Arduino Kit Demo Description

Issue: V2.1

Date: 2019-12-05

Shenzhen Yuejiang Technology Co., Ltd

Disclaimer

To the maximum extent permitted by applicable law, the products described (including its hardware, software and firmware, etc.) in this document are provided **AS IS**, which may have flaws, errors or faults. Yuejiang makes no warranties of any kind, express or implied, including but not limited to, merchantability, satisfaction of quality, fitness for a particular purpose and non-infringement of third party rights. In no event will Yuejiang be liable for any special, incidental, consequential or indirect damages resulting from the use of our products and documents.

Before using our product, please thoroughly read and understand the contents of this document and related technical documents that are published online, to ensure that the robotic arm is used on the premise of fully understanding the robotic arm and related knowledge. Please use this document with technical guidance from professionals. Even if follow this document or any other related instructions, Damages or losses will be happen in the using process, Dobot shall not be considered as a guarantee regarding to all security information contained in this document.

The user has the responsibility to make sure following the relevant practical laws and regulations of the country, in order that there is no significant danger in the use of the robotic arm.

Shenzhen Yuejiang Technology Co., Ltd

Address: Address: Floor 9-10, Building 2, Chongwen Garden, Nanshan iPark, Liuxian Blvd, Nanshan District, Shenzhen, Guangdong Province, China

Website: www.dobot.cc

Preface

Purpose

This document describes the Demo environment setup of the Arduino Kit, the Demo module connection, and key code descriptions for entry-starting creators and non-electronics enthusiasts.

Intended Audience

This document is intended for:

- Customer Engineer
- Sales Engineer
- Installation and Commissioning Engineer
- Technical Support Engineer

Change History

Date	Change Description
2019/12/05	Update our address
2019/06/18	The first release

Symbol Conventions

The symbols that may be founded in this document are defined as follows.

Symbol	Description
⚠DANGER	Indicates a hazard with a high level of risk which, if not avoided, could result in death or serious injury
⚠WARNING	Indicates a hazard with a medium level or low level of risk which, if not avoided, could result in minor or moderate injury, robotic arm damage
⚠NOTICE	Indicates a potentially hazardous situation which, if not avoided, can result in robotic arm damage, data loss, or unanticipated result
🕮NOTE	Provides additional information to emphasize or supplement important points in the main text

Contents

1. Introduction

1.1 Overview

Arduino kit includes Arduino Mega2560 controller board, LED indicators, buttons, joystick, Grove speech recognizer and Pixy vision sensor, of which the demos are based on the Arduino Mega2560 controller board and developed by Dobot. This document describes the connection of each module, the realization processes with these demos, making the makers and the non-electronic professional electronic enthusiasts get started quickly and inspire their creative thinking.

1.2 Software Environment

Arduino is an open-source platform used for building electronics projects, consisting Arduino IDE and its core libraries. Please download Arduino **1.8.2**, of which the path is https://www.arduino.cc/en/Main/OldSoftwareReleases#previous.

After Installing the Arduino IDE, you need to configure it, the steps are shown as follows.

Step 1 Decompress the Arduino Demo package, and add Dobot, Pixy2 and SmartKit files in the **arduino kit\libraries** directory to the **Arduino IDE installation\arduino-1.8.2\libraries** or to the **C:\Users\Administrator\Documents\Arduino\libraries** directory.

If the operation is successful, you can view the corresponding libraries on the **Sketch > Include Library** menu of Arduino page after you launch the Arduino IDE,, as shown in Figure 1.1.

🕮NOTE

In Arduino demos, the Dobot Magician, Pixy vision sensor and SmartKit (LED indicators, buttons, joystick, Grove speech recognizer) may be used, so you need to add their APIs to load them into Arduino.

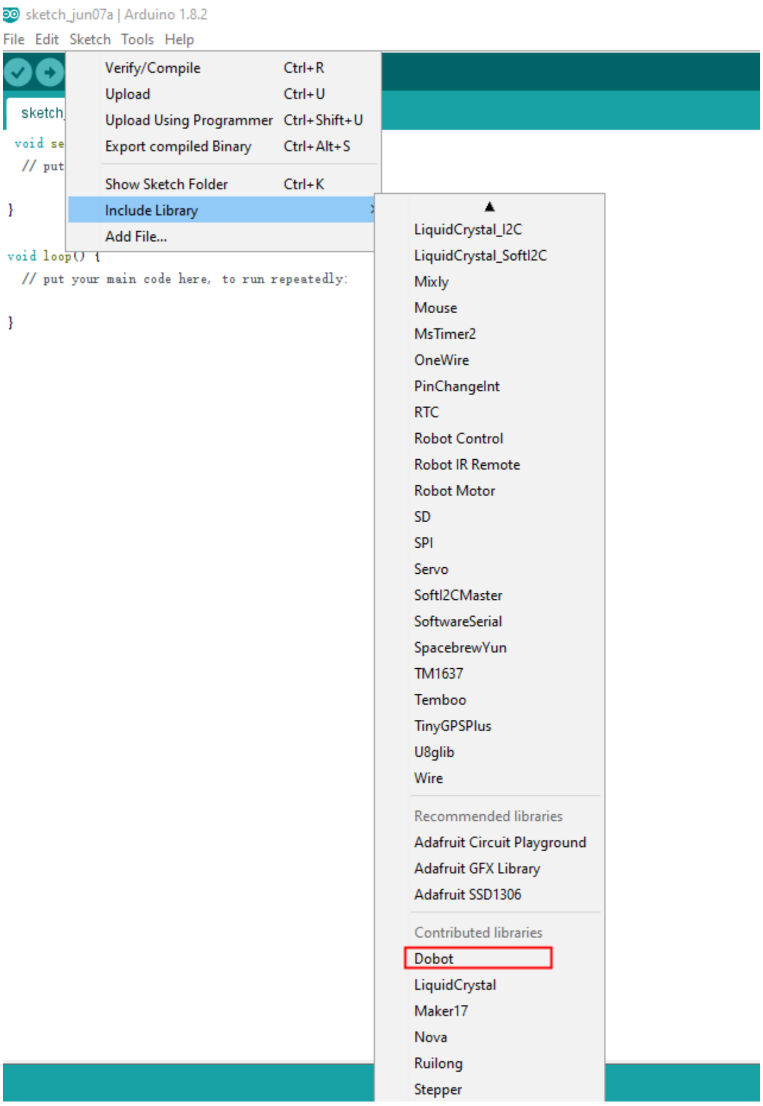


Figure 1.1 Add Dobot library

Step 2 Launch the Arduino IDE.

Step 3 Select **Arduino/Genuino Mega or Mega 2560** on the **Tools > Board** menu, select **ATmega2560 (Mega 2560)** on the **Tools > Processor** menu, select the right serial port on the **Tools > Port** menu.

⚠NOTICE

In Arduino demos, if the Dobot Magician, SmarkKit or Pixy vision sensor is used, please open the corresponding demo with Arduino IDE and select the corresponding library on the **Sketch > Include Library** menu. If the speech recognizer is used, please select **SoftwareSerial** on the **Sketch > Include Library** menu to build software serial communication.

2. FlickerLED Demo

2.1 Introduction

This Demo uses the Arduino to turn on and off the LED indicator.

2.2 Hardware Connection

The LED indicator and Arduino Mega2560 controller board are required in this demo. Figure 2.1 shows its connection process.

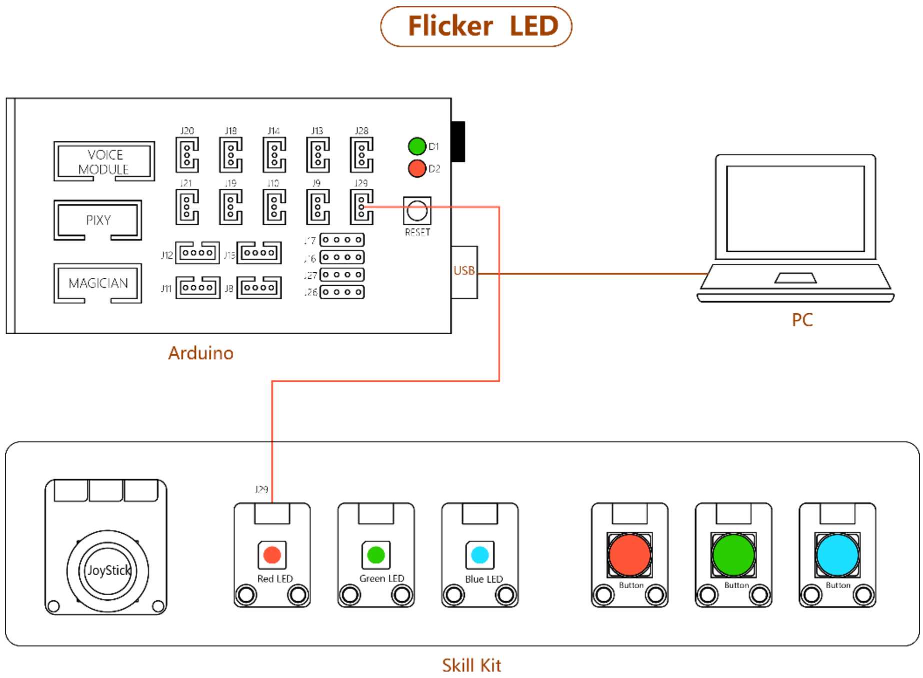


Figure 2.1 FlickerLED Connection

NOTE

If you connect the skill kit to other interfaces on the Arduino Mega2560, you also need to modify the right interfaces that the skill kit connects to in the **SmartKit.h** file.

Program 2.1 Define the interfaces that the skill kit connects to

```
#define JOYSTICK_XPIN     7     // Interface that the X-axis of the JoyStick connects to
#define JOYSTICK_YPIN     6     // Interface that the Y-axis of the JoyStick connects to
#define JOYSTICK_ZPIN     A5    // Interface that the Z-axis of the JoyStick connects to


#define LED_REDPIN        9     // Interface that the red LED indicator connects to
#define LED_GREENPIN      A1    // Interface that the green LED indicator connects to
#define LED_BLUEPIN       A3    // Interface that the blue LED indicator connects to
```

```
#define BUTTON_REDPIN      A0     // Interface that the red button connects to
#define BUTTON_GREENPIN A2     // Interface that the green button connects to
#define BUTTON_BLUEPIN     A4     // Interface that the blue button connects to
```

2.3 Realization Process

Figure 2.2 shows its realization process.

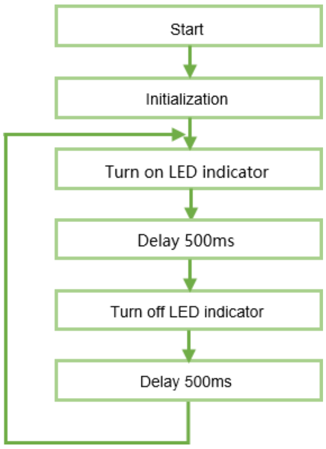


Figure 2.2 Realization process

2.4 Critical Code Description

Before debugging this demo, please select **SmartKit** library on the **Sketch** > **Include Library** menu.

（1） Initialization.

Program 2.2 Initialization

```
void setup(){
  SmartKit_Init();                         //Initialization
}
```

（2） Set the pin to HIGH or LOW to control the LED indicator.

Program 2.3 Set High/Low level

```
void loop(){
    SmartKit_LedTurn(RED, ON);              //Turn on the red LED indicator
    delay(500);
    SmartKit_LedTurn(RED, OFF);             // Turn off the red LED indicator
    delay(500);
}
```

3. AlarmLED Demo

3.1 Introduction

This Demo uses the Arduino to turn on and off three different colored LED indicators. Only one LED indicator is on at a time.

3.2 Hardware Connection

The three LED indicators and Arduino Mega2560 controller board are required in this demo. Figure 3.1 shows its connection process.

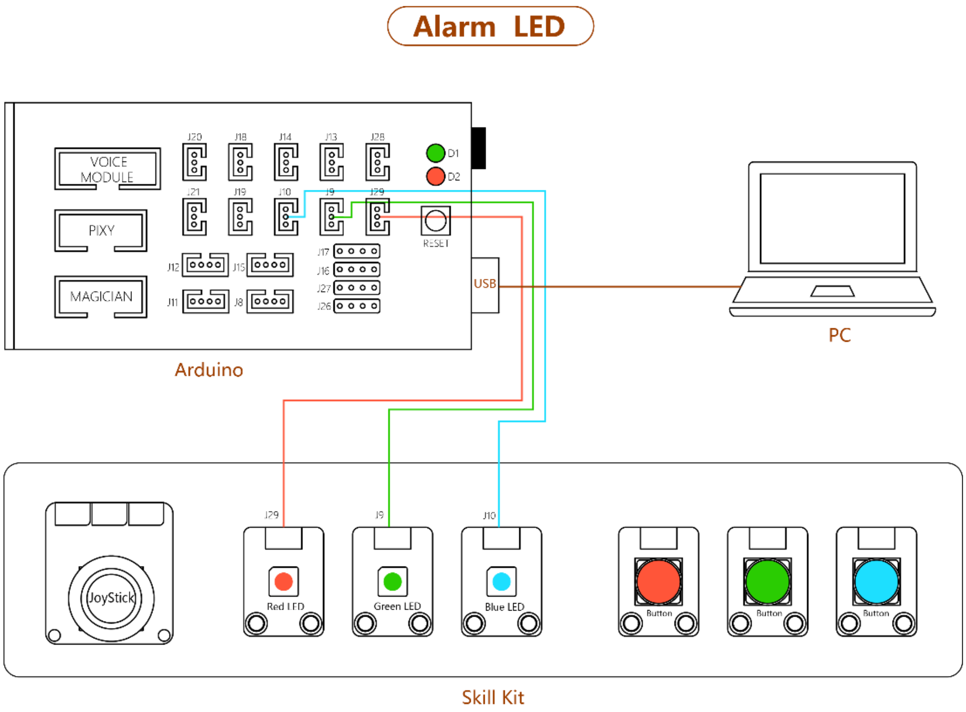


Figure 3.1 AlarmLED Connection

📖NOTE

If you connect the skill kit to other interfaces on the Arduino Mega2560, you also need to modify the right interfaces that the skill kit connects to in the **SmartKit.h** file.

Program 3.1 Define the interfaces that the skill kit connects to

```
#define JOYSTICK_XPIN     7     // Interface that the X-axis of the JoyStick connects to
#define JOYSTICK_YPIN     6     // Interface that the Y-axis of the JoyStick connects to
#define JOYSTICK_ZPIN     A5    // Interface that the Z-axis of the JoyStick connects to

#define LED_REDPIN        9     // Interface that the red LED indicator connects to
#define LED_GREENPIN      A1    // Interface that the green LED indicator connects to
#define LED_BLUEPIN       A3    // Interface that the blue LED indicator connects to
```

```
#define BUTTON_REDPIN     A0    // Interface that the red button connects to
#define BUTTON_GREENPIN A2    // Interface that the green button connects to
#define BUTTON_BLUEPIN    A4    // Interface that the blue button connects to
```

3.3 Realization Process

Figure 3.2 shows its realization process.

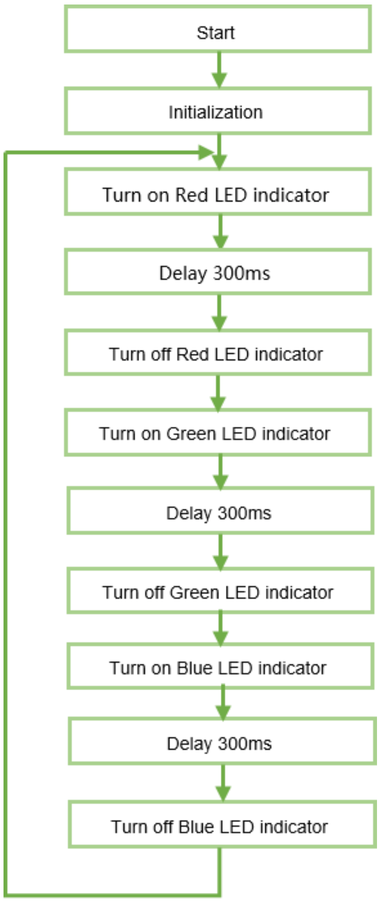


Figure 3.2 Realization process

3.4 Critical Code Description

Before debugging this demo, please select **SmartKit** library on the **Sketch** > **Include Library** menu.

（1） Initialization.

Program 3.2 Initialization

```
void setup(){
  SmartKit_Init();                    //Initialization
}
```

（2） Set the pins to HIGH or LOW to control the three LED indicators.

