Service Robot for Garbage Segregation

USER MANUAL

by

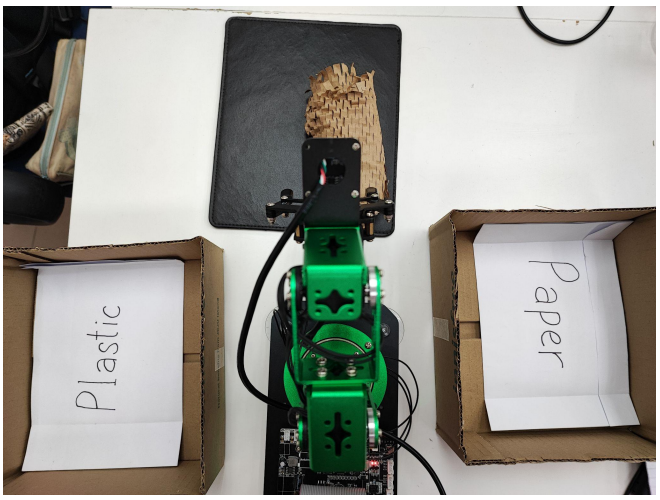
Chan Joey
Lee Jing Hui

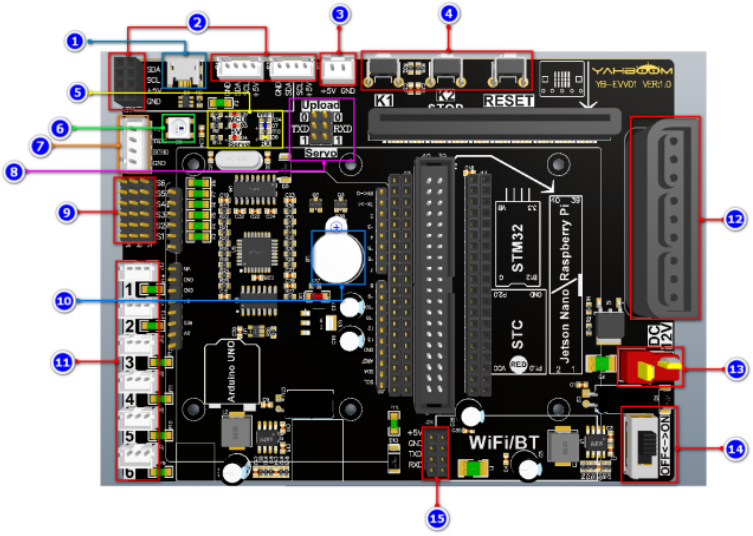
Table of Content

Related code can refer to the github link provided.
https://github.com/jinghui00/Robotic-Garbage-Sorter.git
https://github.com/joey0622/fyp2_Joey_PickAndPlaceRobot.git

Introduction

This project is using a robot named Yahboom Dofbot Raspberry Pi 4B. It has 6-DOF, using Ubuntu 20.04 and ROS Noetic.





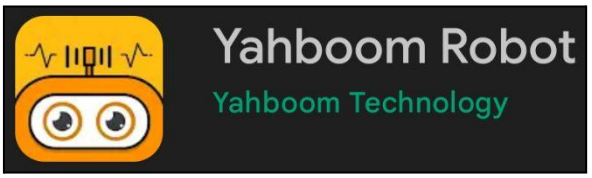
- **K1 button**: In the default mode, press the K1 button once to reset the bus servo, and the DOFBOT keeps upright. Double-click the K1 button to reset the servo quickly.
- **K2 button**: Short press is the emergency stop function of the bus servo to turn off the torque of the bus servo; long press for about 10 seconds is to turn off and turn on the BootLoader function of the underlying single-chip microcomputer. After the BootLoader function is turned on, the RGB light shows a marquee effect.
- **RESET key**: The reset function of the coprocessor (STM8), STM32 and Arduino UNO.

Start the robot

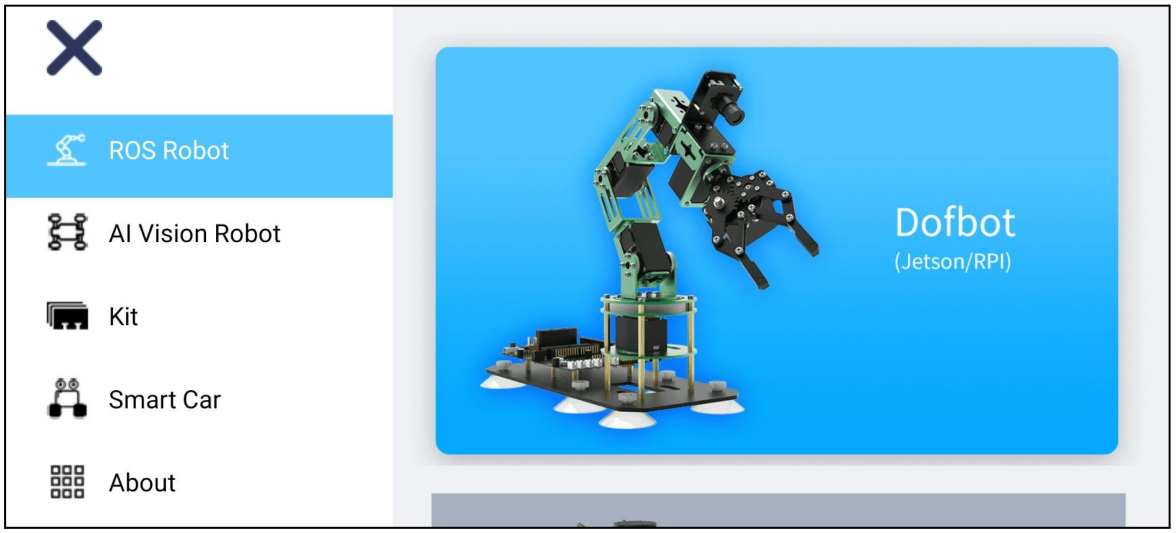
1. Plug in and switch on the robot.
2. Wait until the buzzer whistle 3 times, then the robot is successfully started

Connect robot to WiFi

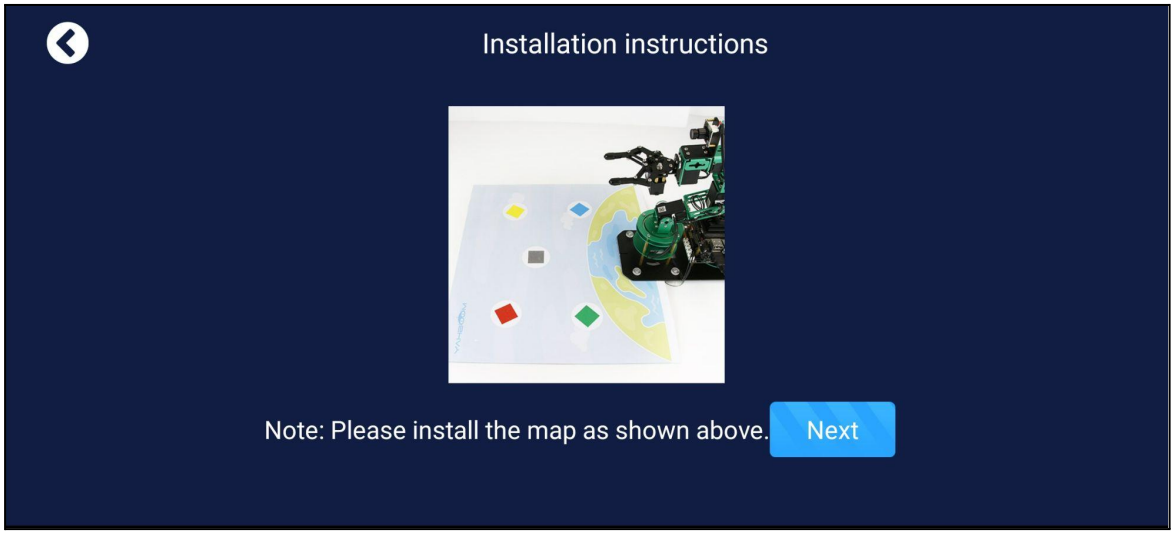
1. In your smartphone/mobile phone, go to Google Play Store (Android) or App Store (iOS) to install the “YahboomRobot” application.



