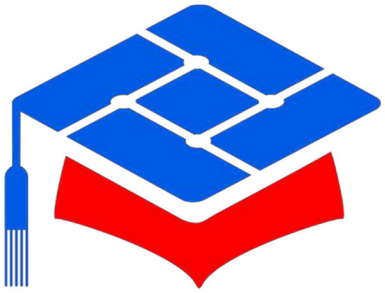

第四届

全国大学生集成电路创新创业大赛

CICIEC

项目技术文档

参赛题目：基于无剑100 开源平台软硬件结合实现电机控制

队伍编号：ASH067476

团队名称：智信同德

目录

济小台

基于无剑 100 SoC 的多功能自校准电动云台

1 背景

云台是安装、固定摄像机的支撑设备，已在多种实际应用场景下得到应用[1]，例如监控、安防、测控、测绘、机器视觉等，按其安装方式可分为固定云台和电动云台两种。固定云台安装好摄像机后调整摄像机的水平和俯仰的角度，达到最好的工作姿态后锁定调整机构，适用于监视范围不大的情况。电动云台在控制信号的作用下，云台变化运动状态以扩大所安装摄像机的监视范围，更适用于对大范围进行扫描监视。电动云台上的摄像机既可自动扫描监视区域，也可在人为操纵下跟踪监视对象。电动云台的主要性能指标有：转动速度、转动角度、带载能力（载重量）、应用环境等。电动云台目前根据不同的应用环境，使用的驱动部件类型主要有，舵机、直流无刷电机、步进电机、交流电机等，各有优势。其中步进电机式云台相比其它方案，具有以下优点：使用简单输入脉冲即可控制、程序开发成本低、开环控制也能获得较高精度、可同时获得控制精度和低延迟、低速环境下扭矩大，带载能力强、不易损坏，维修成本低。本次大赛报告在无剑 100 开源平台上实现步进电机控制，完成电动云台运动控制功能。

2　系统简介

“济小台”云台系统适用于在固定位置安装的电动监控云台，采用 57/42 步进电机控制两个方向的转动，可以在 360 度全方位大空间中快速精确运转，适合大范围监控，进行精确快速监视、物体捕捉或者用于移动场景下的增稳控制，适用于仿生机器人的头部运动单元等多种应用场合，是一款高度灵活精准的电动云台。

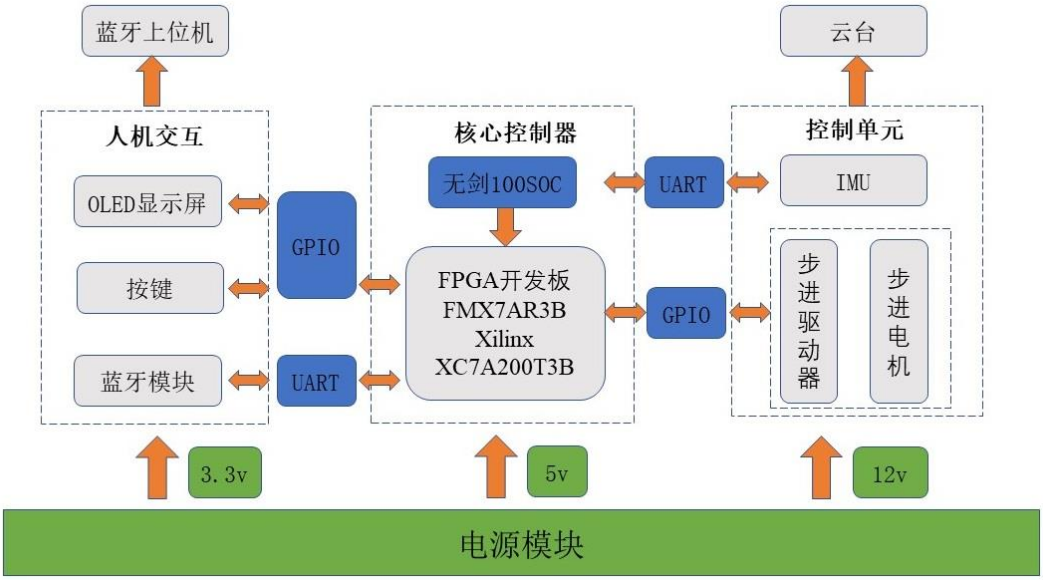

图 1 系统总体构造框图

系统总体构造如图 1 所示，主控核心为由平头哥半导体有限公司研发的以 Xilinx XC7A200T3BBIN 为核心搭载无剑 100 开源 RISC_V SOC 的现场可编程逻辑门阵列（FPGA）开发板 FMX7AR3B，系统主要功能模块包括主板供电模块、步进电机、步进驱动器、九轴惯性测量单元（IMU）、有机发光二极管（OLED）显示屏以及蓝牙通讯单元。开发完成的原型系统具有如下特点：

1）控制精度高，响应快：1.4ms/deg的运行速度，0.1^0的控制精度；

2）智能化控制：使用平头哥半导体公司的开源无剑 100 SoC 作为主控核心，协同控制，实现系统快速灵活编程，实现各种控制功能；

3）蓝牙通讯协同：通过搭载的蓝牙通讯模块，使得手机、电脑等设备实时接入控制云台姿态，监视异常情况，保留 WiFi 开发接口，后续工作中与 internet 通讯，实现远程家庭智能互联。

4）自平衡校准(角度锁定)：使用 IMU 实时获取位置信息，在用户指令执行完毕

后进入位置锁定状态，即使整个系统外界承载物发生角度的偏移，也可以保证云台上的负载能够稳定保持姿态角度。

5）集成度高：整个硬件系统集中化设计，空间资源利用度高，体积更小，方便使用和部件的更替。

3　设计方案

3.1 总体方案概述

项目总体方案如图 2 中所示。整个项目基于平头哥无剑 100 开源平台[1, 2]（wujian100_open，基于 MCU 的 SoC 平台），通过 EDA 工具进行前端仿真并制作 FPGA 进行测试。我们首先使用官方原版的 SOC 设计源码在 xilinx vivado 中综合生成比特流（bitstream）文件，再下载到 XC7A200T 系列 FPGA 中，之后使用配套的 SDK 以及图形化开发套件-C-Sky Develop Kit(CDK)进行基于此 MCU 内核的开发，完成包括云台控制算法和驱动程序在内的核心模块功能开发，以及按键/IMU/蓝牙/OLED 等主要外设的数据传输和处理程序。