Program 3.3 Set High/Low level

```
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 Introduction

This demo uses the joystick to control the brightness of the LED indicator.

4.2 Hardware Connection

The LED indicator, joystick and Arduino Mega2560 are required in this demo. Figure 4.1 shows its connection process.

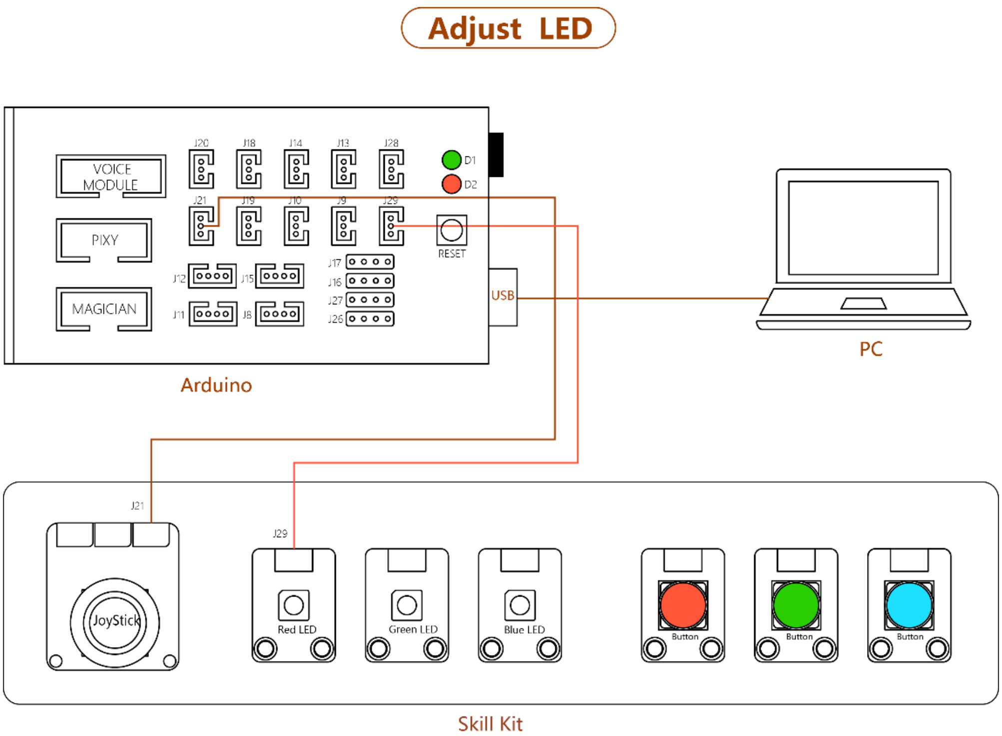


Figure 4.1 AdjustLED Connection

📖NOTE

If you connect the skill kit to other interfaces on the Arduino Mega2560, you also need to modify the right interfaces that the skill kit connects to in the **SmartKit.h** file.

Program 4.1 Define the interfaces that the skill kit connects to

```
#define JOYSTICK_XPIN     7     // Interface that the X-axis of the JoyStick connects to
#define JOYSTICK_YPIN     6     // Interface that the Y-axis of the JoyStick connects to
#define JOYSTICK_ZPIN     A5    // Interface that the Z-axis of the JoyStick connects to


#define LED_REDPIN        9     // Interface that the red LED indicator connects to
#define LED_GREENPIN      A1    // Interface that the green LED indicator connects to
#define LED_BLUEPIN       A3    // Interface that the blue LED indicator connects to
```

```
#define BUTTON_REDPIN     A0     // Interface that the red button connects to
#define BUTTON_GREENPIN A2      // Interface that the green button connects to
#define BUTTON_BLUEPIN    A4     // Interface that the blue button connects to
```

4.3 Realization Process

This demo controls the brightness of the LED indicator by moving the joystick along X-axis. Figure 4.2 shows its realization process.

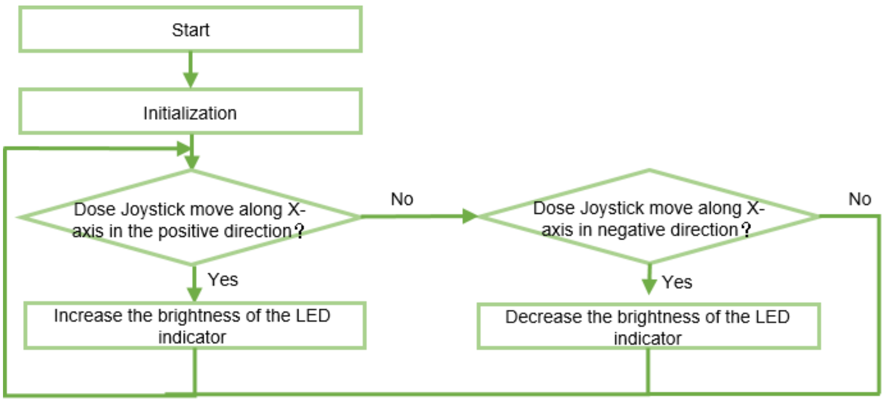


Figure 4.2 Realization process

4.4 Critical Code Description

Before debugging this demo, please select **SmartKit** library on the **Sketch** > **Include Library** menu.

(1) Initialization.

Program 4.2 Initialization

```
void setup(){
  SmartKit_Init();                    //Initialization
}
```

(2) Define the brightness variation frequency of LED indicator.

📖NOTE

When moving joystick along X-axis or Y-axis, the analog values change from 0 to 1023, as shown in Figure 4.3. The homing position of the joystick is at (x,y: 512,508).

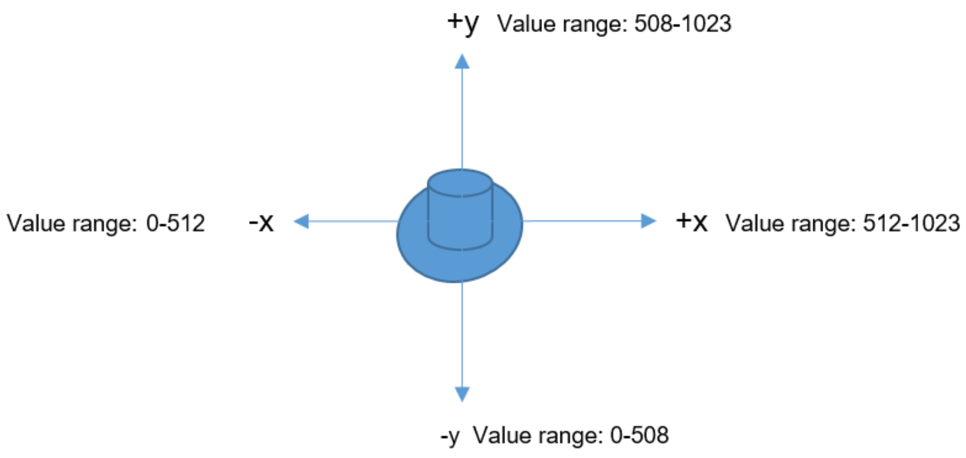


Figure 4.3 Analog value range

Program 4.3 Define the brightness variation of LED indicator

```
double xValueTOAnalogScale = 1023/255;          // Calculate the joystick value and transform to the
                                                   output analog value
value = SmartKit_JoyStickReadXYValue(AXISX);     // Get the analog value of the X axis of the joystick
```

（3） Adjust the brightness of the LED indicator over joystick.

Program 4.4 Adjust the brightness of the LED indicator over joystick

```
value = value / xValueTOAnalogScale;
analogWrite(LED_REDPIN, value);                  //Adjust the brightness of the LED indicator
```

5. Button Demo

5.1 Introduction

This demo uses three different colored buttons to turn on and off the corresponding colored LED indicators respectively.

5.2 Hardware Connection

The three buttons, three LED indicators and Arduino Mega250 are required in this demo. Figure 5.1 shows its connection process.

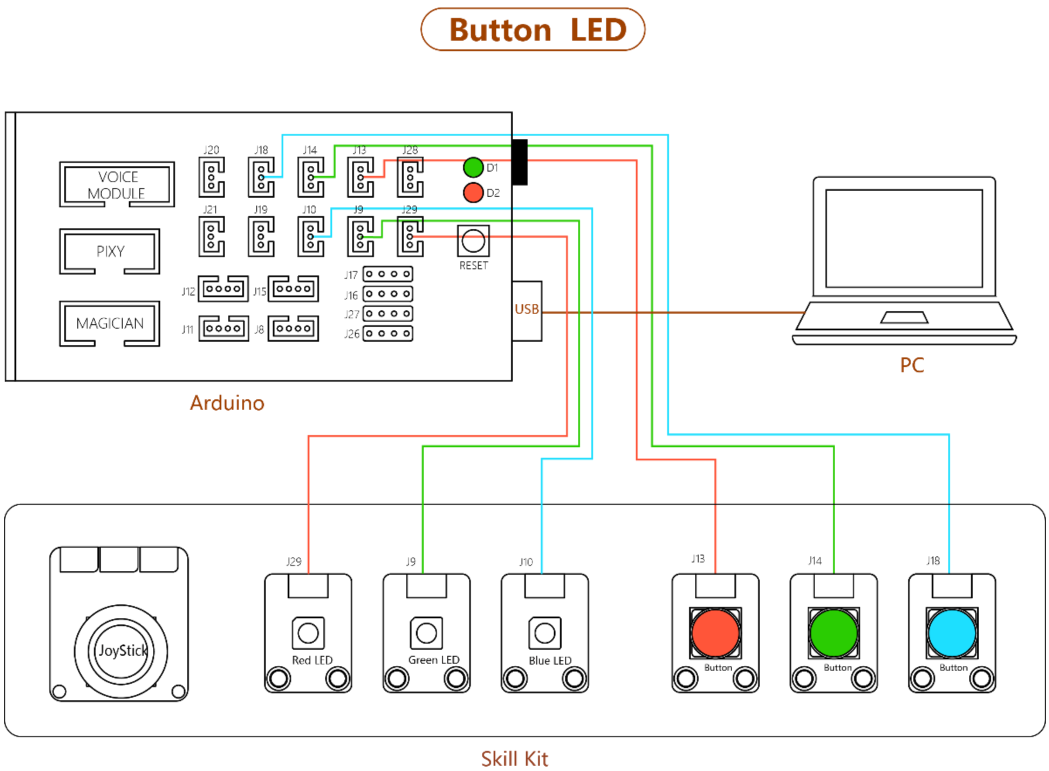


Figure 5.1 ButtonLED Connection

📖NOTE

If you connect the skill kit to other interfaces on the Arduino Mega2560, you also need to modify the right interfaces that the skill kit connects to in the **SmartKit.h** file.

Program 5.1 Define the interfaces that the skill kit connects to

```
#define JOYSTICK_XPIN     7    // Interface that the X-axis of the JoyStick connects to
#define JOYSTICK_YPIN     6    // Interface that the Y-axis of the JoyStick connects to
#define JOYSTICK_ZPIN     A5   // Interface that the Z-axis of the JoyStick connects to

#define LED_REDPIN        9    // Interface that the red LED indicator connects to
#define LED_GREENPIN      A1   // Interface that the green LED indicator connects to
```

```
#define LED_BLUEPIN        A3     // Interface that the blue LED indicator connects to


#define BUTTON_REDPIN      A0     // Interface that the red button connects to
#define BUTTON_GREENPIN A2     // Interface that the green button connects to
#define BUTTON_BLUEPIN    A4     // Interface that the blue button connects to
```

5.3 Realization Process

Figure 5.2 shows its realization process.

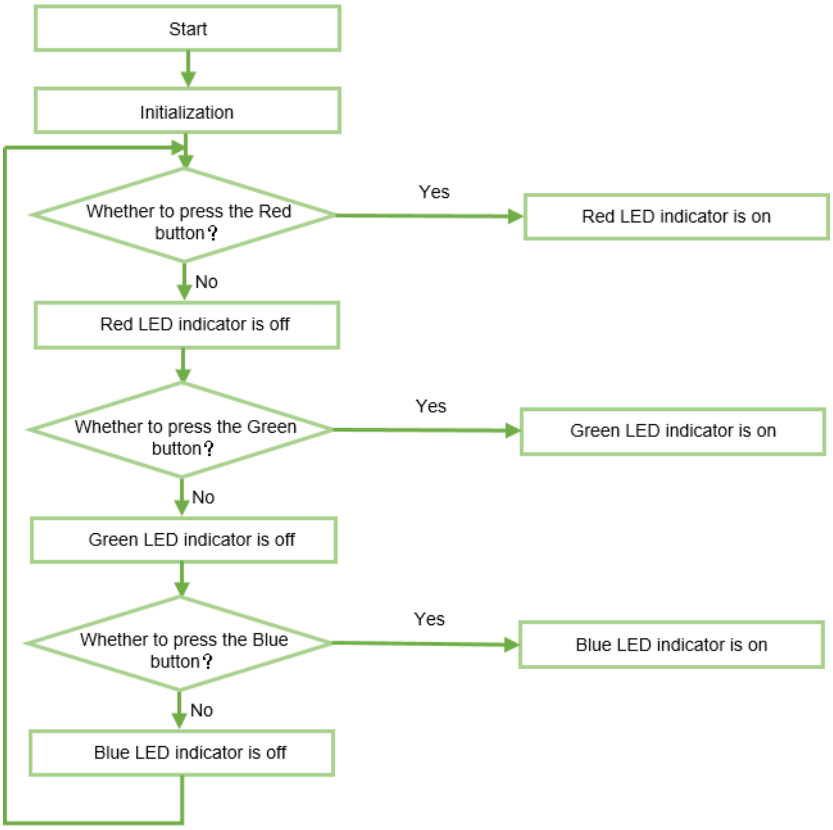


Figure 5.2 Realization process

5.4 Critical Code Description

Before debugging this demo, please select **SmartKit** library on the **Sketch** > **Include Library** menu.

(1) Initialization.

Program 5.2 Initialization

```
void setup(){
  SmartKit_Init();                       //Initialization
}
```

（2） Use the buttons to turn on and off the LED indicators.

Program 5.3 Use buttons to turn on and off the LED indicators

```
    if (SmartKit_ButtonCheckState(color) == TRUE)        // Check the button status
    {
        SmartKit_LedTurn(color, ON);                     //Turn on the LED indicator
    }
    else
    {
        SmartKit_LedTurn(color, OFF);                    //Turn off the LED indicator
    }
……
…..
```

6. SeedVoiceLED Demo

6.1 Introduction

The demo uses Grove speech recognizer to turn on and off three different colored LED indicators (Red, Green, and Blue).

6.2 Hardware Connection

The speech recognizer, three LED indicators and Arduino Mega2560 are required in this demo. Figure 6.1 shows its realization process.

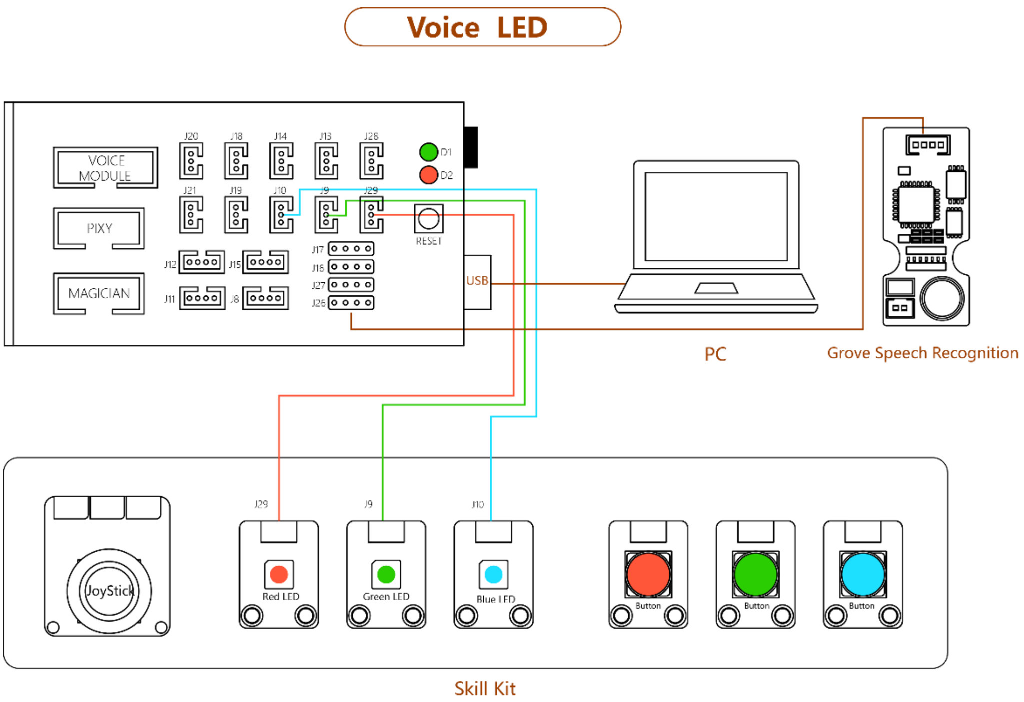


Figure 6.1 SeedVoiceLED Connection

NOTE

If you connect the skill kit to other interfaces on the Arduino Mega2560, you also need to modify the right interfaces that the skill kit connects to in the **SmartKit.h** file.

Program 6.1 Define the interfaces that the skill kit connects to

```
#define JOYSTICK_XPIN     7     // Interface that the X-axis of the JoyStick connects to
#define JOYSTICK_YPIN     6     // Interface that the Y-axis of the JoyStick connects to
#define JOYSTICK_ZPIN     A5    // Interface that the Z-axis of the JoyStick connects to

#define LED_REDPIN        9     // Interface that the red LED indicator connects to
#define LED_GREENPIN      A1    // Interface that the green LED indicator connects to
#define LED_BLUEPIN       A3    // Interface that the blue LED indicator connects to
```



