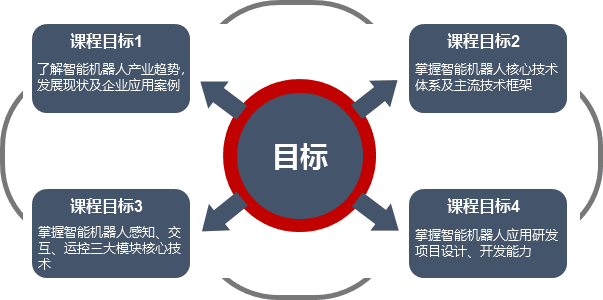
**智能机器人实训讲义**

## 课程能力要求



智能机器人课程目标

在本门课程当中，围绕智能机器人开展理论课程和实验课程的教学，并配套实验项目；学生通过本门课程的学习，需要掌握了解智能机器人的应用开发能力以及一定的基础理论知识，具体目标如上图所示。

本门课程围绕星途机器人进行二次应用研发，不仅仅需要学会怎么去用基础能力模块，而且需要结合机器人的能力模块以及AIUI技能包进行多场景应用，自我学习如何去用、如何用好是本门课程需要掌握的能力；更深层次的，需要学生掌握一定的理论基础，这为后续创新实践提供一定支撑。

## 智能机器人开发平台简介

本门课程的实验平台采用的是智能机器人开发平台“小途”，该产品由芜湖星途机器人科技有限公司自主研发，能够自主导航并实现智能化语音交互、人脸识别等一系列操作。小途集成了科大讯飞AIUI人机语音交互、人脸识别、图像分析处理、室内自主导航等核心技术；同时具备事项办理、咨询、接待、引导、闲聊等多种应用功能，可以作为大型公共场所、办事大厅的移动智能服务窗口。

AIUI 是科大讯飞提供的一套人机智能交互解决方案，旨在实现人机交互无障碍，使人与机器之间可以通过语音、图像、手势等自然交互方式，进行持续、双向、自然地沟通。现阶段 AIUI 提供以语音交互为核心的交互解决方案，全链路聚合了语音唤醒、语音识别、语义理解、内容（信源）平台、语音合成等模块。可以应用于智能手机（终端）、机器人、音箱、车载、智能家居、智能客服等多种领域，让产品不仅能听会说，而且能理解会思考。[AIUI开放平台](http://aiui.iflyos.cn/" \t "_blank)主要包含了语义技能（Skill）、问答库（Q&A）编辑以及AIUI应用（硬件）云端配置的能力，并为不同形态产品提供了不同的接入方式。接入了AIUI的应用和设备可以轻松实现查询天气、播放音视频资源、设置闹钟以及控制智能家居等能力。

#### \*\*\*AIUI平台使用简介

（1）开发前准备

点击讯飞开放平台页面右上角“注册”按钮，在注册页面根据提示信息填写基本资料即可。

（2）创建应用

登录讯飞开放平台后，有两种方式可以创建应用：

1）点击“控制台”->“创建应用”，请根据提示信息，完成应用的创建；

2）点击“产品服务”，进入某个服务页面，如“语音基础能力”，选择右侧的“使用服务”，在弹出的窗口中点击“创建新应用”，请根据提示信息，完成应用的创建。

（3）添加服务

点击“我的应用”，找到需要添加功能的应用，点击下面的“添加新服务”，添加需要的服务。

（4）appid和SDK的关联

创建一个应用时，会自动关联一个appid，appid和对应的SDK具有一致性，如：创建Android平台的应用A，关联的appid是12345678，即12345678和应用A对应的SDK是一一对应关系。

（5）应用开发

围绕应用的具体需求设计，结合当前AIUI能力设计具体业务逻辑，选择合适Android技术进行编码实现，然后编译至智能机器人开发平台上进行调试和演示。

AIUI平台的使用，是本门课程必须掌握的一个环节，应用开发当中涉及到的语音交互、图像识别、语音合成等技术，均需要从平台上调用，一些具有创新的应用均需要这些技术的支撑。具体实现细节，在实验指导书当中给出具体实施流程、步骤以及注意事项，请各位同学在实验课中进行学习和验证。

开始学习前，我们还需要了解如下这些人工智能基础技术：

1. **语音识别技术**

语音识别技术（Auto Speech Recognize，简称ASR）所要解决的问题是让机器能够“听懂”人类的语音，将语音中包含的文字信息“提取”出来，相当于给机器安装上“耳朵”，使其具备“能听”的功能。

语音听写是基于自然语言处理技术，将自然语言转换为文本输出。语音听写技术与语音识别技术的不同在于，语音听写不需要基于某个具体的命令词列表，其识别范围是整个语种内的词条。

1. **语音合成技术**

语音合成，又称文语转换（Text to Speech，TTS）技术，涉及声学、语言学、数字信号处理、计算机科学等多个学科技术，是中文信息处理领域的一项前沿技术，解决的主要问题是如何将文字信息转化为可听的声音信息，也即“让机器像人一样开口说话”。

1. **语义理解技术**

语义理解指将一句自然语言转化为计算机可读的结构化数据。语义理解也是AIUI 的核心功能。例如“明天上海天气怎么样”这句自然语言中，经过AIUI语义理解引擎，得到这句话所对应具有固定格式的Json数据。

1. **人脸识别技术**

人脸识别技术是指利用分析比较的计算机技术识别人脸。人脸识别是一项热门的计算机技术研究领域，其中包括人脸追踪侦测、自动调整影像放大、夜间红外侦测、自动调整曝光强度等技术。目前主要包含三个部分：人脸检测、人脸比对和人脸跟踪，而其具体识别过程主要包括四个组成部分：人脸图像采集及检测、人脸图像预处理、人脸图像特征提取、人脸图像匹配与识别。当前，人脸识别技术比较成熟，2014年汤晓鸥团队原创的人脸识别算法准确率已经超过98.52%，首次超越人脸识别能力（97.53%），在现阶段很多公司已经将目前成熟的人脸识别技术进行能力集成公布在开放平台上供开发者使用，如讯飞的AIUI开放平台：https://www.xfyun.cn/。

1. **OCR技术**

OCR技术是光学字符识别的缩写（Optical Character Recognition），是通过扫描等光学输入方式将各种票据、报刊、书籍、文稿及其它印刷品的文字转化为图像信息，再利用[文字](https://baike.baidu.com/item/%E6%96%87%E5%AD%97" \t "_blank)识别技术将图像信息转化为可以使用的计算机输入技术。如在银行票据光盘缩微系统中，可以自动提取票据要素，可减轻操作员的工作量，减少重复劳动，尤其是在与银行事后且监督系统相结合后，可以替代原先的操作人员完成事后监督工作。由计算机自动识别票据上的日期、帐号、金额等要素，通过银行事后监督系统与业务系统中的数据进行比较，完成传统的事后监督操作；配有印章验证系统后，自动将凭证图像中的印章与系统中预留的印鉴进行比较，完成印章的真伪识别。

1. **Json数据解析**

Json（JavaScript Object Notation,JS对象简谱）是一种轻量级的数据交换格式。它基于ECMAScript（欧洲计算机协会制定的JS规范）的一个子集，采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得Json成为理想的数据交换语言。易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。在AIUI平台交互中，均以Json数据进行，使用过程中，只需要做好Json数据解析即可。

1. **Android技术**

在本次机器人应用开发基础中，主要采用Android技术，涉及到技术内容有Android技术主要有界面布局设计、控件设计、事件响应、事件控件、Activity等，涉及到的内容均为Android基础知识。

以上技术均在后续实验部分以及编码过程中关联，因此各位同学在使用AIUI技能的同时，可根据各自兴趣进行针对性的学习。

本次机器人应用开发实验中，主要从如下几个方面进行：

（1）从AIUI开放平台使用开始，结合账号申请、能力包导入、二次开放流程展开；

（2）围绕开发过程所涉及到的技术，对Android开发基础进行介绍，并设计不同场景

下的软件页面；

1. 围绕AIUI平台能力（语言识别、合成、OCR、图像识别等）、机器人开发平台模块，结合场景设计小实验，并给出大实验案例，形成相应实验指导书给出实验目的与内容介绍、实验操作步骤以及注意事项。

|  |  |
| --- | --- |
| 智能机器人语音技术 | 服务机器人开发环境搭建 |
| 小实验1：语音识别技术体验 |
| 小实验2：语音合成技术体验 |
| 小实验3：语义理解技术体验 |
| 大实验：语音技术综合实验 |
| 智能机器人图像技术 | 小实验1：人脸识别技术体验 |
| 小实验2：文字识别技术体验 |
| 小实验3：证件识别技术体验 |
| 大实验：人证合一技术体验 |
| 智能机器人智能导航技术 | 小实验1：地图管理及配置 |
| 小实验2：移动技术体验 |
| 大实验：智能导航技术体验 |
| 机器人+开发套件实验课总结及讨论 | 小组成果演示及答辩 |

语音识别技术体验 - 实验指导书

# 一、实验目标

本次实验通过AIUI的配置及调用，开发一个关于语音识别的场景应用，让学生了解语音识别技术如何在机器人上使用，同时对语音识别技术的使用途径和步骤进行详细讲解，让学生理解起来更加轻松，本实验的具体实验目标如下：

1. 了解AIUI开放平台账号申请、应用创建及配置；
2. 掌握使用开发套件完成机器人智能应用的开发与调试；
3. 掌握语音识别能力接口的调用开发。

# 实验内容

语音交互技术作为人工智能领域的核心组成部分，也是智能机器人技术发展的重要要素。语音交互技术主要包含以下几个方面：

* 1. 语音合成技术，将文本内容合成为自然语音，并通过设备发声；
  2. 语音识别技术，将自然语音通过转写引擎识别为文本内容；
  3. 语义理解技术，对自然语言文本的理解，识别其具体表达的意思。

本次实验通过一个开发一个实际的简单智能应用，来对机器人使用的语音识别技术使用方法做详细的讲解，实验的具体流程如下：

## 2.1 实验环境

本实验的完成需要依赖硬件和软件的开发环境，硬件是需要一定配置的电脑，还需要语音交互套件；软件是需要Android开发集成环境，在本地电脑上完成Android集成开发环境的搭建，然后将测试程序运行到语音交互套件，能正常运行说明环境搭建完成，具体的实验环境如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **分类** | **环境1具体描述** | **环境2具体描述** |
| 1 | 硬件环境 | 机器人开发平台 | 本地电脑（Windows10）  CPU：酷睿i3 四代 2.0GHz以上  内存：4G以上 |
| 2 | 软件环境 | ADB | Android Studio软件 |
| 3 | 依赖模块/库 | 机器人基础能力包  commonAbility-release.aar | AIUI能力包  aiuilibrary-release.aar |
| 机器人日志辅助包  starrobot-logability.aar |  |

## 2.2 术语解释

本实验涉及到的专业术语包含：

1. 讯飞AIUI开放平台
2. ADB调试
3. 语音识别

# 实验步骤

## 步骤1：项目工程搭建及能力包导入

机器人使用的是Android平台，所有机器人的智能应用都是基于Android环境运行，机器人应用开发跟普通的Android应用开发有很大类似，在进行机器人智能应用开发之前，我们需要先在开发电脑上搭建安卓的开发环境，包括运行的JDK的安装及配置、开发工具的安装及配置等。

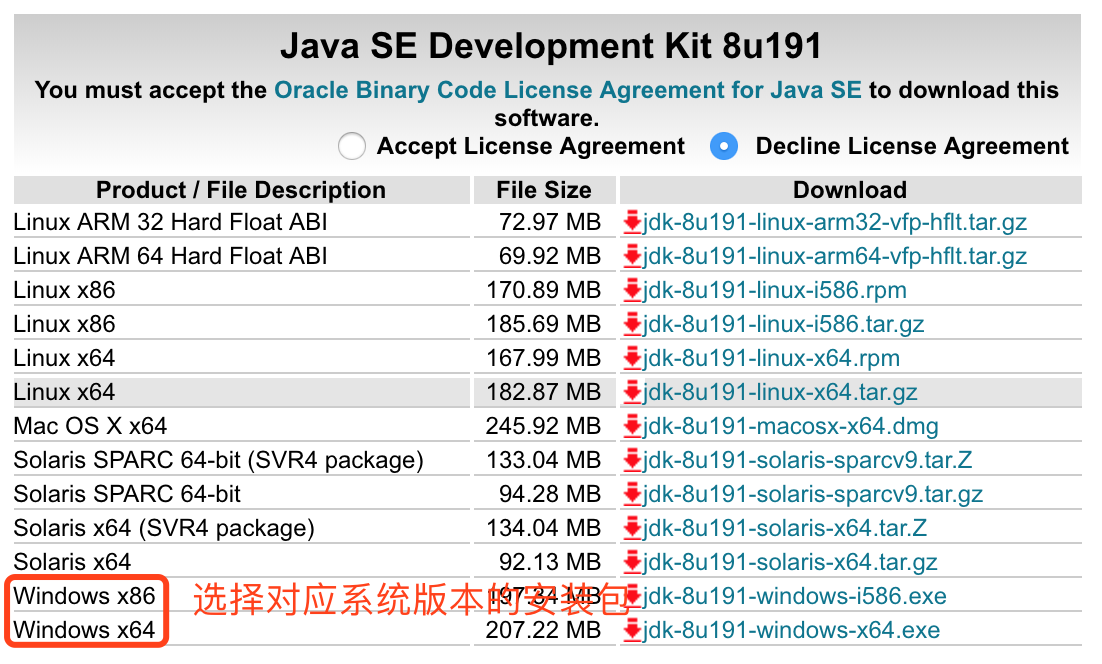
实验的具体流程如下：

1. 前期准备：一台windows系统的电脑，确保电脑联网，机器人开发平台通过USB与电脑连接；
2. 环境搭建：下载JDK和Android Studio安装包，并在电脑上安装；
3. 环境配置：配置JDK的环境变量，设置Android Studio软件的相关参数；
4. 调试连接：调试机器人开发平台与电脑的连接状态，确保Android Studio软件中能够连接上机器人开发平台设备；
5. 应用创建：创建测试应用，并进行相关程序开发；
6. 编译运行：将测试应用编译到机器人开发平台上运行，确保应用能够根据流程正常运行；
7. 实验结束：将应用开发程序保存到电脑合适位置，电脑与机器人开发平台断开USB连接，机器人开发平台装箱。

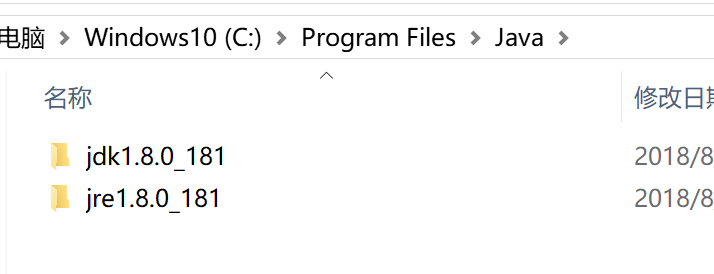
JDK(文件名：jdk-8u181-windows-x64.exe)

JDK的官方下载地址：

<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>，打开页面后会看到如下界面

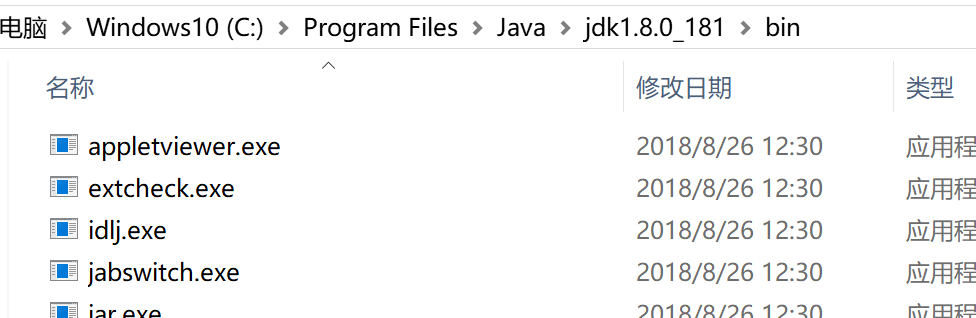


需要根据自己的电脑系统，选择相应的安装包版本，建议选择安装jdk-8u181-windows-x64.exe版本。下载之前需要先点击界面上的“Accept License Agreement”选项，JDK下载完成后双击进行安装。JDK的安装过程比较简单，基本上就是在安装界面点击“下一步”按钮即可，在安装的时候只需要注意将JDK和JRE安装到同一个目录，JDK默认安装成功后，会在系统目录下出现两个文件夹，一个代表JDK，一个代表JRE。



JDK的全称是Java SE Development Kit，也就是Java 开发工具箱。SE表示标准版。JDK是Java的核心，包含了Java的运行环境（Java Runtime Environment）、Java工具和给开发者开发应用程序时调用的Java类库。

我们可以打开JDK的安装目录下的Bin目录，里面有许多后缀名为exe的可执行程序，如下图所示：



这些都是JDK包含的工具，通过配置JDK的变量环境，我们可以方便地调用这些工具及它们的命令。

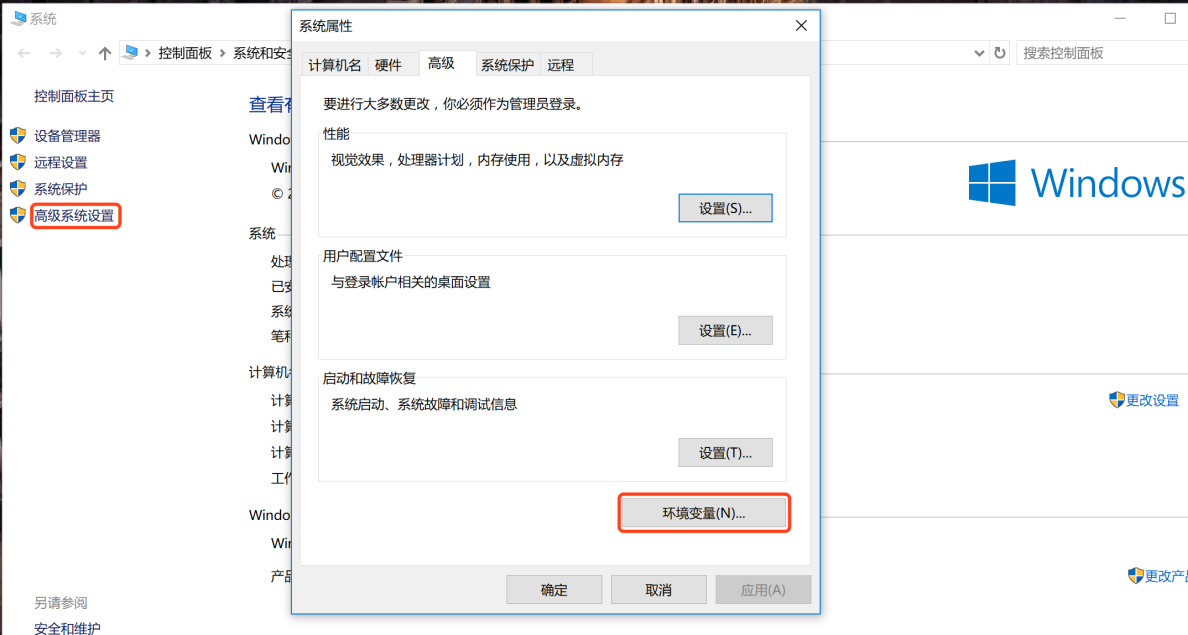
JDK包含的基本工具主要有：

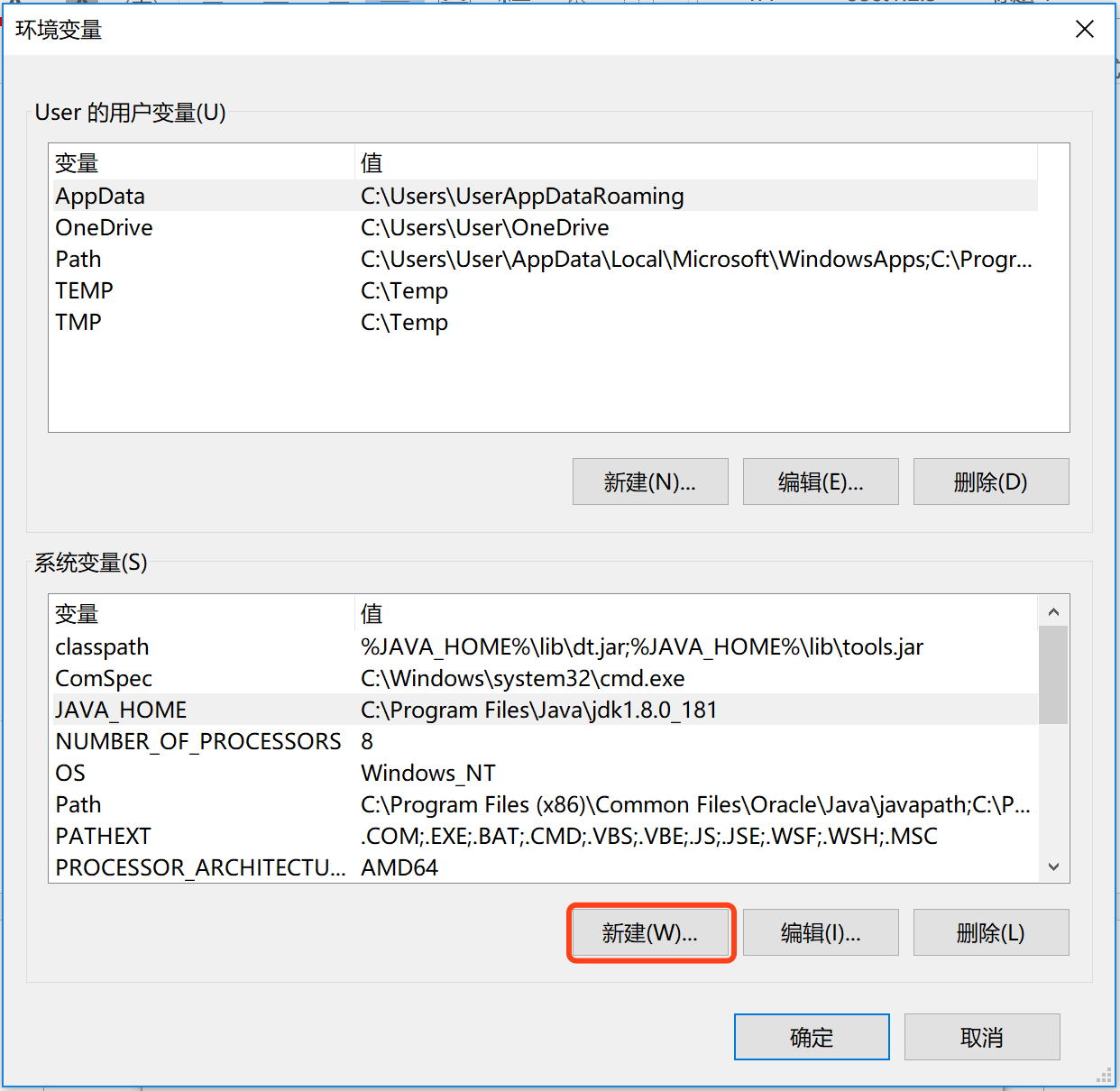
* javac：Java编译器，将源代码转成字节码。
* jar：打包工具，将相关的类文件打包成一个文件。
* javadoc：文档生成器，从源码注释中提取文档。
* java：运行编译后的java程序。