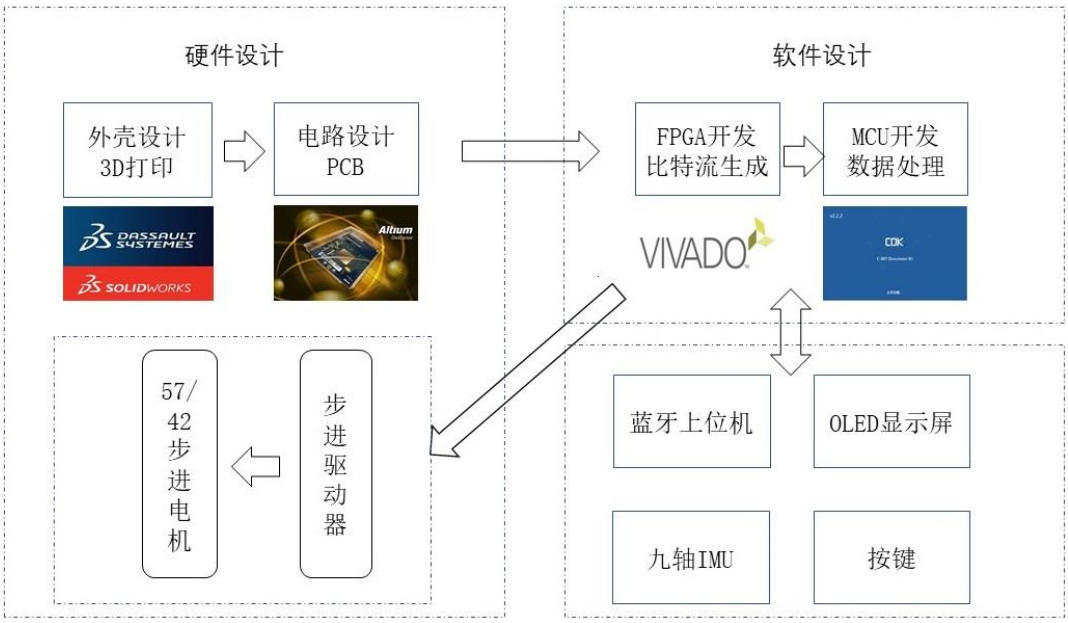

图 2 项目总体方案

所设计与实现的原型系统实物照片如图 3 中所示，多角度展示了整个系统的硬件结构。

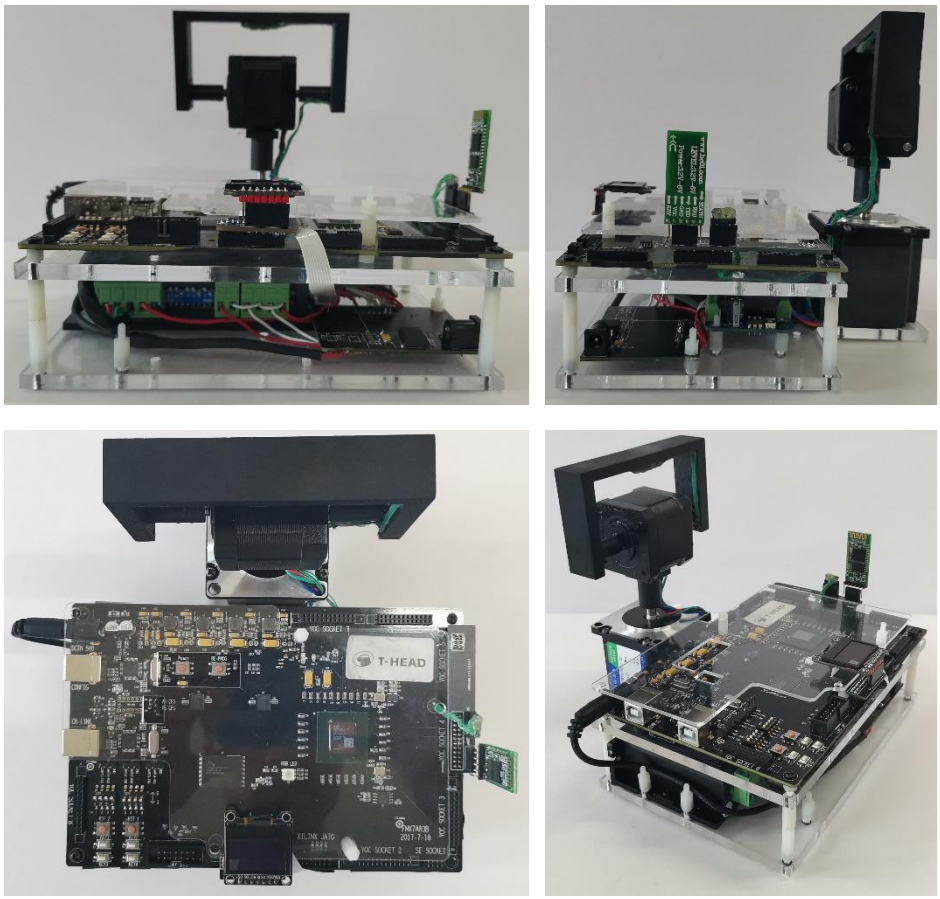

图 3 原型系统实物照片

驱动机构的硬件设计，我们选择一台 57 步进电机和一台 42 步进电机分别提供水平和竖直方向上的 360 度的自由旋转运动，并自主设计电源供应和信号传输电路，选择蓝牙模块进行无线交互，人机交互模块则利用按键和 OLED 模块进行，IMU 模块用于角度自校准，最后将整个系统结构集成一体化，整个云台在无线控制下可以较高精度和响应速度稳定运行，在系统位置发生偏移时能够及时检测到状态变化并加以控制，保证云台的位置姿态保持稳定不变。

系统程序运行流程如图 4 所示。首先连接电源，按下RE_PROG键后，FPGA 主动读取快闪存储器（FLASH）中的数据进行配置,等待 LED D3 亮起即为配置成功，再连接 USB 线通过电脑端的 CDK 进行调试。主函数初始化通用输入/输出接口（GPIO）和异步串列传输（UART）后，设置串口$UART0$和$UART1$通过中断触发读取先入先出（FIFO）的方式，按键通过外部中断触发读取，OLED 通过 GPIO 模拟串行外设接口（SPI）协议进行控制显示。串口$UART0$接收到来自蓝牙

的指令后，首先解析指令翻译成电机片选序号及其旋转方向和脉冲数量，之后在主循环进行电机控制，使用 GPIO 产生固定数量的脉冲宽度调制信号（PWM）和方向控制信号输入到步进驱动器中，达到控制目的。并通过 OLED 显示电机当前的位置，可以起到验证指令解析是否正确和记录云台位置的作用。在用户指令解析、执行完毕后保持的时间内，串口$UART1$以$100Hz$的频率接收来自 IMU 的欧拉角数据，并计算判断出当前电机是否有位置的偏移，同时进行实时调整，保持云台的稳定。

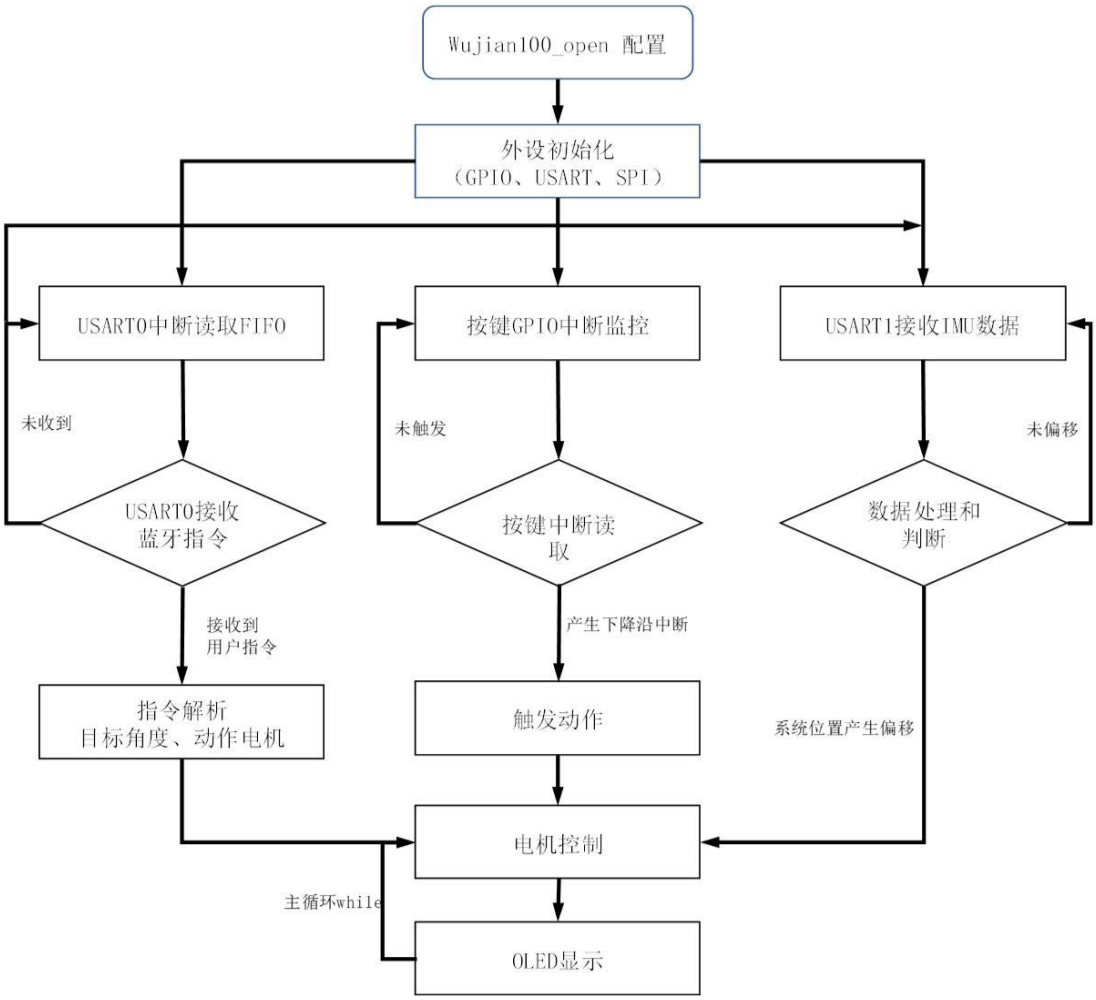

图 4 程序流程图

3.2 硬件设计

3.2.1 电源供应

整个系统可分为两部分电源，一是为步进驱动器和电机供电的直流 12V 电源，另一个是为FPGA开发板供电的直流5V电源。两部分电源共用一个$AC\sim220V$

转 $DC+12V\sim8A$的电源适配器。步进驱动器和电机由 12V 直流电源直接供电，经检测总电流在 2A 以内，满足电源适配器的功率要求。FPGA 开发板 XC7A200T3B 由直流 12V 电源经降压电路降至 5V 后供电，降压电路采用 LM1085 稳压方案，输入电压最大输出电流可达 3A，可以满足 FPGA 3A 电流供电需求。

