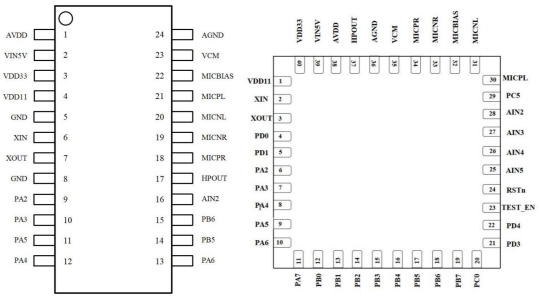
ASRPRO 配置模式使用手册

一、概述

芯片概述

本产品是针对低成本离线语音应用方案开发的一款通用、便携、低功耗、高性能的语音识别芯片，采用了第三代语音识别技术，能支持 DNN\TDNN\RNN 等神经网络及卷积运算，支持语音识别、声纹识别、语音增强、语音检测等功能，具备强劲的回声消除和环境噪声抑制能力，语音识别效果优于其它语音芯片。该芯片方案还支持汉语、英语、日语等多种全球语言，可广泛应用于家电、照明、玩具、可穿戴设备、工业、汽车等产品领域，搭配天问 Block 图形化编程软件，快速实现语音交互及控制和各类智能语音方案应用。



天问提供 SSOP24 和 QFN40 两种封装类型和 2M、4M 两种 Flash 容量类型。具体参数请查看对应的规格书。

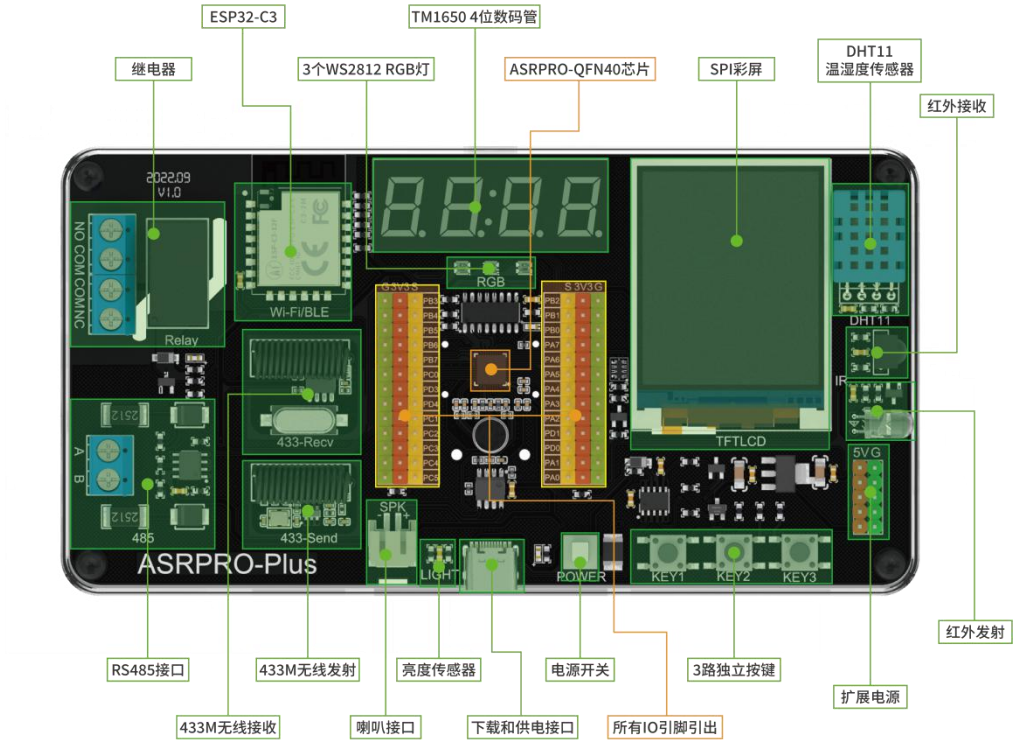
芯片特点

支持离线神经网络计算，支持单麦克风降噪增强，单麦克风回声消除，360 度全方位拾音，可抑制环境噪音，保证嘈杂环境中语音识别的准确性。进行离线语音识别不依赖网络，时延小，性能高，可实现 98%以上的高识别率，10 米超远距离识别，响应时间小于 0.1S。

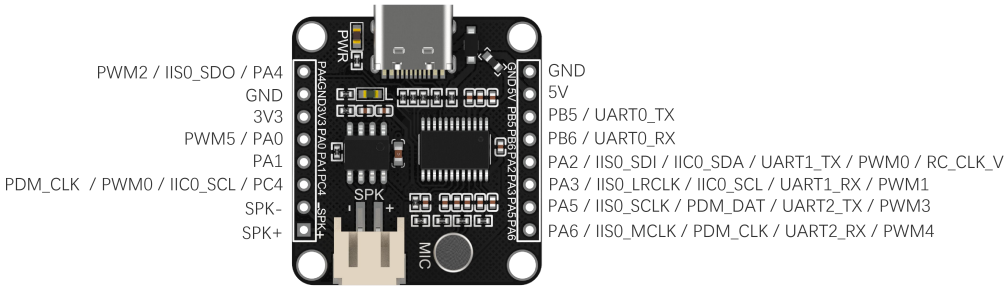
硬件概述

天问基于 ASRPRO 芯片目前推出了 6 种类型，供开发者选择。

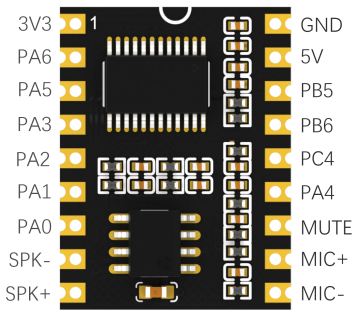
1. ASRPRO-Plus 开发板，一款全功能带语音识别的物联网开发板，方便学习。板载 RS485、433M 无线收发、红外收发、ESP32-C3(2.4GHz Wi-Fi 和 Bluetooth 5LE)、SPI 彩屏、数码管、RGB 灯、光敏传感器、DHT11 温湿度传感器、1 路继电器输出模块。



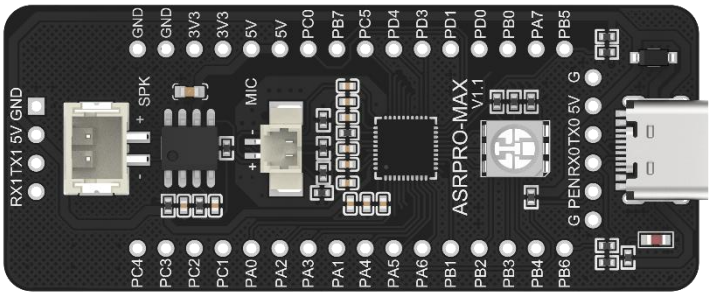
2. ASRPRO 基础开发板，长宽为 30x28mm，板载麦克风、指示灯，用户只需要外接喇叭就可以使用，下载程序需要搭配 STC-LINK 下载器。



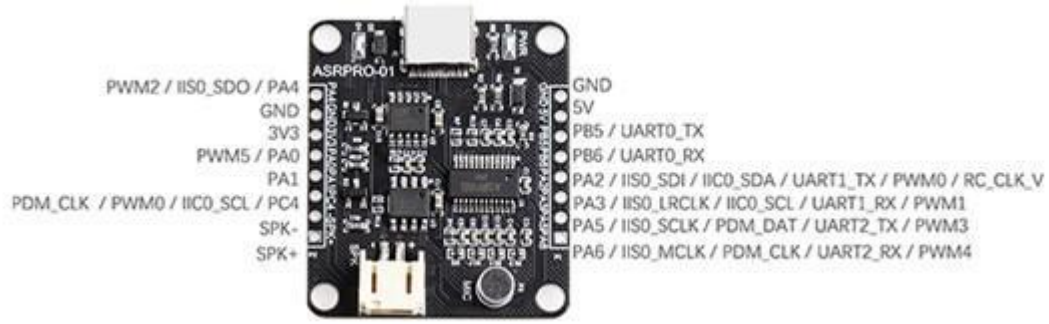
3. ASRPRO-CORE 核心板，模块体积小巧，长宽为 18x23mm，对外接口采用 2 排邮票孔和插针孔，方便采用回流贴片使用和焊接插针使用，喇叭和麦克风都需要自己外接，下载程序需要搭配 STC-LINK 下载器。



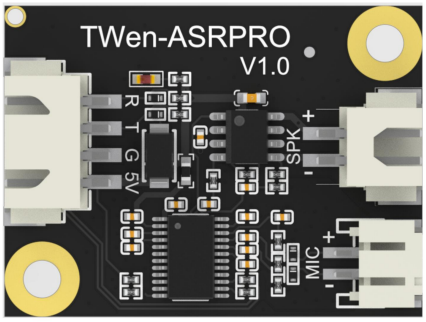
4. ASRPRO-MAX开发板，采用QFN40封装芯片，26路IO口独立引出，长宽为62.0x25.5mm，板载RGB，用户只需要外接喇叭和咪头就可以使用，下载程序需要搭配STC-LINK下载器。



5. 鹿小班 ASRPRO 基础开发板，在 ASRPRO 基础开发板的基础上额外集成了下载芯片，一根 Type-C 线就可以下载程序，并且开发板上有自动断电电路可以实现一键下载，不需要额外的 STC-LINK 下载器。



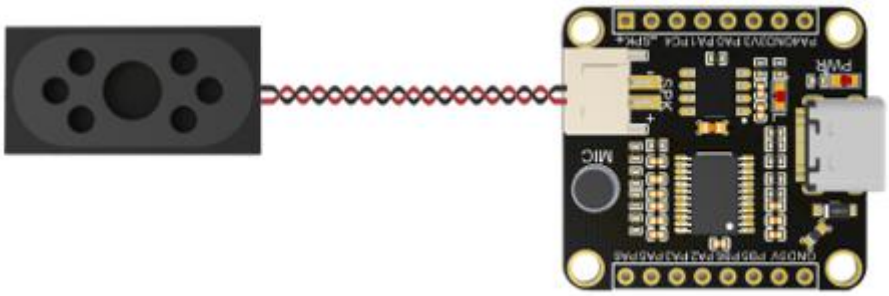
6. ASRPRO 串口模块，只引出了串口、喇叭、麦克风供用户和其它主控搭配使用。

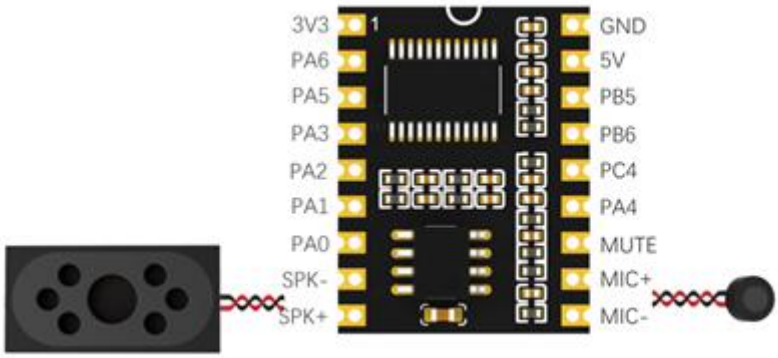


二、开机测试

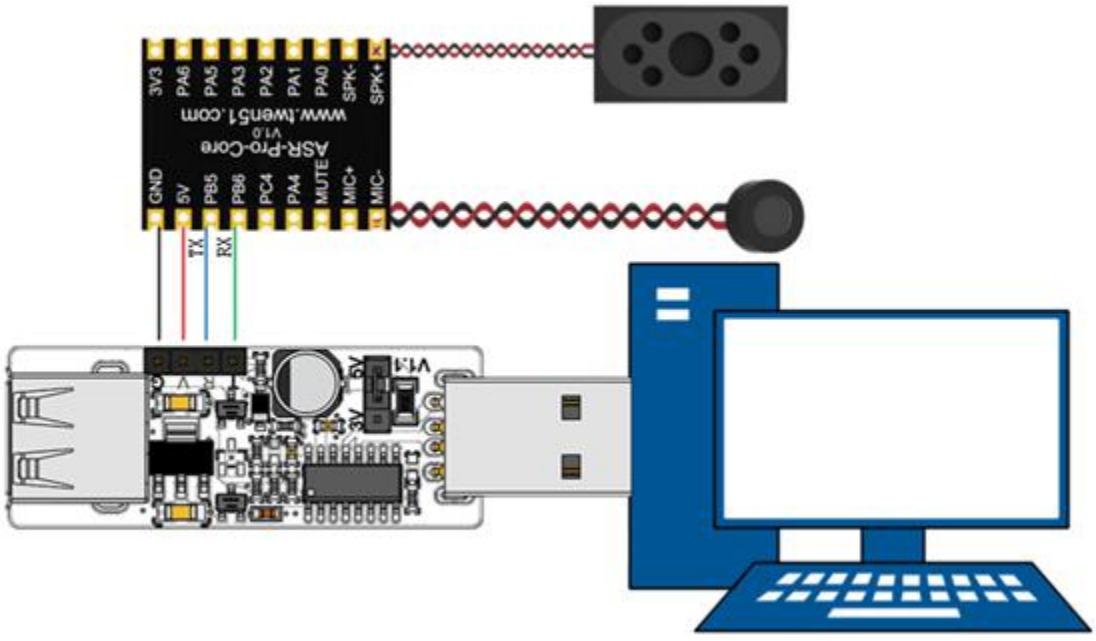
ASRPRO 核心板和基础版开机测试

第一步：开发板需要连接喇叭、核心板需要连接喇叭和咪头。





第二步：用 STC-LINK 连接电脑，再使用 Type-C 数据线将开发板与其连接。



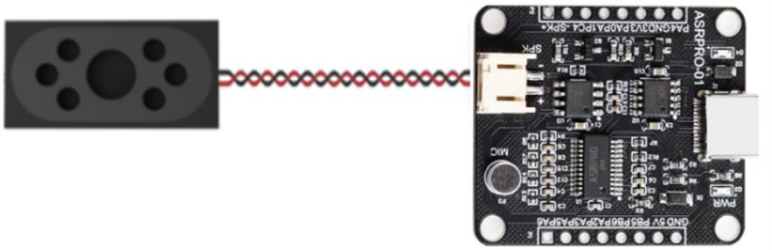
第三步：上电后，会播报欢迎词“欢迎使用智能管家，用智能管家唤醒我”，接下来你可

以用“智能管家”唤醒词唤醒设备，接着用不同的命令词控制设备，详见下表。

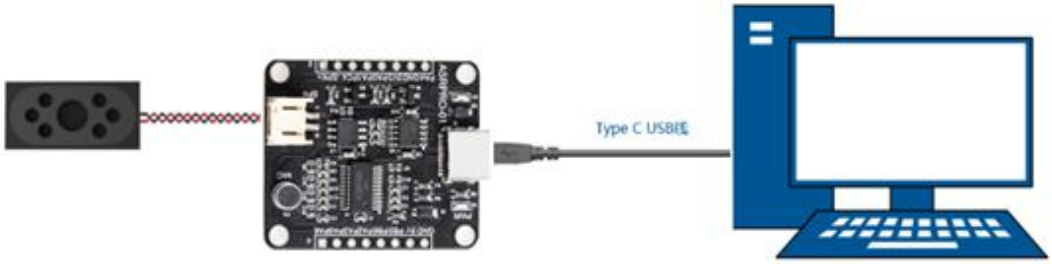
类型	识别词	回复语音
欢迎词	欢迎使用智能管家，用智能管家唤醒我	
退出语音	我退下了，用智能管家唤醒我	
唤醒词	智能管家	我在
命令词	打开灯光	好的，灯光已打开
命令词	关闭灯光	好的，灯光已关闭

ASRPRO 鹿小班基础版开机测试

第一步：开发板需要外接喇叭，喇叭为 PH2.0 接口。下图为开发板实物图



第二步：开发板板载 USB 转 TTL 芯片，只需要一根 Type-C 线就可以实现一键下载。



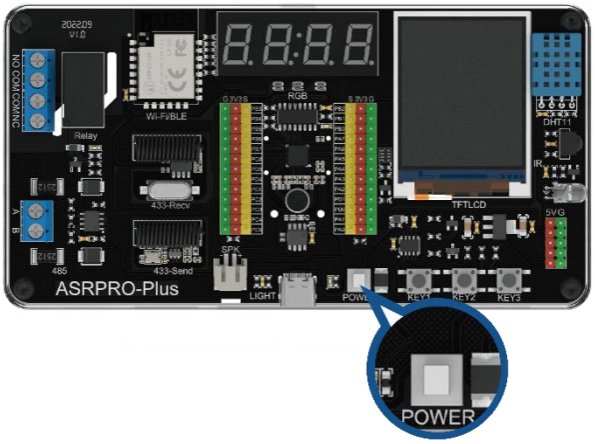
第三步：上电后，会播报欢迎词“欢迎使用智能管家，用智能管家唤醒我”，接下来你可以用“智能管家”唤醒词唤醒设备，接着用不同的命令词控制设备，详见下表。

类型	识别词	回复语音
欢迎词	欢迎使用智能管家，用智能管家唤醒我	
退出语音	我退下了，用智能管家唤醒我	
唤醒词	智能管家	我在
命令词	打开灯光	好的，灯光已打开
命令词	关闭灯光	好的，灯光已关闭

ASRPRO-Plus 开机测试

第一步：使用 Type-C 数据线将开发板连接到电脑上

第二步：打开开发板上的电源开关 POWER，此时旁边指示灯亮起



第三步：上电后，会播报欢迎词“欢迎使用语音助手，用天问五幺唤醒我”，接下来你可以用“天问五幺”唤醒词唤醒设备，接着用不同的命令词控制设备，详见下表。

类型	识别词	回复语音	备注
唤醒词	天问五幺	我在	
命令词	打开灯光	好的，马上打开灯光	打开板载 RGB 灯、继电器、发送无线开关命令 1
命令词	关闭灯光	好的，马上关闭灯光	关闭板载 RGB 灯、继电器、发送无线开关命令 2
命令词	当前天气	播报当前城市天气情况	需要 ESP32 模块联网，默认热点名称：Twen，密码：12345678
命令词	当前时间	播报当前网络时间	需要 ESP32 模块联网，默认热点名称：Twen，密码：12345678
命令词	当前温度	播报板载 DHT11 温度	需要完全断电复位
命令词	当前湿度	播报板载 DHT11 湿度	需要完全断电复位
命令词	当前亮度	播报板载光敏值	
命令词	打开电视	小米电视已打开	红外发送小米电视电源键命令
命令词	关闭电视	小米电视已关闭	红外发送小米电视电源键命令

命令词	打开一号继电器	马上执行	485 控制的 4 路继电器模块（MODBUS 协议）
命令词	关闭一号继电器	马上执行	
命令词	打开二号继电器	马上执行	
命令词	关闭二号继电器	马上执行	
命令词	打开三号继电器	马上执行	
命令词	关闭三号继电器	马上执行	
命令词	打开四号继电器	马上执行	
命令词	关闭四号继电器	马上执行	
命令词	打开所有继电器	马上执行	
命令词	关闭所有继电器	马上执行	

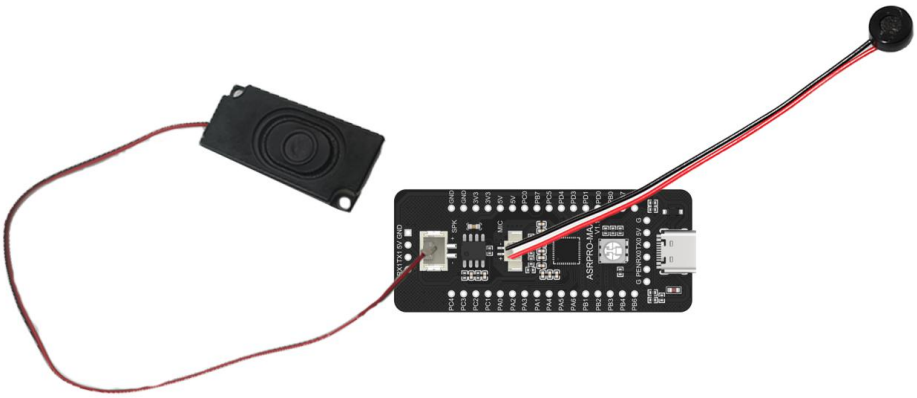
第四步：板载按键控制说明如下表

KEY1	KEY2	KEY3
控制板载继电器打开	控制板载继电器关闭	控制彩屏背光

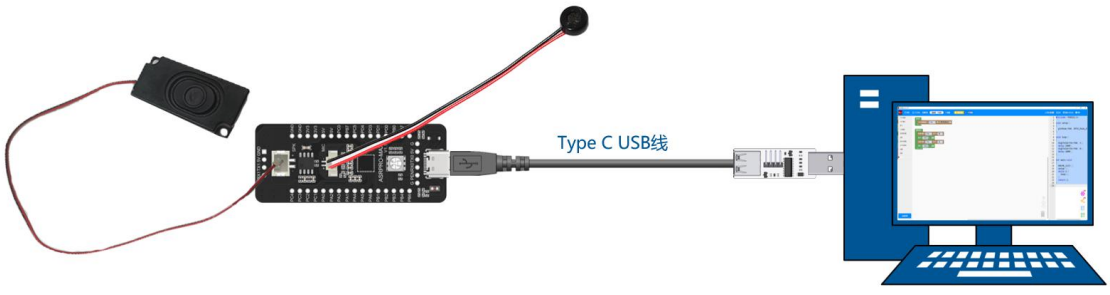
ASRPRO-MAX 开机测试

第一步：连接麦克风和喇叭到开发板上

注意麦克风极性，红色线为正，黑色线为负，不要插反。



第二步： 用 STC-Link 连接 ASRPRO-MAX 到电脑，给开发板供电。



第三步：上电后，会播报欢迎词“欢迎使用语音助手，用天问五幺唤醒我”，接下来你可以用“天问五幺”唤醒词唤醒设备，接着用不同的命令词控制设备，详见下表。

类型	识别词	回复语音
欢迎词	欢迎使用智能管家，用天问五幺唤醒我	
退出语音	我退下了，用天问五幺唤醒我	
唤醒词	天问五幺	我在
命令词	打开灯光	好的，马上执行
命令词	关闭灯光	好的，马上执行

三、下载与安装天问 Block 软件

下载软件

1.浏览器打开天问官方网站 http://twen51.com/。
2.点击天问 Block 下载

安装软件

根据提示默认安装，注意安装过程中，根据提示安装 CH340 驱动。

四、运行天问 Block 软件

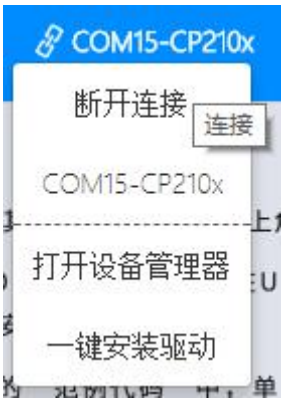
选择主板

第一次打开软件，会让你选择主板，请选择 ASRPRO

ASRPRO

检查串口连接

检查串口是否连接成功，如果未显示驱动可以一键安装驱动。



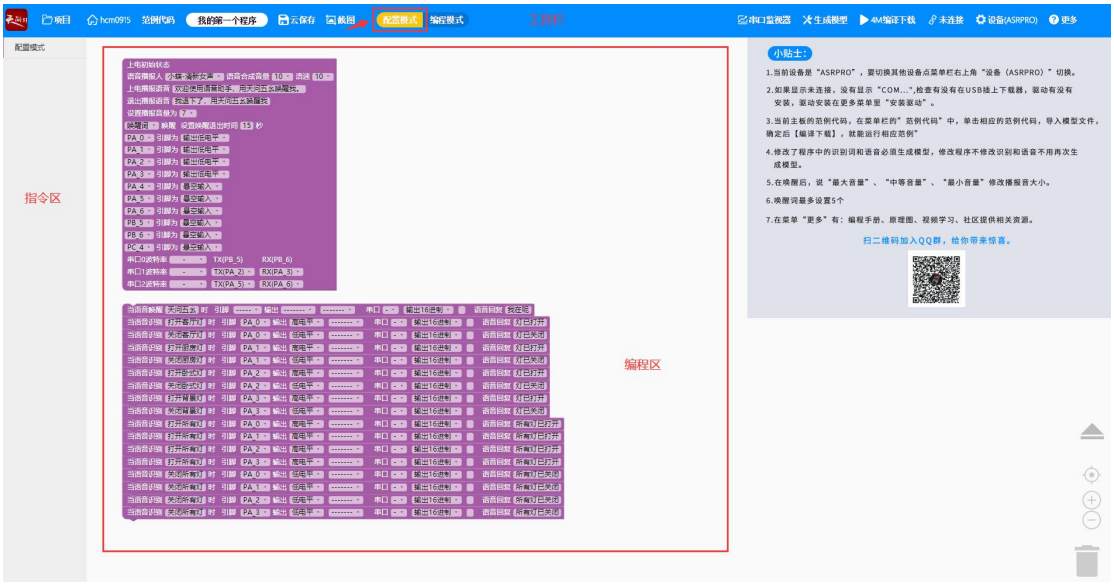
使用 STC-LINK 下载器

使用板载 CH340 下载芯片

选择开发模式

本教程是配置模式教程，因此需要切换成配置模式

界面说明



在配置模式下，页面总共分为 3 个部分，工具栏、指令区和编程区

工具栏：有最基本的文件操作、撤消、重做图标，还可直接打开范例代码进行编译下载，还有串口监视器、生成模型、编译下载等图标，每个图标对应操作的一个功能。还可进行登录个人账号，云保存程序等操作。在“更多”中还可查看编程手册、原理图、学习视频、设置等功能。

指令区：包含了配置模式的最基本指令

编程区：将图形化指令拖拽至编程区进行合理修改组合编程

五、运行程序

打开范例程序

打开范例代码 1.智能语音对话，在跳出的对话框“是否导入并覆盖模型文件”选择确定

设置编译下载模式

ASRPRO 基础开发板和核心板默认采用 ASRPRO 2M 的芯片，即芯片 FLASH 容量是 2M，具体可查看开发板上芯片标注，需选择 2M 下载模式。

ASRPRO-Plus 采用 ASRPRO 4M 的芯片，即芯片 FLASH 容量是 4M，可以选择 4M 下载模式也可选择 2M 下载模式，当程序比较大时，需选择 4M 下载模式。

在更多-设置-编译模式中进行 2M 编译下载和 4M 编译下载切换，本教程主要以 ASRPRO-Plus 展开讲解，所以选择 4M 编译下载。

编译下载范例程序

范例代码都已经生成模型，导入模型后，直接点击编译下载即可。

修改程序

修改程序，如修改唤醒词为好好搭搭

当修改了语音相关设置时，需要重新生成模型

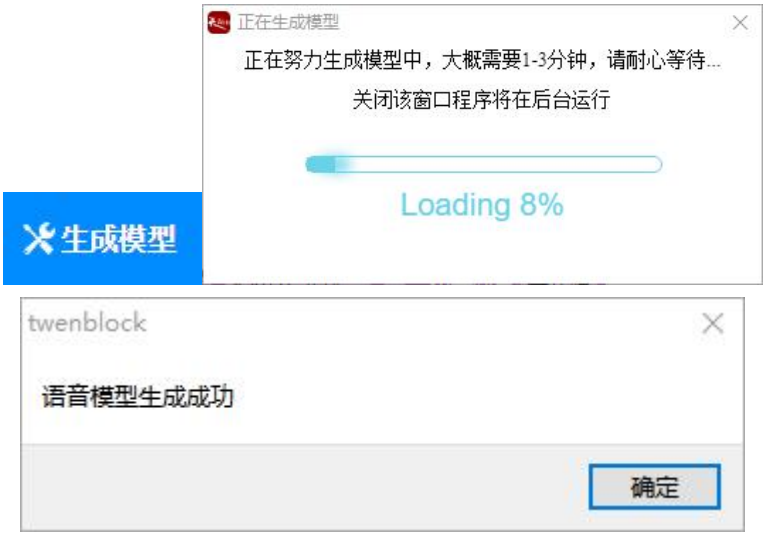
登录账号与实名认证

在使用生成模型功能时，需要登录账号（没有账号可进行免费注册）并进行实名认证

在更多中点击实名认证，输入手机号码以及验证码即可实名成功，注意一个号码只能绑定一个账号

生成模型与编译下载

点击生成模型



点击编译下载

六、云保存与本地保存

云保存

在登录账号的状态下点击工具栏的云保存，根据需要选择保存分享操作。

在菜单栏-项目-项目中心-我的项目中，即可查看到云保存的项目，可随时打开

本地保存

1.在菜单栏-项目-保存（图形文件），选择路径进行保存，注意保存的文件下次打开编译下载前需要重新生成模型

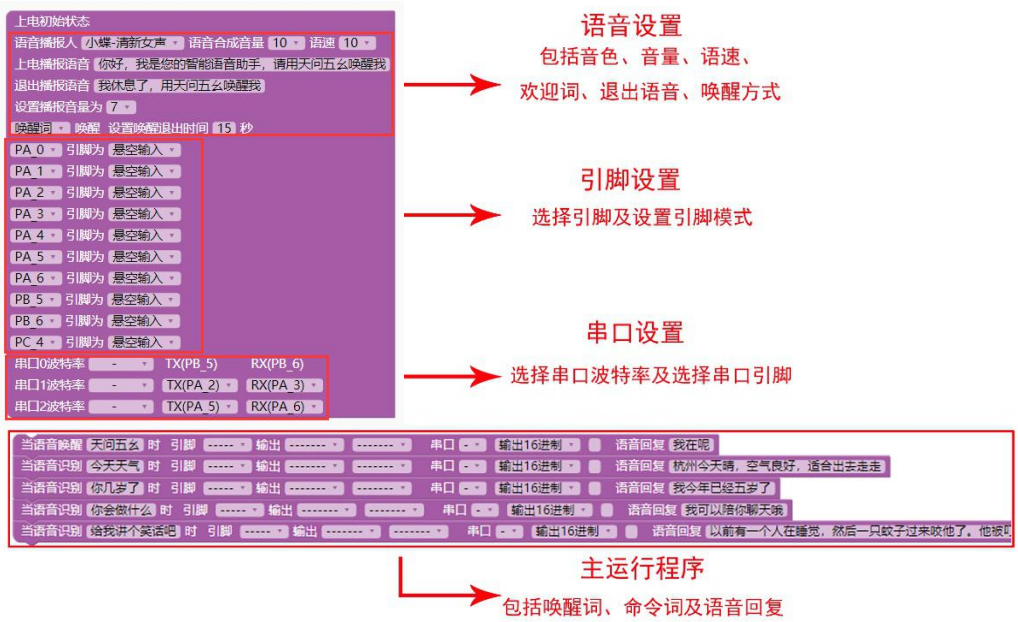
2.在菜单栏-项目-项目保存（含模型），选择路径进行保存，保存的文件下次打开可以直接编译下载

范例 1.1 智能语音对话

一、功能简介

本范例通过学习如何修改语音模型（唤醒词、命令词及回复词），实现智能的语音对话功能，达到用户可以自定义对话内容的目的。

二、范例分析

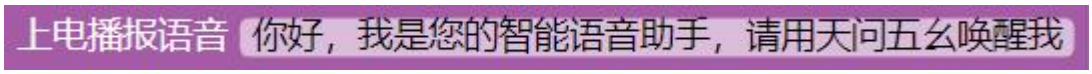


上电初始状态：打开开发板时，每个引脚的初始状态。

三、具体指令讲解

1.上电初始状态设置

用于上电初始状态下语音播报人的相关参数。参数 1 用于设置语音播放人的音色，可选择 39 种不同的音色，（包含 26 种中文 13 种英文）；参数 2 设置语音合成音量，可选择 10 个级别音量大小(注意这里是指生成的 MP3 文件音量)；参数 3 用于调整语音播放的快慢。



