

语音模块 使用手册

PROGRAMMABLE ROBOT KIT



目录

1. 模块介绍.....	4
2. 工作原理.....	4
3. 模块特点.....	错误！未定义书签。
4. 模块参数.....	错误！未定义书签。
5. 使用示例.....	4
5.1 Arduino 示例.....	5
5.1.1 准备工作.....	5
5.1.2 示例程序.....	6
5.1.3 实现效果.....	7
5.2 stm32 示例.....	8
5.2.1 准备工作.....	8
5.2.2 示例程序.....	9
5.2.3 实现效果.....	14
5.3 Raspberry 示例.....	15
5.3.1 准备工作.....	15
5.3.2 示例程序.....	19
5.3.3 实现效果.....	20
5.4 micro:bit 示例.....	21
5.4.1 准备工作.....	26
5.4.2 示例程序.....	27

5.4.3 实现效果.....	29
-----------------	----

产品介绍篇

1.概述

语音识别模块是基于嵌入式的语音识别技术的模块，主要包括语音识别芯片和一些其他的附属电路，能够方便的与主控芯片进行通讯，开发者可以方便的将该模块嵌入到产品中使用，实现语音交互的目的。

2.工作原理

采用 IIC 通信，用户只需要把识别的关键词语以字符串的形式传送进芯片，即可在下次识别中立即生效。

该模块有三种使用模式，用户可通过编程，设置两种不同的使用模式。

◆ **按钮检测模式：**系统的主控 MCU 在接收到外界一个触发后（比如用户按下按键），将会启动芯片上的一个定时识别过程（比如 5 s），此时需要用户在这个定时过程中说出要识别的语音关键词语。当这个过程结束后，需要用户再次触发才能再次启动一个识别过程。

◆ **循环检测模式：**系统的主控 MCU 反复启动的识别过程。如果没有人说话就没有识别结果，则每次识别过程的定时到时后会再启动一个识别过程；如果有识别结果，则根据识别作相应处理后（比如播放某个声音作为回答）再启动一个识别过程。

◆ **口令检测模式：**口令模式需要一个关键词来唤醒，唤醒后才可以进行识别，默认唤醒关键词为第一句，识别结束后，想再次进行识别，还需唤醒它，类似小艾同学。

3.应用领域

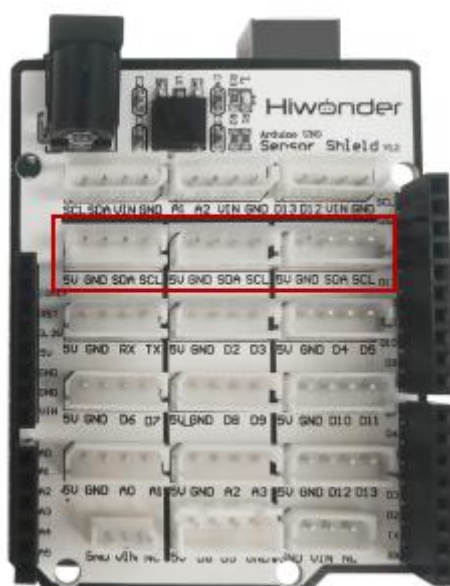
主打傻瓜式的简易操作、出色的语音识别性能，使其应用非常广泛，例如像智能家居、对话机器人、教育机器人、车载调度终端等方面。

案例快速上手篇

案例一：Arduino 示例

1.准备工作

本节示例采用 Arduino UNO+Arduino 扩展板，传感器需要使用 4P 线连接至 Arduino 扩展板上任意一个 IIC 接口。



如果不使用 Arduino 扩展板，则要使用杜邦线将传感器连接至 Arduino 开发板上。

开发板类型	接口
Arduino Board	I2C/TWI pins
Arduino UNO/Ethernet	A4 (SDA) , A5 (SCL)
Arduino Mega2560	20 (SDA) , 21 (SCL)
Arduino Leonardo	2 (SDA) , 3 (SCL)

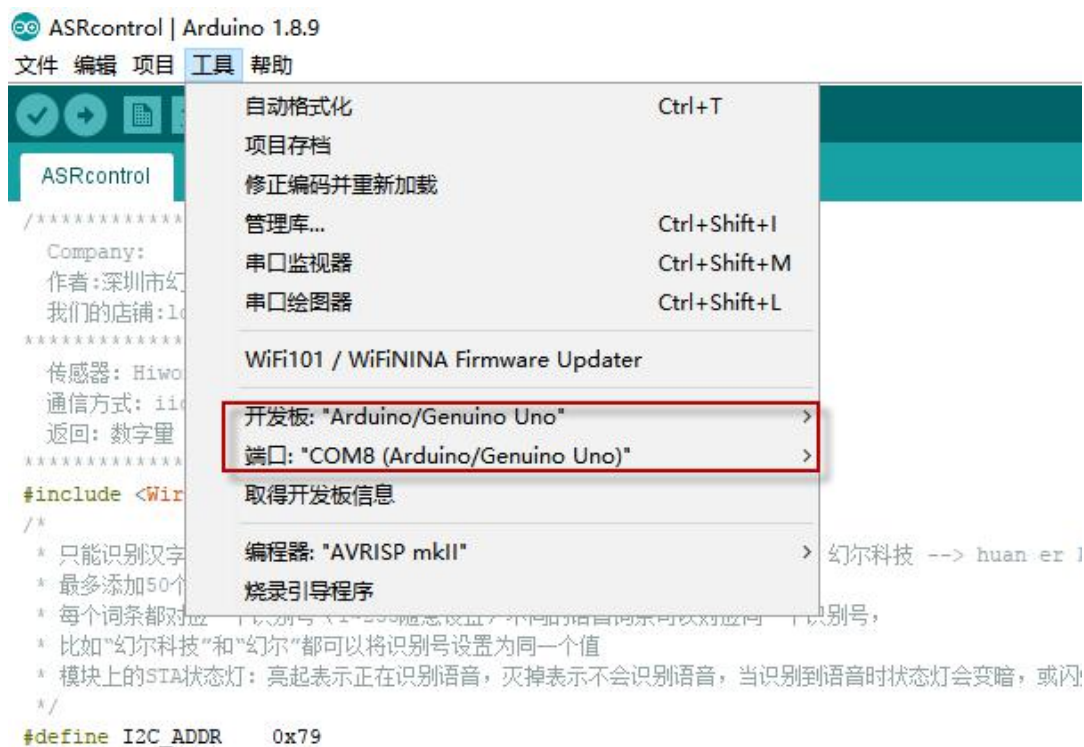
Arduino Due	20 (SDA) , 21 (SCL) , SDA1, SCL1
-------------	----------------------------------

2.实验流程

- 1) 将装有扩展板的 Arduino UNO 开发板通过 USB 数据线连入电脑。
- 2) 打开本文件夹内的 Arduino 示例程序。



- 3) 选择正确的开发板及端口（本节以 UNO 为例，示例所用端口号为 8，用户需根据实际情况自行选择）。



- 4) 在程序中一共有三种模式可选择并测试（默认为循环识别模式）。如果需要更改口

令或按键模式，需要在下方图示位置修改数值（因图文展示有限，详情可参考代码注释）。

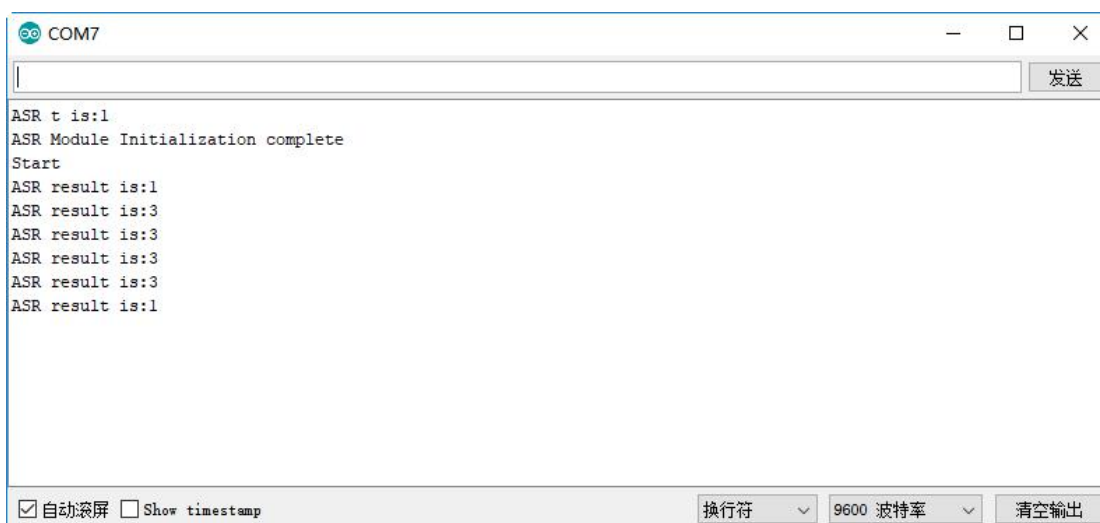
```
void setup()
{
  uint8_t ASRMode = 1; //1: 循环识别模式    2: 口令模式，以第一个词条为口令    3: 按键模式，按下开始识别
  Wire.begin();
  Serial.begin(9600);
}
```