原理图如图 5 中所示，包括 5V 稳压，3.3V 稳压，接口转接等模块。印刷电路板（PCB）如图 6 中所示，包括电源供应标准圆形插口和转接板接口。

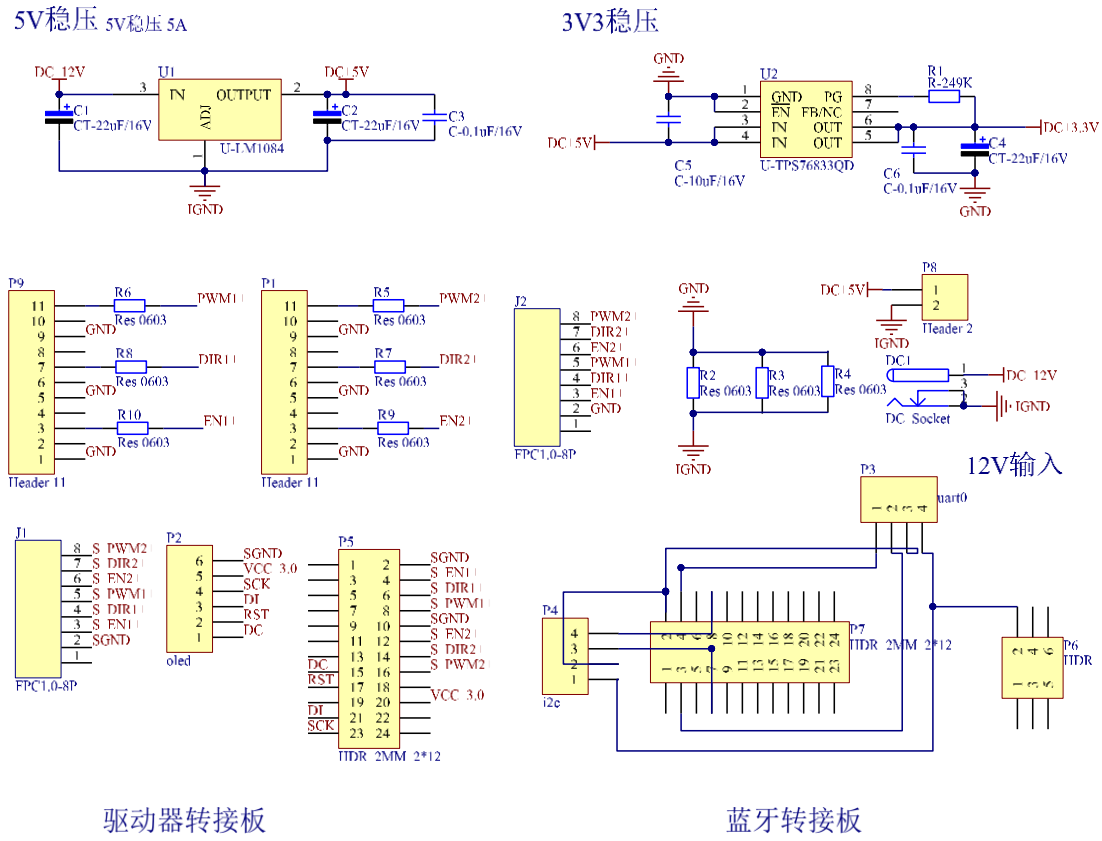

图 5 电源模块电路原理图

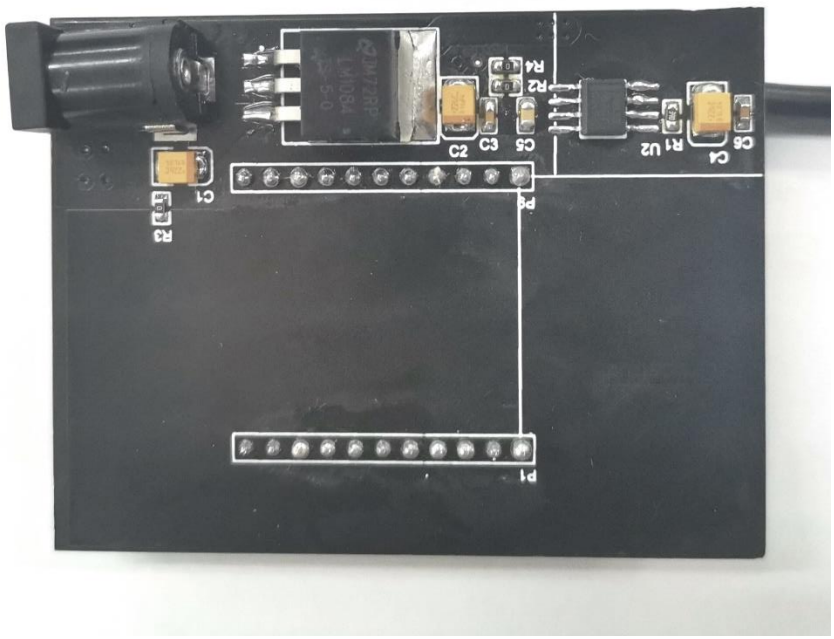

图 6 稳压供电及转接板 PCB 实物照片

3.2.2　云台驱动方案

如图 7 驱动方案系统框图所示，动力系统主要由一个普菲德 57 步进电机和一个 42 步进电机构成，其电气参数如表 1 所示。无剑 100 FPGA 的 I/O 口连接到 DM542 步进驱动器，其细分数高达 25600，即可以将电机的最小步距角缩小到 25600 倍，电流范围为$1.0\sim4.2A$，工作电压为$DC\ 9\sim42V$，分别实现两个旋转方向（水平、垂直）的自由度运动姿态控制。

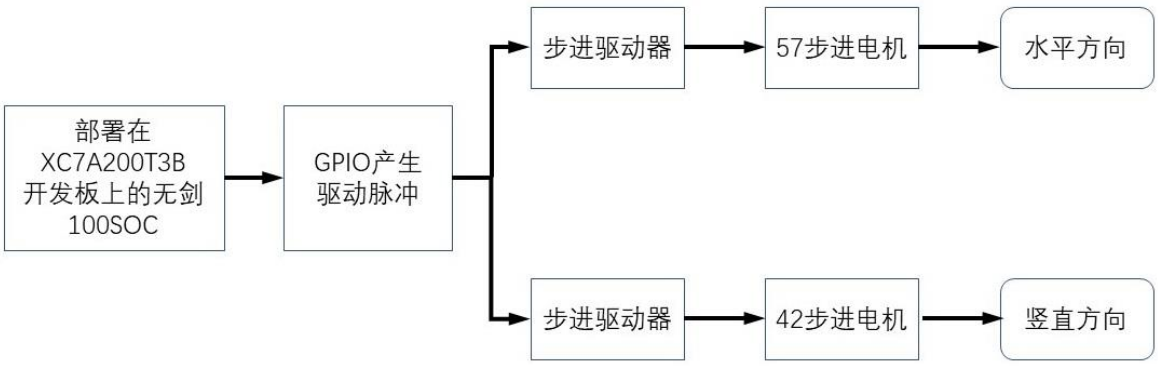

图 7 驱动方案系统框图

表 1 步进电机参数

	参数	57 步进电机	42 步进电机
1	步距角	1.8°	1.8°
2	电压	$DC12V$	$DC12V$
3	电流	MAX $3A$	MAX $1.5A$
4	电阻	$1.36 \pm 10\% \Omega$	1.9Ω
5	电感	$3.75 \pm 20\% mH$	$4mH$
6	保持力矩	$25kg \cdot m$	$0.4N \cdot m$
7	扭矩	$2.3N \cdot m$	$0.4N \cdot m$
8	绝缘等级	B	B

其中 XC7A200T3B 开发板与步进电机驱动器的连接方式为共阳极连接，如图 8 所示；步进电机驱动器与步进电机的连接方式如图 9 所示。步进使用 6400 细分，即步距角为360°/6400=0.05625°，搭配上防丢步算法，空载电机角度控制精度在水平和垂直方向实测最好效果可至0.1°。

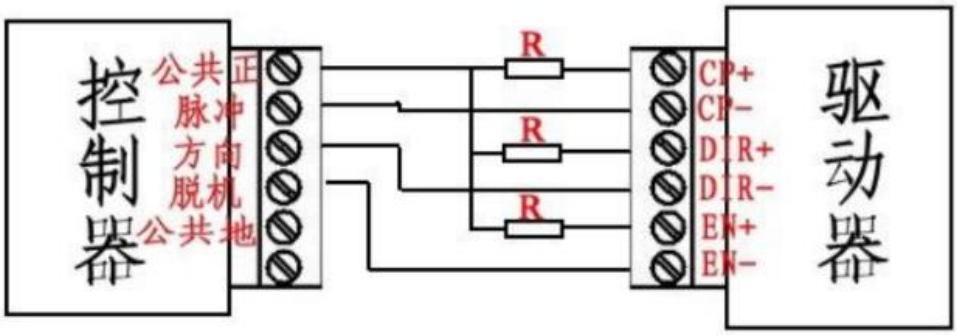

图 8 控制器和驱动器连接方式

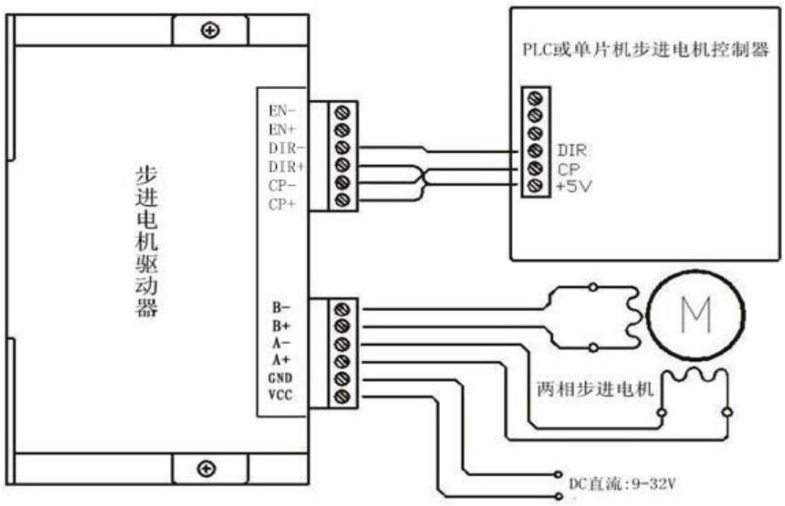

图 9 控制器、驱动器以及电机的总体连接方式

3.2.3　中央控制器和外设模块

a.中央控制器

中央控制器采用平头哥开源 wujian100 SoC，是一个基于 FPGA 的 SoC 平台，支持通过 EDA 工具进行前端仿真和 FPGA 片上测试。此 MCU 含有已经设计好的 32 个 GIPO 口、12 个 PWM 输入输出口、3 个可复用的串口/SPI/IIC 的外设资源[1]，足够满足项目需求。本设计使用中央控制器根据用户指令生成对应的固定脉冲数量的 PWM，通过安全稳定并且简约美观的 FPC 排线把步进驱动器的 PWM 信号以及方向信号输入端口和中央控制器的 PWM 信号和控制信号生成端口相连。

b.外设模块

外设模块包括开发板上的按键[3]、用于显示控制器内部运行状态的 OLED 显示模块以及用于无线通讯的蓝牙模块，如图 10 中所示。按键使用 GPIO 进行读取；OLED 使用 SPI 协议进行指令控制和数据读写；蓝牙模块通过串口与其他模块进行透传模式通信，即插即用，方便灵活。两者均通过转接板与中央控制器连接，避免使用杜邦线连接而导致的数据干扰、断连等造成传输不稳定的潜在风险。

