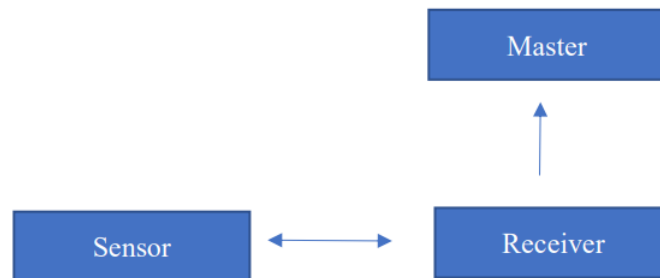


Case Study 2

Zhaofeng Tian

gy8217

1. Project Introduction



Three Nodes – Architecture

There are three nodes: Sensor – which is connected to a hardware and sends a reading (assume a random number between 1 and 100) to another node 'Receiver.' The receiver records the values and if the average of five consecutive readings is 60 or above, then the receiver should 'abort' the operation of sensor. Otherwise, the receiver should send a report to another node 'Master' once in every minute regarding the average value collected in the last minute. However, if the sensor is aborted then the message should be sent to the master immediately.

Implement the above requirements using Python scripts and ROS. Document the screenshots and explanations of your run and submit the python package as well.

2. Solution & Code Explanation

- Using action communication to implement Sensor and Receiver.
- Using topic communication to implement the communication between Receiver and Master.

Action file:

Ave.action

The goal part means the threshold of five consecutive readings to abort the server, set as 60 in this project.

Result Part:

average_five: the average value of five consecutive readings.

circles: how many minutes past.

Feedback Part:

random_number: the random number generated

average_minute: the average value of the random numbers in the past one minute.

```
float32 threshold
---
float32 average_five
float32 circles
---
uint32 random_number
uint32 average_minute
```

Sensor (server):

```
def do_calculator(goal):
    start_time = time.time()

    result = AveResult()
    feedback = AveFeedback()
    result.average_five = 0
    result.circles = 0.0
    count = 0.0
    array = []

    while result.average_five < goal.threshold:

        feedback.random_number = random.randint(0,100)
        array.append(feedback.random_number)
        count += 1
        if count % 5 == 0:
            result.average_five = sum(array[-5:]) / 5.0
        if count % 60 == 0:
            feedback.average_minute = sum(array[-60:]) / 60.0

        server.publish_feedback(feedback)
        time.sleep(0.995)

    if count < 60:
        feedback.average_minute = sum(array) / len(array)
    else:
        feedback.average_minute = sum(array[-60:]) / 60
    server.publish_feedback(feedback)
    # BEGIN PART_8
    result = AveResult()
    result.average_five = sum(array[-5:]) / 5
    result.circles = count / 70.0
    # server.set_succeeded(result, "Exit calculator because five consecutive readings is 60 or above")
    server.set_aborted(result, "Exit calculator because five consecutive readings is 60 or above")
    print (time.time() - start_time)
```

Initialize average_five and circles with 0, using variable count to count every five numbers and every 60 numbers to trigger the respective calculations.

In the while loop, random numbers are generated each of which is assigned to feedback.random_number and appended to the variable array. Then increase count by 1. Two 'if' conditional statements are used to calculate the average values of five consecutive numbers and numbers in the past minute. Feedback the random number and average value of last 60 numbers (represents average value of the numbers in the past one minute).

Sleep time is set to 0.995, it should have been 1.0, but setting to 0.995 is for the compensation of computing time.

When the average value of five consecutive numbers exceeds 60, exit the while loop. If the time does not reach one minute, which means there are not at least 60 numbers in the array, calculate the average value base on what we have in the array and feedback. Else if we have more than 60 numbers, we calculate the average value of the last 60 numbers in the array and then feedback.

