

# GQY 机器人接口文档

宁波 GQY 视讯股份有限公司

## 郑重声明

本手册内容若有变动，恕不另行通知。未得到宁波 GQY 视讯股份有限公司明确的书面许可，不得为任何目的、以任何形式或手段（电子的或机械的）复制或传播手册的任何部分。本文档可能涉及宁波 GQY 视讯股份有限公司的专利（或正在申请的专利）、商标、版权或其他知识产权，除非得到新世纪机器人有限公司的明确书面许可协议，本文档不授予使用这些专利（或正在申请的专利）、商标、版权或其他知识产权的任何许可协议。本手册提及的其它产品和公司名称均可能是各自所有者的商标。

版权所有©宁波 GQY 视讯股份有限公司

目录

GQY 机器人接口文档..... 1

版本信息..... 4

概述 ..... 4

    目的..... 4

    范围..... 4

GQY 机器人开放资源和开发接口..... 5

    GQY 机器人开放资源..... 5

    GQY 机器人开发接口..... 6

        获得语音识别和语音理解结果接口 ..... 6

        打开或者关闭语音接口..... 7

        访问语音合成接口..... 8

        机器人命令控制..... 10

        导航控制接口(导航到某处， 暂停和继续导航)..... 12

        导航结果接受接口..... 14

        表情与运动控制接口..... 16

        运动控制接口..... 22

        人脸识别接口..... 24

        机器人状态接口..... 27

定制知识库..... 30

定制广告..... 31

定制 VIP 识别 ..... 32

附录 ..... 33

# 版本信息

版本	日期	说明
V1.0	2017-7-4	试用版
V1.1	2017-8-18	校正
V1.2	2017-9-4	校正
V1.3	2017-9-19	增加机器人版本差别
V1.4	2018-3-28	增加关机指令
V1.5	2018-6-8	增加其他功能
V1.6	2018-12-29	增加遥控模式
V1.7	2019-2-19	修复一下已知问题

## 概述

GQY 机器人接口文档是介绍 GQY 机器人对客户开放的资源和接口，指导客户如何使用这些接口以及和二次开发配套的软件和文档说明。

## 目的

本手册目录目的是帮助客户了解和开发 GQY 机器人，方便地使用 GQY 机器人的开放的资源和开发接口。

## 范围

本手册描述 GQY 机器人开发资源和接口，不描述机器人结构和与开放资源及接口无关的机器人功能。

# GQY 机器人开放资源和开发接口

## GQY 机器人开放资源

开放资源	说明	开放额度
语音识别	客户程序可以请求语音识别的结果	无限制
语义理解	客户程序可以请求语义理解的结果；客户可以定制知识库	无限制
语音合成	客户程序可以请求发声	无限制
机器人状态控制	通过命令让机器人处于某种状态或者行为	无限制
导航控制及其结果的接受	客户程序可以请求导航功能和获悉导航结果；结合机器人管理 APP 实现导航到目的地	无限制，机器人管理 APP 的介绍参考附录
表情与动作控制	<ul style="list-style-type: none"><li>● 客户程序可以请求眼睛和嘴部表情</li><li>● 客户程序可以请求单个头部动作和单个手臂动作；也可以请求整套动作</li></ul>	<ul style="list-style-type: none"><li>● 眼睛表情（27 种静态+11 动态），嘴部表情（4 种静态+6 动态），表情图片具体参考<a href="#">表情表格</a></li><li>● 2 个头部动作和 4 个手臂动作</li><li>● 13 套整体动作（卖萌、敬礼、飞吻等）</li></ul>
运动控制	客户程序可以请求运动控制	前进, 左转, 右转, 转一圈
超声波感知	机器人具有感知是否有人靠近和离开的能力，客户程序可以请求该服务	无限制
人脸识别	客户程序可以请求人脸识别的结果	VIP 号, 年龄, 性别, 表情和颜值等属性
Windows 程序运行环境	在机器人核心服务不受影响的情况下，允许客户的 Windows 程序运行在 GQY 机器人的 Windows10 系统中。	有限
（选配）外设访问	客户程序可以请求外设访问如打印机，身份证和银行卡读卡器	

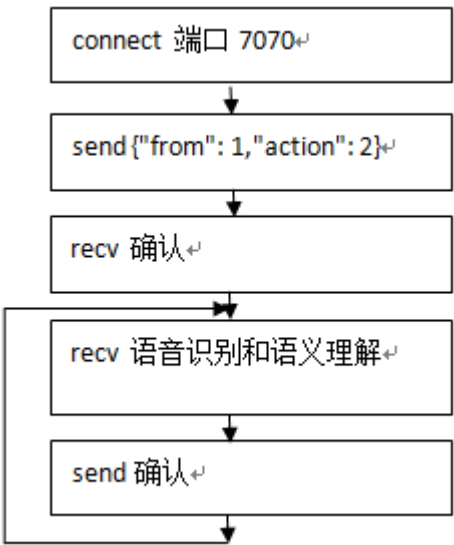
资源表格

# GQY 机器人开发接口

说明：使用该接口需要关闭 CBC 界面程序；中文编码为 utf8。

## 获得语音识别和语音理解结果接口

客户程序通过 socket 访问端口 7070，具体流程图：



握手请求 json:

```
{
  "from":1,
  "action": 2
}
```

语音识别和语音理解结果示例:

```
{
  "question":"你叫什么名字",
  "behavior":0,
  "parameter" : "",
  "content" : "我是 GQY 银行大堂经理小民",
  "context" : "",
  "url" : ""
}
```

