浙江理工大學

Zhejiang Sci-Tech University

本 科 毕 业 设 计

Bachelor'S THESIS

论文题目：农村生活污水处理远程监测采集器设计

专业班级：机械电子工程 13(2)班

姓名学号：夏灵能 2013330300327

指导教师：张建义

递交日期：2017. 5.10

浙 江 理 工 大 学

机械与自动控制学院

毕业论文诚信声明

我谨在此保证：本人所写的毕业论文，凡引用他人的研究成果均已在参考文献或注释中列出。论文主体均由本人独立完成，没有抄袭、剽窃他人已经发表或未发表的研究成果行为。如出现以上违反知识产权的情况，本人愿意承担相应的责任。

声明人（签名）：

年　月　日

摘　要

详细介绍了农村生活污水处理远程监测采集器设计过程中关于芯片选型、原理图设计、PCB 设计及制作、PCB 焊接、嵌入式软件开发等方面的细节内容，并且简要介绍了农村生活污水处理远程监测采集系统的构成及各部分的作用。农村生活污水处理远程监测采集器采用 ST 公司的 STM32F030C8T6 芯片为主控芯片，通过 RS485 总线从多个水质传感设备采集 PH、溶解氧浓度、温度、ORP、电导率、浊度等水质数据，通过 I2C 总线控制 OLED 屏显示水质数据，并通过 NRF24L01 模块将数据上传服务器。实验结果表明，远程监测采集器能够对水质数据进行有效采集并且准确显示。

关键词：水质监测；无线通讯；远程传输；STM32F030

Abstract

A detailed introduction to design of remote monitoring collector for rural sewage treatment from chip selection, design of schematic diagram, design and production of PCB , PCB welding, embedded software development. And briefly described the structure of rural sewage treatment remote monitoring and collecting system and roles of each part. Remote monitoring collector for rural sewage treatment using STM32F030C8T6 chip from ST company as the main control chip, reading water quality data like PH, dissolved oxygen concentration, temperature, ORP, conductivity, turbidity from a plurality of water quality sensing equipment through the RS485 bus, controlling OLED screen to display water quality data through the I2C bus, and uploads data to server through NRF24L01 module. The experimental results show that the remote monitoring collector can collect and display the water quality data accurately.

Key words：water examination; wireless communication; remote transmission; STM32F030

目　录

第 1 章 绪论

1.1 背景和意义

1.1.1 背景

自从上个世纪 80 年代诞生于美国以来，关于远程数据采集器的研究已经持续了 30 多个年头。从最早的 8 位处理器到后来的 16 位处理器再到现在的 32 位处理器甚至 64 位处理器，这些年来，远程数据采集器见证了嵌入式处理器的发展以及该领域的各种技术革新。到今天，远程数据采集器的开发技术已经相当成熟，但是仍然有许多新技术源源不断地加入进来，并且随着智能家居等概念的兴起，该领域的研究将再次成为热点。

当前，远程数据采集技术主要应用于远程无线抄表（包括水表、电表、燃气表等）、工厂生产监测、环境数据采集等。国内外也有将远程数据采集技术应用于水质监测以及水处理设备数据采集的先例。由于该技术目前处于成熟阶段，且有丰富的应有经验，可以应用在水质监测领域。本课题是该技术在农村生活污水处理这个特定场景下的应用。

1.1.2 意义

水是生命之源，没有人能否认水对生命的重要性。我国淡水资源丰富，总量为 28000 亿立方米，占全球淡水资源总量的 6%。仅此于巴西、俄罗斯、加拿大，处于世界第四的水平。然而我国水资源存在两个严重问题。一是人均水资源短缺，我国作为人口大国，人均淡水资源仅为世界人均水资源的四分之一，在世界上名列第 110 位；二是水污染严重，由于工业废水及生活污水等的大量排放，导致相关疾病频发。浙江省省委十三届四次全会，做出了“五水共治”决策：治污水，防洪水，排涝水，保供水，抓节水。其中治污水排在首位，可见对污水治理的重视程度。

在污水处理的对象中，除了高度发展的城市，还有广大处于发展中的农村。相比城市污水处理厂的大规模集中处理方式，农村的污水处理站规模更小也更加分散，管理起来难度更大。本课题为了帮助解决农村生活污水处理站数据采集困难的问题设计了远程监测采集器，方便了处理站的数据采集工作，同时也降低了

农村污水处理的成本，使得农村生活污水处理工作能够有序地顺利展开。

1.2 国内外研究现状

AMR（Automation Meter Reader）技术诞生于上个世纪 80 年代的美国[1]，距今已有 30 多年的时间，是远程数据采集器使用的主要技术。由于远程数据采集给人们的生产生活带来极大的便利，所以这方面的研究一直是国内外专家学者关注的热点。

随着计算机技术以及通信技术的高速进步，AMR 技术也在不断进步。目前，无论是在国内还是国外，远程数据采集器使用的主控芯片主要是以 MSP430 系列为代表的 16 位处理器以及以 STM32 系列为代表的 32 位处理器。相较于早期只能使用 8 处理器的年代，如今的远程数据采集器的性能普遍远胜当年的同类产品。目前推出的商用远程数据采集器较少采用 8 位处理器，其主要原因在于 8 位处理器的低性能并不能带来低成本，在与同价位的 16 位处理器或 32 位处理器产品的对比中毫无优势。

目前，远程数据采集器上传数据到服务器的主要途径仍然是 GPRS 网络。由于 2G 通信通广时间早，技术成熟，在远程数据采集器中广泛采用。远程数据采集器在接收到数据后通常使用 GSM 模块将数据发送到服务器[2]或者直接发送到手持设备[3]。虽然当下在移动端，尤其是手机端，4G 网络通信已经相当普遍，但是由于采用 4G 网络通信的成本比采用 GPRS 通信方式要高，所以，在 GPRS 通信速度能够满足需求的情况下，厂家仍然会选择使用 GPRS 通信。只有在数据量大且要求速度高的情况下，如高清视频传输，才会普遍采用 4G 通讯。

目前，远程数据采集器的应用场景主要是民用表（水表、电表[4]、燃气表等）的无线远传抄表。另外也广泛应用于农业、水质[5]、家居[6]等领域的环境数据的采集。随着智能家居、智能农业等概念的兴起，以及人们对水资源的日益关心，远程数据采集器在相关领域的应用正在不断涌现。

第 2 章 农村生活污水处理远程监测采集器原理

2.1 农村生活污水处理远程监测采集器所在系统的系统框架

整套农村生活污水处理远程监测采集系统由水质传感设备、农村生活污水处理远程监测采集器、服务器这 3 部分组成。

各部分在系统中的作用：

水质传感设备：负责水质信号的采集、处理、模数转换、采样以及数据上传。通过水质传感设备搭载的传感器将水质信息转化为电信号，并完成信号的前期处理，如：滤波、放大等。然后水质传感设备对完成前期处理的信号进行模数转换，使模拟信号转变为适合输入计算机系统的数字信号。再对转换好的数字信号进行采样，计算采样数据得出实际的水质信息，并将当前水质信息显示在水质传感设备的屏幕上。当主机，即远程监测采集器，请求水质数据时，水质传感设备将数据按照特定的协议进行打包并上传。

农村生活污水处理远程监测采集器：负责从水质传感设备获取水质信息，并将水质信息上传服务器。远程监测采集器定期向各个从机，即水质传感设备，发送命令以请求各种水质数据。然后对接受到的数据包进行解析，并记录、显示各种水质数据。再定期将记录的各种数据上传服务器。

服务器：负责接收远程监测采集器上传的数据，并为用户通过网络获取这些数据提供平台。

2.2 系统中所采用的水质传感设备

在远程监测采集系统中一共配备了 5 台水质传感设备，分别是 PH190 控制器、奥新压力仪、电导率仪、在线溶氧仪、浊度在线分析仪。

其中，PH190 控制器能够提供 PH 值、温度、ORP 等数据；奥新压力仪提供压力数据；电导率仪提供电导率、温度等数据；在线溶氧仪提供溶解氧含量、温度数据；浊度在线分析仪提供浊度数据。所以整套系统可以测量 PH 值、温度、ORP、水压、电导率、溶解氧含量、浊度等信息。

2.3 水质传感设备与农村生活污水处理远程监测采集器的通讯方式及通讯协议

2.3.1 RS485

远程监测采集器采用 RS485 总线与水质传感设备进行通信。

RS485 是隶属于 OSI 模型物理层的电气特性规定为 2 线，半双工，多点通信的标准。他的电气特性和 RS232 大不一样。用缆线两端的电压差值来表示传递信号。RS485 仅仅规定了接收端和发送端的电气特性。他没有规定或推荐任何数据协议[7]。