Receiver (client):

```
trigger_value = 0.0
|
def feedback_cb(feedback):
    print('[Feedback] Random number is: %f'%(feedback.random_number))
    global trigger_value
    if feedback.average_minute != trigger_value:
        print('[Feedback] Average value of readings in the past one minute: %f'%(feedback.average_minute))
        pub.publish(feedback.average_minute)
        trigger_value = feedback.average_minute

rospy.init_node('Receiver')
pub = rospy.Publisher('counter', Float32, queue_size = 10)
client = actionlib.SimpleActionClient('reading', AveAction)
client.wait_for_server()

goal = AveGoal()
goal.threshold = 60.0

client.send_goal(goal, feedback_cb=feedback_cb)
client.wait_for_result()

print('[Result] State: %d'%(client.get_state()))
print('[Result] Status: %s'%(client.get_goal_status_text()))
print('[Result] The average value of last five consecutive readings: %f'%(client.get_result().average_five))
print('[Result] How many minutes have past: %d'%(client.get_result().circles))
```

To solve the problem that random_number should be recorded every second, but the average value of numbers in the past minute should be published one time per minute and published to the Master, I build a trigger_value parameter. If the average_minute dose not equal to trigger_value, print and publish it, then

assign its value to the trigger_value. The feedback of its values in one minute (if possible) are actually same, so only when the value changes, which means we have counted another 60 times and have obtained a new value, the publisher will be triggered.

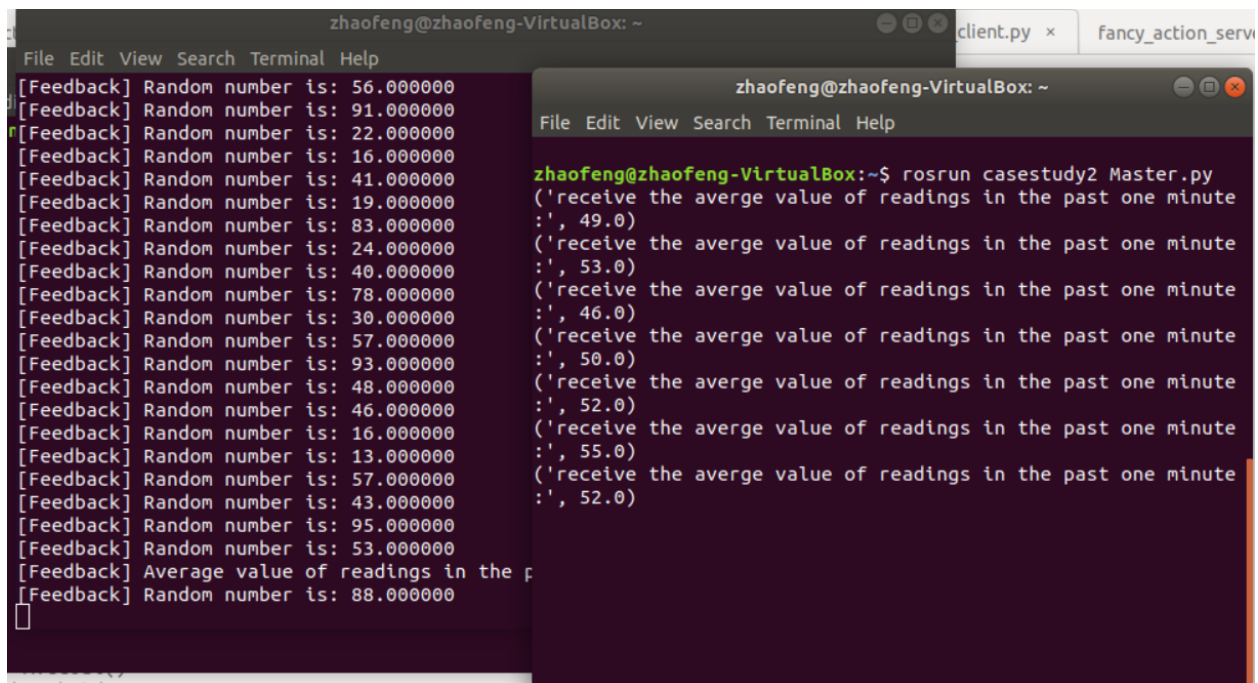
Master:

```
def callback(data):
    print("receive the average value of readings in the past one minute:", data.data)
rospy.init_node('Master')
sub = rospy.Subscriber('counter', Float32, callback)
rospy.spin()
```

Just subscribe to a certain topic, get the average value of random numbers generated in the past minute and print them.