图 10 OLED 显示屏

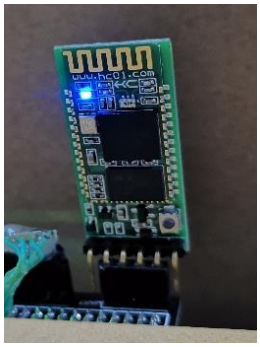

图 11 蓝牙模块

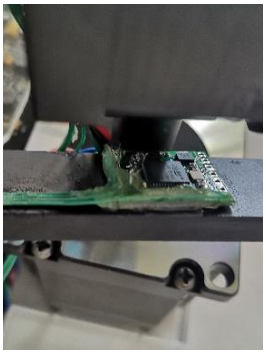

图 12 固定安装的 IMU

3.3 软件设计

3.3.1 无剑 100 SoC 在 FPGA 上的实现

a.比特流文件的生成

平头哥在 GitHub 上发布的开源 wujian100_open 项目中包括了如图 11 所示的目录树。在 windows 系统下生成比特流文件需要使用到 xilinx vivado 软件，本设计使用的版本为运行在 windows10 环境下的 vivado 2018.3 HLX 版本，需要用到目录树中 soc 与 fpga 中的文件。本团队在 Lianglonghui [4] 发布在平头哥芯片开放社区（OCC）的博文基础上，进一步完善和综合，成功生成比特流（bit）文件，并撰写英文版教程发布在 jiayi's blog 公开[5]，其流程环节简要介绍如下。

```
Directory Structure
 |--Project                //开源项目目录
 |--riscv_toolchain        //工具链
 |--wujian100_open         //
     |--case               //模拟实例
     |--doc                //用户手册
     |--fpga               //FPGA 脚本
     |--lib                //用于模拟的编译脚本
     |--regress            //回归结果
     |--sdk                //软件开发套件（sdk）
     |--soc                //SoC 的 RTL 源码
     |--tb                 //testbench
     |--tools              //模拟脚本与设定文件
     |--workdir            //模拟目录
     |--LICENSE
     |--README.md
```

图 13 目录树

首先我们将在 fpga 文件夹中的$wjian100_open_fpga_top.v$（顶层文件）与 soc 文件夹中的所有文件（设计源文件）加入源文件目录；然后添加官方提供的xdc约束文件$XC7A200T3B.xdc$；随后将$apb0_params.v$、$apb1\_params.v$、$timers\_params.v$、$wdt_params.v$四个头文件类型改为“Verilog Header”头文件，并将$wujian100_open_fpga_top.v$文件置顶，作为首先编译的文件以避免一些编译错误。

由于官方提供的$XC7A200T3B.xdc$文件中不包含时序约束条件，我们同时也需要增加图 14 中的时序约束代码至约束文件中以创建系统时钟。

```
create_clock -name {EHS} [get_ports PIN_EHS] -period 50 -waveform {0 25}
create_clock  -name {JTAG_CLK} [get_ports PAD_JTAG_TCLK] -period 1000 -waveform {0 500}

set_clock_groups -asynchronous -name {clkgroup_1} -group [get_clocks {EHS JTAG_CLK}]

set_false_path -through [get_ports PIN_EHS]

#set_clock_groups -name {Inferred_clkgroup_0} -asynchronous -group [get_clocks
{wujian100_open_top|PAD_JTAG_TCLK}]

set_property ASYNC_REG TRUE [get_cells
{x_aou_top/x_rtc0_sec_top/x_rtc_pdu_top/x_rtc_clr_sync/pclk_load_sync2_reg}]
set_property ASYNC_REG TRUE [get_cells
{x_aou_top/x_rtc0_sec_top/x_rtc_pdu_top/x_rtc_clr_sync/rtc_load_sync2_reg}]
set_property ASYNC_REG TRUE [get_cells
{x_aou_top/x_rtc0_sec_top/x_rtc_pdu_top/x_rtc_clr_sync/pclk_load_sync1_reg}]
set_property ASYNC_REG TRUE [get_cells
{x_aou_top/x_rtc0_sec_top/x_rtc_pdu_top/x_rtc_clr_sync/rtc_load_sync1_reg}]
set_property ASYNC_REG TRUE [get_cells {x_cpu_top/CPU/x_cr_had_top/A15d/A74/A10b_reg}]
set_property ASYNC_REG TRUE [get_cells {x_cpu_top/CPU/x_cr_had_top/A15d/A74/A18597_reg}]
set_property ASYNC_REG TRUE [get_cells {x_cpu_top/CPU/x_cr_had_top/A15d/A1862d/A10b_reg}]
set_property ASYNC_REG TRUE [get_cells
{x_cpu_top/CPU/x_cr_had_top/A15d/A1862d/A18597_reg}]
set_property ASYNC_REG TRUE [get_cells {x_cpu_top/CPU/x_cr_had_top/A15d/A75/A10b_reg}]
set_property ASYNC_REG TRUE [get_cells {x_cpu_top/CPU/x_cr_had_top/A15d/A75/A18597_reg}]
```

图 14 时序约束语句

随后，根据上述博客中[4]指出的问题，需要改正官方约束文件中的某一时钟管脚的变量名，由$PAD_JTAG_TCLK_c$更正为PAD_JTAG_TCLK。

最后进入综合环节生成比特流文件。实际测试结果中，在 Xlinx XC7A200T3B 开发板下的资源消耗如表 2 中所示，动态与静态功率消耗如图 15 中所示，可以观察到总体资源使用率少于 25%，动态功率低于 0.1W，总功率低于 0.2W。作为一款低功耗 SoC，无剑 100 展示出了优秀的功率特性以及低资源消耗的特性。

表 2 无剑 100 的资源消耗统计表

资源名称	使用数量	总共可用	利用率（百分比）
LUT	26744	133800	19.99
LUTRAM	72	46200	0.16
FF	13415	267600	5.01
BRAM	64	365	17.53
IO	62	285	21.75
BUFG	2	32	6.25