```
#define BUTTON_REDPIN      A0      // Interface that the red button connects to
#define BUTTON_GREENPIN A2      // Interface that the green button connects to
#define BUTTON_BLUEPIN     A4      // Interface that the blue button connects to
```

6.3 Realization Process

Figure 6.2 shows its realization process.

NOTE

Please speak out the command **Hicell** to wake up the Grove speech recognizer before using it. If successful, the LED on the speech recognizer will turn red. Then, you can speak out the command word. If the command word is detected, the LED will turn blue.

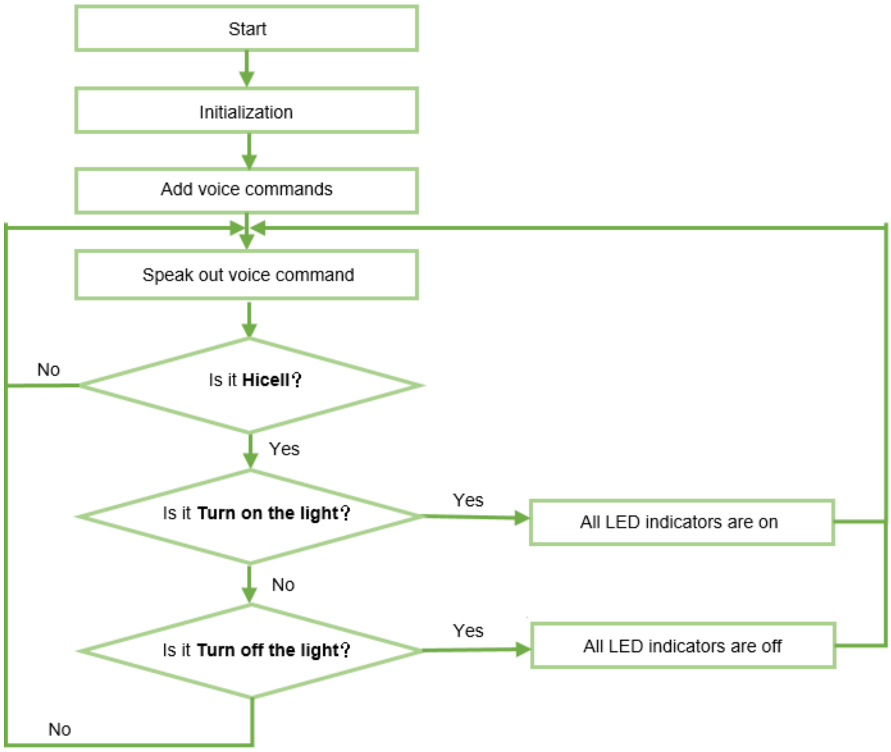


Figure 6.2 Realization process

6.4 Critical Code Description

Before debugging this demo, please select **SoftwareSerial** and **SmartKit** library on the **Sketch** > **Include Library** menu.

（1） Initialization.

Program 6.2 Initialization

```
void setup()
{
```

```
        Serial.begin(115200);
        SmartKit_Init();
        SmartKit_VoiceENGStart();
        Serial.println("Start...");
}
```

（2） Control the LED indicator by voice commands.

Program 6.3 Control the LED indicator by voice commands

```
void loop()
{
        if(SmartKit_VoiceENGVoiceCheck(1) == TRUE)
        {
                SmartKit_LedTurn(RED,ON);
                SmartKit_LedTurn(BLUE,ON);
                SmartKit_LedTurn(GREEN,ON);
                Serial.println(voiceBuffer[0]);
         }
         else if(SmartKit_VoiceENGVoiceCheck(2) == TRUE)
         {
              SmartKit_LedTurn(RED,OFF);
              SmartKit_LedTurn(BLUE,OFF);
              SmartKit_LedTurn(GREEN,OFF);
              Serial.println(voiceBuffer[1]);
         }
}
```

⚠NOTICE

Grove speech recognizer only supports 22 voice commands without user-defined. The commands with values are as shown in Program 6.4.

Program 6.4 Command description

```
const char *voiceBuffer[] =
{
        "Turn on the light",                                        //return 1
        "Turn off the light",                                       //return 2
        "Play music",                                               //return 3
        "Pause",                                                    //return 4
```



```
    "Next",                                  //return 5
    "Previous",                              //return 6
    "Up",                                    //return 7
    "Down",                                  //return 8
    "Turn on the TV",                        //return 9
    "Turn off the TV",                       //return 10
    "Increase temperature",                  //return 11
    "Decrease temperature",                  //return 12
    "What's the time",                       //return 13
    "Open the door",                         //return 14
    "Close the door",                        //return 15
    "Left",                                  //return 16
    "Right",                                 //return 17
    "Stop",                                  //return 18
    "Start",                                 //return 19
    "Mode 1",                                //return 20
    "Mode 2",                                //return 21
    "Go",                                    //return 22
};
```


7. MoveBlock Demo

7.1 Introduction

The demo uses Arduino to control Dobot Magician for picking and placing cubes.

7.2 Hardware Connection

Arduino Mega2560, Dobot Magician and suction cup kit are required in this demo. Figure 7.1 shows its connection process.

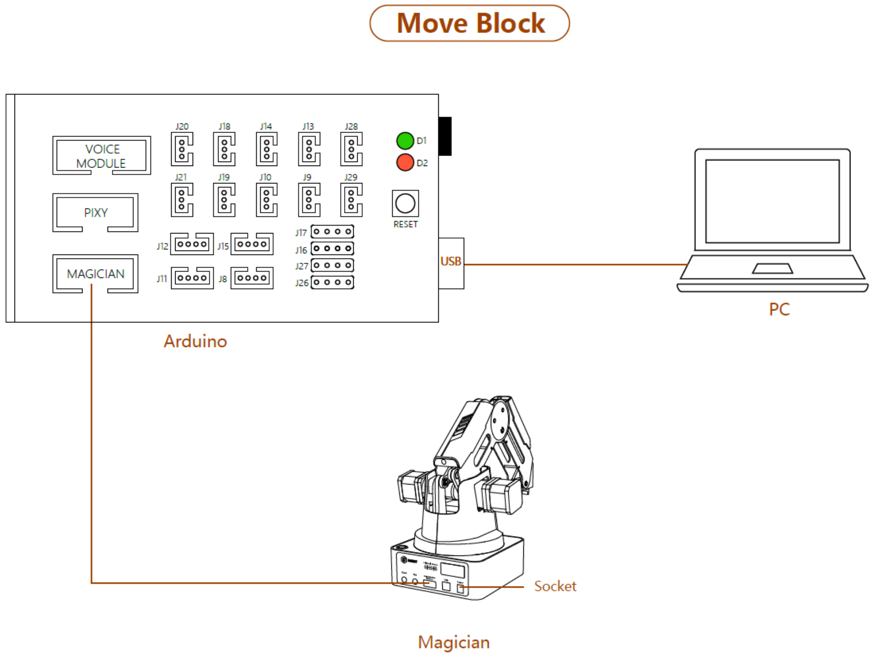


Figure 7.1 MoveBlock Connection

For details how to connect Dobot Magician and suction cup kit, please see *Appendix B Installing Suction Cup Kit.*

📖NOTE

If you connect the skill kit to other interfaces on the Arduino Mega2560, you also need to modify the right interfaces that the skill kit connects to in the **SmartKit.h** file.

Program 7.1 Define the interfaces that the skill kit connects to

```
#define JOYSTICK_XPIN     7    // Interface that the X-axis of the JoyStick connects to
#define JOYSTICK_YPIN     6    // Interface that the Y-axis of the JoyStick connects to
#define JOYSTICK_ZPIN     A5   // Interface that the Z-axis of the JoyStick connects to


#define LED_REDPIN        9    // Interface that the red LED indicator connects to
#define LED_GREENPIN      A1   // Interface that the green LED indicator connects to
#define LED_BLUEPIN       A3   // Interface that the blue LED indicator connects to
```

```
#define BUTTON_REDPIN     A0    // Interface that the red button connects to
#define BUTTON_GREENPIN A2    // Interface that the green button connects to
#define BUTTON_BLUEPIN    A4    // Interface that the blue button connects to
```

7.3 Realization Process

If the cube is moved from point A to point B and then from point B back to point A with multiple times. Figure 7.2 shows its realization process.

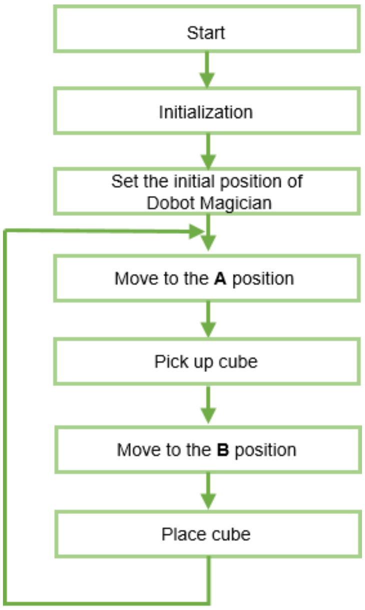


Figure 7.2 Realization process

7.4 Critical Code Desrciption

Dobot Magician communicates with Arduino over UART interface (10 PIN) on the base of the Dobot Magician, using the Dobot protocol. We have provided **Dobot** library which encapsulates part of the Dobot Magician APIs, being called directly to control Dobot Magician. Before debugging this Demo, please select the **Dobot** library on the **Sketch** > **Include Library** menu.

⚠NOTICE

Please long press the **Key** button on the back of base of Dobot Magician to operate homing before debugging this Demo.

Before debugging this Demo, please connect Dobot Magician and DobotStudio and operate homing. And then press the **Unlock** key on the Forearm and drag Dobot Magician to move to the positions where the cube is to be placed (A point and B point), then record their Cartesian coordinates from the **operation panel** pane of DobotStudio page to write in this demo for picking and placing cube.

Figure 7.3 Obtain Cartesian coordinates

（1） Define the coordinates of point A and point B

The coordinates can be obtained from the **Operation Panel** of DobotStudio page.

Program 7.2 Define the coordinates of point A and point B

```
//Coordinates of A point
#define block_positio_X 263                  //X-coordinate of A point
#define block_position_Y 3                   //Y-coordinate of A point
#define block_position_Z -40                 //Z-coordinate of A point
#define block_position_R 0                   //R-coordinate of A point

//Coordinates of B point
```

```
#define Des_position_X 207                                  //X-coordinate of B point
#define Des_position_Y -171                                 //Y-coordinate of B point
#define Des_position_Z -46                                  //Z-coordinate of B point
#define Des_position_R 0                                    // R-coordinate of B point
```

（2） Dobot Magician moves from point A to point B with multiple times.

Program 7.3 Dobot Magician moves from point A to point B with multiple times

```
while(count > 0)
{
  Dobot_SetPTPCmdEx(JUMP_XYZ,block_position_X, block_position_Y,
                    block_position_Z, block_position_R);            //Move to point A
  Dobot_SetEndEffectorSuctionCupEx(true);                           //Turn on air pump to pick up cube
  Dobot_SetPTPCmdEx((JUMP_XYZ,block_position_X, block_position_Y,-4, block_position_R);
                                                                    //Lift a certain height
  Dobot_SetPTPCmdEx(JUMP_XYZ, Des_position_X, Des_position_Y,
                    Des_position_Z, Des_position_R);                //Move to point B
  Dobot_SetEndEffectorSuctionCupEx(false);                          //Turn off air pump to place cube
  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);                      //Move back to the initial position
  Dobot_SetPTPCmdEx(JUMP_XYZ, Des_position_X, Des_position_Y,
                    Des_position_Z, Des_position_R);                //Move to point B
  Dobot_SetEndEffectorSuctionCupEx(true);                           //Turn on air pump to pick up cube
  Dobot_SetPTPCmdEx(MOVL_XYZ, Des_position_X, Des_position_Y, -10, Des_position_R);
                                                                    //Lift a certain height
  Dobot_SetPTPCmdEx(JUMP_XYZ,block_position_X, block_position_Y,
                    block_position_Z, block_position_R);            //Move to point A
  Dobot_SetEndEffectorSuctionCupEx(false);                          //Turn off air pump to place cube
  Dobot_SetPTPCmdEx(MOVJ_XYZ, 178, -4, 40, 0);                      //Move back to the initial position
  count--;
}
```

8. SeedVoiceDobot Demo

8.1 Introduction

This demo uses Grove speech recognizer to control the Dobot Magician movement and the air pump start-stop.

8.2 Hardware Connection

Grove speech recognizer, Dobot Magician, air pump and Arduino Mega2560 are required in this demo. The connection between Dobot Magician and Arduino is shown in Figure 8.1.

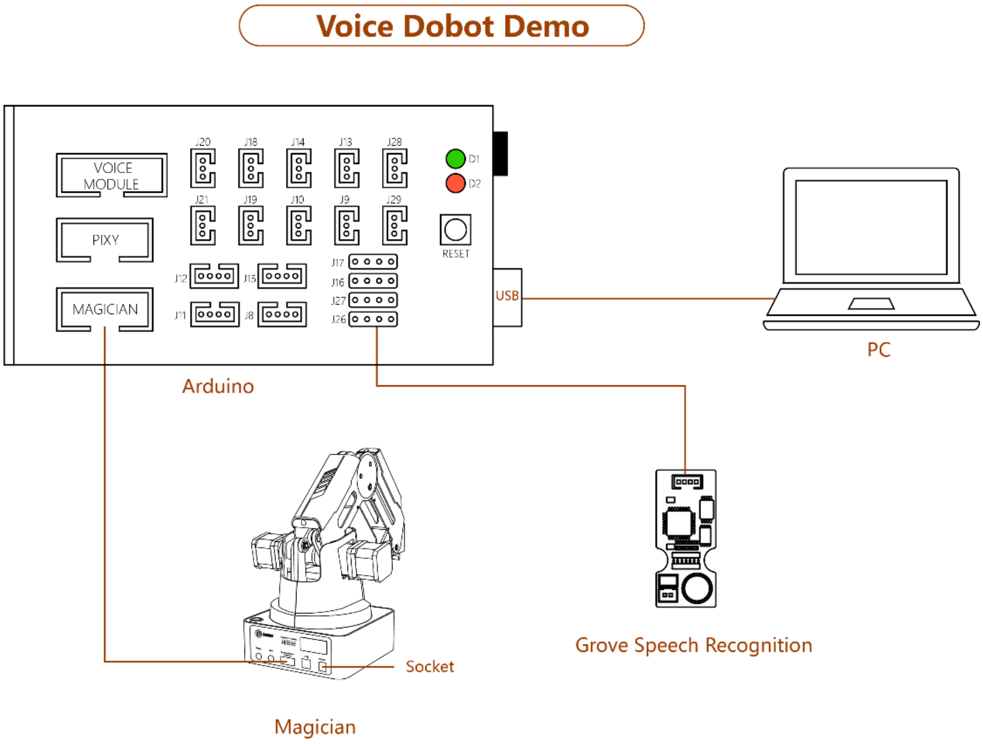


Figure 8.1 SeedVoiceDobot Connection

The connection between Dobot Magician and air pump is shown in Figure 8.2.

Figure 8.2 Air pump Connection

📖NOTE

If you connect the skill kit to other interfaces on the Arduino Mega2560, you also need to modify the right interfaces that the skill kit connects to in the **SmartKit.h** file.

Program 8.1 Define the interfaces that the skill kit connects to

```
#define JOYSTICK_XPIN      7     // Interface that the X-axis of the JoyStick connects to
#define JOYSTICK_YPIN      6     // Interface that the Y-axis of the JoyStick connects to
#define JOYSTICK_ZPIN      A5    // Interface that the Z-axis of the JoyStick connects to

#define LED_REDPIN         9     // Interface that the red LED indicator connects to
#define LED_GREENPIN       A1    // Interface that the green LED indicator connects to
#define LED_BLUEPIN        A3    // Interface that the blue LED indicator connects to

#define BUTTON_REDPIN      A0    // Interface that the red button connects to
#define BUTTON_GREENPIN A2       // Interface that the green button connects to
#define BUTTON_BLUEPIN     A4    // Interface that the blue button connects to
```

8.3 Realization Process

In this demo, we use Grove speech recognizer to control Dobot Magician and air pumps. Figure 8.3 shows the realization process.

📖NOTE

Please speak out the command **Hicell** to wake up the speech recognizer before using it. If successful, the LED on the speech recognizer will turn red. Then, you can speak out the command word. If the command word is detected, the LED will turn blue.

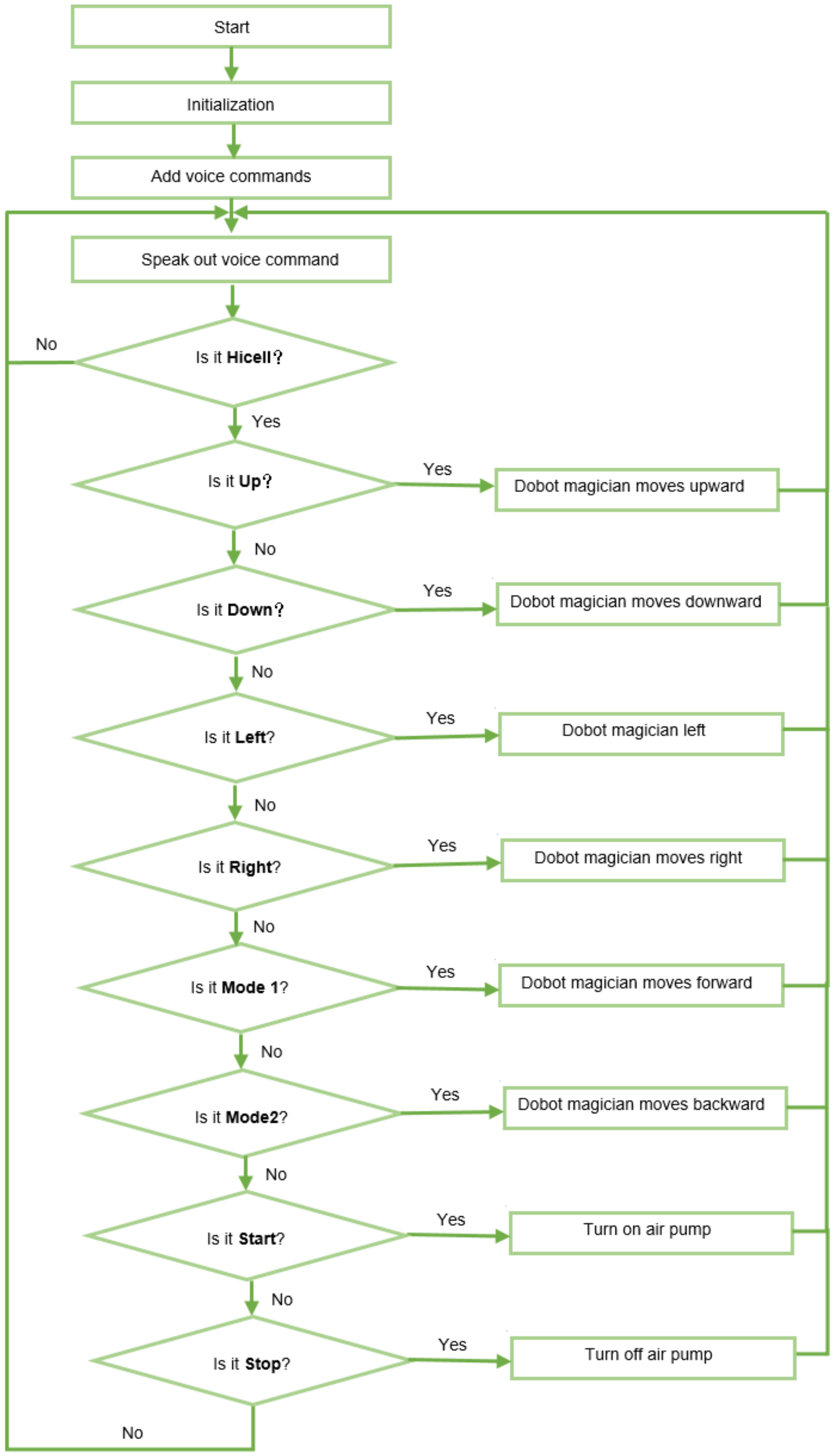


Figure 8.3 Realization process

8.4 Critical Code Description

This demo uses Grove speech recognizer to control Dobot Magician, the **Dobot** library, **SmartKit** library and **SoftwareSerial** library need to be called. Before debugging this Demo, please select **Dobot**, **SmartKit**, **and SoftwareSerial** on the **Sketch** > **Include Library** menu.

⚠NOTICE

Please long press the **Key** button on the back of base of Dobot Magician to operate homing before debugging this Demo.

（1） Initialization.

Program 8.2 Initialization

