，可满足不同应用的需求。

（6）低功耗模式：具备多种低功耗模式，可在不同场景下灵活选择，以实现低功耗运行和延长电池寿命。

（7）丰富的开发支持：意法半导体提供了完善的开发工具和软件库，可快速进行软件开发和系统调试。

2.2 微流控芯片介绍

图 2 所示为微流控芯片，微流控芯片是一种微型化的实验平台，用于在微米尺度上进行流体控制和生物/化学分析。它通常由微流道网络、控制元件和传感器组成，可以实现对微流体的精确控制和监测。而本系统通过微流控芯片，能极大降低紫外氧化纯水时间与清洗速率，提高检测效率，正是微流控芯片这种微小尺寸使得系统能够在微量样本下进行高效的实验操作。本系统也正是因为使用了微流控芯片，极大精简了系统的大小，让系统更易移植。

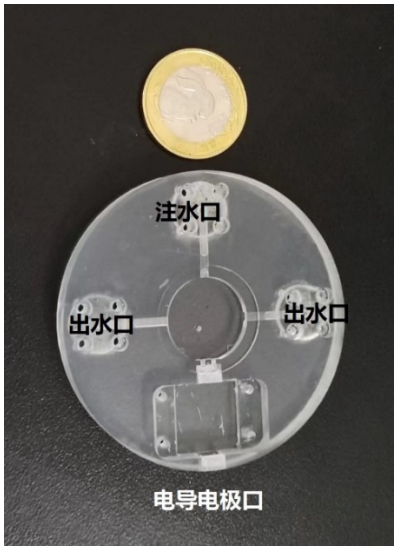


图 2 微流控芯片

2.3 纯水 TOC 检测系统框图

该纯水 TOC 检测是面向用户使用的系统，用户所有操作通过触摸屏实现。如图 3 就是纯水 TOC 检测系统框图。

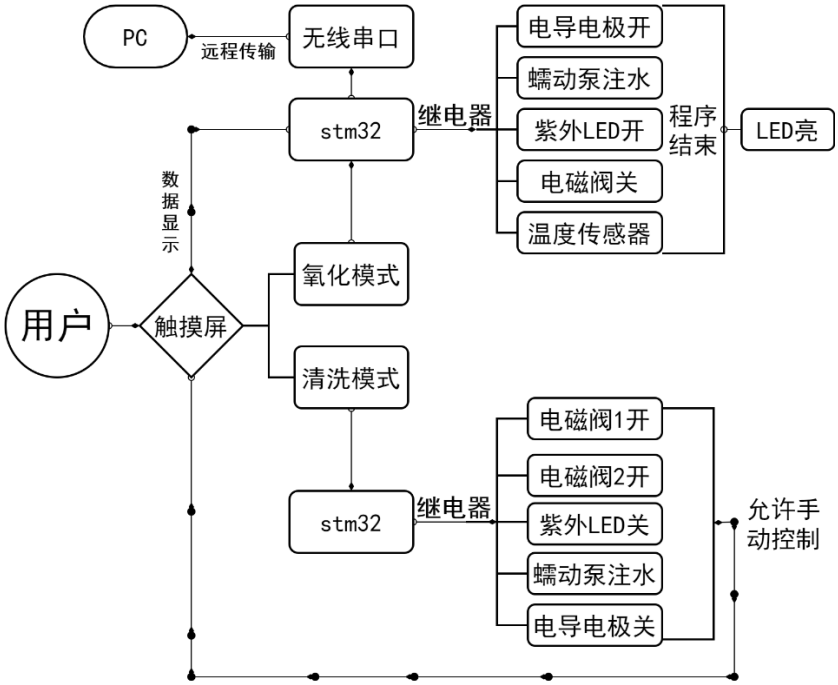


图 3 纯水 TOC 检测系统框图

用户通过触摸屏选择模式，系统则会根据预设的程序对设备进行自动控制。

如果切换到氧化模式，stm32 会先发出指令打开温度传感器与电导电极，同时蠕动泵注水，期间电磁阀保持阀门紧闭。纯水注入完毕后，温度与初始电导率值将返送给 stm32。stm32 再发出指令打开紫外 LED 灯进行纯水氧化，氧化完毕之后，将会再报送结束电导率值给 stm32，stm32 最后通过差值公式计算得出 TOC 值，并将数据通过无线串口无线远程报送给电脑，数据会以 txt 格式按年月日时分秒自动存储在电脑上，最后再打开 LED 白灯来告诉用户程序执行完毕。

如果切换到清洗模式，stm32 会先发出指令，通过预设指令自动打开蠕动泵与 2 个电磁阀，清洗预设时间后将自动关闭。用户不管是否在自动清洗过程中，都能手动对阀门与泵进行开关，这种方式将更加自由地满足用户的需求。

3 系统硬件设计

3.1 继电器与设备驱动板

3.1.1 驱动板简介

本系统自行设计了一块驱动板，以将各设备的驱动集成在一起，方便用户的使用，减少插线的需求。该驱动板只需通过排插插入开发板，芯片即可发出指令进行控制。芯片集成了 5V 电源输入与 12V 电源输入，5V 电源无需继电器，而 12V 电源过大，需要

通过 4 个继电器对相应 4 个 12V 设备进行供电，如图 4 为设备驱动板实物图与背板排插。

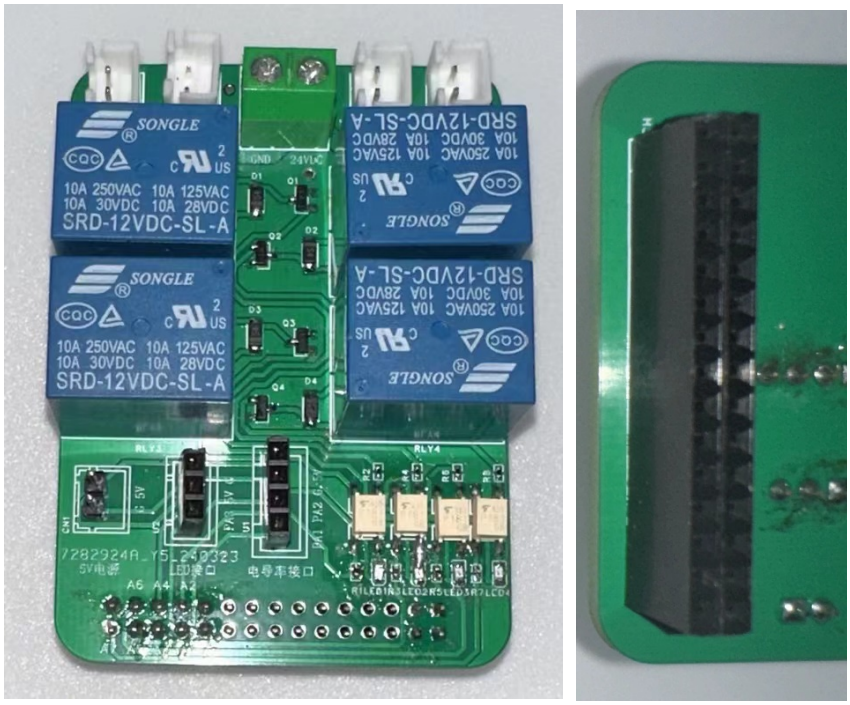


图 4 设备驱动板实物图与背板排插

3.1.2 驱动板电路设计

该驱动板有两种供电的方式：开发板 5V 直流供电，外部 220V 交流转 12V 直流电源适配器供电。5V 供电设备有 LED 模组与电导率模块。12V 供电有 2 个电磁阀，1 个紫外灯，1 个蠕动泵。如图 5 为 5VDC 与 5V 设备原理图。