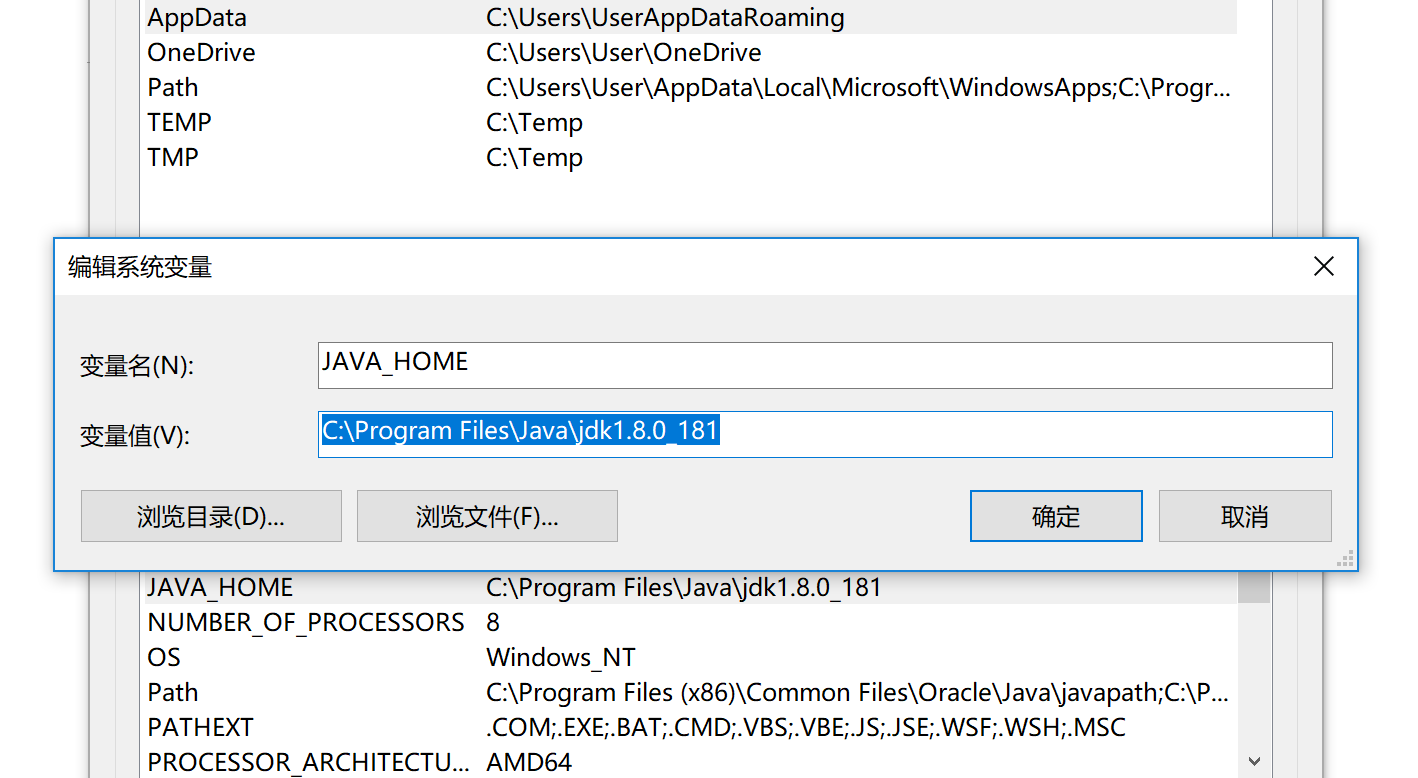
**JDK变量环境的配置**

由于在程序开发时需要使用的JDK命令不属于windows自己的命令，所以要想使用，就需要进行路径配置。

单击“计算机-属性-高级系统设置”，单击“环境变量”。在“系统变量”栏下单击“新建”，创建新的系统环境变量。







需要创建的系统环境变量及相关流程如下：

1. 新建->变量名“JAVA\_HOME”，变量值“C:\Program Files\Java\jdk1.8.0\_181”（即JDK的安装路径，需要根据自己安装的实际情况写）
2. 新建->变量名“Path”，变量值“;%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin”
3. 新建->变量名“CLASSPATH”,变量值“.;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar”

完成环境变量的配置后，需要进行确认，方法如下：

打开电脑的“命令提示符”窗口，输入java -version命令，如出现如下所示的JDK的编译器信息，说明JDK环境已经配置成功。（注意：这里必须输入java -version.exe文件实际所在的文件夹才会运行成功）

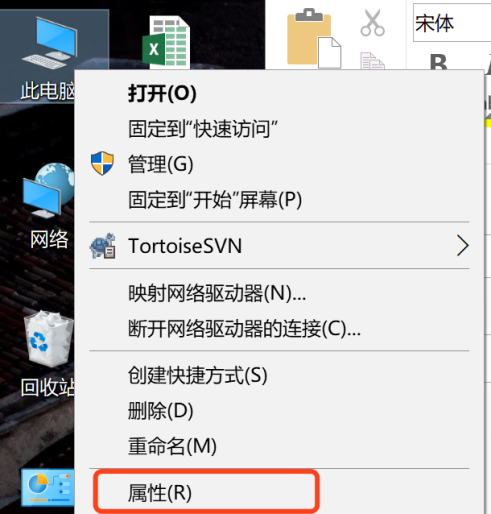
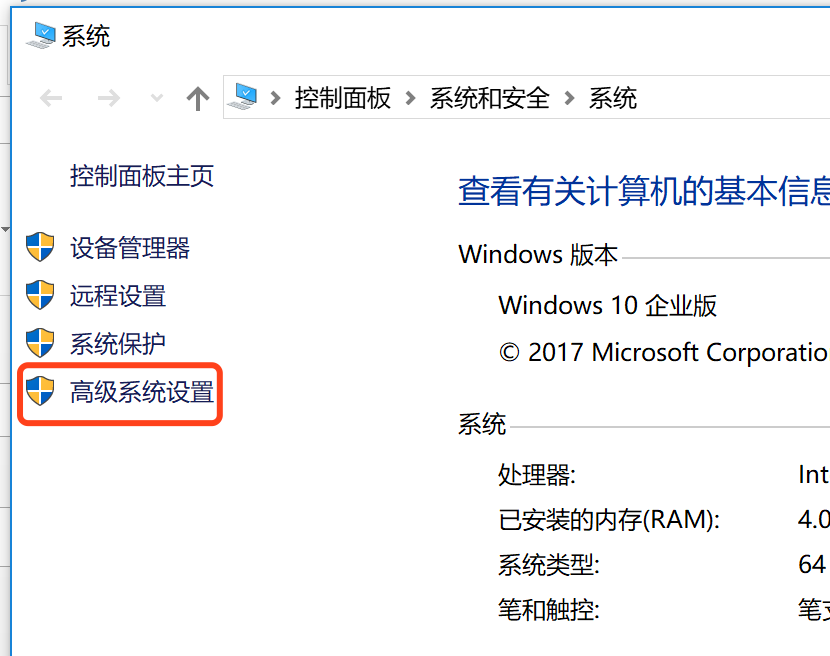


**ADB工具环境配置**

ADB下载地址：<http://adbshell.com/upload/>adb.zip

将ADB工具包（文件名：abd.zip）进行解压缩，记住解压缩的目录地址；

右键计算机，属性--高级系统设置--环境变量，打开环境变量配置窗口，如下：图：

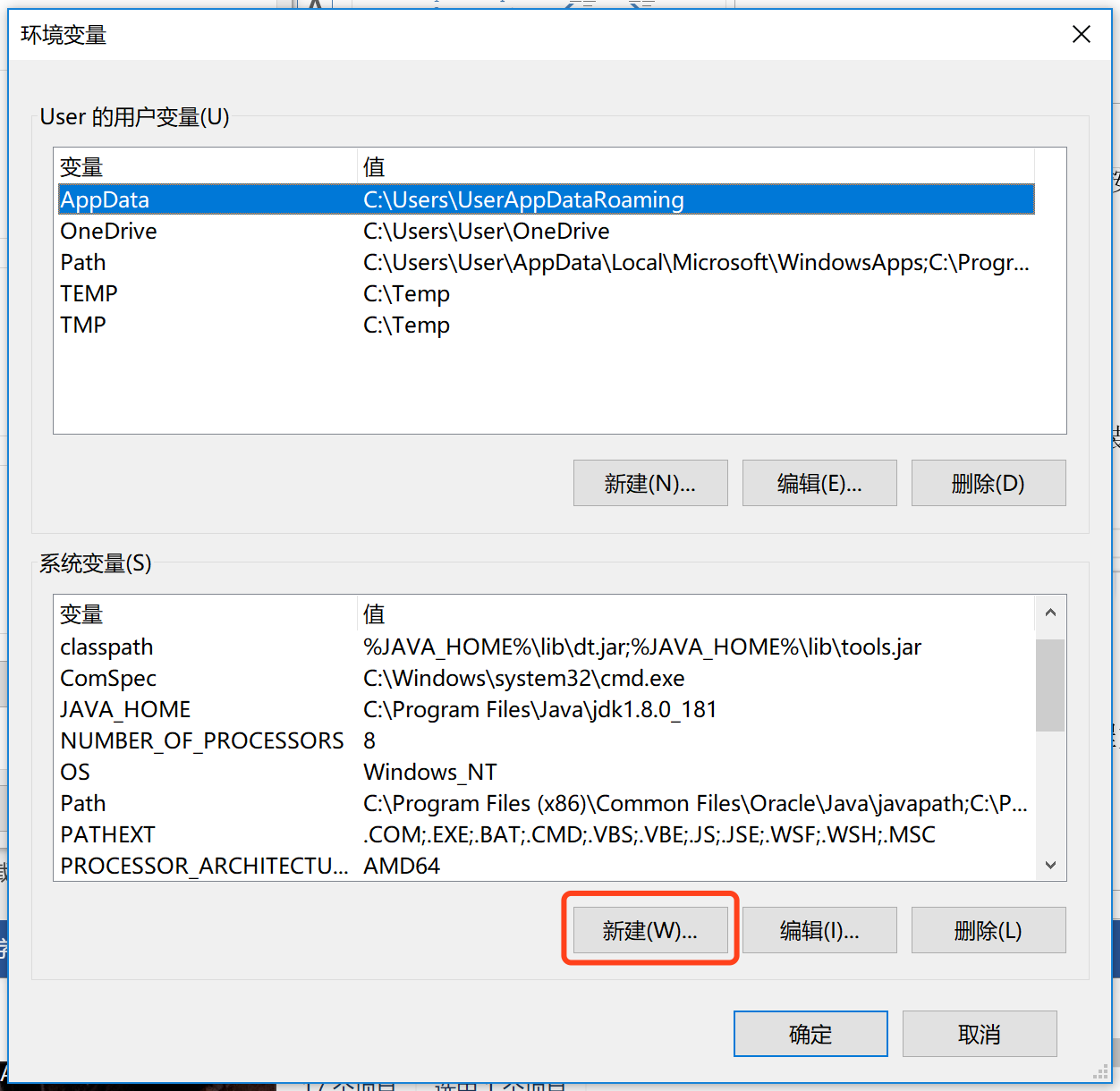
 

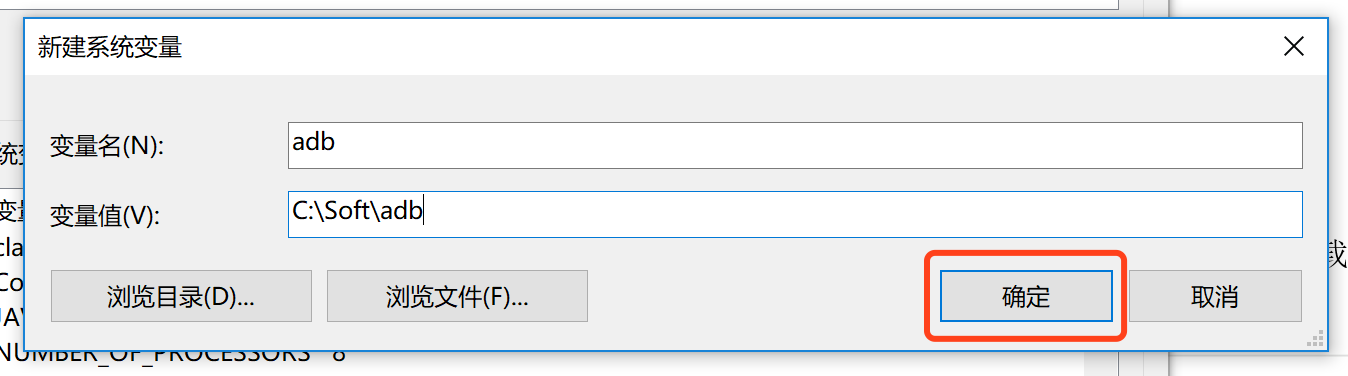


在打开的环境变量界面，点击下方的系统变量中新建按钮，在弹出的新建系统变量界面输入如下信息后点击确定按钮：

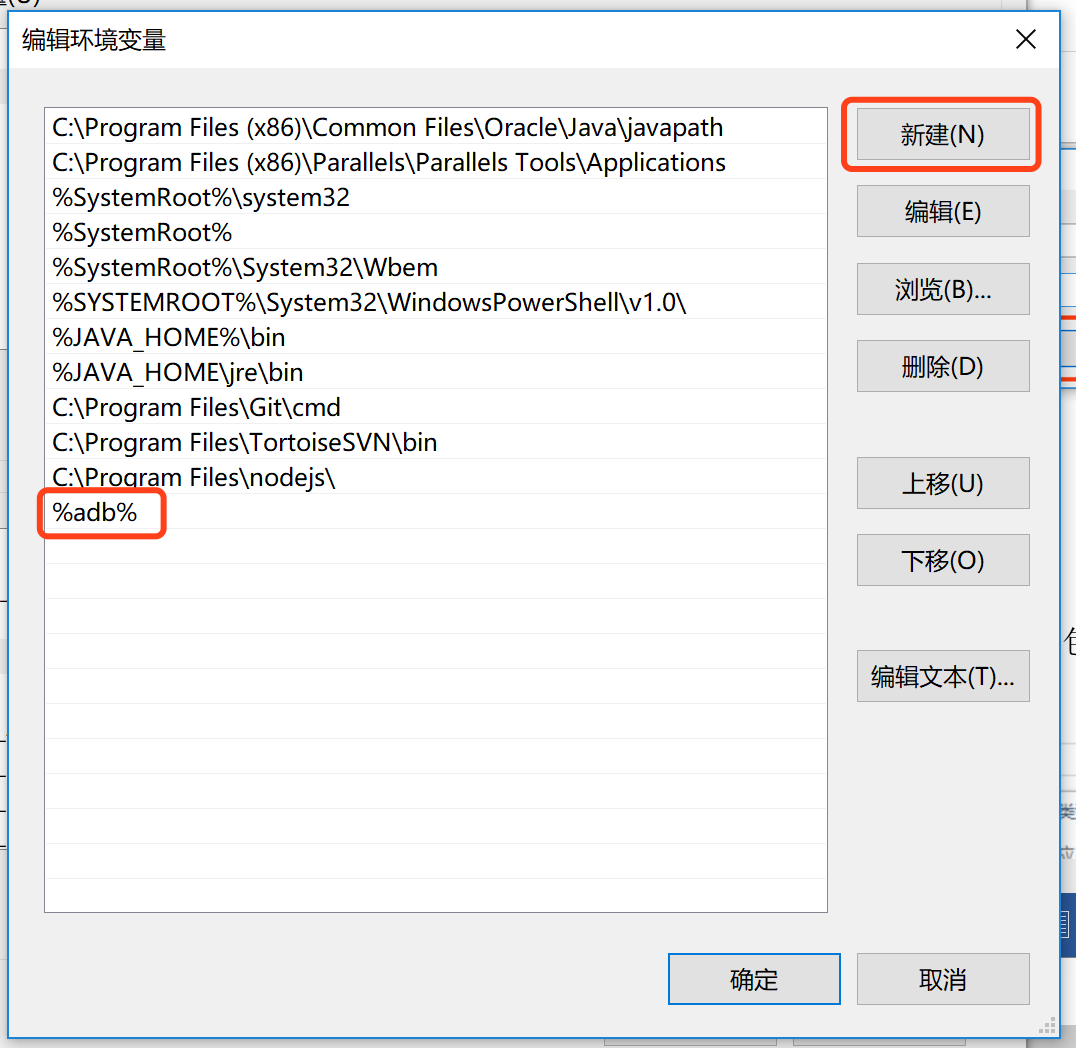
变量名：adb

变量值：adb工具解压保存目录（可以点击“浏览目录”按钮进行选择）

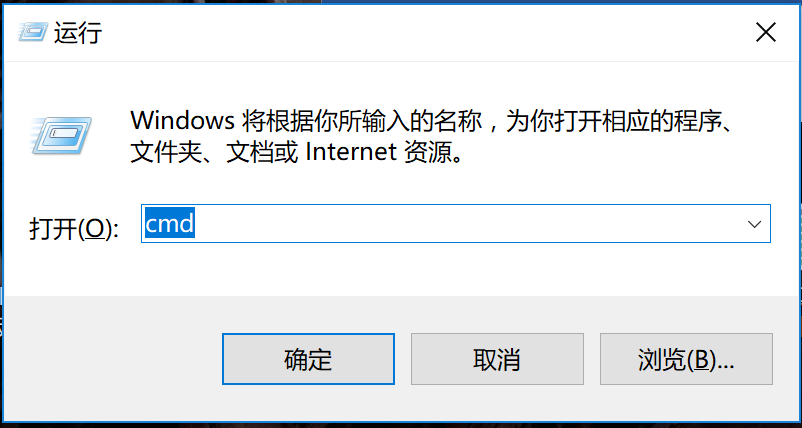




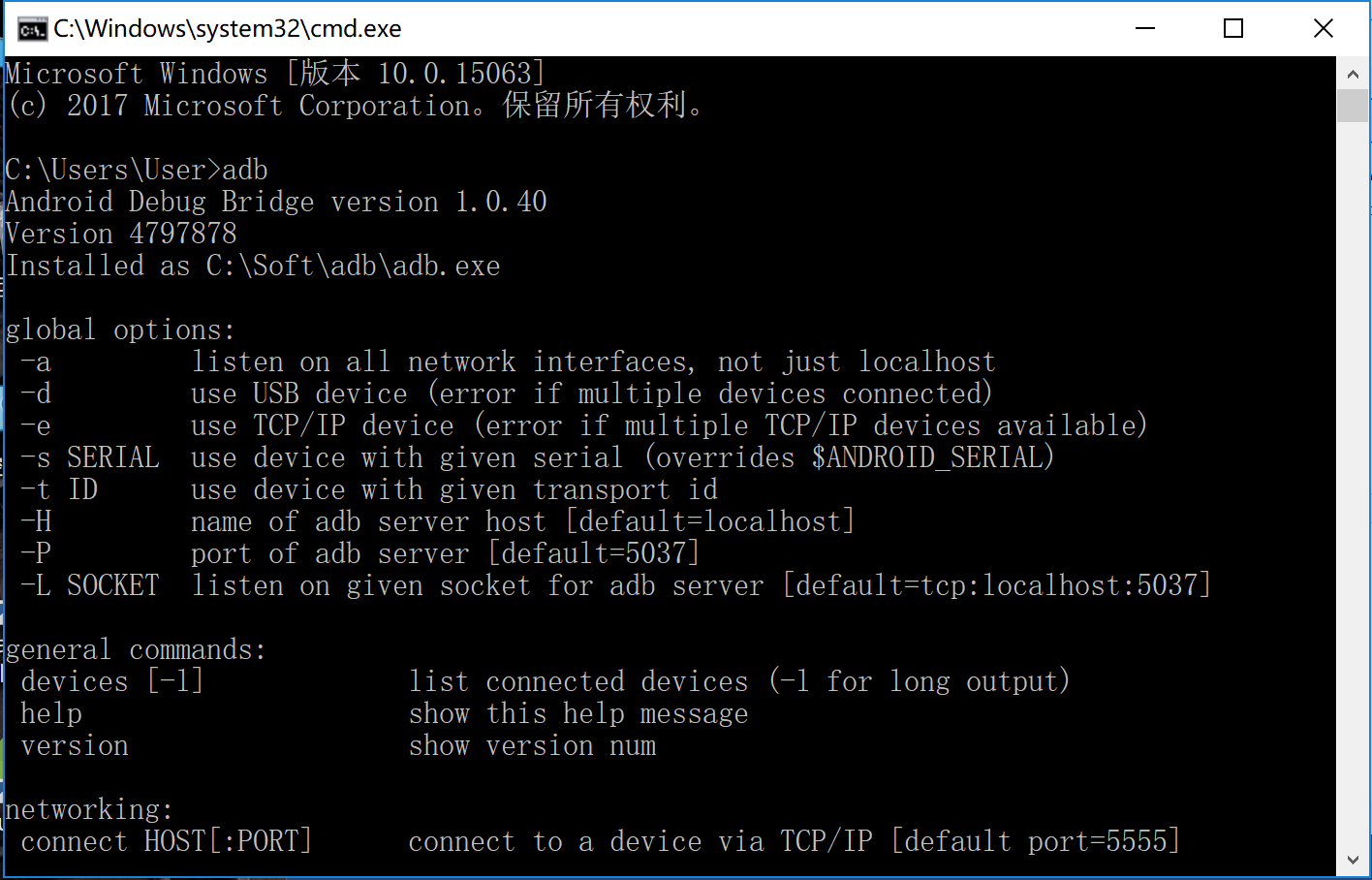
双击系统变量列表中的Path项，在弹出的编辑系统变量界面，点击右侧的“新建”按钮，然后输入%adb%后点击确定按钮；



