基于 rk3588 的智能示波器系统

白菜披萨

摘要

基于 RK3588 的智能示波器系统创新性地开发了一套具备自然语音控制能力的嵌入式解决方案。该系统以 ELF2 开发板为核心平台，采用“嵌入式语音识别+轻量化 AI 决策”双核架构，通过语音指令实现对示波器五大核心功能的非接触式操控，包括放大/缩小时基、放大/缩小垂直挡位以及自动调节（autoscale）。用户可通过语音命令便捷调整波形显示，获得比传统手动操作更精准高效的体验，尤其适用于需双手操作或恶劣环境的场景。

在技术实现上，系统由三大模块协同工作：ASRPRO 语音识别模块接收指令并转化为 GPIO 编码信号；ELF2 开发板内置 DeepSeek-R1 大模型解析编码，通过深度思考生成对应的 SCPI 指令；最后通过 USB-TMC 协议控制示波器执行操作。关键创新点包括采用纯 Python 方案（PyVISA 和 PyVISA-py 开源库）替代传统 IVI 驱动，实现跨平台（Windows/Linux）通信，以及利用提示词工程优化 AI 指令生成流程，解决了专业 SCPI 指令的学习门槛问题。

系统性能指标优异：语音识别响应时间<0.1 秒、准确率>98%、有效距离>10 米，支持时基调节范围（10μs/div~1ms/div）和垂直档位调节范围（50mV/div~5V/div）。实际测试验证了功能的可靠性：例如输入“自动调节”指令后，示波器成功优化了原始杂乱波形；语音控制时基缩放（如从 200μs 缩至 100μs 使波形变宽）和垂直档位调整（如从 200mV 放大至 500mV 使波形变矮）均能精准响应。

该系统在工业自动化、教育实验、野外监测及医疗设备维护领域具有广泛应用潜力。未来可扩展波形参数实时测量（如 Vpp、频率等 12 项参数）、智能异常检测及报告生成功能。开发过程中克服了硬件适配（如 GPIO 引脚复用）、模型部署优化（DeepSeek-R1 本地推理耗时）和通信协议兼容性（USBTMC 驱动集成）等挑战，体现了模块化设计的高效性。

第一部分　作品概述

1.1 功能与特性

本作品创新性地开发了一款具备自然语音控制能力的智能示波器系统。该系统以 ELF2 开发板为核心平台，采用“嵌入式语音识别+轻量化 AI 决策”双核架构，实现了对示波器功能的非接触式操控。

目前通过语音识别，系统主要控制示波器的五个基本功能：

放大/缩小时基：下达“放大时基”或“缩小时基”命令，系统采集当前参数并自动调整到合适的放大或缩小倍数。

放大/缩小垂直档位：下达“放大垂直档位”或“缩小垂直档位”命令，系统采集当前参数并自动调整到合适的放大或缩小倍数。

自动调节：下达“自动调节”命令，系统将执行一次 autoscale 操作。

用户通过这些语音命令，可便捷地调整示波器，使输入波形呈现更易观察的形状。相比传统的手动按钮和旋钮调节，本系统提供了一种更为精准和简便的操作体验。

1.2 应用领域

1.工业自动化与生产线测试

在汽车制造、半导体测试等需双手操作的环境中，工程师可通过语音实时调节示波器参数，同步监测传感器信号（如焊接电流、电机驱动波形），显著提升调试效率并保障操作安全。

2.科研与教育实验

高校电子工程实验室中，学生可通过语音指令快速调整波形显示效果，避免频繁旋钮操作分散注意力，更专注于电路原理分析与信号特性研究，提升实践教学质量。

3.野外与复杂环境监测

地质勘探、电力巡检等场景常需在恶劣环境下操作。语音系统减少物理接触需求，配合便携式示波器，可快速完成地震检波信号或电网谐波分析，

降低操作失误风险。

4.医疗电子设备维护

在生物电信号（如心电图机）检测中，技术人员通过语音调节垂直挡位捕捉放大微弱信号，避免手动操作引入干扰，保障医疗设备调试的准确性。

1.3 主要技术特点

本方案通过在 RK3588 系统内核中加装 USBTMC 驱动，实现了嵌入式设备与测试仪器的标准化通信。USBTMC（USB Test & Measurement Class）作为 IEEE 488.2 协议的 USB 实现，为示波器等仪器提供了即插即用的控制接口。系统通过内核模块动态加载驱动，无需重新编译固件，同时兼容 USB2.0/3.0 协议，确保高速数据传输。

基于 SCPI（可编程仪器标准命令）的指令集，RK3588 可通过字符串命令精确控制示波器的参数配置如时基、垂直挡位。例如，发送:TIMebase[MAIN]:SCALe <scale>设置时基具体数值。通信过程采用请求-响应模式，支持 ASCII 和二进制数据传输格式，实测指令延迟低于 10ms。

该方案扩展了 RK3588 在自动化测试领域的应用，尤其适用于需要多仪器协同的场景。通过二次封装 SCPI 指令库，可进一步降低开发门槛，提升系统集成效率。

1.4 主要性能指标

GPIO 有效电平	0~3.3V
ASRPRO 输出电平	0~3.3V
语音识别响应时间	<0.1s
语音识别距离上限	>10m
语音识别准确率	>98%
深入思考耗时典型值	30s
可调节时基范围	10 μ s/div~1ms/div
可调节垂直档位范围	50mV/div~5V/div

