Arduino 人工智能套件

Demo 说明

文档版本：V2.1

发布日期：2019-12-05

深圳市越疆科技有限公司

免责申明

在法律允许的最大范围内，本手册所描述的产品（含其硬件、软件、固件等）均“按照现状”提供，可能存在瑕疵、错误或故障，越疆不提供任何形式的明示或默示保证，包括但不限于适销性、质量满意度、适合特定目的、不侵犯第三方权利等保证；亦不对使用本手册或使用本公司产品导致的任何特殊、附带、偶然或间接的损害进行赔偿。

在使用本产品前详细阅读本使用手册及网上发布的相关技术文档并了解相关信息，确保在充分了解机器人及其相关知识的前提下使用机械臂。越疆建议您在专业人员的指导下使用本手册。该手册所包含的所有安全方面的信息都不得视为Dobot的保证，即便遵循本手册及相关说明，使用过程中造成的危害或损失依然有可能发生。

本产品的使用者有责任确保遵循相关国家的切实可行的法律法规，确保在越疆机械臂的使用中不存在任何重大危险。

越疆科技有限公司

地址：深圳市南山区留仙大道3370号南山智园崇文区2号楼9-10楼

网址：http://cn.dobot.cc/

前 言

目的

本文档描述了Arduino人工智能套件Demo环境搭建、Demo各模块连接以及关键代码说明，供入门创客以及非电子专业的电子爱好者参考。

读者对象

本手册适用于：

- 客户工程师
- 技术支持工程师

修订记录

时间	修订记录
2019/12/05	第三次发布 修复bug，完善代码，增加初始化中文语音API接口
2019/07/05	第二次发布 修改语音模块连接图
2019/06/12	第一次发布

符号约定

在本手册中可能出现下列标志，它们所代表的含义如下。

符号	说明
⚠危险	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害
⚠警告	表示有中度或低度潜在危害，如果不能避免，可能导致人员轻微伤害、机械臂毁坏等情况
⚠注意	表示有潜在风险，如果忽视这些文本，可能导致机械臂损坏、数据丢失或不可预知的结果
📖说明	表示是正文的附加信息，是对正文的强调和补充

目 录

1. 简介

1.1 概述

Arduino人工智能套件包括Arduino Mega2560控制板、LED指示灯、开关按键、摇杆、LD3320语音识别模块以及Pixy视觉识别模块。Arduino人工智能套件Demo 基于Arduino Mega2560，由Dobot开发。本文档根据Demo对Arduino人工智能套件各模块连接、Demo实现的逻辑框图等进行详细说明，使入门创客以及非电子专业的电子爱好者快速入门，启发创新思维。

1.2 环境搭建

Arduino是一款便捷灵活、方便升级的开源电子平台，包括Arduino开发工具Arduino IDE和核心库。下载Arduino 1.8.2版本，其下载路径为https://www.arduino.cc/en/Main/OldSoftwareReleases#previous。下载时请选择操作系统对应的版本。

安装Arduino IDE后，还需对环境进行配置，操作步骤如下：

步骤 1 解压已获取的Arduino人工智能套件Demo，将SmartKit、Dobot、Pixy2文件夹添加至“Arduino IDE安装目录\arduino-1.8.2\libraries”目录。

如果添加成功，打开Arduino IDE后可在Arduino界面的“项目 > 加载库”查看对应的库，如图 1.1所示。

说明

因为Arduino人工智能套件Demo需用到SmartKit（LED指示灯、开关按键、摇杆、LD3320语音识别模块）、机械臂、Pixy视觉识别模块，所以调试Demo前需添加对应函数库并加载至Arduino。

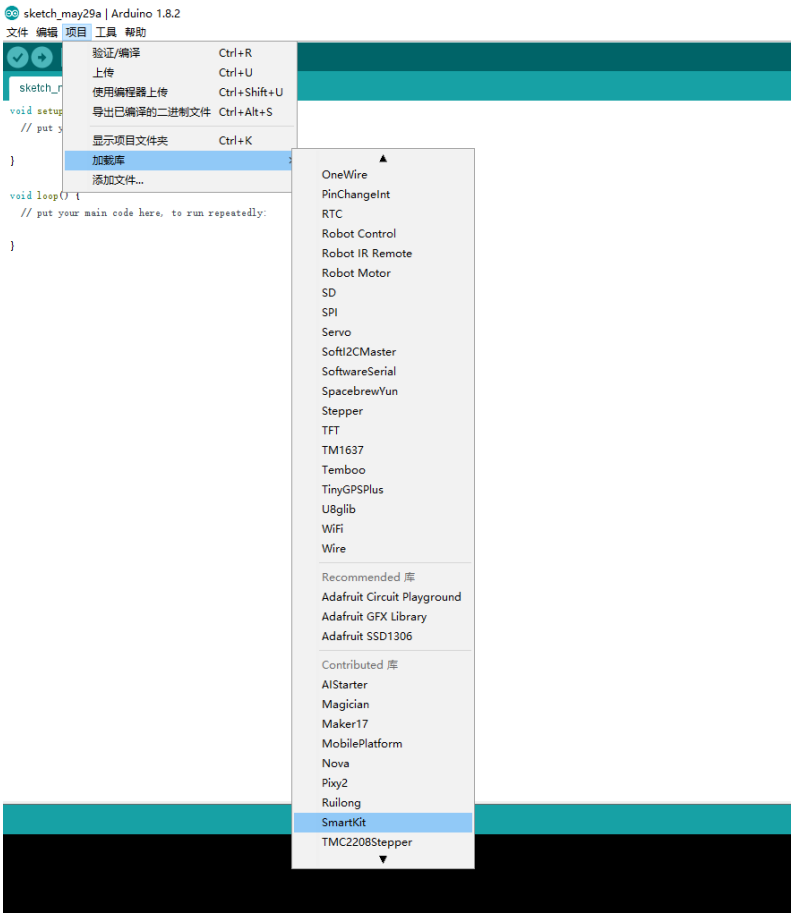


图 1.1 添加SmartKit库

步骤 2 打开Arduino IDE软件。

步骤 3 在Arduino界面的“工具 > 开发板”选择“Arduino/Genuino Mega or Mega 2560”，“工具 > 处理器”选择“ATmega2560（Mega 2560）”，在“工具 > 端口”选择相应的串口。

⚠注意

如果Arduino人工智能套件Demo用到了SmartKit、机械臂、视觉识别模块，打开对应Demo后，请在Arduino界面的“项目 > 加载库”选择对应的库将其导入工程。

2. FlickerLED Demo

2.1 介绍

该Demo通过Arduino控制LED指示灯点亮和熄灭。

2.2 硬件连接

该Demo所需模块：LED模块与Arduino Mega2560，其连接如图 2.1所示。

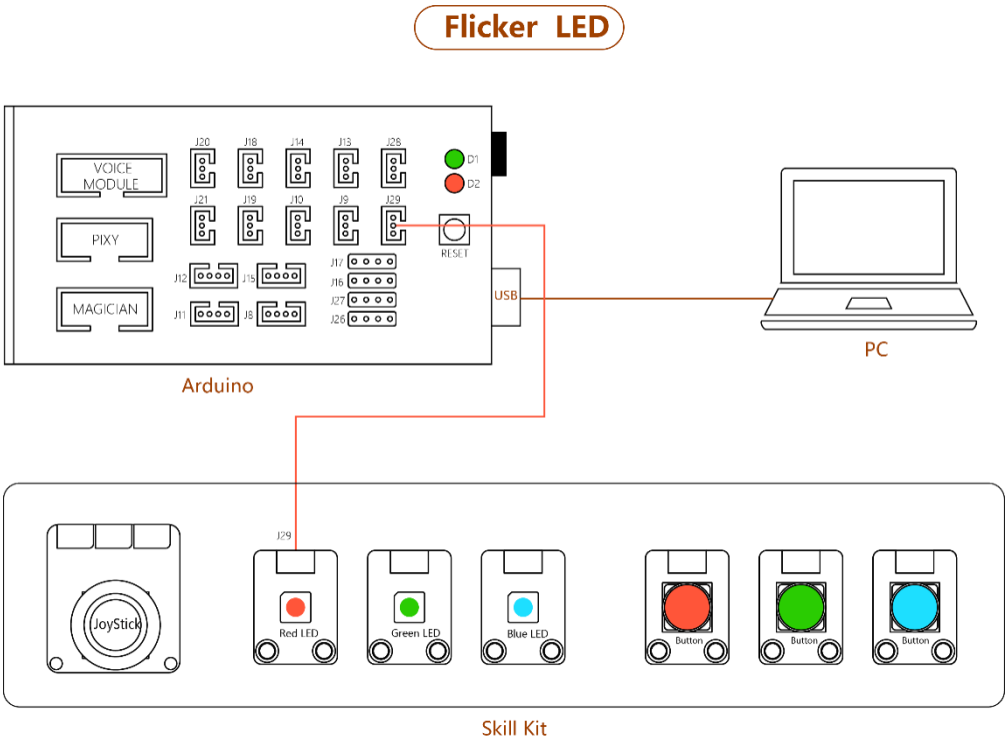


图 2.1 FlickerLED 连接示意图

📖说明

如果用户需将套件连接至Arduino Mega2560上其他接口，还需在“SmartKit.h”文件中修改套件连接的对应接口。

程序 2.1 定义连接至 Arduino 的引脚

```
#define JOYSTICK_XPIN     7      /* JoyStick X 轴连接的引脚 */
#define JOYSTICK_YPIN     6      /* JoyStick Y 轴连接的引脚 */
#define JOYSTICK_ZPIN     A5     /* JoyStick Z 轴连接的引脚 */

#define LED_REDPIN        9      /* 红色灯连接引脚 */
#define LED_GREENPIN      A1     /* 绿色灯连接引脚 */
#define LED_BLUEPIN       A3     /* 蓝色灯连接引脚 */

#define BUTTON_REDPIN     A0     /* 红色按钮连接引脚 */
```

```
#define BUTTON_GREENPIN A2     /* 绿色按钮连接引脚 */
#define BUTTON_BLUEPIN   A4     /* 蓝色按钮连接引脚 */
```

2.3 实现流程

Demo实现流程如图 2.2所示。

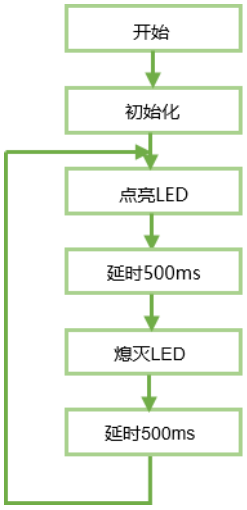


图 2.2 Demo 实现流程

2.4 关键代码说明

该Demo需要调用SmartKit库。调用该Demo前，请在Arduino界面的“项目 > 加载库”选择对应的库将其导入工程。

（1） 程序初始化。

程序 2.2 初始化

```
void setup(){
  SmartKit_Init();                    //初始化
}
```

（2） 设置引脚的高低电平以控制 LED 指示灯。

程序 2.3 设置高低电平

```
void loop(){
    SmartKit_LedTurn(RED, ON);          //点亮 LED 指示灯
    delay(500);
    SmartKit_LedTurn(RED, OFF);         //熄灭 LED 指示灯
    delay(500);
}
```


3. AlarmLED Demo

3.1 介绍

该Demo通过Arduino控制三个不同颜色的LED指示灯点亮和熄灭，每次仅一个LED指示灯点亮。

3.2 硬件连接

该Demo所需模块：三个LED指示灯与Arduino Mega2560，其连接如图 3.1所示。

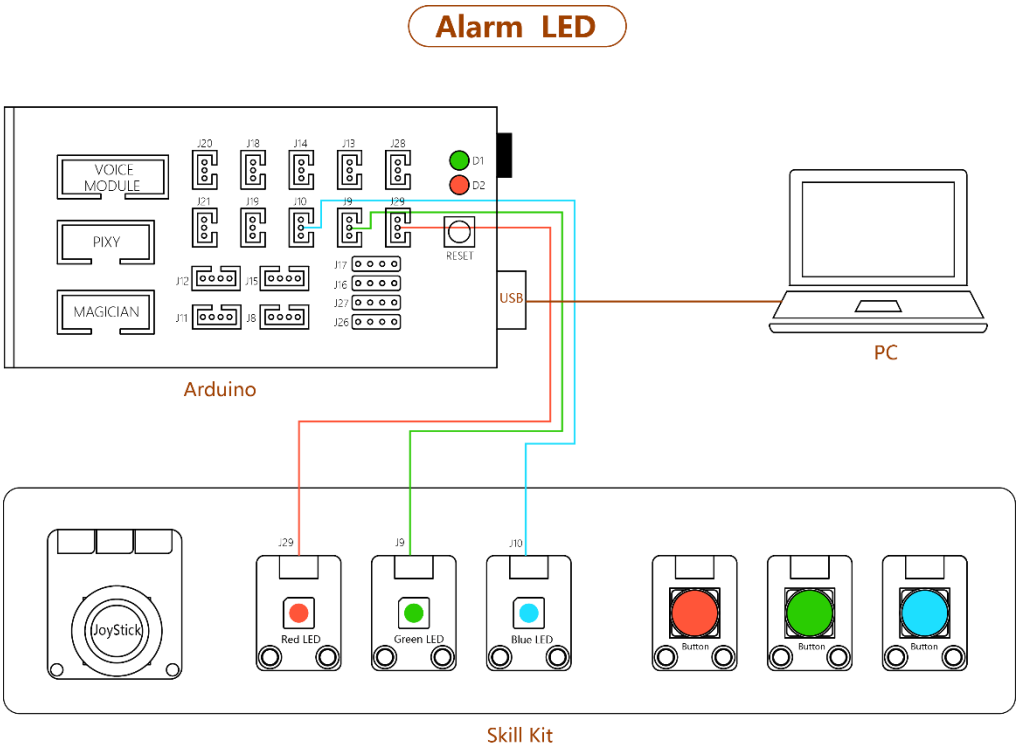


图 3.1 AlarmLED 连接示意图

📖说明

如果用户需将套件连接至Arduino Mega2560上其他接口，还需在“SmartKit.h”文件中修改套件连接的对应接口。

程序 3.1 定义连接至 Arduino 的引脚

```
#define JOYSTICK_XPIN     7      /* JoyStick X 轴连接的引脚 */
#define JOYSTICK_YPIN     6      /* JoyStick Y 轴连接的引脚 */
#define JOYSTICK_ZPIN     A5     /* JoyStick Z 轴连接的引脚 */

#define LED_REDPIN        9      /* 红色灯连接引脚 */
#define LED_GREENPIN      A1     /* 绿色灯连接引脚 */
#define LED_BLUEPIN       A3     /* 蓝色灯连接引脚 */

#define BUTTON_REDPIN     A0     /* 红色按钮连接引脚 */
```

```
#define BUTTON_GREENPIN A2     /* 绿色按钮连接引脚 */
#define BUTTON_BLUEPIN   A4     /* 蓝色按钮连接引脚 */
```

3.3 实现流程

Demo实现流程如图 3.2所示。

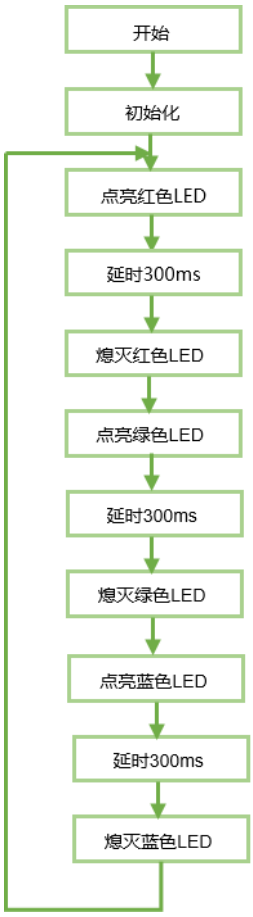


图 3.2 Demo 实现流程

3.4 关键代码说明

该Demo需要调用SmartKit库。调用该Demo前，请在Arduino界面的“项目 > 加载库”选择对应的库将其导入工程。

（1） 程序初始化，关闭所有 LED 指示灯。

程序 3.2 初始化

```
void setup() {
    SmartKit_Init();
    SmartKit_LedTurn(RED, OFF);                    //关闭 LED 指示灯
    SmartKit_LedTurn(BLUE, OFF);
    SmartKit_LedTurn(GREEN, OFF);
}
```

（2） 设置高低电平以控制 LED 指示灯。

程序 3.3 设置高低电平

```
void loop() {
    SmartKit_LedTurn(RED, ON);
    delay(300);
    SmartKit_LedTurn(RED, OFF);
    SmartKit_LedTurn(GREEN, ON);
    delay(300);
    SmartKit_LedTurn(GREEN, OFF);
    SmartKit_LedTurn(BLUE, ON);
    delay(300);
    SmartKit_LedTurn(BLUE, OFF);
}
```