用于设置上电时 ASRPRO 自动播放的语音内容。可在输入框内自由修改文字内容，输入内容为中文模式时语音内容需要全部为中文、英文模式时语音内容需要全部为英文，使用英文模式可参考编程模式下的英文案例，英文模式时输入的文字内容不能包含汉字。上电播

报语音内容可设置为空。

用于设置退出工作状态时自动播报的退出语音内容。可在输入框内自由修改文字内容，输入内容为中文模式时语音内容需要全部为中文、英文模式时语音内容需要全部为英文，使用英文模式可参考编程模式下的案例，英文模式时输入的文字内容不能包含汉字。退出播报语音内容可设置为空。

用于设置播报音量范围 1-7，注意这里语音播报音量设置的 DAC 输出，不要与语音合成音量混淆。当语音播报音量很小时推荐优先设置增大播报音量，若增大语音播报音量后还是音量很小，再将语音合成音量调大（推荐 18，过大影响音质）。

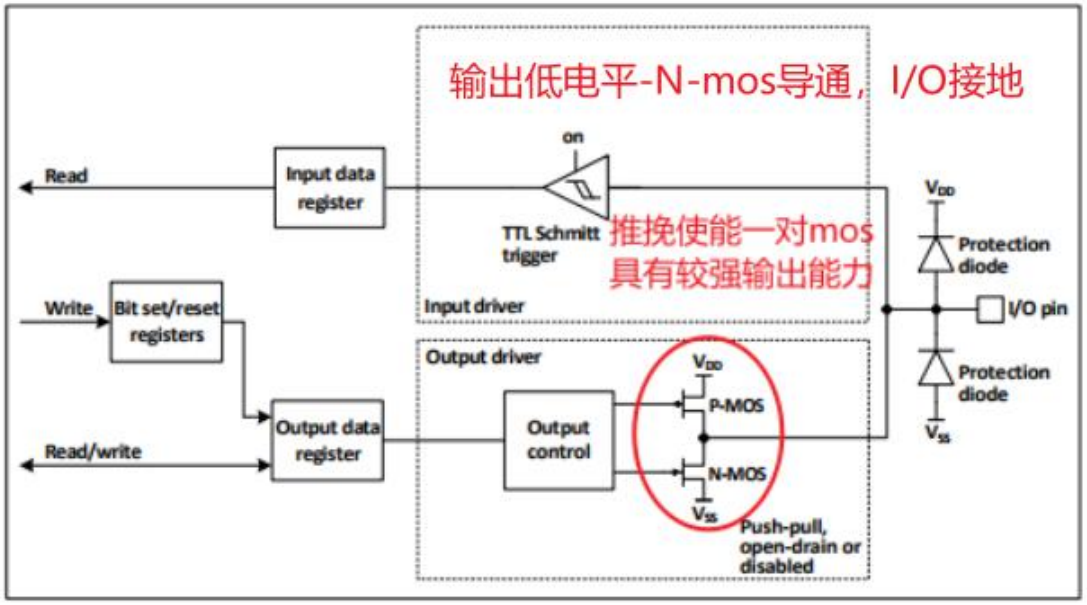
用于设置唤醒状态，以及唤醒的退出时间。可选唤醒词唤醒和永远唤醒两种，其中永远唤醒即为不需要唤醒，可直接使用命令词。

```
extern void set_state_enter_wakeup(uint32_t exit_wakup_ms);
```

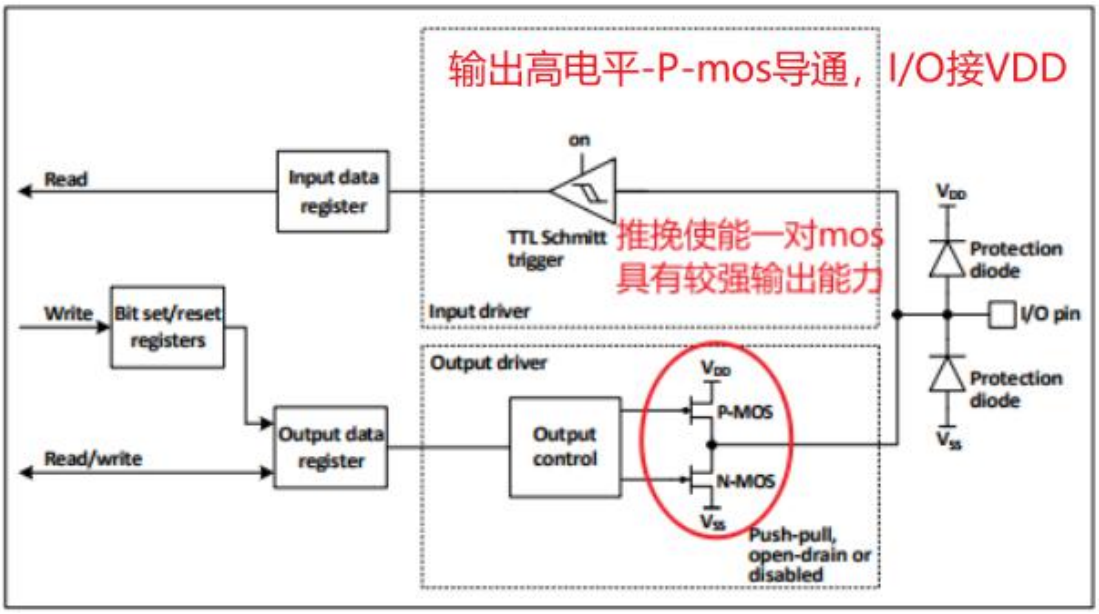
设置唤醒退出时间最多为 2^{32} 毫秒，即 4294967296 毫秒，4294967.296 秒。（传入参数为 32 位无符号整型）

用于设置 PA0-PD5 28 个引脚的状态，分别为输出低电平、输出高电平、上拉输入，下拉输入、悬空输入。（本范例没有用到引脚，所以全部设置悬空输入）

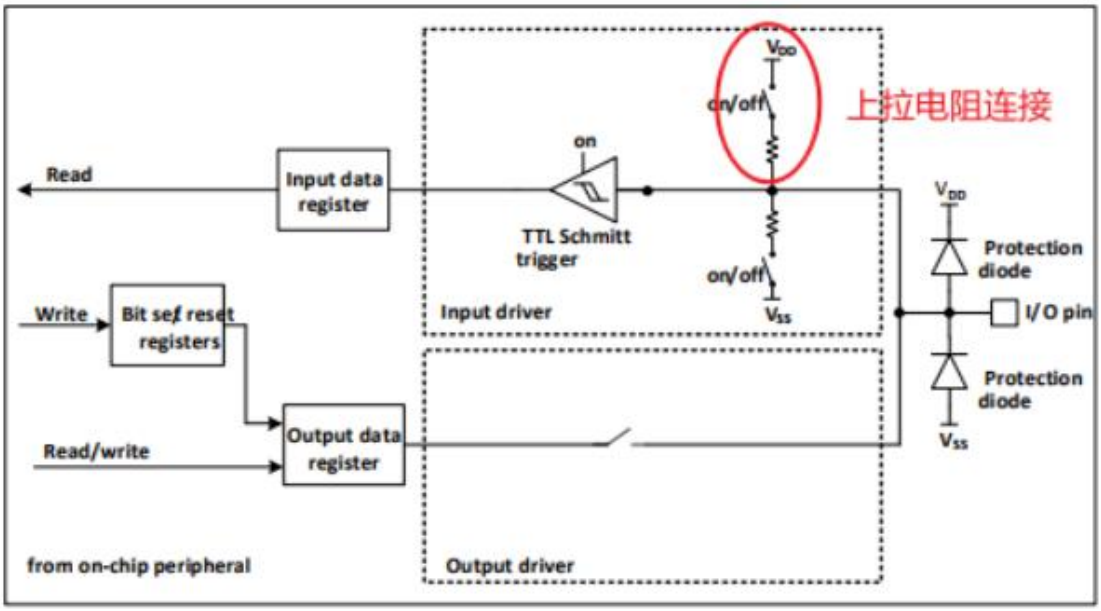
输出低电平：Npos 管导通，io 口接地，因此电平为低（VDD 一般为输入电源正极，VSS 一般为电源地）。



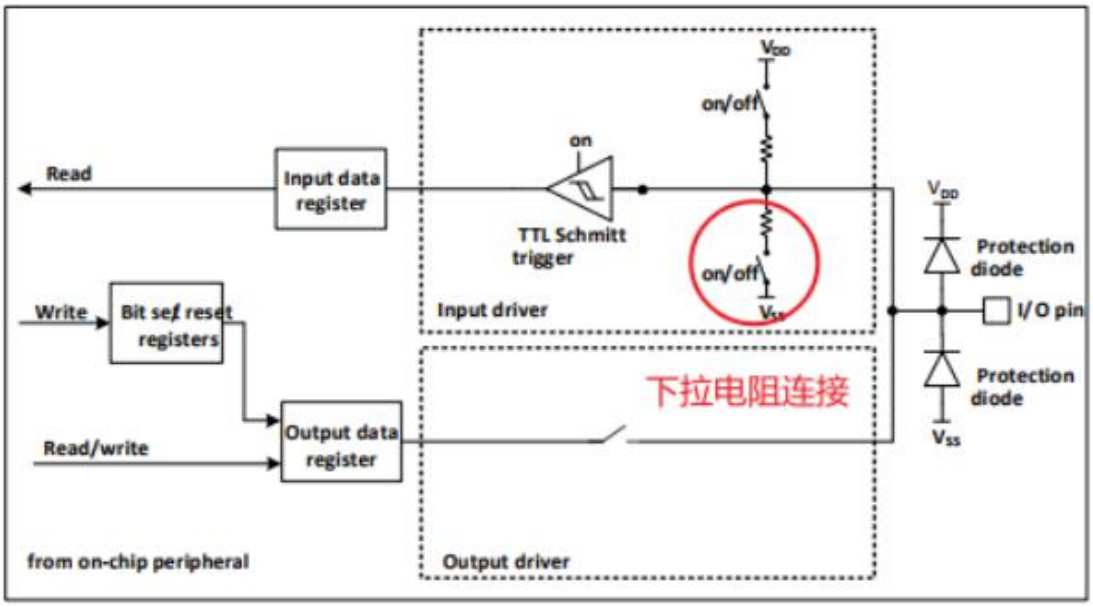
输出高电平：Ppos 管导通，io 口接 VDD，因此电平为高。



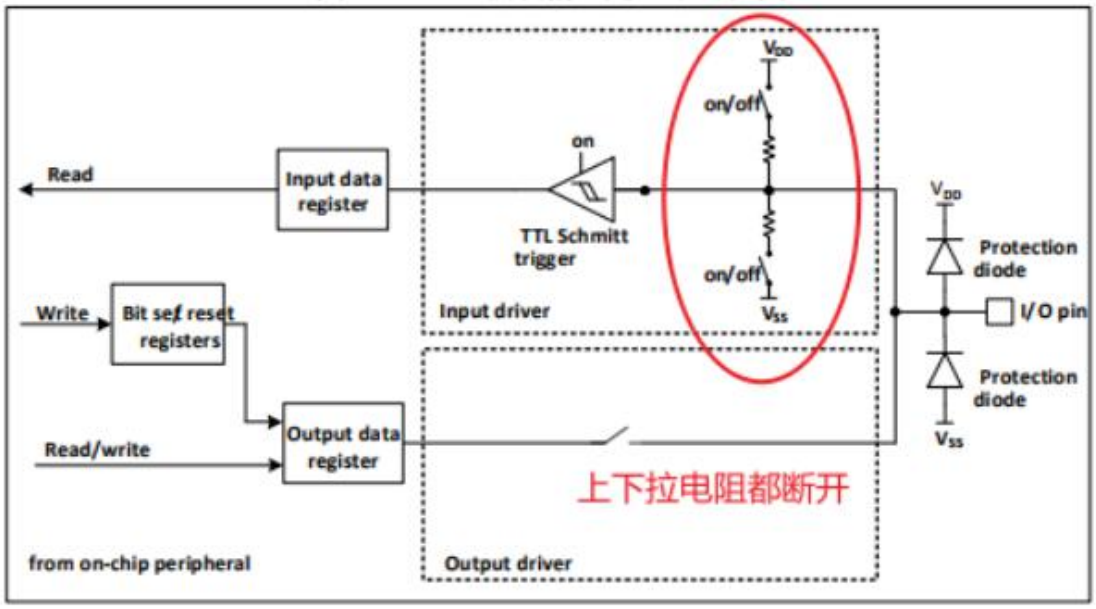
上拉输入：内部上拉电阻连接到 IO，因此 IO 口电平为高。



下拉输入：内部下拉电阻连接到 IO，因此 IO 口电平为低。



悬空输入：内部上下拉电阻都断开，因此空闲时的电平是不确定的。



可设置 3 个串口波特率，需要注意对应的引脚。(本范例没有用到串口，所以为空)

2.主程序设置

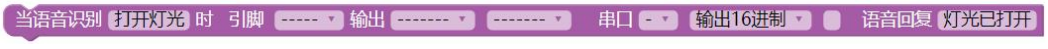
指令说明

智能语音对话使用到的 2 个模块，分别是唤醒模块和命令模块。

用户和语音芯片交互过程：先通过唤醒词唤醒设备，再通过命令词控制设备动作，同时回复对应的回复语（回复语可以为空）。

唤醒模块：可设置唤醒词。

唤醒词是指将产品从待机状态切换到工作状态的词语，可以有效防止误触发。唤醒词最多 5 个。语音回复可以为空。(唤醒词格式参考附录)



命令模块：可设置命令词。

命令词是指用户对语音互动产品发出一定的指令，以此与其进行沟通的词语。根据芯片容量的不同，最大可以设置 300 个。语音回复可以为空。(命令词格式参考附录)

程序实现

使用唤醒模块与命令模块结合。制作简单对话，制作如下图的对话：

3.程序修改

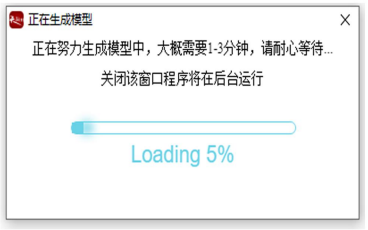
在范例的基础上修改“天问五幺”为其他唤醒词，或添加新的唤醒词模块，如设置唤醒词为“智能管家”、“小爱同学”等等你喜欢的名称；其回复的语音内容也可进行修改（也可为空）。

修改“今天天气”为其他命令词，或添加新的命令词模块，修改命令词为“今天温度”、“你喜欢什么”等等需要的命令；其回复的语音内容也可进行修改（可以为空）。

四、下载运行

1.范例代码可以编译下载，点击右上角 4M 编译下载

2.修改过语音相关内容都需重新生成模型才可以编译下载。点击生成模型，生成模型完成后，再点击 4M 编译下载。（注意：生成模型需要注册账号，并实名认证）



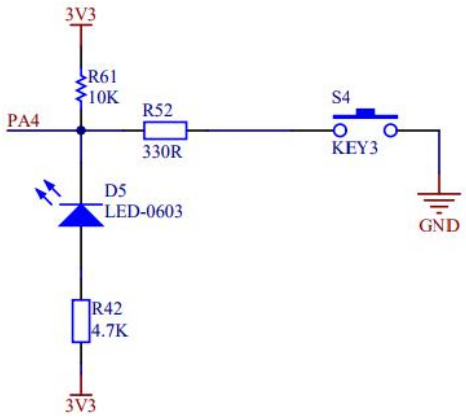
范例 1.2 语音控制板载 LED

一、功能简介

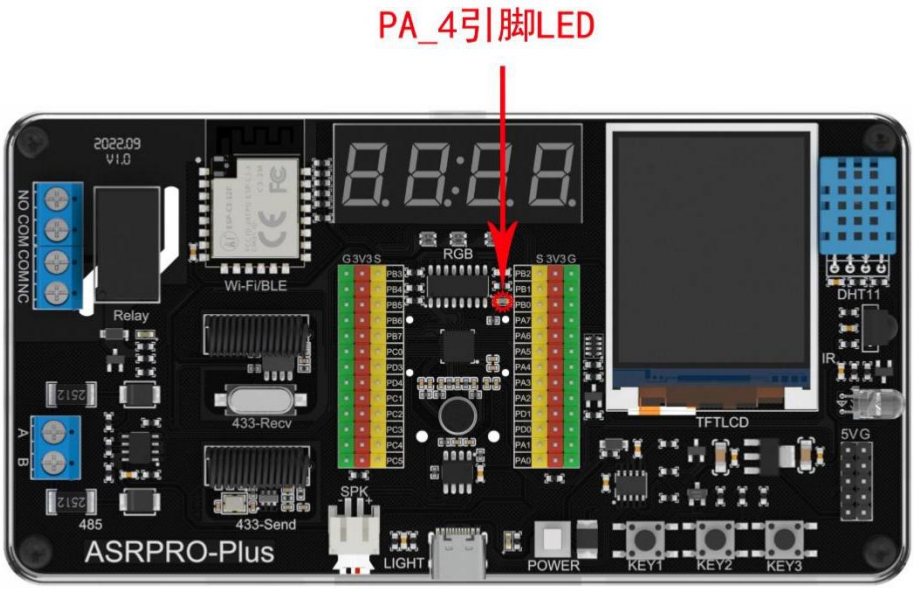
本范例通过学习如何添加命令词的执行动作，实现语音控制板载 LED 灯的功能，达到用户可以用语音同时控制单个设备的目的。

二、范例分析

三、电路与实物说明

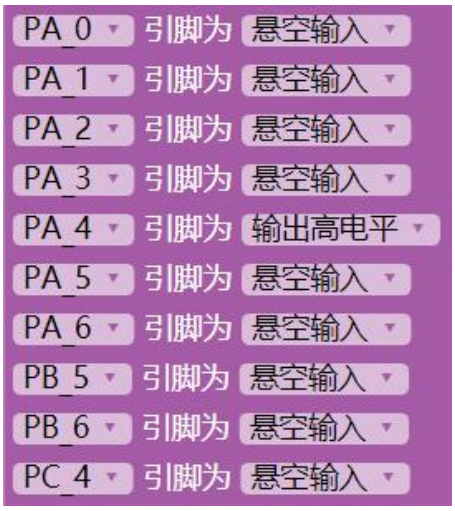


注：当我们查看板载灯的原理图时我们发现，LED 的正极接的是 3V3 的电源，只有当 PA_4 引脚输出低电平时，两者之间才会产生电压差，LED 才能正向导通并发光；当 PA_4 引脚输出高电平时，没有电流通过，LED 熄灭。



四、具体指令讲解

1.上电初始状态设置



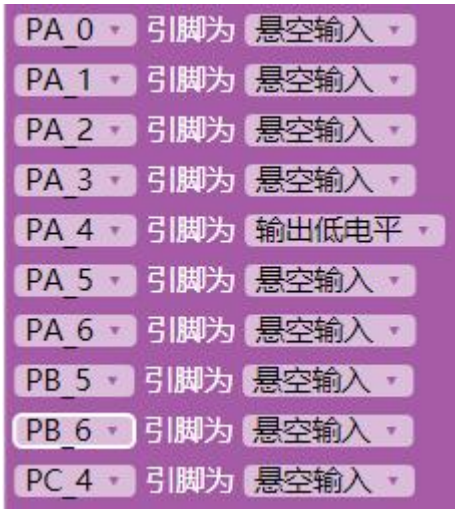
初始设置 PA_4 引脚为输出高电平。

2.主程序设置

控制灯光命令模块设置：选择引脚 PA_4，由于 PA_4 接上拉电阻，所以低电平点亮。这里“打开灯光”设置输出为低电平，“关闭灯光”设置为高电平。

3.程序修改

如果想要在开机时灯是打开的状态，需修改范例中板载灯的初始状态为输出低电平；



主程序增加命令词“灯光变换”，电平输出状态设置电平翻转。

电平翻转是指改变电平状态，如果当前电平为低电平，用电平翻转即修改为高电平，反之，当前是高电平，用电平翻转即修改为低电平。

范例 1.3 语音控制单继电器

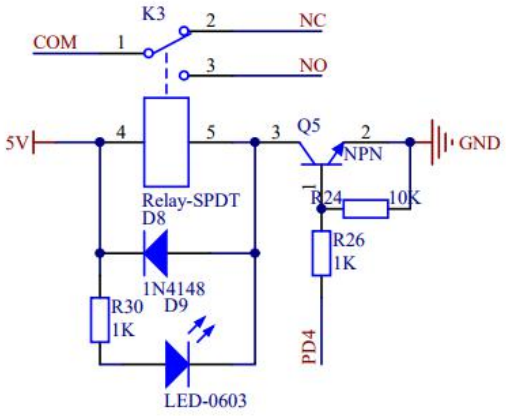
一、功能简介

本范例通过学习如何添加继电器的执行动作，实现语音控制继电器的功能，达到用户可以用语音控制单个设备的目的。

二、范例分析

三、电路与实物说明

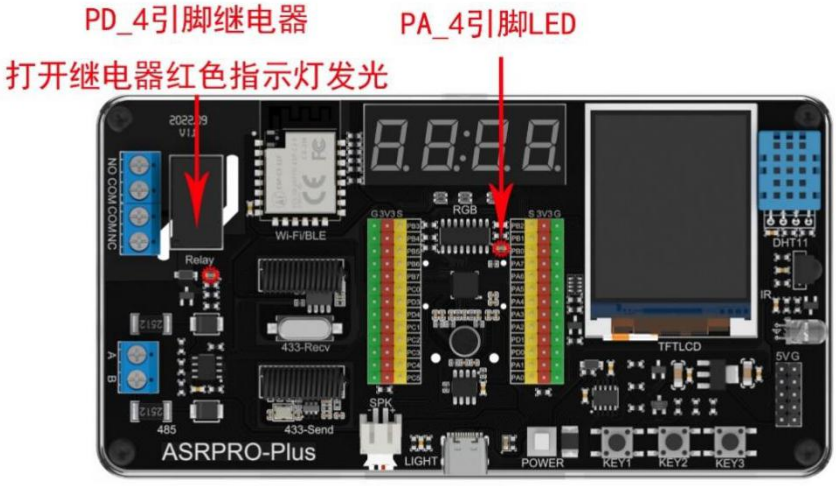
继电器电路原理图如下：

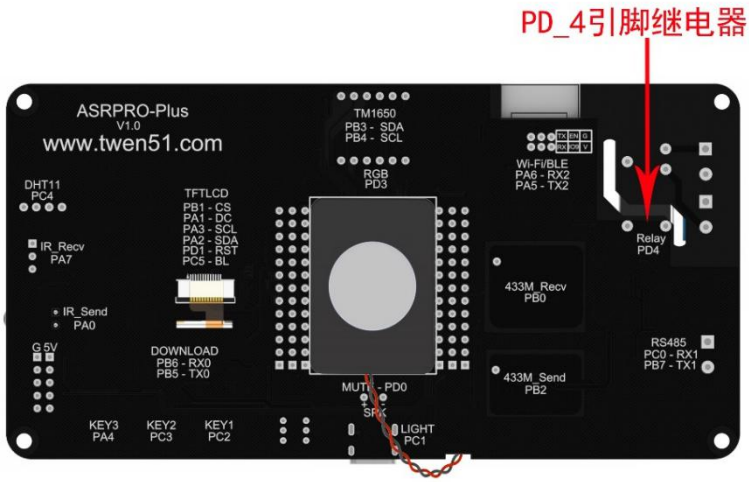


当 NPN 三极管基极被 R24 下拉到电源负极，所以当信号输入端不接或者输入低电平的时候，基极的电压都是 0V，此时基极处于截止状态，集电极和发射极不导通，所以继电器不导通，LED 指示灯不亮；当信号输入端输入高电平，三极管基极也处于高电平，则集电极和发射极导通，继电器吸合，LED 指示灯亮。

继电器外接设备如下图所示：

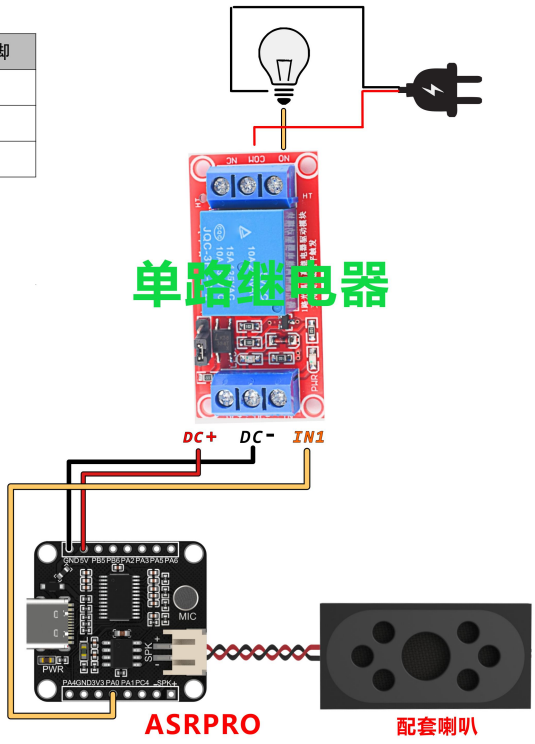
ASRPRO-Plus 实物引脚图如下：





ASRPRO 开发板外接单路继电器示例图如下：

单路继电器	ASRPRO 引脚
DC+	5V
DC-	GND
IN1	PA_0



继电器输出端

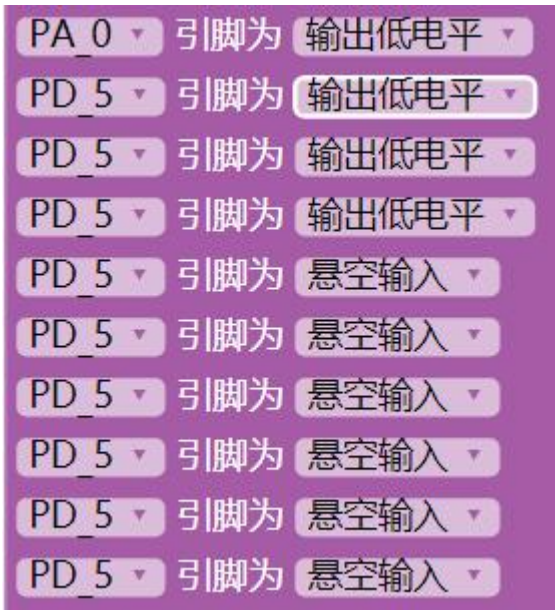
1. NO：继电器常开接口；
2. COM：继电器公共接口；
3. NC：继电器常闭接口；

接口说明

1. DC+：接电源正极（5V）；
2. DC-：接电源负极；
3. IN1：根据每一路的设置均可以高或低电平控制；

四、具体指令讲解

1.上电初始状态设置



可设置 PA0-PD5 28 个引脚的状态，分别为上拉输入，下拉输入、输出低电平、输出高电平、悬空输入。

PA_0 继电器低电平为关闭，所以初始化引脚为低电平。

2.主程序设置

单个继电器控制命令模块设置：

选择引脚 PA_0，语音识别“打开继电器”输出高电平，“关闭继电器”输出低电平。

3.四路以及八路继电器

四个以及八个继电器控制和单个控制原理相同，需要注意的是一个 io 控制一个继电器。

范例 1.4 语音控制四路继电器

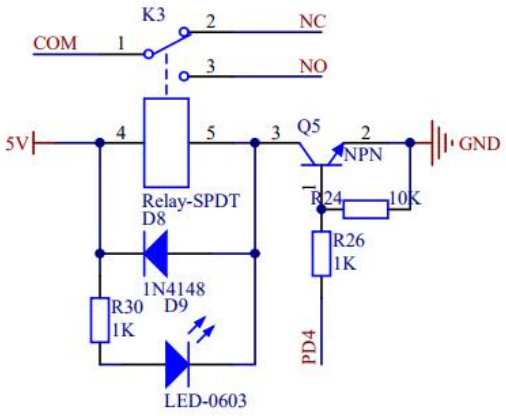
一、功能简介

本范例通过学习如何添加继电器的执行动作，实现语音控制继电器的功能，达到用户可以用语音同时控制多个设备的目的。

二、范例分析

三、电路与实物说明

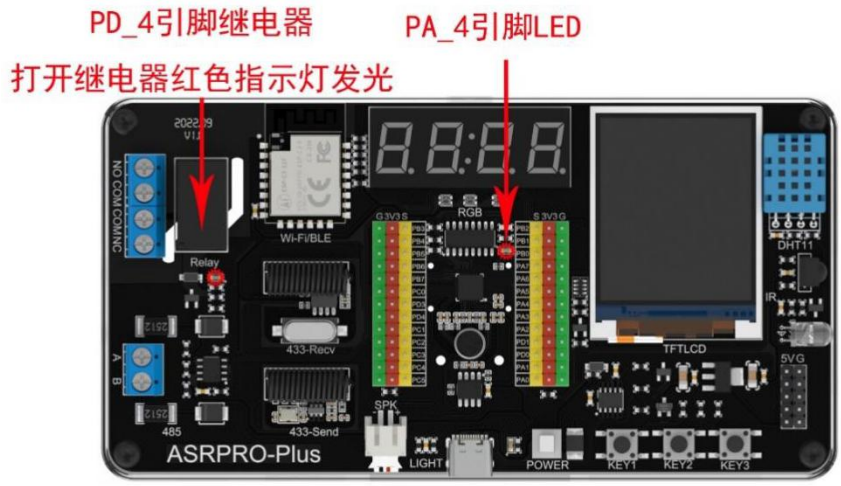
继电器电路原理图如下：

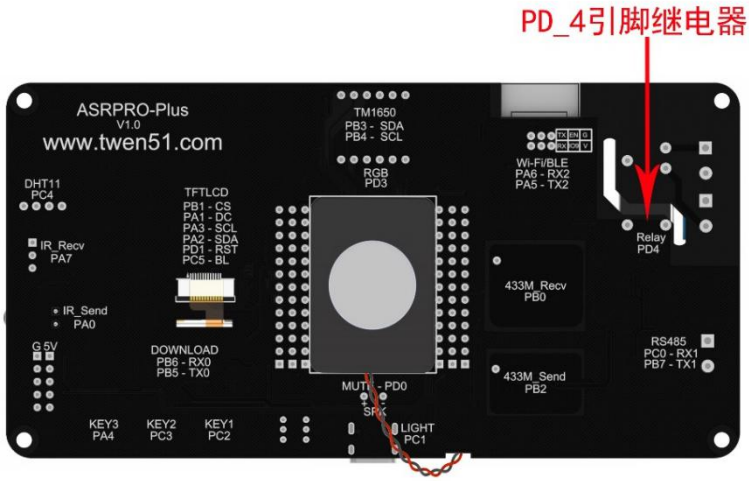


当 NPN 三极管基极被 R24 下拉到电源负极，所以当信号输入端不接或者输入低电平的时候，基极的电压都是 0V，此时基极处于截止状态，集电极和发射极不导通，所以继电器不导通，LED 指示灯不亮；当信号输入端输入高电平，三极管基极也处于高电平，则集电极和发射极导通，继电器吸合，LED 指示灯亮。