USB2.0 高速设备端口	1 个，后面板，兼容 USB TMC

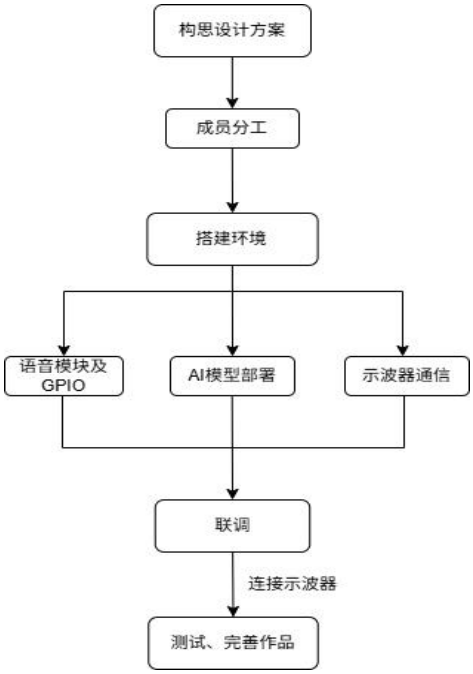
1.5 主要创新点

1.传统方案通常依赖厂商专用的 IVI 驱动程序实现仪器通信，而本设计创新性地采用纯 Python 方案，通过安装 PyVISA 和 PyVISA-py 两个开源软件包，构建轻量级通信后端。PyVISA 提供统一的 SCPI 指令接口，PyVISA-py 则基于 Python 实现 VISA 协议，无需安装驱动即可通过 USB、LAN 或 GPIB 控制示波器。

2.该方案具有跨平台特性（支持 Windows/Linux），通过 Python 脚本可直接实现参数配置（如:ACQuire:POINts 1000）、数据采集与解析，显著降低系统依赖性和开发复杂度。

3.AI 实现语音指令到 SCPI 命令的转换，节省了学习大量 SCPI 指令的时间。

1.6 设计流程

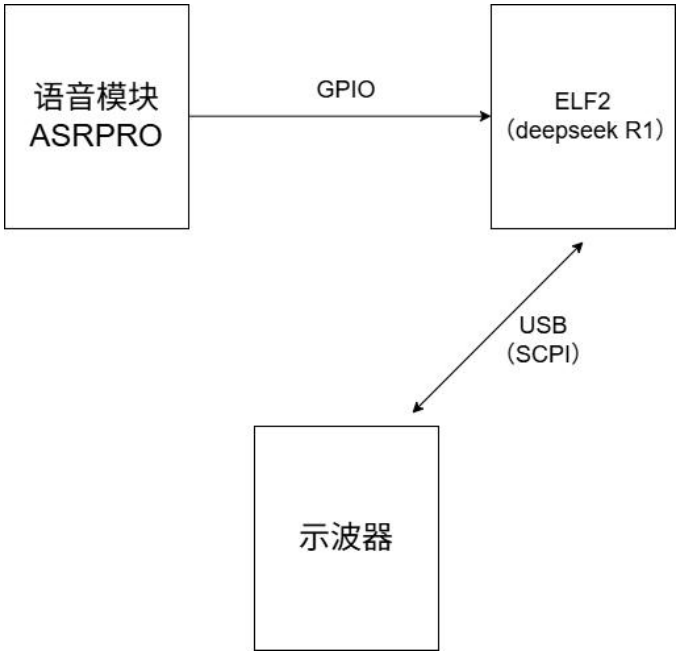


本项目采用模块化开发流程，首先由团队共同完成方案构思和需求分析，确定系统整体架构。随后搭建开发环境并进行任务分解，主要划分为三个开发方向：1）语音控制模块与 GPIO 接口开发；2）AI 模型训练与部署；3）示

波器通信功能实现。三个模块并行开发完成后进入系统联调阶段，重点解决数据交互和时序配合问题。最后通过实际连接示波器进行功能验证测试，测试过程中发现的问题将反馈至对应模块进行迭代优化，直至系统达到设计要求。

第二部分　系统组成及功能说明

2.1 整体介绍



本系统由三大核心模块组成：

1、语音识别模块

采用 ASRPRO 语音识别模块，预先配置关键词与对应电平组合。当接收到语音命令时，模块输出预设的电平信号，通过 GPIO 传输至 ELF2 开发板。

2、AI 指令生成模块

ELF2 开发板内置 DeepSeek-R1 大模型。接收 GPIO 信号后，开发板解析对应的语音指令，由 AI 接收语音指令并通过深度思考生成示波器功能的 SCPI 指令。