4. AdjustLED Demo

4.1 介绍

该Demo通过摇杆控制LED指示灯的亮度。

4.2 硬件连接

该Demo所需模块：LED模块与Arduino Mega2560，其连接如图 4.1所示。

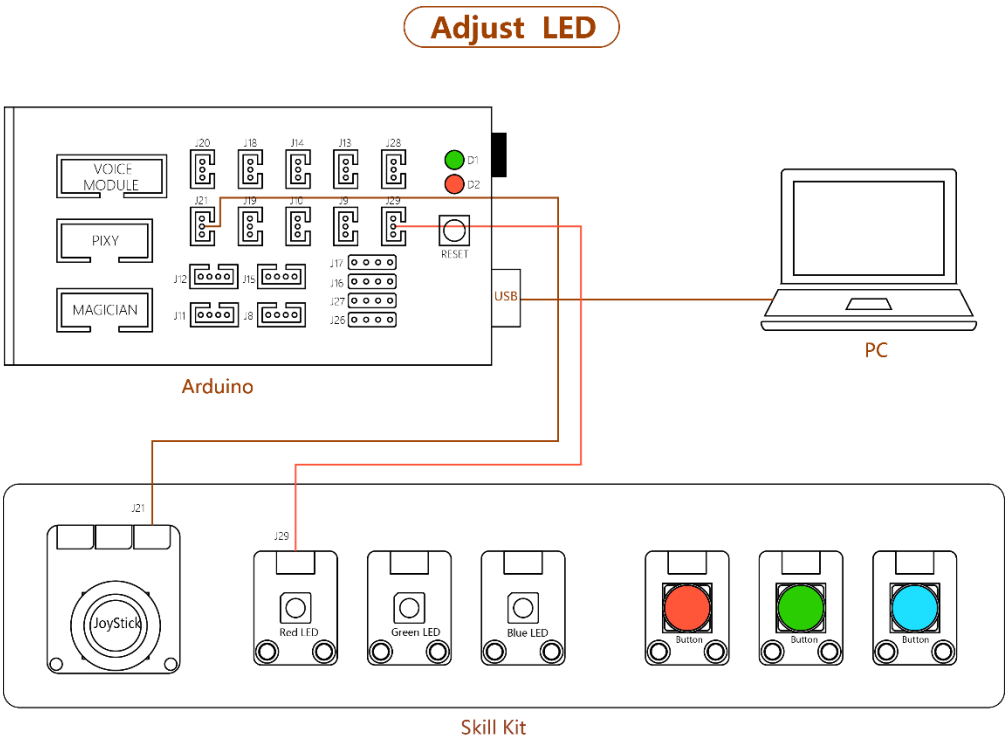


图 4.1 AdjustLED 连接示意图

📖说明

如果用户需将套件连接至Arduino Mega2560上其他接口，还需在“SmartKit.h”文件中修改套件连接的对应接口。

程序 4.1 定义连接至 Arduino 的引脚

```
#define JOYSTICK_XPIN     7      /* JoyStick X 轴连接的引脚 */
#define JOYSTICK_YPIN     6      /* JoyStick Y 轴连接的引脚 */
#define JOYSTICK_ZPIN     A5     /* JoyStick Z 轴连接的引脚 */

#define LED_REDPIN        9      /* 红色灯连接引脚 */
#define LED_GREENPIN      A1     /* 绿色灯连接引脚 */
#define LED_BLUEPIN       A3     /* 蓝色灯连接引脚 */

#define BUTTON_REDPIN     A0     /* 红色按钮连接引脚 */
```

```
#define BUTTON_GREENPIN A2     /* 绿色按钮连接引脚 */
#define BUTTON_BLUEPIN   A4     /* 蓝色按钮连接引脚 */
```

4.3 实现流程

该Demo通过摇杆X轴控制LED指示灯的亮度，Demo实现流程如图 4.2所示。

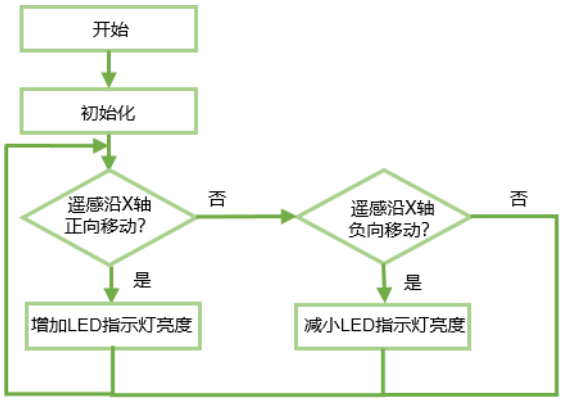


图 4.2 Demo 实现流程

4.4 关键代码说明

该Demo需要调用SmartKit库。调用该Demo前，请在Arduino界面的“项目 > 加载库”选择对应的库将其导入工程。

（1） 程序初始化。

程序 4.2 初始化

```
void setup(){
  SmartKit_Init();                    //初始化
}
```

（2） 定义 LED 指示灯亮度变化频率。

说明

移动摇杆X轴或Y轴时，其模拟量输出范围为0~1023，如图 4.3所示。当摇杆静止时，X轴模拟量输出为512，Y轴模拟量输出为508。

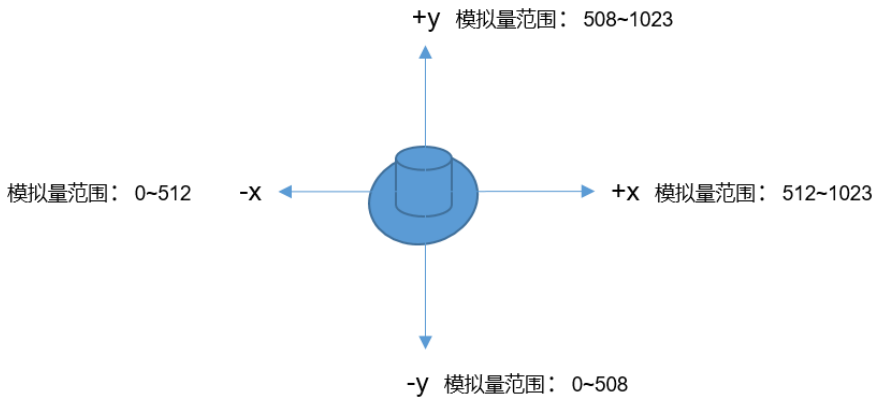


图 4.3 模拟量输出范围

程序 4.3 定义 LED 指示灯亮度变化频率

```
double xValueTOAnalogScale = 1023/255;                    //计算摇杆的值转换成 I/O 输出模拟值的比例
value = SmartKit_JoyStickReadXYValue(AXISX);              //获取摇杆的 x 轴模拟量
```

（3） 通过摇杆调节 LED 指示灯亮度。

程序 4.4 通过摇杆调节 LED 指示灯亮度

```
value = value / xValueTOAnalogScale;
analogWrite(LED_REDPIN, value);                           //调节 LED 指示灯亮度
```


5. Button Demo

5.1 介绍

该Demo通过三个不同颜色的按键控制对应颜色LED指示灯的点亮与熄灭。

5.2 硬件连接

该Demo所需模块：三个按键模块、三个LED指示灯与Arduino Mega2560。其连接如图5.1所示。

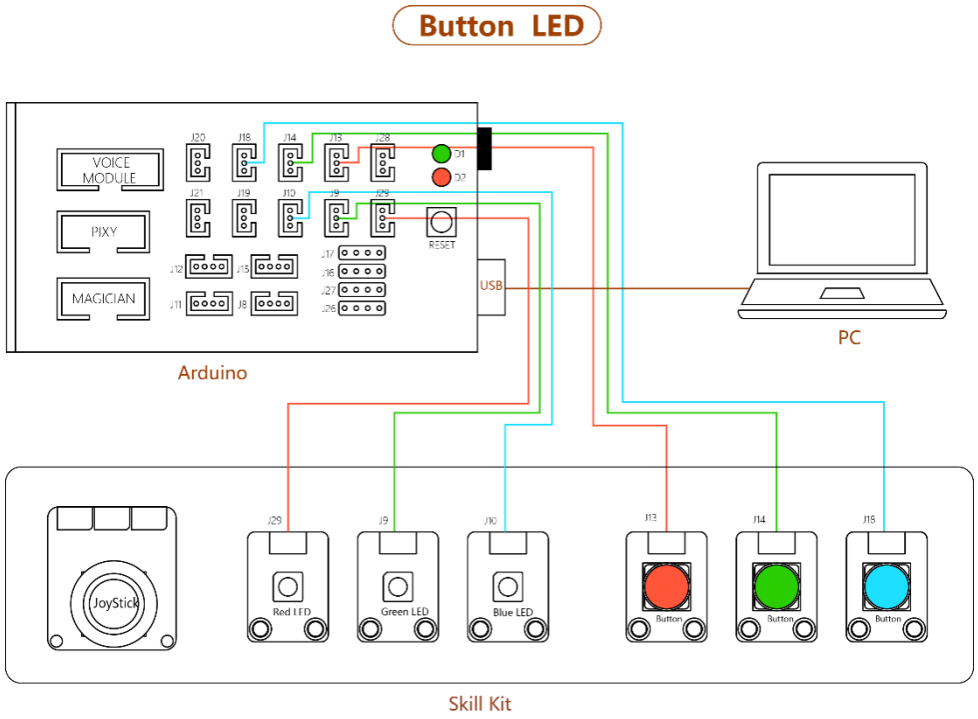


图 5.1 ButtonLED 连接示意图

📖说明

如果用户需将套件连接至Arduino Mega2560上其他接口，还需在“SmartKit.h”文件中修改套件连接的对应接口。

程序 5.1 定义连接至 Arduino 的引脚

```
#define JOYSTICK_XPIN     7      /* JoyStick X 轴连接的引脚 */
#define JOYSTICK_YPIN     6      /* JoyStick Y 轴连接的引脚 */
#define JOYSTICK_ZPIN     A5     /* JoyStick Z 轴连接的引脚 */

#define LED_REDPIN        9      /* 红色灯连接引脚 */
#define LED_GREENPIN      A1     /* 绿色灯连接引脚 */
#define LED_BLUEPIN       A3     /* 蓝色灯连接引脚 */

#define BUTTON_REDPIN     A0     /* 红色按钮连接引脚 */
```

```
#define BUTTON_GREENPIN A2     /* 绿色按钮连接引脚 */
#define BUTTON_BLUEPIN   A4     /* 蓝色按钮连接引脚 */
```

5.3 实现流程

实现流程如图 5.2所示。

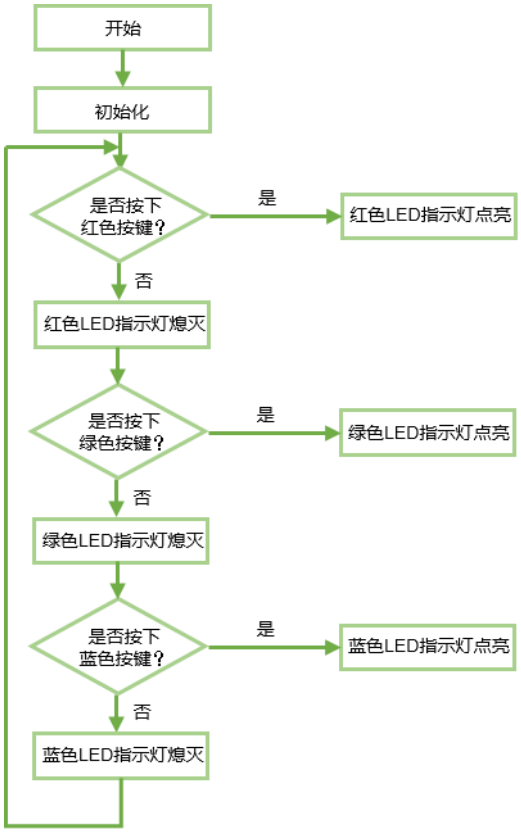


图 5.2 Demo 实现流程

5.4 关键代码说明

该Demo需要调用SmartKit库。调用该Demo前，请在Arduino界面的“项目 > 加载库”选择对应的库将其导入工程。

（1） 程序初始化。

程序 5.2 程序初始化

```
void setup(){
  SmartKit_Init();                    //初始化
}
```

（2） 通过按键控制 LED 灯点亮与熄灭。

程序 5.3 按键控制 LED

```
    if (SmartKit_ButtonCheckState(color) == TRUE)        //检查按键状态
    {
        SmartKit_LedTurn(color, ON);                     //点亮 LED 指示灯
    }
    else
    {
        SmartKit_LedTurn(color, OFF);                    //熄灭 LED 指示灯
    }
……
…..
```

6. VoiceLED Demo

6.1 介绍

该Demo通过LD3320语音识别模块控制不同颜色的（红、绿、蓝）LED指示灯点亮和熄灭。

6.2 硬件连接

该 Demo 所需模块：LD3320 语音识别模块、三个 LED 指示灯与 Arduino Mega2560，其连接如图 6.1所示。

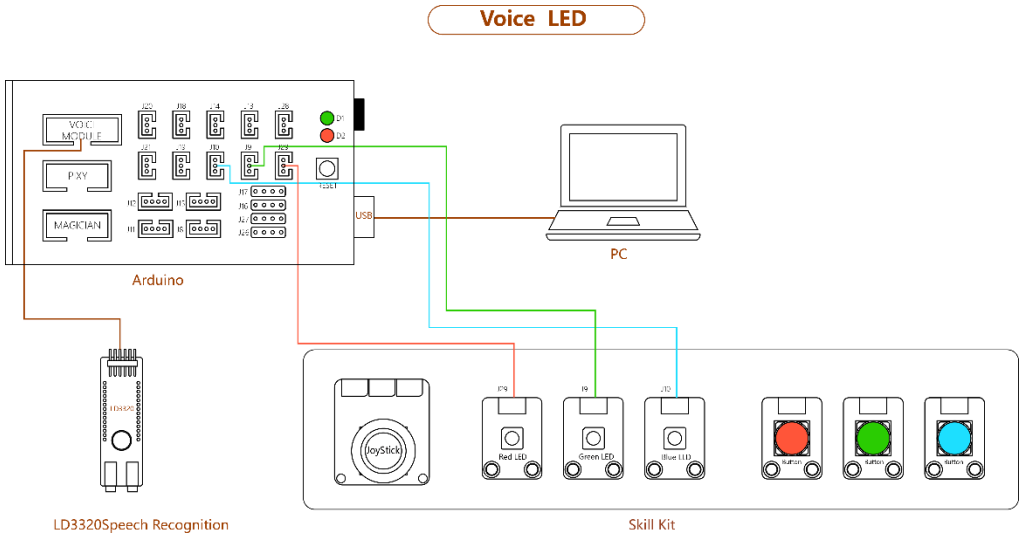


图 6.1 VoiceLED 连接示意图

⚠注意

LD3320语音模块未做防反插设计，所以连接至Arduino时请务必按图 6.2所示进行连接。

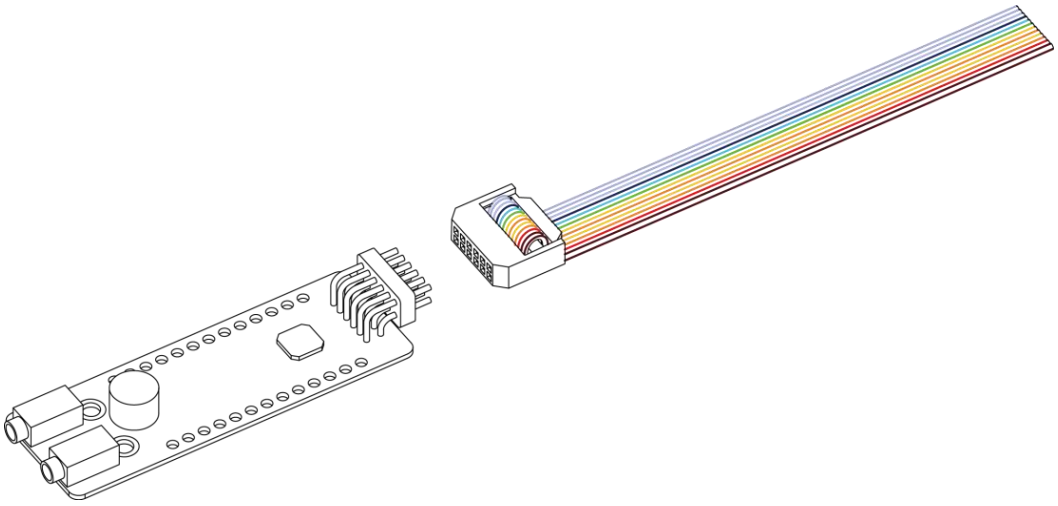


图 6.2 语音模块连接

📖说明

如果用户需将套件连接至Arduino Mega2560上其他接口，还需在“SmartKit.h”文件中修改套件连接的对应接口。

程序 6.1 定义连接至 Arduino 的引脚

```
#define JOYSTICK_XPIN     7      /* JoyStick X 轴连接的引脚 */
#define JOYSTICK_YPIN     6      /* JoyStick Y 轴连接的引脚 */
#define JOYSTICK_ZPIN     A5     /* JoyStick Z 轴连接的引脚 */

#define LED_REDPIN        9      /* 红色灯连接引脚 */
#define LED_GREENPIN      A1     /* 绿色灯连接引脚 */
#define LED_BLUEPIN       A3     /* 蓝色灯连接引脚 */

#define BUTTON_REDPIN     A0     /* 红色按钮连接引脚 */
#define BUTTON_GREENPIN A2     /* 绿色按钮连接引脚 */
#define BUTTON_BLUEPIN    A4     /* 蓝色按钮连接引脚 */
```

6.3 实现流程

实现流程如图 6.3所示。

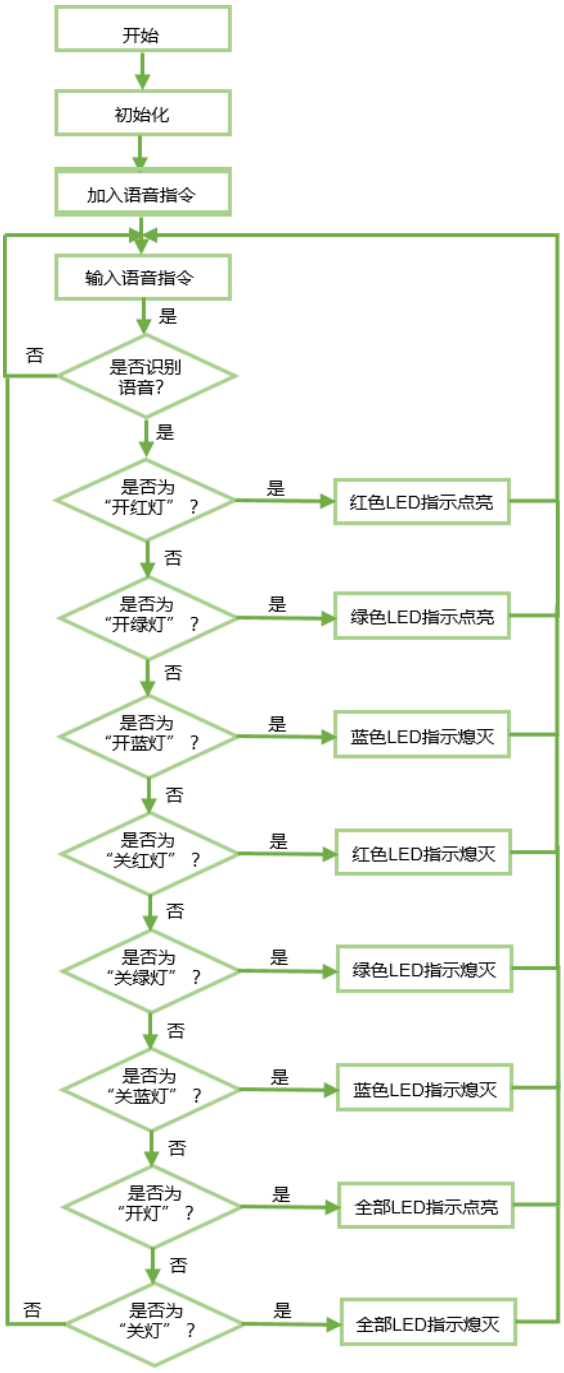


图 6.3 Demo 实现说明

6.4 关键代码说明

该Demo需要调用SmartKit库。调用该Demo前，请在Arduino界面的“项目 > 加载库”选择对应的库将其导入工程。

（1） 程序初始化并增加语音指令。

📖说明

LD3320语音识别模块支持用户自由编辑50条关键词语条。

程序 6.2 增加语音指令

```
void setup() {
    Serial.begin(115200);
    SmartKit_VoiceCNInit();
    SmartKit_Init();
    SmartKit_VoiceCNAddCommand("kai hong deng", 0);
    SmartKit_VoiceCNAddCommand("kai lv deng", 1);
    SmartKit_VoiceCNAddCommand("kai lan deng", 2);
    SmartKit_VoiceCNAddCommand("guan hong deng", 3);
    SmartKit_VoiceCNAddCommand("guan lv deng", 4);
    SmartKit_VoiceCNAddCommand("guan lan deng", 5);
    SmartKit_VoiceCNAddCommand("kai deng", 6);
    SmartKit_VoiceCNAddCommand("guan deng", 7);
    SmartKit_VoiceCNStart();
}
```