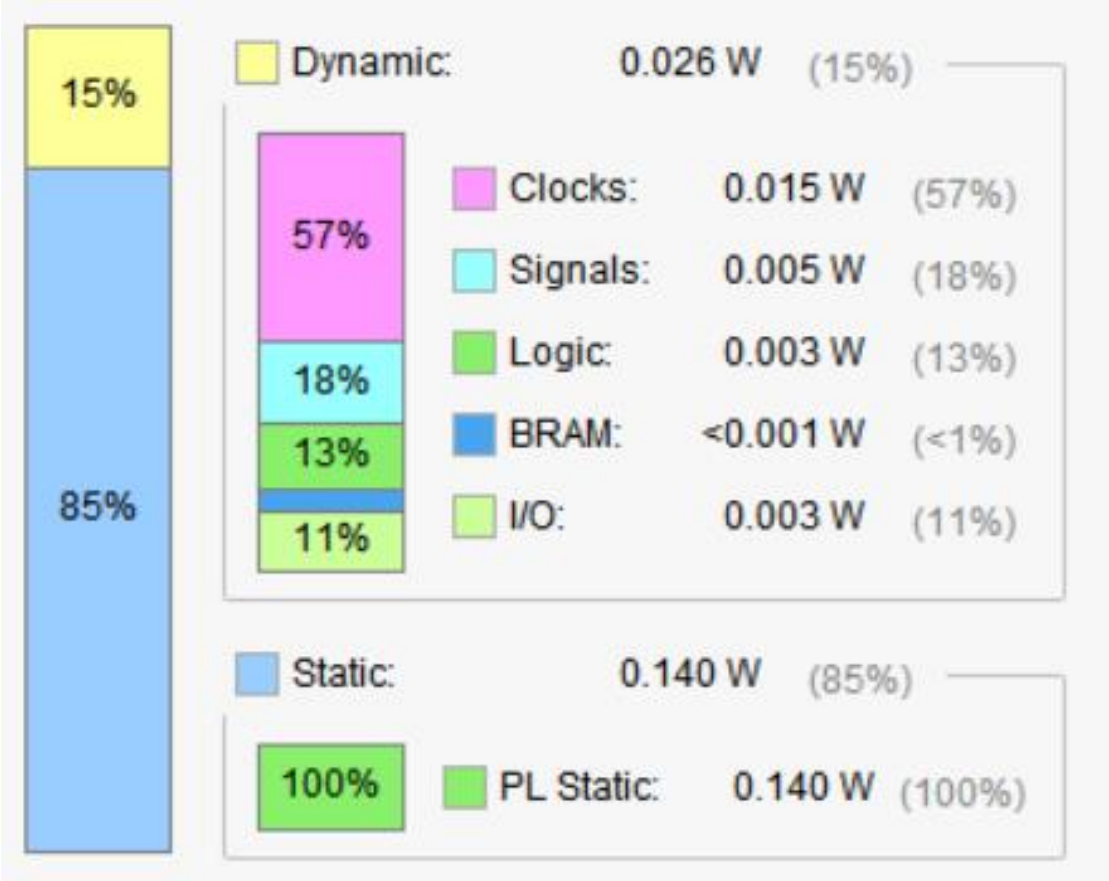

图 15 功率消耗示意图

在实现中可以看到如图 16 所示的版图设计，同样可以发现其器件布局仅占到版图中不到 25%的区域，也说明了无剑 100 的低资源消耗水平。

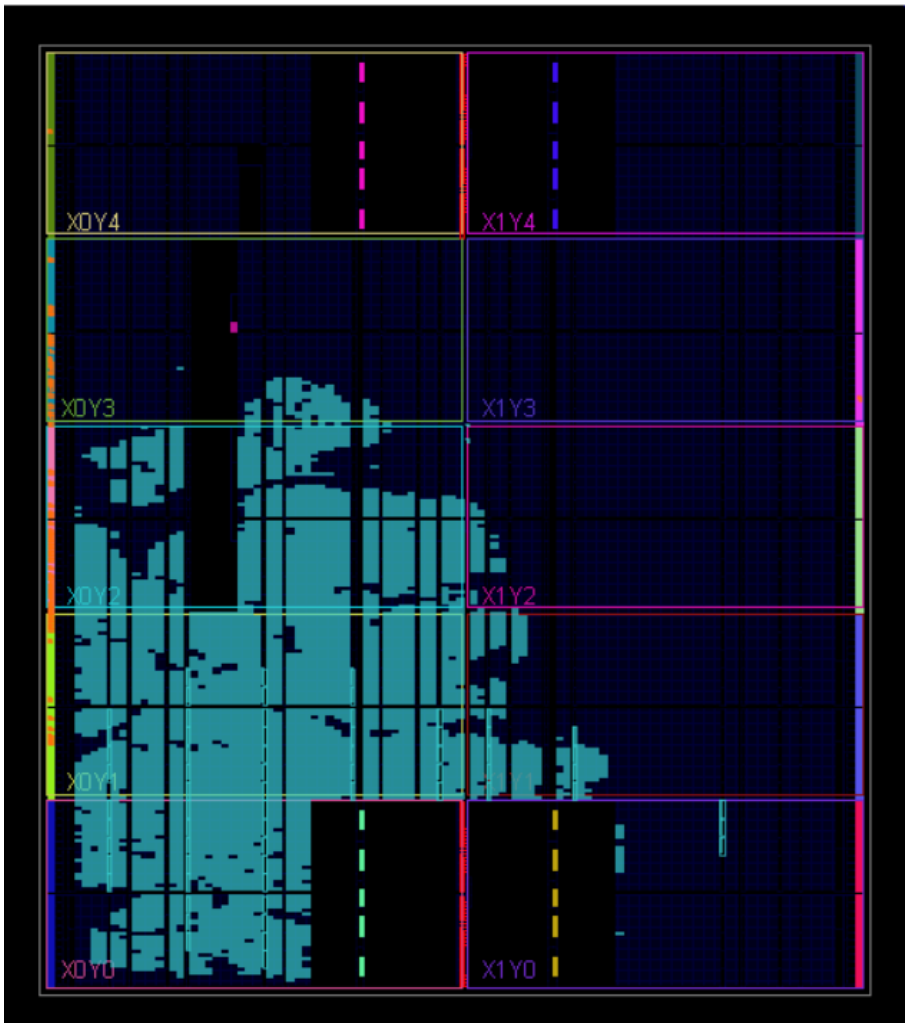

图 16 FPGA 版图

b.FPGA 配置方式设置

FPGA 的配置方式有三种：主动串行配置模式（AS），被动串行配置模式（PS），JTAG 模式，本方案中分别采用 PS 模式和 PS 模式进行比特流文件的配置。前者是利用板上外置 MCU 作为主控制器，读取存储在 SD 卡中的比特流文件，写入到 FPGA 的配置存储器中，具体做法就是把 3.3.1 中生成的比特流文件存储在 SD 卡中，并命名为$cfg.bit$,之后按照步骤上电，按下re_prog等待配置成功即可。后者利用板载的 16MB Flash 进行配置，即把比特流烧入 Flash 中，FPGA 每次上电后，作为主控制器，从 FLASH 中读取数据，进行配置。具体做法是使用 Platform USB Cable 连接开发板和电脑，通过 vivado 把 bit 文件转换为 mcs 文件，并烧写进 Flash 中，最后板上跳线选择 AS 模式，按下re_prog即可成功配置[2]。

综合两种配置方式，本团队发现，使用被动模式可以在 soc 需要修改时，直

接更改 SD 卡内的文件即可，从而可以方便快捷的进行离线程序修改，但是由于位置原因，使用此模式配置的成功率不高，需要多次re_prog才可以，甚至无效。相比之下，使用 FLASH 的 AS 模式虽然改写固件较麻烦，但是每次上电后的配置成功率极高，基本上每次re_prog后即可快速完成，无需等待。因此本方案最后选择使用 FLASH 存储的 AS 模式配置方案。

3.3.2 电机控制和状态自校准锁定

本节叙述驱动电机运转的实现过程和防止步进电机运动过程中出现丢步，增大误差的 S 曲线算法[6-9]以及在进入角度锁定状态后的自校准算法的实现。

a.电机驱动程序

步进电机的驱动信号可以分为脉冲信号和方向控制信号。其中方向控制信号使用一个 GPIO 口产生，输出高电平可使得电机向预设正方向旋转，电平反转后即可使得电机向反方向旋转。步进电机按步长方式进行运动，每一步及对应一个脉冲。此电机步距角为 1.8 度，也即在细分数为 1 时，输入 1 个脉冲会驱动电机旋转 1.8 度。设细分数为 n，则每个脉冲驱动电机旋转角度为：

$$Angle = \frac{360}{n}(n = 1,2,3,\dots) \tag{1}$$

由于驱动器最高细分数为 25600，即理想情况下，驱动精确度上限输出一个脉冲驱动驱动电机选旋转$^{360°}/_{25600} = 0.0140625°$。而脉冲的频率决定了步进电机的运行速度，相当于步进电机最终的响应速度。

设步进电机频率为f，细分数为n,则一个脉冲周期为$1/f$，步进电机旋转运动一周需要脉冲数为

$$N_{pulse} = n \tag{2}$$

则所需时间为

$$t = \frac{n}{f}s \tag{3}$$

步进电机的运动速度w为

$$w = \frac{f}{n}rps \tag{4}$$

故根据实际应用调节频率即可满足不同运行速度的需要，在本设计中细分数取 6400，即$n = 6400$，输入的 PWM 频率在恒速阶段为$f = 10kHz$,代入上式可

得步进电机运行速度为

$$w = \frac{f}{n} = \frac{10000}{6400} \approx 2rps \tag{5}$$

即每秒 2 转。

产生步进电机需要的 PWM 脉冲信号有三种方案：一是使用内置的 PWM 发生器，比如无剑 100SoC 提供共 12 个 PWM 通道，只需要配置好相应的寄存器即可产生需要的 PWM 波，可以应对如对频率和占空比等比较高且需长时间保持脉冲的情况，但是如果需要固定数量脉冲就需要进行脉冲计数，操作起来稍微复杂；二是使用定时器中断和 GPIO 电平翻转，同样有频率高的特性，但使用定时器中断产生的 PWM 波形的脉冲宽度调节起来不太方便；三是单独使用 GPIO 口电平反转和适当延时产生 PWM 波形，这种方式的优势在于脉冲数容易控制、频率与占空比即时调节，缺点在于 GPIO 的主线频率（多为$50kHz$以下）无法产生高频率的脉冲。总结三者优缺本方案选用上述的方案三，无剑 100 SoC 主频可达 $20MHz$,GPIO 的电平可以依赖高速总线实现快速反转的功能以满足步进驱动所需频率，实测最高可达$50Khz$.

b.电机控制算法（S 曲线）

步进电机是一种把脉冲信号转换为角位移的机电元件，具有控制精度高，控制简单等特点，即使是开环控制也能获得较高的控制精度。但是由于固有原因，步进电机在启动时，由于此时需要的加速电流较大，线性度不够，会出现启动慢、启动丢步的现象；同理，在停止时也会出现过冲的现象，这是影响步进电机精度的一个较大原因，所以对步进电机启动和停止时的加速度，速度作合理的规划有很大的实际意义。

综合考虑后本方案选择使用 S 曲线加减速控制算法。S 曲线算法下的电机加速度和速度变化较为平稳且收敛较快，具有较好的效果。S 曲线算法如图 14 所示，这里采用的是五段式 S 曲线[8]，即在从启动到停止整个运动过程中电机的加速度可分为 5 个线性段。也可以看出来速度变化较为平稳。