5) 将程序上传至开发板即可。

3.实现效果

对准语音识别模块任意说出以下字样（程序设定）：“开始”、“大家庭”、“幻尔科技”、“幻尔”、“深圳市”，识别成功语音识别模块的 STA 指示灯会闪烁，并将在串口监视器中打印对应的数字结果。





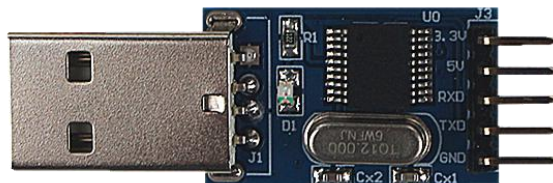
案例二：stm32 示例

1.准备工作

1) 本节示例使用的是 stm32F103 单片机和开源小车控制板。根据程序设定，需要将传感器的用 4PIN 线连接至控制板的 IIC 接口。



2) 控制板使用 7.5V 锂电池供电，下载程序使用 USB 转 TTL 串口下载器。



3) 安装文件夹内的 Keil 工具并激活完毕（详细安装及使用方法参考文件夹内的说明文档）。

4) 安装文件夹内的串口调试助手工具，以便于程序烧录后查看结果。

2.实验流程

1) 打开传感器对应的 stm32 的程序。

CORE	2019/9/27 星期...	文件夹	
Obj	2019/9/27 星期...	文件夹	
STM32F10x_FWLib	2019/9/27 星期...	文件夹	
USER	2019/9/27 星期...	文件夹	
OpenArmSTM32.uvgui.Administrator	2019/9/26 星期...	ADMINISTRATO...	70 KB
OpenArmSTM32.uvgui.cp	2019/9/26 星期...	CP 文件	72 KB
OpenArmSTM32.uvgui_Administrator....	2019/9/26 星期...	BAK 文件	70 KB
OpenArmSTM32.uvopt	2019/9/26 星期...	UVOPT 文件	21 KB
OpenArmSTM32.uvproj	2019/9/26 星期...	碘ision4 Project	21 KB
OpenArmSTM32_Target 1.dep	2019/9/27 星期...	DEP 文件	55 KB
OpenArmSTM32_uvopt.bak	2019/9/26 星期...	BAK 文件	21 KB
OpenArmSTM32_uvproj.bak	2019/9/26 星期...	BAK 文件	22 KB

2) 传感器接口定义位置如下图所示, GPIO_Pin_1 和 GPIO_Pin_12 分别对应控制板的 E2 和 E3 接口。

```

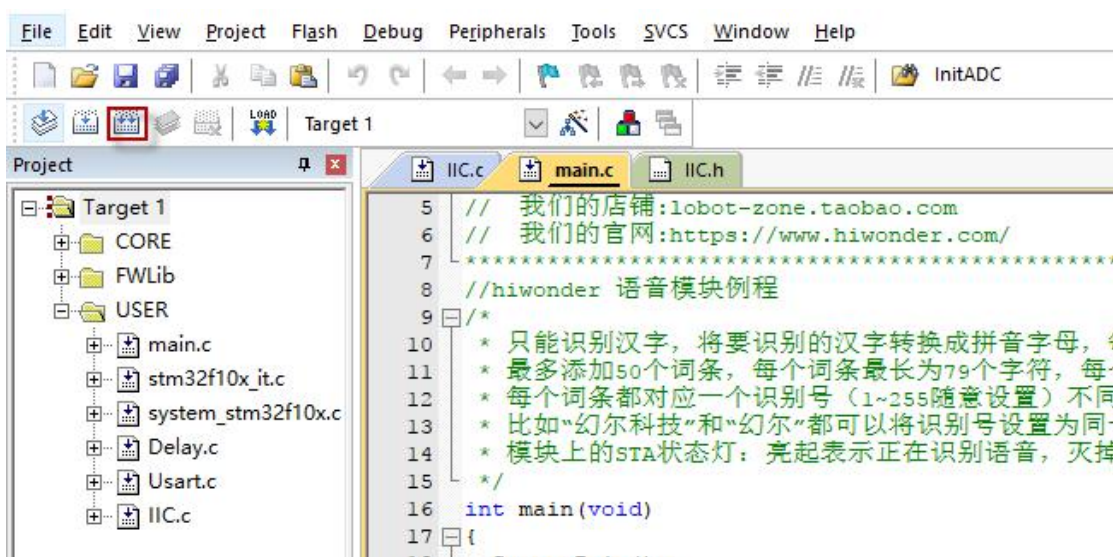
1  #ifndef __IIC_H
2  #define __IIC_H
3
4  //*****软件模拟IIC,
5  //*****修改宏定义即可
6  //*****不同芯片时注意时钟和延时函数
7  #define IIC_IO_SDA      GPIO_Pin_12  //SDA的IO口
8  #define IIC_IO_SCL      GPIO_Pin_1  //SCL的IO口
9  #define GPIOX           GPIOB        //GPIOx选择
10 #define CLOCK           RCC_APB2Periph_GPIOB //时钟信号
11
12 #define IIC_SCL          PBout(1) //SCL
13 #define IIC_SDA          PBout(12) //输出SDA
14 #define READ_SDA        PBin(12) //输入SDA
15
16 #define Asr_Addr         0xF2
17 #define Asr_Read         0xF3
18

```

3) 程序添加词条位置及模式如下图所示, 这里我们以默认设定一循环识别模式为例。
如果需要更改模式或添加词条, 请参考代码详情注释。

```
23 int main(void)
24 {
25     SystemInit();
26     InitDelay(72); //初始化延时函数
27     Usart1_Init(); //串口初始化
28     IIC_Init(); //IIC初始化
29     DelayMs(200);
30
31     if (1) //添加的词条和识别模式是可以掉电保存的，第一次设置完成后，可以将此段注释掉，即将1改为0，然后重新下载一次程序
32     {
33         AsrErase(); //擦除
34         DelayMs(60);
35
36         AsrAddWords(1, (u8*)TEXT1_Buffer, strlen(TEXT1_Buffer));
37         DelayMs(60);
38         AsrAddWords(2, (u8*)TEXT2_Buffer, strlen(TEXT2_Buffer));
39         DelayMs(60);
40         AsrSetMode(1); //设置模式
41         //1: 循环识别模式。状态灯常亮（默认模式）
42         //2: 口令模式，以第一个词条为口令。状态灯常灭，当识别到口令时常亮，等待识别到新的语音，并且读取识别结果后即灭掉
43         //3: 按键模式，按下开始识别，不按不识别。支持掉电保存。状态灯随按键按下而亮起，不按不亮
44         DelayMs(60);
45     }
46
47     while (1)
48     {
49         u8 result;
50         result = Asr_Result(); //返回识别结果，即识别到的词条编号
51         if (result)
52         {
53             printf("%d", result);
54             printf("\n");
55         }
56         DelayMs(500);
57     }
58 }
59
60 }
```