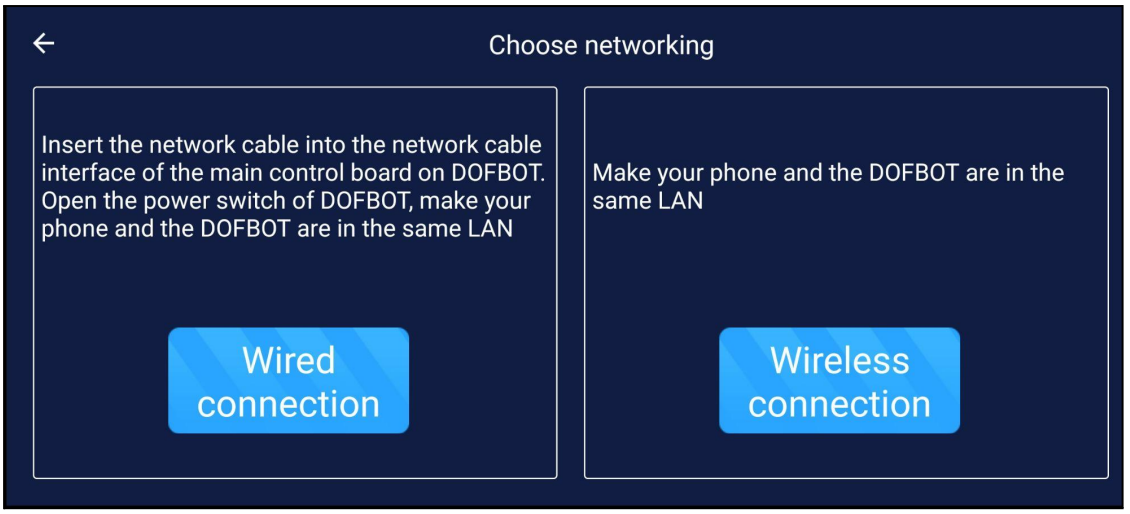
2. Open the application, choose “ROS Robot” and click “Dofbot(Jetson/RPI)”.



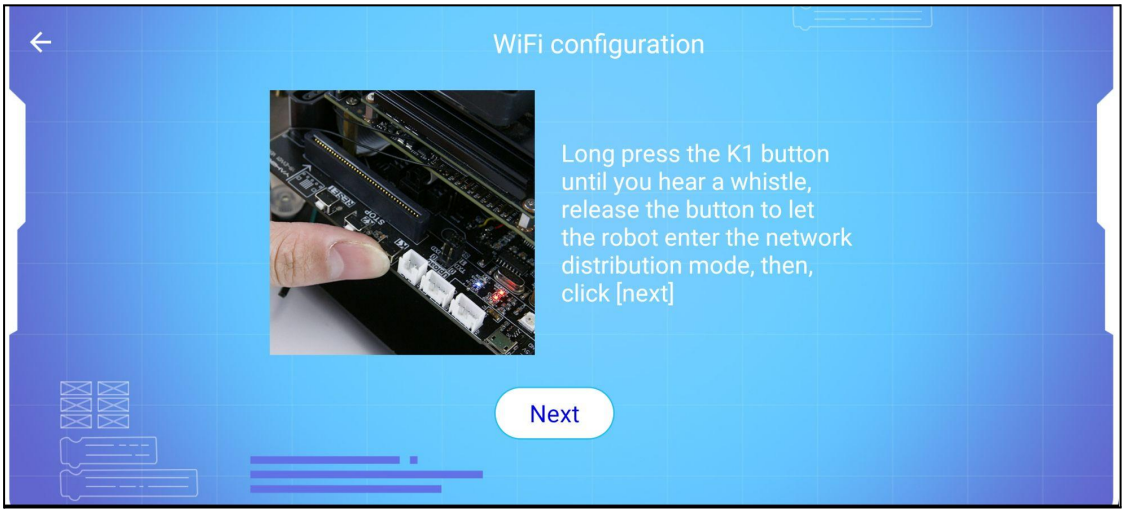
3. Ignore the installation instructions. Click “Next”.



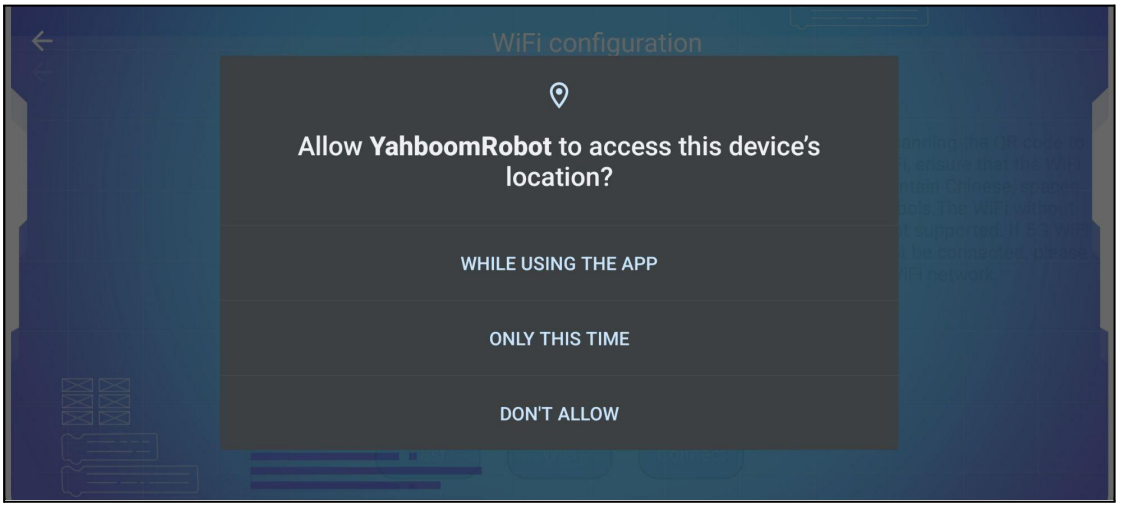
4. Choose “Wireless connection”.



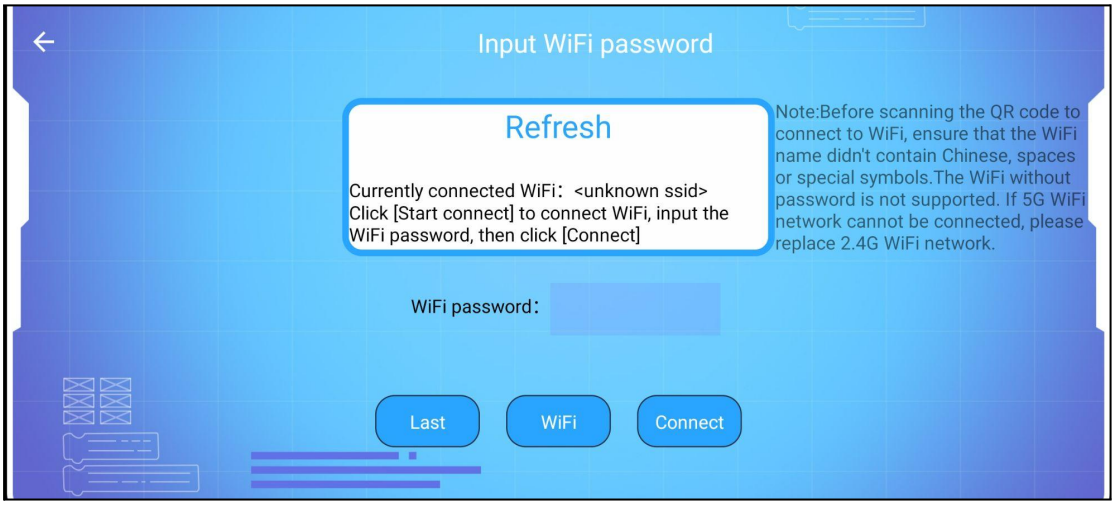
5. Follow the instructions.



6. Give permission to access location.

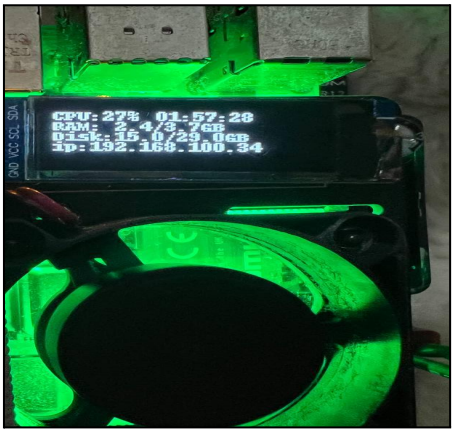


7. Follow the instructions and be aware of the “Note”. Click “WiFi” to choose which WiFi to connect to. After entering the WiFi password, click “Connect”.



8. A QR code will be generated. Place the QR code about 20cm in front of the camera. Wait for the robot to whistle 3 times (network connection complete). Click “OK”.

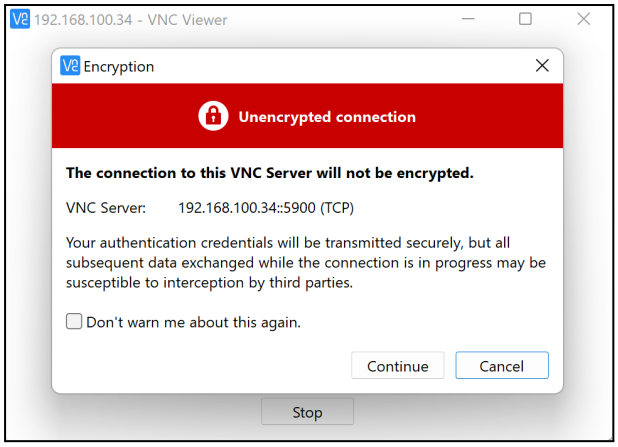
9. OLED will show the IP address based on your wifi connection.

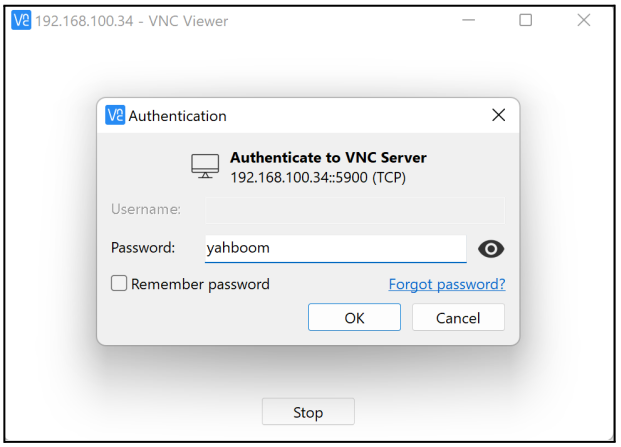


Connect laptop to robot Ubuntu system using VNC Viewer

1. On your laptop, install the VNC Viewer application.
2. Open VNC Viewer.

3. Enter the IP address “192.168.100.34” shown on the robot’s OLED.
4. A window will pop out showing the screen of the Ubuntu system. Press the continue and type in the password “yahboom” to enter the raspberry pi system.

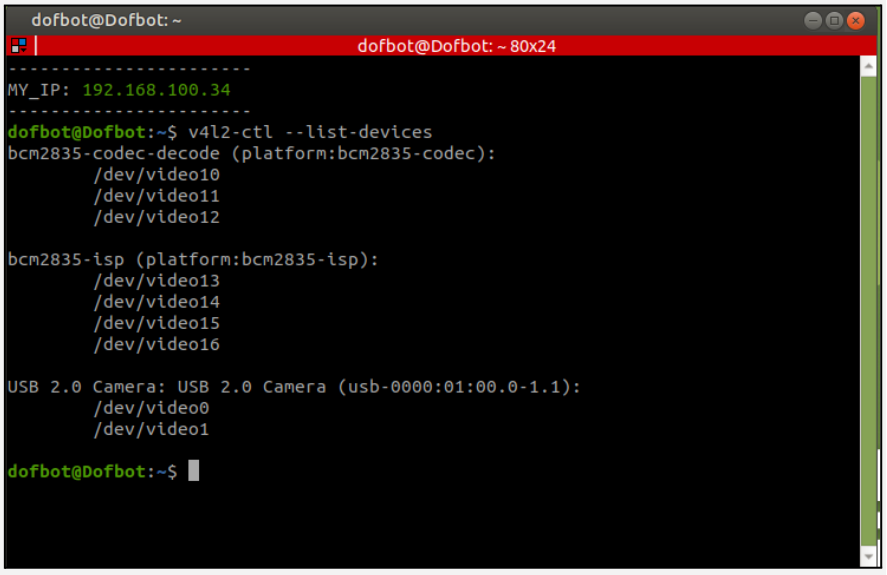




Run the program

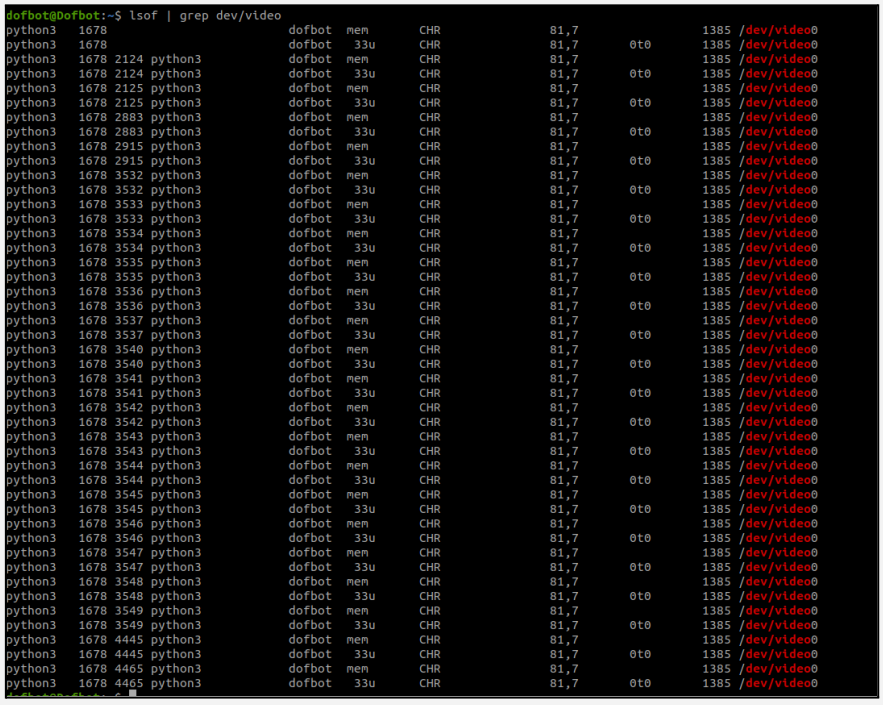
1. Open terminal.
2. Check the camera index. Ensure USB Camera is /dev/video0 and /dev/video1.

   `v4l2-ctl --list-devices`



3. Check if the camera is in use. Diagram below shows /dev/video0 is in use by job id 1385.

   `lsof | grep dev/video`



   3.1 If there are processes that are using the camera, kill that process.

   `kill -9 <job_id>` From the diagram above, the command is `kill -9 1385`.