RS485 拥有以下特点：

接收电平低，不易损坏芯片。以两线间的电压差为+（2~6）V 表示逻辑“1”；以两线间的电压差为-（2~6）V 表示逻辑“0”。相比 RS232，RS485 接口电平更低，不容易损坏接口电路的芯片。

传输速率高。当长度为 10 米时，传输速率高达 35Mbps；当长度为 1.2 千米时，传输速率可达 100Kbps。

抗干扰能力强。采用平衡驱动器和差分接收器组合的 RS485 接口拥有更好的抗共模干扰能力。

2.3.2 各传感设备协议

根据各个水质传感设备厂家提供的通讯协议说明文件，各水质传感设备均采用 MODBUS 标准协议。在此基础上，各个设备根据自身需要传输的数据自定义了数据段的结构。

2.4 农村生活污水处理远程监测采集器硬件设计

2.4.1 需求分析

远程监测采集器需要实现：

需求 1：通过 RS485 总线与多个水质传感设备进行数据传输

使用 RS485 总线进行通信，需要占用主控芯片的 UART 资源。但是一个 RS485 驱动器的驱动能力至少可以驱动 32 个接收器[7]，所以所选用的主控芯片

只需拥有 1 个以上 UART 资源。但是为了使可能用到的串口通讯不和 RS485 总线通讯发生冲突，当然可以进行分时复用解决这个问题，选用拥有 2 个及以上 UART 资源的主控芯片，而且目前市面上主流的主控芯片普遍拥有 2 个以上 UART 口。

需求 2：显示水质数据

为了显示水质数据，需要为远程监测采集器配备一块显示屏。目前市面上主流的显示屏分为 LCD 和 OLED 两种。从操作方便和性价比两方面综合考虑，决定为远程监测采集器配备一块支持 I2C 总线通讯的 0.96 英寸 OLED 显示屏模块，分辨率为 128*64 。在 I2C 通讯过程中以远程监测采集器为主机设备，OLED 显示模块为从机设备[8]。相较于相同分辨率的 LCD 显示屏模块，该 OLED 显示屏模块价格更低，尺寸更小，功耗更低，而且技术更加先进。考虑到显示的数据内容比较简单，这个分辨率的 OLED 屏幕模块是相当合适的选择。为了操作该 OLED 模块，主控芯片需要拥有 I2C 总线资源。

需求 3：将水质数据上传服务器

无线远传设备大多采用 GPRS 或者 4G 将数据上传服务器，这种设计适合无线远传设备与服务器在空间上存在较大距离的场景。考虑到农村生活污水处理厂一般不会位于荒郊野外，一般都能实现通电通网的条件，本系统采用 raspberry pi 3B+搭建一个服务器，并使用 raspberry pi 3B+自带的引脚连接 NRF24L01 模块，与远程监测采集器上搭载的 NRF24L01 模块通过 2.4Ghz 无线通讯。采用这种设计，除了充分使用农村生活污水处理厂的宽带资源外，每个远程监测采集器每年可以节省可观的移动数据流量费用。为了使用 NRF24L01 模块，主控芯片需要拥有 SPI 总线资源。

需求 4：临时保存水质数据

为了实现数据批量上传服务器，需要在两次数据上传之间零时保存水质数据。由于每个水质传感设备每次上传的数据量较小，仅为几个 BYTES 或者十几个 BYTES，所以只需选择一块 FLASH 稍微大一点的主控芯片，就可以省下一块 FLASH 芯片及其外围电路。

2.4.2 主控芯片选型及核心电路设计

目前市场上主流的主控芯片按照位宽大致可分为 3 类，分别是：8 位 MCU（如 51 系列、STM8 系列等）、16 位 MCU（如 MSP430 系列）、32 位 MCU（如 STM32 系列）。由于远程监测采集器对于性能基本没有太高要求，即使是性能相对来说最弱的 51 系列主控芯片也完全能够实现功能。但是考虑到 51 为代表的 8 位处理器年代久远，性价比较低，虽然还在市面上流通，但更大的作用是用于单片机入门的教学，在价格相同甚至更低，性能数倍于 8 位处理器的 32 位处理器面前毫无优势可言，因为现在是一个花 8 位处理器的钱能买 32 位处理器的时代。

当下最流行的 32 位处理器应该非 ST 公司推出的 STM32 系列莫属。该系列 MCU 使用者众多，资料丰富，所以开发难度较低，是用于远程监测采集器设计开发的理想 MCU。

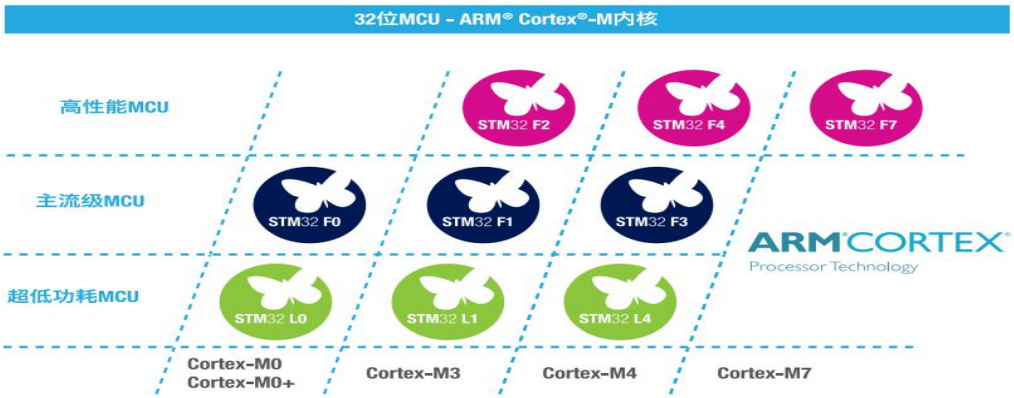

图 2-1 ST 公司 32 位 MCU ARM Cortex-M 内核产品[9]

如图 2-1 所示，ST 公司的 32 位 ARM Cortex-M 内核 MCU 一共有 9 个系列。目前市面上最流行的开发板主要为基于 Cortex-M3 内核的主流级 MCU STM32F1 系列和基于 Cortex-M4 内核的高性能 MCU STM32F4 系列 。但是由于没有过高的性能要求以及超低功耗要求，基于 Cortex-M0 内核的主流级 MCU STM32F0 系列足以满足远程监测采集器设计的需求。

STM32F0 系列又可以细分为 4 个小类，分别为 STM32F0x0 超值型、STM32F0x1 基本型、STM32F0x2 USB 产品线、STM32F0x8 超低电压。由于没有特殊需求，所以排除 STM32F0x2 USB 产品线和 STM32F0x8 超低电压。在剩

下的 STM32F0x0 超值型、STM32F0x1 基本型当中，处于成本考虑决定从 STM32F0x0 超值型系列芯片中选取一款芯片用于远程监测采集器的设计。STM32F0x0 超值型系列各 MCU 资源如图 2-2 所示。

产品型号	主频 (MHz)	内核	FLASH (KB)	RAM (KB)	EEPROM (B)	封装	通用IO	最低工作电压	最高工作电压	16位定时器	32位定时器	电机控制定时器	低功耗定时器	高分辨率定时器	12位ADC转换单元	12位ADC通道	16位ADC转换单元	16位ADC通道	12位DAC	比较器	放大器	SPI	QUADSPI
STM32F0x0超值型 – 48 MHz																							
STM32F030C6T6	48	ARM Cortex-M0	32	4	0	LQFP48	39	2.4	3.6	5	0	1	0	0	1	13	0	0	0	0	0	1	0
STM32F030C8T6	48	ARM Cortex-M0	64	8	0	LQFP48	39	2.4	3.6	7	0	1	0	0	1	16	0	0	0	0	0	2	0
STM32F030F4P6	48	ARM Cortex-M0	16	4	0	TSSOP20	15	2.4	3.6	5	0	1	0	0	1	12	0	0	0	0	0	1	0
STM32F030K6T6	48	ARM Cortex-M0	32	4	0	LQFP32	26	2.4	3.6	5	0	1	0	0	1	13	0	0	0	0	0	1	0
STM32F030R8T6	48	ARM Cortex-M0	64	8	0	LQFP64	55	2.4	3.6	7	0	1	0	0	1	16	0	0	0	0	0	2	0
STM32F030RCT6	48	ARM Cortex-M0	256	32	0	LQFP64	51	2.4	3.6	8	0	1	0	0	1	18	0	0	0	0	0	2	0
STM32F030CCT6	48	ARM Cortex-M0	256	32	0	LQFP48	37	2.4	3.6	8	0	1	0	0	1	12	0	0	0	0	0	2	0
STM32F070C6T6	48	ARM Cortex-M0	32	6	0	LQFP48/ UQFN48	38	2.4	3.6	5	0	1	0	0	1	13	0	0	0	0	0	2	0
STM32F070CBT6	48	ARM Cortex-M0	128	16	0	LQFP48/ UQFN48	37	2.4	3.6	8	0	1	0	0	1	13	0	0	0	0	0	2	0
STM32F070F6P6	48	ARM Cortex-M0	32	6	0	TSSOP20	16	2.4	3.6	5	0	1	0	0	1	12	0	0	0	0	0	1	0
STM32F070RBT6	48	ARM Cortex-M0	128	16	0	LQFP64	51	2.4	3.6	8	0	1	0	0	1	19	0	0	0	0	0	2	0