返回参数说明

参数	类型	说明
question	string	语音识别的结果(问题)
behavior	int	在此应用中可以忽略
parameter	string	在此应用中可以忽略

content	string	语义理解的结果(答案)
context	string	在此应用中可以忽略
url	string	在此应用中可以忽略

c 语言例程:

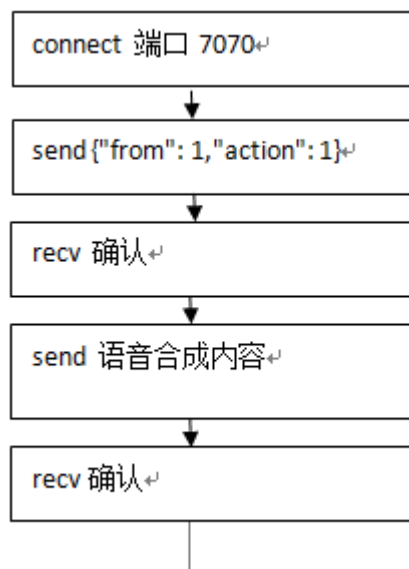
```
const char *req="{\"from\":1,\"action\": 2}";
char buff[1024];
```

```
SOCKET client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
struct sockaddr_in serAddrssound;
serAddrssound.sin_family = AF_INET;
serAddrssound.sin_port = htons(7070);
serAddrssound.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");//机器人 IP
client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
connect(client_socket, (struct sockaddr *)&serAddrssound, sizeof(serAddrssound));
```

```
send(client_socket, req,strlen(req) , 0);
recv(client_socket ,\"ok\",2,0)
while(1)
{
recv(client_socket, buff, 1024, 0);
send(client_socket, \"ok\",2 , 0);
// process the result
}
```

## 打开或者关闭语音接口

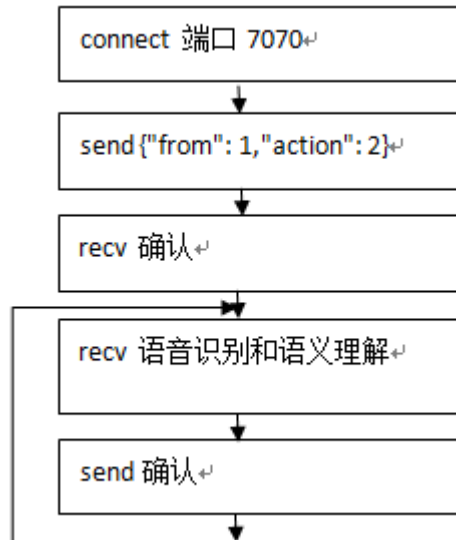
客户程序通过 socket 访问端口 7070，打开主发接口，具体流程图：



握手请求 json:

```
{
  "from":1,
  "action": 1
}
```

同时打开主收接口：



握手请求 json:

```
{
  "from":1,
  "action": 2
}
```

只听模式打开

通过主发发送命令：

```
{
  "cmd":36,
  "subcmd":1,
  "content":""
}
```

只听模式关闭

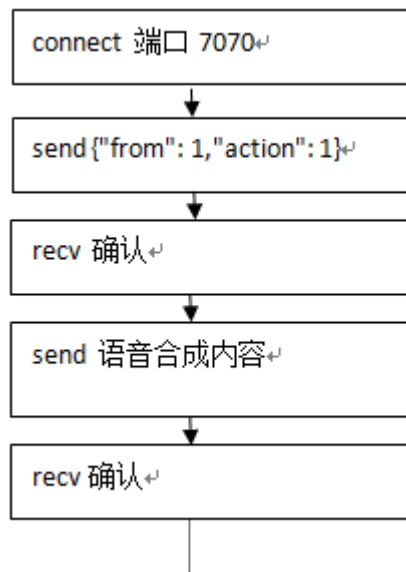
通过主发发送命令：

```
{
  "cmd":37,
  "subcmd":0,
  "content":""
}
```

## 访问语音合成接口

客户程序通过 socket 访问端口 7070，具体流程图：





握手请求 json:

```

{
  "from":1,
  "action": 1
}

```

语音合成内容示例:

```

{
  "cmd":0,          //语音合成时 可选
  "subcmd":0,       //语音合成时 可选
  "content":"你好，欢迎来到 GQY 公司"
}

```

语音合成请求参数说明

参数	类型	说明
content	string	发声内容（utf-8 格式）（为空时打断当前说话，而不是 Null）

c 语言例程:

```

const char *req="{\"from\":1,\"action\": 1}";
char buff[1024];

```

```

SOCKET client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
struct sockaddr_in serAddrsound;
serAddrsound.sin_family = AF_INET;
serAddrsound.sin_port = htons(7070);
serAddrsound.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");//填写机器人 IP
client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
connect(client_socket, (struct sockaddr *)&serAddrsound, sizeof(serAddrsound));