```
void setup()
{
    Serial.begin(115200);
    Dobot_Init();
    SmartKit_Init();
    SmartKit_VoiceENGStart();
    Serial.println("Start...");
}
```

（2） Move Dobot Magician by voice commands.

Program 8.3 Move Dobot Magician by voice commands

```
if(SmartKit_VoiceENGVoiceCheck(7) == TRUE)
    {
        Dobot_SetPTPCmd(MOVL_INC,0,0,30,0);                    //magician moves upward
        Serial.println(voiceBuffer[6]);
     }
     else if(SmartKit_VoiceENGVoiceCheck(8) == TRUE)
     {
        Dobot_SetPTPCmd(MOVL_INC,0,0,-30,0);                   //magician moves downward
        Serial.println(voiceBuffer[7]);
     }
……
……
```

⚠NOTICE

Grove speech recognizer only supports 22 voice commands without user-defined. The commands and their return values are as shown in Program 8.4.

Program 8.4 Command description

```
const char *voiceBuffer[] =
{
```

```
    "Turn on the light",                  //return 1
    "Turn off the light",                 //return 2
    "Play music",                         //return 3
    "Pause",                              //return 4
    "Next",                               //return 5
    "Previous",                           //return 6
    "Up",                                 //return 7
    "Down",                               //return 8
    "Turn on the TV",                     //return 9
    "Turn off the TV",                    //return 10
    "Increase temperature",               //return 11
    "Decrease temperature",               //return 12
    "What's the time",                    //return 13
    "Open the door",                      //return 14
    "Close the door",                     //return 15
    "Left",                               //return 16
    "Right",                              //return 17
    "Stop",                               //return 18
    "Start",                              //return 19
    "Mode 1",                             //return 20
    "Mode 2",                             //return 21
    "Go",                                 //return 22
};
```

9. JoyStick Demo

9.1 Introduction

This demo uses joystick and three buttons to control the Dobot Magician movement and the air pump start-stop.

9.2 Hardware Connection

Joystick module, button, Dobot Magician, air pump, and Arduino Mega2560 are required in this demo. The connections between them are shown in Figure 9.1.

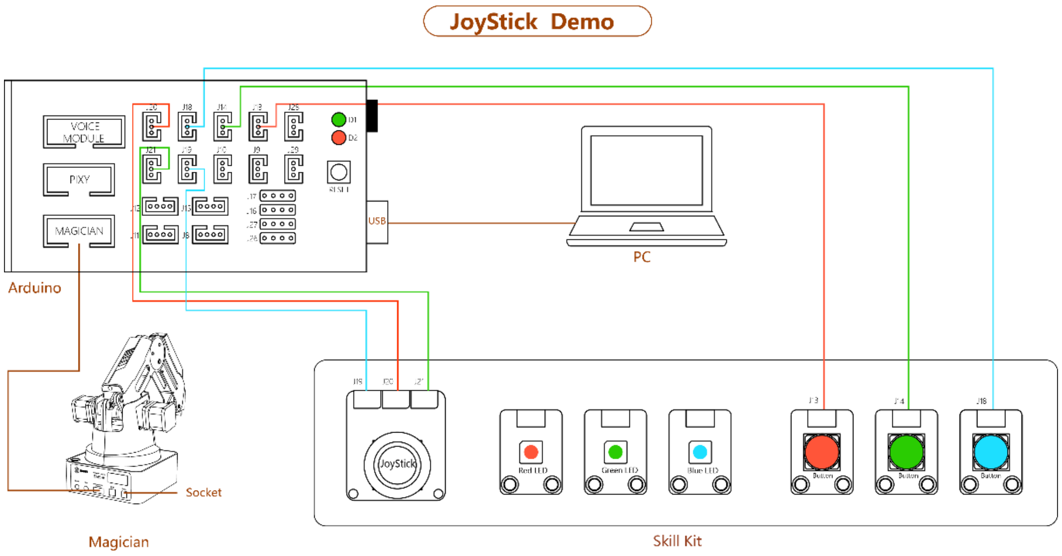


Figure 9.1 JoyStick Connection

The connection between Dobot Magician and air pump is shown in Figure 9.2.

Figure 9.2 Air pump Connection

📖NOTE

If you connect the skill kit to other interfaces on the Arduino Mega2560, you also need to modify the right interfaces that the skill kit connects to in the **SmartKit.h** file.

Program 9.1 Define the interfaces that the skill kit connects to

```
#define JOYSTICK_XPIN      7     // Interface that the X-axis of the JoyStick connects to
#define JOYSTICK_YPIN      6     // Interface that the Y-axis of the JoyStick connects to
#define JOYSTICK_ZPIN      A5    // Interface that the Z-axis of the JoyStick connects to

#define LED_REDPIN         9     // Interface that the red LED indicator connects to
#define LED_GREENPIN       A1    // Interface that the green LED indicator connects to
#define LED_BLUEPIN        A3    // Interface that the blue LED indicator connects to

#define BUTTON_REDPIN      A0    // Interface that the red button connects to
#define BUTTON_GREENPIN A2       // Interface that the green button connects to
#define BUTTON_BLUEPIN     A4    // Interface that the blue button connects to
```

9.3 Realization Process

In this demo, we move Dobot Magician forward, backward, left and right by moving joystick along X-axis or Y-axis, control the moving speed by joystick button, move Dobot Magician upward by red button, move Dobot Magician downward by green button and control the air pump start-stop by blue button. Figure 9.3 shows its realization process.

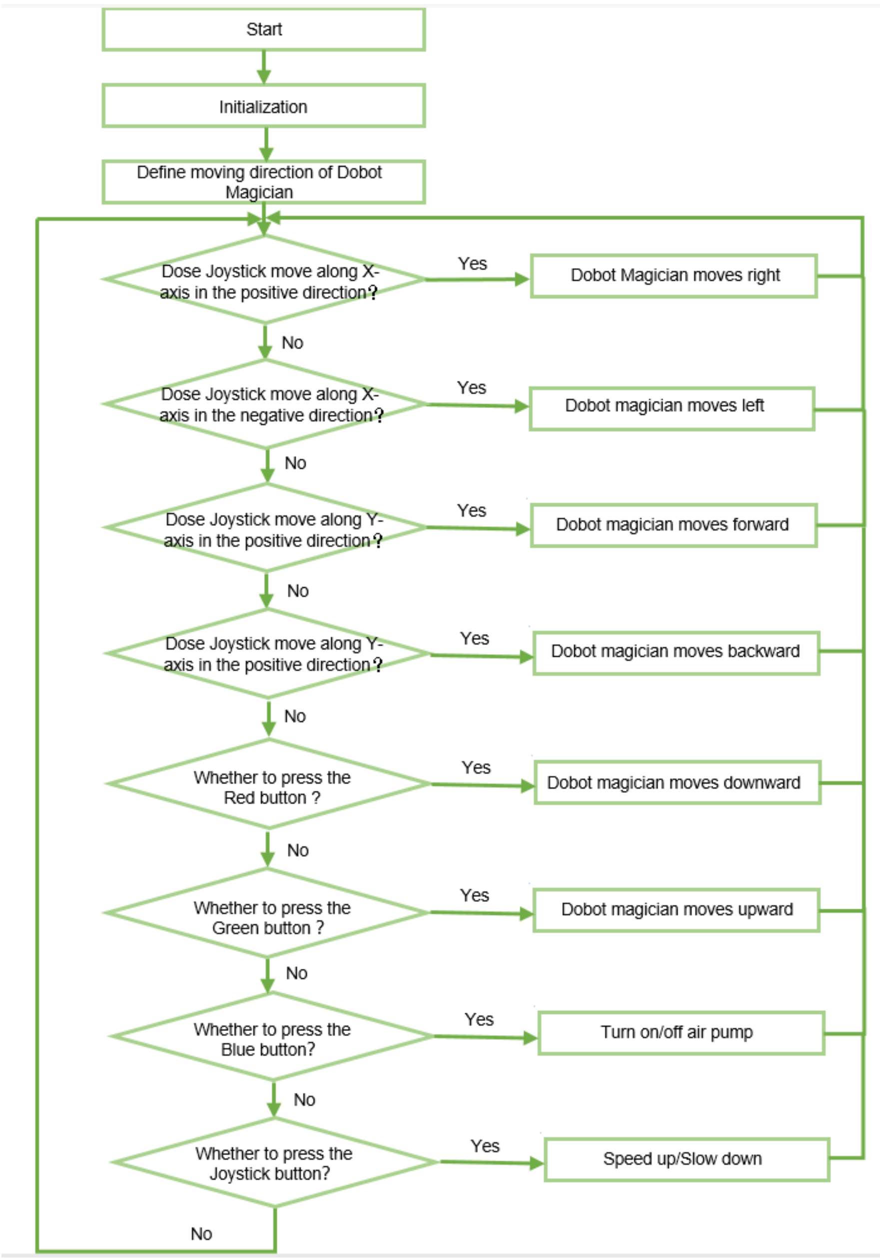


Figure 9.3 Realization process

9.4 Critical Code Description

This demo uses joystick to control Dobot Magician, the **Dobot** library and **SmartKit** library need to be called. Before debugging this Demo, please select **Dobot** and **SmartKit** on the **Sketch** > **Include Library** menu.

⚠NOTICE

Please long press the **Key** button on the back of the base of Dobot Magician to operate the homing procedure before debugging this Demo.

（1） Initialization.

Program 9.2 Initialization

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

（2） Define the moving direction of Dobot Magician and the air pump start-stop based on the moving direction of joystick and buttons.

📖NOTE

When moving joystick along X-axis or Y-axis, the analog values change from 0 to 1023, as shown Figure 9.4. The homing position of the joystick is at (x,y: 512,508).

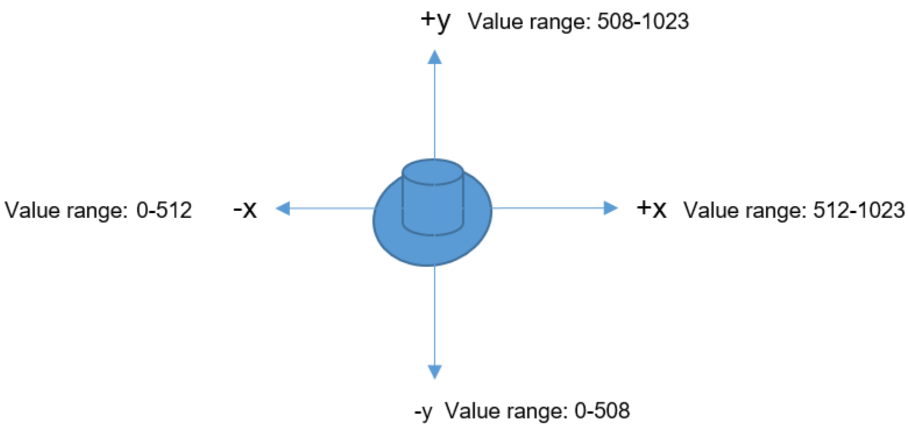


Figure 9.4 Value range

Program 9.3 Define the moving direction of Dobot Magician based on joystick and buttons

```
    int x = 0,y = 0,z = 0,b1 = 0,b2 = 0,b3 = 0;
    int direction = 0;                              //Define direction of JoyStick
    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){                        //JoyStick moves alongt Y-axis in the positive direction
        direction = 1;
  }
```

```
        else if(y < 400){                        // JoyStick moves alongt Y-axis in the negative direction
            direction = 2;
    }
…...
……
```

（3） Control the Dobot Magician and air pump by the joystick

Program 9.4 Control the Dobot Magician and air pump

```
  switch(direction){
    case 1:
        Serial.println("forward");
        Dobot_SetPTPCmdEx(MOVL_INC,20,0,0,0);        //Dobot Magician moves forward
        Serial.print("x=");
        Serial.println(x);
        Serial.print("y=");
        Serial.println(y);
        break;
case 2:
        Serial.println("backwardss");
        Dobot_SetPTPCmdEx(MOVL_INC,-20,0,0,0);              // Dobot Magician moves backward
        Serial.print("x=");
        Serial.println(x);
        Serial.print("y=");
        Serial.println(y);
        break;
……
……
}
```

10. DobotPixy Demo

10.1 Introduction

This demo uses Pixy vision sensor and Dobot Magician to recognize and pick-place different color cubes.

10.2 Hardware Connection

Arduino Mega2560, Pixy vision sensor, Dobot Magician and suction cup kit are required in this demo. For details about the installation and configuration of Pixy vision sensor, please see *Appendix C Pixy Install and Configure Pixy*. For the installation of suction kit, please see *Appendix B Installing Suction Cup Kit*.

The connection between them is shown in Figure 10.1.

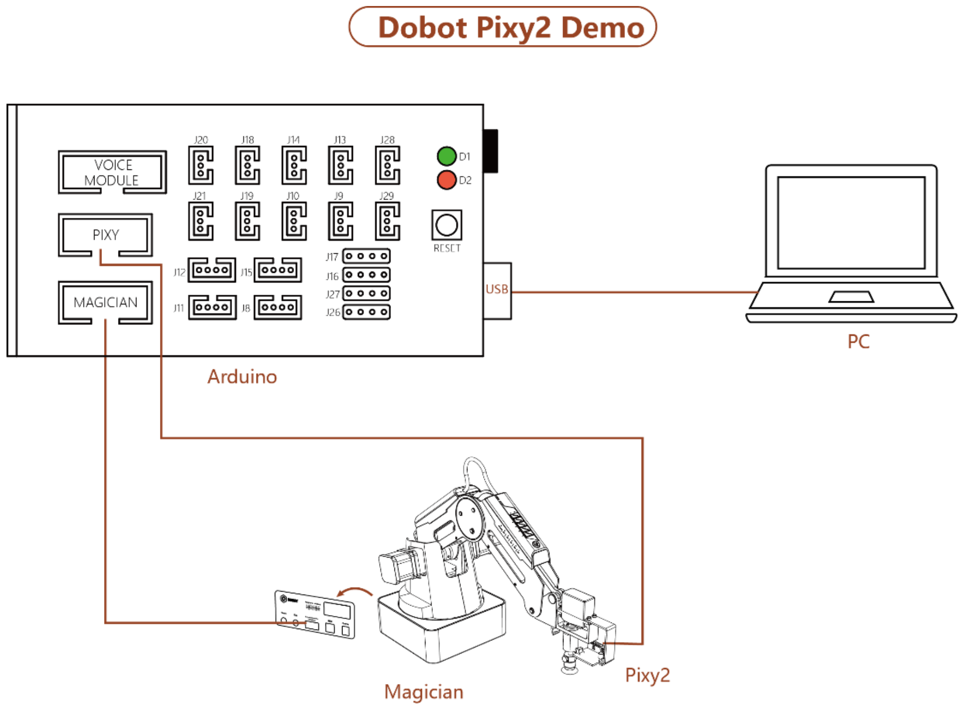


Figure 10.1 DobotPixy2 Connection

NOTE

If you connect the skill kit to other interfaces on the Arduino Mega2560, you also need to modify the right interfaces that the skill kit connects to in the **SmartKit.h** file.

Program 10.1 Define the interfaces that the skill kit connects to

```
#define JOYSTICK_XPIN      7    // Interface that the X-axis of the JoyStick connects to
#define JOYSTICK_YPIN      6    // Interface that the Y-axis of the JoyStick connects to
#define JOYSTICK_ZPIN      A5   // Interface that the Z-axis of the JoyStick connects to


#define LED_REDPIN         9    // Interface that the red LED indicator connects to
```

```
#define LED_GREENPIN       A1      // Interface that the green LED indicator connects to
#define LED_BLUEPIN        A3      // Interface that the blue LED indicator connects to