继电器外接设备如下图所示：

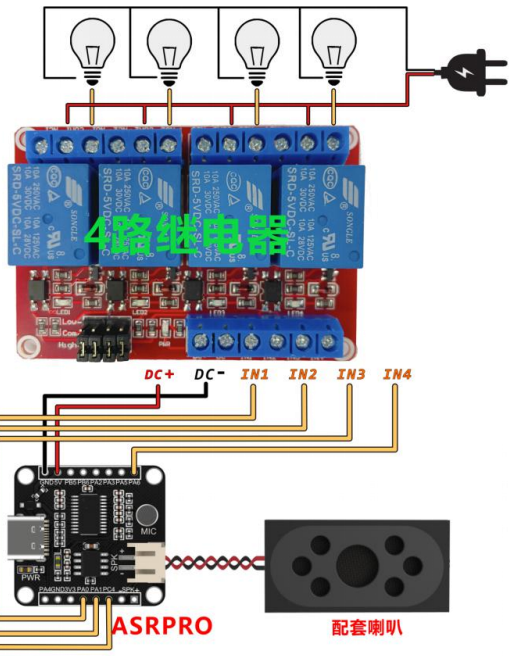
ASRPRO-Plus 实物引脚图如下：





ASRPRO 开发板外接 4 路继电器示例图如下：

4路继电器	ASRPRO 引脚
DC+	5V
DC-	GND
IN1	PA_0
IN2	PA_1
IN3	PC_4
IN4	PA_6



继电器输出端

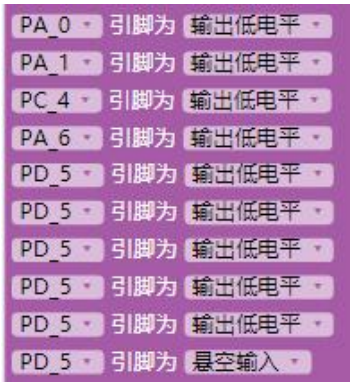
1. NO1-NO4：继电器常开接口；
2. COM1-COM4：继电器公共接口；
3. NC1-NC4：继电器常闭接口；

接口说明

1. DC+：接电源正极（5V）；
2. DC-：接电源负极；
3. IN1-IN4：根据每一路的设置均可以高或低电平控制；

四、具体指令讲解

1.上电初始状态设置



可设置 PA0-PD5 28 个引脚的状态，分别为上拉输入，下拉输入、输出低电平、输出高电平、悬空输入。

PA_0、PA_1、PC_4、PA_6 继电器低电平为关闭，所以初始化引脚为低电平。

2.主程序设置

四路继电器控制命令模块设置：

选择引脚 PA_0，语音识别“打开一号继电器”输出高电平，“关闭一号继电器”输出低电平。

选择引脚 PA_1，语音识别“打开二号继电器”输出高电平，“关闭二号继电器”输出低电平。

选择引脚 PC_4，语音识别“打开三号继电器”输出高电平，“关闭三号继电器”输出低电平。

选择引脚 PA_6，语音识别“打开四号继电器”输出高电平，“关闭四号继电器”输出低电平。

选择引脚 PA_0、PA_1、PC_4、PA_6，语音识别“打开所有继电器”输出高电平，“关闭所有继电器”输出低电平。

范例 1.5 语音控制八路继电器

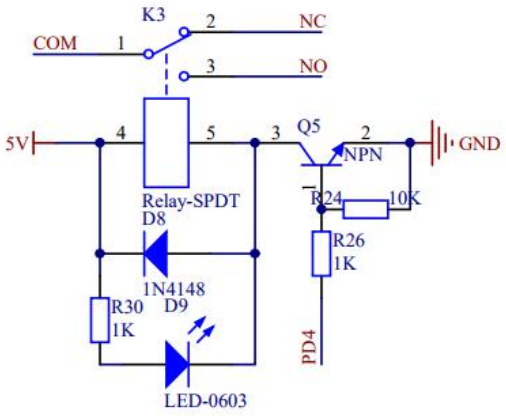
一、功能简介

本范例通过学习如何添加继电器的执行动作，实现语音控制继电器的功能，达到用户可以用语音同时控制多个设备的目的。

二、范例分析

三、电路与实物说明

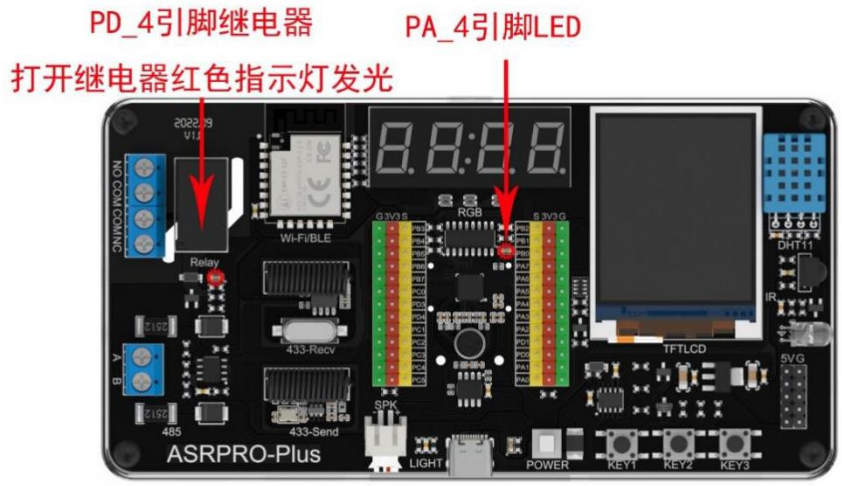
继电器电路原理图如下：

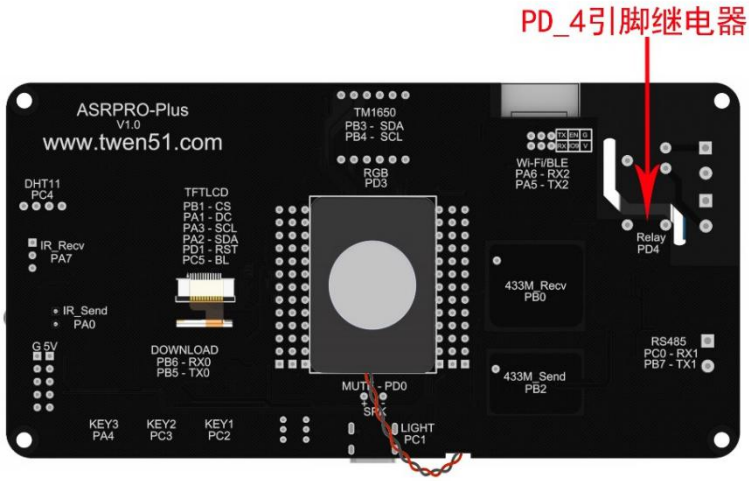


当 NPN 三极管基极被 R24 下拉到电源负极，所以当信号输入端不接或者输入低电平的时候，基极的电压都是 0V，此时基极处于截止状态，集电极和发射极不导通，所以继电器不导通，LED 指示灯不亮；当信号输入端输入高电平，三极管基极也处于高电平，则集电极和发射极导通，继电器吸合，LED 指示灯亮。

继电器外接设备如下图所示：

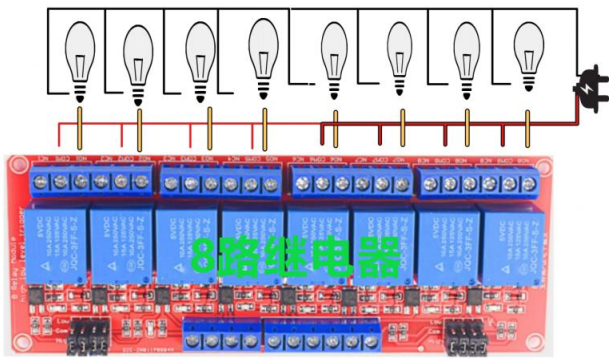
ASRPRO-Plus 实物引脚图如下：





ASRPRO 开发板外接八路继电器示例图如下：

8路继电器	ASRPRO引脚
DC+	5V
DC-	GND
IN1	PA_0
IN2	PA_1
IN3	PC_4
IN4	PA_6
IN5	PA_5
IN6	PA_3
IN7	PA_2
IN8	PB_6

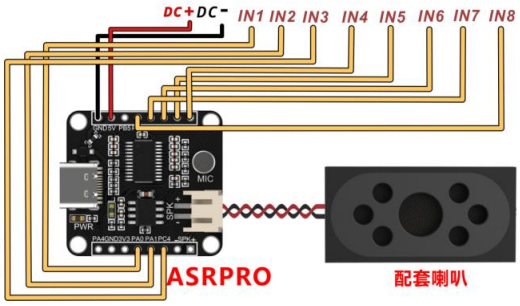


继电器输出端

1. NO1-NO8：继电器常开接口；
2. COM1-COM8：继电器公共接口；
3. NC1-NC8：继电器常闭接口；

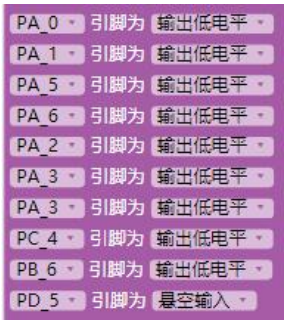
接口说明

1. DC+：接电源正极（5V）；
2. DC-：接电源负极；
3. IN1-IN8：根据每一路的设置均可以高或低电平控制；



四、具体指令讲解

1.上电初始状态设置

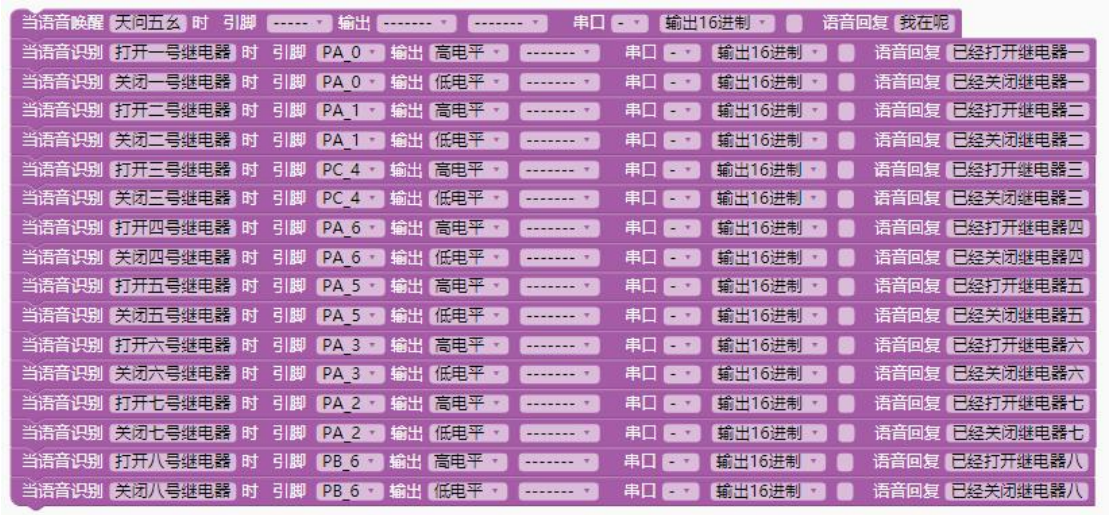


可设置 PA0-PD5 28 个引脚的状态，分别为上拉输入，下拉输入、输出低电平、输出高电平、悬空输入。

PA_0、PA_1、PA_5、PA_6、PA_2、PA_3、PC_4、PB_6 继电器低电平为关闭，所以初始化引脚为低电平。

注意：此时 8 号继电器使用到 PB_6 引脚，此引脚作为下载口引脚，使用下载时继电器应先断开 PB_6 引脚，否则可能出现下载程序时继电器不受控。

2.主程序设置



八路继电器控制命令模块设置：

选择引脚 PA_0，语音识别“打开一号继电器”输出高电平，“关闭一号继电器”输出低电平。

选择引脚 PA_1，语音识别“打开二号继电器”输出高电平，“关闭二号继电器”输出低电平。

选择引脚 PC_4，语音识别“打开三号继电器”输出高电平，“关闭三号继电器”输出低电平。

选择引脚 PA_6，语音识别“打开四号继电器”输出高电平，“关闭四号继电器”输出低电平。

选择引脚 PA_5，语音识别“打开五号继电器”输出高电平，“关闭五号继电器”输出低电平。

选择引脚 PA_3，语音识别“打开六号继电器”输出高电平，“关闭六号继电器”输出低电平。

选择引脚 PA_2，语音识别“打开七号继电器”输出高电平，“关闭七号继电器”输出低电平。

选择引脚 PB_6，语音识别“打开八号继电器”输出高电平，“关闭八号继电器”输出低电平。

范例 1.6 语音控制单继电器 10 秒后关闭

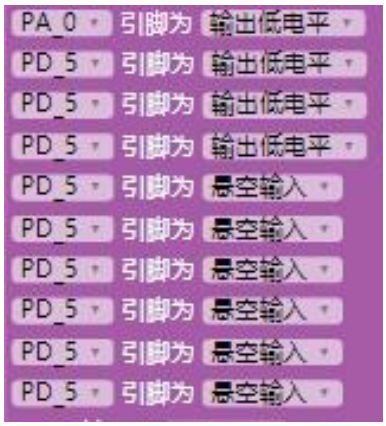
一、功能简介

本范例通过学习语音对话与引脚脉冲状态的设置，实现语音控制继电器延时闭合的功能，达到语音控制引脚脉冲控制继电器延时的目的。

二、范例分析

三、具体指令讲解

1.上电初始状态设置



2.主程序设置

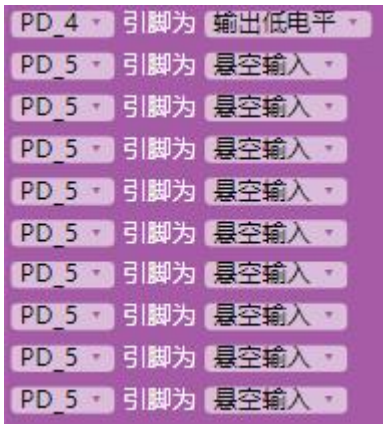
语音控制继电器延时命令模块设置：

语音识别“打开继电器”选择引脚 PA_0，输出高脉冲，可设置脉冲时间（时间范围大于 0 毫秒小于 32767 毫秒）和脉冲个数。

语音识别“关闭继电器”输出低电平。

3.程序修改

PD_4 继电器低电平为关闭，所以初始化引脚为低电平。其余引脚设置悬空状态。



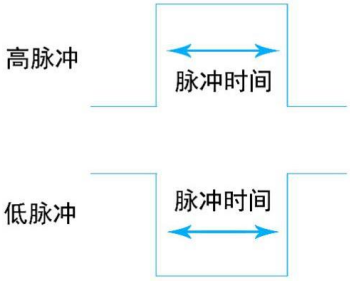
语音控制继电器延时命令模块设置：

语音识别“打开继电器”选择引脚 PD_4，输出高脉冲，脉冲时间 10 秒和脉冲个数 1。

语音识别“关闭继电器”输出低电平。

4.程序设置说明：

从脉冲信号可以看出，本范例初始设置为低电平，而在继电器延时控制中，我们通过输出高脉冲，控制脉冲时间与脉冲个数，达到延时控制的目的。反之，如果我们初始设置高电平，可以通过低脉冲达到延时控制的目的，在设置中根据具体需求进行灵活变动。

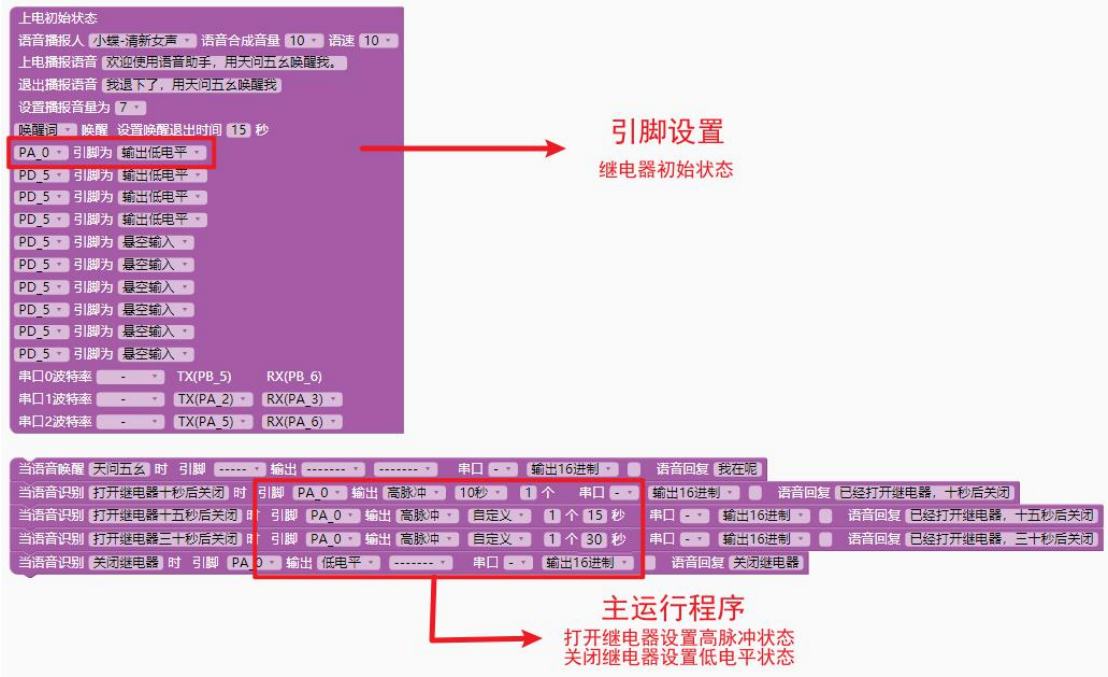


范例 1.7 语音控制继电器延时后关闭

一、功能简介

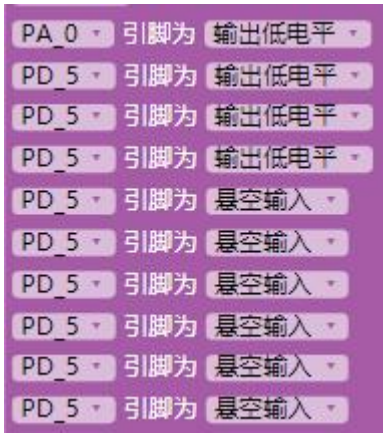
本范例通过学习语音对话与引脚脉冲状态的设置，实现语音控制继电器延时闭合的功能，达到语音控制引脚脉冲控制继电器延时的目的。

二、范例分析



三、具体指令讲解

1.上电初始状态设置



2.主程序设置

语音控制继电器延时命令模块设置：

语音识别“打开继电器十秒后关闭”选择引脚 PA_0，输出高脉冲，脉冲时间 10 秒和脉冲个数 1。

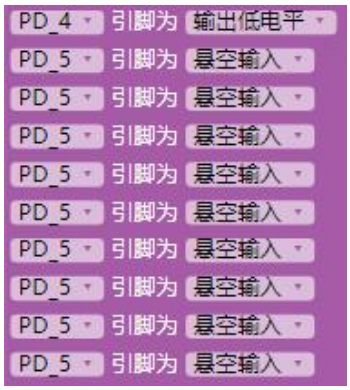
语音识别“打开继电器十五秒后关闭”选择引脚 PA_0，输出高脉冲，脉冲时间 15 秒和脉冲个数 1。

语音识别“打开继电器三十秒后关闭”选择引脚 PA_0，输出高脉冲，脉冲时间 30 秒和脉冲个数 1。

语音识别“关闭继电器”输出低电平。

3.程序修改

PD_4 继电器低电平为关闭，所以初始化引脚为低电平。其余引脚设置悬空状态。



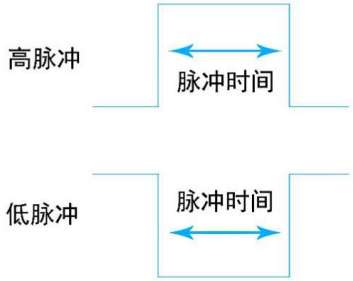
语音控制继电器延时命令模块设置：

语音识别“打开继电器”选择引脚 PD_4，输出高脉冲，脉冲时间 10 秒和脉冲个数 1。

语音识别“关闭继电器”输出低电平。

4.程序设置说明

从脉冲信号可以看出，本范例初始设置为低电平，而在继电器延时控制中，我们通过输出高脉冲，控制脉冲时间与脉冲个数，达到延时控制的目的。反之，如果我们初始设置高电平，可以通过低脉冲达到延时控制的目的，在设置中根据具体需求进行灵活变动。

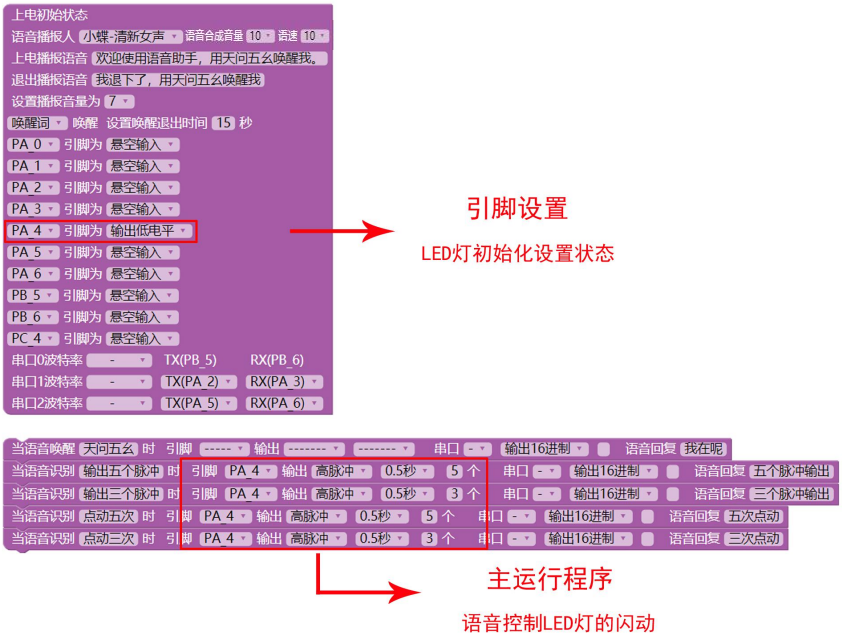


范例 1.8 语音控制引脚（脉冲）点动

一、功能简介

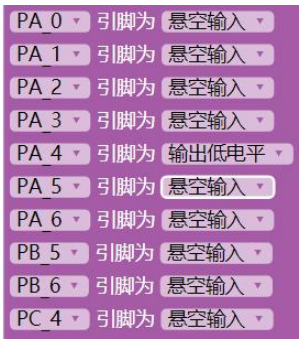
本范例通过学习如何控制引脚的脉冲，实现指定引脚的高低电平延时控制的功能，达到用户可以控制设备脉冲的目的。

二、范例分析



三、具体指令讲解

1.上电初始状态设置



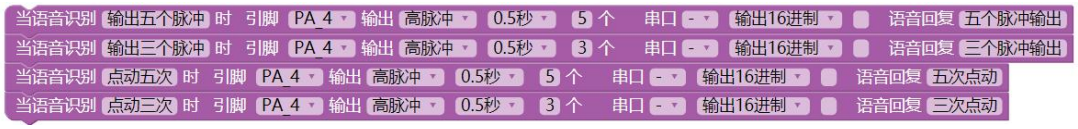
PA_4 引脚 LED 灯输出低电平

2.主程序设置

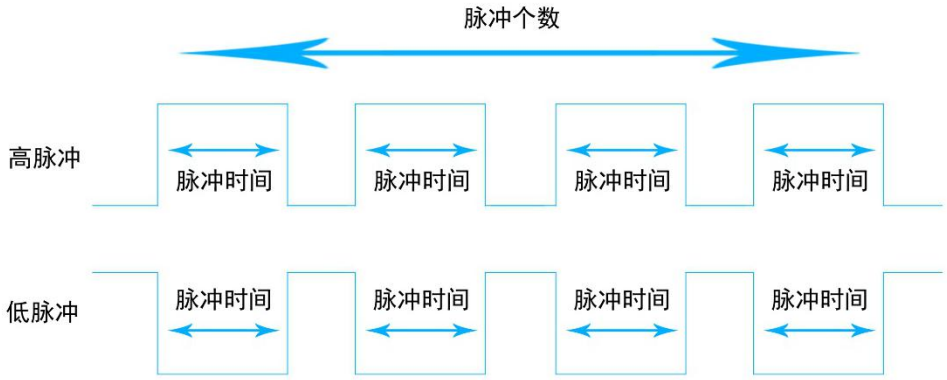
语音控制命令模块设置：

语音识别“输出五个脉冲”或者“点动五次”，控制引脚 PA_4 输出高脉冲，时间 0.5 秒，5 个，回复相应语音。

语音识别“输出三个脉冲”或者“点动三次”，控制引脚 PA_4 输出高脉冲，时间 0.5 秒，3 个，回复相应语音。



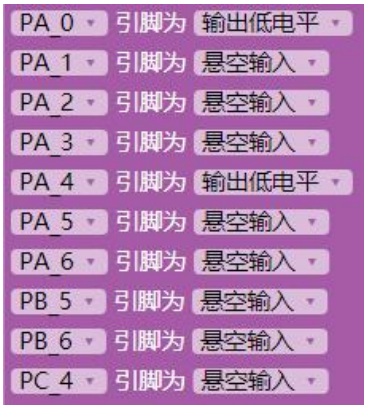
从脉冲信号可以看出，本范例初始设置为低电平，LED 是亮的，我们通过控制脉冲时间与脉冲个数，控制 LED 亮灭状态，实现 LED 灯的闪动。



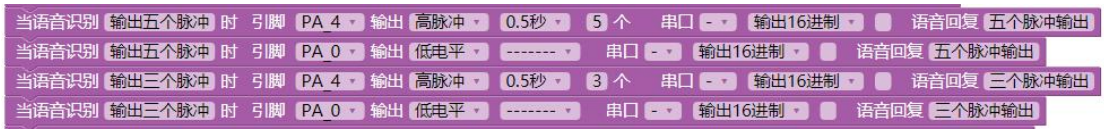
3.程序修改

本范例通过脉冲控制的 LED 的亮灭，下面我们对电机进行脉冲控制的修改。

上电初始状态设置，PA_0 引脚输出低电平，PA_4 引脚输出低电平。



语音控制命令模块设置：语音识别“输出五个脉冲”，控制引脚 PA_4 输出高脉冲，时间 0.5 秒，5 个，PA_0 输出低电平，回复相应语音。（同理输出三个脉冲设置增加引脚 PA_0 输出低电平）



范例 1.9 引脚低电平控制播报

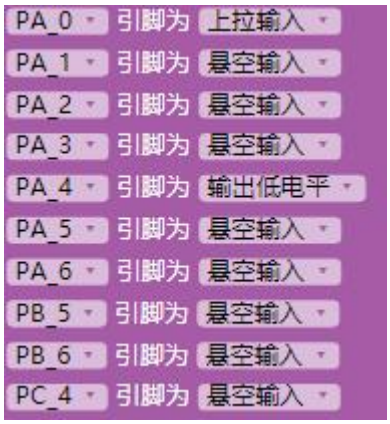
一、功能简介

本范例通过学习低电平控制播报语音，实现指定引脚的高低电平控制语音播报的功能。

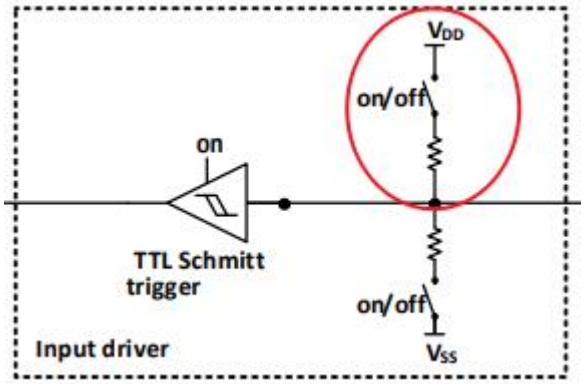
二、范例分析

三、具体指令讲解

1.上电初始状态设置

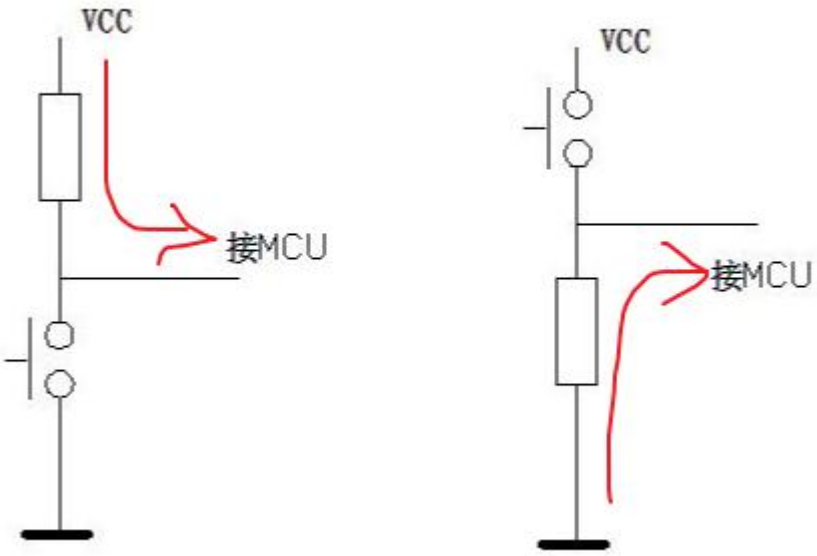


GPIO 内部上拉：



PA0 上电初始化为上拉输入，内部上拉电阻连接到 VDD，当没有外部输入时，引脚电平为高。

扩展：按键输入与上下拉



第一种：由于存在上拉电阻，空闲时 MCU 检测此引脚为高，当按键按下时，引脚直接接地 MCU 检测电平为低。因此，使用内部上拉电阻时，MCU 需要检查引脚是否出现低电平，低电平出现即按键按下。

第二种：由于存在下拉电阻，空闲时 MCU 检测此引脚为低，当按键按下时，引脚直接接 VCC，MCU 检测电平为高。因此，使用内部下拉电阻时，MCU 需要检查引脚是否出现高电平，高电平出现即按键按下。

2.主程序设置

PA0 上拉输入，PA0 电平为高，当 PA0 为低电平时，控制语音输出“按下了低电平”。

3.程序修改

PA5 设置为上拉输入，电平为高，当 PA5 的电平为低时，语音播报“按下了低电平”。

范例 1.10 引脚高电平控制播报

一、功能简介

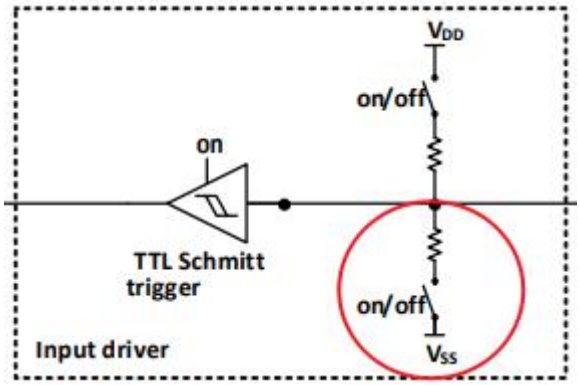
本范例通过学习高电平控制播报语音，实现指定引脚的高低电平控制语音播报的功能。

二、范例分析

三、具体指令讲解

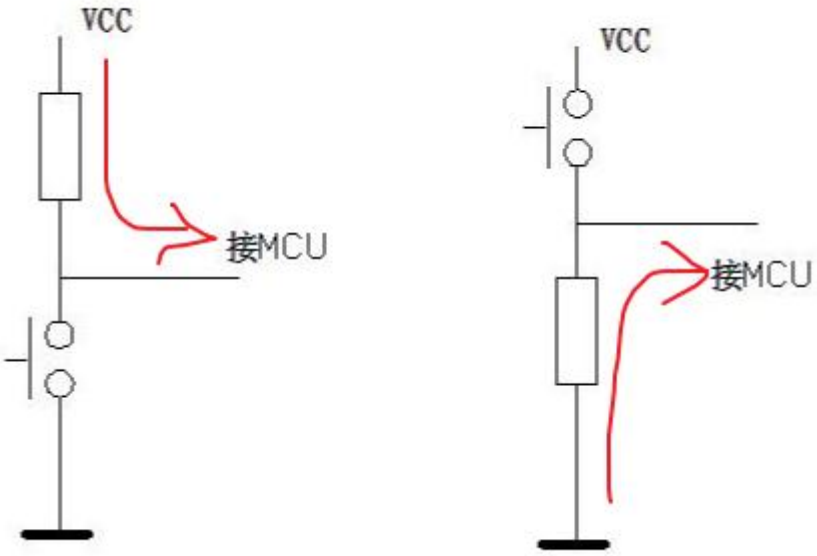
1.上电初始状态设置

GPIO 内部下拉：



PA0 上电初始化为下拉输入，下拉电阻连接到 VSS，当没有外部输入时，引脚电平为低。

扩展：按键输入与上下拉



第一种：由于存在上拉电阻，空闲时 MCU 检测此引脚为高，当按键按下时，引脚直接接地 MCU 检测电平为低。因此，使用内部上拉电阻时，MCU 需要检查引脚是否出现低电平，低电平出现即按键按下。

