电子测量课程设计报告

基于FPGA的智能温控风扇

组长：　　李筠颀

组员：　　梁候柱、李志聪、张海涛

摘要

本设计基于 FPGA 技术，结合 ESP32 微控制器和手机小程序交互技术，设计并实现了一款基于 PWM（脉宽调制）控制的智能温控风扇系统。该系统可对指定 4 线风扇 AUB0912VH 进行转速控制以及转速测量，可根据该温度传感器反馈的实时温度数据来实现风扇转速的自动化控制，所有的数据均可以在手机端进行动态展示，完成所有要求。

系统主要分为 FPGA 部分，ESP32 部分，手机小程序部分。FPGA 部分负责读取温度传感器数据、对风扇的控制和检测以及 rs232 串口的收发；ESP32 则负责将数据通过无线低功耗蓝牙（BLE）协议上传至手机端；手机端采用微信小程序进行数据交互，负责温度和转速数据的实时动态展示，以及用户在对风扇的远程操控，如模式切换和转速设定。整个系统分工明确，清晰协调。

我们选用 Intel 的 EP4CE10F17C8 芯片进行 FPGA 开发，verilog HDL 进行硬件描述，各功能的波形仿真全部通过；ESP32 使用 Platform IO 嵌入式开发平台进行开发，使用基于 C++的 Arduino 框架进行逻辑编写，运用 NimBLE 库进行低功耗蓝牙开发；手机小程序采用 Uniapp 跨端平台设计，运用 Ucharts 高性能跨端图标库实现数据图表动态显示。

通过本次课程设计，我们深入探讨了 FPGA、ESP32、Uniapp 小程序开发、PWM 控制、BLE 通信等技术的应用。提升了我们的硬件设计思维，掌握了物联网体系的设计逻辑。

关键词：FPGA；PWM 控制；BLE 协议；Uniapp

目录

摘要 II

一、设计要求 4

二、方案设计 4

2.1 系统总体设计 4

2.2 FPGA 系统设计 6

2.2.1 概述 6
2.2.2 串口接收模块 7
2.2.3 串口发送模块 7
2.2.4 逻辑控制模块 8
2.2.5 PWM 生成模块 9
2.2.6 温度传感器控制模块 9
2.2.7 测速模块 12

2.3 ESP32 系统设计 13

2.4 手机小程序设计 14

2.4.1 概述 14
2.4.2 BLE 通信 15
2.4.3 工作流程 16

三、测试情况 16

3.1 FPGA 仿真与综合 16

3.2 电路参数测量 17

四、总结 17

一、设计要求

设计一款有界面控制的温控风扇系统。该平台能够监测风扇的转速、监测现场使用温度的变化，控制风扇转速（风量），实施针对性散热。温度高了加快转速，温度低了降低 转速。温度恒定转速恒定。并在实验板上调试并实现所要求功能和技术指标，用示波器测 试转速以及各项指标，撰写实验报告，最后提交验收并答辩。

功能内容：

1. 掌握给定型号 AUB0912VH 的台达 4 线风扇的内部驱动控制原理，参数指标等。（10 分）

2. 人工测量并风扇最大以及最小转速时的电流电压；（10 分）

3. 采用 STM32 开发板设计 PWM 控制程序，用独立电源、开发板、手机搭建控制系统。APP 实施 PWM 控制交互。（40 分）

4. 设计测量程序，测量出风扇的转速范围。建立控制曲线，并与对应的温度传感器 建立控制策略，实施温控。（20 分）

5. 将整个控制系统及其功能移植到 FPGA 开发平台。（20 分）

二、方案设计

2.1 系统总体设计

系统主要分为 FPGA 部分，ESP32 部分，手机小程序三大部分：FPGA 部分负责读取温度传感器数据、PWM 波形的生成、转速的捕获以及 rs232 串口的收发；ESP32 作为中间层，作用是数据传输的桥梁。负责手机与 FPGA 间的远端数据通讯；手机端采用微信小程序进行数据交互，负责温度和转速数据的实时动态展示，以及用户在对风扇的远程操控，如模式切换和转速设定。整个系统分工明确，清晰协调。系统框图如下图 2-1 所示。