（2） 根据语音指令控制 LED 指示灯。

程序 6.3 根据语音指令控制 LED 指示灯

```
void loop() {
    if (SmartKit_VoiceCNVoiceCheck(0) == TRUE)          //检查是否检测到语音序列 0
    {
        SmartKit_LedTurn(RED, ON);                      //点亮 LED 指示灯
        Serial.println("turn on the red LED");
    }
……
……
}
```

7. MoveBlock Demo

7.1 介绍

该Demo通过Arduino控制机械臂完成物块的搬运。

7.2 硬件连接

该Demo所需模块：Arduino Mega2560、Dobot Magician与吸盘套件，Dobot Magician与Arduino连接如图 7.1所示。

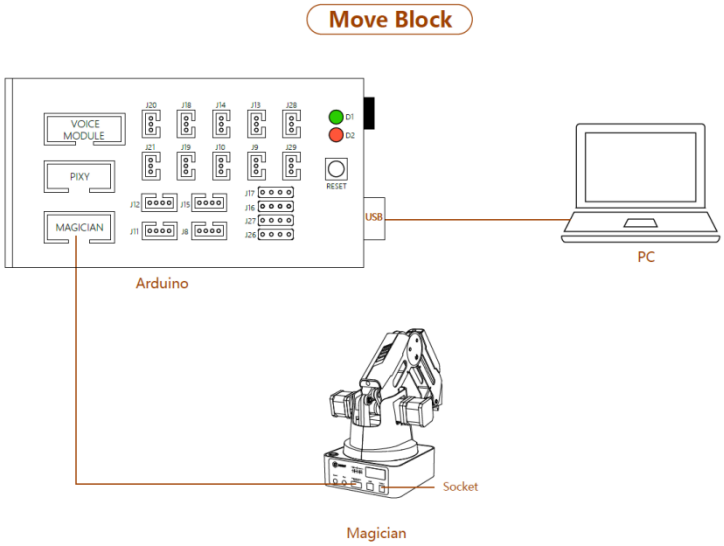


图 7.1 MoveBlock 连接示意图

吸盘套件与Dobot Magician连接请参见附录B 安装吸盘套件。

说明

如果用户需将套件连接至Arduino Mega2560上其他接口，还需在“SmartKit.h”文件中修改套件连接的对应接口。

程序 7.1 定义连接至 Arduino 的引脚

```
#define JOYSTICK_XPIN     7      /* JoyStick X 轴连接的引脚 */
#define JOYSTICK_YPIN     6      /* JoyStick Y 轴连接的引脚 */
#define JOYSTICK_ZPIN     A5     /* JoyStick Z 轴连接的引脚 */

#define LED_REDPIN        9      /* 红色灯连接引脚 */
#define LED_GREENPIN      A1     /* 绿色灯连接引脚 */
#define LED_BLUEPIN       A3     /* 蓝色灯连接引脚 */

#define BUTTON_REDPIN     A0     /* 红色按钮连接引脚 */
#define BUTTON_GREENPIN A2       /* 绿色按钮连接引脚 */
```

```
#define BUTTON_BLUEPIN    A4     /*  蓝色按钮连接引脚  */
```

7.3 实现流程

假设物块从A点搬运到B点，再从B点搬回A点，并循环多次，其实现流程如图 7.2所示。

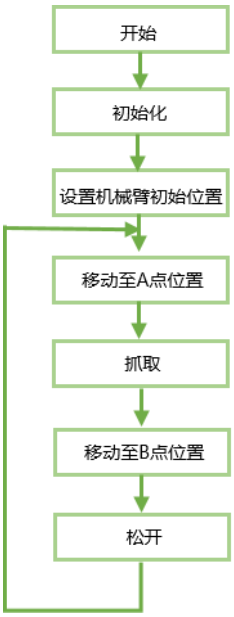


图 7.2 Demo 实现流程

7.4 关键代码说明

Dobot Magician通过底座背面的UART接口（10PIN）与Arduino进行通信，采用Dobot通信协议。我们已提供Dobot库，该库封装了部分Dobot Magician API，可直接调用即可控制Dobot Magician。调试该Demo前，请在Arduino界面的“项目 > 加载库”选择对应的库将其导入工程。

⚠注意

调试Demo前请长按Dobot Magician底座背面的“Key”键进行回零操作。

调试Demo前，需将Dobot Magician和DobotStudio连接，将Dobot Magician回零后，按住小臂上的圆形按钮并拖动小臂移动至物块待放置的位置（假设A点和B点），然后在DobotStudio的“操作面板”界面记录物块位置坐标，以便写入Demo中完成物块搬运，如图7.3所示。

图 7.3 获取位置坐标

（1） 定义 A 点和 B 点坐标。

坐标可通过 DobotStudio 界面的“操作面板”获取。

```
// A 点坐标
#define block_positio_X 263                    //A 点 X 轴坐标
#define block_position_Y 3                     //A 点 Y 轴坐标
#define block_position_Z -40                   //A 点 Z 轴坐标
#define block_position_R 0                     //A 点 R 轴坐标

//B 点坐标
#define Des_position_X 207
```

```
#define Des_position_Y -171
#define Des_position_Z -46
#define Des_position_R 0
```

（2） 机械臂从 A 点移动至 B 点，并循环多次。

程序 7.2 机械臂从 A 点移动至 B 点，并循环多次

```
while(count > 0)
{
  Dobot_SetPTPCmdEx(JUMP_XYZ,block_position_X, block_position_Y,
                    block_position_Z, block_position_R);          //移动至 A 点
  Dobot_SetEndEffectorSuctionCupEx(true);                         //打开气泵
  Dobot_SetPTPCmdEx((JUMP_XYZ,block_position_X, block_position_Y,-4, block_position_R);
                                                                  //抬升一定高度
  Dobot_SetPTPCmdEx(JUMP_XYZ, Des_position_X, Des_position_Y,
                    Des_position_Z, Des_position_R);              //移动至 B 点
  Dobot_SetEndEffectorSuctionCupEx(false);                        //关闭气泵
  Dobot_SetPTPCmdEx(JUMP_XYZ, Des_position_X, Des_position_Y, -20, Des_position_R);
                                                                  //Lift a certain height
  Dobot_SetPTPCmdEx(MOVJ_XYZ, 178, -4, 40, 0);                    //回到初始位置
  Dobot_SetPTPCmdEx(JUMP_XYZ, Des_position_X, Des_position_Y,
                    Des_position_Z, Des_position_R);              //移动至 B 点
  Dobot_SetEndEffectorSuctionCupEx(true);                         //打开气泵
  Dobot_SetPTPCmdEx(MOVL_XYZ, Des_position_X, Des_position_Y, -10, Des_position_R);
                                                                  //抬升一定高度
  Dobot_SetPTPCmdEx(JUMP_XYZ,block_position_X, block_position_Y,
                    block_position_Z, block_position_R);          //移动至 A 点
  Dobot_SetEndEffectorSuctionCupEx(false);                        //关闭气泵
  Dobot_SetPTPCmdEx(MOVJ_XYZ, 178, -4, 40, 0);                    //回到初始位置
  count--;
}
```

8. VoiceDobot Demo

8.1 介绍

该Demo通过LD3320语音识别模块控制机械臂移动以及气泵的启停。

8.2 硬件连接

该Demo所需模块：LD3320语音识别模块、Dobot Magician、气泵盒与Arduino Mega2560。LD3320语音识别模块、Dobot Magician与Arduino连接如图 8.1所示。

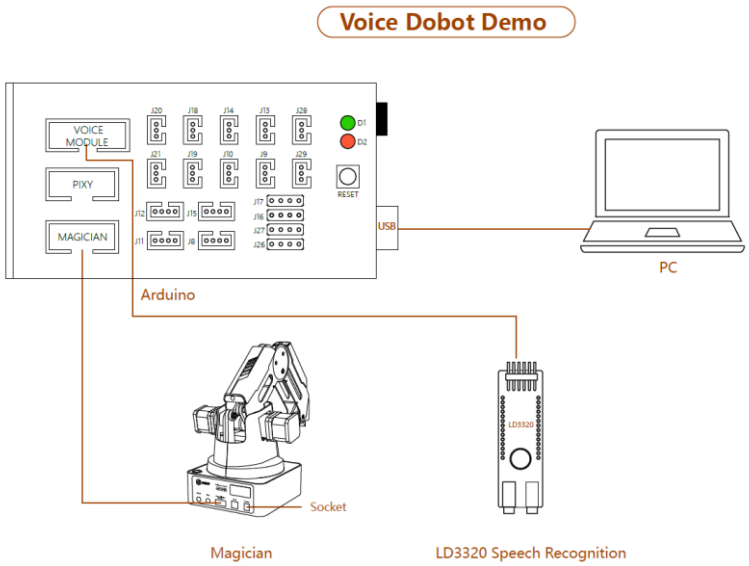


图 8.1 VoiceDobot 连接示意图

⚠注意

LD3320语音模块未做防反插设计，所以连接至Arduino时请务必按图 8.2所示进行连接。

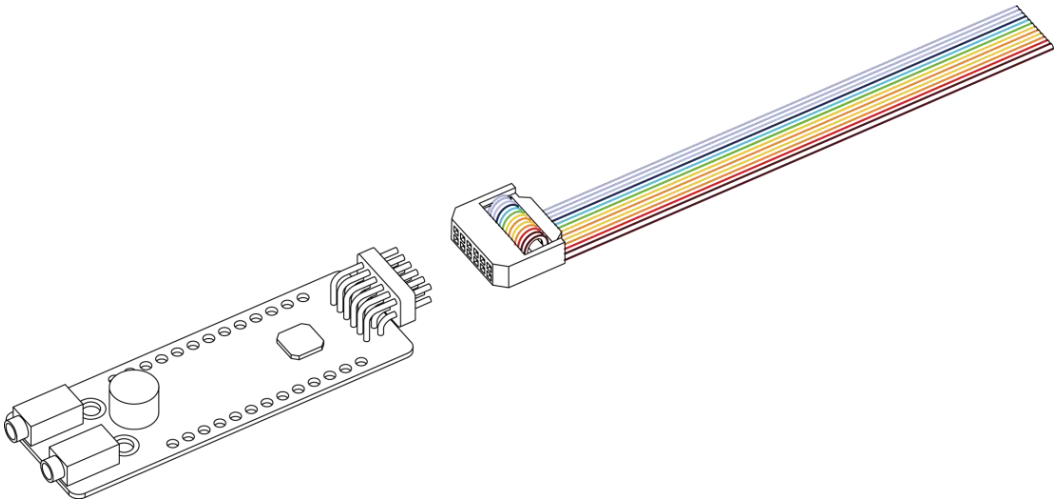


图 8.2 语音模块连接

Dobot Magician与气泵连接如图 8.3所示。

图 8.3 气泵连接图

📖说明

如果用户需将套件连接至Arduino Mega2560上其他接口，还需在“SmartKit.h”文件中修改套件连接的对应接口。

程序 8.1 定义连接至 Arduino 的引脚

```
#define JOYSTICK_XPIN     7      /* JoyStick X 轴连接的引脚 */
#define JOYSTICK_YPIN     6      /* JoyStick Y 轴连接的引脚 */
#define JOYSTICK_ZPIN     A5     /* JoyStick Z 轴连接的引脚 */

#define LED_REDPIN        9      /* 红色灯连接引脚 */
#define LED_GREENPIN      A1     /* 绿色灯连接引脚 */
#define LED_BLUEPIN       A3     /* 蓝色灯连接引脚 */

#define BUTTON_REDPIN     A0     /* 红色按钮连接引脚 */
#define BUTTON_GREENPIN A2     /* 绿色按钮连接引脚 */
#define BUTTON_BLUEPIN    A4     /* 蓝色按钮连接引脚 */
```

8.3 实现流程

该Demo通过LD3320语音识别模块控制机械臂上、下、左、右、前、后移动，并控制气泵的启停，其实现流程如图 8.4所示。

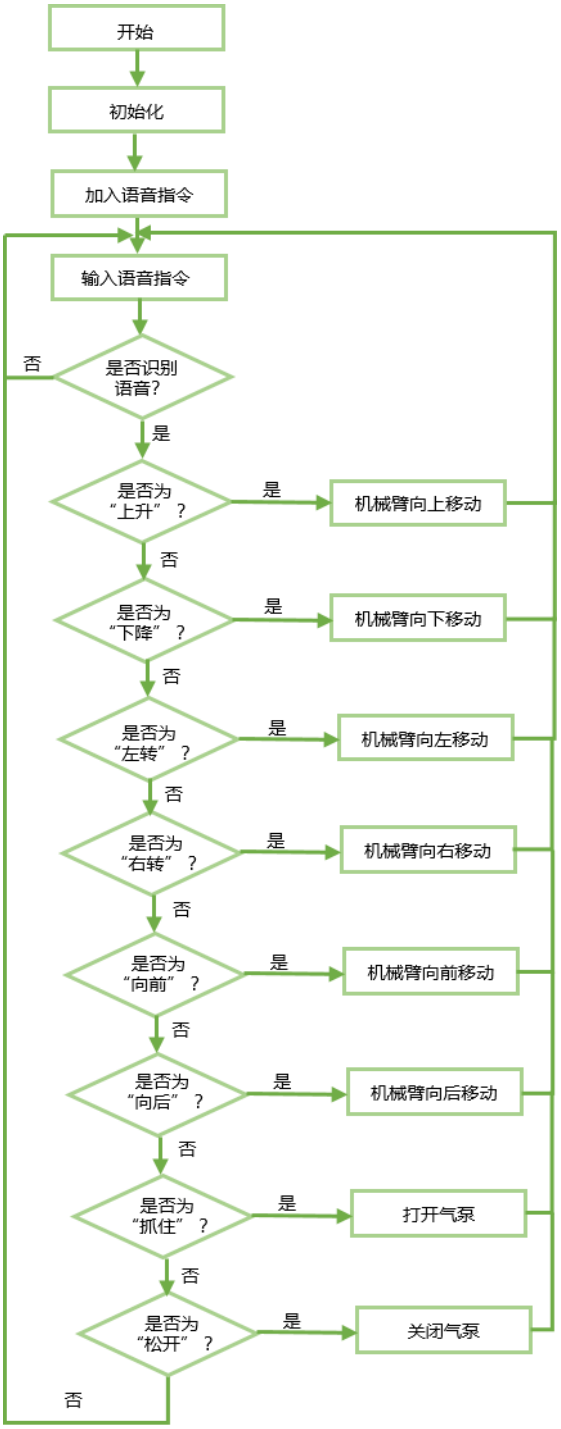


图 8.4 Demo 实现流程

8.4 关键代码说明

该Demo通过LD3320语音识别模块控制机械臂，需调用SmartKit和Dobot Magician库。调试该Demo前，请在Arduino界面的“项目 > 加载库”选择对应的库将其导入工程。

⚠注意

调试Demo前请长按Dobot Magician底座背面的“Key”键进行回零操作。

（1） 程序初始化并增加语音指令。

📖说明

用户可自行增加语音指令，比如增加指令标签相同，指令内容为“上”、“上升”、“升”的多条语音指令，以便提高识别准确率，方便控制机械臂。LD3320语音识别模块支持用户自由编辑50条关键词语条。

程序 8.2 声明语音识别对象并初始化

```
void setup() {
    Serial.begin(115200);

Dobot_Init();
SmartKit_VoiceCNInit();
SmartKit_Init();
SmartKit_VoiceCNAddCommand("shang sheng",0);                    //增加语音指令
    SmartKit_VoiceCNAddCommand("shang",0);
    SmartKit_VoiceCNAddCommand("sheng",0);
    SmartKit_VoiceCNAddCommand("xia jiang",1);
    SmartKit_VoiceCNAddCommand("xia",1);
    SmartKit_VoiceCNAddCommand("jiang",1);
    SmartKit_VoiceCNAddCommand("zuo yi",2);
    SmartKit_VoiceCNAddCommand("zuo",2);
    SmartKit_VoiceCNAddCommand("you zhuan",3);
    SmartKit_VoiceCNAddCommand("you",3);
    SmartKit_VoiceCNAddCommand("qian jin",4);
    SmartKit_VoiceCNAddCommand("qian",4);
    SmartKit_VoiceCNAddCommand("hou tui",5);
    SmartKit_VoiceCNAddCommand("hou",5);
    SmartKit_VoiceCNAddCommand("da kai qi beng",6);
    SmartKit_VoiceCNAddCommand("guan bi qi beng",7);
    /*************************************************
     *The instructions to be identified here are added*
     *************************************************/
     SmartKit_VoiceCNStart();}
}
```

（2） 根据语音指令移动机械臂。

程序 8.3 根据语音指令移动机械臂

```
if (SmartKit_VoiceCNVoiceCheck(0) == TRUE)
    {
        Dobot_SetPTPCmd(MOVL_INC, 0, 0, 30, 0);                    //机械臂向上移动 30mm
        Serial.println("up");
    }
    else if (SmartKit_VoiceCNVoiceCheck(1) == TRUE)
    {
        Dobot_SetPTPCmd(MOVL_INC, 0, 0, -30, 0);                   //机械臂向下移动 30mm
        Serial.println("down");
    }
……
……
```

9. JoyStick Demo

9.1 介绍

该demo通过摇杆和三个按键模块控制机械臂和气泵。

9.2 硬件连接

该demo所需模块：摇杆模块、按键模块、Dobot Magician、气泵与Arduino Mega2560。摇杆模块、Dobot Magician与Arduino连接如图 9.1所示。