```

```

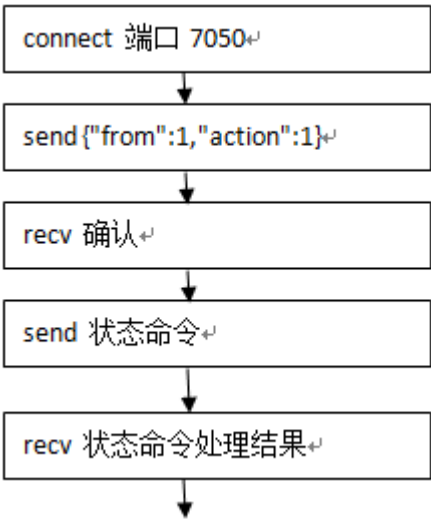
send(client_socket, req,strlen(req) , 0);

```

```
recv(client_socket ,buff ,1024,0)//接受确认
const char *ttsB="{\"content\": \"你好， 欢迎来到 GQY 公司\\\"}\" //xxx 是发声的 utf8
send(client_socket, ttsB,strlen(ttsB) , 0);
recv(client_socket, buff, 1024, 0);//接受确认。
```

机器人命令控制

客户程序通过 socket 访问端口 7050， 具体流程：



握手请求 json:

```
{
  "from":1,
  "action": 1
}
```

状态命令示例:

```
{
  "from": 1,
  "cmd":xxx,
  "subcmd":yyy
}
```

状态命令参数说明

参数	类型	说明
from	int	固定为 1 或者 4
cmd	int	命令 xxx: 命令详细看下面列表
subcmd	int	参数

命令定义:

命令	值	参数	备注
----	---	----	----

休眠模式	64	无	
空闲模式	65	无	
跳舞模式	66	1/2	舞曲 1，或者舞曲 2
唱歌模式	67	无	
跟随模式	68	无	
语音命令	69	参数	详见下表
遥控模式	71	参数	详见下表
导航模式	73	目标点	
关机模式	75	无	
行走模式	101	-128~127	每个值代表 5cm，正值向前，负值向后

语音命令参数列表

功能	参数
左转	1
右转	2
转圈	3
过来	4
再转一点	5
敬礼	6
卖萌	7
握手	8
飞吻	9
摆 post	10
欢迎	11
拥抱	12
得瑟	14
摇头	17
Byebye	22

遥控模式参数表格：

前进 X	转向 Y
范围： -128~127	范围： -128~127

参数=X<<8+Y;

关机命令处理结果示例：

```
{
  "from": 3,
  "cmd": 75,
  "subcmd": 0,
  "resp": 1
}
```

### 关机命令处理结果参数说明

参数	类型	说明
from	int	固定为 3
cmd	int	关机命令 75
subcmd	int	0，无意义，保留
resp	int	1 表示接受状态命令， 2 表示拒绝状态

C 语言例程：

```
const char *req="{\"from\":1,\"action\": 1}";  
char buff[1024];
```

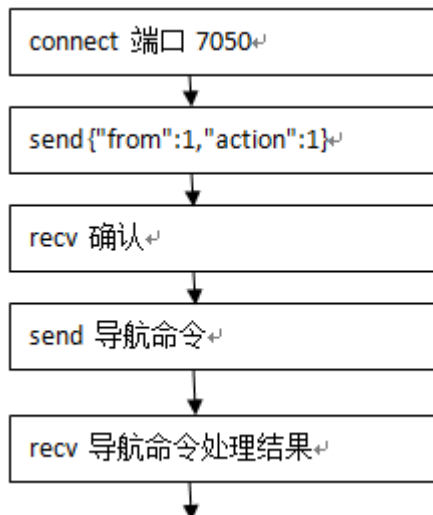
```
SOCKET client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);  
struct sockaddr_in serAddrsound;  
serAddrsound.sin_family = AF_INET;  
serAddrsound.sin_port = htons(7050);  
serAddrsound.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");//填写机器人 IP  
client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);  
connect(client_socket, (struct sockaddr *)&serAddrsound, sizeof(serAddrsound));
```

```
send(client_socket, req,strlen(req) , 0);  
recv(client_socket ,buff,1024,0);//接受确认
```

```
const char *n1="{\"from\": 1,\"cmd\":75,\"subcmd\":0};//关闭机器人  
send(client_socket, n1,strlen(n1) , 0);  
recv(client_socket, buff, 1024, 0);//接受状态命令处理结果， 查看 resp 键值是接受还是拒绝。
```

## 导航控制接口(导航到某处， 暂停和继续导航)

客户程序通过 socket 访问端口 7050， 具体流程：



握手请求 json:

```

{
  "from":1,
  "action": 1
}

```

导航命令示例:

```

{
  "from": 1,
  "cmd":73,
  "subcmd":10
}

```

导航命令参数说明

参数	类型	说明
from	int	固定为 1 或者 4
cmd	int	导航命令，固定为 73
subcmd	int	<ul style="list-style-type: none"> <li>0-20 机器人导航到达位置号,使用机器人管理 APP 设置;</li> <li>255，暂停导航(进入了导航模式有效，即发过 0-20 导航点命令，且没有到达预定位置)</li> <li>254，继续导航(进入了导航模式有效)</li> </ul>

导航命令处理结果示例:

```

{
  "from": 3,
  "cmd":73,
  "subcmd":10,
  "resp":1
}

```

导航命令处理结果参数说明

参数	类型	说明
----	----	----

from	int	固定为 3
cmd	int	导航命令，固定为 73
subcmd	int	导航命令下发的导航点或者 254, 255
resp	int	1 表示接受导航， 2 表示拒绝导航