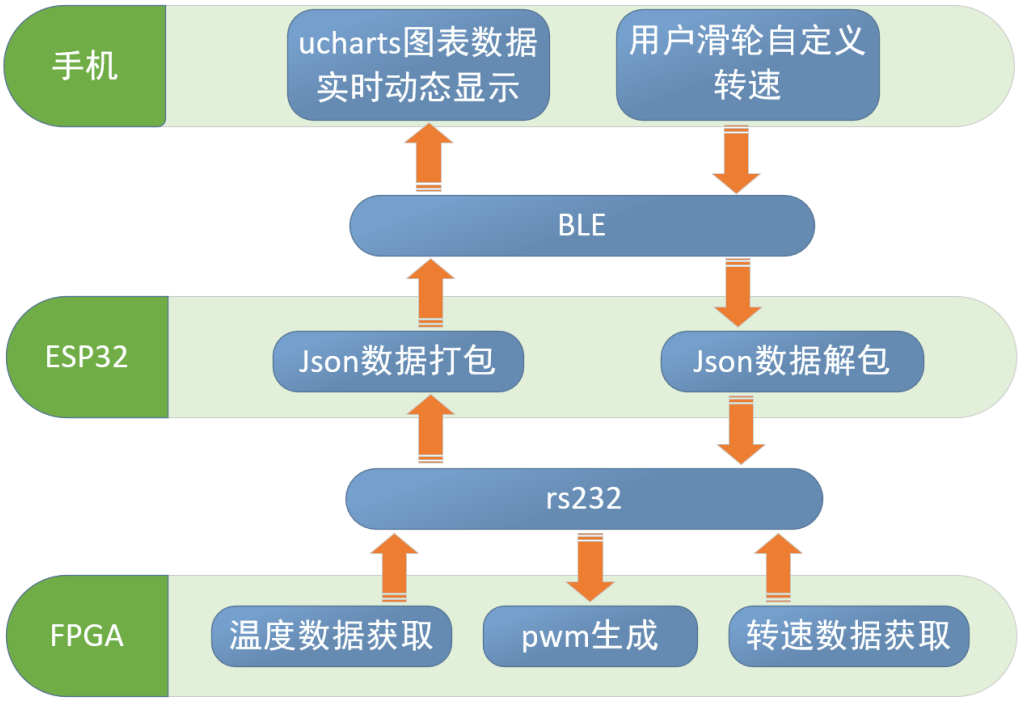

图 2-1 系统框图

欲完成该系统功能，数据通讯是关键。数据传输路径可分为监测和控制路线。监测路线是上行数据，主要包含温度和转速数据。FPGA 通过 rs232 上传温度和转速的纯数据格式，格式规定为 3 字节数据，具体对应数据位如下图 2-2 所示。数据到达 ESP32 将转化为 uniapp 方便处理的 Json 数据格式并通过蓝牙上传，最终 Json 数据在 uniapp 端进行解包，更新 ucharts 图表的数据；控制路线是下行数据，主要包含模式和速度数据，ESP32 将 Json 数据转化成 2 字节数据传给 FPGA，具体数据规定如下图所示，最高数据位 0 对应自动模式，1 则对应手动模式。

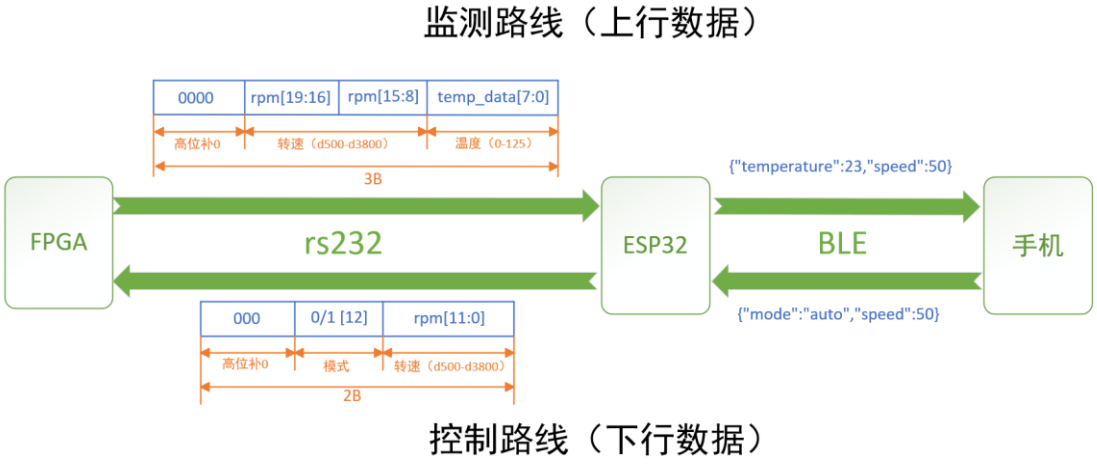

图 2-2 数据传输

2.2 FPGA 系统设计

2.2.1 概述

欲完成 FPGA 系统的设计，需要遵循：总体框图设计 —> 各子模块波形设计 —> 各子模块波形仿真 —> 顶层文件编写 —> 综合上板验证 的设计流程。在这里我们设计 6 个子模块，各子模块完成各功能。FPGA 系统框图如下图所示。系统主要运行逻辑如下：

- uart_rx 用于接收来自 ESP32 的串口数据，接收过后将提取 2 字节中的前 13 位有效数据（具体数据格式在 2.1 有具体说明）给主逻辑控制单元 logic_ctrl；
- logic_ctrl 接收来自温度传感器读取模块 ds18b20_ctrl 的温度数据以及 uart_rx 的下行数据。根据这两个数据编写温控逻辑以及模式切换逻辑，最终通过转速-占空比转换表输出占空比数据给 pwm_ctrl 模块；
- pwm_ctrl 将根据接收的占空比值实时改变输出的 pwm 波形；
- ds18b20_ctrl 控制模块将根据 ds18b20 传感器的单数据线通讯协议进行命令控制和数据读取；
- speed_acq 模块将接收来自风扇的速度反馈信号 fg，根据 AUB0912VH 的数据手册，将信号解析成转速数据。
- uart_tx 将接收温度数据以及转速数据，将两个数据打包成 3 字节的数据（具体数据格式在 2.1 有具体说明）以 1s 的时间间隔传输给 ESP32。