3、示波器控制模块

通过 USB 连接 ELF2 开发板与示波器并做好适配。示波器执行接收到的

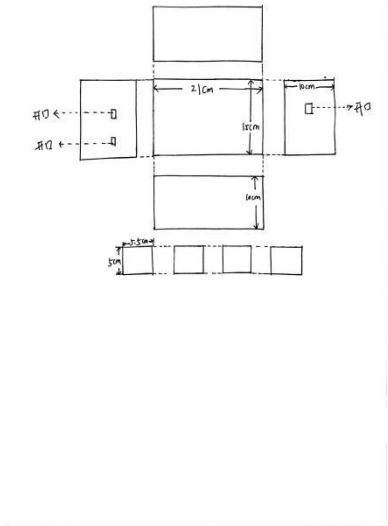
SCPI 指令，实时完成参数调整等操作。

2.2 硬件系统介绍

2.2.1 硬件整体介绍

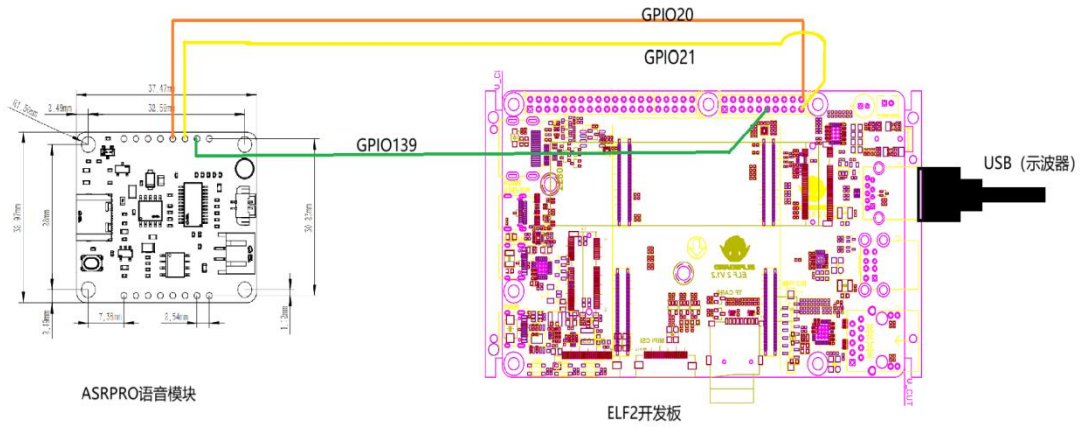
本系统采用高强度雪弗板外壳作为主体结构，内部集成 RK3588 高性能主控模块（ELF2 开发板）和智能语音交互模块（ASRPRO，连接扬声器），构成完整的嵌入式硬件平台。外壳设计兼顾功能性与实用性，设有三个精密开孔：示波器数据接口用于连接外部测试设备，电源输入接口确保稳定供电，以及调试接口支持 PC 连接进行系统开发和维护。整套系统结构紧凑、模块化程度高，适用于智能终端、工业控制等多种应用场景。

2.2.2 机械设计介绍

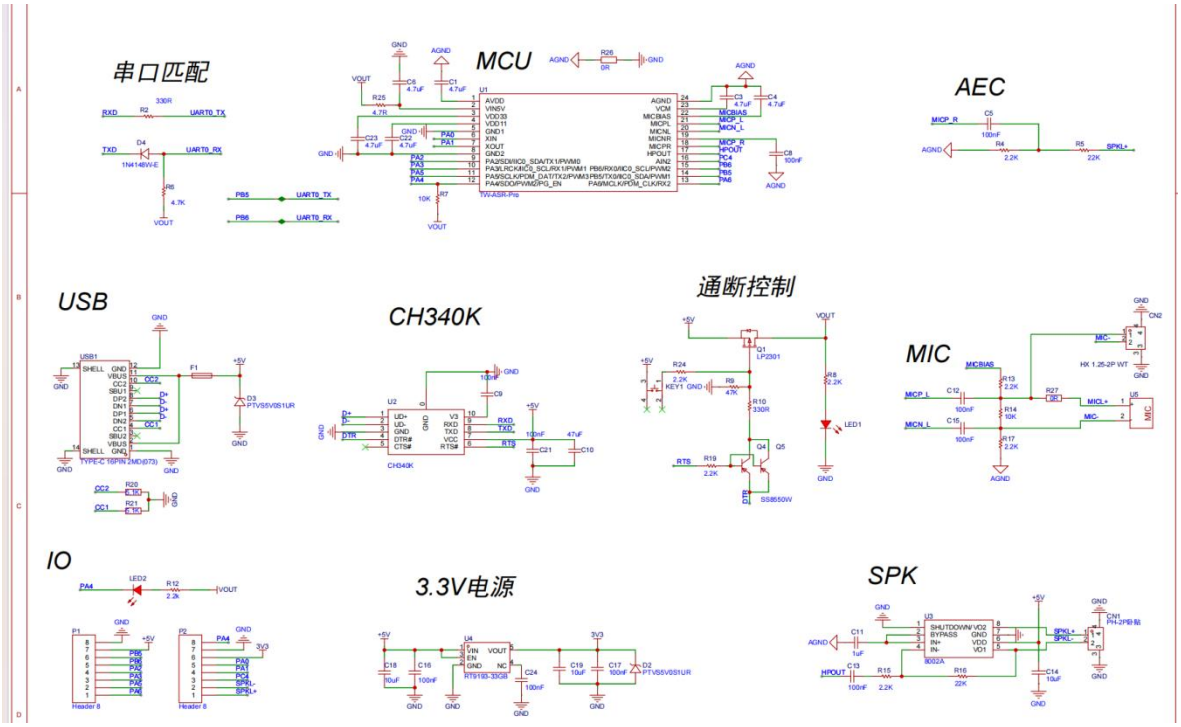


使用雪弗板裁剪粘贴制作的外壳参数如上图所示，包括底板、侧板以及四个固定在内部棱角处的限位板（用于放置 ELF2 开发板）。

2.2.3 电路各模块介绍



关键连线图



ASRPRO 原理图

ELF2 相关引脚图

2.3 软件系统介绍

2.3.1 软件整体介绍

本系统软件部分主要分为三部分：

1、语音识别部分：ASR-PRO 模块上部分——识别并通过 GPIO 编码向 ELF2 开发板传递语音识别结果；ELF2 开发板上部分——将 GPIO 编码转化为语音指令，并生成对应的模型输入。

