济小台——基于无剑 100 SoC 的安防监控云台

李珈毅[1]，李伟博[2]，翁锦煜[3]，张志明[4]
（同济大学电子与信息工程学院，上海 200092）

摘要：“济小台”电动云台系统，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，监控范围大、可搭载重负载，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。系统具有如下特点：高达 1.4ms/deg 的运行速度，0.1°的控制精度；使用无剑 100 SoC 作为主控核心，系统快速灵活编程；通过蓝牙按键等外设实时交互控制云台姿态；状态锁定自校准功能锁定目标角度，降低外界干扰。

关键词：电动云台；步进电机；无剑 100 SOC；

0 引言

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移动云台两种。移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。目前根据不同的应用环境，云台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。本文介绍在无剑 100 开源平台上开发与实现“济小台”步进电机云台的方法和流程。

1 系统硬件方案设计

1.1 方案概述

本文基于平头哥无剑 100 开源平台[2]设计，其支持通过 EDA 工具进行前端仿真和制作可编程逻辑器件（FPGA）进行测试。系统总体构造框图如图 1 中所示，主控核心硬件为主控 FMX7AR3B 开发板[3-4]，动力单元由两台 57/42 步进电

[1]参赛队员，电子科学与技术系 2016 级本科生，研发组长。主要负责方案设计、研发，技术报告撰写等。

[2]参赛队员，电子科学与技术系 2016 级本科生，技术骨干。主要负责原型系统开发调试、技术细节优化等。

[3]参赛队员，电子科学与技术系 2017 级本科生，技术骨干。主要负责产品设计、技术细节优化等。

[4]指导教师，电子与信息工程学院，工学博士，讲师，研究方向：检测技术与自动化装置等。

机、电源供应和信号传输电路板、蓝牙模块、OLED 模块等组成。所设计与实现的原型系统实物照片如图 2 中所示，多角度展示整机系统的硬件结构。

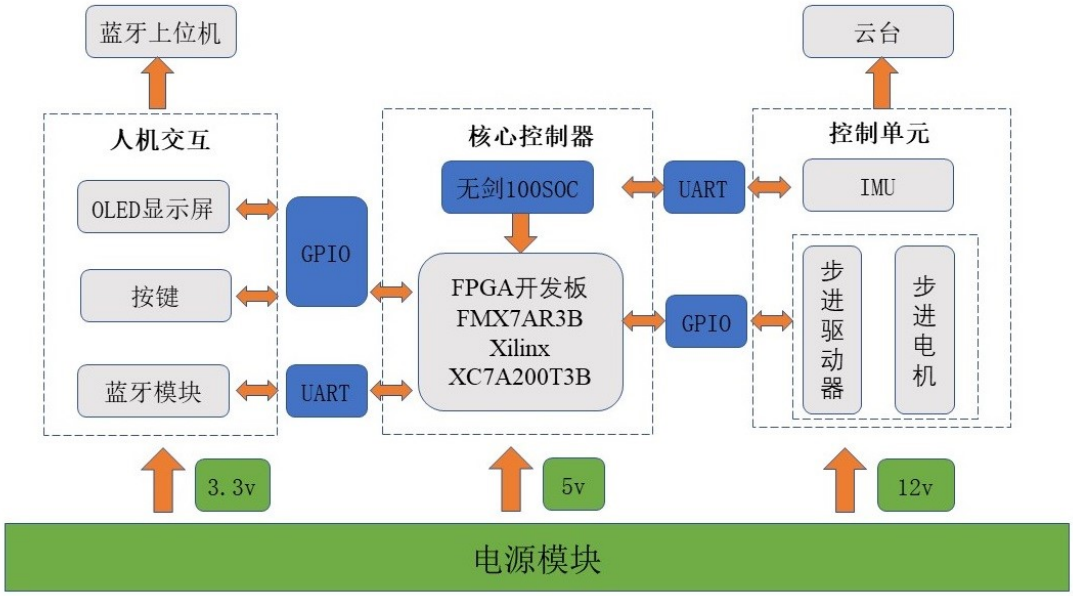


图 1　系统总体构造框图

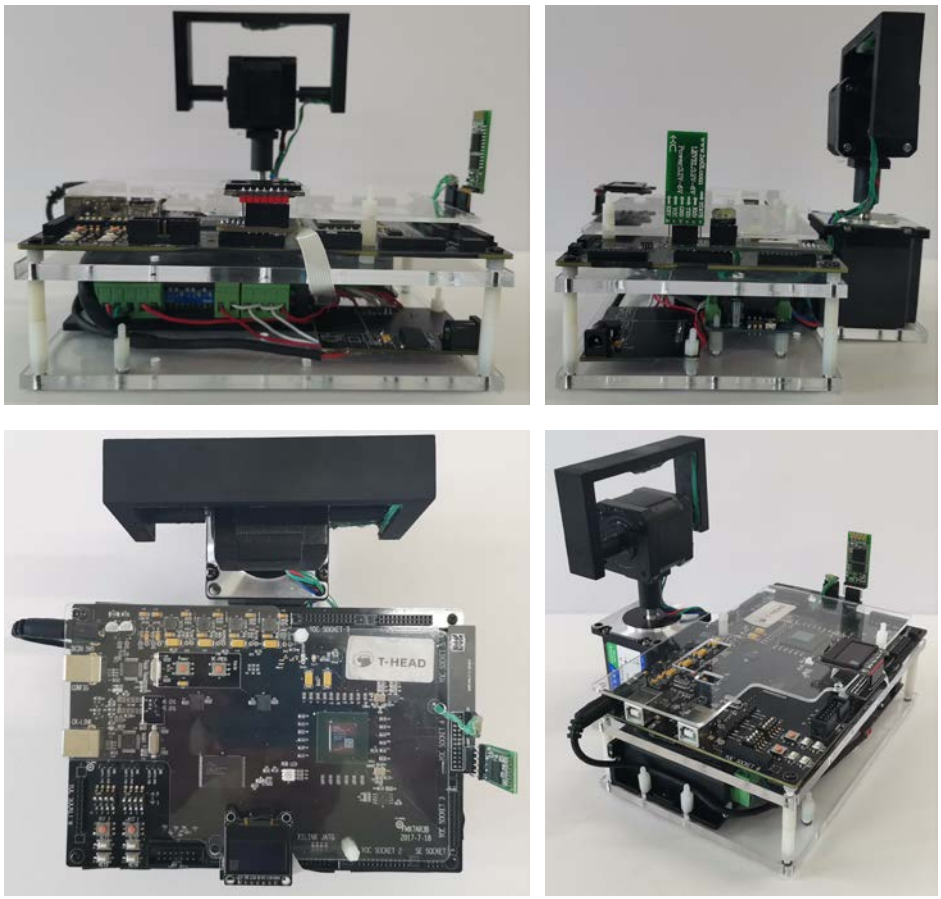


图 2　原型系统实物照片

1.2　电路设计

1.2.1　电源供应

系统有两部分电源，包括为步进驱动器和电机供电的直流+12V 电源，以及为 FPGA 开发板供电的直流+5V 电源。电源共用一个$AC\sim220V$转$DC12V/8A$的电源适配器。由直流 12V 电源经降压电路降至 5V 后供电给 FPGA 开发板，降压电路采用典型的 LM1085/1084 线性稳压集成电路方案，最大输出电流可达 3A。电路原理图如图 3 所示，包括 5V 稳压、3.3V 稳压、接口转接等模块。印刷电路板 PCB 实物图如图 4 所示，包括电源供应标准圆形插口和转接板接口。