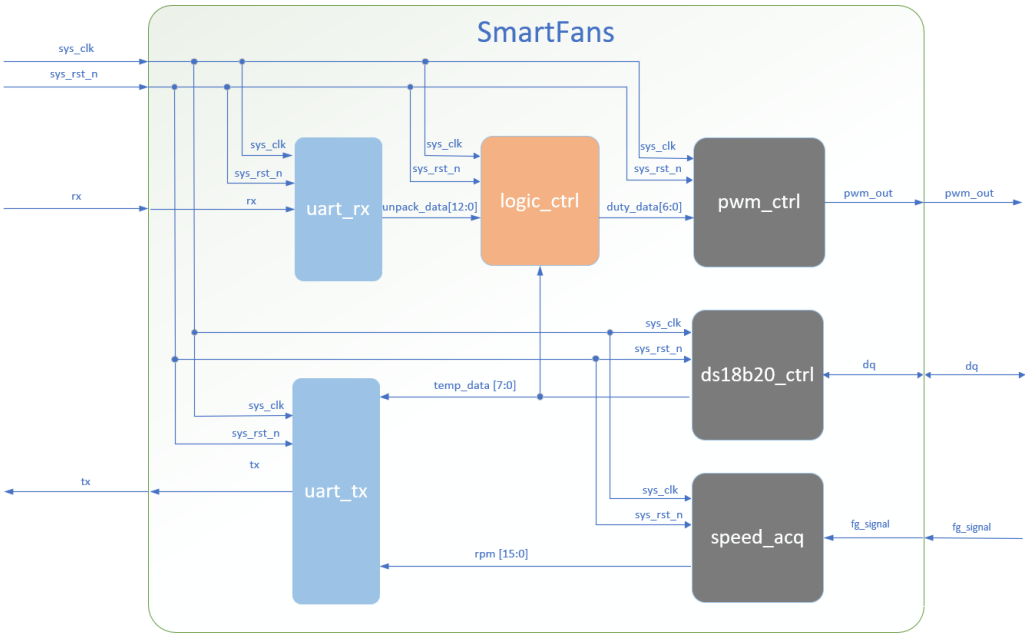

图 2-3 FPGA 系统框图

2.2.2 串口接收模块

该模块是一个基于 FPGA 的串口接收模块，主要用于接收并解包串口数据。模块通过三级寄存器（rx_reg1、rx_reg2、rx_reg3）对输入信号 rx 进行同步处理，消除亚稳态问题，并检测串口数据的起始位（下降沿）。当检测到起始位时，start_nedge 信号产生一个高电平脉冲，启动数据接收过程。模块通过波特率计数器 baud_cnt 按照设定的波特率（115200）进行数据采样，bit_flag 信号在波特率周期的中间时刻拉高，表示此时采样的数据最稳定。接收到的数据通过移位操作存储在 rx_data 寄存器中，bit_cnt 计数器记录接收到的数据位数。模块支持接收两字节数据，byte_cnt 用于计数接收的字节数。当两字节数据接收完成后，output_flag 信号拉高，触发 unpack_data 输出解包后的数据。整个过程确保了数据的准确接收和稳定输出。该模块波形设计图如下图 2-4 所示。

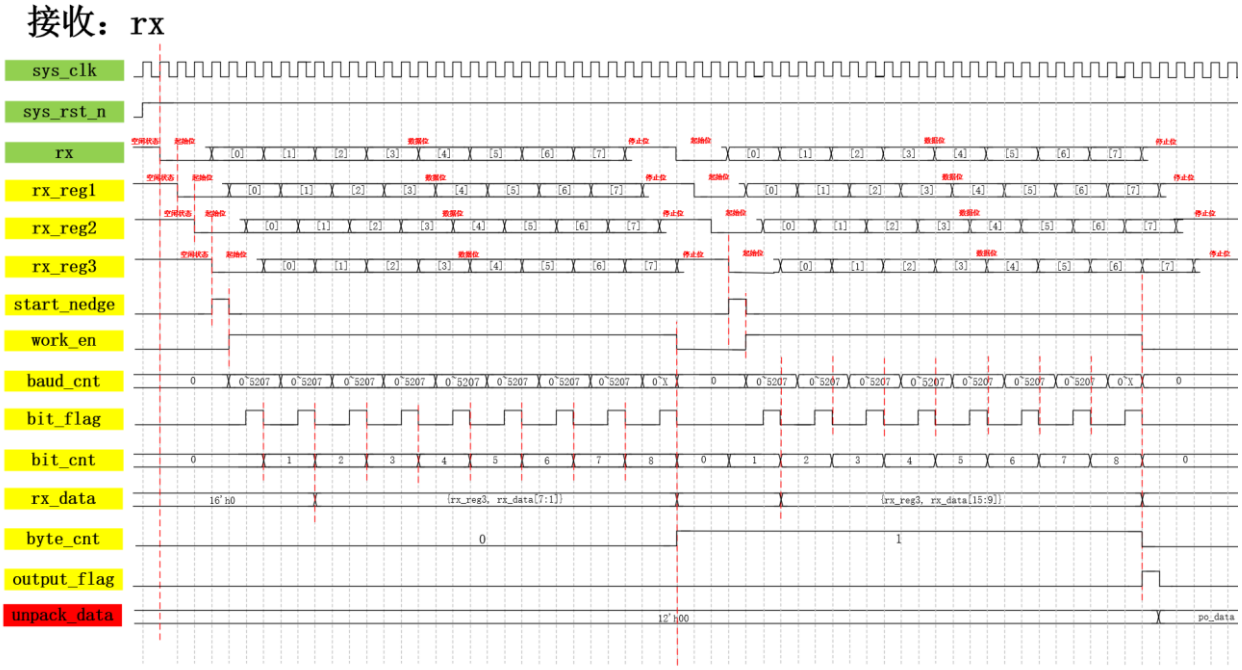

图 2-4 串口接收波形设计图

2.2.3 串口发送模块

该模块是一个基于 FPGA 的串口发送模块，用于将温度和转速数据通过串口发送出去。模块的工作流程如下：首先，系统时钟 sys_clk 和复位信号 sys_rst_n 初始化模块状态。当状态机处于 START 状态时，send_flag 信号拉高，表示开始发送数据。模块将输入的 8 位

温度数据 temp_data 和 16 位转速数据 rpm 打包成 24 位的 send_data。状态机进入 SEND_BYTE 状态后，波特率计数器 baud_cnt 开始计数，当计数到 1 时，bit_flag 信号拉高，表示可以进行数据位的发送。bit_cnt 计数器用于记录当前发送的比特数，依次发送起始位、数据位和停止位。每个字节发送完成后，状态机进入 WAIT_1s 状态，等待 1 秒后再返回 START 状态，准备下一次发送。最终，数据通过 tx 信号按照 RS232 协议逐位输出，确保数据的正确传输。整个过程通过状态机和计数器实现了数据的定时发送和协议控制。该模块波形设计图如下图 2-5 所示。