第二种：由于存在下拉电阻，空闲时 MCU 检测此引脚为低，当按键按下时，引脚直接接 VCC，MCU 检测电平为高。因此，使用内部下拉电阻时，MCU 需要检查引脚是否出现高电平，高电平出现即按键按下。

2.主程序设置

PA0 下拉输入，PA0 电平为低，当 PA0 为高电平时，控制语音输出“按下了高电平”。

3.程序修改

PA5 设置为下拉输入，电平为低，当 PA5 的电平为高时，语音播报“按下了高电平”。

范例 2.1 串口 0 输出字符串

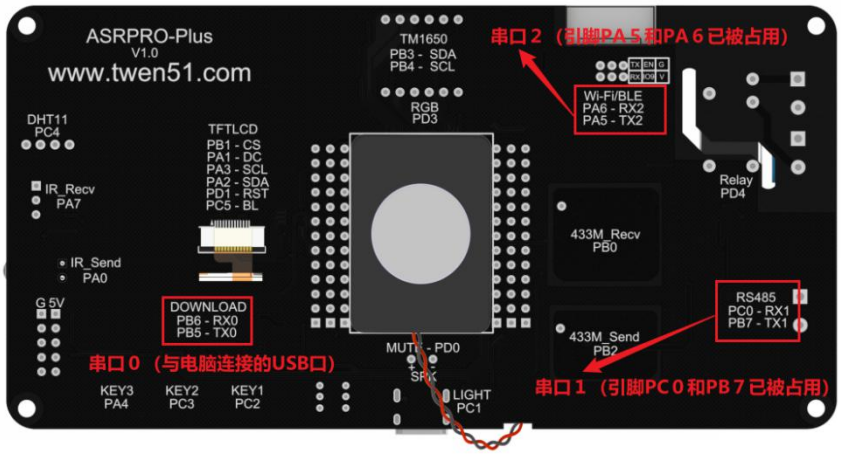
一、功能简介

本范例通过学习如何使用串口自动发送字符串数据，实现串口输出字符串的功能，达到用户可以正确使用串口通讯的目的。

二、范例分析

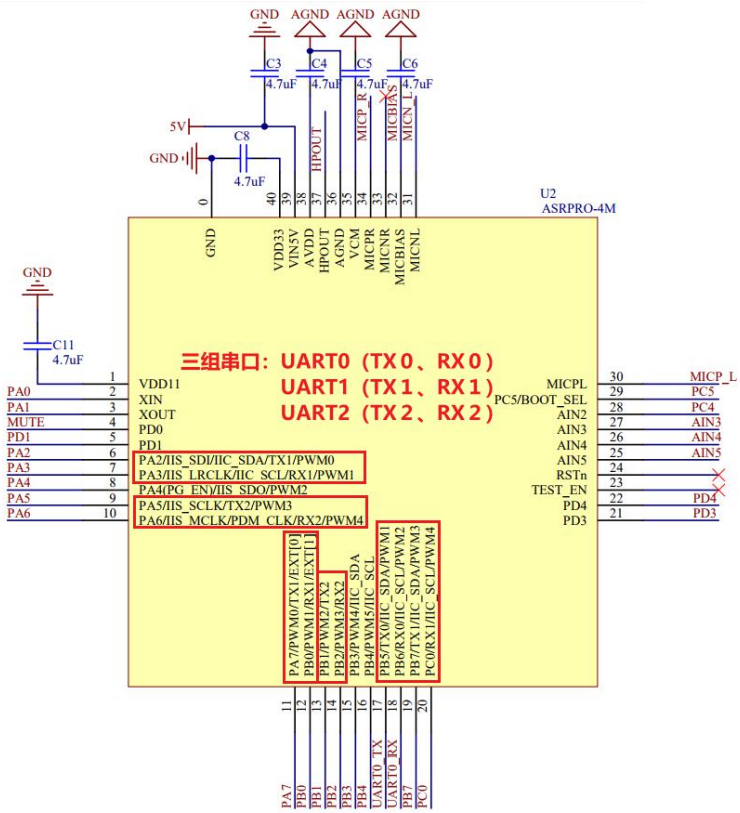
三、电路与实物说明

ASRPRO-Plus 开发板串口所处位置如下图所示：



可以看到具有串口通讯功能的的 PA5、PA6、PC0 和 PB7 引脚已经被占用了，如果要查看串口通讯的数据，可以使用烧写器连接其他具有串口通讯的引脚，并借助串口监视器或 STC-ISP 软件查看不同串口的通讯数据。

芯片内部电路原理图如下图所示：



可以看到 ASRPRO-Plus 开发板的芯片共有三组串口，分别是 UART0、UART1 和 UART2。

UART 是最常见的串行通讯，广泛应用于单片机和单片机之间通讯。比如 WiFi 模块，串口液晶屏等。串口通信经过信号转换，可以进行 RS232、RS422、RS485 通信，广泛应用于设备之间远程通信。

串口通讯采用 2 条通讯线：发送数据线 TX，接收数据线 RX。要实现不同设备之间的数据收发，需要将通讯设备 1 的 TX 与通讯设备 2 的 RX 相连，将通讯设备 1 的 RX 与通讯设备 2 的 TX 相连，就可以同时进行数据的发送和接收。

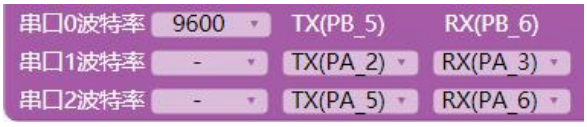
在本范例中，我们需要设置三组串口的引脚位置，串口 0 的位置已经固定（PB5 和 PB6），串口 1 和串口 2 的位置可以通过编程修改确定（避开已被占用引脚）。

四、具体指令讲解

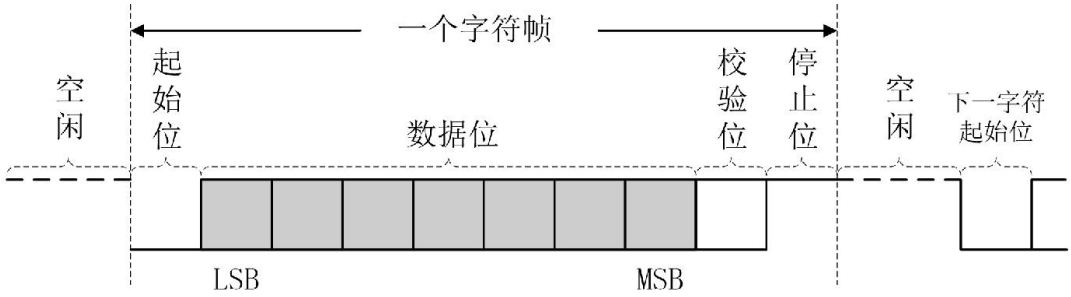
1.上电初始状态设置

PA_4 输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。

可设置 3 个串口波特率。



波特率（BaudRate）表示数据传送速率，即每秒钟传送的二进制位数。波特率通常单位是 bit/s，比较常用的波特率有 9600，57600，115200 等等。



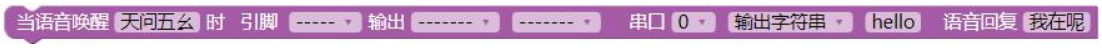
设置串口发送与接收的引脚，其中串口 0 固定发送引脚 PB_5，接收 PB_6，串口 0 的位置是 USB 接口上。串口 1 可以设置的发送引脚有 PA_2、PA_7、PB_7，接收引脚有 PA_3、PB_0、PC_0。串口 2 可以设置的发送引脚有 PA_5、PB_1、PB_6，接收引脚有 PA_6、PB_2。根据具体需要与其它设备进行通讯的情况，进行选择。（在本范例中，我们需要设置一组串口的引脚位置，串口 0 的位置已经固定 PB5 和 PB6，串口 1 和串口 2 的位置可以通过编程

修改确定,避开已被占用引脚）

2.主程序设置

语音控制唤醒词输出 16 进制命令模块设置：

语音识别“天问五幺”，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 hello。



语音控制继电器输出字符串命令模块设置：

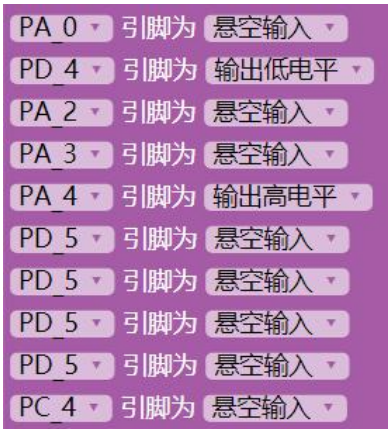
语音识别“打开继电器”选择引脚 PA_4，输出高电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 on。

语音识别“关闭继电器”选择引脚 PA_4，输出低电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 off。

3.程序修改

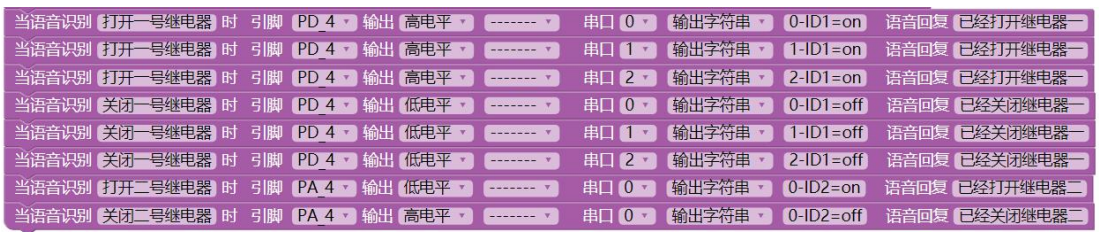
PD_4 继电器输出低电平，PA_4 模拟第二个继电器，输出高电平。

所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。



语音控制继电器输出字符串命令模块设置：

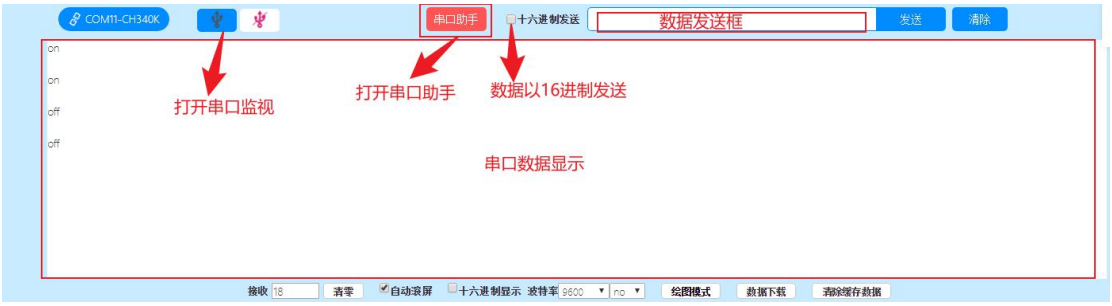
语音识别“打开一号继电器”选择引脚 PD_4，输出高电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 0-ID1=on。语音识别“关闭一号继电器”选择引脚 PD_4，输出低电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 0-ID1=off。（串口 1 和串口 2 同理设置）。



4.串口监视器查看串口 0 输出字符串

打开串口监视器：点击右上角串口监视器按钮

本范例串口监视器设置：COM 端口对应选择主板的 COM 端口，COM 端口打开如图左边按钮显示蓝色为打开状态，波特率对应程序设置，本范例为 9600。



串口 0 输出显示区域：当我们发出语音命令到主板，主板会输出对应字符串并在串口输出显示区域显示。

字符串的本质就是多个字节组成的字符。

如串口发送"hello"，实际是串口发送 5 个字节，第一个字节'h'（16 进制是 0x68)、第二个字节'e'(16 进制是 0x65)、第三个字节'l'(16 进制是 0x6c)、第四个字节'l'(16 进制是 0x6c)、第五个字节'o'(16 进制是 0x6f)。

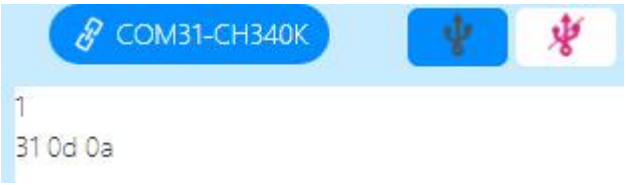
我们可以用串口原始输出 16 进制数，就可在串口助手看到"hello"字符串。

对应参考 ASCII 码表具体如下所示：

Bin	Oct	Dec	Hex	缩写/字符	解释	Bin	Feb	Apr	Hex	缩写/字符	解释
(二进制)	(八进制)	(十进制)	(十六进制)			(二进制)	(八进制)	(十进制)	(十六进制)		
0011 0000	60	48	0x30	0	字符0	0100 1110	116	78	0x4E	N	大写字母N
0011 0001	61	49	0x31	1	字符1	0100 1111	117	79	0x4F	O	大写字母O
0011 0010	62	50	0x32	2	字符2	0101 0000	120	80	0x50	P	大写字母P
0011 0011	63	51	0x33	3	字符3	0101 0001	121	81	0x51	Q	大写字母Q
0011 0100	64	52	0x34	4	字符4	0101 0010	122	82	0x52	R	大写字母R
0011 0101	65	53	0x35	5	字符5	0101 0011	123	83	0x53	S	大写字母S
0011 0110	66	54	0x36	6	字符6	0101 0100	124	84	0x54	T	大写字母T
0011 0111	67	55	0x37	7	字符7	0101 0101	125	85	0x55	U	大写字母U
0011 1000	70	56	0x38	8	字符8	0101 0110	126	86	0x56	V	大写字母V
0011 1001	71	57	0x39	9	字符9	0101 0111	127	87	0x57	W	大写字母W
0011 1010	72	58	0x3A	:	冒号	0101 1000	130	88	0x58	X	大写字母X
0011 1011	73	59	0x3B	;	分号	0101 1001	131	89	0x59	Y	大写字母Y
0011 1100	74	60	0x3C	<	小于	0101 1010	132	90	0x5A	Z	大写字母Z
0011 1101	75	61	0x3D	=	等号	0101 1011	133	91	0x5B	[	开方括号
0011 1110	76	62	0x3E	>	大于	0101 1100	134	92	0x5C	\	反斜杠
0011 1111	77	63	0x3F	?	问号	0101 1101	135	93	0x5D	]	闭方括号
0100 0000	100	64	0x40	@	电子邮件符号	0101 1110	136	94	0x5E	^	脱字符
0100 0001	101	65	0x41	A	大写字母A	0101 1111	137	95	0x5F	_	下划线
0100 0010	102	66	0x42	B	大写字母B	0110 0000	140	96	0x60	`	开单引号
0100 0011	103	67	0x43	C	大写字母C	0110 0001	141	97	0x61	a	小写字母a
0100 0100	104	68	0x44	D	大写字母D	0110 0010	142	98	0x62	b	小写字母b
0100 0101	105	69	0x45	E	大写字母E	0110 0011	143	99	0x63	c	小写字母c
0100 0110	106	70	0x46	F	大写字母F	0110 0100	144	100	0x64	d	小写字母d
0100 0111	107	71	0x47	G	大写字母G	0110 0101	145	101	0x65	e	小写字母e
0100 1000	110	72	0x48	H	大写字母H	0110 0110	146	102	0x66	f	小写字母f
0100 1001	111	73	0x49	I	大写字母I	0110 0111	147	103	0x67	g	小写字母g
1001010	112	74	0x4A	J	大写字母J	0110 1000	150	104	0x68	h	小写字母h
0100 1011	113	75	0x4B	K	大写字母K	0110 1001	151	105	0x69	i	小写字母i
0100 1100	114	76	0x4C	L	大写字母L	0110 1010	152	106	0x6A	j	小写字母j
0100 1101	115	77	0x4D	M	大写字母M	0110 1011	153	107	0x6B	k	小写字母k

以串口 0 输出字符串“1“为例设置程序如图：

串口输出显示为“1”，勾选 16 进制显示，串口输出为“31 0d 0a”，其中 1 转换 16 进制为 31，0d 0a 转换 16 进制为换行。



5.串口监视器-串口助手使用

当使用串口监视器出现一些奇怪的现象时，可以使用串口助手查看串口接收的数据。

打开串口助手：

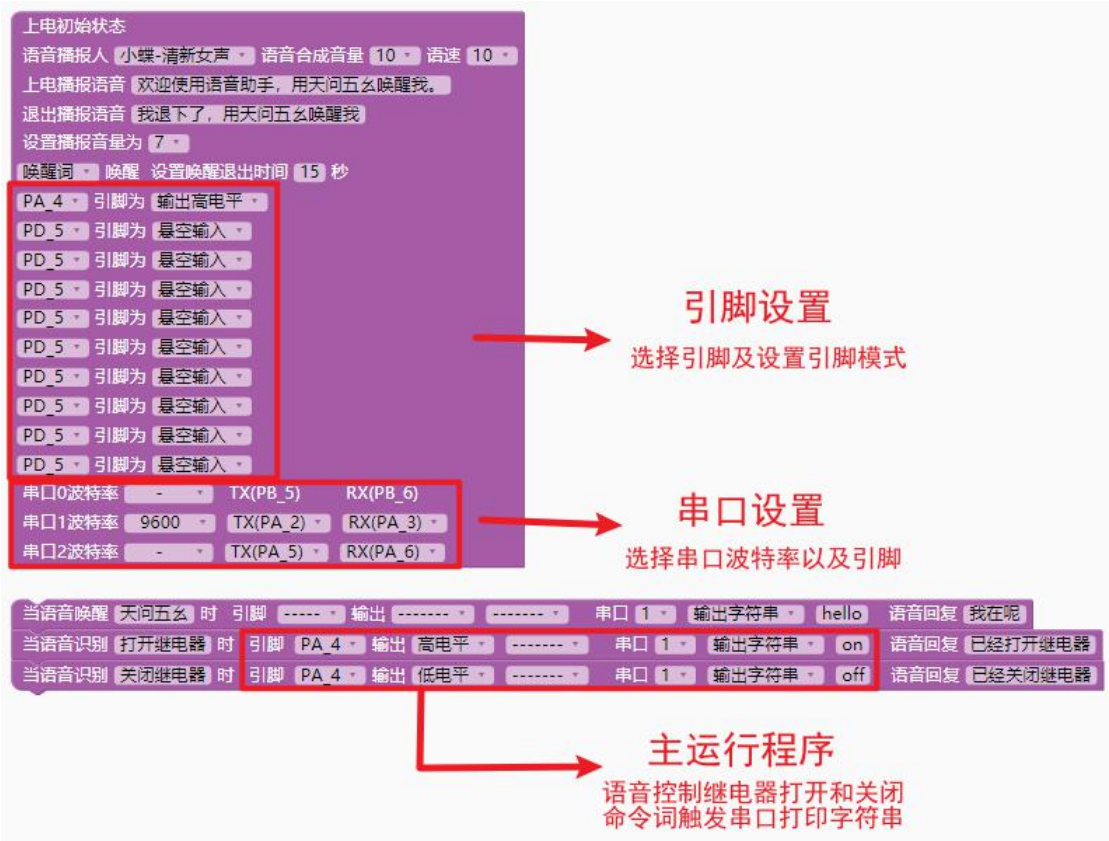
常用功能介绍：

范例 2.2 串口 1 输出字符串

一、功能简介

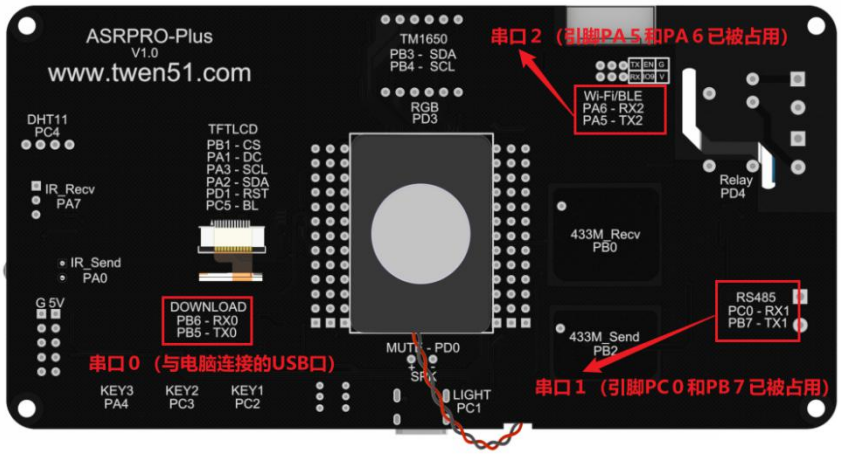
本范例通过学习如何使用串口自动发送字符串数据，实现串口输出字符串的功能，达到用户可以正确使用串口通讯的目的。

二、范例分析



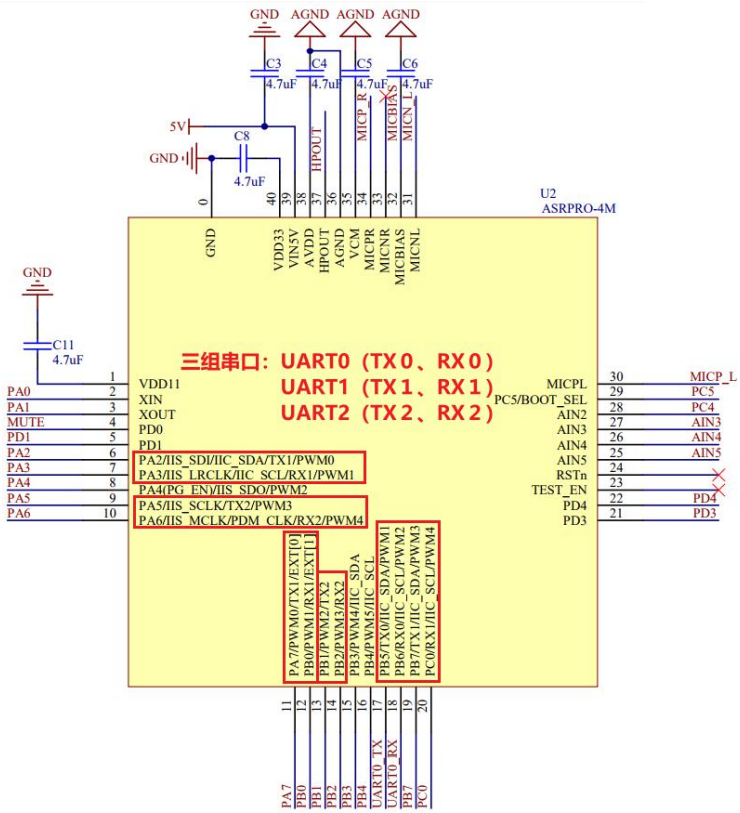
三、电路与实物说明

ASRPRO-Plus 开发板串口所处位置如下图所示：



可以看到具有串口通讯功能的的 PA5、PA6、PC0 和 PB7 引脚已经被占用了，如果要查看串口通讯的数据，可以使用烧写器连接其他具有串口通讯的引脚，并借助串口监视器或 STC-ISP 软件查看不同串口的通讯数据。

芯片内部电路原理图如下图所示：



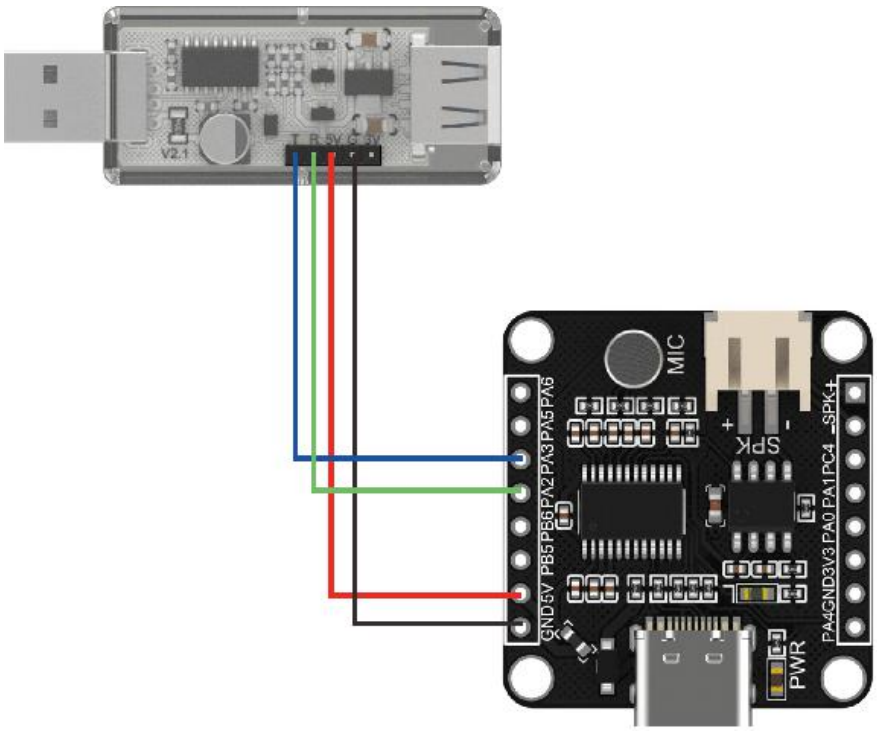
可以看到 ASRPRO-Plus 开发板的芯片共有三组串口，分别是 UART0、UART1 和 UART2。

UART 是最常见的串行通讯，广泛应用于单片机和单片机之间通讯。比如 WiFi 模块，串口液晶屏等。串口通信经过信号转换，可以进行 RS232、RS422、RS485 通信，广泛应用于设备之间远程通信。

串口通讯采用 2 条通讯线：发送数据线 TX，接收数据线 RX。要实现不同设备之间的数据收发，需要将通讯设备 1 的 TX 与通讯设备 2 的 RX 相连，将通讯设备 1 的 RX 与通讯设备 2 的 TX 相连，就可以同时进行数据的发送和接收。

在本范例中，我们需要设置一组串口的引脚位置，串口 0 的位置已经固定（PB5 和 PB6），串口 1 的位置（PA2 和 PA3），串口 1 和串口 2 的位置可以通过编程修改确定（避开已被占用引脚）。

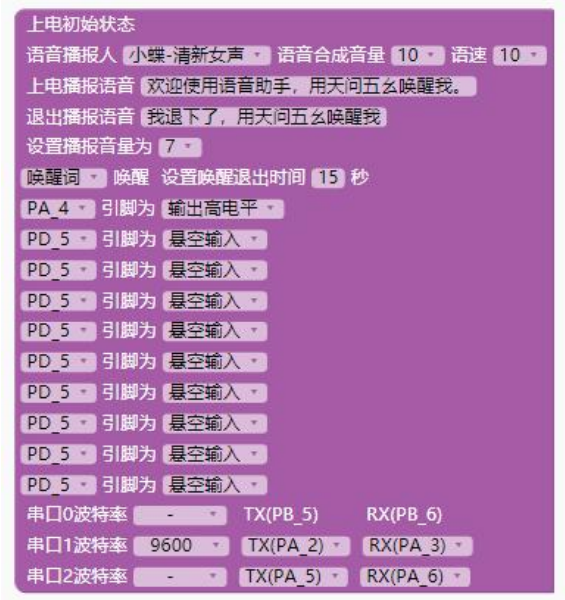
ASRPRO 开发板实物连接图：



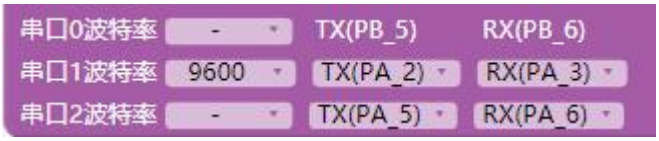
四、具体指令讲解

1.上电初始状态设置

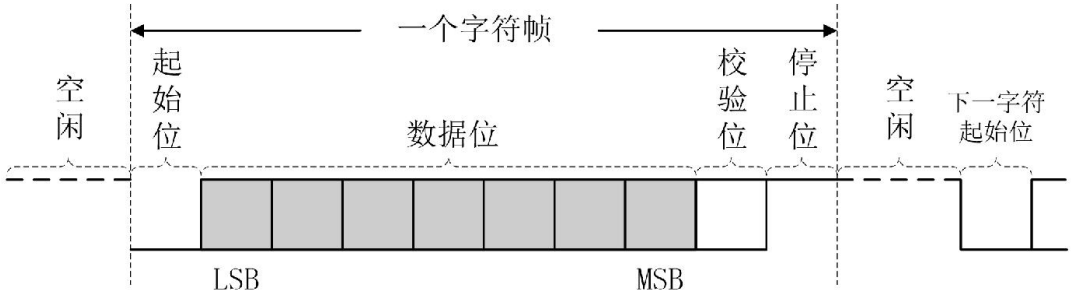
PA_4 输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。



可设置 3 个串口波特率。



波特率（BaudRate）表示数据传送速率，即每秒钟传送的二进制位数。波特率通常单位是 bit/s，比较常用的波特率有 9600，57600，115200 等等。

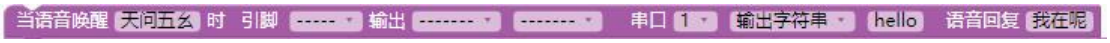


设置串口发送与接收的引脚，其中串口 0 固定发送引脚 PB_5，接收 PB_6，串口 0 的位置是 USB 接口上。串口 1 可以设置的发送引脚有 PA_2、PA_7、PB_7，接收引脚有 PA_3、PB_0、PC_0。串口 2 可以设置的发送引脚有 PA_5、PB_1、PB_6，接收引脚有 PA_6、PB_2。根据具体需要与其它设备进行通讯的情况，进行选择。（在本范例中，我们需要设置一组串口的引脚位置，串口 0 的位置已经固定 PB5 和 PB6，串口 1 和串口 2 的位置可以通过编程修改确定,避开已被占用引脚）

2.主程序设置

语音控制唤醒词输出 16 进制命令模块设置：

语音识别“天问五幺”，串口选择 1，输出模式选择输出字符串，串口 1 输出的字符串为 hello。



语音控制继电器输出字符串命令模块设置：

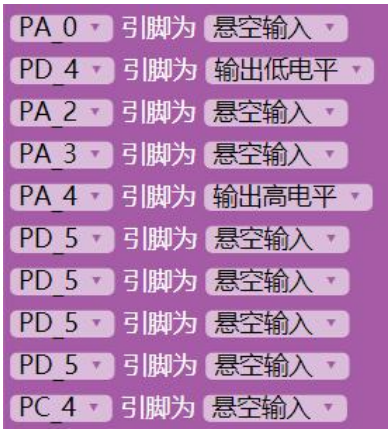
语音识别“打开继电器”选择引脚 PA_4，输出高电平，串口选择 1，输出模式选择输出字符串，串口 1 输出的字符串为 on。