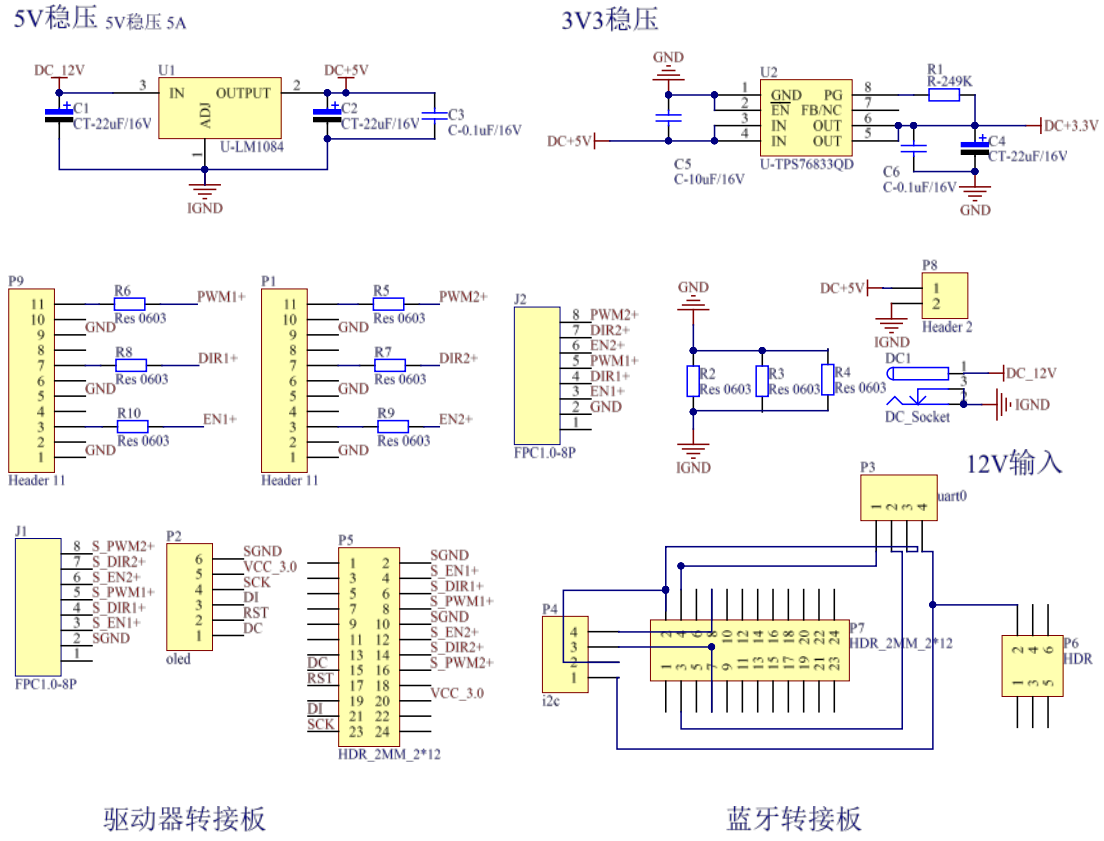


图 3　电源模块电路原理图

图 4 稳压供电及转接板 PCB 实物照片

1.2.2　云台驱动设计

云台驱动方案设计框图如图 5 中所示，中央控制器采用采用平头哥开源无剑 100 SoC。该 MCU 含有 32 个通用输入/输出（GPIO）口、12 个脉冲宽度调制（PWM）输入输出口、3 个可复用的串口/SPI/IIC 的外设资源。