4. Check if there is a rosnode running.

   ```
   rosnode list
   ```

   4.1 If there are rosnode running, kill all rosnode.

   ```
   killall --exact roslaunch
   ```

   ```
   killall –exact rosrun
   ```

   ```
   rosnode kill -a
   ```

5. Go to the directory containing the bash script.

   ```
   cd dofbot_ws/src/dofbot_moveit/scripts
   ```

6. Program running.

   6.1 Whole program from bin detection to garbage classification and arm grasping

   ```
   ./run.sh
   ```

   6.2 Only bin detection

   ```
   ./detection_bin.sh
   ```

   6.3 Only garbage classification and arm grasping

   ```
   ./detection_garbage.sh
   ```

Code modification/explanation/execution

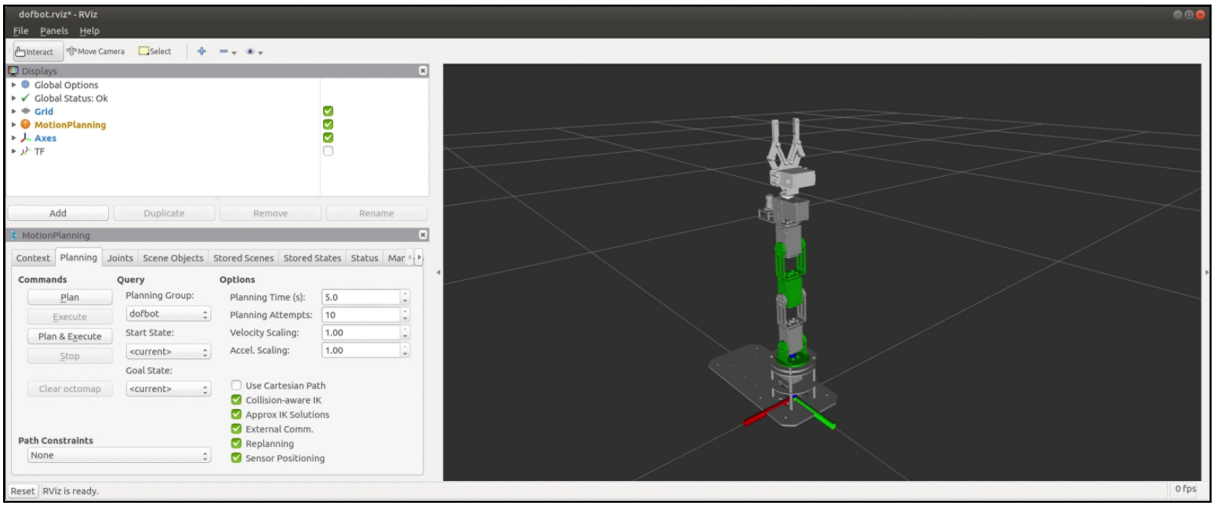
1. Arm manipulation

<u>Control and determine the first five joints of Dofbot for moving to garbage and bin (not including the gripper)</u>

The `dofbot_ws/src/dofbot_moveit/scripts` folder contains the code needed to perform arm control. In the terminal :

- Launch Rviz. Wait until "`Rviz is ready`" (shown at the bottom left of the Rviz window).
- Another window also pops out showing the slider of joint states. There are only 5 joints that can be controlled (joint 1 - joint 5), not including the gripper (joint 6).

```
roslaunch dofbot_config demo.launch
```



- Create a subscriber python file that subscribes to the topic in real-time and drive the robotic arm to move in real-time. The code is in the path dofbot_ws/src/dofbot_moveit/scripts/00_dofbot_move.py.

  *Note: The gripper does not participate in the positive and negative solution of the movement, it needs to be controlled separately. However, do not set the 6th joint to 0°.

```python
#!/usr/bin/env python3
# coding: utf-8

import rospy
import Arm_Lib
from math import pi
from sensor_msgs.msg import JointState

RA2DE = 180 / pi


def topic(msg): # Create a subscriber callback function

    if not isinstance(msg, JointState): return

    # Define the joint angle container, the last one is the angle of the gripper.
    # Gripper needs to be controlled separately. Do not set the No.6 servo to 0°.
    joints = [0.0, 0.0, 0.0, 0.0, 0.0, 90.0]

    # Convert the received angle in radian [-1.57,1.57] to degree [0,180]
    for i in range(5): joints[i] = (msg.position[i] * RA2DE) + 90

    sbus.Arm_serial_servo_write6_array(joints, 100)


if __name__ == '__main__':
    sbus = Arm_Lib.Arm_Device()
    rospy.init_node("ros_dofbot")
    # Create a subscriber
    subscriber = rospy.Subscriber("/joint_states", JointState, topic)
    rate = rospy.Rate(2)
    rospy.spin()
```

- In a new terminal, run the python file.

  ```
  rosrun dofbot_moveit 00_dofbot_move.py
  ```

  After running the command, the arm will be initialized according to the values in the joint angle container.

  You can change the value by moving the slider. The arm movement is shown in the Rviz window as well as in real-time on the robot.

  * Using this slider can help you to determine the values of each joint to move to the garbage and the bin. Then, you can insert the value in the arm_move_*.py file to perform your task. For example, in move_down() of arm_move_down.py.

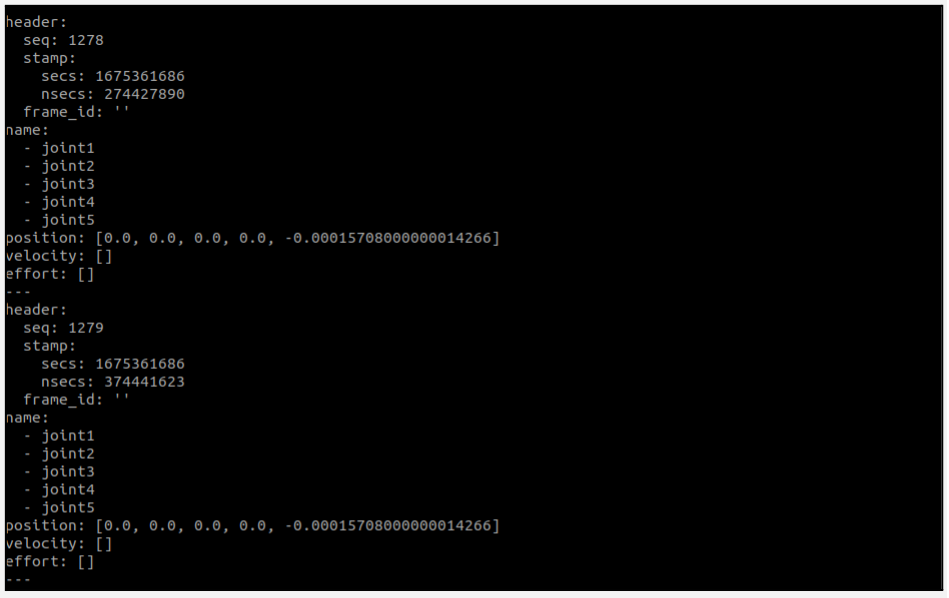
  ```
  dofbot.set_joint_value_target([0.00,-0.95,-0.36,-1.57,-0.00])
  ```

- You can view the topic joint/states using the command below.
    - List all the ROS topics to determine whether /joint_states topic exists.

      `rostopic list`

    - Display the information of /joint_states topic. The "position" shows the joint state value in radian.

      `rostopic echo /joint_states`

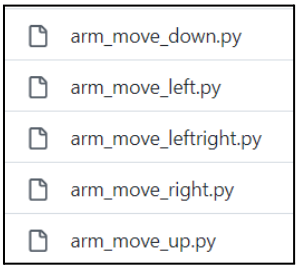


Run the arm movement to move to garbage or bin (not including the gripper)

The dofbot_ws/src/dofbot_moveit/scripts folder contains the code needed to perform arm control. These files are used to perform different arm movements. Gripper is ignored. To run these files, you must have Rviz running first. Then, choose which movement you want and run the corresponding file. For example, to let the arm move to the middle downwards.

To open Rviz: `roslaunch dofbot_config demo.launch`

New terminal: `rosrun dofbot_moveit arm_move_down.py`



Run the gripper of the arm to grasp objects (not including the other 5 joints)

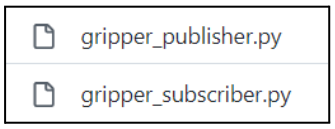
The `dofbot/src/dofbot_moveit/src/scripts/` folder contains the code needed to perform gripper control. To run these files, you must have Rviz running first.

* Note: The subscriber node must be run before the publisher node.

To open Rviz: `roslaunch dofbot_config demo.launch`

New terminal: `rosrun dofbot_moveit gripper_subscriber.py`

New terminal: `rosrun dofbot_moveit gripper_publisher.py`



The ROS topic named “gripper_topic” is used in the publisher and subscriber node to communicate the message.