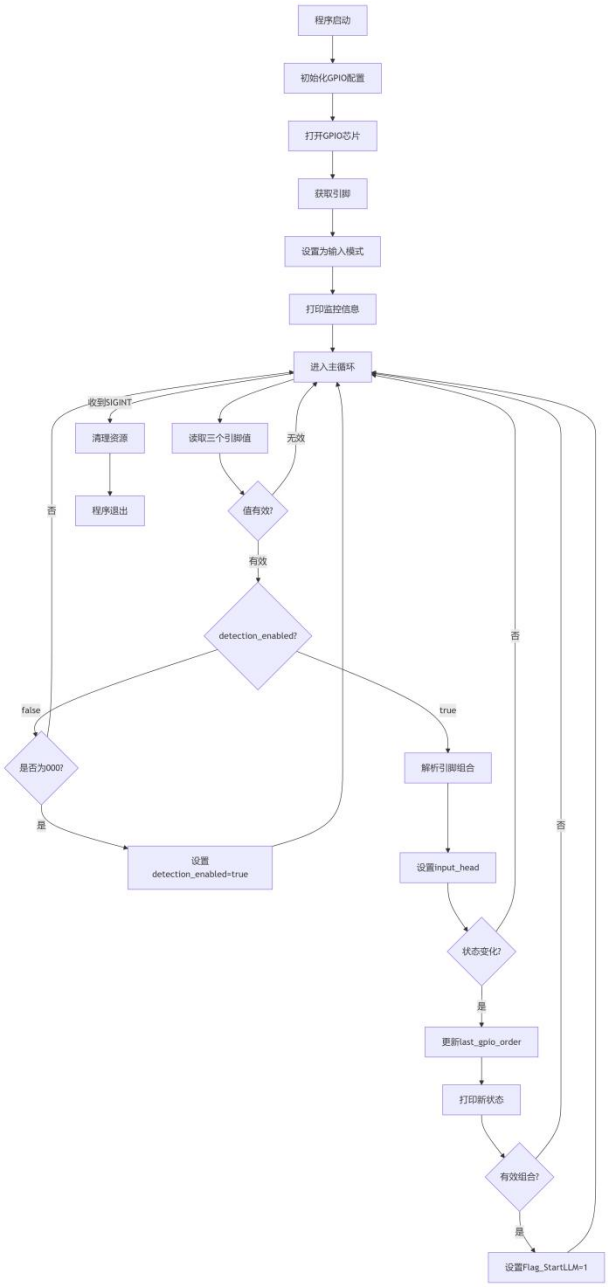
2、模型推理部分：接收语音指令-SCPI 代码转化规则与语音指令，执行模型推理并输出 SCPI 代码。

3、示波器控制部分：ELF2 开发板上部分——识别模型推理中的 SCPI 代码并执行对应的 Python 程序将具体 SCPI 指令发给示波器执行，控制示波器做出响应。

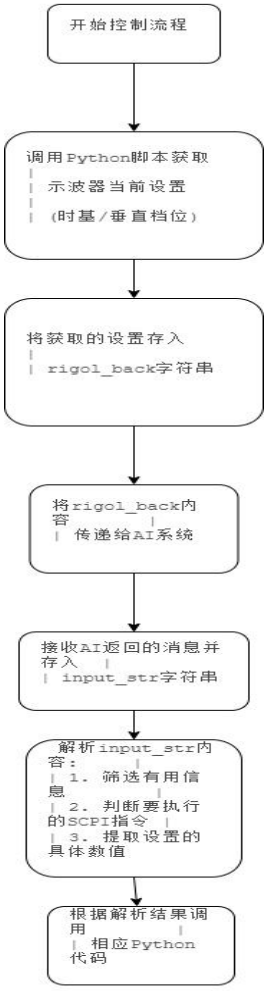
2.3.2 软件各模块介绍

在 ASR-PRO 上，软件系统经过开机时短暂的初始化阶段后进入等待唤醒词阶段，在该阶段若模块读取到外部唤醒词“精灵同学”将进入唤醒状态，此时发出的语音指令会被识别并被模块内部单片机编码，经 GPIO 传输至 ELF2 开发板接入模型输入等待推理；若在唤醒状态中一段时间内未收到语音指令，模块将退出唤醒状态，重新开始等待唤醒词。