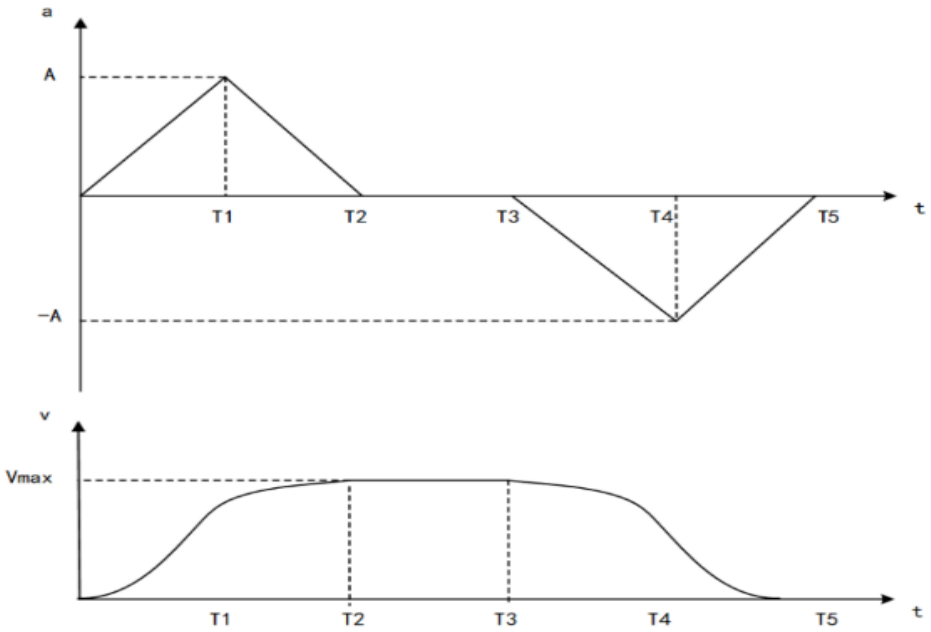

图 17 S 曲线算法示意图

设v为步进电机的稳定运动速度，也即 S 曲线中的最大速度，对应加速度为 0 的时段；最大加速度为A,最小加速度为$-A$,最大速度为V_{max},加速度线性系数为k.则整个运动过程可分为 5 段，如表 3 所示。

表 3 运动过程中加速度与速度的变化

	加速度$\boldsymbol{A(m/s^2)}$	速度$\boldsymbol{v(m/s)}$	
1	kt	$\frac{1}{2}kt^2$	(6)
2	$A-kt$	$\frac{1}{2}At_1{}^2+\frac{1}{2}(2A-kt)\cdot(t-t_1)$	(7)
3	0	$V_{max}=\frac{1}{2}At_2$	(8)
4	$-kT+kt_3$	$V_{max}-\frac{1}{2}(t-t_3)(kt_3-kt)$	(9)
5	$kt-kt5$	$V_{max}-\frac{1}{2}A(t_4-t_3)-\frac{1}{2}(kT-kt_5+A)(T-t_4)$	(10)

步进电机启动加速度的大小和脉冲频率有着直接关系，所以可以假设脉冲频率和加速度存在线性相关关系，可以使用脉冲频率f代替加速度。经过实际带载测试和迭代求解，本方案中关于 S 曲线中各参数的详细配置为：

步进驱动器选择 6400 细分，计算可得 1 个脉冲对应的步进角为

$$\frac{360}{6400} = 0.05625° \tag{11}$$

假设$F_{max} = 20kHz$,$F_{min} = 1kHz$,每个非线性加速时间段内取 10 个脉冲，以平滑加减速，即可防止丢步，又能快速响应。以$0 \sim t_1$区间为例选择 10 个频率点，范围为$F_{min} \sim F_{max}$,则脉冲频率步长为

$$\frac{F_{max}}{10} = \frac{20000}{10} = 2kHz \tag{12}$$

c.自校准锁定

自校准状态锁定流程如图 18 所示，在系统处理完用户指令后，进入角度锁定状态。在此期间，程序时刻获取系统基准面(底盘)的三维空间的角度变化信息，当角度和锁定的初始状态不一致发生改变时，根据解算得到的角度偏移量，计算出反馈给电机的旋转量，及时的控制电机向和偏移方向相反的效果运动，从而做到云台角度的自校准锁定。

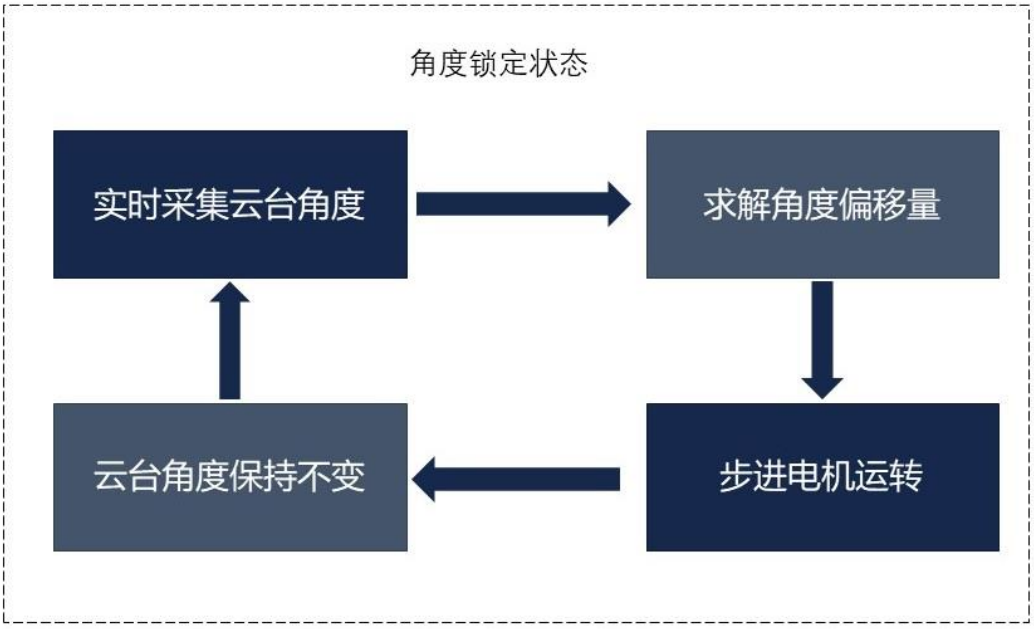

图 18 自校准锁定流程

3.3.3 蓝牙模块和 IMU 模块

本章节叙述系统运行过程中使用的蓝牙和 IMU 获取数据和处理数据的具体过程。

a.蓝牙模块通信和数据解析

本方案设计中使用了无剑 100 中的$UART0$模块通过中断触发接收方式从

FIFO 中进行读取数据和蓝牙进行通信进而实现数据的无线发送和接收[10]。上位机发送的是用 7 个字符表示的自定义指令，指令格式定义如表 4 中所示。

表 4 蓝牙指令格式

指令 （7 个字符）	意义	
M/L （首字符）	指令校验	M：活动模式 L：锁定模式
+/−	旋转方向	+：定义方向顺时针旋转 −：定义方向逆时针旋转
XXXX	脉冲数量	0~9999
1/2 （尾字符）	动作电机位置	1：竖直电机 2：水平电机

指令解析流程图如图 19 所示。当中央控制器接收到来自上位机发送的指令后首先进行校验，通过判断首位字符，确定是移动还是锁定，之后判断第二位含义是是‘+’或者‘−’。如果不符合规范则回传上位机“*error*!”，如果正确进行下一步解析。解析首先要把 4 个代表脉冲数的字符转化为对应的整数，结合第二位字符，生成一个有符号数，代表脉冲数和方向。之后解析末位字符，判断出目标电机的位置。在解析完整条指令后，回传给上位机同样的指令表明解析成功。

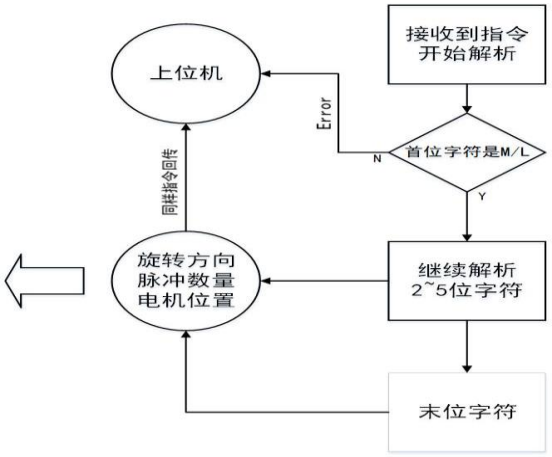

图 19 蓝牙指令解析流程图

b.IMU 模块数据的获取和处理

使用 UART1 模块采用中断触发读取 FIFO 的方式获取 IMU 的数据。IMU 九轴传感器[11]以$10Hz$的速率向$UART1$发送含有 11 个字符的数据帧，包含校验位和欧拉角数据。采用和蓝牙类似的方式读取 11 位的数据帧，然后处理计算出来欧拉角。在云台锁定模式下，实时检测角度的变化，并做出相应的反应。比如，在锁定模式开始时，初始状态云台的偏航角为a。在实时获取角度的过程中，如果检测到偏航角为b,此时则需要向电机发送动作指令，产生和角度变化量$(b-a)$对应的脉冲信号，进行角度偏移的校正。

3.3.4　OLED 显示屏（SPI 通讯）

OLED 显示屏使用 SPI 协议与主控制器进行通信作为人机交互界面，显示控制器内部变量的运行状态。无剑 100 SoC 虽然具有硬件 SPI 功能可以用来控制 OLED，但是整个系统不同硬件资源比如 GPIO、UART、SPI 等 I/O 端口的布局在本设计中无法统一。