产品型号	I2S	I2C	高速I2C	U(S)ART	低功耗UART	CAN	SDIO	FSMC	FMC	从USB	全速USB OTG	高速USB OTG	以太网	MDIOS	段式LCD	TFT LCD	DSI HOST	SAI	SPDIFRX	DFSDM	DCMI	SWPMI	TRNG	AES	DES/TDES	SHA	HMAC
STM32F030C6T6	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STM32F030C8T6	0	2	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STM32F030F4P6	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STM32F030K6T6	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STM32F030R8T6	0	2	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STM32F030RCT6	0	2	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STM32F030CCT6	0	2	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STM32F070C6T6	0	1	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STM32F070CBT6	0	2	0	4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STM32F070F6P6	0	1	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STM32F070RBT6	0	2	0	4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

图 2-2 STM32F0x0 超值型系列 MCU[9]

由于需要临时保存水质数据，所以不考虑 FLASH 容量为 32KB 及以下的 MCU。在 FLASH 容量为 64KB 及以上的 MCU 中选择了 STM32F030C8T6 这款 MCU 用于远程监测采集器的设计开发。当然要使用一块芯片开发产品之前，除了要考虑芯片的资源以及性能之外，还需要考虑芯片的使用条件限制。由于远程监测采集器并不需要在极端环境下工作，STM32F030C8T6 可以满足条件.

Symbol	Parameter	Conditions	Min	Max	Unit
f_{HCLK}	Internal AHB clock frequency		0	48	MHz
f_{PCLK}	Internal APB clock frequency		0	48	
V_{DD}	Standard operating voltage		2.4	3.6	V
V_{DDA}	Analog operating voltage	Must have a potential equal to or higher than V_{DD}	2.4	3.6	V
$V_{IN}^{(1)}$	Input voltage on FT and FTf pins		V_{SS} –0.3	V_{DD} +4.0	V
	Input voltage on TTa pins		V_{SS} –0.3	4.0	V
	Input voltage on any other pin		V_{SS} –0.3	4.0	V
P_D	Power dissipation at T_A = 85 °C for suffix 6(2)	LQFP64	-	444	mW
		LQFP48	-	364	
		LQFP32	-	357	
		TSSOP20	-	182	
TA	Ambient temperature for 6 suffix version	Maximum power dissipation	–40	85	°C
		Low power dissipation(3)	–40	105	
TJ	Junction temperature range	6 suffix version	–40	105	°C

图 2-3 STM32F030 操作条件[10]

图 2-3 为芯片所在系列的数据手册对其工作条件的描述。

STM32F030C8T6 每片 4 元人民币左右的零售价甚至低于一些 51 内核的 8 位 MCU。尽管 Cortex-M0 最低配置只有 12000 逻辑门，在规模上同 8 位和 16 位处理器差不多，却具有完整的 32 位核心，其优势是 8 位和 16 位设备所不能比拟的[11]。

STM32F030C8T6 核心电路设计

再强大的 MCU 也不可能离开最基本的外围电路工作，STM32F030C8T6 也不例外，合适的核心电路设计是芯片正常工作的基础。

核心电路需要解决芯片的供电、Boot 选择、时钟、复位电路等基本需求。

对于供电电压，在数据手册 STM32F030x4 STM32F030x6 STM32F030x8 中的 3.5 章 Power management 中的 3.5.1 节 Power supply schemes 中有如下定义：

VDD = 2.4 to 3.6 V: external power supply for I/Os and the internal regulator. Provided externally through VDD pins[10].

VDDA = 2.4 to 3.6 V: external analog power supply for ADC, Reset blocks, RCs and PLL. The VDDA voltage level must be always greater or equal to the VDD voltage level and must be provided first[10].

即该手册定义的芯片需要提供两类电源，分别为模拟电源 VDDA 和供电电源 VDD，且模拟电源必须首先供应同时不得低于供电电源。鉴于两者电压都处于 2.4V 到 3.6V 之间，出于简化电路的考虑，将两者的供电电压统一为 3.3V，并通过 0 欧姆电阻将两者相连接。

对于 Boot 选择，在数据手册中的第 3.3 章 Boot modes 中有如下定义：

At startup, the boot pin and boot selector option bit are used to select one of three boot options:

Boot from User Flash

Boot from System Memory

Boot from embedded SRAM[10]

The boot loader is located in System Memory. It is used to reprogram the Flash memory by using USART on pins PA14/PA15 or PA9/PA10[10].

不同于一般芯片拥有 BOOT0、BOOT1 两个引脚控制 BOOT 选项，数据手册定义的这些芯片只留出一个 BOOT0，另外一个通过串口访问 System Memory 中的 boot loader 进行操作，以软件的方式实现。所以，BOOT 电路只需能够配置 BOOT0 脚的电平即可。

对于时钟，在数据手册中的第 3.6 章 Clocks and startup 中有如下定义：

System clock selection is performed on startup, however the internal RC 8 MHz oscillator is selected as default CPU clock on reset. An external 4-32 MHz clock can be selected, in which case it is monitored for failure. If failure is detected, the system automatically switches back to the internal RC oscillator[10].

即复位后选择以内部 8M 晶振为默认 CPU 时钟，但也可以选择外部 4-32M 晶振。使用外部晶振时会监测是否起振失败，若失败则自动切换回内部晶振。

所以理论上完全可以省略外部晶振，但考虑到外部晶振的最高频率可达内部晶振的 4 倍，决定留出外部晶振电路，以备后期可能的高频应用之需。

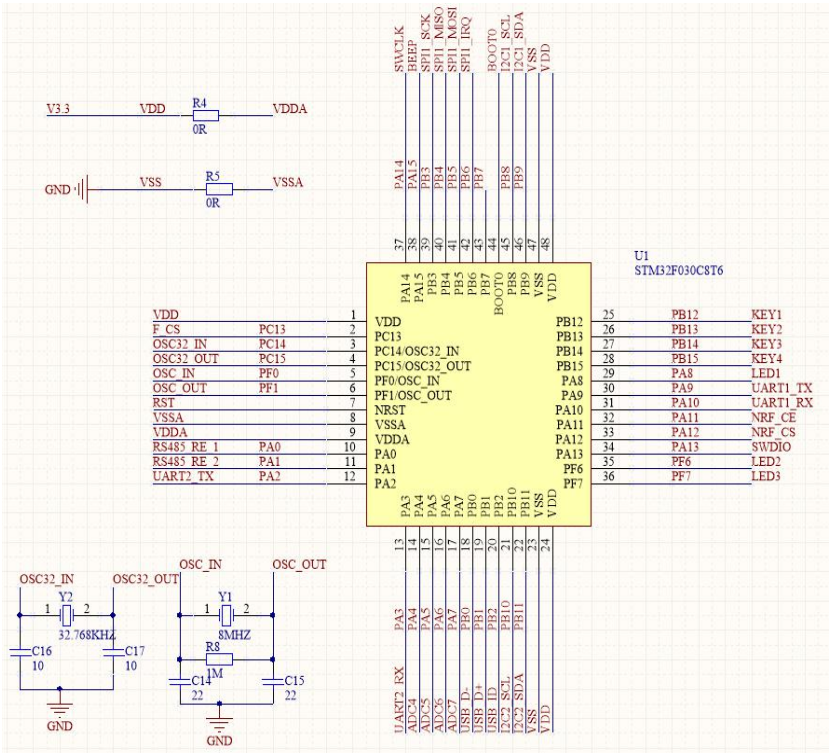

图 2-4 STM32F030C8T6 核心电路原理图

最终，基于 STM32F030C8T6 设计的远程监测采集器核心电路如图 2-4 所示。

2.4.3 RS485 通讯模块设计

与远程监测采集器连接的各个水质传感设备来自不同的厂家，但是他们都支持使用 RS485 总线通讯。为了实现与这些水质传感设备之间的数据交互，远程监测采集器必须支持 RS485 通讯并且兼容各水质传感设备厂家定义的通讯协议。