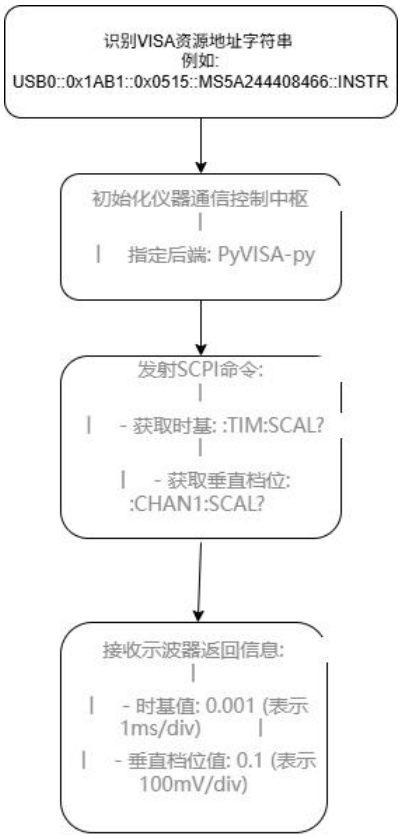
在 ELF2 开发板上，软件系统在初始化阶段依次经过 GPIO 初始化、模型初始化，之后等待语音模块发送编码。在接收到语音模块编码信息后，对编码进行解码，将 GPIO 编码转化为语音指令，调用 Python 程序控制示波器读取当前示波器档位信息，结合语音指令-SCPI 代码转化规则生成对应的模型输入。之后启动模型推理，模型推理会将用户问题、思考过程、最终回答打印到串口上，以供开发者、使用者用 Putty 查看，并将模型输出存储到一个字符串中。使用正则表达式对模型输出进行处理，可以得到控制示波器的 SCPI 代码，根据代码类型调用不同的 Python 程序，实现对示波器的控制。



语音识别方面，程序启动后，开发板首先进行 GPIO 的初始化工作，包括打开要用到的 GPIO 芯片、获取 GPIO 引脚并将其设置为输入模式。随后开发板持续获取三个 GPIO 引脚上的电平值，读取的电平值被储存在**关键变量 values 数组**中。当电平值第一次出现 000 组合时，初始化解锁，启用检测。接着开发板继续利用 **values** 监测 GPIO 的电平值，一旦监测到的电平组合与预设的有效组合相同，将会对关键**字符串变量 input_head** 进行赋值，更新最后记录的状态并设置 Flag_StartLLM=1 触发后续处理。而其他无效组合仅打印状态，不触发赋值和标志设置。赋值后的**字符串变量 input_head** 将会作为 AI 大模型的主要输入内容。



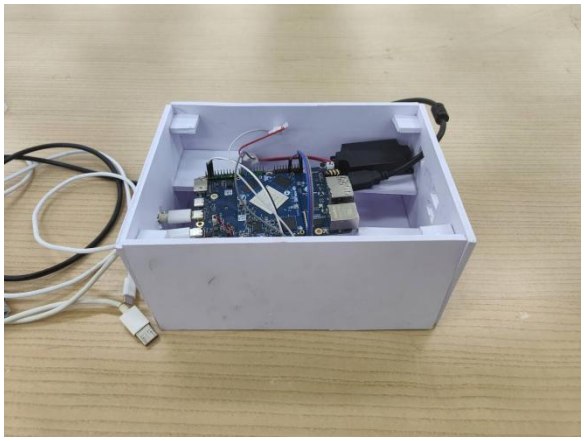
C 语言中关于示波器控制部分，先调用相应 python 代码获取当前示波器时基和垂直档位具体数值，存到 rigol_back_time 和 rigol_back_vertical 字符串里然后传给 AI，在将 AI 返回的消息存到 input_str 字符串里并从中筛选有用信息，判断接下来要执行的具体是哪一个 SCPI 指令，并从 input_str 中读取到具体设置的数值，然后调用相应的 python 代码。

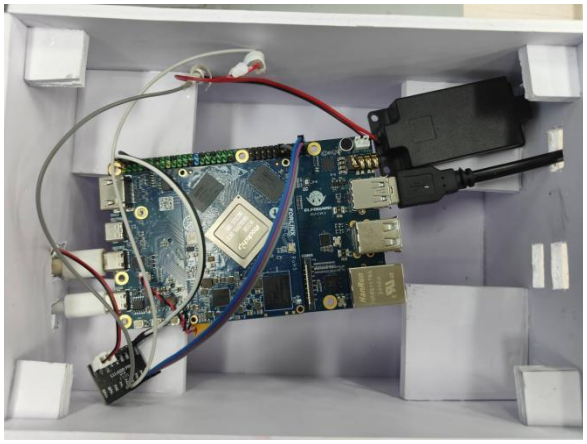


先识别示波器 VISA 资源地址字符串，然后初始化与仪器通信的控制中枢，指定后端驱动为纯 Python 实现的 PyVISA-py 后端，最后发射 SCPI 命令并接受示波器返回的信息。