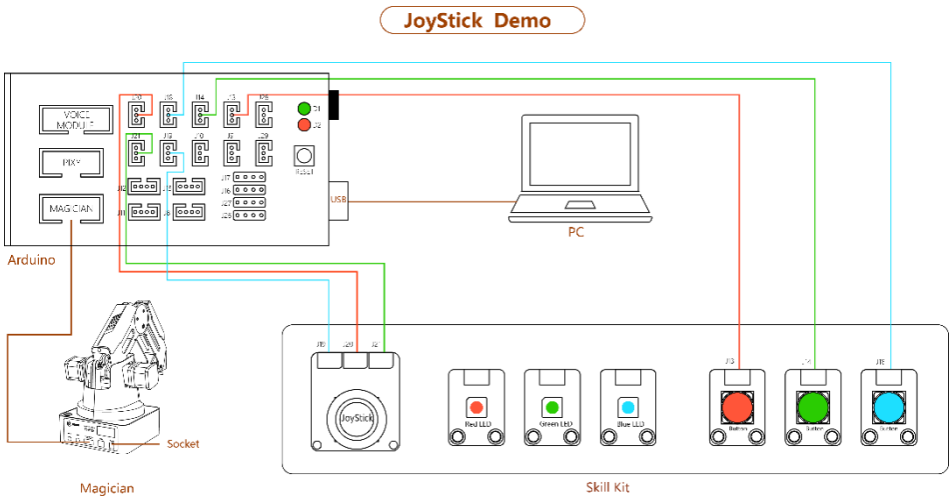


图 9.1 JoyStick 连接示意图

气泵与机械臂的连接如图 9.2所示。

图 9.2 气泵连接图

说明

如果用户需将套件连接至Arduino Mega2560上其他接口，还需在“SmartKit.h”文件中修改套件连接的对应接口。

程序 9.1 定义连接至 Arduino 的引脚

```
#define JOYSTICK_XPIN     7      /* JoyStick X 轴连接的引脚 */
```

```
#define JOYSTICK_YPIN     6      /* JoyStick Y 轴连接的引脚  */
#define JOYSTICK_ZPIN     A5     /* JoyStick Z 轴连接的引脚  */

#define LED_REDPIN        9      /*  红色灯连接引脚  */
#define LED_GREENPIN      A1     /*  绿色灯连接引脚  */
#define LED_BLUEPIN       A3     /*  蓝色灯连接引脚  */

#define BUTTON_REDPIN     A0     /*  红色按钮连接引脚  */
#define BUTTON_GREENPIN A2     /*  绿色按钮连接引脚  */
#define BUTTON_BLUEPIN    A4     /*  蓝色按钮连接引脚  */
```

9.3 实现流程

该Demo通过摇杆X、Y轴控制机械臂前后左右移动，摇杆按键控制机械臂的移动速度，按键1控制机械臂向上移动，按键2控制机械臂向下移动，按键3控制气泵启停。其实现流程如图 9.3所示。

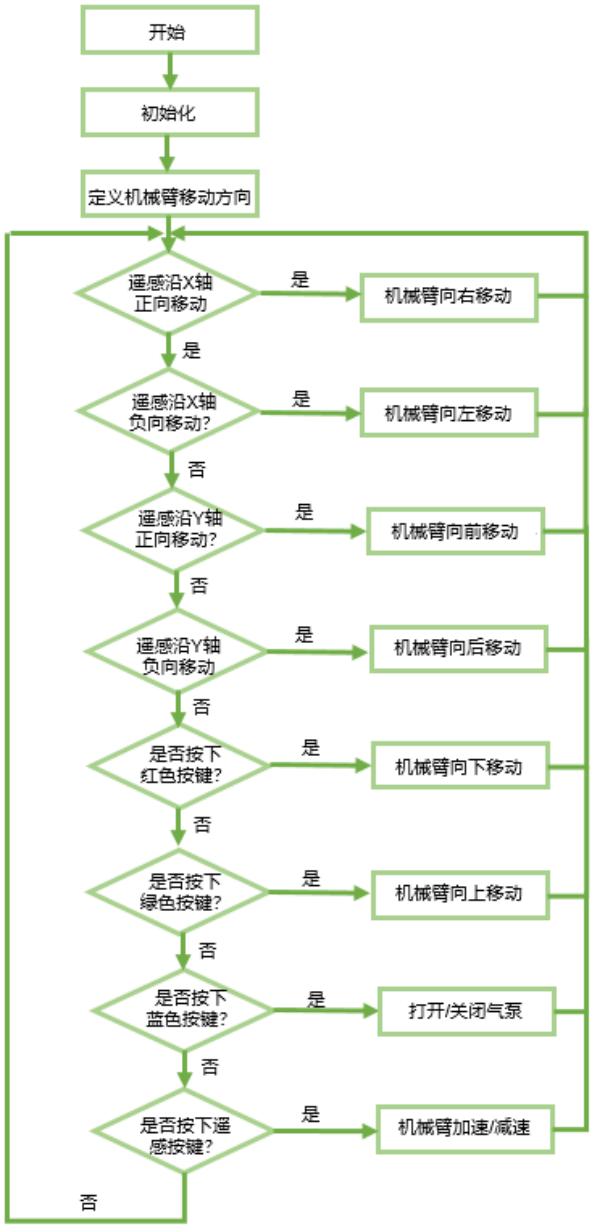


图 9.3 Demo 实现流程

9.4 关键代码说明

该Demo通过摇杆控制机械臂，需调用Dobot库和SmartKit库。调用该Demo前，请在Arduino界面的“项目 > 加载库”选择对应的库将其导入工程。。

⚠注意

调试Demo前请长按Dobot Magician底座背面的“Key”键进行回零操作。

（1） 程序初始化。

程序 9.2 程序初始化

```
void setup()
{
    Dobot_Init();
    SmartKit_Init();
}
```

（2） 根据摇杆移动的方向以及按键来定义机械臂移动方向以及气泵的启停。

说明

移动摇杆X轴或Y轴时，其模拟量输出范围为0~1023，如图 9.4所示。摇杆静止时，X轴模拟量输出为512，Y轴模拟量输出为508。

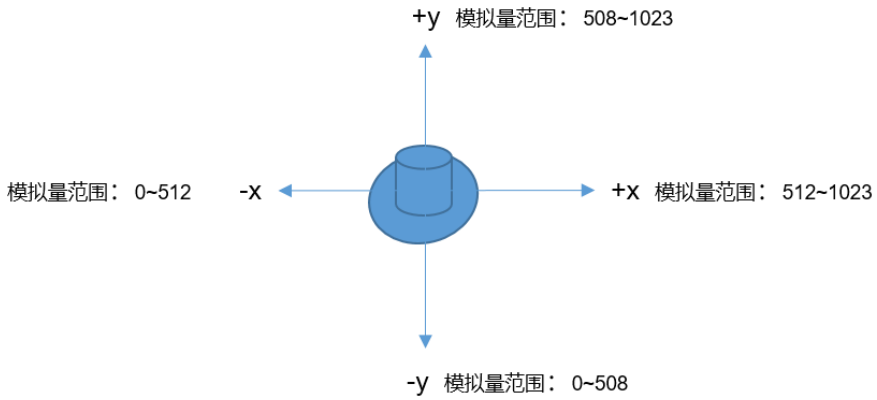


图 9.4 模拟量输出范围

程序 9.3 通过摇杆及按键定义机械臂移动方向

```
int x = 0,y = 0,z = 0,b1 = 0,b2 = 0,b3 = 0;
  int direction = 0;                                    //定义摇杆方向
      x = SmartKit_JoyStickReadXYValue(AXISX);
    y = SmartKit_JoyStickReadXYValue(AXISY);
    z = SmartKit_JoyStickCheckPressState();
    b1 = SmartKit_ButtonCheckState(RED);
    b2 = SmartKit_ButtonCheckState(GREEN);
    b3 = SmartKit_ButtonCheckState(BLUE);
    if(y > 600){                                        //摇杆沿 Y 轴正方向移动
    direction = 1;
  }
  else if(y < 400){                                     // 摇杆沿 Y 轴负方向移动
    direction = 2;
```

```
  }
…...
……
```

（3） 通过移动摇杆来控制机械臂移动及气泵启停。

程序 9.4 通过摇杆控制机械臂和气泵

```
  switch(direction){
     case 1:
          Serial.println("forward");
          Dobot_SetPTPCmdEx(MOVL_INC,20,0,0,0);          //机械臂向前移动
          Serial.print("x=");
          Serial.println(x);
          Serial.print("y=");
          Serial.println(y);
          break;
      case 2:
          Serial.println("backward");
          Dobot_SetPTPCmdEx(MOVL_INC,-20,0,0,0);                  //  机械臂向后移动
          Serial.print("x=");
          Serial.println(x);
          Serial.print("y=");
          Serial.println(y);
          break;
……
……
}
```

10. DobotPixy Demo

10.1 介绍

该Demo通过Pixy视觉识别模块和Dobot Magician来识别不同颜色的物块并分拣。

10.2 硬件连接

该Demo所需模块：Arduino Mega2560、Pixy视觉识别模块、Dobot Magician以及吸盘套件。Pixy视觉识别模块安装与配置请参见*附录C 安装与配置Pixy*，吸盘套件安装方法请参见附录B 安装吸盘套件。

Pixy视觉识别模块、Dobot Magician与Arduino连接如图 10.1所示。

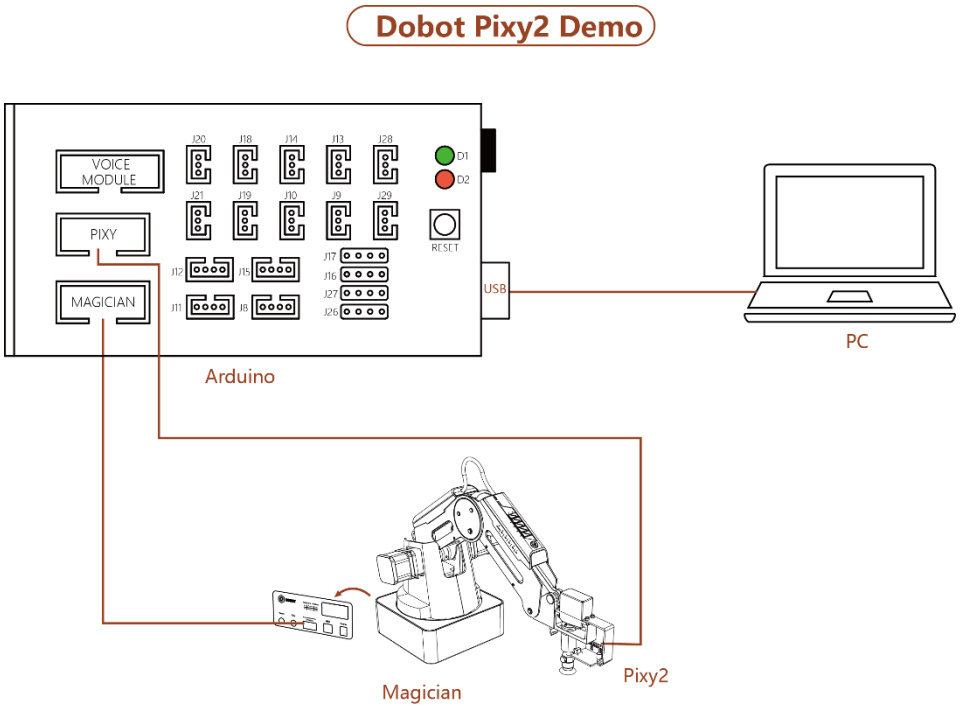


图 10.1 DobotPixy2 连接示意图

说明

如果用户需将套件连接至Arduino Mega2560上其他接口，还需在“SmartKit.h”文件中修改套件连接的对应接口。

程序 10.1 定义连接至 Arduino 的引脚

```
#define JOYSTICK_XPIN     7      /* JoyStick X 轴连接的引脚 */
#define JOYSTICK_YPIN     6      /* JoyStick Y 轴连接的引脚 */
#define JOYSTICK_ZPIN     A5     /* JoyStick Z 轴连接的引脚 */

#define LED_REDPIN        9      /* 红色灯连接引脚 */
#define LED_GREENPIN      A1     /* 绿色灯连接引脚 */
```

```
#define LED_BLUEPIN         A3     /*  蓝色灯连接引脚  */


#define BUTTON_REDPIN      A0     /*  红色按钮连接引脚  */
#define BUTTON_GREENPIN A2     /*  绿色按钮连接引脚  */
#define BUTTON_BLUEPIN     A4     /*  蓝色按钮连接引脚  */
```

10.3 实现流程

假设识别四种不同颜色（红、黄、绿、蓝）的物块，每个颜色的物块各两个。将物块置于视觉范围内（视觉识别模块安装在Dobot Magician末端），待Pixy视觉识别模块识别到物块颜色后，Dobot Magicia对该颜色的物块进行分拣，详细实现流程如图 10.2所示。

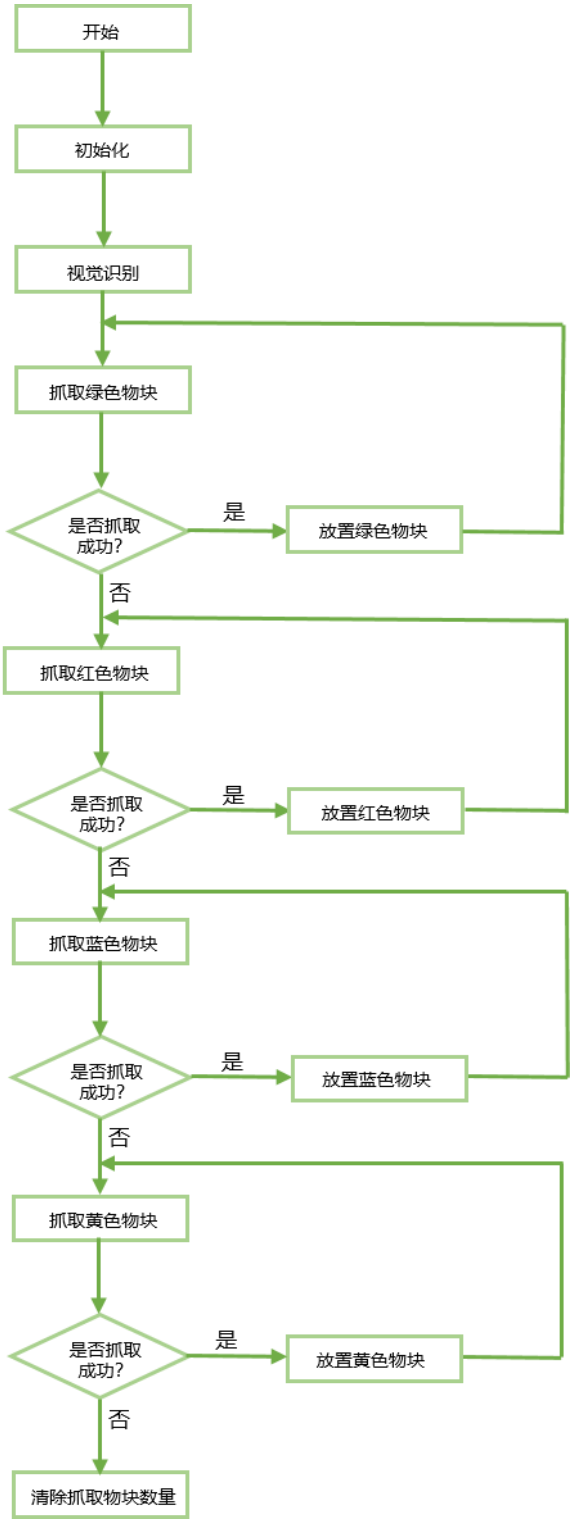


图 10.2 Demo 实现流程

10.4 关键代码说明

该Demo通过Pixy视觉识别模块和机械臂进行物块分拣，除调用SmartKit，Dobot Magician库外，还需调用Pixy库。调试该Demo前，请在Arduino界面的“项目 > 加载库”选择对应的

库将其导入工程。

⚠注意

调试Demo前请长按Dobot Magician底座背面的“Key”键进行回零操作。

（1） 程序初始化。

设置物块位置，需将 Dobot Magician 和 DobotStudio 连接，按住小臂上的圆形按钮并拖动小臂移动至物块位置，然后在 DobotStudio 的“操作面板”界面记录该位置坐标，以便写入 Demo 中。

视觉识别初始化流程请参见附录 D 视觉识别初始化流程。

程序 10.2 程序初始化

```
SmartKit_VISSetAT(197.2155, 0.0679, 61.0561, 25.5385);        //设置视觉识别位置（摄像头位置）
SmartKit_VISSetPixyMatrix(11, 153, 44, 45, 135, 91, 45, 46, 260, 11, 47, 43);     //设置物块的图像坐标
SmartKit_VISSetColorSignature(RED, 1);                        //设置物块颜色标签
SmartKit_VISSetColorSignature(GREEN, 2);
SmartKit_VISSetColorSignature(BLUE, 3);
SmartKit_VISSetColorSignature(YELLOW, 4);
SmartKit_VISSetDobotMatrix(245.1905, -61.6963, 214.2730, 1.5698, 169.7256, 67.9938);        //设置物块笛卡尔坐标，根据物块的图像坐标和笛卡尔坐标获取变换矩阵
SmartKit_VISSetGrapAreaZ(-65);
SmartKit_VISSetBlockTA(RED, 120, -135.9563, -66.9085, 0);          //设置红色物块放置区域
SmartKit_VISSetBlockTA(GREEN, 160, -135.9563, -66.9085, 0);
SmartKit_VISSetBlockTA(BLUE, 200, -135.9563, -66.9085, 0);
SmartKit_VISSetBlockTA(YELLOW, 240, -135.9563, -66.9085, 0);
SmartKit_VISSetBlockHeight(RED, 26);                          //设置物块高度
SmartKit_VISSetBlockHeight(GREEN, 26);
SmartKit_VISSetBlockHeight(BLUE, 26);
SmartKit_VISSetBlockHeight(YELLOW, 26);
SmartKit_VISInit();                                           //视觉识别初始化
SmartKit_Init();
```

