1. Voice control of robotic arm movements

Command word table

command word	command word
Clamp	Up
Release	Down
Applause	Turn left
Nod	Right
Pray	Kneel
Fright	Initial position
The car moves forward	The car moves backward
Car turns left	Car turns right
The car rotates left	The car rotates right
Car stops	Car sleeps
Red light on	Green light on
bright blue light	bright yellow light
Turn on the running light	Turn on the gradient light
Turn on the breathing light	Display battery level
Buzzer alarm	

1.1. Function description

By interacting with the voice recognition module on the X3 Plus, you can not only achieve the basic control of the car and light strips in 14.3, but also control the movements of the robotic arm, including up, down, left, and right states, and some settings. Action groups, such as the robot arm "dancing", the robot arm "nodding", etc.

1.2. Start

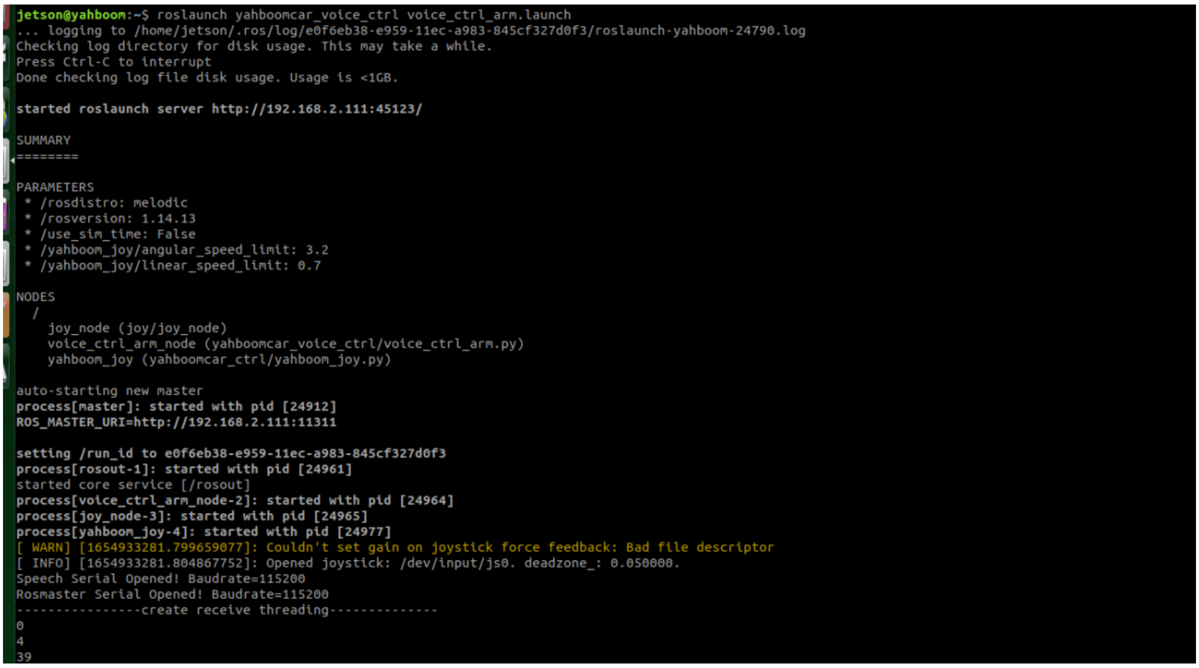
```
#Raspberry Pi 5 master needs to enter docker first, please perform this step
#If running the script into docker fails, please refer to ROS/07, Docker tutorial
~/run_docker.sh
```

1.2.1. Function package path

```
~/yahboomcar_ws/src/yahboomcar_voice_ctrl/
```

1.2.2. Start

```
roslaunch yahboomcar_voice_ctrl voice_ctrl_arm.launch
```



```
jetson@yahboom:~$ roslaunch yahboomcar_voice_ctrl voice_ctrl_arm.launch
... logging to /home/jetson/.ros/log/e0f6eb38-e959-11ec-a983-845cf327d0f3/roslaunch-yahboom-24790.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.2.111:45123/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.13
 * /use_sim_time: False
 * /yahboom_joy/angular_speed_limit: 3.2
 * /yahboom_joy/linear_speed_limit: 0.7

NODES
  /
    joy_node (joy/joy_node)
    voice_ctrl_arm_node (yahboomcar_voice_ctrl/voice_ctrl_arm.py)
    yahboom_joy (yahboomcar_ctrl/yahboom_joy.py)

auto-starting new master
process[master]: started with pid [24912]
ROS_MASTER_URI=http://192.168.2.111:11311

setting /run_id to e0f6eb38-e959-11ec-a983-845cf327d0f3
process[rosout-1]: started with pid [24961]
started core service [/rosout]
process[voice_ctrl_arm_node-2]: started with pid [24964]
process[joy_node-3]: started with pid [24965]
process[yahboom_joy-4]: started with pid [24977]
[ WARN] [1654933281.799659077]: Couldn't set gain on joystick force feedback: Bad file descriptor
[ INFO] [1654933281.804867752]: Opened joystick: /dev/input/js0. deadzone_: 0.050000.
Speech Serial Opened! Baudrate=115200
Rosmaster Serial Opened! Baudrate=115200
----------------create receive threading--------------
0
4
39
```

