Yolov5 Garbage Sorting with Voice Broadcast (Jetson Nano)

This course is exclusive to Jetson-Nano board users. Orin board users please refer to the course content in [12.2 Garbage Sorting with Voice Broadcast (Jetson Orin)].

Before running the function, you need to close the App and large programs. For the closing method, refer to [4.Preparation] - [1. Manage APP control services].

1. Function Description

Control garbage sorting through voice commands and broadcast the name and type of the currently sorted garbage through voice.

2. Startup and Operation

2.1. Startup

This case runs on the host machine.

First, you need to start roscore. Open the terminal and enter the following:

```
roscore
```

Then you need to start the ROS node service. Open the terminal and enter the following:

```
rosrun dofbot_pro_info kinemarics_dofbot_pro
```

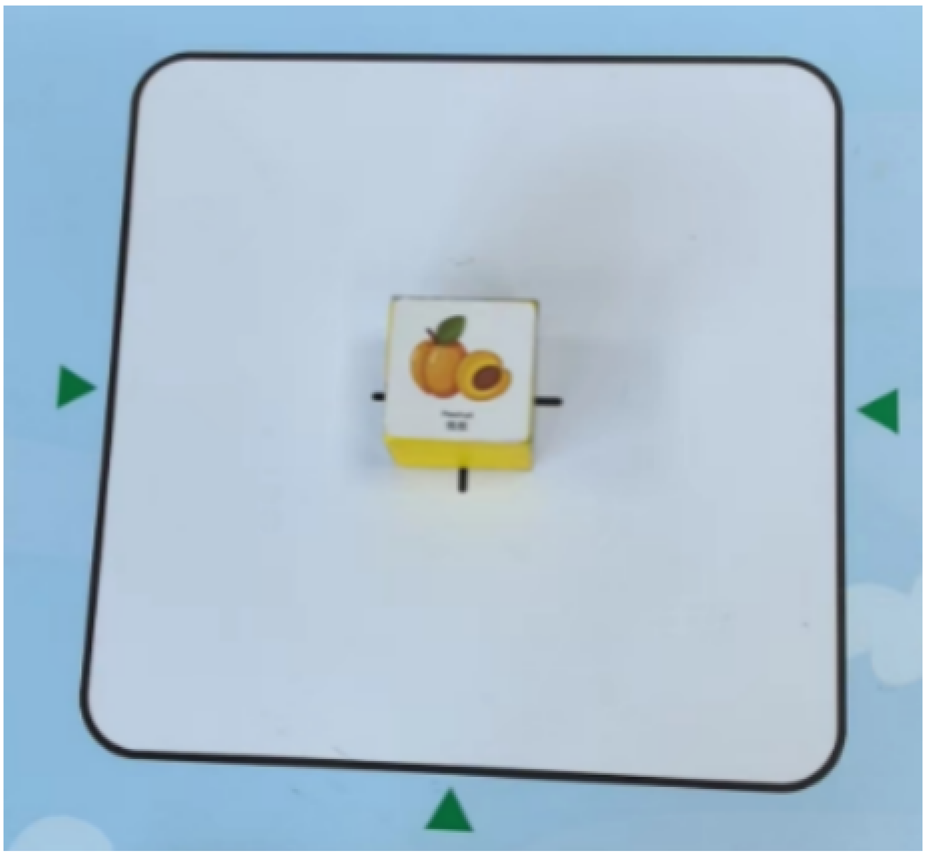
Open a third terminal and enter the following command:

```
python3 ~/dofbot_voice/scripts/garbage_sorting_broadcast.py
```

The Jetson-Nano board startup will be relatively slow. Please wait patiently until the [Model-Loading] on the image screen disappears, indicating initialization is complete.

2.2. Operation Steps

Place the garbage label code block in the center of the image, then say "Hello, yahboom" to the voice module. The voice module will reply "here" to indicate successful wake-up. Then say "What garbage is this?" to the voice module. After the program recognizes it, the voice module will reply and broadcast what this garbage is and its type, and the robotic arm will grab this block and place it in the designated position according to its garbage type. Taking the figure below as an example, according to the recognition result, it will reply "This is a peach pit, which belongs to dry garbage". Then the robotic arm will lower its gripper to grab it and place it in the "dry garbage" area marked on the map.