（2） 视觉识别物块并将物块搬运至对应位置。如果同一类颜色有多个物块时，分拣时将多个物块堆叠。

程序 10.3 分拣物块

```
SmartKit_VISRun();                    //视觉识别，获取物块数量、颜色、坐标等
```

```
    color = GREEN;
    Dobot_SetPTPJumpParams(10);                    //设置机械臂抬升高度
    while (SmartKit_VISGrabBlock(color, 1, 0) == TRUE)        //抓取物块
    {
        Dobot_SetPTPJumpParams(30);                    //设置机械臂抬升高度
        SmartKit_VISPlaceBlock(color);                 //放置物块
    };
     SmartKit_VISSetBlockPlaceNum(color, 0);           //清除物块放置数量
……
……
```

11. VoicePixy Demo

11.1 介绍

该Demo通过Pixy视觉识别模块、LD3320语音识别模块，并结合Dobot Magician来识别不同颜色的物块并分拣。

11.2 硬件连接

该Demo所需模块：Arduino Mega2560、Pixy视觉识别模块、LD3320语音识别模块、LED模块、按键、Dobot Magician以及吸盘套件。Pixy视觉识别模块安装与配置请参见附录C 安装与配置Pixy，吸盘套件安装方法请参见附录B 安装吸盘套件。

Pixy视觉识别模块、LD3320语音识别模块、LED模块、按键以及Dobot Magician与Arduino之间连接如图 11.1所示。

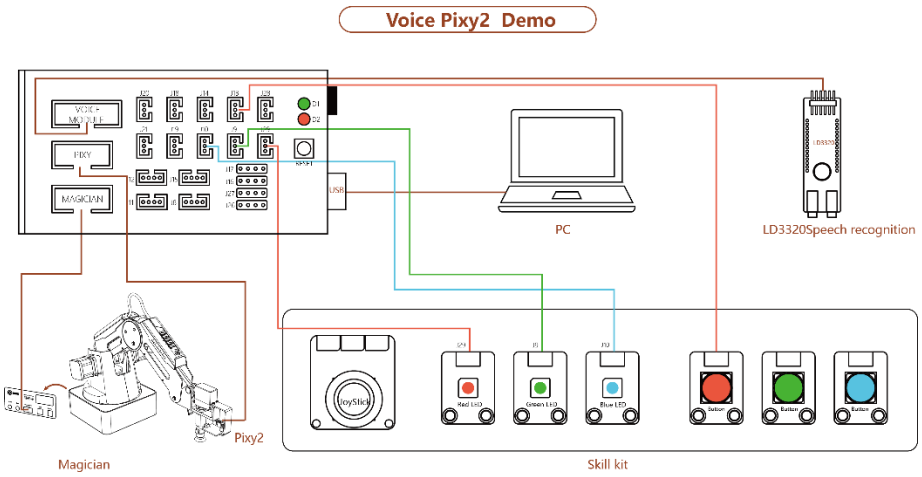


图 11.1 VoicePixy2 连接示意图

⚠注意

LD3320语音模块未做防反插设计，所以连接至Arduino时请务必按图 11.2所示进行连接。

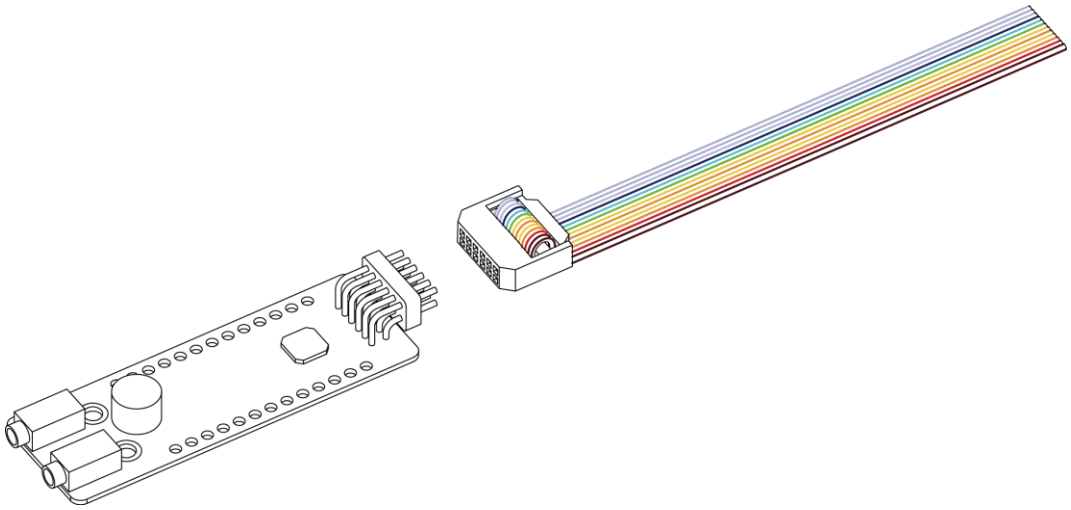


图 11.2 语音模块连接

说明

如果用户需将套件连接至Arduino Mega2560上其他接口，还需在“SmartKit.h”文件中修改套件连接的对应接口。

程序 11.1 定义连接至 Arduino 的引脚

```
#define JOYSTICK_XPIN     7      /* JoyStick X 轴连接的引脚 */
#define JOYSTICK_YPIN     6      /* JoyStick Y 轴连接的引脚 */
#define JOYSTICK_ZPIN     A5     /* JoyStick Z 轴连接的引脚 */

#define LED_REDPIN        9      /* 红色灯连接引脚 */
#define LED_GREENPIN      A1     /* 绿色灯连接引脚 */
#define LED_BLUEPIN       A3     /* 蓝色灯连接引脚 */

#define BUTTON_REDPIN     A0     /* 红色按钮连接引脚 */
#define BUTTON_GREENPIN A2     /* 绿色按钮连接引脚 */
#define BUTTON_BLUEPIN    A4     /* 蓝色按钮连接引脚 */
```

11.3 实现流程

假设识别四个不同颜色（红、黄、绿、蓝）的物块。将物块置于视觉范围内（视觉识别模块安装在Dobot Magician末端），当用户按下按键，所有LED指示灯亮时，此时对着语音识别模块（LD3320）上的麦克风说出待识别的颜色，并松开按键。待LD3320语音识别模块识别指令后，所有LED指示灯灭。Pixy视觉模块识别到该颜色后，Dobot Magicia对该颜色的物块进行分拣，详细实现流程如图 11.3所示。

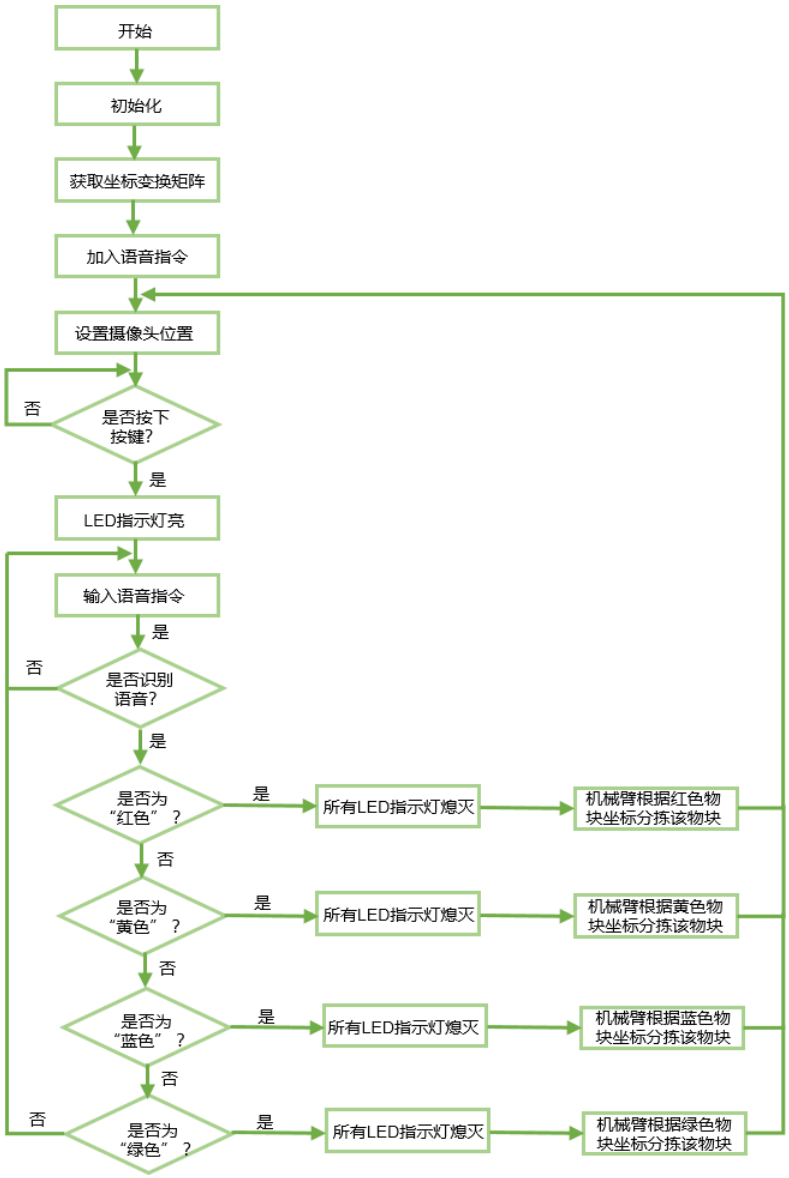


图 11.3 Demo 实现流程

11.4 关键代码说明

该Demo通过Pixy视觉模块、LD3320语音识别模块，并结合Dobot Magician来识别不同颜色的物块并分拣，除调用SmartKit，Dobot Magician库外，还需调用Pixy库。调试该Demo前，请在Arduino界面的“项目 > 加载库”选择对应的库将其导入工程即可。

⚠注意

调试Demo前请长按Dobot Magician底座背面的“Key”键进行回零操作。

（1） 程序初始化。

设置物块位置，需将 Dobot Magician 和 DobotStudio 连接，按住小臂上的圆

形按钮并拖动小臂移动至物块待放置的位置，然后在 DobotStudio 的“操作面板”界面记录该位置坐标，以便写入 Demo 中。

视觉识别初始化流程请参见附录 D 视觉识别初始化流程。

程序 11.2 程序初始化

```
SmartKit_VISSetAT(197.2155, 0.0679, 61.0561, 25.5385);     //设置视觉识别位置（摄像头位置）
SmartKit_VISSetPixyMatrix(11, 153, 44, 45, 135, 91, 45, 46, 260, 11, 47, 43);     //设置物块的图像坐标
SmartKit_VISSetColorSignature(RED, 1);                    //设置物块颜色标签
SmartKit_VISSetColorSignature(GREEN, 2);
SmartKit_VISSetColorSignature(BLUE, 3);
SmartKit_VISSetColorSignature(YELLOW, 4);
SmartKit_VISSetDobotMatrix(245.1905, -61.6963, 214.2730, 1.5698, 169.7256, 67.9938);        设置物块笛卡尔坐标，根据物块笛卡尔坐标和图形坐标获取变换矩阵
SmartKit_VISSetGrapAreaZ(-65);
SmartKit_VISSetBlockTA(RED, 120, -135.9563, -66.9085, 0);                    //设置红色物块放置区域
SmartKit_VISSetBlockTA(GREEN, 160, -135.9563, -66.9085, 0);
SmartKit_VISSetBlockTA(BLUE, 200, -135.9563, -66.9085, 0);
SmartKit_VISSetBlockTA(YELLOW, 240, -135.9563, -66.9085, 0);
SmartKit_VISSetBlockHeight(RED, 26);                          //设置红色物块高度
SmartKit_VISSetBlockHeight(GREEN, 26);
SmartKit_VISSetBlockHeight(BLUE, 26);
SmartKit_VISSetBlockHeight(YELLOW, 26);
SmartKit_VISInit();                                    //视觉识别初始化
SmartKit_VoiceCNInit();
SmartKit_Init();
SmartKit_VoiceCNAddCommand("hong se",1);          //增加语音指令
SmartKit_VoiceCNAddCommand("lan se",2);
SmartKit_VoiceCNAddCommand("lv se",3);
SmartKit_VoiceCNAddCommand("huang se",4);
SmartKit_VoiceCNStart();
```

（2） 根据语音指令识别物块颜色并搬运至对应位置。

程序 11.3 分拣物块

```
if(SmartKit_ButtonCheckState(RED)==DOWN)                    //检查是否按下按键
    {
        SmartKit_LedTurn(RED, ON);
        SmartKit_LedTurn(BLUE, ON);
```

```
        SmartKit_LedTurn(GREEN, ON);                          //点亮所有指示灯
        if (SmartKit_VoiceCNVoiceCheck(3) == TRUE)            //检测用户是否说出相应物块的颜色
        {
            color = GREEN;
            SmartKit_LedTurn(RED, OFF);                       //熄灭所有指示灯
            SmartKit_LedTurn(BLUE, OFF);
            SmartKit_LedTurn(GREEN, OFF);
            while(SmartKit_VISRun() == TRUE)                  //视觉识别,, 获取物块数量、颜色、坐标等
            {
                Dobot_SetPTPJumpParams(10);                   //设置机械臂抬升高度
                while (SmartKit_VISGrabBlock(color, 1, 0) == TRUE)     抓取相应颜色物块
                {
                    Dobot_SetPTPJumpParams(30);               //设置机械臂抬升高度
                    SmartKit_VISPlaceBlock(color);            //放置相应颜色物块
                }
            }
            SmartKit_VISSetBlockPlaceNum(color, 0);
        }
……
……
```

附录A 常用函数说明

SmartKit 常用函数说明

SmartKit库封装了Arduino人工智能套件常用的函数。用户调试Demo前需将SmartKit库加载至Arduino库中。

附表 1 SmartKit 初始化

原型	void SmartKit_Init(void)
功能	SmartKit初始化，包含摇杆、指示灯、按钮、语音识别
参数	无
返回	无

附表 2 按键状态检查

原型	int SmartKit_ButtonCheckState(char color)
功能	检查按键状态
参数	color：按键颜色，取值范围：BLUE，GREEN，RED
返回	按键状态：UP 或 DOWN

附表 3 读取摇杆数值

原型	int SmartKit_JoyStickReadXYValue(int axis)
功能	读取摇杆数值
参数	axis：摇杆移动方向；取值范围：AXISX， AXISY
返回	摇杆的数值

附表 4 检测摇杆按键状态

原型	int SmartKit_JoyStickCheckPressState(void)
功能	检查摇杆按键状态
参数	无
返回	按键状态：UP 或 DOWN

附表 5 检查 LED 指示灯的状态

原型	int SmartKit_LedCheckState(char color)
功能	检查LED指示灯的状态

参数	color：LED 指示灯颜色；取值范围：BLUE，GREEN，RED
返回	LED 指示灯状态：ON 或 OFF

附表 6 控制 LED 指示灯的状态

原型	int SmartKit_LedTurn(char color, int state)
功能	控制LED指示灯的状态
参数	color：LED 指示灯颜色；取值范围：BLUE，GREEN，RED state：LED 指示灯状态；取值范围：ON 或 OFF
返回	无

附表 7 中文语音初始化

原型	Void SmartKit_VoiceCNInit(void)
功能	初始化中文语音
参数	无
返回	无

附表 8 添加语音指令

原型	int SmartKit_VoiceCNAddCommand(char *Voice, int num)
功能	添加语音指令
参数	Voice：语音指针 num：语音序列号
返回	无
示例	SmartKit_VoiceCNAddCommand("xia jiang",1);

📖说明

用户可自行增加语音指令，比如增加指令标签相同，指令内容为“上”、“上升”、“升”的多条语音指令，，以便提高识别准确率，方便控制机械臂。LD3320语音识别模块支持用户自由编辑50条关键词语条。

附表 9 添加语音指令检测

原型	int SmartKit_VoiceCNVoiceCheck(int num)
功能	语音指令检测
参数	num：语音序列号

返回	TRUE：检测到语音指令 FALSE：未检测到语音指令

附表 10 开始检测语音指令

原型	void SmartKit_VoiceCNStart(void)
功能	开始检测语音指令
参数	无
返回	无

附表 11 视觉识别初始化

原型	void SmartKit_VISInit(void)
功能	视觉识别初始化 调用该指令前需调用 SmartKit_VISSetDobotMatrix(float x1, float y1,float x2, float y2,float x3, float y3)和 SmartKit_VISSetPixyMatrix(float x1, float y1, float length1, float wide1,float x2, float y2, float length2, float wide2,float x3, float y3, float length3, float wide3)指令，以便获取变换矩阵
参数	无
返回	无

📖说明

变换矩阵：根据三个点的图像坐标 $\mathbf{A}\begin{bmatrix}x_1 & x_2 & x_3\\ y_1 & y_2 & y_3\\ 1 & 1 & 1\end{bmatrix}$ 和对应笛卡尔坐标 $\mathbf{B}\begin{bmatrix}x_1' & x_2' & x_3'\\ y_1' & y_2' & y_3'\\ 1 & 1 & 1\end{bmatrix}$ 获取变换矩阵 R_T，以便视觉识别模块获取到物块的图像坐标后，根据变换矩阵获取对应机笛卡尔坐标，使机械臂移动至该位置。其公式为 $\mathbf{B} = \boldsymbol{R_T A}$。

附表 12 设置机械臂变换矩阵

原型	void SmartKit_VISSetDobotMatrix(float x1, float y1,float x2, float y2,float x3, float y3)
功能	设置机械臂变换矩阵 即视觉识别范围内放置三个物块，使机械臂运动至三个物块中心位置，获取它们的笛卡尔坐标
参数	x1：物块 1 的 X 轴坐标 y1：物块 1 的 Y 轴坐标

	x2：物块 2 的 X 轴坐标 y2：物块 2 的 Y 轴坐标 x3：物块 3 的 X 轴坐标 y3：物块 3 的 Y 轴坐标
返回	无

附表 13 设置 Pixy 变换矩阵

原型	void SmartKit_VISSetPixyMatrix(float x1, float y1, float length1, float wide1,float x2, float y2, float length2, float wide2,float x3, float y3, float length3, float wide3)
功能	设置Pixy变换矩阵 视觉识别范围内放置三个物块，通过PixyMon获取它们的图像坐标
参数	x1：物块 1 的 X 轴坐标 y1：物块 1 的 Y 轴坐标 length1：物块 1 的长度 wide1：物块 1 的宽度 x2：物块 2 的 X 轴坐标 y2：物块 2 的 Y 轴坐标 length2：物块 2 的长度 wide2：物块 2 的宽度 x3：物块 3 的 X 轴坐标 y3：物块 3 的 Y 轴坐标 length3：物块 3 的长度 wide3：物块 3 的宽度
返回	无