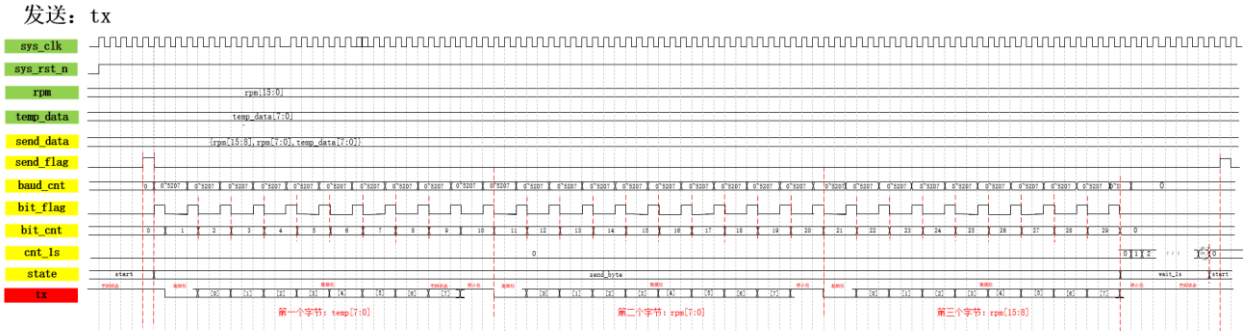

图 2-5 串口发送波形设计图

2.2.4 逻辑控制模块

用于根据输入的速度和温度数据生成 PWM 占空比信号，控制风扇的转速。模块的输入包括系统时钟 sys_clk、复位信号 sys_rst_n、12 位的速度数据 speed（来自 unpack_data 的低 12 位）、1 位的模式信号 mode（来自 unpack_data 的最高位）以及 8 位的温度数据 temp_data。模块的输出为 7 位的占空比数据 duty_data，用于控制 PWM 信号的占空比。

模块的工作流程如下：当系统复位时，duty_data 被初始化为 0。在正常工作状态下，模块根据 mode 信号选择工作模式。如果 mode 为 1，表示手动模式，模块调用 speed_to_duty 函数，将输入的速度数据 speed 映射为占空比。如果 mode 为 0，表示自动模式，模块调用 temp_to_duty 函数，根据输入的温度数据 temp_data 生成占空比。

在手动模式下，速度到占空比的映射分为三个区间：500-1000 RPM、1000-1950 RPM 和 1950-3800 RPM，每个区间有不同的转换系数（Per_Duty_500_1000、Per_Duty_1000_1950、Per_Duty_1950_3800），确保速度与占空比的线性映射，具体参数如下表所示。在自动模式下，温度到占空比的映射根据温度范围（0-125°C）分为多个区间，温度越高，占空比越大，风扇转速越快，最高占空比为 100（对应最大转速 3800 RPM）。

表 2-1 占空比转速映射表

	500~1000 RPM	1000~1950 RPM	1950~3800 RPM
占空比范围	0~20 %	20~50 %	50~100 %
转换系数（RPM/%）	25	32	37

2.2.5 PWM 生成模块

该模块是一个 PWM 控制模块，用于生成 25kHz 的 PWM 信号，控制风扇的转速。模块的输入包括系统时钟 sys_clk、复位信号 sys_rst_n 以及 7 位的占空比数据 duty_data（范围 0-100）。模块的输出为 PWM 信号 pwm_out，用于控制风扇的转速。

模块的工作流程如下：首先，duty_data 通过计算转换为对应的计数值 duty_count，公式为 duty_count = (duty_data * 2000) / 100，其中 2000 是 PWM 计数器的最大值，对应 25kHz 的频率。接着，pwm_counter 计数器在每个时钟周期递增，直到达到 1999 后复位为 0，实现 25kHz 的 PWM 频率。

在 PWM 信号生成过程中，当 pwm_counter 的值小于 duty_count 时，pwm_out 输出高电平；否则输出低电平。通过这种方式，模块根据输入的占空比数据 duty_data 生成相应的 PWM 信号，控制风扇的转速。复位时，pwm_counter 和 pwm_out 均被清零，确保系统从初始状态开始工作。该模块实现了精确的 PWM 控制，能够根据占空比数据动态调整风扇转速。该模块波形设计图如下图 2-6 所示。

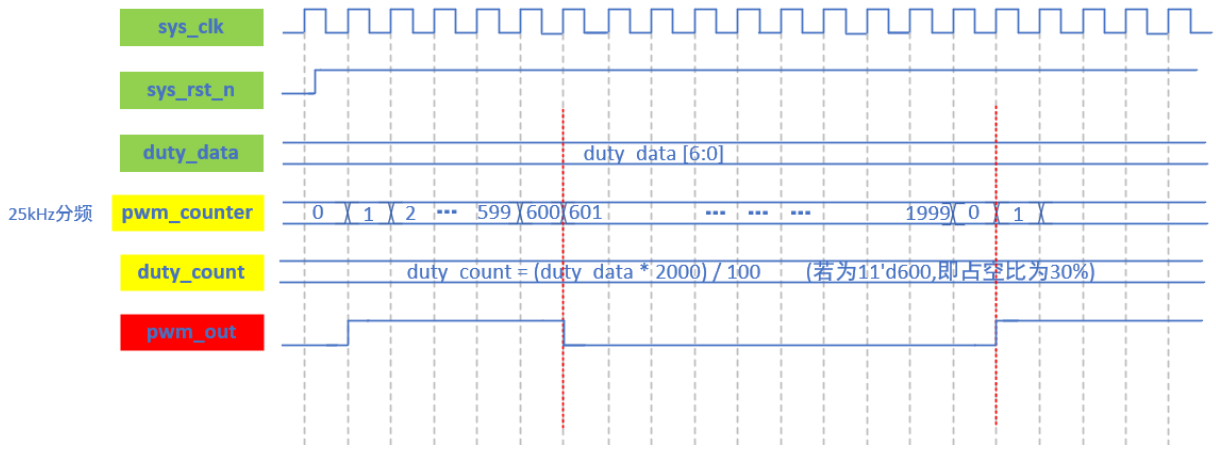

图 2-6 pwm 生成模块波形设计图

2.2.6 温度传感器控制模块

首先，模块使用 50MHz 的系统时钟 sys_clk 和低电平有效的复位信号 sys_rst_n 进行初

始化。复位时，所有寄存器和状态机被清零，确保系统从初始状态开始工作。通过分频计数器 cnt 生成 1us 的时钟信号 clk_1us，用于精确控制 DS18B20 的通信时序。

模块采用状态机控制 DS18B20 的通信流程，状态包括初始化（S_INIT）、写命令（S_WR_CMD）、等待温度转换（S_WAIT）、再次初始化（S_INIT_AGAIN）、读命令（S_RD_CMD）和读温度（S_RD_TEMP）。状态机根据 cnt_1us 计数器和 bit_cnt 位计数器进行状态跳转，确保每个状态的时序符合 DS18B20 的通信协议。