到这里我们就完成了ADB工具的环境配置了，下面，我们要测试是否配置成功，按WIN+R键，进入命令行，输入cmd，回车，打开命令提示符窗口；



在打开的命令提示符窗口，输入adb，回车，如果界面显示如下，说明已经配置成功了。（注意：这里必须输入adb.exe文件实际所在的文件夹才会运行成功）



接下来我们使用USB线连接机器人开发平台与电脑，开发平台处接口为“主控1调试”，如下图所示：



完成连接后，我们通过ADB工具可以查看机器人开发平台与电脑的连接状态。

连接后开发平台若显示“允许USB调试吗？”，勾选“一律允许使用这台计算机进行调试”并确定。

首先，通过前文提供的ADB下载地址下载ADB工具包，解压到电脑上的桌面位置；

然后，按键盘上的windows键，在运行输入框位置输入cmd，回车打开“命令提示符”软件；

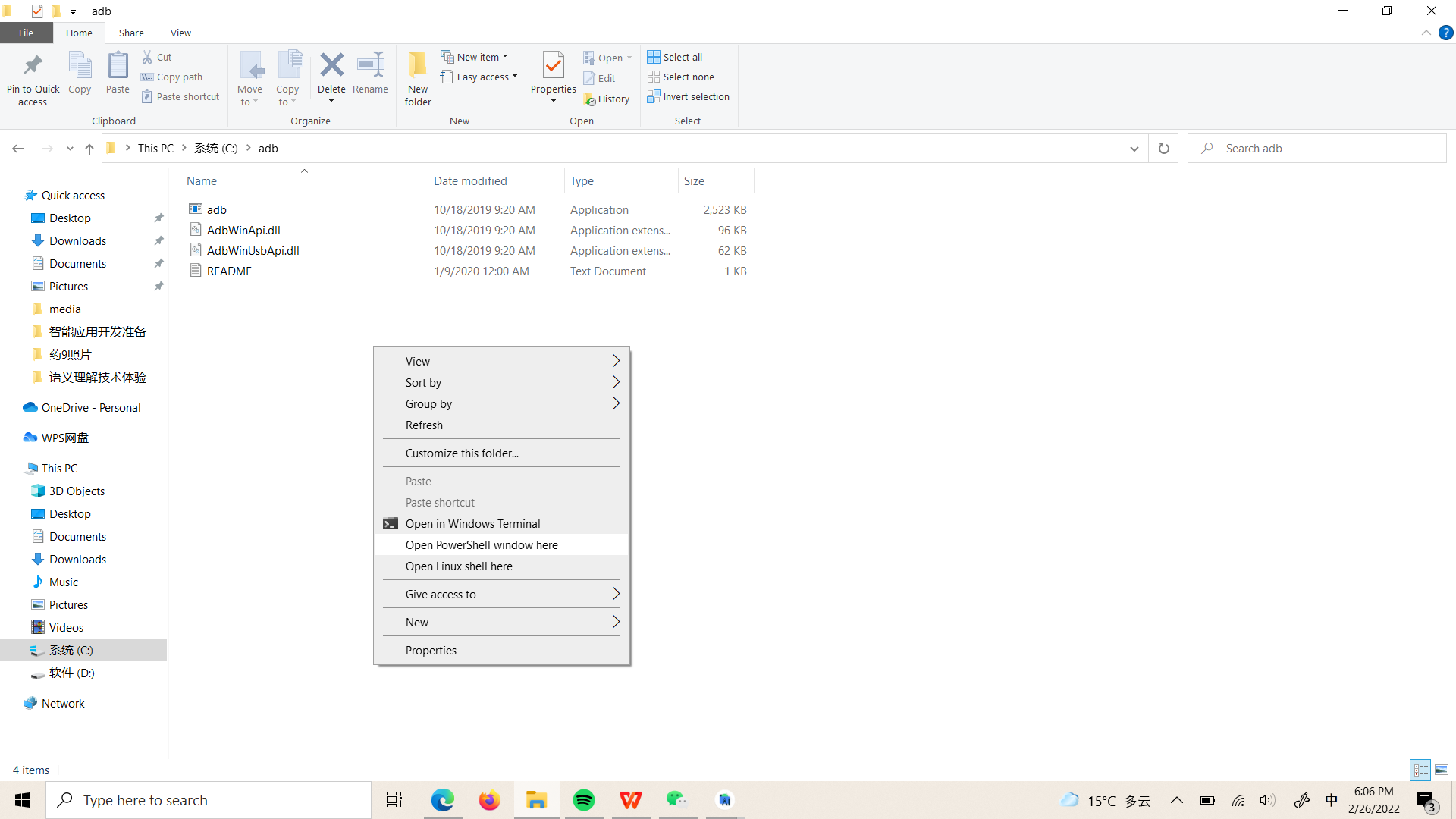
输入指令：cd desktop，回车（命令行此时进入了电脑的桌面目录）

输入指令：cd adb，回车（进入adb目录）

输入指令：adb devices，回车

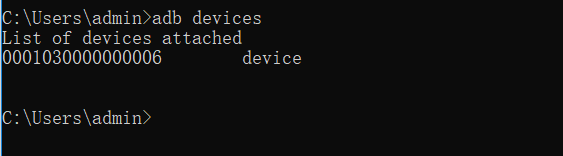
或：

手动进入之前安装的adb文件夹，在空白处按住Shift键的同时，点击鼠标右键，在弹出的菜单中选择打开Powershell（命令）窗口



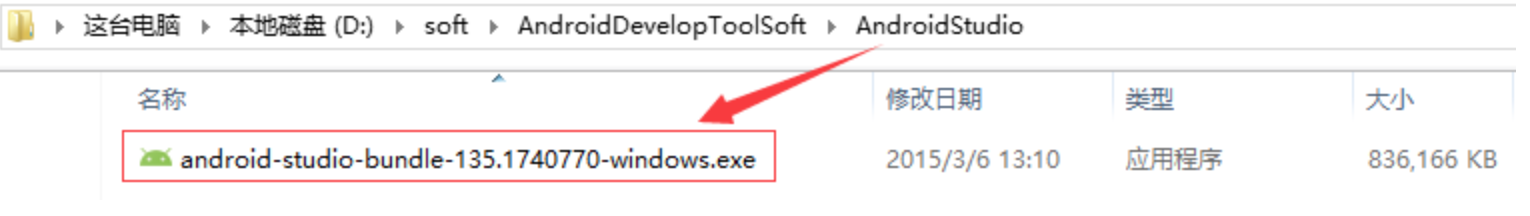
输入指令：adb devices，回车

当命令提示符软件界面显示如下图所示的设备信息时，说明机器人开发平台已经与电脑正确连接上。



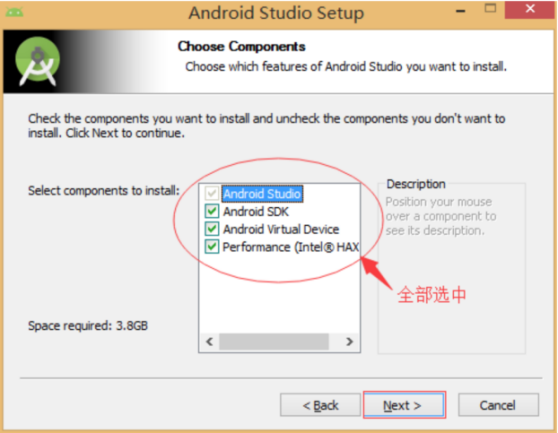
**Android Studio的安装（安装包：android-studio-ide-171.4443003-windows.exe）**

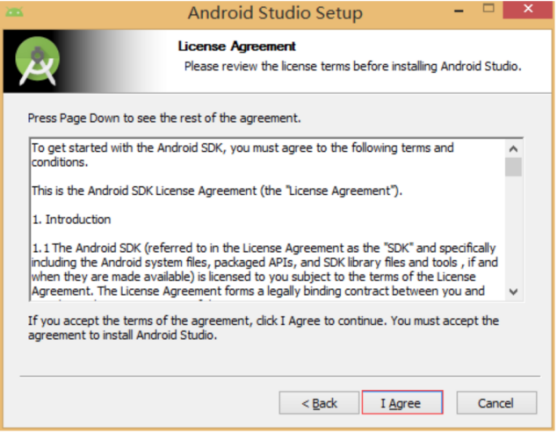
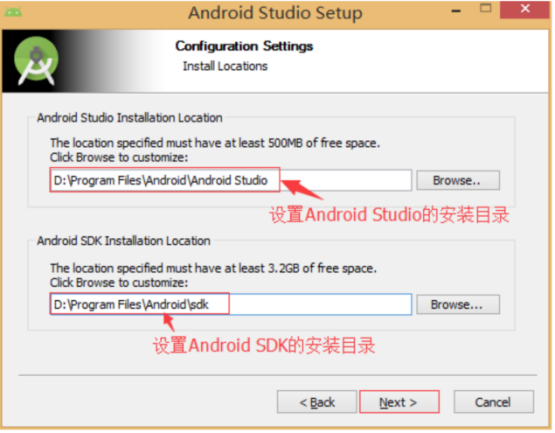
首先是从地址<https://developer.android.google.cn/studio/>下载Android Studio安装包，下载完成后得到一个如下图所示的安装包：

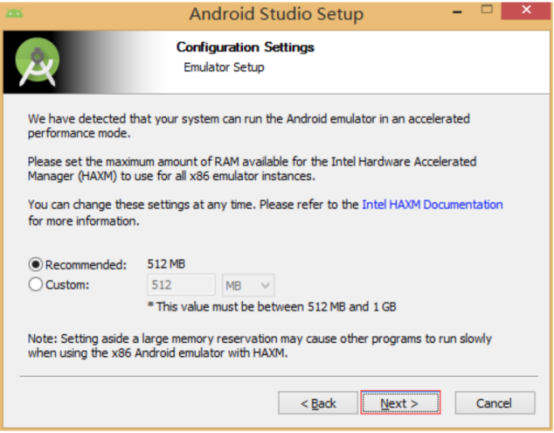
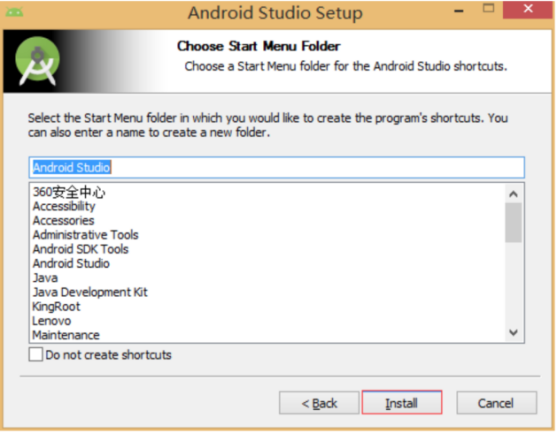


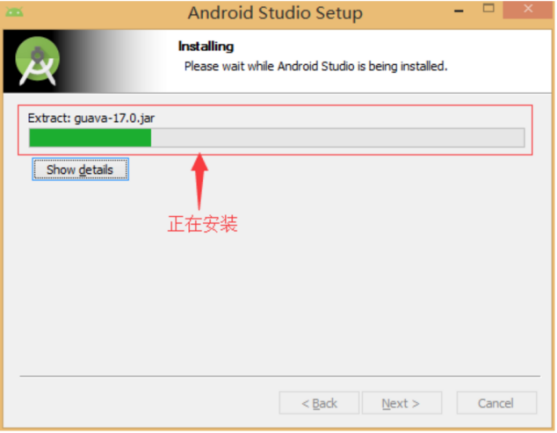
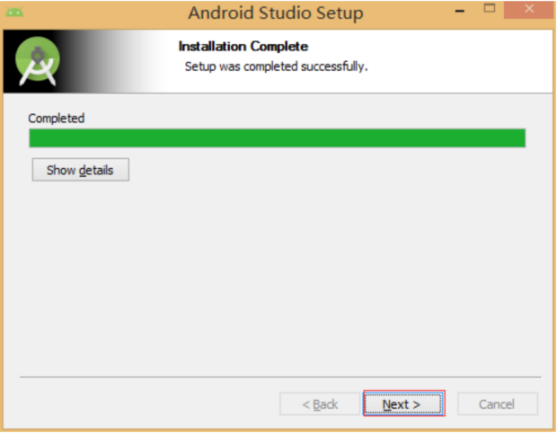
建议下载**安装包：android-studio-ide-171.4443003-windows.exe**

鼠标双击安装包启动安装程序，Android Studio安装过程如下图所示：

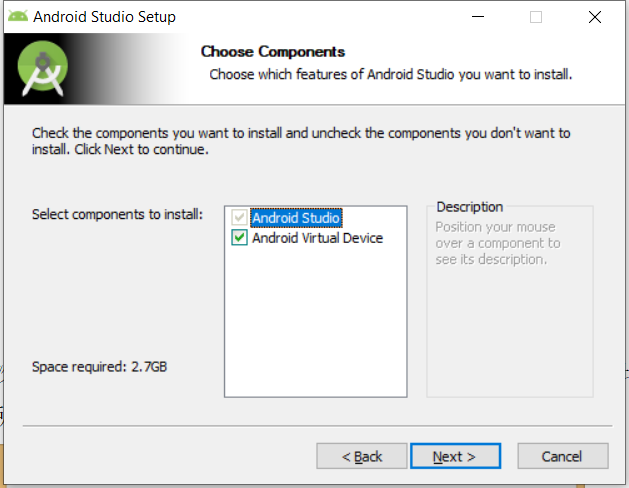
 

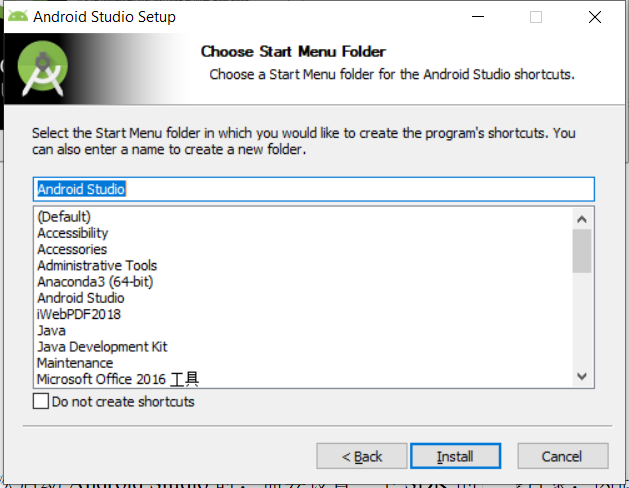
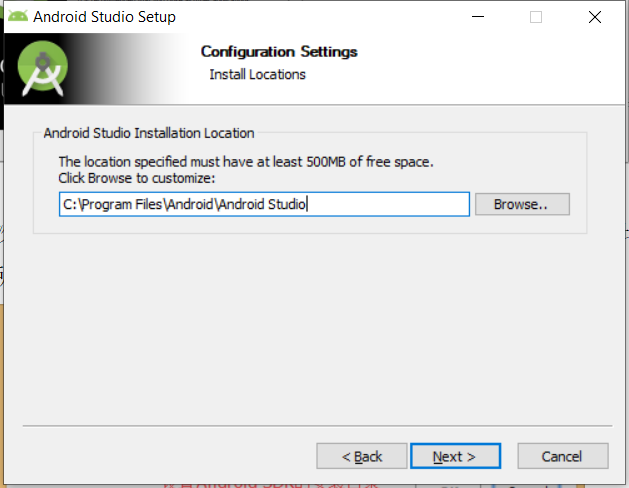
 

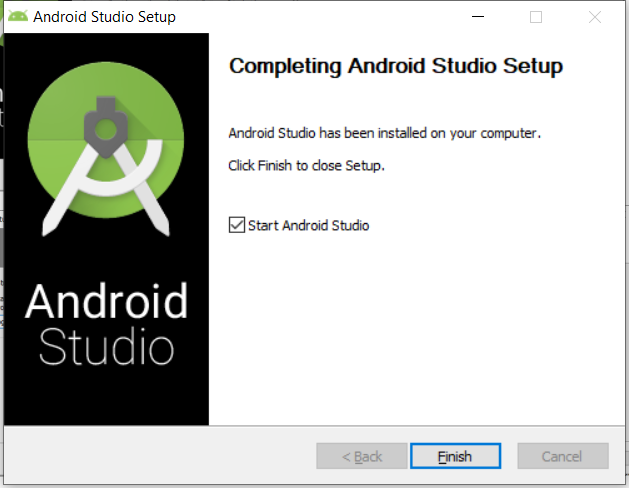
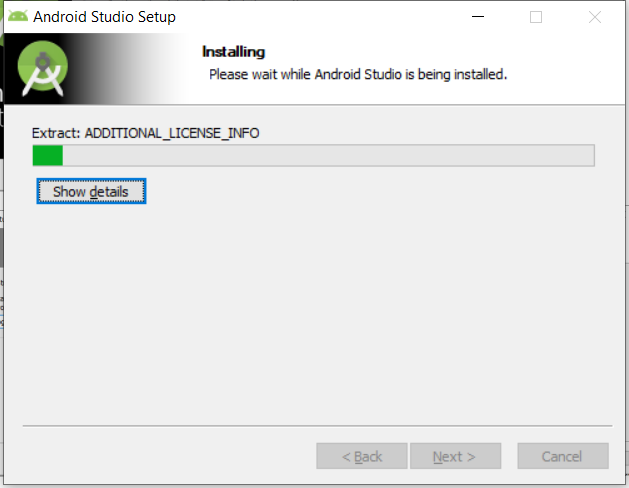