附表 14 设置抓取区域 Z 轴坐标

原型	void SmartKit_VISSetGrapAreaZ(float z)
功能	设置抓取区域Z轴坐标，即木块所在位置的高度
参数	z：抓取位置的高度
返回	无

附表 15 获取抓取区域 Z 轴坐标

原型	float SmartKit_VISGetGrapAreaZ(void)

功能	获取抓取区域Z轴坐标，即木块所在位置的高度
参数	无
返回	抓取位置的高度

附表 16 设置机械臂视觉区域坐标

原型	void SmartKit_VISSetAT(float x, float y, float z, float r)
功能	设置机械臂视觉识别区域坐标，即设置摄像头所在位置
参数	x：摄像头位置的 X 轴坐标 y：摄像头位置的 Y 轴坐标 z：摄像头位置的 Z 轴坐标 r：摄像头位置的 R 轴坐标
返回	无

附表 17 设置物块颜色标识

原型	char SmartKit_VISSetColorSignature(char color, char signature)
功能	设置物块颜色标识
参数	color：物块颜色；取值范围：RED，BLUE，YELLOW，GREEN signature：颜色标识；取值范围：1~7
返回	True：设置成功 FALSE：设置失败

附表 18 设置各颜色物块的放置区域

原型	char SmartKit_VISSetBlockTA(char color, float x, float y, float z, float r)
功能	设置各颜色物块的放置区域
参数	color：物块颜色；取值范围：RED，BLUE，YELLOW，GREEN x：放置区域的 X 轴坐标 y：放置区域的 Y 轴坐标 z：放置区域的 Z 轴坐标 r：放置区域的 R 轴坐标
返回	TRUE：设置成功 FALSE：设置失败

附表 19 获取视觉识别检测到的各颜色物块数量

原型	char SmartKit_VISGetBlockCheckNum(char color)

功能	获取视觉识别检测到的各颜色物块数量
参数	color：物块颜色；取值范围：RED，BLUE，YELLOW，GREEN
返回	物块数量

附表 20 设置各颜色物块放置数量

原型	char SmartKit_VISSetBlockPlaceNum(char color, int placeNum)
功能	设置各颜色物块放置数量
参数	color：物块颜色；取值范围：RED，BLUE，YELLOW，GREEN placeNum：放置的数量
返回	TURE：设置成功 FALSE：设置失败

附表 21 获取各颜色物块放置数量

原型	char SmartKit_VISGetBlockPlaceNum(char color)
功能	获取各颜色物块放置数量
参数	color：物块颜色；取值范围：RED，BLUE，YELLOW，GREEN
返回	物块数量

附表 22 设置各颜色物块高度

原型	char SmartKit_VISSetBlockHeight(char color, float height)
功能	设置各颜色物块高度
参数	color：物块颜色；取值范围：RED，BLUE，YELLOW，GREEN height：物块高度
返回	TRUE：设置成功 FALSE：设置失败

附表 23 清除视觉识别检测到的物块数量

原型	void SmartKit_VISBlockParmCheckNumClear(void)
功能	清除视觉识别检测到的物块数量
参数	无
返回	无

附表 24 视觉识别

原型	char SmartKit_VISRun(void)
功能	执行视觉识别，获取物块数量、颜色、坐标等
参数	无
返回	TRUE：识别成功 FALSE：识别失败，可能未检测到物块或物块颜色标记设置有误

附表 25 抓取物块

原型	char SmartKit_VISGrabBlock(char color, int blockNum, float r)
功能	抓取物块
参数	color：物块颜色；取值范围：RED，BLUE，YELLOW，GREEN blockNum：抓取物块的编号 r：抓取物块时旋转的角度
返回	TRUE：抓取成功 FALSE：抓取失败

附表 26 放置物块

原型	char SmartKit_VISPlaceBlock(char color)
功能	放置物块
参数	color：物块颜色；取值范围：RED，BLUE，YELLOW，GREEN
返回	TRUE：放置成功 FALSE：放置失败

Dobot Magician 常用函数说明

Dobot Magician通过底座背面的UART接口（10PIN）与Arduino进行通信，采用Dobot通信协议。我们已提供Dobot库，该库封装了部分Dobot Magician API，可直接调用即可控制Dobot Magician。本节对Demo中常用函数进行说明。更多Dobot Magician API说明，请参见《Dobot Magician API接口文档》。

附表 27 设置机械臂运动模式以及目标点坐标

原型	void Dobot_SetPTPCmdEx(uint8_t Model,float x,float y,float z,float r)
功能	设置机械臂运动模式以及目标点坐标
参数	Mode 取值如下： enum {

	JUMP_XYZ, //JUMP 模式，（x,y,z,r）为笛卡尔坐标系下的目标点坐标 MOVJ_XYZ, //MOVJ 模式，（x,y,z,r）为笛卡尔坐标系下的目标点坐标 MOVL_XYZ, //MOVL 模式，（x,y,z,r）为笛卡尔坐标系下的目标点坐标 JUMP_ANGLE, //JUMP 模式，（x,y,z,r）为关节坐标系下的目标点坐标 MOVJ_ANGLE, //MOVJ 模式，（x,y,z,r）为关节坐标系下的目标点坐标 MOVL_ANGLE, //MOVL 模式，（x,y,z,r）为关节坐标系下的目标点坐标 MOVJ_INC, //MOVJ 模式，（x,y,z,r）为关节坐标系下的坐标增量 MOVL_INC, //MOVL 模式，（x,y,z,r）为笛卡尔坐标系下的坐标增量 MOVJ_XYZ_INC, //MOVJ 模式，（x,y,z,r）为笛卡尔坐标系下的坐标增量 JUMP_MOVL_XYZ, //JUMP 模式，平移时运动模式为 MOVL。（x,y,z,r）为笛卡尔坐标系下的坐标增量 }; x,y,z,r //目标点坐标参数，可为笛卡尔坐标、关节坐标、笛卡尔坐标增量或关节坐标增量
返回	无

附表 28 控制气泵启停

原型	void Dobot_SetEndEffectorSuctionCupEx(bool issuck)
功能	控制气泵启停
参数	issuck：是否开启气泵。true：开启；false：停止
返回	无

附表 29 设置机械臂移动速度和加速度比率

原型	void Dobot_SetJOGCommonParamsEx(float velocityRatio, float accelerationRatio)
功能	设置机械臂移动速度和加速度比率
参数	velocityRatio：速度比率 accelerationRatio：加速度比率

返回	无

附表 30 获取机械臂位姿

原型	float Dobot_GetPoseEx(uint8_t temp)
功能	获取机械臂位姿

参数	temp：机械臂坐标轴
	temp 取值范围如下：
	enum {
	X，……//X 轴
	Y，……//Y 轴
	Z，……//Z 轴
	R，……//R 轴
	JOINT1，……//关节坐标轴 1
	JOINT2，……//关节坐标轴 2
	JOINT3，……//关节坐标轴 3
	JOINT4，……//关节坐标轴 4
	};

返回	返回已获取的某个轴的坐标值

附表 31 获取设备时钟

原型	uint32_t Dobot_GetDeviceTimeEx(void)
功能	获取设备时钟
参数	无
返回	返回设备时钟时间

附表 32 设置滑轨状态

原型	void Dobot_SetDeviceWIthLEx(bool isWithL)
功能	设置机械臂滑轨状态
参数	isWithL:是否开启滑轨。1：开启滑轨，0：关闭滑轨
返回	无

附表 33 执行回零功能

原型	void Dobot_SetHOMECmdEx(void)
功能	执行机械臂回零功能

参数	无
返回	无

附表 34 设置末端坐标偏移量

原型	void Dobot_SetEndEffectorParamsEx(float x,float y, float z)
功能	设置末端坐标偏移参数，一般在末端安装了执行器才需设置。
参数	x：X 轴方向偏移量 y：Y 轴方向偏移量 z：Z 轴方向偏移量
返回	无

附表 35 设置激光状态

原型	void Dobot_SetEndEffectorLaserEx(uint8_t isEnable,float power)
功能	设置激光状态
参数	isEnable：输出电平。0：低电平，1：高电平 power：占空比，取值范围 0~100
返回	无

附表 36 设置抓取状态

原型	void Dobot_SetEndEffectorGripperEx(bool isEnable,bool isGriped)
功能	设置末端抓取工具状态
参数	isEnable:末端使能。0：未使能，1：使能 isGriped:控制夹爪抓取或释放。0：释放，1：抓取
返回	无

附表 37 设置点动模式下各关节坐标轴的速度和加速度

原型	void Dobot_SetJOGJointParamsEx(float velocityJ1,float accelerationJ1,float velocityJ2,float accelerationJ2,float velocityJ3,float accelerationJ3,float velocityJ4,float accelerationJ4)
功能	设置机械臂点动时各关节坐标轴的速度和加速度
参数	velocityJ1、velocityJ2、velocityJ3、velocityJ4：分别为点动模式四轴的关节速度 accelerationJ1、accelerationJ2、accelerationJ3、accelerationJ4：分别为点动模式四轴的关节加速度
返回	无

附表 38 执行点动指令

原型	void Dobot_SetJOGCmdEx(uint8_t model)
功能	执行点动指令，设置点动相关参数后可调用该接口

参数	model：点动命令
	model取值说明如下：
	enum {
	AP_DOWN, ……//X+/Joint1+
	AN_DOWN, ……//X-/Joint1-
	BP_DOWN, …… //Y+/Joint2+
	BN_DOWN, …… //Y-/Joint2-
	CP_DOWN, …… //Z+/Joint3+
	CN_DOWN, …… //Z-/Joint3-
	DP_DOWN, ……//R+/Joint4+
	DN_DOWN, …… //R-/Joint4-
	LP_DOWN, ……//L+
	LN_DOWN ……//L-
	};

返回	无

附表 39 设置 PTP 运动的速度百分比和加速度百分比

原型	void Dobot_SetPTPCommonParamsEx(float velocityRatio,float accelerationRatio)
功能	设置 PTP 运动的速度百分比和加速度百分比
参数	velocityRatio： PTP 模式速度百分比，关节坐标轴和笛卡尔坐标轴共用 accelerationRatio：PTP 模式加速度百分比，关节坐标轴和笛卡尔坐标轴共用
返回	无

附表 40 设置 PTP 模式下各关节坐标轴的速度和加速度

原型	void Dobot_SetPTPJointParamsEx(float velocityJ1,float accelerationJ1,float velocityJ2,float accelerationJ2,float velocityJ3,float accelerationJ3,float velocityJ4,float accelerationJ4)
功能	设置PTP模式下各关节坐标轴的速度和加速度
参数	velocityJ1、velocityJ2、velocityJ3、velocityJ4：分别为 PTP 模式下四轴的关节速度

	accelerationJ1、accelerationJ2、accelerationJ3、accelerationJ4：分别为 PTP 模式下四轴的关节加速度
返回	无

附表 41 设置 PTP 模式下滑轨速度和加速度

原型	void Dobot_SetPTPLParamsEx(float velocityRatio, float accelerationRatio)
功能	设置PTP模式下滑轨速度和加速度
参数	velocityRatio：PTP 模式下滑轨速度 accelerationRatio：PTP 模式下滑轨加速度
返回	无

附表 42 设置 JUMP 模式下抬升高度

原型	void Dobot_SetPTPJumpParamsEx(float jumpHeight)
功能	设置JUMP模式下抬升高度
参数	jumpHeight：抬升高度
返回	无

附表 43 执行带滑轨的 PTP 指令

原型	void Dobot_SetPTPWithLCmdEx(uint8_t Model,float x,float y,float z,float r,float l)
功能	执行带滑轨的PTP指令
参数	Model：PTP 模式 Model取值说明如下： enum { JUMP_XYZ, …//JUMP 模式，（x,y,z,r）为笛卡尔坐标系下的目标点坐标 MOVJ_XYZ, … //MOVJ 模式，（x,y,z,r）为笛卡尔坐标系下的目标点坐标 MOVL_XYZ,… //MOVL 模式，（x,y,z,r）为笛卡尔坐标系下的目标点坐标 JUMP_ANGLE, … //JUMP 模式，（x,y,z,r）为关节坐标系下的目标点坐标 MOVJ_ANGLE, … //MOVJ 模式，（x,y,z,r）为关节坐标系下的目标点坐标 MOVL_ANGLE, …//MOVL 模式，（x,y,z,r）为关节坐标系下的目标点坐标 MOVJ_INC, …//MOVJ 模式，（x,y,z,r）为关节坐标系下的坐标增量 MOVL_INC, …//MOVL 模式，（x,y,z,r）为笛卡尔坐标系下的坐标增量 MOVJ_XYZ_INC, …//MOVJ 模式，（x,y,z,r）为笛卡尔坐标系下的坐标增量 JUMP_MOVL_XYZ, … //JUMP 模式，平移时运动模式为MOVL。

	（x,y,z,r）为笛卡尔坐标系下的坐标增量 }; x、y、z、r：为坐标参数，可为笛卡尔坐标、关节坐标、笛卡尔坐标增量或关节坐标增量 l：滑轨运行距离
返回	无

附表 44 设置 I/O 复用

原型	void Dobot_SetIOMultiplexingEx(uint8_t address,uint8_t function)
功能	设置I/O复用

参数	address：I/O 地址，取值范围：1~20,详细说明请参考附录 E V1 版本机械臂 I/O 接口复用说明
	function： I/O 功能
	function 取值说明如下：
	typedef enum {
	IOFunctionDummy, …… //不配置功能
	IOFunctionDO, ……//I/O 输出
	IOFunctionPWM, ……//PWM 输出
	IOFunctionDI, ……//I/O 输入
	IOFunctionADC ……//A/D 输入
	IOFunctionDIPU…… //上拉输入
	IOFunctionDIPD ……//下拉输入
	};

返回	无

附表 45 设置 I/O 输出电平

原型	void Dobot_SetIODOEx(uint8_t address,uint8_t value)
功能	设置I/O输出电平
参数	address：I/O 地址。取值范围：1~20 。 value：输出电平。0：低电平。1：高电平
返回	无

附表 46 设置 PWM 输出

原型	void Dobot_SetIOPWMEx(uint8_t address, float freq, float duty)
功能	设置PWM输出

参数	address：I/O 地址 freq：PWM 频率。取值范围：10HZ~1MHz duty：PWM 占空比。取值范围：0~100
返回	无

附表 47 读取 I/O 输入电平

原型	uint8_t Dobot_GetIODIEx(uint8_t address)
功能	读取I/O输入电平
参数	address：I/O 地址
返回	返回 I/O 输入电平值

附表 48 读取 A/D 输入

原型	uint16_t Dobot_GetIOADCEx(uint8_t address)
功能	读取A/D 输入
参数	address：I/O 地址
返回	返回 A/D 输入值

附表 49 设置扩展电机速度

原型	void Dobot_SetEMotorEx(uint8_t address, uint8_t enable, uint32_t speed)
功能	设置扩展电机速度
参数	address：电机编号。0：Stepper1， 1：Stepper2 enable：电机控制使能。0：未使能，1：使能 speed：电机控制速度（脉冲个数每秒）
返回	无

附表 50 设置扩展电机速度和移动距离

原型	void Dobot_SetEMotorSEx(uint8_t address, uint8_t enable, uint32_t speed, uint32_t deltaPulse)
功能	设置扩展电机速度和移动距离。当需要以一定速度运行一段距离时可调用此函数
参数	address：电机编号。0：Stepper1， 1：Stepper2 enable：电机控制使能。0：未使能，1：使能 speed：电机控制速度（脉冲个数每秒） deltaPulse：电机移动距离(脉冲个数)

返回	无

附表 51 设置使能颜色传感器

原型	void Dobot_SetColorSensorEx(uint8_t enable, uint8_t port)
功能	设置使能颜色传感器

参数	enable：使能标志。0：未使能，1：使能
	port：颜色传感器连接至机械臂的接口，请选择对应的接口
	port取值说明如下：
	enum {
	IF_PORT_GP1;
	IF_PORT_GP2;
	IF_PORT_GP4;
	IF_PORT_GP5;
	};

返回	无

附表 52 获取颜色传感器读数

原型	uint8_t Dobot_GetColorSensorEx(uint8_t color)
功能	获取颜色传感器读数
参数	color： 0：红色，1：绿色，2：蓝色
返回	返回传感器数值

附表 53 设置丢步检测阈值接口说明

原型	void Dobot_SetLostStepSetEx(float lostStepValue)
功能	设置丢步检测阈值，若不调用该接口，则默认丢步检测阈值为 5
参数	lostStepValue：丢步阈值
返回	无

附表 54 执行丢步检测

原型	void Dobot_SetLostStepCmdEx(void)
功能	执行丢步检测
参数	无
返回	无

附表 55 设置红外传感器

原型	void Dobot_SetIRSwitchEx(uint8_t enable, uint8_t port)
功能	设置红外传感器

参数	enable：使能标志。0：未使能。1：使能
	port：红外传感器连接至机械臂的接口，请选择对应的接口
	port取值说明如下：
	enum {
	IF_PORT_GP1;
	IF_PORT_GP2;
	IF_PORT_GP4;
	IF_PORT_GP5;
	};

返回	无

附表 56 获取红外传感器读数

原型	uint8_t GetIRSwitchEx(uint8_t port)
功能	获取红外传感器读数

参数	port：红外传感器连接至机械臂的接口，请选择对应的接口
	port 取值说明如下：
	enum {
	IF_PORT_GP1;
	IF_PORT_GP2;
	IF_PORT_GP4;
	IF_PORT_GP5;
	};

返回	返回传感器数值

Arduino 常用函数说明

Arduino人工智能套件Demo基于Arduino Mega2560开发，需调用Arduino API，本节对Arduino人工智能套件Demo中常用函数进行说明。

附表 57 配置指定数字引脚为输入或输出

原型	void pinMode(uint8_t pin, uint8_t mode)
功能	将指定的数字引脚配置为输入或输出
参数	pin：引脚编号

	mode：INPUT 或 OUTPUT
返回	无

附表 58 给定数字引脚写入高电平或低电平

原型	void digitalWrite(uint8_t pin, uint8_t value)
功能	给定一个数字引脚，写入高电平或低电平
参数	pin：引脚编号 value：HIGH 或 LOW
返回	无

附表 59 读取指定数字引脚的值

原型	int digitalRead(uint8_t pin)
功能	读取指定数字引脚的值
参数	pin：引脚编号
返回	HIGH 或 LOW

附表 60 给定模拟引脚写入值

原型	void analogWrite(uint8_t pin, int value)
功能	给定模拟引脚写入模拟值，用于控制LED指示灯的亮度或控制电机的转速
参数	pin：引脚编号 value：0~255。 0：OFF；1：ON
返回	无