在初始化状态（S_INIT 和 S_INIT_AGAIN），模块发送至少 480us 的低电平复位脉冲，并检测 DS18B20 的存在脉冲（低电平响应）。如果检测到存在脉冲，状态机跳转到下一个状态。在写命令状态（S_WR_CMD），模块发送跳过 ROM 命令（0xCC）和温度转换命令（0x44），启动温度转换。温度转换完成后，状态机进入等待状态（S_WAIT），等待 750ms 以确保温度转换完成。

在再次初始化后，模块发送跳过 ROM 命令（0xCC）和读温度命令（0xBE），进入读温度状态（S_RD_TEMP）。在该状态，模块逐位读取 DS18B20 的温度数据，并将其存储在 data_tmp 寄存器中。读取的温度数据为 16 位，其中高 5 位为符号位，低 11 位为温度值。模块根据符号位判断温度的正负，并将温度数据转换为整数形式存储在 data 寄存器中。最终，温度数据通过 temp_data 输出，单位为摄氏度。

模块通过 dq_en 和 dq_out 信号控制双向数据总线 dq。在写操作时，dq_en 为高电平，dq_out 输出数据；在读操作时，dq_en 为低电平，dq 为高阻态，模块从 DS18B20 读取数据。温度传感器控制模块状态机状态转移图具体如下。