RS485 电平无法直接与 STM32F030C8T6 相连接，需要使用电平转换芯片[7]。这里选用的电平转换芯片为 RS485 电路中常用的 SP3485 芯片。根据 SP3485 芯片的 DATASHEET 中的描述：The SP3481 and the SP3485 are a family of +3.3V low power half-duplex transceivers that meet the specifications of the RS-485 and RS-422 serial protocols[12].该芯片可以使用 3.3 伏电源供电且支持 RS485 协议。

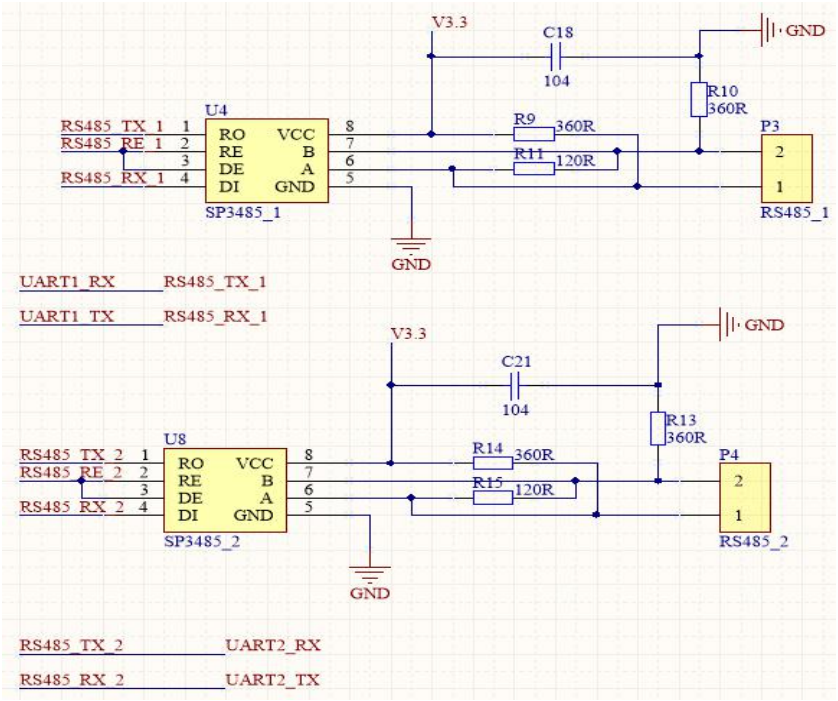

图 2-5 RS485 模块原理图

如图 2-5 所示，远程监测采集器上设计了两个 RS485 模块，其中 RS485_1 由串口 1 控制，其中 RS485_2 由串口 2 控制。每个 RS485 分别有一个 120 欧姆的终端匹配电阻和两个 360 欧姆的偏置电阻，偏置电阻保证了空闲时信号不会波动。

2.4.4 OLED 显示模块设计

在需求分析中已经确定要为远程监测采集器配备一块支持 I2C 通讯的 0.96 寸 OLED 屏幕。出于简化设计和降低成本的考虑，没有为远程监测采集器采用裸屏，而是采用成熟的 OLED 显示屏模块。这样可以在 PCB 上省略 OLED 屏的外围电路，而只需要提供 I2C 总线接口以及电源和地。

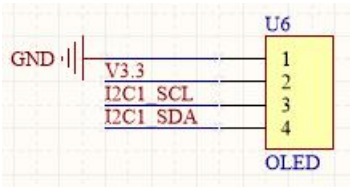

图 2-6 OLED 模块原理图

如图 2-6 所示，只需要为 OLED 模块留出 1 个 4 针接口，简化电路图的同时提高了电路的可靠性。

2.4.5 供电模块及电压监测电路设计

目前，嵌入式设备中最常用的电源电压是 5V 和 3.3V。截至目前，在整个远程监测采集器设计中，无论是核心电路还是 RS485 电路还是 OLED 模块电路所使用的供电电压都是 3.3V，之后需要用到的 NRF24L01 模块也可以使用 3.3V 电源供电。并且在远程监测采集器中没有使用到其它电压值的供电的器件。所以，理论上整个远程监测采集器只需要设计 3.3V 电源即可。但是考虑到 5V 电源供电器件广泛应用，若只提供 3.3V 电源将会对后续的功能升级造成很大的限制，所以决定加上 5V 电源的设计。最终，远程监测采集器将提供 5V 和 3.3V 两种电压值的供电模块。

Parameter	Device	Conditions	Min	Typ	Max	Units
Reference Voltage (Note 2)	AMS1117	I_{OUT} = 10 mA 10mA ≤ I_{OUT} ≤ 1A, 1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V	1.238 **1.225**	1.250 **1.250**	1.262 **1.270**	V V
Output Voltage (Note 2)	AMS1117-1.5	0 ≤ I_{OUT} ≤ 1A , 3.0V≤ V_{IN} ≤ 12V	1.485 **1.476**	1.500 **1.500**	1.515 **1.524**	V V
	AMS1117-1.8	0 ≤ I_{OUT} ≤ 1A , 3.3V≤ V_{IN} ≤ 12V	1.782 **1.773**	1.800 **1.800**	1.818 **1.827**	V V
	AMS1117-2.5	0 ≤ I_{OUT} ≤ 1A , 4.0V≤ V_{IN} ≤ 12V	2.475 **2.460**	2.500 **2.500**	2.525 **2.560**	V V
	AMS1117-2.85	0 ≤ I_{OUT} ≤ 1A , 4.35V≤ V_{IN} ≤ 12V	2.82 **2.79**	2.850 **2.850**	2.88 **2.91**	V V
	AMS1117-3.3	0 ≤ I_{OUT} ≤ 1A , 4.75V ≤ V_{IN} ≤ 12V	3.267 **3.235**	3.300 **3.300**	3.333 **3.365**	V V
	AMS1117-5.0	0 ≤ I_{OUT} ≤ 1A , 6.5V ≤ V_{IN} ≤ 12V	4.950 **4.900**	5.000 **5.000**	5.050 **5.100**	V V

图 2-7 Electrical characteristics for AMS1117[13]

经过筛选，决定采用广泛使用的 AMS1117 系列稳压芯片，具体型号分别为 AMS1117_3.3 和 AMS1117_5.0 。如图 2-7 中 AMS1117 系列芯片的电气特性所示，这两块电源稳压芯片的最大输入电压均为 12V，所以远程监测采集器采用 12V 直流电源适配器供电。另外远程监测采集器还配备了移动端常用的 Micro USB 接口,可以通过电压值为 5V 的手机电源适配器为之供电，在电流需求不高的情况下可以通过电脑的 USB 口对远程监测采集器进行供电，这种供电方式在使用电脑通过 USB 与远程监测采集器进行数据交互的应用场景下显得十分便利。

为防止在使用 USB 时将 5V 电压直连 12V 供电电压而导致损坏 5V 电压设备的情况，在电路中加入二极管保证电流只能从 5V 流向供电电压而不可逆向流动。

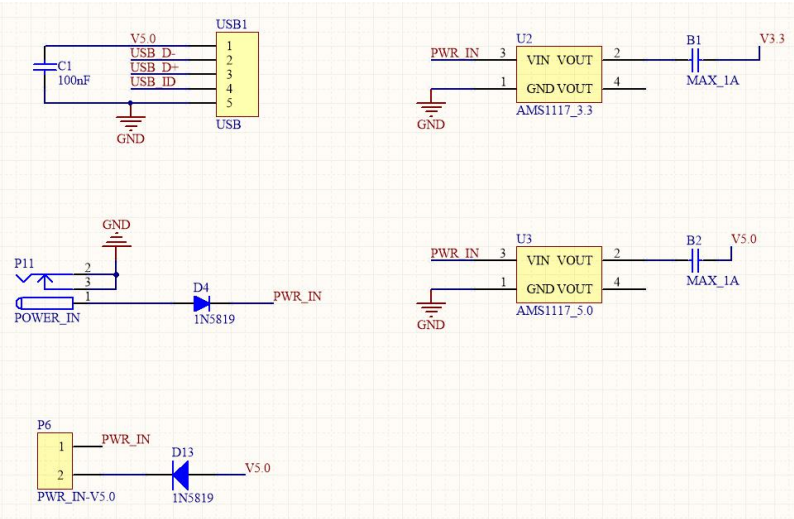

图 2-8 电源模块原理图

为了使远程监测采集器拥有监测自身电压情况的能力，以便在电压异常时及时报警，所以决定使用 STM32F030C8T6 自带的 ADC 资源监测电源模块电压状况。通过 ADC 可以将连续的模拟信号转换成适合于数字处理的二进制数[14]。图 2-9 为电压监测模块的原理图。