- Create a subscriber for gripper.

  ```
  sub = rospy.Subscriber('gripper_topic', Float64, gripper_callback)
  ```

  Create a gripper subscriber callback function to receive the gripper position message and perform gripper action on the arm in real-time.

  ```
  gripper_position = msg.data
  ```

  ```
  sbus.Arm_serial_servo_write(6, gripper_position, 1000)
  ```

- Create a publisher for gripper.

```
15      # Create the publisher object
16      pub = rospy.Publisher('gripper_topic', Float64, queue_size=10)
17
18      # Read the angle of joint 6 (gripper)
19      msg = Float64()
20      angle = sbus.Arm_serial_servo_read(6)
21      print("Angle=", angle)
```

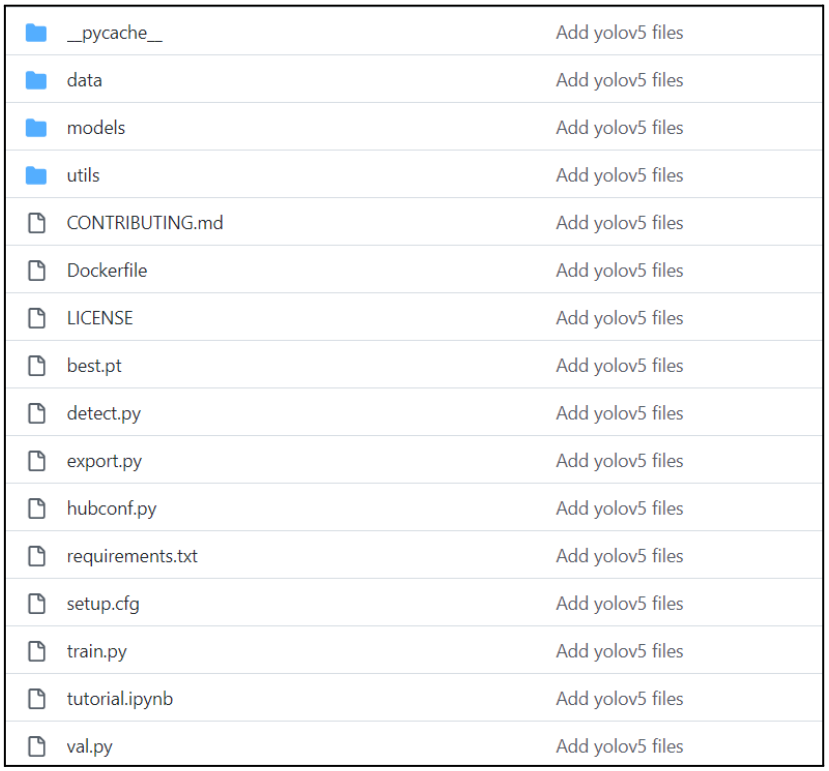
  Determine to open or close the gripper and the corresponding degree value. You may change the angle values which suits your use.

```
23      # Make sure the angle is not None before using it in the calculation
24      if angle is not None:
25          if angle <= 60.0: # If  gripper is open
26              msg.data = 140.0 # Close the gripper
27          elif angle > 60.0 and angle <= 180.0: # If gripper is close
28              msg.data = 30.0 # Open the gripper
29      else:
30          # Handle the case where the angle is None
31          print("Error: Angle is not defined")
```

2. Garbage classification

The dofbot_ws/src/dofbot_moveit/src folder contains the code needed to perform garbage detection and classification. In the directory dofbot_ws/src/dofbot_moveit/src/yolov5, install the requirements of the YOLOv5 model.

```
pip3 install -r requirements.txt
```



__pycache__	Add yolov5 files
data	Add yolov5 files
models	Add yolov5 files
utils	Add yolov5 files
CONTRIBUTING.md	Add yolov5 files
Dockerfile	Add yolov5 files
LICENSE	Add yolov5 files
best.pt	Add yolov5 files
detect.py	Add yolov5 files
export.py	Add yolov5 files
hubconf.py	Add yolov5 files
requirements.txt	Add yolov5 files
setup.cfg	Add yolov5 files
train.py	Add yolov5 files
tutorial.ipynb	Add yolov5 files
val.py	Add yolov5 files

* Note: The garbage classification is using another detect.py file in the directory dofbot_ws/src/dofbot_moveit/src, not the detect.py file in the directory dofbot_ws/src/dofbot_moveit/src/yolov5. This is because the detect.py file in the directory dofbot_ws/src/dofbot_moveit/src has made some changes to implement the YOLOv5 model in ROS.

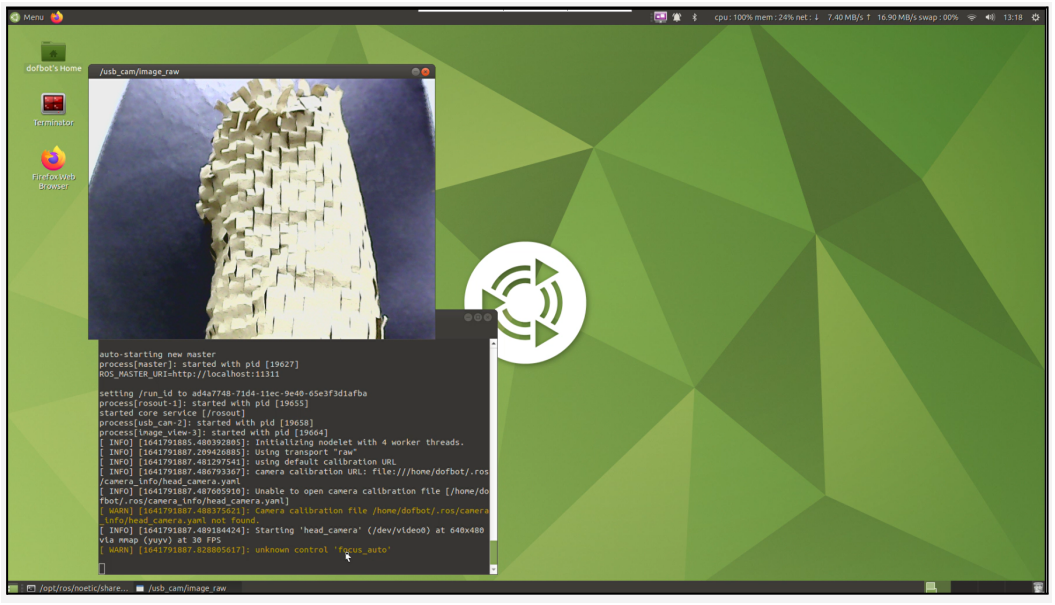
```
39  class Yolov5Detector:
40      def __init__(self):
41          self.conf_thres = rospy.get_param("~confidence_threshold")
42          self.iou_thres = rospy.get_param("~iou_threshold")
43          self.agnostic_nms = rospy.get_param("~agnostic_nms")
44          self.max_det = rospy.get_param("~maximum_detections")
```

All the initialization value of the parameters is obtained from dofbot_ws/src/dofbot_moveit/launch/yolov5.launch using `rospy.get_param()` function.

