Industrial Automation
Fall 2024
Assignment 2

OUTLINE:

1. Each student - individually, needs to follow all the steps below and:
hand in all exercise scripts (-ex.py) to TA by the due date as explained below.

2. Copy all the assignment's given scripts in to your raspberrypi's following directory (bin):

In your raspberrypi's home directory:

```
$ mkdir bin
```

Download all assignment's scripts and copy them all to bin directory from your local computer using scp:

```
$ scp *.* <your_uni>@<your_ip_address>:/home/<your_uni>/bin/.
```

3. Follow all scripts and exercises step-by-step meticulously and make sure you undertand every single word of them.

4. The exercise scripts has '-ex.py' suffix in their file names. These should be zipped and sent to TA by the due date.

5. The rest of scripts are building blocks of your learning; skipping those will be detrimental for your future works.

Make sure you are comfortable with all the scripts, your scripts work with your computer and are stored in ~/bin/ directory.

6. If a scripts is labeled as 'DEMO', you need to demonstrate it works in your robot to your TA.

A group-based demonstration for this assignment is enough for these types of scripts. This means that only one person of the group will need to show their DEMO script works.

7. Imortant industrial practice:

"Be mindful and considerate to the others who will likely need to read your code. Always put comments and reminders."

8. If you have completed your RaspberryPi0.txt steps, then you'll know your robot's computer's ip address.

Then, you'll be able to ssh to your robot's computer through your laptop - no need for external monitor, mouse, keyboard.

For instance, Columbia's network, gives 10.206.17.37 for our robot.

Check if it assigns the same ip address when you shut down and re-connect.

9. This exercise has 3 sections as following:

Section A: Warming up with python and programming

Section B: Building Robot Shell

Section C: Building Web Services

------------ Section A: Warming up with python and programming ------------

1. controller01.py:

Write this code yourself and make sure that you understand all lines of it.

2. controller02.py:

Points to notice in this script:
- The script is very well commented.
- How we define main() function and pass arguments from it.
- How made a function
- How we define our hyper parameters
- How we import libraries

3. controller03.py:

This script detects your GPIO devices.

In our case we have connected crickit to it.

Make sure you understand all lines about I2C in RaspberryPi0.txt and your're comfortable with this script.

4. controller04.py:

Shows the way that touch sensors of circkit work.

5. controller05-ex.py:

This script should be written by you. Follow the instruction in the script.

6. controller06.py:

The very basic of motor drive; showing:
- how to drive both motors to 0.5 seconds;
- how to choose throttle power.

7. controller07-ex.py:

You need to complete this script. When the sensor 1 is touched, then:
- LED turns to red,
- The robot runs full power (0.9 throttle always for fast speed) for 0.3 seconds

8. controller08.py:

This script teaches you:
- how to pass user arguments,
- how to write an object-oriented script using 'class'
- how to instantiate an object and play with it.

Try to understand every single part of it.

9. controller09-ex.py:

This is an extension of controller08.py.

If you successfully run controller08.py, you would notice that the robot does not go straight.

The reason is that the applied power - mechanically to both motores, are not the same.

There are three solutions for this:

a) Change the throttle value experimentally - by multiplying a fixed
coefficient to the assigned speed value.

b) Apply encoder - speed detectors, to both motors and control their speed by a
feedback signal.

c) Apple a gyroscope sensor to detect robot's angular velocity and then correct
/ adjust the motor speeds.

In this lab, we will use method 'a' since our focus is not robotics.

Our goal is to adjust self.SyncForwardR and self.SyncBackwardR variables which will
be robot-dependent.

These are the coefficients which will be applied to the motors' throttle values.

Couple of points to notice:
- we apply coefficient on only one motor since the adjustment is relative,
- the coefficients for forward and backward are (may be) different.

You need to first assign values to these variables and then complete the move()
method in this class.

You'll need to do a few trial and errors to find correct parameters for the robot
to move straight for about 24 inches.

You'll use your found parameters to the next scripts.

10. motor-ex.py

This is an important library you need to complete. The rest of the course, will use
this library.

Then, any problem in this script, will cause your future scripts, assignments and
final project fail.

Follow the step-by-step instructions as below:
- Learn python's functools.partial. We'll use this in this script.
- Complete the sync parameters SyncForwardR and SyncBackwardR as you found in
controller09-ex.py
- Notice how we defined THROTTLE_SPEED and MOTOR dictionaries for the script to
use.
- Complete the required lines where there is a question mark.
- Once you're done and you're sure that the script works fine, copy the script

to motor.py since you'll use it later on.

```
$ cp motor-ex.py motor.py
```

------------ Section B: Building Robot Shell ------------

From now on, we'll use your motor.py library in our other python scripts.

Shell is a command line prompt - similar to python with the difference that it's designed to operate robots.

We'd like to develop a complete shell to provide something like and waits for a command to execute:

```
$ (robot)
```

We'll use python's powerful cmd library for this purpose.

1. shell1.py

In this script, we create a class called RobotShell with the type of cmd.Cmd.
Try to comprehend how this script works and run it.

When the robot shell comes up, you could type ? to see available commands.

As you would notice, the available commands are forward and backward which calls motor library to execute them.

It also uses the default value of these functions - it is not able to pass arguments - such as adjusting speed and duration.

2. shell2.py

We'll add command line option to this script as so the user will be able to pass the speed and duration amount.

Quit option is also added to the script. When user types 'Quit', it exits the shell.

3. shell3-ex.py

Since the motion commands need 'duration' and 'speed' for a full command customization, we'll need a smarter method to break down the passed arguments in the shell. For this purpose, we have a separate function to handle scenarios.