附表 61 读取指定模拟引脚的值

原型	int analogRead(uint8_t pin)
功能	读取指定模拟引脚的值
参数	pin：引脚编号
返回	0~1023

Pixy 常用函数说明

当Arduino人工智能套件Demo使用了Pixy视觉模块时，还需调用Pixy API。本节对Demo

中常用函数进行说明。

附表 62 返回 Pixy 识别的物体数量

原型	uint16_t getBlocks()

功能	返回Pixy所识别的物体数量以及每个识别到的物块的数据，包括：
	• 视觉识别模块为Pixy：
	  • pixy.blocks[i].signature：被识别物体的标记编号（颜色）（1~7）
	  • pixy.blocks[i].x：被识别物体中心位置在X方向的坐标（0~319）
	  • pixy.blocks[i].y：被识别物体中心位置在Y方向的坐标（0~319）
	  • pixy.blocks[i].width：被识别物体的宽度（1~320）
	  • pixy.blocks[i].height：被识别物体的高度（1~200）
	• 视觉识别模块为Pixy2：
	  • pixy.ccc.blocks[i].m_signature：被识别物体的标记编号（颜色）（1~7）
	  • pixy.ccc.blocks[i].m_x：被识别物体中心位置在X方向的坐标（0~319）
	  • pixy.ccc.blocks[i].m_y：被识别物体中心位置在Y方向的坐标（0~319）
	  • pixy.ccc.blocks[i].m_width：被识别物体的宽度（1~320）
	  • pixy.ccc.blocks[i].m_height：被识别物体的高度（1~200）

参数	无
返回	返回Pixy所识别的物体数量

LD3320 语音模块常用函数说明

当Arduino人工智能套件Demo使用了LD3320语音模块时，还需调用语音模块API。本节对Demo中常用函数进行说明。

附表 63 增加语音指令

原型	void VoiceRecognition::addCommand(char *pass, int num)
功能	增加语音指令 最多仅支持50个命令，每条命令不超过75个字节
参数	pass：指令内容 num：指令标签，可重复
返回	无

附录B 安装吸盘套件

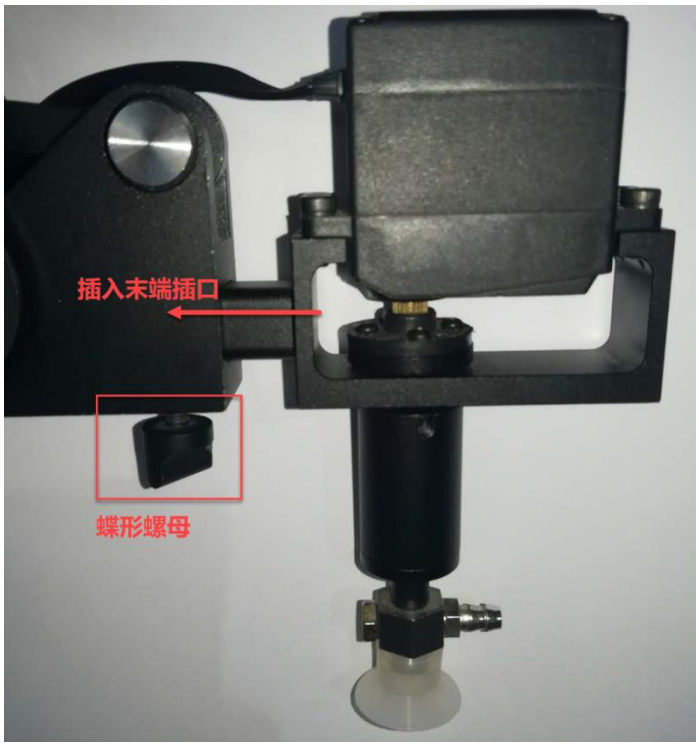
操作步骤

利用Dobot Magician进行物块分拣时，需通过吸盘套件来进行吸取和释放物块，所以末端需安装吸盘套件。

步骤 1 将气泵盒的电源线SW1接在机械臂后面板的“SW1”接口，信号线GP1接在“GP1”接口。如附图 1所示。

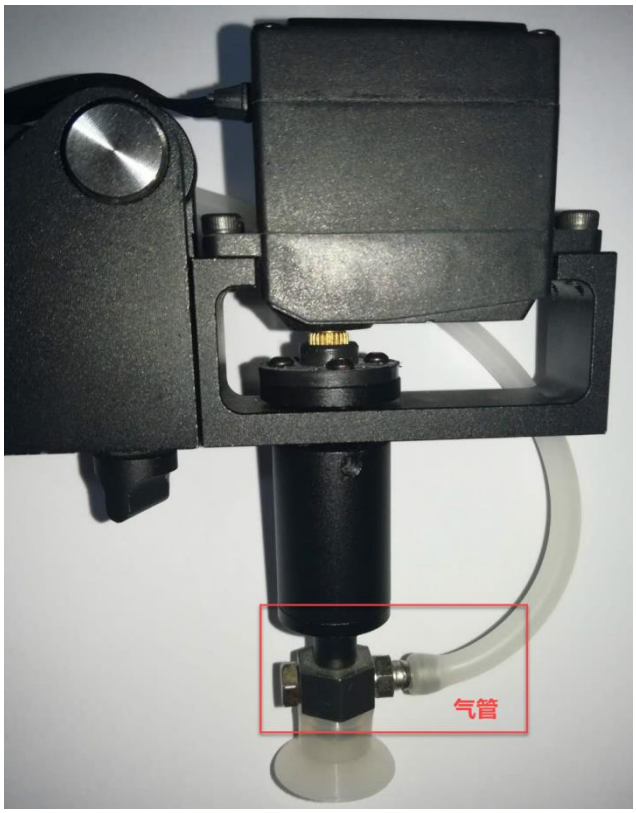
附图 1 连接气泵盒和机械臂

步骤 2 将吸盘套件插入机械臂末端插口中，然后通过蝶形螺母拧紧。如附图 2所示。



附图 2 安装吸盘套件

将气泵盒的气管连接在吸盘的气管接头上。如附图 3所示。



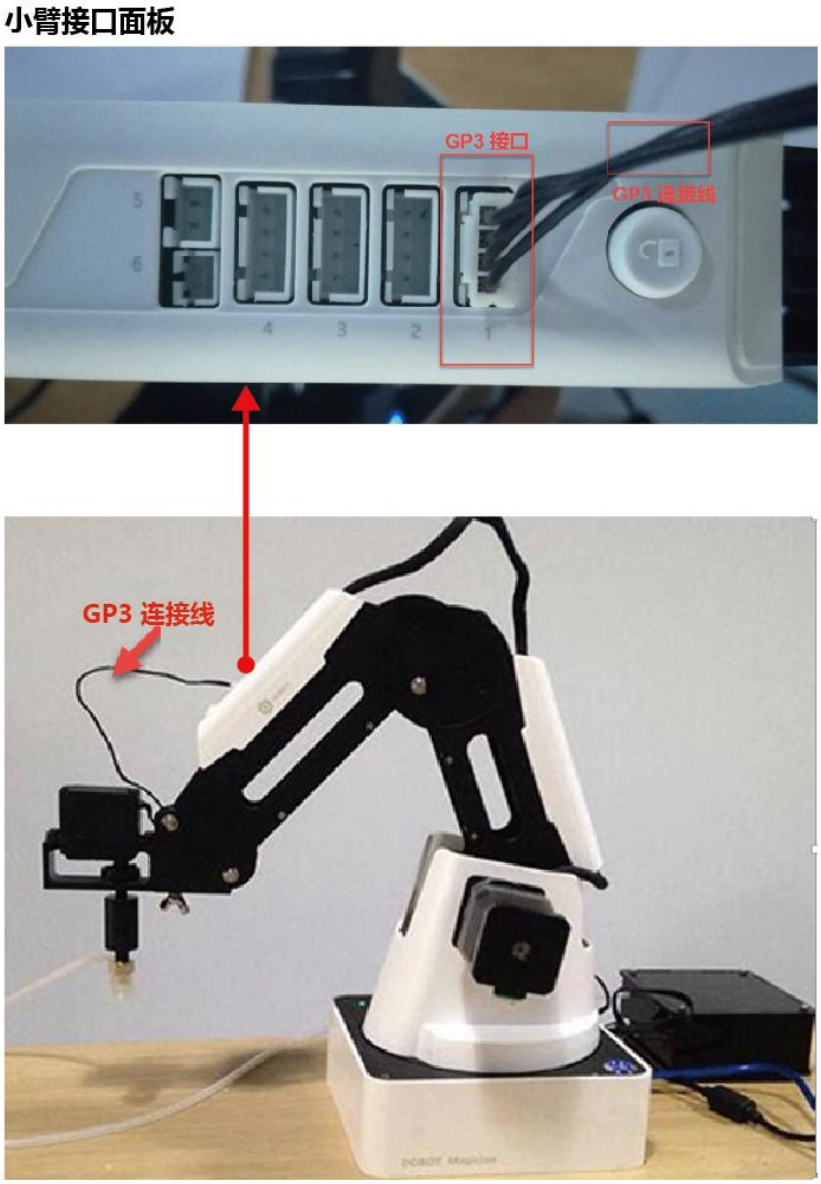
附图 3 连接气管

说明

如果吸盘套件配套的吸盘无法吸住小物块，可将吸盘更换为Arduino人工智能套件配套的吸盘，如附图 4所示。

附图 4 Arduino 人工智能套件配套的吸盘

步骤 3 将舵机连接线GP3接在小臂的“GP3”接口。如附图 5所示。



附图 5 连接“GP3”接口

附录C 安装与配置 Pixy

在调试DobotPixy Demo前，还需安装与配置Pixy视觉识别模块。

前提条件

- 已安装PixyMon，可从Arduino人工智能套件Demo中获取PixyMon安装包，获取路径：arduino套装/PixyMon。
- Dobot Magician末端已安装吸盘套件。
- 已获取小物块。

操作步骤

步骤 1 松开舵机上两颗M3*8杯头内六角螺丝，如附图 6所示。

附图 6 松开舵机上的螺丝

步骤 2 通过视觉夹具将Pixy2安装在舵机上，如附图 7所示。

附图 7 安装 Pixy2 视觉识别模块

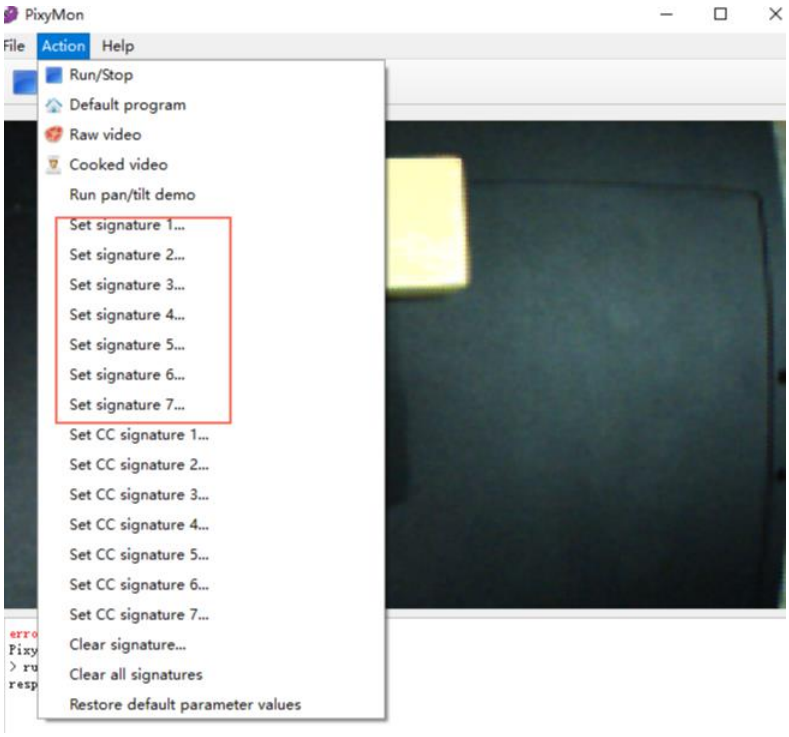
步骤 3 通过USB连接Pixy和PC。

步骤 4 将小物块放置于视觉范围内。

步骤 5 调节视觉识别模块上的摄像头焦距，将视野调制最佳状态，直至小物块能在PixyMon页面出现比较清晰的图形。

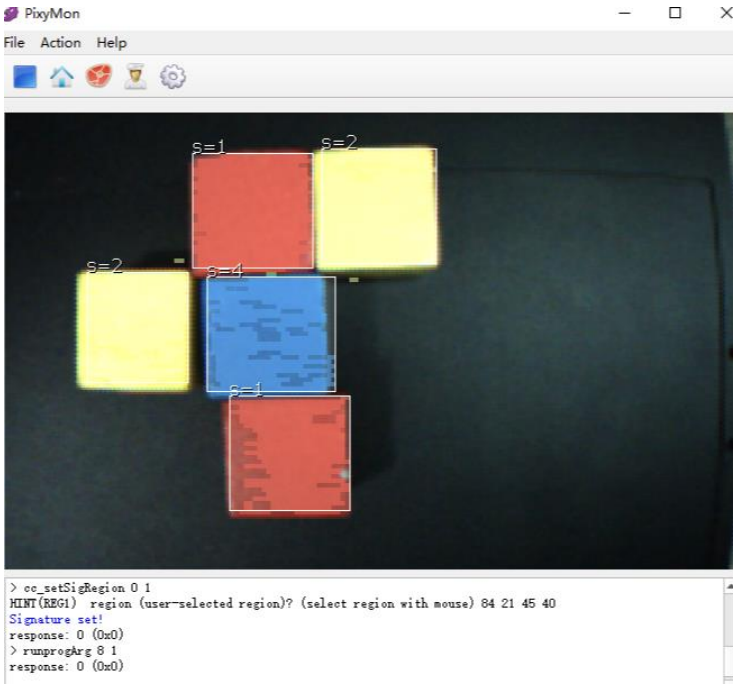
步骤 6 在PixyMon页面选择“Action > Set signature *x*”，选中物块的某一区域，对物块进行颜色标记，如附图 8所示。

每个颜色的物块，进行一次标记即可。Pixy最多仅支持标记7种颜色。



附图 8 颜色标记

颜色标记结果如附图 9所示。

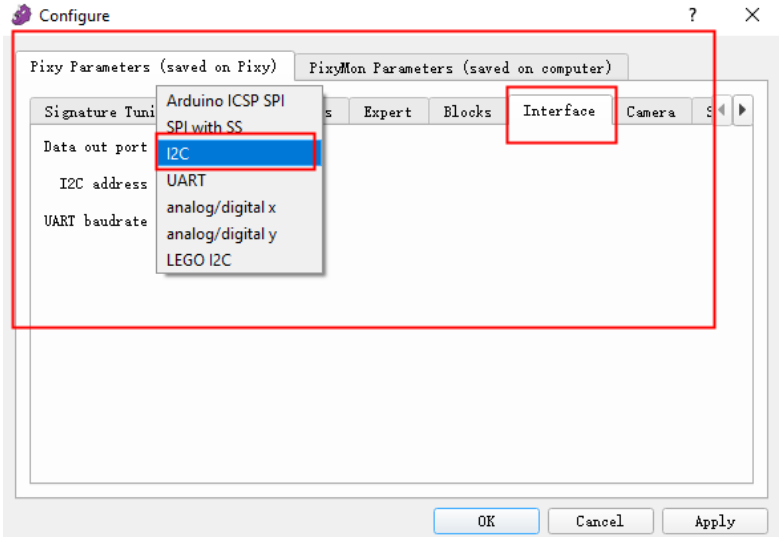


附图 9 标记结果

步骤 7 在PixyMon页面单击  。

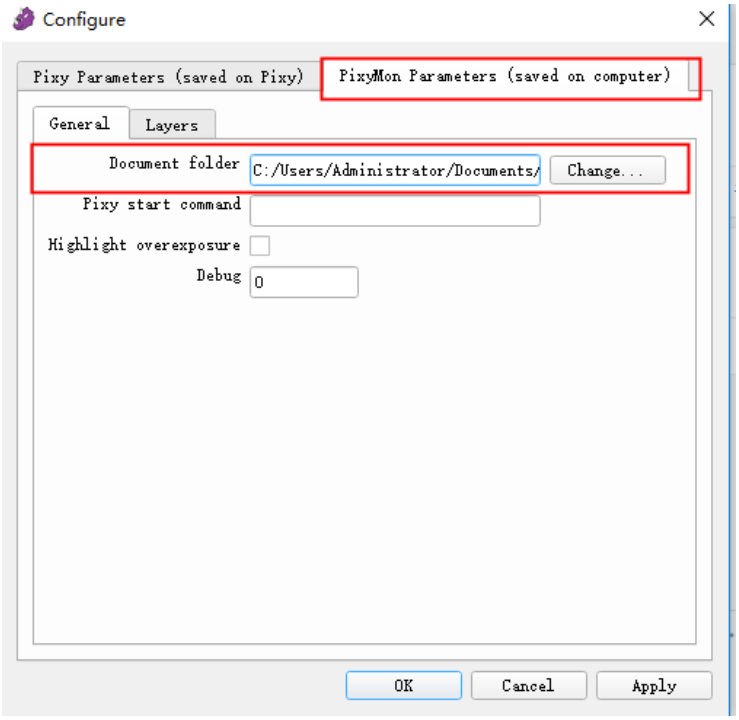
弹出“Configure”页面。

步骤 8 在“Configure”页面的“Pixy Parameters (saved on Pixy) > Interface”页签将“Data out port”选择为“I2C”，如附图 10所示。



附图 10 设置 Data out port

步骤 9 在“Configure”页面的“Pixy Parameters (saved on computer) > General”页签将“Document folder”选择为Pixymon软件的安装目录，如附图 11所示。



