

**工程实践与劳动教育报告**



题 目 树莓派智能陪伴型机械臂

成员姓名 王珂、刘思颖、杨慕云、张煦涵

班 级 软外2202班

教师评语：

|  |
| --- |
| **教师评语及成绩：**  **不跨到下一页。先对全组进行整体评价（含作品主要功能、亮点和不足等），然后对各个成员的表现分别进行评价。  作品功能点、创新性、技术含金量、参与度、理解深度、Debug能力。** |

团队分工与贡献度说明

|  |  |  |  |
| --- | --- | --- | --- |
| 成员姓名 | 主要分工 | 总贡献度（%） | 备注 |
| 王珂 | 1. 完成了总体环境搭建； 2. 完成了项目的整体构思与设计； 3. 完成了项目汇总和调试； 4. 完成了语音识别的业务逻辑实现； 5. 完成了语音播报模块的业务逻辑实现； 6. 完成了报告的特色模块撰写。 | 28% | 组长 |
| 刘思颖 | 1. 完成了故事（二维码识别）模块的业务逻辑实现； 2. 完成了弹琴模块的业务逻辑实现； 3. 完成了飞行棋和弹琴模块机械臂控制的业务逻辑实现； 4. 完成了飞行棋和弹琴模块的场景搭建和方位确认； 5. 完成了报告的主要模块撰写。 | 24% | 组员 |
| 杨慕云 | 1. 完成了飞行棋的业务逻辑实现； 2. 完成了舞蹈（手势识别）模块的业务逻辑实现； 3. 完成了机械臂舞蹈动作的设计与实现； 4. 完成了ppt的制作。 5. 完成了报告的部分模块撰写。 | 24% | 组员 |
| 张煦涵 | 1. 完成了飞行棋和弹琴模块机械臂控制的业务逻辑实现； 2. 完成了飞行棋和弹琴模块的场景搭建和方位确认； 3. 完成了颜色识别业务逻辑实现； 4. 完成了视频剪辑。 | 24% | 组员 |

# **一、系统简介、主要功能、特色与亮点**

## 1.1 系统简介

1. 应用场景

随着人们对于人机交互和机器人技术日益增长的兴趣和需求，机械臂不仅可以作为一个功能强大的工具，执行例如搬运、组装或检查等任务，更重要的是，它还可以作为一个陪伴者存在。在家庭、医疗或者办公场所，它能够提供智能化的支持和互动，增强人们的生活质量和工作效率。故我们组参考了市面上的一些陪伴型机器人，推出了我们自己设计的树莓派和机械臂结合的孩童陪伴型机械臂——小花，这样我们塑造了一个温馨的家庭场景，机械臂作为家庭中的成员之一，可以与家庭成员互动，起到很好的陪伴的作用。

1. 业务逻辑

当孩子需要陪伴时，即可启动我们的智能陪伴型机械臂，机械臂通过识别语音识别孩子想要玩耍的内容，对应来陪孩子互动玩耍。其中机械臂的语音互动识别可以对应有以下几个内容，分别为‘游戏’中‘观战’对应机械臂自己和自己玩飞行棋而‘参战’对应机械臂与孩子一起玩飞行棋、‘故事’对应讲冷笑话、‘天气’对应播报今天的天气、‘演奏’对应机械臂控制方位来弹八色琴、‘跳舞’对应机械臂识别手势来跳不同的舞蹈。

其中，飞行棋模块具体流程如下图1.1.1。图1.1.1的地图分为两个路线，两个起始位置和一个骰子识别位置，共两个玩家参与游戏。如果是‘观战’则两个玩家都为机械臂，开始时机械臂投掷骰子，通过识别骰子颜色来确认点数，直到投到6才可以从起始位置移到路线的起点，如果两轮都没有投到6则直接在第三次投掷时移到路线的起点，哪个玩家先到终点则哪个玩家胜利。如果是‘参战’则一个玩家为孩子，一个玩家为机械臂，规则与‘观战’一致，只是在识别骰子的颜色后加了播报点数方便行走。



图1.1.1 飞行棋地图

故事模块为机械臂通过蜂鸣器长鸣提示可以开始识别二维码，短鸣提示识别中，识别二维码通过调用python的pyzbar库，如果识别成功则对应语音播报讲识别到的数字对应的冷笑话，如果失败则会长鸣提示再次开始识别，直到识别成功为止。

跳舞模块通过调用百度手势识别的api来识别不同的手势，通过识别出的手势来语音播报识别结果并控制机械臂跳对应的不同的舞蹈。

演奏模块则是通过对应测出八音琴每个琴音的相应位置和机械臂抓取的相应位置，机械臂长鸣提示开始抓取琴棒，抓取成功后开始读取琴谱来控制机械臂敲击不同的琴键，这样完成一首优美的曲子，完成后将琴棒放回。

天气模块则是使用爬虫来抓取当天的天气来播报。

## 1.2 系统主要功能

1. 二维码识别功能：扫描二维码并获取二维码中的信息作为不同冷笑话的编号；
2. 手势识别功能：识别手势并解析其中的对应的信息作为不同舞蹈的编号；
3. 颜色识别功能：识别不同的颜色作为骰子的不同代表点数；
4. 机械臂控制功能：控制机械臂的正常移动、抓取物品、舞蹈动作和敲击；
5. 语音播报功能：播报需要的相应内容；
6. 语音识别功能：识别不同的内容对应不同的互动方式；
7. 爬虫功能：爬取当天的天气。

## 1.3 特色与亮点

1. 使用opencv处理基础图像并且调用百度api，利用百度现有模型进行手势识别不同的信息；
2. 使用opencv处理相关基础图片，并且利用pyzbar库中函数扫描二维码获取信息；
3. 使用爬虫功能来爬取需要的信息；
4. 飞行棋的特色思路和棋盘设计与制作；
5. 弹琴的特色思路，有趣且适合孩童。

# 二、系统硬件设计与实现

## 2.1 系统硬件总体结构

2.1.1 树莓派主板

树莓派由注册于英国的慈善组织“Raspberry Pi 基金会”开发，Eben·Upton/埃·厄普顿为项目带头人。2012年3月，英国剑桥大学埃本·阿普顿（Eben Epton）正式发售世界上最小的台式机，又称卡片式电脑，外形只有信用卡大小，却具有电脑的所有基本功能，这就是Raspberry Pi电脑板，中文译名"树莓派"。

自问世以来，受众多计算机发烧友和创客的追捧，曾经一“派”难求。别看其外表“娇小”，内“心”却很强大，视频、音频等功能通通皆有，可谓是“麻雀虽小，五脏俱全”。主板图如图2.1.1所示：

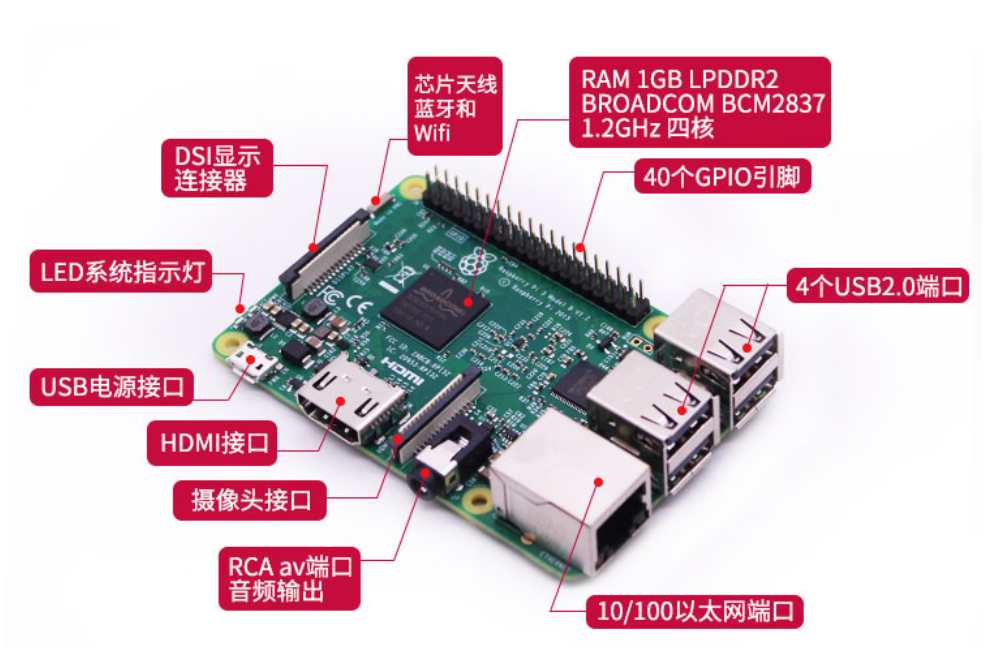


图2.1.1 树莓派3B+主控板

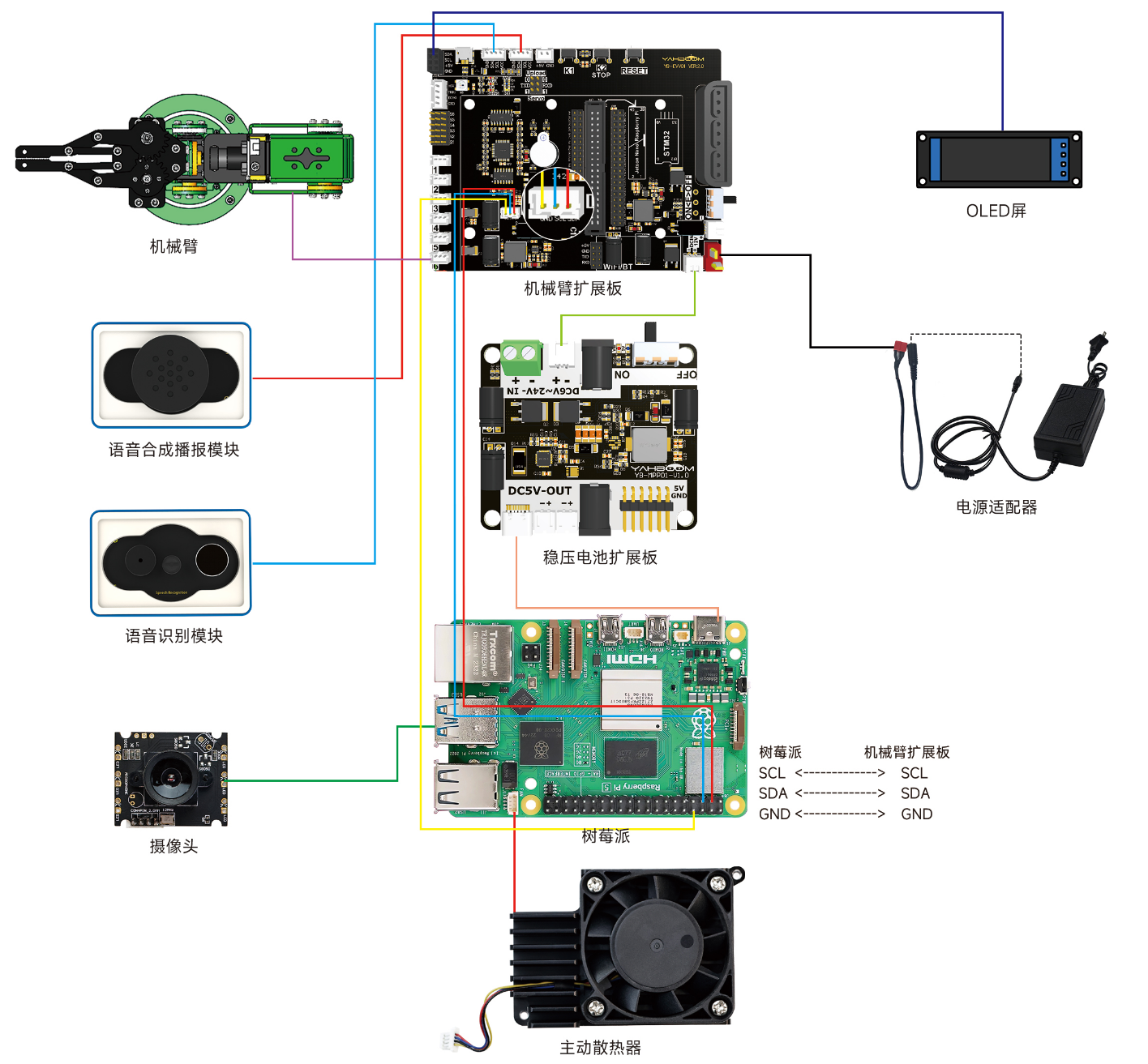


图2.1.2 树莓派接线示意图

## 2.2 系统核心模块

**2.2.1 机械臂模块**

机械臂模块是 DOFBOT AI 视觉机械臂系统的核心部分，负责实现机械臂的运动和执行各种任务。它通过结合人工智能和视觉感知技术，具备了自主感知和决策能力。它可以通过图像处理和深度学习算法实现目标检测、物体跟踪、路径规划等高级功能，从而适应不同的应用场景，如装配、拾取放置、质检等。这种机械臂模块的灵活性和智能化使其成为现代工业自动化和机器人应用领域的重要工具。机械臂模块如图2.2.1和2.2.2所示。

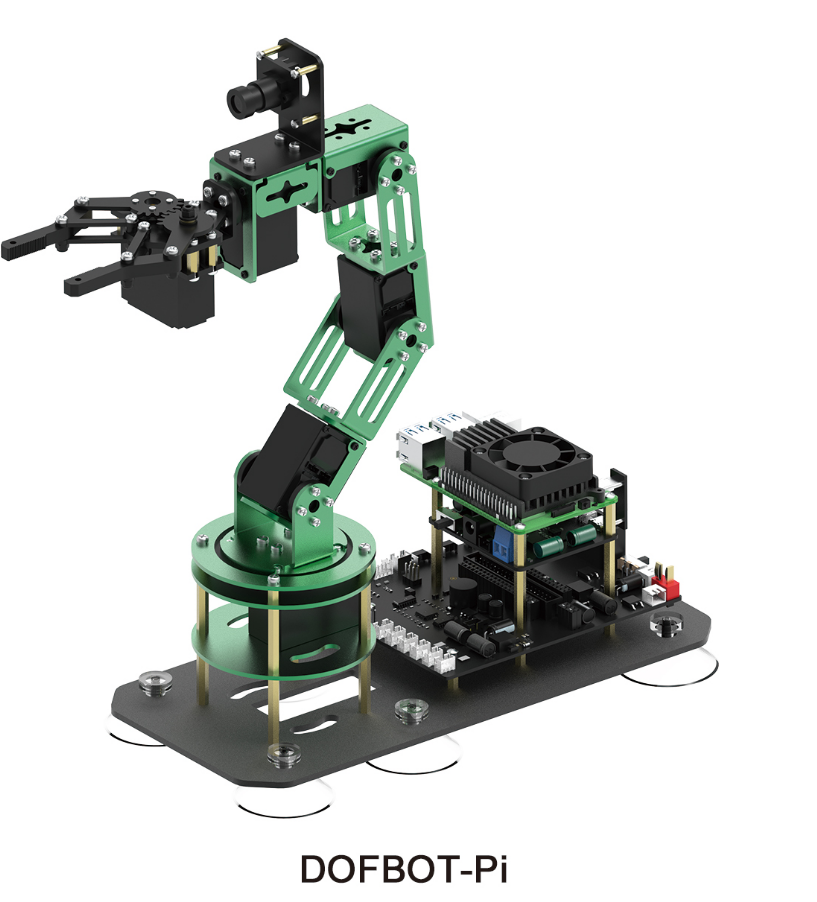


图2.2.1 树莓派机械臂展示图

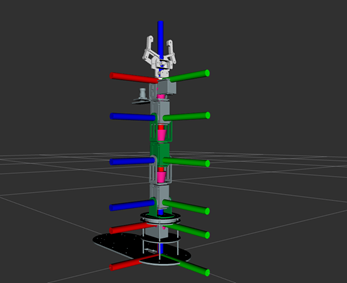


图2.2.2 树莓派机械臂结构图

**2.2.2 语音识别模块介绍**

语音识别模块是基于LD3320和内部MCU，该模块通过内部MCU对LD3320进行初始化配置，用户可以直接通过I2C录入识别词条和词条对应的序号，设置模块的模式（循环检测，口令触发，按键触发），获取识别到的结果，无需了解内部处理即可直接使用，且模块上还集成了蜂鸣器和RGB灯可通过I2C控制RGB灯的颜色和设置识别提示声的开关。

模块集成了LD3320，无需用户训练和录入语音，通过录入对应的识别词语的拼音和序号，当识别到对应的词语的时会在结果寄存器里面更新为识别给过对应的序号。例如1“hong deng”，当识别到红灯这个词时候会将1更新到结果寄存器，也就是录入对应的词语的拼音，注意这个拼音是没有音调的。模块最多可以设置50个识别词条，识别结果可以是单字，词组，短句，长度不能超过10个汉字或者79个字节的拼音字符串，通讯方式为I2C通讯。模块有三种模式，循环模式模块时刻处于语音识别状态，口令模式当检测到口令时候进行语音识别，按键模式当按下按键后处于语音识别模式，模块没有设置板载按键，通过主控板设置按键检测程序，检测到按键后通过I2C发送按键命令触发识别，该按键模式可以有客户定义任意的自己用的按键。

模块的识别原理非特定人语音识别技术ASR，是一个基于关键词语列表的匹配识别技术算法本质是在提取输入声音的特征后，在关键词语列表中寻找一个相似度最高。因此，任何的声音输入进语音识别芯片，都会去和关键词语列表中的词语进行匹配对比，并且也都会依次打分。这有时候可能会造成一些错误识别。通过借助编程技巧中的增加垃圾词吸收可以有效消除错误识别，在一些准确性场景比较高的场景中需经过大量测试准确性和稳定性符合要求再应用到实际中，以免错误识别造成不可挽回的事故。



图2.2.3 语音识别模块

**2.2.3 语音播报模块介绍**

语音播报模块是基于XFS5152CE和内部MCU，该模块通过内部MCU对XFS5152CE进行配置，芯片支持任意中文、英文文本的合成，可以采用 GB2312、GBK、BIG5 和 UNICODE 四种编码方式。每次合成的文本量最多可达4K字节。模块对文本进行分析，对常见的数字、号码、时间、日期、度量衡符号等格式的文本，芯片能够根据内置的文本匹配规则进行正确的识别和处理；对一般多音字也可以依据其语境正确判断读法；另外针对同时有中文和英文的文本，可实现中英文混读。支持多种控制命令如合成文本、停止合成、暂停合成、恢复合成、状态查询等。可设置对应字母的播报方式，如字母发音、单词发音等，还可以通过设置通过汉语拼音进行汉语播报。内置多个发音人物，可以通过设置选择对应的发音人进行语音播放，语音播报的语速、语调、音量等均可以设置。模块还有其他如数字符号等的特殊播报设置。通过I2C通讯对模块进行设置，按照规定的帧进行设置即可，操作方便快捷。



图2.2.4 语音播报模块

# 系统软件设计

## 3.1 系统软件总体设计思路

我们讲机械臂的功能分为5个模块，分别为飞行棋、跳舞、故事、天气和演奏模块。机械臂通过语音识别来判断当前属于什么模块要完成什么功能。

其中游戏模块分为观战与参战，在飞行棋游戏中，机械臂可以独立完成游戏（观战模式），或者与孩子一起玩（参战模式）。游戏流程为通过投掷骰子、识别点数、移动棋子，直到决出胜者。

故事模块为机械臂通过蜂鸣器提示用户扫描二维码，成功识别后讲述对应的冷笑话。

跳舞模块为通过百度手势识别API，机械臂可以识别孩子的手势，并跳出相应的舞蹈。

演奏模块为机械臂通过精确控制，可以抓取琴棒并按照琴谱敲击八音琴，演奏出美妙的音乐。

天气模块为通过爬虫技术，机械臂可以实时获取并播报当天的天气情况。

总体运行流程图如图3.1.1所示，其中分别流程图如图3.2.1，图3.2.2.1，图3.2.3，图3.2.4，图3.2.5所示。

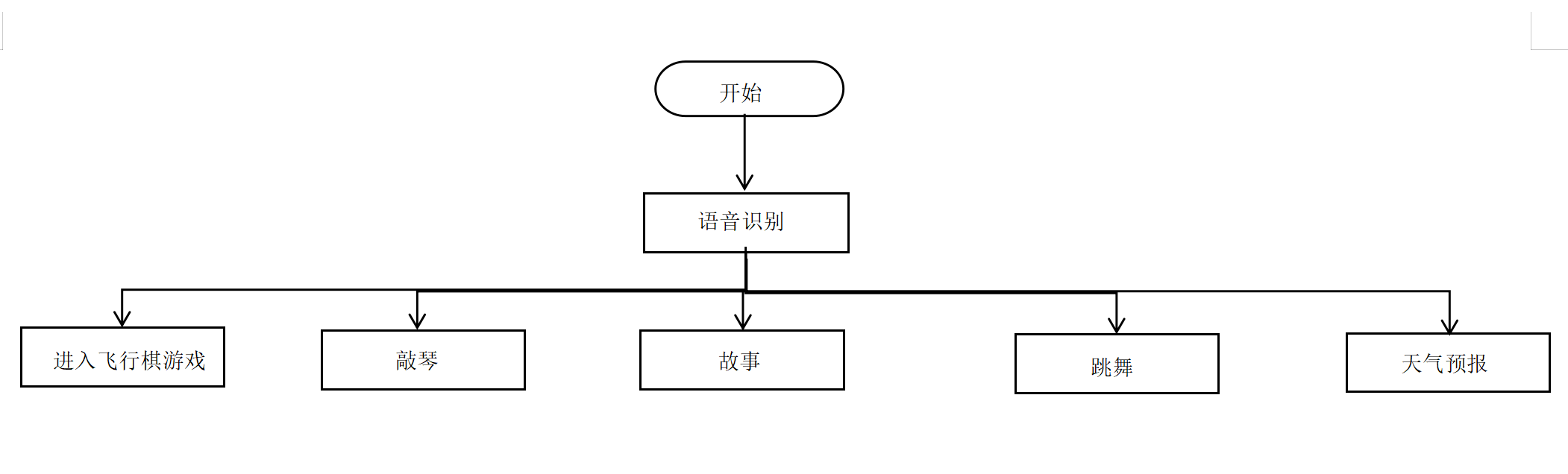


图3.1.1 总体流程图

## 3.2 系统核心功能实现

我们在树莓派平台上完成了手势识别，二维码识别等相关的环境搭建，具体的语句为：

1. pip install opencv
2. pip install pyzbar
3. pip install baidu-aip

结合树莓派平台上的现有包，我们完成了整个流程的实现。

### **3.2.1飞行棋游戏模块**

### **3.2.1.1逻辑**

该模块分为机器对战模式和人机对战模式，两者都需要投掷骰子来确定玩家的步数。开始游戏时，两位玩家的棋子都被放在“家”中，直到投掷到6才能放到开始位置。同时，为了游戏更好的进行，我们设置了保底机制，如果一位玩家第三次还没有投到6则直接把棋子放到开始位置。

在人机对战模式中，机械臂会对人类玩家投掷的骰子进行识别并播报点数。最终游戏的输赢也会对应机械臂不同的舞蹈姿态。

如下图3.2.1为流程图。

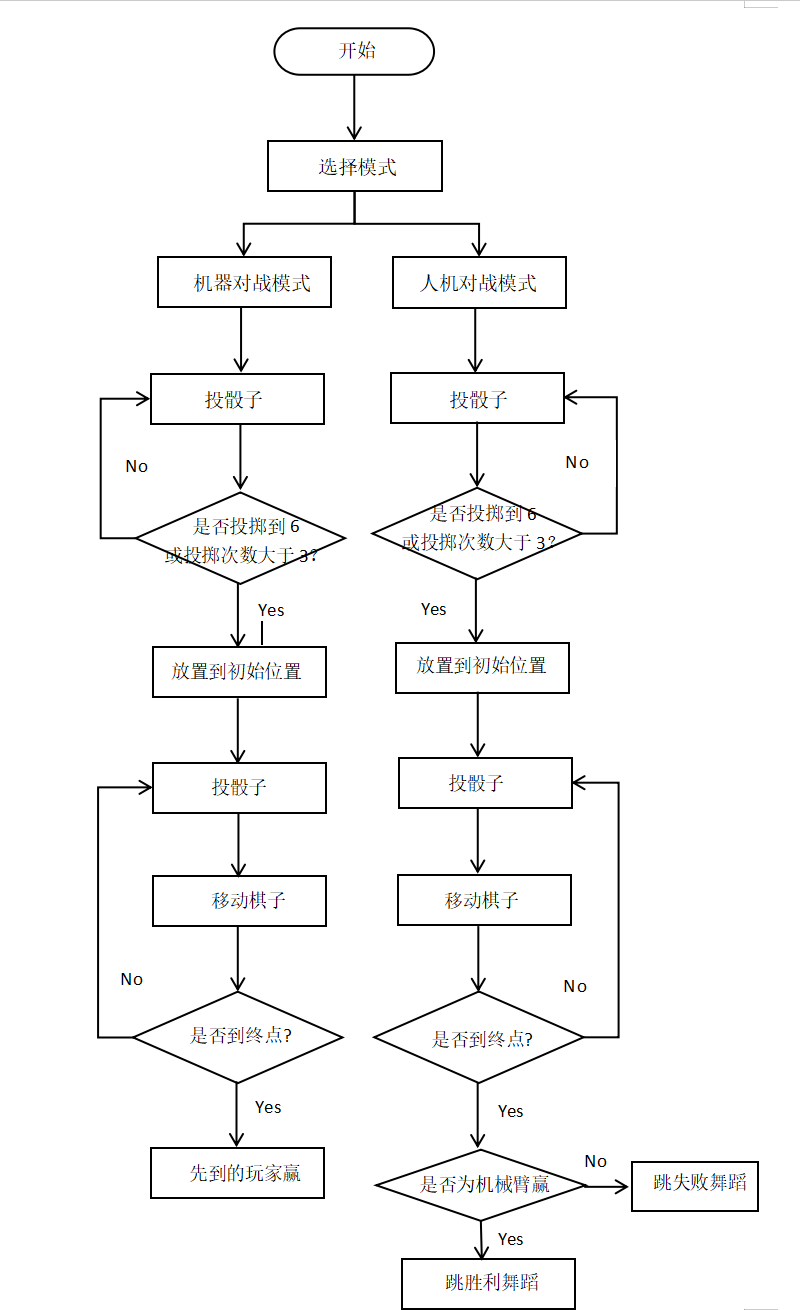


图3.2.1 飞行棋流程图

1. def move\_player(self, player, steps):
2. if player == 1:
3. if self.player1\_position == -1:  *# 棋子未在棋盘上*
4. if steps == 6 or self.dice\_count1 == 2:
5. self.player1\_position = 0  *# 将棋子放到初始位置*
6. self.position\_move0\_1(self.p1\_1, self.p1)
7. print(f"Player {player} places the piece at the starting position.")
8. self.dice\_count1 = 0
9. print(f"the dice\_number is dice\_count1={self.dice\_count1}")
10. else:
11. print(f"Player {player} needs to roll a 6 to start.")
12. print(f"the dice\_number is dice\_count1={self.dice\_count1}")
13. self.dice\_count1 += 1
14. print(f"the dice\_number is dice\_count1={self.dice\_count1}")
15. return False
16. else:
17. old\_player1\_position = self.player1\_position
18. self.player1\_position += steps
19. if self.player1\_position > self.board\_length:
20. steps = self.board\_length - old\_player1\_position
21. self.player1\_position = self.board\_length
22. print(f"Player {player} moves to position {self.player1\_position}")
23. self.position\_move2(self.p1, steps)
24. if self.player1\_position == self.board\_length:
25. b\_time = 3
26. self.Arm.Arm\_Buzzer\_On(b\_time)
27. print(f"Player {player} wins!")
28. return True
29. elif player == 2:
30. if self.player2\_position == -1:  *# 棋子未在棋盘上*
31. if steps == 6 or self.dice\_count2 == 2:
32. self.player2\_position = 0  *# 将棋子放到初始位置*
33. self.position\_move0\_2(self.p2\_2, self.p2)
34. print(f"Player {player} places the piece at the starting position.")
35. self.dice\_count2 = 0
36. print(f"the dice\_number is dice\_count2={self.dice\_count2}")
37. else:
38. print(f"Player {player} needs to roll a 6 to start.")
39. print(f"the dice\_number is dice\_count2={self.dice\_count2}")
40. self.dice\_count2 += 1
41. print(f"the dice\_number is dice\_count2={self.dice\_count2}")
42. return False
43. else:
44. old\_player2\_position = self.player2\_position
45. self.player2\_position += steps
46. if self.player2\_position > self.board\_length:
47. steps = self.board\_length - old\_player2\_position
48. self.player2\_position = self.board\_length
49. print(f"Player {player} moves to position {self.player2\_position}")
50. self.position\_move2(self.p2, steps)
51. if self.player2\_position == self.board\_length:
52. b\_time = 3
53. self.Arm.Arm\_Buzzer\_On(b\_time)
54. print(f"Player {player} wins!")
55. return True
56. else:
57. print(f"Unknown player: {player}")
58. return False
59. return False

机器对战游戏逻辑主要代码

1. def move\_player(self, player, steps):
2. if player == 1:
3. if self.player1\_position == -1:  *# Piece not on board*
4. if steps == 6 or self.dice\_count1 == 2:
5. self.player1\_position = 0  *# Place piece at start position*
6. self.position\_move0\_1(self.p1\_1, self.p1)
7. print(f"Player {player} places the piece at the starting position.")
8. self.dice\_count1 = 0
9. print(f"Dice count for Player 1: {self.dice\_count1}")
10. else:
11. print(f"Player {player} needs to roll a 6 to start.")
12. self.dice\_count1 += 1
13. print(f"Dice count for Player 1: {self.dice\_count1}")
14. return False
15. else:
16. old\_player1\_position = self.player1\_position
17. self.player1\_position += steps
18. if self.player1\_position > self.board\_length:
19. steps = self.board\_length - old\_player1\_position
20. self.player1\_position = self.board\_length
21. print(f"Player {player} moves to position {self.player1\_position}")
22. self.position\_move2(self.p1, steps)
23. if self.player1\_position == self.board\_length:
24. print(f"Player {player} wins!")
25. self.win()
26. b\_time = 6
27. self.Arm.Arm\_Buzzer\_On(b\_time)
28. return True
29. elif player == 2:
30. print("Player 2's turn (human player):")
31. print("Rolling dice...")
32. b\_time = 7
33. self.Arm.Arm\_Buzzer\_On(b\_time)
34. time.sleep(3)
35. p\_vidio = [90, 78, 24, 13, 90]
36. self.arm\_move(p\_vidio, 1000)
37. time.sleep(3)
38. steps = self.getdice\_num()
39. while steps == -1:
40. print("Retrying dice throw...")
41. steps = self.getdice\_num()
42. print(f"Player {player} rolls {steps}")
43. if self.player2\_position == -1:  *# Piece not on board*
44. if steps == 6 or self.dice\_count2 == 2:
45. self.player2\_position = 0  *# Place piece at start position*
46. print(f"Player {player} places the piece at the starting position.")
47. self.dice\_count2 = 0
48. print(f"Dice count for Player 2: {self.dice\_count2}")
49. else:
50. print(f"Player {player} needs to roll a 6 to start.")
51. self.dice\_count2 += 1
52. print(f"Dice count for Player 2: {self.dice\_count2}")
53. return False
54. else:
55. old\_player2\_position = self.player2\_position
56. self.player2\_position += steps
57. if self.player2\_position > self.board\_length:
58. steps = self.board\_length - old\_player2\_position
59. self.player2\_position = self.board\_length
60. print(f"Player {player} moves to position {self.player2\_position}")
61. if self.player2\_position == self.board\_length:
62. print(f"Player {player} wins!")
63. self.fail()
64. b\_time = 3
65. self.Arm.Arm\_Buzzer\_On(b\_time)
66. return True
67. else:
68. print(f"Unknown player: {player}")
69. return False
70. return False

人机对战游戏逻辑主要代码

### **3.2.1.2模块功能**

该模块包含了语音播报、颜色识别、机械臂控制功能。

### **3.2.1.2.1语音播报模块**

语音播报模块是基于XFS5152CE和内部MCU，对播报内容、播报方式等进行设置。

1. Reader\_Type = {
2. 'Reader\_XiaoYan': 3,
3. 'Reader\_XuJiu': 51,
4. 'Reader\_XuDuo': 52,
5. 'Reader\_XiaoPing': 53,
6. 'Reader\_DonaldDuck': 54,
7. 'Reader\_XuXiaoBao': 55
8. }
9. def person\_speak(self, num):
10. myspeak = speak()
11. if num == 1:
12. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])
13. myspeak.Speech\_text("请小主人走1步", myspeak.EncodingFormat\_Type["GB2312"])
14. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:
15. time.sleep(0.1)
16. elif num == 2:
17. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])
18. myspeak.Speech\_text("请小主人走2步", myspeak.EncodingFormat\_Type["GB2312"])
19. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:
20. time.sleep(0.1)
21. elif num == 3:
22. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])
23. myspeak.Speech\_text("请小主人走3步", myspeak.EncodingFormat\_Type["GB2312"])
24. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:
25. time.sleep(0.1)
26. elif num == 4:
27. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])
28. myspeak.Speech\_text("请小主人走4步", myspeak.EncodingFormat\_Type["GB2312"])
29. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:
30. time.sleep(0.1)
31. elif num == 5:
32. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])
33. myspeak.Speech\_text("请小主人走5步", myspeak.EncodingFormat\_Type["GB2312"])
34. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:
35. time.sleep(0.1)
36. elif num == 6:
37. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])
38. myspeak.Speech\_text("请小主人走6步", myspeak.EncodingFormat\_Type["GB2312"])
39. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:
40. time.sleep(0.1)

语音播报主要代码

### **3.2.1.2.2颜色识别模块**

颜色识别通过对不同颜色的hsv区间设置来判断颜色，并设置了识别的范围，能够更精确的识别出需要识别的物体的颜色。

1. def getdice\_num(self):
2. cap=self.cap
3. if not cap.isOpened():
4. print("Error: Could not open camera.")
5. return -1
6. ret, frame = cap.read()
7. cv2.imshow("image", frame)
8. if not ret:
9. print("Error: Could not read frame.")
10. return -1
11. height, width, \_ = frame.shape
12. center\_size = 40
13. x\_start = width // 2 - center\_size // 2
14. y\_start = height // 2 - center\_size // 2
15. x\_end = x\_start + center\_size
16. y\_end = y\_start + center\_size
17. center\_region = frame[y\_start:y\_end, x\_start:x\_end]
18. print(f"center\_region: {center\_region}")
19. hsv = cv2.cvtColor(center\_region, cv2.COLOR\_BGR2HSV)
20. color\_ranges = {
21. 1: [(0, 126, 191), (181, 253, 255)],  *# Red*
22. 2: [(104, 150, 160), (113, 253, 255)],  *# Blue*
23. 3: [(57, 76, 111), (75, 253, 255)],  *# Green*
24. 4: [(28, 63, 240), (32, 253, 255)],  *# Yellow*
25. 5: [(0, 0, 0), (120, 218, 65)],  *# Black*
26. 6: [(80, 5, 100), (125, 200, 255)]  *# White*
27. }
28. max\_pixels = 0
29. detected\_dice\_num = -1
30. mapping = {
31. 1: "red",
32. 2: "blue",
33. 3: "green",
34. 4: "yellow",
35. 5: "black",
36. 6: "white"
37. }
38. for dice\_num, ranges in color\_ranges.items():
39. mask = cv2.inRange(hsv, ranges[0], ranges[1])
40. pixels = cv2.countNonZero(mask)
41. detected\_hsv = cv2.bitwise\_and(hsv, hsv, mask=mask)
42. print(f"Detected HSV values: {detected\_hsv}")
43. if pixels > max\_pixels:
44. max\_pixels = pixels
45. detected\_dice\_num = dice\_num
46. if detected\_dice\_num == -1:
47. print("Camera error")
48. else:
49. print(f"The color is: {mapping[detected\_dice\_num]}, the number is: {detected\_dice\_num}")
50. return detected\_dice\_num

颜色识别主要代码

### **3.2.1.2.3机械臂控制模块**

通过对机械臂的6个舵机在各个角度的数值的定义来实现输和赢后的机械臂动作。

1. def win(self):
2. guiwei = [90, 90, 90, 90, 90, 90, 500]
3. times = 0.003
4. self.arm\_move(guiwei)
5. time.sleep(1)
6. self.Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
7. time.sleep(times)
8. self.Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
9. time.sleep(times)
10. self.Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
11. time.sleep(self.time\_sleep)
12. self.Arm.Arm\_serial\_servo\_write(2, 180 - 135, self.time\_1)
13. time.sleep(times)
14. self.Arm.Arm\_serial\_servo\_write(3, 135, self.time\_1)
15. time.sleep(times)
16. self.Arm.Arm\_serial\_servo\_write(4, 45, self.time\_1)
17. time.sleep(self.time\_sleep)
18. self.Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
19. time.sleep(times)
20. self.Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
21. time.sleep(times)
22. self.Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
23. time.sleep(self.time\_sleep)

机械臂控制部分代码

### **3.2.2手势识别跳舞模块**

### **3.2.2.1逻辑**

机械臂在识别玩家的不同手势后跳对应的舞蹈动作。

如下图3.2.2.1为流程图。

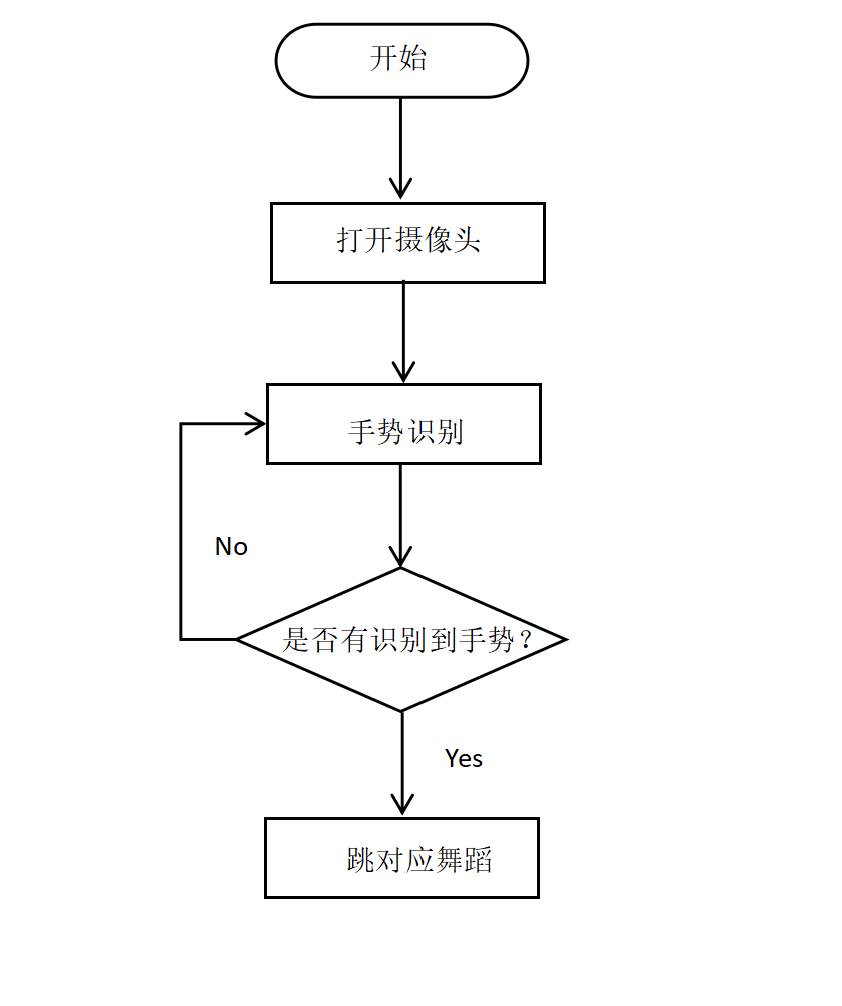


图3.2.2.1 跳舞流程图

### **3.2.2.2模块功能**

该模块包含了语音播报、手势识别、机械臂控制功能。

### **3.2.2.2.1手势识别功能**

该功能调用了百度API，通过摄像头并按帧率截取出多个JPEG图像，对图像进行识别检测，若有定义的手势则输出结果。如下图3.2.2.2为调用图。

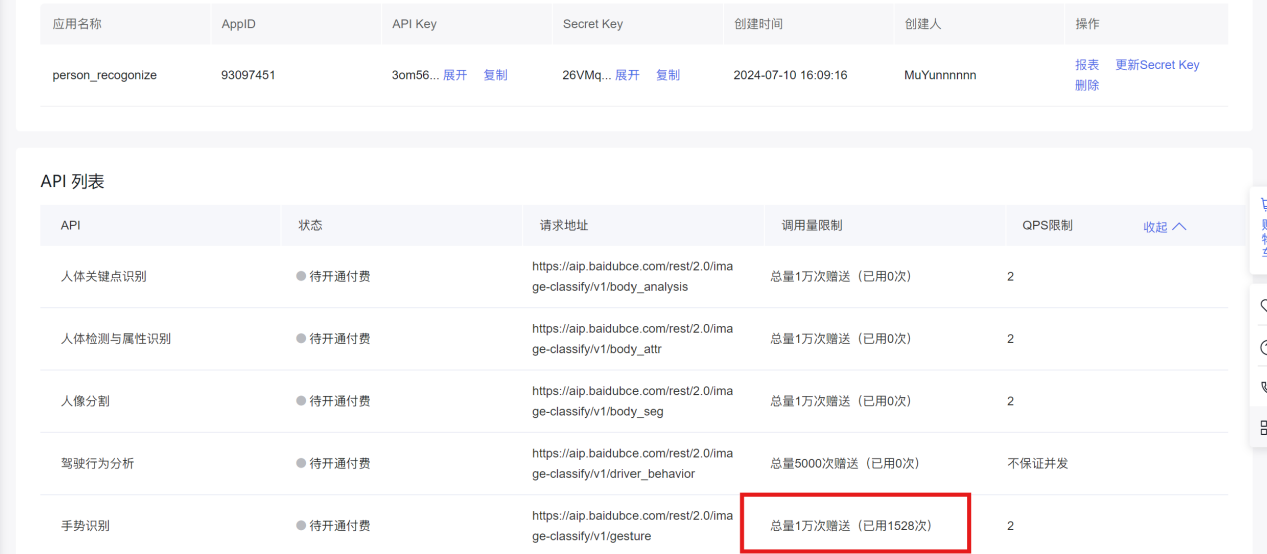


图3.2.2.2百度API调用

1. def \_\_init\_\_(self) -> None:
2. *# Replace with your own Baidu AI credentials*
3. APP\_ID = '93097451'
4. API\_KEY = '3om56y5ScOOn36Em2UJXbMZ4'
5. SECRET\_KEY = '26VMq5kQqjdfNfkalZVNq6pjdF5yYNcs'
6. self.client = AipBodyAnalysis(APP\_ID, API\_KEY, SECRET\_KEY)
7. self.hand = {'One': '数字 1', 'Two': '数字 2', 'Three': '数字 3', 'Four': '数字 4',
8. 'Five': '数字 5', 'Six': '数字 6', 'Seven': '数字 7',
9. 'Eight': '数字 8', 'Nine': '数字 9', 'Fist': '拳头', 'Ok': 'OK',
10. 'Prayer': '祈祷', 'Congratulation': '作揖', 'Honour': '作别',
11. 'Heart\_single': '比心心', 'Thumb\_up': '点赞', 'Thumb\_down': 'Diss',
12. 'ILY': '我爱你', 'Palm\_up': '掌心向上', 'Heart\_1': '双手比心 1',
13. 'Heart\_2': '双手比心 2', 'Heart\_3': '双手比心 3', 'Rock': 'Rock',
14. 'Insult': '竖中指', 'Face': '脸'}
15. def bgr8\_to\_jpeg(self, image):
16. image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)
17. \_, jpeg = cv2.imencode('.jpg', image\_rgb)
18. return jpeg.tobytes()
19. def cv2ImgAddText(self, img, text, left, top, textColor=(0, 255, 0), textSize=20):
20. if (isinstance(img, np.ndarray)):  *# 判断是否OpenCV图片类型*
21. img = Image.fromarray(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))
22. *# 创建一个可以在给定图像上绘图的对象*
23. draw = ImageDraw.Draw(img)
24. *# 字体的格式*
25. fontStyle = ImageFont.truetype(
26. "SettingsIcons.ttf", textSize, encoding="utf-8")
27. *# 绘制文本*
28. draw.text((left, top), text, textColor, font=fontStyle)
29. *# 转换回OpenCV格式*
30. return cv2.cvtColor(np.asarray(img), cv2.COLOR\_RGB2BGR)
31. def start(self):
32. myspeak = speak()
33. *#         g\_camera = cv2.VideoCapture(1)*
34. g\_camera = self.cap
35. *# 打开摄像头*
36. g\_camera.set(3, 640)
37. g\_camera.set(4, 480)
38. g\_camera.set(5, 30)  *# Frame rate*
39. g\_camera.set(cv2.CAP\_PROP\_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
40. g\_camera.set(cv2.CAP\_PROP\_BRIGHTNESS, 20)  *# Brightness*
41. g\_camera.set(cv2.CAP\_PROP\_CONTRAST, 50)  *# Contrast*
42. g\_camera.set(cv2.CAP\_PROP\_EXPOSURE, 100)  *# Exposure*
43. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
44. myspeak.Speech\_text("开始识别", myspeak.EncodingFormat\_Type["GB2312"])
45. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
46. time.sleep(0.1)
47. print("识别中")
48. time.sleep(3)
49. flag = True
50. try:
51. while flag:
52. Arm.Arm\_serial\_servo\_write6(90, 135, 20, 25, 90, 30, 1000)
53. ret, frame = g\_camera.read()
54. if not ret:
55. print("Error: Could not read frame.")
56. time.sleep(3)
57. self.image\_widget = widgets.Image(format='jpeg', width=600, height=500)
58. display(self.image\_widget)
59. frame\_jpeg = self.bgr8\_to\_jpeg(frame)
60. gesture\_result = self.client.gesture(frame\_jpeg)
61. raw\_result = str(gesture\_result)
62. parsed\_result = demjson.decode(raw\_result)
63. try:
64. res = parsed\_result['result'][0]['classname']
65. except:
66. print('识别结果：未能识别到手势')
67. img = frame
68. else:
69. print('识别结果：' + self.hand.get(res, '未知手势'))

手势识别主要代码

### **3.2.2.2.2机械臂控制功能**

通过对机械臂的6个舵机在各个角度的数值的定义来实现输和赢后的机械臂动作。

1. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_2)
2. time.sleep(self.times)
3. Arm.Arm\_serial\_servo\_write(1, 90, self.time\_1)
4. time.sleep(self.times)
5. Arm.Arm\_serial\_servo\_write(1, 90, self.time\_1)
6. time.sleep(self.times)
7. Arm.Arm\_serial\_servo\_write(5, 90, self.time\_1)
8. time.sleep(self.time\_sleep)
9. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
10. time.sleep(self.times)
11. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
12. time.sleep(self.time\_sleep)
13. Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
14. time.sleep(self.times)
15. Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
16. time.sleep(self.times)
17. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
18. time.sleep(self.time\_sleep)

机械部控制部分代码

**3.2.3二维码识别故事模块**

### **3.2.3.1逻辑**

机械臂通过蜂鸣器长鸣提示可以开始识别二维码，短鸣提示识别中，识别二维码通过调用python的pyzbar库，如果识别成功则对应语音播报讲识别到的数字对应的冷笑话，如果失败则会长鸣提示再次开始识别，直到识别成功为止。

如图3.2.3为流程图。

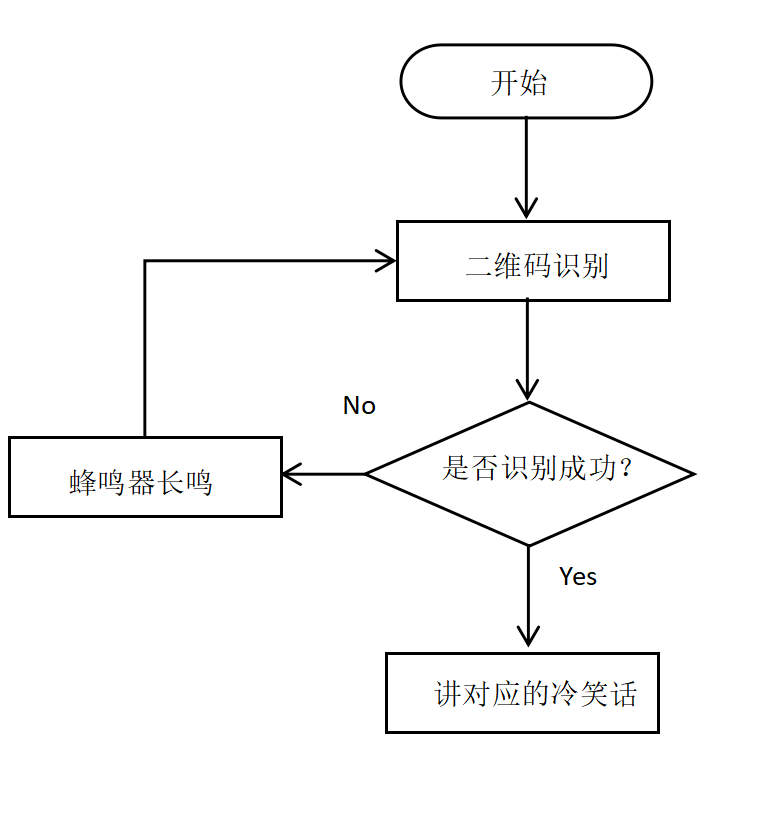


图3.2.3二维码讲故事流程图

### **3.2.3.2模块功能**

该模块包含了语音播报、二维码识别功能。

### **3.2.2.2.1二维码识别功能**

实时读取摄像头当前帧并进行处理，将处理的得到的图像使用pyzbar库进行解码。将二维码标注在图像上并输出到指定变量中以待后续使用。

1. import cv2
2. import pyzbar.pyzbar as pyzbar
3. from Arm\_Lib import Arm\_Device
4. import ipywidgets.widgets as widgets
5. import time
6. class QRCodeScanner:
7. def \_\_init\_\_(self):
8. self.arm = Arm\_Device()
9. time.sleep(0.1)
10. def arm\_move(self, p, s\_time=500):
11. for i in range(5):
12. id = i + 1
13. self.arm.Arm\_serial\_servo\_write(id, p[i], s\_time)
14. time.sleep(0.01)
15. time.sleep(s\_time / 1000)
16. def decode\_display(self, image, image1):
17. print('begin code')
18. barcodes = pyzbar.decode(image)
19. for barcode in barcodes:
20. (x, y, w, h) = barcode.rect
21. cv2.rectangle(image1, (x, y), (x + w, y + h), (0, 0, 255), 2)
22. barcodeData = barcode.data.decode("utf-8")
23. barcodeType = barcode.type
24. print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData))
25. if barcodeData == '1':
26. print('1 is ok')
27. elif barcodeData == '2':
28. print('2 is ok')
29. elif barcodeData == '3':
30. print('3 is ok')
31. elif barcodeData == '4':
32. print('4 is ok')
33. elif barcodeData == '5':
34. print('5 is ok')
35. else:
36. print('error')
37. return image1
38. def bgr8\_to\_jpeg(self, image):
39. image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)
40. \_, jpeg = cv2.imencode('.jpg', image\_rgb)
41. return jpeg.tobytes()
42. def capture\_qr\_codes(self):
43. p = [90, 135, 20, 25, 90, 30]
44. self.arm\_move(p, 500)
45. time.sleep(1)
46. cap = cv2.VideoCapture(1)
47. if not cap.isOpened():
48. print("Error: Could not open camera.")
49. return -1  *# 返回-1表示摄像头打开失败*
50. b\_time = 10
51. a\_time = 1
52. self.arm.Arm\_Buzzer\_On(b\_time)
53. time.sleep(10)
54. self.arm.Arm\_Buzzer\_On(a\_time)
55. print("begin photo")
56. while True:
57. *# 读取当前帧*
58. ret, img = cap.read()
59. *# 打开摄像头*
60. cap.set(3, 640)
61. cap.set(4, 480)
62. cap.set(5, 30)  *# Frame rate*
63. cap.set(cv2.CAP\_PROP\_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
64. cap.set(cv2.CAP\_PROP\_BRIGHTNESS, 40)  *# Brightness*
65. cap.set(cv2.CAP\_PROP\_CONTRAST, 50)  *# Contrast*
66. cap.set(cv2.CAP\_PROP\_EXPOSURE, 156)  *# Exposure*
67. *# Read the initial frame*
68. ret, frame = cap.read()
69. *# Display widget for camera feed*
70. image\_widget = widgets.Image(format='jpeg', width=600, height=500)
71. display(image\_widget)
72. image\_widget.value = self.bgr8\_to\_jpeg(frame)
73. b\_time = 10
74. self.arm.Arm\_Buzzer\_On(b\_time)
75. print('1')
76. if not ret:
77. print("Error: Could not read frame.")
78. *# 转为灰度图像*
79. print(f"Frame type: {type(frame)}, Frame shape: {frame.shape}")
80. gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)
81. *# 准备开始识别*
82. c\_time = 1
83. time.sleep(1)
84. self.arm.Arm\_Buzzer\_On(c\_time)
85. time.sleep(0.1)
86. self.arm.Arm\_Buzzer\_On(c\_time)
87. *# 进入识别*
88. im = self.decode\_display(gray, img)
89. cv2.imshow("image", im)
90. print(2)
91. key = cv2.waitKey(5)
92. print(3)
93. if key == 27:
94. break
95. try:
96. qr\_scanner = QRCodeScanner()
97. qr\_scanner.capture\_qr\_codes()
98. except KeyboardInterrupt:
99. print("\nProgram closed!")
100. finally:
101. time.sleep(1)

二维码识别核心代码

**3.2.4爬虫天气模块**

### **3.2.4.1逻辑**

使用爬虫来抓取当天的天气来播报。

如图3.2.4为流程图。

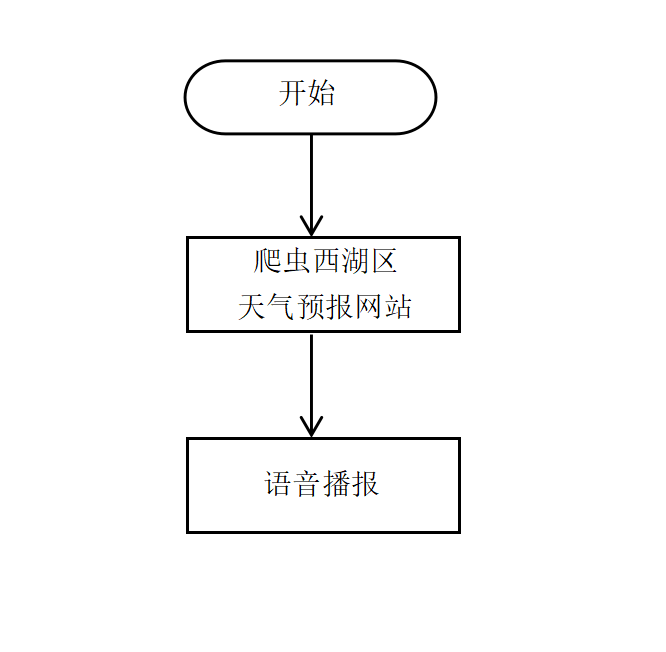


图3.2.4 天气流程图

### **3.2.4.2模块功能**

该模块包含了语音播报、爬虫功能。

### **3.2.2.2.1爬虫功能**

获取城市天气信息并返回文本格式。

1. def weather():
2. headers = {
3. 'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36',
4. }
5. res = requests.get('https://tianqi.moji.com/weather/china/zhejiang/xihu-district', headers=headers)
6. soup = BeautifulSoup(res.text, "html.parser")
7. temp = soup.find('div', class\_='wea\_weather clearfix').em.get\_text()
8. weather = soup.find('div', class\_='wea\_weather clearfix').b.get\_text()
9. sd = soup.find('div', class\_='wea\_about clearfix').span.get\_text()
10. sd\_num = re.search(r'\d+', sd).group()
11. wind = soup.find('div', class\_='wea\_about clearfix').em.get\_text()
12. aqi = soup.find('div', class\_='wea\_alert clearfix').em.get\_text()
13. aqi\_num = re.search(r'\d+', aqi).group()
14. info = soup.find('div', class\_='wea\_tips clearfix').em.get\_text()
15. *# 格式化湿度和空气质量指数*
16. sd = '湿度百分之%s' % sd\_num
17. aqi = '空气质量指数%s' % aqi\_num
18. *# 获取当前日期*
19. today = datetime.now().date().strftime('%Y年%m月%d日')
20. *# 构造返回的天气信息文本*
21. text = f'小主人好！今天是{today}，天气{weather}，温度{temp}摄氏度，{sd}，{wind}，{aqi}，{info}'
22. return text

爬虫核心代码

**3.2.5弹奏模块**

### **3.2.5.1逻辑**

通过对应测出八音琴每个琴音的相应位置和机械臂抓取的相应位置，机械臂长鸣提示开始抓取琴棒，抓取成功后开始读取琴谱来控制机械臂敲击不同的琴键，这样完成一首优美的曲子，完成后将琴棒放回。

如图3.2.5所示流程图。

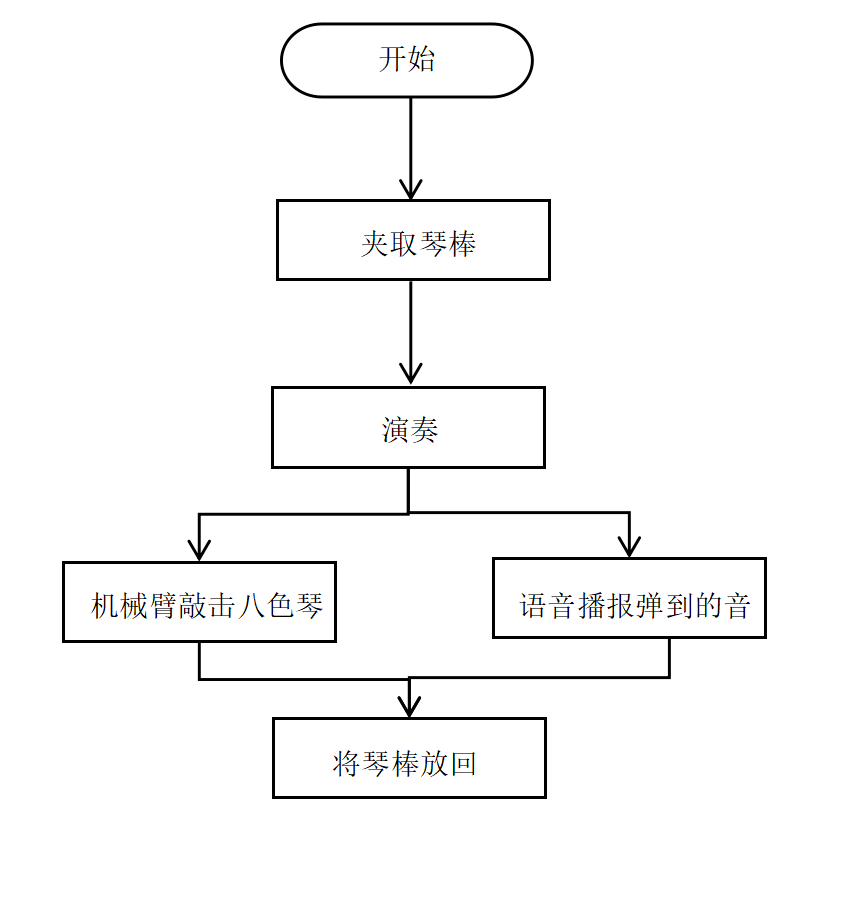


图3.2.5 演奏流程图

### **3.2.5.2模块功能**

该模块包含了语音播报、机械臂控制功能。

### **3.2.2.2.1**机械臂控制功能

通过确认八个琴音的位置，控制机械臂移动来敲击琴键。

1. *#!/usr/bin/env python3*
2. *#coding=utf-8*
3. import time
4. from Arm\_Lib import Arm\_Device
5. import smbus
6. import time
7. from speak import speak
8. class RoboticArmController:
9. def \_\_init\_\_(self):
10. self.arm = Arm\_Device()  *# Initialize the robotic arm*
11. self.init\_arm\_positions()  *# Initialize arm to default position*
12. self.time\_sleep = 0.5  *# Define default sleep time*
13. self.bus = smbus.SMBus(1)
14. self.seki2c\_addr = 0x30
15. self.date\_head = 0xfd
16. global i2c\_addr
17. *# Define arm positions for different notes*
18. self.positions = {
19. 0: [160, 140, 20, 25, 89],
20. 1: [63, 129, 25, 40, 180],
21. 2: [71, 138, 13, 40, 180],
22. 3: [77, 140, 13, 40, 180],
23. 4: [83, 148, 6, 40, 180],
24. 5: [89, 148, 5, 40, 180],
25. 6: [96, 148, 5, 40, 180],
26. 7: [100, 148, 5, 40, 180],
27. 8: [106, 148, 5, 40, 180],
28. }
29. Reader\_Type = {
30. 'Reader\_XiaoYan': 3,  *# 为 3，设置发音人为小燕(女声, 推荐发音人)*
31. 'Reader\_XuJiu': 51,  *# 为 51，设置发音人为许久(男声, 推荐发音人)*
32. 'Reader\_XuDuo': 52,  *# 为 52，设置发音人为许多(男声)*
33. 'Reader\_XiaoPing': 53,  *# 为 53，设置发音人为小萍(女声*
34. 'Reader\_DonaldDuck': 54,  *# 为 54，设置发音人为唐老鸭(效果器)*
35. 'Reader\_XuXiaoBao': 55  *# 为 55，设置发音人为许小宝(女童声)*
36. }  *# 选择发音人 [m?]'*
37. def init\_arm\_positions(self):
38. self.arm.Arm\_serial\_servo\_write6(90, 143, 20, 25, 89, 90, 500)
39. time.sleep(1)
40. def arm\_clamp\_block(self, enable):
41. if enable == 0:
42. self.arm.Arm\_serial\_servo\_write(6, 40, 400)
43. else:
44. self.arm.Arm\_serial\_servo\_write(6, 180, 400)
45. time.sleep(self.time\_sleep)
46. def arm\_move(self, p, s\_time=500):
47. for i in range(5):
48. id = i + 1
49. self.arm.Arm\_serial\_servo\_write(id, p[i], s\_time)
50. time.sleep(0.01)
51. time.sleep(s\_time / 1000)
52. def arm\_qiao(self, enable):
53. if enable == 0:
54. self.arm.Arm\_serial\_servo\_write(4, 30, 200)  *# Lift*
55. else:
56. self.arm.Arm\_serial\_servo\_write(4, 0, 100)  *# Strike*
57. time.sleep(self.time\_sleep)
58. def arm\_qiaoji(self):
59. time.sleep(0.01)
60. self.arm\_qiao(1)
61. time.sleep(1)
62. self.arm\_qiao(0)
63. time.sleep(0.01)
64. def arm\_return(self):
65. c\_time = 2
66. self.arm.Arm\_Buzzer\_On(c\_time)
67. self.arm\_move(self.positions[0], 500)
68. self.arm\_clamp\_block(0)
69. def person\_speak(self, num):
70. myspeak=speak()
71. if num == 1:
72. myspeak.SetReader(myspeak.Reader\_Type["Reader\_DonaldDuck"])  *# 选择播音人许多*
73. myspeak.Speech\_text("哆", myspeak.EncodingFormat\_Type["GB2312"])
74. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
75. time.sleep(0.1)
76. elif num == 2:
77. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XuJiu"])  *# 选择播音人许多*
78. myspeak.Speech\_text("来", myspeak.EncodingFormat\_Type["GB2312"])
79. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
80. time.sleep(0.1)
81. elif num == 3:
82. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XuDuo"])  *# 选择播音人许多*
83. myspeak.Speech\_text("咪", myspeak.EncodingFormat\_Type["GB2312"])
84. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
85. time.sleep(0.1)
86. elif num == 4:
87. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
88. myspeak.Speech\_text("发", myspeak.EncodingFormat\_Type["GB2312"])
89. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
90. time.sleep(0.1)
91. elif num == 5:
92. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
93. myspeak.Speech\_text("嗦", myspeak.EncodingFormat\_Type["GB2312"])
94. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
95. time.sleep(0.1)
96. elif num == 6:
97. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoPing"])  *# 选择播音人许多*
98. myspeak.Speech\_text("拉", myspeak.EncodingFormat\_Type["GB2312"])
99. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
100. time.sleep(0.1)
101. elif num == 7:
102. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoPing"])  *# 选择播音人许多*
103. myspeak.Speech\_text("西", myspeak.EncodingFormat\_Type["GB2312"])
104. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
105. time.sleep(0.1)
106. elif num == 8:
107. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XuXiaoBao"])  *# 选择播音人许多*
108. myspeak.Speech\_text("do", myspeak.EncodingFormat\_Type["GB2312"])
109. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
110. time.sleep(0.1)
111. def main\_game(self):
112. myspeak=speak()
113. try:
114. *# Initialize*
115. self.init\_arm\_positions()
116. *# Grab hammer*
117. b\_time = 10
118. self.arm.Arm\_Buzzer\_On(b\_time)
119. time.sleep(4)
120. a\_time = 3
121. self.arm.Arm\_Buzzer\_On(a\_time)
122. self.arm\_clamp\_block(1)
123. time.sleep(2)
124. self.arm.Arm\_serial\_servo\_write(5, 180, 400)
125. time.sleep(3)
126. *# Play musical sequence*
127. for note in q1:
128. if note in self.positions:
129. self.arm\_move(self.positions[note], 300)
130. time.sleep(0.01)
131. self.arm\_qiaoji()
132. self.person\_speak(note)
133. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
134. time.sleep(0.1)
135. print(note)
136. else:
137. print(f"Note {note} is not mapped to any position.")
138. time.sleep(0.01)
139. self.arm\_return()
140. except KeyboardInterrupt:
141. print("\nProgram closed!")
142. finally:
143. *# Reset arm to default position*
144. guiwei = [90, 90, 90, 90, 90, 90, 500]
145. self.arm\_move(guiwei, 1000)
146. time.sleep(1)
147. *# Clean up or disconnect arm if applicable*
148. *# self.arm.disconnect()  # Uncomment this line if there's a disconnect method*
149. *# Define your musical sequence*
150. q1 = [1, 1, 5, 5, 6, 6, 5, 4, 4, 3, 3, 2, 2, 1, 5, 5, 4, 4, 3, 3, 2, 5, 5, 4, 4, 3, 3, 2]
151. *# Main program execution*
152. if \_\_name\_\_ == "\_\_main\_\_":
153. arm\_controller = RoboticArmController()
154. arm\_controller.main\_game()

机械臂控制核心代码

# 四、系统执行结果与调试说明

## 4.1 系统执行结果

**4.1.1 智能陪伴型机械臂**

启动系统进入程序，机械臂首先会智能播报欢迎词“欢迎来到你的专属智能乐园，我是亲爱的小花。你想要做什么？”，用户通过语音输入想要进入的模块，机械臂进行语音识别关键词。

如果用户说“小花”，蜂鸣器就会响一声“滴”。

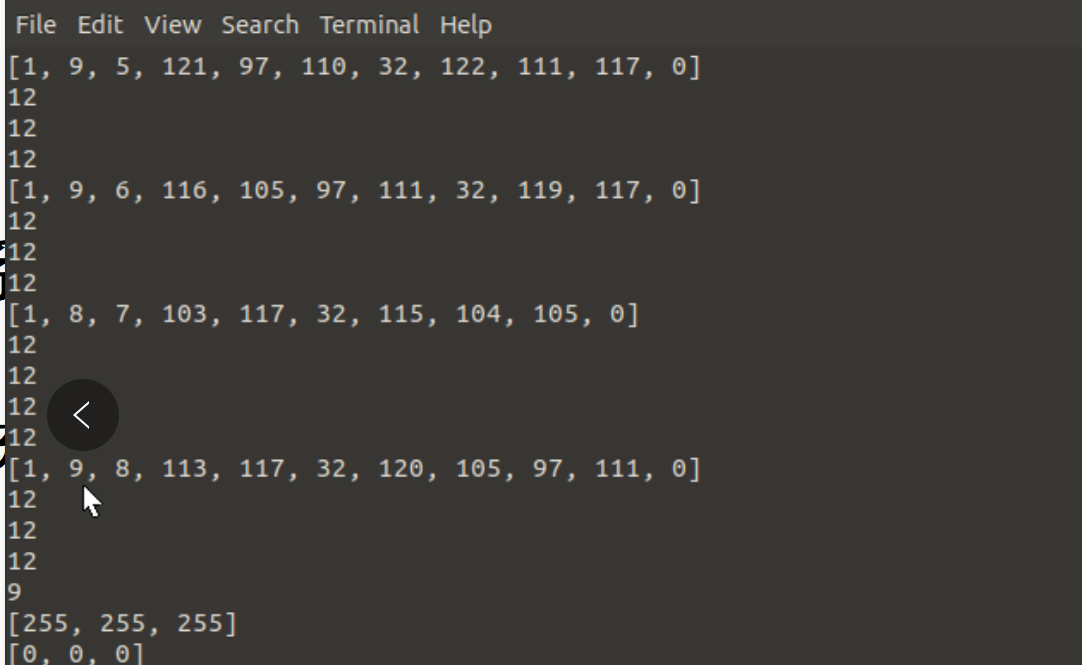


图4.1.1 系统完成准备，播报欢迎词代码运行截图

运行结果符合预期要求，控制台上输出了所需要的前期设备准备的反馈消息。

**4.1.2 天气预报模块**

用户对语音识别模块说一句：“天气”，语音播报模块就会发出提示音，然后软件就会利用爬虫的技术去爬取“墨迹天气”中西湖区当天的天气，并将数据填写到下列播报语句中进行播报，“小主人好！今天是{today}，天气{weather}，温度{temp}摄氏度，{sd}，{wind}，{aqi}，{info}”。具体的情况如下图所示：

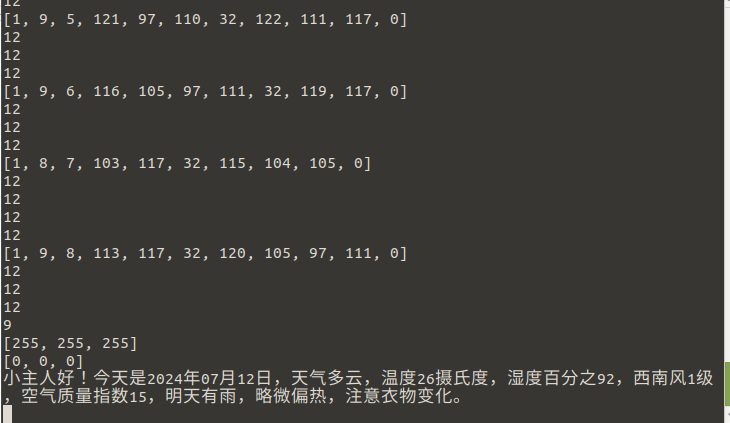


图4.1.2 天气预报功能代码运行结果截图

运行结果符合预期要求，控制台上输出了当天的正确天气信息。

**4.1.3 演奏模块**

用户对语音识别模块说一句：“演奏”，语音播报模块就会发出提示音，然后机械臂语音播报，"小花即将变成演奏大师，请小主人准备聆听"，进入演奏模块。

程序读取简谱，转化成位置信息传给机械臂，机械臂操纵琴锤完成演奏，并用不同的语音进行播报对应的音。

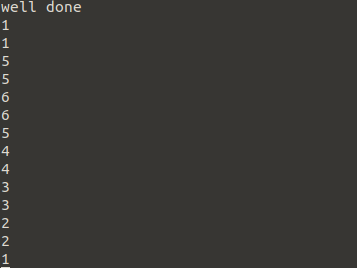


图4.1.3 演奏模块代码运行结果截图

运行结果符合预期要求，控制台上输出了所演奏曲子的简谱内容。

**4.1.4 冷笑话模块**

用户对语音识别模块说一句：“故事”，语音播报模块就会发出提示音，然后机械臂语音播报，"小花要开始说花花的笑话了，请小主人不要吃东西哦"，进入讲冷笑话模块。

用户需要向机械臂提供一张包含了数字（1-5）信息的二维码，机械臂会对该模块该二维码进行识别，读取内部信息，将信息回传。通过该信息匹配对应的笑话，传给机械臂语音播报模块。

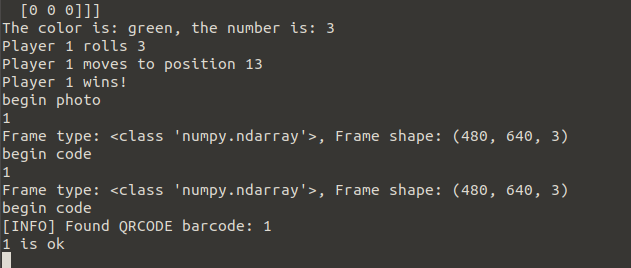


图4.1.4 冷笑话模块代码运行结果截图

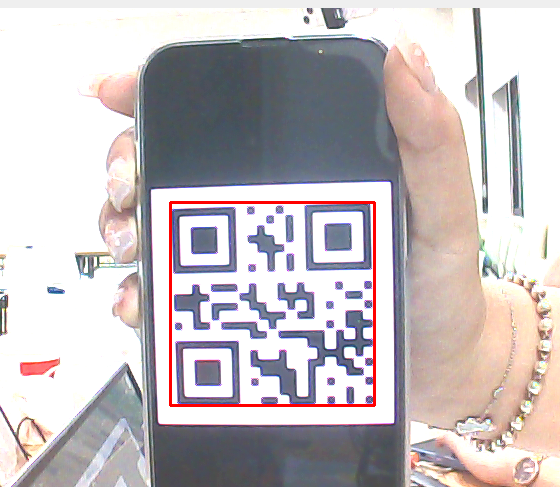


图4.1.5 二维码识别

运行结果符合预期要求，控制台上输出了识别到的二维码图像以及对应信息。

**4.1.5 跳舞模块**

用户对语音识别模块说一句：“跳舞”，语音播报模块就会发出提示音，然后机械臂语音播报，"小花就要开始跳舞啦，请小主人给小花一个手势，然后搬好小板凳准备观看哦"，进入讲冷笑话模块。

用户提供给机械臂一个手势，机械臂进行识别，识别完成后进行语音播报识别结果。不同的识别结果对应不同的舞蹈，对识别结果进行判断，然后将机械臂的动作位置发送给机械臂完成舞蹈。

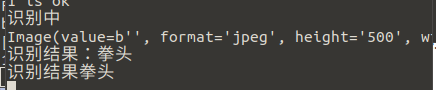


图4.1.5 跳舞模块代码运行结果截图

运行结果符合预期要求，控制台上输出了识别到的结果。

**4.1.6 飞行棋模块**

用户对语音识别模块说一句：“游戏”，语音播报模块就会发出提示音，然后机械臂语音播报，"欢迎来到小花的飞行棋世界，请问小主人是想自己玩还是和小花一起玩？"，进入飞行棋模块。

用户选择“参战”，对语音识别模块说“参战”，语音播报模块就会发出提示音，然后机械臂语音播报，"即将进行人机对战，请小主人做好准备"，进入人机对战模块。人机对战在结束后，机械臂会根据输赢展示不同的舞蹈。

用户选择“参战”，对语音识别模块说“参战”，语音播报模块就会发出提示音，然后机械臂语音播报，"即将进行机器对战，请小主人观战"，进入机器对战模块。

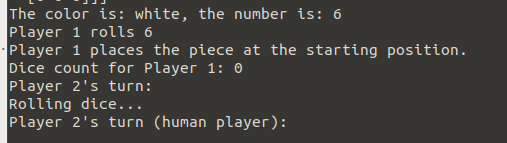


图4.1.6 飞行棋人机对战代码运行结果截图

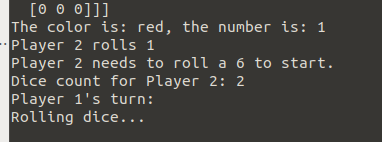


图4.1.7 飞行棋机器模块代码运行结果截图

运行结果符合预期要求，控制台上输出了识别到的骰子点数以及计算出了当前玩家的总点数。

## 4.2 调试说明

我们的开发分为摸索机器，设计机械臂，实现设计，功能完善四个阶段。我们在各个阶段遇到了不同的问题，下面将分阶段来介绍。

4.2.1 摸索机器

（1）连接问题

在一开始接触树莓派机械臂的时候，我们始终没有办法让机械臂连接上网，也无法获取到机械臂的正确ip。

解决方法：重新烧录镜像。通过重新烧录镜像，将机械臂整体刷新，重新配置网络。

1. 远程卡顿问题

在使用VNC远程连接机械臂，进入树莓派桌面，会出现卡顿，如果多台机器同时连接会更卡。

解决方法：通过本地使用ip和端口，连接到Jupyter Lab进行代码的编写和运行。

4.2.2 设计机械臂

（1）飞行棋游戏设计

在设计飞行棋游戏的时候，由于机械臂运动范围的限制，不能满足一般市面上的飞行棋棋盘的要求。

解决方法：自行设计飞行棋棋盘。根据机械臂运动的特点，我们自行设计了一款棋盘。

4.2.3 实现设计

（1）颜色识别

在颜色识别方面，由于机械臂的摄像头各种原因，总是存在不能很好的识别颜色。

解决方法：通过反复的颜色校准。

1. 语音识别

在语音识别方面，因为周围嘈杂环境以及语音识别模块的不够精准，没有办法识别过于长的句子。并且语音识别对需要对拼音的精准把控，对前后鼻音不分的南方人不太友好。

解决方法：缩短语音识别关键词，并避免使用后鼻音的关键词

4.2.4 功能完善

（1）摄像头调用

因为我们是让机械臂连接个人热点，保证电脑和机械臂在同一个网络下。不使用机械臂本身的网络，机械臂摄像头的使用编号会经常改变。我们的飞行棋模块、舞蹈模块、冷笑话模块都使用了摄像头，在单独模块调试的时候，每一块的代码都单独调用了摄像头，整合在一起如果摄像头使用编号发生改变会非常难修改。

解决方法：在主程序中直接调用摄像头，取消每个单独模块的摄像头调用。整个程序结束后再释放摄像头。

1. 摄像头资源被占用

主程序打开摄像头，在进入中途进入别的模块退出的时候，虽然写了在抛出中途停止异常的时候释放摄像头资源，但还是会出现摄像头资源未被释放的情况。摄像头资源如果一直被占用，会导致无法再次执行下次程序。

解决方法：在终端通过ps命令检查占用摄像头资源的进程，用kill命令结束该进程，完成释放。

# 五、工程实践与劳动教育

## 5.1 劳动理论总结

劳动理论在社会学、经济学和哲学等多个领域都有深入探讨，它们揭示了劳动的本质、价值和社会作用。以下是几个核心的劳动理论和价值观的概述：

1.马克思的劳动价值论：卡尔·马克思认为，商品的价值是由生产该商品的社会必要劳动时间决定的。这强调了劳动在创造物质财富中的基础作用。

2.韦伯的劳动伦理：马克斯·韦伯提出的新教伦理与资本主义精神关联，他认为工作不仅是谋生手段，更是一种道德义务，是个人实现自我价值的方式。

3.亚当·斯密的分工理论：亚当·斯密在《国富论》中阐述了分工对提高生产效率的重要性，指出劳动分工可以提高技能，减少工作转换的时间，从而增加生产力。

4.熊彼特的创新理论：约瑟夫·熊彼特认为，经济发展的动力在于创新，包括新技术、新产品、新市场、新的资源组织方式等，这为劳动带来了更高的价值和效率。

5.杜威的经验主义教育：约翰·杜威主张“做中学”，强调实践活动在学习过程中的重要性，认为劳动教育能促进学生全面成长，培养解决问题的能力。

6.现代劳动心理学：现代理论关注工作满意度、职业发展、工作与生活的平衡，强调健康的工作环境和积极的工作态度对于个人幸福和社会福祉的影响。

## 5.2 工程实践与劳动教育体会及感想

在观看关于劳动理论的视频后，我深刻体会到工匠精神在工程实践中的重要性。工匠精神不仅仅是追求技术上的精湛，更是对工作的热爱、对细节的专注和对完美的不懈追求。在工程实践中，这种精神体现在对每一个环节的严格把控，对每一次迭代的精益求精。

例如，在参与一个自动化生产线的设计与实施项目时，我亲眼见证了团队成员如何将看似简单的工序优化到极致，如何通过反复测试和调整来确保设备的稳定性和精度。这种对工作的全情投入和对质量的高标准要求，不仅提高了生产效率，也提升了产品的竞争力。

此外，创新精神与创新意识也是工程实践的关键。在项目中，我们不断尝试新技术和新材料，以期在保持成本可控的前提下，提升产品性能。这种创新不仅为劳动本身带来了更高价值，也激发了团队的创造力，促进了个人能力的提升。

通过这次工程实践与劳动教育，我学到了以下几点：

建立正确的劳动观：认识到劳动不仅仅是谋生手段，更是实现个人价值、推动社会进步的重要途径。

培养工匠精神：注重细节，追求卓越，对待每一项工作都以专业和热情的态度。

强化创新意识：鼓励探索和实验，勇于面对失败，从错误中学习，持续改进。

理解劳动与科技的关系：认识到科技的发展为劳动提供了更多可能性，同时劳动也是科技发展的推动力。

工程实践让我深刻理解了劳动理论在实际工作中的应用，增强了我的职业技能和职业素养，也让我更加珍惜和尊重每一份劳动成果。

# 附录 程序清单

所有程序代码全部位于代码压缩包内：

1. startgame.py:主程序代码，完成模块准备，串接起所有模块。
2. import cv2 as cv
3. import threading
4. import random
5. from time import sleep
6. import Arm\_Lib
7. import smbus
8. from person\_DiceGame import person\_DiceGame
9. from robot\_DiceGame import robot\_DiceGame
10. from play\_music import RoboticArmController
11. from code import QRCodeScanner
12. from dance import dance
13. import re
14. import requests
15. from datetime import datetime
16. from bs4 import BeautifulSoup
17. Arm = Arm\_Lib.Arm\_Device()
18. joints\_0 = [90, 135, 20, 25, 90, 30]
19. Arm.Arm\_serial\_servo\_write6\_array(joints\_0, 1000)
20. import time
21. bus = smbus.SMBus(1)
22. i2c\_addr = 0x0f  *# Speech recognition module address*
23. asr\_add\_word\_addr = 0x01  *# Entry add address*
24. asr\_mode\_addr = 0x02  *# Recognition mode setting address, the value is 0-2, 0: cyclic recognition mode 1: password mode, 2: button mode, the default is cyclic detection*
25. asr\_rgb\_addr = 0x03  *# RGB lamp setting address, need to send two bits, the first one is directly the lamp number 1: blue 2: red 3: green*
26. *# The second byte is brightness 0-255, the larger the value, the higher the brightness*
27. asr\_rec\_gain\_addr = 0x04  *# Identification sensitivity setting address, sensitivity can be set to 0x00-0x7f, the higher the value, the easier it is to detect but the easier it is to misjudge*
28. *# It is recommended to set the value to 0x40-0x55, the default value is 0x40*
29. asr\_clear\_addr = 0x05  *# Clear the operation address of the power-off cache, clear the cache area information before entering the information*
30. asr\_key\_flag = 0x06  *# Used in key mode, set the startup recognition mode*
31. asr\_voice\_flag = 0x07  *# Used to set whether to turn on the recognition result prompt sound*
32. asr\_result = 0x08  *# Recognition result storage address*
33. asr\_buzzer = 0x09  *# Buzzer control register, 1 bit is on, 0 bit is off*
34. asr\_num\_cleck = 0x0a  *# Check the number of entries*
35. asr\_vession = 0x0b  *# firmware version number*
36. asr\_busy = 0x0c  *# Busy and busy flag*
37. i2c\_speech\_addr = 0x30  *# ÓïÒô²¥±¨Ä£¿éµØÖ·*
38. speech\_date\_head = 0xfd
39. *# 天气爬虫*
40. def weather():
41. headers = {
42. 'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36',
43. }
44. res = requests.get('https://tianqi.moji.com/weather/china/zhejiang/xihu-district', headers=headers)
45. soup = BeautifulSoup(res.text, "html.parser")
46. temp = soup.find('div', class\_='wea\_weather clearfix').em.get\_text()
47. weather = soup.find('div', class\_='wea\_weather clearfix').b.get\_text()
48. sd = soup.find('div', class\_='wea\_about clearfix').span.get\_text()
49. sd\_num = re.search(r'\d+', sd).group()
50. wind = soup.find('div', class\_='wea\_about clearfix').em.get\_text()
51. aqi = soup.find('div', class\_='wea\_alert clearfix').em.get\_text()
52. aqi\_num = re.search(r'\d+', aqi).group()
53. info = soup.find('div', class\_='wea\_tips clearfix').em.get\_text()
54. *# 格式化湿度和空气质量指数*
55. sd = '湿度百分之%s' % sd\_num
56. aqi = '空气质量指数%s' % aqi\_num
57. *# 获取当前日期*
58. today = datetime.now().date().strftime('%Y年%m月%d日')
59. *# 构造返回的天气信息文本*
60. text = f'小主人好！今天是{today}，天气{weather}，温度{temp}摄氏度，{sd}，{wind}，{aqi}，{info}'
61. return text
62. time\_1 = 500
63. time\_2 = 1000
64. time\_sleep = 0.5
65. def I2C\_WriteBytes(str\_):
66. global i2c\_speech\_addr
67. for ch in str\_:
68. try:
69. bus.write\_byte(i2c\_speech\_addr, ch)
70. time.sleep(0.01)
71. except:
72. print("write I2C error")
73. EncodingFormat\_Type = {
74. 'GB2312': 0x00,
75. 'GBK': 0X01,
76. 'BIG5': 0x02,
77. 'UNICODE': 0x03
78. }
79. def Speech\_text(str\_, encoding\_format):
80. str\_ = str\_.encode('gb2312')
81. size = len(str\_) + 2
82. DataHead = speech\_date\_head
83. Length\_HH = size >> 8
84. Length\_LL = size & 0x00ff
85. Commond = 0x01
86. EncodingFormat = encoding\_format
87. Date\_Pack = [DataHead, Length\_HH, Length\_LL, Commond, EncodingFormat]
88. I2C\_WriteBytes(Date\_Pack)
89. I2C\_WriteBytes(str\_)
90. def SetBase(str\_):
91. str\_ = str\_.encode('gb2312')
92. size = len(str\_) + 2
93. DataHead = speech\_date\_head
94. Length\_HH = size >> 8
95. Length\_LL = size & 0x00ff
96. Commond = 0x01
97. EncodingFormat = 0x00
98. Date\_Pack = [DataHead, Length\_HH, Length\_LL, Commond, EncodingFormat]
99. I2C\_WriteBytes(Date\_Pack)
100. I2C\_WriteBytes(str\_)
101. def TextCtrl(ch, num):
102. if num != -1:
103. str\_T = '[' + ch + str(num) + ']'
104. SetBase(str\_T)
105. else:
106. str\_T = '[' + ch + ']'
107. SetBase(str\_T)
108. ChipStatus\_Type = {
109. 'ChipStatus\_InitSuccessful': 0x4A,
110. 'ChipStatus\_CorrectCommand': 0x41,
111. 'ChipStatus\_ErrorCommand': 0x45,
112. 'ChipStatus\_Busy': 0x4E,
113. 'ChipStatus\_Idle': 0x4F
114. }
115. def GetChipStatus():
116. global i2c\_speech\_addr
117. AskState = [0xfd, 0x00, 0x01, 0x21]
118. try:
119. I2C\_WriteBytes(AskState)
120. time.sleep(0.05)
121. except:
122. print("I2CRead\_Write error")
123. try:
124. Read\_result = bus.read\_byte(i2c\_speech\_addr)
125. return Read\_result
126. except:
127. print("I2CRead error")
128. Style\_Type = {
129. 'Style\_Single': 0,
130. 'Style\_Continue': 1
131. }
132. def SetStyle(num):
133. TextCtrl('f', num)
134. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
135. time.sleep(0.002)
136. Language\_Type = {
137. 'Language\_Auto': 0,
138. 'Language\_Chinese': 1,
139. 'Language\_English': 2
140. }
141. def SetLanguage(num):
142. TextCtrl('g', num)
143. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
144. time.sleep(0.002)
145. Articulation\_Type = {
146. 'Articulation\_Auto': 0,
147. 'Articulation\_Letter': 1,
148. 'Articulation\_Word': 2
149. }
150. def SetArticulation(num):
151. TextCtrl('h', num)
152. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
153. time.sleep(0.002)
154. Spell\_Type = {
155. 'Spell\_Disable': 0,
156. 'Spell\_Enable': 1
157. }
158. def SetSpell(num):
159. TextCtrl('i', num)
160. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
161. time.sleep(0.002)
162. Reader\_Type = {
163. 'Reader\_XiaoYan': 3,
164. 'Reader\_XuJiu': 51,
165. 'Reader\_XuDuo': 52,
166. 'Reader\_XiaoPing': 53,
167. 'Reader\_DonaldDuck': 54,
168. 'Reader\_XuXiaoBao': 55
169. }
170. def SetReader(num):
171. TextCtrl('m', num)
172. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
173. time.sleep(0.002)
174. NumberHandle\_Type = {
175. 'NumberHandle\_Auto': 0,
176. 'NumberHandle\_Number': 1,
177. 'NumberHandle\_Value': 2
178. }
179. def SetNumberHandle(num):
180. TextCtrl('n', num)
181. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
182. time.sleep(0.002)
183. ZeroPronunciation\_Type = {
184. 'ZeroPronunciation\_Zero': 0,
185. 'ZeroPronunciation\_O': 1
186. }
187. def SetZeroPronunciation(num):
188. TextCtrl('o', num)
189. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
190. time.sleep(0.002)
191. NamePronunciation\_Type = {
192. 'NamePronunciation\_Auto': 0,
193. 'NamePronunciation\_Constraint': 1
194. }
195. def SetNamePronunciation(num):
196. TextCtrl('r', num)
197. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
198. time.sleep(0.002)
199. def SetSpeed(speed):
200. TextCtrl('s', speed)
201. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
202. time.sleep(0.002)
203. def SetIntonation(intonation):
204. TextCtrl('t', intonation)
205. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
206. time.sleep(0.002)
207. def SetVolume(volume):
208. TextCtrl('v', volume)
209. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
210. time.sleep(0.002)
211. OnePronunciation\_Type = {
212. 'OnePronunciation\_Yao': 0,
213. 'OnePronunciation\_Yi': 1
214. }
215. def SetOnePronunciation(num):
216. TextCtrl('y', num)
217. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
218. time.sleep(0.002)
219. Rhythm\_Type = {
220. 'Rhythm\_Diasble': 0,
221. 'Rhythm\_Enable': 1
222. }
223. def SetRhythm(num):
224. TextCtrl('z', num)
225. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
226. time.sleep(0.002)
227. def SetRestoreDefault():
228. TextCtrl('d', -1)
229. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
230. time.sleep(0.002)
231. *# Write entry*
232. def AsrAddWords(idnum, str):
233. global i2c\_addr
234. global asr\_add\_word\_addr
235. words = []
236. words.append(asr\_add\_word\_addr)
237. words.append(len(str) + 2)
238. words.append(idnum)
239. for alond\_word in str:
240. words.append(ord(alond\_word))
241. words.append(0)
242. print(words)
243. for date in words:
244. bus.write\_byte(i2c\_addr, date)
245. time.sleep(0.03)
246. *# Set RGB*
247. def RGBSet(R, G, B):
248. global i2c\_addr
249. global asr\_rgb\_addr
250. date = []
251. date.append(R)
252. date.append(G)
253. date.append(B)
254. print(date)
255. bus.write\_i2c\_block\_data(i2c\_addr, asr\_rgb\_addr, date)
256. *# Read result*
257. def I2CReadByte(reg):
258. global i2c\_addr
259. bus.write\_byte(i2c\_addr, reg)
260. time.sleep(0.05)
261. Read\_result = bus.read\_byte(i2c\_addr)
262. return Read\_result
263. *# Wait busy*
264. def Busy\_Wait():
265. busy = 255
266. while busy != 0:
267. busy = I2CReadByte(asr\_busy)
268. print(asr\_busy)
269. '''
270. The mode and phrase have the function of power-down save, if there is no modification after the first entry, you can change 1 to 0
271. '''
272. cleck = 1
273. if 1:
274. bus.write\_byte\_data(i2c\_addr, asr\_clear\_addr, 0x40)  *# Clear the power-down buffer area*
275. Busy\_Wait()  *# Wait for the module to be free*
276. print("Cache cleared")
277. bus.write\_byte\_data(i2c\_addr, asr\_mode\_addr, 1)
278. Busy\_Wait()
279. print("The mode is set")
280. AsrAddWords(0, "xiao hua")
281. Busy\_Wait()
282. AsrAddWords(1, "you xi")
283. Busy\_Wait()
284. AsrAddWords(2, "tian qi")
285. Busy\_Wait()
286. AsrAddWords(3, "can zhan")
287. Busy\_Wait()
288. AsrAddWords(4, "guan zhan")
289. Busy\_Wait()
290. AsrAddWords(5, "yan zou")
291. Busy\_Wait()
292. AsrAddWords(6, "tiao wu")
293. Busy\_Wait()
294. AsrAddWords(7, "gu shi")
295. Busy\_Wait()
296. AsrAddWords(8, "qu xiao")
297. Busy\_Wait()
298. while cleck != 9:
299. cleck = I2CReadByte(asr\_num\_cleck)
300. print(cleck)
301. bus.write\_byte\_data(i2c\_addr, asr\_rec\_gain\_addr, 0x40)  *# Set the sensitivity, the recommended value is 0x40-0x55*
302. bus.write\_byte\_data(i2c\_addr, asr\_voice\_flag, 1)  *# Set switch sound*
303. bus.write\_byte\_data(i2c\_addr, asr\_buzzer, 1)  *# buzzer*
304. RGBSet(255, 255, 255)
305. time.sleep(1)
306. RGBSet(0, 0, 0)
307. bus.write\_byte\_data(i2c\_addr, asr\_buzzer, 0)  *# buzzer*
308. SetReader(Reader\_Type["Reader\_XiaoYan"])
309. SetVolume(20)
310. Speech\_text("欢迎来到你的专属智能乐园，我是亲爱的小花。你想要做什么。", EncodingFormat\_Type["GB2312"])
311. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
312. time.sleep(0.1)
313. def main():
314. global HSV\_learning, model
315. *# 打开摄像头*
316. capture = cv.VideoCapture(1)
317. capture.set(3, 640)
318. capture.set(4, 480)
319. capture.set(5, 30)  *# 设置帧率*
320. *#     当摄像头正常打开的情况下循环执行*
321. while capture.isOpened():
322. *#    while True:*
323. try:
324. result = I2CReadByte(asr\_result)
325. time.sleep(0.01)
326. if result == 1:
327. Speech\_text("欢迎来到小花的飞行棋世界，请问小主人是想自己玩还是和小花一起玩？",
328. EncodingFormat\_Type["GB2312"])
329. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
330. time.sleep(0.1)
331. elif result == 2:
332. text = weather()
333. print(text)
334. Speech\_text(text, EncodingFormat\_Type["GB2312"])
335. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
336. time.sleep(0.1)
337. elif result == 3:
338. Speech\_text("即将进行人机对战，请小主人做好准备",
339. EncodingFormat\_Type["GB2312"])
340. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
341. time.sleep(0.1)
342. *#                 person\_DiceGame().start\_person\_game()*
343. pd = person\_DiceGame()
344. pd.cap = capture
345. pd.start\_person\_game()
346. elif result == 4:
347. Speech\_text("即将进行机器对战，请小主人观战",
348. EncodingFormat\_Type["GB2312"])
349. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
350. time.sleep(0.1)
351. *#                 robot\_DiceGame().start\_robot\_game()*
352. rd = robot\_DiceGame()
353. rd.cap = capture
354. rd.start\_robot\_game()
355. elif result == 5:
356. Speech\_text("小花即将变成演奏大师，请小主人准备聆听",
357. EncodingFormat\_Type["GB2312"])
358. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
359. time.sleep(0.1)
360. time.sleep(10)
361. RoboticArmController().main\_game()
362. elif result == 6:
363. Speech\_text("小花就要开始跳舞啦，请小主人给小花一个手势，然后搬好小板凳准备观看哦",
364. EncodingFormat\_Type["GB2312"])
365. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
366. time.sleep(0.1)
367. *#                 dance().start()*
368. tw = dance()
369. tw.cap = capture
370. tw.start()
371. elif result == 7:
372. Speech\_text("小花要开始说花花的笑话了，请小主人不要吃东西哦",
373. EncodingFormat\_Type["GB2312"])
374. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
375. time.sleep(0.1)
376. *#                 QRCodeScanner().capture\_qr\_codes()*
377. story = QRCodeScanner()
378. story.cap = capture
379. cap.capture\_qr\_codes()
380. elif result == 7:
381. *# result = 255*
382. text = weather()
383. Speech\_text("ok", EncodingFormat\_Type["GB2312"])
384. while GetChipStatus() != ChipStatus\_Type['ChipStatus\_Idle']:
385. time.sleep(0.1)
386. cv.destroyAllWindows()
387. *# del Arm*
388. *# del bus*
389. print(" Program closed! ")
390. pass
391. break
392. except KeyboardInterrupt:
393. capture.release()
394. del Arm
395. del bus
396. print(" Program closed! ")
397. pass
398. *# def main():*
399. *#     while True:*
400. *#         result = I2CReadByte(asr\_result)*
401. *#         if result == 1:*
402. *#             print(1)*
403. *#*
404. *# display(controls\_box,output)*
405. *# threading.Thread(target=camera, ).start()*
406. try:
407. main()
408. except KeyboardInterrupt:
409. capture.release()
410. *#     del Arm*
411. *#     del bus*
412. *#     print(" Program closed! ")*
413. *#     pass*
414. speak.py:语音播报模块代码。
415. *#!/usr/bin/env python3*
416. *#coding=utf-8*
417. import smbus
418. import time
419. class speak:
420. def \_\_init\_\_(self):
421. self.i2c\_addr = 0x30
422. self.date\_head = 0xfd
423. self.bus = smbus.SMBus(1)
424. def I2C\_WriteBytes(self, str\_):
425. for ch in str\_:
426. try:
427. self.bus.write\_byte(self.i2c\_addr, ch)
428. time.sleep(0.01)
429. except:
430. print("write I2C error")
431. EncodingFormat\_Type = {
432. 'GB2312': 0x00,
433. 'GBK': 0X01,
434. 'BIG5': 0x02,
435. 'UNICODE': 0x03
436. }
437. def Speech\_text(self, str\_, encoding\_format):
438. str\_ = str\_.encode('gb2312')
439. size = len(str\_) + 2
440. DataHead = self.date\_head
441. Length\_HH = size >> 8
442. Length\_LL = size & 0x00ff
443. Commond = 0x01
444. EncodingFormat = encoding\_format
445. Date\_Pack = [DataHead, Length\_HH, Length\_LL, Commond, EncodingFormat]
446. self.I2C\_WriteBytes(Date\_Pack)
447. self.I2C\_WriteBytes(str\_)
448. def SetBase(self, str\_):
449. str\_ = str\_.encode('gb2312')
450. size = len(str\_) + 2
451. DataHead = self.date\_head
452. Length\_HH = size >> 8
453. Length\_LL = size & 0x00ff
454. Commond = 0x01
455. EncodingFormat = 0x00
456. Date\_Pack = [DataHead, Length\_HH, Length\_LL, Commond, EncodingFormat]
457. self.I2C\_WriteBytes(Date\_Pack)
458. self.I2C\_WriteBytes(str\_)
459. def TextCtrl(self, ch, num):
460. if num != -1:
461. str\_T = '[' + ch + str(num) + ']'
462. self.SetBase(str\_T)
463. else:
464. str\_T = '[' + ch + ']'
465. self.SetBase(str\_T)
466. ChipStatus\_Type = {
467. 'ChipStatus\_InitSuccessful': 0x4A,  *# 初始化成功回传*
468. 'ChipStatus\_CorrectCommand': 0x41,  *# 收到正确的命令帧回传*
469. 'ChipStatus\_ErrorCommand': 0x45,  *# 收到不能识别命令帧回传*
470. 'ChipStatus\_Busy': 0x4E,  *# 芯片忙碌状态回传*
471. 'ChipStatus\_Idle': 0x4F  *# 芯片空闲状态回传*
472. }
473. def GetChipStatus(self):
474. AskState = [0xfd, 0x00, 0x01, 0x21]
475. try:
476. self.I2C\_WriteBytes(AskState)
477. time.sleep(0.05)
478. except:
479. print("I2CRead\_Write error")
480. try:
481. Read\_result = self.bus.read\_byte(self.i2c\_addr)
482. return Read\_result
483. except:
484. print("I2CRead error")
485. Style\_Type = {
486. 'Style\_Single': 0,  *# 为 0，一字一顿的风格*
487. 'Style\_Continue': 1  *# 为 1，正常合成*
488. }  *# 合成风格设置 [f?]*
489. def SetStyle(self, num):
490. self.TextCtrl('f', num)
491. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
492. time.sleep(0.002)
493. Language\_Type = {
494. 'Language\_Auto': 0,  *# 为 0，自动判断语种*
495. 'Language\_Chinese': 1,  *# 为 1，阿拉伯数字、度量单位、特殊符号等合成为中文*
496. 'Language\_English': 2  *# 为 1，阿拉伯数字、度量单位、特殊符号等合成为中文*
497. }  *# 合成语种设置 [g?]*
498. def SetLanguage(self, num):
499. self.TextCtrl('g', num)
500. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
501. time.sleep(0.002)
502. Articulation\_Type = {
503. 'Articulation\_Auto': 0,  *# 为 0，自动判断单词发音方式*
504. 'Articulation\_Letter': 1,  *# 为 1，字母发音方式*
505. 'Articulation\_Word': 2  *# 为 2，单词发音方式*
506. }  *# 设置单词的发音方式 [h?]*
507. def SetArticulation(self, num):
508. self.TextCtrl('h', num)
509. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
510. time.sleep(0.002)
511. Spell\_Type = {
512. 'Spell\_Disable': 0,  *# 为 0，不识别汉语拼音*
513. 'Spell\_Enable': 1  *# 为 1，将“拼音＋1 位数字（声调）”识别为汉语拼音，例如： hao3*
514. }  *# 设置对汉语拼音的识别 [i?]*
515. def SetSpell(self, num):
516. self.TextCtrl('i', num)
517. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
518. time.sleep(0.002)
519. Reader\_Type = {
520. 'Reader\_XiaoYan': 3,  *# 为 3，设置发音人为小燕(女声, 推荐发音人)*
521. 'Reader\_XuJiu': 51,  *# 为 51，设置发音人为许久(男声, 推荐发音人)*
522. 'Reader\_XuDuo': 52,  *# 为 52，设置发音人为许多(男声)*
523. 'Reader\_XiaoPing': 53,  *# 为 53，设置发音人为小萍(女声*
524. 'Reader\_DonaldDuck': 54,  *# 为 54，设置发音人为唐老鸭(效果器)*
525. 'Reader\_XuXiaoBao': 55  *# 为 55，设置发音人为许小宝(女童声)*
526. }  *# 选择发音人 [m?]*
527. def SetReader(self, num):
528. self.TextCtrl('m', num)
529. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
530. time.sleep(0.002)
531. NumberHandle\_Type = {
532. 'NumberHandle\_Auto': 0,  *# 为 0，自动判断*
533. 'NumberHandle\_Number': 1,  *# 为 1，数字作号码处理*
534. 'NumberHandle\_Value': 2  *# 为 2，数字作数值处理*
535. }  *# 设置数字处理策略 [n?]*
536. def SetNumberHandle(self, num):
537. self.TextCtrl('n', num)
538. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
539. time.sleep(0.002)
540. ZeroPronunciation\_Type = {
541. 'ZeroPronunciation\_Zero': 0,  *# 为 0，读成“zero*
542. 'ZeroPronunciation\_O': 1  *# 为 1，读成“欧”音*
543. }  *# 数字“0”在读 作英文、号码时 的读法 [o?]*
544. def SetZeroPronunciation(self, num):
545. self.TextCtrl('o', num)
546. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
547. time.sleep(0.002)
548. NamePronunciation\_Type = {
549. 'NamePronunciation\_Auto': 0,  *# 为 0，自动判断姓氏读音*
550. 'NamePronunciation\_Constraint': 1  *# 为 1，强制使用姓氏读音规则*
551. }  *# 设置姓名读音 策略 [r?]*
552. def SetNamePronunciation(self, num):
553. self.TextCtrl('r', num)
554. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
555. time.sleep(0.002)
556. *# 设置语速 [s?] ? 为语速值，取值：0～10*
557. def SetSpeed(self, speed):
558. self.TextCtrl('s', speed)
559. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
560. time.sleep(0.002)
561. *# 设置语调 [t?] ? 为语调值，取值：0～10*
562. def SetIntonation(self, intonation):
563. self.TextCtrl('t', intonation)
564. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
565. time.sleep(0.002)
566. *# 设置音量 [v?] ? 为音量值，取值：0～10*
567. def SetVolume(self, volume):
568. self.TextCtrl('v', volume)
569. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