C 语言例程：

```
const char *req="{\"from\":1,\"action\": 1}";
char buff[1024];
```

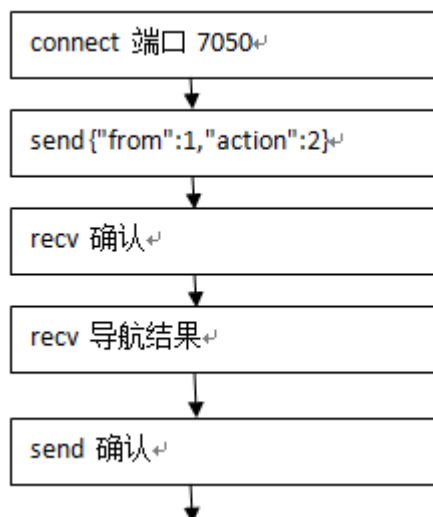
```
SOCKET client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
struct sockaddr_in serAddrssound;
serAddrssound.sin_family = AF_INET;
serAddrssound.sin_port = htons(7050);
serAddrssound.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");//填写机器人 IP
client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
connect(client_socket, (struct sockaddr *)&serAddrssound, sizeof(serAddrssound));
```

```
send(client_socket, req,strlen(req) , 0);
recv(client_socket ,buff,1024,0);//接受确认
```

```
const char *n1="{\"from\": 1,\"cmd\":73,\"subcmd\":1};//导航到一号地点
send(client_socket, n1,strlen(n1) , 0);
recv(client_socket, buff, 1024, 0);//接受导航命令处理结果， 查看 resp 键值是接受还是拒绝
```

## 导航结果接受接口

客户程序通过 socket 访问端口 7050，具体流程：



握手请求 json:

```
{
  "from":1,
  "action": 2
}
```

导航结果示例:

```
{
  "from": 3,
  "cmd":73,
  "subcmd":10,
  "resp":3
}
```

导航结果参数说明:

参数	类型	说明
from	int	固定为 3
cmd	int	导航命令，固定为 73
subcmd	int	导航命令下发的导航点 (0-20)或者 0xff/0xfe
resp	int	1 表示导航正在进行 2 表示导航失败 3 表示导航成功 4 表示导航临时遇到障碍

c 语言例程:

```
const char *req="{\"from\":1,\"action\": 2}";
const char *n1="ok";
char buff[1024];
```

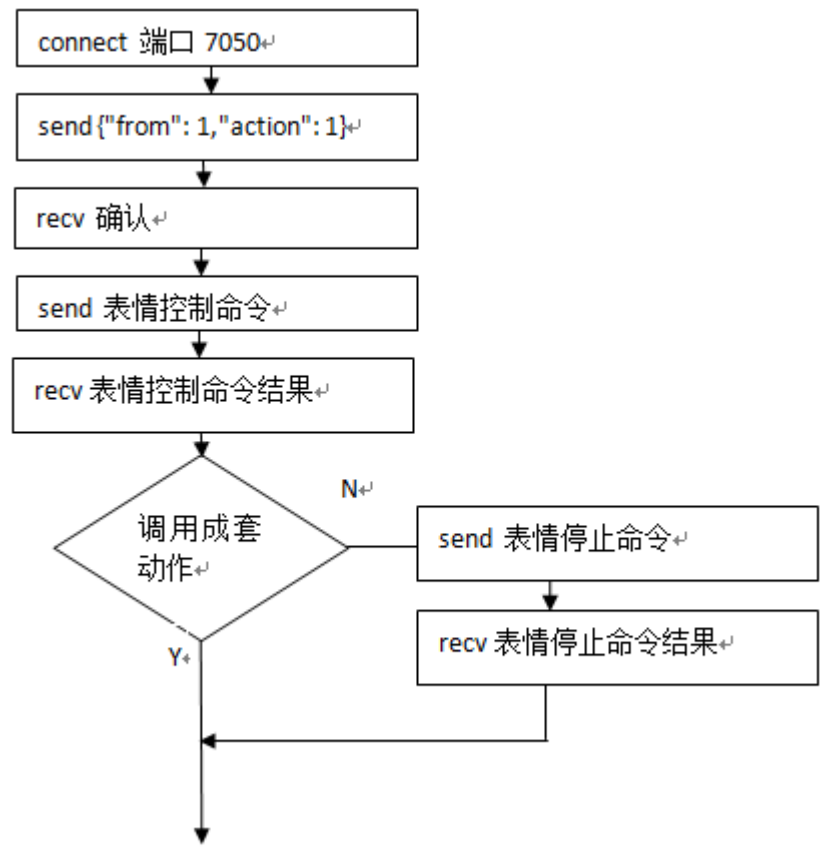
```
SOCKET client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
struct sockaddr_in serAddrsound;
serAddrsound.sin_family = AF_INET;
serAddrsound.sin_port = htons(7050);
serAddrsound.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");//填写机器人 IP
client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
connect(client_socket, (struct sockaddr *)&serAddrsound, sizeof(serAddrsound));
```

```
send(client_socket, req,strlen(req) , 0);
recv(client_socket ,buff,1024,0);//接受确认
```

```
recv(client_socket, buff, 1024, 0);//接受导航结果，查看
send(client_socket, n1,strlen(n1) , 0);
```