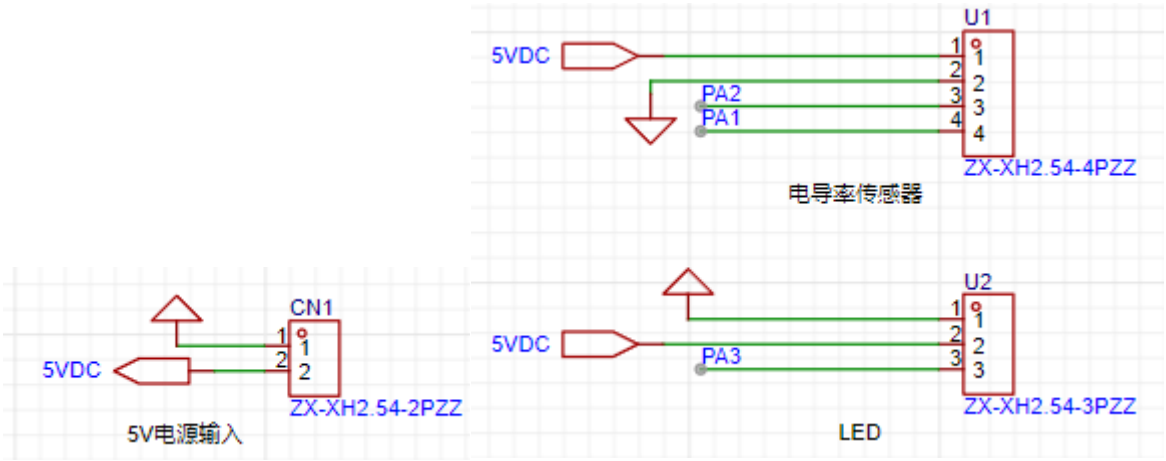


图 5 5VDC 与 5V 设备原理图

由原理图可见 5V 电源输入直连开发板 5V 引脚为 LED 模组与电导率模块供电，且 LED 的驱动信号连接 PA3 口。电导率模组由于其需要温度补偿，已集成 DS18B20 温度传感器。由原理图可知，stm32 通过 PA2 口对温度传感器进行设备驱动与数据的读取，通过 PA1 口对电导率电极进行设备驱动与数据的读取，其中 PA1 初始化为板载 ADC 通道，将电导电极测出来的模拟信号转化为数字信号显示输出在屏幕上。

如图 6 为 12V 电源输入原理图与电源适配器实物图。

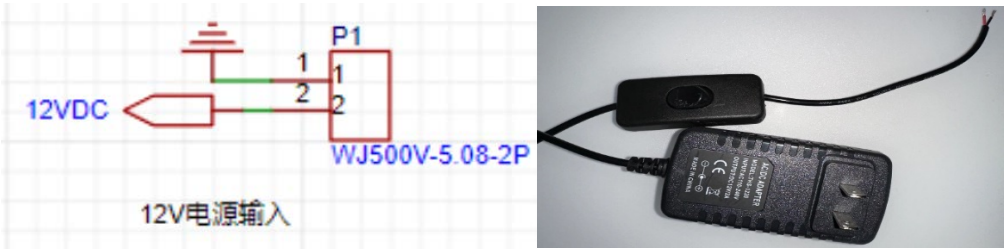
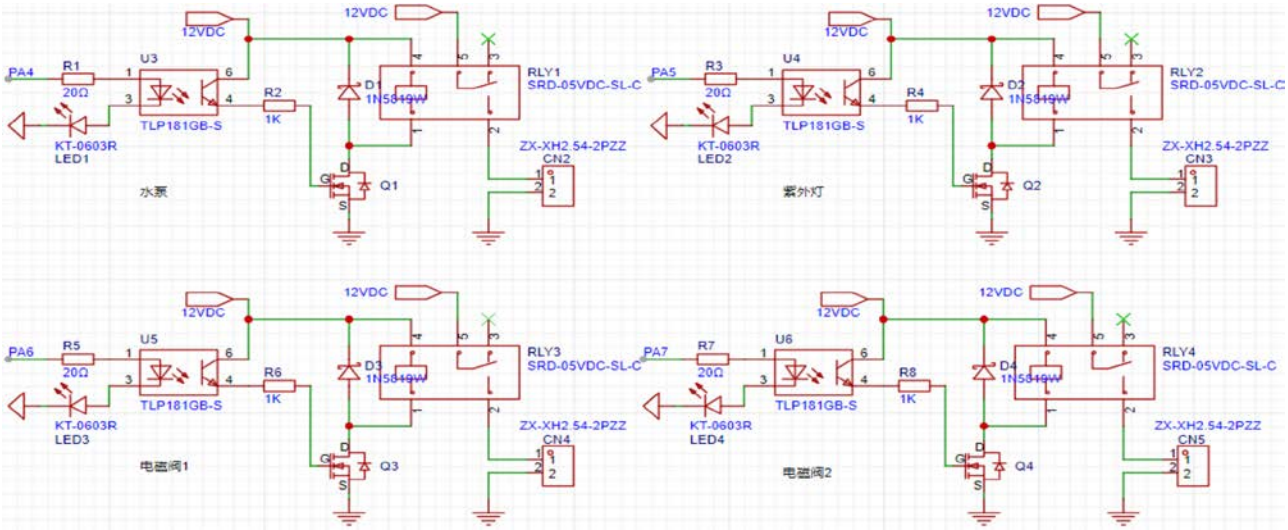


图 6 12V 电源输入原理图与电源适配器实物图

如图 7 为 12V 继电器控制原理图。



如图 7 12V 继电器控制原理图

由核心板板载资源和原理图可知，核心板无法供给 12V 电源，因此 12V 为外部电源适配器供给。由图可知，PA4 驱动水泵，PA5 驱动紫外灯，PA6、PA7 驱动电磁阀。继电器控制为常闭型，当 IO 口给予驱动信号时，LED 灯亮示意设备运行，且左侧 12VDC 电流经过继电器电磁铁使电磁铁导通，动触点 5 随即与下方静触点 2 闭合，右侧 12VDC

电流随即到达设备，设备开始运行。使用继电器不仅起开关作用，更重要的是提供了电气隔离，通过电磁原理，可以在控制信号与被控制电路之间提供隔离，从而使得不同电路之间的电流不会相互影响，避免了价格较为昂贵的开发板被击穿烧毁。

如图 8 为驱动板与开发板 IO 口连接的排插原理图。

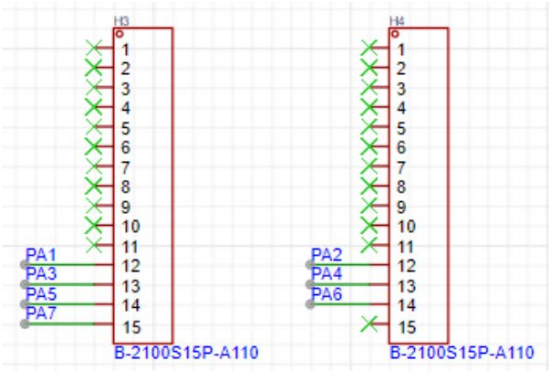


图 8 排插原理图

3.1.3 驱动板 PCB

如图 9 为驱动板正反面 PCB 图。

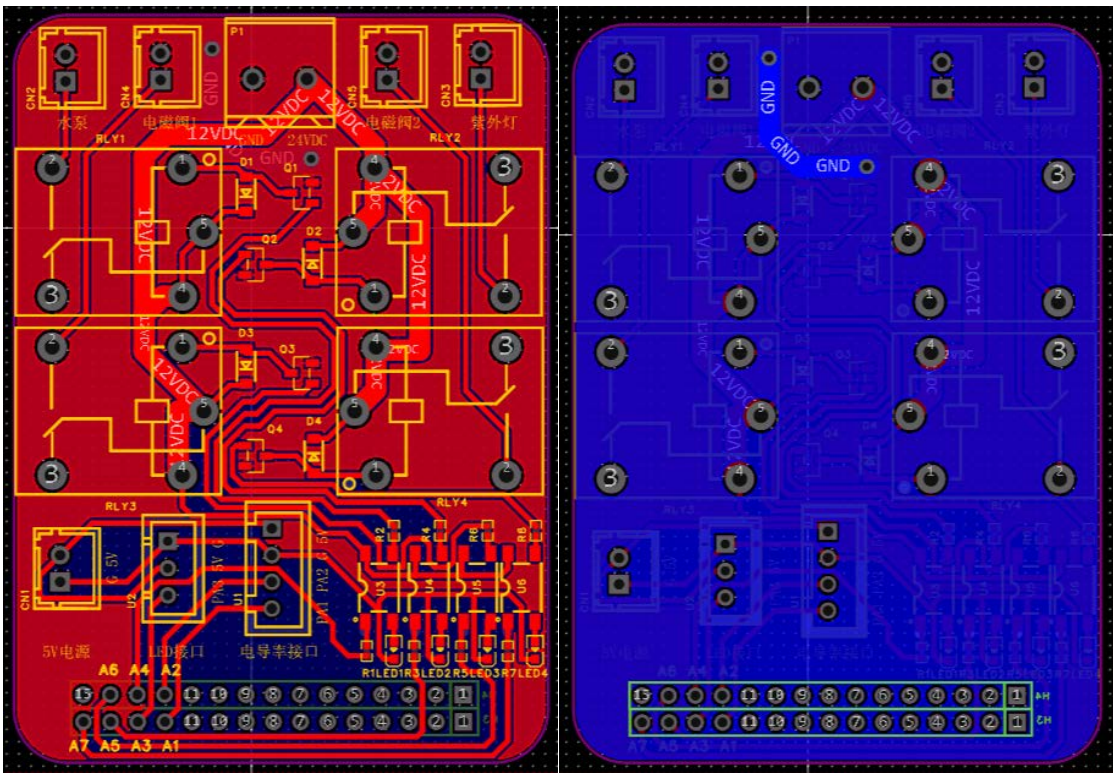


图 9 驱动板正反面 PCB 图

3.2 电导率传感器模块

3.2.1 模块简介

本系统的电导率检测采用了 WAAAX 维可思公司生产的集成检测模块。该模块通过 BNC 接头与电导率电极进行连接；集成温度传感器接口，方便进行温度补偿。如图 10 所示。