附图 11 设置 Document folder

步骤 10 拔掉连接Pixy与PC之间的USB连接线。

附录D 视觉识别初始化流程

视觉识别初始化流程包括设置摄像头位置、物块笛卡尔坐标、物块图像坐标、物块所在平面高度、物块放置区域、物块颜色高度以及物块颜色标签。详细如下所示。

步骤 1 连接机械臂和DobotStudio，并进行回零操作。本节不做详细说明，请参见《Dobot Magician 用户手册》。

步骤 2 安装Pixy并打开PixyMon软件。本节不做详细说明，请参见附录C 安装与配置Pixy。

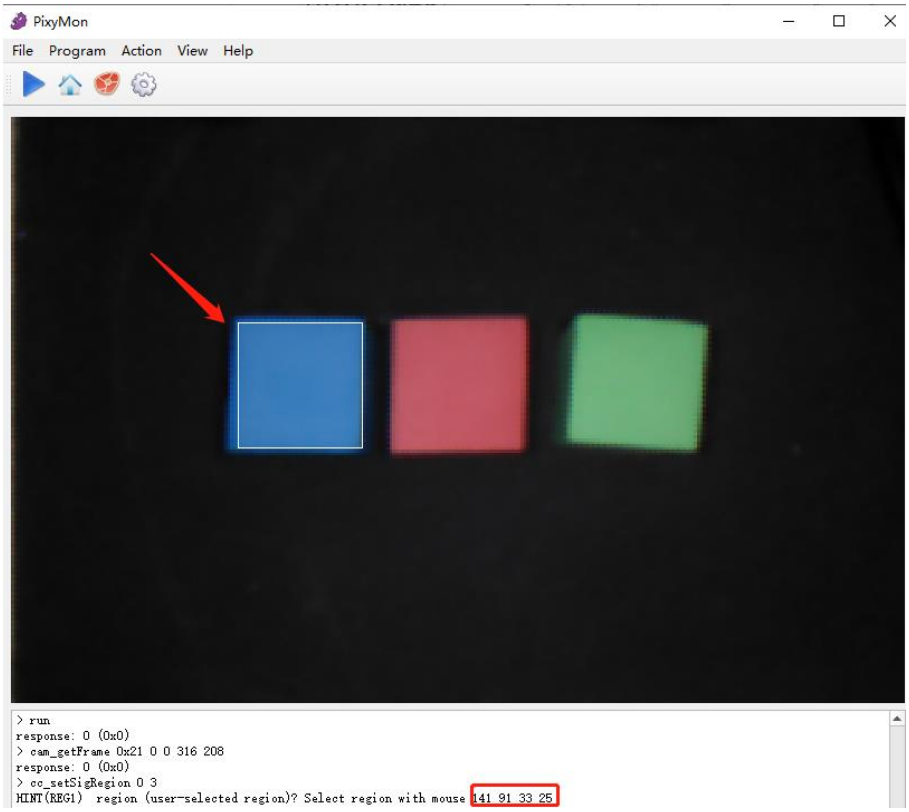
步骤 3 设置摄像头位置。

将机械臂移动至合适位置，以便在视觉范围内能识别物块。此时在DobotStudio界面的操作面板记录X、Y、Z、R的坐标，将其值按顺序写入SmartKit_VISSetAT(float x, float y, float z, float r)函数中。

步骤 4 将三个物块放置于视觉范围内。

步骤 5 设置三个物块图像坐标。

在PixyMon界面单击“Action > Set signature 1…”，分别框选三个物块对角线，此时在PixyMon界面显示对应物块的图像坐标，如附图 12所示。并按顺序写入SmartKit_VISSetPixyMatrix(float x1, float y1, float length1, float wide1, float x2, float y2, float length2, float wide2, float x3, float y3, float length3, float wide3)函数中。



附图 12 获取图像坐标

步骤 6 设置三个物块笛卡尔坐标。

按**步骤 5**的顺序移动机械臂至对应物块正上方，在DobotStudio界面的操作面板记录X、Y的坐标，并依次按顺序写入SmartKit_VISSetDobotMatrix(float x1, float y1,float x2, float y2, float x3, float y3)函数中。

⚠注意

物块的图像坐标和笛卡尔坐标需一一对应，否则获取变换矩阵失败。

步骤 7 设置物块所在平面高度。

移动机械臂至物块所在平面，在DobotStudio界面记录Z轴坐标，并写入SmartKit_VISSetGrapAreaZ(float z)函数中。

步骤 8 设置各颜色物块放置区域。

移动机械臂至各颜色物块待放置区域，在DobotStudio界面依次记录X、Y、Z、R的坐标，并写入SmartKit_VISSetBlockTA(char color, float x, float y, float z, float r)函数中。

步骤 9 设置各颜色物块高度。

将各颜色物块高度分别写入SmartKit_VISSetBlockHeight(char color, float height)函数中。

物块高度需用户自行通过测量工具获取，单位：毫米。

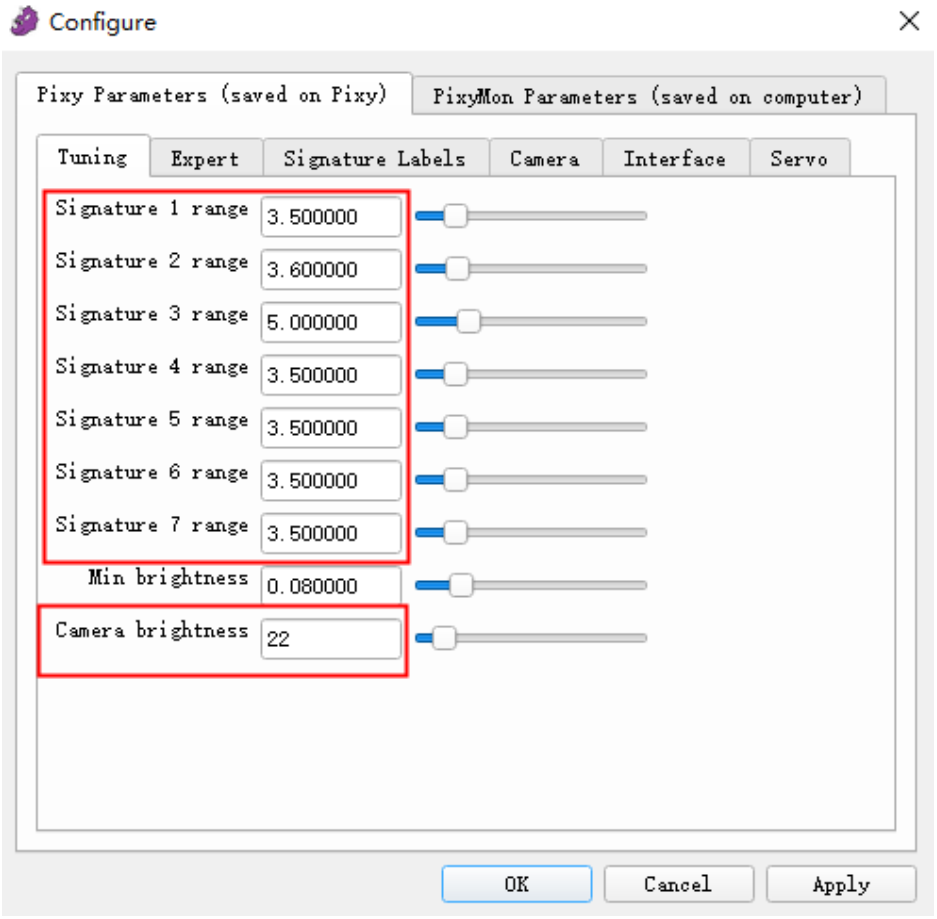
步骤 10 设置物块颜色标签。

在PixyMon界面单击“Action > Set signature x…”设置物块颜色标签，并写入SmartKit_VISSetColorSignature(char color, char signature)中。

其中，x表示对应物块颜色标签。每个颜色的物块，进行一次标记即可。Pixy最多仅支持标记7种颜色。

📖说明

若Pixy识别效果较差，可在“File > Configure > Pixy Parameters(saved on Pixy)”页签微调“Signature x range”以及“Camera brightness”，使得识别效果达到最佳，如附图 13所示。

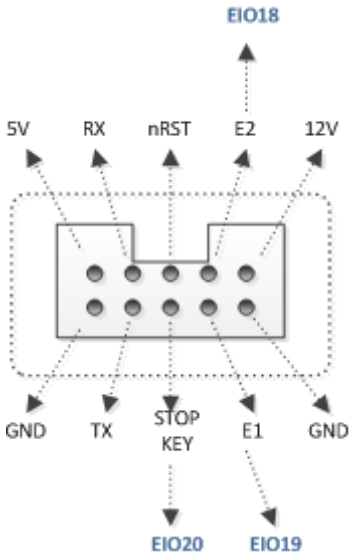


附图 13 调节范围和亮度

附录E V1 版本机械臂 I/O 接口复用说明

底座 UART 接口 I/O 复用说明

UART接口如附图 14所示，其I/O复用说明如附表 64所示。



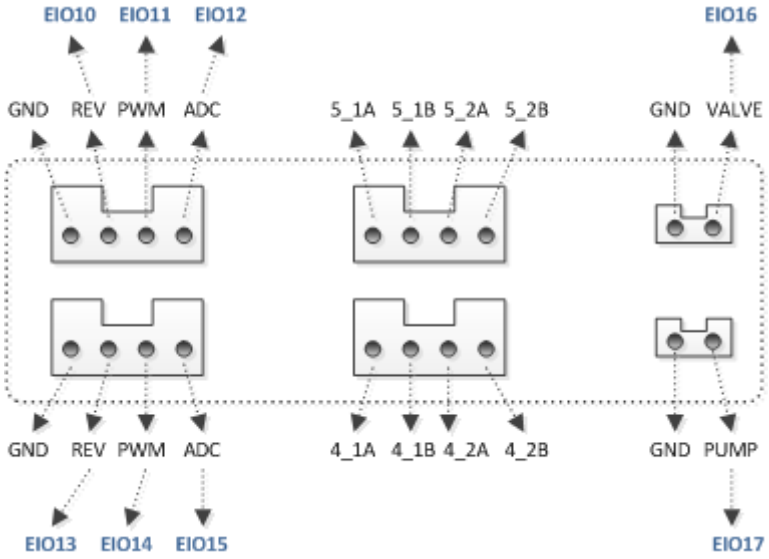
附图 14 UART 接口

附表 64 UART 接口 I/O 复用说明

I/O编址	电压	电平输出	PWM	电平输入	ADC
18	3.3V	√	-	√	-
19	3.3V	√	-	√	-
20	3.3V	√	-	√	-

外设接口 I/O 复用说明

底座外设接口如附图 15所示，其I/O复用说明如附表 65所示。



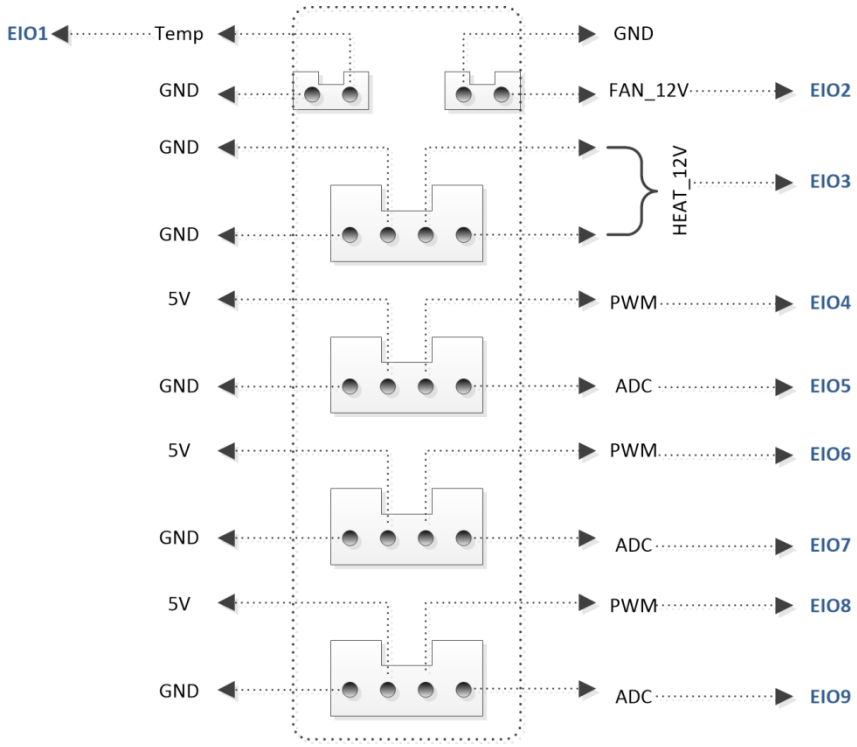
附图 15 底座外设接口

附表 65 外设接口 I/O 复用说明

I/O编址	电压	电平输出	PWM	电平输入	ADC
10	5V	√	-	-	-
11	3.3V	√	√	√	-
12	3.3V	√	-	√	√
13	5V	√	-	-	-
14	3.3V	√	√	√	-
15	3.3V	√	-	√	√
16	12V	√	-	-	-
17	12V	√	-	-	-

小臂 I/O 接口复用说明

小臂外设接口如附图 16所示，其I/O复用说明如附表 66所示。



附图 16 小臂外设接口

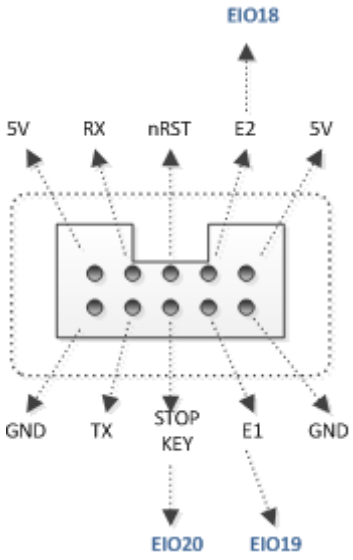
附表 66 外设接口 I/O 复用说明

I/O编址	电压	电平输出	PWM	电平输入	ADC
1	3.3V	-	-	√	-
2	12V	√	-	-	-
3	12V	√	-	-	-
4	3.3V	√	√	√	-
5	3.3V	√	-	√	√
6	3.3V	√	√	√	-
7	3.3V	√	-	√	√
8	3.3V	√	√	√	-
9	3.3V	√	-	√	√

附录F V2 版本机械臂 I/O 接口复用说明

底座 UART 接口 I/O 复用说明

UART接口如附图 17所示，其I/O复用说明如附表 67所示。



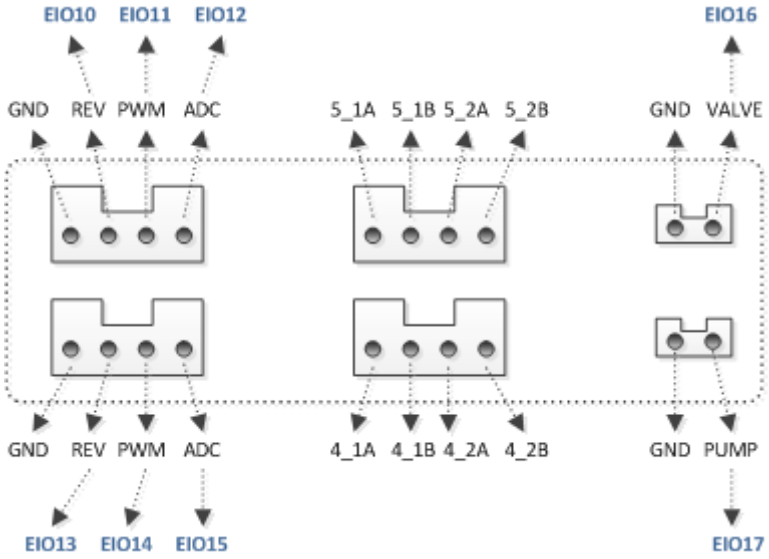
附图 17 UART 接口

附表 67 UART 接口 I/O 复用说明

I/O编址	电压	电平输出	PWM	电平输入	ADC
18	3.3V	√	-	-	-
19	3.3V	-	-	√	-
20	3.3V	-	-	√	-

外设接口 I/O 复用说明

底座外设接口如附图 18所示，其I/O复用说明如附表 65所示。



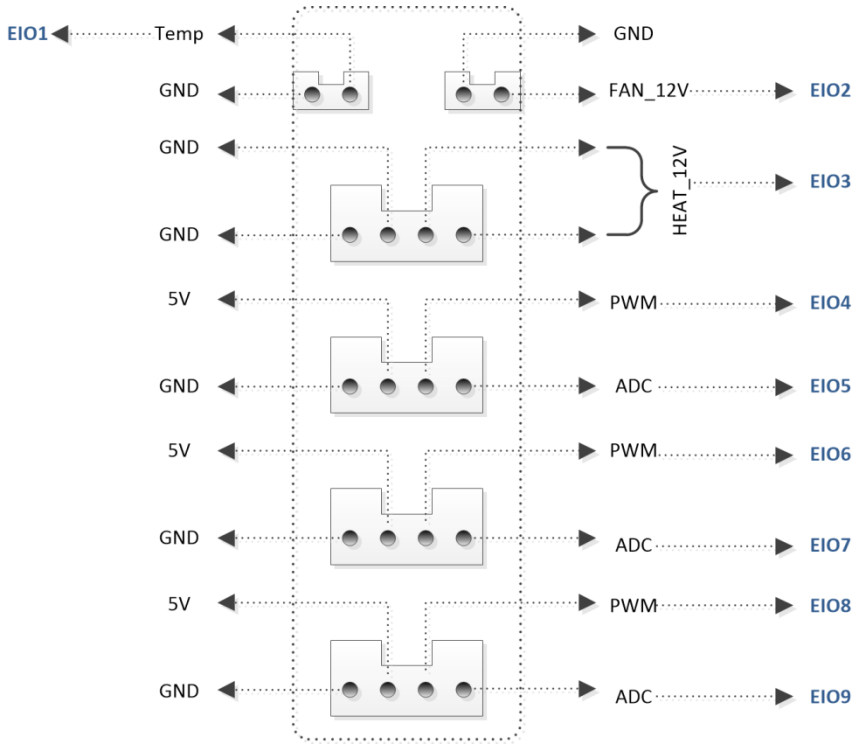
附图 18 底座外设接口

附表 68 外设接口 I/O 复用说明

I/O编址	电压	电平输出	PWM	电平输入	ADC
10	5V	√	-	-	-
11	3.3V	√	√	-	-
12	3.3V	-	-	√	-
13	5V	√	-	-	-
14	3.3V	√	√	√	-
15	3.3V	√	-	√	√
16	12V	√	-	-	-
17	12V	√	-	-	-

小臂 I/O 接口复用说明

小臂外设接口如附图 19所示，其I/O复用说明如附表 69所示。



附图 19 小臂外设接口

附表 69 外设接口 I/O 复用说明

I/O编址	电压	电平输出	PWM	电平输入	ADC
1	3.3V	-	-	√	-
2	12V	√	-	-	-
3	12V	√	-	-	-
4	3.3V	√	√	-	-
5	3.3V	-	-	√	-
6	3.3V	√	√	-	-
7	3.3V	-	-	√	-
8	3.3V	√	√	-	-
9	3.3V	-	-	√	√