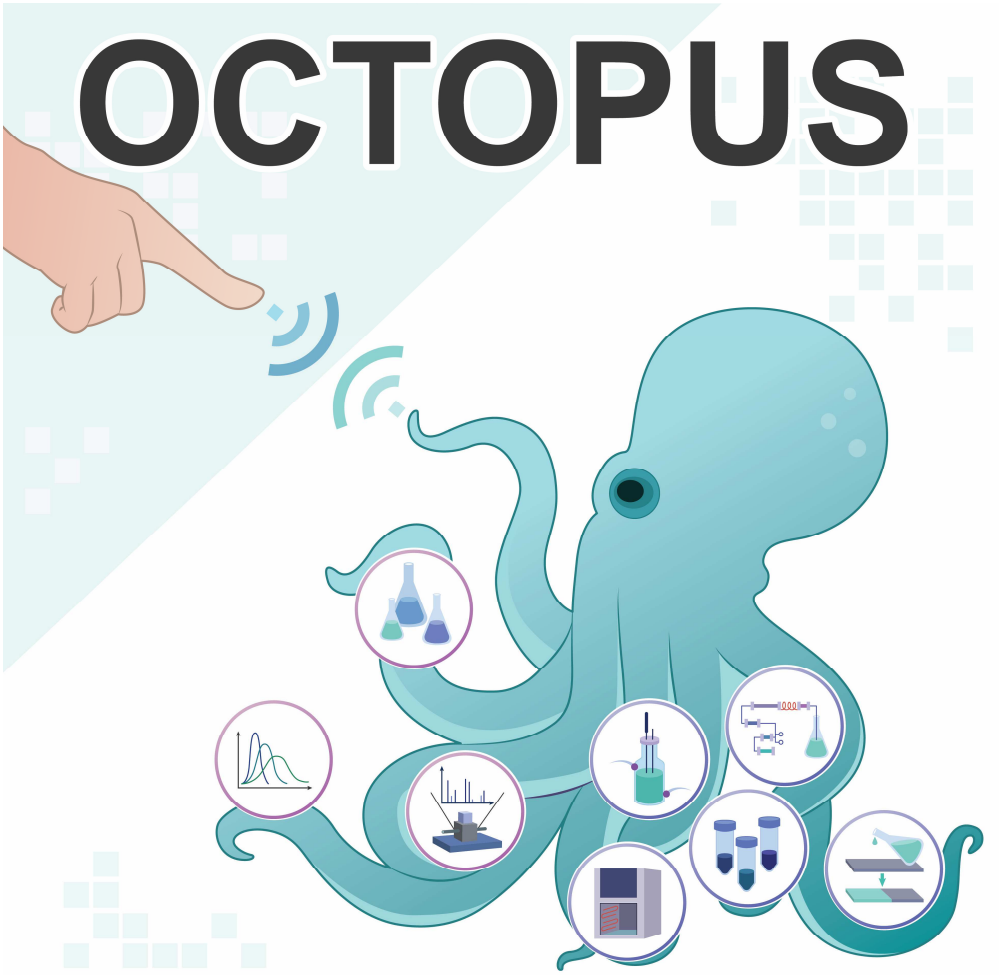
Supplementary Guidelines for using **Copilot of OCTOPUS**

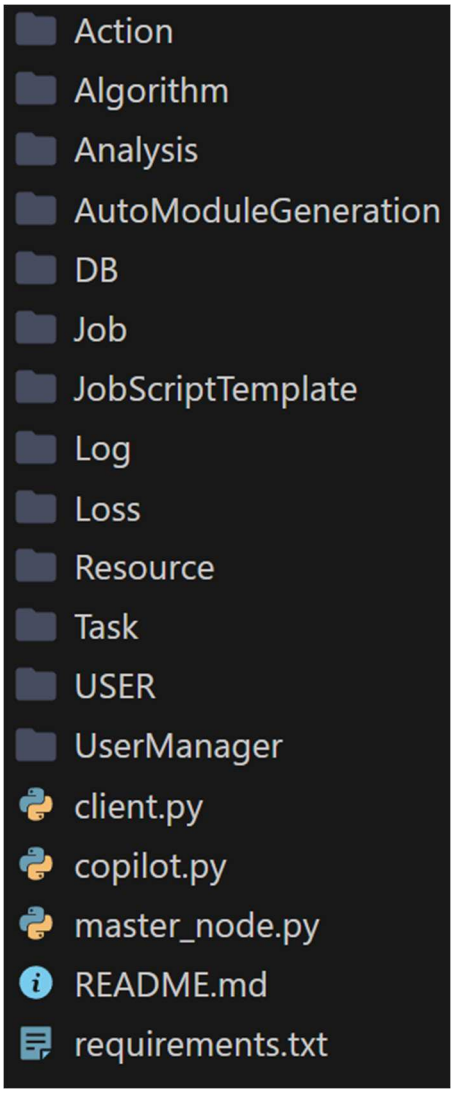
This file provides a detailed and step-by-step protocol to assist potential users in using **Copilot of OCTOPUS** without difficulty.



Hyuk Jun Yoo, Kwan-Young Lee*, Donghun Kim* and Sang Soo Han*

Table of contents

1. Overview: directory description of OCTOPUS



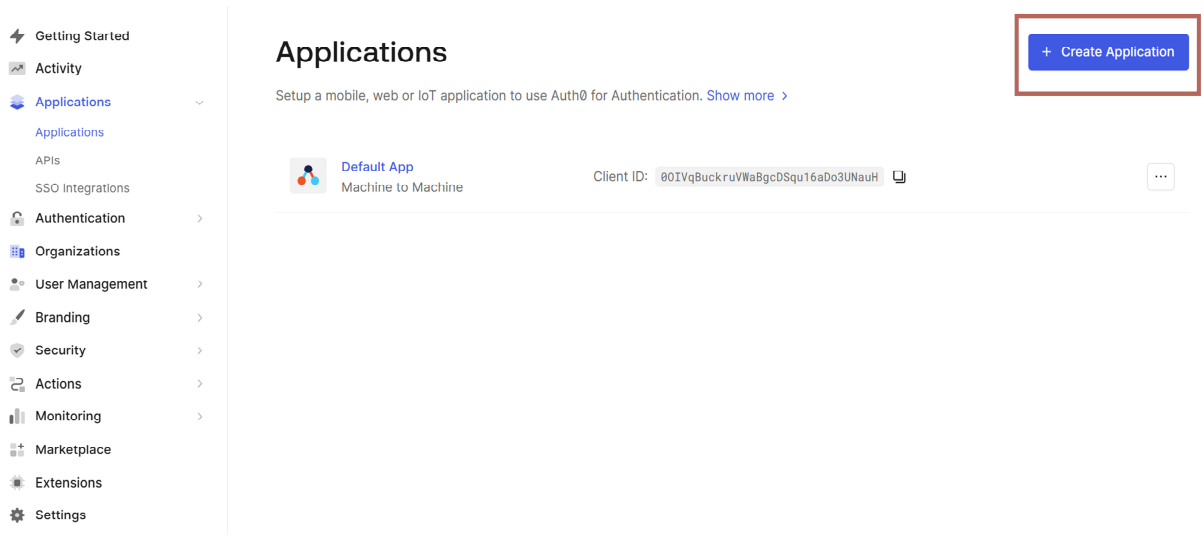
Directory name	Description
Action	Script of ActionTranslator, ActionExecutor
Algorithm	Script of customized AI model
Analysis	Script of customized analysis function from raw spectrum data for each task
AutoModuleGeneration	Script of GPT, feedback system, rule-based generation function
DB	Script of MongoDB manager
Job	Script of JobScheduler, JobTrigger, Job
JobScriptTemplate	JSON file of job script for each module
Log	Script of logger and log files
Loss	Script of loss function
Resource	Script of ResourceManager
Task	Script of TaskGenerator, TaskScheduler, and task template, task Pydantic, device actions for each task
USER	User folder included user of job script, generated data with csv format, log, and pre-trained model object file
UserManager	Script of UserManager for login process with Auth0

2. Prerequisite of installation : Auth0 setting

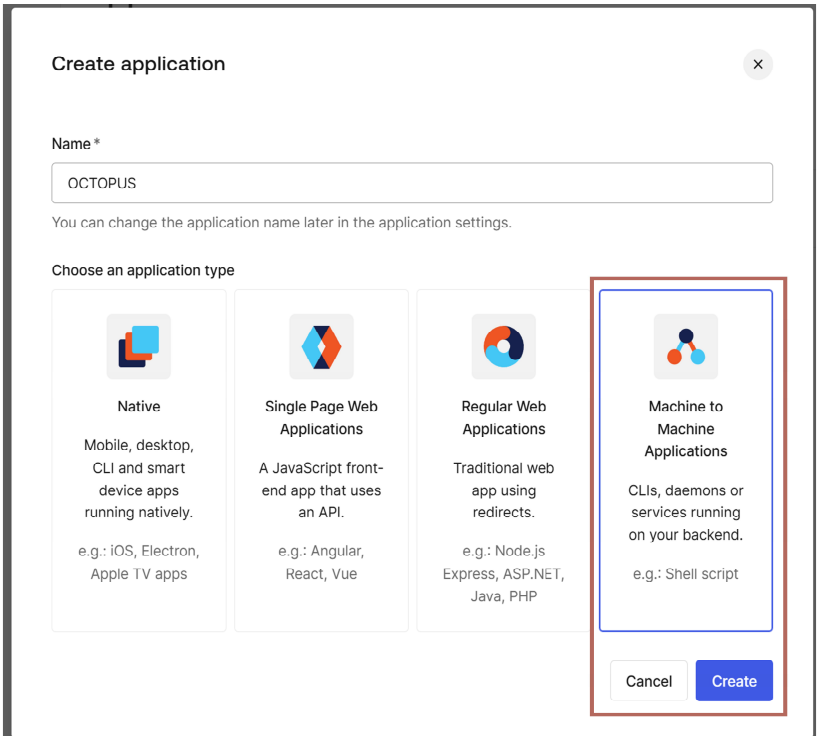
2.1 Sign up Auth0

Please sign up in https://auth0.com/.

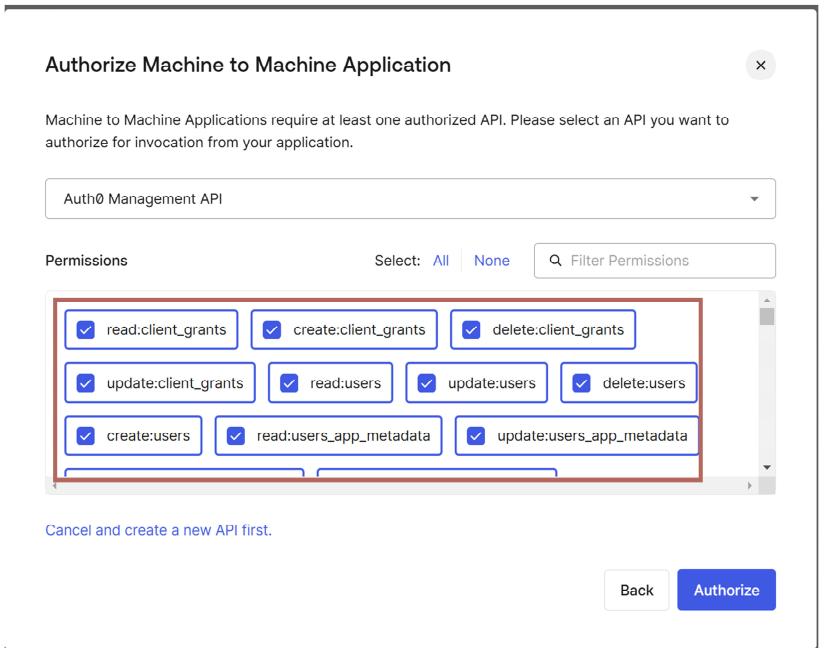
2.2 Create application in Auth0 page



Click “Create Application” and generate application

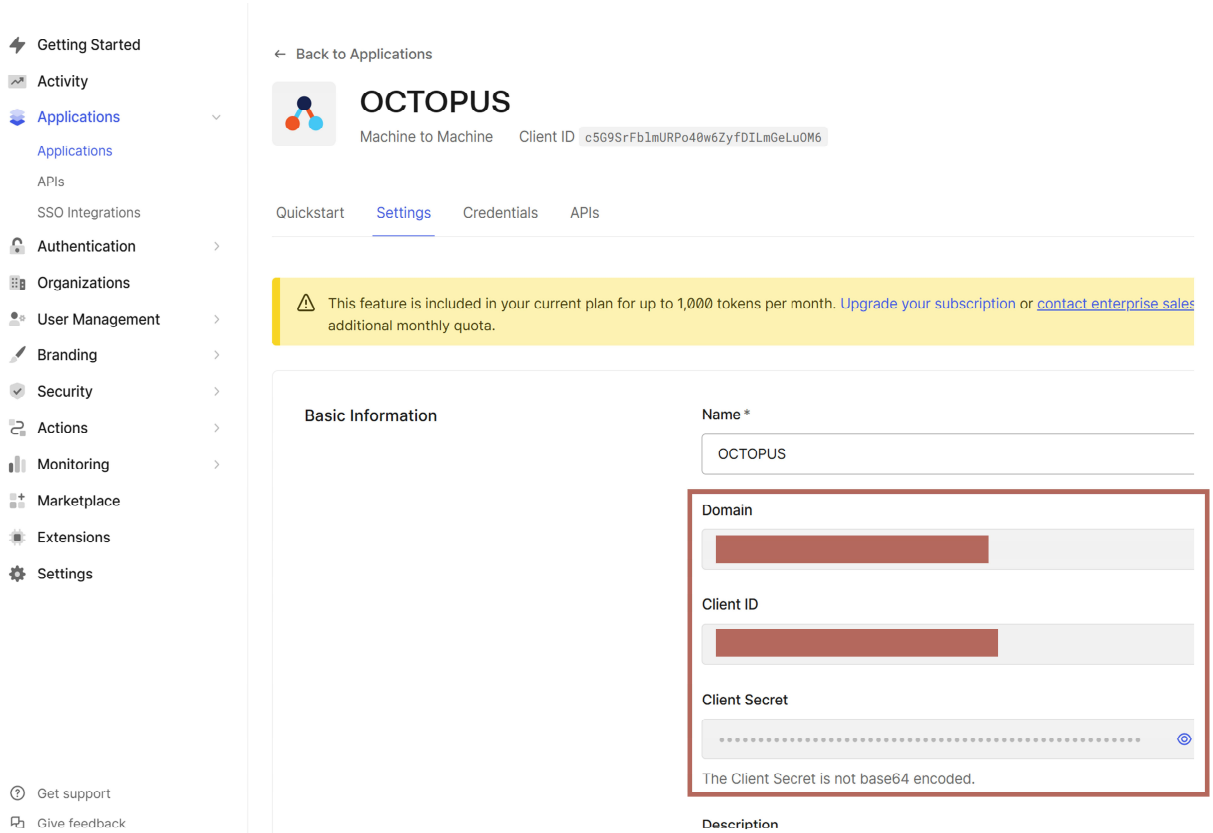


Click “Machine to Machine Applications” button, and click “Create” button.

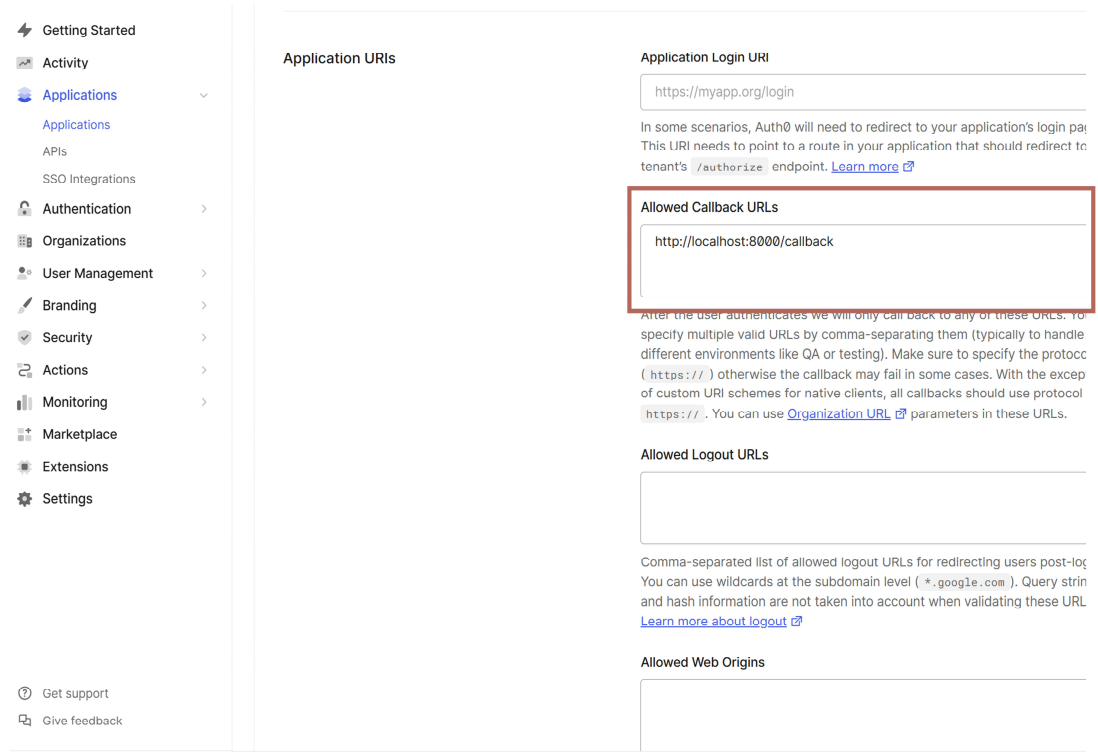


Please click and check all items, and click “Authorize” button.