语音识别“关闭继电器”选择引脚 PA_4，输出低电平，串口选择 1，输出模式选择输出字符串，串口 1 输出的字符串为 off。（串口 0 和串口 2 同理设置）

3.程序修改

PD_4 继电器输出低电平，PA_4 模拟第二个继电器，输出高电平。

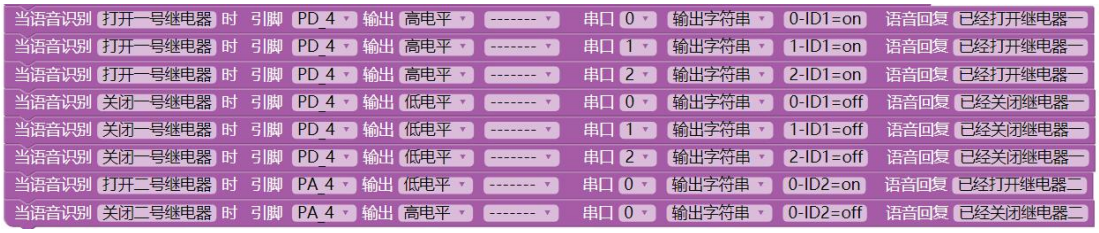
所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。



语音控制继电器输出字符串命令模块设置：

语音识别“打开一号继电器”选择引脚 PD_4，输出高电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 0-ID1=on。

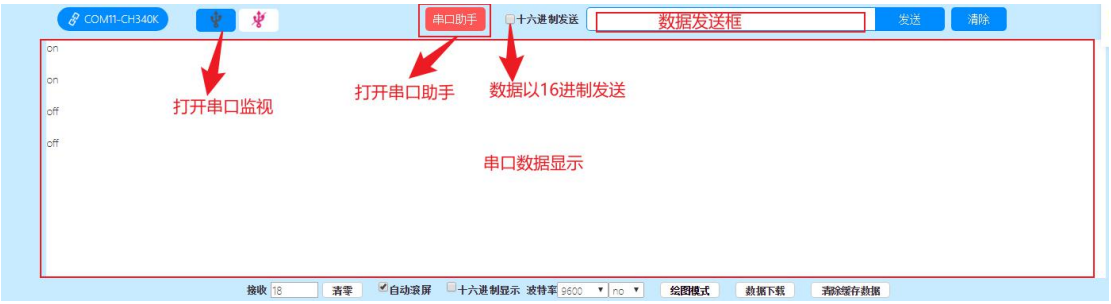
语音识别“关闭一号继电器”选择引脚 PD_4，输出低电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 0-ID1=off。（串口 1 和串口 2 同理设置）。



4.串口监视器查看串口 1 输出字符串

打开串口监视器：点击右上角串口监视器按钮

本范例串口监视器设置：COM 端口对应选择主板的 COM 端口，COM 端口打开如图左边按钮显示蓝色为打开状态，波特率对应程序设置，本范例为 9600。



串口 1 输出显示区域：当我们发出语音命令到主板，主板会输出对应字符串并在串口输出显示区域显示。

字符串的本质就是多个字节组成的字符。

如串口发送"hello"，实际是串口发送 5 个字节，第一个字节'h'（16 进制是 0x68)、第二个字节'e'(16 进制是 0x65)、第三个字节'l'(16 进制是 0x6c)、第四个字节'l'(16 进制是 0x6c)、第五个字节'o'(16 进制是 0x6f)。

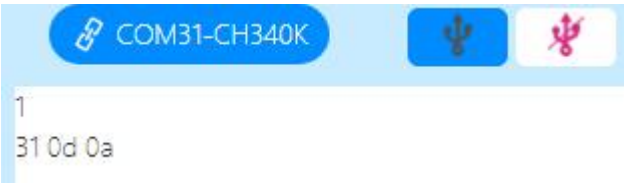
我们可以用串口原始输出 16 进制数，就可在串口助手看到"hello"字符串。

对应参考 ASCII 码表具体如下所示：

Bin	Oct	Dec	Hex	缩写/字符	解释	Bin	Feb	Apr	Hex	缩写/字符	解释
(二进制)	(八进制)	(十进制)	(十六进制)			(二进制)	(八进制)	(十进制)	(十六进制)		
0011 0000	60	48	0x30	0	字符0	0100 1110	116	78	0x4E	N	大写字母N
0011 0001	61	49	0x31	1	字符1	0100 1111	117	79	0x4F	O	大写字母O
0011 0010	62	50	0x32	2	字符2	0101 0000	120	80	0x50	P	大写字母P
0011 0011	63	51	0x33	3	字符3	0101 0001	121	81	0x51	Q	大写字母Q
0011 0100	64	52	0x34	4	字符4	0101 0010	122	82	0x52	R	大写字母R
0011 0101	65	53	0x35	5	字符5	0101 0011	123	83	0x53	S	大写字母S
0011 0110	66	54	0x36	6	字符6	0101 0100	124	84	0x54	T	大写字母T
0011 0111	67	55	0x37	7	字符7	0101 0101	125	85	0x55	U	大写字母U
0011 1000	70	56	0x38	8	字符8	0101 0110	126	86	0x56	V	大写字母V
0011 1001	71	57	0x39	9	字符9	0101 0111	127	87	0x57	W	大写字母W
0011 1010	72	58	0x3A	:	冒号	0101 1000	130	88	0x58	X	大写字母X
0011 1011	73	59	0x3B	;	分号	0101 1001	131	89	0x59	Y	大写字母Y
0011 1100	74	60	0x3C	<	小于	0101 1010	132	90	0x5A	Z	大写字母Z
0011 1101	75	61	0x3D	=	等号	0101 1011	133	91	0x5B	[	开方括号
0011 1110	76	62	0x3E	>	大于	0101 1100	134	92	0x5C	\	反斜杠
0011 1111	77	63	0x3F	?	问号	0101 1101	135	93	0x5D	]	闭方括号
0100 0000	100	64	0x40	@	电子邮件符号	0101 1110	136	94	0x5E	^	脱字符
0100 0001	101	65	0x41	A	大写字母A	0101 1111	137	95	0x5F	_	下划线
0100 0010	102	66	0x42	B	大写字母B	0110 0000	140	96	0x60	`	开单引号
0100 0011	103	67	0x43	C	大写字母C	0110 0001	141	97	0x61	a	小写字母a
0100 0100	104	68	0x44	D	大写字母D	0110 0010	142	98	0x62	b	小写字母b
0100 0101	105	69	0x45	E	大写字母E	0110 0011	143	99	0x63	c	小写字母c
0100 0110	106	70	0x46	F	大写字母F	0110 0100	144	100	0x64	d	小写字母d
0100 0111	107	71	0x47	G	大写字母G	0110 0101	145	101	0x65	e	小写字母e
0100 1000	110	72	0x48	H	大写字母H	0110 0110	146	102	0x66	f	小写字母f
0100 1001	111	73	0x49	I	大写字母I	0110 0111	147	103	0x67	g	小写字母g
1001010	112	74	0x4A	J	大写字母J	0110 1000	150	104	0x68	h	小写字母h
0100 1011	113	75	0x4B	K	大写字母K	0110 1001	151	105	0x69	i	小写字母i
0100 1100	114	76	0x4C	L	大写字母L	0110 1010	152	106	0x6A	j	小写字母j
0100 1101	115	77	0x4D	M	大写字母M	0110 1011	153	107	0x6B	k	小写字母k

以串口 1 输出字符串“1\n“为例设置程序如图：

串口输出显示为“1”，勾选 16 进制显示，串口输出为“31 0d 0a”，其中 1 转换 16 进制为 31，0d 0a 转换 16 进制为换行。



5.串口监视器-串口助手使用

当使用串口监视器出现一些奇怪的现象时，可以使用串口助手查看串口接收的数据。
打开串口助手：

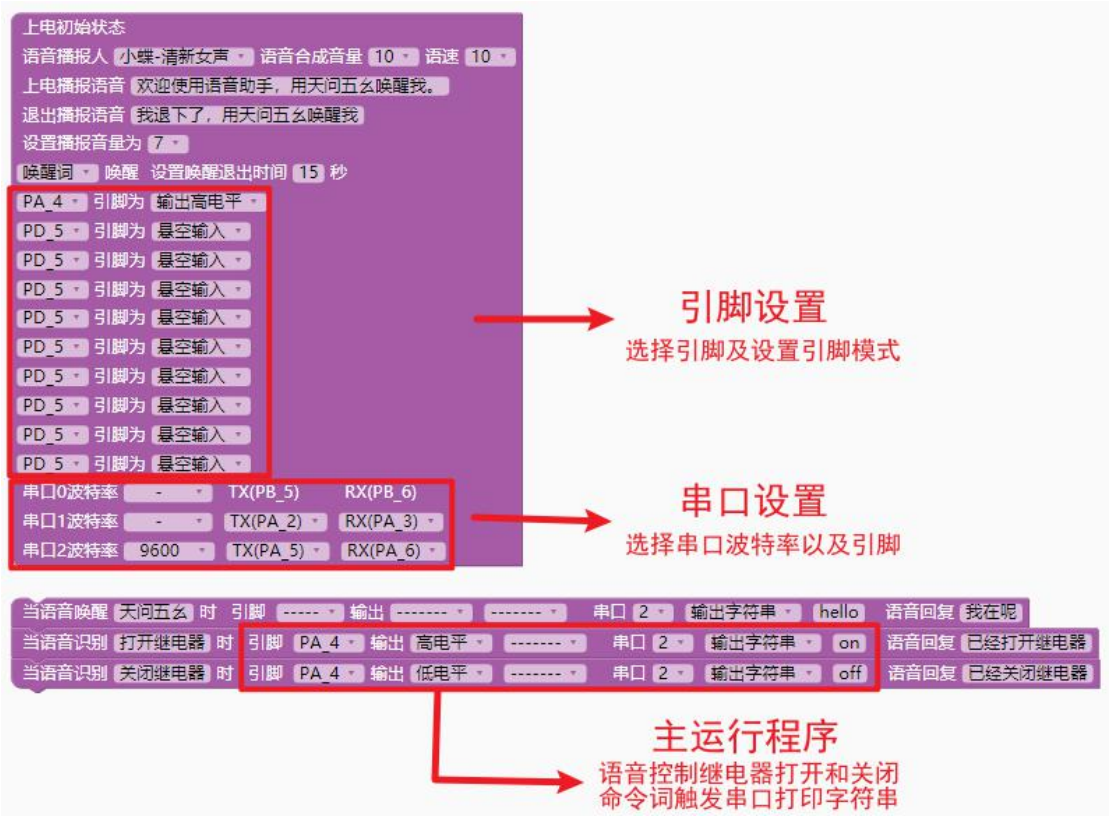
常用功能介绍：

范例 2.3 串口 2 输出字符串

一、功能简介

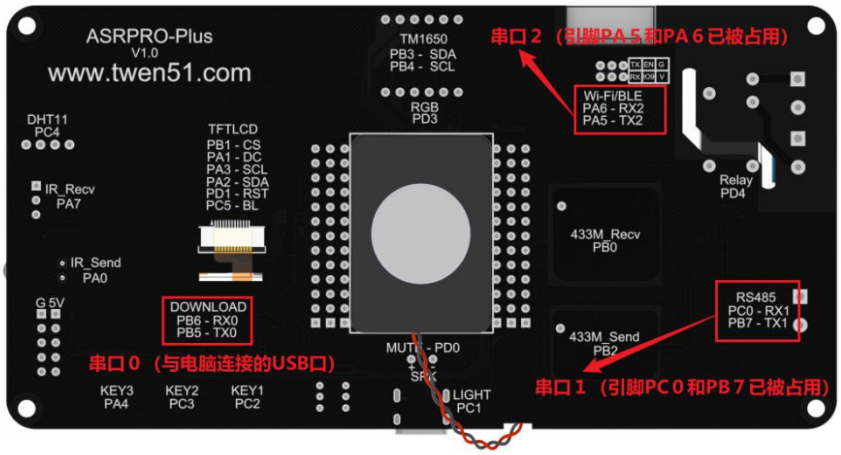
本范例通过学习如何使用串口自动发送字符串数据，实现串口输出字符串的功能，达到用户可以正确使用串口通讯的目的。

二、范例分析



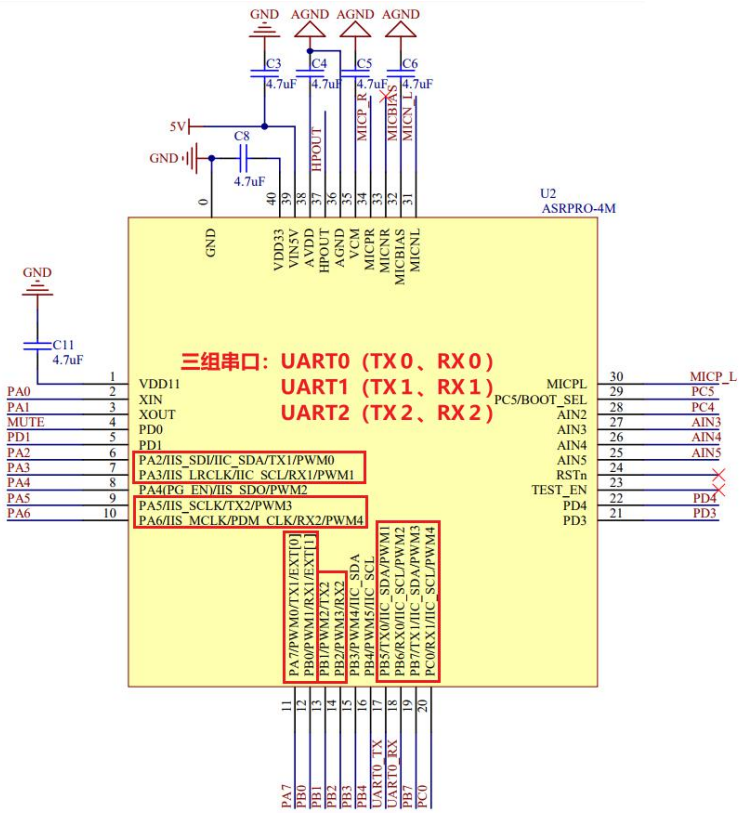
三、电路与实物说明

ASRPRO-Plus 开发板串口所处位置如下图所示：



可以看到具有串口通讯功能的的 PA5、PA6、PC0 和 PB7 引脚已经被占用了，如果要查看串口通讯的数据，可以使用烧写器连接其他具有串口通讯的引脚，并借助串口监视器或 STC-ISP 软件查看不同串口的通讯数据。

芯片内部电路原理图如下图所示：



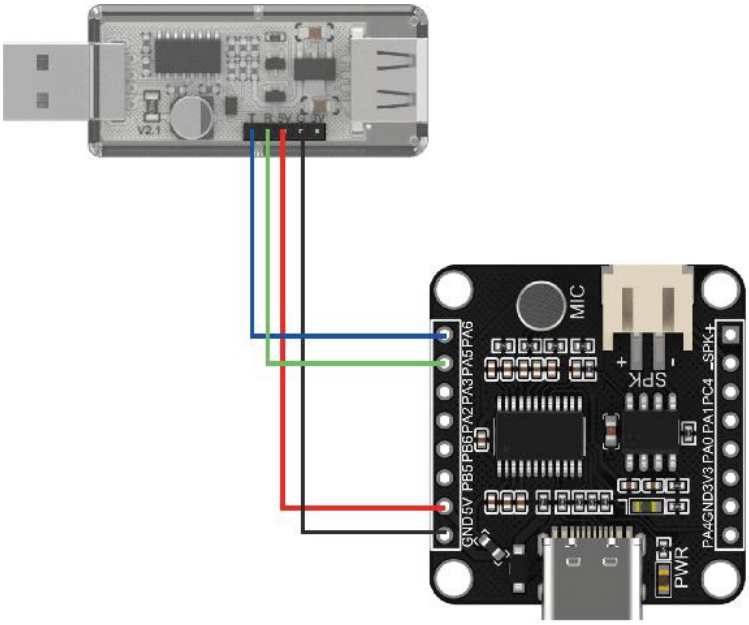
可以看到 ASRPRO-Plus 开发板的芯片共有三组串口，分别是 UART0、UART1 和 UART2。

UART 是最常见的串行通讯，广泛应用于单片机和单片机之间通讯。比如 WiFi 模块，串口液晶屏等。串口通信经　过信号转换，可以进行 RS232、RS422、RS485 通信，广泛应用于设备之间远程通信。

串口通讯采用 2 条通讯线：发送数据线 TX，接收数据线 RX。要实现不同设备之间的数据收发，需要将通讯设备 1 的 TX 与通讯设备 2 的 RX 相连，将通讯设备 1 的 RX 与通讯设备 2 的 TX 相连，就可以同时进行数据的发送和接收。

在本范例中，我们需要设置一组串口的引脚位置，串口 0 的位置已经固定（PB5 和 PB6），串口 1 和串口 2 的位置可以通过编程修改确定（避开已被占用引脚）。

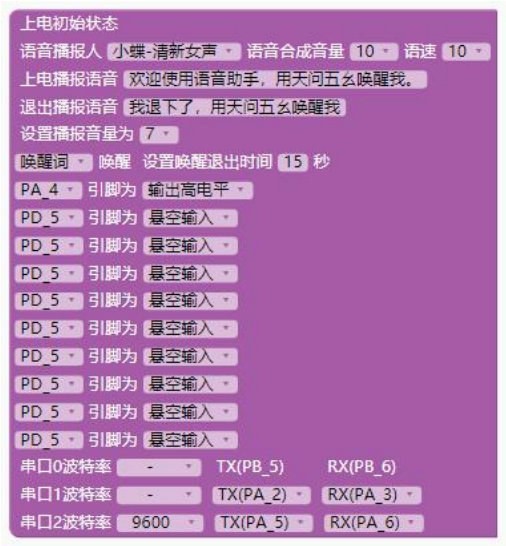
ASRPRO 开发板实物连接：



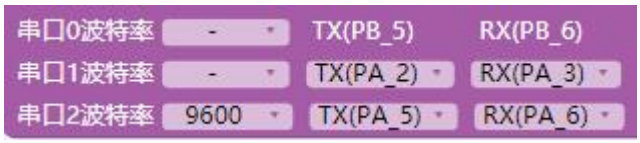
四、具体指令讲解

1.上电初始状态设置

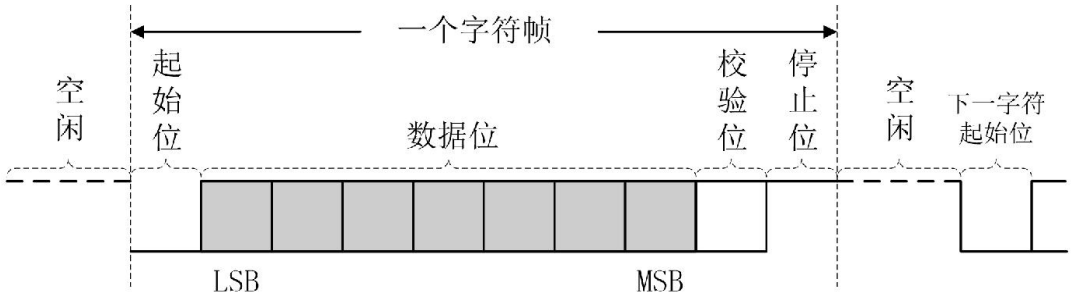
PA_4 输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。



可设置 3 个串口波特率。



波特率（BaudRate）表示数据传送速率，即每秒钟传送的二进制位数。波特率通常单位是 bit/s，比较常用的波特率有 9600，57600，115200 等等。



设置串口发送与接收的引脚，其中串口 0 固定发送引脚 PB_5，接收 PB_6，串口 0 的位置是 USB 接口上。串口 1 可以设置的发送引脚有 PA_2、PA_7、PB_7，接收引脚有 PA_3、PB_0、PC_0。串口 2 可以设置的发送引脚有 PA_5、PB_1、PB_6，接收引脚有 PA_6、PB_2。根据具体需要与其它设备进行通讯的情况，进行选择。（在本范例中，我们需要设置一组串口的引脚位置，串口 0 的位置已经固定 PB5 和 PB6，串口 1 和串口 2 的位置可以通过编程修改确定,避开已被占用引脚）

2.主程序设置

语音控制唤醒词输出 16 进制命令模块设置：

语音识别“天问五幺”，串口选择 2，输出模式选择输出字符串，串口 2 输出的字符串为 hello。

语音控制继电器输出字符串命令模块设置：

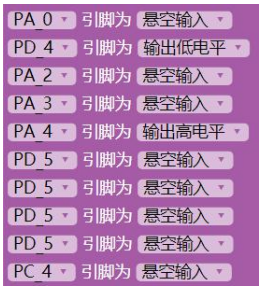
语音识别“打开继电器”选择引脚 PA_4，输出高电平，串口选择 2，输出模式选择输出字符串，串口 2 输出的字符串为 on。

语音识别“关闭继电器”选择引脚 PA_4，输出低电平，串口选择 2，输出模式选择输出字符串，串口 2 输出的字符串为 off。（串口 0 和串口 1 同理设置）

3.程序修改

PD_4 继电器输出低电平，PA_4 模拟第二个继电器，输出高电平。

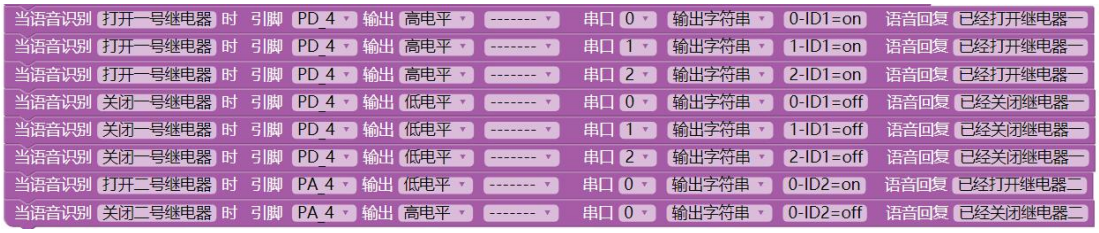
所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。



语音控制继电器输出字符串命令模块设置：

语音识别“打开一号继电器”选择引脚 PD_4，输出高电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 0-ID1=on。

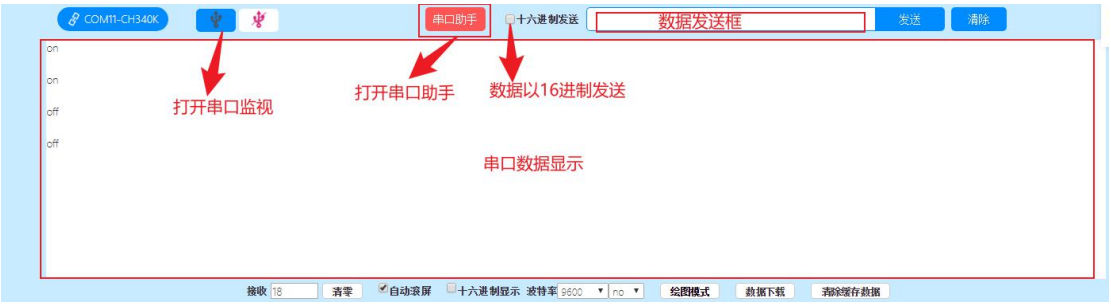
语音识别“关闭一号继电器”选择引脚 PD_4，输出低电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 0-ID1=off。（串口 1 和串口 2 同理设置）。



4.串口监视器查看串口 2 输出字符串

打开串口监视器：点击右上角串口监视器按钮

本范例串口监视器设置：COM 端口对应选择主板的 COM 端口，COM 端口打开如图左边按钮显示蓝色为打开状态，波特率对应程序设置，本范例为 9600。



串口 2 输出显示区域：当我们发出语音命令到主板，主板会输出对应字符串并在串口输出显示区域显示。

字符串的本质就是多个字节组成的字符。

如串口发送"hello"，实际是串口发送 5 个字节，第一个字节'h'（16 进制是 0x68）、第二个字节'e'(16 进制是 0x65)、第三个字节'l'(16 进制是 0x6c)、第四个字节'l'(16 进制是 0x6c)、第五个字节'o'(16 进制是 0x6f)。

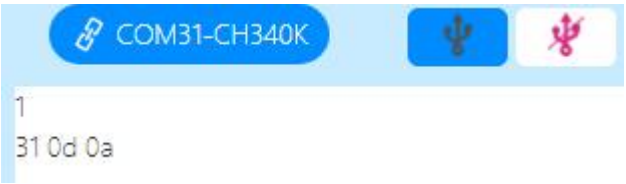
我们可以用串口原始输出 16 进制数，就可在串口助手看到"hello"字符串。

对应参考 ASCII 码表具体如下所示：

Bin	Oct	Dec	Hex	缩写/字符	解释	Bin	Feb	Apr	Hex	缩写/字符	解释
(二进制)	(八进制)	(十进制)	(十六进制)			(二进制)	(八进制)	(十进制)	(十六进制)		
0011 0000	60	48	0x30	0	字符0	0100 1110	116	78	0x4E	N	大写字母N
0011 0001	61	49	0x31	1	字符1	0100 1111	117	79	0x4F	O	大写字母O
0011 0010	62	50	0x32	2	字符2	0101 0000	120	80	0x50	P	大写字母P
0011 0011	63	51	0x33	3	字符3	0101 0001	121	81	0x51	Q	大写字母Q
0011 0100	64	52	0x34	4	字符4	0101 0010	122	82	0x52	R	大写字母R
0011 0101	65	53	0x35	5	字符5	0101 0011	123	83	0x53	S	大写字母S
0011 0110	66	54	0x36	6	字符6	0101 0100	124	84	0x54	T	大写字母T
0011 0111	67	55	0x37	7	字符7	0101 0101	125	85	0x55	U	大写字母U
0011 1000	70	56	0x38	8	字符8	0101 0110	126	86	0x56	V	大写字母V
0011 1001	71	57	0x39	9	字符9	0101 0111	127	87	0x57	W	大写字母W
0011 1010	72	58	0x3A	:	冒号	0101 1000	130	88	0x58	X	大写字母X
0011 1011	73	59	0x3B	;	分号	0101 1001	131	89	0x59	Y	大写字母Y
0011 1100	74	60	0x3C	<	小于	0101 1010	132	90	0x5A	Z	大写字母Z
0011 1101	75	61	0x3D	=	等号	0101 1011	133	91	0x5B	[	开方括号
0011 1110	76	62	0x3E	>	大于	0101 1100	134	92	0x5C	\	反斜杠
0011 1111	77	63	0x3F	?	问号	0101 1101	135	93	0x5D	]	闭方括号
0100 0000	100	64	0x40	@	电子邮件符号	0101 1110	136	94	0x5E	^	脱字符
0100 0001	101	65	0x41	A	大写字母A	0101 1111	137	95	0x5F	_	下划线
0100 0010	102	66	0x42	B	大写字母B	0110 0000	140	96	0x60	`	开单引号
0100 0011	103	67	0x43	C	大写字母C	0110 0001	141	97	0x61	a	小写字母a
0100 0100	104	68	0x44	D	大写字母D	0110 0010	142	98	0x62	b	小写字母b
0100 0101	105	69	0x45	E	大写字母E	0110 0011	143	99	0x63	c	小写字母c
0100 0110	106	70	0x46	F	大写字母F	0110 0100	144	100	0x64	d	小写字母d
0100 0111	107	71	0x47	G	大写字母G	0110 0101	145	101	0x65	e	小写字母e
0100 1000	110	72	0x48	H	大写字母H	0110 0110	146	102	0x66	f	小写字母f
0100 1001	111	73	0x49	I	大写字母I	0110 0111	147	103	0x67	g	小写字母g
1001010	112	74	0x4A	J	大写字母J	0110 1000	150	104	0x68	h	小写字母h
0100 1011	113	75	0x4B	K	大写字母K	0110 1001	151	105	0x69	i	小写字母i
0100 1100	114	76	0x4C	L	大写字母L	0110 1010	152	106	0x6A	j	小写字母j
0100 1101	115	77	0x4D	M	大写字母M	0110 1011	153	107	0x6B	k	小写字母k

以串口 2 输出字符串“1\n“为例设置程序如图：

串口输出显示为“1”，勾选 16 进制显示，串口输出为“31 0d 0a”，其中 1 转换 16 进制为 31，0d 0a 转换 16 进制为换行。



5.串口监视器-串口助手使用

当使用串口监视器出现一些奇怪的现象时，可以使用串口助手查看串口接收的数据。

打开串口助手：

常用功能介绍:


串口调试小助手 1.3
端 口
COM1
波特率
9600
校验位
None (无)
数据位
8
停止位
1
打开串口
查看任务管理器-端口
串口波特率检验位等设置，必须与发送方一致
设置完成后，点击打开串口
清空接收区
接收区
停止显示
自动清空
十六进制显示
点击勾选，接收以16进制显示
接收文件
None
保存数据
更改路径
C:\COMDATA\
填写框后点击发送
清空重填
发送区
自动发送
手动发送
发送框
收/发计数
(字节)
自动发送周期
1000
毫秒
选择发送文件
还没有选择文件
发送文件
以固定时间自动发送
Rx:0
Tx:0
计数清零
退出

范例 2.4 串口 0-1-2 同时输出字符串

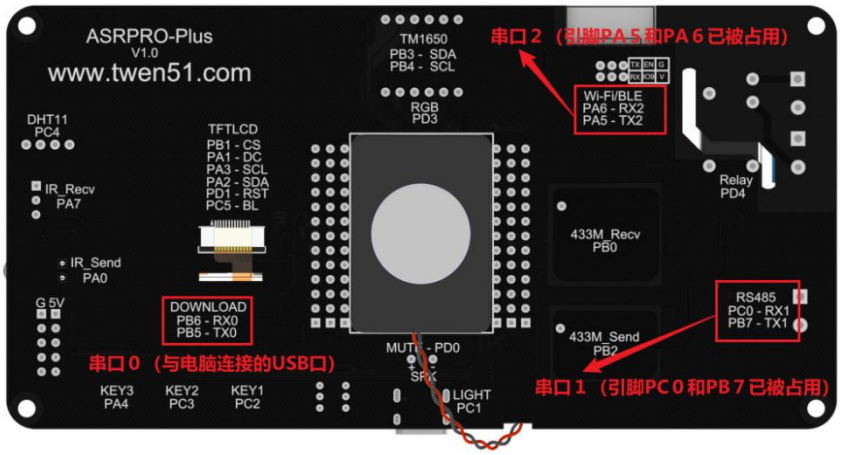
一、功能简介

本范例通过学习如何使用串口自动发送字符串数据，实现串口输出字符串的功能，达到用户可以正确使用串口通讯的目的。

二、范例分析

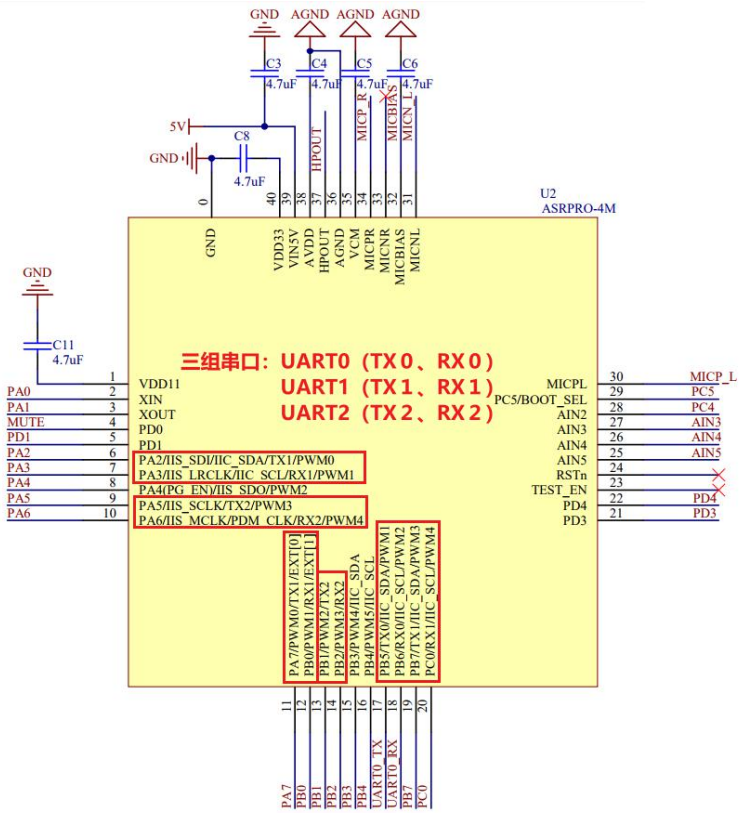
三、电路与实物说明

ASRPRO-Plus 开发板串口所处位置如下图所示：



可以看到具有串口通讯功能的的 PA5、PA6、PC0 和 PB7 引脚已经被占用了，如果要查看串口通讯的数据，可以使用烧写器连接其他具有串口通讯的引脚，并借助串口监视器或 STC-ISP 软件查看不同串口的通讯数据。