#define BUTTON_REDPIN      A0      // Interface that the red button connects to
#define BUTTON_GREENPIN A2      // Interface that the green button connects to
#define BUTTON_BLUEPIN    A4      // Interface that the blue button connects to
```

10.3 Realization Process

If there are eight cubes with different colors (red, yellow, green, blue), each color has two cubes. Place these cubes in the visual range (the Pixy vision sensor is installed on the end of Dobot Magician). After a color is detected by Pixy vision sensor, Dobot Magician will pick and place the corresponding cubes. Figure 10.2 shows its realization process.

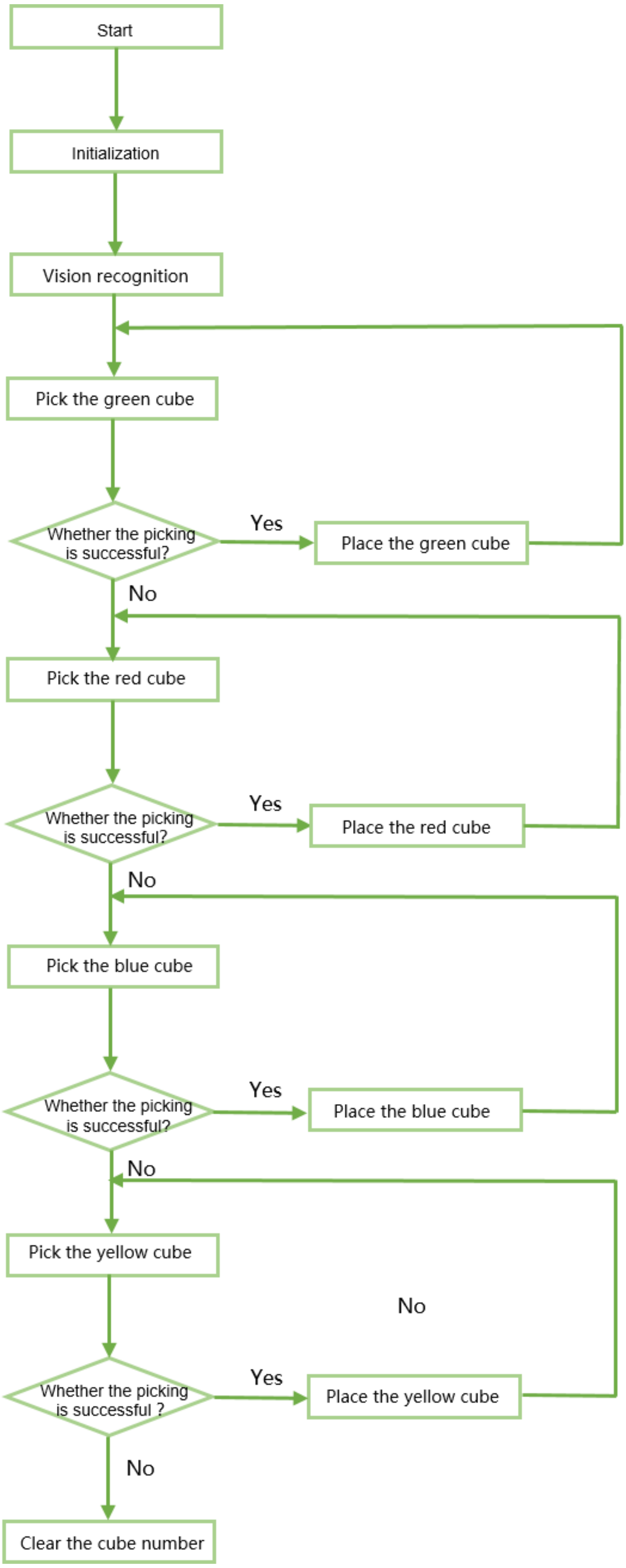


Figure 10.2 Realization process

10.4 Critical Code Description

This demo uses Pixy vision sensor and Dobot Magician to pick and place cubes. The **Dobot** library, **SmartKit** library, and **Pixy** library need to be called. Before debugging this Demo, please select **Dobot**, **SmartKit**, and **Pixy** on the **Sketch** > **Include Library** menu.

⚠NOTICE

Please long press the **Key** button on the back of base of Dobot Magician to operate the homing procedure before debugging this Demo.

（1） Initialization.

When setting the cube position, please connect Dobot Magician and DobotStudio. And press the **Unlock** key on the Forearm and drag Dobot Magician to move to the cube positions, then record their Cartesian coordinates from the **operation panel** pane of DobotStudio page to write in this demo for picking and placing cube.

For details about the vision recognition initialization process, please see Appendix D Vision Recognition Initialization Process.

Program 10.2 Initialization

```
SmartKit_VISSetAT(197.2155, 0.0679, 61.0561, 25.5385);          //Set the Pixy position (Camera position)
SmartKit_VISSetPixyMatrix(11, 153, 44, 45, 135, 91, 45, 46, 260, 11, 47, 43);       // Set the image coordinates of the cubes
SmartKit_VISSetColorSignature(RED, 1);                           // Set the color signature of the cube
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);        //Set the Cartesian coordinates of the cubes to obtain the rotation matrix based on the image coordinates and the Cartesian coordinates.
SmartKit_VISSetGrapAreaZ(-65);
SmartKit_VISSetBlockTA(RED, 120, -135.9563, -66.9085, 0);          // Set the placing position of the red cube
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);                               // Set the cube height
SmartKit_VISSetBlockHeight(GREEN, 26);
SmartKit_VISSetBlockHeight(BLUE, 26);
SmartKit_VISSetBlockHeight(YELLOW, 26);
SmartKit_VISInit();                                                // Initial the Pixy
SmartKit_Init();
```


（2） Pick and place cubes. If there are multiple cubes in the same color, these cubes will be piled when placing them.

Program 10.3 Pick and place cubes

```
    SmartKit_VISRun();            //Vision recognition: Obtain the numbers, colors and coordinates of the cubes.


    color = GREEN;
    Dobot_SetPTPJumpParams(10);                    //Set the lifting height
    while (SmartKit_VISGrabBlock(color, 1, 0) == TRUE)        // Pick the cube
    {
        Dobot_SetPTPJumpParams(30);                    //Set the lifting height
        SmartKit_VISPlaceBlock(color);                // Place the cube
    };
     SmartKit_VISSetBlockPlaceNum(color, 0);          // Clear the placing number
……
……
```

Appendix A Common Function Description

Common Function of SmartKit

SmartKit library encapsulates the common functions that users need to use. Please load the SamrtKit library to the Arduino before debugging demos.

Attached table 1 SmartKit initialization

Prototype	void SmartKit_Init(void)
Description	SmartKit initialization, includes joystick, LED indicators, buttons and voice recognition
Parameter	None
Return	None

Attached table 2 Button status check

Prototype	int SmartKit_ButtonCheckState(char color)
Description	Check the button status
Parameter	color: Button color. Value range: BLUE, GREEN, RED
Return	Button status: UP or DOWN

Attached table 3 Obtain the joystick value

Prototype	int SmartKit_JoyStickReadXYValue(int axis)
Description	Obtain the joystick value
Parameter	axis: Joystick direction. Value range: AXISX, AXISY
Return	Joystick value

Attached table 4 Joystick's button status check

Prototype	int SmartKit_JoyStickCheckPressState(void)
Description	Check the joystick's button status
Parameter	None
Return	Button status: UP or DOWN

Attached table 5 LED indicator status check

Prototype	int SmartKit_LedCheckState(char color)
Description	Check the LED indicator status
Parameter	color: LED indicator color. Value range: BLUE, GREEN, RED
Return	LED indicator status: ON or OFF

Attached table 6 LED indicator status control

Prototype	int SmartKit_LedTurn(char color, int state)
Description	Control the LED indicator status
Parameter	color: LED indicator color. Value range: BLUE, GREEN, RED state: LED indicator status. Value range: ON or OFF
Return	None

Attached table 7 Voice command check

Prototype	int SmartKit_VoiceENGVoiceCheck(int num)
Description	Check the voice command
Parameter	num: Vocie command number
Return	TURE: the voice command is checked. False: No voice command

Attached table 8 Start to check the voice command

Prototype	void SmartKit_VoiceENGStart(void)
Description	Start to check the voice command
Parameter	None
Return	None

⚠NOTICE

Grove speech recognizer only supports 22 voice commands without user-defined. The commands and their return values are as shown in Attached program 1.

Attached program 1 Command description

```
const char *voiceBuffer[] =
{
    "Turn on the light",                                    //return 1
```