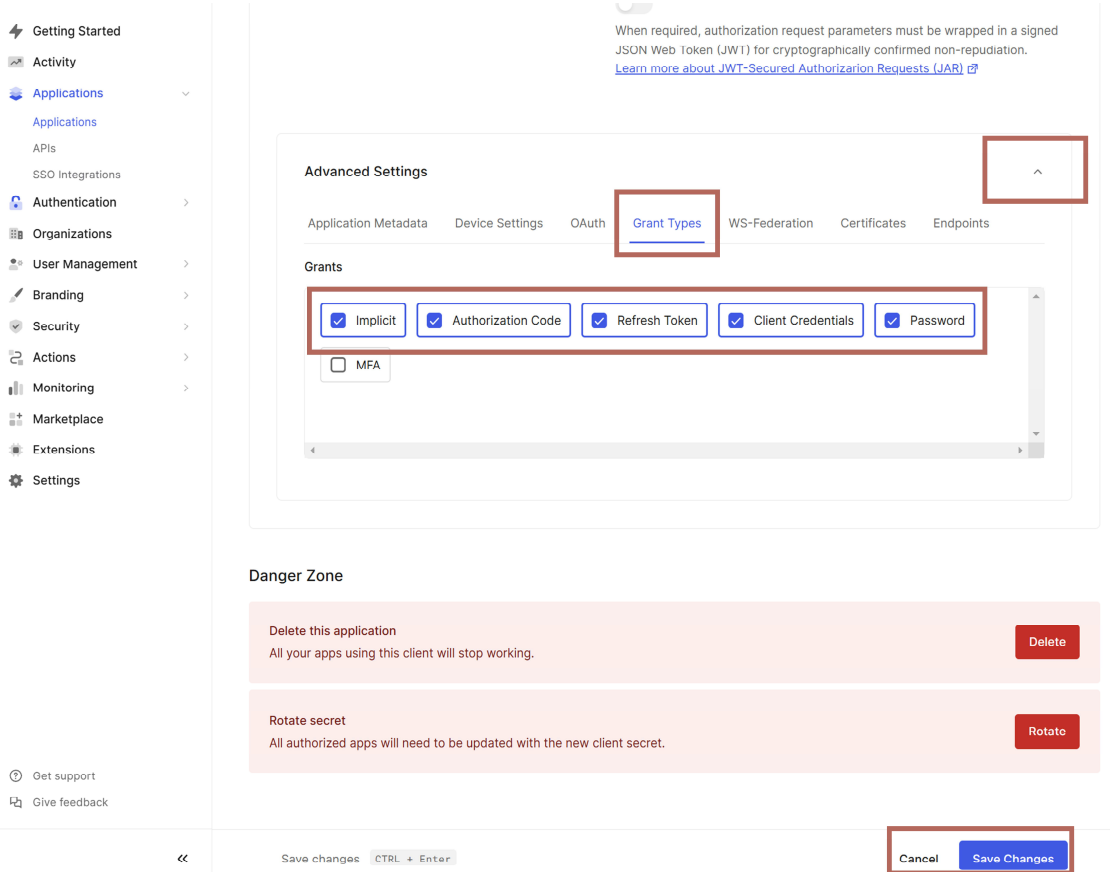
2.4 Set Auth0 administrator page



Please copy the domain, client ID, Client Secret, and save them in another text file. They are different for each user account.

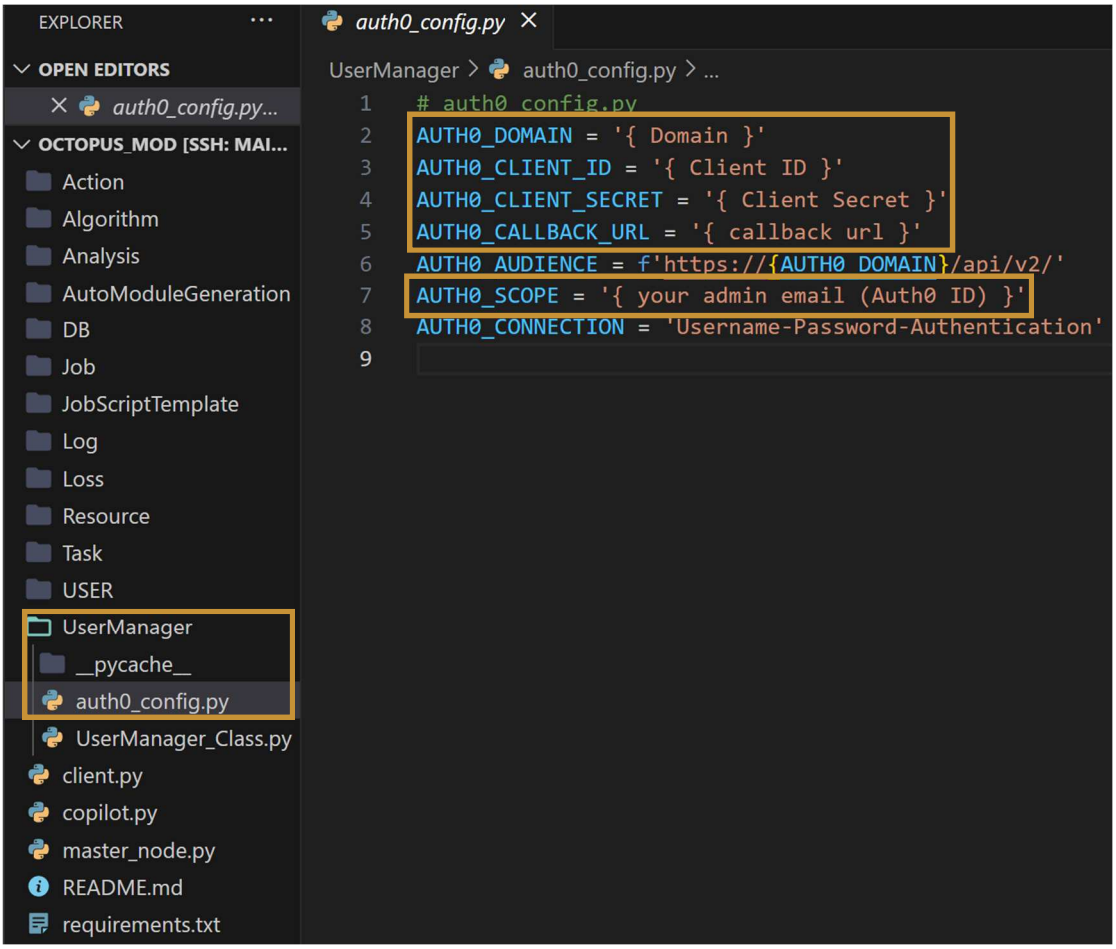


Scroll down to "Application URIs" and enter https://localhost:8000/callback in "Allowed Callback URLs."



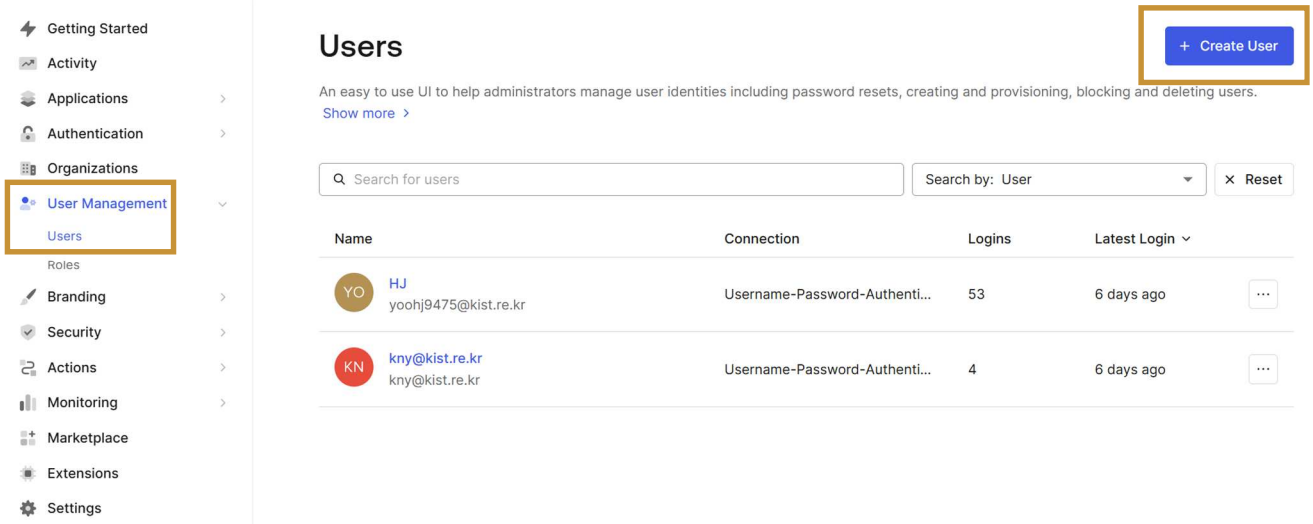
Scroll down to "Advanced Settings," click the "Grant Types" button, and select the items inside the red box.

2.5 Connect Auth0 administrator page to OCTOPUS

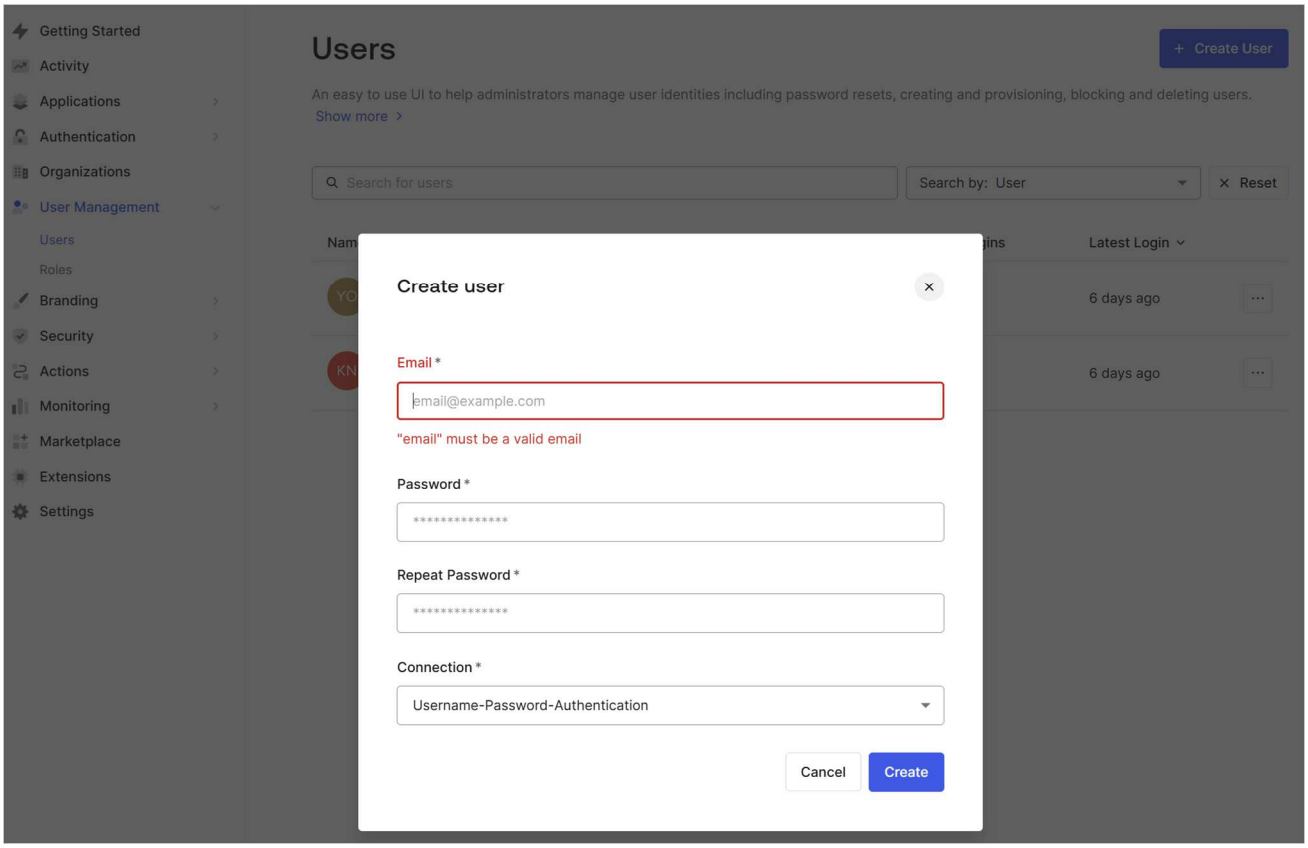


Enter the OCTOPUS folder and open the “UserManager/auth0_config.py” file. In this file, enter the previously saved domain, client ID, client secret, and callback URL in order. In “UserManager/auth0_config.py” file, for "AUTH0_SCOPE," enter the Auth0 account ID in email format.

2.6 Register user in Auth0 page



Click on "Users" under "User Management" in the left toolbar, then click the "Create User" button.



Enter the email and password to be registered, then click the "Create" button to register the user.

3. Installation & environment setting

3.1 Installation

Copy and paste the following git clone command into the terminal window to download the OCTOPUS directory.

```
git clone https://github.com/KIST-CSRC/Octopus.git
```

3.2 Environment setting

Using pip (recommended)

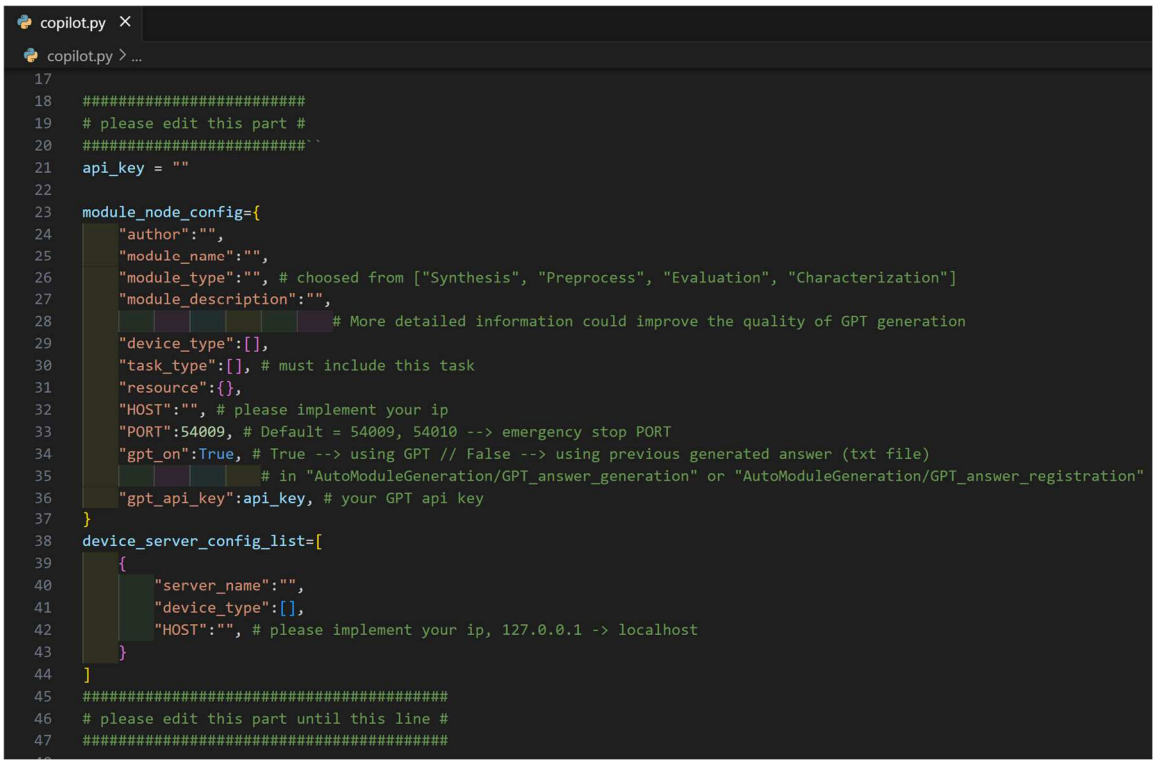
```
pip install -r requirements.txt
```

Using conda

```
conda env create -f requirements.txt
```

4. Copilot of OCTOPUS for module generation and registration

4.1 Set module information in `copilot.py`

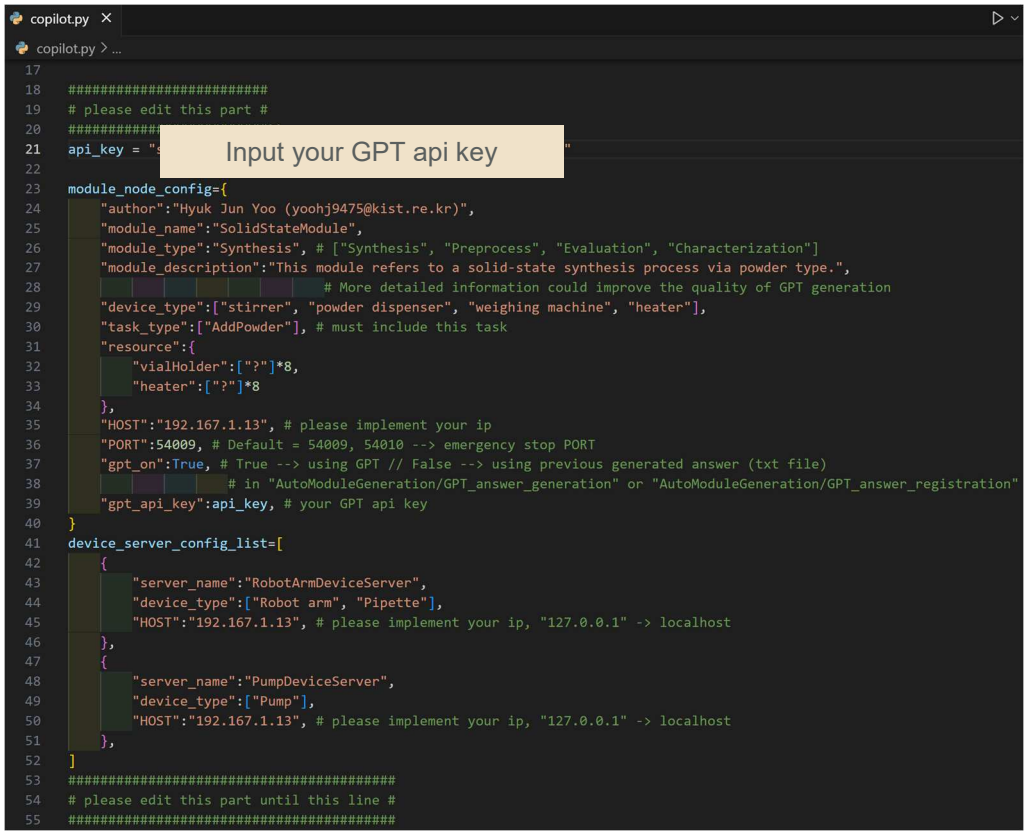


```python
copilot.py
copilot.py > ...
17
18  ########################
19  # please edit this part #
20  ########################``
21  api_key = ""
22
23  module_node_config={
24      "author":"",
25      "module_name":"",
26      "module_type":"", # choosed from ["Synthesis", "Preprocess", "Evaluation", "Characterization"]
27      "module_description":"",
28                          # More detailed information could improve the quality of GPT generation
29      "device_type":[],
30      "task_type":[], # must include this task
31      "resource":{},
32      "HOST":"", # please implement your ip
33      "PORT":54009, # Default = 54009, 54010 --> emergency stop PORT
34      "gpt_on":True, # True --> using GPT // False --> using previous generated answer (txt file)
35                     # in "AutoModuleGeneration/GPT_answer_generation" or "AutoModuleGeneration/GPT_answer_registration"
36      "gpt_api_key":api_key, # your GPT api key
37  }
38  device_server_config_list=[
39      {
40          "server_name":"",
41          "device_type":[],
42          "HOST":"", # please implement your ip, 127.0.0.1 -> localhost
43      }
44  ]
45  ########################################
46  # please edit this part until this line #
47  ########################################
```

The following illustration shows only the lines that the user needs to fill in within the “copilot.py file”. By entering just a few pieces of information, the code for the components owned by the master node will be automatically generated according to the logic explained above.