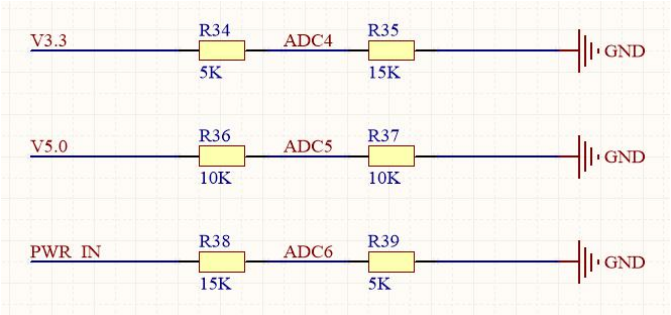

图 2-9 电压监测模块原理图

2.4.6 NRF24L01 无线通讯模块设计

远程监测采集器采用 NRF24L01 模块通过 2.4Ghz 无线与服务器进行数据交互。出于简化设计和降低成本的考虑，没有为远程监测采集器重新设计 NRF24L01 芯片外围电路及天线电路而是采用现成的 NRF24L01 模块。这样设计的好处是可以方便更换损坏的模块，而只需要提供 SPI 总线接口、片选、使能、电源和地。

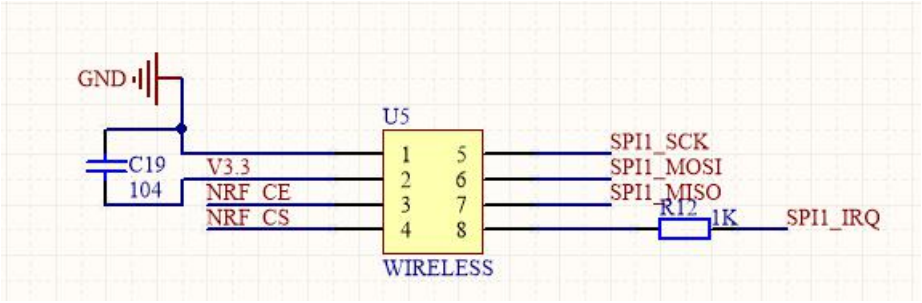

图 2-10 NRF24L01 模块原理图

2.4.7 农村生活污水处理远程监测采集器 PCB 设计及焊接

在完成远程采集监测采集器的原理图设计之后，进行了远程监测采集器的 PCB 设计。PCB 采用双层设计，双面敷铜，成品尺寸为 50mm*50mm 。

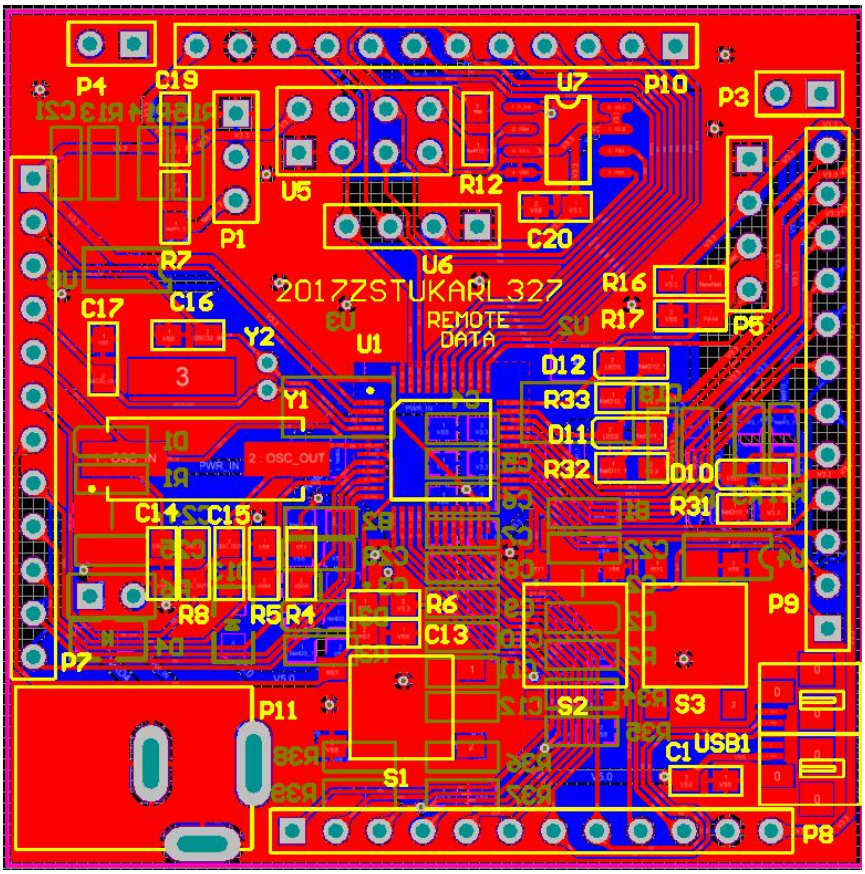

图 2-11 远程检测采集器 PCB 图

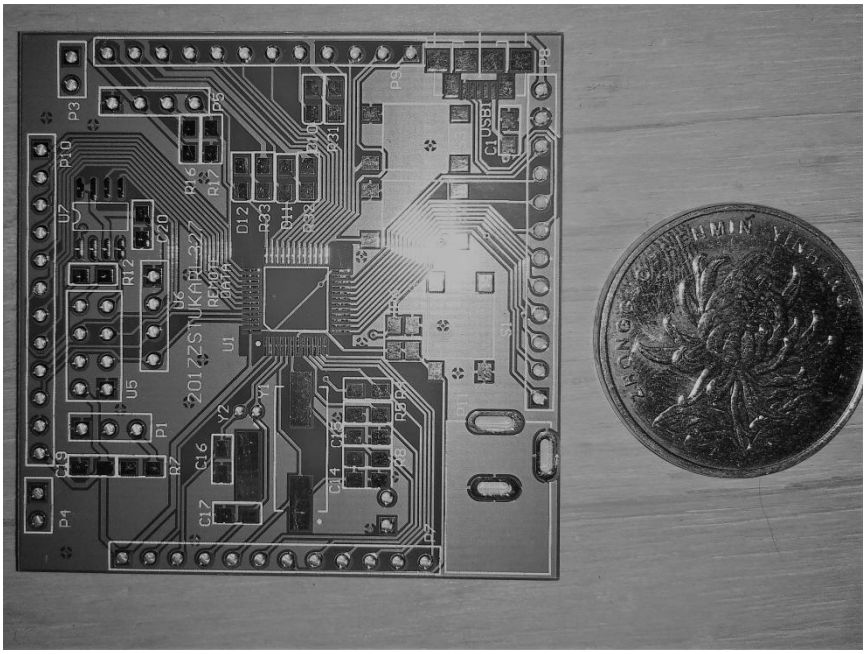

图 2-12 远程监测采集器 PCB 实物图

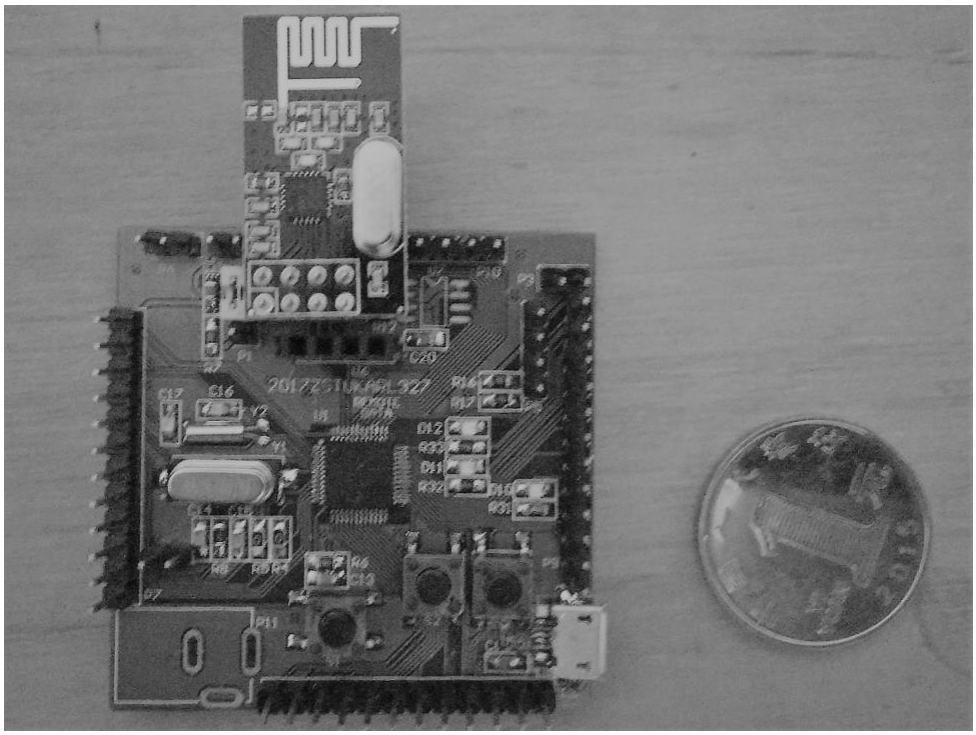

图 2-13 远程监测采集器焊接实物图

图 2-12 为使用远程监测采集器 PCB 图制作出来的 PCB 实物图，而图 2-13 则是使用远程监测采集器 PCB 实物焊接完成的成品。

2.5 农村生活污水处理远程监测采集器软件设计

2.5.1 En.stm32cubef0 Firmware 介绍及移植

En.stm32cubef0 是 ST 公司为 STM32F0 系列 MCU 提供的官方固件库。下载路径为：http://www.st.com/en/embedded-software/stm32cubef0.html 在该页面点击 GET SOFTWARE 将会跳到该页底部，再点击 STM32CubeF0 后面的 GET SOFTWARE 将开始下载，但是要在官网下载文件需要事先注册账号。

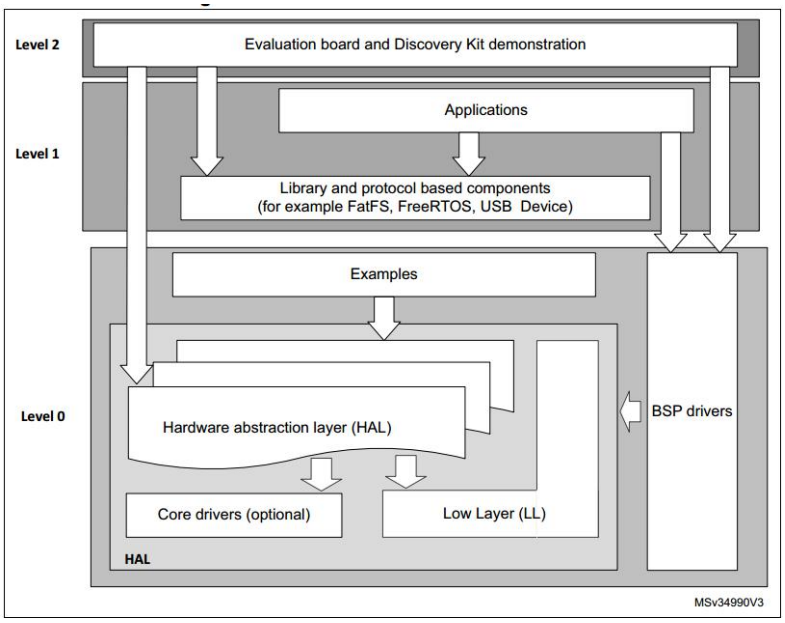

图 2-14 STM32CubeF0 firmware architecture[15]

stm32cubef0 固件的结构如图 2-14 所示，提供了从底层到应用层丰富的 API。若使用官方的评估板进行开发，则可以直接调用官方提供的所有 API。但是由于远程检测采集器的 PCB 为自行绘制的，与官方评估板的管脚定义及板载资源都不同，无法直接使用 API。当然差异主要集中在 BSP 这一层，通过修改 BSP 将代码移植到远程监测采集器上，但是由于 PCB 差异较大，而且远程需要实现的功能比较简单，移植代码并不划算。所以决定只使用官方的 HAL 层驱动而自己编写 BSP 层驱动以及其上代码。

HAL 层驱动的源码和头文件位置：

en.stm32cubef0\STM32Cube_FW_F0_V1.7.0\Drivers\STM32F0xx_HAL_Driver 文件夹中的 Src 和 Inc 文件夹中，根据项目的需求添加到工程中。

启动文件 startup_stm32f030x8.s 的路径为：

en.stm32cubef0\STM32Cube_FW_F0_V1.7.0\Drivers\CMSIS\Device\ST\STM32F0xx\Source\Templates\arm。

C 文件 system_stm32f0xx.c 的路径为：

en.stm32cubef0\STM32Cube_FW_F0_V1.7.0\Drivers\CMSIS\Device\ST\STM32F0xx\Source\Templates。

接口文件 stm32f0xx.h 的路径为：

en.stm32cubef0\STM32Cube_FW_F0_V1.7.0\Drivers\CMSIS\Device\ST\STM32F0xx\Include。在接口文件中需要取消其中对启动文件的注释才能调用启动文件工作。

在自己新建的工程中移植完上述文件后就可以使用官方固件库进行开发了。

2.5.2 RS485 通讯模块驱动程序设计

RS485 通讯模块驱动主要编写了 RS485_Init、RS485_Send_Data、RS485_Receive_Data、RS485_Check、Error_Handler 这 5 个函数，此外还重写了固件库中自带的，使用 _weak 声明的 HAL_UART_TxCpltCallback、HAL_UART_RxCpltCallback、HAL_UART_ErrorCallback 这 3 个函数。