```
    "Turn off the light",                                    //return 2
    "Play music",                                            //return 3
    "Pause",                                                 //return 4
    "Next",                                                  //return 5
    "Previous",                                              //return 6
    "Up",                                                    //return 7
    "Down",                                                  //return 8
    "Turn on the TV",                                        //return 9
    "Turn off the TV",                                       //return 10
    "Increase temperature",                                  //return 11
    "Decrease temperature",                                  //return 12
    "What's the time",                                       //return 13
    "Open the door",                                         //return 14
    "Close the door",                                        //return 15
    "Left",                                                  //return 16
    "Right",                                                 //return 17
    "Stop",                                                  //return 18
    "Start",                                                 //return 19
    "Mode 1",                                                //return 20
    "Mode 2",                                                //return 21
    "Go",                                                    //return 22
};
```

Attached table 9 Voice recognition initialization

Prototype	void SmartKit_VISInit (void)
Description	Initial voice recognition Before calling this API, you need to call SmartKit_VISSetDobotMatrix(float x1, float y1,float x2, float y2,float x3, float y3) and 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) to obtain the rotation matrix
Parameter	None
Return	None

NOTE

Rotation matrix: According to the image coordinates **A** $\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}$ and the corresponding Cartesian coordinates $\mathbf{B}\begin{bmatrix} x_1' & x_2' & x_3' \\ y_1' & y_2' & y_3' \\ 1 & 1 & 1 \end{bmatrix}$ of three points to obtain the rotation matrix, so that after the image coordinates of a cube are obtained by the Pixy vision sensor, the corresponding Cartesian coordinates where Dobot Magician will move to can be calculated with the $\mathbf{R_T}$ matrix, as shown in. The formula is $\mathbf{B} = \boldsymbol{R_T A}$.

Attached table 10 Set robot matrix

Prototype	void SmartKit_VISSetDobotMatrix(float x1, float y1,float x2, float y2,float x3, float y3)
Description	Set robot matrix. Namely, Place three cubes in the vision area and make the robot move to the centers of the three cubes to obtain their Cartesian coordinates.
Parameter	x1: X-axis coordinate of the cube1 y1: Y-axis coordinate of the cube1 x2: X-axis coordinate of the cube2 y2: Y-axis coordinate of the cube2 x3: X-axis coordinate of the cube3 y3: Y-axis coordinate of the cube3
Return	None

Attached table 11 Set Pixy matrix

Prototype	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)
Description	Set pixy matrix. Namely, Place three cubes in the vision area and obtain their image coordinates by the PixyMon.
Parameter	x1: X-axis coordinate of the cube1 y1: Y-axis coordinate of the cube1 length1: Length of the cube1 wide1: Width of the cube1 x2: X-axis coordinate of the cube2 y2: Y-axis coordinate of the cube2 length2: Length of the cube2 wide2: Width of the cube2 x3: X-axis coordinate of the cube3 y3: Y-axis coordinate of the cube3

	length3: Length of the cube3 wide3: Width of the cube3
Return	None

Attached table 12 Set the Z-axis coordinate of the pickup area

Prototype	void SmartKit_VISSetGrapAreaZ(float z)
Description	Set the Z-axis coordinate of the pickup area
Parameter	z: The height of the pickup area
Return	None

Attached table 13 Obtain the Z-axis coordinate of the pickup area

Prototype	float SmartKit_VISGetGrapAreaZ(void)
Description	Obtain the Z-axis coordinate of the pickup area
Parameter	None
Return	The height of the pickup area

Attached table 14 Set the vision area

Prototype	void SmartKit_VISSetAT(float x, float y, float z, float r)
Description	Set the vision area. Namely, set the position of the pixy
Parameter	x: X-axis coordinate of the vision area y: Y-axis coordinate of the vision area z: Z-axis coordinate of the vision area r: R-axis coordinate of the vision area
Return	None

Attached table 15 Set the color signature of the cube

Prototype	char SmartKit_VISSetColorSignature(char color, char signature)
Description	Set the color signature of the cube
Parameter	color: Cube color. Value range: RED, BLUE, YELLOW, GREEN signature: color signature. Value range: 1-7
Return	TRUE: The setting is successful FALSE: The setting is failed

Attached table 16 Set the placing position of the cube

Prototype	char SmartKit_VISSetBlockTA(char color, float x, float y, float z, float r)
Description	Set the placing position of the cube
Parameter	color: Cube color. Value range: RED, BLUE, YELLOW, GREEN x: X-axis coordinate of the placing position y: Y-axis coordinate of the placing position z: Z-axis coordinate of the placing position r: R-axis coordinate of the placing position
Return	TRUE: The setting is successful FALSE: The setting is failed

Attached table 17 Obtain the cube numbers that the pixy detects

Prototype	char SmartKit_VISGetBlockCheckNum(char color)
Description	Obtain the cube numbers of each color that the pixy detects
Parameter	color: Cube color. Value range: RED, BLUE, YELLOW, GREEN
Return	Cube numbers

Attached table 18 Set the numbers of cubes that need to be placed

Prototype	char SmartKit_VISSetBlockPlaceNum(char color, int placeNum)
Description	Set the numbers of cubes of each color that need to be placed
Parameter	color: Cube color. Value range: RED, BLUE, YELLOW, GREEN placeNum: Cube numbers
Return	TRUE: The setting is successful FALSE: The setting is failed

Attached table 19 Obtain the numbers of cubes that are placed

Prototype	char SmartKit_VISGetBlockPlaceNum(char color)
Description	Obtain the numbers of cubes of each color that are placed
Parameter	color: Cube color. Value range: RED, BLUE, YELLOW, GREEN
Return	Cube numbers

Attached table 20 Set the cube height

Prototype	char SmartKit_VISSetBlockHeight(char color, float height)
Description	Set the cube height

Parameter	color: Cube color. Value range: RED, BLUE, YELLOW, GREEN height: cube height
Return	TRUE: The setting is successful FALSE: The setting is failed

Attached table 21 Clear the numbers of cubes that the pixy detects

Prototype	void SmartKit_VISBlockParmCheckNumClear(void)
Description	Clear the numbers of cubes that the pixy detectss
Parameter	None
Return	None

Attached table 22 Vision recognition

Prototype	char SmartKit_VISRun(void)
Description	Execute the vision recognition to obtain the numbers, colors, coordinates of cubes
Parameter	None
Return	TRUE: The vision recognition is successful FALSE: The vision recognition is failed. The cubes may be not detected or the signature setting is wrong

Attached table 23 Pick cube

Prototype	char SmartKit_VISGrabBlock(char color, int blockNum, float r)
Description	Pick cubes
Parameter	color: Cube color. Value range: RED, BLUE, YELLOW, GREEN blockNum: The number of the cube to be picked r: Rotation angle when picking a cube
Return	TRUE: The pickup is successful FALSE: The pickup is failed

Attached table 24 Place cube

Prototype	char SmartKit_VISPlaceBlock(char color)
Description	Place cubes
Parameter	color: Cube color. Value range: RED, BLUE, YELLOW, GREEN
Return	TRUE: The placement is successful

	FALSE: The placement is failed

Common Function of Dobot Magician

Dobot Magician communicates with Arduino over UART interface (10 PIN) on the base of the Dobot Magician, using the Dobot protocol. We have provided Dobot library which encapsulates part of the Dobot Magician API, being called directly to control Dobot Magician. This topic describes the common functions that are used in Arduino demo. For details about Dobot Magician API, please see *Dobot Magician API Description*.

Attached table 25 Set the motion mode and the target coordinates

Prototype	void Dobot_SetPTPCmdEx(uint8_t Model, float x, float y, float z, float r)
Description	Set the motion mode and the target coordinates

Parameter	Details for Model:
	enum {
	JUMP_XYZ, //JUMP mode, (x,y,z,r) is the target point in Cartesian coordinate system
	MOVJ_XYZ, //MOVJ mode, (x,y,z,r) is the target point in Cartesian coordinate system
	MOVL_XYZ, //MOVL mode, (x,y,z,r) is the target point in Cartesian coordinate system
	JUMP_ANGLE, //JUMP mode, (x,y,z,r) is the target point in Joint coordinate system
	MOVJ_ANGLE, //MOVJ mode, (x,y,z,r) is the target point in Joint coordinate system
	MOVL_ANGLE, //MOVL mode, (x,y,z,r) is the target point in Joint coordinate system
	MOVJ_INC, //MOVJ mode, (x,y,z,r) is the angle increment in Joint coordinate system
	MOVL_INC, //MOVL mode, (x,y,z,r) is the Cartesian coordinate increment in Joint coordinate system
	MOVJ_XYZ_INC, //MOVJ mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian coordinate system
	JUMP_MOVL_XYZ, //JUMP mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian coordinate system
	};
	x、y、z、r : Coordinate parameters in PTP mode. (x,y,z,r) can be set to Cartesian coordinate, joints angle, or increment of them

Return	None

Attached table 26 Control the start-stop of air pump

Prototype	void Dobot_SetEndEffectorSuctionCupEx(bool issuck)
Description	Control the start-stop of the air pump
Parameter	Issuck: Whether to turn on the air pump. true: Turn on; false: Turn off
Return	None

Attached table 27 Set the speed ratio and acceleration ratio of Dobot Magician

Prototype	void Dobot_SetJOGCommonParamsEx(float velocityRatio, float accelerationRatio)
Description	Set the speed ratio and acceleration ratio of Dobot Magician
Parameter	velocityRatio: Speed ratio accelerationRatio: Acceleration ratio
Return	None

Attached table 28 Get the real-time pose of Dobot

Prototype	float Dobot_GetPoseEx(uint8_t temp)
Description	Get the real-time pose of Dobot

Parameter	temp: Cartesian coordinate system axis
	Details for temp:
	enum{
	X,……//X axis
	Y,……//Y axis
	Z,……//Z axis
	R,……//R axis
	JOINT1,……//joint axis 1
	JOINT2,……//joint axis 2
	JOINT3,……//joint axis 3
	JOINT4,……//joint axis 4
	};

Return	Return the value of axis or joint angle

Attached table 29 Get the device clock

Prototype	uint32_t Dobot_GetDeviceTimeEx(void)
Description	Get the device clock
Parameter	None
Return	Return the device clock

Attached table 30 Set the sliding rail status

Prototype	void Dobot_SetDeviceWIthLEx(bool isWithL)
Description	Set the Sliding rail status
Parameter	isWithL: 0:Disabled, 1:Enabled
Return	None

Attached table 31 Execute the homing function

Prototype	Void Dobot_SetHOMECmdEx(void)
Description	Execute the homing function
Parameter	None
Return	None

Attached table 32 Set the offset of end-effector

Prototype	void Dobot_SetEndEffectorParamsEx(float x,float y, float z)
Description	Set the offset of the end-effector. If the end-effector is installed, this API is required
Parameter	x: the X-axis offset of end-effector y: the Y-axis offset of end-effector z: the Z-axis offset of end-effector
Return	None

Attached table 33 Enable laser

Prototype	void Dobot_SetEndEffectorLaserEx (uint8_t isEnable, float power)

Description	Enable laser
Parameter	isEnable: Control laser. 0: Disabled, 1: Enabled power: PWM duty cycle:0-100
Return	None

Attached table 34 Set gripper status

Prototype	void Dobot_SetEndEffectorGripperEx(bool isEnable,bool isGriped)
Description	Set gripper status
Parameter	isEnable: Control end-effector. 0: Disabled, 1: Enabled isGriped: Control the gripper to grab or release. 0:Released, 1: Grabbed