芯片内部电路原理图如下图所示：



可以看到 ASRPRO-Plus 开发板的芯片共有三组串口，分别是 UART0、UART1 和 UART2。

UART 是最常见的串行通讯，广泛应用于单片机和单片机之间通讯。比如 WiFi 模块，串口液晶屏等。串口通信经过信号转换，可以进行 RS232、RS422、RS485 通信，广泛应用于设备之间远程通信。

串口通讯采用 2 条通讯线：发送数据线 TX，接收数据线 RX。要实现不同设备之间的数据收发，需要将通讯设备 1 的 TX 与通讯设备 2 的 RX 相连，将通讯设备 1 的 RX 与通讯设备 2 的 TX 相连，就可以同时进行数据的发送和接收。

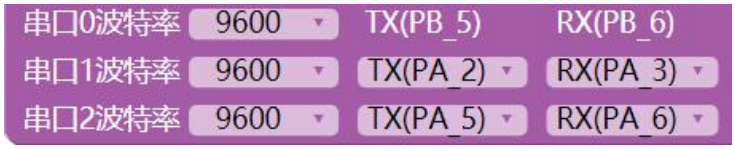
在本范例中，我们需要设置三组串口的引脚位置，串口 0 的位置已经固定（PB5 和 PB6），串口 1 和串口 2 的位置可以通过编程修改确定（避开已被占用引脚）。

四、具体指令讲解

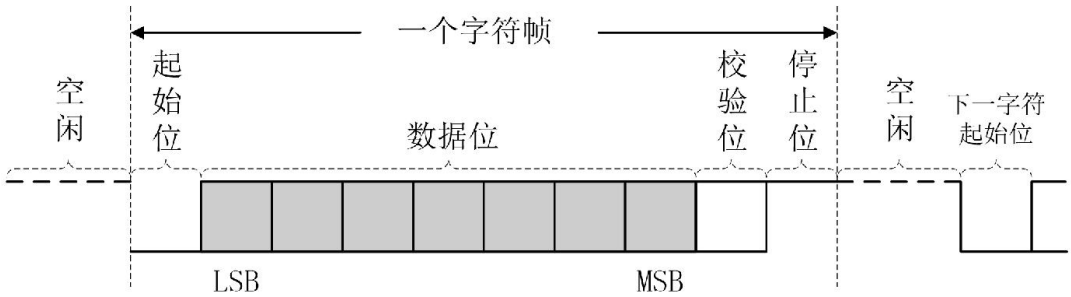
1.上电初始状态设置

PA_4 输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。

可设置 3 个串口波特率。



波特率（BaudRate）表示数据传送速率，即每秒钟传送的二进制位数。波特率通常单位是 bit/s，比较常用的波特率有 9600，57600，115200 等等。

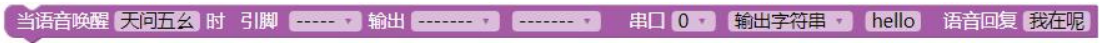


设置串口发送与接收的引脚，其中串口 0 固定发送引脚 PB_5，接收 PB_6，串口 0 的位置是 USB 接口上。串口 1 可以设置的发送引脚有 PA_2、PA_7、PB_7，接收引脚有 PA_3、PB_0、PC_0。串口 2 可以设置的发送引脚有 PA_5、PB_1、PB_6，接收引脚有 PA_6、PB_2。根据具体需要与其它设备进行通讯的情况，进行选择。（在本范例中，我们需要设置三组串口的引脚位置，串口 0 的位置已经固定 PB5 和 PB6，串口 1 和串口 2 的位置可以通过编程修改确定,避开已被占用引脚）

2.主程序设置

语音控制唤醒词输出 16 进制命令模块设置：

语音识别“天问五幺”，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 hello。



语音控制继电器输出字符串命令模块设置：

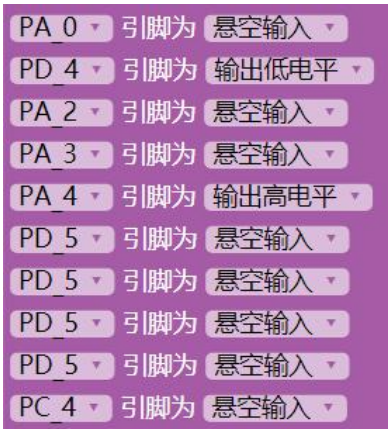
语音识别“打开继电器”选择引脚 PA_4，输出高电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 on。

语音识别“关闭继电器”选择引脚 PA_4，输出低电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 off。（串口 1 和串口 2 同理设置）

3.程序修改

PD_4 继电器输出低电平，PA_4 模拟第二个继电器，输出高电平。

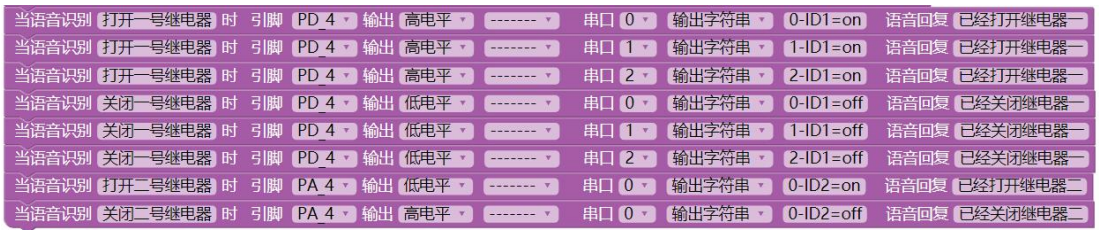
所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。



语音控制继电器输出字符串命令模块设置：

语音识别“打开一号继电器”选择引脚 PD_4，输出高电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 0-ID1=on。

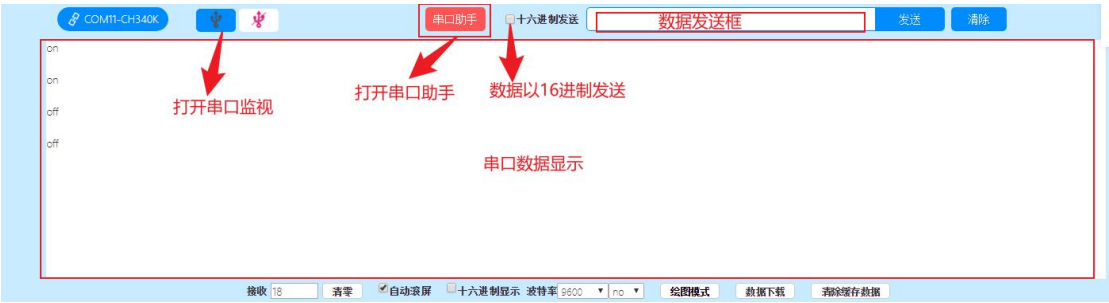
语音识别“关闭一号继电器”选择引脚 PD_4，输出低电平，串口选择 0，输出模式选择输出字符串，串口 0 输出的字符串为 0-ID1=off。（串口 1 和串口 2 同理设置）。



4.串口监视器查看串口输出字符串

打开串口监视器：点击右上角串口监视器按钮

本范例串口监视器设置：COM 端口对应选择主板的 COM 端口，COM 端口打开如图左边按钮显示蓝色为打开状态，波特率对应程序设置，本范例为 9600。



串口 0 输出显示区域：当我们发出语音命令到主板，主板会输出对应字符串并在串口输出显示区域显示。

字符串的本质就是多个字节组成的字符。

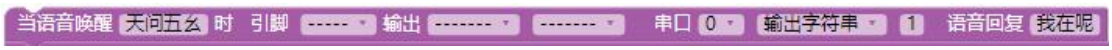
如串口发送"hello"，实际是串口发送 5 个字节，第一个字节'h'（16 进制是 0x68)、第二个字节'e'(16 进制是 0x65)、第三个字节'l'(16 进制是 0x6c)、第四个字节'l'(16 进制是 0x6c)、第五个字节'o'(16 进制是 0x6f)。

我们可以用串口原始输出 16 进制数，就可在串口助手看到"hello"字符串。

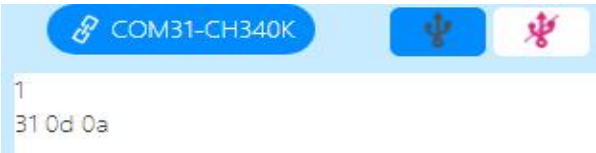
对应参考 ASCII 码表具体如下所示：

Bin	Oct	Dec	Hex	缩写/字符	解释	Bin	Feb	Apr	Hex	缩写/字符	解释
(二进制)	(八进制)	(十进制)	(十六进制)			(二进制)	(八进制)	(十进制)	(十六进制)		
0011 0000	60	48	0x30	0	字符0	0100 1110	116	78	0x4E	N	大写字母N
0011 0001	61	49	0x31	1	字符1	0100 1111	117	79	0x4F	O	大写字母O
0011 0010	62	50	0x32	2	字符2	0101 0000	120	80	0x50	P	大写字母P
0011 0011	63	51	0x33	3	字符3	0101 0001	121	81	0x51	Q	大写字母Q
0011 0100	64	52	0x34	4	字符4	0101 0010	122	82	0x52	R	大写字母R
0011 0101	65	53	0x35	5	字符5	0101 0011	123	83	0x53	S	大写字母S
0011 0110	66	54	0x36	6	字符6	0101 0100	124	84	0x54	T	大写字母T
0011 0111	67	55	0x37	7	字符7	0101 0101	125	85	0x55	U	大写字母U
0011 1000	70	56	0x38	8	字符8	0101 0110	126	86	0x56	V	大写字母V
0011 1001	71	57	0x39	9	字符9	0101 0111	127	87	0x57	W	大写字母W
0011 1010	72	58	0x3A	:	冒号	0101 1000	130	88	0x58	X	大写字母X
0011 1011	73	59	0x3B	;	分号	0101 1001	131	89	0x59	Y	大写字母Y
0011 1100	74	60	0x3C	<	小于	0101 1010	132	90	0x5A	Z	大写字母Z
0011 1101	75	61	0x3D	=	等号	0101 1011	133	91	0x5B	[	开方括号
0011 1110	76	62	0x3E	>	大于	0101 1100	134	92	0x5C	\	反斜杠
0011 1111	77	63	0x3F	?	问号	0101 1101	135	93	0x5D	]	闭方括号
0100 0000	100	64	0x40	@	电子邮件符号	0101 1110	136	94	0x5E	^	脱字符
0100 0001	101	65	0x41	A	大写字母A	0101 1111	137	95	0x5F	_	下划线
0100 0010	102	66	0x42	B	大写字母B	0110 0000	140	96	0x60	`	开单引号
0100 0011	103	67	0x43	C	大写字母C	0110 0001	141	97	0x61	a	小写字母a
0100 0100	104	68	0x44	D	大写字母D	0110 0010	142	98	0x62	b	小写字母b
0100 0101	105	69	0x45	E	大写字母E	0110 0011	143	99	0x63	c	小写字母c
0100 0110	106	70	0x46	F	大写字母F	0110 0100	144	100	0x64	d	小写字母d
0100 0111	107	71	0x47	G	大写字母G	0110 0101	145	101	0x65	e	小写字母e
0100 1000	110	72	0x48	H	大写字母H	0110 0110	146	102	0x66	f	小写字母f
0100 1001	111	73	0x49	I	大写字母I	0110 0111	147	103	0x67	g	小写字母g
1001010	112	74	0x4A	J	大写字母J	0110 1000	150	104	0x68	h	小写字母h
0100 1011	113	75	0x4B	K	大写字母K	0110 1001	151	105	0x69	i	小写字母i
0100 1100	114	76	0x4C	L	大写字母L	0110 1010	152	106	0x6A	j	小写字母j
0100 1101	115	77	0x4D	M	大写字母M	0110 1011	153	107	0x6B	k	小写字母k

以串口 0 输出字符串“1\n“为例设置程序如图：



串口输出显示为“1”，勾选 16 进制显示，串口输出为“31 0d 0a”，其中 1 转换 16 进制为 31，0d 0a 转换 16 进制为换行。



5.串口监视器-串口助手使用

当使用串口监视器出现一些奇怪的现象时，可以使用串口助手查看串口接收的数据。

打开串口助手：

常用功能介绍:


串口调试小助手 1.3
端 口
COM1
波特率
9600
校验位
None (无)
数据位
8
停止位
1
打开串口
查看任务管理器-端口
串口波特率检验位等设置，必须与发送方一致
设置完成后，点击打开串口
清空接收区
接收区
停止显示
自动清空
十六进制显示
点击勾选，接收以16进制显示
接收文件
None
保存数据
更改路径
C:\COMDATA\
填写框后点击发送
清空重填
发送区
自动发送
手动发送
发送框
收/发计数
(字节)
自动发送周期
1000
毫秒
选择发送文件
还没有选择文件
发送文件
以固定时间自动发送
Rx:0
Tx:0
计数清零
退出

范例 2.5 串口 0 输出十六进制数

一、功能简介

本范例通过学习如何使用串口自动发送十六进制数据，实现串口输出十六进制数的功能，达到用户可以正确使用串口通讯的目的。

二、范例分析

三、具体指令讲解

1.主程序设置

语音控制唤醒词输出 16 进制命令模块设置：
语音识别“天问五幺”，串口选择 0，输出模式选择输出 16 进制，串口 0 输出的 16 进制为 A0 00 00 。

对于十六进制数，我们一般使用字首“0x”，例如“0x5A3”。开头的“0”令解析器更易辨认数，而“x”则代表十六进制（就如“O”代表八进制）。在“0x”中的“x”可以大写或小写。如果同时发送多个 16 进制数，需要用空格隔开。

图形块里不用写 0x 前缀，软件自动处理好了，方便用户快速输入。

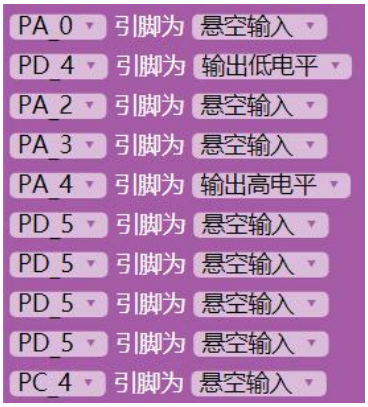
语音控制继电器输出 16 进制命令模块设置：

语音识别“打开继电器”选择引脚 PA_4，输出高电平，串口选择 0，输出模式选择输出 16 进制，串口 0 输出的 16 进制为 A0 00 01。

语音识别“关闭继电器”选择引脚 PA_4，输出低电平，串口选择 0，输出模式选择输出 16 进制，串口 0 输出的 16 进制 A0 00 02（串口 1 和串口 2 同理设置）

2.程序修改

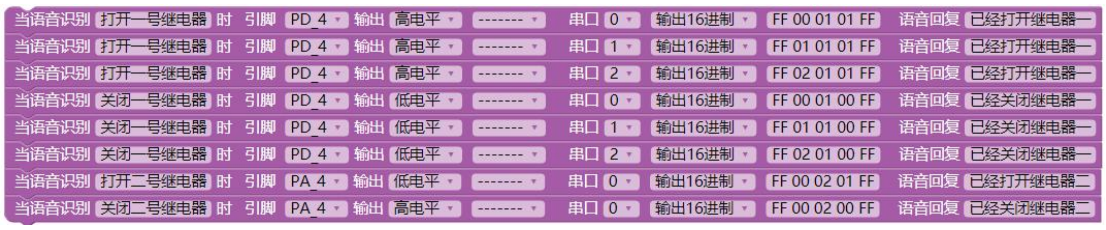
PD_4 继电器输出低电平，PA_4 模拟第二个继电器，输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。



语音控制继电器输出 16 进制命令模块设置：

语音识别“打开一号继电器”选择引脚 PD_4，输出高电平，串口选择 0，输出模式选择输出 16 进制，串口 0 输出的 16 进制为 FF 00 01 01 FF。

语音识别“关闭一号继电器”选择引脚 PD_4，输出低电平，串口选择 0，输出模式选择输出 16 进制，串口 0 输出的 16 进制为 FF 00 01 00 FF。（串口 1 和串口 2 同理设置）。



3.串口监视器查看串口 0 输出 16 进制

打开串口监视器：点击右上角串口监视器按钮

本范例串口监视器设置：勾选十六进制显示。

串口 0 输出显示区域：当我们发出语音命令道主板，主板会输出对应字符串并在串口输出显示区域显示。（如果不勾选 16 进制显示，会出现乱码）

4.串口监视器-串口助手使用

当使用串口监视器出现一些奇怪的现象时，可以使用串口助手查看串口接收的数据。

打开串口助手：

常用功能介绍：

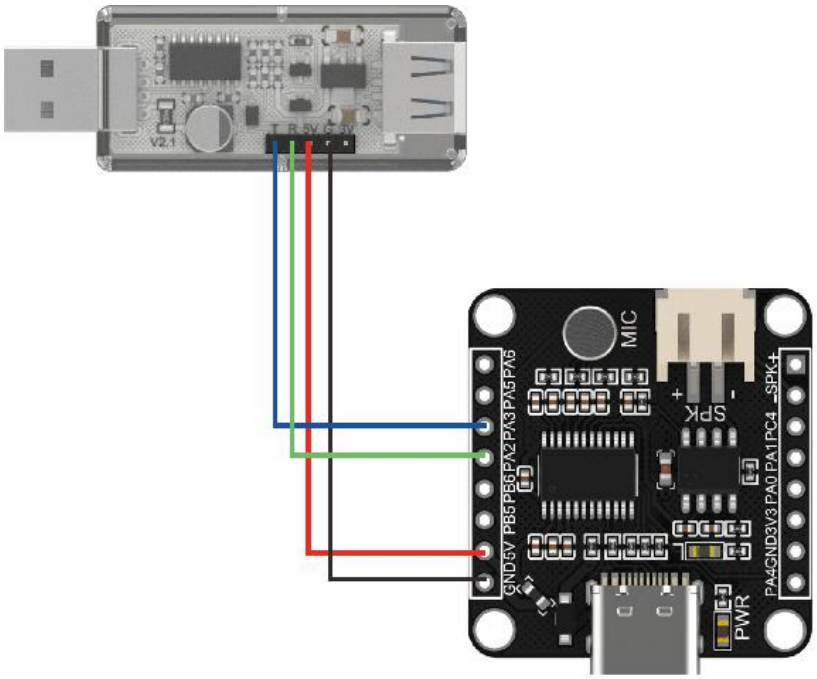
范例 2.6 串口 1 输出十六进制数

一、功能简介

本范例通过学习如何使用串口自动发送十六进制数据，实现串口输出十六进制数的功能，达到用户可以正确使用串口通讯的目的。

二、范例分析

ASRPRO 开发板实物连接：



三、具体指令讲解

1.主程序设置

语音控制唤醒词输出 16 进制命令模块设置：

语音识别“天问五幺”，串口选择 1，输出模式选择输出 16 进制，串口 1 输出的 16 进制为 A0 01 00 。

对于十六进制数，我们一般使用字首“0x”，例如“0x5A”。开头的“0”令解析器更易辨认数，而“x”则代表十六进制（就如“O”代表八进制）。在“0x”中的“x”可以大写或小写。如果同时发送多个 16 进制数，需要用空格隔开。

图形块里不用写 0x 前缀，软件自动处理好了，方便用户快速输入。

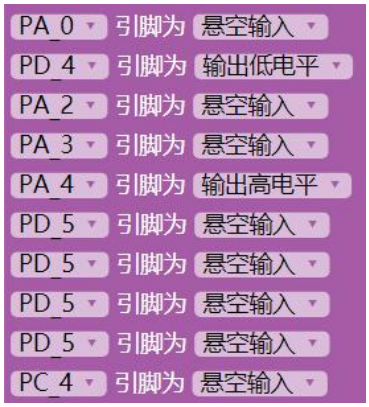
语音控制继电器输出 16 进制命令模块设置：

语音识别“打开继电器”选择引脚 PA_4，输出高电平，串口选择 1，输出模式选择输出 16 进制，串口 1 输出的 16 进制为 A0 01 01。

语音识别“关闭继电器”选择引脚 PA_4，输出低电平，串口选择 1，输出模式选择输出 16 进制，串口 1 输出的 16 进制 A0 01 02（串口 0 和串口 2 同理设置）

2.程序修改

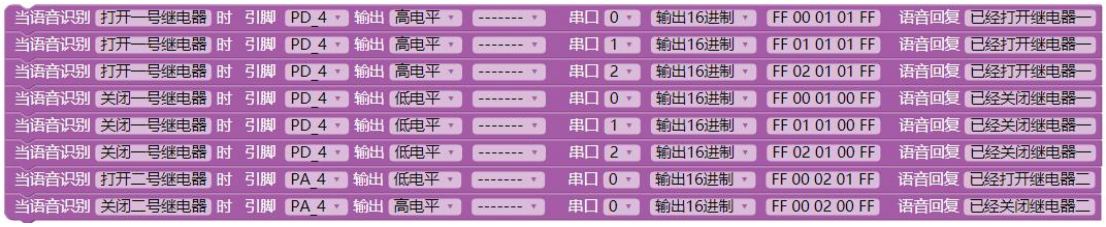
PD_4 继电器输出低电平，PA_4 模拟第二个继电器，输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。



语音控制继电器输出 16 进制命令模块设置：

语音识别“打开一号继电器”选择引脚 PD_4，输出高电平，串口选择 0，输出模式选择输出 16 进制，串口 0 输出的 16 进制为 FF 00 01 01 FF。

语音识别“关闭一号继电器”选择引脚 PD_4，输出低电平，串口选择 0，输出模式选择输出 16 进制，串口 0 输出的 16 进制为 FF 00 01 00 FF。（串口 1 和串口 2 同理设置）。



3.串口监视器查看串口 1 输出 16 进制

打开串口监视器：点击右上角串口监视器按钮

本范例串口监视器设置：勾选十六进制显示。

串口 1 输出显示区域：当我们发出语音命令道主板，主板会输出对应字符串并在串口输出显示区域显示。（如果不勾选 16 进制显示，会出现乱码）

4.串口监视器-串口助手使用

当使用串口监视器出现一些奇怪的现象时，可以使用串口助手查看串口接收的数据。

打开串口助手：

常用功能介绍：

范例 2.7 串口 2 输出十六进制数

一、功能简介

本范例通过学习如何使用串口自动发送十六进制数据，实现串口输出十六进制数的功能，达到用户可以正确使用串口通讯的目的。

二、范例分析

三、具体指令讲解

1.主程序设置

语音控制唤醒词输出 16 进制命令模块设置：
语音识别“天问五幺”，串口选择 2，输出模式选择输出 16 进制，串口 2 输出的 16 进制为 A0 02 00 。

对于十六进制数，我们一般使用字首“0x”，例如“0x5A”。开头的“0”令解析器更易辨认数，而“x”则代表十六进制（就如“O”代表八进制）。在“0x”中的“x”可以大写或小写。如果同时发送多个 16 进制数，需要用空格隔开。

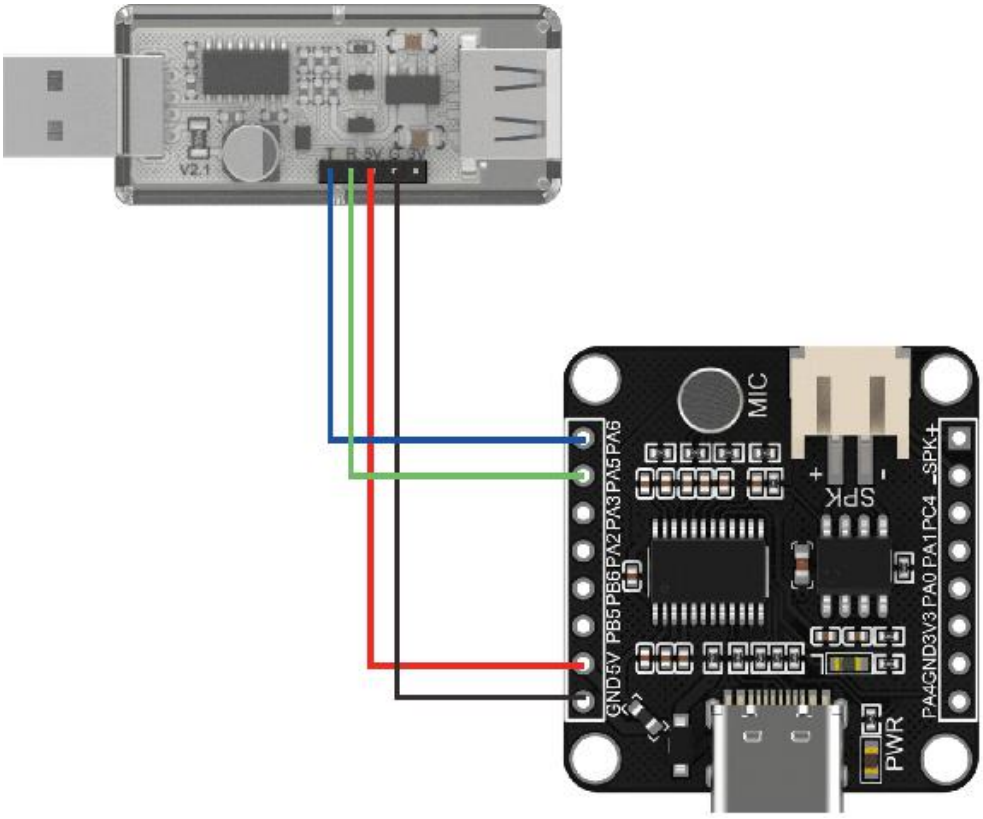
图形块里不用写 0x 前缀，软件自动处理好了，方便用户快速输入。

语音控制继电器输出 16 进制命令模块设置：

语音识别“打开继电器”选择引脚 PA_4，输出高电平，串口选择 2，输出模式选择输出 16 进制，串口 2 输出的 16 进制为 A0 02 01。

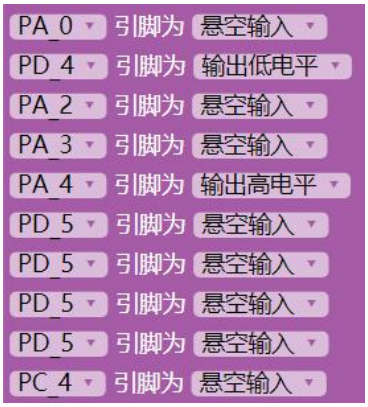
语音识别“关闭继电器”选择引脚 PA_4，输出低电平，串口选择 2，输出模式选择输出 16 进制，串口 2 输出的 16 进制 A0 02 02（串口 0 和串口 1 同理设置）

实物连接：



2.程序修改

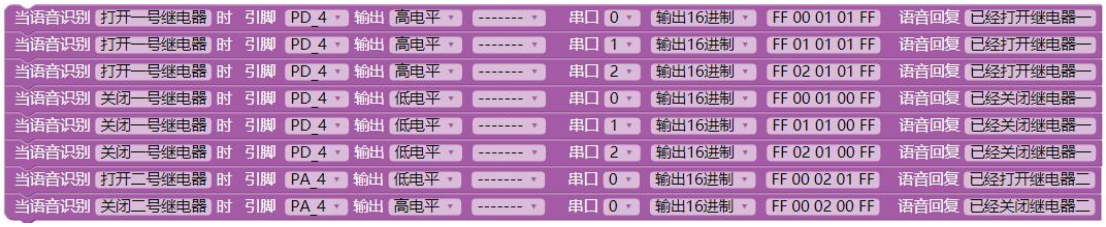
PD_4 继电器输出低电平，PA_4 模拟第二个继电器，输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。



语音控制继电器输出 16 进制命令模块设置：

语音识别“打开一号继电器”选择引脚 PD_4，输出高电平，串口选择 2，输出模式选择输出 16 进制，串口 2 输出的 16 进制为 FF 00 02 01 FF。

语音识别“关闭一号继电器”选择引脚 PD_4，输出低电平，串口选择 2，输出模式选择输出 16 进制，串口 2 输出的 16 进制为 FF 00 02 00 FF。（串口 0 和串口 2 同理设置）。



3.串口监视器查看串口 2 输出 16 进制

打开串口监视器：点击右上角串口监视器按钮

本范例串口监视器设置：勾选十六进制显示。

串口 0 输出显示区域：当我们发出语音命令道主板，主板会输出对应字符串并在串口输出显示区域显示。（如果不勾选 16 进制显示，会出现乱码）

4.串口监视器-串口助手使用

当使用串口监视器出现一些奇怪的现象时，可以使用串口助手查看串口接收的数据。

打开串口助手：

常用功能介绍:


串口调试小助手 1.3
端 口
COM1
波特率
9600
校验位
None (无)
数据位
8
停止位
1
打开串口
查看任务管理器-端口
串口波特率检验位等设置，必须与发送方一致
设置完成后，点击打开串口
清空接收区
接收区
停止显示
自动清空
十六进制显示
点击勾选，接收以16进制显示
接收文件
None
保存数据
更改路径
C:\COMDATA\
填写框后点击发送
清空重填
发送区
自动发送
手动发送
发送框
收/发计数
自动发送周期
1000
毫秒
选择发送文件
还没有选择文件
发送文件
(字节)
STATUS:COM Port Closed
以固定时间自动发送
Rx:0
Tx:0
计数清零
退出

范例 2.8 串口 0-1-2 同时输出十六进制数

一、功能简介

本范例通过学习如何使用串口自动发送十六进制数据，实现串口输出十六进制数的功能，达到用户可以正确使用串口通讯的目的。

二、范例分析

三、具体指令讲解

1.主程序设置

语音控制唤醒词输出 16 进制命令模块设置：

语音识别“天问五幺”，串口选择 0，输出模式选择输出 16 进制，串口 0 输出的 16 进制为 A0 00 00 。

语音识别“天问五幺”，串口选择 1，输出模式选择输出 16 进制，串口 1 输出的 16 进制为 A0

01 00 。

语音识别“天问五幺”，串口选择 2，输出模式选择输出 16 进制，串口 2 输出的 16 进制为 A0 02 00 。

对于十六进制数，我们一般使用字首“0x”，例如“0x5A3”。开头的“0”令解析器更易辨认数，而“x”则代表十六进制（就如“O”代表八进制）。在“0x”中的“x”可以大写或小写。如果同时发送多个 16 进制数，需要用空格隔开。

图形块里不用写 0x 前缀，软件自动处理好了，方便用户快速输入。

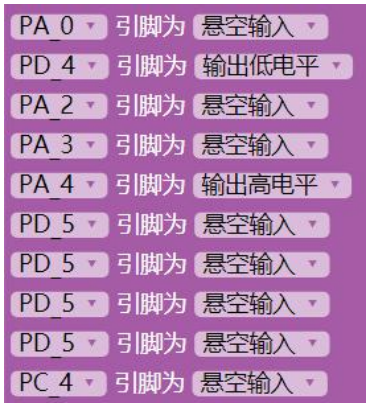
语音控制继电器输出 16 进制命令模块设置：

语音识别“打开继电器”选择引脚 PA_4，输出高电平，串口选择 0、1、2，输出模式选择输出 16 进制，串口 0 输出的 16 进制为 A0 00 01。

语音识别“关闭继电器”选择引脚 PA_4，输出低电平，串口选择 0、1、2，输出模式选择输出 16 进制，串口 0 输出的 16 进制 A0 00 02（串口 1 和串口 2 同理设置）