```
void RS485_Init(u32 bound)
{
  GPIO_InitTypeDef  gpioinitstruct;
  __HAL_RCC_GPIOA_CLK_ENABLE();
  gpioinitstruct.Pin = RS485_2_RE_PIN;
  gpioinitstruct.Mode = GPIO_MODE_OUTPUT_PP;
  gpioinitstruct.Pull = GPIO_NOPULL;
  gpioinitstruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(RS485_2_RE_GPIO_PORT, &gpioinitstruct);
  UartHandle.Instance        = USART2;
  UartHandle.Init.BaudRate   = bound;
  UartHandle.Init.WordLength = UART_WORDLENGTH_8B;
  UartHandle.Init.StopBits   = UART_STOPBITS_1;
  UartHandle.Init.Parity     = UART_PARITY_NONE;
  UartHandle.Init.HwFlowCtl  = UART_HWCONTROL_NONE;
  UartHandle.Init.Mode       = UART_MODE_TX_RX;
  UartHandle.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if(HAL_UART_DeInit(&UartHandle) != HAL_OK)
  {
    Error_Handler();
  }
  if(HAL_UART_Init(&UartHandle) != HAL_OK)
  {
    Error_Handler();
  }
}
```

图 2-15 RS485_Init 函数截图

函数 RS485_Init 用于初始化 RS485。由于 RS485 使用串口资源进行工作，所以在配置好结构体后调用固件库中提供的 HAL_UART_Init 函数进行初始化。

```
void RS485_Send_Data(u8 *buf,u8 len)
{
  RS485_2_RE_HIGH();
   /* The board sends the message and expects to receive it back */

  /*##-2- Start the transmission process #####################################*/
  /* While the UART in reception process, user can transmit data through
     "aTxBuffer" buffer */
  if(HAL_UART_Transmit(&UartHandle,(uint8_t*)buf, len,1000) != HAL_OK)
  {
    Error_Handler();
  }
  /*##-3- Wait for the end of the transfer ###################################*/

  /* Reset transmission flag */
  UartReady = RESET;
  RS485_2_RE_LOW();
}
```

图 2-16 RS485_Send_Data 函数截图

函数 RS485_Send_Data 用于发送数据。由于 RS485 有专门引脚控制发送和接收模式，所以在函数的开始先将控制发送状态的引脚拉高，然后使用固件库中提供的 HAL_UART_Transmit_IT 函数进行数据传输。等到数据发送结束后拉低控制发送状态的引脚，切换到接收模式。

```
void RS485_Receive_Data(u8 *buf,u8 len)
{
  /*##-4- Put UART peripheral in reception process ###########################*/
  while(HAL_UART_Receive_IT(&UartHandle, (uint8_t *)buf, len) != HAL_OK)
  {
    Error_Handler();
  }
}
```