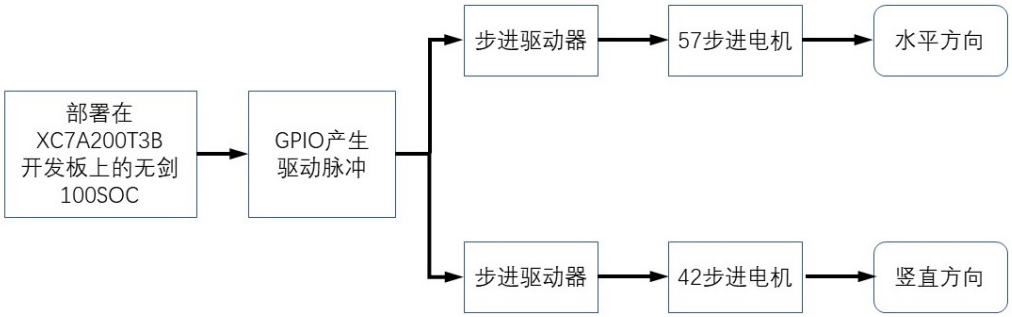


图 5 驱动方案系统框图

动力系统如图 6 所示，由一个普菲德 57 步进电机和一个 42 步进电机构成，通过两个细分高达 25600，电流范围为$1.0\sim4.2A$，工作电压为$DC\ 9\sim42V$的 DM542 步进驱动器分别与无剑 100 SoC 连接，连接方式为共阳极连接。步进使用 6400 细分，即步距角为$360°/6400 = 0.05625°$度，搭配 S 曲线算法，空载电机角度控制精度可达$0.1°$。

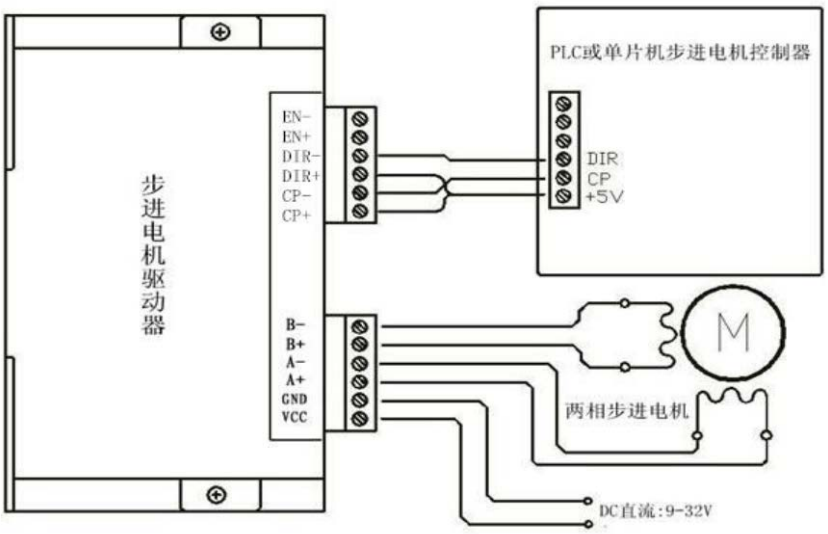
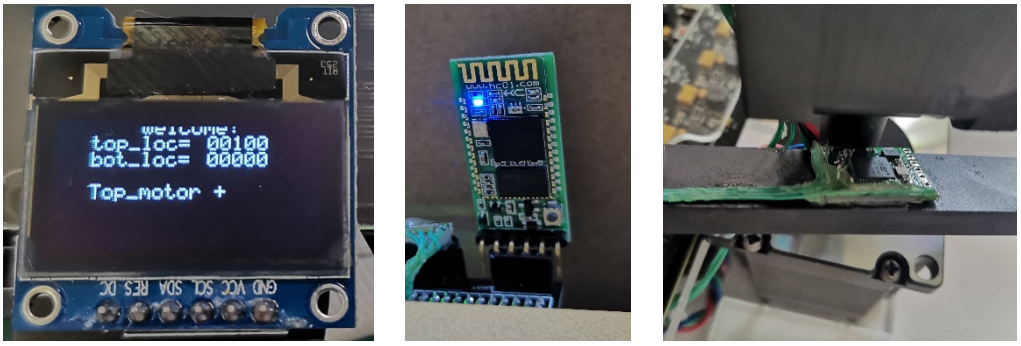


图 6 控制器、驱动器以及电机的总体连接方式

1.2.3 外设功能模块

外设包括开发板上的按键、用于显示控制器内部运行状态的 OLED 显示模块、用于无线通讯的蓝牙模块[5]以及获取三维角度信息的 IMU 模块[6]等。OLED 使用 SPI 协议进行指令控制和数据读写；蓝牙模块和 IMU 模块则是通过串口与主控制器进行通信。外设均通过 2.2 节中设计的转接板与中央控制器连接。



（a）OLED 显示屏　　（b）蓝牙模块　　（c）固定安装的 IMU

图 7 云台系统附属外设功能模块

2 比特流生成及无剑 100 的 FPGA 实现

本团队在 Lianglonghui 的博文[7]基础上，进一步完善和改进，成功生成比特流文件，并撰写英文版教程发布在 jiayi's blog 公开[8]，其流程环节简要介绍如下。

首先将设计源文件及约束文件加入源文件目录；随后增加图 8 中所示的代码块以创建系统时钟；最后综合生成比特流文件。

```
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 8 时序约束语句

资源消耗如表 1 中所示，动态与静态功率消耗如图 9 所示，FPGA 版图设计实现如图 10 所示。

表 1 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

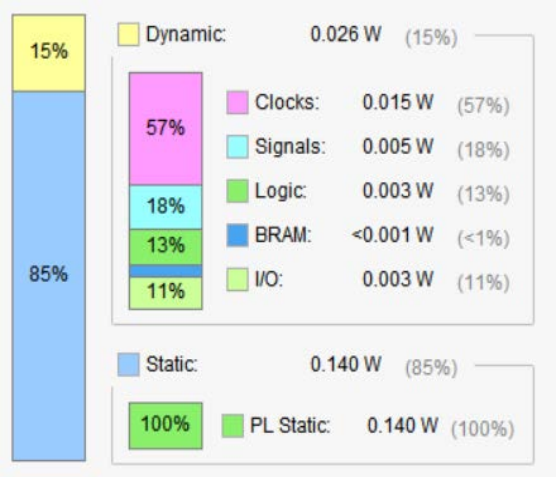


图 9 功率消耗示意图

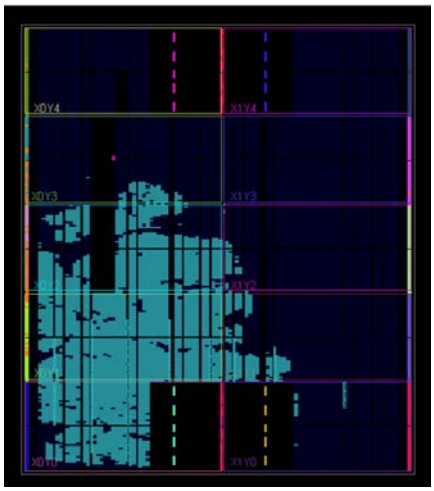


图 10 FPGA 版图

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 AS 模式进行比特流文件的配置。后者利用板载的16MB $FLASH$进行配置，即把比特流文件转换为mcs文件烧入$FLASH$中。FPGA 上电后按下RE_PROG使用主控制器从 FLASH 中读取数据进行配置。综合两种配置方式，使用 FLASH 的 AS 模式配置成功率高。因此本方案选择使用该配置方案。

3　程序运行流程

系统程序运行流程如图 11 所示。首先连接电源，按下RE_PROG键后，FPGA 主动读取 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 GPIO 和 UART 后，设置串口$UART0$和$UART1$通过中断触发读取 FIFO 的方式，按键通过外部中断触发读取，OLED 通过 GPIO 模拟 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，使用通用输入/输出接口(GPIO)产生固定数量的脉冲（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 10Hz 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。