图 10 电导率传感器模块

表 1 模块引脚定义

序号	引脚定义	功能描述	备注
1	VCC	供电电压正极，5V	不可使用 3.3V
2	GND	供电电压负极	
3	T1	温度传感器数据输出口	
4	A0	模拟量输出口	输出电压范围 0~3.4V

如表所示，模块同时可获取电导率与温度两个测量值，但电导率值输出为模拟量，需经过开发板内置集成的 ADC 模块进行模数量转换。

3.2.2 铂电导电极与 DS18B20 温度传感器

（1）电导池又称电导电极，由两片固定在玻璃支架上的铂片组成。其距离与面积之比 L/A 称为电导池（电极）常数(cell constant)。每一支电导率电极有固定的电导池常数值。如图 11 所示为电导电极，电导电极较为昂贵，从成本考虑出发，本系统所选用的电导电极，目的仅为实现正常的测量功能。电导池常数为 K=0.991，支持测量范围：

0~20mS/cm，推荐测量范围：1~15mS/cm，测量温度：0-40℃。如需更换更高精度的电导电极，需确保为 BNC 接口。该电导电极使用一段时间后精度会改变，需要采用 2 点校准方法，使用 1413uS/cm 和 12.88mS/cm 的电导率标准液（杜思舟，2021）。

图 11 铂电导电极

（2）DS18B20 是数字温度传感器，由 Dallas Semiconductor（现为 Maxim Integrated）公司制造。它使用单总线接口通信，并可直接连接到微控制器。DS18B20 具有高精度和数字输出，可测量温度并输出至微控制器。其特点包括数字输出、高精度、广泛温度范围（-55°C 至+125°C）以及唯一的 64 位串行代码。因此 DS18B20 被广泛应用于温度监测领域，如温度控制系统、环境监测系统和气象站。考虑到其符合实验室高精度的标准，且恰好适配电导率传感器模块，不需要额外的电平处理，选用 DS18B20 能极大缩小开发效率。如图 12 为 DS18B20 数字温度传感器

图 12 DS18B20

3.2.3 TOC 计算值

本系统使用的是直接电导率法：通过测试经过氧化反应的样品的总碳含量和未经过

氧化反应的样品总无机碳的含量差值来测定总有机碳含量。以本实验为例，温度和电导率传感器采集紫外灯开启过程前后的电导率值和温度值，即紫外灯开启时开始采集一次电导率值，紫外灯关闭时采集一次电导率值，将二者的电导率差值带入假定公式:y=119.13x+59.164（y 为 TOC 值，x 为电导率差值，TOC 值单位为 μg/L），即可算出 TOC 含量。

3.3 其他外设简介

（1）紫外氧化法具有能耗低、完全无污染、反应可控性高、反应彻底迅速等良好特性（王思华，2022），考虑到成本与氧化能力，本系统采用 12V、波长 270-280nm、辐射功率 15-20mW 且发光角度为 120 度的紫外灯进行有机物的氧化，由于紫外灯工作时会产生大量热能，为避免紫外灯过热烧毁，该紫外灯还外接线并联了 12V 小风扇，将与紫外灯一同启动。图示 13 为紫外灯实物图。

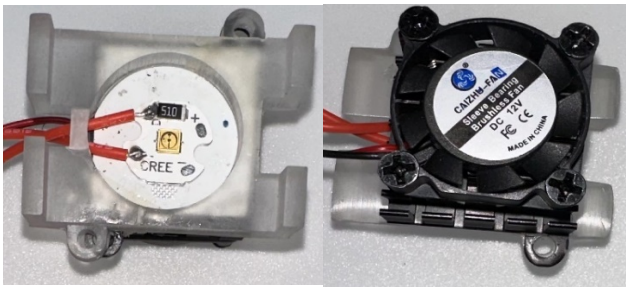


图 13 紫外灯

（2）该系统采用了 2 个 12V 常闭型电磁阀，预设的冲洗模式会自动打开 2 个电磁阀，以防止冲洗时，会有一侧死角残留测后水。但系统预留了手动控制模式，用户可根据需求选择性打开全部阀门又或者其中一个阀门。图示 14 为常闭型 12V 电磁阀实物图。

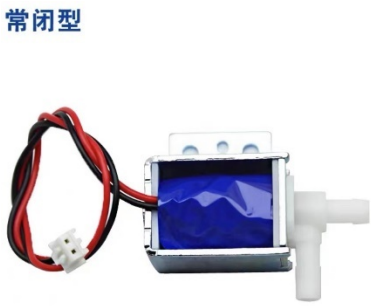


图 14 电磁阀

（3）本系统采用了蠕动泵，蠕动泵是利用蠕动原理进行液体输送的一种特殊泵。它由柔软管和滚轮或滚筒组成，通过挤压柔软管推动液体。蠕动泵优点包括不需密封、适用于高粘度液体和多种应用场景，但其缺点也很明显，12V 的驱动意味着能耗高，且不断蠕动导致了易损坏需要定期更换部件。图示 15 为 12V 蠕动泵实物图。

图 15 蠕动泵

（4）为了较为醒目的提示系统氧化模式的运行结束，本系统使用了 LED 大白灯模组，该模组由板载 5V 供电，通过一条信号线连接至 stm32 来接收信号。图示 16 为 LED 模组。

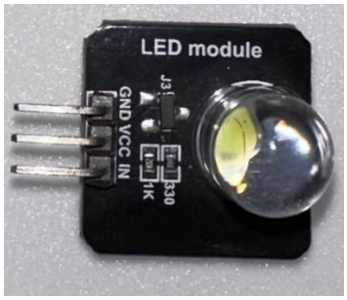


图 16 LED 模组

（5）为实现远程传输数据且能对数据自动保存，该系统采用了 JDY-40 无线串口，其原理运用了 2.4GHz 的无线透传技术，通常成对使用，一收一发，传输距离为视距 120 米，只需要了解串口知识就可以对产品进行应用，实现了系统免开发易移植的优点。图 17 为 JDY-40 无线串口。