Return	None

Attached table 35 Set the velocity and acceleration of the joints coordinate axis in jogging mode

Prototype	void Dobot_SetJOGJointParamsEx(float velocityJ1,float accelerationJ1,float velocityJ2,float accelerationJ2,float velocityJ3,float accelerationJ3,float velocityJ4,float accelerationJ4)
Description	Set the velocity and acceleration of the joints coordinate axis in jogging mode
Parameter	velocityJ1、velocityJ2、velocityJ3、velocityJ4: joints velocity in jogging mode accelerationJ1、accelerationJ2、accelerationJ3、accelerationJ4: joints acceleration in jogging mode
Return	None

Attached table 36 Execute the jogging command

Prototype	void Dobot_SetJOGCmdEx(uint8_t model)
Description	Execute the Jogging command. Please call this API after setting the related parameters of jogging
Parameter	model : Jogging command Details for model: 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- };
Return	None

Attached table 37 Set the velocity ratio and the acceleration ratio in PTP mode

Prototype	void Dobot_SetPTPCommonParamsEx(float velocityRatio,float accelerationRatio)
Description	Set the velocity ratio and the acceleration ratio in PTP mode
Parameter	velocityRatio: velocity ratio in PTP mode accelerationRatio: velocity acceleration in PTP mode
Return	None

Attached table 38 Set the velocity and acceleration of the joint coordinate axis in PTP mode

Prototype	void Dobot_SetPTPJointParamsEx(float velocityJ1,float accelerationJ1,float velocityJ2,float accelerationJ2,float velocityJ3,float accelerationJ3,float velocityJ4,float accelerationJ4)
Description	Set the velocity and acceleration of the joint coordinate axis in PTP mode
Parameter	velocityJ1、velocityJ2、velocityJ3、velocityJ4: Joint velocity in PTP mode accelerationJ1、accelerationJ2、accelerationJ3、accelerationJ4: Joint acceleration in PTP mode
Return	None

Attached table 39 Set the velocity and acceleration of sliding rail in PTP mode

Prototype	void Dobot_SetPTPLParamsEx(float velocityRatio, float accelerationRatio)
Description	Set the velocity and acceleration of sliding rail in PTP mode
Parameter	velocityRatio: Sliding rail velocity in PTP mode accelerationRatio: Sliding rail acceleration in PTP mode

Return	None

Attached table 40 Set the lifting height in JUMP mode

Prototype	void Dobot_SetPTPJumpParamsEx(float jumpHeight)
Description	Set the lifting height in JUMP mode
Parameter	jumpHeight: The lifting height
Return	None

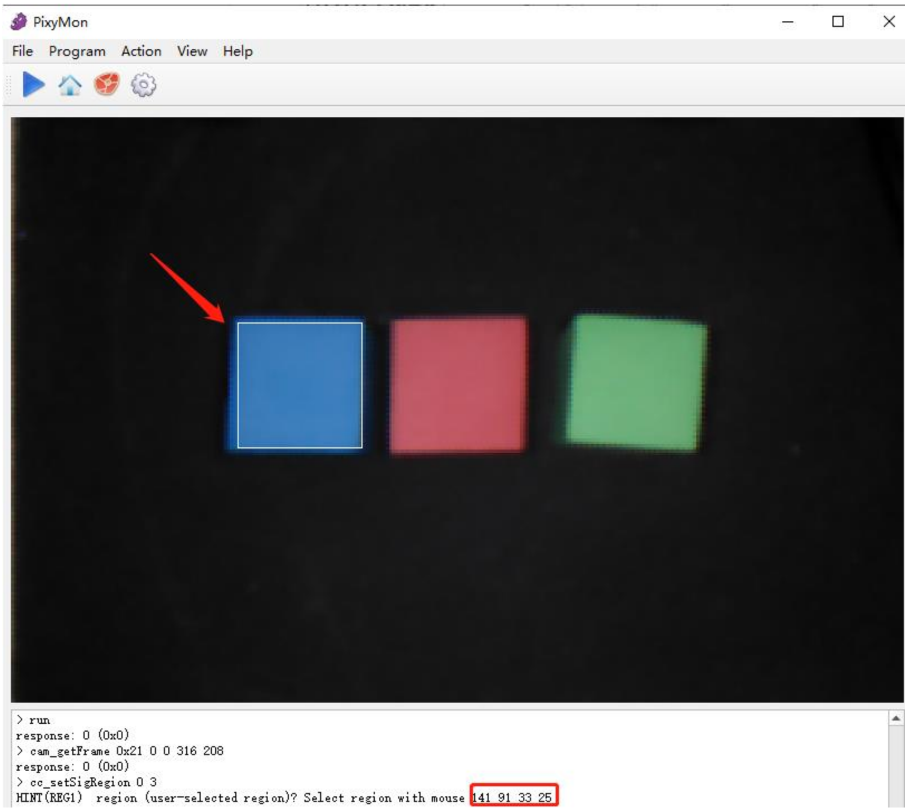
Attached table 41 Execute a PTP command with the sliding rail

Prototype	void Dobot_SetPTPWithLCmdEx(uint8_t Model,float x,float y,float z,float r,float l)
Description	Execute a PTP command with the sliding rail
Parameter	Model: PTP mode Details for Model: enum { JUMP_XYZ, //JUMP mode, (x,y,z,r) is the target point in Cartesian coordinate system MOVJ_XYZ, //MOVJ mode, (x,y,z,r) is the target point in Cartesian coordinate system MOVL_XYZ, //MOVL mode, (x,y,z,r) is the target point in Cartesian coordinate system JUMP_ANGLE, //JUMP mode, (x,y,z,r) is the target point in Joint coordinate system MOVJ_ANGLE, //MOVJ mode, (x,y,z,r) is the target point in Joint coordinate system MOVL_ANGLE, //MOVL mode, (x,y,z,r) is the target point in Joint coordinate system MOVJ_INC, //MOVJ mode, (x,y,z,r) is the angle increment in Joint coordinate system MOVL_INC, //MOVL mode, (x,y,z,r) is the Cartesian coordinate increment in Joint coordinate system MOVJ_XYZ_INC, //MOVJ mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian coordinate system UMP_MOVL_XYZ, //JUMP mode, (x,y,z,r) is the Cartesian coordinate increment in Cartesian

	}; x、y、z、r: Coordinate parameters in PTP mode.(x,y,z,r) can be set to Cartesian coordinate, joints angle, or increment of them l: The distance that sliding rail moves
Return	None

Attached table 42 Set the I/O multiplexing

Prototype	void Dobot_SetIOMultiplexingEx(uint8_t address,uint8_t function)
Description	Set the I/O multiplexing
Parameter	address: I/O address(1~20),please refer to *Appendix D Vision Recognition Initialization Process* *Vision* recognition initialization process includes vision area setting, Cartesian coordinates of the cubes setting, image coordinates of the cubes setting, Z-axis coordinate of the pickup area setting, placing position setting, cube height setting, and color signature setting. For details, please see as follows. **Step 1** Connect the Dobot Magician to DobotStudio and execute the homing procedure. For details, please see *Dobot Magician User Guide*. **Step 2** Install the Pixy and launch the PixyMon. For details, please see Appendix C Pixy Install and Configure Pixy. **Step 3** Set the vision area. Namely, set the pixy position. Move the Dobot Magician to a right position to make the Pixy detect cubes in the vision area and record the X, Y, Z, R vules on the DobotStudio page, then write these values to the SmartKit_VISSetAT(float x, float y, float z, float r) function. **Step 4** Place three cubes in the vision area. **Step 5** Set the image coordinates of the three cubes. Click **Action > Set signature 1…** on the PixyMon page and select the diagonal line of the three cubes respectively, then the image coordinates of the corresponding cube will be displayed on the PixyMon page, as shown in Attached figure 13. Write them in order in the 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) function.



Attached figure 13 Obtain the image coordinates

Step 6 Set the Cartesian coordinates of the three cubes.

Move the Dobot Magician to the center of the three cubes in the order of **Step 5** and record X, Y values on the DobotStudio. Then write them in order in the SmartKit_VISSetDobotMatrix(float x1, float y1,float x2, float y2, float x3, float y3) function.

Step 7 ⚠NOTICE

Step 8 The image coordinates of the cube need to be corresponding to the Cartesian coordinates. Otherwise, the rotation matrix will fail to be obtained 。

Set the Z-axis coordinate of the pickup area.

Move the Dobot Magicain to the plane where the cube is located and record the Z value on the DobotStudio, then write it in the SmartKit_VISSetGrapAreaZ(float z) function.

Step 9 Set the placing positions of the cubes of each color.

Step 10 Move the Dobot Magician to the positions where cubes of each color are to be placed and record X, Y, Z, R values on the DobotStudio page. Write them in the SmartKit_VISSetBlockTA(char color, float x, float y, float z, float r) function.

Set the height of the cubes of each color.

Write the height of the cubes of each color in the SmartKit_VISSetBlockHeight(char color, float height) function.

The cube height shall be obtained by user with measuring tools. Unit: mm.

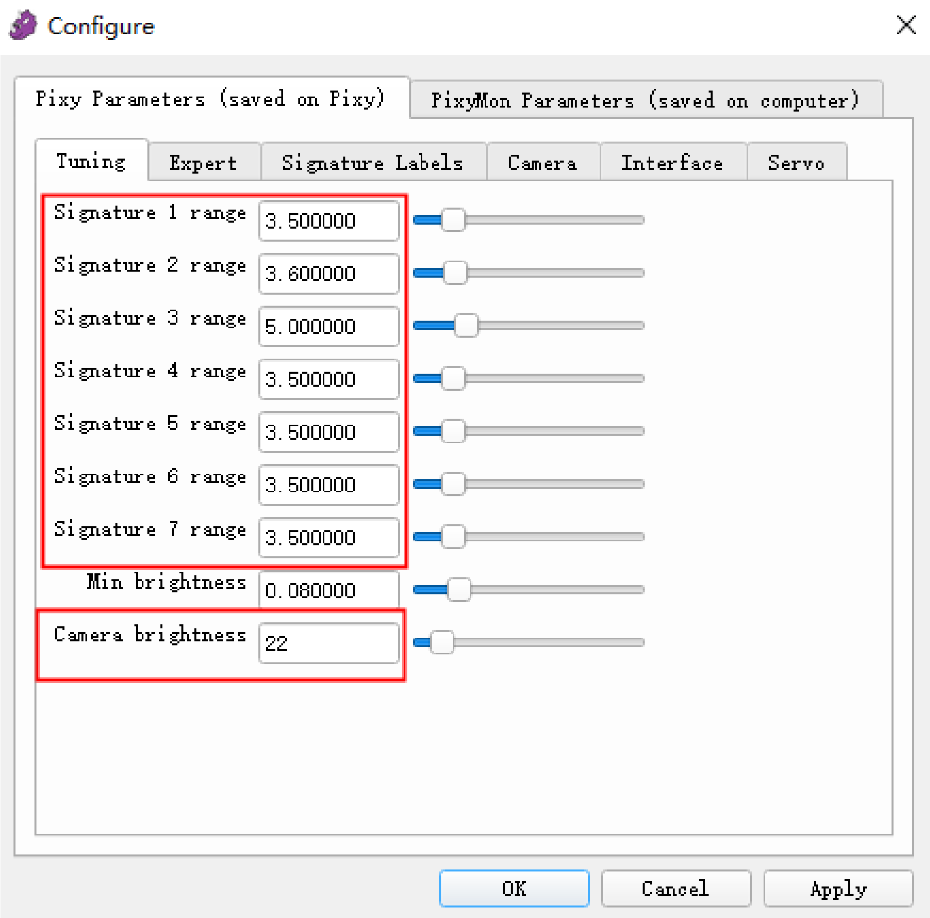
Set the color signature of the cubes.

Step 11 Click Action > Set signature x… on the PixyMon page to set the color signature and write them in the SmartKit_VISSetColorSignature(char color, char signature) function.

x indicates the color signature. For each color, set signature once. Pixy can only set 7 signature labels.

Step 12 🕮NOTE

Step 13 If the vision recognition effect is poor, you can adjust Signature x range and Camera brightness on the **File > Configure > Pixy Parameters(saved on Pixy)** tab to improve the accuracy, as shown in Attached figure 14.



Step 14

Step 15 Attached figure 14 Adjust range and brightness

Multiplexed **I/O Interface Description** for details

function: I/O multiplexing function

Details for function:

	typedef enum {
	IOFunctionDummy; //Invalid
	IOFunctionDO; // I/O output
	IOFunctionPWM; // PWM output
	IOFunctionDI; //I/O input
	IOFunctionADC; //A/D input
	IOFunctionDIPU; //Pull-up input
	IOFunctionDIPD //Pull-down input
	}

Return	None

Attached table 43 Set I/O output

Prototype	void Dobot_SetIODOEx(uint8_t address,uint8_t value)
Description	Set I/O output
Parameter	address: /O address(1~20) value: Low level(0), High level(1)
Return	None

Attached table 44 Set PWM output

Prototype	void Dobot_SetIOPWMEx(uint8_t address, float freq, float duty)
Description	Set PWM output
Parameter	address: I/O address(1~20) freq: PWM frequency(10HZ~1MHz) duty: PWM duty circle(0~100)
Return	None

Attached table 45 Get I/O input

Prototype	uint8_t Dobot_GetIODIEx(uint8_t address)
Description	Get I/O input
Parameter	address: I/O address(1~20)
Return	return I/O input

Attached table 46 Get A/D input

Prototype	uint16_t Dobot_GetIOADCEx(uint8_t address)
Description	Get A/D input
Parameter	address: I/O address(1~20)
Return	Return A/D input

Attached table 47 Set the velocity of extended motor

Prototype	void Dobot_SetEMotorEx(uint8_t address, uint8_t enable, uint32_t speed)
Description	Set the velocity of extended motor
Parameter	address: Motor index. 0:Stepper1, 1:Stepper2 enable: Control motor. 0:Disabled, 1:Enabled speed: Motor velocity（Pulse number per second）
Return	None

Attached table 48 Set the velocity and movement distance of extended motor

Prototype	void Dobot_SetEMotorSEx(uint8_t address, uint8_t enable, uint32_t speed, uint32_t deltaPulse)
Description	Set the velocity and movement distance of extended motor, The Dobot will move for some distance at a constant velocity after calling this API
Parameter	address: Motor index. 0:Stepper1, 1:Stepper2 enable: Control motor.。 0:Disabled, 1:Enabled speed: Control motor（Pulse number per second） deltaPulse: The distance that motor moves(Pulse number)
Return	None

Attached table 49 Enable the color sensor

Prototype	void Dobot_SetColorSensorEx(uint8_t enable, uint8_t port)
Description	Enable the color sensor
Parameter	enable: 0:Disabled, 1:Enabled port: the Dobot interface that the color sensor is connected to. Please select the

	corresponding interface
	Details for port:
	enum {
	IF_PORT_GP1;
	IF_PORT_GP2;
	IF_PORT_GP4;
	IF_PORT_GP5;
	};

Return	None

Attached table 50 Get the color sensor value

Prototype	uint8_t Dobot_GetColorSensorEx(uint8_t color)
Description	Get the color sensor value
Parameter	color: 0:Red, 1:Green, 2:Blue
Return	Retun the color sensor value

Attached table 51 Set the losing-step threshold

Prototype	void Dobot_SetLostStepSetEx(float lostStepValue)
Description	Set the losing-step threshold. If you do not call this API, the default threshold is 5
Parameter	lostStepValue: losing-step threshold
Return	None

Attached table 52 Execute losing-step command

Prototype	void Dobot_SetLostStepCmdEx(void)
Description	Execute losing-step command
Parameter	None
Return	None

Attached table 53 Set the IR sensor

Prototype	void Dobot_SetIRSwitchEx(uint8_t enable, uint8_t port)

Description	Set the IR sensor

Parameter	enable: 0:Disabled, 1:Enabled
	port: the Dobot interface that the IR sensor is connected to. Please select the corresponding interface
	Details for port:
	enum {
	IF_PORT_GP1;
	IF_PORT_GP2;
	IF_PORT_GP4;
	IF_PORT_GP5;
	};

Return	None

Attached table 54 Get the IR sensor value

Prototype	uint8_t GetIRSwitchEx(uint8_t port)
Description	Get the IR sensor value

Parameter	port: the Dobot interface that the IR sensor is connected to. Please select the corresponding interface
	Details for port:
	enum {
	IF_PORT_GP1;
	IF_PORT_GP2;
	IF_PORT_GP4;
	IF_PORT_GP5;
	};

Return	Return the IR sensor value

Common Function of Arduino

Arduino demo is developed based on Arduino Mega2560 and the Arduino APIs need to be called. This topic describes the common functions that are used in Arduino demo.

Attached table 55 Set the specified pin to INPUT or OUTPUT

Prototype	void pinMode(uint8_t pin, uint8_t mode)
Description	Set the specified digital pin to **INPUT** or **OUTPUT**
Parameter	pin: Pin number of the pin mode: INPUT or OUTPUT

Return	None

Attached table 56 Set the specified digital pin to HIGH or LOW

Prototype	void digitalWrite(uint8_t pin, uint8_t value)
Description	Set the specified digital pin to **HIGH** or **LOW**