第三部分　完成情况及性能参数

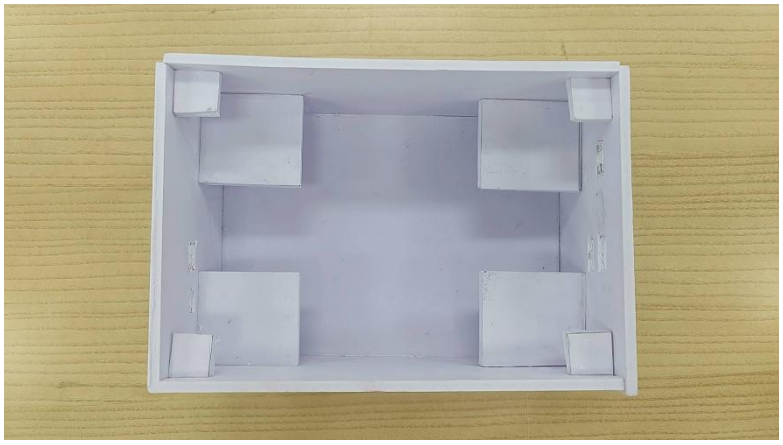
3.1　整体介绍



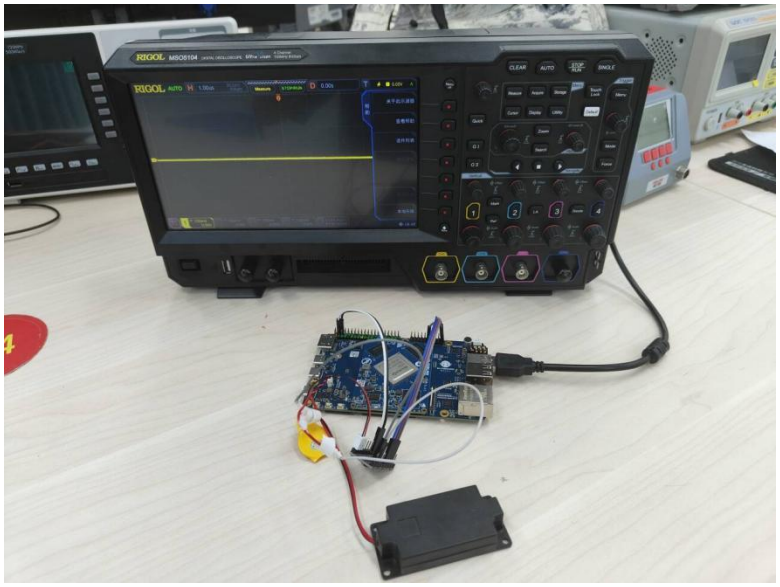


3.2 工程成果

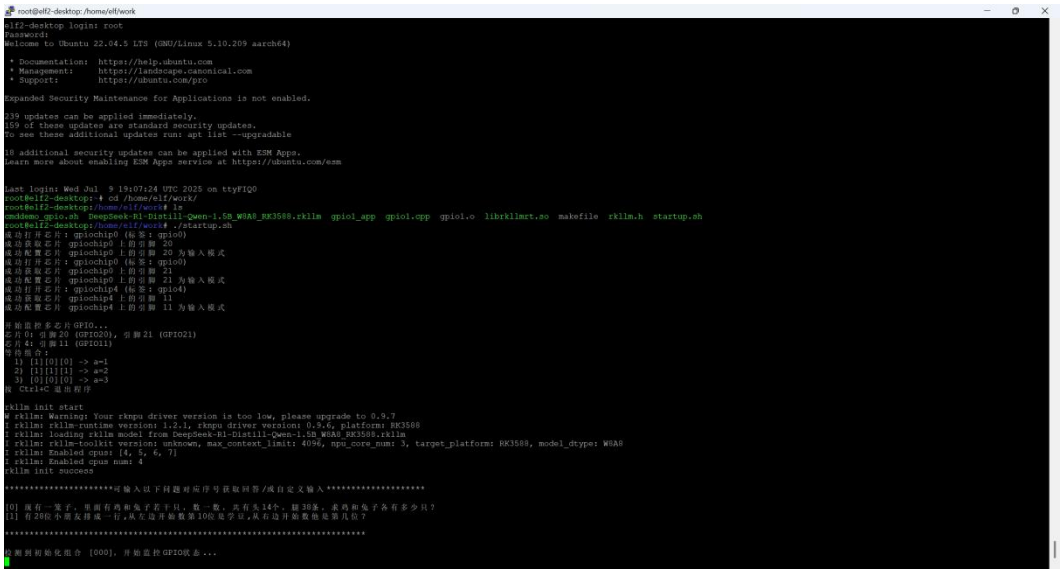
3.2.1 机械成果；



3.2.2 电路成果；



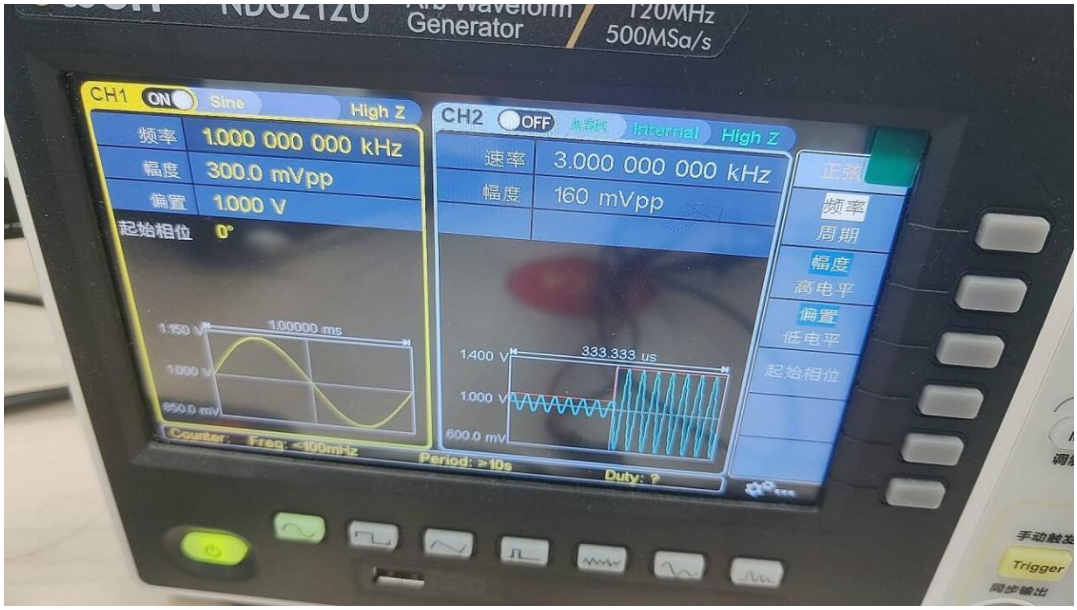
3.2.3 软件成果；



3.3 特性成果

可实现对波形自动调节：

信号发生器输入信号如下图所示



示波器初始波形如下图所示

示波器中没有明显波形，十分不便于观察