1.2.3. Core file voice_ctrl_arm.py

- code path

```
~/yahboomcar/src/yahboomcar_voice_ctrl/scripts
```

- Core code analysis:

  1). Import related library files

```
from Speech_Lib import Speech
from voice_arm_library import *
from Rosmaster_Lib import Rosmaster
```

Speech_Lib: Speech module library, reference path: ~/software/py_install_V0.0.1/py_install/Speech_Lib

voice_arm_library: Robot arm action library, reference path: ~/yahboomcar/src/yahboomcar_voice_ctrl/scripts

Rosmaster_Lib: underlying driver library, reference path: ~/software/py_install/Rosmaster_Lib

2. Create speech recognition objects, drive control objects and robotic arm action objects

```
spe = Speech()
self.car = Rosmaster()
voice_arm = Voice_Arm()
```

3). Main function: Recognize the voice, and execute the corresponding program based on the recognized voice. Take the robot arm to return to the initial position as an example.

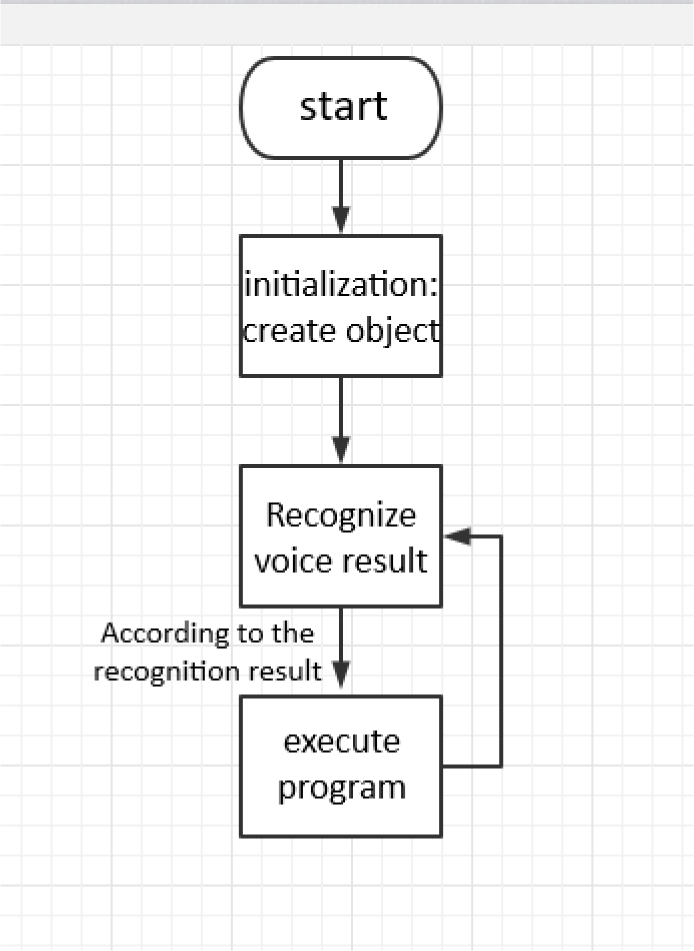
```
speech_r = spe.speech_read()
            if speech_r!=999:
                print(speech_r)
            #print(speech_r)
            if speech_r == 49 :
             spe.void_write(45)
             voice_arm.init_pose()
```

Among them, voice_arm.init_pose() is the program that needs to be executed. At this time, it will jump to the voice_arm_library library and execute the function init_pose() inside. In the init_pose() function, what programs are executed? It is explained below,

```
def init_pose(self):
self.arm_joint.joints =[90.0, 145.0, 0.0, 0.0, 90.0, 31.0]
self.pubPoint.publish(self.arm_joint)
```

Here we define the angle that each joint needs to execute, and then use TargetAngle as the topic to publish the data; then return to the main function and subscribe to the TargetAngle topic; receive the data, enter the callback function, and then pass the self.car.set_uart_servo_angle function Send it to the bottom layer to drive the steering gear.

1.3. Program flow chart


start
initialization:
create object
Recognize
voice result
According to the
recognition result
execute
program