3. Results

- While the average value of five consecutive numbers does not exceed 60, we can get the average value one time per minute in the Master. Below is an example:



The image shows two terminal windows from a VirtualBox environment. The left window displays a series of random numbers generated by a publisher, each preceded by '[Feedback] Random number is:'. The right window shows the output of the Master node, which prints the average value of the readings in the past one minute. The output shows several average values, including 49.0, 53.0, 46.0, 50.0, 52.0, 55.0, and 52.0.

```
zhaofeng@zhaofeng-VirtualBox: ~
File Edit View Search Terminal Help
[Feedback] Random number is: 56.000000
[Feedback] Random number is: 91.000000
[Feedback] Random number is: 22.000000
[Feedback] Random number is: 16.000000
[Feedback] Random number is: 41.000000
[Feedback] Random number is: 19.000000
[Feedback] Random number is: 83.000000
[Feedback] Random number is: 24.000000
[Feedback] Random number is: 40.000000
[Feedback] Random number is: 78.000000
[Feedback] Random number is: 30.000000
[Feedback] Random number is: 57.000000
[Feedback] Random number is: 93.000000
[Feedback] Random number is: 48.000000
[Feedback] Random number is: 46.000000
[Feedback] Random number is: 16.000000
[Feedback] Random number is: 13.000000
[Feedback] Random number is: 57.000000
[Feedback] Random number is: 43.000000
[Feedback] Random number is: 95.000000
[Feedback] Random number is: 53.000000
[Feedback] Average value of readings in the p
[Feedback] Random number is: 88.000000

zhaofeng@zhaofeng-VirtualBox: ~
File Edit View Search Terminal Help
zhaofeng@zhaofeng-VirtualBox:~$ roslaunch casestudy2 Master.py
('receive the average value of readings in the past one minute
:', 49.0)
('receive the average value of readings in the past one minute
:', 53.0)
('receive the average value of readings in the past one minute
:', 46.0)
('receive the average value of readings in the past one minute
:', 50.0)
('receive the average value of readings in the past one minute
:', 52.0)
('receive the average value of readings in the past one minute
:', 55.0)
('receive the average value of readings in the past one minute
:', 52.0)
```

- When the average value of five consecutive numbers exceeds 60, publish the average value immediately. As you can see in the image below, we have

completed 3 circles, which means 3 minutes have past, thus three average values were published to Master. After completing 3 minutes, the next five numbers in the beginning of the fourth minute trigger the exiting process, so we feedback again. Note that the value feedbacked is the average value of these five numbers plus 55 numbers in the third minute. So, we can see four values in the Master, three normal in-loop value, one exiting-loop value. Additionally, the time we get in the server is 184.415 quite matches 3 circles of one minute plus 5 seconds in the fourth circle. The time value being smaller than 185 is a result of that sleep time was set to 0.995 but not 1 exactly.

```

zhaofeng@zhaofeng-VirtualBox: ~
File Edit View Search Terminal Help
[Feedback] Random number is: 77.000000
[Feedback] Random number is: 65.000000
[Feedback] Random number is: 72.000000
[Feedback] Random number is: 39.000000
[Feedback] Random number is: 84.000000
[Feedback] Random number is: 28.000000
[Feedback] Average value of readings in the past one minute: 47.000000
[Feedback] Random number is: 85.000000
[Feedback] Random number is: 67.000000
[Feedback] Random number is: 84.000000
[Feedback] Random number is: 91.000000
[Feedback] Random number is: 100.000000
[Feedback] Random number is: 100.000000
[Feedback] Average value of readings in the past one minute: 50.000000
[Result] State: 4
[Result] Status: Exit calculator because five consecutive readings is 60 or above
[Result] The average value of last five consecutive readings: 85.000000
[Result] How many minutes have past: 3
zhaofeng@zhaofeng-VirtualBox:~$

zhaofeng@zhaofeng-VirtualBox:~$ roslaunch casestudy2 Master.py
('receive the average value of readings in the past one minute:', 54.0)
('receive the average value of readings in the past one minute:', 50.0)
('receive the average value of readings in the past one minute:', 47.0)
('receive the average value of readings in the past one minute:', 50.0)

zhaofeng@zhaofeng-VirtualBox:~$ roslaunch casestudy2 fancy
_action_server.py
49.8513970375
^Czhaofeng@zhaofeng-VirtualBox:~$ roslaunch casestudy2 fancy
_action_server.py
184.414740086

```

- Rqt-graph is shown below:

