

Control the end coordinates of the robot arm

1. Explanation of important codes

Code path: ~/jetcobot_ws/src/jetcobot_ctrl/scripts/ctrl_coords.ipynb

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

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

Create seven sliders to control the six coordinate parameters and gripper of the robot arm.

```
def on_slider_gripper(angle):
    print("G7:", angle)
    mc.set_gripper_value(angle, g_speed)

def on_slider_coords(x, y, z, rx, ry, rz):
    coords = [x, y, z, rx, ry, rz]
    mc.send_coords(coords, g_speed, 0)
```

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

```
slider_speed = widgets.IntSlider(description='Speed:', value=50, min=1, max=100,
                                  step=1, orientation='horizontal')

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

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

Reset the joint angles of the robotic arm.

```
def reset_joints():
    if button_power_off.icon == 'check':
        mc.power_on()
        time.sleep(1)
```

```
mc.send_angles([0, 0, 0, 0, 0, -45], 50)
mc.set_gripper_value(100, 50)
slider_x.value = 50
slider_y.value = -64
slider_z.value = 419
slider_rx.value = -90
slider_ry.value = -45
slider_rz.value = -90
slider_g.value = 100
slider_speed.value = 50
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
    print("capture release")
    capture.release()
```

Create an end 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)
```

2. 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.