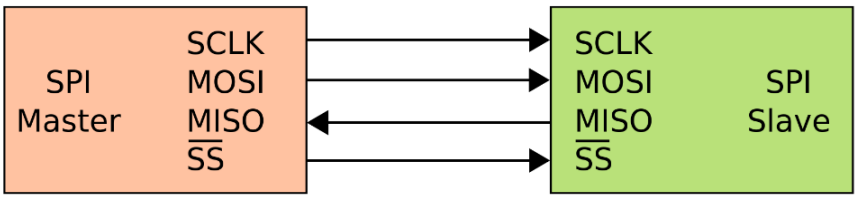

图 20 SPI 主从机示意图

为了能够更集中利用板上资源，以及根据 SPI 协议的特性，最终本设计选择了使用 GPIO 模拟 SPI 的通信方式，SPI 主从机连接示意图如图 20 所示。具体方法是根据 SPI 协议通信时时钟线和信号线的时序，用 GPIO 模拟通信端口和时钟线反转电平并进行合适的延时以模仿 SPI 的行为，可以实现使用任何 4 个通用端口即模拟串列时脉（SCLK）、主机输出从机输入信号（OSI）、主机输入从机输出信号（MISO）和片选信号（SS），即可达到 SPI 通信协议的要求；除了 SPI 通信所要求的 4 个信号线，OLED 显示屏还需要一个复位信号（Reset），使用单独的 GPIO 即可。

SPI 协议的时序图如图 21 所示,显示了不同模式下的信号时序图。本模拟 SPI 采用的是$CPOL=0$且$CPHA=0$的传输模式，即时钟相位为 0，时钟前沿数据采

样，时钟后沿数据输出。

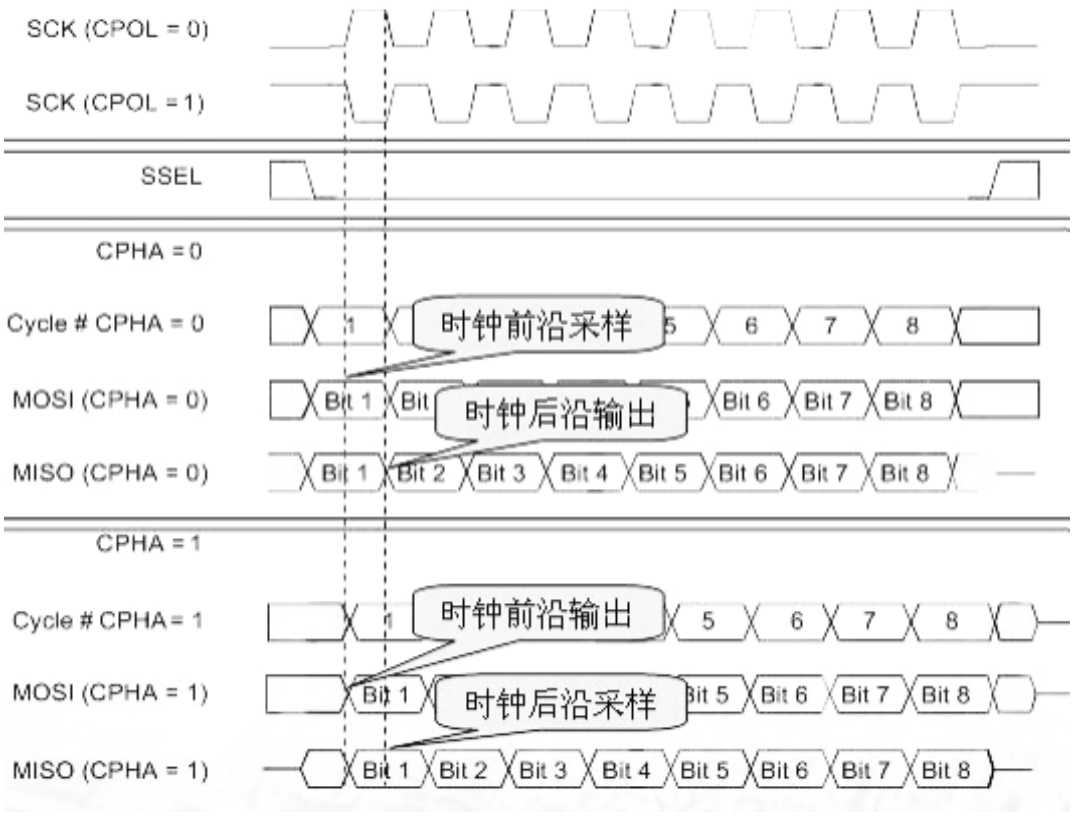

图 21 SPI 通讯协议时序图

3.3.5　片上硬件资源使用

a.通用输入/输出接口（GPIO）

可以看到，本设计中 GPIO 是使用最多的资源，其中包括驱动器的 PWM 信号、方向控制信号以及 OLED 的通信信号和按键的输入读取。由于模拟 SPI 的编写方法和电机驱动信号的产生方法近乎完全一致，因此本部分主要描述步进电机驱动过程和按键处理中 GPIO 的具体使用，SPI 不做赘述。 三种 GPIO 应用的端口分配如下所示。

表 5 按键连接端口

KEY	K3	K4	K5	K7
PIN	PA30	PA29	PA28	PA27

表 6 用于与 OLED 连接的 GPIO 端口定义表

OLED	SPI_CLK	SPI_NS	SPI_MISO	SPI_MOSI	RST
PIN	PA17	NC	PA16	PA12	PA14

表 7 用于与步进电机驱动器连接的 GPIO 端口定义

PIN	Motor1	Motor2
PWM+	PA13	PA6
PWM-	GND	GND
DIR+	PA11	PA4
DIR-	GND	GND
EN+	PA9	PA2
EN-	GND	GND

根据 3.3.2a.电机驱动程序中选定的 GPIO 电平反转加精确延时方案，精确延时使用基于内核时钟计数中断编写的μs级延时函数。具体而言，只需通过电平反转和延时保持即可实现 PWM 的生成。延时时间决定脉冲频率，通过有限次数的循环反转和精确延时以决定脉冲数量。GPIO 持续高电平$50\mu s$，之后反转持续低电平$50\mu s$作为一个脉冲，即脉冲频率为$10kHz$。$10kHz$脉冲示意图如图 22 所示。

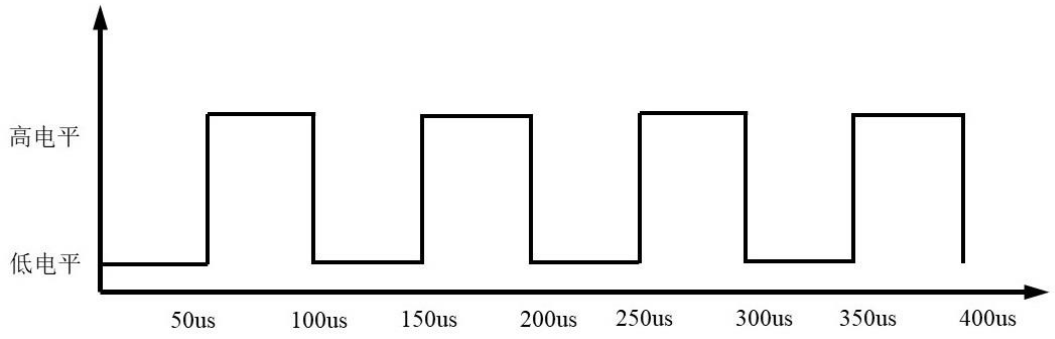

图 22 10kHz 脉冲示意图

按键使用 GPIO 的输入中断模式即可。GPIO 输入中断设置为下降沿触发，当按键未按下时，GPIO 口是高电平，当按下按键是，跳变为低电平，产生中断，在中断函数中识别出引脚号即可判断按键位置。

b.异步串行传输（UART）

无剑 100 的串口含有同步传输和异步传输两种方式[12]。同步方式需要时钟线作为校准，适合大数据量的传输；异步串口不需要时钟校对，但是需要检验位，适合低速，小数据帧的传输。本方案根据实际情况选择了异步串口工作方式，采用中断方式从 FIFO 中进行读取数据。无剑 100 平台的串口读取数据可以使用轮询形式和中断触发方式，查询方式是在主循环中使用串口进行固定大小数据帧的读取后查询接收完成标志位是否置 1 决定是否完成接收，这种方式适用于读取字符数量固定的数据帧传输，一旦传输数据字符数量发生变化，容易引起异常产生；中断触发接收方式则是在串口接收到每一个字符时都触发中断，并储存在 FIFO 中，之后在从 FIFO 中进行读取数据，这种方式便于传输不定大小的数据帧，并且只有数据发送过来时才会进行读取，从而可以避免资源的浪费。

端口$UART0$连接蓝牙串口模块，$UART1$连接 IMU。其配置参数如表 8 所示，包括波特率和数据位等参数的设置。两个模块均是采用中断读取 FIFO 的模式。连续接收 7 或者 11 个字符后，进行一次 FIFO 数据的读取[13]。

表 8 异步串口传输配置表

配置项	参数
波特率（Baud Rate）	115200
传输模式(Mode)	异步传输
校验位(Parity)	无
停止位(Stop Bit)	1 位
数据位(Data Bit)	8 位