图 2-17 RS485_Receive_Data 函数截图

函数 RS485_Receive_Data 用于设置 RS485 接收到的数据存放的位置，通过调用固件库提供的 HAL_UART_Receive_IT 函数实现。实际的数据接收任务由中断服务函数完成。

```
void RS485_Check(void){
  uint8_t ReceiveDataAddr=0x00;
  if(RS485Reg){
    memcpy(aRxBufferBackUp,aRxBuffer,RXBUFFERSIZE);
    memcpy(aRxBuffer,allZero,RXBUFFERSIZE);
    if(RS485Reg&RS485_2_REC){
      ReceiveDataAddr=aRxBufferBackUp[0];
      switch (ReceiveDataAddr){
        case PHMeterAddr:
          memcpy(PHMeterDataBuf,aRxBufferBackUp,PHMETER_DATABUF_SIZE);
          PHMeterReg|=PHMETER_RBUF_UPDATE;
          break;
        case DOMeterAddr:
          memcpy(DOMeterDataBuf,aRxBufferBackUp,DOMETER_DATABUF_SIZE);
          DOMeterReg|=DOMETER_RBUF_UPDATE;
          break;
        default: ;
      }
      memcpy(aRxBufferBackUp,allZero,RXBUFFERSIZE);
    }
  RS485Reg&=~RS485_2_REC;
  }
}
```

图 2-18 RS485_Check 函数截图

函数 RS485_Check 用于在最外层 while 循环中检测是否有新的 RS485 接收事件发生。由于数据接收使用中断服务函数进行，而在中断服务函数中应尽量简化操作以免影响别的中断，或者被别的中断所中断，所以在中断服务函数中接收数据后不会对数据进行进一步处理，而是将标志位置位表示时间发生后退出中断。所以需要使用类似 RS485_Check 这样的函数在 while 循环中 check 是否发生相应事件，若发生则进行细节处理，若未发生则跳过。

在函数 RS485_Check 中主要是根据接收到的数据中的地址信息判断数据的来源。在确定来源之后，将接收到的数据拷贝到特意为该来源设备分配的缓存中，并置位相应标识位以便相应的 Check 函数检查和处理。此时 RS485 接收缓存中的数据已经没有意义了，为了防止影响到下一次的数据接收，需要将接收缓存中的数据清除。然后再调用函数 RS485_Receive_Data 设置下次接收到数据的存放位置。

函数 Error_Handler 如其字面意思所示，用于处理错误。目前并没有对错误类型进行细分，所有在 RS485 操作过程中出现的错误都会进入这里。

HAL_UART_TxCpltCallback 、 HAL_UART_RxCpltCallback 、 HAL_UART_ErrorCallback 这 3 个函数是分别由固件库在发送数据结束、接收数据结束、发生错误时调用的。在其内对相应标志位进行操作就可以判断发送是否完成、接收是否完成、是否发生错误等情况。

2.5.3 OLED 显示模块驱动程序设计

OLED 显示模块驱动源码由淘宝卖家提供，在修改管脚预定义并用固件库提供的，相同功能的函数替换源码中出现的，固件库中没有的函数后移植到本工程中。移植后可以完成如字符显示、数字显示、图片显示、汉字显示的基本功能，足够满足远程监测采集器的需求。

2.5.4 NRF24L01 无线通讯模块驱动程序设计

NRF24L01 无线通讯模块驱动源码同样由淘宝卖家提供，移植过程同 OLED 显示模块驱动源码的移植大同小异。但是无线通讯模块驱动在移植时无法进行有效地调试，因为要调试接收或者发送的前提是另一部分的代码能够正常工作，而要想调试另一部分代码到能够正常工作的前提则是要保证本部分代码能够正常工作，如果本部分代码能够正常工作我还调试啥。所以在没有移植成功的时候，无法判断问题出在发送还是接收或者两者都有问题。

2.5.5 各水质传感设备驱动程序设计

远程监测采集器支持与多款水质监测传感设备进行数据交互，虽然都是通过 RS485 总线通信，且都遵守 MODBUS 协议，然而各款水质传感设备的命令结构以及数据结构都不相同。由于各款水质传感设备的结构差异较大，开发一个兼容各协议的驱动难度过大，所以针对每款水质传感设备的协议对立开发驱动。

虽然数据的结构不同，但是总体的流程是相似的，接下来以 PH190 控制器的驱动为例介绍水质传感设备驱动的工作流程。

在 PH190 控制器的驱动 PHMeter.c 文件中有 14 个函数。其中 ERRHandle 类函数有 3 个，如：PHMeterErrorHandle、PHMeterTOHandle、PHMeterCRCHandle；发送命令类函数有 6 个，如：PHMeterRequestData、PHMeterRequestPH、PHMeterRequestT、PHMeterRequestPHT、PHMeterRequestORP、PHMeterRequestORPT；接收数据类函数有 3 个，如：PHMeterErrReceiveHandle、PHMeterDataReceiveHandle、PHMeterReceiveHandle；通用类函数有两个，如：PHMeterDisplay、PHMeterCheck。

ERRHandle 类函数用于处理发生错误的场景，PHMeterErrorHandle 用于处理通用错误，PHMeterTOHandle 用于处理命令超时的错误，PHMeterCRCHandle 用

于处理 CRC 校验错误的错误。

发送命令类函数用于生成命令并发送。PHMeterRequestPH、PHMeterRequest、PHMeterRequestPHT、PHMeterRequestORP、PHMeterRequestORPT 这 5 个函数会根据协议处理各条命令中存在差异的部分，最后这 5 个函数都会调用函数 PHMeterRequestData 来处理各命令中共同的部分并将生成号的命令通过调用函数 RS485_Send_Data 发出去。

接收数据类函数用于接收数据并进行处理。其中 PHMeterReceiveHandle 为接收类函数总入口，接收到的数据都会在这个函数中通过判断标志位区分是数据还是错误反馈。

若接收到的是数据，则交由函数 PHMeterDataReceiveHandle 处理。在 PHMeterDataReceiveHandle 中会先进行 CRC 校验。若校验失败则直接丢入 PHMeterCRCHandle 中处理，若校验通过则会根据之前发送的命令来确定本条数据所响应的命令，并将数据取出存放到相应寄存器中。

若接收到的是错误反馈，则交由函数 PHMeterErrReceiveHandle 处理。在 PHMeterErrReceiveHandle 中会先进行 CRC 校验。若校验失败则直接丢入函数 PHMeterCRCHandle 中处理，若校验通过则会根据之前发送的命令来确定本条错误所响应的命令，然后重新发送该命令。

通用类函数用于处理通用任务。PHMeterCheck 用于监测是否接收到新的 PH 数据。上文中的函数 RS485_Check 判断数据源地址后，若数据源地址为 PH190 控制器，则会将接收到的数据拷贝到 PHMeter 专用的缓存中，并将寄存器 PHMeterReg 中的新数据接收位置 1。而 PHMeterCheck 则是通过判断寄存器 PHMeterReg 中的新数据接收位来判断是否接收到 PH 数据。若接收到新的 PH 数据，则调用函数 PHMeterReceiveHandle 对数据进行处理。

PHMeterDisplay 通过调用 OLED 驱动在 OLED 屏幕上显示 PHMeter 界面。在界面上会表明当前显示的是 PHMeter 数据，当前状态是否正常以及各项读取到的最新数据。

2.5.6 主函数框架设计

截至目前，远程监测采集器的软件开发部分只完成了一些基础功能，如数据

采集、数据显示等。而数据存储、上传服务器等功能暂时尚未实现。接下来，在此背景下介绍主函数框架。

```
int main(void)
{
  uint8_t roundCnt=0x00;
  HAL_Init();
  /* Configure the system clock to 48 MHz */
  SystemClock_Config();
  RS485_Init(9600);
  OLED_Init();
  BSP_LED_Init(LED1);
  BSP_LED_Init(LED2);
  BSP_LED_Init(LED3);
  OLED_Clear();
  DOMeterDisplay();
  RS485_Receive_Data(aRxBuffer,RXBUFFERSIZE);
  while (1)
  {
    CMDSwitch(roundCnt);
    DisplaySwitch(roundCnt);
    RS485_Check();
    PHMeterCheck();
    DOMeterCheck();
    HAL_GPIO_TogglePin(LED2_GPIO_PORT, LED2_PIN);
    /* Insert delay 100 ms */
    HAL_Delay(100);
  roundCnt++;
  }
}
```

图 2-19 远程监测采集器主函数代码

在主函数最开始定义了一个变量 roundCnt，用于记录在 while 循环中循环的次数，作为后续命令切换以及显示界面切换的依据。

在变量定义之后是硬件抽象层函数的初始化、系统时钟初始化以及 BSP 的初始化。在一系列的初始化之后，OLED 开始显示溶氧仪数据界面。由于此时尚未发送读取数据的命令，所以所有参数值都为初始值 0。

在显示界面后。开始设置中断接收到的数据的存放位置，并开启接收中断，随后进入 While 循环。

在循环开始后，根据当前的圈数发送命令从水质传感设备读取数据。然后，根据当前圈数切换要显示的水质传感设备数据界面。之后检查是否接收到新的数据，有则处理。再之后检查是否接收到新的 PH 表数据，有则处理。再之后，检查是否接收到新的溶氧仪数据，有则处理。最后改变 LED2 状态并延时，提示程序正常运行。