To perform garbage classification

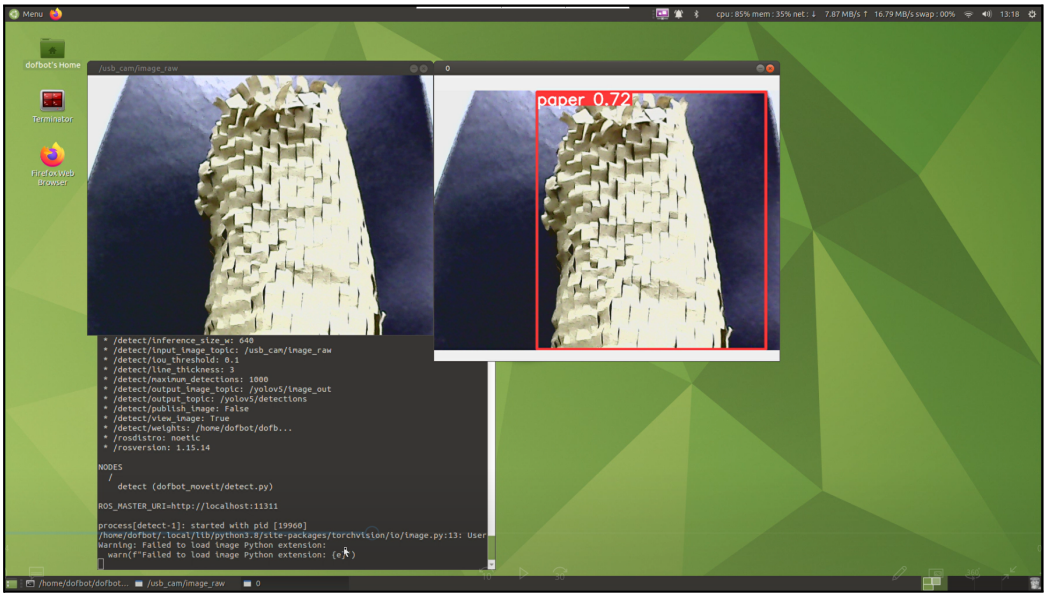
- Open a terminal, bring up the usb camera. Remember to check the camera index and if camera is in use.

```
roslaunch usb_cam usb_cam-test.launch
```

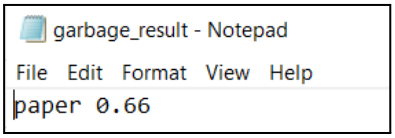


- In a new terminal, run the yolov5 model.

```
roslaunch dofbot_moveit yolov5.launch
```



- The result containing the type of garbage and the score is saved in a text file.



To train new garbage classification model

Go to the github link, refer to YOLOv5_Custom_Training.ipynb. You may directly open the code by clicking the “Open in Colab” button.

To change the weight file and data configuration file

If you trained a new YOLOv5 model, either new weights or more type of recyclable garbage dataset, you can replace the .pt file and add the data in data.yaml file.

- The current weight used is best.pt. It is stored in the path dofbot_ws/src/dofbot_moveit/src/yolov5.
- The data.yaml file is in the path dofbot_ws/src/dofbot_moveit/src/yolov5/data. You may change the nc (number of classes for detection) and add the type of garbage in names.

```
1   nc: 2
2   names: ['Paper', 'Plastic']
```

To change the input parameters of garbage classification

As shown in the figure below is a part of the yolov5.launch file. It specifies the path to weights and data configuration. Besides, you may change the value of confidence_threshold, iou_threshold and other parameters which suit your requirement.

```
cd dofbot_ws/src/dofbot_moveit/launch/yolov5.launch
```

```
<launch>
    <!-- Detection configuration -->
    <arg name="weights"              default="$(find dofbot_moveit)/src/yolov5/best.pt"/>
    <arg name="data"                 default="$(find dofbot_moveit)/src/yolov5/data/data.yaml"/>
    <arg name="confidence_threshold" default="0.50"/>
    <arg name="iou_threshold"        default="0.1"/>
    <arg name="maximum_detections"   default="1000"/>
    <arg name="device"               default="cpu"/>
    <arg name="agnostic_nms"         default="true"/>
    <arg name="line_thickness"       default="3"/>
    <arg name="dnn"                  default="true"/>
    <arg name="half"                 default="false"/>
```

```
    <!-- ROS topics -->
    <arg name="input_image_topic"       default="/usb_cam/image_raw"/>
    <arg name="output_topic"            default="/yolov5/detections"/>
```

- You may change the parameter for input_image_topic to any ROS topic with message type of sensor_msgs/Image or sensor_msgs/CompressedImage if you want to use other camera. (not suggested)
- In our case is using the ros-noetic-usb-cam package to obtain information from the USB 2.0 camera. So, the input_image_topic is /usb_cam/image_raw. To view or change value in the usb cam launch file:

  ```
  roscd usb_cam/launch
  ```

  ```
  sudo nano usb_cam-test.launch
  ```

```
<launch>
  <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="640" />
    <param name="image_height" value="480" />
    <param name="pixel_format" value="yuyv" />
    <param name="color_format" value="yuv422p" />
    <param name="camera_frame_id" value="usb_cam" />
    <param name="io_method" value="mmap"/>
  </node>
  <node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/usb_cam/image_raw"/>
    <param name="autosize" value="true" />
  </node>
</launch>
```

To publish garbage classification result to robot arm

The code is in the directory, dofbot_ws/src/dofbot_moveit/src/publish_result.py.

```
rosrun dofbot_moveit publish_result.py
```

```
start publish result
[INFO] [1641792898.264290]: Detected plastic
```

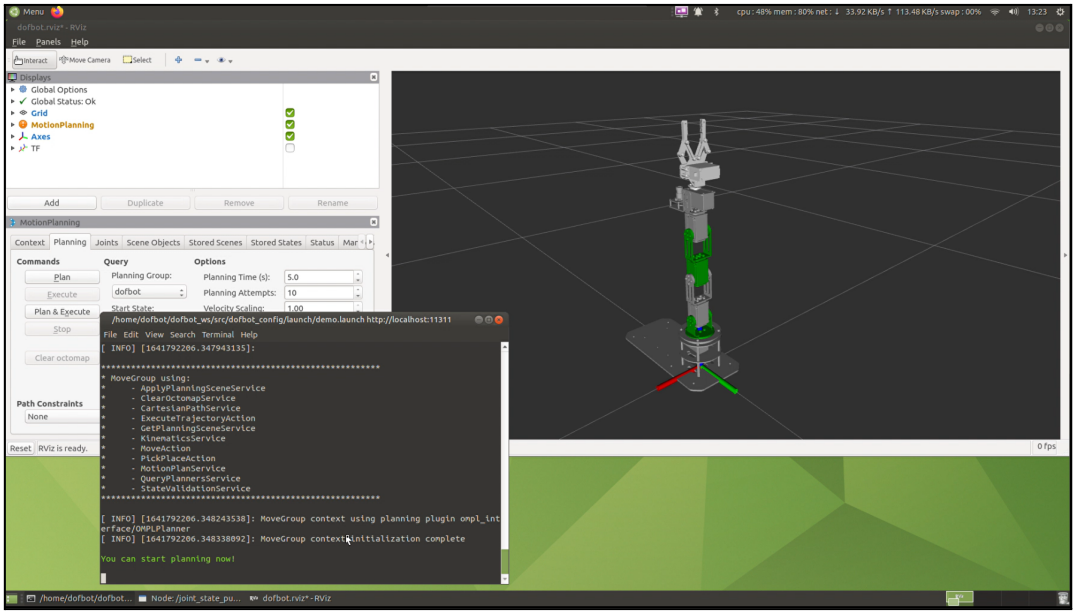
To subscribe garbage classification result to perform corresponding arm movement

* Note: The subscriber node must be run before the publisher node.

To enable the robot arm movement, it must subscribe to the joint states from rviz. Running demo.launch is important as it will publish joint states through rviz. Then, the arm_move_<movement>.py will subscribe to the joint states publisher and publish the desired values for each joint state to move to the bin.

```
roslaunch dofbot_config demo.launch
```

```
rosrun dofbot_moveit arm_move_leftright.py
```



3. Bin classification

The `dofbot_ws/src/dofbot_moveit/src/bin_detect.py` file contains the code needed to perform bin detection and classification.