4) 将所有代码文件生成可执行的文件。

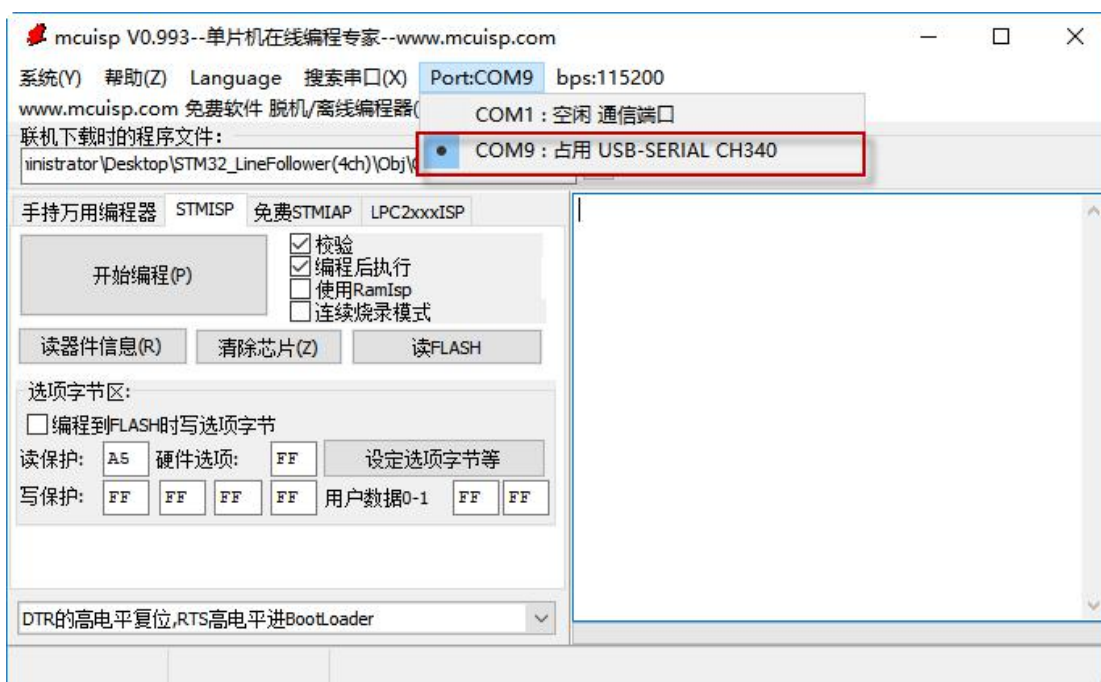


5) 将控制板通过 USB 转 TTL 工具连接至电脑的任意一个 USB 接口。

6) 打开程序烧写工具。



7) 需要选择设备的正确端口，波特率维持不变即可。



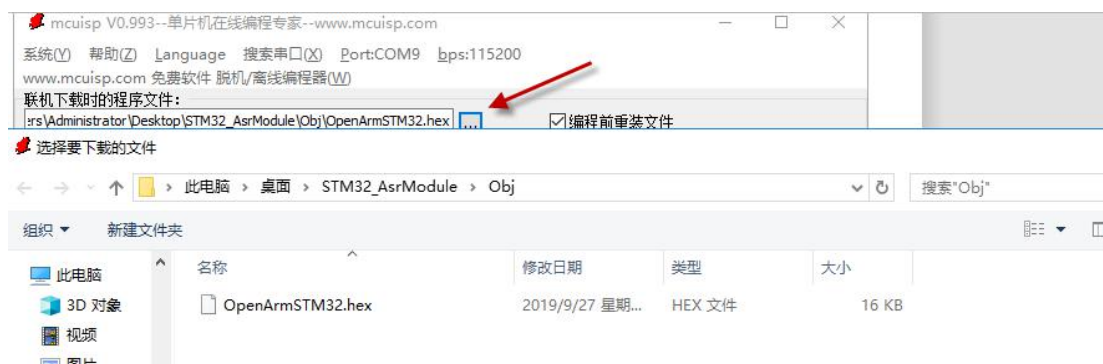
8) 选项卡模式选择“STMISP”，如下图所示，其他保持默认。



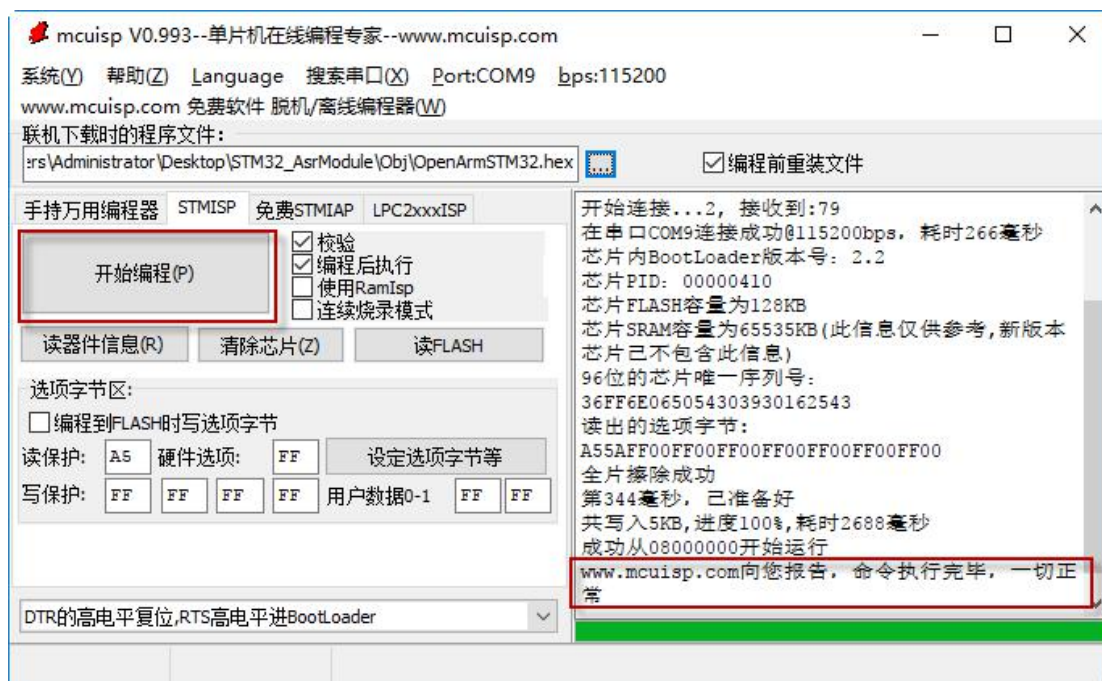
9) 拔掉 stm32 单片机上的跳线帽。(跳线帽用于切换正常运行模式和烧写模式，拔下时为烧写模式，插上时为正常运行模式。拔下跳线帽后上电，或者是在上电情况下拔下跳线帽后按一下核心板上 RST 按钮，即可进入烧写模式)