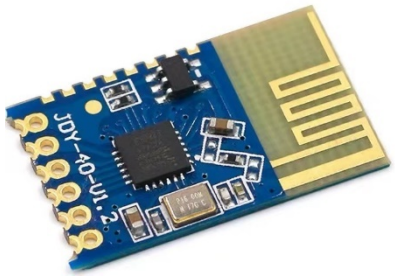


图 17 JDY-40

4　系统软件设计

4.1　系统程序运行流程

图示 18 为程序总体流程图

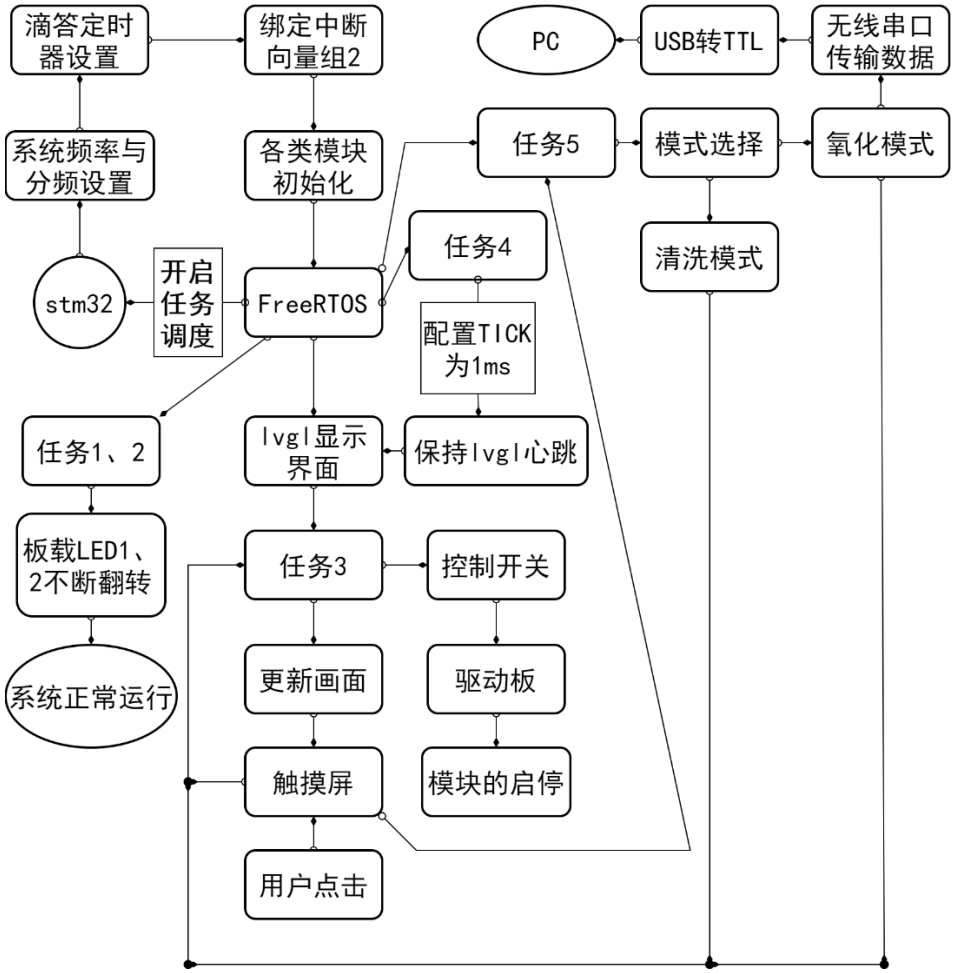


图 18 程序流程图

由图可知程序先对系统的各个设定进行了初始化设置，随后便进入 FreeRTOS 操作

系统。程序开启任务调度后，整个系统即以多任务并行执行的方式运作。

4.2 可视化下位机系统

4.2.1 FreeRTOS 操作系统

以往裸机开发形式，总将所有需要执行的函数写在 int main 中 while(1)的大循环里，串行依次完成所写的代码任务，而有中断发生的时候才能跳转出去处理，因此被称为单任务系统。相较于单任务系统的裸机，移植了 FreeRTOS 操作系统的开发板将可以满足多任务处理的功能需求，极大提高了系统的执行速率。开发者在使用 FreeRTOS 操作系统时，可将代码分为几部分任务，随后在 int main 函数中不需要写大循环，只需开启任务调度，便可对多个任务进行处理。较于裸机，它的并发性更好，能避免延时等情况 CPU 资源的闲置浪费；其次本系统中 FreeRTOS 采取所有任务的优先级都是相同的策略，拥有更好的实时性，只需设定时间节拍就可实现任务执行的自动跳转。最后它将任务模块化，使其具有高内聚、低耦合的特点。

4.2.2 LVGL 显示界面

LVGL(Light and Versatile Graphics Library)是一种开源的嵌入式图形库，用于创建图形用户界面(GUI)。在 FreeRTOS 操作系统上使用 LVGL 有以下几个优点：

（1）轻量级和高效性：LVGL 被设计成轻量级的图形库，它占用的内存和存储空间相对较小。这意味着与 FreeRTOS 相辅相成，能在嵌入式系统中保持高效稳定的运行。

（2）丰富的图形元素和组件：LVGL 提供了丰富的图形元素和组件，且这些元素的设计都经过了优化，使得它们在 FreeRTOS 操作系统上运行时能够保持高效性。

（3）交互性和动画效果：LVGL 支持丰富的交互性和动画效果，例如触摸屏输入、滑动效果等。这些功能恰好满足了本系统所需的可视化操作功能。

（4）易于定制和扩展： LVGL 提供了丰富的配置选项和定制功能，可根据需求进行配置以适应不同场景和硬件平台。

综上所述，LVGL 在 FreeRTOS 操作系统上具有轻量级、高效性、丰富的图形元素和组件、交互性和动画效果以及易于定制和扩展等优点，适合用于本检测系统中的图形界面开发。

4.2.3 TFT-LCD

本系统采用3.2寸16位并口/SPI串口LCDTFT液晶带电阻触摸屏作为可视化界面，TFTLCD 是一种液晶显示屏，它的每一个像素上都设置有一个薄膜晶体管(TFT)，因此得名。该屏支持 16 位真彩显示（65536 色），并内置 ILI9341 驱动芯片，该芯片目前是

我国市面上主流的 TFTLCD 驱动 IC。此外，TFTLCD 还自带电阻式触摸屏，当用户触摸屏幕时，触摸点附近的导电膜会发生电阻变化，控制电路据此可计算出触摸点的位置坐标。该屏幕还采用了 FMSC 新总线接口，可以直接插入本系统所使用的核心开发板上，易于程序的开发和移植。如图 19 为 TFT-LCD 触摸屏。

图 19 TFT-LCD

4.3 上位机数据存储

4.3.1 Python 代码编写与程序生成

无线串口通常成对来使用，其中一个用于发送数据，另一个用于接收数据。因此 PC 端也需要连接一个无线串口，本系统通过 USB 转 TTL 连接无线串口，再将 USB 插入 PC，即可实现串口与 PC 的连接，如图 20 所示。但 PC 仍旧需要对所插入的 USB 进行识别，才能正确的进行串口的数据接收，如图 21，因此本系统将数据传输波特率设为 9600 并对端口名称进行代码检测，检测成功即可进行端口连接配对。

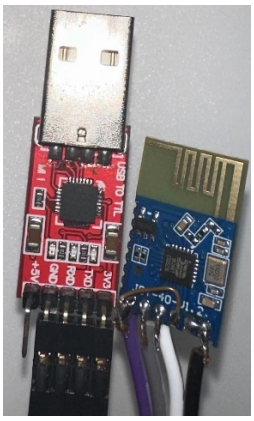


图 20 USB-TTL-无线串口　　图 21 USB to UART COM

为实现在 PC 端无线接收数据，本系统编写程序来判断所接收到的数据，判断正确则按规格对数据进行存储。在氧化模式氧化前，设备端无线串口将会发送初始电导值，因此只需判断接收数据中是否包含“start”字符串即可接收氧化前数据，同理在氧化后，只需判断接收数据中是否包含“stop” 字符串，即可接收氧化后数据。

最后，本上位机系统编写代码，在传输结束后，将自动生成 TXT 格式的文件，并将数据存储在其中。文件名将按照此次测量结束时的年、月、日、时、分、秒来命名。如图 22 为已制作好的 exe 运行程序。

图 22 远程数据接收程序

如图 23 为自动生成的的 txt 格式文件示例。

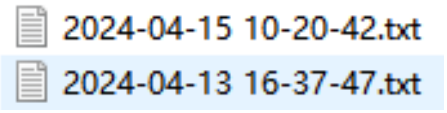


图 23 自动生成的 txt 文件

4.3.2 远程传输流程

以下图 24 为远程传输的详细流程图。

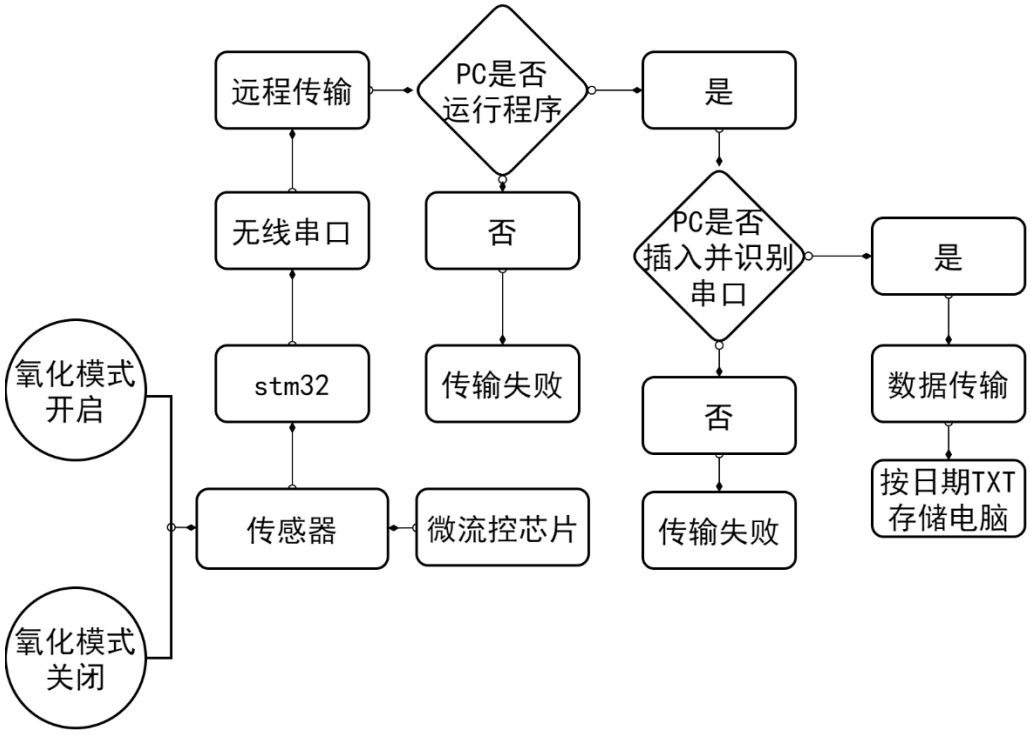


图 24 远程传输流程图

从此流程图可知，每次氧化模式的执行，stm32 都会通过无线串口传送 2 次数据。由于 TOC 计算方法为差值计算法，第一次传送数据在氧化模式前，传送数据为未经氧化的初始纯水电导率值。在氧化结束后，stm32 将会传送第二次数据，但不同的是，此次数据的传送，附带已经通过预设公式计算好的 TOC 值，因此，末次传输会传输结束电导率值与 TOC 计算值这两个数值。