下达语音指令“自动调节”后，Putty 串口输出如下

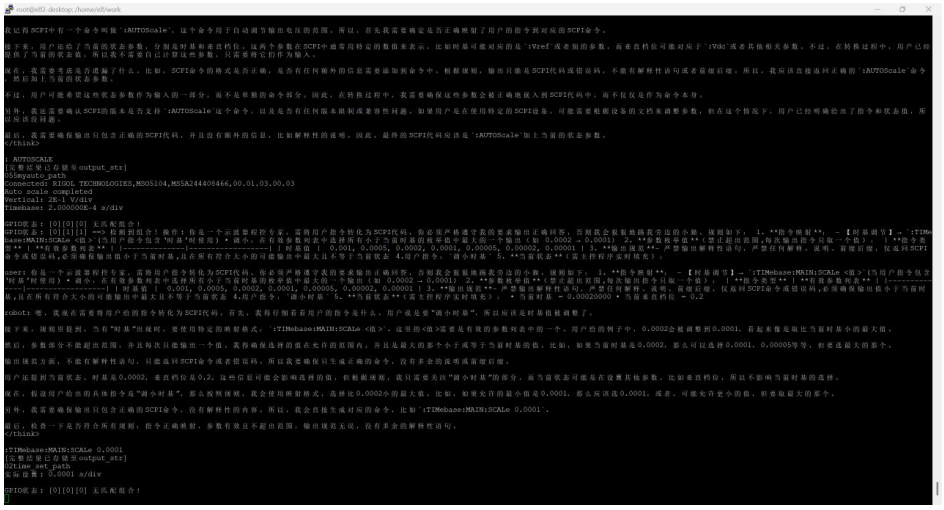
经深度思考后，示波器波形如下

显然，示波器成功执行了“自动调节”操作

可实现对示波器时基单独控制：

与自动调节功能测试方法一致，示波器初始波形如下

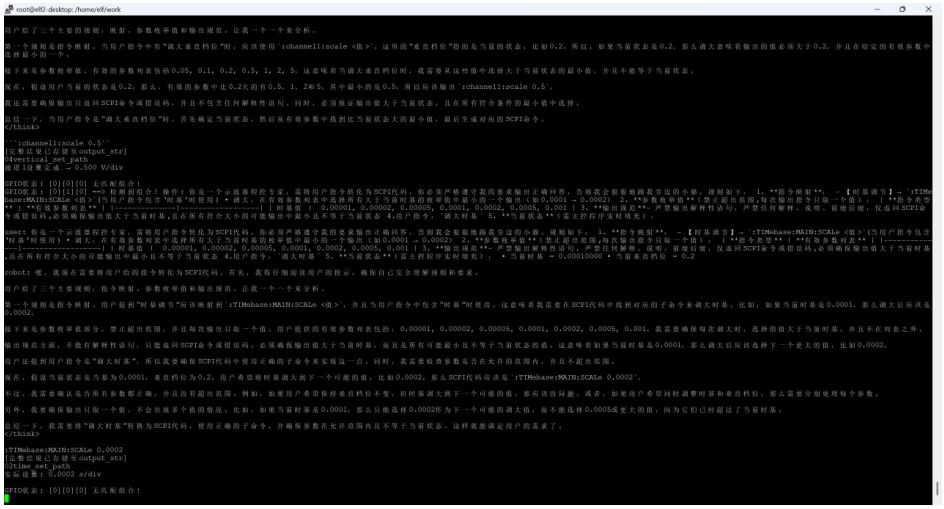
下达语音指令“缩小时基”后，Putty 串口输出如下



经深度思考后，示波器波形如下

时基由 200μs 缩小为 100μs，波形变“宽”

紧接着下达“放大时基”语音命令，Putty 输出如下



经深度思考后，示波器波形如下

时基由 100μs 放大回 200μs，波形变“窄”