4 技术创新点

4.1 步进电机在云台的应用

目前云台多是采用舵机进行各个自由度的方向控制，舵机一般成本较低，外设简单，结构紧凑体积小，但是由于舵机是恒扭矩，功率较小，使得在速度较低，负载较大的运动场景中无法较好的应用。步进电机的功率可以是舵机的几倍之多，可以适用于负载需求较大的场合。除此之外，步进电机在高精度步进驱动器的配合下可以达到较高精确度，而且不同于直流电机，其每一执行步的误差不会积累；步进电机的启动和停止响应都极快，不需要复杂的控制算法。

整个云台系统使用$DC+12V$输入，可以脱离工业电压，直接使用电池供电。整个系统实测运行电流不超过$1A$，总体来说功率不高，适合移动使用。本方案选用步进电机设计云台既可以实现多种负载大小的应用场景中，比如负载较小的监控安防云台，或视觉定位的底层驱动系统，以及负载较大的机器人头部模块等。与此同时，还能达到极高的精确度和响应速度。

4.2 自校准锁定

在用户发送完动作指令后，云台系统接收指令并进行转动到目标位置，之后进入锁定状态。在未接收到用户移动指令时，如果系统外界环境发生三维空间内位置上的偏移，导致整个系统的基准面发生改变，如果不采用自校准，云台的角度也会随之改变，从而无法达到目标位置。当采用了使用 IMU 获取系统基准面的三维空间位置信息后，可以据此判断外界环境是否发生改变，当检测到变化后，启动自校准功能，保持云台的角度仍然是用户的目标角度，达到一个自适应的锁定功能。

4.3 系统资源的极小化运用

在整个设计中仅使用到与步进电机控制相关的 3 个通用输入/输出接口以及 2 个定时器中断与 1 个异步串列传输通讯接口。济小台认同少即是多的设计理念，不以使用资源多、功能复杂冗余为目的，而是着眼于使用尽量少的接口、简洁高效的程序实现最核心实用的功能，从而可以极大的节省算力，为其他应用外设可

以分配出更充足的开发空间，如可以加入图像采集、视觉处理算法等，进而实现云台追踪。

5 团队介绍

李珈毅，电子科学与技术系 2016 级本科生。团队队长，主要负责作品设计、研发，技术方案拟写，完成无剑 100 SoC 的开发研究等工作。

李伟博，电子科学与技术系 2016 级本科生。团队技术总负责人主要负责作品研发、技术细节优化、原型系统构建和调试等工作。

翁锦煜，电子科学与技术系 2017 级本科生。团队技术负责人，主要负责产品设计、技术细节优化、系统调试等工作。

6 后续工作展望

后续研发的重点如下：

a.目前使用的是通用开发板，很多功能使用不上，且占用体积较大，下一步需要根据实际应用所需要的端口资源等硬件资源重新设计核心控制板，把和电机的连接接口，通信接口等集成到一块上面。大幅缩小系统体积。

b.通过蓝牙 mesh 组网将其与其他家用智能设备联网，组成家用智能安防网络；

c.利用 wujian100 SoC 以及 FPGA 的其余资源实现运动监控的功能；

d.使用 WLAN 通讯模块将视频实时上传至家中的 NSA 设备。

7 项目心得体会

无剑 100 作为一款开源的基于 RISC-V 指令集的 SoC 平台，承载了很多国内科研人员的重望，也不负众望，可以说是一款非常优秀的 MCU。其功耗低、占用资源少、轻量、开发简便的特点使得无剑 100 可以作为一款初学者都易上手的 SoC。在开发过程中首先需要进行 SOC 文件的综合烧写等步骤，这一过程中，我们学习到了 FPGA 的使用并通过 RISC-V 架构 MCU 的开源代码进一步了解到了新架构新指令集设计 CPU 的方法。

在使用的过程中，本着方便日后更多同学和开发者能够更快速便捷地使用无剑平台，本团队编写了英文的功能使用手册，发布在 jiayi's Blog 上，共有 12 个详细的使用教程[5, 12, 14-20]，包括使用 windows 的比特流生成、CDK 开发平台与 wujian100 SDK 简介、新项目创建、快速开始项目、通用输入/输出接口的使用、同/异步串列传输的使用、定时器的使用、中断向量控制器的介绍、问题汇总以及片上按键资源的使用。

作为一个电子相关专业的本科生团队，深知中国芯片行业目前举步维艰的困难处境：前有高通苹果在集成芯片领域绞杀，后有英特尔、AMD、ARM 的威胁勒索，中间还夹着随时可能断供生产的台积电。就连嵌入式芯片，都被 ST 微电子、恩智浦、德州仪器等公司围困，即便是走在最前列的海思半导体，也受到重重限制和掣肘。

所以中国芯片，尤其是中国的嵌入式微处理器要在 RISC-V 上杀出一条血路。有幸能够参与到平头哥半导体冠名的该项赛事，让我们得以了解中国公司目前在 RISC-V 指令集上的工作和进步。这是芯片历史上的很小一步，却是中国芯片发展里程上的重大成就。希望平头哥半导体能够在无剑的基础上流片生产出自主设计研发的 MCU，让仍在大学的我们也能有机会学习中国自主的嵌入式芯片与架构并应用到实际的工程实践项目中。

8 参考文献

[1] 罗世伟. 视频监控系统操作与维护[M]. 北京: 电子工业出版社, 2019 年 06 月.

[2] 平头哥半导体有限公司. wujian100_open User Guide v1.0[EB/OL]. 2020-06-22. https://github.com/T-head-Semi/wujian100_open.

[3] 平头哥半导体有限公司. XC7A-FPGA 开发板用户手册(FMX7AR3B).v1.0[EB/OL]. 2020-06-22. https://occ.t-head.cn/community/.

[4] 平头哥半导体有限公司. FMX7AR3B-Schematic[EB/OL]. 2020-06-22. https://occ.t-head.cn/community/.

[5] 汇承科技有限公司. HC-06 V2.0 蓝牙串口通信模块用户手册[EB/OL]. 2020-03-13. https://wenku.baidu.com/view/23d996bfdbef5ef7ba0d4a7302768e9951e76e06.html?re=view.

[6] 维特智能. WT931 姿态角度传感器说明书[EB/OL]. 2020-03-13. http://wiki.wit-motion.com/doku.php?id=wt931%E8%B5%84%E6%96%99#wt931%E8%B5%84%E6%96%99

[7] Lianglonghui. wujian100_open 的 FPGA 实现——如何用 vivado 生成 wujian100 的比特流文件[EB/OL]. 2020-06-22. https://occ.t-head.cn/community/post/detail?spm=a2cl5.1430 0636.0.0.429d180fLO W0E8&id=654091577878118400

[8] LI Jiayi. FPGA Development with wujian100 SoC - Part One: Bitstream Generation[EB/OL]. 2020-03-25. https://shieldjy.github.io/post/FPGA-Development-with-WJ100-SoC-P1.html.

[9] 冯涛,李擎,潘月斗,冯永锋,潘奕琛.步进电机梯形加减速曲线规划控制实验系统设计[J].煤矿机械,2020,41(07):23-25.

[10] 邢然,郑国昆,任晓伟,丁保民.步进电机加减速曲线设计方法研究[J].工程建设与设计,2018(06):48-49+85.

[11] 陈祖霖,黄峰,吴靖,沈英.步进电机 S 曲线调速控制研究[J].福州大学学报(自然科学版),2019,47(05):640-645.

[12] 陈金龙.步进电机多段 S 曲线加减速控制研究与设计[J].电子世界,2020(08):112-113.

[13] Edwin. wujian100 编程——串口应用[E]. 2020-04-26. https://verimake.com/topics/113.

[14] LI Jiayi. FPGA Development with wujian100 SoC - Part Two: CDK Toolkit and wujian100 SDK[EB/OL]. 2020-03-27. https://shieldjy.github.io/post/FPGA-Development-with-WJ1

00-SoC-P2.html.

[15] LI Jiayi. FPGA Development with wujian100 SoC - Part Three: Start a New Project on CDK[EB/OL]. 2020-03-29.https://shieldjy.github.io/post/FPGA-Development-with-WJ100-SoC-P3.html.

[16] LI Jiayi. FPGA Development with wujian100 SoC - Part Four: Hello World[EB/OL]. 2020-03-31. https://shieldjy.github.io/post/FPGA-Development-with-WJ100-SoC-P4.html.

[17] LI Jiayi. FPGA Development with wujian100 SoC - Part Five: GPIO[EB/OL]. 2020-04-06. https://shieldjy.github.io/post/FPGA-Development-with-WJ100-SoC-P5.html.

[18] LI Jiayi. FPGA Development with wujian100 SoC - Part Six: UART[EB/OL]. 2020-04-08. https://shieldjy.github.io/post/FPGA-Development-with-WJ100-SoC-P6.html.

[19] LI Jiayi. FPGA Development with wujian100 SoC - Part SEVEN: TIMER[EB/OL]. 2020-04-09. https://shieldjy.github.io/post/FPGA-Development-with-WJ100-SoC-P7.html.

[20] LI Jiayi. FPGA Development with wujian100 SoC - Part EIGHT: Interrupt(VIC)[EB/OL]. 2020-04-13. https://shieldjy.github.io/post/FPGA-Development-with-WJ100-SoC-P8.html.

[21] LI Jiayi. FPGA Development with wujian100 SoC - Part Ten: Add On-Board Buttons to SoC[EB/OL]. 2020-08-08. https://shieldjy.github.io/post/FPGA-Development-with-WJ100-SoC.html