而在远程数据传输的过程中，导致传输失败的主要变量都出自于电脑端，使用者需要在使用前提前将串口与电脑正确连接。但由于不同使用者的 PC 不同且不同使用者所使用的 USB 转 TTL 不同，在 USB 端口的识别上有概率出现识别不出的情况，当该情况发生的时候，使用者需要查看设备管理器中 USB 端口的名称，并在 python 文件中，将用于检测判断的字符串改为该名称中较为特别的关键词，最后生成新的 python 执行程序。生成执行程序的方法为，调出 win+R，输入 exe 生成命令：pyinstaller .\testuart.py -F，回车即可自动生成。

5　结果分析

5.1　检测系统实物与模块正常运行图

实物如图 25 所示。

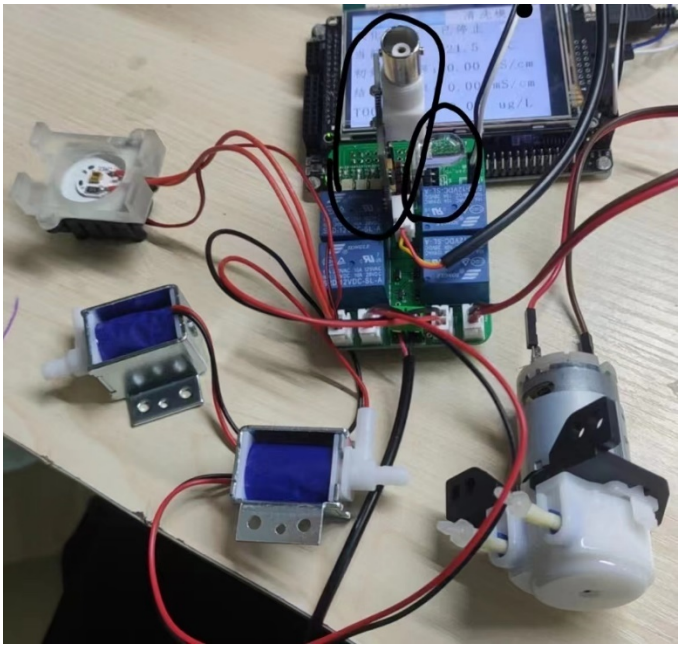


图 25　系统实物图

经系统全部安装完后调试，所有设备正常运行，所有功能正常实现，PCB 板无烧毁。运行 4 个小时，主控芯片温度维持在 28 度。经测试结果来看，系统稳定，符合预期。

如图 26 所示 D1、D2 两个 LED 灯不停翻转，代表系统正常运行。

图 26　D1、D2 翻转图

开启四个 12V 的设备的开关，各继电器指示灯均正常显示，且设备正常工作，如图 27 则为各设备工作图。可看到 4 个红色 LED 点亮，紫外灯点亮工作。

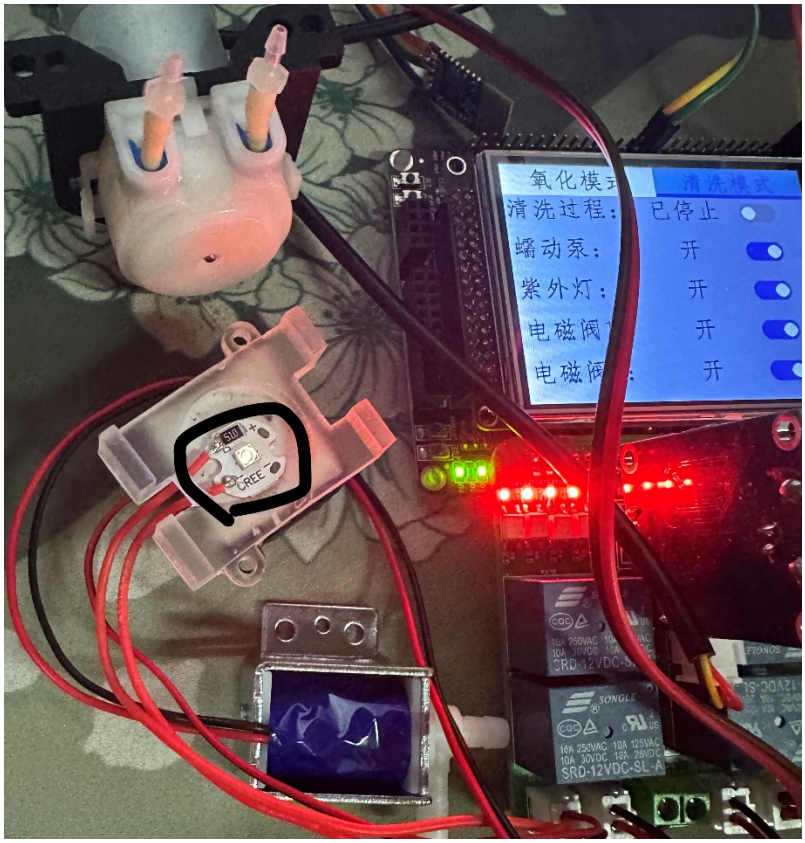


图 27 12V 设备正常运行

在氧化模式结束后，系统会自动点亮 LED 模组亮出闪耀白光提示模式运行完毕。如图 28 所示。

图 28 LED 模组正常工作

5.2 设备端程序运行 GUI 界面

如图 26 为氧化模式的 GUI 界面。其含括一个执行自动氧化过程的按钮，并能实时显示所需测量纯水的当前温度值。并会将始末测量电导率值与计算出的 TOC 值显示在屏幕上，以供实验人员查看数据。

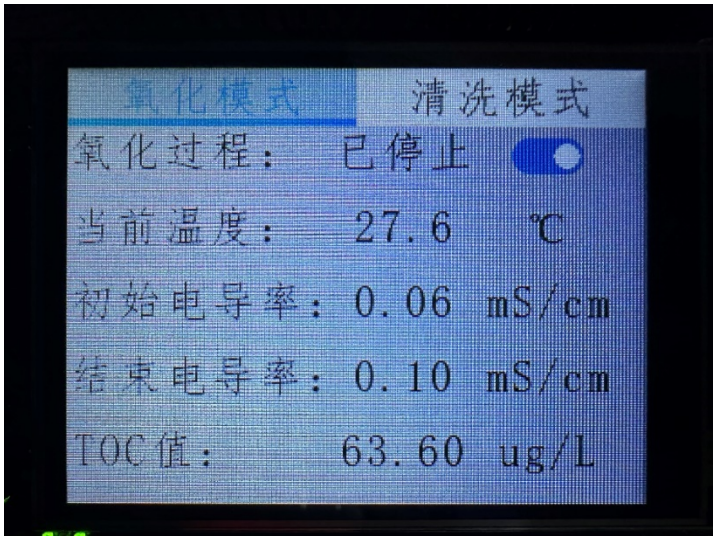


图 29 氧化模式 GUI

如图 27 为清洗模式的 GUI 界面。其拥有 5 个按钮，其中清洗过程按钮打开后，将会一直打开预设的蠕动泵与两个电磁阀，不会关停，直至关闭清洗过程的按钮。但除此之外，使用者还可以直接手动关闭或打开对应设备，以满足更多的使用需求与使用环境。