10) 参考下方图示选项位置，打开第 3 步创建的可执行文件，路径一般为该程序文件夹内的 Obj 内。

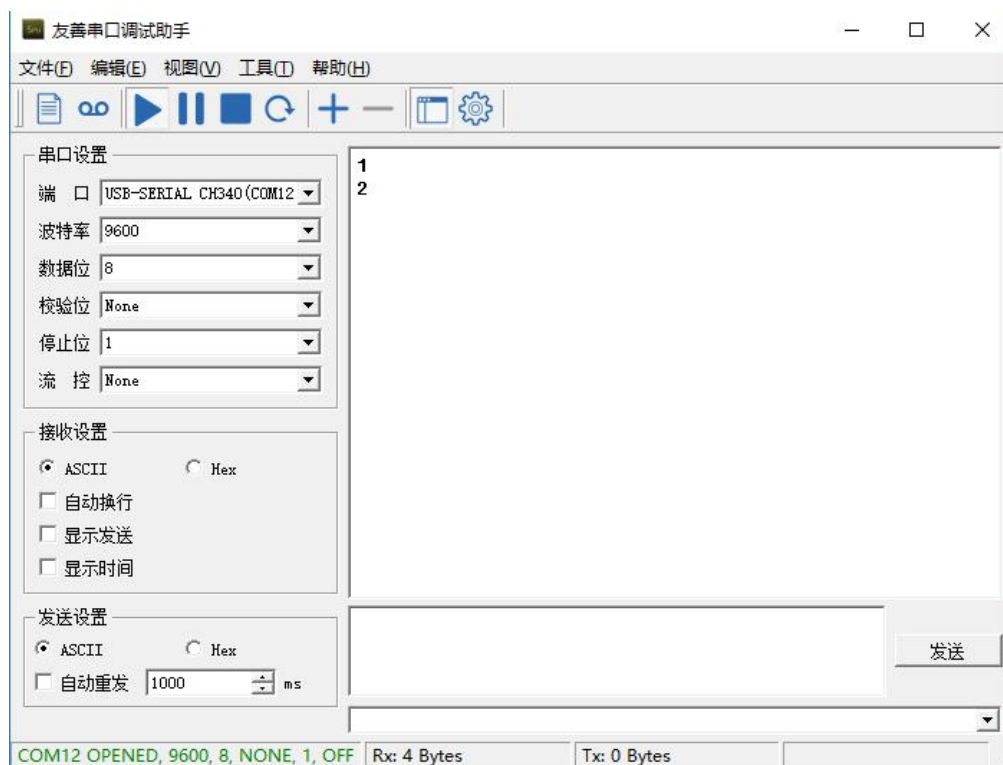


11) 点击“开始编程”按键，将程序烧写至单片机中，编写成功会提示“一切正常”。



3.实现效果

首先打开串口调试工具，选择我们控制板连接的端口，波特率选择 9600，接收设置选择 ASCII 即可。然后对准语音识别模块，任意说出“你好”或“开始”字样，识别成功，模块的 STA 指示灯会闪烁，并且串口调试助手会收到对应的数据。

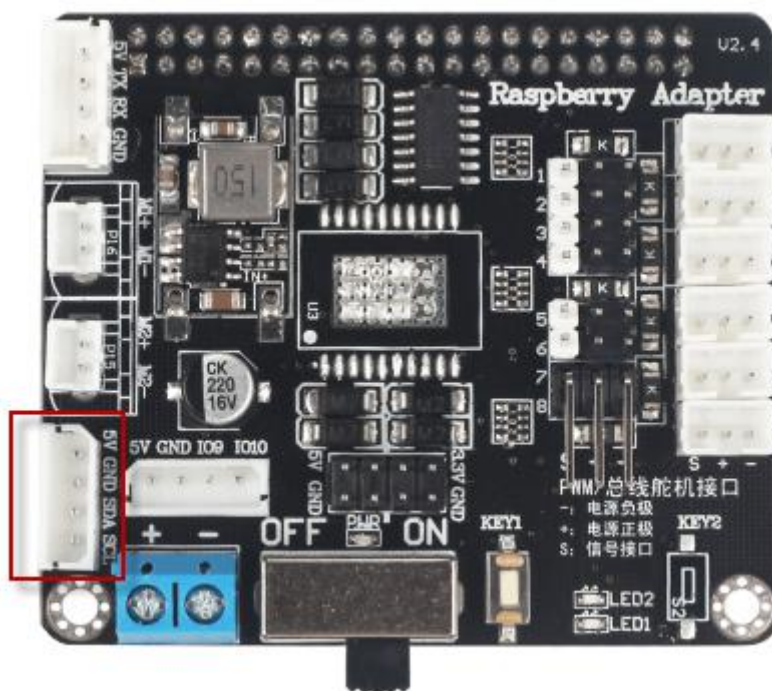


案例三：Raspberry 示例

1.准备工作

◆ 硬件部分

1) 本节示例使用的是树莓派 4B 和树莓派拓展板（树莓派 3B 及 3B+同样适用）。传感器需要通过 4P 线连接至树莓派拓展板的 IIC 接口。拓展板供电使用 7.5V 锂电池。

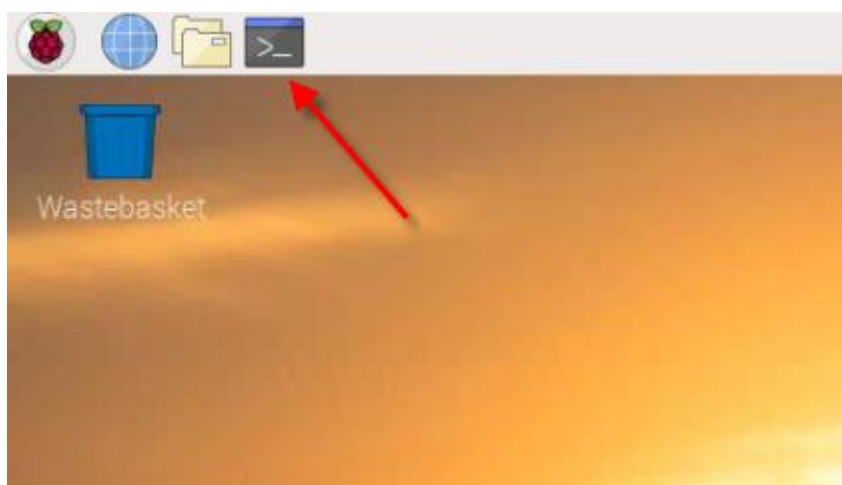


2) 为树莓派准备一个显示器、鼠标及键盘等外设，或者使用远程控制的方式（如果你是树莓派初学者，可以先学习本文件夹下的《树莓派开发入门手册》）。

◆ 软件部分:

注意：本案例使用的镜像为官方镜像 Raspbian。

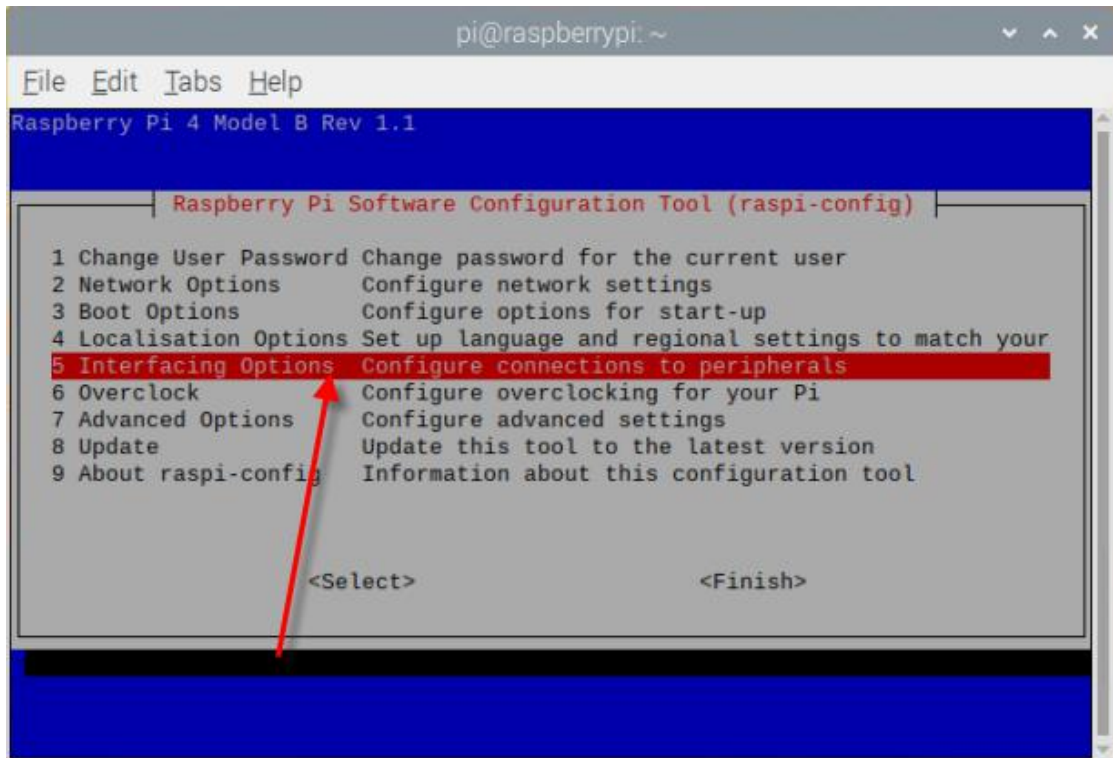
1) 开启树莓派，在系统桌面上打开 LX 终端。



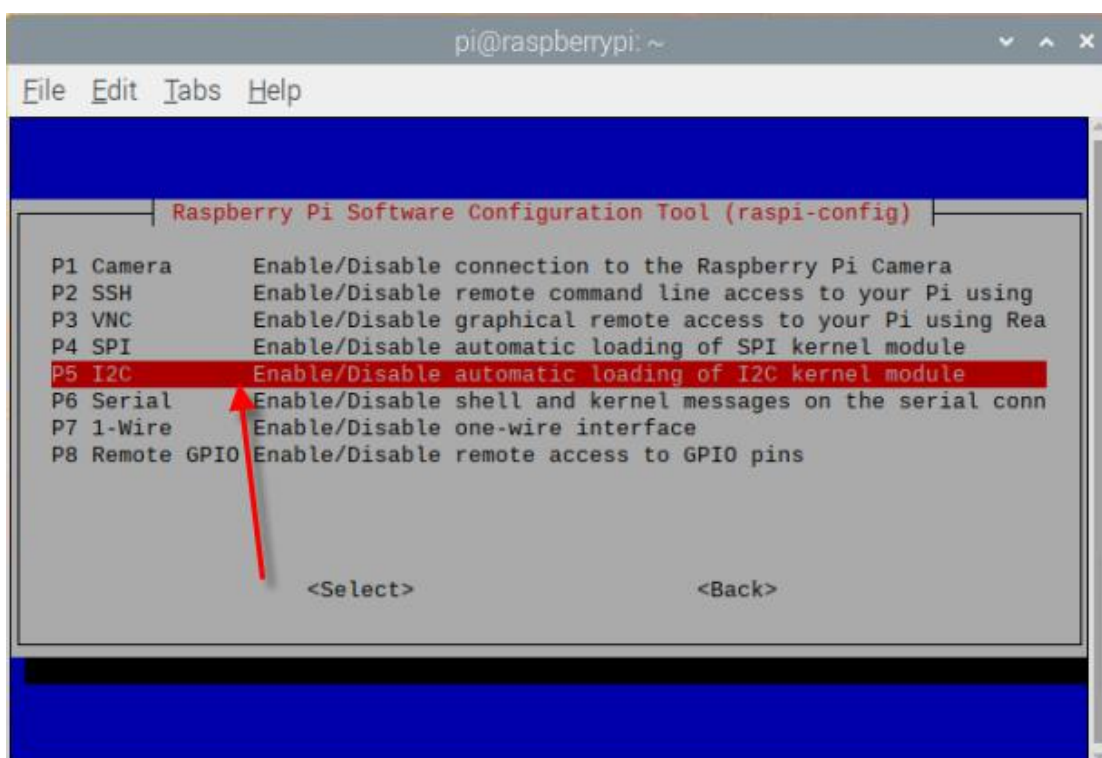
2) 输入下方图示指令，打开树莓派系统配置。


```
pi@raspberrypi:~ $ sudo raspi-config
```

3) 使用键盘“↑↓”键选择图示箭头位置，按回车前往该选项。



4) 选择“P5 I2C”功能会将该树莓派的 IIC 服务开启。



5) 使用“←→”按键选择“Yes”。

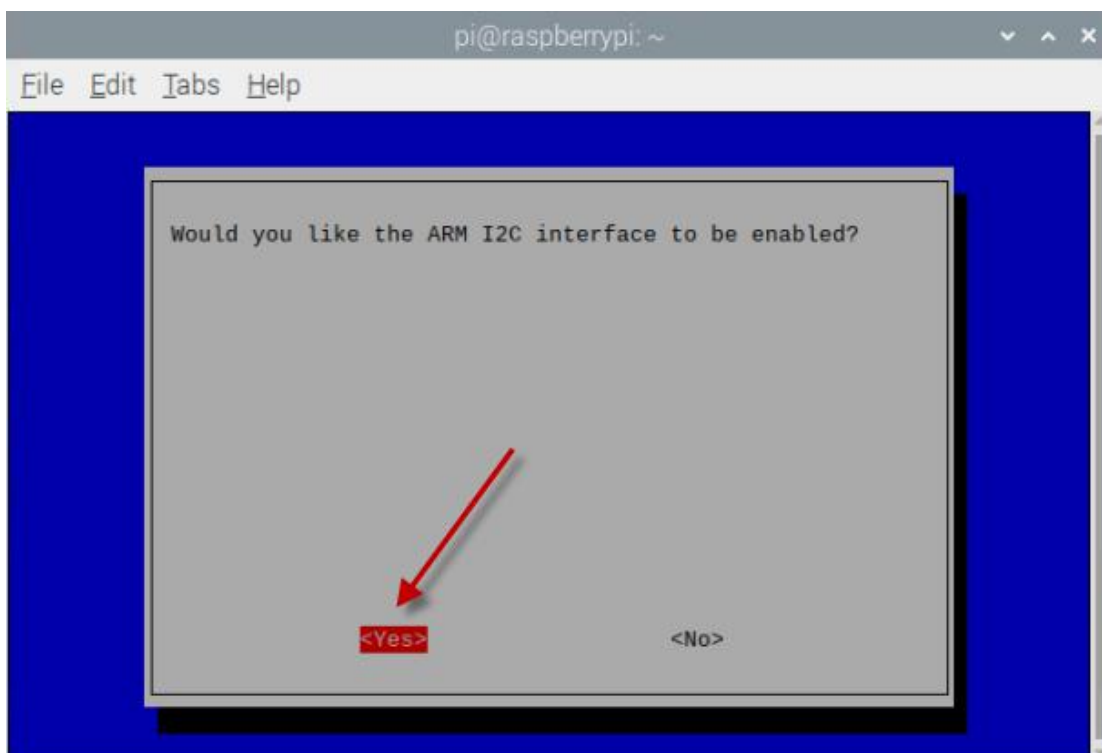


图 5-24 确认开启

6) 开启完成会再次回到一开始设置的蓝色主界面，按“Esc”键便可退出。

7) 输入“sudo reboot”指令，将树莓派重启（必须要做）。

```
pi@raspberrypi:~ $ sudo reboot
```

8) 重启完毕后，再次打开 LX 终端。依次输入下方图示指令安装 I2C 的库文件及 smbus 库（该过程需树莓派全程保持网络畅通）。

```
pi@raspberrypi:~ $ sudo apt-get install i2c-tools -y
```

```
pi@raspberrypi:~ $ sudo apt-get install python-smbus
```

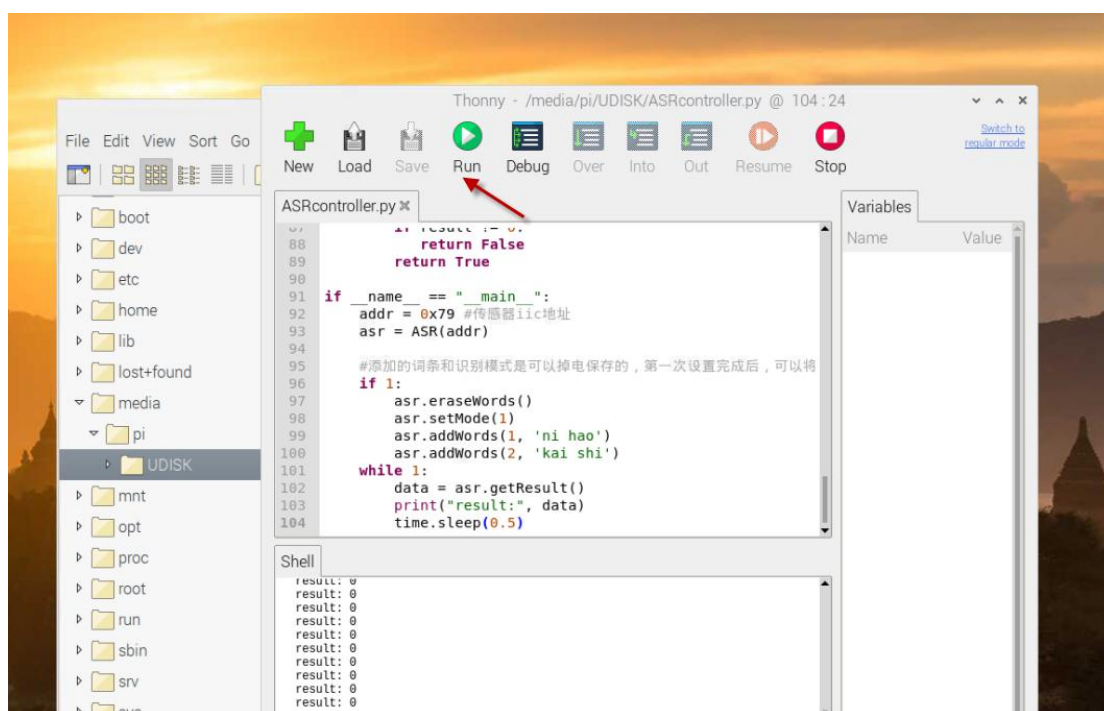
9) 通过下方指令可以查看连接至树莓派拓展板上的语音识别模块。

```
pi@raspberrypi:~ $ sudo i2cdetect -y -a 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: 00  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  79  --  --  --  --  --  --
```