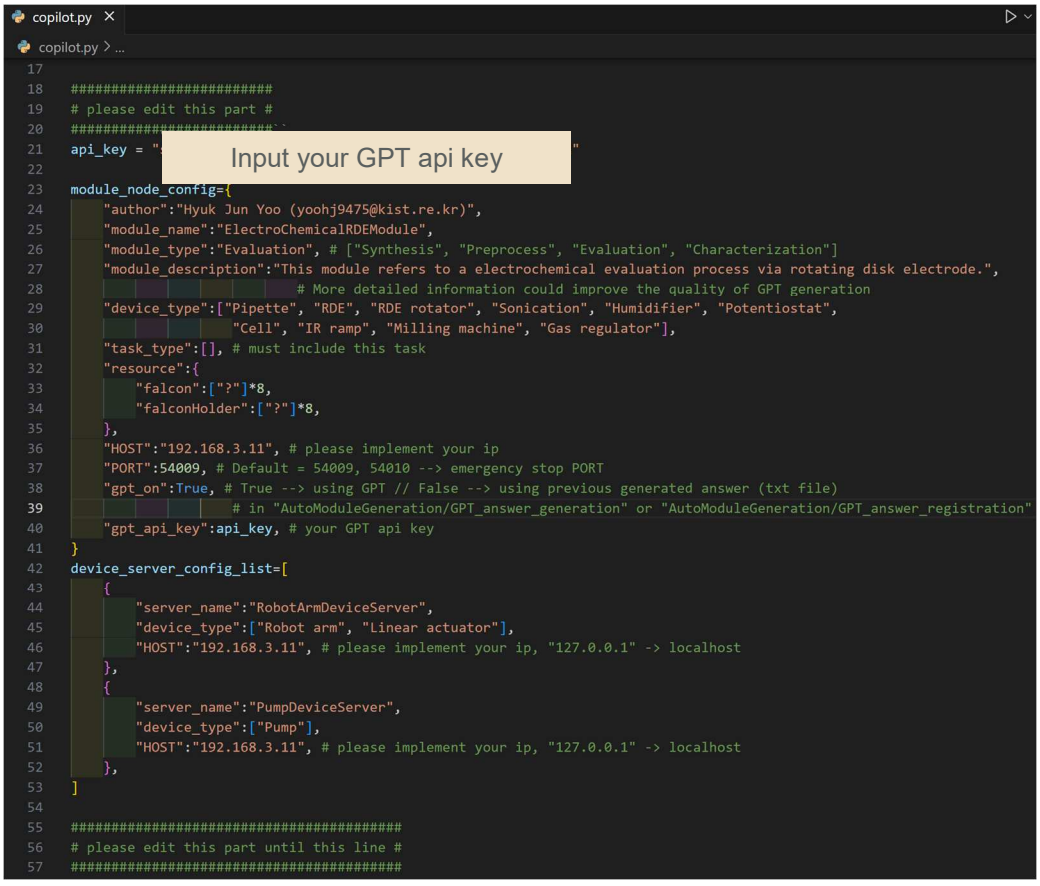
Key of module information	Type	Description of module information
author (=email format).	str	The author name who generated the code for the module node
module_name/server_name	str	The name of module node or device server
module_type	str	The type of module node (please choose from following options: ["Synthesis", "Preprocess", "Evaluation", "Characterization"])
module_description	str	The description of module node (More detailed description could improve the accuracy of code generation.)
device_type	str in list	The names of the devices that need to be registered with the module node
task_type	str in list	The name of the task that must be included in the module node
resource	dict	The resources of the module node
HOST	str	The IP address of the module node
PORT	int	The port number of module node
gpt_on	boolean	Set whether to use the existing GPT answer (=False) , or to ask GPT new answer (True)
gpt_api_key	str	api key of OpenAI

4.1.1 Example of “SolidStateModule”




```python
########################
# please edit this part #
########################
api_key = "Input your GPT api key"

module_node_config={
    "author":"Hyuk Jun Yoo (yoohj9475@kist.re.kr)",
    "module_name":"SolidStateModule",
    "module_type":"Synthesis", # ["Synthesis", "Preprocess", "Evaluation", "Characterization"]
    "module_description":"This module refers to a solid-state synthesis process via powder type.",
                         # More detailed information could improve the quality of GPT generation
    "device_type":["stirrer", "powder dispenser", "weighing machine", "heater"],
    "task_type":["AddPowder"], # must include this task
    "resource":{
        "vialHolder":["?"]*8,
        "heater":["?"]*8
    },
    "HOST":"192.167.1.13", # please implement your ip
    "PORT":54009, # Default = 54009, 54010 --> emergency stop PORT
    "gpt_on":True, # True --> using GPT // False --> using previous generated answer (txt file)
                   # in "AutoModuleGeneration/GPT_answer_generation" or "AutoModuleGeneration/GPT_answer_registration"
    "gpt_api_key":api_key, # your GPT api key
}
device_server_config_list=[
    {
        "server_name":"RobotArmDeviceServer",
        "device_type":["Robot arm", "Pipette"],
        "HOST":"192.167.1.13", # please implement your ip, "127.0.0.1" -> localhost
    },
    {
        "server_name":"PumpDeviceServer",
        "device_type":["Pump"],
        "HOST":"192.167.1.13", # please implement your ip, "127.0.0.1" -> localhost
    },
]
########################################
# please edit this part until this line #
########################################
```

4.1.2 Example of “ElectroChemicalRDEModule”




```python
########################
# please edit this part #
########################
api_key = "Input your GPT api key"

module_node_config={
    "author":"Hyuk Jun Yoo (yoohj9475@kist.re.kr)",
    "module_name":"ElectroChemicalRDEModule",
    "module_type":"Evaluation", # ["Synthesis", "Preprocess", "Evaluation", "Characterization"]
    "module_description":"This module refers to a electrochemical evaluation process via rotating disk electrode.",
                         # More detailed information could improve the quality of GPT generation
    "device_type":["Pipette", "RDE", "RDE rotator", "Sonication", "Humidifier", "Potentiostat",
                   "Cell", "IR ramp", "Milling machine", "Gas regulator"],
    "task_type":[], # must include this task
    "resource":{
        "falcon":["?"]*8,
        "falconHolder":["?"]*8,
    },
    "HOST":"192.168.3.11", # please implement your ip
    "PORT":54009, # Default = 54009, 54010 --> emergency stop PORT
    "gpt_on":True, # True --> using GPT // False --> using previous generated answer (txt file)
                   # in "AutoModuleGeneration/GPT_answer_generation" or "AutoModuleGeneration/GPT_answer_registration"
    "gpt_api_key":api_key, # your GPT api key
}
device_server_config_list=[
    {
        "server_name":"RobotArmDeviceServer",
        "device_type":["Robot arm", "Linear actuator"],
        "HOST":"192.168.3.11", # please implement your ip, "127.0.0.1" -> localhost
    },
    {
        "server_name":"PumpDeviceServer",
        "device_type":["Pump"],
        "HOST":"192.168.3.11", # please implement your ip, "127.0.0.1" -> localhost
    },
]

########################################
# please edit this part until this line #
########################################
```

If the configuration for the module node is complete, please execute the following command to run the copilot of OCTOPUS:

```
python copilot.py
```

4.2 Action generation of experimental device in module node and device server

Below is the result of inputting the devices connected to the actual SolidStateModule into GPT to get recommendations for the actions of each device.

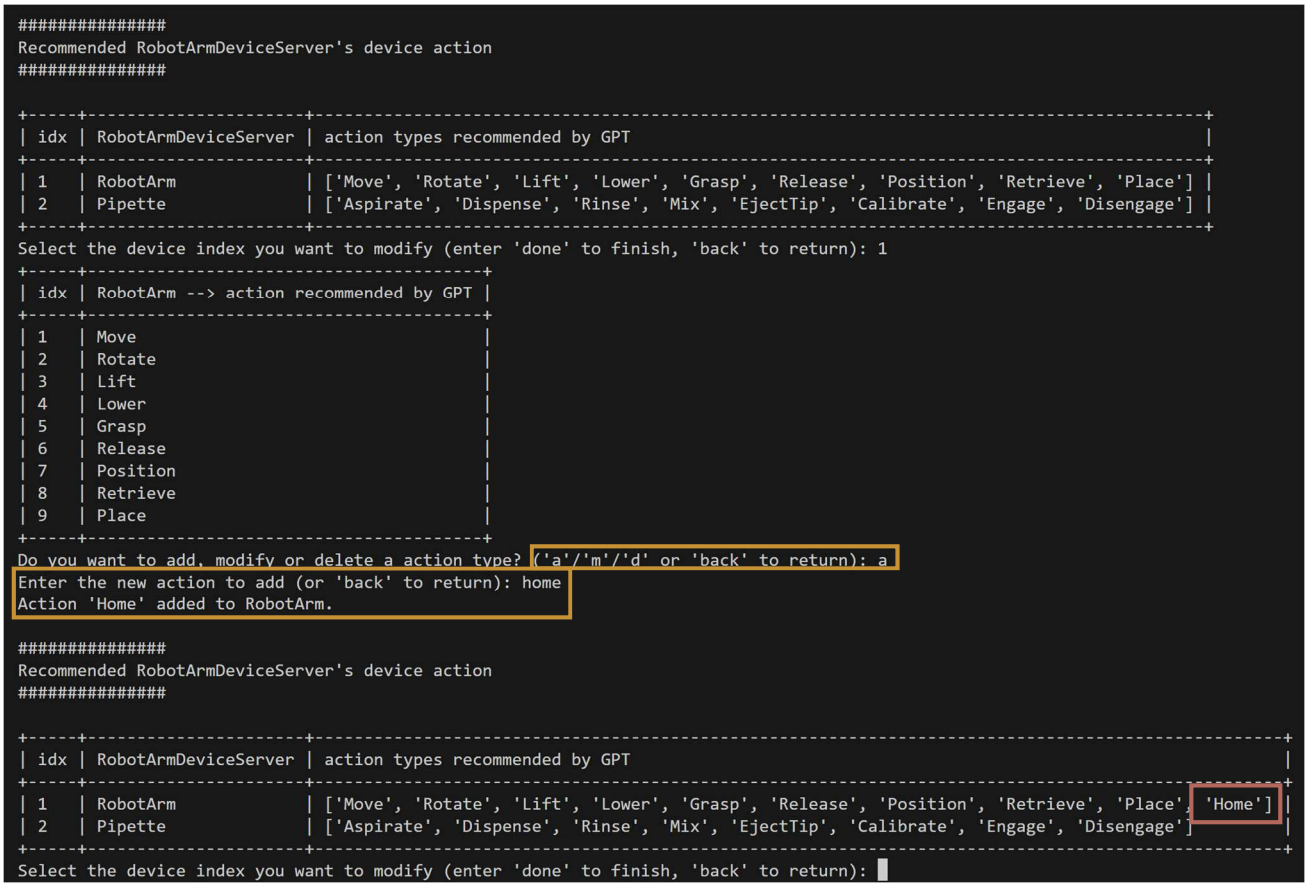
```
###################
SolidStateModule module generation started!
###################
###################
[GPT (ModuleNode or DeviceServer)]: action generation for devices...
###################
{
  "prompt_tokens": 152,
  "completion_tokens": 80,
  "total_tokens": 232
}

###################
[Feedback system (action generation)]
###################

###############
Recommended RobotArmDeviceServer's device action
###############

+-----+----------------------+-----------------------------------------------------------------------------------------------+
| idx | RobotArmDeviceServer | action types recommended by GPT                                                               |
+-----+----------------------+-----------------------------------------------------------------------------------------------+
| 1   | RobotArm             | ['Move', 'Rotate', 'Lift', 'Lower', 'Grasp', 'Release', 'Position', 'Retrieve', 'Place']      |
| 2   | Pipette              | ['Aspirate', 'Dispense', 'Rinse', 'Mix', 'EjectTip', 'Calibrate', 'Engage', 'Disengage']      |
+-----+----------------------+-----------------------------------------------------------------------------------------------+
Select the device index you want to modify (enter 'done' to finish, 'back' to return):
```

However, since the available functions vary by manufacturer, the user needs to add, modify, or delete them in command line interface. Therefore, you can enter the index of the device to add, modify, or delete its actions. Once all actions are correctly entered, you can press 'done' to finish editing the actions.

```
###############
Recommended RobotArmDeviceServer's device action
###############

+-----+----------------------+-----------------------------------------------------------------------------------------------+
| idx | RobotArmDeviceServer | action types recommended by GPT                                                               |
+-----+----------------------+-----------------------------------------------------------------------------------------------+
| 1   | RobotArm             | ['Move', 'Rotate', 'Lift', 'Lower', 'Grasp', 'Release', 'Position', 'Retrieve', 'Place']      |
| 2   | Pipette              | ['Aspirate', 'Dispense', 'Rinse', 'Mix', 'EjectTip', 'Calibrate', 'Engage', 'Disengage']      |
+-----+----------------------+-----------------------------------------------------------------------------------------------+
Select the device index you want to modify (enter 'done' to finish, 'back' to return): 1
+-----+----------------------------------------+
| idx | RobotArm --> action recommended by GPT |
+-----+----------------------------------------+
| 1   | Move                                   |
| 2   | Rotate                                 |
| 3   | Lift                                   |
| 4   | Lower                                  |
| 5   | Grasp                                  |
| 6   | Release                                |
| 7   | Position                               |
| 8   | Retrieve                               |
| 9   | Place                                  |
+-----+----------------------------------------+
Do you want to add, modify or delete a action_type? ('a'/'m'/'d' or 'back' to return):
```

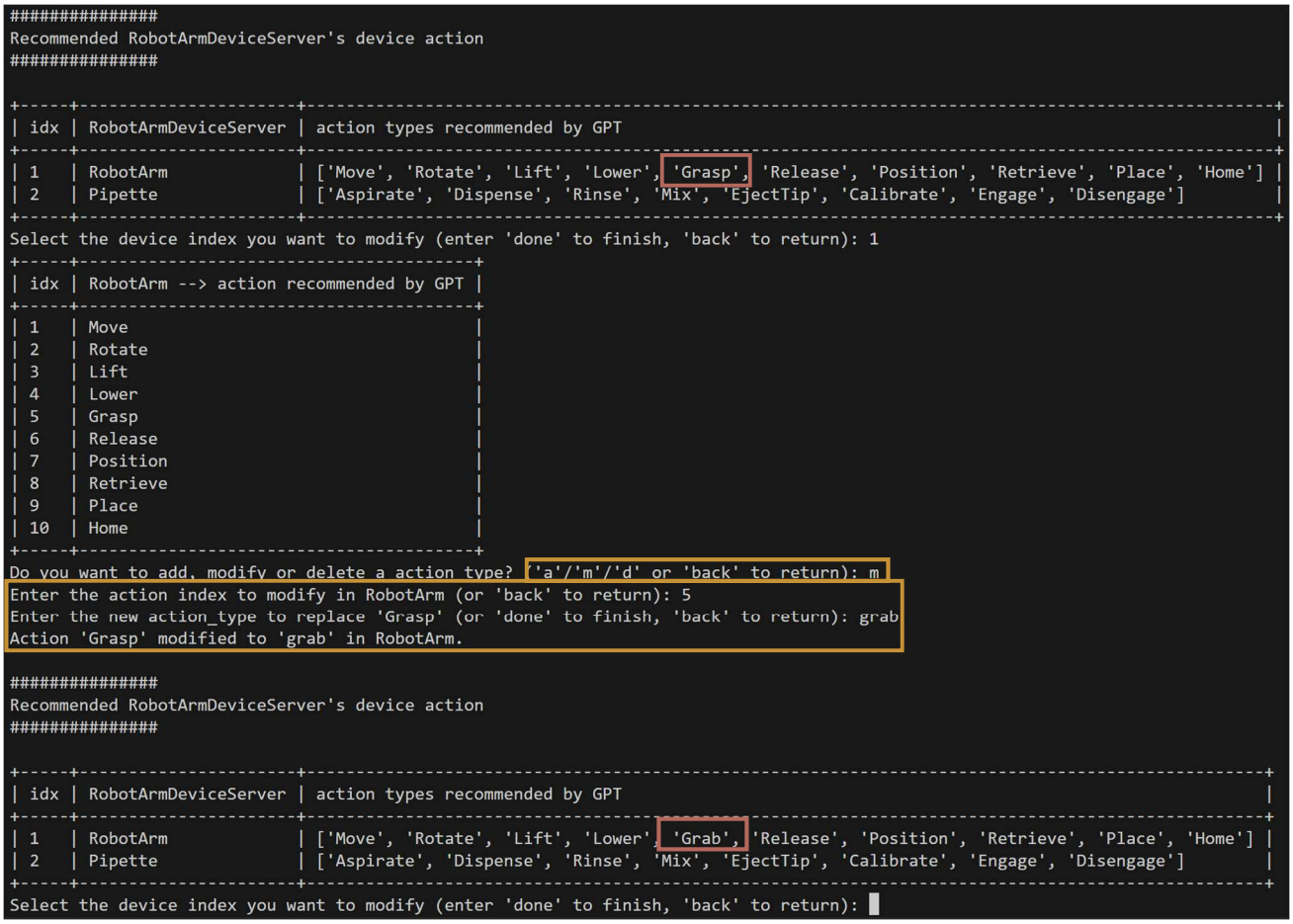
If you enter index 1 to modify the actions of 'RobotArm', all the actions of 'RobotArm' will be displayed in a table format, as shown in the illustration above.