# 表情与运动控制接口

客户程序通过 socket 访问端口 7050，具体流程：



握手请求 json:

```
{
  "from":1,
  "action": 1
}
```

表情控制命令示例:

```
{
  "from":1,
  "cmd":102,
  "subcmd":1792
}
```

表情控制命令参数说明:

参数	类型	说明
from	int	固定为 1
cmd	int	动作命令，固定为 102
subcmd	int	表情或者动作号



表情控制命令结果示例：

```
{  
  "from": 1,  
  "cmd":102,  
  "subcmd":1792,  
  "resp":1  
}
```

表情控制命令结果参数说明：

参数	类型	说明
from	int	固定为 1
cmd	int	动作命令，固定为 102
subcmd	int	表情或者动作号
resp	int	1 表示接受动作指令 2 表示拒绝动作指令

表情停止命令示例：

```
{  
  "from": 1,  
  "cmd":65,  
  "subcmd":0  
}
```

表情控制命令参数说明：

参数	类型	说明
from	int	固定为 1
cmd	int	动作命令，固定为 65
subcmd	int	值为 0，忽略它

表情停止命令结果示例：

```
{  
  "from": 1,  
  "cmd":65,  
  "subcmd":0,  
  "resp":1  
}
```

表情停止命令结果参数说明：

参数	类型	说明
from	int	固定为 1
cmd	int	动作命令，固定为 65
subcmd	int	值为 0，忽略它
resp	int	1 表示接受动作指令 2 表示拒绝动作指令

表情或者动作号的取值格式为三个字节组成的整数如下图：









高	中	低
---	---	---















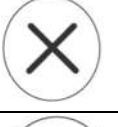

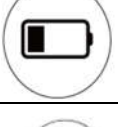

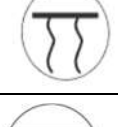
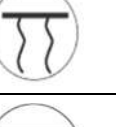




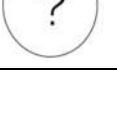

每个字节的取值范围：

**低字节：**1-12，可以调用 GQY 定制好的整套表情和动作；和其他字节是互斥，如果该字节为非 0，其他字节必须为 0。

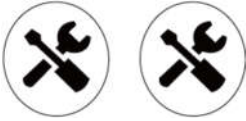
低字节值	动作意义
1	指示平板
2	解答问题
3	卖萌 1
4	卖萌 2
5	迎宾
6	大屏介绍
7	请
8	舞蹈 1
9	握手
10	敬礼
11	摆 pose
12	舞蹈 2

**中字节：**1-46，可以调用眼睛表情；该字节可以和高字节做或运算，实现眼睛和嘴巴动作组合。

中字节值	表达意思	眼睛形状
0	welcome	 
1	桃心	 
2	三角	 
3	微笑	 



4	失落		
5	眼睫毛		
6	圆圈		
7	横线		
8	Hi		
9	Z		
10	波浪		
11	叉		
12	电池没电		
13	哭泣		
14	嚎啕大哭		
15	闪电		
16	问号		

17	向右看	
18	向左看	
19	眩晕	
20	音乐 1	
21	音乐 2	
22	感叹号	
23	充电	
24	心电图	
25	警示符	
26	byebye	
27	Hi 动画	
28	音乐条动画	
29	心电图动画	
30	波浪线动画	
31	哭泣动画	
32	向左看动画	
33	向右看动画	
34	桃心动画	
35	byebye 动画	
36	睡觉动画	
37	思考动画	
38	程序更新动画	
39	眨眼睛一次动画	

40	眨眼睛两次动画	
41	眨眼睛两次后熄灭动画	
42	清屏	
43	故障	
44	电量不足动画	
45	充电动画	

表情表格

高字节：1-14，可以调用嘴巴表情：

高字节值	表达意思	嘴巴形状
0	微笑	
1	失落	
2	口型	
3	横线	
4	清屏	
5	嘴部说话 1	
6	嘴部说话 2	
7	嘴部说话 3	
8	嘴部说话 4	
9	嘴部说话 5	
10	嘴部说话 6	
11	嘴部说话 7	
12	嘴部说话 8	

C 语言例程：  
让眼睛显示圈圈：  
const char \*req="{\"from\":1,\"action\": 1}”;  
char buff[1024];

```

SOCKET client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
struct sockaddr_in serAddrsound;
serAddrsound.sin_family = AF_INET;
serAddrsound.sin_port = htons(7050);
serAddrsound.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");//填写机器人 IP
client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
connect(client_socket, (struct sockaddr *)&serAddrsound, sizeof(serAddrsound));

send(client_socket, req, strlen(req), 0);
recv(client_socket, "ok", 2, 0)//接受确认

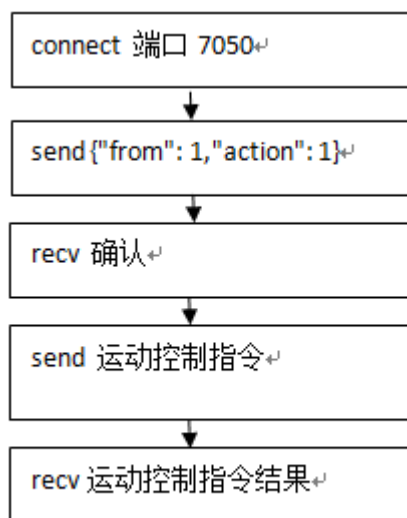
send(client_socket, req, strlen(req), 0);
recv(client_socket, buff, 1024, 0);//接受确认

const char *n1="{\"from\": 1,\"cmd\":102,\"subcmd\":1792}"; //1792 的 16 进制为 0x700
send(client_socket, n1, strlen(n1), 0); //动作控制命令
recv(client_socket, buff, 1024, 0);//接受表情控制命令结果，查看 resp 键值是接受还是拒绝
const char *n2="{\"from\": 1,\"cmd\":65,\"subcmd\":}"; //停止动作命令
send(client_socket, n2, strlen(n2), 0);
recv(client_socket, buff, 1024, 0); //停止动作命令结果

```

## 运动控制接口

客户程序通过 **socket** 访问端口 **7050**，可以控制机器人前进，左转，右转以及转一圈。具体流程：



握手请求 json:

```
{
```

```
"from":1,  
"action": 1  
}
```

运动控制指令示例：

```
{  
  "from":1,  
  "cmd":69,  
  "subcmd":1  
}
```

表情控制命令参数说明：

参数	类型	说明
from	int	固定为 1 或者 4
cmd	int	运动控制命令, 固定为 69
subcmd	int	1 表示左转 2 表示右转 3 表示转圈 4 表示前进

运动控制指令结果示例：

```
{  
  "from":1,  
  "cmd":69,  
  "subcmd":1,  
  "resp":1  
}
```

运动控制指令结果参数说明：

参数	类型	说明
from	int	固定为 3
cmd	int	运动控制命令, 固定为 69
subcmd	int	1 表示左转 2 表示右转 3 表示转圈 4 表示前进
resp	int	1 表示接受 2 表示拒绝

C 语言例程：

```
const char *req="{\"from\":1,\"action\": 1}";  
char buff[1024];
```

```
SOCKET client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);  
struct sockaddr_in serAddrssound;  
serAddrssound.sin_family = AF_INET;
```

```

serAddrsound.sin_port = htons(7050);
serAddrsound.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");//填写机器人 IP
client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
connect(client_socket, (struct sockaddr *)&serAddrsound, sizeof(serAddrsound));

send(client_socket, req, strlen(req), 0);
recv(client_socket, buff, 1024, 0);//接受确认

const char *n1="{\"from\": 1, \"cmd\":69, \"subcmd\":1}";//左转
send(client_socket, n1, strlen(n1), 0);
recv(client_socket, buff, 1024, 0);//接受左转命令结果，查看 resp 键值是接受还是拒绝

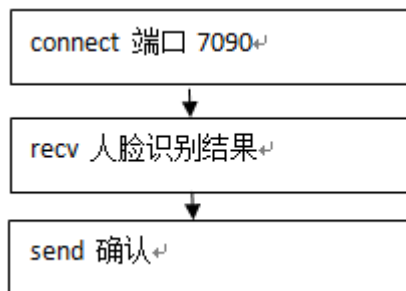
```

## 人脸识别接口

人脸识别接口在 **A2** 和 **A3** 接口是不同的。请认清机器人版本，选择对应的接口。

### A2 机器人：

客户程序通过 socket 访问端口 7090，可以获得人脸识别的结果。具体流程：



人脸识别结果示例：

```

{
  "id": 300011111,
  "gender": 1,
  "emotion": 1,
  "age": 28,
  "attr": 96
}

```

人脸识别结果参数说明：

参数	类型	说明
id	int	VIP 号码
gender	int	性别，0 表示女，1 表示男，2 表示不确定
emotion	int	0 表示愤怒



		1 表示平静 2 表示困惑 3 表示厌恶 4 表示高兴 5 表示悲伤 6 表示惊恐 7 表示诧异 8 表示斜视 9 表示尖叫
age	int	年龄， 0-100
attr	int	颜值， 0-100

c 语言例程：

```

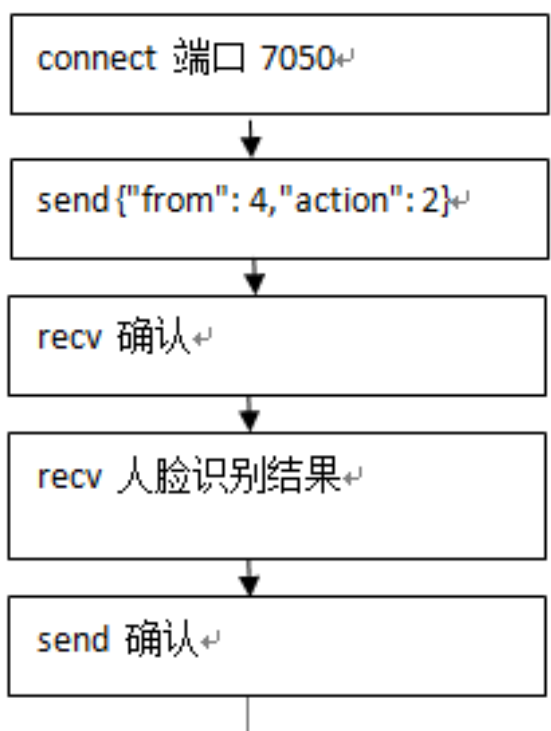
char buff[1024];
SOCKET client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
struct sockaddr_in serAddrsound;
serAddrsound.sin_family = AF_INET;
serAddrsound.sin_port = htons(7090);
serAddrsound.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");//填写机器人 IP
client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
connect(client_socket, (struct sockaddr *)&serAddrsound, sizeof(serAddrsound));

while(1)
{
    recv(client_socket ,buff,1024,0);//接受人脸识别结果
    send(client_socket, "ok",2, 0);//发送确认
    //处理人脸识别结果
}

```

### A3 机器人：

客户程序通过 socket 访问端口 7050，可以获得人脸识别的结果。具体流程：



人脸识别结果示例：

```
{
  "id": 300011111,
  "gender":99,
  "age":26,
  "emotion_t":4,
  "mouth_open":
}
```

人脸识别结果参数说明：

参数	类型	说明
id	int	VIP 号码
gender	int	性别为男性的百分比置信度(0-100)
emotion_t	int	0 表示愤怒 1 表示平静 2 表示困惑 3 表示厌恶 4 表示高兴 5 表示悲伤 6 表示惊恐 7 表示诧异 8 表示斜视 9 表示尖叫
age	int	年龄， 0-100
attr	int	颜值， 0-100

```

char buff[1024];
const char *req="{\"from\":4,\"action\": 2}";

SOCKET client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
struct sockaddr_in serAddrsound;
serAddrsound.sin_family = AF_INET;
serAddrsound.sin_port = htons(7050);
serAddrsound.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");//填写机器人 IP
client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
connect(client_socket, (struct sockaddr *)&serAddrsound, sizeof(serAddrsound));

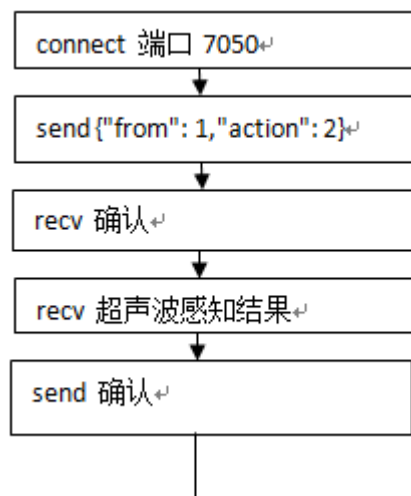
send(client_socket, req,strlen(req) , 0);
recv(client_socket ,buff,1024,0);//接受确认

while(1)
{
    recv(client_socket ,buff,1024,0);//接受人脸识别结果
    send(client_socket, "ok",2, 0);//发送确认
    //处理人脸识别结果
}

```

## 机器人状态接口

客户程序通过 **socket** 访问端口 **7050**，感知机器人前面有人进入或者离开。具体流程：



请求 json:

```
{
```

```
"from":1,
"action",2
}
```

超声波感知结果示例：

```
{
  "from":3,
  "cmd":80,
  "subcmd":systemStatus,
  "errcode":systemError,
  "workmode":workMode,
  "resp":1
}
```

systemStatus 定义

系统 状态	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Status	电机8是否OK	电机7是否OK	电机6是否OK	电机5是否OK	电机4是否OK	电机3是否OK	电机2是否OK	电机1是否OK
	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
	未用	未用	急停开关	未用	未用	人体识别	建闸标记	头部唤醒信号
	Bit23	Bit22~Bit21		Bit20	Bit19	Bit18	Bit17	Bit16
	充电中标记	00:人脸识别处于空闲;10:识别到普通用户;11:识别到VIP用户		说话等待标记	语音说话标记	打印机未连接	未用	低电量
Electricity	Bit31~24							
	电量							

systemError 定义

系统故障	Bit	Bit7	Bit6	Bit5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
error	故障状态L	底盘CAN通讯异常	超声波故障	传感器故障	导航模块故障	驱动电机故障	底盘电机故障	充电故障	电池故障
	Bit	Bit15	Bit14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	故障状态H	工控机CAN通讯异常	人脸识别异常	摄像头打开异常	服务器连接异常	未用	超声波人体识别异常	LED板CAN通讯异常	胸部CAN通讯异常

workMode list

模式	值
休眠模式	64
空闲模式	65
跳舞模式	66
唱歌模式	67
跟随模式	68
语音命令	69
导航模式	73

充电模式	74
关机模式	75
行走模式	101

c 语言例程:

```
char buff[1024];
```

```
const char *req="{\"from\":1,\"action\": 2}";
```

```
SOCKET client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
```

```
struct sockaddr_in serAddrsound;
```

```
serAddrsound.sin_family = AF_INET;
```

```
serAddrsound.sin_port = htons(7050);
```

```
serAddrsound.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");//填写机器人 IP
```

```
client_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
```

```
connect(client_socket, (struct sockaddr *)&serAddrsound, sizeof(serAddrsound));
```

```
send(client_socket, req,strlen(req) , 0);
```

```
recv(client_socket ,\"ok\",2,0)//接受确认
```

```
recv(client_socket ,buff,1024,0);//接受结果
```

```
send(client_socket, \"ok\",2, 0);//发送确认
```