570. time.sleep(0.002)
571. OnePronunciation\_Type = {
572. 'OnePronunciation\_Yao': 0,  *# 为 0，合成号码“1”时读成幺*
573. 'OnePronunciation\_Yi': 1  *# 为 1，合成号码“1”时读成一*
574. }  *# 设置号码中“1”的读法 [y?]*
575. def SetOnePronunciation(self, num):
576. self.TextCtrl('y', num)
577. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
578. time.sleep(0.002)
579. Rhythm\_Type = {
580. 'Rhythm\_Diasble': 0,  *# 为 0，“ \*”和“#”读出符号*
581. 'Rhythm\_Enable': 1  *# 为 1，处理成韵律，“\*”用于断词，“#”用于停顿*
582. }  *# 是否使用韵律 标记“\*”和“#” [z?]*
583. def SetRhythm(self, num):
584. self.TextCtrl('z', num)
585. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
586. time.sleep(0.002)
587. *# 恢复默认的合成参数 [d] 所有设置（除发音人设置、语种设置外）恢复为默认值*
588. def SetRestoreDefault(self):
589. self.TextCtrl('d', -1)
590. while self.GetChipStatus() != self.ChipStatus\_Type['ChipStatus\_Idle']:
591. time.sleep(0.002)
592. robot\_DiceGame:机器对战模块代码。
593. import time
594. import random
595. import cv2
596. from Arm\_Lib import Arm\_Device
597. class robot\_DiceGame:
598. def \_\_init\_\_(self):
599. self.Arm = Arm\_Device()
600. time.sleep(0.1)
601. *# 定义机械臂动作时间*
602. self.time\_1 = 500
603. self.time\_sleep = 0.5
604. *# 初始化游戏参数*
605. self.board\_length = 13
606. self.player1\_position = -1  *# 玩家1的初始位置*
607. self.player2\_position = -1  *# 玩家2的初始位置*
608. self.dice\_count1 = 0  *# 玩家1的投骰子次数*
609. self.dice\_count2 = 0  *# 玩家2的投骰子次数*
610. *# 初始化机械臂动作*
611. self.init\_arm\_positions()
612. self.cap = None
613. self.p1 = [5, 75, 20, 0, 90]  *# p0 = [90, 90, 45, 0, 90]*
614. self.p2 = [5, 60, 40, 10, 90]
615. self.change = 13.2
616. self.p1\_1 = [50, 18, 97, 20, 90]
617. self.p2\_2 = [70, 18, 97, 20, 90]
618. *# 初始化机械臂位置*
619. def init\_arm\_positions(self):
620. self.Arm.Arm\_serial\_servo\_write6(90, 90, 90, 90, 90, 90, 500)
621. time.sleep(1)
622. *# 定义夹取积木块函数，enable=1：夹住，=0：松开*
623. def arm\_clamp\_block(self, enable):
624. if enable == 0:
625. self.Arm.Arm\_serial\_servo\_write(6, 40, 1000)
626. else:
627. self.Arm.Arm\_serial\_servo\_write(6, 150, 400)
628. time.sleep(0.5)
629. *# 定义移动机械臂函数，同时控制 1-5 号舵机运动，p=[S1,S2,S3,S4,S5]*
630. def arm\_move(self, p, s\_time=500):
631. for i in range(5):
632. id = i + 1
633. self.Arm.Arm\_serial\_servo\_write(id, p[i], s\_time)
634. time.sleep(0.01)
635. time.sleep(s\_time / 1000)
636. *# 机械臂向上移动1*
637. def arm\_move\_up1(self):
638. self.Arm.Arm\_serial\_servo\_write(2, 90, 1500)
639. time.sleep(0.1)
640. *# 机械臂向上移动2*
641. def arm\_move\_up2(self):
642. self.Arm.Arm\_serial\_servo\_write(2, 75, 1500)
643. time.sleep(0.1)
644. *# 机械臂向上移动0\_1*
645. def arm\_move\_0\_1(self):
646. p = [5, 107, 9, 0, 90]
647. self.Arm.Arm\_serial\_servo\_write(2, 90, 1000)
648. time.sleep(0.1)
649. self.Arm.Arm\_serial\_servo\_write(3, 45, 1000)
650. time.sleep(0.1)
651. self.arm\_move(p, 1000)
652. time.sleep(0.1)
653. *# 机械臂向上移动0\_2*
654. def arm\_move\_0\_2(self):
655. p = [5, 90, 24, 7, 90]
656. self.Arm.Arm\_serial\_servo\_write(2, 90, 1000)
657. time.sleep(0.1)
658. self.Arm.Arm\_serial\_servo\_write(3, 45, 1000)
659. time.sleep(0.1)
660. self.arm\_move(p, 1000)
661. time.sleep(0.1)
662. *# 识别颜色并返回相应的数字*
663. def getdice\_num(self):
664. *# 打开摄像头*
665. *# while True:*
666. *#     last\_num = None*
667. *#     current\_num = None*
668. *#*
669. *#     while current\_num == last\_num:*
670. *#         current\_num = random.randint(0, 2)*
671. *#*
672. *#     cap = cv2.VideoCapture(current\_num)*
673. *#     if cap.isOpened():*
674. *#         print("Camera ok")*
675. *#         break*
676. *#     print(current\_num)*
677. *#     print("Error: Could not open camera.")*
678. *#     last\_num = current\_num*
679. *# if cap.isOpened():*
680. *#     print("Error: Could not open camera.1 can not")*
681. *#     num=2*
682. *# return -1  # 返回-1表示摄像头打开失败*
683. *# if not cap.isOpened():*
684. *#     print("Error: Could not open camera.")*
685. *#     return -1  # 返回-1表示摄像头打开失败*
686. *# 读取一帧图像*
687. *#         cap = cv2.VideoCapture(1)*
688. *#         if not cap.isOpened():*
689. *#             print("Error: Could not open camera.")*
690. *#             return -1  # 返回-1表示摄像头打开失败*
691. *#         # 读取一帧图像*
692. *#         ret, frame = cap.read()*
693. *#         cap.release()  # 读取一帧图像后立即释放摄像头*
694. *#         if not ret:*
695. *#             print("Error: Could not read frame.")*
696. *#             return -1  # 返回-1表示读取图像失败*
697. *# #         cap=self.cap*
698. *# #         ret, frame = cap.read()*
699. *#         cap.release()*
700. *#         cap = cv2.VideoCapture(1)*
701. cap = self.cap
702. if not cap.isOpened():
703. print("Error: Could not open camera.")
704. return -1  *# 返回-1表示摄像头打开失败*
705. *# 读取一帧图像*
706. ret, frame = cap.read()
707. cv2.imshow("image", frame)
708. *#         cap.release()  # 读取一帧图像后立即释放摄像头*
709. if not ret:
710. print("Error: Could not read frame.")
711. return -1  *# 返回-1表示读取图像失败*
712. *#         cap=self.cap*
713. *#         ret, frame = cap.read()*
714. *#         cap.release()*
715. height, width, \_ = frame.shape
716. *# 定义中心区域的尺寸（例如：50x50像素）*
717. center\_size = 50
718. x\_start = width // 2 - center\_size // 2
719. y\_start = height // 2 - center\_size // 2
720. x\_end = x\_start + center\_size
721. y\_end = y\_start + center\_size
722. *# 获取中心区域图像*
723. center\_region = frame[y\_start:y\_end, x\_start:x\_end]
724. *# 将中心区域图像转换为HSV颜色空间*
725. hsv = cv2.cvtColor(center\_region, cv2.COLOR\_BGR2HSV)
726. *# 定义颜色的HSV范围*
727. color\_ranges = {
728. 1: [(160, 100, 170), (175, 200, 240)],  *# 红色*
729. 2: [(100, 150, 140), (140, 255, 255)],  *# 蓝色*
730. 3: [(75, 200, 100), (90, 255, 180)],  *# 绿色*
731. 4: [(25, 100, 190), (32, 180, 255)],  *# 黄色*
732. 5: [(95, 130, 35), (120, 218, 65)],  *# 黑色*
733. 6: [(80, 5, 100), (125, 200, 255)]  *# 白色*
734. }
735. max\_pixels = 0
736. detected\_dice\_num = -1
737. *# 定义颜色与数字的映射*
738. mapping = {
739. 1: "red",
740. 2: "blue",
741. 3: "green",
742. 4: "yellow",
743. 5: "black",
744. 6: "white"
745. }
746. for dice\_num, ranges in color\_ranges.items():
747. mask = cv2.inRange(hsv, ranges[0], ranges[1])
748. pixels = cv2.countNonZero(mask)
749. if pixels > max\_pixels:
750. max\_pixels = pixels
751. detected\_dice\_num = dice\_num
752. if detected\_dice\_num == -1:
753. print("camera error")
754. else:
755. print(f"the color is: {mapping[detected\_dice\_num]} the number is: {detected\_dice\_num}")
756. *# 返回检测到的数值*
757. return detected\_dice\_num
758. def throw\_dice(self):
759. *# 机械臂移动到拿骰子的位置*
760. pick\_position = [90, 46, 53, 22, 90]
761. p\_vidio = [90, 78, 24, 13, 90]
762. self.Arm.Arm\_serial\_servo\_write(2, 90, 1500)  *# 抬起*
763. time.sleep(1)
764. self.arm\_move(pick\_position, 1000)
765. self.arm\_clamp\_block(1)
766. time.sleep(1)
767. self.Arm.Arm\_serial\_servo\_write(2, 90, 1500)  *# 抬起*
768. time.sleep(1)
769. self.arm\_clamp\_block(0)
770. time.sleep(1)
771. self.Arm.Arm\_serial\_servo\_write(5, 90, 500)
772. time.sleep(0.1)
773. *# 获取投掷结果*
774. self.arm\_move(p\_vidio, 1000)
775. time.sleep(3)
776. result = self.getdice\_num()
777. *# 如果识别失败则重新投掷*
778. while result == -1:
779. print("Retrying dice throw...")
780. result = self.getdice\_num()
781. return result
782. *# 投骰子，返回1到6的随机数*
783. def roll\_dice(self):
784. return self.throw\_dice()
785. *# 定义不同位置的变量参数*
786. def position\_move(self, po1, po2):
787. self.arm\_move(po1, 1000)
788. self.arm\_clamp\_block(1)
789. time.sleep(1)
790. self.arm\_move\_up1()
791. time.sleep(1)
792. self.arm\_move(po2, 1000)
793. self.arm\_clamp\_block(0)
794. time.sleep(1)
795. def position\_move2(self, p, steps):
796. self.arm\_move(p, 1000)
797. self.arm\_clamp\_block(1)
798. time.sleep(1)
799. self.arm\_move\_up1()
800. time.sleep(1)
801. p[0] += self.change \* steps
802. self.arm\_move(p, 1000)
803. self.arm\_clamp\_block(0)
804. time.sleep(1)
805. def position\_move0\_1(self, po1, po2):
806. self.arm\_move(po1, 1000)
807. self.arm\_clamp\_block(1)
808. time.sleep(1)
809. self.arm\_move\_0\_1()
810. time.sleep(1)
811. self.arm\_move(po2, 1000)
812. self.arm\_clamp\_block(0)
813. time.sleep(1)
814. def position\_move0\_2(self, po1, po2):
815. self.arm\_move(po1, 1000)
816. self.arm\_clamp\_block(1)
817. time.sleep(1)
818. self.arm\_move\_0\_2()
819. time.sleep(1)
820. self.arm\_move(po2, 1000)
821. self.arm\_clamp\_block(0)
822. time.sleep(1)
823. def move\_player(self, player, steps):
824. if player == 1:
825. if self.player1\_position == -1:  *# 棋子未在棋盘上*
826. if steps == 6 or self.dice\_count1 == 2:
827. self.player1\_position = 0  *# 将棋子放到初始位置*
828. self.position\_move0\_1(self.p1\_1, self.p1)
829. print(f"Player {player} places the piece at the starting position.")
830. self.dice\_count1 = 0
831. print(f"the dice\_number is dice\_count1={self.dice\_count1}")
832. else:
833. print(f"Player {player} needs to roll a 6 to start.")
834. print(f"the dice\_number is dice\_count1={self.dice\_count1}")
835. self.dice\_count1 += 1
836. print(f"the dice\_number is dice\_count1={self.dice\_count1}")
837. return False
838. else:
839. old\_player1\_position = self.player1\_position
840. self.player1\_position += steps
841. if self.player1\_position > self.board\_length:
842. steps = self.board\_length - old\_player1\_position
843. self.player1\_position = self.board\_length
844. print(f"Player {player} moves to position {self.player1\_position}")
845. self.position\_move2(self.p1, steps)
846. if self.player1\_position == self.board\_length:
847. b\_time = 3
848. self.Arm.Arm\_Buzzer\_On(b\_time)
849. print(f"Player {player} wins!")
850. return True
851. elif player == 2:
852. if self.player2\_position == -1:  *# 棋子未在棋盘上*
853. if steps == 6 or self.dice\_count2 == 2:
854. self.player2\_position = 0  *# 将棋子放到初始位置*
855. self.position\_move0\_2(self.p2\_2, self.p2)
856. print(f"Player {player} places the piece at the starting position.")
857. self.dice\_count2 = 0
858. print(f"the dice\_number is dice\_count2={self.dice\_count2}")
859. else:
860. print(f"Player {player} needs to roll a 6 to start.")
861. print(f"the dice\_number is dice\_count2={self.dice\_count2}")
862. self.dice\_count2 += 1
863. print(f"the dice\_number is dice\_count2={self.dice\_count2}")
864. return False
865. else:
866. old\_player2\_position = self.player2\_position
867. self.player2\_position += steps
868. if self.player2\_position > self.board\_length:
869. steps = self.board\_length - old\_player2\_position
870. self.player2\_position = self.board\_length
871. print(f"Player {player} moves to position {self.player2\_position}")
872. self.position\_move2(self.p2, steps)
873. if self.player2\_position == self.board\_length:
874. b\_time = 3
875. self.Arm.Arm\_Buzzer\_On(b\_time)
876. print(f"Player {player} wins!")
877. return True
878. else:
879. print(f"Unknown player: {player}")
880. return False
881. return False
882. *# 游戏主循环*
883. def main\_game(self):
884. current\_player = 1
885. while True:
886. print(f"Player {current\_player}'s turn:")
887. print("rolling dice...")
888. steps = self.roll\_dice()
889. print(f"Player {current\_player} rolls {steps}")
890. if self.move\_player(current\_player, steps):
891. break
892. *# 切换玩家*
893. current\_player = 2 if current\_player == 1 else 1
894. def start\_robot\_game(self):
895. try:
896. self.main\_game()
897. except KeyboardInterrupt:
898. print("\nProgram closed!")
899. finally:
900. guiwei = [90, 90, 90, 90, 90, 90, 500]
901. self.arm\_move(guiwei, 1000)
902. time.sleep(1)
903. *# 如果有关闭连接的方法，可以在这里使用*
904. *# self.Arm.disconnect()  # 假设 Arm\_Device 没有 disconnect 方法*
905. *# 创建游戏对象并运行*
906. if \_\_name\_\_ == "\_\_main\_\_":
907. game = robot\_DiceGame()
908. game.start\_robot\_game()
909. person\_DiceGame:人机对战模块代码。
910. import time
911. import random
912. import cv2
913. from Arm\_Lib import Arm\_Device  *# Assuming Arm\_Device is the library for your robotic arm*
914. from speak import speak
915. class person\_DiceGame:
916. def \_\_init\_\_(self):
917. *# Create the Arm\_Device object for controlling the robotic arm*
918. self.Arm = Arm\_Device()
919. time.sleep(0.1)
920. *# Define arm action times*
921. self.time\_1 = 1000
922. self.time\_sleep = 0.5
923. *# Initialize game parameters*
924. self.board\_length = 13
925. self.player1\_position = -1  *# Player 1's initial position*
926. self.player2\_position = -1  *# Player 2's initial position*
927. self.dice\_count1 = 0  *# Player 1's dice roll count*
928. self.dice\_count2 = 0  *# Player 2's dice roll count*
929. self.p1 = [5, 75, 20, 0, 90]
930. self.p2 = [5, 60, 40, 10, 90]
931. self.change = 13.2
932. self.p1\_1 = [50, 18, 97, 20, 90]
933. self.p2\_2 = [70, 18, 97, 20, 90]
934. *# Initialize arm positions*
935. self.init\_arm\_positions()
936. self.cap = None
937. def init\_arm\_positions(self):
938. self.Arm.Arm\_serial\_servo\_write6(90, 90, 90, 90, 90, 90, 500)
939. time.sleep(1)
940. def arm\_clamp\_block(self, enable):
941. if enable == 0:
942. self.Arm.Arm\_serial\_servo\_write(6, 0, 400)
943. else:
944. self.Arm.Arm\_serial\_servo\_write(6, 150, 400)
945. time.sleep(0.5)
946. def arm\_move(self, p, s\_time=500):
947. for i in range(5):
948. id = i + 1
949. self.Arm.Arm\_serial\_servo\_write(id, p[i], s\_time)
950. time.sleep(0.01)
951. time.sleep(s\_time / 1000)
952. def arm\_move\_up1(self):
953. self.Arm.Arm\_serial\_servo\_write(2, 90, 1500)
954. time.sleep(0.1)
955. def arm\_move\_0\_1(self):
956. p = [5, 107, 9, 0, 90]
957. self.Arm.Arm\_serial\_servo\_write(2, 90, 1000)
958. time.sleep(0.1)
959. self.Arm.Arm\_serial\_servo\_write(3, 45, 1000)
960. time.sleep(0.1)
961. self.arm\_move(p, 1000)
962. time.sleep(0.1)
963. def arm\_move\_up2(self):
964. self.Arm.Arm\_serial\_servo\_write(2, 75, 1500)
965. time.sleep(0.1)
966. def getdice\_num(self):
967. *#         cap = cv2.VideoCapture(1)*
968. *#         if not cap.isOpened():*
969. *#             print("Error: Could not open camera.")*
970. *#             return -1  # 返回-1表示摄像头打开失败*
971. cap=self.cap
972. if not cap.isOpened():
973. print("Error: Could not open camera.")
974. return -1  *# 返回-1表示摄像头打开失败*
975. *# 读取一帧图像*
976. ret, frame = cap.read()
977. cv2.imshow("image", frame)
978. *#         cap.release()  # 读取一帧图像后立即释放摄像头*
979. if not ret:
980. print("Error: Could not read frame.")
981. return -1  *# 返回-1表示读取图像失败*
982. *#         cap=self.cap*
983. *#         ret, frame = cap.read()*
984. *#         cap.release()*
985. height, width, \_ = frame.shape
986. center\_size = 40
987. x\_start = width // 2 - center\_size // 2
988. y\_start = height // 2 - center\_size // 2
989. x\_end = x\_start + center\_size
990. y\_end = y\_start + center\_size
991. center\_region = frame[y\_start:y\_end, x\_start:x\_end]
992. print(f"center\_region: {center\_region}")
993. hsv = cv2.cvtColor(center\_region, cv2.COLOR\_BGR2HSV)
994. color\_ranges = {
995. 1: [(0, 126, 191), (181, 253, 255)],  *# Red*
996. 2: [(104, 150, 160), (113, 253, 255)],  *# Blue*
997. 3: [(57, 76, 111), (75, 253, 255)],  *# Green*
998. 4: [(28, 63, 240), (32, 253, 255)],  *# Yellow*
999. 5: [(0, 0, 0), (120, 218, 65)],  *# Black*
1000. 6: [(80, 5, 100), (125, 200, 255)]  *# White*
1001. }
1002. max\_pixels = 0
1003. detected\_dice\_num = -1
1004. mapping = {
1005. 1: "red",
1006. 2: "blue",
1007. 3: "green",
1008. 4: "yellow",
1009. 5: "black",
1010. 6: "white"
1011. }
1012. for dice\_num, ranges in color\_ranges.items():
1013. mask = cv2.inRange(hsv, ranges[0], ranges[1])
1014. pixels = cv2.countNonZero(mask)
1015. detected\_hsv = cv2.bitwise\_and(hsv, hsv, mask=mask)
1016. print(f"Detected HSV values: {detected\_hsv}")
1017. if pixels > max\_pixels:
1018. max\_pixels = pixels
1019. detected\_dice\_num = dice\_num
1020. if detected\_dice\_num == -1:
1021. print("Camera error")
1022. else:
1023. print(f"The color is: {mapping[detected\_dice\_num]}, the number is: {detected\_dice\_num}")
1024. return detected\_dice\_num
1025. Reader\_Type = {
1026. 'Reader\_XiaoYan': 3,  *# 为 3，设置发音人为小燕(女声, 推荐发音人)*
1027. 'Reader\_XuJiu': 51,  *# 为 51，设置发音人为许久(男声, 推荐发音人)*
1028. 'Reader\_XuDuo': 52,  *# 为 52，设置发音人为许多(男声)*
1029. 'Reader\_XiaoPing': 53,  *# 为 53，设置发音人为小萍(女声*
1030. 'Reader\_DonaldDuck': 54,  *# 为 54，设置发音人为唐老鸭(效果器)*
1031. 'Reader\_XuXiaoBao': 55  *# 为 55，设置发音人为许小宝(女童声)*
1032. }  *# 选择发音人 [m?]'*
1033. def person\_speak(self, num):
1034. myspeak = speak()
1035. if num == 1:
1036. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1037. myspeak.Speech\_text("请小主人走1步", myspeak.EncodingFormat\_Type["GB2312"])
1038. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1039. time.sleep(0.1)
1040. elif num == 2:
1041. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1042. myspeak.Speech\_text("请小主人走2步", myspeak.EncodingFormat\_Type["GB2312"])
1043. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1044. time.sleep(0.1)
1045. elif num == 3:
1046. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1047. myspeak.Speech\_text("请小主人走3步", myspeak.EncodingFormat\_Type["GB2312"])
1048. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1049. time.sleep(0.1)
1050. elif num == 4:
1051. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1052. myspeak.Speech\_text("请小主人走4步", myspeak.EncodingFormat\_Type["GB2312"])
1053. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1054. time.sleep(0.1)
1055. elif num == 5:
1056. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1057. myspeak.Speech\_text("请小主人走5步", myspeak.EncodingFormat\_Type["GB2312"])
1058. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1059. time.sleep(0.1)
1060. elif num == 6:
1061. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1062. myspeak.Speech\_text("请小主人走6步", myspeak.EncodingFormat\_Type["GB2312"])
1063. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1064. time.sleep(0.1)
1065. def throw\_dice(self):
1066. pick\_position = [90, 46, 53, 22, 90]
1067. p\_vidio = [90, 78, 24, 13, 90]
1068. self.Arm.Arm\_serial\_servo\_write(2, 90, 1500)  *# Lift*
1069. time.sleep(1)
1070. self.arm\_move(pick\_position, 1000)
1071. self.arm\_clamp\_block(1)
1072. time.sleep(1)
1073. self.Arm.Arm\_serial\_servo\_write(2, 90, 1500)  *# Lift*
1074. time.sleep(1)
1075. self.arm\_clamp\_block(0)
1076. time.sleep(1)
1077. self.Arm.Arm\_serial\_servo\_write(5, 90, 500)
1078. time.sleep(0.1)
1079. self.arm\_move(p\_vidio, 1000)
1080. time.sleep(3)
1081. result = self.getdice\_num()
1082. while result == -1:
1083. print("Retrying dice throw...")
1084. result = self.getdice\_num()
1085. return result
1086. def roll\_dice(self):
1087. return self.throw\_dice()
1088. def fail(self):
1089. guiwei = [90, 90, 90, 90, 90, 90, 500]
1090. self.arm\_move(guiwei)
1091. time.sleep(1)
1092. p\_fail = [90, 90, 15, 5, 90]
1093. self.arm\_move(p\_fail)
1094. self.arm\_clamp\_block(1)
1095. time.sleep(1)
1096. self.arm\_clamp\_block(0)
1097. def win(self):
1098. guiwei = [90, 90, 90, 90, 90, 90, 500]
1099. times = 0.003
1100. self.arm\_move(guiwei)
1101. time.sleep(1)
1102. self.Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
1103. time.sleep(times)
1104. self.Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
1105. time.sleep(times)
1106. self.Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1107. time.sleep(self.time\_sleep)
1108. self.Arm.Arm\_serial\_servo\_write(2, 180 - 135, self.time\_1)
1109. time.sleep(times)
1110. self.Arm.Arm\_serial\_servo\_write(3, 135, self.time\_1)
1111. time.sleep(times)
1112. self.Arm.Arm\_serial\_servo\_write(4, 45, self.time\_1)
1113. time.sleep(self.time\_sleep)
1114. self.Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
1115. time.sleep(times)
1116. self.Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