第 3 章 成果展示

3.1 数据显示界面介绍

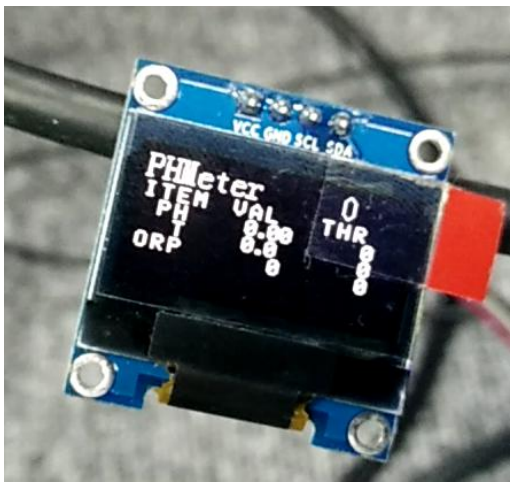

图 3-1 PH 表数据显示界面初始化状态

图 3-2 溶氧仪数据显示界面初始化状态

图 3-1 与图 3-2 分别为 PH 表和溶氧仪的数据显示界面的初始化状态。

在图 3-1 中，第一行左侧显示的 PHMeter 字样表明当前显示的数据来自 PH 表，右侧的数字表示当前的状态，0 代表正常状态。

第二行中，从左到右显示了 ITEM 、VAL 、THR 三个字符，为英文单词 item 、value 、threshold 的缩写，代表项目、值、门限。

第三到五行为数据显示区域，分别显示了 PH 值、温度、ORP 值的数据及门限。

在图 3-2 中，第一行左侧显示的 DOMeter 字样表明当前显示的数据来自溶

氧仪，右侧的数字表示当前的状态，0 代表正常状态。

第二行中，从左到右显示了 ITEM 、MIN 、VAL 、MAX 四个字符，为英文单词 item 、minimum 、value 、maximum 的缩写，代表项目、溶氧低报警值、值、溶氧高报警值。

第三到五行为数据显示区域，分别显示了溶氧值、温度、延时的数据、溶氧低报警值及溶氧高报警值。

3.2 数据显示结果与原表比较

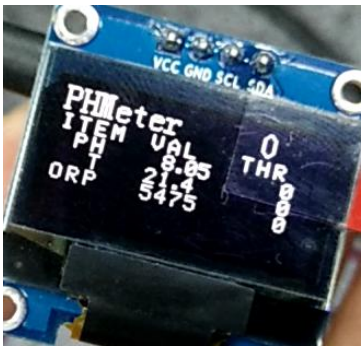

图 3-1 PH 数据显示界面结果

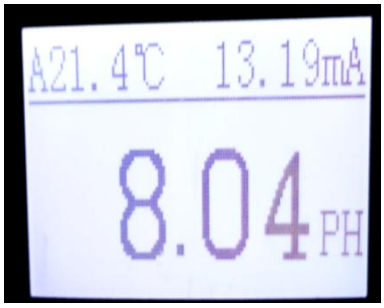

图 3-2 PH 表数据显示

图 3-3 溶氧仪数据显示界面结果

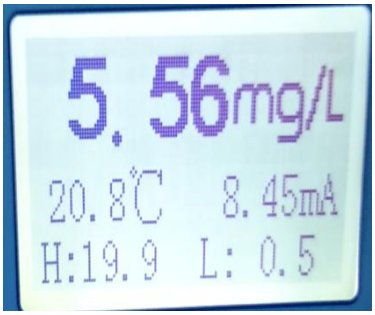

图 3-4 溶氧仪数据显示

由于数据读取命令的发送有一定周期，所以显示的数据并不是实时对应的。另外水质传感设备灵敏度较高，读取的数据会不时发生变动，所以会导致两者显示结果存在细微差异。但是通过调试，可以看到接收数据是准确的，并且远程监测采集器能够正确显示。

3.3 实物接线图

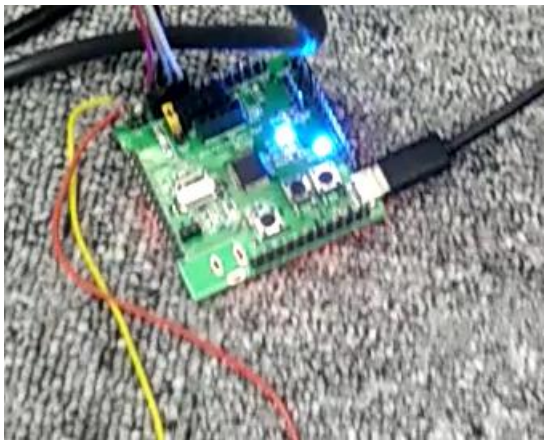

图 3-5 远程监测采集器接线图

图 3-5 为远程监测采集器实物接线图。左上侧两根为 485 总线，左上侧的一束线位 OLED 连接线，右下角位 USB 供电线。需要说明的是 OLED 显示模块本可以直接插在主板上，但是卖家发货时搞错了线序，还好主板上将所有引脚引出，所以使用杜邦线连接在主板的左上角。另外，由于上传服务器功能未开发完成，所以主板上未连接 NRF24L01 模块。

参考文献

[1] 陈鹏. 基于 ARM 的无线抄表系统的研究与应用[D]. 陕西:西安电子科技大学, 2009.

[2] Goel,A. &Mishra,R.S. Remote Data Acquisition Using Wireless - Scada System[J]. International Joumal of Engineering(IJE), 2009, Volume(3): 58~64.

[3] Ionel, R. Vasiu, G. Mischie, S. GPRS based data acquisition and analysis system with mobile phone control[J]. Measurement, 2012, 45: 1462~1470.

[4] 刘亚善. 基于 ARM/GPRS 的远程电量数据采集器的设计[D]. 河北:河北农业大学, 2015.

[5] 张华. 基于远程通信的水处理设备数据采集系统[D]. 河北:河北农业大学, 2011.

[6] 涂瑞. 基于 ARM 的远程室内环境监测系统[D]. 湖南:湖南大学, 2014.

[7] 正点原子. STM32F4 开发指南 V1.1-库函数版本[EB/OL]. http://vdisk.weibo.com/wap/s/uEzlGhhGxtHzk. 2016-10-15.

[8] 黄建华, 宾辰忠, 欧阳宁, 等. ARM Cortex 嵌入式系统开发教程[M]. 陕西:西安电子科技大学出版社, 2012. 174~174

[9] STMicroelectronics. STM32 系列 32 位微控制器（MCU）产品选型手册[EB/OL]. http://www.stmcu.com.cn. 2016-7.

[10] STMicroelectronics. STM32F030x4 STM32F030x6 STM32F030x8[EB/OL]. m.alldatasheet.com/datasheet-pdf/pdf/524580/STMICROELECTRONICS/STM32F030C8.html. 2013.

[11] Joseph Yiu. ARM Cortex-M0 权威指南[M]. 北京:清华大学出版社, 2013. 1~1.

[12] Sipex Corporation. SP3481/SP3485[EB/OL]. http://www.alldatasheet.com/datasheet-pdf/pdf/45930/SIPEX/SP3485EN.html. 2000.

[13] Advanced Monolithic Systems, Inc. AMS1117[EB/OL]. http://www.alldatasheet.com/datasheet-pdf/pdf/49118/ADMOS/AMS1117.html.

[14] 张俊谟. 单片机中级教程[M]. 北京:北京航空航天大学出版社, 2006. 246~246

[15] STMicroelectronics. STM32CubeF0GettingStarted[EB/OL]. http://www.st.com/en/embedded-software/stm32cubef0.html. 2016-10-27

致　谢

这次的毕业设计是在我的导师张建义老师的指导下完成的，我首先要感谢的就是我的导师张建义老师。在设计过程中，自己挖了一个又一个大坑，若不是张老师不时地拉我一把，可能现在我还在坑里。大家都是做技术的，就不客套了，我由衷地感谢张建义老师对我的指导以及给我的帮助。

其次，我要感谢我在嵌入式开发方面的启蒙老师，传感器实验室的刘燕娜老师。是刘老师领着我走上了嵌入式开发这条不归路。感谢刘老师这几年来的指导，让我能在嵌入式开发方面一直走下去。

然后，我要感谢天津易实公司的吴国强老板。国强大哥是我的第一个老板，是在他的指导下，我开始开发真正的嵌入式产品而不再是实验品。在易实的这段时间也是我成长最快的一段时间。

最后，感谢所有在此次毕业设计中给予我帮助的人们。