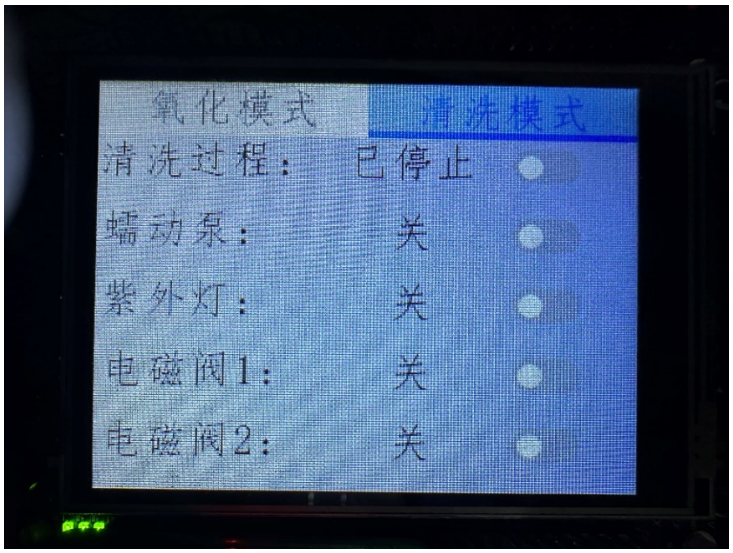


图 30 清洗模式 GUI

5.3 远程传输运行界面

如图 28 即为 exe 程序运行且串口连接正常时，数据传输成功的情况。可见通过对串口名称关键词的提取，电脑成功定位到了相应的 COM8，并接收到了无线传输的数据。数据和预设格式一致，分两次传输，开始为单独的电导率值（即图中 EC 值），结束为电导率值与 TOC 值。

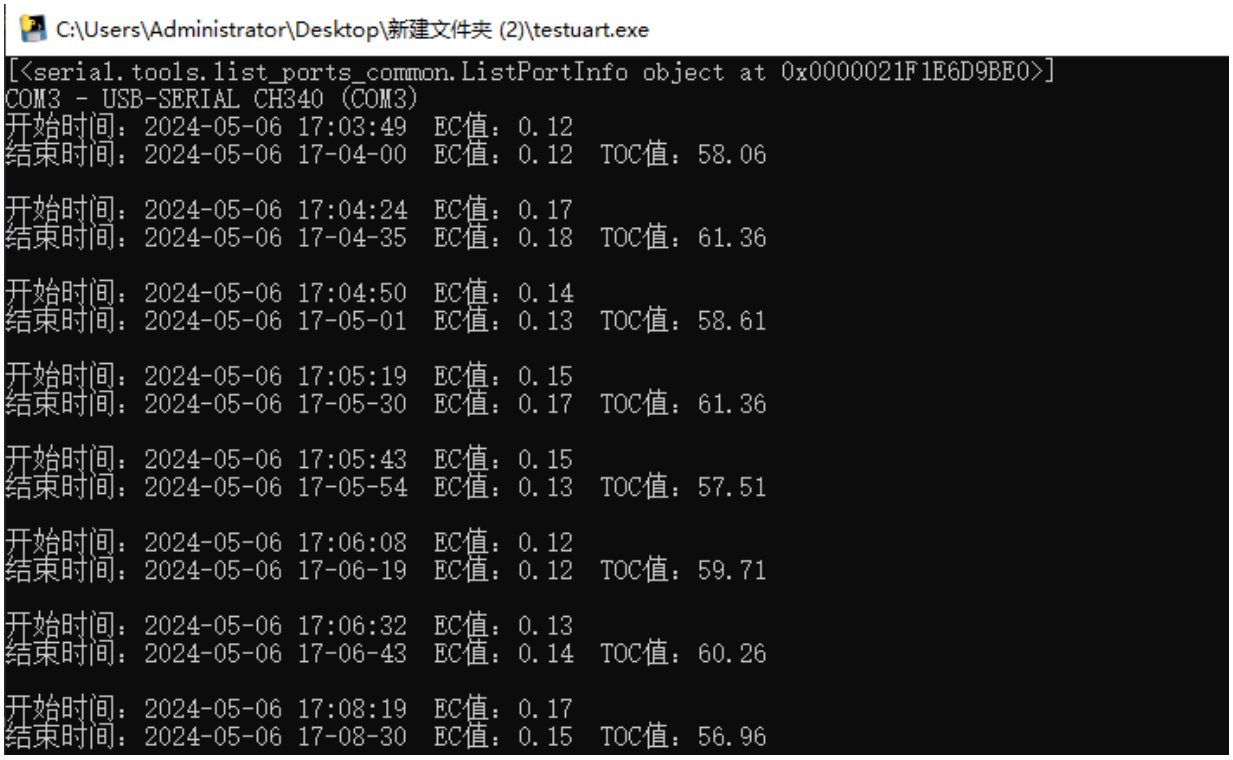


图 31 程序成功运行的界面

如图 29 为自动保存的 txt 文件内容图，打开后，可以验证数据已经完整且成功保存在了 txt 文件中，证明了该远程传输方法的可行性。此远程自动保存文件的方法极大缩减了实验者的工作量，实验者不需要匆忙记录数据，CPU 将自行执行这些操作并保障实验数据的准确性。且该远程存储以精确时间的形式的存储，减少了实验工作者筛查数据时所耗费的时间，相较于传统使用 SD/FT 卡对数据进行存储的方法，还省去了调取 SD 卡这一繁琐的步骤，进一步提高了该系统的精简性。

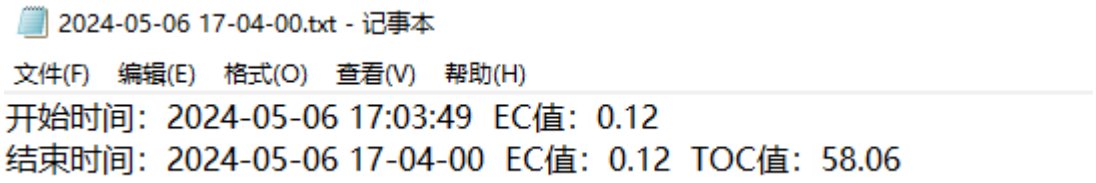


图 32 txt 内容图

6 总结与展望

本次检测系统在机械精简度上取得了较好的成果，已完美完成了期望的要求，所采用的设备大多为模块化、小而巧的设备，且为了提高设备的集成度，还制备了一块 PCB 板。该板将 5V、12V 电源、继电器与各种设备接连在了一起，此 PCB 板大小仅为开发板的一半。相较于以往的紫外氧化 TOC 分析仪，其大小已充分展现了 ARM 架构的精简与优越性。

本系统也实现了可视化触摸屏操作，各系统模式都能通过触摸屏进行触控执行，该系统最初选用 f103 系列芯片进行操作，但移植 FreeRTOS 操作系统与 LVGL 显示界面后，系统界面卡顿严重无法运行，更换 f407 芯片组后，虽然系统仍有些许延迟，但屏显与触控功能的执行已顺利展现，且更换芯片依旧能实现系统，也满足了系统的可移植性。

但该系统还是有些许不足，由于高精度电导电极以及高辐射功率的紫外灯成本过高，无法对更高规格的纯水进行进一步的研究。且电阻式触摸屏依旧拥有不小的问题，其触摸范围不能太大，只能用指尖轻点才能对按钮进行准确操作，给操作带来较为糟糕的体验。至于设备虽然精细程度已经很高了，但仍有明显的改进空间，引人注目的即是错综复杂的线路，希望在今后，我不仅是将驱动板集成起来，还能将所有模块、外设都集成起来，以实现无线化的真正精简性。

参 考 文 献

王思华. 基于紫外氧化法的超纯水 TOC 含量检测仪的研究及应用[D]. 青岛科技大学, 2023.

马军, 刘红斌, 龚承元等. EDI 高纯水系统及其在生物医学领域的应用研究[C]//天津市生物医学工程学会. 天津市生物医学工程学会 2007 年学术年会论文摘要集. 《生物医学工程与临床》, 2007:1.

范浩明, 倪宝培. 一种全自动红外 TOC 分析仪的研制方法[J]. 计量与测试技术, 2023, 50(06): 54-56, 59.

徐涛, 高玉成, 叶振忠等. 水质 TOC 分析仪器的现状及其检测技术的新进展[J]. 仪器仪表学报, 2002, (S3): 224-227.

洪钏. 电导法总有机碳分析仪校准方法及其不确定度评定[J]. 煤炭与化工, 2021, 44(10): 144-145, 153.

梅玉东. 直接电导法测定锅炉给水中 TOC_i 含量的不确定度评定 [J]. 广东化工, 2023, 50(08): 184-186.

杜思舟. 基于 Android 的水库水质监测系统设计[D]. 内蒙古科技大学, 2021.