1117. time.sleep(times)
1118. self.Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1119. time.sleep(self.time\_sleep)
1120. def position\_move(self, po1, po2):
1121. self.arm\_move(po1, 1000)
1122. self.arm\_clamp\_block(1)
1123. time.sleep(1)
1124. self.arm\_move\_up1()
1125. time.sleep(1)
1126. self.arm\_move(po2, 1000)
1127. self.arm\_clamp\_block(0)
1128. time.sleep(1)
1129. def position\_move2(self, p, steps):
1130. self.arm\_move(p, 1000)
1131. self.arm\_clamp\_block(1)
1132. time.sleep(1)
1133. self.arm\_move\_up1()
1134. time.sleep(1)
1135. p[0] += self.change \* steps
1136. self.arm\_move(p, 1000)
1137. self.arm\_clamp\_block(0)
1138. time.sleep(1)
1139. def position\_move0\_1(self, po1, po2):
1140. self.arm\_move(po1, 1000)
1141. self.arm\_clamp\_block(1)
1142. time.sleep(1)
1143. self.arm\_move\_0\_1()
1144. time.sleep(1)
1145. self.arm\_move(po2, 1000)
1146. self.arm\_clamp\_block(0)
1147. time.sleep(1)
1148. def move\_player(self, player, steps):
1149. if player == 1:
1150. if self.player1\_position == -1:  *# Piece not on board*
1151. if steps == 6 or self.dice\_count1 == 2:
1152. self.player1\_position = 0  *# Place piece at start position*
1153. self.position\_move0\_1(self.p1\_1, self.p1)
1154. print(f"Player {player} places the piece at the starting position.")
1155. self.dice\_count1 = 0
1156. print(f"Dice count for Player 1: {self.dice\_count1}")
1157. else:
1158. print(f"Player {player} needs to roll a 6 to start.")
1159. self.dice\_count1 += 1
1160. print(f"Dice count for Player 1: {self.dice\_count1}")
1161. return False
1162. else:
1163. old\_player1\_position = self.player1\_position
1164. self.player1\_position += steps
1165. if self.player1\_position > self.board\_length:
1166. steps = self.board\_length - old\_player1\_position
1167. self.player1\_position = self.board\_length
1168. print(f"Player {player} moves to position {self.player1\_position}")
1169. self.position\_move2(self.p1, steps)
1170. if self.player1\_position == self.board\_length:
1171. print(f"Player {player} wins!")
1172. self.win()
1173. b\_time = 6
1174. self.Arm.Arm\_Buzzer\_On(b\_time)
1175. return True
1176. elif player == 2:
1177. print("Player 2's turn (human player):")
1178. print("Rolling dice...")
1179. b\_time = 7
1180. self.Arm.Arm\_Buzzer\_On(b\_time)
1181. time.sleep(3)
1182. p\_vidio = [90, 78, 24, 13, 90]
1183. self.arm\_move(p\_vidio, 1000)
1184. time.sleep(3)
1185. steps = self.getdice\_num()
1186. while steps == -1:
1187. print("Retrying dice throw...")
1188. steps = self.getdice\_num()
1189. print(f"Player {player} rolls {steps}")
1190. if self.player2\_position == -1:  *# Piece not on board*
1191. if steps == 6 or self.dice\_count2 == 2:
1192. self.player2\_position = 0  *# Place piece at start position*
1193. print(f"Player {player} places the piece at the starting position.")
1194. self.dice\_count2 = 0
1195. print(f"Dice count for Player 2: {self.dice\_count2}")
1196. else:
1197. print(f"Player {player} needs to roll a 6 to start.")
1198. self.dice\_count2 += 1
1199. print(f"Dice count for Player 2: {self.dice\_count2}")
1200. return False
1201. else:
1202. old\_player2\_position = self.player2\_position
1203. self.player2\_position += steps
1204. if self.player2\_position > self.board\_length:
1205. steps = self.board\_length - old\_player2\_position
1206. self.player2\_position = self.board\_length
1207. print(f"Player {player} moves to position {self.player2\_position}")
1208. if self.player2\_position == self.board\_length:
1209. print(f"Player {player} wins!")
1210. self.fail()
1211. b\_time = 3
1212. self.Arm.Arm\_Buzzer\_On(b\_time)
1213. return True
1214. else:
1215. print(f"Unknown player: {player}")
1216. return False
1217. return False
1218. def start\_person\_game(self):
1219. current\_player = 1
1220. try:
1221. while True:
1222. print(f"Player {current\_player}'s turn:")
1223. print("Rolling dice...")
1224. if current\_player == 1:
1225. steps = self.roll\_dice()
1226. print(f"Player {current\_player} rolls {steps}")
1227. else:
1228. print("Player 2's turn (human player):")
1229. b\_time = 7
1230. self.Arm.Arm\_Buzzer\_On(b\_time)
1231. time.sleep(3)
1232. p\_vidio = [90, 78, 24, 13, 90]
1233. self.arm\_move(p\_vidio, 1000)
1234. time.sleep(3)
1235. steps = self.getdice\_num()
1236. self.person\_speak(steps)
1237. while steps == -1:
1238. print("Retrying dice throw...")
1239. steps = self.getdice\_num()
1240. print(f"Player {current\_player} rolls {steps}")
1241. if self.move\_player(current\_player, steps):
1242. break
1243. current\_player = 2 if current\_player == 1 else 1
1244. except KeyboardInterrupt:
1245. cap.release()
1246. print("\nProgram closed!")
1247. finally:
1248. guiwei = [90, 90, 90, 90, 90, 90, 500]
1249. game.arm\_move(guiwei, 1000)
1250. time.sleep(1)
1251. if \_\_name\_\_ == "\_\_main\_\_":
1252. try:
1253. game = person\_DiceGame()
1254. game.start\_person\_game()
1255. except KeyboardInterrupt:
1256. print("\nProgram closed!")
1257. finally:
1258. guiwei = [90, 90, 90, 90, 90, 90, 500]
1259. game.arm\_move(guiwei, 1000)
1260. time.sleep(1)
1261. play\_music:演奏模块代码。
1262. *#!/usr/bin/env python3*
1263. *#coding=utf-8*
1264. import time
1265. from Arm\_Lib import Arm\_Device
1266. import smbus
1267. import time
1268. from speak import speak
1269. class RoboticArmController:
1270. def \_\_init\_\_(self):
1271. self.arm = Arm\_Device()  *# Initialize the robotic arm*
1272. self.init\_arm\_positions()  *# Initialize arm to default position*
1273. self.time\_sleep = 0.5  *# Define default sleep time*
1274. self.bus = smbus.SMBus(1)
1275. self.seki2c\_addr = 0x30
1276. self.date\_head = 0xfd
1277. global i2c\_addr
1278. *# Define arm positions for different notes*
1279. self.positions = {
1280. 0: [160, 140, 20, 25, 89],
1281. 1: [63, 129, 25, 40, 180],
1282. 2: [71, 138, 13, 40, 180],
1283. 3: [77, 140, 13, 40, 180],
1284. 4: [83, 148, 6, 40, 180],
1285. 5: [89, 148, 5, 40, 180],
1286. 6: [96, 148, 5, 40, 180],
1287. 7: [100, 148, 5, 40, 180],
1288. 8: [106, 148, 5, 40, 180],
1289. }
1290. Reader\_Type = {
1291. 'Reader\_XiaoYan': 3,  *# 为 3，设置发音人为小燕(女声, 推荐发音人)*
1292. 'Reader\_XuJiu': 51,  *# 为 51，设置发音人为许久(男声, 推荐发音人)*
1293. 'Reader\_XuDuo': 52,  *# 为 52，设置发音人为许多(男声)*
1294. 'Reader\_XiaoPing': 53,  *# 为 53，设置发音人为小萍(女声*
1295. 'Reader\_DonaldDuck': 54,  *# 为 54，设置发音人为唐老鸭(效果器)*
1296. 'Reader\_XuXiaoBao': 55  *# 为 55，设置发音人为许小宝(女童声)*
1297. }  *# 选择发音人 [m?]'*
1298. def init\_arm\_positions(self):
1299. self.arm.Arm\_serial\_servo\_write6(90, 143, 20, 25, 89, 90, 500)
1300. time.sleep(1)
1301. def arm\_clamp\_block(self, enable):
1302. if enable == 0:
1303. self.arm.Arm\_serial\_servo\_write(6, 40, 400)
1304. else:
1305. self.arm.Arm\_serial\_servo\_write(6, 180, 400)
1306. time.sleep(self.time\_sleep)
1307. def arm\_move(self, p, s\_time=500):
1308. for i in range(5):
1309. id = i + 1
1310. self.arm.Arm\_serial\_servo\_write(id, p[i], s\_time)
1311. time.sleep(0.01)
1312. time.sleep(s\_time / 1000)
1313. def arm\_qiao(self, enable):
1314. if enable == 0:
1315. self.arm.Arm\_serial\_servo\_write(4, 30, 200)  *# Lift*
1316. else:
1317. self.arm.Arm\_serial\_servo\_write(4, 0, 100)  *# Strike*
1318. time.sleep(self.time\_sleep)
1319. def arm\_qiaoji(self):
1320. time.sleep(0.01)
1321. self.arm\_qiao(1)
1322. time.sleep(1)
1323. self.arm\_qiao(0)
1324. time.sleep(0.01)
1325. def arm\_return(self):
1326. c\_time = 2
1327. self.arm.Arm\_Buzzer\_On(c\_time)
1328. self.arm\_move(self.positions[0], 500)
1329. self.arm\_clamp\_block(0)
1330. def person\_speak(self, num):
1331. myspeak=speak()
1332. if num == 1:
1333. myspeak.SetReader(myspeak.Reader\_Type["Reader\_DonaldDuck"])  *# 选择播音人许多*
1334. myspeak.Speech\_text("哆", myspeak.EncodingFormat\_Type["GB2312"])
1335. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1336. time.sleep(0.1)
1337. elif num == 2:
1338. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XuJiu"])  *# 选择播音人许多*
1339. myspeak.Speech\_text("来", myspeak.EncodingFormat\_Type["GB2312"])
1340. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1341. time.sleep(0.1)
1342. elif num == 3:
1343. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XuDuo"])  *# 选择播音人许多*
1344. myspeak.Speech\_text("咪", myspeak.EncodingFormat\_Type["GB2312"])
1345. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1346. time.sleep(0.1)
1347. elif num == 4:
1348. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1349. myspeak.Speech\_text("发", myspeak.EncodingFormat\_Type["GB2312"])
1350. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1351. time.sleep(0.1)
1352. elif num == 5:
1353. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1354. myspeak.Speech\_text("嗦", myspeak.EncodingFormat\_Type["GB2312"])
1355. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1356. time.sleep(0.1)
1357. elif num == 6:
1358. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoPing"])  *# 选择播音人许多*
1359. myspeak.Speech\_text("拉", myspeak.EncodingFormat\_Type["GB2312"])
1360. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1361. time.sleep(0.1)
1362. elif num == 7:
1363. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoPing"])  *# 选择播音人许多*
1364. myspeak.Speech\_text("西", myspeak.EncodingFormat\_Type["GB2312"])
1365. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1366. time.sleep(0.1)
1367. elif num == 8:
1368. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XuXiaoBao"])  *# 选择播音人许多*
1369. myspeak.Speech\_text("do", myspeak.EncodingFormat\_Type["GB2312"])
1370. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1371. time.sleep(0.1)
1372. def main\_game(self):
1373. myspeak=speak()
1374. try:
1375. *# Initialize*
1376. self.init\_arm\_positions()
1377. *# Grab hammer*
1378. b\_time = 10
1379. self.arm.Arm\_Buzzer\_On(b\_time)
1380. time.sleep(4)
1381. a\_time = 3
1382. self.arm.Arm\_Buzzer\_On(a\_time)
1383. self.arm\_clamp\_block(1)
1384. time.sleep(2)
1385. self.arm.Arm\_serial\_servo\_write(5, 180, 400)
1386. time.sleep(3)
1387. *# Play musical sequence*
1388. for note in q1:
1389. if note in self.positions:
1390. self.arm\_move(self.positions[note], 300)
1391. time.sleep(0.01)
1392. self.arm\_qiaoji()
1393. self.person\_speak(note)
1394. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1395. time.sleep(0.1)
1396. print(note)
1397. else:
1398. print(f"Note {note} is not mapped to any position.")
1399. time.sleep(0.01)
1400. self.arm\_return()
1401. except KeyboardInterrupt:
1402. print("\nProgram closed!")
1403. finally:
1404. *# Reset arm to default position*
1405. guiwei = [90, 90, 90, 90, 90, 90, 500]
1406. self.arm\_move(guiwei, 1000)
1407. time.sleep(1)
1408. *# Clean up or disconnect arm if applicable*
1409. *# self.arm.disconnect()  # Uncomment this line if there's a disconnect method*
1410. *# Define your musical sequence*
1411. q1 = [1, 1, 5, 5, 6, 6, 5, 4, 4, 3, 3, 2, 2, 1, 5, 5, 4, 4, 3, 3, 2, 5, 5, 4, 4, 3, 3, 2]
1412. *# Main program execution*
1413. if \_\_name\_\_ == "\_\_main\_\_":
1414. arm\_controller = RoboticArmController()
1415. arm\_controller.main\_game()
1416. dance:舞蹈模块代码。
1417. import cv2
1418. import time
1419. import demjson
1420. import pygame
1421. from aip import AipBodyAnalysis
1422. from PIL import Image, ImageDraw, ImageFont
1423. import numpy as np
1424. import ipywidgets.widgets as widgets
1425. import random
1426. from Arm\_Lib import Arm\_Device
1427. from IPython.display import display
1428. from speak import speak
1429. Arm = Arm\_Device()
1430. class dance:
1431. def \_\_init\_\_(self) -> None:
1432. *# Replace with your own Baidu AI credentials*
1433. APP\_ID = '93097451'
1434. API\_KEY = '3om56y5ScOOn36Em2UJXbMZ4'
1435. SECRET\_KEY = '26VMq5kQqjdfNfkalZVNq6pjdF5yYNcs'
1436. self.client = AipBodyAnalysis(APP\_ID, API\_KEY, SECRET\_KEY)
1437. *# Initialize camera*
1438. *# self.g\_camera = cv2.VideoCapture(1)*
1439. *# Mapping of gesture class names to text descriptions*
1440. self.hand = {'One': '数字 1', 'Two': '数字 2', 'Three': '数字 3', 'Four': '数字 4',
1441. 'Five': '数字 5', 'Six': '数字 6', 'Seven': '数字 7',
1442. 'Eight': '数字 8', 'Nine': '数字 9', 'Fist': '拳头', 'Ok': 'OK',
1443. 'Prayer': '祈祷', 'Congratulation': '作揖', 'Honour': '作别',
1444. 'Heart\_single': '比心心', 'Thumb\_up': '点赞', 'Thumb\_down': 'Diss',
1445. 'ILY': '我爱你', 'Palm\_up': '掌心向上', 'Heart\_1': '双手比心 1',
1446. 'Heart\_2': '双手比心 2', 'Heart\_3': '双手比心 3', 'Rock': 'Rock',
1447. 'Insult': '竖中指', 'Face': '脸'}
1448. *# Define time intervals*
1449. self.times = 1
1450. self.time\_1 = 500
1451. self.time\_2 = 1000
1452. self.time\_sleep = 1
1453. self.cap = None
1454. Reader\_Type = {
1455. 'Reader\_XiaoYan': 3,  *# 为 3，设置发音人为小燕(女声, 推荐发音人)*
1456. 'Reader\_XuJiu': 51,  *# 为 51，设置发音人为许久(男声, 推荐发音人)*
1457. 'Reader\_XuDuo': 52,  *# 为 52，设置发音人为许多(男声)*
1458. 'Reader\_XiaoPing': 53,  *# 为 53，设置发音人为小萍(女声*
1459. 'Reader\_DonaldDuck': 54,  *# 为 54，设置发音人为唐老鸭(效果器)*
1460. 'Reader\_XuXiaoBao': 55  *# 为 55，设置发音人为许小宝(女童声)*
1461. }  *# 选择发音人 [m?]'*
1462. *# Function to convert BGR image to JPEG format*
1463. def bgr8\_to\_jpeg(self, image):
1464. image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)
1465. \_, jpeg = cv2.imencode('.jpg', image\_rgb)
1466. return jpeg.tobytes()
1467. *# 定义转换显示中文函数*
1468. def cv2ImgAddText(self, img, text, left, top, textColor=(0, 255, 0), textSize=20):
1469. if (isinstance(img, np.ndarray)):  *# 判断是否OpenCV图片类型*
1470. img = Image.fromarray(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))
1471. *# 创建一个可以在给定图像上绘图的对象*
1472. draw = ImageDraw.Draw(img)
1473. *# 字体的格式*
1474. fontStyle = ImageFont.truetype(
1475. "SettingsIcons.ttf", textSize, encoding="utf-8")
1476. *# 绘制文本*
1477. draw.text((left, top), text, textColor, font=fontStyle)
1478. *# 转换回OpenCV格式*
1479. return cv2.cvtColor(np.asarray(img), cv2.COLOR\_RGB2BGR)
1480. def start(self):
1481. myspeak = speak()
1482. *#         g\_camera = cv2.VideoCapture(1)*
1483. g\_camera = self.cap
1484. *# 打开摄像头*
1485. g\_camera.set(3, 640)
1486. g\_camera.set(4, 480)
1487. g\_camera.set(5, 30)  *# Frame rate*
1488. g\_camera.set(cv2.CAP\_PROP\_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
1489. g\_camera.set(cv2.CAP\_PROP\_BRIGHTNESS, 20)  *# Brightness*
1490. g\_camera.set(cv2.CAP\_PROP\_CONTRAST, 50)  *# Contrast*
1491. g\_camera.set(cv2.CAP\_PROP\_EXPOSURE, 100)  *# Exposure*
1492. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1493. myspeak.Speech\_text("开始识别", myspeak.EncodingFormat\_Type["GB2312"])
1494. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1495. time.sleep(0.1)
1496. print("识别中")
1497. time.sleep(3)
1498. flag = True
1499. try:
1500. while flag:
1501. Arm.Arm\_serial\_servo\_write6(90, 135, 20, 25, 90, 30, 1000)
1502. ret, frame = g\_camera.read()
1503. if not ret:
1504. print("Error: Could not read frame.")
1505. time.sleep(3)
1506. *# raw = str(self.client.gesture(self.image\_widget.value))*
1507. *# text = demjson.decode(raw)*
1508. self.image\_widget = widgets.Image(format='jpeg', width=600, height=500)
1509. display(self.image\_widget)
1510. *# Capture frame from camera*
1511. *# Convert frame to JPEG format for processing*
1512. frame\_jpeg = self.bgr8\_to\_jpeg(frame)
1513. *# Perform gesture recognition*
1514. gesture\_result = self.client.gesture(frame\_jpeg)
1515. raw\_result = str(gesture\_result)
1516. parsed\_result = demjson.decode(raw\_result)
1517. try:
1518. res = parsed\_result['result'][0]['classname']
1519. except:
1520. print('识别结果：未能识别到手势')
1521. img = frame
1522. else:
1523. print('识别结果：' + self.hand.get(res, '未知手势'))
1524. if res == "Thumb\_up":
1525. print('识别结果' + self.hand.get(res, '未知手势'))
1526. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1527. myspeak.Speech\_text("小花看到你比的大拇哥啦", myspeak.EncodingFormat\_Type["GB2312"])
1528. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1529. time.sleep(0.1)
1530. flag = False
1531. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_2)
1532. time.sleep(self.times)
1533. Arm.Arm\_serial\_servo\_write(1, 90, self.time\_1)
1534. time.sleep(self.times)
1535. Arm.Arm\_serial\_servo\_write(1, 90, self.time\_1)
1536. time.sleep(self.times)
1537. Arm.Arm\_serial\_servo\_write(5, 90, self.time\_1)
1538. time.sleep(self.time\_sleep)
1539. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1540. time.sleep(self.times)
1541. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1542. time.sleep(self.time\_sleep)
1543. Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
1544. time.sleep(self.times)
1545. Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
1546. time.sleep(self.times)
1547. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1548. time.sleep(self.time\_sleep)
1549. Arm.Arm\_serial\_servo\_write(2, 180 - 135, self.time\_1)
1550. time.sleep(self.times)
1551. Arm.Arm\_serial\_servo\_write(3, 135, self.time\_1)
1552. time.sleep(self.times)
1553. Arm.Arm\_serial\_servo\_write(4, 45, self.time\_1)
1554. time.sleep(self.time\_sleep)
1555. Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
1556. time.sleep(self.times)
1557. Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
1558. time.sleep(self.times)
1559. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1560. time.sleep(self.time\_sleep)
1561. Arm.Arm\_serial\_servo\_write(2, 90, self.time\_1)
1562. time.sleep(self.times)
1563. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1564. time.sleep(self.times)
1565. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1566. time.sleep(self.time\_sleep)
1567. Arm.Arm\_serial\_servo\_write(2, 180 - 80, self.time\_1)
1568. time.sleep(self.times)
1569. Arm.Arm\_serial\_servo\_write(3, 80, self.time\_1)
1570. time.sleep(self.times)
1571. Arm.Arm\_serial\_servo\_write(4, 80, self.time\_1)
1572. time.sleep(self.time\_sleep)
1573. Arm.Arm\_serial\_servo\_write(2, 180 - 60, self.time\_1)
1574. time.sleep(self.times)
1575. Arm.Arm\_serial\_servo\_write(3, 60, self.time\_1)
1576. time.sleep(self.times)
1577. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1578. time.sleep(self.time\_sleep)
1579. Arm.Arm\_serial\_servo\_write(2, 180 - 45, self.time\_1)
1580. time.sleep(self.times)
1581. Arm.Arm\_serial\_servo\_write(3, 45, self.time\_1)
1582. time.sleep(self.times)
1583. Arm.Arm\_serial\_servo\_write(4, 45, self.time\_1)
1584. time.sleep(self.time\_sleep)
1585. Arm.Arm\_serial\_servo\_write(2, 90, self.time\_1)
1586. time.sleep(self.times)
1587. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1588. time.sleep(self.times)
1589. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1590. time.sleep(self.times)
1591. time.sleep(self.time\_sleep)
1592. Arm.Arm\_serial\_servo\_write(4, 20, self.time\_1)
1593. time.sleep(self.times)
1594. Arm.Arm\_serial\_servo\_write(6, 150, self.time\_1)
1595. time.sleep(self.times)
1596. time.sleep(self.time\_sleep)
1597. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1598. time.sleep(self.times)
1599. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_1)
1600. time.sleep(self.time\_sleep)
1601. Arm.Arm\_serial\_servo\_write(4, 20, self.time\_1)
1602. time.sleep(self.times)
1603. Arm.Arm\_serial\_servo\_write(6, 150, self.time\_1)
1604. time.sleep(self.time\_sleep)
1605. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1606. time.sleep(self.times)
1607. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_1)
1608. time.sleep(self.times)
1609. Arm.Arm\_serial\_servo\_write(1, 0, self.time\_1)
1610. time.sleep(self.times)
1611. Arm.Arm\_serial\_servo\_write(5, 0, self.time\_1)
1612. time.sleep(self.time\_sleep)
1613. Arm.Arm\_serial\_servo\_write(3, 180, self.time\_1)
1614. time.sleep(self.times)