To setup bin classification

- `pip3 install pytesseract`
- `sudo apt-get install tesseract-ocr`
- In terminal, type "`which pytesseract`" to find the installed path of pytesseract
- Ensure the pytesseract installation path and camera index are correct

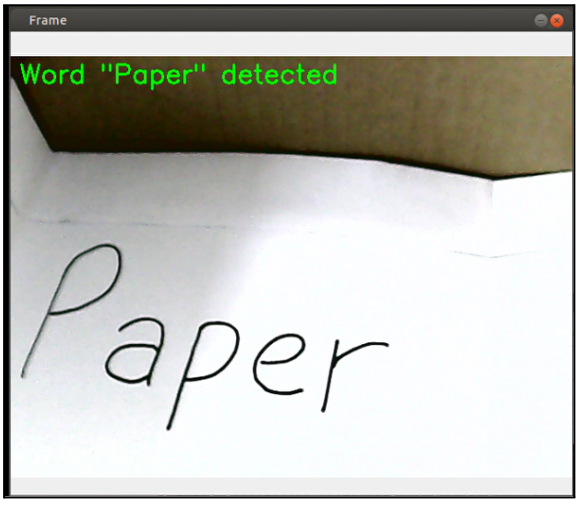
```python
#!/usr/bin/env python3
import rospy
from std_msgs.msg import String
import cv2
import pytesseract

def bin_node():
    # Set the Tesseract path
    pytesseract.pytesseract.tesseract_cmd = "/usr/bin/tesseract"

    # Capture the main image
    capture = cv2.VideoCapture(0)
```

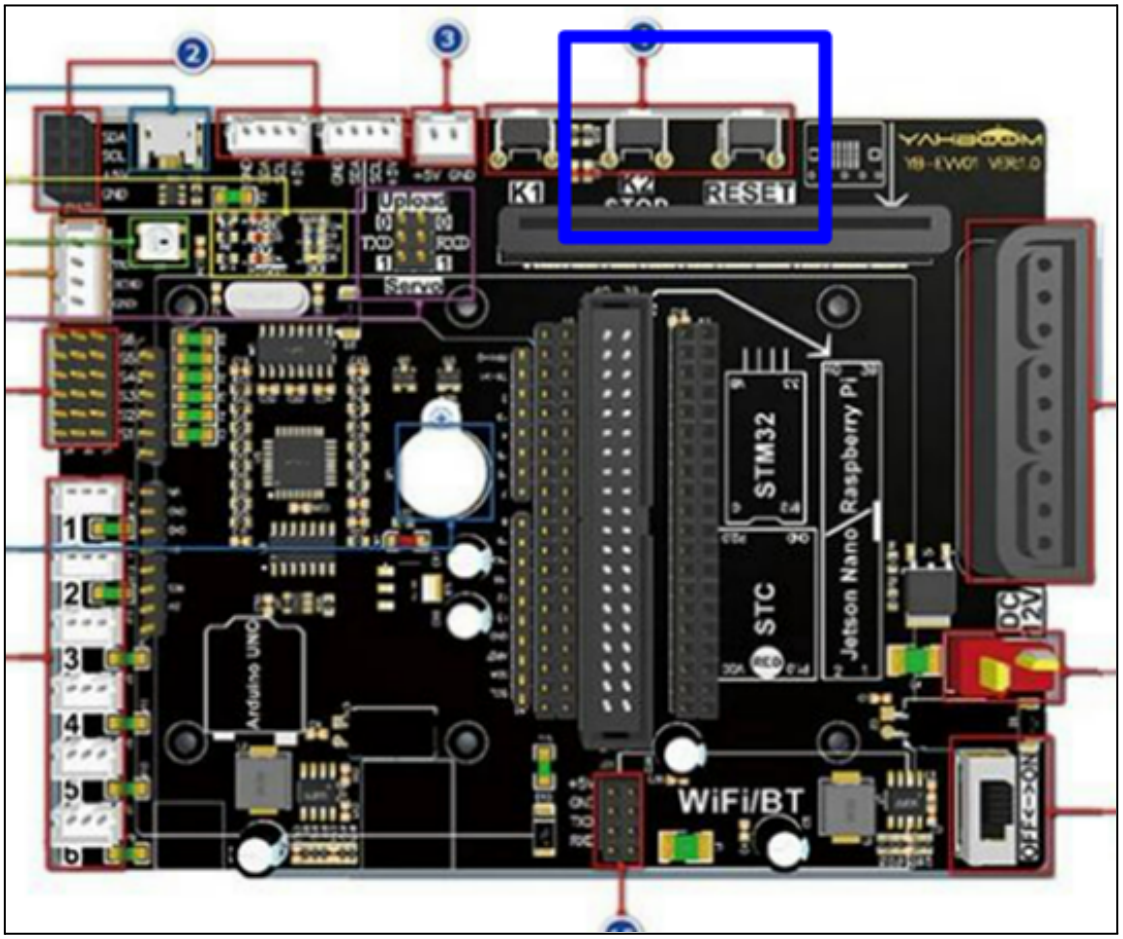
To perform bin classification

`rosrun dofbot_moveit bin_detect.py`



Error handling

1. If the robot is suddenly out of control, press the emergency ‘stop’ button on the robot.



2. If the system is lagging (the screen is changing slowly or the mouse cursor reacts very slow) or the roscore fails to run, kill the running processes.

   Open a new terminal:

   ```
   rosnode kill -a
   ```

   ```
   killall --exact roslaunch
   ```

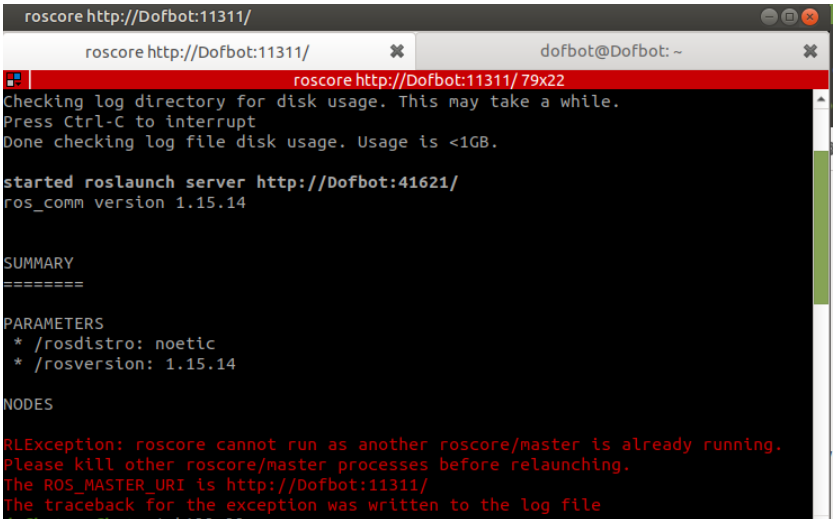
   ```
   killall --exact rosrun
   ```

3. If the system freezed (the screen does not change for a few minutes or the mouse cursor does not react), restart the robot by power off and power on the robot. The system usually freezed when launching Rviz together with garbage classification files (usb_cam-test.launch and yolov5.launch). This is due to the limited CPU and memory of the robot.

4. If the roscore is already running, there will be an error because only one roscore can be run at a time.

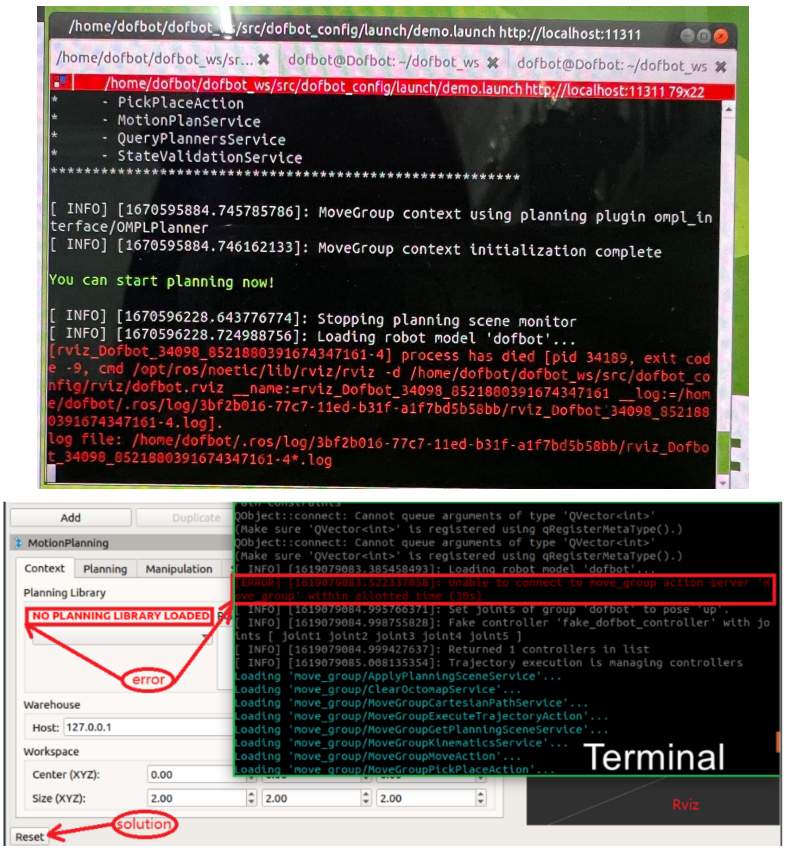
   Open a new terminal:

   `rosnode kill -a`.