4.2.1 Add new action



If the client inputs 'a' to add a new action, you need to enter the name of the new action in command line interface. In the figure above, an action called 'Home,' which moves the robot arm to a predetermined position, was added. Once this task is completed, you can see that 'Home' in red box has been added to the actions of 'RobotArm'.

4.2.2 Modify action



If the client inputs 'm' to modify an action, you need to enter both the existing action name and the new action name in command line interface. In the illustration above, the action 'Grasp,' which is the movement of the robot arm's gripper to pick up an object, was modified to 'Grab.' Once this task is completed, you can see that 'Grasp' has been changed to 'Grab' in the actions of 'RobotArm', via red boxes.

4.2.3 Delete action

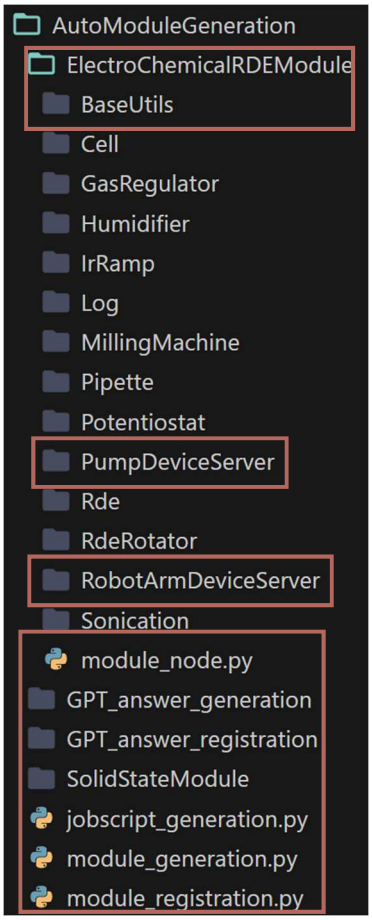
```
###############
Recommended RobotArmDeviceServer's device action
###############

+-----+----------------------+-----------------------------------------------------------------------------------------------------+
| idx | RobotArmDeviceServer | action types recommended by GPT                                                                     |
+-----+----------------------+-----------------------------------------------------------------------------------------------------+
| 1   | RobotArm             | ['Move', 'Rotate', 'Lift', 'Lower', 'Grab', 'Release', 'Position', 'Retrieve', 'Place', 'Home']     |
| 2   | Pipette              | ['Aspirate', 'Dispense', 'Rinse', 'Mix', 'EjectTip', 'Calibrate', 'Engage', 'Disengage']            |
+-----+----------------------+-----------------------------------------------------------------------------------------------------+
Select the device index you want to modify (enter 'done' to finish, 'back' to return): 1
+-----+----------------------------------------+
| idx | RobotArm --> action recommended by GPT |
+-----+----------------------------------------+
| 1   | Move                                   |
| 2   | Rotate                                 |
| 3   | Lift                                   |
| 4   | Lower                                  |
| 5   | Grab                                   |
| 6   | Release                                |
| 7   | Position                               |
| 8   | Retrieve                               |
| 9   | Place                                  |
| 10  | Home                                   |
+-----+----------------------------------------+
Do you want to add, modify or delete a action_type? ('a'/'m'/'d' or 'back' to return): d
Enter the action index to delete in RobotArm (or 'back' to return): 8
Are you sure you want to delete action 'Retrieve'? ('y'/'n', 'back' to return): y
Action 'Retrieve' deleted from RobotArm.

###############
Recommended RobotArmDeviceServer's device action
###############

+-----+----------------------+-----------------------------------------------------------------------------------------------------+
| idx | RobotArmDeviceServer | action types recommended by GPT                                                                     |
+-----+----------------------+-----------------------------------------------------------------------------------------------------+
| 1   | RobotArm             | ['Move', 'Rotate', 'Lift', 'Lower', 'Grab', 'Release', 'Position', 'Place', 'Home']                  |
| 2   | Pipette              | ['Aspirate', 'Dispense', 'Rinse', 'Mix', 'EjectTip', 'Calibrate', 'Engage', 'Disengage']            |
+-----+----------------------+-----------------------------------------------------------------------------------------------------+
Select the device index you want to modify (enter 'done' to finish, 'back' to return):
```

If the client inputs 'd' to delete an existing action, you need to enter the name of the action to be deleted in the command line interface. In the illustration above, the action 'Retrieve,' which involves the robot arm retrieving a specific object, cannot be performed by the currently registered module node. Therefore, the 'Retrieve' action was deleted. Once this task is completed, you can see that 'Retrieve' has been removed from the actions of 'RobotArm.'

```
####################
SolidStateModule module generation completed!
####################
####################
SolidStateModule module registration started!
####################
####################
[SolidStateModule]: get all action type of device from SolidStateModule
####################
+-----+-----------------+-----------------------------------------------------------------------------------------+
| idx | device name     | SolidStateModule action_type                                                            |
+-----+-----------------+-----------------------------------------------------------------------------------------+
| 1   | RobotArm        | ['Move', 'Rotate', 'Lift', 'Lower', 'Grab', 'Release', 'Position', 'Place', 'Home']     |
| 2   | Pipette         | ['Aspirate', 'Dispense', 'Rinse', 'Mix', 'Eject', 'Calibrate']                          |
| 3   | Pump            | ['Initialize', 'Flush', 'Inject']                                                       |
| 4   | Stirrer         | ['Stir', 'Stop']                                                                        |
| 5   | PowderDispenser | ['Dispense', 'Stop']                                                                    |
| 6   | WeighingMachine | ['Weigh', 'Tare', 'Stop']                                                               |
| 7   | Heater          | ['Heat', 'Cool', 'Stop']                                                                |
+-----+-----------------+-----------------------------------------------------------------------------------------+
```

If all device actions have been configured, all the actions of the devices will be displayed in a table format, as shown in the figure above.

4.2.4 Code automated generation of module node and device server



Directory name	Description
ElectroChemicalRDEModule	Script for module node (RDE process)
└─BaseUtils	Functions for read/write of JSON file and socket communication
└─Log	Script of logger and log files
└─PumpDeviceServer	Script of controlling pump as device server
└─RobotArmDeviceServer	Script of controlling robotic arm as device server
└─module_node.py	Interface for controlling devices by communicating with the master node
GPT_answer_generation	A text file containing GPT-based recommended actions for the devices saved in this directory
GPT_answer_registration	A text file containing GPT answers to questions related to the master node saved in this directory
SolidStateModule	Script for module node (solid-state process)
jobscript_generation.py	Script for automated generation of job script template in 'JobScriptTemplate' directory
module_generation.py	Script of automated generation for module node
module_registration.py	Script of automated generation for registration in master node

4.3 Task generation of module node

```
####################
[GPT (TaskGenerator, ActionTranslator)]: task generation...
####################
{
  "prompt_tokens": 188,
  "completion_tokens": 388,
  "total_tokens": 576
}

####################
[Feedback system (task generation)]: current task List
####################
+-----+-----------------------------------+
| idx | current task name                 |
+-----+-----------------------------------+
| 1   | SolidStateModule_LoadPowder       |
| 2   | SolidStateModule_AddPowder        |
| 3   | SolidStateModule_MixPowders       |
| 4   | SolidStateModule_PressPowder      |
| 5   | SolidStateModule_Calcine          |
| 6   | SolidStateModule_Sinter           |
| 7   | SolidStateModule_Cool             |
| 8   | SolidStateModule_Grind            |
| 9   | SolidStateModule_Pelletize        |
| 10  | SolidStateModule_MeasureMass      |
| 11  | SolidStateModule_CharacterizeTest |
| 12  | SolidStateModule_DensityTest      |
| 13  | SolidStateModule_PorosityTest     |
| 14  | SolidStateModule_PhaseAnalysisTest|
+-----+-----------------------------------+
Do you want to add, modify or delete this task? ('a'/'m'/'d') (or 'done' to finish):
```

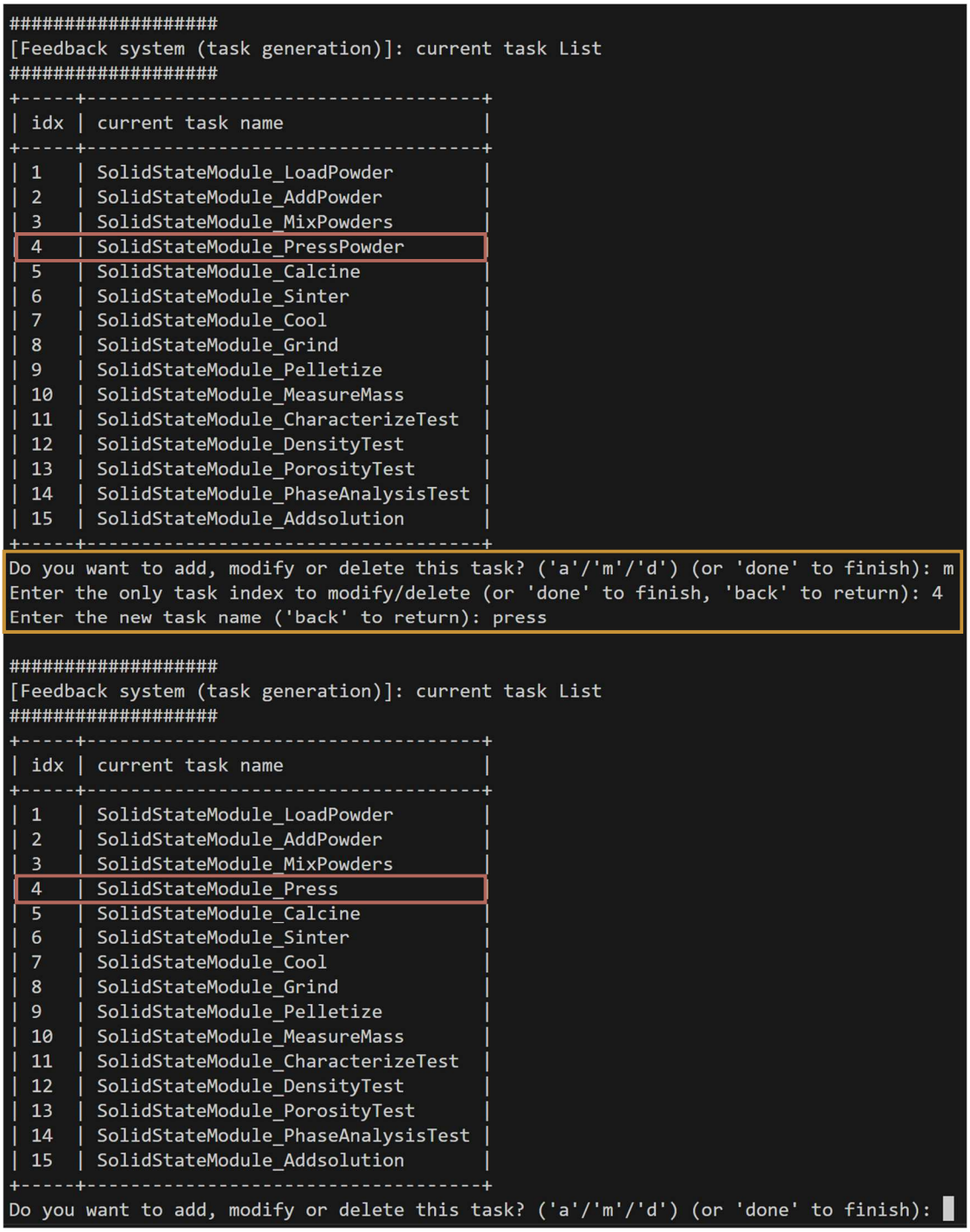
After configuring the device actions, GPT will recommend tasks to perform the module, highlighted in the orange box in the illustration above. Among the tasks recommended by GPT, the researcher can add, modify, or delete task names through the feedback system.

4.3.1 Add new task

```
###################
[Feedback system (task generation)]: current task List
###################
+-----+-----------------------------------+
| idx | current task name                 |
+-----+-----------------------------------+
| 1   | SolidStateModule_LoadPowder       |
| 2   | SolidStateModule_AddPowder        |
| 3   | SolidStateModule_MixPowders       |
| 4   | SolidStateModule_PressPowder      |
| 5   | SolidStateModule_Calcine          |
| 6   | SolidStateModule_Sinter           |
| 7   | SolidStateModule_Cool             |
| 8   | SolidStateModule_Grind            |
| 9   | SolidStateModule_Pelletize        |
| 10  | SolidStateModule_MeasureMass      |
| 11  | SolidStateModule_CharacterizeTest |
| 12  | SolidStateModule_DensityTest      |
| 13  | SolidStateModule_PorosityTest     |
| 14  | SolidStateModule_PhaseAnalysisTest|
+-----+-----------------------------------+
Do you want to add, modify or delete this task? ('a'/'m'/'d') (or 'done' to finish): a
Enter the new task name ('back' to return): AddSolution

###################
[Feedback system (task generation)]: current task List
###################
+-----+-----------------------------------+
| idx | current task name                 |
+-----+-----------------------------------+
| 1   | SolidStateModule_LoadPowder       |
| 2   | SolidStateModule_AddPowder        |
| 3   | SolidStateModule_MixPowders       |
| 4   | SolidStateModule_PressPowder      |
| 5   | SolidStateModule_Calcine          |
| 6   | SolidStateModule_Sinter           |
| 7   | SolidStateModule_Cool             |
| 8   | SolidStateModule_Grind            |
| 9   | SolidStateModule_Pelletize        |
| 10  | SolidStateModule_MeasureMass      |
| 11  | SolidStateModule_CharacterizeTest |
| 12  | SolidStateModule_DensityTest      |
| 13  | SolidStateModule_PorosityTest     |
| 14  | SolidStateModule_PhaseAnalysisTest|
| 15  | SolidStateModule_Addsolution      |
+-----+-----------------------------------+
Do you want to add, modify or delete this task? ('a'/'m'/'d') (or 'done' to finish):
```

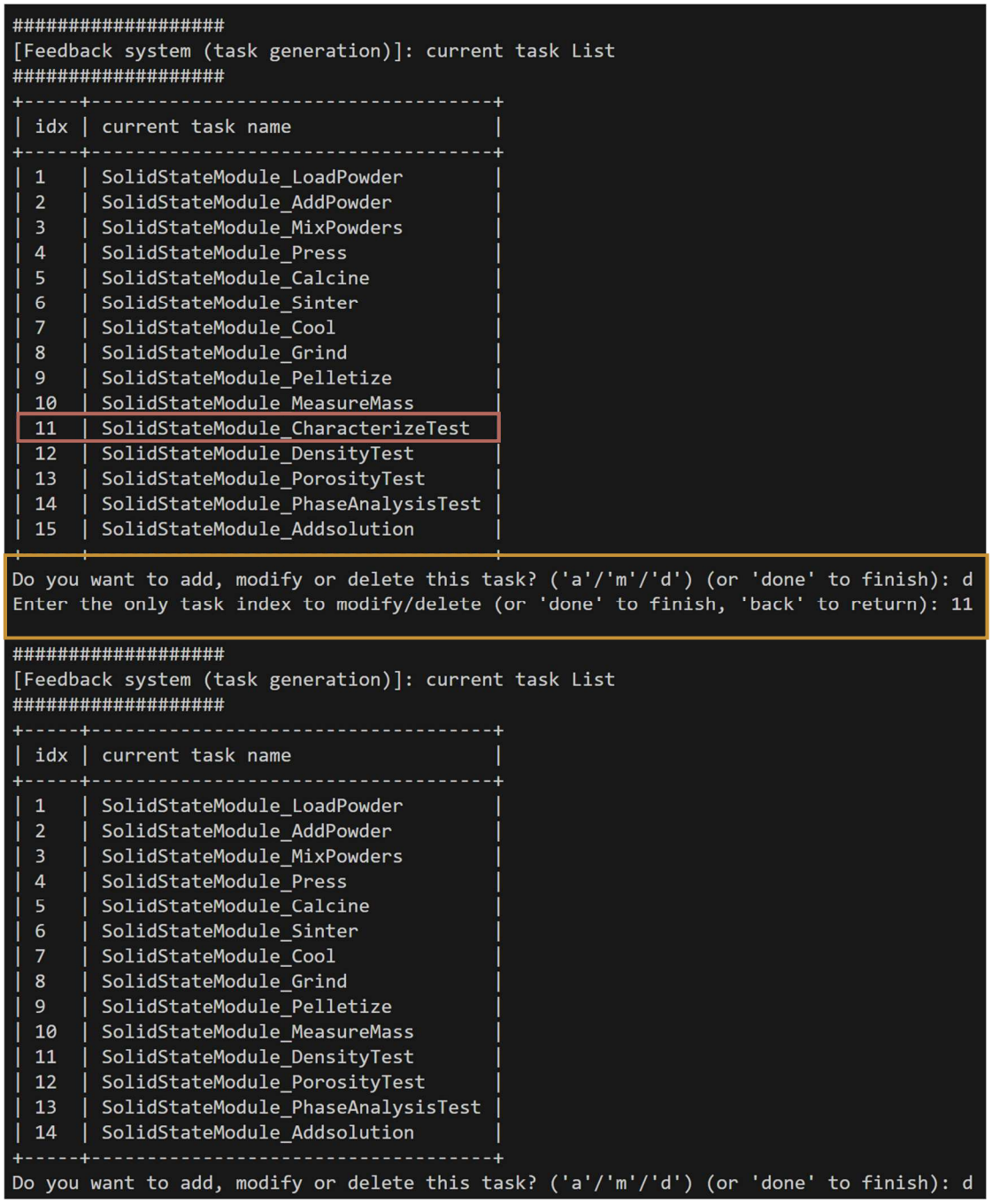
If the client inputs 'a' to add a new task in command line interface, you need to enter the name of the new task in yellow box. In the figure above, a task called 'AddSolution,' which involves adding a solution, was registered. When entering the task name, the module name should be omitted, and only the task name should be inputted. Once this process is completed, you will see 'SolidStateModule_AddSolution' added to the list of tasks in red box.

4.3.2 Modify task



If the client inputs 'm' to modify an existing task in command line interface, you need to enter both the task index and the new task name in yellow box. In the figure above, the task 'PressPowder,' which involves molding powder, was modified to 'Press.' Once this task is completed, you will see that 'SolidStateModule_PressPowder' has been changed to 'SolidStateModule_Press' in red boxes.

4.3.3 Delete task



If the client inputs 'd' to delete an existing task, you need to enter the index of the task to be deleted in the command line interface. In the illustration above, the task 'CharacterizeTest' generated by GPT could not be performed, so the task was deleted. Once all tasks are completed, you will see that 'SolidStateModule_CharacterizeTest' has been removed.

```
###################
[Feedback system (task generation)]: current task List
###################
+-----+-----------------------------+
| idx | current task name           |
+-----+-----------------------------+
| 1   | SolidStateModule_LoadPowder |
| 2   | SolidStateModule_AddPowder  |
| 3   | SolidStateModule_MixPowders |
| 4   | SolidStateModule_Press      |
| 5   | SolidStateModule_Calcine    |
| 6   | SolidStateModule_Sinter     |
| 7   | SolidStateModule_Cool       |
| 8   | SolidStateModule_Grind      |
| 9   | SolidStateModule_Pelletize  |
| 10  | SolidStateModule_MeasureMass|
| 11  | SolidStateModule_Addsolution|
+-----+-----------------------------+
```

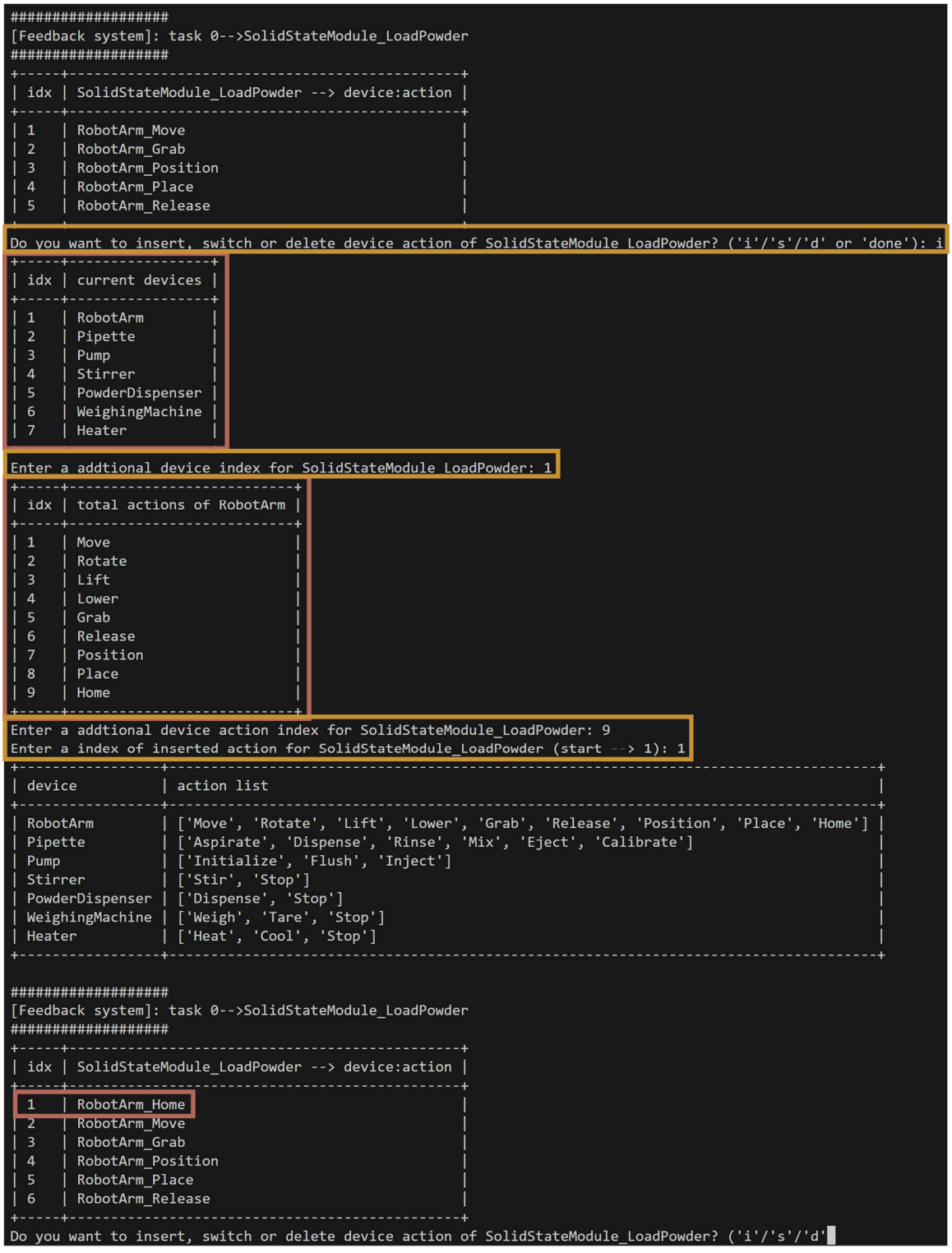
If all process is done, client input ‘done’ in command line interface. This figure show all task of ‘SolidStateModule’.

4.4 Match action sequence for task execution in module node

```
###################
[GPT (ActionTranslator)]: match device-action combination for each task...
###################
{
  "prompt_tokens": 1184,
  "completion_tokens": 1001,
  "total_tokens": 2185
}
[Feedback system (match task-->device-action)]: Devices registration for SolidStateModule_LoadPowder (type 'done' when finished)
+----------------+-------------------------------------------------------------------------------------+
| device         | action list                                                                         |
+----------------+-------------------------------------------------------------------------------------+
| RobotArm       | ['Move', 'Rotate', 'Lift', 'Lower', 'Grab', 'Release', 'Position', 'Place', 'Home'] |
| Pipette        | ['Aspirate', 'Dispense', 'Rinse', 'Mix', 'Eject', 'Calibrate']                      |
| Pump           | ['Initialize', 'Flush', 'Inject']                                                   |
| Stirrer        | ['Stir', 'Stop']                                                                    |
| PowderDispenser| ['Dispense', 'Stop']                                                                |
| WeighingMachine| ['Weigh', 'Tare', 'Stop']                                                           |
| Heater         | ['Heat', 'Cool', 'Stop']                                                            |
+----------------+-------------------------------------------------------------------------------------+

###################
[Feedback system]: task 0-->SolidStateModule_LoadPowder
###################
+-----+-----------------------------------------------+
| idx | SolidStateModule_LoadPowder --> device:action |
+-----+-----------------------------------------------+
| 1   | RobotArm_Move                                 |
| 2   | RobotArm_Grab                                 |
| 3   | RobotArm_Position                             |
| 4   | RobotArm_Place                                |
| 5   | RobotArm_Release                              |
+-----+-----------------------------------------------+
Do you want to insert, switch or delete device action of SolidStateModule_LoadPowder? ('i'/'s'/'d' or 'done'):
```

After configuring the tasks, GPT recommends combinations and sequences of actions to perform each task. The red box in the figure shows the recommended action combination and sequence for 'LoadPowder' via GPT. Unlike action and task generation, configuring the action sequence is crucial for performing chemical experiment tasks correctly. Therefore, researchers can use the feedback logic to insert new actions in action sequence, change the sequence, or delete actions based on GPT's recommended sequence. This feedback system could provide the correct action sequence.

4.4.1 Insert new action in action sequence



If the client inputs 'i' to insert a new action into the sequence, the feedback system will first ask for the type of device to be added. The client will enter the device's index to be added, and then the system will ask for the type of action for that device. When the client inputs the action's index, the feedback system will ask for the index of the position where the action should be inserted in the action sequence. The position index starts at 1, and if the client types 1, the action will be inserted at the very beginning. For example, if 1 is entered, the 'Home' action of 'RobotArm' will be inserted at the first of the action sequence, as shown in the figure.

4.4.2 Switch two actions in action sequence

```
###################
[Feedback system]: task 0-->SolidStateModule_LoadPowder
###################
+-----+-----------------------------------------------+
| idx | SolidStateModule_LoadPowder --> device:action |
+-----+-----------------------------------------------+
| 1   | RobotArm_Home                                 |
| 2   | RobotArm_Move                                 |
| 3   | RobotArm_Grab                                 |
| 4   | RobotArm_Position                             |
| 5   | RobotArm_Place                                |
| 6   | RobotArm_Release                              |
+-----+-----------------------------------------------+
Do you want to insert, switch or delete device action of SolidStateModule_LoadPowder? ('i'/'s'/'d's
Enter a index of the device action to switch: 2
Enter the other index of device action to switch: 3
+-----------------+-----------------------------------------------------------------------------------------+
| device          | action list                                                                             |
+-----------------+-----------------------------------------------------------------------------------------+
| RobotArm        | ['Move', 'Rotate', 'Lift', 'Lower', 'Grab', 'Release', 'Position', 'Place', 'Home'] |
| Pipette         | ['Aspirate', 'Dispense', 'Rinse', 'Mix', 'Eject', 'Calibrate']                          |
| Pump            | ['Initialize', 'Flush', 'Inject']                                                       |
| Stirrer         | ['Stir', 'Stop']                                                                        |
| PowderDispenser | ['Dispense', 'Stop']                                                                    |
| WeighingMachine | ['Weigh', 'Tare', 'Stop']                                                               |
| Heater          | ['Heat', 'Cool', 'Stop']                                                                |
+-----------------+-----------------------------------------------------------------------------------------+

###################
[Feedback system]: task 0-->SolidStateModule_LoadPowder
###################
+-----+-----------------------------------------------+
| idx | SolidStateModule_LoadPowder --> device:action |
+-----+-----------------------------------------------+
| 1   | RobotArm_Home                                 |
| 2   | RobotArm_Grab                                 |
| 3   | RobotArm_Move                                 |
| 4   | RobotArm_Position                             |
| 5   | RobotArm_Place                                |
| 6   | RobotArm_Release                              |
+-----+-----------------------------------------------+
Do you want to insert, switch or delete device action of SolidStateModule_LoadPowder? ('i'/'s'/'d' or 'done'):
```

If the client inputs 's' to switch the order of specific actions within the existing action sequence, they need to enter the index of each action to be switched in command line interface. In the illustration above, the second and third actions were switched. Once this task is completed, you will see that the order of 'RobotArm_Move' and 'RobotArm_Grab' actions has been swapped.

4.4.3 Delete action in action sequence

```
###################
[Feedback system]: task 0-->SolidStateModule_LoadPowder
###################
+-----+-----------------------------------------------+
| idx | SolidStateModule_LoadPowder --> device:action |
+-----+-----------------------------------------------+
| 1   | RobotArm_Home                                 |
| 2   | RobotArm_Grab                                 |
| 3   | RobotArm_Move                                 |
| 4   | RobotArm_Position                             |
| 5   | RobotArm_Place                                |
| 6   | RobotArm_Release                              |
+-----+-----------------------------------------------+
Do you want to insert, switch or delete device action of SolidStateModule_LoadPowder? ('i'/'s'/'d' or 'done'): d
Enter a index of device to delete for SolidStateModule_LoadPowder, or 'back': 4
+----------------+-------------------------------------------------------------------------------------------+
| device         | action list                                                                               |
+----------------+-------------------------------------------------------------------------------------------+
| RobotArm       | ['Move', 'Rotate', 'Lift', 'Lower', 'Grab', 'Release', 'Position', 'Place', 'Home']       |
| Pipette        | ['Aspirate', 'Dispense', 'Rinse', 'Mix', 'Eject', 'Calibrate']                            |
| Pump           | ['Initialize', 'Flush', 'Inject']                                                         |
| Stirrer        | ['Stir', 'Stop']                                                                          |
| PowderDispenser | ['Dispense', 'Stop']                                                                     |
| WeighingMachine | ['Weigh', 'Tare', 'Stop']                                                                |
| Heater         | ['Heat', 'Cool', 'Stop']                                                                  |
+----------------+-------------------------------------------------------------------------------------------+

###################
[Feedback system]: task 0-->SolidStateModule_LoadPowder
###################
+-----+-----------------------------------------------+
| idx | SolidStateModule_LoadPowder --> device:action |
+-----+-----------------------------------------------+
| 1   | RobotArm_Home                                 |
| 2   | RobotArm_Grab                                 |
| 3   | RobotArm_Move                                 |
| 4   | RobotArm_Place                                |
| 5   | RobotArm_Release                              |
+-----+-----------------------------------------------+
Do you want to insert, switch or delete device action of SolidStateModule_LoadPowder? ('i'/'s'/'d' or 'done'):
```

If the client inputs 'd' to delete an existing action from the sequence, they need to enter the index of the action to be deleted in the command line interface. In the illustration above, you can see that the 'RobotArm_Position' action has been deleted from the sequence.

```
###################
[Feedback system (match task-->device-action)]: Final Device:Action List
###################
+----------------------------+-----------------------------------------------------------------------------------------+
| task                       | device:action list                                                                      |
+----------------------------+-----------------------------------------------------------------------------------------+
| SolidStateModule_LoadPowder | ['RobotArm_Home', 'RobotArm_Grab', 'RobotArm_Move', 'RobotArm_Place', 'RobotArm_Release'] |
| SolidStateModule_AddPowder  | ['PowderDispenser_Dispense']                                                            |
| SolidStateModule_MixPowders | ['Stirrer_Stir']                                                                        |
| SolidStateModule_Press      | ['RobotArm_Move', 'RobotArm_Position', 'RobotArm_Release']                              |
| SolidStateModule_Calcine    | ['Heater_Heat', 'Heater_Stop']                                                          |
| SolidStateModule_Sinter     | ['Heater_Heat', 'Heater_Stop']                                                          |
| SolidStateModule_Cool       | ['Heater_Cool', 'Heater_Stop']                                                          |
| SolidStateModule_Grind      | ['RobotArm_Move', 'RobotArm_Grab', 'RobotArm_Release']                                  |
| SolidStateModule_Pelletize  | ['RobotArm_Move', 'RobotArm_Grab', 'RobotArm_Release']                                  |
| SolidStateModule_MeasureMass | ['WeighingMachine_Tare', 'WeighingMachine_Weigh']                                      |
| SolidStateModule_Addsolution | ['Pipette_Aspirate', 'Pipette_Dispense']                                               |
+----------------------------+-----------------------------------------------------------------------------------------+
```

If all process is done, client input ‘done’ in command line interface. Through the GPT and feedback system, each task can be performed by customizing the action sequence for that task. The illustration above shows the resulting action sequences configured for each task.

4.5 Generate task template and type validation

```
###################
[GPT]: task template generation...
###################
token : {
  "prompt_tokens": 1146,
  "completion_tokens": 1656,
  "total_tokens": 2802
}
[Feedback system (modify task template)]: Data modification for SolidStateModule_LoadPowder template (type 'done' when finished)
+-------+---------------------------------------+---------------------------------------+
| index | key of 'SolidStateModule_LoadPowder' | value of 'SolidStateModule_LoadPowder' |
+-------+---------------------------------------+---------------------------------------+
| 1     | Material                              | {'Type': ''}                          |
| 2     | Amount                                | {'Value': 0, 'Dimension': 'g'}        |
| 3     | Device                                | {}                                    |
+-------+---------------------------------------+---------------------------------------+

###################
[Feedback system]: task 0 --> 'SolidStateModule_LoadPowder'
###################
Do you want to add, rename or delete data of 'SolidStateModule_LoadPowder' template? ('a'/'r'/'d' or 'done'):
```

The process involves creating a template in JSON format for the information that needs to be filled out for tasks made through the GPT and feedback system. The task template must include all the necessary information for the actions included in the action sequence. The example above shows the task template for 'SolidStateModule_LoadPowder' recommended by GPT. 'LoadPowder' is the process of loading a stock container filled with powder into the solid dispenser.

4.5.1 Add new key in task template

```
[Feedback system (modify task template)]: Data modification for SolidStateModule_LoadPowder template (type 'done' when finished)
+-------+---------------------------------------+---------------------------------------+
| index | key of 'SolidStateModule_LoadPowder' | value of 'SolidStateModule_LoadPowder' |
+-------+---------------------------------------+---------------------------------------+
| 1     | Material                              | {'Type': ''}                          |
| 2     | Amount                                | {'Value': 0, 'Dimension': 'g'}        |
| 3     | Device                                | {}                                    |
+-------+---------------------------------------+---------------------------------------+

###################
[Feedback system]: task 0 --> 'SolidStateModule_LoadPowder'
###################
Do you want to add, rename or delete data of 'SolidStateModule_LoadPowder' template? ('a'/'r'/'d' or 'done'): a
Enter an additional key for SolidStateModule_LoadPowder: FromTo
Is it quantitive value type for the key? 'y'/'n' or 'back' :'Fromto': n
+-------+---------------------------------------+---------------------------------------+
| index | key of 'SolidStateModule_LoadPowder' | value of 'SolidStateModule_LoadPowder' |
+-------+---------------------------------------+---------------------------------------+
| 1     | Material                              | {'Type': ''}                          |
| 2     | Amount                                | {'Value': 0, 'Dimension': 'g'}        |
| 3     | Device                                | {}                                    |
| 4     | Fromto                                | {'Type': ''}                          |
+-------+---------------------------------------+---------------------------------------+

###################
[Feedback system]: task 0 --> 'SolidStateModule_LoadPowder'
###################
Do you want to add, rename or delete data of 'SolidStateModule_LoadPowder' template? ('a'/'r'/'d' or 'done'):
```

The task template must include information about which material is being loaded. However, the current template does not include the position information for the robot arm. Therefore, if the client inputs 'a' to add new information, they will then enter the key for the new information. In this example, 'FromTo' was entered to denote the position information for the robot arm. When asked if the information is quantitative, the client entered 'n,' indicating that 'FromTo' is not quantitative. Quantitative information refers to data requiring numerical values, such as 'Volume,' 'Concentration,' 'InjectionRate,' 'Weight,' and 'StirRate'. If 'y' is input here, the task's value will be created as {'Value': 0, 'Dimension': ''}, and if 'n' is input, {'Type': ''} will be created. As a result, 'FromTo' as a key, with its value being a dictionary, was added to the template, as shown in the example.

4.5.2 Rename key in task template

```
+-------+-----------------------------------------+-----------------------------------------+
| index | key of 'SolidStateModule_LoadPowder'    | value of 'SolidStateModule_LoadPowder'  |
+-------+-----------------------------------------+-----------------------------------------+
| 1     | Material                                | {'Type': ''}                            |
| 2     | Amount                                  | {'Value': 0, 'Dimension': 'g'}          |
| 3     | Device                                  | {}                                      |
| 4     | Fromto                                  | {'Type': ''}                            |
+-------+-----------------------------------------+-----------------------------------------+

####################
[Feedback system]: task 0 --> 'SolidStateModule_LoadPowder'
####################
Do you want to add, rename or delete data of 'SolidStateModule_LoadPowder' template? ('a'/'r'/'d' or 'done'): r
Enter the index of key to rename: 4
Do you want to rename key ('k') or value ('v')? press 'k'/'v' or 'back: k
Enter the new key to rename: location
Is it quantitive value type for the key? 'y'/'n' or 'back' :n
+-------+-----------------------------------------+-----------------------------------------+
| index | key of 'SolidStateModule_LoadPowder'    | value of 'SolidStateModule_LoadPowder'  |
+-------+-----------------------------------------+-----------------------------------------+
| 1     | Material                                | {'Type': ''}                            |
| 2     | Amount                                  | {'Value': 0, 'Dimension': 'g'}          |
| 3     | Device                                  | {}                                      |
| 4     | Location                                | {'Type': ''}                            |
+-------+-----------------------------------------+-----------------------------------------+

####################
[Feedback system]: task 0 --> 'SolidStateModule_LoadPowder'
####################
Do you want to add, rename or delete data of 'SolidStateModule_LoadPowder' template? ('a'/'r'/'d' or 'done'):
```

If you want to change a specific key to another word, the client can input 'r' to modify it. In the example above, 'Fromto' key was intended to be changed to 'Location.' Therefore, the index of the key to be renamed was entered, and since the modification is not within the value, 'k' was input to change the key word. Then, the new word was entered, and since it is not quantitative information, 'n' was input. As seen in the example, the 'Fromto' key was successfully renamed to 'Location.'

```
[Feedback system (modify task template)]: Data modification for SolidStateModule_AddPowder template (type 'done' when finished)
+-------+-----------------------------------------+-----------------------------------------+
| index | key of 'SolidStateModule_AddPowder'     | value of 'SolidStateModule_AddPowder'   |
+-------+-----------------------------------------+-----------------------------------------+
| 1     | Material                                | {'Type': ''}                            |
| 2     | Amount                                  | {'Value': 0, 'Dimension': 'g'}          |
| 3     | Device                                  | {}                                      |
+-------+-----------------------------------------+-----------------------------------------+

####################
[Feedback system]: task 1 --> 'SolidStateModule_AddPowder'
####################
Do you want to add, rename or delete data of 'SolidStateModule_AddPowder' template? ('a'/'r'/'d' or 'done'): r
Enter the index of key to rename: 2
Do you want to rename key ('k') or value ('v')? press 'k'/'v' or 'back: v
Please input dimension of value (ex. mL, µL or mL/s, mPa, mV... etc) :mg
+-------+-----------------------------------------+-----------------------------------------+
| index | key of 'SolidStateModule_AddPowder'     | value of 'SolidStateModule_AddPowder'   |
+-------+-----------------------------------------+-----------------------------------------+
| 1     | Material                                | {'Type': ''}                            |
| 2     | Amount                                  | {'Value': 0, 'Dimension': 'mg'}         |
| 3     | Device                                  | {}                                      |
+-------+-----------------------------------------+-----------------------------------------+

####################
[Feedback system]: task 1 --> 'SolidStateModule_AddPowder'
####################
Do you want to add, rename or delete data of 'SolidStateModule_AddPowder' template? ('a'/'r'/'d' or 'done'):
```

Unlike the 'LoadPowder' task, the 'AddPowder' task template requires the 'Amount' information. In the example above, the value of the 'Amount' key's 'Dimension' was intended to be renamed from 'g' to 'mg.' Therefore, 'r' was input, followed by the index of the key to be renamed. Then, 'v' was input to indicate that the value inside the key should be modified, and the word to be changed within the 'Dimension' was entered. Through this

process, the 'Dimension' was successfully changed from 'g' to 'mg.'

4.5.3 Delete key in task template

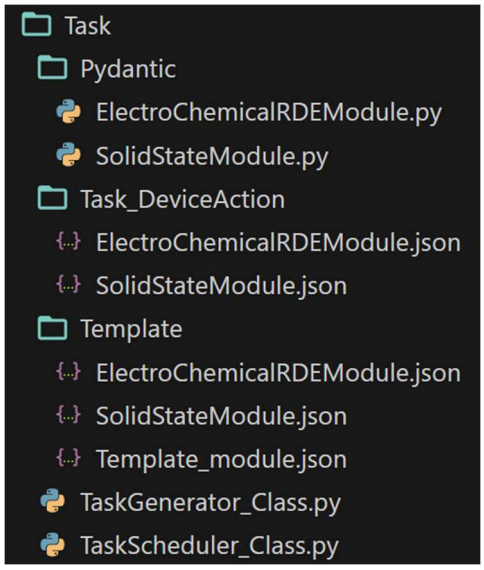
```
+-------+-------------------------------------+---------------------------------------+
| index | key of 'SolidStateModule_LoadPowder' | value of 'SolidStateModule_LoadPowder' |
+-------+-------------------------------------+---------------------------------------+
| 1     | Material                            | {'Type': ''}                          |
| 2     | Amount                              | {'Value': 0, 'Dimension': 'g'}        |
| 3     | Device                              | {}                                    |
| 4     | Location                            | {'Type': ''}                          |
+-------+-------------------------------------+---------------------------------------+

###################
[Feedback system]: task 0 --> 'SolidStateModule_LoadPowder'
###################
Do you want to add, rename or delete data of 'SolidStateModule_LoadPowder' template? ('a'/'r'/'d' or 'done'): d
Enter the key to delete: 2
+-------+-------------------------------------+---------------------------------------+
| index | key of 'SolidStateModule_LoadPowder' | value of 'SolidStateModule_LoadPowder' |
+-------+-------------------------------------+---------------------------------------+
| 1     | Material                            | {'Type': ''}                          |
| 2     | Device                              | {}                                    |
| 3     | Location                            | {'Type': ''}                          |
+-------+-------------------------------------+---------------------------------------+

###################
[Feedback system]: task 0 --> 'SolidStateModule_LoadPowder'
###################
Do you want to add, rename or delete data of 'SolidStateModule_LoadPowder' template? ('a'/'r'/'d' or 'done'):
```

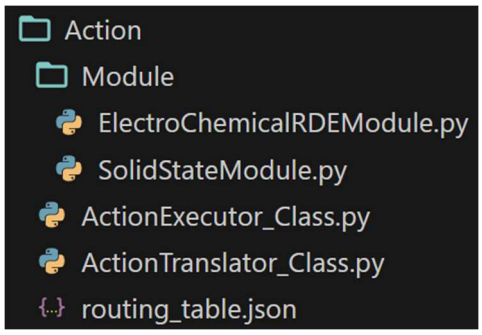
Since the 'LoadPowder' task involves loading a stock container filled with powder into the solid dispenser, the 'Amount' key, which indicates how much to load, is more suitable for the 'AddPowder' task. Therefore, 'd' was input, followed by the key's index, to delete it.

```
###################
[Feedback system (modify task template)]: Final Task Template
###################
+------------------------------+-----------------------------------------------------------------------------------------------------------+
| task                         | template                                                                                                  |
+------------------------------+-----------------------------------------------------------------------------------------------------------+
| SolidStateModule_LoadPowder  | {'Material': {'Type': ''}, 'Device': {}, 'Location': {'Type': ''}}                                        |
| SolidStateModule_AddPowder   | {'Material': {'Type': ''}, 'Amount': {'Value': 0, 'Dimension': 'mg'}, 'Device': {}}                        |
| SolidStateModule_MixPowders  | {'Time': {'Value': 0, 'Dimension': 'min'}, 'Speed': {'Value': 0, 'Dimension': 'rpm'}, 'Device': {}}        |
| SolidStateModule_Press       | {'Pressure': {'Value': 0, 'Dimension': 'Pa'}, 'Time': {'Value': 0, 'Dimension': 'min'}, 'Device': {}}      |
| SolidStateModule_Calcine     | {'Temperature': {'Value': 0, 'Dimension': 'ºC'}, 'Time': {'Value': 0, 'Dimension': 'h'}, 'Atmosphere': {'Type': ''}, 'Device': {}} |
| SolidStateModule_Sinter      | {'Temperature': {'Value': 0, 'Dimension': 'ºC'}, 'Time': {'Value': 0, 'Dimension': 'h'}, 'Device': {}}     |
| SolidStateModule_Cool        | {'Rate': {'Value': 0, 'Dimension': 'ºC/min'}, 'Device': {}}                                               |
| SolidStateModule_Grind       | {'Time': {'Value': 0, 'Dimension': 'min'}, 'Device': {}}                                                  |
| SolidStateModule_Pelletize   | {'Pressure': {'Value': 0, 'Dimension': 'Pa'}, 'Time': {'Value': 0, 'Dimension': 'min'}, 'Device': {}}      |
| SolidStateModule_MeasureMass | {'Method': {'Type': ''}, 'Device': {}}                                                                    |
| SolidStateModule_Addsolution | {'Material': {'Type': ''}, 'Volume': {'Value': 0, 'Dimension': 'mL'}, 'Concentration': {'Value': 0, 'Dimension': 'M'}, 'Device': {}} |
+------------------------------+-----------------------------------------------------------------------------------------------------------+
```

If all process is done, client input 'done' in command line interface. Each task template can be customized through the GPT and feedback system. The figure shows the resulting task templates configured for each task of the 'SolidStateModule'.

4.5.4 Code automated generation of action translator, task template and task Pydantic





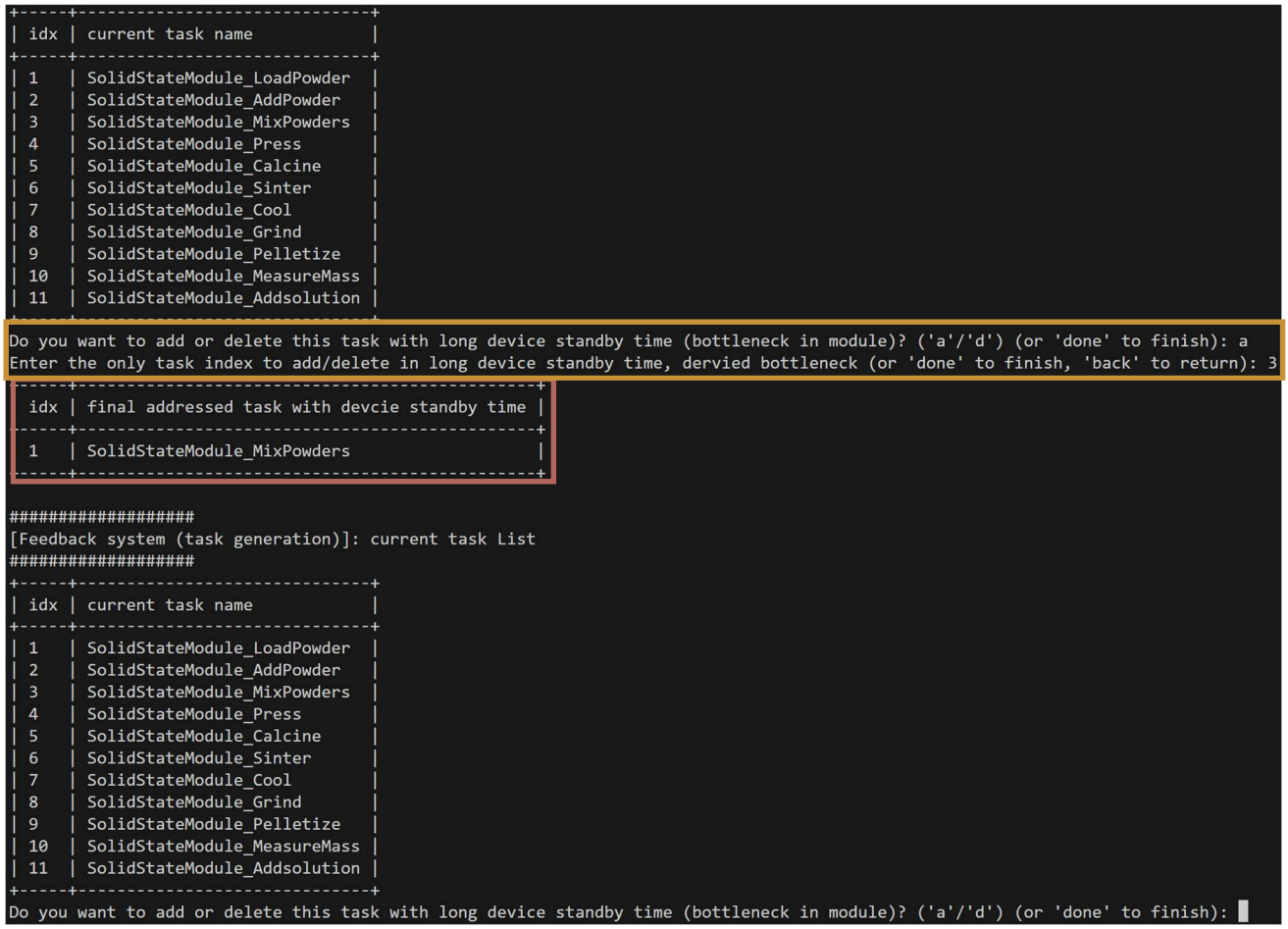
Directory name	Description
Task	The directory related to tasks
└─Pydantic	Script of type validation of task template via Pydantic package
└─Task_DeviceAction	A JSON file that stores the action sequences for all tasks for each module
└─Template	A JSON file that stores templates containing the necessary information for each task for each module
└─TaskGenerator_Class.py	Script of task generator
└─TaskScheduler_Class.py	Script of task scheduler
Action	The directory related to actions
└─Module	Script of module node for action translation
└─ActionExecutor_Class.py	Script of action executor
└─ActionTranslator_Class.py	Script of action translator
└─routing_table.json	A JSON file that stores IP/Port number of each module

4.6 Address long device standby times for job trigger

Device standby times, where no significant actions are performed in terms of devices, may substantially impair job execution efficiency. An example of device standby times is the chemical reaction period ("React" task) in the "BatchSynthesisModule", where chemical reactions occur in chemical vessels but no actions are performed in terms of devices.

In this chapter, the process of registering a task with a long device standby time for the job trigger's decision policy is demonstrated. The illustration above shows all the tasks currently available in the 'SolidStateModule.'

4.6.1 Add task with long device standby time



If the client inputs 'a' to add a task with a long device standby time, they need to enter the index of the task. Since the 'MixPowder' task in 'SolidStateModule' has a long device standby time, 'SolidStateModule_MixPowder' will be added.

4.6.2 Delete task with long device standby time

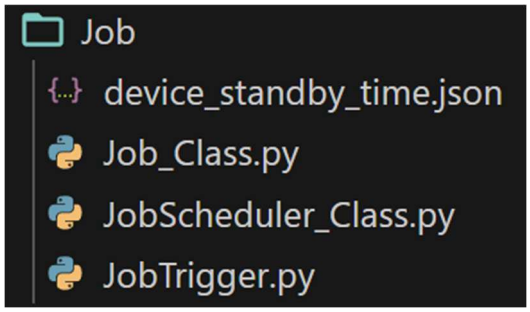
```
+-----+-----------------------------------------------+
| idx | final addressed task with devcie standby time |
+-----+-----------------------------------------------+
| 1   | SolidStateModule_MixPowders                   |
| 2   | SolidStateModule_Addsolution                  |
+-----+-----------------------------------------------+

###################
[Feedback system (task generation)]: current task List
###################
+-----+-----------------------------+
| idx | current task name           |
+-----+-----------------------------+
| 1   | SolidStateModule_LoadPowder |
| 2   | SolidStateModule_AddPowder  |
| 3   | SolidStateModule_MixPowders |
| 4   | SolidStateModule_Press      |
| 5   | SolidStateModule_Calcine    |
| 6   | SolidStateModule_Sinter     |
| 7   | SolidStateModule_Cool       |
| 8   | SolidStateModule_Grind      |
| 9   | SolidStateModule_Pelletize  |
| 10  | SolidStateModule_MeasureMass |
| 11  | SolidStateModule_Addsolution |
+-----+-----------------------------+
Do you want to add or delete this task with long device standby time (bottleneck in module)? ('a'/'d') (or 'done' to finish): d
Enter the only task index to add/delete in long device standby time, dervied bottleneck (or 'done' to finish, 'back' to return): 2
+-----+-----------------------------------------------+
| idx | final addressed task with devcie standby time |
+-----+-----------------------------------------------+
| 1   | SolidStateModule_MixPowders                   |
+-----+-----------------------------------------------+

###################
[Feedback system (task generation)]: current task List
###################
+-----+-----------------------------+
| idx | current task name           |
+-----+-----------------------------+
| 1   | SolidStateModule_LoadPowder |
| 2   | SolidStateModule_AddPowder  |
| 3   | SolidStateModule_MixPowders |
| 4   | SolidStateModule_Press      |
| 5   | SolidStateModule_Calcine    |
| 6   | SolidStateModule_Sinter     |
| 7   | SolidStateModule_Cool       |
| 8   | SolidStateModule_Grind      |
| 9   | SolidStateModule_Pelletize  |
| 10  | SolidStateModule_MeasureMass |
| 11  | SolidStateModule_Addsolution |
+-----+-----------------------------+
Do you want to add or delete this task with long device standby time (bottleneck in module)? ('a'/'d') (or 'done' to finish):
```

If a task has been incorrectly added and needs to be deleted, the client can input 'd' and then enter the index of the registered task to delete it. In the example above, the 'AddSolution' task was incorrectly registered, so 'd' and 2 were input to delete it.

4.6.3 Code automated generation of device_standby_time.json

```
+-----+------------------------------+
| idx | current task name            |
+-----+------------------------------+
| 1   | SolidStateModule_LoadPowder  |
| 2   | SolidStateModule_AddPowder   |
| 3   | SolidStateModule_MixPowders  |
| 4   | SolidStateModule_Press       |
| 5   | SolidStateModule_Calcine     |
| 6   | SolidStateModule_Sinter      |
| 7   | SolidStateModule_Cool        |
| 8   | SolidStateModule_Grind       |
| 9   | SolidStateModule_Pelletize   |
| 10  | SolidStateModule_MeasureMass |
| 11  | SolidStateModule_Addsolution |
+-----+------------------------------+
Do you want to add or delete this task with long device standby time (bottleneck in module)? ('a'/'d') (or 'done' to finish): done

###################
[Feedback system (address device standby time)]: Final task List
###################
+-----+-----------------------------------------------+
| idx | final addressed task with devcie standby time |
+-----+-----------------------------------------------+
| 1   | SolidStateModule_MixPowders                   |
+-----+-----------------------------------------------+
```

If all process is done, client input ‘done’ in command line interface. The figure shows the addressed task with long device standby times in ‘SolidStateModule’.



Directory name	Description
device_standby_time.json	A JSON file that addresses tasks with long device standby time of each module
Job_Class.py	Script of job generation
JobScheduler_Class.py	Script of job scheduler
JobTrigger.py	Script of job trigger

4.7 Masking table generation for safe task execution

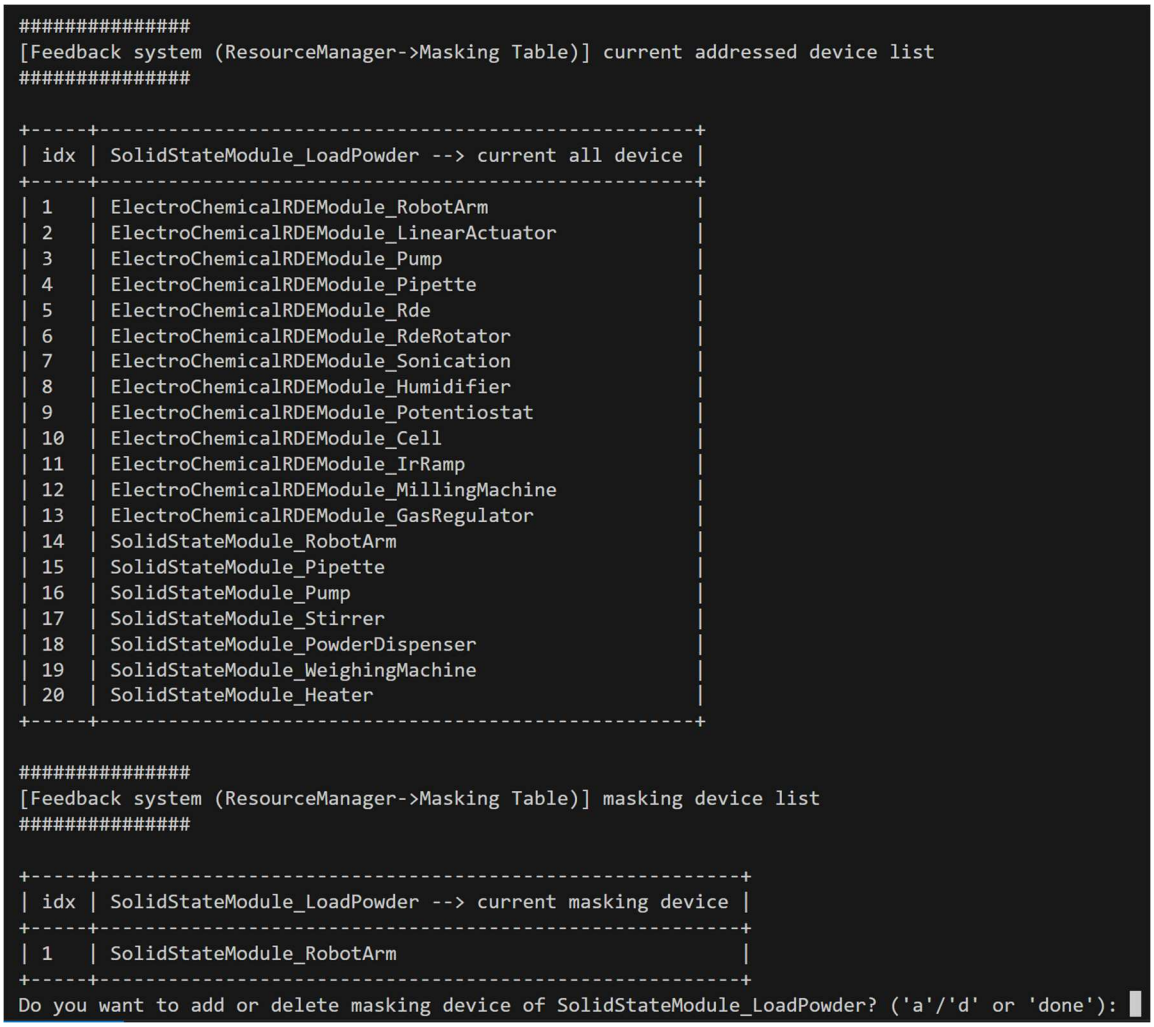
```
###############
[Feedback system (ResourceManager->Masking Table)] (Optional) add new devices for each previous task
###############

+-----+-----------------------------------------+-----------------------------------------------------------------------------------------------------+
| idx | task name                               | current masking table                                                                               |
+-----+-----------------------------------------+-----------------------------------------------------------------------------------------------------+
| 1   | ElectroChemicalRDEModule_SetupElectrode | ['ElectroChemicalRDEModule_RobotArm', 'ElectroChemicalRDEModule_Rde', 'SolidStateModule_RobotArm'] |
| 2   | ElectroChemicalRDEModule_Calibrate      | ['ElectroChemicalRDEModule_Potentiostat', 'ElectroChemicalRDEModule_RdeRotator']                    |
| 3   | ElectroChemicalRDEModule_LoadSample     | ['ElectroChemicalRDEModule_Pipette']                                                                |
| 4   | ElectroChemicalRDEModule_AdjustRotation | ['ElectroChemicalRDEModule_RdeRotator']                                                             |
| 5   | ElectroChemicalRDEModule_ApplyPotential | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
| 6   | ElectroChemicalRDEModule_RecordCurrent  | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
| 7   | ElectroChemicalRDEModule_PerformCV      | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
| 8   | ElectroChemicalRDEModule_PerformLPR     | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
| 9   | ElectroChemicalRDEModule_PerformEIS     | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
| 10  | ElectroChemicalRDEModule_ChangeSolution | ['ElectroChemicalRDEModule_Pump']                                                                   |
| 11  | ElectroChemicalRDEModule_CleanElectrode | ['ElectroChemicalRDEModule_Sonication']                                                             |
| 12  | ElectroChemicalRDEModule_AnalyzeData    | []                                                                                                  |
| 13  | ElectroChemicalRDEModule_PerformanceTest | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
+-----+-----------------------------------------+-----------------------------------------------------------------------------------------------------+
Is there a system for sharing newly registered SolidStateModule and devices? ('y'/'n'):
```

This chapter demonstrates the process of registering the task-specific masking table for the resource manager. The figure above shows the masking table for all tasks in the 'SolidStateModule.' Therefore, if you want to add the devices from existing modules to the task-specific masking table of the new module being registered, input 'y'; otherwise, input 'n'.

```
###############
[Feedback system (ResourceManager->Masking Table)] (Optional) add new devices for each previous task
###############

+-----+-----------------------------------------+-----------------------------------------------------------------------------------------------------+
| idx | task name                               | current masking table                                                                               |
+-----+-----------------------------------------+-----------------------------------------------------------------------------------------------------+
| 1   | ElectroChemicalRDEModule_SetupElectrode | ['ElectroChemicalRDEModule_RobotArm', 'ElectroChemicalRDEModule_Rde', 'SolidStateModule_RobotArm'] |
| 2   | ElectroChemicalRDEModule_Calibrate      | ['ElectroChemicalRDEModule_Potentiostat', 'ElectroChemicalRDEModule_RdeRotator']                    |
| 3   | ElectroChemicalRDEModule_LoadSample     | ['ElectroChemicalRDEModule_Pipette']                                                                |
| 4   | ElectroChemicalRDEModule_AdjustRotation | ['ElectroChemicalRDEModule_RdeRotator']                                                             |
| 5   | ElectroChemicalRDEModule_ApplyPotential | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
| 6   | ElectroChemicalRDEModule_RecordCurrent  | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
| 7   | ElectroChemicalRDEModule_PerformCV      | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
| 8   | ElectroChemicalRDEModule_PerformLPR     | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
| 9   | ElectroChemicalRDEModule_PerformEIS     | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
| 10  | ElectroChemicalRDEModule_ChangeSolution | ['ElectroChemicalRDEModule_Pump']                                                                   |
| 11  | ElectroChemicalRDEModule_CleanElectrode | ['ElectroChemicalRDEModule_Sonication']                                                             |
| 12  | ElectroChemicalRDEModule_AnalyzeData    | []                                                                                                  |
| 13  | ElectroChemicalRDEModule_PerformanceTest | ['ElectroChemicalRDEModule_Potentiostat']                                                           |
+-----+-----------------------------------------+-----------------------------------------------------------------------------------------------------+
Is there a system for sharing newly registered SolidStateModule and devices? ('y'/'n'): y
Enter the only task index to add new devices in masking table (or 'done' to finish, 'back' to return): 3
+-----+-----------------------------------+
| idx | new SolidStateModule --> current devices |
+-----+-----------------------------------+
| 1   | RobotArm                          |
| 2   | Pipette                           |
| 3   | Pump                              |
| 4   | Stirrer                           |
| 5   | PowderDispenser                   |
| 6   | WeighingMachine                   |
| 7   | Heater                            |
+-----+-----------------------------------+
Enter the only device index to add in masking table (or 'done' to finish, 'back' to return): 1
Add new device (SolidStateModule_RobotArm) in masking table: ['ElectroChemicalRDEModule_Pipette', 'SolidStateModule_RobotArm']
Enter the only device index to add in masking table (or 'done' to finish, 'back' to return): done
Is there a system for sharing newly registered SolidStateModule and devices? ('y'/'n'): n
```

Let's assume that 'ElectroChemicalRDEModule' was already registered and you are registering 'SolidStateModule.' If the user wants to add the 'RobotArm' of 'SolidStateModule' to the masking table of the 'LoadSample' task in 'ElectroChemicalRDEModule', you should input 'y' and then enter the index of the task. The figure above input '3', which represents 'ElectroChemicalRDEModule_LoadSample' task. Following this, the command line interface will prompt the user to enter the index of the 'SolidStateModule's 'RobotArm' that you want to register. The figure above input '1', which represents 'SolidStateModule_RobotArm' task.

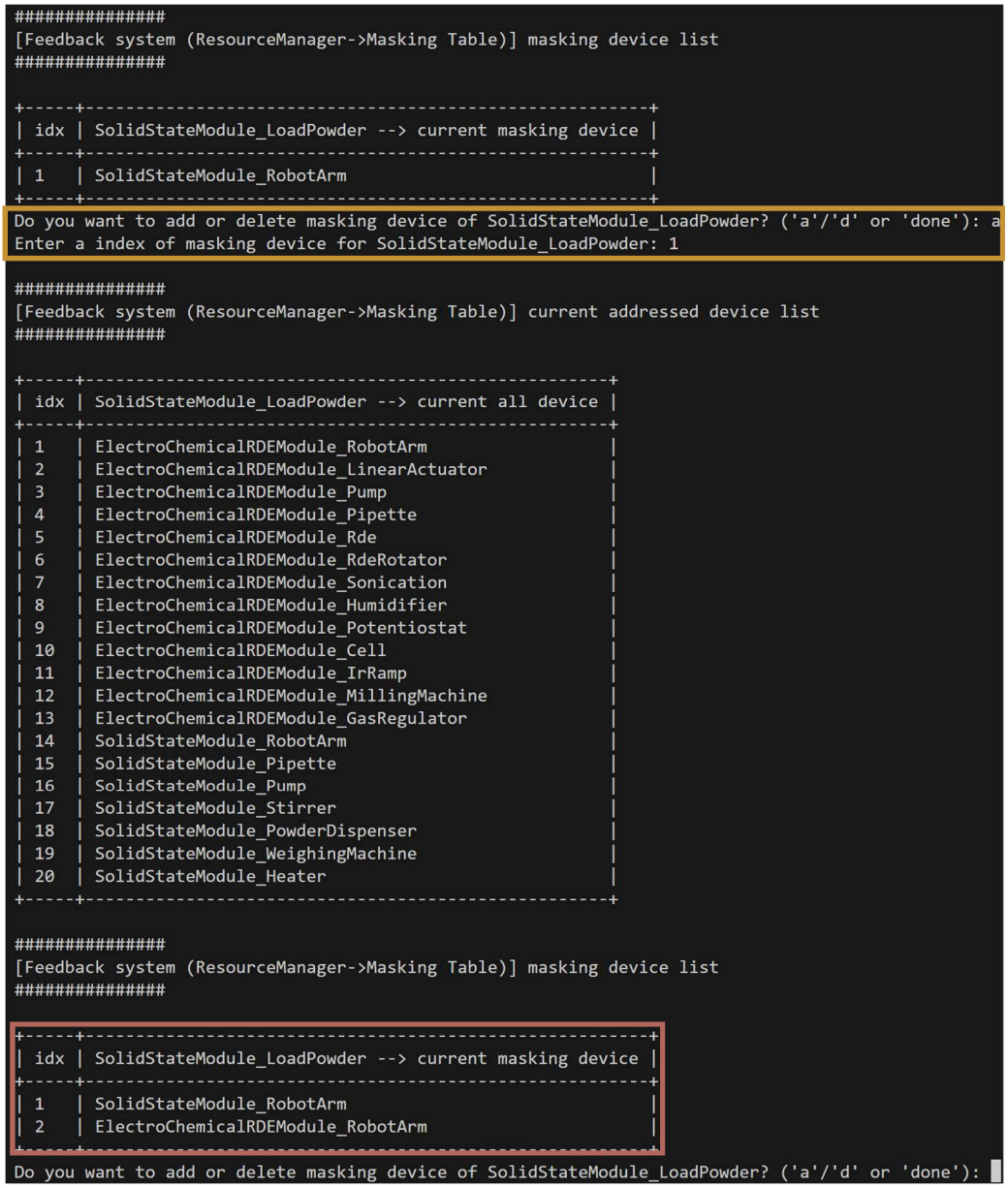
If you want to add another device, you should input the index of the device. Once the modifications to the masking table of the 'LoadSample' task in 'ElectroChemicalRDEModule' are complete, input 'done.' When the modifications to the task-specific masking table of 'SolidStateModule' are finished, input 'n'.



After completing the registration of the new module's device into the masking tables of the previously registered module, the next step is to modify the task-specific masking table of the new module. Therefore, the figure above shows all devices registered in OCTOPUS and asks if you want to modify the masking table of each task in new module.

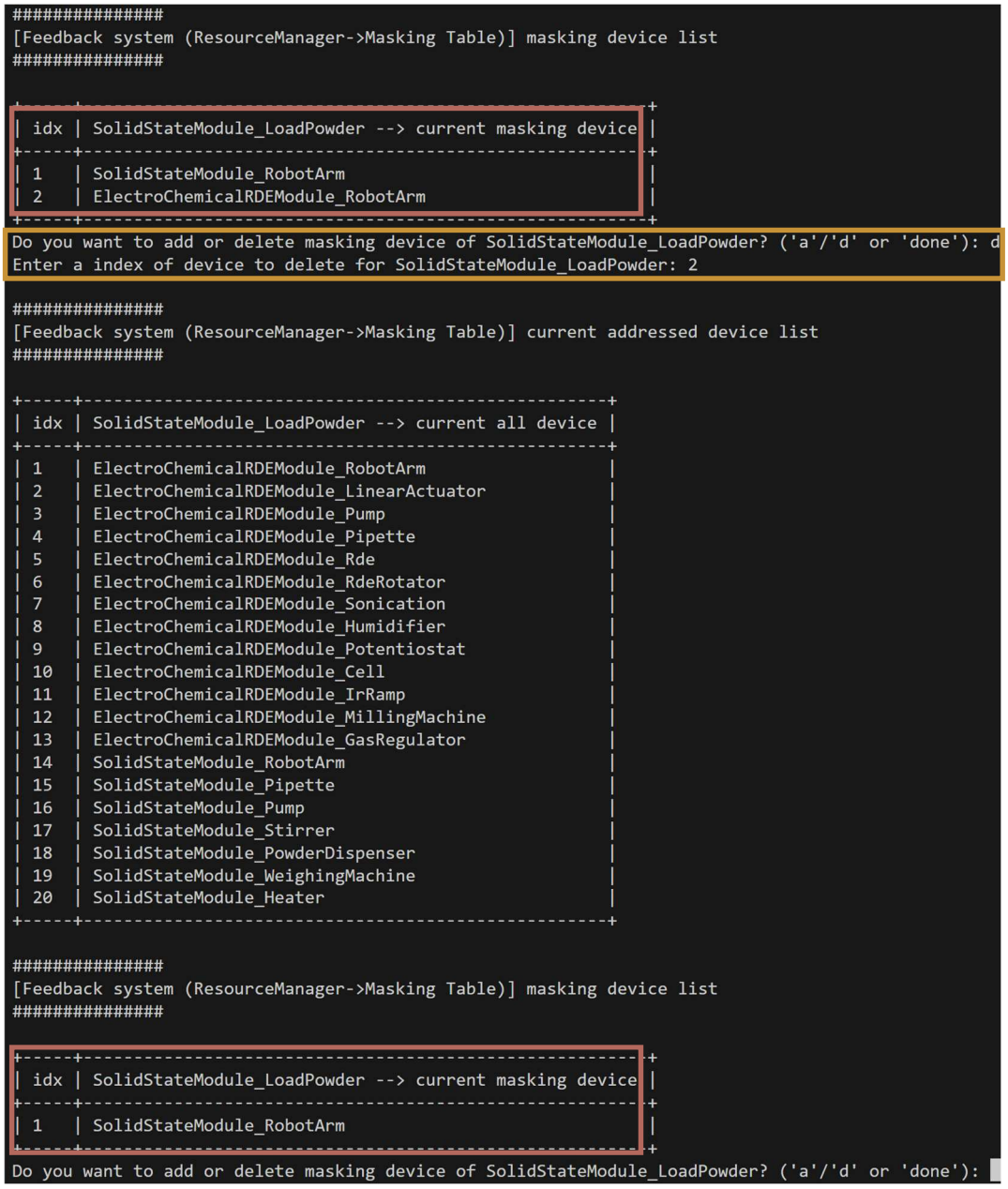
If the you want to register a new device into the masking table of each task, shown by the feedback system, you should input 'a'. To delete a registered device, you should input 'd'. To finish modifying the masking table, you should input 'done'.