3. Core Code Analysis

`Source code path: ~/dofbot_voice/scripts/garbage_sorting_broadcast.py`

```
#Import voice broadcast garbage type library, this library is located at
/home/jetson/dofbot_pro/dofbot_garbage_yolov5/speech_garbage_identify.py
from speech_garbage_identify import speech_garbage_identify

#Call the garbage recognition function, the input parameter is the current image,
then return the recognized processed image and recognition results
self.frame, msg = self.garbage_identify.garbage_run(self.frame)

#Traverse the recognition results msg, determine the number self.garbage_num and
type self.garbage_class represented by the garbage name according to the value of
name
for key, pos in msg.items(): name = key
if name == "Zip_top_can":           (self.garbage_num, self.garbage_class) =
('00', '01')
if name == "Old_school_bag":        (self.garbage_num, self.garbage_class) =
('01', '01')
if name == "Newspaper":             (self.garbage_num, self.garbage_class) =
('02', '01')
if name == "Book":                  (self.garbage_num, self.garbage_class) =
('03', '01')
if name == "Toilet_paper":          (self.garbage_num, self.garbage_class) =
('04', '02')
if name == "Peach_pit":             (self.garbage_num, self.garbage_class) =
('05', '02')
if name == "Cigarette_butts":       (self.garbage_num, self.garbage_class) =
('06', '02')
if name == "Disposable_chopsticks": (self.garbage_num, self.garbage_class) =
('07', '02')
```

```
if name == "Egg_shell":              (self.garbage_num, self.garbage_class) =
('08', '03')
if name == "Apple_core":             (self.garbage_num, self.garbage_class) =
('09', '03')
if name == "Watermelon_rind":        (self.garbage_num, self.garbage_class) =
('10', '03')
if name == "Fish_bone":              (self.garbage_num, self.garbage_class) =
('11', '03')
if name == "Expired_tablets":        (self.garbage_num, self.garbage_class) =
('12', '04')
if name == "Expired_cosmetics":      (self.garbage_num, self.garbage_class) =
('13', '04')
if name == "Used_batteries":         (self.garbage_num, self.garbage_class) =
('14', '04')
if name == "Syringe":                (self.garbage_num, self.garbage_class) =
('15', '04')
if name == "None":                   (self.garbage_num, self.garbage_class) =
('None', 'None')
#Get voice recognition result
result = mySpeech.speech_read()
#If the current voice recognition result is 94, it means asking what the current
garbage is
if result == 94:
    if self.garbage_num == '00':
        mySpeech.void_write(94)

    elif self.garbage_num == '01':
        mySpeech.void_write(95)

    elif self.garbage_num == '02':
        mySpeech.void_write(96)

    elif self.garbage_num == '03':
        mySpeech.void_write(97)

    elif self.garbage_num == '04':
        mySpeech.void_write(109)

    elif self.garbage_num == '05':
        mySpeech.void_write(108)

    elif self.garbage_num == '06':
        mySpeech.void_write(107)

    elif self.garbage_num == '07':
        mySpeech.void_write(106)

    elif self.garbage_num == '08':
        mySpeech.void_write(105)

    elif self.garbage_num == '09':
        mySpeech.void_write(104)

    elif self.garbage_num == '10':
        mySpeech.void_write(103)

    elif self.garbage_num == '11':
        mySpeech.void_write(102)
    elif self.garbage_num == '12':
        mySpeech.void_write(101)

    elif self.garbage_num == '13':
        mySpeech.void_write(100)

    elif self.garbage_num == '14':
        mySpeech.void_write(99)

    elif self.garbage_num == '15':
        mySpeech.void_write(98)
#Start thread to grab garbage label code block
threading.Thread(target=self.single_garbage_grap, args=
(self.garbage_class,)).start()
def single_garbage_grap(self, name):
    '''
    :param name: recognized garbage category
    '''
    self.arm.Arm_Buzzer_On(1)
    sleep(0.5)
    # Hazardous waste -- red 04
    if name == "04":
        # print("Hazardous waste")
        # Move to trash can position and drop corresponding pose
        joints_down = self.robot.P_HAZARDOUS_WASTE
        self.move(joints_down)
        # Movement complete
        self.status = 'waiting'
    # Recyclable waste -- blue 01
    if name == "01":
        # print("Recyclable waste")
        joints_down = self.robot.P_RECYCLABLE_WASTE
        self.move(joints_down)
        self.status = 'waiting'
    # Kitchen waste -- green 03
    if name == "03":
        # print("Kitchen waste")
        joints_down = self.robot.P_KITCHEN_WASTE
        self.move(joints_down)
        self.status = 'waiting'
    # Other waste -- gray 02
    if name == "02":
        # print("Other waste")
        joints_down = self.robot.P_OTHER_WASTE
        self.move(joints_down)
        self.status = 'waiting'
#Movement function
def move(self, joints_down):
    joints_uu = self.robot.P_TOP
    # Move over the object's position
    self.arm.Arm_serial_servo_write6_array(joints_uu, 1000)
    sleep(1)
    # Release the jaws
    self.arm.Arm_serial_servo_write(6, self.release_joint, 500)
    sleep(0.5)
    # Move to object position
    joints_center = self.robot.P_CENTER
```

```
        joints_center[5] = self.release_joint
        self.arm.Arm_serial_servo_write6_array(joints_center, 1000)
        sleep(1.5)
        # Gripping, clamping jaws
        self.arm.Arm_serial_servo_write(6, self.grap_joint, 500)
        sleep(0.5)
        # Set up
        self.arm.Arm_serial_servo_write6_array(joints_uu, 1000)
        sleep(1)
        # Lift to the top of the corresponding position
        self.arm.Arm_serial_servo_write(1, joints_down[0], 1000)
        sleep(1)
        # Lift to the corresponding position
        self.arm.Arm_serial_servo_write6_array(joints_down, 1000)
        sleep(1.5)
        # Release the object, release the gripper
        self.arm.Arm_serial_servo_write(6, self.release_joint, 500)
        sleep(0.5)
        # Put up
        self.arm.Arm_serial_servo_write(2, 90, 1000)
        sleep(1)
        # Move to initial position
        self.arm.Arm_serial_servo_write6_array(self.robot.P_LOOK_MAP, 1000)
        sleep(1.5)
```