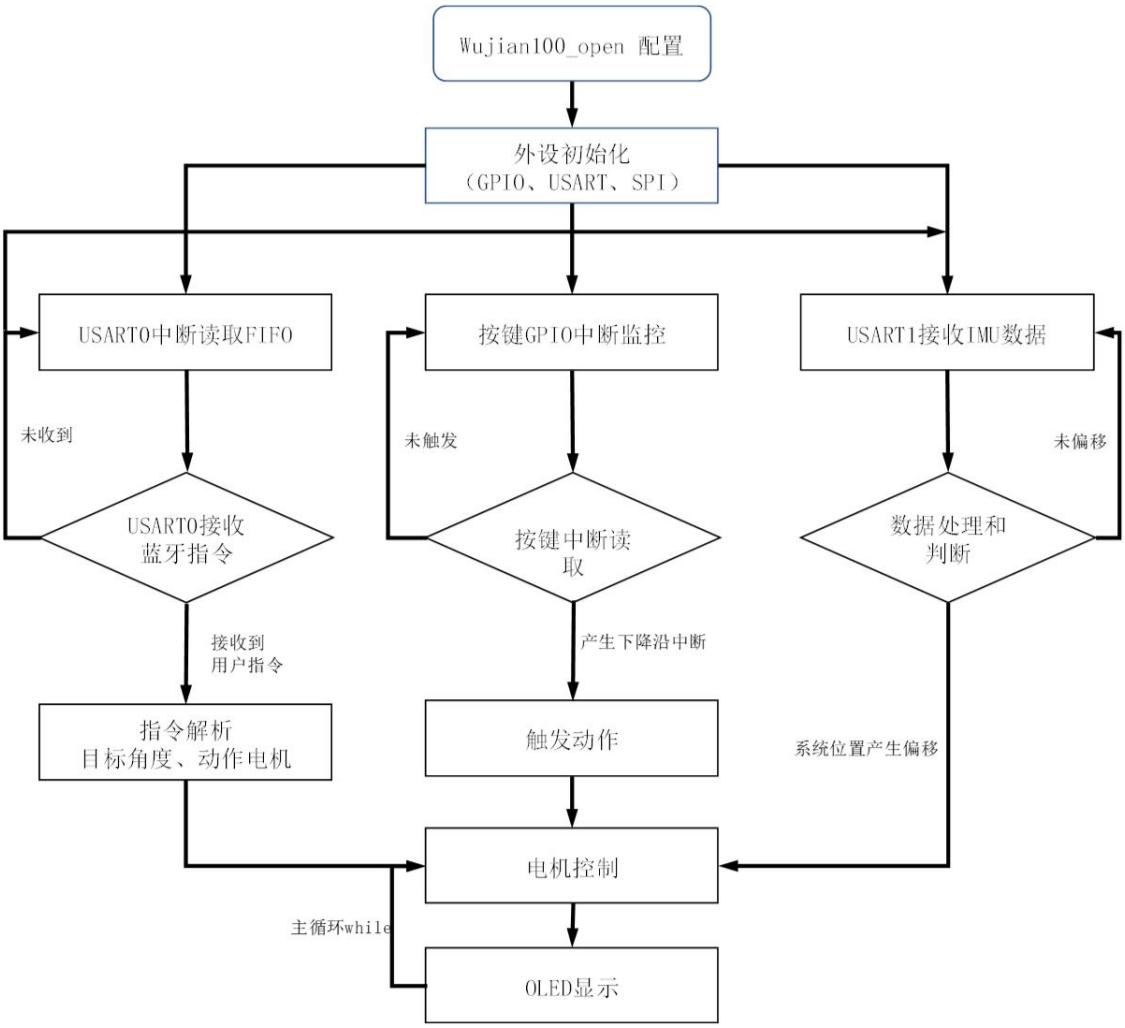


图 11 程序流程图

4　电机驱动和状态锁定自校准算法

4.1　电机驱动 S 曲线算法

由于步进电机在启动时由于需要的加速电流较大，线性度不够，会导致启动慢、丢步；在停止时会出现过冲。所以需要对步进电机启动和停止时的加速度、速度作合理的规划[9-10]。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 产生，输出高电平可使得电机向预设正方向旋转，反之同理。步进电机按步长方式进行运动，每一步及对应一个脉冲。本设计使用 S 曲线加减速控制算法。该算法下的电机加速度和速度变化较为平稳且收

敛较快。如图 12 所示，这里采用的是五段式 S 曲线[11-12]，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。

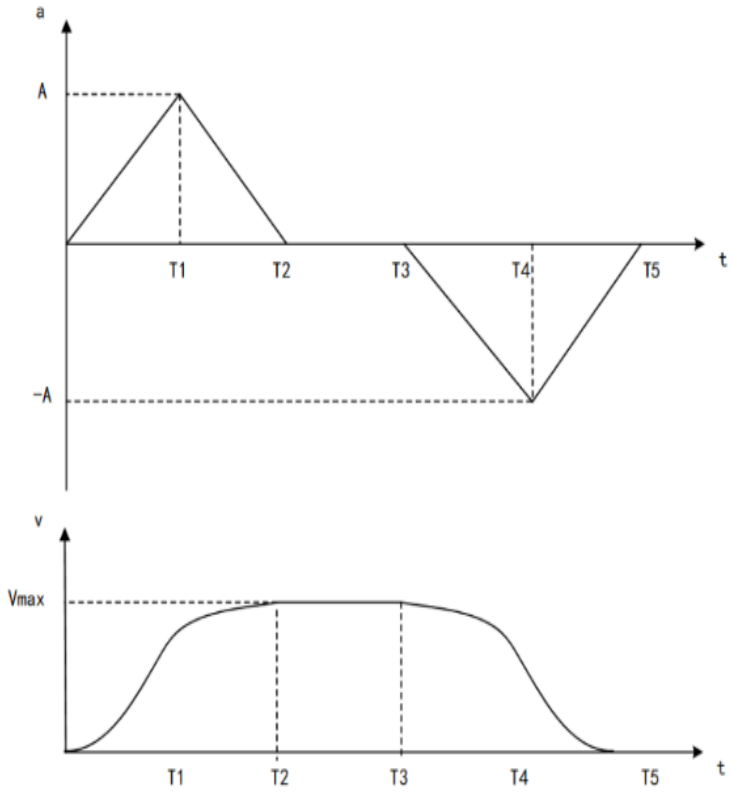


图 12 S 曲线算法示意图

设v为步进电机的稳定运动速度，即 S 曲线中的最大速度；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 2 所示。

表 2 运动过程中加速度与速度的变化

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

步进电机启动加速度的大小和脉冲频率有着直接关系。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

$$\frac{360}{6400} = 0.05625° \tag{6}$$

假设$F_{max} = 30kHz$,$F_{min} = 1kHz$,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。

4.2　状态锁定自校准算法

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。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，进行角角度偏移的校正。

5　片上外设资源使用

5.1　通用输入/输出接口(GPIO)

本设计中通用输入/输出接口（GPIO）是使用最多的资源，其中包括驱动器的脉冲宽度调制（PWM）信号、方向控制信号、OLED 的指令和数据传输、以及按键的输入读取。

5.1.1 按键

板上按键使用 GPIO 输入中断的方式进行按键的触发和识别，使用的端口定义见表 3。

表 3 按键对应的 GPIO 口端口定义

KEY	K3	K4	K5	K7
PIN	PA30	PA29	PA28	PA27

5.1.2 串行外设接口（SPI）

OLED 显示屏使用串行外设接口（SPI）协议与主控制器进行通信作为人机交互界面，显示控制器运行状态。本设计使用 GPIO 模拟 SPI 的通信方式，具体方法是用 GPIO 模拟通信端口和时钟线反转电平并进行延时以模仿 SPI 的行为。除信号线外，还需要一个复位信号（Reset）。具体使用的 GPIO 端口如表 4 所示。

表 4 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

5.1.3 脉冲宽度调制信号（PWM）

产生步进电机需要的 PWM 脉冲信号有两种方案：

一是使用硬件 PWM 发生器，优点为频率高且占空比可调，缺点为脉冲需要软件计数；二是使用 GPIO 口定时反转电平产生 PWM 波形，优点为脉冲数可记、频率与占空比易调。本设计选用方案二，使用基于内核时钟计数中断编写的μs级延时，无剑 100 SoC 的 GPIO 电平可以实现快速反转满足步进驱动所需频率（$10kHz$），实测最高可达$50kHz$。$10Khz$脉冲示意图如图 13 所示。GPIO 端口定义如表 5 所示。

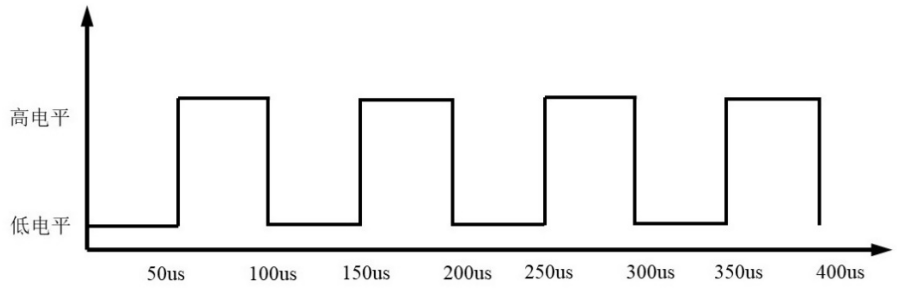


图 13 脉冲示意图

表 5 步进电机驱动器连接的 GPIO 端口定义表

PIN	Motor1	Motor2

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

5.2　异步串列传输（USART）

本设计使用端口 USART0 外接蓝牙串口模块 HC06 与上位机（手机、电脑等设备）通信，使用 USART1 连接九轴 IMU WT931。串口采用中断方式从 FIFO 中读取来自上位机发送的自定义指令，指令格式定义如表 7 所示[13]。

表 7 蓝牙指令格式

指令（7 个字符）	意义
M/L	指令校验： M：Move L：Lockd
+/−（首字符）	电机旋转方向： +：定义方向顺时针旋转 −：定义方向逆时针旋转
xxxx(中间 4 个字符)	脉冲数量： 0~9999
1/2（尾字符）	动作电机位置： 1：top motor 2：bottom motor

指令解析流程如图 14 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，校验通过后首先把字符转化为对应的整数，结合首位字符，生成有符号数代表脉冲数和方向；随后解析末位字符，判断目标电机的位置。解析完成后，回传上位机指令。

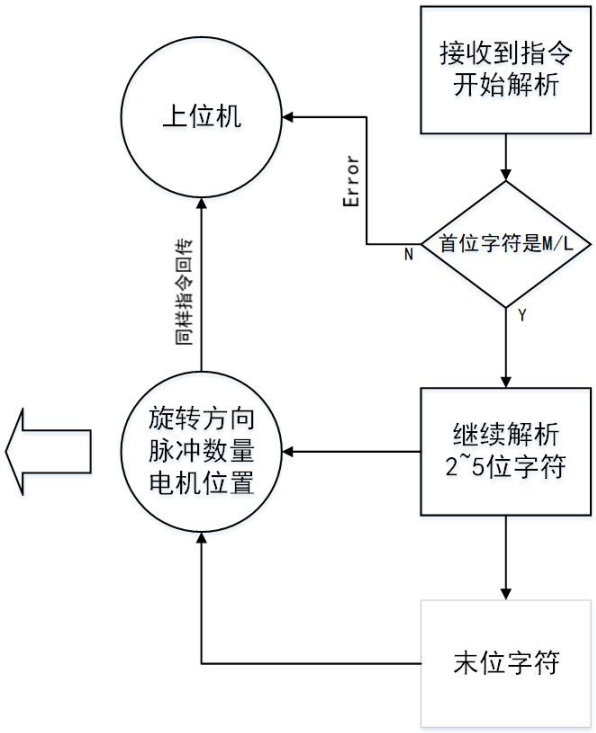


图 14 蓝牙指令解析流程图

6 结语

“济小台”基于步进电机设计具有功率高、精度高且无累计误差、启停响应快、控制算法简单的优点；使用$DC\mp12V$输入，使用电池或直流电源供电，运行电流不超过$1A$，具有功率低、移动性强的优点；状态锁定自校准功能确保云台能够按照用户目标保持稳定，降低外界干扰；其次在整个设计中仅使用到与步进电机控制、按键和 OLED 相关的 14 个通用输入/输出接口与 2 个异步串列传输通讯接口，节省算力与资源，为其他应用外设可以分配出更充足的开发空间。

无剑 100 作为一款开源的基于 RISC-V 指令集的 SoC 平台，其功耗低、占用资源少、轻量、开发简便。在使用的过程中，本团队总结经验编写了英文的功能使用手册，并公开发布在 jiayi’s Blog 上以供更多同学和开发者参考[8, 14-21]。

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 0636.0.0.429d180fLOW0E8&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-WJ100-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