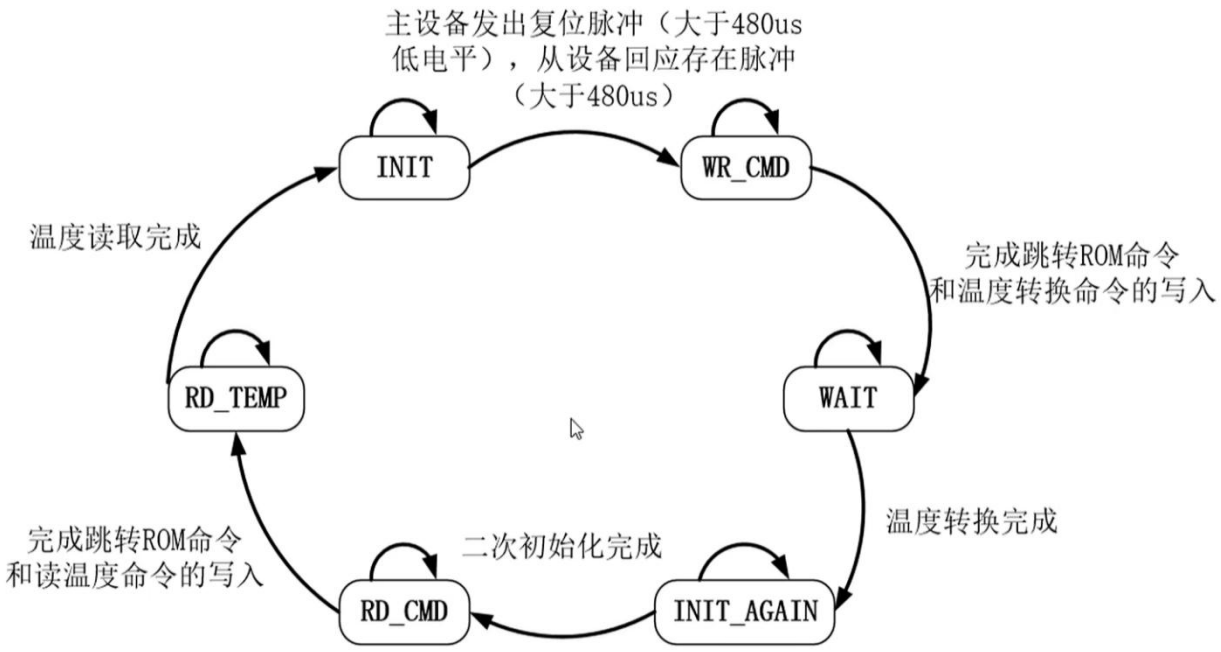

图 2-7 状态转移图

每种状态对应的波形设计图如下。

初始化、写指令

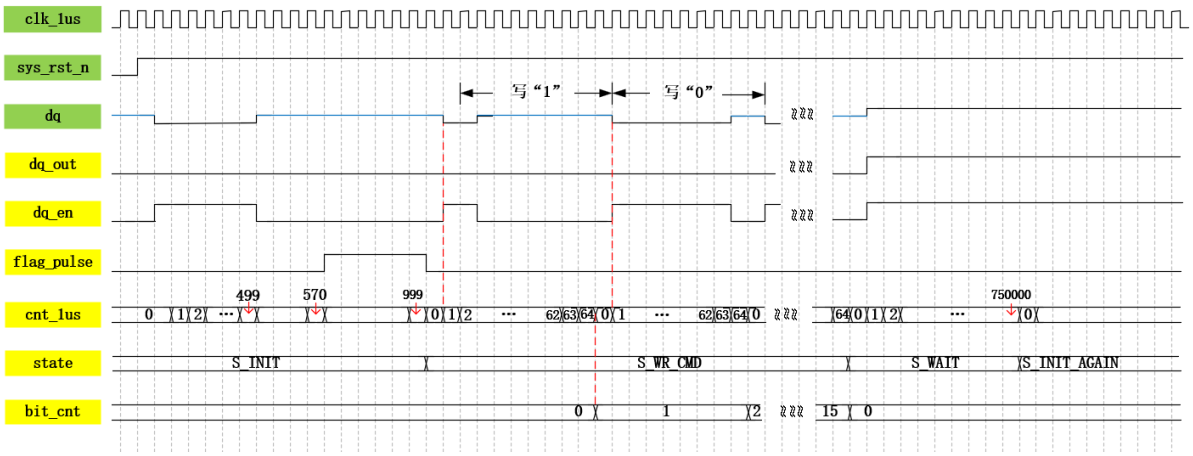

图 2-8 初始化、写指令波形设计图

再初始化、读命令

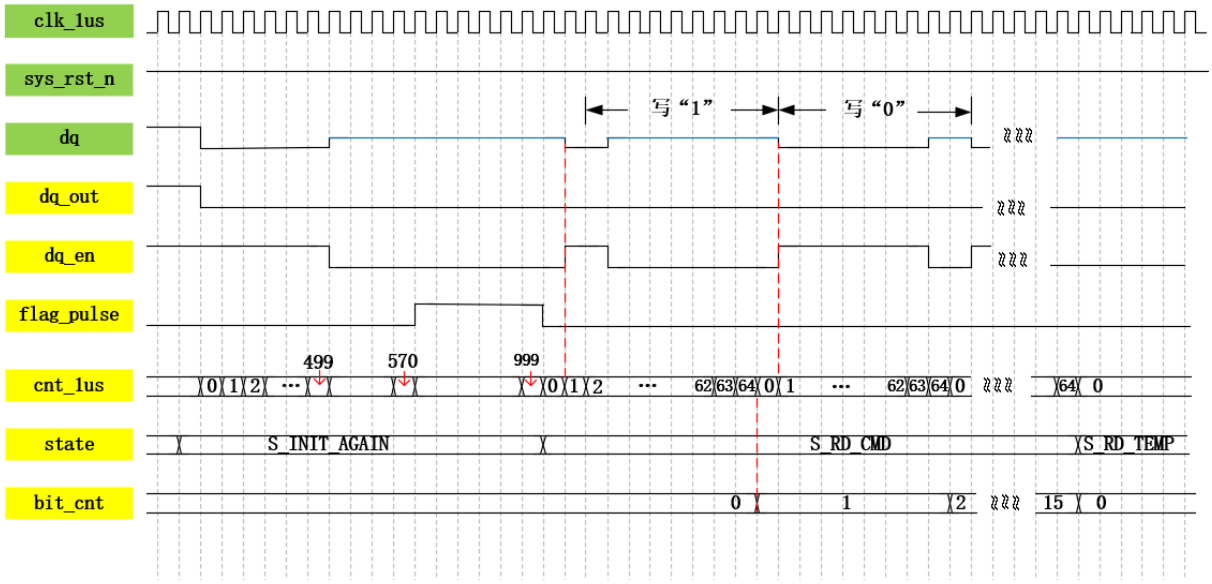

图 2-9 再初始化、读命令波形设计图

输出数据

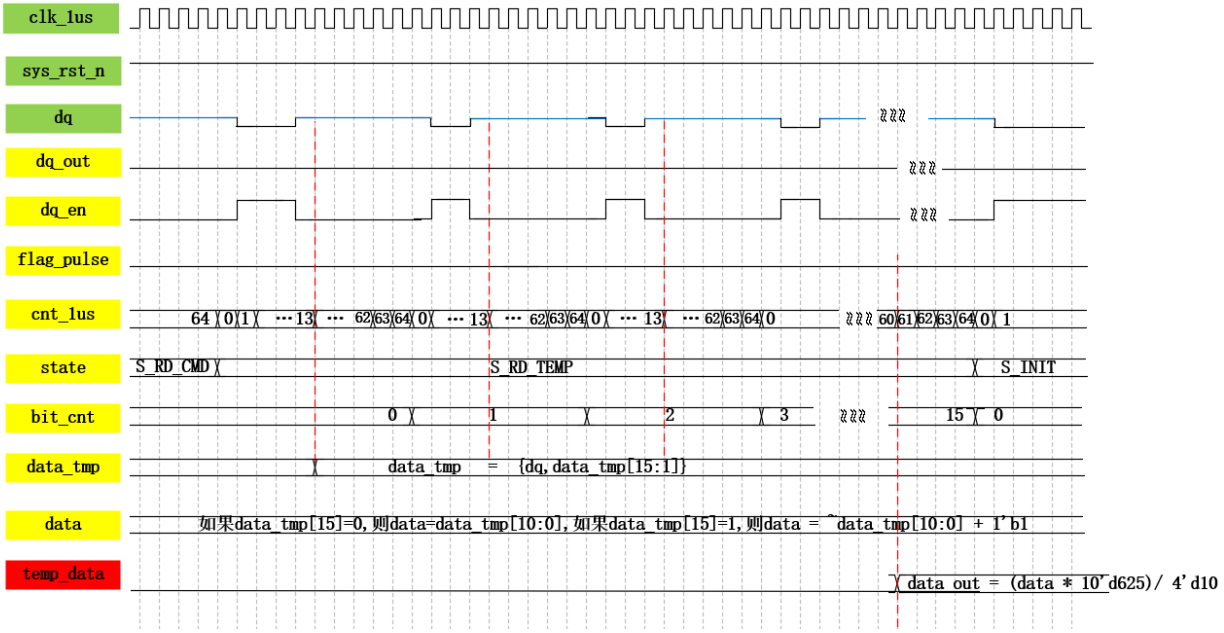

图 2-10 输出数据波形设计图

2.2.7 测速模块

用于测量风扇的转速并将其转换为 RPM（每分钟转数）。模块的输入包括系统时钟

sys_clk、复位信号 sys_rst_n 以及来自风扇的频率生成器信号 fg_signal。模块的输出为 16 位的转速数据 rpm。

模块的设计思路如下：首先，通过两级寄存器（fg_signal_d1 和 fg_signal_d2）对 fg_signal 进行同步处理，消除亚稳态问题，并检测 fg_signal 的上升沿。当检测到上升沿时，模块记录当前时钟计数器的值 ts_count，并清零计数器 counter，开始测量下一个周期。counter 在每个时钟周期递增，直到下一个上升沿到来。

转速的计算公式为 rpm = (60 * CLK_FREQ) / (4 * ts_count)，其中 CLK_FREQ 为系统时钟频率（50MHz），ts_count 为一个完整周期的时钟计数值。通过该公式，模块将周期计数值转换为 RPM 值，并输出到 rpm 寄存器中。复位时，所有计数器被清零，确保系统从初始状态开始工作。