2.程序修改

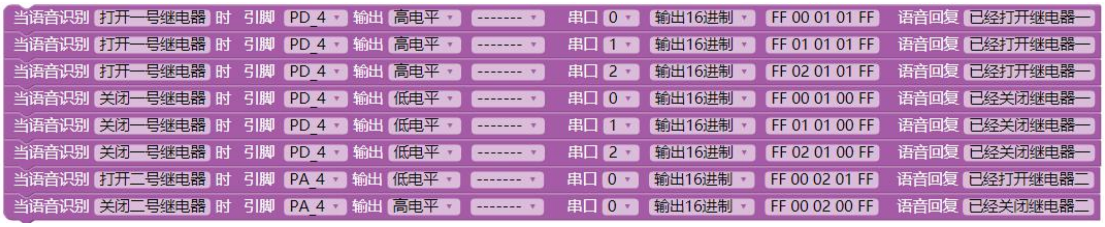
PD_4 继电器输出低电平，PA_4 模拟第二个继电器，输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。



语音控制继电器输出 16 进制命令模块设置：

语音识别“打开一号继电器”选择引脚 PD_4，输出高电平，串口选择 0，输出模式选择输出 16 进制，串口 0 输出的 16 进制为 FF 00 01 01 FF。

语音识别“关闭一号继电器”选择引脚 PD_4，输出低电平，串口选择 0，输出模式选择输出 16 进制，串口 0 输出的 16 进制为 FF 00 01 00 FF。（串口 1 和串口 2 同理设置）。



3.串口监视器查看串口 16 进制数据

打开串口监视器：点击右上角串口监视器按钮

本范例串口监视器设置：勾选十六进制显示。

串口输出显示区域：当我们发出语音命令道主板，主板会输出对应字符串并在串口输出显示区域显示。（如果不勾选 16 进制显示，会出现乱码）

4.串口监视器-串口助手使用

当使用串口监视器出现一些奇怪的现象时，可以使用串口助手查看串口接收的数据。

打开串口助手：

常用功能介绍:


串口调试小助手 1.3
端 口
COM1
波特率
9600
校验位
None (无)
数据位
8
停止位
1
打开串口
查看任务管理器-端口
串口波特率检验位等设置，必须与发送方一致
设置完成后，点击打开串口
清空接收区
接收区
停止显示
自动清空
十六进制显示
点击勾选，接收以16进制显示
接收文件
None
保存数据
更改路径
C:\COMDATA\
填写框后点击发送
清空重填
发送区
自动发送
手动发送
发送框
收/发计数
(字节)
自动发送周期
1000
毫秒
选择发送文件
还没有选择文件
发送文件
以固定时间自动发送
Rx:0
Tx:0
计数清零
退出

范例 2.9 串口 0 接收到字符串打开继电器

一、功能简介

本范例通过学习如何使用串口接收字符串数据，实现串口接收字符串并通过接收的字符串判断是否打开继电器，达到用户可以正确使用串口通讯的目的。

二、范例分析

三、具体指令讲解

1.主程序设置

当串口接收到数据后，和程序设定的数据进行比较（可以是字符串或者是 16 进制），如果相同时引脚输出高低电平。

这里程序收到“on”时，PA4 输出高电平即继电器打开，同时语音播报“已经打开继电器”；当收到“off”时 PA4 输出低电平即继电器关闭，同时语音播报“已经关闭继电器”；

2.程序修改

PA_4 模拟继电器，输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。

语音识别“打开继电器”选择引脚 PA_4，输出高电平。

语音识别“关闭继电器”选择引脚 PA_4，输出低电平。

串口收到字符串“aabbcc”时，PA4 输出高电平即继电器打开。

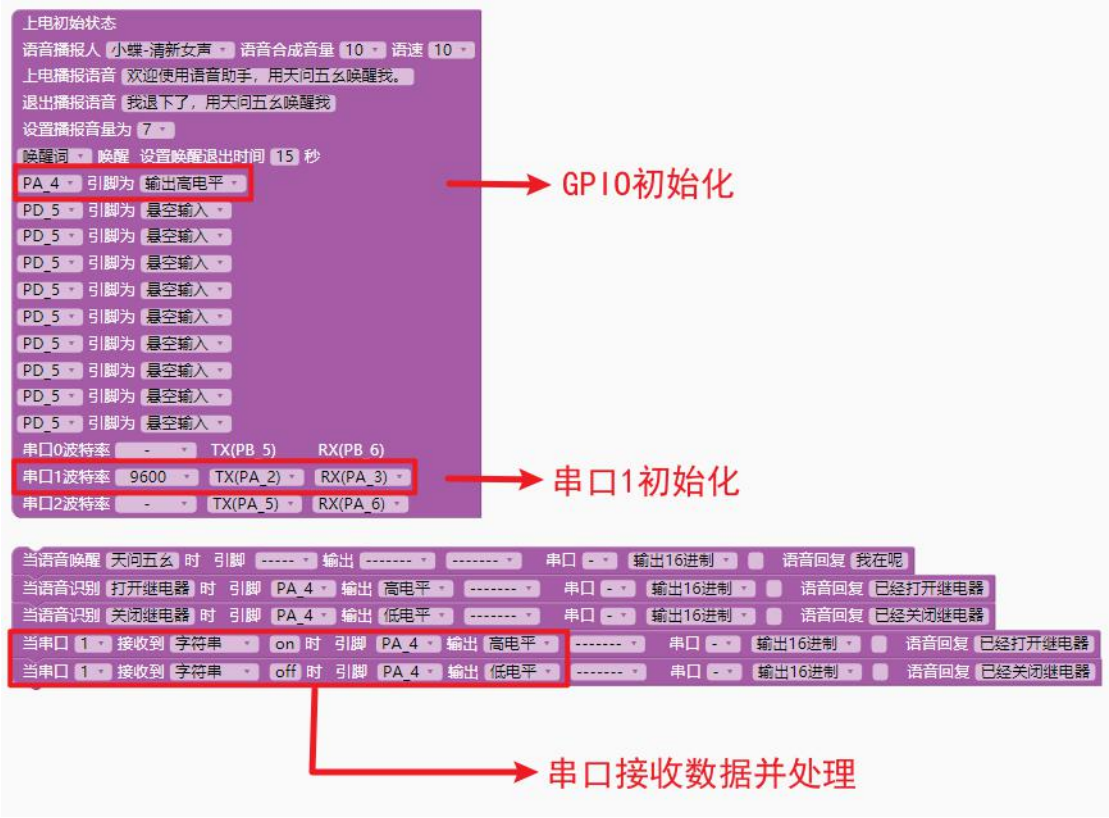
串口收到字符串“ddeeff”时，PA4 输出低电平即继电器关闭。

范例 2.10 串口 1 接收到字符串打开继电器

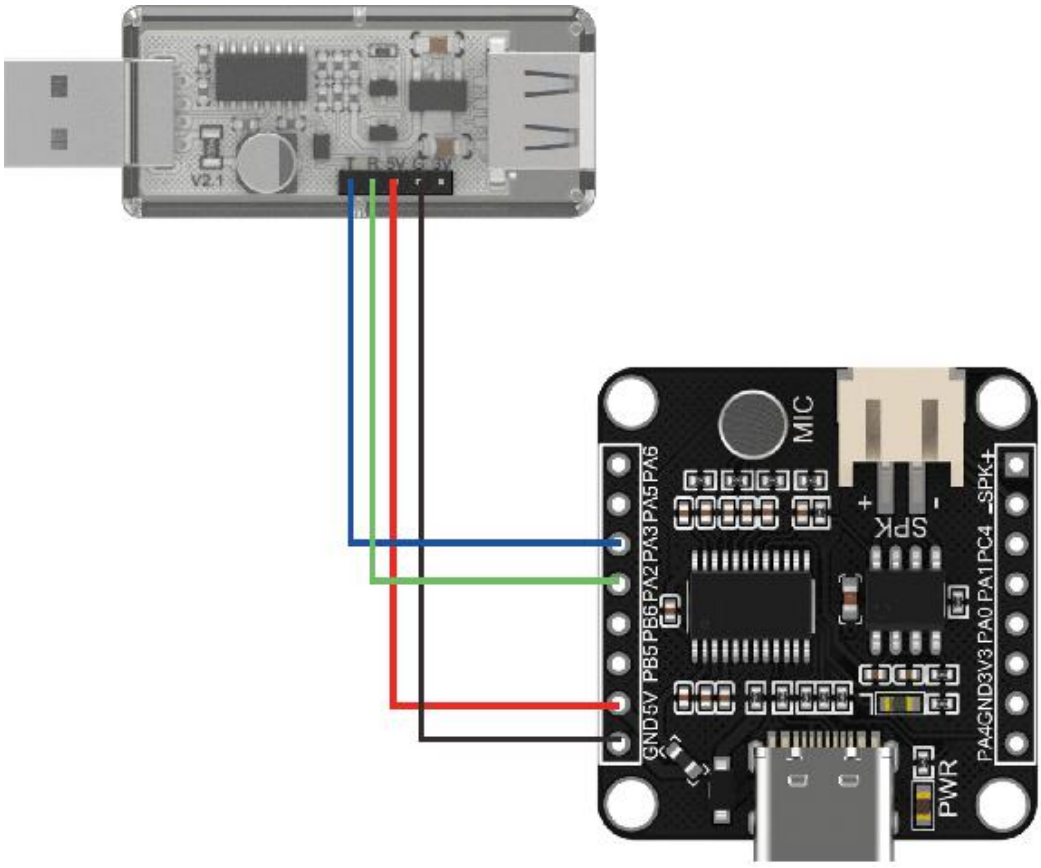
一、功能简介

本范例通过学习如何使用串口接收字符串数据，实现串口接收字符串并通过接收的字符串判断是否打开继电器，达到用户可以正确使用串口通讯的目的。

二、范例分析



ASRPRO 开发板实物连接：



三、具体指令讲解

1.主程序设置

当串口接收到数据后，和程序设定的数据进行比较（可以是字符串或者是 16 进制），如果相同时引脚输出高低电平。

这里程序收到“on”时，PA4 输出高电平即继电器打开，同时语音播报“已经打开继电器”；当收到“off”时 PA4 输出低电平即继电器关闭，同时语音播报“已经关闭继电器”；

2.程序修改

PA_4 模拟继电器，输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。

语音识别“打开继电器”选择引脚 PA_4，输出高电平。

语音识别“关闭继电器”选择引脚 PA_4，输出低电平。

串口收到字符串“aabbcc”时，PA4 输出高电平即继电器打开。

串口收到字符串“ddeeff”时，PA4 输出低电平即继电器关闭。

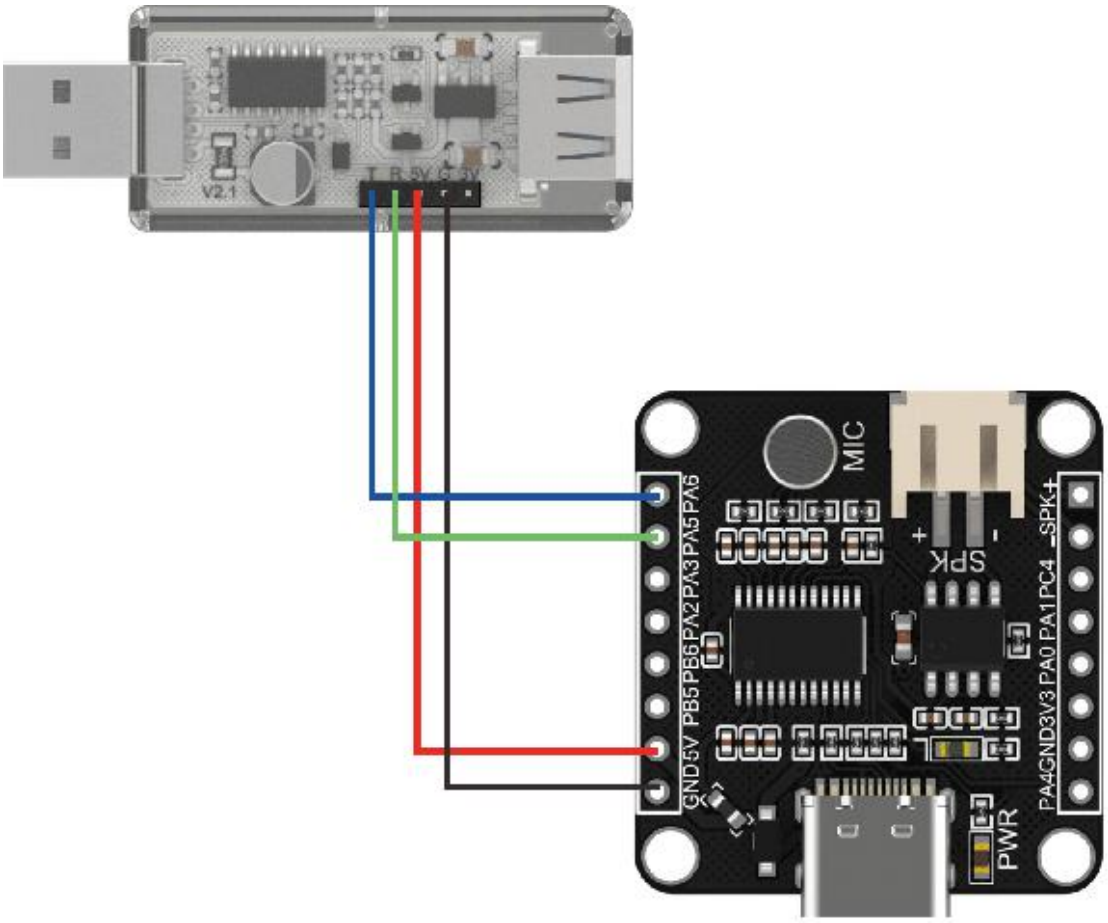
范例 2.11 串口 2 接收到字符串打开继电器

一、功能简介

本范例通过学习如何使用串口接收字符串数据，实现串口接收字符串并通过接收的字符串判断是否打开继电器，达到用户可以正确使用串口通讯的目的。

二、范例分析

ASRPRO 开发板实物连接：



三、具体指令讲解

1.主程序设置

当串口接收到数据后，和程序设定的数据进行比较（可以是字符串或者是 16 进制），如果相同时引脚输出高低电平。

这里程序收到“on”时，PA4 输出高电平即继电器打开，同时语音播报“已经打开继电器”；当收到“off”时 PA4 输出低电平即继电器关闭，同时语音播报“已经关闭继电器”；

2.程序修改

PA_4 模拟继电器，输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。

语音识别“打开继电器”选择引脚 PA_4，输出高电平。

语音识别“关闭继电器”选择引脚 PA_4，输出低电平。

串口收到字符串“aabbcc”时，PA4 输出高电平即继电器打开。

串口收到字符串“ddeeff”时，PA4 输出低电平即继电器关闭。

范例 2.12 串口 0 接收十六进制数打开继电器

一、功能简介

本范例通过学习如何使用串口接收十六进制数数据，实现串口接收十六进制数并通过接收的十六进制数判断是否打开继电器，达到用户可以正确使用串口通讯的目的。

二、范例分析

三、具体指令讲解

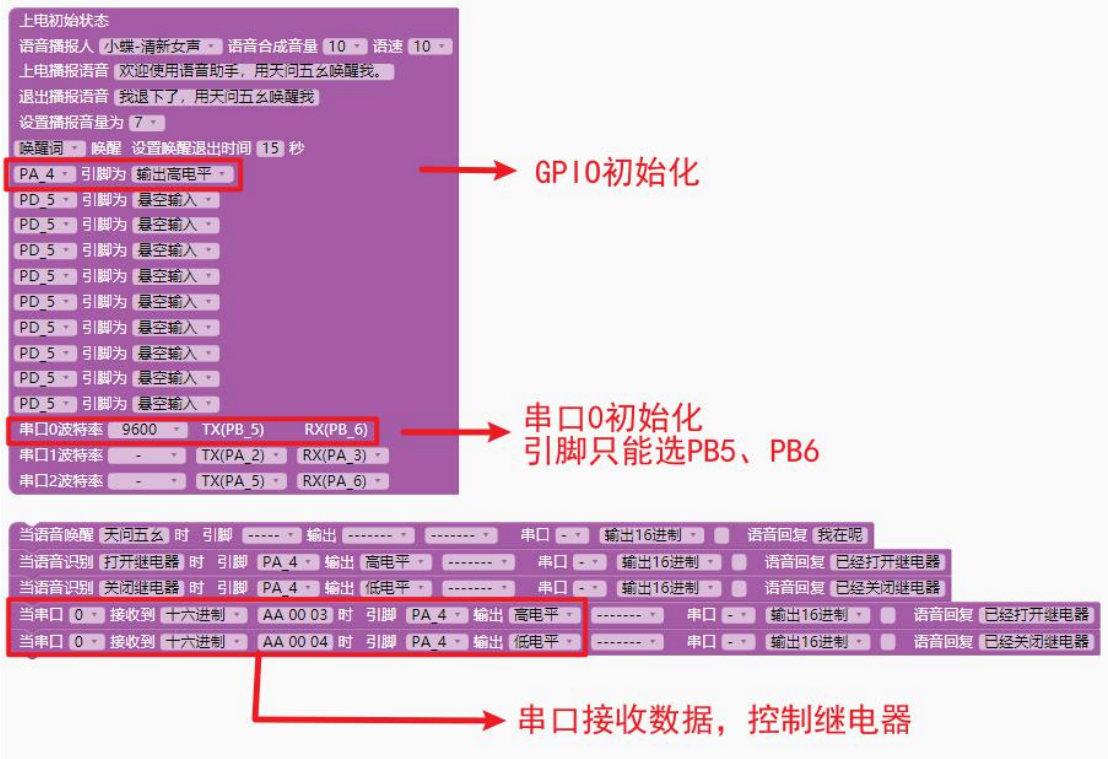
1.主程序设置

当串口接收到数据后，和程序设定的数据进行比较（可以是字符串或者是 16 进制），如果相同时引脚输出高低电平。

这里程序收到十六进制数“AA 00 01”时，PA4 输出高电平即继电器打开，同时语音播报“已经打开继电器”；

当收到十六进制数“AA 00 02”时 PA4 输出低电平即继电器关闭，同时语音播报“已经关闭继电器”；

2.程序修改



PA_4 模拟继电器，输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。

语音识别“打开继电器”选择引脚 PA_4，输出高电平。

语音识别“关闭继电器”选择引脚 PA_4，输出低电平。

串口收到十六进制数“AA 00 03”时，PA4 输出高电平即继电器打开。

串口收到十六进制数“AA 00 04”时，PA4 输出低电平即继电器关闭。

范例 2.13 串口 1 接收十六进制数打开继电器

一、功能简介

本范例通过学习如何使用串口接收十六进制数数据，实现串口接收十六进制数并通过接收的十六进制数判断是否打开继电器，达到用户可以正确使用串口通讯的目的。

二、范例分析

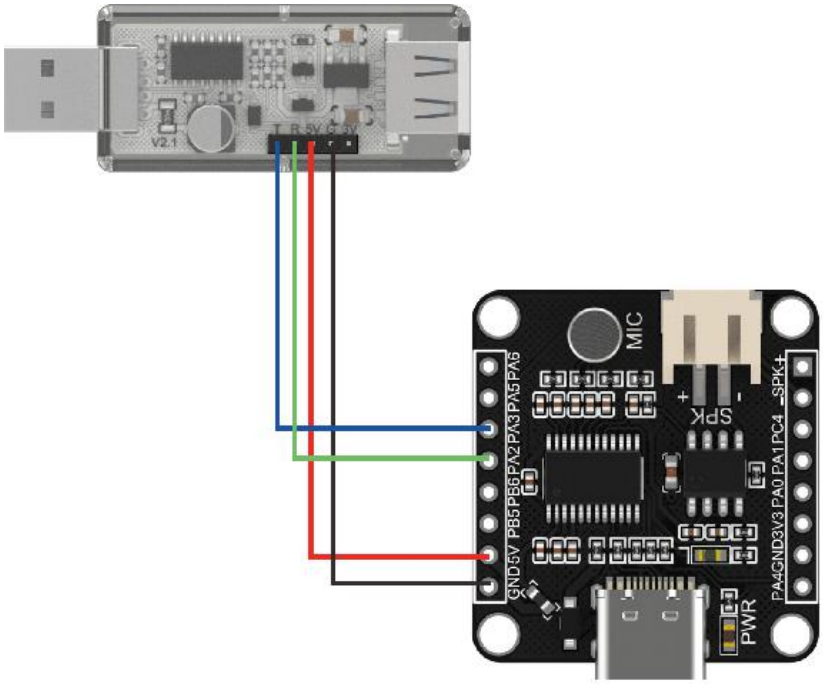
三、具体指令讲解

1.主程序设置

当串口接收到数据后，和程序设定的数据进行比较（可以是字符串或者是 16 进制），如果相同时引脚输出高低电平。这里程序收到十六进制数“AA 01 01”时，PA4 输出高电平即继电器打开，同时语音播报“已经打开继电器”；当收到十六进制数“AA 01 02”时 PA4 输出低电

平即继电器关闭，同时语音播报“已经关闭继电器”；

ASRPRO 开发板实物连接：



2.程序修改

PA_4 模拟继电器，输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。

语音识别“打开继电器”选择引脚 PA_4，输出高电平。语音识别“关闭继电器”选择引脚 PA_4，输出低电平。串口收到十六进制数“AA 01 03”时，PA4 输出高电平即继电器打开。串口收到十六进制数“AA 01 04”时，PA4 输出低电平即继电器关闭。

范例 2.14 串口 2 接收十六进制数打开继电器

一、功能简介

本范例通过学习如何使用串口接收十六进制数数据，实现串口接收十六进制数并通过接收的十六进制数判断是否打开继电器，达到用户可以正确使用串口通讯的目的。

二、范例分析

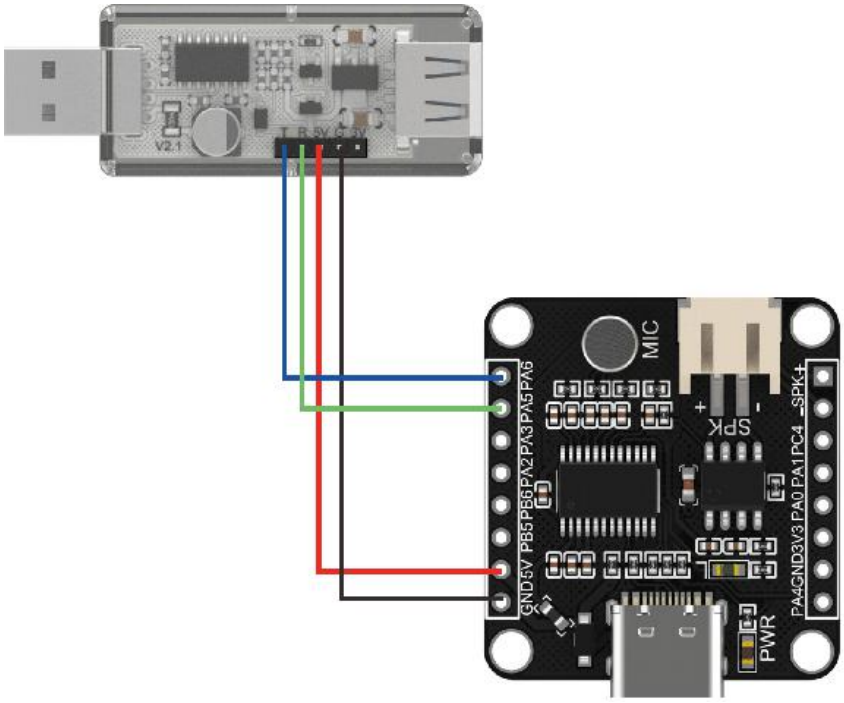
三、具体指令讲解

1.主程序设置

当串口接收到数据后，和程序设定的数据进行比较（可以是字符串或者是 16 进制），如果相同时引脚输出高低电平。这里程序收到十六进制数“AA 00 01”时，PA4 输出高电平即继电器打开，同时语音播报“已经打开继电器”；当收到十六进制数“AA 00 02”时 PA4 输出低电

平即继电器关闭，同时语音播报“已经关闭继电器”；

ASRPRO 开发板实物连接：



2.程序修改

PA_4 模拟继电器，输出高电平。所有涉及到串口的引脚在初始化设置中不进行设置。全部用 PD_5 代替，状态设置为悬空输入。

语音识别“打开继电器”选择引脚 PA_4，输出高电平。语音识别“关闭继电器”选择引脚 PA_4，输出低电平。串口收到十六进制数“AA 00 03”时，PA4 输出高电平即继电器打开。串口收到十六进制数“AA 00 04”时，PA4 输出低电平即继电器关闭。

附录一：语音识别设置注意事项

关于中英文语音识别还有一些注意事项，这里给出以下使用建议：

1. 一般为 4-6 个字，4 个字最佳，过短容误识高，过长不便用户呼叫和记忆；
2. 命令词中相邻汉字的声韵母区分度越大越好；
3. 符合用户的语言习惯，尽量采用常用说法，内容具体直接；
4. 应避免使用日常用语，如：“吃饭啦”；
5. 生僻字和零声母字应尽量避免，如“语音识别”中“语音”两个字均为零声母字；
6. 命令词中的字最好不要有语气词，如“啊”、“呢”等；
7. 应避免使用叠词，如：“你好你好”；
8. 中文命令词中只能由汉字组成,不允许有空格,逗号等其他字符；
9. 命令词中的数字需要以汉字表示，如“调高一度”；
10. 若您还未确定命令词，建议您从平台的“命令词推荐”中选择。
11. 英文建议由 2-4 个单词(4-6 个音节)组成，过短容误识高，过长不便用户记忆；
12. 英文命令词间音节区分度越大越好；
13. 英文的语音符合用户的语言习惯，尽量采用常用说法，内容具体直接；
14. 英文的唤醒词、命令词应避免使用日常用语，如：“HI、HELLO”；
15. 避免使用相似音节，词的发音清晰响度要大，如避免同时使用 TURN-ON 和 TURN-OFF ；
16. 应避免使用叠词，如：“HELLO -HELLO；

附录二：ASRPRO 与其他单片机串口通讯的范例说明

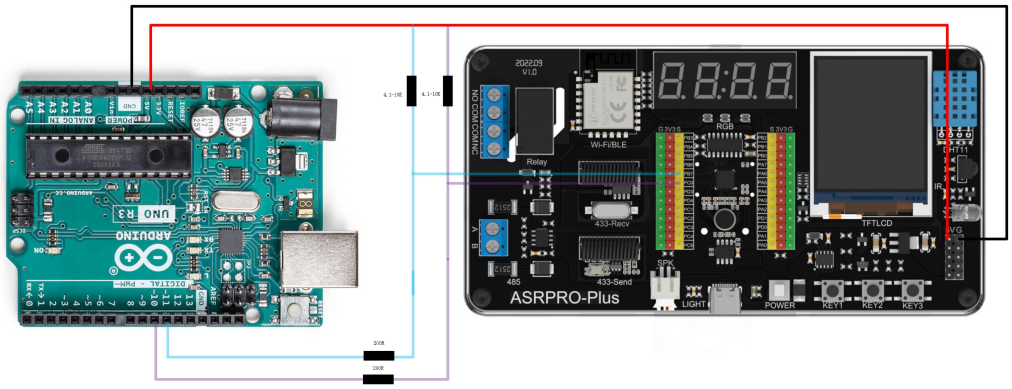
一、概述

ASRPRO 有 3 组串口，UART0 预留为程序升级接口，方便后期升级。如需和其它 MCU 通讯建议使用 UART1 或者 UART2。

ASRPRO IO 口为 3.3V 电平，为了可靠性，建议设置 TX、RX 引脚内部上下电阻无效，同时设置 TX 为开漏模式，外接上拉电阻到 5V，串联电阻，电路示意图如下所示：

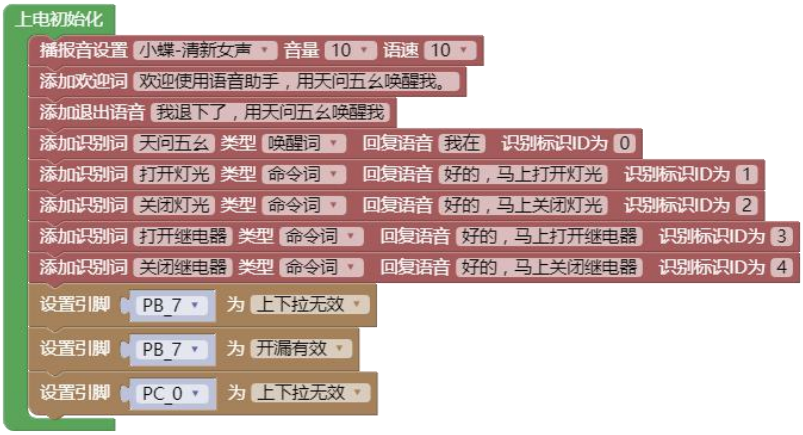
二、Arduino UNO（5V 单片机）

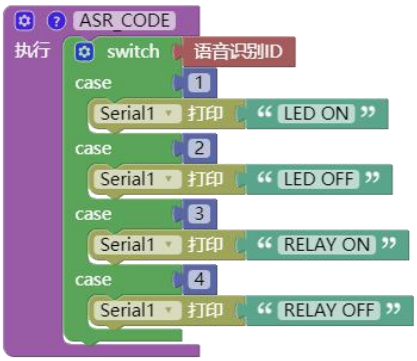
1. 电路连接



2. 范例 1：ASRPRO 语音发送串口控制 Arduino 执行动作

1) ASRPRO 端程序





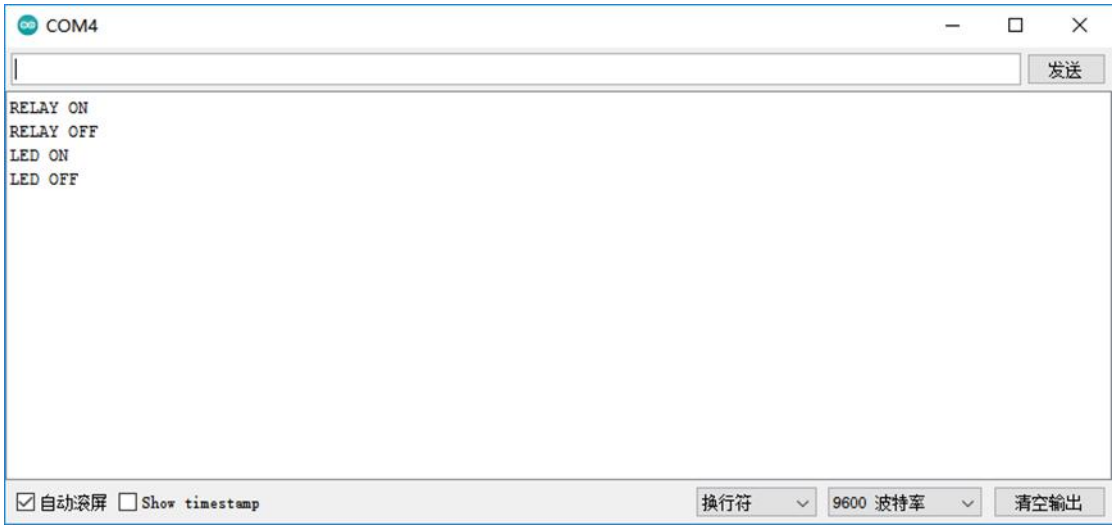
2) Arduino 端程序

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

String value;

#define LED_PIN   13
#define RELAY_PIN 12

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  mySerial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
  pinMode(RELAY_PIN, OUTPUT);
```