Schäfer, S.,H., Katharina,et al. A new setup for the measurement of total organic carbon in ultrapure water systems[J]. Sensors 2022, 22(5), 2004.

Jeongyun Choi, Jinwook Chung. Evaluation of urea removal by persulfate with UV irradiation in an ultrapure water production system [J]. Water Research 2019, 411-416, 158.

附录A

```
#include "stm32f4xx.h"
#include "FreeRTOS.h"
#include "task.h"
#include "Driver.h"
#include "ili9341.h"
#include "lvgl.h"
#include "lv_port_disp_template.h"
#include "lv_port_indev_template.h"
#include "gui_guider.h"
#include "events_init.h"
#include "stdio.h"
#include "string.h"
#include "stdlib.h"
#include "SDcard.h"
#include "time.h"
#include "light.h"
#include "motor.h"
#include "svo.h"
#include "svt.h"
#include "led.h"
#include "toc_sensor.h"
#include "uart.h"
#include "console_logic.h"
//#include "prohead.h"
//#include "diskio.h"
//#include "ff.h"
#define MOTOR_DELAY_S    10                  //蠕动泵开启时间单位s
#define LIGHT_DELAY_S    20    //3*60        //紫外灯开启时间单位s
```

```
#define CLEAR_DELAY_S    10                    //清洗模式时间单位s
//FATFS    FileSys; //file system object
//FIL        File;//File object
//FRESULT    FileResult; //File function return code (FRESULT)
//UINT NbCount ; //file number count
//__align(4) u8 TxBuffer[512] = "你好测试";
//void DIS_Delayms(uint16_t Num)
//{
//          uint16_t Timer;
//    while(Num--)
//    {
//          Timer = 25000;
//          while(Timer--);
//    }
//}
lv_ui guider_ui;
unsigned char touch_state_val = 0;
char display_str[20] = {0};
uint8_t sdcard_state = 0;
void vTask1(void *pvParameters)      //翻转LED0
{
   GPIO_InitTypeDef GPIO_InitStructure;
   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOF, ENABLE);
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;//LED1对应IO口
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;//普通输出模式
   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;//推挽输出
   GPIO_InitStructure.GPIO_Speed = GPIO_Low_Speed;//100MHz
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_DOWN;//上拉
   GPIO_Init(GPIOF, &GPIO_InitStructure);
   while(1)
```

```
    {
      GPIO_SetBits(GPIOF,GPIO_Pin_9);
      vTaskDelay(300);
      GPIO_ResetBits(GPIOF,GPIO_Pin_9);
      vTaskDelay(300);
    }
}
void vTask2(void *pvParameters)      //翻转LED1
{
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;//LED1对应IO口
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;//普通输出模式
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;//推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Low_Speed;//100MHz
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_DOWN;//上拉
    GPIO_Init(GPIOF, &GPIO_InitStructure);
    while(1)
    {
      GPIO_SetBits(GPIOF,GPIO_Pin_10);
      vTaskDelay(500);
      GPIO_ResetBits(GPIOF,GPIO_Pin_10);
      vTaskDelay(500);
    }
}
void vTask3(void *pvParameters)    //LVGL线程，ADC获取
{
    LCD_ili9341Init();                                    //屏幕初始化
    lv_init();
    lv_port_disp_init();
    lv_port_indev_init();
```

```
        setup_ui(&guider_ui);
        events_init(&guider_ui);
        printf(" test hello world\r\n");
        while(1)
        {
          Touch_GetSite(XDown, YDown);
          //My_RTC_Get();
          DS18B20_Get_Temp();
        console_logic_run(motor_state,light_state,svo_state,svt_state,oxidize_state,clear_state
);
          memset(display_str,0,sizeof(display_str)/sizeof(display_str[0]));
           sprintf(display_str,"%.1f",temp_value);
           lv_obj_set_style_text_font(guider_ui.screen_temp_lable, &lv_font_simfang_24, 0);
           lv_label_set_text(guider_ui.screen_temp_lable, display_str);
          memset(display_str,0,sizeof(display_str)/sizeof(display_str[0]));
           sprintf(display_str,"%.2f",EC_Start_value);
           lv_obj_set_style_text_font(guider_ui.screen_start_conductivity_lable,
&lv_font_simfang_24, 0);
           lv_label_set_text(guider_ui.screen_start_conductivity_lable, display_str);
          memset(display_str,0,sizeof(display_str)/sizeof(display_str[0]));
           sprintf(display_str,"%.2f",EC_Stop_value);
           lv_obj_set_style_text_font(guider_ui.screen_end_conductivity_lable,
&lv_font_simfang_24, 0);
           lv_label_set_text(guider_ui.screen_end_conductivity_lable, display_str);
          memset(display_str,0,sizeof(display_str)/sizeof(display_str[0]));
           sprintf(display_str,"%.2f",toc_value);
           lv_obj_set_style_text_font(guider_ui.screen_toc_lable, &lv_font_simfang_24, 0);
           lv_label_set_text(guider_ui.screen_toc_lable, display_str);
          if(oxidize_state == 1)    //oxidize mode
          {
```

```
            lv_obj_set_style_text_font(guider_ui.screen_oxidize_state_lable,
&lv_font_simfang_24, 0);
            lv_label_set_text(guider_ui.screen_oxidize_state_lable, "运行中");
        }
        if(oxidize_state == 0)
        {
            lv_obj_set_style_text_font(guider_ui.screen_oxidize_state_lable,
&lv_font_simfang_24, 0);
            lv_label_set_text(guider_ui.screen_oxidize_state_lable, "已停止");
        }
        if(clear_state == 1)    //clear mode
        {
            lv_obj_set_style_text_font(guider_ui.screen_clear_state_lable,
&lv_font_simfang_24, 0);
            lv_label_set_text(guider_ui.screen_clear_state_lable, "运行中");
        }
        if(clear_state == 0)
        {
            lv_obj_set_style_text_font(guider_ui.screen_clear_state_lable,
&lv_font_simfang_24, 0);
            lv_label_set_text(guider_ui.screen_clear_state_lable, "已停止");
        }
        if(motor_state == 1)    //motor
        {
            lv_obj_set_style_text_font(guider_ui.screen_motor_state_lable,
&lv_font_simfang_24, 0);
            lv_label_set_text(guider_ui.screen_motor_state_lable, "开");
            lv_obj_add_state(guider_ui.screen_motor_sw, LV_STATE_CHECKED);
        }
        if(motor_state == 0)
```

```
        {
            lv_obj_set_style_text_font(guider_ui.screen_motor_state_lable,
&lv_font_simfang_24, 0);
            lv_label_set_text(guider_ui.screen_motor_state_lable, "关");
            lv_obj_clear_state(guider_ui.screen_motor_sw, LV_STATE_CHECKED);
        }
        if(light_state == 1)    //light
        {
            lv_obj_set_style_text_font(guider_ui.screen_light_state_lable,
&lv_font_simfang_24, 0);
            lv_label_set_text(guider_ui.screen_light_state_lable, "开");
            lv_obj_add_state(guider_ui.screen_light_sw, LV_STATE_CHECKED);
        }
        if(light_state == 0)
        {
            lv_obj_set_style_text_font(guider_ui.screen_light_state_lable,
&lv_font_simfang_24, 0);
            lv_label_set_text(guider_ui.screen_light_state_lable, "关");
            lv_obj_clear_state(guider_ui.screen_light_sw, LV_STATE_CHECKED);
        }

        if(svo_state == 1)    //svo
        {
            lv_obj_set_style_text_font(guider_ui.screen_svo_state_lable,
&lv_font_simfang_24, 0);
            lv_label_set_text(guider_ui.screen_svo_state_lable, "开");
            lv_obj_add_state(guider_ui.screen_svo_sw, LV_STATE_CHECKED);
        }
        if(svo_state == 0)
        {
```

```
                lv_obj_set_style_text_font(guider_ui.screen_svo_state_lable,
&lv_font_simfang_24, 0);
                lv_label_set_text(guider_ui.screen_svo_state_lable, "关");
                lv_obj_clear_state(guider_ui.screen_svo_sw, LV_STATE_CHECKED);
            }
            if(svt_state == 1)    //svt
            {
                lv_obj_set_style_text_font(guider_ui.screen_svt_state_lable,
&lv_font_simfang_24, 0);
                lv_label_set_text(guider_ui.screen_svt_state_lable, "开");
                lv_obj_add_state(guider_ui.screen_svt_sw, LV_STATE_CHECKED);
            }
            if(svt_state == 0)
            {
                lv_obj_set_style_text_font(guider_ui.screen_svt_state_lable,
&lv_font_simfang_24, 0);
                lv_label_set_text(guider_ui.screen_svt_state_lable, "关");
                lv_obj_clear_state(guider_ui.screen_svt_sw, LV_STATE_CHECKED);
            }
            lv_task_handler();
            vTaskDelay(1);
          }
      }
      void vTask4(void *pvParameters)    //保持LVGL心跳
      {
         while(1)
         {
          lv_tick_inc(1);
          vTaskDelay(1);
         }
```

```
}
void vTask5(void *pvParameters)
{
   uint16_t motor_delay = 0;
   uint16_t light_delay = 0;
   uint16_t clear_delay = 0;
   while(1)
   {
    if(clear_state == 1)
    {
          clear_state = 2;
          motor_state = 1;
          svo_state      = 1;
          svt_state      = 1;
          led_state      = 0;
    }
    if(clear_state == 0)//3
    {
          motor_state = 0;
          svo_state      = 0;
          svt_state      = 0;
    }
    if(oxidize_state == 1)
    {
          motor_delay++;
         light_delay++;
          toc_value           = 0;
          if(motor_delay <= MOTOR_DELAY_S)
          {
               motor_state = 1;
```

```
                if(motor_delay == MOTOR_DELAY_S)
                {
                    motor_state = 0;
                    EC_Start_value = 0;
                 EC_Stop_value    = 0;
                    EC_Start_Get();
                    printf(" start ec %.2f",EC_Start_value);
                }
            }
            if(motor_delay > MOTOR_DELAY_S)
            {
            }
            if((MOTOR_DELAY_S <light_delay)    &&    (light_delay<
LIGHT_DELAY_S))
            {
                light_state = 1;
            }
            if(light_delay > LIGHT_DELAY_S)
            {
                light_state = 0;
                motor_delay = 0;
              light_delay = 0;
                oxidize_state = 0;
                led_state = 1;
                EC_Stop_Get();
                TOC_Get();
                printf(" stop ec %.2f TOC %.2f\r\n",EC_Stop_value,toc_value);
            }
        }
        vTaskDelay(1000);     //1秒延时
```

```
    }
}
int main(void)
{
    Driver_MCU_Init();                                          //系统频率/分频设置
    SysTICK_SET();                                               //滴答定时器设置
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); //绑定中断向量组2

// disk_initialize(0);
// f_mount(&FileSys,"",0);
// FileResult = f_open(&File, "0:/EU_TT.TXT",FA_CREATE_ALWAYS|FA_WRITE);
// if(FileResult == FR_OK)
// {
//         //printf("Creat EU_DEMO.TXT File OK \r\n");
//         FileResult = f_write(&File,TxBuffer,sizeof(TxBuffer),&NbCount);//往文件里面写数据
//         f_close(&File);
// }
// f_mount(0, "", 0);
    My_RTC_Init();
    UARTON();
    LIGHT_Init();
    MOTOR_Init();
    SVO_Init();
    SVT_Init();
    LED_Init();
    Adc_Toc_Init();
    DS18B20_Init();
    xTaskCreate(vTask1,"LED1",64,NULL,1,NULL);          //翻转LED1(单片机状态指示)
```

```
        xTaskCreate(vTask2,"LED2",64,NULL,1,NULL);          //翻转LED2(单片机状态指
示)
        xTaskCreate(vTask3,"TASK3",1024,NULL,1,NULL);     //更新画面、获取触摸值、
控制水泵、TOC获取
一定注意分配堆栈大小，大了无法创建成功。
        xTaskCreate(vTask4,"TASK4",128,NULL,1,NULL);       //保持LVGL心跳。
        xTaskCreate(vTask5,"TASK5",512,NULL,1,NULL);       //控制逻辑
        vTaskStartScheduler();
        while(1)
        {
          ;
        }
    }
    #include "console_logic.h"
    uint8_t    oxidize_state = 0;
    uint8_t    clear_state = 0;
    void console_logic_run(uint8_t motor_s,uint8_t light_s,uint8_t svo_s,uint8_t
svt_s,uint8_t oxidize_s,uint8_t clear_s)
    {
       MOTOR_Console(motor_s);
       LIGHT_Console(light_s);
       SVO_Console(svo_s);
       SVT_Console(svt_s);
       LED_Console(led_state);
    }
```