Parameter	pin: Pin number of the pin mode: HIGH or LOW
Return	None

Attached table 57 Read the specified digital pin

Prototype	int digitalRead(uint8_t pin)
Description	Read the specified digital pin
Parameter	pin: Pin number of the pin
Return	HIGH or LOW

Attached table 58 Write the specified analog pin

Prototype	void analogWrite(uint8_t pin, int value)
Description	Write the specified analog pin for controlling the brightness of LED indicator or the motor speed
Parameter	pin: Pin number of the pin value: 0-255. 0: OFF; 255: ON
Return	None

Attached table 59 Read the specified analog pin

Prototype	int analogRead(uint8_t pin)
Description	Read the specified analog pin
Parameter	pin: Pin number of the pin
Return	0-1023

Pixy Common Function of Pixy

When the Pixy vision sensor is used in Arduino demo, the Pixy APIs need to be called. This topic describes the common functions in Arduino demo.

Attached table 60 Return the number of cubes Pixy has detected

Prototype	uint16_t getBlocks()

Description	Obtain the number of cubes Pixy has detected. You can then look in the pixy.blocks[] array for information about each detected object (one array member for each detected object.) Each array member (i) contains the following fields
	• If the vision sensor is Pixy:
	  • pixy.blocks[i].signature： The signature number (color) of the detected cube (1-7).
	  • pixy.blocks[i].x： The X-coordinate of the center of the detected object (0-319).
	  • pixy.blocks[i].y： The Y-coordinate of the center of the detected object (0-319).
	  • pixy.blocks[i].width： The width of the detected cube (1-320).
	  • pixy.blocks[i].height： The height of the detected cube (1-200).
	• If the vision sensor is Pixy2:
	  • pixy.ccc.blocks[i].m_signature： The signature number (color) of the detected cube (1-7).
	  • pixy.ccc.blocks[i].m_x： The X-coordinate of the center of the detected object (0-319).
	  • pixy.ccc.blocks[i].m_y： The Y-coordinate of the center of the detected object (0-319).
	  • pixy.ccc.blocks[i].m_width： The width of the detected cube (1-320).
	  • pixy.ccc.blocks[i].m_height： The height of the detected cube (1-200).

Parameter	None
Return	Return the number of cubes Pixy has detected

Appendix B Installing Suction Cup Kit

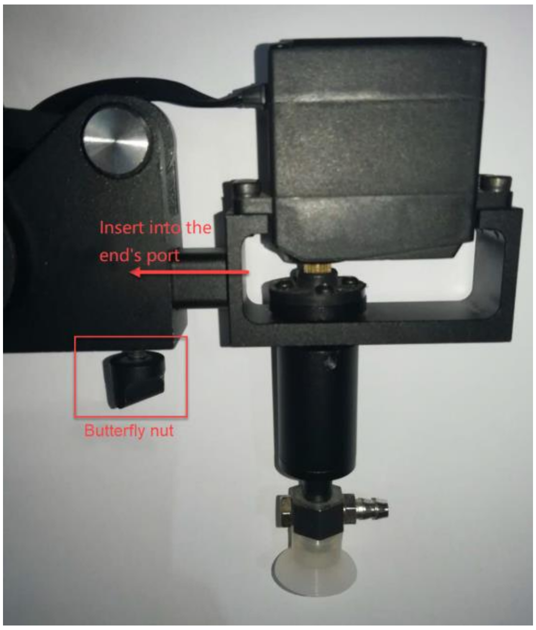
Procedure

When picking and placing cubes with Dobot Magician, the suction cup kit should be installed on the end of the Dobot Magician

Step 1 Connect the air pump's power cable **SW1** to the **SW1** connector on the Dobot Magician base' rear panel and the signal cable **GP1** to the **GP1** connector, as shown in Attached figure 1.

Attached figure 1 Connect the air pump to the Dobot Magician

Step 2 Insert a suction cup kit into the end's port, and fasten it with a butterfly nut, as shown in Attached figure 2.



Attached figure 2 Install a suction cup kit

Step 3 Connect the air pump's air tube to the air tube connector of the suction cup kit, as shown in Attached figure 3.

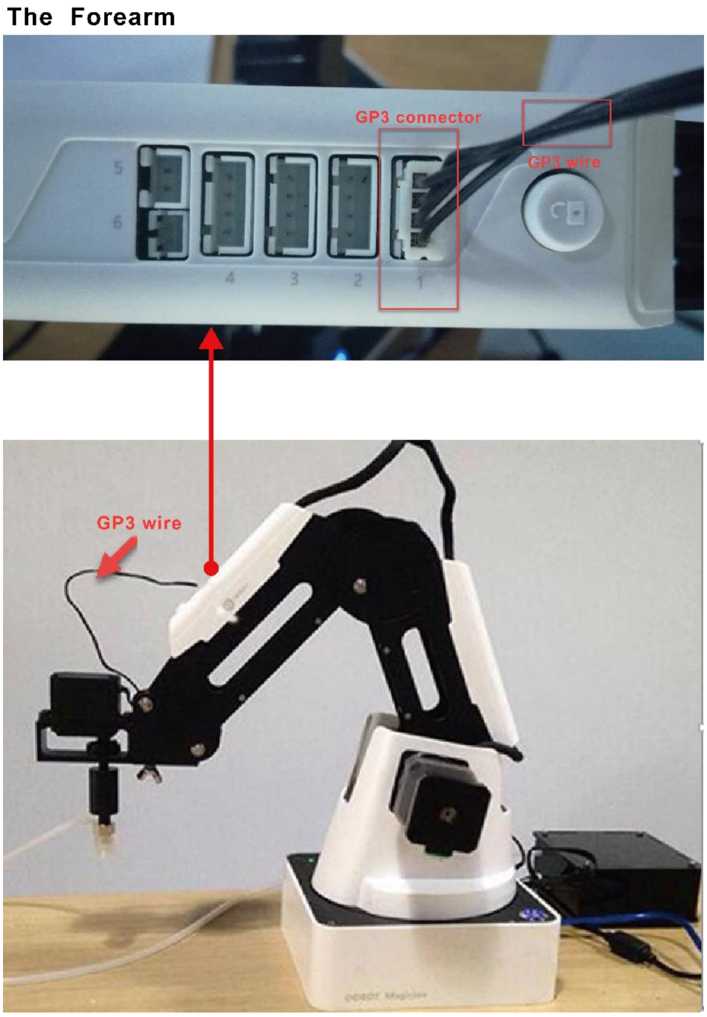
Attached figure 3 Install an air tube

📖NOTE

If the suction cup matched with the suction cup kit cannot catch cubes, please replace with the suction cup matched with the Arduino kit, as shown in

Attached figure 4 Suction cup matched with the Arduino kit

Step 4 Connect the servo's **GP3** cable to the **GP3** connector on the Forearm, as shown in Attached figure 4.



Attached figure 5 Connect the servo's GP3 cable to the GP3 connector

Appendix C Pixy Install and Configure Pixy

Before debugging the DobotPixy Demo, you need to install and configure the Pixy vision sensor.

Prerequisites

- The PixyMon has been installed, which is obtained from **arduino kit/PixyMon**.
- The suction kit has been installed on the end of Dobot Magician.
- The cubes have been obtained.

Procedure

Step 1 Loosen the two M3*8 hexagon socket head cap screws on the servo, as shown in Attached figure 6.

Attached figure 6 loosen the screws

Step 2 Mount the Pixy vision sensor on the servo with a connecting board, as shown in Attached figure 7. The vision sensor shown in Attached figure 7 is Pixy.

Attached figure 7 Install Pixy vision sensor

If the vision sensor is Pixy2, please mount it on the servo with the vision fixture, as shown in Attached figure 8.

Attached figure 8 Install Pixy2 vision sensor

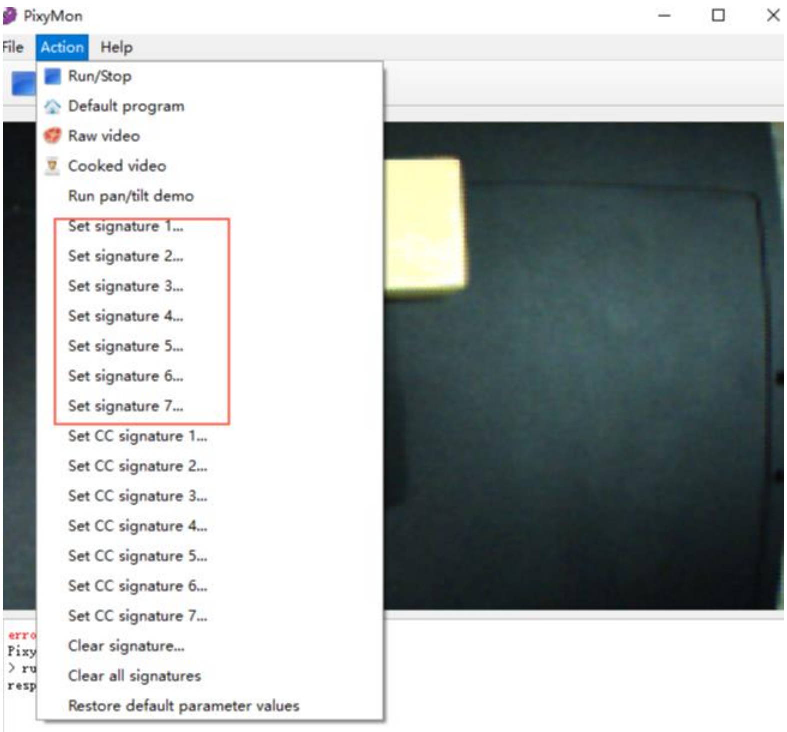
Step 3 Connect the Pixy vision sensor and PC with USB cable.

Step 4 Place cubes on the vision range.

Step 5 Rotate around the focal length of the camera on the Pixy vision sensor to adjust the camera view to the optimum state, until the cubes can be appeared clearly on the **PixyMon** page.

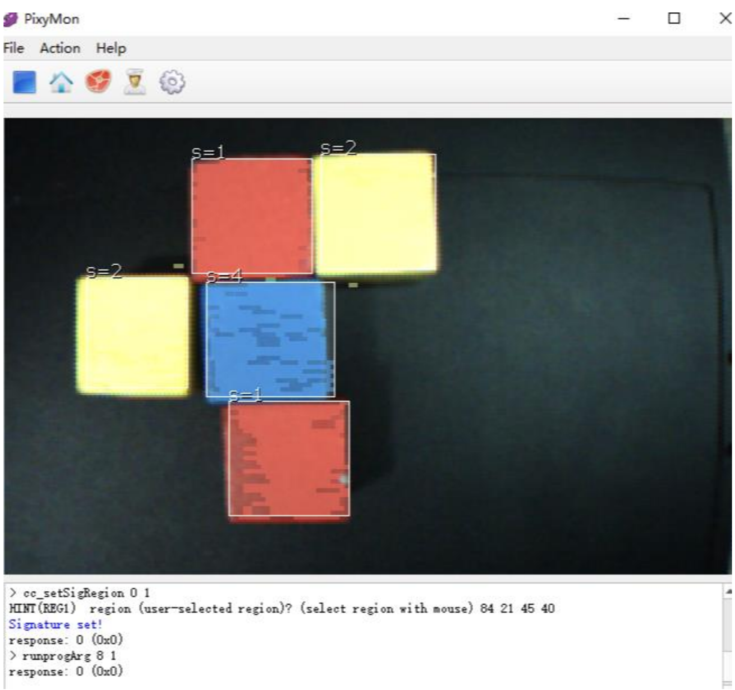
Step 6 Select **Action > Set signature *x*** on the **PixyMon** page and select an area on a cube image to set signature number, as shown in Attached figure 9.

For each color, set signature once. Pixy can only set 7 signature labels.



Attached figure 9 Set signature number

Attached figure 10 shows the setting result.

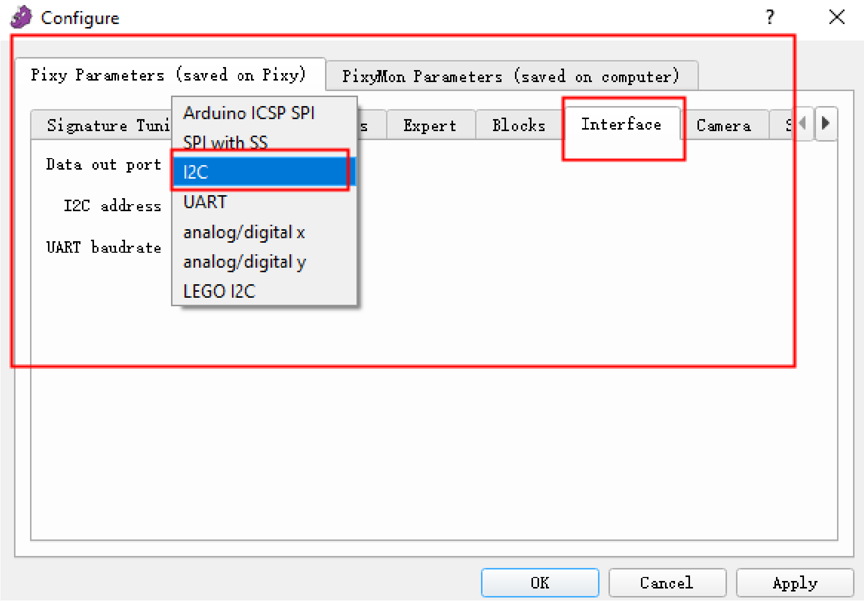


Attached figure 10 Setting result

Step 7 Click  on the **PixyMon** page.

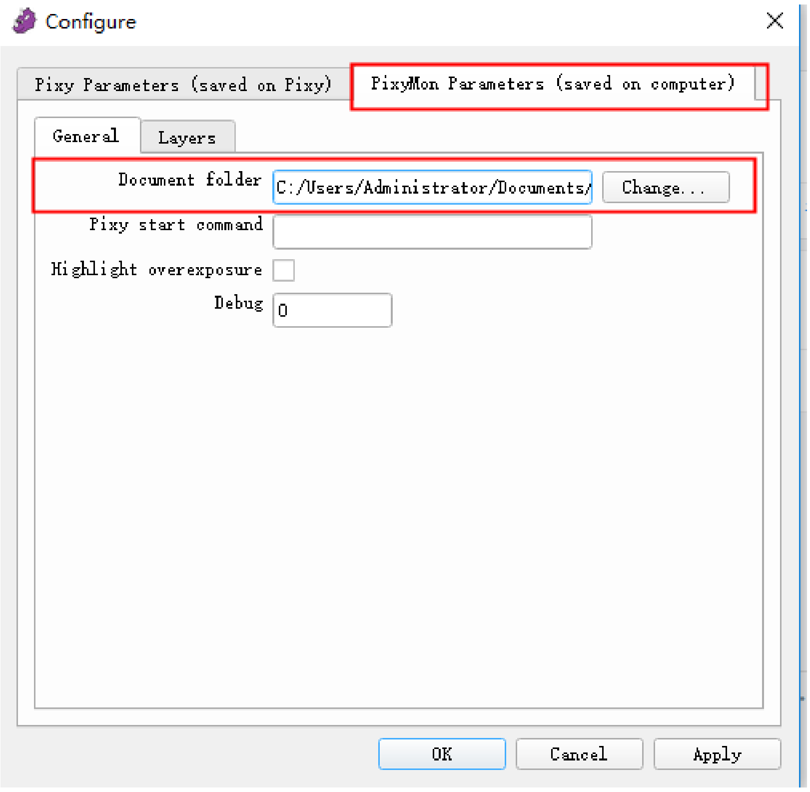
The **Configure** page is displayed.

Step 8 Set **Data out port** to **I2C** on the **Pixy Parameters (saved on Pixy) > Interface** tab of the **Configure** page, as shown in Attached figure 11.



Attached figure 11 Set Data out port

Step 9 Set **Document folder** to the installation directory on the **Pixy Parameters (saved on computer) > General** tab of the **Configure** page, as shown in Attached figure 12.



Attached figure 12 Set Document folder

Step 10 Remove the USB connection between the Pixy and PC.

Appendix D Vision Recognition Initialization Process

Vision recognition initialization process includes vision area setting, Cartesian coordinates of the cubes setting, image coordinates of the cubes setting, Z-axis coordinate of the pickup area setting, placing position setting, cube height setting, and color signature setting. For details, please see as follows.

Step 1 Connect the Dobot Magician to DobotStudio and execute the homing procedure. For details, please see *Dobot Magician User Guide*.

Step 2 Install the Pixy and launch the PixyMon. For details, please see Appendix C Pixy Install and Configure Pixy.

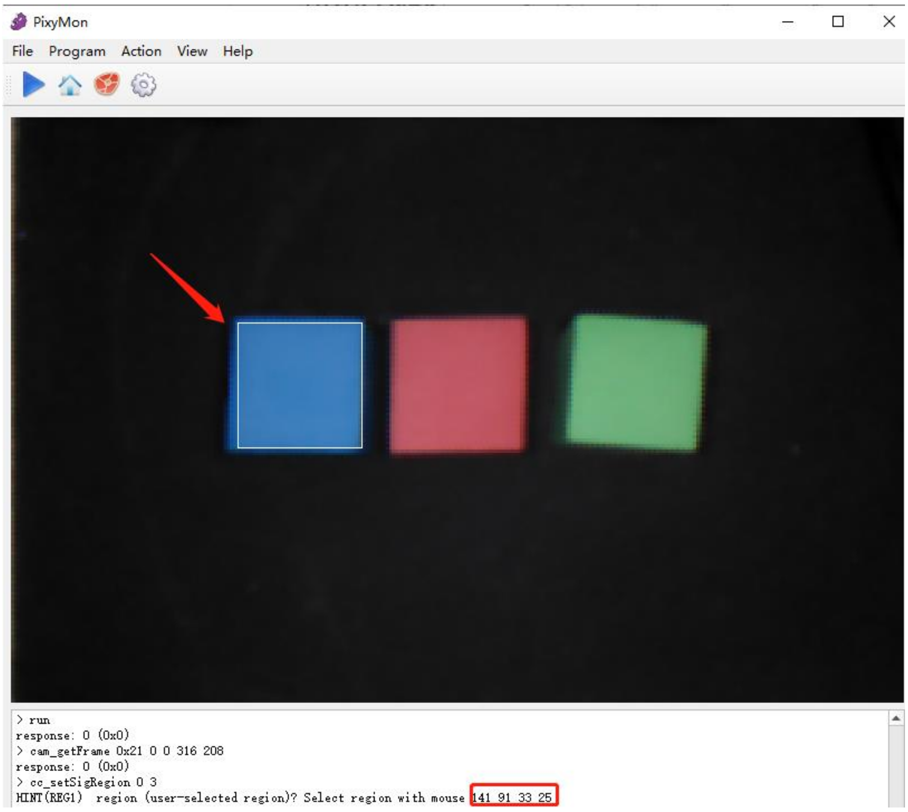
Step 3 Set the vision area. Namely, set the pixy position.

Move the Dobot Magician to a right position to make the Pixy detect cubes in the vision area and record the X, Y, Z, R vules on the DobotStudio page, then write these values to the SmartKit_VISSetAT(float x, float y, float z, float r) function.

Step 4 Place three cubes in the vision area.

Step 5 Set the image coordinates of the three cubes.

Click **Action > Set signature 1…** on the PixyMon page and select the diagonal line of the three cubes respectively, then the image coordinates of the corresponding cube will be displayed on the PixyMon page, as shown in Attached figure 13. Write them in order in the 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) function.



Attached figure 13 Obtain the image coordinates

Step 6 Set the Cartesian coordinates of the three cubes.

Move the Dobot Magician to the center of the three cubes in the order of **Step 5** and record X, Y values on the DobotStudio. Then write them in order in the SmartKit_VISSetDobotMatrix(float x1, float y1,float x2, float y2, float x3, float y3) function.

⚠NOTICE

The image coordinates of the cube need to be corresponding to the Cartesian coordinates. Otherwise, the rotation matrix will fail to be obtained 。

Step 7 Set the Z-axis coordinate of the pickup area.

Move the Dobot Magicain to the plane where the cube is located and record the Z value on the DobotStudio, then write it in the SmartKit_VISSetGrapAreaZ(float z) function.

Step 8 Set the placing positions of the cubes of each color.

Move the Dobot Magician to the positions where cubes of each color are to be placed and record X, Y, Z, R values on the DobotStudio page. Write them in the SmartKit_VISSetBlockTA(char color, float x, float y, float z, float r) function.

Step 9 Set the height of the cubes of each color.

Write the height of the cubes of each color in the SmartKit_VISSetBlockHeight(char color, float height) function.

The cube height shall be obtained by user with measuring tools. Unit: mm.

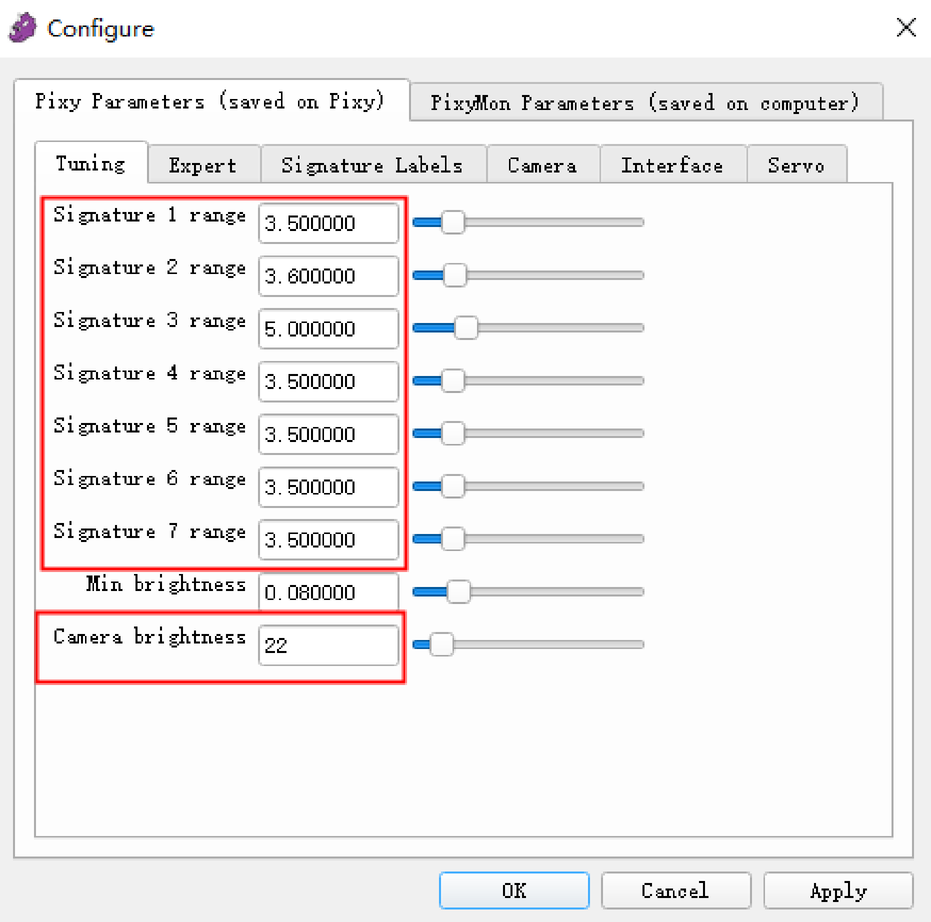
Step 10 Set the color signature of the cubes.

Click **Action** > **Set signature x…** on the PixyMon page to set the color signature and write them in the SmartKit_VISSetColorSignature(char color, char signature) function.

x indicates the color signature. For each color, set signature once. Pixy can only set 7 signature labels.

📖NOTE

If the vision recognition effect is poor, you can adjust **Signature x range** and Camera brightness on the **File** > **Configure** > **Pixy Parameters(saved on Pixy)** tab to improve the accuracy, as shown in Attached figure 14.

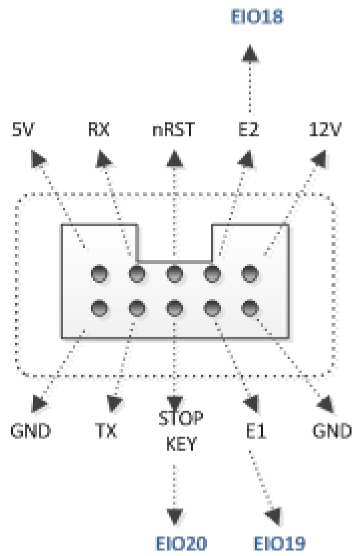


Attached figure 14 Adjust range and brightness

Appendix E Multiplexed I/O Interface Description of V1 Dobot Magician

Multiplexed UART Interface Description

Attached figure 15 shows the UART interface on the base, Attached table 61 lists the multiplexed I/O description.



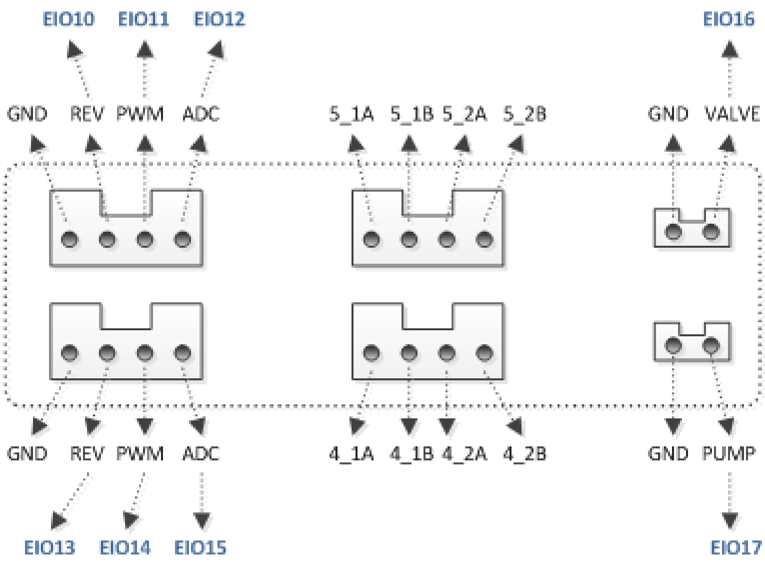
Attached figure 15 UART interface

Attached table 61 Multiplex I/O Description

I/O addressing	Voltage	Level Output	PWM	Level Input	ADC
18	3.3V	√	-	√	-
19	3.3V	√	-	√	-
20	3.3V	√	-	√	-

Multiplexed Peripheral Interface Description

Attached figure 16 shows the peripheral interface on the base, and Attached table 62 lists the multiplexed I/O description.



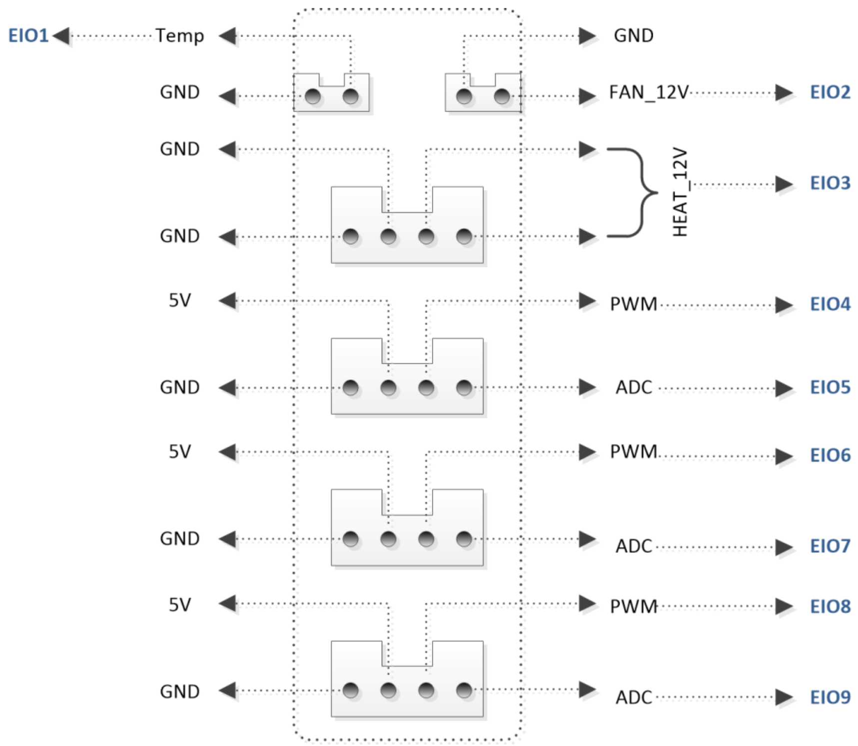
Attached figure 16 Peripheral Interface

Attached table 62 Multiplexed I/O Description

I/O addressing	Voltage	Level Output	PWM	Level Input	ADC
10	5V	√	-	-	-
11	3.3V	√	√	√	-
12	3.3V	√	-	√	√
13	5V	√	-	-	-
14	3.3V	√	√	√	-
15	3.3V	√	-	√	√
16	12V	√	-	-	-
17	12V	√	-	-	-

Multiplexed Forearm I/O Interface Description

Attached figure 17 shows the peripheral interface on the Forearm, Attached table 63 lists the multiplexed I/O description.



Attached figure 17 Peripheral interface in the Forearm

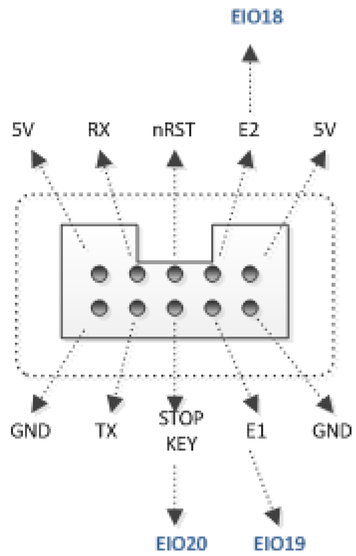
Attached table 63 Multiplexed I/O description

I/O addressing	Voltage	Level Output	PWM	Level Input	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	√	-	√	√

Appendix F Multiplexed I/O Interface Description of V2 Dobot Magician

Multiplexed UART Interface Description

Attached figure 18 shows the UART interface on the base, Attached table 64 lists the multiplexed I/O description.



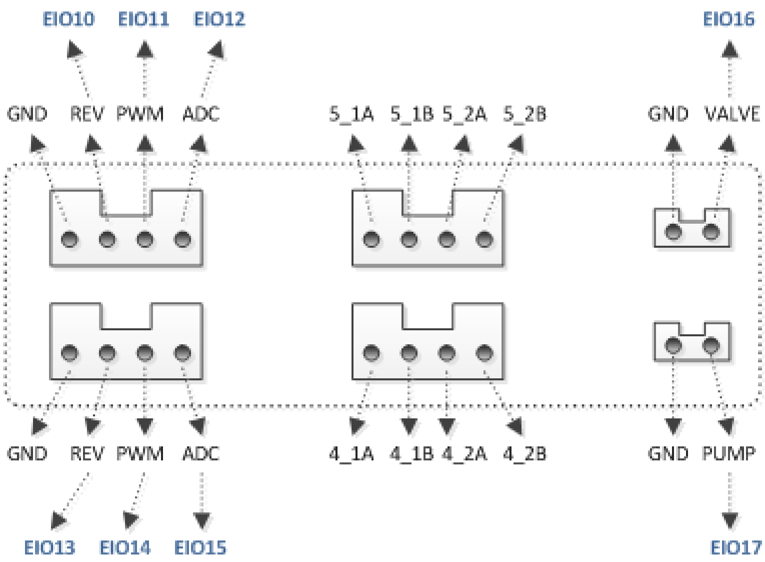
Attached figure 18 UART interface

Attached table 64 Multiplex I/O Description

I/O addressing	Voltage	Level Output	PWM	Level Input	ADC
18	3.3V	√	-	-	-
19	3.3V	-	-	√	-
20	3.3V	-	-	√	-

Multiplexed Peripheral Interface Description

Attached figure 19 shows the peripheral interface on the base, and Attached table 65 lists the multiplexed I/O description.



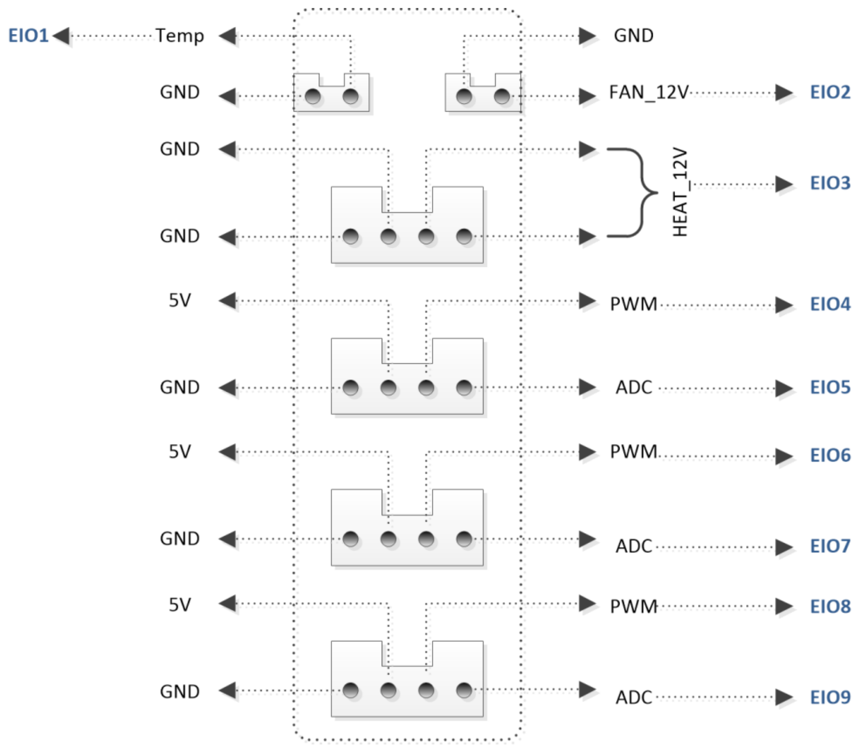
Attached figure 19 Peripheral Interface

Attached table 65 Multiplexed I/O Description

I/O addressing	Voltage	Level Output	PWM	Level Input	ADC
10	5V	√	-	-	-
11	3.3V	√	√	-	-
12	3.3V	-	-	√	-
13	5V	√	-	-	-
14	3.3V	√	√	√	-
15	3.3V	√	-	√	√
16	12V	√	-	-	-
17	12V	√	-	-	-

Multiplexed Forearm I/O Interface Description

Attached figure 20 shows the peripheral interface on the Forearm, Attached table 63 lists the multiplexed I/O description.



Attached figure 20 Peripheral interface in the Forearm

Attached table 66 Multiplexed I/O description

I/O addressing	Voltage	Level Output	PWM	Level Input	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	-	-	√	√