2.实验流程

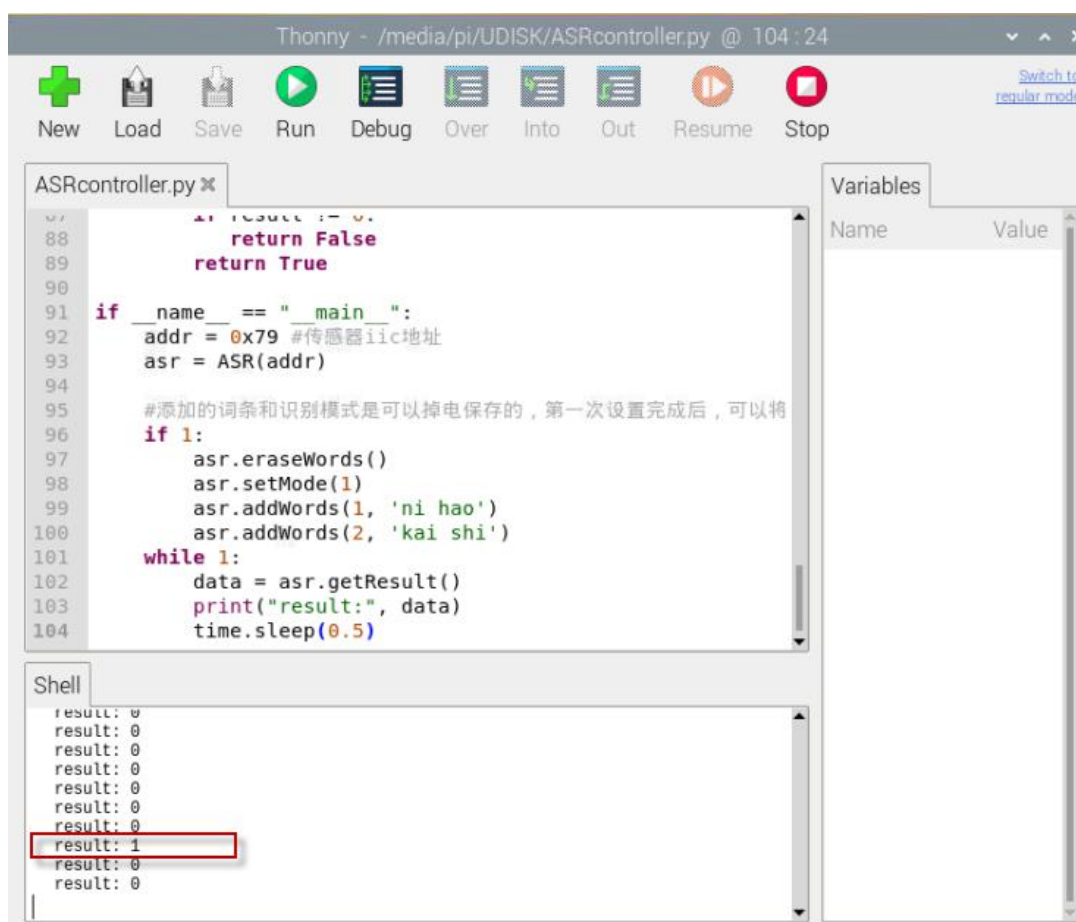
1) 将资料内的语音识别程序通过 U 盘或者远程传输至树莓派内（这里以 U 盘为例）。

2) 直接打开该程序，执行程序。



3.实现效果

对准语音识别模块说出“你好”或“开始”的字样，当语音模块识别到后，模块的 STA 指示灯会闪烁并且屏幕 Shell 界面会输出 1。



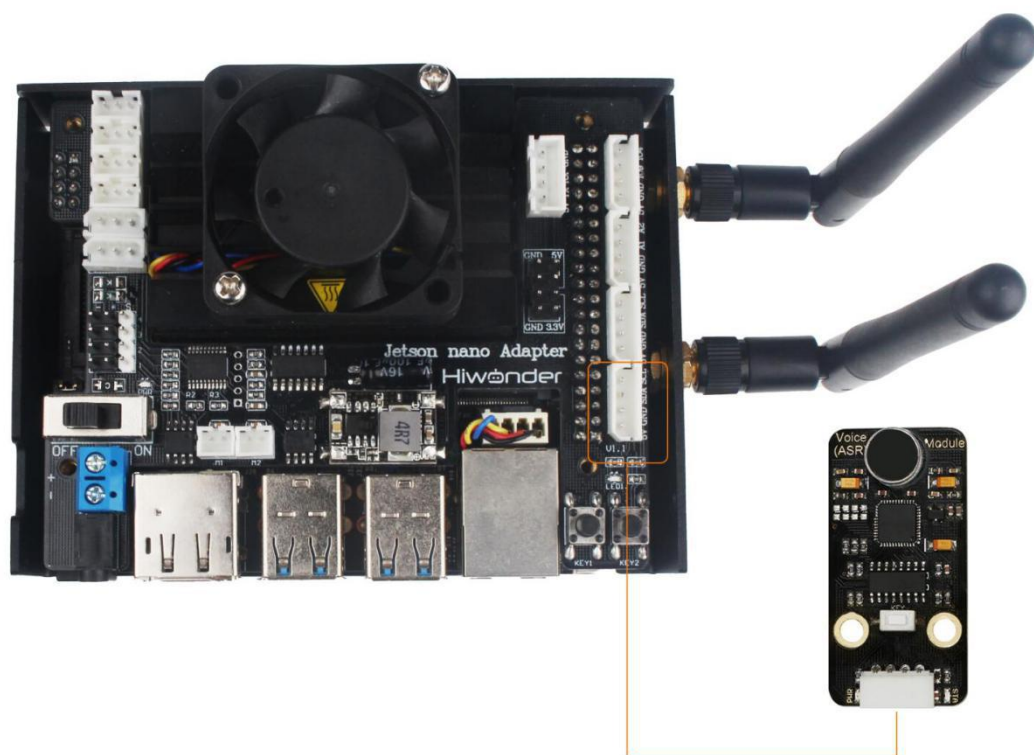
案例四：JETSON NANO 示例

1.准备工作

◆ 硬件部分：

1) 本节示例使用的是 jetson nano 主板和 jetson nano 扩展板，通过 5v4A 适配器来进行供电。

2) 语音识别传感器通过 4P 线连接至 jetson nano 扩展板的 IIC 接口。Jetson nano 的镜像为官方镜像。



3) 为 Jetson nano 准备一个显示器、鼠标及键盘等外设，或者使用远程控制的方式（如果你是 Jetson nano 初学者，可以先学习本文件夹下的《Jetson nano 开发入门手册》）。