# 定制知识库

GQY 机器人提供了便捷的定制语义理解知识库方法，具体步骤如下：

1. 在 U 盘中，建立目录：新知识库
2. 在客户知识库目录下创建 **exec** 表格文件，文件名如：**1.xls**，文件夹中可以多个 **exec** 表格文件;文件名不能是中文。
3. 每个 **exec** 文件有两列：问题和答案，表必须保留前两行表头内容，从第三行开始填入自己定义的问题和答案。
4. 在 **name** 列填写自定义问题，在对应的 **content** 列填写自定义答案，多个问题或答案可用“|”分隔，例如 **name** 填“今天天气|天气怎么样|天气”，对应 **content** 填“天气晴|晴空万里|好天气”，每一句句首句尾均不加标点；

在 GitHub 中可以下载完整 excel 表格文件示例，参考附录 1。

5. 参考附录 1 中的客户知识库目录。
6. 插入 U 盘到机器人充电座后盖的 USB 接口，如下图红圈标示处：

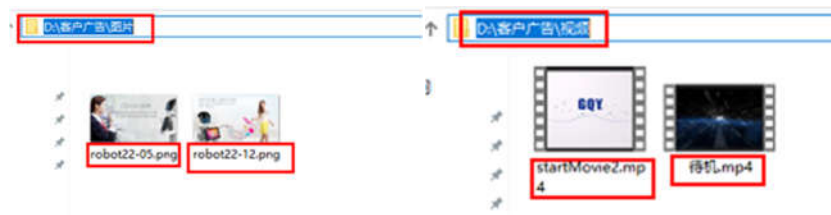


7. 重启机器人，完成客户知识库的定制

# 定制广告

GQY 机器人提供了便捷的定制客户广告方法，具体步骤如下：

1. 在 U 盘中，建立目录：客户广告\图片，客户广告\视频
2. 客户广告\图片目录下保存客户的广告图片，格式为 png，分辨率推荐为 1600\*1000
3. 客户广告\视频目录下保存客户的广告视频，格式为 mp4
4. 如下图：



5. 插入 U 盘到机器人充电座后盖的 USB 接口，如下图红圈标示处：



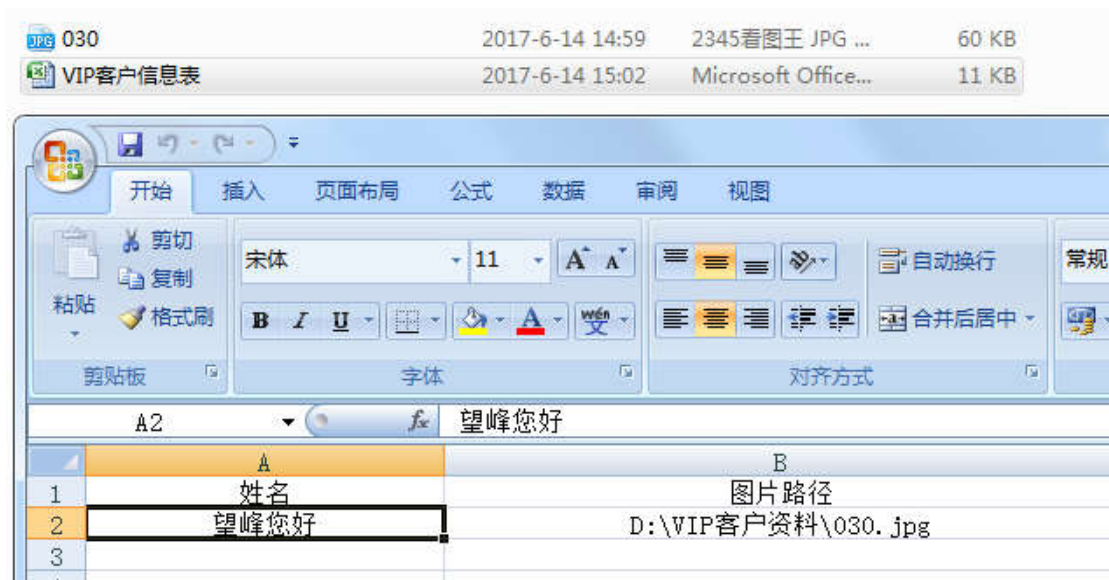
7. 重起机器人，完成客户广告的定制。

# 定制 VIP 识别

GQY 机器人提供了便捷的定制客户 VIP 识别，具体步骤如下：

更换主题和增加 VIP 数据的方法。具体的使用方法如下：

1. 在 U 盘中，建立目录：VIP 客户资料。
2. 在目录中保存 VIP 图片,格式为 jpg，分辨率为 1280\*960 左右，大小为 500K 左右，单个人脸的正面照。
3. 创建 VIP 客户信息表.xlsx 文件，建立照片和称呼的对应。
4. 示例: VIP 客户资料目录下的 030.jpg 照片和对应的称呼文档。



- 4.插入 U 盘到机器人充电座后盖的 USB 接口，如下图红圈标示处：



- 7.打开 VIP 录入程序，完成 VIP 识别定制。



# 附录

该附录包括目录和文件：

1. 外设：包括了外设的帮助文档和头文件
  2. 客户知识库：包括了 excel 文件，展示如何编写知识库
  3. 机器人管理 app:介绍机器人管理 APP 功能和使用方法
- 下载地址：<https://github.com/NingBoGQY/userguide>