**运行Android Studio**

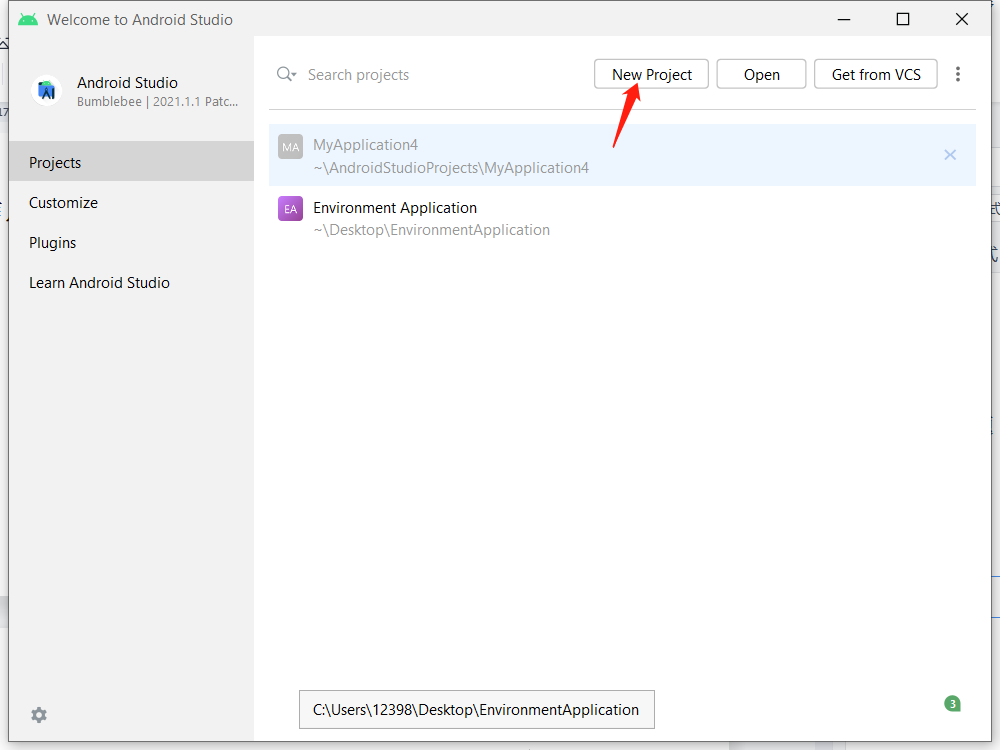
Android Studio启动过程如下图所示：

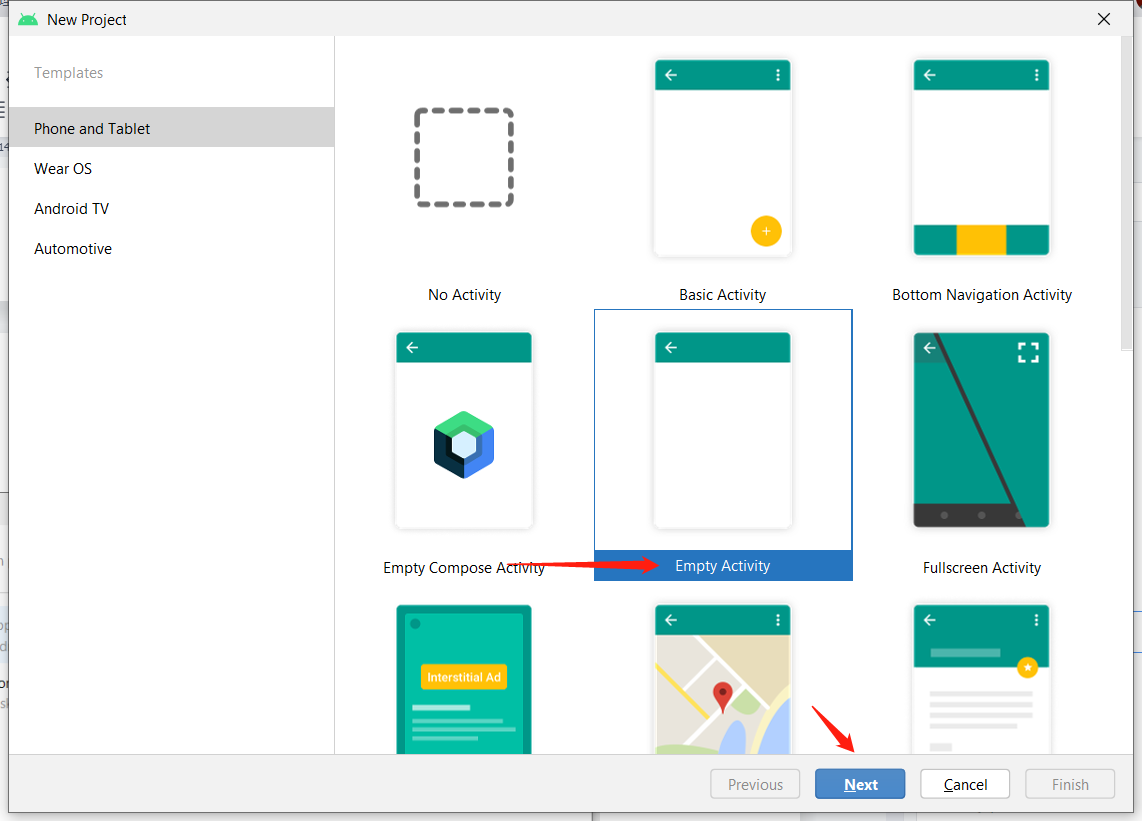




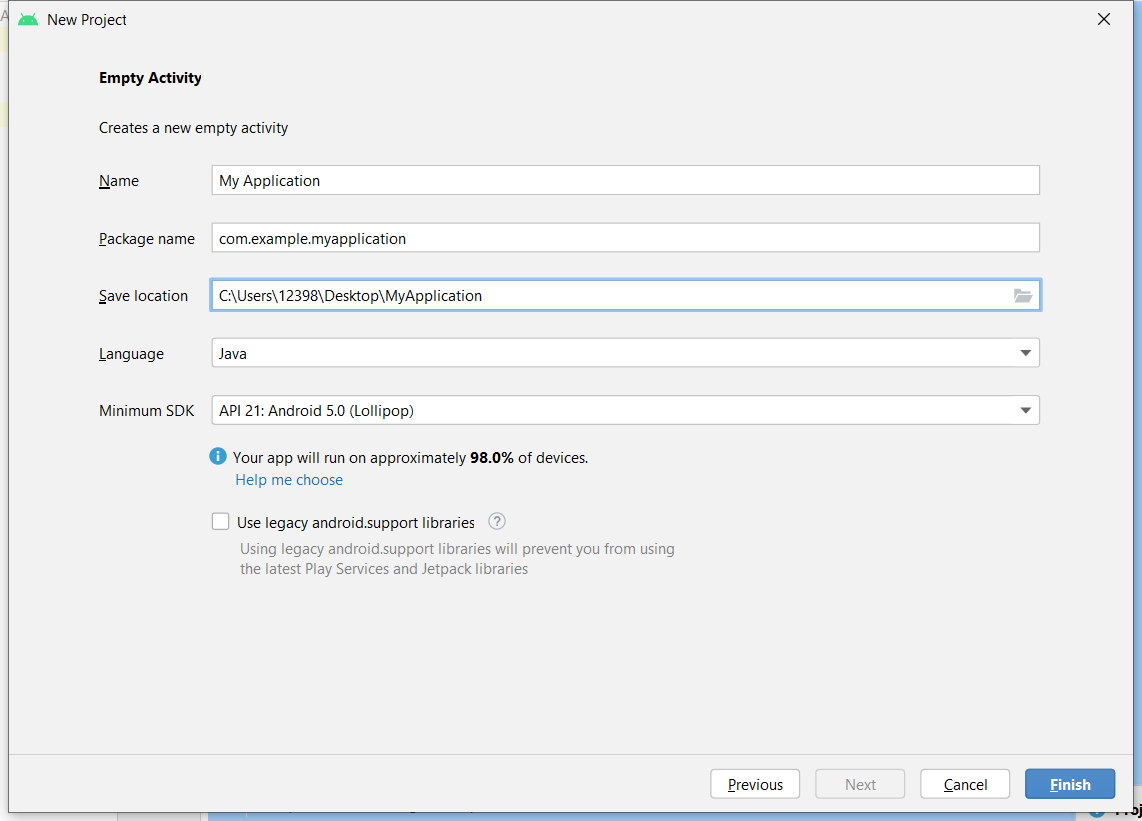


打开Android Studio之后，默认会按步骤帮我们创建一个app的项目；首先，我们选择创建新的项目，点击如下图所示按钮：

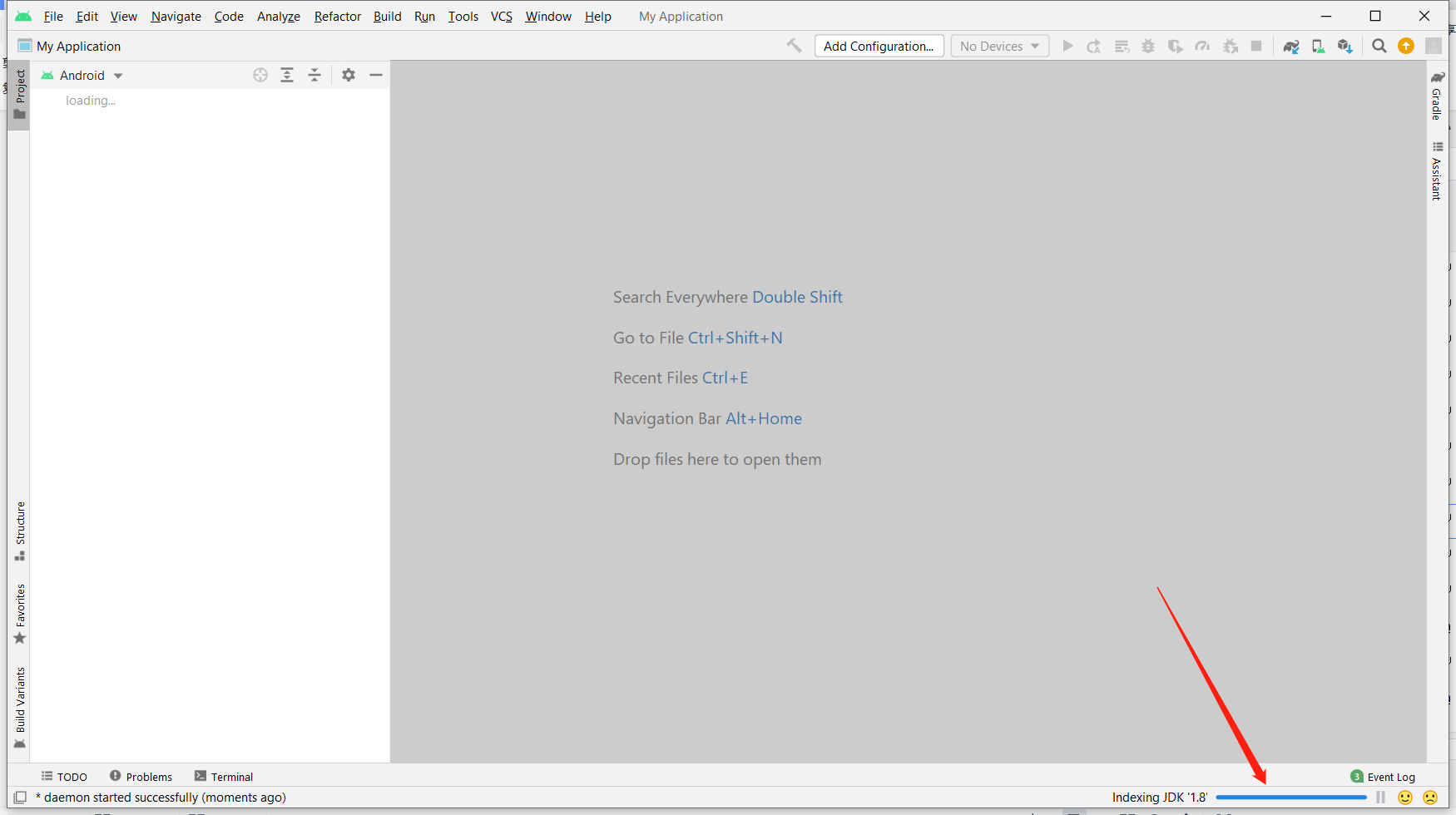




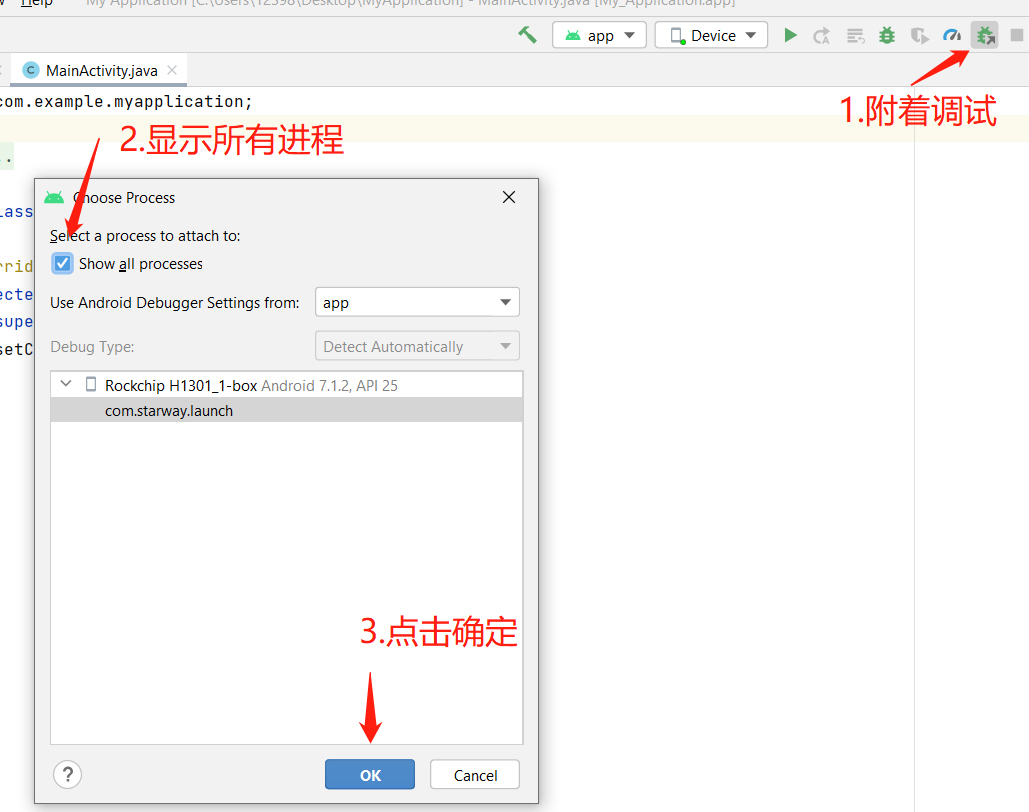
然后，输入示例应用的相关信息的界面，这里可以全部默认，直接点击下方的Finish按钮，如下图：



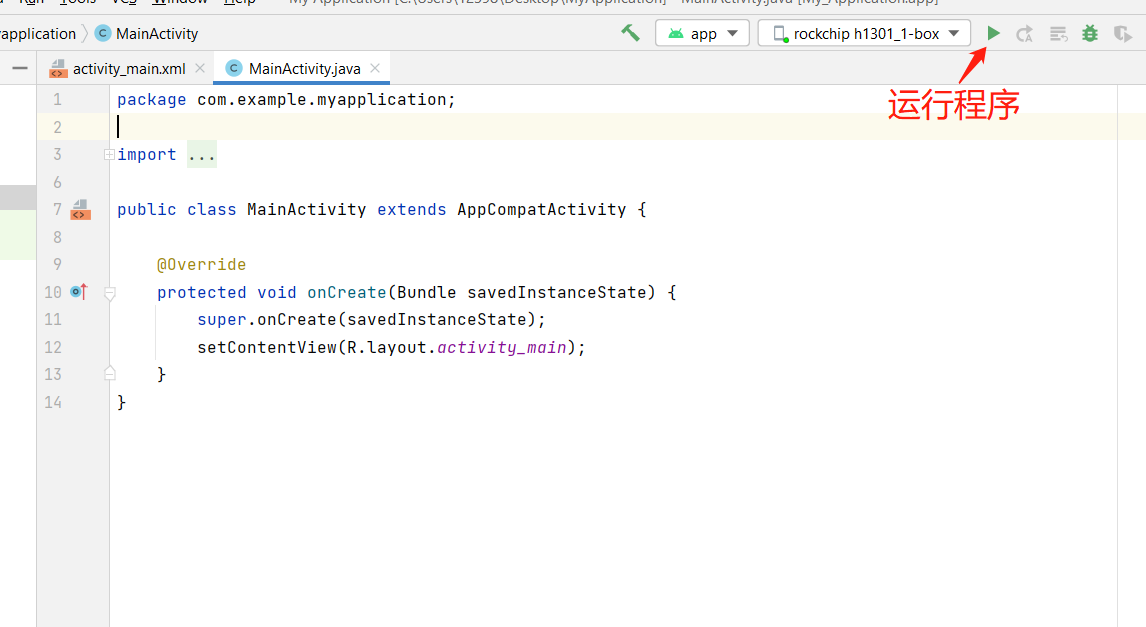
接着，进入项目页面，Andriod Studio会自动下载安装相关文件。等待最下方状态条提示的相关配置完成（等待时间较长），即可进行项目的开发。



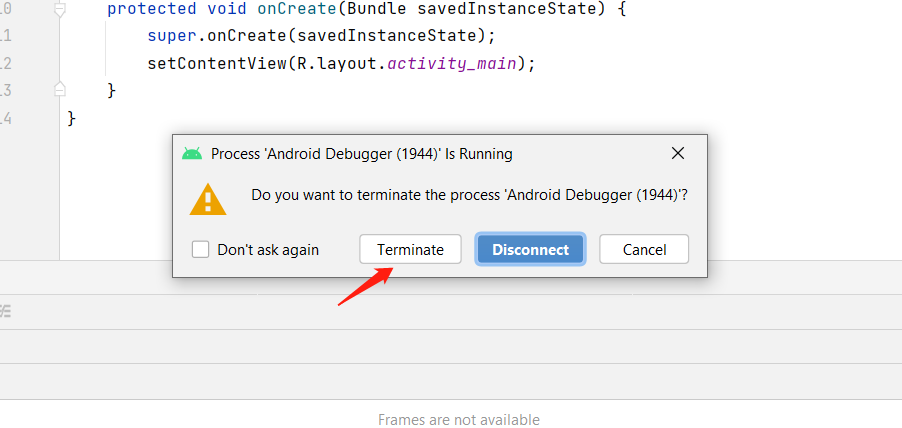
当ADB指令下能够显示机器人开发平台设备时，在Android Studio的界面上就可以查看设备的连接信息了，按下图步骤开始程序调试：



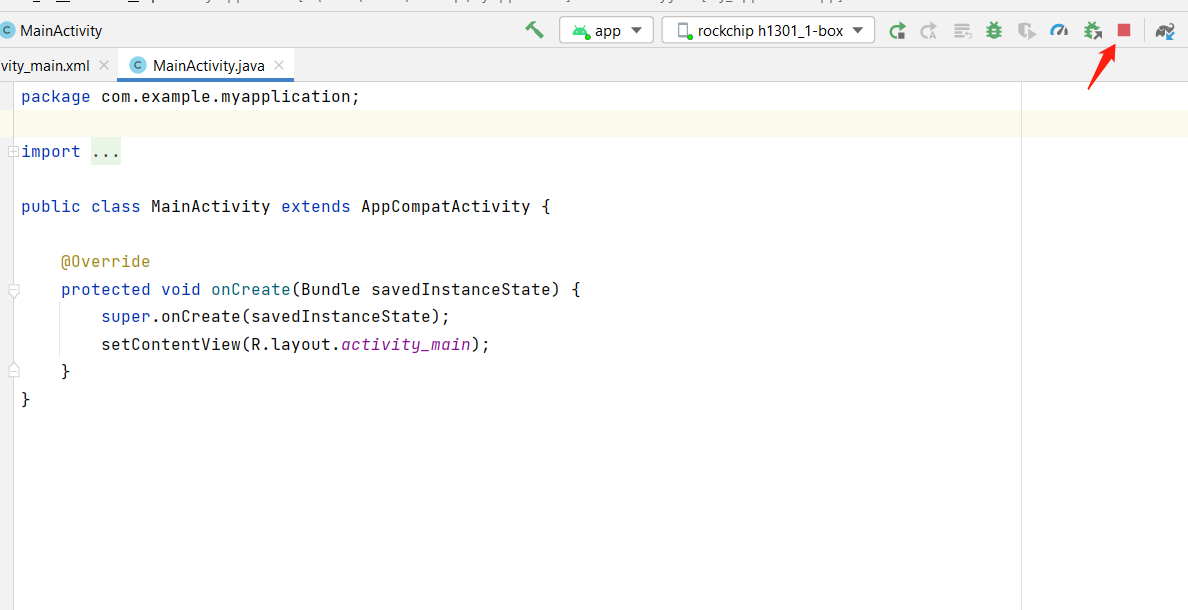
我们直接点击如下编译运行按钮



若出现如下提示，点击终止或断开连接皆可,不会影响程序运行



当应用能够正常完成编译后，机器人开发平台屏幕界面上会显示“Hello World!”的界面。点击终止按钮，结束程序运行。



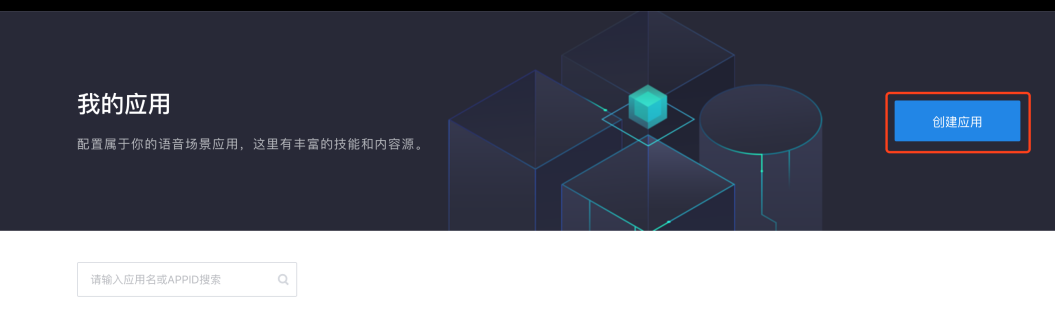
到这里，我们就完成了智能机器人开发前的准备工作，后面就可以进入我们的实际应用开发环节了。

## 步骤2：AIUI账号申请及平台设置

由于本次实验开发语音合成功能应用使用的是科大讯飞的AIUI语音合成功能，我们需要先到讯飞的AIUI平台申请账号，并完成平台上相关的功能设置，具体步骤如下：

* 1. 使用浏览器访问AIUI开放平台，地址：<https://aiui.iflyos.cn/index-aiui>；
  2. 点击右上角“注册”按钮，进入注册页面，根据要求完成注册信息的填写，注册账号；
  3. 使用注册的账号登录AIUI开放平台，点击页面上方“我的应用”菜单，进入应用管理页面，在应用管理页面点击“创建”应用按钮，开始添加应用，如下图：





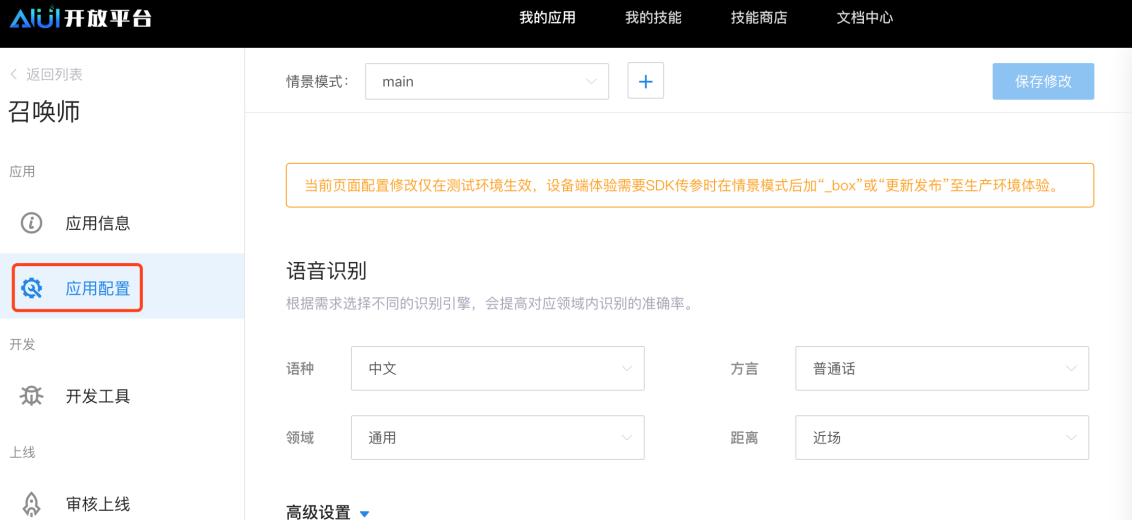
* 1. 在应用添加界面，输入应用的相关信息（输入应用名称，应用分类选择“服务机器人”）后，点击“确认创建”按钮完成应用的添加，如下图：



* 1. 完成了应用的创建后，就进入了应用的信息显示界面，在这里，我们可以看到应用中需要使用的appid和appkey；



* 1. 点击页面右侧的“应用配置”菜单，进入应用配置页面，在这里，可以对应用的语音识别参数进行设置。



可根据产品特点及主要客户群体按需选择语种、方言、领域和距离，智能机器人及开发板套件采用了6麦麦克风环形阵列，为远程拾音模式，可选择远场拾音模式。

在“高级设置”中，可以对语音识别展示效果的选项，可以为“识别结果添加标点”、“识别结果优先阿拉伯数字”和“progressive流式识别”。

另外，为了提高某些专有名词的识别率，可以在平台上传热词文件。

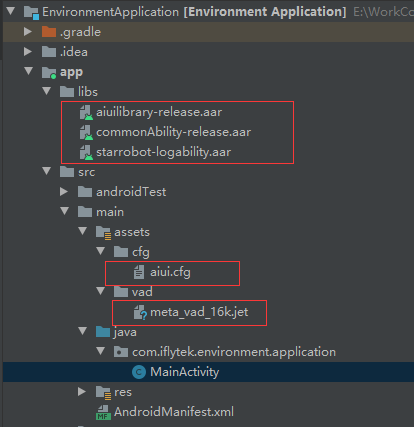
注：针对免费申请的账号，AIUI开放平台对应用做了每天500次交互的使用限制，如果需要对交互次数进行提升，请联系客服人员处理。

**机器人能力包导入**

完成了项目工程的创建后，我们现在开始导入机器人开发的能力包，首先，我们导入应用必需的机器人基础能力开发包和需要依赖的日志能力包，以及AIUI能力开发包（注：实验中所有使用到的能力包都在“机器人应用开发能力包资源”目录中），主要流程如下：

1. 将机器人基础能力开发包commonAbility-release.aar和starrobot-logability.aar以及AIUI能力包aiuilibrary-release.aar拷贝到应用的/app/libs文件夹中；
2. 在/app/src/main目录下创建assets文件夹，再在assets目录下创建cfg和vad两个目录；
3. 将AIUI的配置文件aiui.cfg拷贝到cfg目录下，将AIUI的vad文件meta\_vad\_16k.jet拷贝到vad目录下；

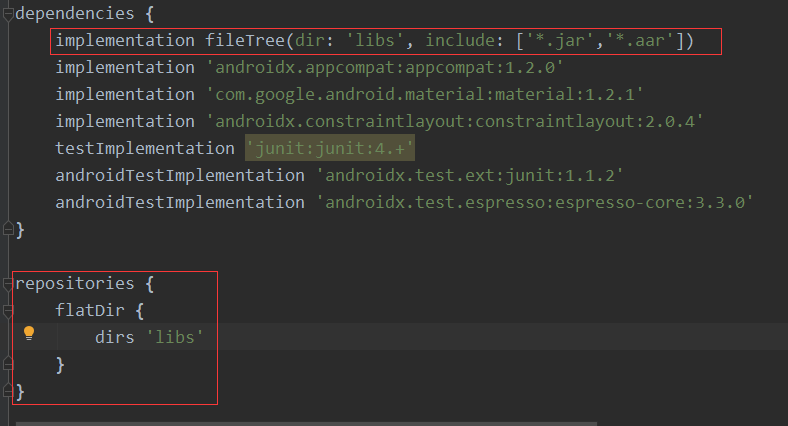
完成以上三个步骤后，项目整体目录结构如下图所示：



**能力包导入的配置**

完成了应用的能力包导入后，我们需要对工程的能力包依赖关系及权限配置等信息进行配置，主要流程如下：

1. 在build.gradle(app)文件中添加能力包的依赖，如下图所示：



添加的代码如下：

|  |
| --- |
| *//代码仓库地址*  repositories{  flatDir{  dirs **'**libs**'**  }  }  *//编译依赖lib下的jar和aar*  implementation fileTree(dir: 'libs', include: ['\*.jar','\*.aar']) |

修改对应SDK版本，如下图所示



1. 在AndroidManifest.xml文件中添加应用权限设置，如下所示：



具体要添加的配置代码如下，只需要将下面黄色背景的代码拷贝到自己创建的工程文件的AndroidManifest.xml对应的地方即可。

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.starway.myapplication">   *<!--六麦唤醒-->* <uses-permission android:name="com.android.permission.IFLYTEK\_MIC\_WAKEUP" />  *<!-- 读取内存卡权限 -->* <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"/>  *<!-- 连接网络权限 -->* <uses-permission android:name="android.permission.INTERNET"/>  *<!-- 允许程序录制音频 -->* <uses-permission android:name="android.permission.RECORD\_AUDIO"/>  *<!-- 允许程序访问有关的网络信息 -->* <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>  *<!-- 允许程序改变网络连接状态 -->* <uses-permission android:name="android.permission.CHANGE\_NETWORK\_STATE"/>  *<!-- 允许程序访问Wi-Fi网络状态信息 -->* <uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"/>  <uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE"/>  *<!-- 允许应用写（非读）用户的外部存储器 -->* <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>  *<!-- 允许程序读取所有者数据 -->* <uses-permission android:name="android.permission.READ\_OWNER\_DATA"/>  *<!-- 允许一个程序写入但不读取所有者数据 -->* <uses-permission android:name="android.permission.WRITE\_OWNER\_DATA"/>   <application  android:name=".DemoApplication"  android:allowBackup="true"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  android:roundIcon="@mipmap/ic\_launcher\_round"  android:supportsRtl="true"  android:theme="@style/AppTheme">  <activity android:name=".MainActivity"  android:launchMode="singleTask">  <intent-filter>  <action android:name="android.intent.action.MAIN"/>  <category android:name="android.intent.category.HOME" />  <category android:name="android.intent.category.DEFAULT"/>  <category android:name="android.intent.category.LAUNCHER"/>  </intent-filter>  </activity>  </application>  </manifest> |
|  |

**功能集成示例**

在项目工程的MainActivity类中添加机器人的能力初始化方法，并在场景初始化的onCreate方法中添加调用，如下所示：

|  |
| --- |
| package com.example.myapplication;//**该行不复制，使用自己的项目名称**  import androidx.appcompat.app.AppCompatActivity;  import android.os.Bundle;  import android.os.Handler;  import com.starway.starrobot.commonability.RobotType;  import com.starway.starrobot.commonability.StarCommonAbility;  import com.starway.starrobot.commonability.hardware.EmojiHelper;  import com.starway.starrobot.logability.StarLogAbility;  public class MainActivity extends AppCompatActivity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);  initRobotAbility();  }  /\*\*  \* 初始化机器人基础能力  \*/  private void initRobotAbility() {  //日志初始化  StarLogAbility.getInstance().initAbility(this);  StarCommonAbility.getInstance().initAbility(this.getApplicationContext(), RobotType.TYPE\_TEACHING, new StarCommonAbility.onResultCallback() {  @Override  public void onResult(boolean isSuccess, String hard\_code) {  if (isSuccess) {  //硬件和业务状态初始化  //这里根据不同的标识返回不同硬件的初始化成功状态  switch (hard\_code) {  case "emoji": //表情硬件初始化成功  //设置基础表情  EmojiHelper.doEmojiBase();  //延迟展示Love表情  showEmojiEffectDelay();  break;  default:  break;  }  }  }  });  }  */\*\*  \* 延迟展示机器人表情（Love)  \*/* private void showEmojiEffectDelay()  {  *//延迟3秒展示Love表情* new Handler().postDelayed(new Runnable() {  @Override  public void run() {  EmojiHelper.*doEmojiLove*();  }  },3000);  } } |

注：能力初始化方法中的RobotType.TYPE\_TEACHING参数表示的是当前运行的平台，包括以下几种：

* RobotType.TYPE\_A A款机器人
* RobotType.TYPE\_B B款机器人
* RobotType.TYPE\_TEACHING 机器人开发平台

**配置文件修改**

由于Android Studio版本的区别，为了保证程序正常编译运行，需要对相关配置文件进行修改。将以下代码框内的代码复制替换至对应文件内即可

**build.gradle(Project)**

|  |
| --- |
| buildscript {  repositories {  google()  jcenter()  }  dependencies {  classpath "com.android.tools.build:gradle:4.1.1"  // NOTE: Do not place your application dependencies here; they belong  // in the individual module build.gradle files  }  }  allprojects {  repositories {  google()  jcenter()  }  }  task clean(type: Delete) {  delete rootProject.buildDir  } |

**settings.gradle**

|  |
| --- |
| rootProject.name = "My Application"//注意和自己的项目名保持一致  include ':app' |

**代码运行**

在示例代码中，我们首先添加了一个初始化机器人能力的方法initRobotAbility，在这个方法里面，我们对机器人的基础硬件能力进行了初始化，并且在初始化成功状态返回时（表情硬件初始化成功），执行了延迟显示Love表情的方法，然后在场景的onCreate方法中对这个机器人能力初始化方法进行调用。

完成了这些基本代码的集成，我们将项目工程编译到机器人开发平台上运行，第一次运行时，如果进行录音等操作，可能会出现异常崩溃的情况，这是因为我们还没有给应用赋予相应的权限，机器人的Android系统为7.1.2，因为Android系统版本在6.0以上的时候，需要动态的申请权限，如果没有动态申请就需要手动到应用设置中赋予权限，如果我们需要手动进行设置，具体步骤如下：

1）首先，我们打开系统的命令提示符窗口（WIN+R组合键打开运行窗口，输入cmd后回车）；

2） 在命令提示符窗口使用ADB工具打开机器人开发平台的系统设置界面，输入如下指令后回车：

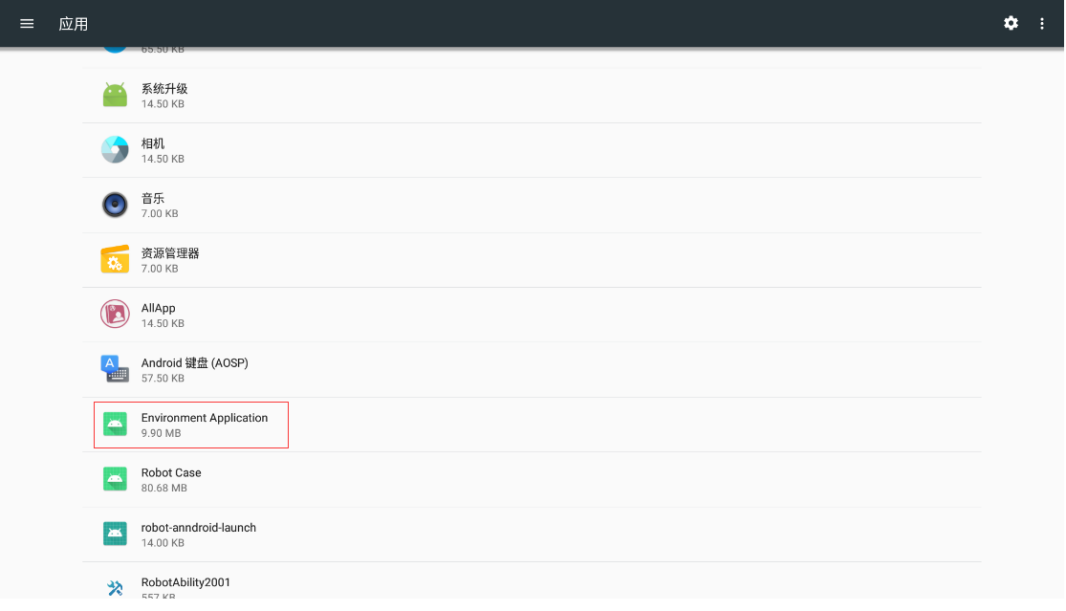
adb shell am start –n com.android.settings/com.android.settings.Settings

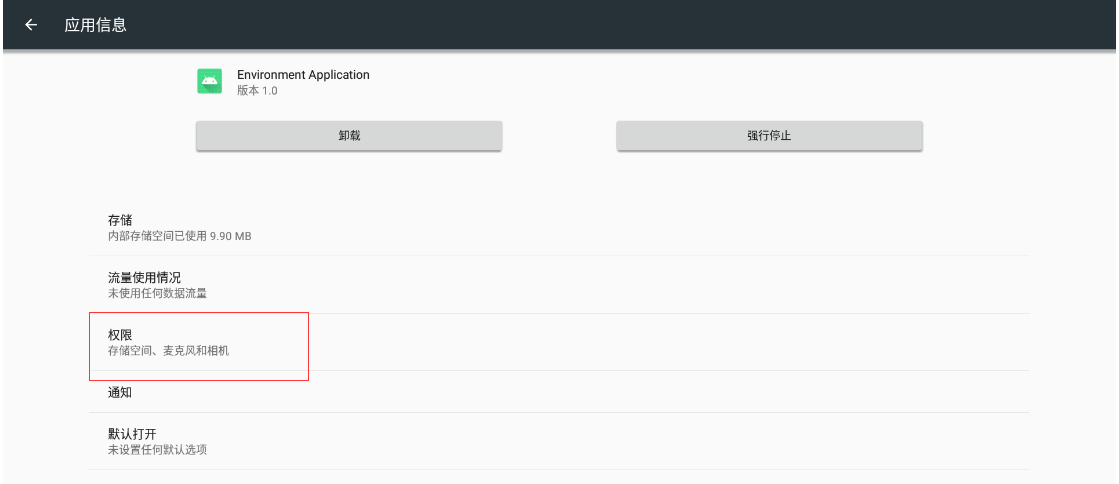


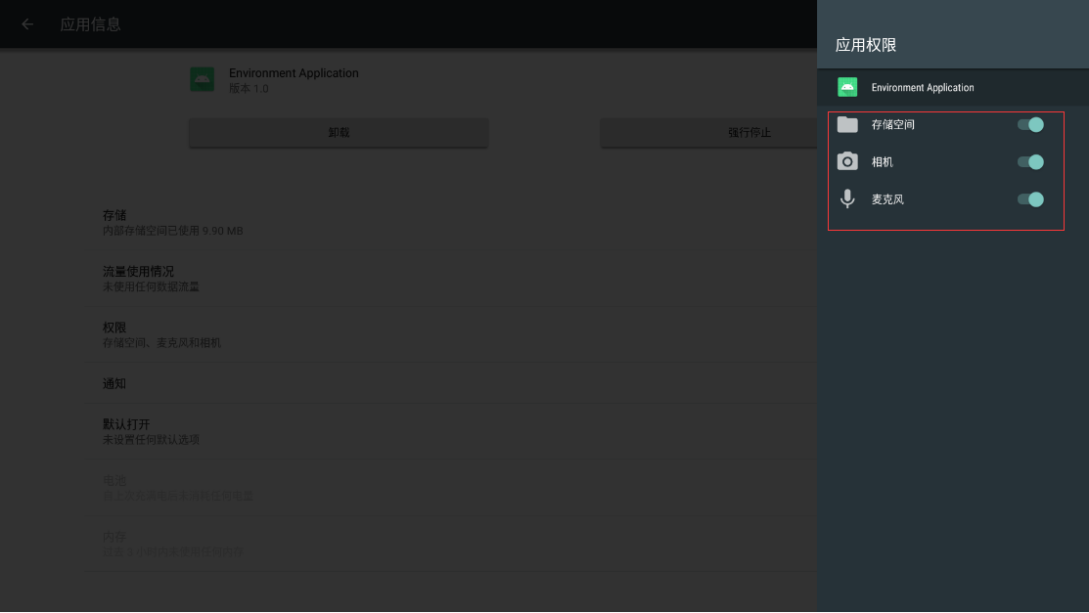
若通过ADB进入失败，则需手动在开发平台操作：

1. 开机输入权限密码后进入选择WIFI页面，多次点击屏幕右下角进入测试页面。
2. 测试页面的下方进入系统设置-打开系统设置进入设置页面。









开启了应用的权限后，需要重新编译应用到机器人开发平台，这时候，应用就可以正常运行了。

当应用在机器人开发平台上启动，三秒后，可以看到机器人的表情显示了“爱心”的图片，这就说明我们的能力包集成示例已经成功运行了。

## 步骤3：项目AIUI能力导入及配置

完成了工程项目的搭建后，我们需要在项目工程中集成AIUI能力，具体步骤如下：

1） 首先，导入配置文件，将aiui.cfg文件导入到应用工程目录：src/main/assets/cfg/（该文件示例工程同目录中可获取）；

2） 修改aiui.cfg文件中的配置信息，具体如下：

|  |
| --- |
| 1. /\* AIUI参数配置 \*/ 2. { 3. /\* login相关参数 \*/ 4. "login":{ 5. "appid":"\*\*\*\*\*\*\*", 6. "key":"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" 7. } 8. /\* 交互参数 \*/ 9. "interact":{ 10. "interact\_timeout":"60000", 11. "result\_timeout":"5000" 12. }, 13. /\* 全局设置 \*/ 14. "global":{ 15. "scene":"\*\*\*\*\*\*\*\*" 16. }, |

修改代码中\*标识的位置，主要包括appid，appkey，scene（场景），将AIUI平台创建的应用相关信息填入。

scene（场景）处参数为AIUI应用页面的场景模式名+”\_box”，如下图参数应为

1. "global":{
2. "scene":"main\_box"

},



## 步骤4：功能代码开发

首先，在我们的应用工程MainActivity.java同文件下新建MyApp.java文件，通过MyApp.java的onCreate()进行机器人硬件能力的初始化，只有完成了硬件能力的注册后，才能使用机器人的六麦的麦克风阵列进行录音，具体代码如下：

（**非完整代码，导入类的方法参考：常见问题-代码错误**）

|  |
| --- |
| public class MyApp extends Application {  @Override  public void onCreate() {  super.onCreate();  init();  }  private void init() {  StarLogAbility.getInstance().initAbility(this);  //基础能力初始化  StarCommonAbility.getInstance().initAbility(this,  RobotType.TYPE\_TEACHING, new StarCommonAbility.onResultCallback() {  @Override  public void onResult(boolean isSuccess, String hardCode) {  if (isSuccess) {  //硬件和业务状态初始化  switch (hardCode) {  case PartCode.HARDWARE\_PARTCODE.CODE\_EMOJI:  //设置初始表情  EmojiHelper.doEmojiBase();  break;  case PartCode.HARDWARE\_PARTCODE.CODE\_GPIO:  //默认加载的时候，将拾音方向设置为默认正前方的0度。  GPIOHelper.getInstance().setMainMic(0);  break;  case PartCode.HARDWARE\_PARTCODE.CODE\_CENTER\_LIGHT:  //关闭腹部灯带  CenterLightHelper.takeCenterLightOff();  break;  default:  break;  }  }  }  });  }  } |

以上完成了硬件能力的初始化，在我们的应用工程MainActivity.java文件的onCreate中进行机器人AIUI的能力初始化，同时启动AIUI服务，主要代码如下：

|  |
| --- |
| @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  *//初始化AIUI能力*  AIUIAbility.getInstance().initAIUIAbility(this);  *//设置AIUI监听*  AIUIAbility.getInstance().addNLPListener(this);  *//启动AIUI*  AIUIAbility.getInstance().start();  }  */\*\*  \* 解析AIUI结果，处理语义指令  \* @param semantic  \*/*  @Override  public void onAiuiResponse(String s) {  }  @Override  public void onAiuiWakeUp() {  }  @Override  public void onAiuiSleep() {  }  @Override  public void onAiuiEvent(AIUIEvent aiuiEvent) {  }  @Override  public void onError(int i) {  } |

在onPause方法中，停止AIUI服务，代码如下：

|  |
| --- |
| @Override protected void onPause() {  super.onPause();  *//停止AIUI服务* AIUIAbility.*getInstance*().release (); } |

到这里，我们就已经在应用工程中完成了AIUI能力的集成了，接下来，我们修改界面布局文件，添加一个用来显示语音识别后文本的控件。

打开/app/src/main/res/layout/activity\_main.xml这个布局文件，选择code模式，修改其中的代码如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">  <TextView  android:id="@+id/txtArea"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text=""  android:textSize="32sp"  android:layout\_centerInParent="true"/>  </RelativeLayout> |

在这里，添加了一个TextView控件，位于屏幕的中间位置，并将其ID设置为“txtArea”，这个控件将用来显示语音识别后的文本。

下面，将AIUI返回的结果进行解析，并将相应的文本进行显示，修改MainActivity类文件中的onAiuiResponse方法，代码如下：

|  |
| --- |
| */\*\**  *\* 解析AIUI结果，处理语义指令*  *\* @param bean*  *\*/*  @Override  public void onAiuiResponse(String bean) {  if (TextUtils.isEmpty(bean)) {  return;  }  try {  JSONObject object = new JSONObject(bean);  if (null == object) {  return;  }  JSONObject intentObject = object.optJSONObject("intent");  if (null == intentObject) {  return;  }  if(intentObject.has("text")){  TextView txtView = (TextView) findViewById(R.id.txtArea);  txtView.setText(intentObject.getString("text"));  }  } catch (Exception e) {  e.printStackTrace();  }  } |

在build.gradle(app)文件中添加如下代码：

|  |
| --- |
| ndk {  abiFilters "armeabi", "armeabi-v7a"  } |



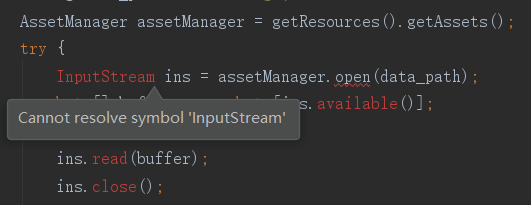
到这里，我们就完成所有功能代码的集成了，下面我们将机器人开发平台与开发电脑通过USB进行连接，然后将应用编译到机器人开发平台进行运行，具体的方法参考“实验准备：开发环境搭建”中编译调试部分。

当应用在机器人开发平台上正常运行时，可以尝试在开发平台的麦克风位置进行语音输入，当我们说的话能够在屏幕上显示出来的时候，实验就达到了我们的预期效果，如下图所示：



# 常见问题

* 1. **程序运行错误**
* 错误信息：程序出现崩溃异常
* 解决方法：出现该问题可能是因为程序录音没有权限，机器人的系统Android7.1.2需要动态申请权限，代码中如果没有动态进行麦克风权限的申请，需要手动到系统设置页面打开录音的权限，进入系统设置可以使用adb命令：adb shell am start -n com.android.settings/com.android.settings.Settings
  1. **代码错误**
* 错误信息：在添加应用代码的过程中，出现“cannot resolve symbol ‘xxxxx’ ”的提示信息，如下图所示：



* 解决方法：出现该问题是因为当前文件中没有导入相应的类，只需要将光标置于需要导入的类对象名称位置（如上图“InputStream”），然后同时按下键盘上的Alt + Enter键即可自动导入该类。
  1. **编译错误**
* 错误信息：应用编译运行时，没有播放合成的语音，并且调试信息里有报错：errorCode:10147

解决方法：检查AIUI配置文件中的appid和appkey的设置是否正确，是否跟AIUI平台创建的应用appid和appkey一致。

语音合成技术体验 - 实验指导书

# 一、实验目标

本次实验通过AIUI的配置及调用，开发一个关于语音合成的智能应用，让学生了解语音合成技术如何在机器人上使用，同时对语音合成技术的使用途径和步骤进行详细讲解，让学生理解起来更加轻松，本实验的具体实验目标如下：

1）了解AIUI开放平台账号申请、应用创建及配置；

2）掌握使用开发套件完成机器人智能应用的开发与调试；

3）掌握语音合成能力接口的调用开发。

# 二、实验内容

本次实验通过一个开发一个实际的简单智能应用，来对机器人使用的语音合成技术使用方法做详细的讲解。

# 三、实验步骤

## 步骤1：AIUI开放平台应用添加及设置

参考“语音识别技术体验\_实验指导书”完成AIUI开发平台的应用添加，获取应用的appid和appkey；

## 步骤2：项目工程搭建及能力包导入

参考“语音识别技术体验\_实验指导书”

## 步骤3：项目AIUI能力导入及配置

参考“语音识别技术体验\_实验指导书”

## 步骤4：功能代码开发

在本次实验中，我们完成以下的实验场景：

1. 当应用接收到语音“你是谁”时，通过语音合成文本“我是小途，很高兴为您服务。”进行回答；
2. 当应用接收到语音“你会干什么”时，通过语音合成文本“我可以陪你聊天，为你解答问题，还可以给你唱歌哦。”进行回答。

首先，在的应用工程MainActivity.java文件的onCreate方法中进行AIUI的能力初始化，并在文件中添加AIUI的回调方法；修改页面布局文件，添加一个ID为txtArea的TextView文本控件，具体代码请参考“语音识别技术体验\_实验指导书”；

然后，在onCreate方法中进行语言合成能力的初始化，同时将文本控件的显示文本设置为“您好，请问有什么可以帮您？”，代码如下

|  |
| --- |
| @Override protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main); AIUIAbility.getInstance().initAIUIAbility(this);   *//初始化语音合成能力*  SpeechHelper.getInstance().initSpeech(this);  AIUIAbility.getInstance().addNLPListener(this);  AIUIAbility.getInstance().start();  ((TextView)findViewById(R.id.txtArea)).setText("您好，请问有什么可以帮您？"); } |

接下来，我们在onAiuiResponse方法中进行语义结果的解析，同时添加相应的功能逻辑代码，如下：

|  |
| --- |
| */\*\**  *\* 解析AIUI结果，处理语义指令*  *\* @param semantic*  *\*/*  @Override  public void onAiuiResponse(String semantic) {  if (TextUtils.isEmpty(semantic)) {  return;  }  try {  JSONObject object = new JSONObject(semantic);  if (null == object) {  return;  }  JSONObject intentObject = object.optJSONObject("intent");  if (null == intentObject) {  return;  }  final TextView txtArea = (TextView)findViewById(R.id.txtArea);  if(intentObject.has("text")){  String txt = intentObject.getString("text");  if(TextUtils.equals(txt, "你是谁？")){  txtArea.setText("我是小途，很高兴为您服务。");  SpeechHelper.getInstance().speak("我是小途，很高兴为您服务。");  }  else if(TextUtils.equals(txt, "你会干什么？")){  txtArea.setText("我可以陪你聊天，为你解答问题，还可以给你唱歌哦。");  SpeechHelper.getInstance().speak("我可以陪你聊天，为你解答问题，还可以给你唱歌哦。", new TTS.OnSpeakCallback() {  @Override  public void onSpeak(String s) {  //需要在主线程中进行界面元素的修改  runOnUiThread(new Runnable() {  @Override  public void run() {  txtArea.setText("您好，请问有什么可以帮您？");  }  });  }  });  }  }  } catch (Exception e) {  e.printStackTrace();  }  } |

到这里，我们就已经完成所有应用功能代码的开发了，下面，我们将机器人开发平台与开发电脑通过USB进行连接，然后将应用编译到机器人开发平台进行运行，具体的方法参考“服务机器人开发环境搭建\_实验指导书”中编译调试部分。

应用在机器人开发平台上运行时，我们来验证具体的效果，流程如下：

1. 在开发平台的麦克风位置语音说“你是谁”，平台会进行语音播报 “我是小途，很高兴为您服务。”，同时界面上显示文本“我是小途，很高兴为您服务。”；
2. 在开发平台的麦克风位置语音说“你会干什么”，平台会进行语音播报“我可以陪你聊天，为你解答问题，还可以给你唱歌哦。”，同时界面上显示文本“我是小途，很高兴为您服务。”，当语音播报完成后，界面显示的文本重新变回“您好，请问有什么可以帮您？”。当以上流程在应用中验证没有问题时，实验就已经达到了我们的预期效果。

# 四、常见问题

**编译错误**

* 错误信息：应用编译运行时，没有播放合成的语音，并且调试信息里有报错：errorCode:10147
* 解决方法：检查AIUI配置文件中的appid和appkey的设置是否正确，是否跟AIUI平台创建的应用appid和appkey一致。

语义理解技术体验 - 实验指导书

# 一、实验目标

本次实验通过AIUI开放平台的语义技能的配置，结合机器人客户端智能应用的开发，让学生了解AIUI语义理解的调用过程，同时对语义配置的流程和步骤进行详细讲解，让学生理解掌握起来更加轻松，本实验的具体实验目标如下：

1）了解AIUI商店技能、自定义技能和问答库的添加基本流程；

2）了解AIUI平台语义设计和配置方法；

3）掌握基础的语义技能中意图、语料、实体及辅助词的配置方法。

# 二、实验内容

本次实验通过完成一个智能语音控制的音乐播放器，来深入了解语义平台的使用流程以及相应的语义功能在项目中的实现方法。

# 三、实验步骤

## 步骤1：AIUI开放平台应用添加及设置

在使用AIUI开放平台的语义理解功能之前，我们需要先参考“语音识别技术体验\_实验指导书”完成应用的添加，然后，在应用配置页面开启应用的“语义理解”功能，如下图所示：



开启了应用的语义理解功能之后，我们就可以对应用的语义技能进行配置了，包括商店技能、自定义问答和自定义技能的添加，其中自定义问答和自定义技能需要在技能工作室中进行配置。

##### 1.1 商店技能

AIUI开放平台的技能商店中内置了平台的所有开放技能，包括每种技能的使用介绍、语义协议。开放的技能覆盖吃住行生活娱乐等众多垂直领域，帮助开发者快速实现基本需求。例如：

* 通过天气技能，查询天气（“今天的天气怎么样”）
* 通过音乐技能，播放音乐（“来一首刘德华的歌”）
* 通过空调技能，控制家居设备空调（“好热呀，打开空调”）

技能商店中的很多开放技能都提供免费使用，我们可以直接在应用中集成这些技能；在应用配置页面的“语义技能”区域点击下方的“商店技能”，然后勾选需要的技能后保存修改即可。

在弹出的技能添加界面，选择我们需要添加的商店技能（在本次实验中，我们添加“天气”，“航班”，“笑话”三个商店技能），然后点击界面下方的“确定”按钮，完成技能的添加：



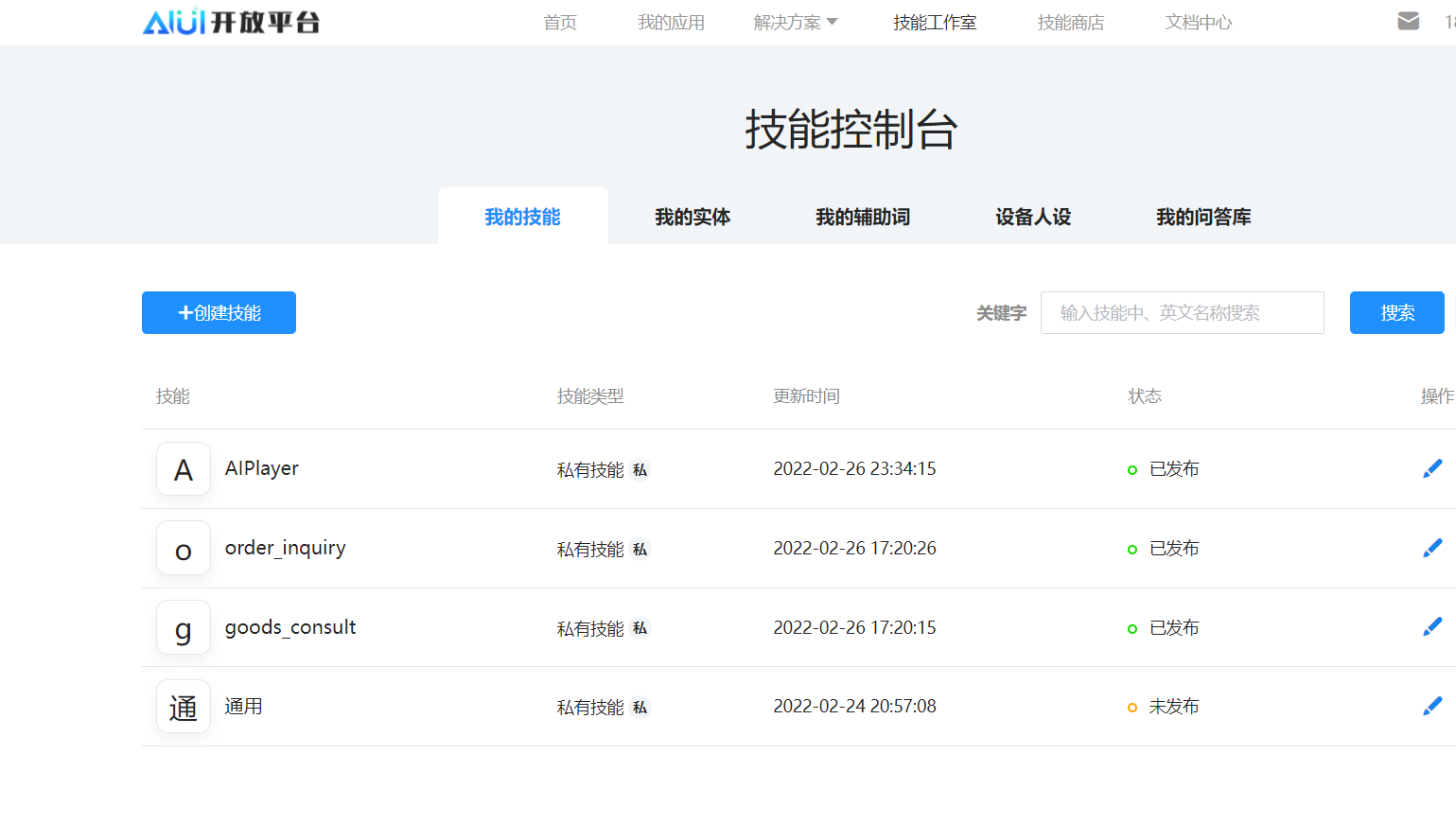
完成技能的添加后，我们需要点击页面左上角的“保存修改”按钮，保存我们应用的配置信息。

##### 1.2 技能工作室

当商店技能无法满足我们个性化的需求，需要处理特定的任务，例如，实现校园卡业务的咨询和办理，日常的带路等任务时，我们可以利用技能工作室（Skill Studio）完成自定义技能的设计、开发、测试、发布等工作；开发者可以通过可视化的界面，快速高效地将自己的创意、产品或服务，通过语音技能传达给智能硬件的用户。

点击AIUI开放平台页面上方的“技能工作室”菜单，通过引导，进入“技能控制台”页面。





###### *1.2.1我的问答库*

在某些较为简单场景中，我们希望智能硬件设备在接收到指定问题时能够回答固定的一种或几种答复时，例如Q：“你的名字是什么”A：“我叫小途，是你的贴心秘书”，可以利用“我的问答库”功能实现。

问答库的QA没有语法上的要求，按正常表述习惯录入即可。同时平台支持“一问一答”、“一问多答”和“多问多答”的配置。

操作步骤：

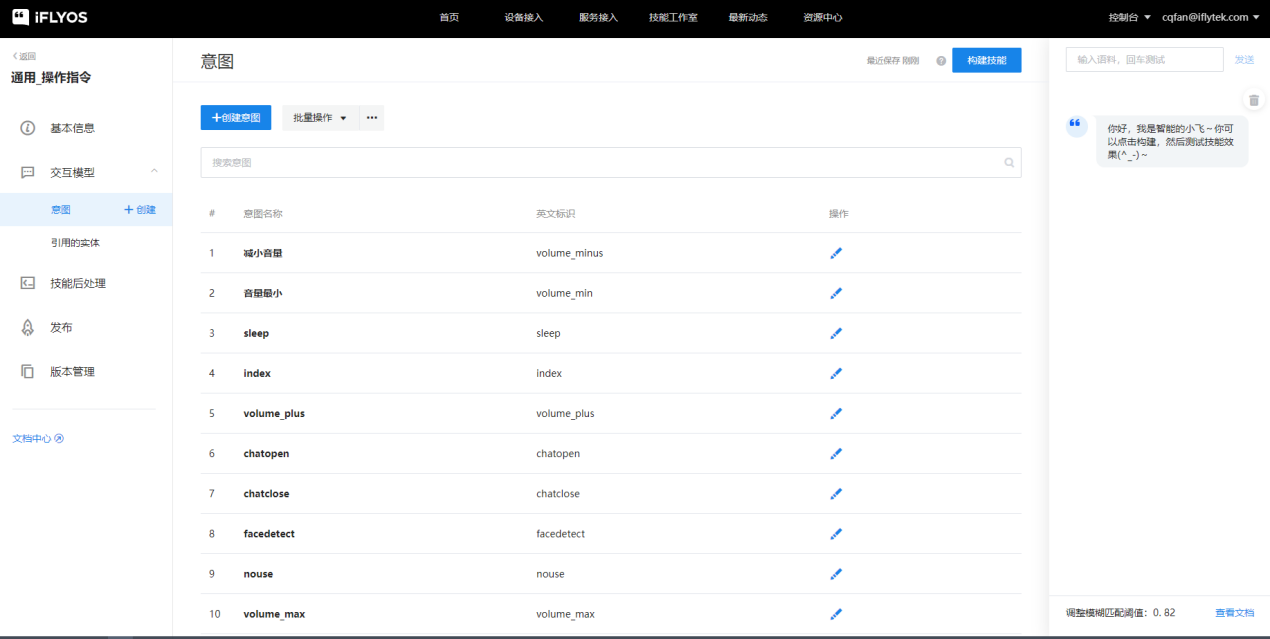
1. 点击“创建问答库”按钮，填入问答库名称，点击右上角“构建问题”即可完成创建。
2. 继续创建主题，填入想要的问题和答案即可。

另外，问答对可以通过excel模板编辑后进行批量导入和导出操作。



###### *1.2.2我的技能*

语音技能（Skill）是指智能硬件可以使用自然的语言交流提供服务的能力，实现不同服务的能力，可以定义为一个技能。例如，实现天气查询服务，我们定义了“天气”技能；实现机器人控制服务，我们定义了“机器人控制”技能，如下图所示：



**意图**指用户在技能中可能的请求或目的。作为“机器人控制”技能，可以满足用户增大音量、减小音量、查询电量等需求。

用户在完成某一意图时，可能会有很多种提问方式，这些提问方式我们统称为语料。为保证良好的体验，开发者需要尽可能完善用户所有的提问方式。

下表展示了“机器人控制”技能中的意图和语料。

|  |  |  |
| --- | --- | --- |
| **技能** | **意图** | **语料** |
| **机器人控制** | 增大声音 | 声音大一点  大点声 |
|  | 减小声音 | 声音小点  我听不见 |
|  | 电量查询 | 你还有多少电  你没电了吗 |

以业务办理技能下的“办理材料”意图为例，用户的提问方式可能有：

* 办理市民卡需要哪些材料
* 办理学生卡需要哪些材料
* 什么材料能办理市民卡
* 什么材料能办理学生卡

在如上的语料中，有1个关键槽位“办事项名称”，槽位用花括号{ } 标识。具体办事项的集合，我们称之为实体，可以在具体配置中引用。

为了提高语料配置效率，我们可以把上例中的语料抽象为：

* 办理{todolist}需要哪些材料
* 什么材料能办理{todolist}

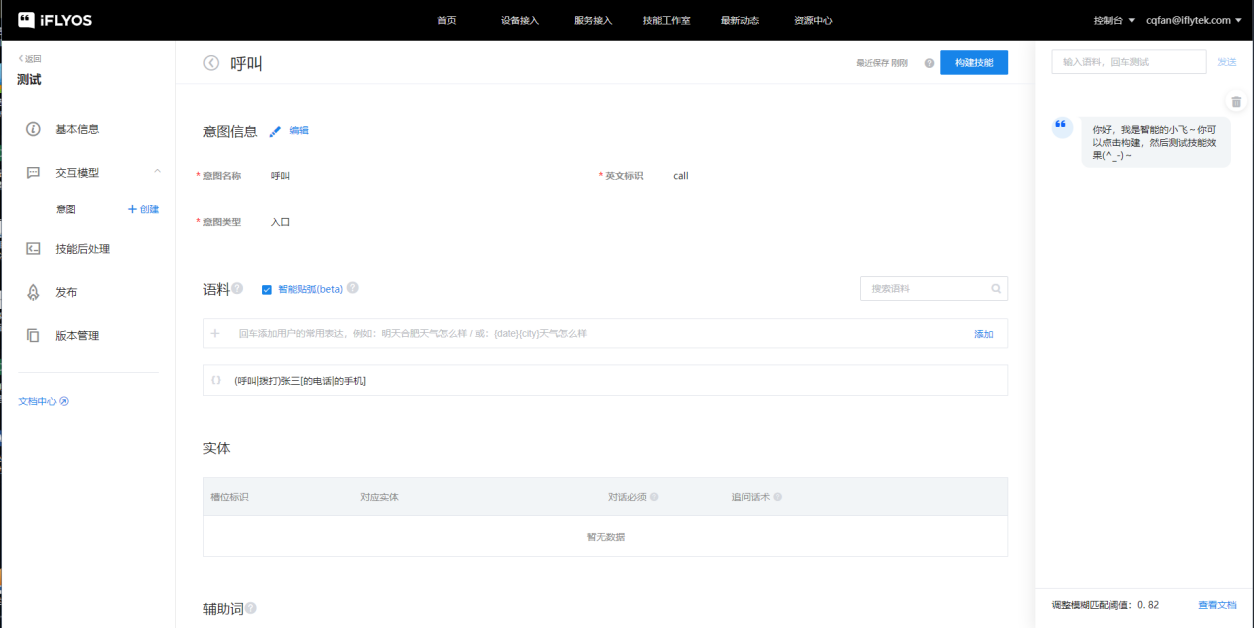
以上纯文本抽象过后的语句我们称为“模板语料”。

为了方便快速编写相似语料，模板语料提供了可选的写法。

|  |  |
| --- | --- |
| **符号** | **释义** |
| **（ ）** | 必选符，不可缺省 |
| **[ ]** | 可选符，可缺省 |
| **{ }** | 语义槽 |
| **|** | 或 |

帮我（呼叫|拨打）张三[的电话|的手机]，等同于：

* 帮我呼叫张三
* 帮我拨打张三的电话
* 帮我拨打张三的手机



注意：一句语料中允许出现至多5个中括号和小括号。技能设置完成后，点击发布才可以部署到开发平台上。

###### *1.2.3我的实体*

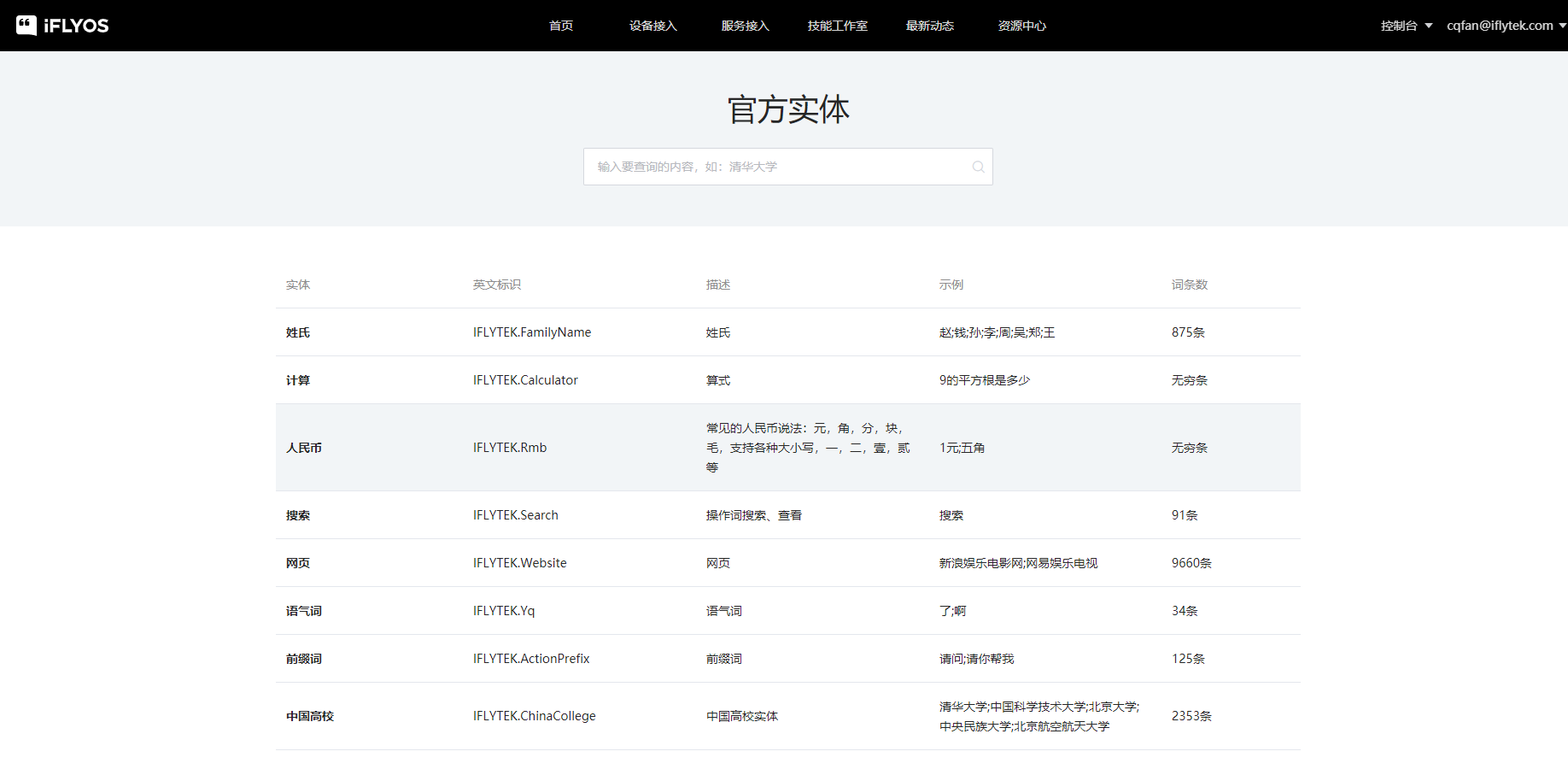
语料中，同一类型的名词将会被总结为实体。

示例：

* 明天合肥天气怎么样
* 后天北京天气怎么样

抽象成模板语料为：{time}{city}天气怎么样

其中，time为时间实体，city为城市实体。平台提供了部分较为通用的官方实体，可在语料配置中引用。



对于无法穷举的实体，平台提供了**通配实体**，可设定1-20个任意字符。使用时可根据具体情况设定字符个数。例如，语义槽为车牌号时，我们可以引用通配实体，并设定通配实体字符个数为7-8。

对于官方实体无法满足的实体需求，需要我们在平台建立自己的实体，即**静态实体**。

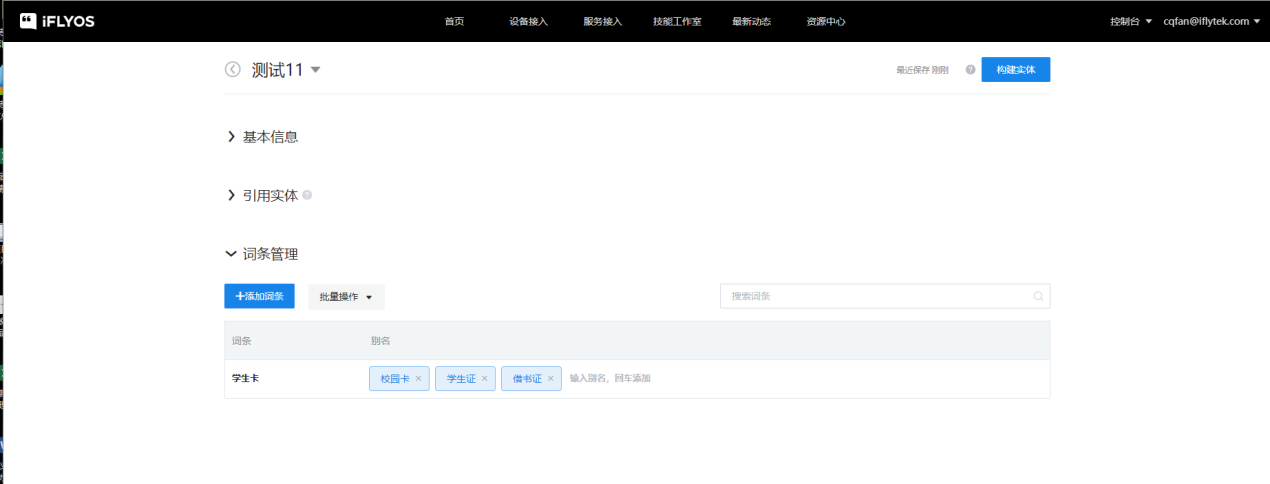
示例：

* 我要办理{todolist}

todolist语义槽中我们希望填入具体的办事项名称，办事项名称有学生证和市民卡两项。另外，我们发现，对于学生卡的说法，还有借书证、校园卡、学生证等，可一并填入别名中。

操作步骤：

1. 新建实体，实体类型为静态实体
2. 添加词条，学生卡、市民卡
3. 给词条扩充别名



###### *1.2.4我的辅助词*

为了满足开发者快速编写语料的需求，同时兼顾语料的可维护性，平台提出了辅助词的概念。

实体和辅助词在本质上都是同一类词的集合，一些会严重影响到我们理解的词，我们倾向使用实体去定义，另一些不会太影响我们理解的词，我们则倾向于用辅助词去定义。

在如下语料中：

* [帮我][定一下]从{合肥}到{广州}的[飞机票]
* [我要][买一张]从{合肥}到{广州}的[机票]
* [给我][预约]从{合肥}到{广州}的[航班]

语料的主体结构没有变化，开发者实际关注的信息也只有出发地和目的地，但是用户的表达方式却千变万化，此时我们可以把这句话抽象成：

{helpMe}{book}从{startCity}到{endCity}的{ticket}

其中，startCity和endCity为实体，helpMe、book和ticket的为辅助词。这些辅助词包含的内容如下：

|  |  |  |
| --- | --- | --- |
| helpme | book | ticket |
| 帮我 | 定一下 | 飞机票 |
| 我要 | 买一张 | 机票 |
| 给我 | 预约 | 航班 |
| 我想 | 购买 | 客机 |
| …… | …… | …… |

##### 1.3 自定义技能和自定义问答

完成了技能工作室操作流程知识的学习，我们开始为本次实验添加相关的实验语义数据。

首先，我们在技能工作室的“我的问答库”中添加一个名称为“demo\_qa”的问答库，并将下表的数据添加到问答库中：

|  |  |  |
| --- | --- | --- |
| **主题** | **问题** | **回答** |
| 姓名 | 你是谁 | 我是小途 |
| 你叫什么名字 | 我的名字叫小途，很高兴为您服务 |
| 能力 | 你会干嘛 | 我会陪你聊天，为你解答问题，还能唱歌给你听哦 |
| 你能做什么 |

完成了自定义问答的添加后，我们来继续添加自定义技能的语义数据，，参考下列表格，完成自定义技能、意图和语料的设置：

1、技能、意图和语料

|  |  |  |
| --- | --- | --- |
| **技能** | **意图** | **语料模板** |
| AIPlayer | PlayPlayerMusic | {want\_listen}{songer}的歌 |
| {want\_listen}{songer}唱的歌 |
| PlayMusic | {want\_listen}{song} |
| PlayPause | 暂停 |
| 停一下 |
| PlayResume | 继续 |

2、实体

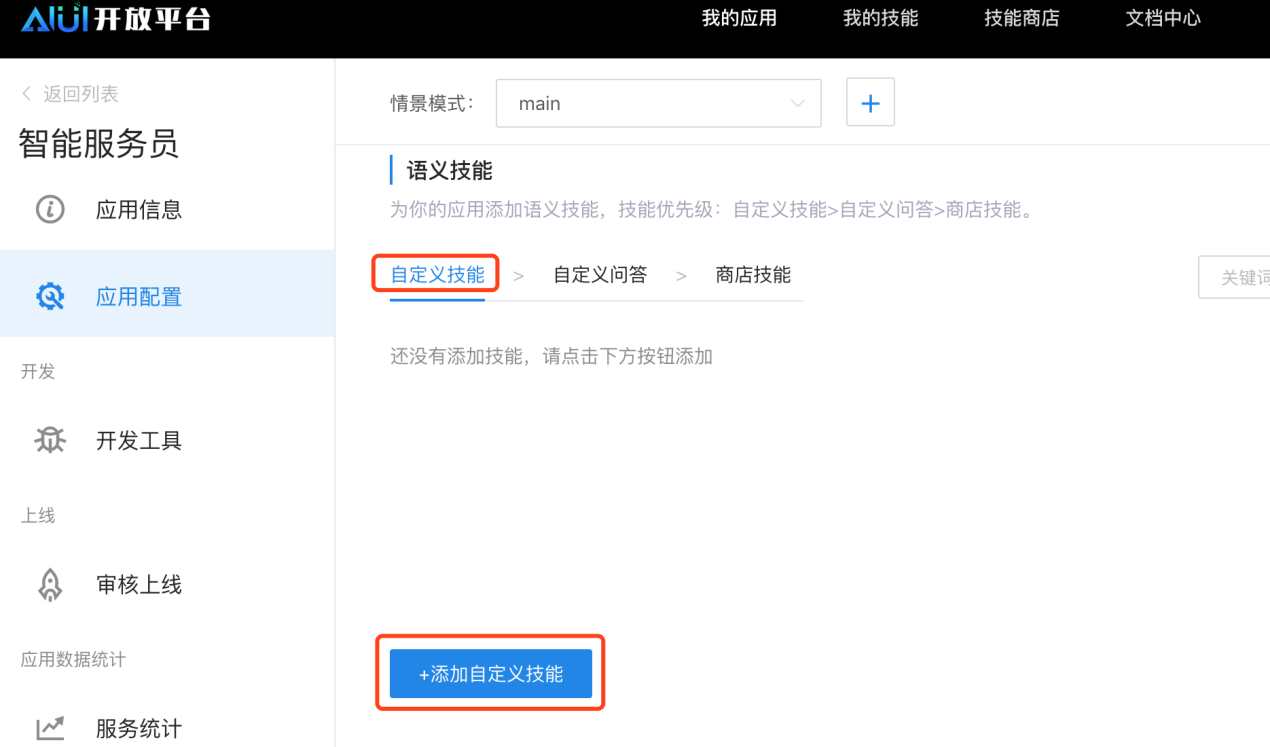
|  |  |  |
| --- | --- | --- |
| **实体名** | **词条名** | **别名** |
| play\_player | player1 | 刘德华 |
| palyer2 | 周杰伦 |
| player3 | 张靓颖 |
| play\_music | song1 | 练习 |
| song2 | 搁浅 |
| song3 | 我的梦 |

3、辅组词

|  |  |
| --- | --- |
| **辅组词名** | **别名** |
| play\_help | 我想听 |
| 来一首 |
| 唱一个 |

完成技能工作室中本次实验语义的数据设置后，我们开始将相应的自定义语义技能和自定义问答库添加到我们的实验应用中。

添加自定义技能与问答：在应用配置页面，点击语义技能区域的“用户自定义”链接，然后点击勾选下方已经发布的技能或问答，最后保存修改。





完成自定义技能和自定义问答的添加后，点击应用配置页面的“保存修改”按钮使配置生效。

## 步骤2：项目工程搭建及能力包导入

参考“语音识别技术体验\_实验指导书”

## 步骤3：项目AIUI能力导入及配置

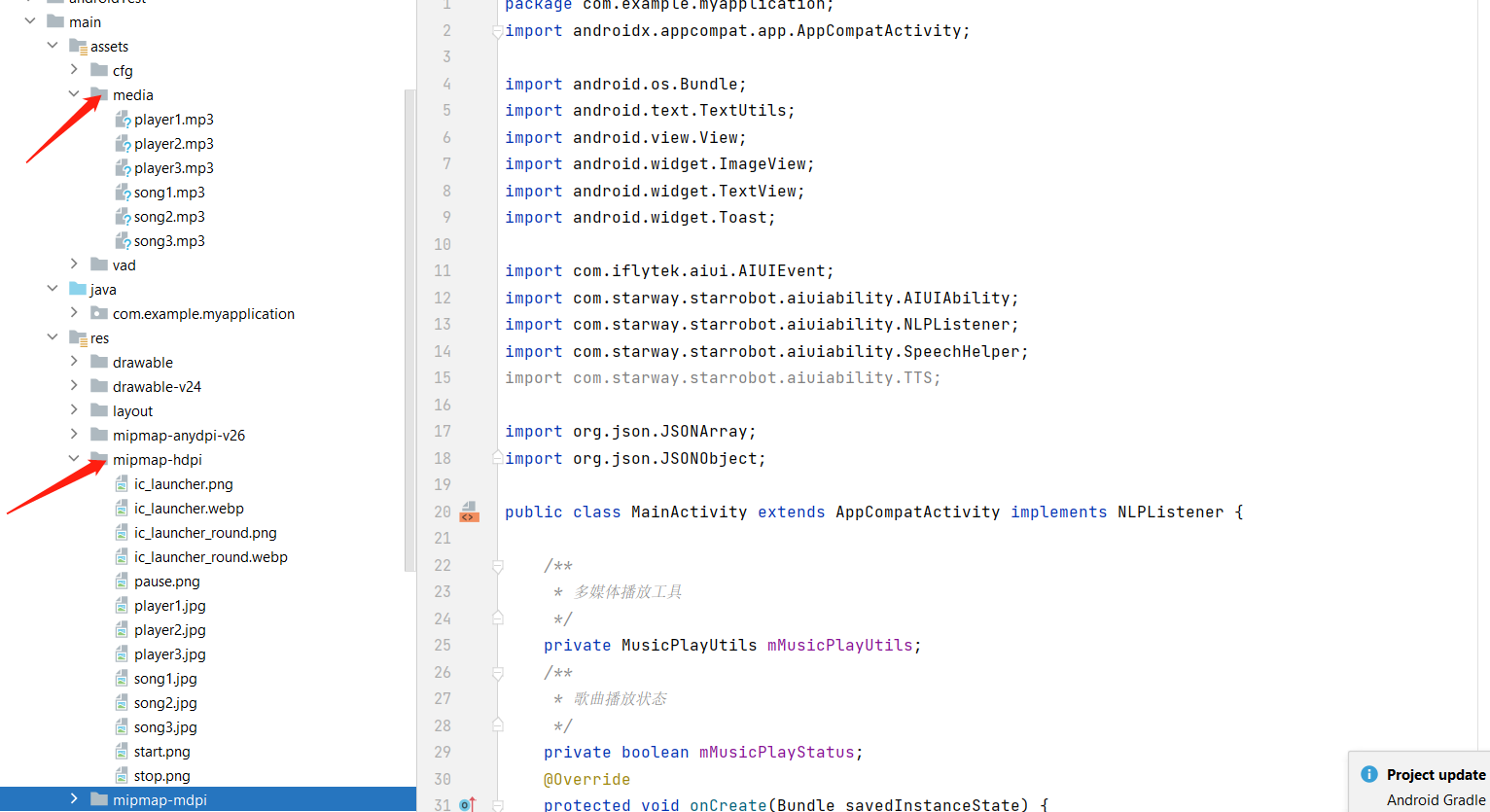
参考“语音识别技术体验\_实验指导书”

## 步骤4：功能代码开发

首先，在我们的应用工程MainActivity.java文件的onCreate中进行AIUI的能力初始化，并在文件中添加相应的回调方法，具体代码请参考“语音识别技术体验\_实验指导书”；

将音频资源media文件夹复制至app\src\main\assets文件夹内

将图标资源mipmap-hdpi文件夹复制至app\src\main\res文件夹内



接下来，我们修改布局文件，完成一个音频播放器的UI界面，具体代码如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">  <RelativeLayout  android:id="@+id/answerArea"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  <TextView  android:id="@+id/txtArea"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_centerInParent="true"  android:textAlignment="center"  android:visibility="gone"  android:text=""/>  </RelativeLayout>  <LinearLayout  android:id="@+id/palyerArea"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="horizontal">  <LinearLayout  android:layout\_width="500dp"  android:layout\_height="match\_parent"  android:orientation="vertical"  android:padding="50dp">  <ImageView  android:id="@+id/player\_icon"  android:layout\_width="400dp"  android:layout\_height="250dp"  android:src="@mipmap/icon"/>  <TextView  android:id="@+id/player\_txt"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:textAlignment="center"  android:layout\_marginTop="20dp"  android:text="正在播放XXX的歌曲XXX"/>  </LinearLayout>  <LinearLayout  android:layout\_width="2px"  android:layout\_height="match\_parent"  android:background="@color/white">  </LinearLayout>  <RelativeLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  <ImageView  android:id="@+id/play\_btn"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_centerInParent="true"  android:src="@mipmap/start"/>  </RelativeLayout>  </LinearLayout>  </RelativeLayout> |

由于本次实验需要开发的语音智能播放器包含了音频播放的功能，所以需要添加一个用于音频文件播放控制的功能类，创建一个MusicPlayUtils.java文件，与MainActivity类文件同目录位置，代码如下：

|  |
| --- |
| */\*\**  *\* 音乐播放工具类*  *\*/*  public class MusicPlayUtils {  private Context mContext;  private MediaPlayer mMediaPlayer;  */\*\**  *\* 监听播放完成*  *\*/*  public interface onPlayCompletedCallback {  void onCompleted();  }  public MusicPlayUtils(Context ctx){  mContext = ctx;  mMediaPlayer = new MediaPlayer();  }  */\*\**  *\* 判断是否正在播放*  *\* @return*  *\*/*  public boolean isPlaying() {  try {  return mMediaPlayer.isPlaying();  }  catch (Exception e) {  return false;  }  }  */\*\**  *\* 停止播放*  *\*/*  public void stop() {  if (null != mMediaPlayer) {  mMediaPlayer.stop();  mMediaPlayer.reset();  }  }  */\*\**  *\* 暂停播放*  *\*/*  public void pause(){  if(null != mMediaPlayer && mMediaPlayer.isPlaying()){  mMediaPlayer.pause();  }  }  */\*\**  *\* 继续播放*  *\*/*  public void resume(){  if(null != mMediaPlayer){  mMediaPlayer.start();  }  }  */\*\**  *\* 播放*  *\* @param fileName 音频文件地址*  *\* @param mCallback 播完的回调*  *\* @return*  *\* @throws JSONException*  *\*/*  public void playLocalMedia(String fileName, final onPlayCompletedCallback mCallback) {  if (null == mMediaPlayer) {  return;  }  if(mMediaPlayer.isPlaying()){  Log.e("MediaPlayer", "MediaPlayer is busying now!");  return;  }  try {  AssetManager am = mContext.getAssets();  mMediaPlayer.setDataSource(am.openFd(fileName));  mMediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {  @Override  public void onCompletion(MediaPlayer mp) {  mCallback.onCompleted();  }  });  mMediaPlayer.prepare();  mMediaPlayer.start();  } catch (IOException e) {  e.printStackTrace();  }  }  } |

接下来，我们在MainActivity类文件中进行UI控件及播放器相关能力的初始化，代码如下：

|  |
| --- |
| */\*\**  *\* 多媒体播放工具*  *\*/*  private MusicPlayUtils mMusicPlayUtils;  */\*\**  *\* 歌曲播放状态*  *\*/*  private boolean mMusicPlayStatus;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  *//AIUI能力的初始化*  AIUIAbility.getInstance().initAIUIAbility(this);  AIUIAbility.getInstance().addNLPListener(this);  AIUIAbility.getInstance().start();  *//语音合成初始化*  SpeechHelper.getInstance().initSpeech(this);  SpeechHelper.getInstance().setVoicer("xiaoyan");  mMusicPlayStatus = false;  *//播放器初始化*  mMusicPlayUtils = new MusicPlayUtils(this);  *//界面播放按钮，点击*  ImageView playBtn = (ImageView)findViewById(R.id.play\_btn);  playBtn.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View view) {  if(mMusicPlayStatus){  if(mMusicPlayUtils.isPlaying()) {  musicPlayPause();  }  else{  musicPlayResume();  }  }  else{  Toast.makeText(MainActivity.this, "当前没有选择播放的歌曲", Toast.LENGTH\_SHORT);  }  }  });  } |

然后，在MainActivity类文件中的onAiuiResponse方法中进行语义结果的解析，调用新增的方法处理相应的语义指令，完成智能语音音乐播放器功能逻辑代码，具体代码如下：

|  |
| --- |
| */\*\**  *\* 解析AIUI结果，处理语义指令*  *\* @param semantic*  *\*/*  @Override  public void onAiuiResponse(String semantic) {  if (TextUtils.isEmpty(semantic)) {  return;  }  try {  JSONObject object = new JSONObject(semantic);  JSONObject intentObject = object.optJSONObject("intent");  if (null == intentObject) {  return;  }  *//通过判断返回的结果语义中是否包含answer字段，确认是否是“商店技能”或“自定义问答”*  if(intentObject.has("answer")){  *//显示回答文本区域，隐藏播放器界面*  findViewById(R.id.answerArea).setVisibility(View.VISIBLE);  findViewById(R.id.playerArea).setVisibility(View.GONE);  *//显示回答文本*  final TextView answerView = (TextView)findViewById(R.id.txtArea);  JSONObject answerObject = intentObject.optJSONObject("answer");  String answerText = answerObject.getString("text");  answerView.setText(answerText);  *//语音合成*  SpeechHelper.getInstance().speak(answerText);  }  else if(intentObject.has("service") && TextUtils.equals(intentObject.getString("service"), "OS8501282703.AIPlayer")){  JSONArray semanticArray = intentObject.optJSONArray("semantic");  JSONObject semanticObject = (JSONObject)semanticArray.get(0);  String intent = semanticObject.getString("intent");  switch (intent){  case "PlayPlayerMusic":*//播放指定歌手的歌曲*  playMusic(semanticObject);  break;  case "PlayMusic":*//播放指定歌曲*  playMusic(semanticObject);  break;  case "PlayPause":*//播放暂停*  if(mMusicPlayStatus) {  musicPlayPause();  }  break;  case "PlayResume":*//播放继续*  if(mMusicPlayStatus) {  musicPlayResume();  }  break;  }  }  } catch (Exception e) {  e.printStackTrace();  }  }  */\*\**  *\* 暂停播放*  *\*/*  private void musicPlayPause(){  mMusicPlayUtils.pause();  ImageView playBtn = (ImageView)findViewById(R.id.play\_btn);  playBtn.setImageResource(R.mipmap.start);  }  */\*\**  *\* 继续播放*  *\*/*  private void musicPlayResume(){  mMusicPlayUtils.resume();  ImageView playBtn = (ImageView)findViewById(R.id.play\_btn);  playBtn.setImageResource(R.mipmap.pause);  }  */\*\**  *\* 播放歌曲*  *\* @param semanticObject*  *\*/*  private void playMusic(JSONObject semanticObject){  try {  JSONArray slotsArray = semanticObject.optJSONArray("slots");  JSONObject slotsObject = (JSONObject) slotsArray.get(0);  String normValue = slotsObject.getString("normValue");  String value = slotsObject.getString("value");  playMusic(normValue);  findViewById(R.id.playerArea).setVisibility(View.VISIBLE);  findViewById(R.id.answerArea).setVisibility(View.GONE);  int resId =  getResources().getIdentifier(normValue, "mipmap", this.getPackageName());  ImageView playerIcon = (ImageView)findViewById(R.id.player\_icon);  playerIcon.setImageResource(resId);  TextView playerTxt = (TextView)findViewById(R.id.player\_txt);  if(TextUtils.equals(semanticObject.getString("intent"), "PlayMusic")){  playerTxt.setText("正在播放歌曲“" + value + "”");  }  else {  playerTxt.setText("正在播放" + value + "的歌曲");  }  ImageView playBtn = (ImageView)findViewById(R.id.play\_btn);  playBtn.setImageResource(R.mipmap.pause);  }  catch (Exception e){  e.printStackTrace();  }  }  */\*\**  *\* 播放歌曲*  *\* @param song 歌曲名称*  *\*/*  private void playMusic(String song){  mMusicPlayStatus = true;  mMusicPlayUtils.playLocalMedia(  "media/" + song + ".mp3", new MusicPlayUtils.onPlayCompletedCallback() {  @Override  public void onCompleted() {  mMusicPlayStatus =false;  TextView playerTxt = (TextView)findViewById(R.id.player\_txt);  playerTxt.setText("");  ImageView playBtn = (ImageView)findViewById(R.id.play\_btn);  playBtn.setImageResource(R.mipmap.start);  }  });  } |

注：蓝色代码处替换自己账号的标识号



到这里，我们就已经完成所有应用功能代码的开发了，下面，我们将机器人开发平台与开发电脑通过USB进行连接，然后将应用编译到机器人开发平台进行运行，应用在机器人开发平台上运行时，当应用启动后，我们按照如下流程，对我们的实验成果进行校验：

1. 对开发平台说“今天天气怎么样？”，开发平台能够将今天的具体天气进行语音回答，同时在界面上显示相应的回答文本；
2. 对开发平台说“你能做什么”，开发平台能够语音回答“我会陪你聊天，

为你解答问题，还能唱歌给你听哦”，同时在屏幕上显示相应的文本；

1. 对开发平台说“播放刘德华的歌曲”，开发平台的屏幕显示智能播放器界

面，播放器的显示“刘德华”的图片，按照显示播放状态（需要替换相应的图片以实现）

1. 对开发平台说“暂停”或“继续”，开发平台会做出响应，播放按钮显示

相应状态（需要替换图片以实现）

对以上功能流程的验证，如果在平台上可以得到正确的反馈时，说明我们的实验已经达到了预期效果。

# 四、常见问题

* 错误信息：应用编译运行时，没有播放合成的语音，并且调试信息里有报错：errorCode:10147
* 解决方法：检查AIUI配置文件中的appid和appkey的设置是否正确，是否跟AIUI平台创建的应用appid和appkey一致。

语音技术综合应用 - 实验指导书

# 一、实验目标

本次实验通过设计一个智能语音交互应用场景，应用中使用到了语音识别、语音合成、语义理解等技术，让学生了解语音交互智能技术如何在机器人上使用，同时对语音交互技术的使用途径和步骤进行详细讲解，让学生理解起来更加轻松，本实验的具体实验目标如下：

1）熟练掌握语音合成、语音识别、语义理解相关技术的使用方法；

2）能够基于业务需求完成简单的语义设计；

3）掌握简单的语音交互场景设计，并能完成语音交互智能应用的开发。

# 二、实验内容

本次实验通过实现一个自动问答的商城智能语音客服系统，对语音合成和语音识别、语义理解等AI技术在实际智能应用中的综合应用，达到人机智能交互的效果。

# 三、实验步骤

## 步骤1：交互场景需求设计

在本次实验中，我们基于机器人开发平台完成一个智能语音客服系统，该系统围绕在线商城的业务咨询场景，可以识别和理解用户通过语音提出的问题，并对问题进行语音回答；

该系统需要支持以下场景的语音交互问答：

1. 闲聊（包括天气问询、维基百科、唐诗宋词）
2. 商品咨询（包括商品描述、价格、库存信息等）
3. 订单查询（查询订单状态、物流状态等）

所有的这些功能，都是通过人机交互的方式，由机器人开发平台与用户进行直接的语言沟通进行展现。

## 步骤2：AIUI开放平台应用添加及设置

（1）参考“语音识别技术体验\_实验指导书”完成AIUI开放平台的应用添加，获取应用的appid和appkey；

（2）在应用配置页面开启应用的“语义理解”功能；

（3）添加商店技能

在应用配置页面完成商店技能“天气”、“百科”、“诗词对答”的添加；

（4）添加自定义问答

* 在“技能工作室”中添加问答库“shop\_qa”
* 在问答库“shop\_qa”中添加问答数据，内容参考下表

|  |  |  |
| --- | --- | --- |
| **主题** | **问题** | **答案** |
| 你好 | 在吗 | 您好，请问有什么可以帮您 |
| 你好 | 亲，很高兴为您服务 |
| 有人吗 | 在呢，有什么可以为您服务 |
| 你会什么 | 你能做什么 | 我可以查询商品和订单信息哦 |
| 我可以问什么问题 | 您可以问商品信息也可以问订单信息哦 |

* 在应用配置页面完成自定义问答“shop\_qa”的添加

（5）添加自定义技能

* 根据场景在“技能工作室”中添加自定义AIUI技能商品咨询“goods\_[consult](http://www.baidu.com/link?url=G49DFIL75p_h1-KQ_b2vNzl6D3horZQbvabKuxIzxkcmzsblvHeud-DMYTlOY3uG_SdtQ7vWzJs3QWlT57ah_LsyvflvwsV5Fx2Qrlg7qrS" \t "https://www.baidu.com/_blank)”和订单查询“order\_inquiry”
* 如用户进入商品咨询技能，可能想要根据商品名称查询对应的商品描述、商品价格、商品库存等意图信息；订单查询技能，需要根据订单号查询订单及物流状态意图信息。我们可以定义添加对应的意图及语料。
* 根据下面语义模板表的数据，完成自定义技能商品咨询“goods\_[consult](http://www.baidu.com/link?url=G49DFIL75p_h1-KQ_b2vNzl6D3horZQbvabKuxIzxkcmzsblvHeud-DMYTlOY3uG_SdtQ7vWzJs3QWlT57ah_LsyvflvwsV5Fx2Qrlg7qrS" \t "https://www.baidu.com/_blank)”和订单查询“order\_inquiry”的语义数据添加，数据的添加方法可参考“语义理解技术体验\_实验指导书”相关操作步骤。