◆ 软件部分：

由于传感器需要使用 I2C 接口，因此在使用 I2C 接口进行通讯时，需要准备以下步骤：

1) 输入“sudo apt-get update”命令来访问源列表里的网址，并读取软件列表。

```
hiwonder@hiwonder-desktop:~$ sudo apt-get update
获取:1 file:/var/cuda-repo-10-0-local-10.0.326 InRelease
忽略:1 file:/var/cuda-repo-10-0-local-10.0.326 InRelease
获取:2 file:/var/visionworks-repo InRelease
忽略:2 file:/var/visionworks-repo InRelease
```

2) 安装 I²C tool 用于设备地址扫描。在终端输入指令“sudo apt-get install libi2c-dev i2c-tools”进行安装。输入密码后会弹出确认安装信息，此时输入“Y”按下回车即可。

（安装过程需 Jetson nano 全程保持网络畅通）


```
hiwonder@hiwonder-desktop: ~  
hiwonder@hiwonder-desktop:~$ sudo apt-get install libi2c-dev i2c-tools  
[sudo] password for hiwonder:
```

```
hiwonder@hiwonder-desktop: ~  
libqt5quickwidgets5 libqt5sensors5 libqt5sql5 libqt5test5 libqt5webchannel5  
libqt5webkit5 libxcb-composite0 libxcb-cursor0 libxcb-damage0 os-prober  
python3-dbus.mainloop.pyqt5 python3-icu python3-pam python3-pyqt5  
python3-pyqt5.qtsvg python3-pyqt5.qtwebkit python3-sip  
qml-module-org-kde-kquickcontrolsaddons qml-module-qtmultimedia  
qml-module-qtquick2 rdate tasksel tasksel-data  
使用'sudo apt autoremove'来卸载它(它们)。  
下列【新】软件包将被安装:  
libi2c-dev  
升级了 0 个软件包, 新安装了 1 个软件包, 要卸载 0 个软件包, 有 203 个软件包未被升  
级。  
需要下载 7,104 B 的归档。  
解压缩后会消耗 30.7 kB 的额外空间。  
您希望继续执行吗? [Y/n] Y  
获取:1 http://ports.ubuntu.com/ubuntu-ports bionic/universe arm64 libi2c-dev arm  
64 4.0-2 [7,104 B]  
已下载 7,104 B, 耗时 5秒 (1,356 B/s)  
debconf: delaying package configuration, since apt-utils is not installed  
正在选中未选择的软件包 libi2c-dev。  
(正在读取数据库 ... 系统当前共安装有 134992 个文件和目录。)  
正准备解包 .../libi2c-dev_4.0-2_arm64.deb ...  
正在解包 libi2c-dev (4.0-2) ...  
正在设置 libi2c-dev (4.0-2) ...  
hiwonder@hiwonder-desktop:~$
```

3) 安装 python2 和 python3 的 I2C 库,分别输入“sudo apt-get install python-smbus”指令和“sudo apt-get install python3-smbus”指令即可,这里以安装 python3 的 I2C 库为例:

```

hiwonder@hiwonder-desktop: ~
正在设置 libi2c-dev (4.0-2) ...
hiwonder@hiwonder-desktop:~$ sudo apt-get install python3-smbus
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
下列软件包是自动安装的并且现在不需要了:
apt-clone archdetect-deb bogl-bterm busybox-static cryptsetup-bin
dpkg-repack gir1.2-timzone-1.0 gir1.2-xkl-1.0 grub-common
kde-window-manager kinit kio kpackagetool5 kwayland-data kwinn-common
kwinn-data kwinn-x11 libdebiann-installer4 libkdecorations2-5v5
libkdecorations2-private5v5 libkf5activities5 libkf5attica5
libkf5completion-data libkf5completion5 libkf5declarative-data
libkf5declarative5 libkf5doctools5 libkf5globalaccel-data libkf5globalaccel5
libkf5globalaccel-private5 libkf5idle-time5 libkf5jobwidgets-data
libkf5jobwidgets5 libkf5kcmutils-data libkf5kcmutils5 libkf5kio-core5
libkf5kiontln5 libkf5kio-widgets5 libkf5newstuff-data libkf5newstuff5
libkf5newstuff-core5 libkf5package-data libkf5package5 libkf5plasma5
libkf5quickaddons5 libkf5solid5 libkf5solid5-data libkf5sonnet5-data
libkf5sonnet-core5 libkf5sonnet-ui5 libkf5textwidgets-data libkf5textwidgets5
libkf5wayland-client5 libkf5wayland-server5 libkf5xmlgui-bin libkf5xmlgui-data
libkf5xmlgui5 libkscreenlocker5 libkwinn4-effect-builtins1 libkwineffects11
libkwinn-glutils11 libkwinn-xrenderutils11 libqgsttools-p1 libqt5designer5
libqt5help5 libqt5multimedia5 libqt5multimedia5-plugins
libqt5multimedia-quick-p5 libqt5multimedia-quick5 libqt5opengl5

```

4) 使用指令“`apt-cache policy i2c-tools`”来检查安装情况，出现如下图所示即为安装成功。

```

hiwonder@hiwonder-desktop: ~
下列【新】软件包将被安装：
python3-smbus
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 203 个软件包未被升级。
需要下载 6,832 B 的归档。
解压缩后会消耗 32.8 kB 的额外空间。
获取:1 http://ports.ubuntu.com/ubuntu-ports bionic/universe arm64 python3-smbus
arm64 4.0-2 [6,832 B]
已下载 6,832 B，耗时 0秒 (14.1 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
正在选中未选择的软件包 python3-smbus:arm64。
(正在读取数据库 ... 系统当前共安装有 135000 个文件和目录。)
正准备解包 .../python3-smbus_4.0-2_arm64.deb ...
正在解包 python3-smbus:arm64 (4.0-2) ...
正在设置 python3-smbus:arm64 (4.0-2) ...
hiwonder@hiwonder-desktop:~$ apt-cache policy i2c-tools
i2c-tools:
  已安装: 4.0-2
  候选: 4.0-2
  版本列表:
*** 4.0-2 500
500 http://ports.ubuntu.com/ubuntu-ports bionic/universe arm64 Packages
100 /var/lib/dpkg/status
hiwonder@hiwonder-desktop:~$

```

5) 通过下方指令可以查看连接至 Jetson nano 上的语音识别模块，这里依然需要输入密码。


```
hiwonder@hiwonder-desktop:~$ sudo i2cdetect -y -r -a 1
[sudo] password for hiwonder:
```

6) 由于语音识别模块的 IIC 的地址默认为 0x79, 因此设备的 I2C 地址将会出现在下方列表。

```
hiwonder@hiwonder-desktop:~$ sudo i2cdetect -y -r -a 1
[sudo] password for hiwonder:
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: 00  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  79  --  --  --  --  --  --
```

2.实验流程

1) 将资料内的语音识别程序通过 U 盘或者远程传输至 Jetson nano 内 (这里以 U 盘为例)。



2) 定位到程序文件存放的目录, 然后输入 “sudo python3 ASRcontroller.py” 运行程序。

```
hiwonder@hiwonder-desktop:~/Desktop$ sudo python3 ASRcontroller.py
[sudo] password for hiwonder:
```