4.7.1 Add new device in masking table for each task

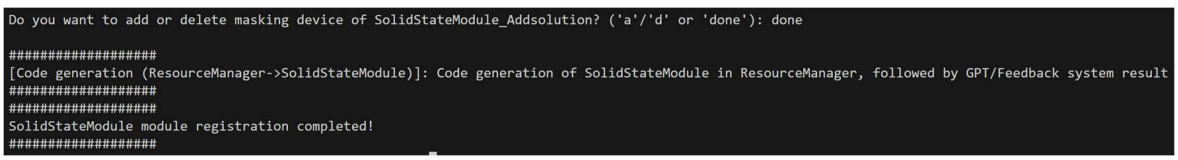


For example, let's assume that the same robot arm is shared between 'SolidStateModule' and 'ElectroChemicalRDEModule.' To register 'ElectroChemicalRDEModule_RobotArm' into the masking table of 'SolidStateModule_LoadPowder,' input the index (=1) corresponding to 'RobotArm' in 'ElectroChemicalRDEModule'. As a result, as shown in the red box in the illustration above, 'ElectroChemicalRDEModule_RobotArm' will be newly registered in the masking table of 'SolidStateModule_LoadPowder.'

4.7.2 Delete device in masking table for each task

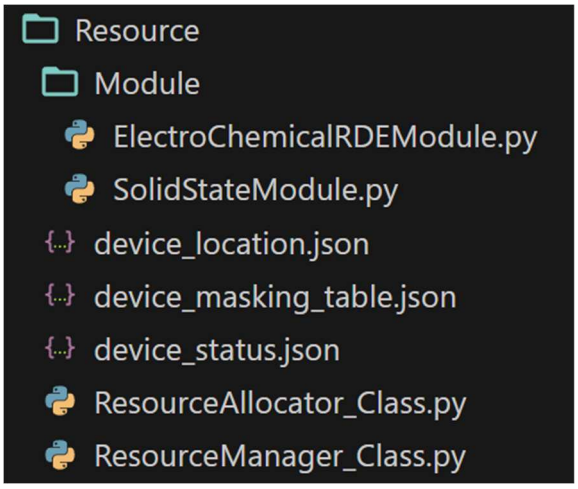


This time, let's delete a device registered in the masking table. To delete 'ElectroChemicalRDEModule_RobotArm' from the masking table of 'SolidStateModule_LoadPowder,' input the index corresponding to 'RobotArm' in 'ElectroChemicalRDEModule,' which is 2. As a result, as shown in the red box in the illustration above, 'ElectroChemicalRDEModule_RobotArm' will be removed from the masking table of 'SolidStateModule_LoadPowder.'



Once all modifications to the masking table are complete, input 'done' to exit the feedback system for masking table modifications. After that, the Copilot of OCTOPUS will proceed with code generation related to the masking table.

4.7.3 Code automated generation of resource manager



Directory/Script name	Description
Module	Resource allocator for each module
device_location.json	Device resources (location information) for each module
device_masking_table.json	Device status table included all devices for each module
device_status.json	Device masking table for all task of each module
ResourceAllocator_Class.py	Script of resource allocator inherited by Module/{module name}.py
ResourceManager_Class.py	Script of resource manager

5. Appendix: Manual modification in OCTOPUS

5.1 Connect device to module node

To perform the defined tasks using devices, you must define the communication methods (RS232, RS486 or ethernet) and data transmission protocols according to the specifications set by each manufacturer. This requires consulting the manuals or libraries provided by the manufacturers. For example, if you use a syringe pump from “Tecan”, refer to the “TecanAPI” library provided by Tecan to select the communication method and enter the data transmission protocol in the code. If you created a custom Arduino-based device, you should predefine the data transmission protocol based on serial communication and register it with the module node.

5.1.1 Set device information

```
class PowderDispenser:

    def __init__(self, logger_obj, device_name="PowderDispenser"):
        ########################################################################################################
        # Please configure the device information according to the PowderDispenser manufacturer guidelines.
        # ex) self.info['Port']='COM3' or '/dev/ttyUSB0'
        # ex) self.info['BaudRate']=9600
        ########################################################################################################

        self.info={}
        self.info['DeviceName']=device_name

        self.logger_obj=logger_obj
        self.device_name=device_name

        ########################################################################################################
        # Please configure the types of actions according to the PowderDispenser manufacturer guidelines.
        # if some device is made of yourself (ex. arduino),
        # >>> self.arduinoData = serial.Serial(self.info["Port"], self.info["BaudRate"])
        # if some device import manufacturer library and generate object in here
        # >>> from Tecan import TecanAPI
        # >>> self.syringe_pump = TecanAPI()
        ########################################################################################################
```

Device information requires the input of device settings. Since device information is stored in the task template by the master node, providing more detailed device information can enhance the reliability of the task.

```
import sys
import serial

class PowderDispenser:

    def __init__(self, logger_obj, device_name="PowderDispenser"):
        ########################################################################################################
        # Please configure the device information according to the PowderDispenser manufacturer guidelines.
        # ex) self.info['Port']='COM3' or '/dev/ttyUSB0'
        # ex) self.info['BaudRate']=9600
        ########################################################################################################

        self.info={}
        self.info['DeviceName']=device_name
        self.info["PowderList"]=["IrCl3","RuCl3","NiCl2","CarbonBlack"]
        self.info["Resolution"]="0.001g/step"
        self.info['Port']='COM3'
        self.info['BaudRate']=9600

        self.logger_obj=logger_obj
        self.device_name=device_name
        self.arduino = serial.Serial(self.info["Port"], self.info["BaudRate"])

        ########################################################################################################
        # Please configure the types of actions according to the PowderDispenser manufacturer guidelines.
        # if some device is made of yourself (ex. arduino),
        # >>> self.arduinoData = serial.Serial(self.info["Port"], self.info["BaudRate"])
        # if some device import manufacturer library and generate object in here
        # >>> from Tecan import TecanAPI
        # >>> self.syringe_pump = TecanAPI()
        ########################################################################################################
```

For example, in the case of a powder dispenser based on Arduino, you can include the powder type, the resolution, device port and baud rate as device information. The figure above represents manual modification of constructor (=‘__init__’) function in ‘PowderDispenser.py’

5.1.2 Translate action to device protocol based on manufacturer manual

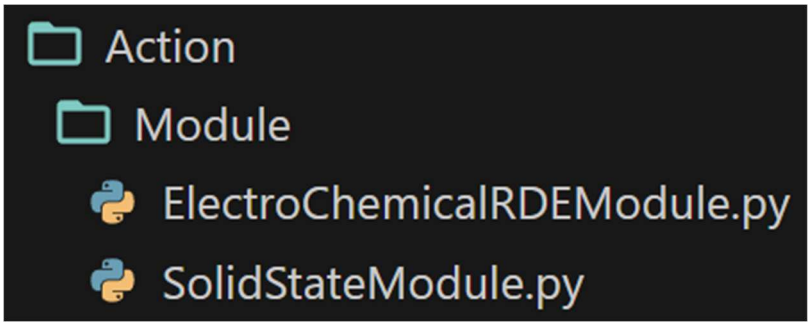
```python
class PowderDispenser:
    def Dispense(self, action_data:str, mode_type:str="virtual") -> str:
        '''
        :param action_data (str) :
        :param mode_type="virtual" (str): set virtual or real mode

        :return: return_res_msg
        '''
        current_action_type=sys._getframe().f_code.co_name
        device_name="{} ({})".format(self.device_name, mode_type)
        debug_msg="Start {} action, action_data={}".format(current_action_type, action_data)
        self.logger_obj.debug(device_name=device_name, debug_msg=debug_msg)
        action_data_list=action_data.split("&")

        if mode_type=="real":
            ##############################################################################################
            # Please configure the types of actions according to the device manufacturer guidelines. #
            ##############################################################################################
            action_data_list
            pass
        elif mode_type=="virtual":
            pass

        debug_msg="Finish {} action, action_data={}".format(current_action_type, action_data)
        self.logger_obj.debug(device_name=device_name, debug_msg=debug_msg)

        return_res_msg="[{}] : {}".format(device_name, debug_msg)
        return return_res_msg
```

After code generation through the Copilot of OCTOPUS, the action functions of the module node's devices are not implemented as shown in the red box in the illustration. Therefore, you need to translate the actions into device protocols according to the manufacturer's manual.

```python
class PowderDispenser:
    def Dispense(self, action_data:str, mode_type:str="virtual") -> str:
        '''
        :param action_data (str) :
        :param mode_type="virtual" (str): set virtual or real mode

        :return: return_res_msg
        '''
        current_action_type=sys._getframe().f_code.co_name
        device_name="{} ({})".format(self.device_name, mode_type)
        debug_msg="Start {} action, action_data={}".format(current_action_type, action_data)
        self.logger_obj.debug(device_name=device_name, debug_msg=debug_msg)
        action_data_list=action_data.split("&")

        if mode_type=="real":
            ##############################################################################################
            # Please configure the types of actions according to the device manufacturer guidelines. #
            ##############################################################################################
            material, amount, dimension = action_data_list
            self.arduino.write("{}&{}&{}".format(material, amount, dimension))
        elif mode_type=="virtual":
            pass

        debug_msg="Finish {} action, action_data={}".format(current_action_type, action_data)
        self.logger_obj.debug(device_name=device_name, debug_msg=debug_msg)

        return_res_msg="[{}] : {}".format(device_name, debug_msg)
        return return_res_msg
```

For example, in the case of a powder dispenser made with Arduino, the coding that entered material name, amount, and dimension is already done. Therefore, the ‘action_data_list’ should include these three information, and should be included diverse information depending on device. Use the ‘write’ function of 'self.arduino', defined in the constructor function, to execute the 'Dispense' function of the powder dispenser made with Arduino.

5.2 Complete action data



After completing the modifications to the module node, you need to modify the script file located in the 'Action/Module' directory.

```
class SolidStateModule(ActionExecutor):
    def SolidStateModule_AddPowder(self, task_info_list:list, jobID:int, location_dict:dict, TaskLogger_obj:object, mode_type="virtual"):
        # action_type_dict : Please refer action type of this module
        """
        Please construct action_data, using below information (action_type_list, location_dict, task_dict, task_idx).

        1. action_type_list (list)=['RobotArm_Move', 'RobotArm_Grasp', 'PowderDispenser_Dispense', 'RobotArm_Release']
        2. location_dict (dict)= --> result of ResouceManager.allocateResource
        3. task_dict ()={
            "Task": "SolidStateModule_AddPowder",
            "Data": {
                "Material": {
                    "Type": ""
                },
                "Amount": {
                    "Value": 0,
                    "Dimension": "g"
                },
                "Device": {}
            }
        }
        4. task_idx
        """
```

When you open the script saved under the module name, you will find several functions, defined for each task name of the module. These functions are defined for task execution, and the Copilot of OCTOPUS has automatically generated the functions to execute actions according to the defined action sequences.

The important point is that you need to define the action data for each action using 'task_dict,' which contains task information, 'location_dict,' which includes resources assigned to the module, and 'task_idx,' which indicates the sequence of operations.

```
class SolidStateModule(ActionExecutor):
    def SolidStateModule_AddPowder(self, task_info_list:list, jobID:int, location_dict:dict, TaskLogger_obj:object, mode_type="virtual"):
        """
        data_dict=dict()
        # data_dict["Reference"]=reference_dict, # just example. please modify depending on your module, and add new code in "Analysis.Analysis.calculateData"
        # data_dict["Absorbance"]=absorbance_dict, # just example. please modify depending on your module, and add new code in "Analysis.Analysis.calculateData"

        # please extract action_data in task_dict, and location dict from resource manager
        self.ResourceManager_obj.updateStatus(current_func_name, True)
        data_dict=self.executeAction(self.__SolidStateModule_name,jobID,"RobotArm","Move","please extract action_data in task_dict",mode_type, TaskLogger_obj, data_dict)

        # please extract action_data in task_dict, and location dict from resource manager
        self.ResourceManager_obj.updateStatus(current_func_name, True)
        data_dict=self.executeAction(self.__SolidStateModule_name,jobID,"RobotArm","Grasp","please extract action_data in task_dict",mode_type, TaskLogger_obj, data_dict)

        # please extract action_data in task_dict, and location dict from resource manager
        self.ResourceManager_obj.updateStatus(current_func_name, True)
        data_dict=self.executeAction(self.__SolidStateModule_name,jobID,"PowderDispenser","Dispense","please extract action_data in task_dict",mode_type,TaskLogger_obj,data_dict)

        # please extract action_data in task_dict, and location dict from resource manager
        self.ResourceManager_obj.updateStatus(current_func_name, True)
        data_dict=self.executeAction(self.__SolidStateModule_name,jobID,"RobotArm","Release","please extract action_data in task_dict",mode_type, TaskLogger_obj, data_dict)
        ##############################################################################################
        # Please configure the sequence and types of actions according to the module's robotic setting. #
        ##############################################################################################
        self.ResourceManager_obj.updateStatus(current_func_name, False)

    TaskLogger_obj.debug(self.__SolidStateModule_name, "Finish "+current_func_name+" Queue")

    return res_msg
```

Since the action data has already been defined in Chapter 5.1, you need to modify red boxes to customized action data on module node's device script using 'task_dict,' 'location_dict,' and 'task_idx', according to the previously defined format from module node's device script.

```python
class SolidStateModule(ActionExecutor):
    def SolidStateModule_AddPowder(self, task_info_list:list, jobID:int, location_dict:dict, TaskLogger_obj:object, mode_type="virtual"):
        """
        data_dict=dict()
        # data_dict["Reference"]=reference_dict, # just example. please modify depending on your module, and add new code in "Analysis.Analysis.calculateData"
        # data_dict["Absorbance"]=absorbance_dict, # just example. please modify depending on your module, and add new code in "Analysis.Analysis.calculateData"

        # please extract action_data in task_dict, and location dict from resource manager
        self.ResourceManager_obj.updateStatus(current_func_name, True)
        action_data=location_dict["Stirrer"][task_idx]
        data_dict=self.executeAction(self.__SolidStateModule_name,jobID,"RobotArm","Move",action_data,mode_type, TaskLogger_obj, data_dict)

        # please extract action_data in task_dict, and location dict from resource manager
        self.ResourceManager_obj.updateStatus(current_func_name, True)
        action_data="None"
        data_dict=self.executeAction(self.__SolidStateModule_name,jobID,"RobotArm","Grasp",action_data,mode_type, TaskLogger_obj, data_dict)

        # please extract action_data in task_dict, and location dict from resource manager
        self.ResourceManager_obj.updateStatus(current_func_name, True)
        action_data=[task_dict["Material"], task_dict["Amount"]["Value"], task_dict["Amount"]["Dimension"]]
        data_dict=self.executeAction(self.__SolidStateModule_name,jobID,"PowderDispenser","Dispense",action_data,mode_type,TaskLogger_obj,data_dict)

        # please extract action_data in task_dict, and location dict from resource manager
        self.ResourceManager_obj.updateStatus(current_func_name, True)
        action_data="None"
        data_dict=self.executeAction(self.__SolidStateModule_name,jobID,"RobotArm","Release",action_data,mode_type, TaskLogger_obj, data_dict)
        ##############################################################################################
        # Please configure the sequence and types of actions according to the module's robotic setting. #
        ##############################################################################################
        self.ResourceManager_obj.updateStatus(current_func_name, False)

    TaskLogger_obj.debug(self.__SolidStateModule_name, "Finish "+current_func_name+" Queue")

    return res_msg
```

For example, the 'AddPowder' task is composed of the action sequence, such as 'RobotArm_Move,' 'RobotArm_Grasp,' 'PowderDispenser_Dispense,' and 'RobotArm_Release.' You need to customize the action data for each action. The powder dispenser was predefined in the module node to receive material name, amount, and dimension when executing the "Dispense" action. Therefore, the action data is extracted from the task_dict and customized into a single list. Additionally, the 'RobotArm_Move' action can use the resource index assigned by the resource manager as action data. After defining the action data, the 'self.executeAction' function is used to command the module node to execute the action by inputting the module name, job ID, device name, action type, action data, mode type, and data dictionary.

5.3 [Optional] Define `calculateData` function in `Analysis` directory

Analysis > Module > ElectroChemicalRDEModule.py > ...

```python
# !/usr/bin/env python3
# -*- coding: utf-8 -*-
# ##
# @brief    [ElectroChemicalRDEModule] Analysis function of ElectroChemicalRDEModule for raw spectrum
# author Hyuk Jun Yoo (yoohj9475@kist.re.kr)
# GENEREATION 2024-06-26 13:51:49

import json
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os, time
from collections import OrderedDict

def RecordCurrent(data_dict:dict):

    pass
    # return result, clean_dict

def PerformCV(data_dict:dict):

    pass
    # return result, clean_dict

def PerformLPR(data_dict:dict):

    pass
    # return result, clean_dict

def PerformEIS(data_dict:dict):

    pass
    # return result, clean_dict

def AnalyzeData(data_dict:dict):

    pass
    # return result, clean_dict
```

After code generation via the Copilot of OCTOPUS, you will find that functions are created in the 'Analysis/Module' folder to preprocess raw data or spectrum data for each task of the module, if needed. In above figure, the functions received raw data or spectrum data from the 'ElectroChemicalModule' to extract specific properties, such as overpotential, current density, or stability about electrochemical properties. In such cases, you need to define how to handle the raw data or spectrum data in each function.

For example, 'PerformCV' refers to CV (Cyclic Voltammetry) analysis. Therefore, you should define the necessary hyperparameters for CV in the function, such as the number of cycles, scan rate, current value, and impedance value. This allows data processing using the predefined hyperparameters.

More detailed information of defining analysis script, please refer https://github.com/KIST-CSRC/Octopus/blob/main/Analysis/Module/UVVisModule.py.