可实现对示波器垂直档位单独控制：

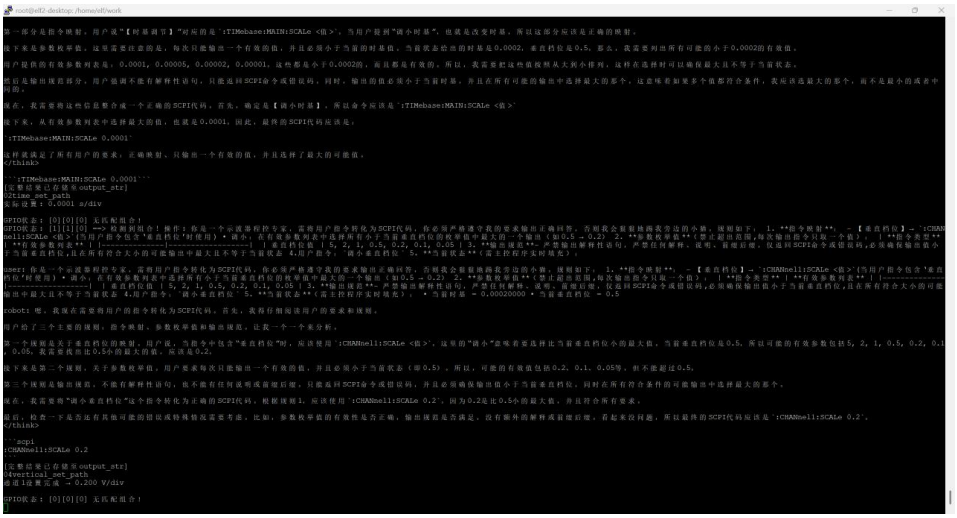
示波器初始波形如下

下达语音指令“放大垂直档位”后，Putty 串口输出如下

经深度思考后，示波器波形如下

垂直档位由 200mV 放大到 500mV，波形变“矮”

紧接着下达“缩小垂直档位”语音命令，Putty 输出如下



经深度思考后，示波器波形如下

垂直档位由 500mV 缩小回 200mV，波形变“高”

第四部分　总结

4.1 可扩展之处

系统将扩展 12 项波形参数实时测量功能，涵盖电压类（Vpp/Vmax/Vmin/Vavg/Vrms）、时间类（频率/周期/占空比/上升下降时间/脉宽）及特殊参数（过冲/振铃）。通过滑动窗口均值滤波与三点插值算法，在 100MHz 带宽下将频率测量误差控制在 0.1%以内。显示系统将开发专业级可视化界面，支持原始/滤波波形双图层叠加显示与参数动态标注，同时新增一键截图（PNG/JPG/SVG）、多格式数据导出（CSV/Excel/MATLAB.mat）

及自动化报告生成功能。加入智能辅助系统，重点构建基于深度强化学习的异常检测模型，建立可迭代专家知识库，提供探头设置优化、采样模式调整等实时建议，并通过学习用户习惯实现个性化参数推荐。针对 DeepSeek-R1 在语音触发后的深度思考耗时问题，将优化推理引擎提升响应效率。上述功能将分阶段落地实施。

4.2 心得体会

参与基于 RK3588 开发语音控制示波器功能的项目，于我们而言是一段充满挑战与成长的经历。这次项目不仅让我们深入了解了嵌入式开发的各个环节，更让我们体会到团队协作、问题解决的重要性。

最初，我们在语音识别功能实现上遇到了困难。起初选用的是官方配套的 LD3320 语音模块，但在实际使用过程中，我们发现该模块控制起来较为复杂，且与我们整体项目的适配性不强。经过反复讨论和测试，我们决定更换为 ASRPRO 语音识别模块。这一改变带来了新的问题：通信方式需要从原来的 UART 通信转为 GPIO 编码通信。更棘手的是，我们发现板子上的绝大多数 GPIO 口不知为何被锁定，仅有少数几个可以正常使用。为了解决这个问题，我们首先仔细查看了开发板上所有 GPIO 口的状态，并通过引脚复用的方式，成功获得了足够数量的 GPIO 口，最终实现了 ASRPRO 与开发板的稳定通信。这一过程虽然曲折，但让我们深刻体会到，面对困难时，团队成员之间的沟通与协作至关重要，只有大家齐心协力，才能找到解决问题的最佳方案。

在大模型应用开发阶段，我们遇到了更大的挑战。首先是资料不足的问题，我们组大部分时间都花在查找资料和反复试错的过程中。其次，本地部署的 deepseek 模型算力有限，而我们的问题又过于专业，导致在不对模型进行训练的情况下，很难达到理想的效果。考虑到我们并没有足够的算力来训练模型，最后我们采用了提示词工程的方式，通过不断优化提示词，反复测试和调整，最终实现了较高的正确率。这个过程虽然艰难，但让我们明白了一个道理：在资源有限的情况下，灵活调整策略，充分发挥现有工具的潜力，往往能够取得意想不到的效果。

在搭建 SCPI 通信环境时，我们再次遇到了难题。原本计划使用官方的 IVI

驱动程序来实现与示波器的通信，但该驱动程序与 RK3588 系统的 Linux 内核版本不符，导致无法正常使用。我们查阅了大量资料，最终发现可以使用 pyvisa 纯 Python 端替代 IVI 驱动程序，成功实现了与示波器的通信。这一经历让我们深刻认识到，在技术开发过程中，遇到问题时，广泛查阅资料、积极寻找替代方案，是突破困境的关键所在。

回顾整个项目过程，虽然遇到了许多困难和挑战，但正是这些困难，让我们不断成长，积累了宝贵的经验。我们不仅提升了自己的技术能力，更在心态和团队协作方面有了长足的进步。未来，我们将继续保持这种积极的态度，勇敢面对各种挑战，不断提升自己，争取在更多的项目中取得更好的成绩。

第五部分　参考文献

1、 ASRPRO 开发板规则书 V1.1

2、 ASRPRO 编程模式开发手册

3、 ELF2-RK3588 本地部署 DeepSeek-R1

4、 基于 RK3588 的 AI 模型训练到部署

5、 ELF2 引脚复用表 20241108

6、 ELF2 开发板快速启动手册

7、 ELF2 开发板编译手册

8、 ELF2 开发板软件系统开发教程

9、 Linux 系统基础入门

10、MSO5000_DataSheet_CN_tcm4-4430

11、MSO5000_ProgrammingGuide_CN