3.实现效果

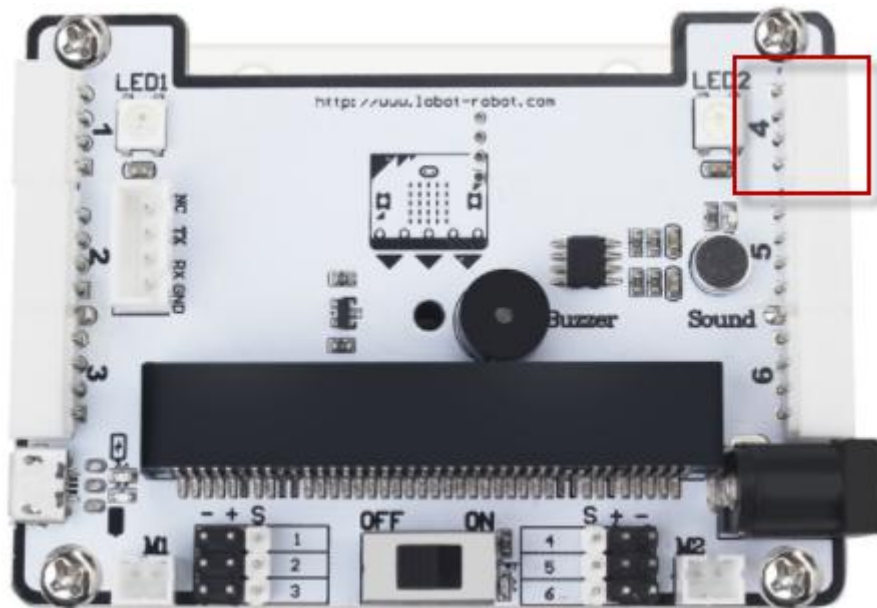
3) 程序运行时，对着模块说“你好”，则模块上的蓝灯会闪烁，并且终端打印出“你好”。对着模块说“开始”，同样的块上的蓝灯会闪烁，并且打印出“开始”，其他情况便打印 None。

```
hiwonder@hiwonder-desktop:~/Desktop$ sudo python3 ASRcontroller.py
[sudo] password for hiwonder:
result: None
result: None
result: None
result: None
result: None
result: None
result: None
result: None
result: None
result: None
result: None
result: 你好
result: None
result: None
result: None
result: None
result: None
result: 开始
result: None
result: None
```

案例五：micro:bit 示例

1.准备工作

本节示例使用的是 micro:bit 主板和 micro:bit 扩展板。需要将语音识别传感器通过 4P 线连接至 micro:bit 扩展板的 4 号接口。

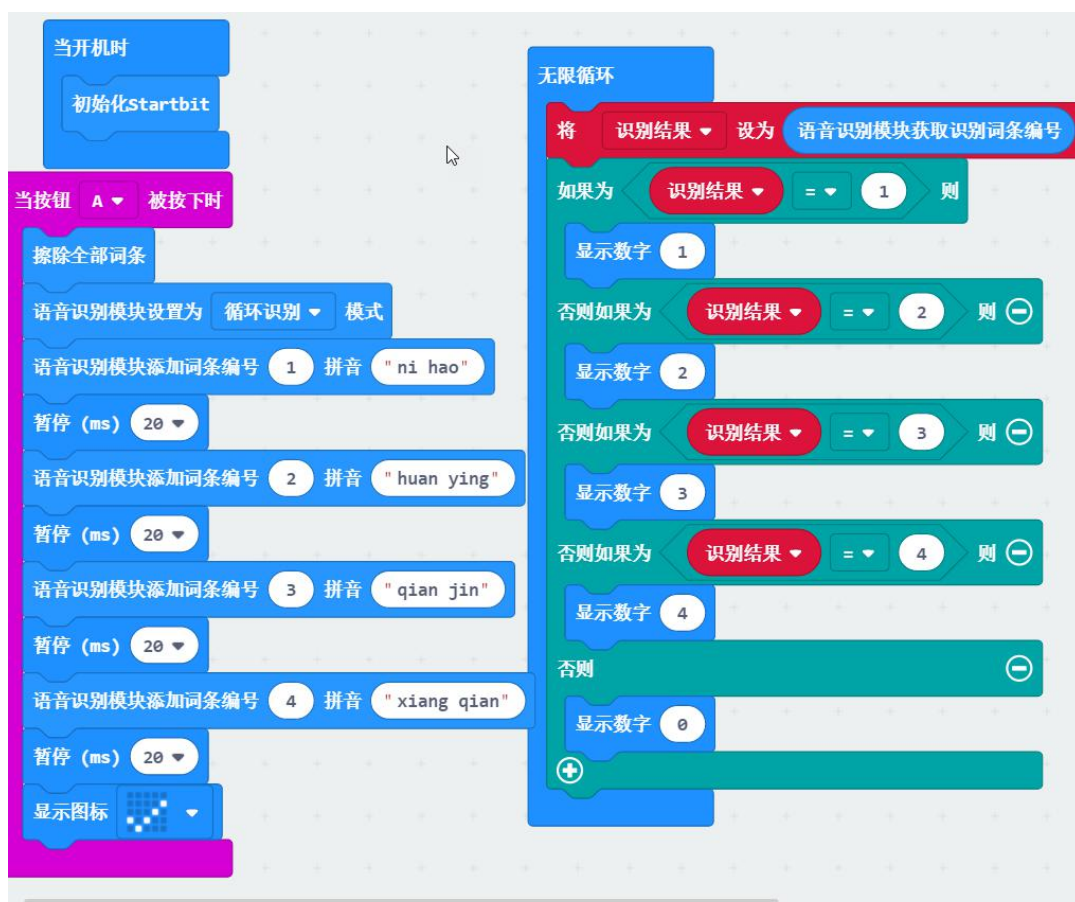


2.实验流程

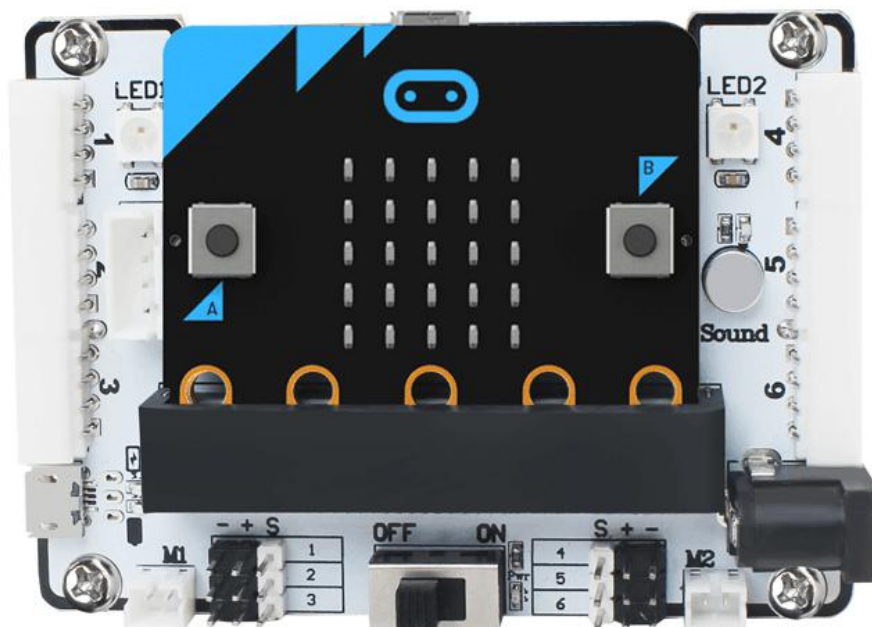
- 1) 点击此[链接](#)，进入 micro:bit 官网在线编程平台。
- 2) 点击网页此按键（如下方图示）或将本示例的 micro:bit 程序直接拖动至网页页面内。



- 3) 将 micro:bit 主板插入电脑任意一个 USB 接口，把程序下载至该 micro:bit 所在的盘符内。



4) 下载完成后，将 micro:bit 主板插入 micro:bit 扩展板内即可。



3.实现效果

1) 重新打开扩展板开关，按下 micro:bit 主板 A 键，对准语音识别模块任选一词（“你好、欢迎、前进、向前”）。

2) 当识别时，模块的 STA 指示灯会闪烁，micro:bit 主板的点阵会显示“√”的图案。识别成功，主板的点阵将会显示（1-3）的数字。

附件篇

规格参数

尺寸	48mm×24mm
工作电压	5V
通信方式	IIC 通信，使用 4PIN 连接线连接主控器
安装方式	兼容乐高系列
三种识别模式	循环识别模式、口令模式、按键模式
识别要求	每次识别最多可以设置 50 项候选识别句，识别句可以是单字，词组或短句，长度为不超过 10 个汉字或者 79 个字节的拼音串。
板载芯片状态指示灯	循环识别模式下，状态灯常亮。口令模式下，以第一个词条为口令，当识别到口令词时常亮，等待识别到新的语音，并且读取识别结果后灭掉。按键模式

	下，按下按键开始识别且状态灯亮起，不按时不识别且状态灯不亮，支持掉电保存。
识别范围：	安静环境下，语音接收最大距离为 3 米；嘈杂环境下语音接收最大距离为 30cm。