```
}

void loop() { // run over and over
  if (mySerial.available()) {
    value = (mySerial.readString());
    Serial.println(value);
    if (value == "LED ON")
    {
      digitalWrite(LED_PIN,HIGH);
    }
    else if (value == "LED OFF")
    {
      digitalWrite(LED_PIN,LOW);
    }
    else if (value == "RELAY ON")
    {
      digitalWrite(RELAY_PIN,HIGH);
    }
    else if (value == "RELAY OFF")
    {
      digitalWrite(RELAY_PIN,LOW);
    }
  }
}
```

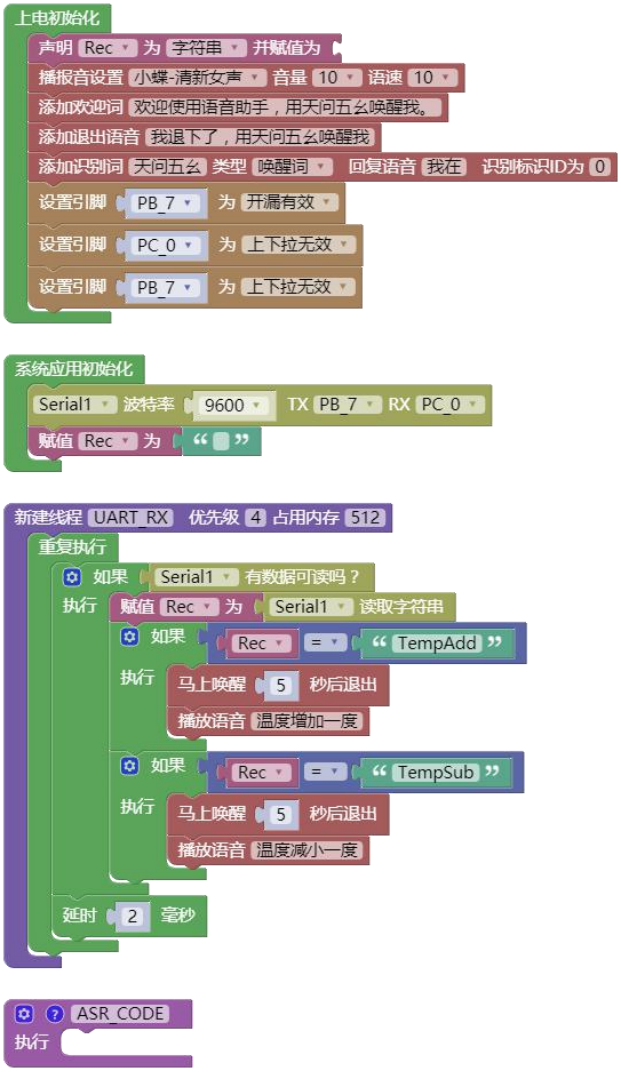
3) 程序效果

通过用“天问五幺”语音唤醒后，分别说测试语音“打开灯光”、“关闭灯光”、“打开继电器”、“关闭继电器”，Arduino 端接收到串口命令后会执行对应引脚的控制和串口打印。



3. 范例 2：Arduino 发送串口控制 ASRPRO 播放语音

1) ASRPRO 端程序



2) Arduino 端程序

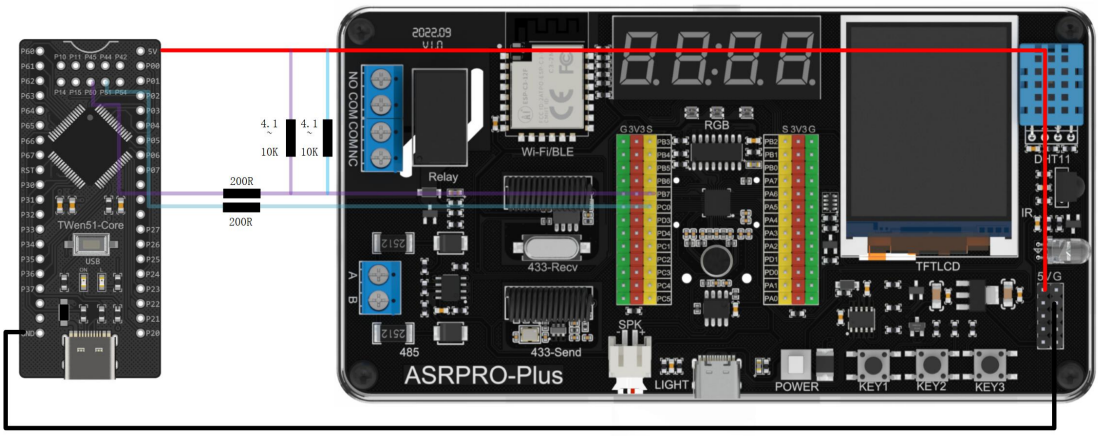
```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX
void setup() {
  mySerial.begin(9600);
}
void loop() { // run over and over
  mySerial.print("TempAdd");
  delay(5000);
  mySerial.print("TempSub");
  delay(5000);
}
```

3) 程序效果

Arduino 端间隔 5 秒串口发送“TempAdd”、"TempSub"，ASRPRO 接收到串口命令后会马上唤醒自动播报语音“温度增加一度”、“温度减小一度”。

三、STC（5V 单片机）

1. 电路连接

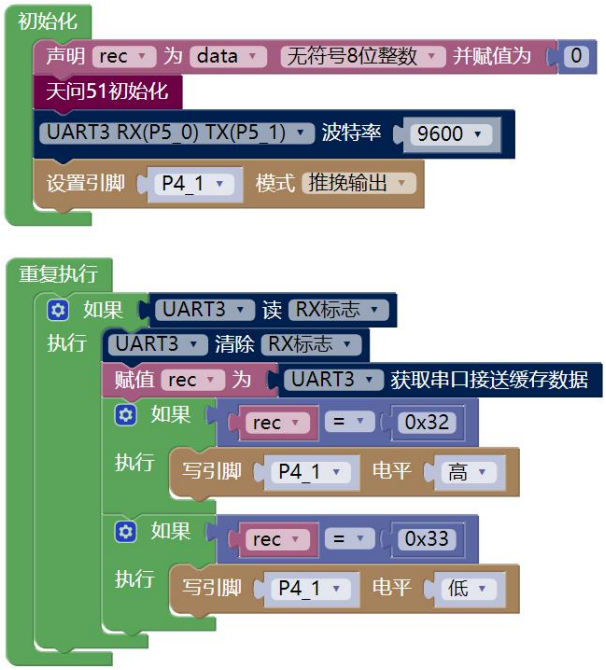


本案例以 P5_0 为 RX，P5_1 为 TX 举例。P5_0 与 ASRPRO 的 TX 连接，也就是接在 PB7，P5_1 与 ASRPRO 的 RX 连接，也就是接在 PC0，注意 STC8 的电源插脚接到 5V，GND 引脚互相连接。

2. 范例：ASRPRO 与 STC8 进行串口通讯语音控制 STC8 板载灯

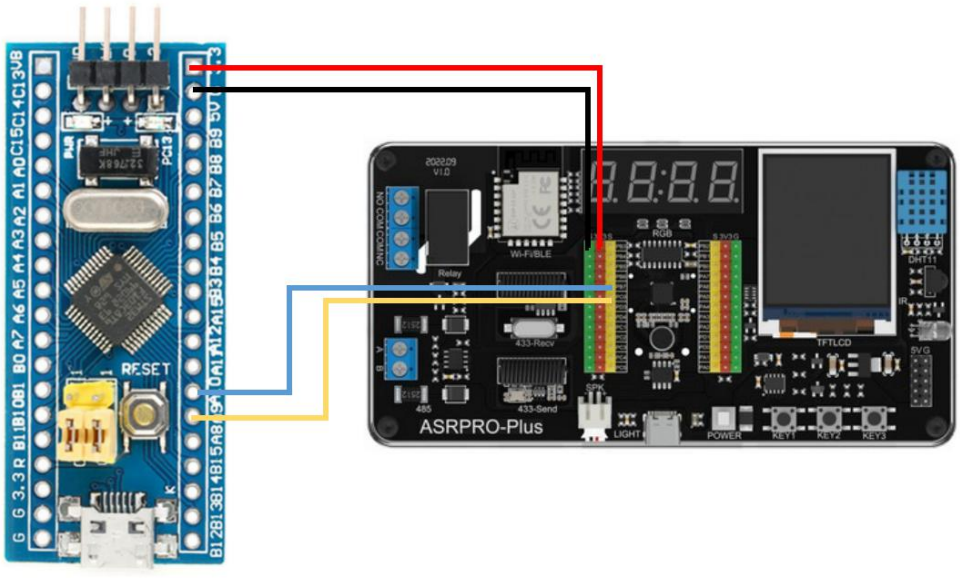
1) ASRPRO 端程序

2) STC8 端程序



四、STM32（3V 单片机）

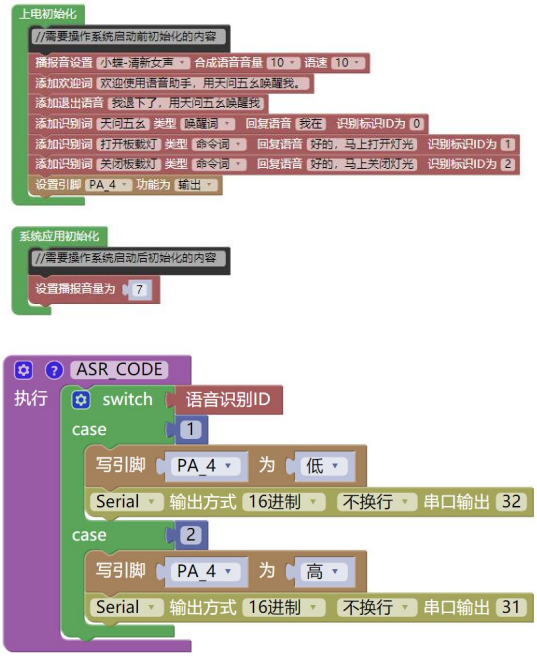
1. 电路连接



ASRPRO 的 PB7（TX）引脚接 STM32 的 A10（RX）引脚，PC0（RX）引脚接 STM32 的 A9（TX）引脚。

2. 范例 1：ASRPRO 语音发送串口控制 STM32 执行动作

1) ASRPRO 端程序



2) STM32 部分程序

main.c

```c
#include "stm32f10x.h"
#include "usart.h"
#include "led.h"
#include "delay.h"
u16 USART_RX_STA=0;                         //接收状态标记
static u16 fac_ms = 0;
int main(void)
{
    LED_Init();
    MyUSART_Init();
    delay_init();
    while(1)
    {
    }
}
```

usart.c

```c
#include "usart.h"
#include "led.h"
```

```
//重定向 C 库函数 printf 到串口，重定向后可使用 printf 函数
int fputc(int ch,FILE *f)
{
    /* 发送一个字节数据到串口 */
    USART_SendData(USART1,(uint8_t) ch);
    while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
    return (ch);
}
//重定向 C 库函数 scanf 到串口,重写向后可使用 scanf、getchar 等函数
int fgetc(FILE *f)
{
    /* 等待串口输入数据 */
    while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
    return (int)USART_ReceiveData(USART1);
}
void MyUSART_Init()
{
    /* 定义 GPIO、NVIC 和 USART 初始化的结构体 */
    GPIO_InitTypeDef GPIO_InitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;
    USART_InitTypeDef USART_InitStructure;
    /* 使能 GPIO 和 USART 的时钟 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);
    /* 将 USART TX（A9）的 GPIO 设置为推挽复用模式 */
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_Init(GPIOA,&GPIO_InitStructure);
    /* 将 USART RX（A10）的 GPIO 设置为浮空输入模式 */
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IN_FLOATING;
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_10;
    GPIO_Init(GPIOA,&GPIO_InitStructure);

    /* 配置串口 */
    USART_InitStructure.USART_BaudRate=9600;                                    /
/波特率了设置为 9600
```

```
    USART_InitStructure.USART_HardwareFlowControl=USART_HardwareFlowControl_None;
//不使用硬件流控制
    USART_InitStructure.USART_Mode=USART_Mode_Tx|USART_Mode_Rx;
//使能接收和发送
    USART_InitStructure.USART_Parity=USART_Parity_No;                              /
/不使用奇偶校验位
    USART_InitStructure.USART_StopBits=USART_StopBits_1;                           /
/1 位停止位
    USART_InitStructure.USART_WordLength=USART_WordLength_8b;                      /
/字长设置为 8 位
    USART_Init(USART1, &USART_InitStructure);

    /* Usart1 NVIC 配置 */
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);                //设置 NVIC 中断分
组 2
    NVIC_InitStructure.NVIC_IRQChannel=USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelCmd=ENABLE;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority=1;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority=0;
    NVIC_Init(&NVIC_InitStructure);

    /*初始化串口，开启串口接收中断 */
    USART_ITConfig(USART1,USART_IT_RXNE,ENABLE);
    /* 使能串口 1 */
    USART_Cmd(USART1,ENABLE);

}
/* USART1 中断函数 */
void USART1_IRQHandler(void)
{
    uint8_t ucTemp;                                        //接收数据
    if(USART_GetITStatus(USART1,USART_IT_RXNE)!=RESET)
    {
        ucTemp = USART_ReceiveData(USART1);
        USART_SendData(USART1,ucTemp);
        if(ucTemp == 0x32)
        {
            LED_ON();
```

```
        }
        if(ucTemp == 0x31)
         {
            LED_OFF();
        }
    }
}
/* 发送一个字节 */
void Usart_SendByte( USART_TypeDef * pUSARTx, uint8_t ch)
{
    /* 发送一个字节数据到 USART */
    USART_SendData(pUSARTx,ch);

    /* 等待发送数据寄存器为空 */
    while (USART_GetFlagStatus(pUSARTx, USART_FLAG_TXE) == RESET);
}
/* 发送字符串 */
void Usart_SendString( USART_TypeDef * pUSARTx, char *str)
{
    unsigned int k=0;
  do
  {
      Usart_SendByte( pUSARTx, *(str + k) );
      k++;
  } while(*(str + k)!='\0');

  /* 等待发送完成 */
  while(USART_GetFlagStatus(pUSARTx,USART_FLAG_TC)==RESET)
  {}
}


}
```

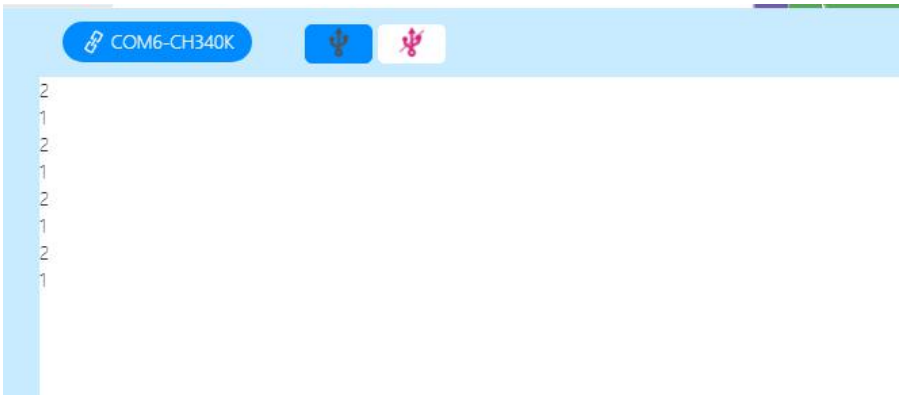
led.c

```
#include "led.h"
void LED_Init(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;
```

```
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB|RCC_APB2Periph_AFIO, ENABLE);    //
使能 B 端口时钟
    GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);
     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;         //推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;//速度 50MHz
    GPIO_Init(GPIOB, &GPIO_InitStructure);    //初始化 GPIOB
    GPIO_SetBits(GPIOB,GPIO_Pin_10);
//  GPIO_SetBits(GPIOA,GPIO_Pin_8);
}
void LED_OFF(void)
{
    GPIO_SetBits(GPIOB,GPIO_Pin_10);
}
void LED_ON(void)
{
    GPIO_ResetBits(GPIOB,GPIO_Pin_10);
}
```

3) 程序效果

通过用“天问五幺”语音唤醒后，分别说测试语音“打开灯光”、“关闭灯光”，STM32 端接收到串口命令后会执行对应引脚的控制和串口打印，“2”代表打开，“1”代表关闭。



3. 范例 2：STM32 串口发送控制 ASRPRO 播报语音

1) ASRPRO 端程序

2) STM32 部分程序

main.c

```
#include "stm32f10x.h"
#include "usart.h"
#include "led.h"
#include "delay.h"

u16 USART_RX_STA=0;                         //接收状态标记
static u16 fac_ms = 0;
//void delay_init(void);
//void delay_ms(u16 nms);
int main(void)
{
```

```
    LED_Init();
    MyUSART_Init();
    delay_init();
    while(1)
    {
        Usart_SendString( USART1,"LED ON");
        LED_ON();
        delay_ms(5000);
        Usart_SendString( USART1,"LED OFF");
        LED_OFF();
        delay_ms(5000);
    }
}
```

usart.c

```
#include "usart.h"
#include "led.h"

//重定向 C 库函数 printf 到串口，重定向后可使用 printf 函数
int fputc(int ch,FILE *f)
{
    /* 发送一个字节数据到串口 */
    USART_SendData(USART1,(uint8_t) ch);
    while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
    return (ch);
}
//重定向 C 库函数 scanf 到串口,重写向后可使用 scanf、getchar 等函数
int fgetc(FILE *f)
{
    /* 等待串口输入数据 */
    while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
    return (int)USART_ReceiveData(USART1);
}
void MyUSART_Init()
{
    /* 定义 GPIO、NVIC 和 USART 初始化的结构体 */
    GPIO_InitTypeDef GPIO_InitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;
    USART_InitTypeDef USART_InitStructure;
```

```
    /* 使能 GPIO 和 USART 的时钟 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);
    /* 将 USART TX（A9）的 GPIO 设置为推挽复用模式 */
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_Init(GPIOA,&GPIO_InitStructure);
    /* 将 USART RX（A10）的 GPIO 设置为浮空输入模式 */
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IN_FLOATING;
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_10;
    GPIO_Init(GPIOA,&GPIO_InitStructure);

    /* 配置串口 */
    USART_InitStructure.USART_BaudRate=9600;                                    /
/波特率了设置为 9600
    USART_InitStructure.USART_HardwareFlowControl=USART_HardwareFlowControl_None;
//不使用硬件流控制
    USART_InitStructure.USART_Mode=USART_Mode_Tx|USART_Mode_Rx;                  /
/使能接收和发送
    USART_InitStructure.USART_Parity=USART_Parity_No;                           /
/不使用奇偶校验位
    USART_InitStructure.USART_StopBits=USART_StopBits_1;                        /
/1 位停止位
    USART_InitStructure.USART_WordLength=USART_WordLength_8b;                   /
/字长设置为 8 位
    USART_Init(USART1, &USART_InitStructure);

    /* Usart1 NVIC 配置 */
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);                 //设置 NVIC 中断分
组 2
    NVIC_InitStructure.NVIC_IRQChannel=USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelCmd=ENABLE;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority=1;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority=0;
    NVIC_Init(&NVIC_InitStructure);

    /*初始化串口，开启串口接收中断 */
```

```
    USART_ITConfig(USART1,USART_IT_RXNE,ENABLE);
    /* 使能串口 1 */
    USART_Cmd(USART1,ENABLE);

}
/* USART1 中断函数 */
void USART1_IRQHandler(void)
{
    uint8_t ucTemp;                                          //接收数据
    if(USART_GetITStatus(USART1,USART_IT_RXNE)!=RESET)
    {
        ucTemp = USART_ReceiveData(USART1);
        USART_SendData(USART1,ucTemp);
        if(ucTemp == 0x32)
        {
            LED_ON();
        }
        if(ucTemp == 0x31)
        {
            LED_OFF();
        }
    }
}
```

```
/* 发送一个字节 */
void Usart_SendByte( USART_TypeDef * pUSARTx, uint8_t ch)
{
    /* 发送一个字节数据到 USART */
    USART_SendData(pUSARTx,ch);

    /* 等待发送数据寄存器为空 */
    while (USART_GetFlagStatus(pUSARTx, USART_FLAG_TXE) == RESET);
}
/* 发送字符串 */
void Usart_SendString( USART_TypeDef * pUSARTx, char *str)
{
    unsigned int k=0;
  do
```

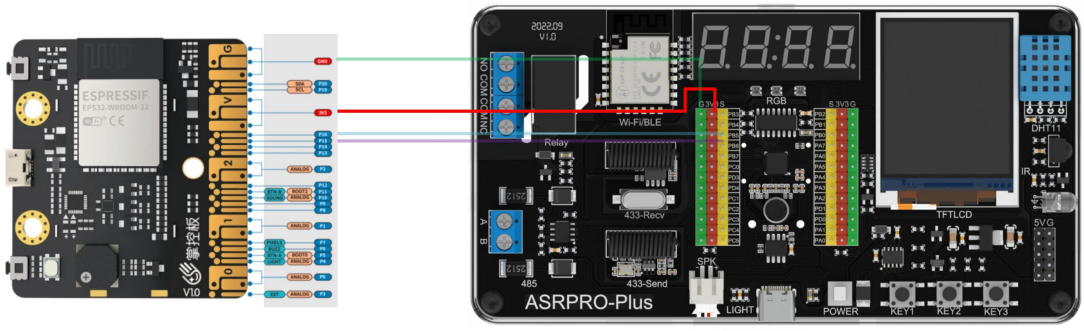
```
  {
      Usart_SendByte( pUSARTx, *(str + k) );
      k++;
  } while(*(str + k)!='\0');


  /* 等待发送完成 */
  while(USART_GetFlagStatus(pUSARTx,USART_FLAG_TC)==RESET)
  {}
}
```

3) 程序效果

STM32 端间隔一段时间串口发送“LED ON”、"LED OFF"，ASRPRO 接收到串口命令后会马上唤醒自动播报语音“灯光已打开”、“灯光已关闭”。

五、ESP32（3V 单片机）

1. 电路连接



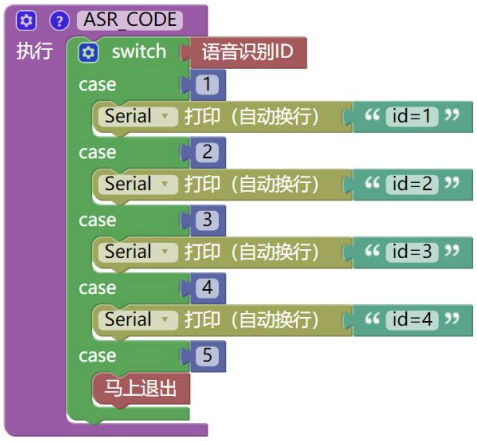
掌控板的 P15 引脚的 TX，接 ASRPRO 的 RX，也就是接在 PB6；P16 引脚接到 ASRPRO RX 引脚（PB5），两者的 3V 引脚互相连接，GND 引脚互相连接。

2. 范例：ASRPRO 与掌控板进行串口通讯

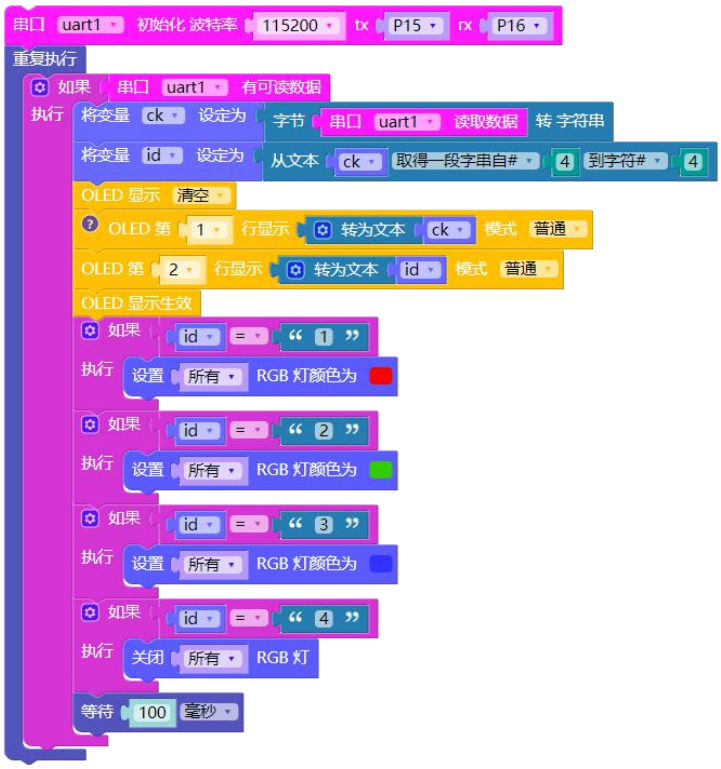
语音控制 ASRPRO 发送串口数据，并控制掌控板的板载 RGB 灯显示不同的颜色。

1) ASRPRO 端程序


上电初始化
播报音设置 小蝶-清新女声 音量 6 语速 10
添加欢迎词 你好，我是您的智能语音助手，请用天问五幺唤醒我
添加退出语音 我休息了，用天问五幺唤醒我
添加识别词 天问五幺 类型 唤醒词 回复语音 我在 识别标识ID为 0
添加识别词 打开红灯 类型 命令词 回复语音 好的 识别标识ID为 1
添加识别词 打开绿灯 类型 命令词 回复语音 好的 识别标识ID为 2
添加识别词 打开蓝灯 类型 命令词 回复语音 好的 识别标识ID为 3
添加识别词 关闭灯光 类型 命令词 回复语音 好的 识别标识ID为 4
添加识别词 退下吧 类型 命令词 回复语音 好的 识别标识ID为 5
唤醒词 唤醒
Serial 波特率 115200 TX PB_5 RX PB_6

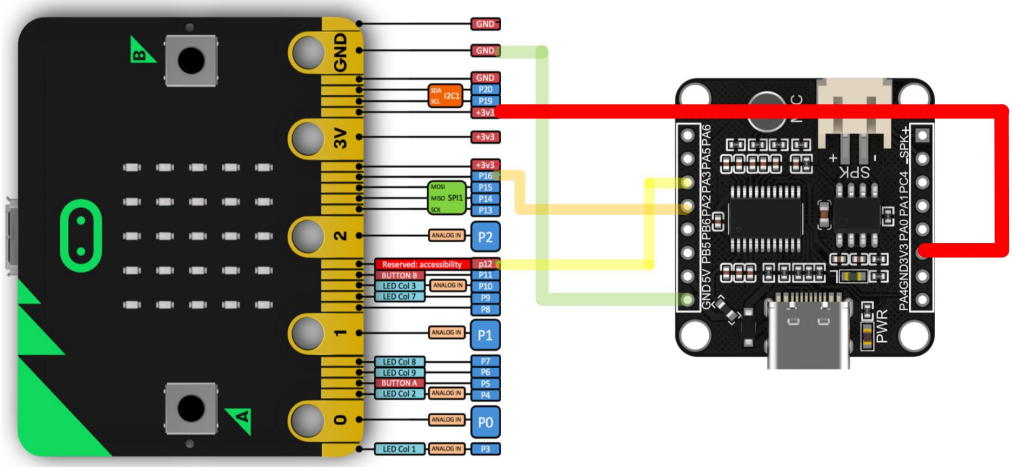

ASR_CODE
执行
switch 语音识别ID
case 1
Serial 打印（自动换行） “ id=1 ”
case 2
Serial 打印（自动换行） “ id=2 ”
case 3
Serial 打印（自动换行） “ id=3 ”
case 4
Serial 打印（自动换行） “ id=4 ”
case 5
马上退出

2) 掌控板端程序


串口 uart1 初始化 波特率 115200 tx P15 rx P16
重复执行
如果 串口 uart1 有可读数据
执行
将变量 ck 设定为 字节 串口 uart1 读取数据 转 字符串
将变量 id 设定为 从文本 ck 取得一段字串自# 4 到字符# 4
OLED 显示 清空
OLED 第 1 行显示 转为文本 ck 模式 普通
OLED 第 2 行显示 转为文本 id 模式 普通
OLED 显示生效
如果 id = “ 1 ”
执行 设置 所有 RGB 灯颜色为
如果 id = “ 2 ”
执行 设置 所有 RGB 灯颜色为
如果 id = “ 3 ”
执行 设置 所有 RGB 灯颜色为
如果 id = “ 4 ”
执行 关闭 所有 RGB 灯
等待 100 毫秒

六、micro:bit（3V 单片机）

1. 电路连接



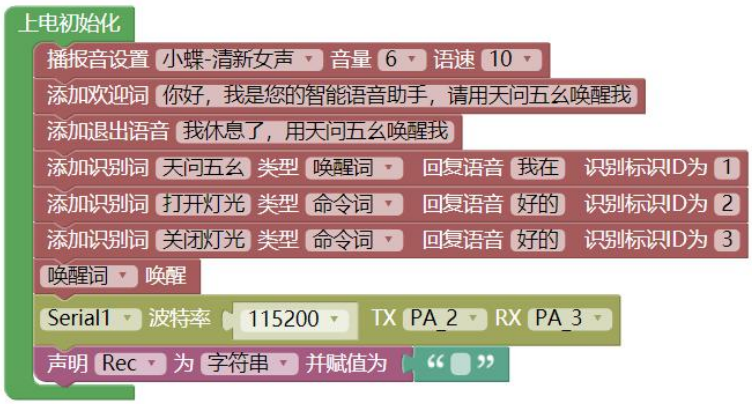
在进行串口通信时，两个设备进行双向通信，此时两个设备的 RX 和 TX 要交错连接。例如 micro:bit，定义 P16 为 RX 口，则要接到 ASRPRO 的 TX 上，也就是 PA2。

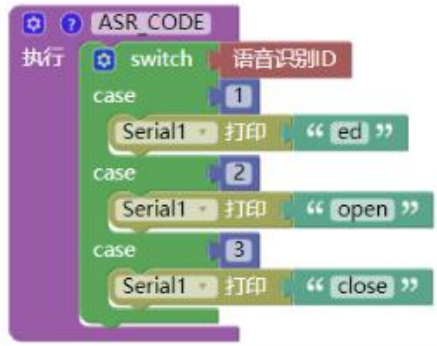
micro:bit 的 P16 引脚的 RX，接 ASRPRO 的 TX，也就是接在 PA2；P12 引脚接到 ASRPRO 的 RX 引脚（PA3），两者的 3V 引脚互相连接，GND 引脚互相连接。

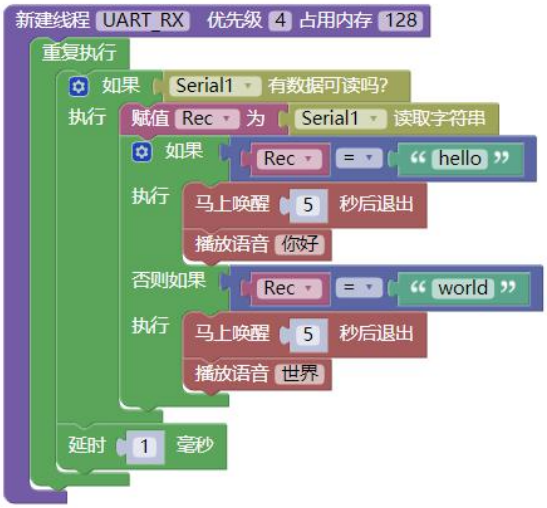
2. 范例：ASRPRO 与 micro:bit 进行串口通讯

按下 micro:bit 的 AB 按键，通过串口可以给 ASRPRO 发送字符串 hello 和 world；当 ASRPRO 接收到后，就会回复对应的语音；当对 ASRPRO 说出“打开灯光、关闭灯光”时，ASRPRO 和 micro:bit 的串口信息会显示在 micro:bit 的点阵屏上。

1) ASRPRO 程序







2) micro:bit 程序