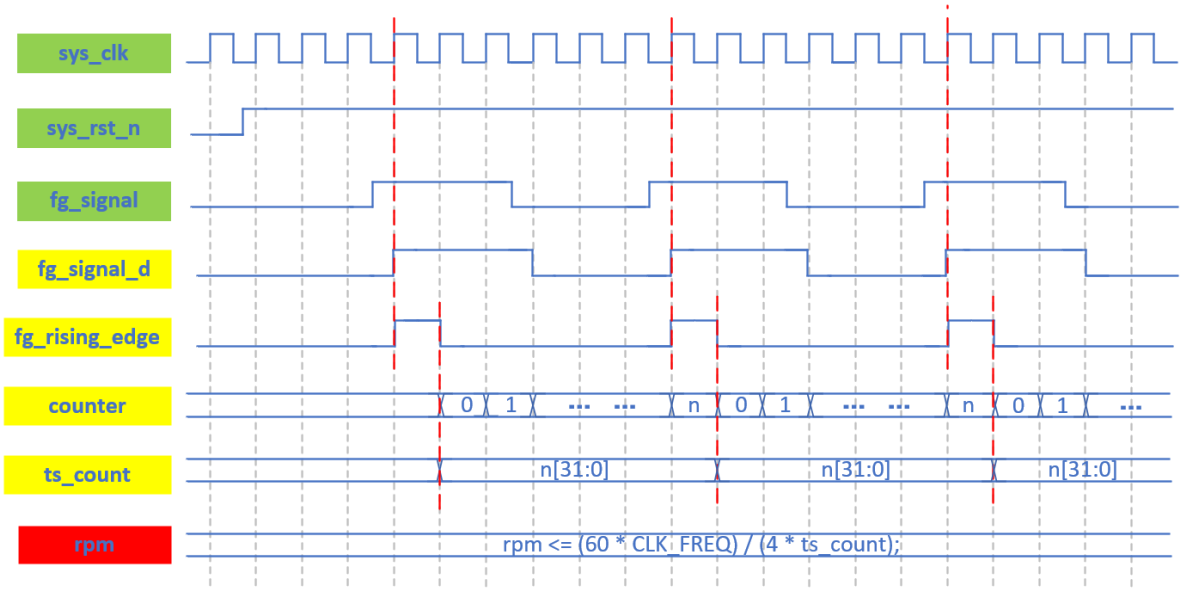

图 2-11 测速模块波形设计

2.3 ESP32 系统设计

用于实现智能风扇系统的数据传输与控制。模块的主要功能是通过 BLE 与手机端进行通信，同时通过串口与 FPGA 进行数据交互。以下是代码的主要功能描述：

- BLE 初始化与广播：使用 NimBLEDevice 库初始化 BLE 设备，设置设备名称为 SmartFan，并创建一个 BLE 服务器。定义一个 128 位的 UUID 服务，并创建两个特征值：一个用于接收手机端发送的数据（pCharacteristicRX），另一个用于向手机端发送数据（pCharacteristicTX）。启动 BLE 服务并开始广播，等待手机端连接。

- 串口数据处理：通过 processSerialData 函数从串口读取 FPGA 发送的数据。数据格式为 3 字节，第 1 字节为温度数据，第 2 和第 3 字节组合为风扇转速数据。将读取的温度和转速数据打包为 JSON 格式，并通过 BLE 特征值 pCharacteristicTX 发送到手机端，同时触发通知（notify），以便手机端实时更新数据。
- BLE 数据处理：通过 processBLEData 函数接收手机端通过 BLE 发送的 JSON 数据。数据包含风扇的工作模式（自动或手动）和设定的转速。将接收到的模式（mode）和转速（speed）数据解包，并根据模式设置数据的第 13 位（手动模式为 1，自动模式为 0），同时将转速数据打包为 16 位格式。如果数据发生变化，则将打包后的数据通过串口发送到 FPGA，用于控制风扇的转速和工作模式。
- 主循环：在 loop 函数中，循环调用 processSerialData 和 processBLEData 函数，分别处理串口和 BLE 数据，确保数据的实时传输和处理。

2.4 手机小程序设计

2.4.1 概述

手机小程序基于 Uniapp 框架开发，采用 JavaScript 作为核心编程语言，结合 Vue.js 的响应式与组件化特性实现跨端兼容性，最终项目部署于微信小程序端。UI 设计采用 Uniapp 原生组件（button、slider），并通过 HTML/CSS 进行页面布局与样式设计；数据可视化通过高性能跨端图表库 Ucharts 实现，其基于 Canvas 的高性能渲染保障了图表的流畅性。我们使用官方组件的圆弧进度条（arcbar）和仪表盘（gauge）进行开发，以动态展示温度与转速；蓝牙通信则依赖 Uniapp 的 BLE API，支持设备搜索、连接及数据收发，做到对硬件的实时数据收发。主页以及蓝牙设备连接界面如图 2-12 和图 2-13 所示。

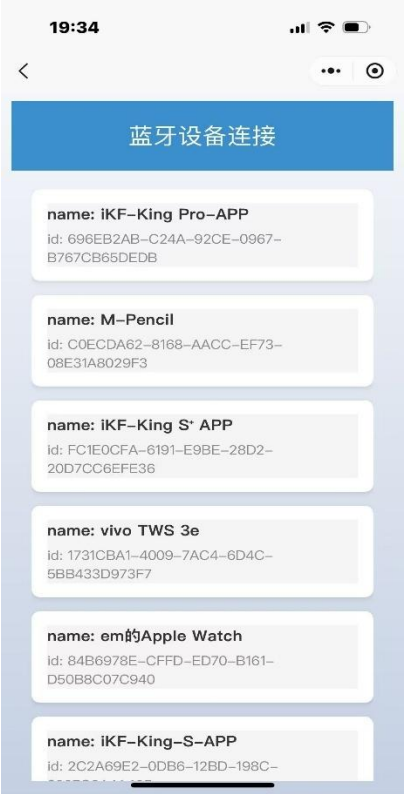

图 2-12 蓝牙设备连接页面

图 2-13 主页

2.4.2 BLE 通信

BLE 是一种低功耗的蓝牙技术，适用于需要长时间运行的设备，如智能家居、健康监测设备等。Uniapp 的 BLE API 封装了跨平台的蓝牙操作，开发者可以方便地在 iOS 和 Android 平台上实现蓝牙功能。具体工作流程如下：