附录B

```
import serial
import serial.tools.list_ports
import time
class Uart:
    def __init__(self):
        self.ser = None
        self.port = None
        self.baudrate = 9600
        self.timeout = 0.1
        self.connected = False
    def list_ports(self):
        ports = serial.tools.list_ports.comports()
        for port, desc, hwid in sorted(ports):
            print("{}: {} [{}]".format(port, desc, hwid))
    def connect(self, port):
        self.port = port
        self.ser = serial.Serial(port, self.baudrate, timeout=self.timeout)
        self.connected = True
    def disconnect(self):
        self.ser.close()
        self.connected = False
    def send(self, data):
        self.ser.write(data)
    def receive(self):
        return self.ser.read_all()
    def read(self, size):
        return self.ser.read(size)
    def write(self, data):
```

```
            self.ser.write(data)
        def readline(self):
            return self.ser.readline()
        def readlines(self):
            return self.ser.readlines()
        def flush(self):
            self.ser.flush()
        def reset_input_buffer(self):
            self.ser.reset_input_buffer()
        def reset_output_buffer(self):
            self.ser.reset_output_buffer()
        def in_waiting(self):
            return self.ser.in_waiting
        def out_waiting(self):
            return self.ser.out_waiting
        def close(self):
            self.ser.close()
        def __del__(self):
            if self.connected:
                self.close()
# Path: main.py
import time
uart = Uart()
filename = ""
last_time = ""
fist_data = ""
last_data = ""
uart_is_active          = False
uart_connected_flag = False
```

```
use_com = "";
while True:
    if uart_connected_flag == True:
        time.sleep(0.5)
        tt = uart.receive()
        if tt:
            tt = tt.decode("utf-8")
            #print(tt)
            if "start" in tt:
                fist_data = "开始时间：%s    EC
值：%s" %(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()), tt.split(" ")[3])
                print(fist_data)
            if "stop" in tt:
                last_time = time.strftime("%Y-%m-%d %H-%M-%S",
time.localtime())
                last_data = "结束时间：%s    EC值：%s    TOC值：%s"%(last_time,
tt.split(" ")[3], tt.split(" ")[5])
                print(last_data)
                filename = last_time + ".txt"
                with open(filename, "a",encoding="utf-8") as f:
                    f.write(fist_data + "\n")
                    f.write(last_data + "\n")
                    f.close()
    if uart_connected_flag == False:
        use_com = input("请输入串口号：")
        port_list = list(serial.tools.list_ports.comports())
        print(port_list)
        if len(port_list) > 0:
            for i in    list(range(len(port_list))):
                print(port_list[i])
```

```
if "Silicon" in port_list[i][1]:
        uart.connect(port_list[i][0])
        uart_connected_flag = True
        break
```

致　　谢

辗转四年，时光飞逝，站在大学时光的尾声，我内心充满了感慨与感恩，因为在这四年，有数不清的人伴我成长，让我如今硕果累累。

首先，我要向我的指导教师严炳辉讲师表示由衷的感谢。在我听讲上课的时候，严老师就给予了我极大的鼓励与支持，会为我分析难题，讲解原理，我内心感到无比荣幸。之后因为升学，我需要推荐人举荐，于是我便找到了严老师，希望严老师做我的推荐老师。我本以为这将会是很坎坷的一次询问，不曾想严老师立马同意了我的请求。而我也不负期望，收获了许多学校的邀请。后来老师找到我做这个课题，我感到受宠若惊，但我知道老师给予了我很大的期望，于是我欣然接受了下来。在我对课题迷茫的时候，老师主动帮我联系了其他学院的老师为我排解疑难；在我对时间把握不定的时候，老师给予了我极大的时间宽限；因此我在这着重感谢我的指导老师。

其次我要向我的父母表达感激，感谢我的父母给予了我这么好的生活条件，让我衣食无虑，让我得以全身心的投入到学习中。我要感激我得父母，不管我做出什么决定都无条件理解与支持我，我由衷感谢这22年来您们对我的无微不至，对我的无限宽容。

我要感谢我的女朋友，又或者说是未婚妻，如果没遇见她，我不可能在大一就确定好了努力的方向，我也不可能取得这么多优异的成绩与比赛奖项。是她一直在我身旁鞭笞我前行，鼓励我前进。而今我们即将共同步入新的环境，希望我们依旧能共同努力，齐步向前。

由于自身的文采水平有限，这段时光其实还有许多人伴我前行，但我就不逐步列举了，只希望未来诸位前程似锦、意气风发。最后衷心感谢能看完此篇文章的各位读者与老师，欢迎各位批评指正。