1. 技能、意图和语料

|  |  |  |
| --- | --- | --- |
| **技能** | **意图** | **语料** |
| 商品咨询  goods\_[consult](http://www.baidu.com/link?url=G49DFIL75p_h1-KQ_b2vNzl6D3horZQbvabKuxIzxkcmzsblvHeud-DMYTlOY3uG_SdtQ7vWzJs3QWlT57ah_LsyvflvwsV5Fx2Qrlg7qrS" \t "https://www.baidu.com/_blank) | 查询商品描述  describe | * 我想了解下讯飞翻译机的信息   **[**{want}{Inquiry}**]**{goods\_name}的信息   * 能介绍下阿尔法蛋吗 * 介绍下阿尔法蛋   [能|能否|能够]{[introduce](http://www.baidu.com/link?url=PUbbZISK1FbR2-vU7OjhWjZLxwzfTHMnfhJNkBBqwxrks09XHmZwS5NrmJ3vFFR4iioChKyS0W49rloKKDid3wC6iczQqf_b4EKRk4b-AYi" \t "https://www.baidu.com/_blank)}{goods\_name}   * 阿尔法蛋是什么   {goods\_name}是什么   * 讯飞录音笔有哪些功能 * 我想了解下讯飞翻译机的功能   **[**{want}{Inquiry}**]**{goods\_name}{function} |
| 商品价格  price | * 我问下讯飞录音笔卖多少钱 * 阿尔法蛋卖多少钱 * 我想查询下讯飞录音笔的价格 * 讯飞翻译机的价格   **[**{want}**][**{Inquiry}**]**{goods\_name}{price}   * 能否帮我查下阿尔法蛋的价格吗   [{request}][帮我][{Inquiry}]  {goods\_name}{price}   * 阿尔法蛋怎么卖   {goods\_name}怎么卖 |
| 商品库存  stock | * 讯飞翻译机还有货吗 * 问一下阿尔法蛋有货吗 * 讯飞翻译机有多少库存 * 讯飞录音笔还有多少货 * ...   **[**{want}**][**{Inquiry}**]**{goods\_name}{stock} |
| 商品名称  goods | * 讯飞翻译机   {goods\_name} |
| 订单查询  order\_inquiry | 查询订单状态  order\_status | * 我买的翻译机发货了吗? * 讯飞翻译机有没有发货？   [我][买|购买]{goods\_name}**(**发货了吗**|**有没有发货**|**发货没有**)**   * 我买的东西发货了吗? * 商品发货没有？   [我][买|购买]{goods}**(**发货了吗**|**有没有发货**|**发货没有**)**   * 我想查询下订单信息 * 我要查订单   [{want}][{Inquiry}]订单**[**信息**]** |
| 订单号  order | * 订单号88888888   [订单号]{order\_number}   * 我要查询订单88888888的信息   [{want}][{Inquiry}][订单]{order\_number}[的信息] |

1. 实体

|  |  |  |
| --- | --- | --- |
| **实体名** | **词条名** | **别名** |
| goods\_name | trade1 | 讯飞翻译机 |
| trade2 | 阿尔法蛋 |
| trade3 | 讯飞录音笔 |
| IFLYTEK.Number | 1-20位数字 支持汉字数字和阿拉伯数字 | |

订单号的语义槽 {order\_number} 为纯数字的数据，我们可以直接调用平台的开放实体【IFLYTEK.number】可以很好的解决。

1. 辅助词

|  |  |  |  |
| --- | --- | --- | --- |
| **辅组词名** | **别名** | **辅组词名** | **别名** |
| want | 我想 | Inquiry | 查询 |
| 我要 | 查下 |
| 帮我 | 问一下 |
| request | 请你 | price | 多少钱 |
| 麻烦你 | 多少价格 |
| 是否可以 | 价格 |
| goods | 东西 | stock | 有没有货 |
| 商品 | 多少货 |
| 物品 | 多少库存 |
| [introduce](http://www.baidu.com/link?url=PUbbZISK1FbR2-vU7OjhWjZLxwzfTHMnfhJNkBBqwxrks09XHmZwS5NrmJ3vFFR4iioChKyS0W49rloKKDid3wC6iczQqf_b4EKRk4b-AYi" \t "https://www.baidu.com/_blank) | 介绍下 | function | 哪些功能 |
| 说说 | 功能有哪些 |
| 讲一下 | 有些什么功能 |

在应用配置页面完成自定义技能“goods\_[consult](http://www.baidu.com/link?url=G49DFIL75p_h1-KQ_b2vNzl6D3horZQbvabKuxIzxkcmzsblvHeud-DMYTlOY3uG_SdtQ7vWzJs3QWlT57ah_LsyvflvwsV5Fx2Qrlg7qrS" \t "https://www.baidu.com/_blank)”和“order\_inquiry”的添加

（6）保存应用配置。

## 步骤3：项目工程搭建及能力包导入

参考“语音识别技术体验\_实验指导书”

## 步骤4：项目AIUI能力导入及配置

参考“语音识别技术体验\_实验指导书”

## 步骤5：功能代码开发

1. AIUI能力集成

在我们的应用工程MainActivity.java文件的onCreate中进行AIUI的能力初始化，并在文件中添加相应的回调方法，具体代码请参考“语音识别技术体验\_实验指导书”；

1. 添加页面布局

添加页面布局文件content\_adapter.xml，用来显示每一条消息信息，代码如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  <TextView  android:id="@+id/msg"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:background="@color/white"  android:layout\_alignParentLeft="true"/>  </RelativeLayout> |

修改页面布局文件activity\_main.xml，需要添加相应的控件来在界面上展示用户进行问题咨询时的问答信息，具体的代码如下:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">  <RelativeLayout  android:id="@+id/txtArea"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  <TextView  android:id="@+id/txtShow"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_centerInParent="true"/>  </RelativeLayout>  <ListView  android:id="@+id/contentArea"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:visibility="gone">  </ListView >  </RelativeLayout> |

1. 应用数据添加和解析

一般在应用开发过程中，应用的业务数据都是存储在后台服务器的数据库中，应用通过接口的方式调用后台服务获取业务数据，在本次实验中，后台服务相关知识不作为我们的实验内容，所以我们使用本地的数据来存储相关业务数据，通过数据解析的方式获取数据。

我们在项目工程目录/app/src/main/assets下面添加一个数据文件，命名为data.cfg，文件代码如下：

|  |
| --- |
| {  "products":  {  "讯飞翻译机":  {  "price":2999,  "stock":85,  "desc":"讯飞翻译机2.0是科大讯飞于2018年4月20日推出的新一代人工智能翻译产品。它采用神经网络机器翻译、语音识别、语音合成、图像识别、离线翻译以及四麦克风阵列等多项人工智能技术，支持语种覆盖近200个国家和地区，支持离线翻译、方言识别、拍照翻译、全球上网；除了4种中文方言的识别 ，讯飞翻译机2.0还能够识别加拿大、英国、澳大利亚、印度、新西兰五个国家的带有口音的英语 。",  "ability":"讯飞翻译机2.0支持的功能包括语音识别、语音合成、图像识别、在线翻译、离线翻译等。"  },  "阿尔法蛋":  {  "price":699,  "stock":150,  "desc":"阿尔法蛋将人工智能与儿童教育深度结合，旨在为每个孩子提供人工智能学习助手。围绕不同年龄段儿童的成长特性，阿尔法蛋设计了科学的内容体系，为孩子提供精选的成长资源。阿尔法蛋依托淘云科技独有的、专为孩子定制的人工智能技术，为孩子提供有趣高效的学习方式，提高其学习兴趣和效率。",  "ability":"阿尔法蛋支持的功能包括语音唤醒、在线点播、百科问答、定时提醒、生活娱乐、学习助手等，很强大哦！"  },  "讯飞录音笔":  {  "price":1500,  "stock":210,  "desc":"讯飞智能录音笔是一款智能录音设备，具有声音的存储、编辑、转写、查看、分享功能。拾音和声音转文字等能力合为一体，录音转为文字呈现在屏幕，并可进行标签添加和编辑。产品适用于演讲、会议、培训、课堂、取证等多场景，提高语音记录和资料整理效率，产品配备指纹加密功能。"  }  },  "orders":  {  "13718137491":  {  "status":"已发货",  "logistics":"到达华东物流转运中心",  "amount":3960  },  "22387789132":  {  "status":"已支付，未发货",  "logistics":"",  "amount":1500  }  }  } |

然后，在MainActivity.java文件中对数据文件进行解析，完成解析后将业务数据存储到全局变量中，具体代码如下：

|  |
| --- |
| */\*\**  *\* 业务数据*  *\*/*  private JSONObject mBusinessData;  */\*\**  *\* 解析业务数据并存储*  *\*/*  private JSONObject loadBusinessData(){  String data\_path = "data.cfg";  AssetManager assetManager = getResources().getAssets();  try {  InputStream ins = assetManager.open(data\_path);  byte[] buffer = new byte[ins.available()];  ins.read(buffer);  ins.close();  String data\_content = new String(buffer);  JSONObject dataObject = new JSONObject(data\_content);  return dataObject;  }  catch (Exception e){  Log.e("MainActivity", "载入业务数据出现异常");  e.printStackTrace();  }  return null;  } |

在onCreate方法中调用该数据解析方法，完成业务数据的初始化，代码如下：

|  |
| --- |
| @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  *//AIUI能力的初始化*  AIUIAbility.getInstance().initAIUIAbility(this);  AIUIAbility.getInstance().addNLPListener(this);  AIUIAbility.getInstance().start();  *//语音合成初始化*  SpeechHelper.getInstance().initSpeech(this);  SpeechHelper.getInstance().setVoicer("xiaoyan");  *//业务数据初始化*  mBusinessData = this.loadBusinessData();  } |

1. 添加功能代码

语义理解的功能实现比较重要的一个环节是语义结果的解析，与之前的实验不同的是，我们不再使用较为繁琐的Json对象转换的方式进行数据解析，而是创建一个专门用于Json数据解析的类，用来配合Gson工具进行Json数据解析，创建一个新的文件SemanticBean.java，具体代码如下：

|  |
| --- |
| public class SemanticBean {  *//用户说的话*  private String text = "";  *//响应码*  private String rc = "-1";  *//指令类型*  private String operation = "";  *//服务ID*  private String service = "";  *//语义结果*  private List<Semantic> semantic;  *//回答 有answer则无semantic*  private Answer answer;  public boolean isValid() {  return true;  }  public void setInputText(String text) {  this.text = text;  }  public String getInputText() {  return text;  }  public String getRc() {  return rc;  }   public String getService() {  return service;  }  public void setService(String ser) {  this.service = ser;  }  public String getOperation() {  return operation;  }  public String getAnswer() {  String str = answer != null ? answer.text : "";  return str;  }  public void setAnswer(Answer answer) {  this.answer = answer;  }  public List<Semantic> getSemantic() {  return semantic;  }  public void setSemantic(List<Semantic> semantic) {  this.semantic = semantic;  }  */\*\* end \*\*/*  */\*\*  \* 语义类  \*/*  public static class Semantic {   private String intent;   private List<Slots> slots;   public List<Slots> getSlots() {  return slots;  }  public void setSlots(List<Slots> slots) {  this.slots = slots;  }  public String getIntent() {  return intent;  }  public void setIntent(String intent) {  this.intent = intent;  }  }  public static class Slots {  public String name;   public String value;  public String normValue;  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getValue() {  return value;  }  public void setValue(String value) {  this.value = value;  }  public String getNormValue() {  return normValue;  }  public void setNormValue(String normValue) {  this. normValue = normValue;  }  }   public static class Location {  public String type;  public String poi;  public String city;  }   */\*\*  \* 回答  \*/*  public static class Answer {  *//回答的文本，jsonObject中其他字段暂时无用*  public String text;  } } |

接下来，我们在MainActivity.java定义一个方法，用来处理商店技能和自定义问答的内容显示，使用页面控件中ListView将问答的对话信息进行展示，同时采用语音的方式进行回答，下面是具体的代码添加流程：

1. 定义全局变量

|  |
| --- |
| */\*\**  *\* 需要展示的文本数据列表*  *\*/*  private List<String[]> dataList=new ArrayList<String[]>();  */\*\**  *\* ListAdapter*  *\*/*  private ListAdapter mListAdapter;  */\*\**  *\* 界面控件*  *\*/*  private ListView mListView;  private RelativeLayout mTextArea;  private TextView mShowTextView;  */\*\**  *\* 延时处理Handler*  *\*/*  private Handler mHandler;  private ArrayList mDataList; |

1. 在OnCreate方法中进行业务数据、UI界面及语音合成服务的初始化操作

|  |
| --- |
| @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  *//AIUI能力的初始化*  AIUIAbility.getInstance().initAIUIAbility(this);  AIUIAbility.getInstance().addNLPListener(this);  AIUIAbility.getInstance().start();  *//初始化语音合成服务*  SpeechHelper.getInstance().initSpeech(this);  *//设置发音人*  SpeechHelper.getInstance().setVoicer("xiaoyan");  *//业务数据初始化*  mDataList = new ArrayList();  mHandler = new Handler();  mBusinessData = this.loadBusinessData();  *//UI界面*  initView();  }  */\*\**  *\* UI界面显示*  *\*/*  private void initView(){  mListView = (ListView)findViewById(R.id.contentArea);  mListAdapter = new ListAdapter(dataList);  mListView.setAdapter((android.widget.ListAdapter) mListAdapter);  mTextArea = (RelativeLayout)findViewById(R.id.txtArea);  mShowTextView = (TextView)findViewById(R.id.txtShow);  mShowTextView.setText("您好，请问有什么可以帮您？");  } |

1. 处理对话信息

|  |
| --- |
| */\*\**  *\* 显示回答文本（处理商店技能和自定义问答技能）*  *\*/*  private void showAnswerText(String inputText, String answerText){  //界面显示文本信息  dataList.add(new String[]{"1", inputText});  dataList.add(new String[]{"2", answerText});  //取消等待处理的事件  mHandler.removeCallbacksAndMessages(null);  if(mListView.getVisibility() == View.GONE){  mListView.setVisibility(View.VISIBLE);  mTextArea.setVisibility(View.GONE);  }  mListAdapter.notifyDataSetChanged();  SpeechHelper.getInstance().speak(answerText);  mHandler.postDelayed(new Runnable() {  @Override  public void run() {  SpeechHelper.getInstance().speak("请问还有什么可以帮您？");  }  }, 15\*1000);  mHandler.postDelayed(new Runnable() {  @Override  public void run() {  mDataList.clear();  mListAdapter.notifyDataSetChanged();  mListView.setVisibility(View.GONE);  mTextArea.setVisibility(View.VISIBLE);  }  }, 20\*1000);  }  */\*\**  *\* ListAdapter*  *\*/*  public class ListAdapter extends BaseAdapter{  List<String[]> mTitles;  public ListAdapter(List<String[]> titles){  this.mTitles = titles;  }  @Override  public int getCount(){  return mTitles.size();  }  @Override  public Object getItem(int position){  return mTitles.get(position);  }  @Override  public long getItemId(int position){  return position;  }  @Override  public View getView(int position, View convertView, ViewGroup parent){  convertView=LayoutInflater.from(getApplicationContext())  .inflate (R.layout.content\_adapter, parent, false);  TextView titleView = convertView.findViewById(R.id.msg);  String type = mTitles.get(position)[0];  String title = mTitles.get(position)[1];  titleView.setText(title);  *//布局默认显示在右边，如果是用户的发言，调整为显示在左边*  if(TextUtils.equals("1", type)){  RelativeLayout.LayoutParams rlp=new RelativeLayout  .LayoutParams(RelativeLayout.LayoutParams.WRAP\_CONTENT,  RelativeLayout.LayoutParams.WRAP\_CONTENT);  rlp.addRule(RelativeLayout.ALIGN\_PARENT\_LEFT);  titleView.setLayoutParams(rlp);  }  return convertView;  }  } |

1. 业务逻辑处理（商品咨询及订单查询等）

我们定义一个新的方法，用来处理自定义技能的相关功能逻辑，包括了用户对商品的咨询和订单查询等语义的处理，具体代码如下：

|  |
| --- |
| */\*\**  *\* 处理自定义技能*  *\*/*  private void handleShopSkill(SemanticBean bean){  //判断服务类型  String service\_tail=bean.getService().substring(0,bean.getService().indexOf("."));  String service=bean.getService().substring(service\_tail.length()+1,bean.getService().length());  switch (service){  case "goods\_consult":*//商品咨询*  handleGoodsConsult(bean);  break;  case "order\_inquiry":*//订单查询*  handleOrderInquery(bean, 1);  break;  default:  break;  }  }  */\*\**  *\* 商品咨询处理*  *\*/*  private void handleGoodsConsult(SemanticBean bean){  *//获取语义结果*  SemanticBean.Semantic semantic = bean.getSemantic().get(0);  SemanticBean.Slots goods\_slot = semantic.getSlots().get(0);  String goods\_name = goods\_slot.getValue();  try {  JSONObject productsObject = mBusinessData.optJSONObject("products");  if (productsObject.has(goods\_name)) {  JSONObject dataObject = productsObject.optJSONObject(goods\_name);  switch (semantic.getIntent()) {  case "describe":  String desc = dataObject.getString("desc");  showAnswerText(bean.getInputText(), desc);  break;  case "price":  String price = dataObject.getString("price");  String price\_desc = goods\_name + "的价格是" + price + "元";  showAnswerText(bean.getInputText(), price\_desc);  break;  case "stock":  String stock = dataObject.getString("stock");  String stock\_desc = goods\_name + "目前的库存还有" + stock + "件";  showAnswerText(bean.getInputText(), stock\_desc);  break;  case "goods":  String goods\_desc = dataObject.getString("desc");  String goods\_ability = dataObject.getString("ability");  showAnswerText(bean.getInputText(), goods\_desc + goods\_ability);  break;  default:  break;  }  } else {  *//语音合成*  SpeechHelper.getInstance().speak("对不起，没有查询到您要咨询的商品信息！");  }  }  catch (Exception e){  Log.e("MainActivity", "处理商品咨询语义出现异常");  e.printStackTrace();  }  }  */\*\**  *\* 订单查询处理*  *\* @param bean 语义结果*  *\*/*  private void handleOrderInquery(SemanticBean bean){  SemanticBean.Semantic semantic = bean.getSemantic().get(0);  List<SemanticBean.Slots> slots = semantic.getSlots();  try{  JSONObject orderObjects = mBusinessData.optJSONObject("orders");  switch (semantic.getIntent()){  case "order\_status":  showAnswerText(bean.getInputText(),  "好的，请告诉我您要查询的订单号！");  break;  case "order":  String order\_id = slots.get(0).getValue();  if(orderObjects.has(order\_id)){  JSONObject dataObject = orderObjects.optJSONObject(order\_id);  if(TextUtils.equals(dataObject.getString("status"), "已发货")){  showAnswerText(bean.getInputText(), "您的订单已经发货，  当前的物流状态为："+dataObject.getString("logistics"));  }  else{  showAnswerText(bean.getInputText(),  "您的订单"+dataObject.getString("status"));  }  }  else{  showAnswerText(bean.getInputText(),  "对不起，没有查询到您的订单信息，请确认订单号是否正确！");  }  break;  }  }  catch (Exception e){  e.printStackTrace();  }  } |

然后，我在AIUI的onAiuiResponse回调方法中进行语义数据解析，将语义指令调用相应的方法进行处理，具体代码如下：

|  |
| --- |
| */\*\**  *\* 解析AIUI结果，处理语义指令*  *\* @param semantic*  *\*/*  @Override  public void onAiuiResponse(String semantic) {  if (TextUtils.isEmpty(semantic)) {  return;  }  try{  JSONObject jsonObject = new JSONObject(semantic);  Gson gson = new Gson();  SemanticBean semanticBean =  (SemanticBean) gson.fromJson(jsonObject.getString("intent"), SemanticBean.class);  if(null == semanticBean){  return;  }  *//处理商店技能和自定义问答，显示回答文本*  if (!TextUtils.isEmpty(semanticBean.getAnswer())) {  showAnswerText(semanticBean.getInputText(), semanticBean.getAnswer());  }  *//处理自定义技能*  else if(!TextUtils.isEmpty(semanticBean.getService())){  handleShopSkill(semanticBean);  }  }  catch (Exception e){  Log.e("MainActivity", "处理语音结果出现异常");  e.printStackTrace();  }  } |

到这里，我们就已经完成所有应用功能代码的开发了，下面，我们将机器人开发平台与开发电脑通过USB进行连接，然后将应用编译到机器人开发平台进行运行，具体的方法参考“服务机器人开发环境搭建\_实验指导书”中编译调试部分。

当应用在机器人开发平台上运行时，我们开始按照以下测试用例对应用功能进行校验，查看运行效果：

|  |  |  |
| --- | --- | --- |
| **测试方式** | **测试步骤** | **期望反馈效果** |
| 启动应用 | 应用启动后，查看应用界面 | 应用正常运行，界面显示文本“请问有什么可以帮您？” |
| 语音输入 | “今天天气怎么样？” | 语音回答天气情况，界面以对话方式显示天气情况文本信息 |
| “来一首李白的静夜思” | 语音阅读诗词“静夜思”，界面显示文本信息 |
| “有人吗？” | 语音回答“在呢，有什么可以为您服务”，界面显示相应文本信息 |
| “你能做什么” | 语音回答“我可以查询商品和订单信息哦”，界面显示相应文本信息 |
| “能介绍下阿尔法蛋吗” | 语音播放商品“阿尔法蛋”的介绍，界面显示相应文本信息 |
| “讯飞录音笔有哪些功能” | 语音播放商品“讯飞录音笔”的功能介绍，界面显示相应文本信息 |
| “讯飞翻译机什么价格？” | 语音回答商品“讯飞翻译机”的价格，界面显示相应文本信息 |
| "讯飞翻译机还有货吗？“ | 语音回答商品”讯飞翻译机“的库存信息，界面显示相应文本信息 |
| "请问我买的翻译机发货了吗?" | 语音提示“好的，请告诉我您要查询的订单号！” |
| “88888888” | 在提示用户输入订单号的状态，查询订单号为“238172394908”的订单信息，如果没有查找到，语音提示“没有查询到您的订单信息，请确认您的订单号是否正确。”；如果查到订单，语音回答订单状态及物流状态 |
| 非提示用户输入订单号的状态，忽略，无效果 |
| “我要查询订单88888888的信息” | 查询订单信息，如果没有查找到，语音提示“没有查询到您的订单信息，请确认您的订单号是否正确。”；如果查到订单，语音回答订单状态及物流状态 |

根据测试用例的步骤对应用进行相应的功能测试，如果都可以得到正确的反馈，那么就说明我们的实验已经达到了我们的预期效果。

# 参考资料与扩展阅读

1. Android开发工具网站：https://www.androiddevtools.cn/
2. Android开发者社区：<https://developer.android.google.cn/guide/platform/>
3. 科大讯飞AIUI开放平台：<https://aiui.xfyun.cn/>
4. 科大讯飞开放平台文档中心：https://www.xfyun.cn/doc/