Additionally, we added precmd() method to the RobotShell class to print whatever command is passed.

Make sure you understand all the parts well.

------------ Section C: Building Web Services ------------

In this section, we want to explore three main ways of connecting your local computer to your robot's computer.

We will explore three fields:

C.1) ssh
C.2) Remote execution using subprocess
C.3) URL and HTTP protocol
C.1) ssh:

In order to connect to your robot from your local computer (client):

1. Create an ssh key in your local computer:

- check if your computer already has it; it should be located here:

```
$ ls ~/.ssh/id_rsa
```

- if not, you may create it:

```
$ ssh-keygen -t rsa
```

This will prompt the default location as above; accept it.

Now, you should have id_rsa file in ~/.ssh/id_rsa

2. Transfer your id_rsa to your remote (raspberry pi) machine:

```
$ ssh-copy-id <your_uni>@<your_ip_address>
```

3. Once your local computer's id is added to your raspberrypi, your ssh won't ask your password again.

4. Now you can directly call a script from your local computer.

We'll run webservice1.py in your raspberrypi's bin folder.

Read this script carefully to see how it operates robot from your local
computer.

```
$ ssh <your_uni>@<your_ip_address> '~/pyenv/bin/python ~/bin/webservice1.py
spin_left --duration 1 --speed 3'
```

This script uses getattr() to get the attributes of a function in a given
library.

See how it works in this script and familiarize yourself with this function as
it's one of the python's powerful tools.

C.2) subprocess script in your client computer; you should run this script from
your local computer.

subprocess is powerful python library which executes scripts outside of the current
script. In this section, we'd like to run a script from your local computer to connect to your
robot automatically.

webservice2.py: follow the instructions in the script to completely understand how
it calls webservice1.py to perform tasks.

Reminder: webservice1.py is seating in your robot's computer and you're calling
that script from your local computer using webservice2.py

C.3) Connecting via url and http protocol.

There are three famous options to create web services in python:
- tornado: https://www.tornadoweb.org
- django: https://www.djangoproject.com
- flask: https://flask.palletsprojects.com

We will use tornado in this since course since it suits well our needs.

1. Install tornado in your robot's computer:

```
$ ~/pyenv/bin/pip install tornado
```

2. Prepare webservice3.py script in your robot's computer:
- Follow the instructions in the script to finalize wrt your robot.
- Make sure your assigned port is correct - ask if you are not sure.
- Read all comments in the script and make sure you understand what each line

do.
- When you understand all parts, run it in your robot's computer:

```
$ ./webservice3.py
```

This should be running w/o issue until you cancel or turn your robot's computer off.

- In order to test if the app is listening to the requests, from a separate terminal of your robot, use curl command:

Assuming that your robot's port is 8888:

```
$ curl http://localhost:8888/
```

If this does not work:

```
$ curl http://<your_ip_address>:8888/
```

3. Now, we want to dive deeper and post customized requests using curl.

Prepare webservice4.py in your computer:

- Make sure your assigned port is correct - ask if you are not sure.
- Read all comments in the sctipt and make sure you understand what each line do.
- Run webservice4.py in your robot's computer.
- In order to test if the app is listening to the requests, from a separate terminal of your robot, use the following curl commands:

```
$ curl http://<your_ip_address>:8888/ # assuming that you are assigning port number 8888

$ curl http://<your_ip_address>:8888/forward # run forward command

$ curl -X POST http://<your_ip_address>:8888/spi_right -d '{"duration": 1, "speed": 3}'

$ curl -X POST http://<your_ip_address>:8888/backward -d '{"speed": 1}'
```

- Try different command lines to see how your script works.

- As you would notice, we can pass arguments into a computer that listens through a port.

4. In this section, we will talk to the robot's computer from a remote computer.
- We will use python's power urllib library.

- Run webservice4.py in your robot's computer and leave it running.
- In your remote computer, enter to python environment and:

```
>>> MY_PORT = 8888 # replace this number with your group's assigned number
>>> from urllib.request import urlopen
>>> urlopen('http://<your_ip_address>:f{MY_PORT}/').read() # listen to port of the robot's computer
```

- Now you can pass arguments from your remote computer to your robot's computer:

```
>>> urlopen('http://<your_ip_address>:f{MY_PORT}/spin_right', data=b'{"speed": 1, "duration": 3}').read()
```

- Note that the result of execution can be passed to a variable:

```
>>> result = urlopen('http://raspberrypi:f{MY_PORT}/').read()
>>> import json
>>> json.loads(result).keys() # shows the fields of the retrieved information
```

5. Now, it's time to finish a script that seats on your local computer and controls the robot.

In this part, we'll use webservice4-client1-ex.py which uses same protocol as above: url rquests.

This script should transfer commands to webservice4.py which is running in the robot's computer.

Make sure you understand all parts of this code which is crucial to any url-based applications.

6. As the last part of this assignment, we'll learn http protocols.

The robot's script webservice4.py is still running in its terminal (if not, make sure it is).

Complete webservice4-client2-ex.py and run in your local computer. This will bridge your computer to the listening port through the http library.

This is your DEMO script. The instruction is mentioned above.

NOTE:

Network speed is VERY crucial for industrial automation pplications.
That's why at the end of this script, we compute the connection latency.

Make sure you compute your connections latency and consider it as an attribute
of your robot.

Q: Why is it important?

A: Imagine, you need to send a command to a machine which must run your command within 1 millisecond.
If the network latency (timeout) is 10 ms in average, then your connection and solution underperforms and violates the expected requirements!

Now that you have finished all the parts, you can zip in all -ex.py scripts and hand in to TA.