1615. Arm.Arm\_serial\_servo\_write(4, 0, self.time\_1)
1616. time.sleep(self.time\_sleep)
1617. Arm.Arm\_serial\_servo\_write(6, 180, self.time\_1)
1618. time.sleep(self.time\_sleep)
1619. Arm.Arm\_serial\_servo\_write(6, 0, self.time\_2)
1620. time.sleep(self.time\_sleep)
1621. print("well done")
1622. elif res == "Fist":
1623. print('识别结果' + self.hand.get(res, '未知手势'))
1624. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1625. myspeak.Speech\_text("小花看到你比的拳头啦", myspeak.EncodingFormat\_Type["GB2312"])
1626. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1627. time.sleep(0.1)
1628. flag = False
1629. Arm.Arm\_serial\_servo\_write(2, 180 - 60, self.time\_1)
1630. time.sleep(self.times)
1631. Arm.Arm\_serial\_servo\_write(3, 60, self.time\_1)
1632. time.sleep(self.times)
1633. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1634. time.sleep(self.time\_sleep)
1635. Arm.Arm\_serial\_servo\_write(2, 180 - 45, self.time\_1)
1636. time.sleep(self.times)
1637. Arm.Arm\_serial\_servo\_write(3, 45, self.time\_1)
1638. time.sleep(self.times)
1639. Arm.Arm\_serial\_servo\_write(4, 45, self.time\_1)
1640. time.sleep(self.time\_sleep)
1641. Arm.Arm\_serial\_servo\_write(2, 90, self.time\_1)
1642. time.sleep(self.times)
1643. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1644. time.sleep(self.times)
1645. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1646. time.sleep(self.times)
1647. time.sleep(self.time\_sleep)
1648. Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
1649. time.sleep(self.times)
1650. Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
1651. time.sleep(self.times)
1652. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1653. time.sleep(self.time\_sleep)
1654. Arm.Arm\_serial\_servo\_write(2, 180 - 135, self.time\_1)
1655. time.sleep(self.times)
1656. Arm.Arm\_serial\_servo\_write(3, 135, self.time\_1)
1657. time.sleep(self.times)
1658. Arm.Arm\_serial\_servo\_write(4, 45, self.time\_1)
1659. time.sleep(self.time\_sleep)
1660. Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
1661. time.sleep(self.times)
1662. Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
1663. time.sleep(self.times)
1664. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1665. time.sleep(self.time\_sleep)
1666. Arm.Arm\_serial\_servo\_write(2, 90, self.time\_1)
1667. time.sleep(self.times)
1668. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1669. time.sleep(self.times)
1670. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1671. time.sleep(self.time\_sleep)
1672. Arm.Arm\_serial\_servo\_write(2, 180 - 80, self.time\_1)
1673. time.sleep(self.times)
1674. Arm.Arm\_serial\_servo\_write(3, 80, self.time\_1)
1675. time.sleep(self.times)
1676. Arm.Arm\_serial\_servo\_write(4, 80, self.time\_1)
1677. time.sleep(self.time\_sleep)
1678. Arm.Arm\_serial\_servo\_write(4, 20, self.time\_1)
1679. time.sleep(self.times)
1680. Arm.Arm\_serial\_servo\_write(6, 150, self.time\_1)
1681. time.sleep(self.times)
1682. time.sleep(self.time\_sleep)
1683. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1684. time.sleep(self.times)
1685. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_1)
1686. time.sleep(self.time\_sleep)
1687. Arm.Arm\_serial\_servo\_write(4, 20, self.time\_1)
1688. time.sleep(self.times)
1689. Arm.Arm\_serial\_servo\_write(6, 150, self.time\_1)
1690. time.sleep(self.time\_sleep)
1691. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1692. time.sleep(self.times)
1693. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_1)
1694. time.sleep(self.times)
1695. Arm.Arm\_serial\_servo\_write(1, 0, self.time\_1)
1696. time.sleep(self.times)
1697. Arm.Arm\_serial\_servo\_write(5, 0, self.time\_1)
1698. time.sleep(self.time\_sleep)
1699. Arm.Arm\_serial\_servo\_write(3, 180, self.time\_1)
1700. time.sleep(self.times)
1701. Arm.Arm\_serial\_servo\_write(4, 0, self.time\_1)
1702. time.sleep(self.time\_sleep)
1703. Arm.Arm\_serial\_servo\_write(6, 180, self.time\_1)
1704. time.sleep(self.time\_sleep)
1705. Arm.Arm\_serial\_servo\_write(6, 0, self.time\_2)
1706. time.sleep(self.time\_sleep)
1707. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_2)
1708. time.sleep(self.times)
1709. Arm.Arm\_serial\_servo\_write(1, 90, self.time\_1)
1710. time.sleep(self.times)
1711. Arm.Arm\_serial\_servo\_write(1, 90, self.time\_1)
1712. time.sleep(self.times)
1713. Arm.Arm\_serial\_servo\_write(5, 90, self.time\_1)
1714. time.sleep(self.time\_sleep)
1715. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1716. time.sleep(self.times)
1717. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1718. time.sleep(self.time\_sleep)
1719. print("well done")
1720. elif res == "Ok":
1721. print('识别结果' + self.hand.get(res, '未知手势'))
1722. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1723. myspeak.Speech\_text("小花看到你比的OK啦", myspeak.EncodingFormat\_Type["GB2312"])
1724. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1725. time.sleep(0.1)
1726. flag = False
1727. Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
1728. time.sleep(self.times)
1729. Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
1730. time.sleep(self.times)
1731. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1732. time.sleep(self.time\_sleep)
1733. Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
1734. time.sleep(self.times)
1735. Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
1736. time.sleep(self.times)
1737. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1738. time.sleep(self.time\_sleep)
1739. Arm.Arm\_serial\_servo\_write(2, 180 - 135, self.time\_1)
1740. time.sleep(self.times)
1741. Arm.Arm\_serial\_servo\_write(3, 135, self.time\_1)
1742. time.sleep(self.times)
1743. Arm.Arm\_serial\_servo\_write(4, 45, self.time\_1)
1744. time.sleep(self.time\_sleep)
1745. Arm.Arm\_serial\_servo\_write(2, 90, self.time\_1)
1746. time.sleep(self.times)
1747. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1748. time.sleep(self.times)
1749. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1750. time.sleep(self.time\_sleep)
1751. Arm.Arm\_serial\_servo\_write(2, 180 - 80, self.time\_1)
1752. time.sleep(self.times)
1753. Arm.Arm\_serial\_servo\_write(3, 80, self.time\_1)
1754. time.sleep(self.times)
1755. Arm.Arm\_serial\_servo\_write(4, 80, self.time\_1)
1756. time.sleep(self.time\_sleep)
1757. Arm.Arm\_serial\_servo\_write(2, 180 - 60, self.time\_1)
1758. time.sleep(self.times)
1759. Arm.Arm\_serial\_servo\_write(3, 60, self.time\_1)
1760. time.sleep(self.times)
1761. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1762. time.sleep(self.time\_sleep)
1763. Arm.Arm\_serial\_servo\_write(2, 180 - 45, self.time\_1)
1764. time.sleep(self.times)
1765. Arm.Arm\_serial\_servo\_write(3, 45, self.time\_1)
1766. time.sleep(self.times)
1767. Arm.Arm\_serial\_servo\_write(4, 45, self.time\_1)
1768. time.sleep(self.time\_sleep)
1769. Arm.Arm\_serial\_servo\_write(2, 90, self.time\_1)
1770. time.sleep(self.times)
1771. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1772. time.sleep(self.times)
1773. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1774. time.sleep(self.times)
1775. time.sleep(self.time\_sleep)
1776. Arm.Arm\_serial\_servo\_write(3, 180, self.time\_1)
1777. time.sleep(self.times)
1778. Arm.Arm\_serial\_servo\_write(4, 0, self.time\_1)
1779. time.sleep(self.time\_sleep)
1780. Arm.Arm\_serial\_servo\_write(6, 180, self.time\_1)
1781. time.sleep(self.time\_sleep)
1782. Arm.Arm\_serial\_servo\_write(6, 0, self.time\_2)
1783. time.sleep(self.time\_sleep)
1784. Arm.Arm\_serial\_servo\_write(4, 20, self.time\_1)
1785. time.sleep(self.times)
1786. Arm.Arm\_serial\_servo\_write(6, 150, self.time\_1)
1787. time.sleep(self.times)
1788. time.sleep(self.time\_sleep)
1789. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1790. time.sleep(self.times)
1791. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_1)
1792. time.sleep(self.time\_sleep)
1793. Arm.Arm\_serial\_servo\_write(4, 20, self.time\_1)
1794. time.sleep(self.times)
1795. Arm.Arm\_serial\_servo\_write(6, 150, self.time\_1)
1796. time.sleep(self.time\_sleep)
1797. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1798. time.sleep(self.times)
1799. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_1)
1800. time.sleep(self.times)
1801. Arm.Arm\_serial\_servo\_write(1, 0, self.time\_1)
1802. time.sleep(self.times)
1803. Arm.Arm\_serial\_servo\_write(5, 0, self.time\_1)
1804. time.sleep(self.time\_sleep)
1805. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_2)
1806. time.sleep(self.times)
1807. Arm.Arm\_serial\_servo\_write(1, 90, self.time\_1)
1808. time.sleep(self.times)
1809. Arm.Arm\_serial\_servo\_write(1, 90, self.time\_1)
1810. time.sleep(self.times)
1811. Arm.Arm\_serial\_servo\_write(5, 90, self.time\_1)
1812. time.sleep(self.time\_sleep)
1813. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1814. time.sleep(self.times)
1815. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1816. time.sleep(self.time\_sleep)
1817. print("well done")
1818. elif res == "Face":
1819. print('识别结果' + self.hand.get(res, '未知手势'))
1820. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1821. myspeak.Speech\_text("小花看到小主人的脸啦", myspeak.EncodingFormat\_Type["GB2312"])
1822. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1823. time.sleep(0.1)
1824. flag = False
1825. Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
1826. time.sleep(self.times)
1827. Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
1828. time.sleep(self.times)
1829. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1830. time.sleep(self.time\_sleep)
1831. Arm.Arm\_serial\_servo\_write(2, 180 - 135, self.time\_1)
1832. time.sleep(self.times)
1833. Arm.Arm\_serial\_servo\_write(3, 135, self.time\_1)
1834. time.sleep(self.times)
1835. Arm.Arm\_serial\_servo\_write(4, 45, self.time\_1)
1836. time.sleep(self.time\_sleep)
1837. Arm.Arm\_serial\_servo\_write(2, 180 - 120, self.time\_1)
1838. time.sleep(self.times)
1839. Arm.Arm\_serial\_servo\_write(3, 120, self.time\_1)
1840. time.sleep(self.times)
1841. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1842. time.sleep(self.time\_sleep)
1843. Arm.Arm\_serial\_servo\_write(2, 90, self.time\_1)
1844. time.sleep(self.times)
1845. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1846. time.sleep(self.times)
1847. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1848. time.sleep(self.time\_sleep)
1849. Arm.Arm\_serial\_servo\_write(2, 180 - 80, self.time\_1)
1850. time.sleep(self.times)
1851. Arm.Arm\_serial\_servo\_write(3, 80, self.time\_1)
1852. time.sleep(self.times)
1853. Arm.Arm\_serial\_servo\_write(4, 80, self.time\_1)
1854. time.sleep(self.time\_sleep)
1855. Arm.Arm\_serial\_servo\_write(2, 180 - 60, self.time\_1)
1856. time.sleep(self.times)
1857. Arm.Arm\_serial\_servo\_write(3, 60, self.time\_1)
1858. time.sleep(self.times)
1859. Arm.Arm\_serial\_servo\_write(4, 60, self.time\_1)
1860. time.sleep(self.time\_sleep)
1861. Arm.Arm\_serial\_servo\_write(2, 180 - 45, self.time\_1)
1862. time.sleep(self.times)
1863. Arm.Arm\_serial\_servo\_write(3, 45, self.time\_1)
1864. time.sleep(self.times)
1865. Arm.Arm\_serial\_servo\_write(4, 45, self.time\_1)
1866. time.sleep(self.time\_sleep)
1867. Arm.Arm\_serial\_servo\_write(2, 90, self.time\_1)
1868. time.sleep(self.times)
1869. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1870. time.sleep(self.times)
1871. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1872. time.sleep(self.times)
1873. time.sleep(self.time\_sleep)
1874. Arm.Arm\_serial\_servo\_write(4, 20, self.time\_1)
1875. time.sleep(self.times)
1876. Arm.Arm\_serial\_servo\_write(6, 150, self.time\_1)
1877. time.sleep(self.times)
1878. time.sleep(self.time\_sleep)
1879. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1880. time.sleep(self.times)
1881. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_1)
1882. time.sleep(self.time\_sleep)
1883. Arm.Arm\_serial\_servo\_write(4, 20, self.time\_1)
1884. time.sleep(self.times)
1885. Arm.Arm\_serial\_servo\_write(6, 150, self.time\_1)
1886. time.sleep(self.time\_sleep)
1887. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_2)
1888. time.sleep(self.times)
1889. Arm.Arm\_serial\_servo\_write(1, 90, self.time\_1)
1890. time.sleep(self.times)
1891. Arm.Arm\_serial\_servo\_write(1, 90, self.time\_1)
1892. time.sleep(self.times)
1893. Arm.Arm\_serial\_servo\_write(5, 90, self.time\_1)
1894. time.sleep(self.time\_sleep)
1895. Arm.Arm\_serial\_servo\_write(3, 90, self.time\_1)
1896. time.sleep(self.times)
1897. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1898. time.sleep(self.time\_sleep)
1899. Arm.Arm\_serial\_servo\_write(3, 180, self.time\_1)
1900. time.sleep(self.times)
1901. Arm.Arm\_serial\_servo\_write(4, 0, self.time\_1)
1902. time.sleep(self.time\_sleep)
1903. Arm.Arm\_serial\_servo\_write(6, 180, self.time\_1)
1904. time.sleep(self.time\_sleep)
1905. Arm.Arm\_serial\_servo\_write(6, 0, self.time\_2)
1906. time.sleep(self.time\_sleep)
1907. Arm.Arm\_serial\_servo\_write(4, 90, self.time\_1)
1908. time.sleep(self.times)
1909. Arm.Arm\_serial\_servo\_write(6, 90, self.time\_1)
1910. time.sleep(self.times)
1911. Arm.Arm\_serial\_servo\_write(1, 0, self.time\_1)
1912. time.sleep(self.times)
1913. Arm.Arm\_serial\_servo\_write(5, 0, self.time\_1)
1914. time.sleep(self.time\_sleep)
1915. else:
1916. img = frame
1917. *# Update the image widget with the annotated frame*
1918. *# self.image\_widget.value = self.bgr8\_to\_jpeg(img)*
1919. except KeyboardInterrupt:
1920. cap.release()
1921. print("程序已关闭！")
1922. pass
1923. if \_\_name\_\_ == "\_\_main\_\_":
1924. *#     arm\_controller = RoboticArmController()*
1925. *#     arm\_controller.main\_game()*
1926. dance = dance()
1927. dance.start()
1928. code:冷笑话模块代码。
1929. import cv2
1930. import pyzbar.pyzbar as pyzbar
1931. from Arm\_Lib import Arm\_Device
1932. import ipywidgets as widgets
1933. import time
1934. import random
1935. from speak import speak
1936. class QRCodeScanner:
1937. def \_\_init\_\_(self):
1938. self.arm = Arm\_Device()
1939. time.sleep(0.1)
1940. self.cap = None
1941. def arm\_move(self, p, s\_time=500):
1942. for i in range(5):
1943. id = i + 1
1944. self.arm.Arm\_serial\_servo\_write(id, p[i], s\_time)
1945. time.sleep(0.01)
1946. time.sleep(s\_time / 1000)
1947. Reader\_Type = {
1948. 'Reader\_XiaoYan': 3,  *# 为 3，设置发音人为小燕(女声, 推荐发音人)*
1949. 'Reader\_XuJiu': 51,  *# 为 51，设置发音人为许久(男声, 推荐发音人)*
1950. 'Reader\_XuDuo': 52,  *# 为 52，设置发音人为许多(男声)*
1951. 'Reader\_XiaoPing': 53,  *# 为 53，设置发音人为小萍(女声*
1952. 'Reader\_DonaldDuck': 54,  *# 为 54，设置发音人为唐老鸭(效果器)*
1953. 'Reader\_XuXiaoBao': 55  *# 为 55，设置发音人为许小宝(女童声)*
1954. }  *# 选择发音人 [m?]'*
1955. def decode\_display(self, image, image1, flag):
1956. print('begin code')
1957. myspeak = speak()
1958. barcodes = pyzbar.decode(image)
1959. for barcode in barcodes:
1960. (x, y, w, h) = barcode.rect
1961. cv2.rectangle(image1, (x, y), (x + w, y + h), (0, 0, 255), 2)
1962. barcodeData = barcode.data.decode("utf-8")
1963. barcodeType = barcode.type
1964. print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData))
1965. if barcodeData == '1':
1966. print('1 is ok')
1967. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1968. myspeak.Speech\_text("为什么小白兔不嫁给斑马呢,因为兔妈妈说纹身不是好孩子",
1969. myspeak.EncodingFormat\_Type["GB2312"])
1970. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1971. time.sleep(0.1)
1972. flag = False
1973. elif barcodeData == '2':
1974. print('2 is ok')
1975. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1976. myspeak.Speech\_text("为什么蚕宝宝很有钱,因为它会结茧", myspeak.EncodingFormat\_Type["GB2312"])
1977. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1978. time.sleep(0.1)
1979. flag = False
1980. elif barcodeData == '3':
1981. print('3 is ok')
1982. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1983. myspeak.Speech\_text("猴子不喜欢什么线,平行线,因为没有相交", myspeak.EncodingFormat\_Type["GB2312"])
1984. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1985. time.sleep(0.1)
1986. flag = False
1987. elif barcodeData == '4':
1988. print('4 is ok')
1989. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1990. myspeak.Speech\_text("巧克力和西红柿打架，为什么是巧克力赢,因为巧克力棒嘛",
1991. myspeak.EncodingFormat\_Type["GB2312"])
1992. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
1993. time.sleep(0.1)
1994. flag = False
1995. elif barcodeData == '5':
1996. print('5 is ok')
1997. myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  *# 选择播音人许多*
1998. myspeak.Speech\_text("鲨鱼吃了绿豆会怎么样,变成了绿豆沙", myspeak.EncodingFormat\_Type["GB2312"])
1999. while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  *# 等待当前语句播报结束*
2000. time.sleep(0.1)
2001. flag = False
2002. else:
2003. print('error')
2004. return image1, flag
2005. def bgr8\_to\_jpeg(self, image):
2006. image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)
2007. \_, jpeg = cv2.imencode('.jpg', image\_rgb)
2008. return jpeg.tobytes()
2009. def main(self):
2010. myspeak = speak()
2011. flag = True
2012. p = [90, 135, 20, 25, 90, 30]
2013. self.arm\_move(p, 500)
2014. time.sleep(1)
2015. *#         cap = cv2.VideoCapture(1)*
2016. cap = self.cap
2017. if not cap.isOpened():
2018. print("Error: Could not open camera.")
2019. return -1  *# 返回-1表示摄像头打开失败*
2020. b\_time = 10
2021. a\_time = 1
2022. self.arm.Arm\_Buzzer\_On(b\_time)
2023. time.sleep(10)
2024. self.arm.Arm\_Buzzer\_On(a\_time)
2025. print("begin photo")
2026. *#         myspeak.SetReader(myspeak.Reader\_Type["Reader\_XiaoYan"])  # 选择播音人许多*
2027. *#         myspeak.Speech\_text("小花要开始说花花的笑话了，请小主人不要吃东西哦", myspeak.EncodingFormat\_Type["GB2312"])*
2028. *#         while myspeak.GetChipStatus() != myspeak.ChipStatus\_Type['ChipStatus\_Idle']:  # 等待当前语句播报结束*
2029. *#             time.sleep(0.1)*
2030. while flag:
2031. *# 读取当前帧*
2032. ret, img = cap.read()
2033. *# 打开摄像头*
2034. cap.set(3, 640)
2035. cap.set(4, 480)
2036. cap.set(5, 30)  *# Frame rate*
2037. cap.set(cv2.CAP\_PROP\_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
2038. cap.set(cv2.CAP\_PROP\_BRIGHTNESS, 40)  *# Brightness*
2039. cap.set(cv2.CAP\_PROP\_CONTRAST, 50)  *# Contrast*
2040. cap.set(cv2.CAP\_PROP\_EXPOSURE, 156)  *# Exposure*
2041. *# Read the initial frame*
2042. ret, frame = cap.read()
2043. *# Display widget for camera feed*
2044. *#             image\_widget = widgets.Image(format='jpeg', width=600, height=500)*
2045. *#             display(image\_widget)*
2046. *#             image\_widget.value = self.bgr8\_to\_jpeg(frame)*
2047. b\_time = 10
2048. self.arm.Arm\_Buzzer\_On(b\_time)
2049. print('1')
2050. if not ret:
2051. print("Error: Could not read frame.")
2052. *# 转为灰度图像*
2053. print(f"Frame type: {type(frame)}, Frame shape: {frame.shape}")
2054. gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)
2055. *# 准备开始识别*
2056. c\_time = 1
2057. time.sleep(1)
2058. self.arm.Arm\_Buzzer\_On(c\_time)
2059. time.sleep(0.1)
2060. self.arm.Arm\_Buzzer\_On(c\_time)
2061. *# 进入识别*
2062. im, flag = self.decode\_display(gray, img, flag)
2063. cv2.imshow("image", im)
2064. key = cv2.waitKey(5)
2065. if key == 27:
2066. break
2067. def capture\_qr\_codes(self):
2068. try:
2069. self.main()
2070. except KeyboardInterrupt:
2071. cap.release()
2072. print("\nProgram closed!")
2073. finally:
2074. time.sleep(1)
2075. if \_\_name\_\_ == "\_\_main\_\_":
2076. *#     game = robot\_DiceGame()*
2077. *#     game.start\_robot\_game()*
2078. try:
2079. qr\_scanner = QRCodeScanner()
2080. qr\_scanner.capture\_qr\_codes()
2081. except KeyboardInterrupt:
2082. *#         cap.release()*
2083. print("\nProgram closed!")
2084. finally:
2085. time.sleep(1)