- 初始化蓝牙适配器：使用 uni.openBluetoothAdapter 初始化蓝牙模块。
- 扫描设备：使用 uni.startBluetoothDevicesDiscovery 扫描附近的 BLE 设备，并通过回调函数获取设备列表。
- 连接设备：使用 uni.createBLEConnection 连接目标设备。
- 发现服务与特征值：连接成功后，使用 uni.getBLEDeviceServices 和 uni.getBLEDeviceCharacteristics 获取设备的服务和特征值。
- 读写数据：通过 uni.readBLECharacteristicValue 和 uni.writeBLECharacteristicValue 进行数据的读取和写入。
- 监听通知：使用 uni.notifyBLECharacteristicValueChange 监听特征值的通知，实时

接收设备发送的数据。

2.4.3 工作流程

用户首先进入主页，此时未连接设别状态时数据为 0。点击右上角的“+”进入蓝牙设备连接界面。此时手机会扫描附近信号强的蓝牙设备并显示在页面上。当点击设备后手机会自动连接 ESP32，返回主页就能看到硬件上传的数据了。自动模式下，转速是受温度控制。按下按钮切换为手动模式，页面会自动弹出一个滑轮，最小值值为 500RPM，最大值为 3800RPM，用户可自定义调节滑轮，以设置对应的风扇转速。

三、测试情况

3.1 FPGA 仿真与综合

经过不断的调试，功能实现成功，仿真与实际上板测试均可完成指定要求，综合结果与设计框图完全一致。下图 3-1 为 Quartus 工具综合后的 rtl 视图。由图可见于设计图 2-1 完全一致，综合成功。系统仿真图如图 3-2 所示。

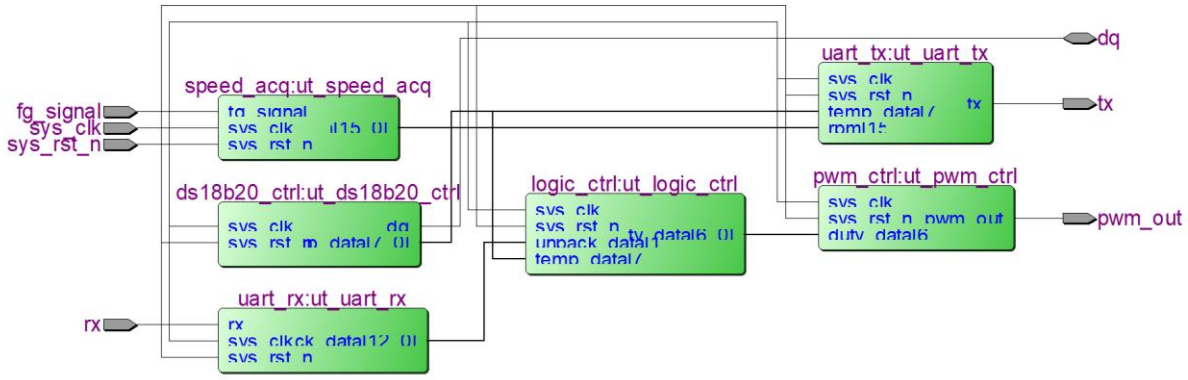

图 3-1 Quartus 综合 rtl 视图

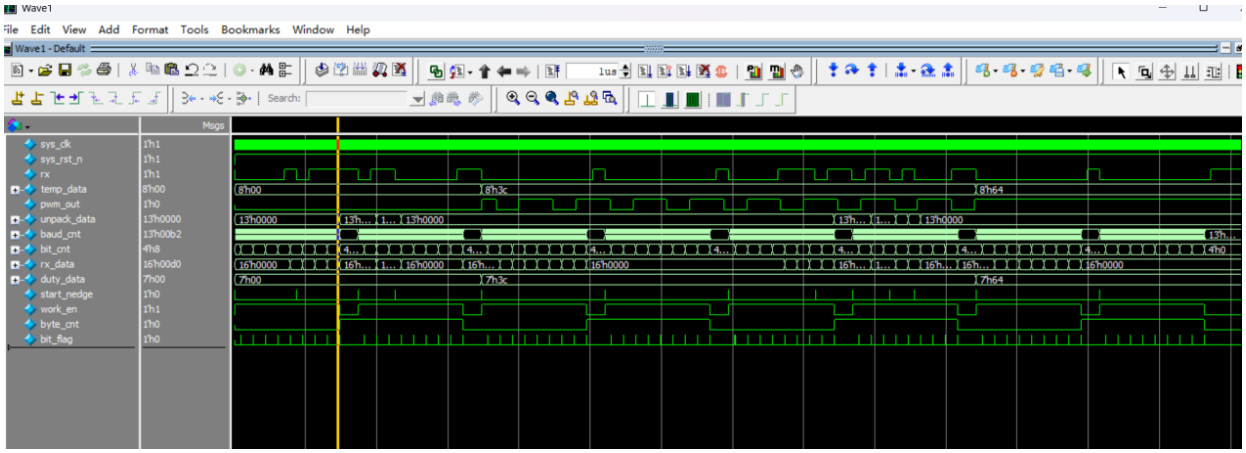

图 3-2 波形仿真图

电路由系统时钟和复位信号控制；pwm_out 信号输出等占空比的 PWM 矩形波，说明成功输出给定占空比的 PWM 信号；temp_data 信号输出温度数据；unpack_data 信号代表数据包的数据；由以上波形仿真图可以看出，系统输出波形 pwm_out 的占空比可以根据我的模拟输入的两组 rx 数据以及 temp_data 来实时改变占空比。仿真成功。

3.2 电路参数测量

测试风扇最大转速以及最小转速对应的电压电流，测试结果如下：

表 3-1 最大/小转速对应电压电流关系

PWM 占空比\电压电流	电压	电流
100%	12V	0.41A
0%	12V	0.02A

四、总结

通过本次基于 FPGA 的智能温控风扇系统的设计与实现，我们成功地将 FPGA、ESP32 和手机小程序技术相结合，构建了一个完整的物联网控制系统。通过模块化设计和多次测试，系统在实时性、稳定性和扩展性方面表现良好。本次设计不仅提升了我们的硬件设计能力和物联网系统开发经验，还增强了团队协作与项目管理能力，收获颇多。