9、QR code instructions

9.1 Introduction to gameplay

This course mainly uses the robot's camera to obtain the camera's picture, identify the QR code information, and control the robot's movement based on the QR code information.

9.2Core content analysis

Initialize the hexapod robot and set the camera pan/tilt angle. The default is S1=90 and S2=30. The initial angle can be modified according to actual needs.

```
from MutoLib import Muto
g_bot = Muto()
g_bot.Gimbal_1_2(90, 30)
```

Import the QR code parsing library pybar

```
import pyzbar.pyzbar as pyzbar
from PIL import Image
```

If pybar is not installed on your system, please open a terminal and run the following command to install it.

```
pip3 install pyzbar
sudo apt install libzbar-dev
```

Analyze the grayscale image and extract the information and image position of the QR code in the image. If there is no QR code in the image, the information is None.

```
def detect_qrcode(image):
    # 转为灰度图像 Convert to grayscale image
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    barcodes = pyzbar.decode(gray)
    for barcode in barcodes:
        # 提取二维码的数据和边界框的位置 The data of the QR code and the position of
the bounding box are extracted
        (x, y, w, h) = barcode.rect
        barcodeData = barcode.data.decode("utf-8")
        barcodeType = barcode.type
        # print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData))
        car_control(barcodeData)
        return barcodeData, (x, y, w, h)
    return None, (0, 0, 0, 0)
```

Control the robot movement according to the string command of info. Among them, g_time_pass is the time of command movement, which can be adjusted according to actual needs, and the unit is 100 milliseconds. By default, the robot will automatically stop after executing the command for 2 seconds.

```
def robot_action(data):
    global g_time_pass
    if g_time_pass >= 0:
        return
    print("Data:", data)
    if data == "forward":
        g_bot.forward()
        g_time_pass = 10 * 2
    elif data == "back":
        g_bot.back()
        g_time_pass = 10 * 2
    elif data == "left":
        g_bot.left()
        g_time_pass = 10 * 2
    elif data == "right":
        g_bot.right()
        g_time_pass = 10 * 2
    elif data == "turnleft":
        g_bot.turnleft()
        g_time_pass = 10 * 2
    elif data == "turnright":
        g_bot.turnright()
        g_time_pass = 10 * 2
    elif data == "stop":
        g_time_pass = -1
        g_bot.zero_reset()
```

Define a task_time task to stop the robot automatically and not execute other QR code instructions during the movement of the robot. g_time_pass is in units of 100 milliseconds. When g_time_pass is greater than 0, it automatically decreases by 1 every 100 milliseconds and stops automatically when timeout occurs.

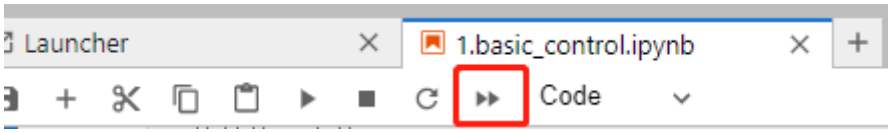
```
def task_time():
    global g_time_pass
    while True:
        if g_time_pass > 0:
            g_time_pass = int(g_time_pass - 1)
            if g_time_pass == 0:
                g_bot.stay_put()
                time.sleep(.1)
                g_time_pass = -1
        time.sleep(.1)
```

9.3 operation

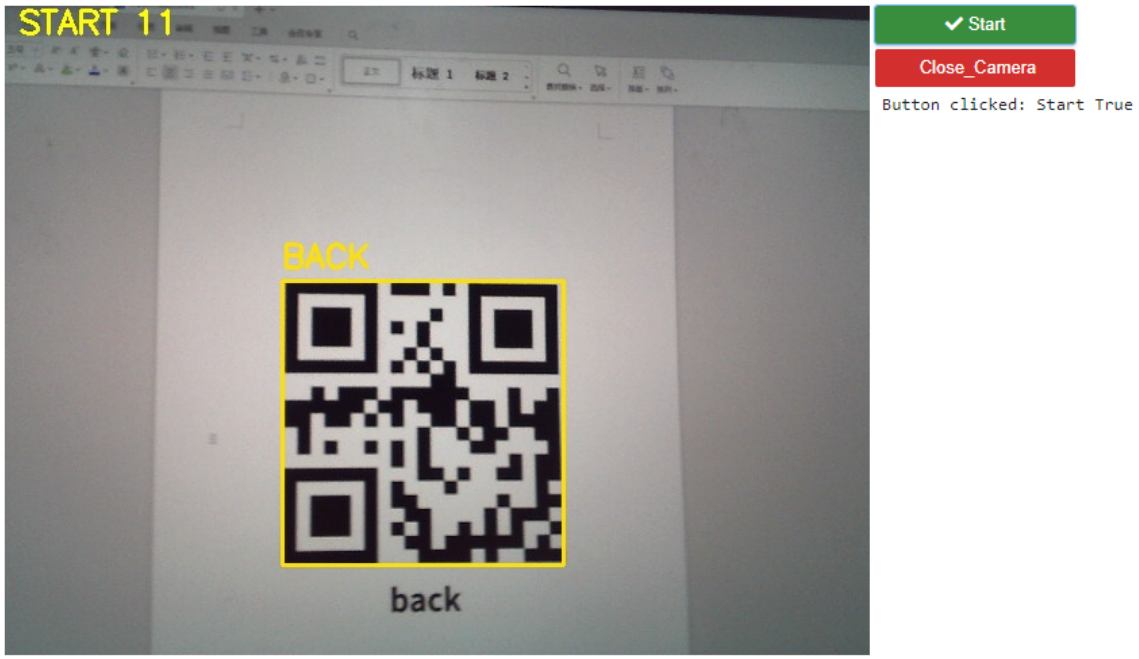
Open the jupyterLab client and find the code path:

```
muto/Samples/AI_Samples/08_qrcode_recognition/qrcode_recognition.ipynb
```

Click to run all cells, and then scroll to the bottom to see the generated controls.



Click the Start button below to start identifying QR codes and execute instructions. The QR code that can be recognized in the current routine is QRCode. The information is "forward" for forward, "back" for backward, "left" for left translation, "right" for right translation, "turnleft" for left rotation, and "turnright" for Rotate right, "stop" means stop.



Finally click the Close_Camera button to close the camera.

Appendix (QR code picture):