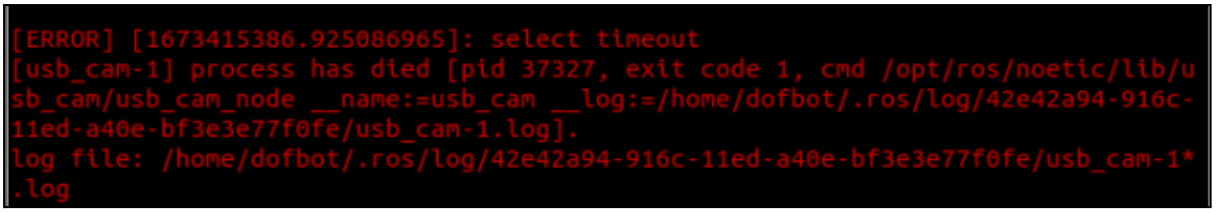
5. Rviz Process has died and no planning library loaded.

   Click the [Reset] button and wait after loading. It is related to the performance of the device, and sometimes it takes a few more attempts or restarts.

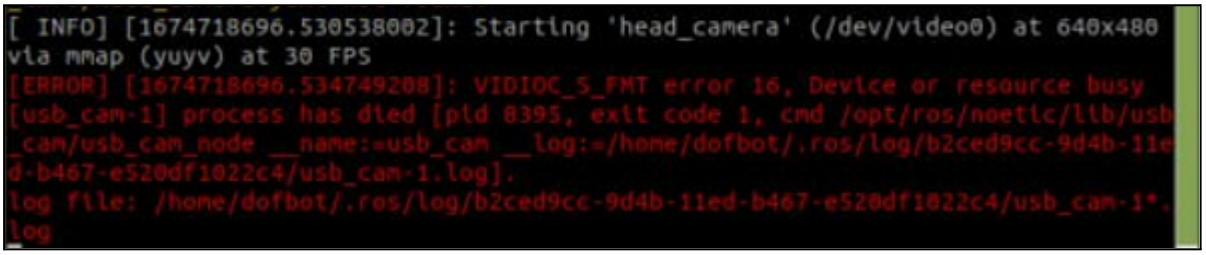


6. Usb camera process has died.

   Close the terminal and relaunch the usb camera.



7. If you face the error camera device is busy, check which process is using the camera.



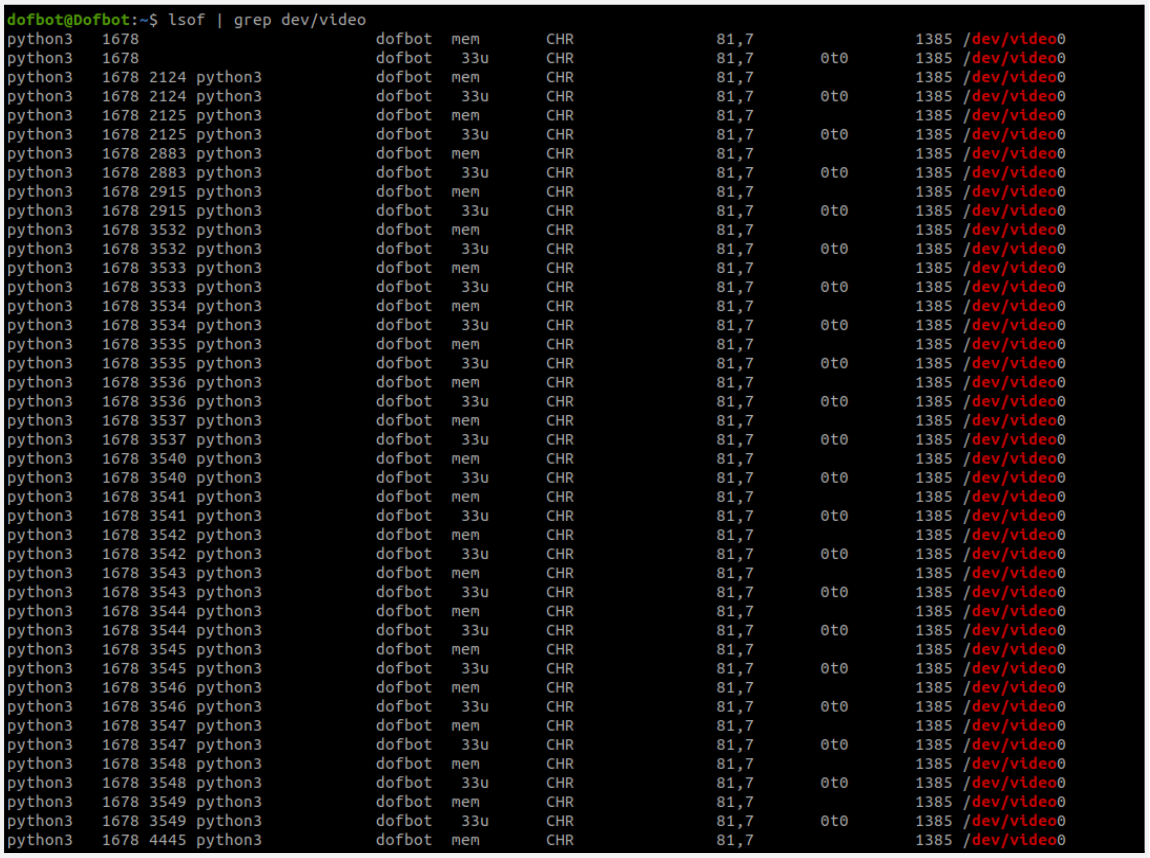
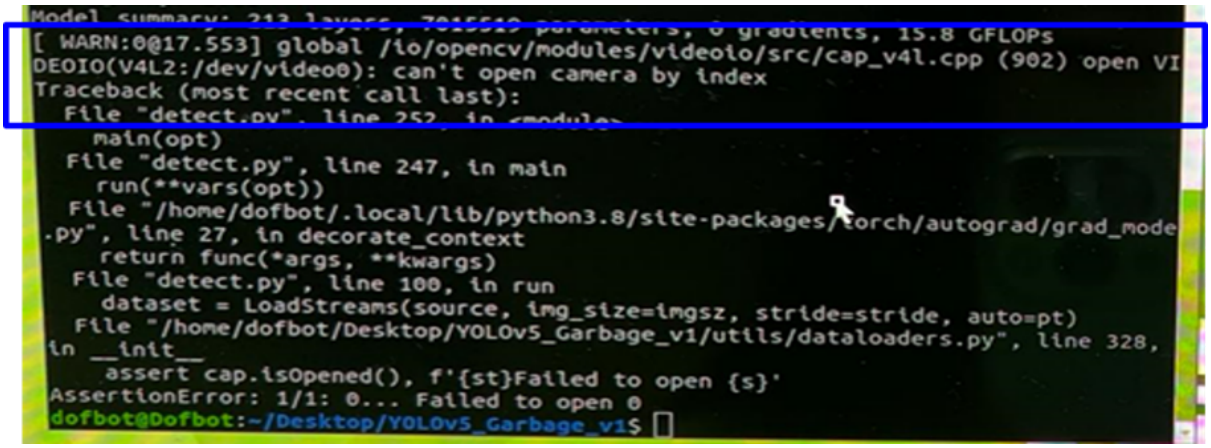


   Diagram below shows /dev/video0 is in use by job id 1385. Kill that process.

```
lsof | grep dev/video
```

```
kill -9 1385
```

8. If you face the error can’t open camera by index, check the camera index.



Ensure the USB 2.0 Camera is showing /dev/video0 and /dev/video1.

Else, restart your robot.

```
v4l2-ctl --list-devices
```