PC web programming

1. Web programming instructions

PC web programming uses the Jupyter lab platform, calling built-in buttons and sliders to generate sliders that control the six joints and movement speed of the robot arm. Each time you drag the slider, you can control the rotation of the corresponding servo of the robot arm.

2. Important code explanation

Code path:

```
/home/jetson/dofbot_pro/dofbot_ctrl/scripts/PC_Control.ipynb
```

Create three buttons to control the reset, enable, and uninstall functions of the robot arm.

```
def on_button_clicked(b):
with output:
print("Button clicked:", b.description)
if b.description == 'Reset':
reset_joints()
elif b.description == 'Power_on':
Arm.Arm_serial_set_torque(True)
b.icon = 'check'
button_power_off.icon = 'uncheck'
elif b.description == 'Power_off':
Arm.Arm_serial_set_torque(False)
b.icon = 'check'
button_power_on.icon = 'uncheck'
```

Create six sliders to control the six joints of the robot arm.

```
def on_slider_S1(angle):
 print("J1:", angle)
 mc.send_angle(1, angle, g_speed)
def on_slider_S2(angle):
 print("J2:", angle)
 mc.send_angle(2, angle, g_speed)
def on_slider_S3(angle):
 print("J3:", angle)
 mc.send_angle(3, angle, g_speed)
def on_slider_S4(angle):
 print("J4:", angle)
 mc.send_angle(4, angle, g_speed)
def on_slider_S5(angle):
 print("J5:", angle)
 mc.send_angle(5, angle, g_speed)
def on_slider_S6(angle):
 print("J6:", angle)
 mc.send_angle(6, angle, g_speed)
```

Create a slider to control the servo's movement speed.

```
slider_speed = widgets.IntSlider(description='Speed:', value=1000, min=0,
max=2000, step=100, orientation='horizontal')

def on_slider_speed(value):
global g_speed
g_speed = value
print("speed:", value)

widget_speed = widgets.interactive(on_slider_speed, value=slider_speed)
```

Reset the joint angles of the robot arm.

```
def reset_joints():
if button_power_off.icon == 'check':
Arm.Arm_serial_set_torque(True)
time.sleep(1)
Arm.Arm_serial_servo_write6_array([90, 164, 18, 0, 90, 30], 1000)
slider_S1.value = 90
slider_S2.value = 164
slider_S3.value = 18
slider_S4.value = 0
slider_S5.value = 90
slider_S6.value = 30
slider_speed.value = 1000
button_power_on.icon = 'check'
button_power_off.icon = 'uncheck'
```

Create a camera display window, read the camera image in real time and display it.

```
imgbox = widgets.Image(format='jpg', width=640, height=480,
layout=widgets.Layout(align_self='center'))
model = 'Start'
```

```
def camera():
global model
capture = cv.VideoCapture(0)
capture.set(cv.CAP_PROP_FRAME_WIDTH, 640)
capture.set(cv.CAP_PROP_FRAME_HEIGHT, 480)
while capture.isOpened():
try:
_, img = capture.read()
if model == 'Exit':
break
imgbox.value = cv.imencode('.jpg', img)[1].tobytes()
except:
break
with output:
print("capture release")
capture.release()
```

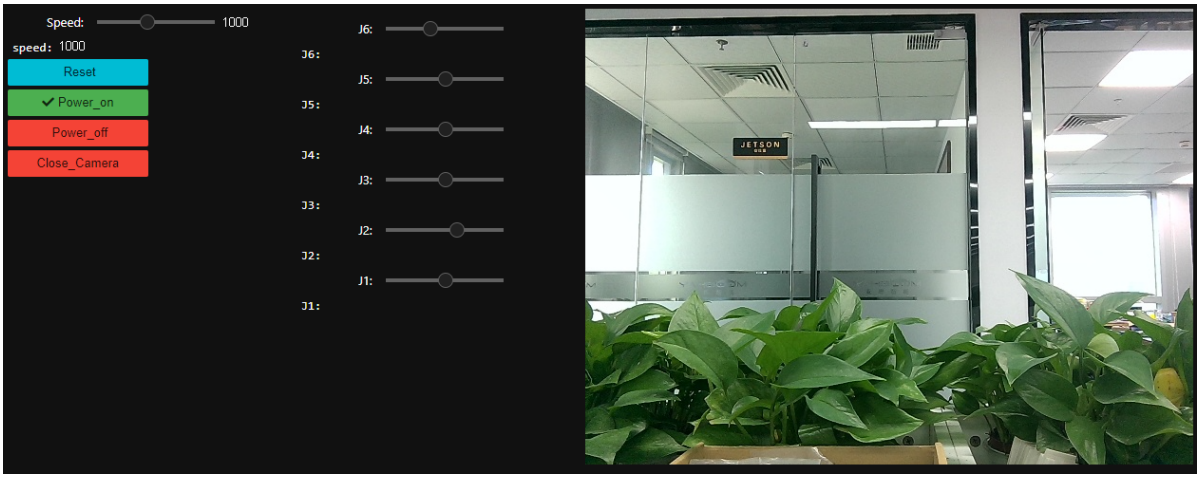
Create a close button to end the program and release resources.

```
button_close = widgets.Button(description='Close_Camera', button_style='danger')
def button_close_Callback(value):
global model
model = 'Exit'
with output: print(model)
button_close.on_click(button_close_Callback)
```

3. Run the program

Click the Run the entire program button on the jupyterlab toolbar, and then pull it to the bottom.

You can see that the relevant control controls are displayed on the left and the camera display screen is displayed on the right.



The corresponding positions of the robotic arm joints are shown in the figure below.