Supplementary Information for

OCTOPUS: Operation **C**ontrol System for **T**ask **O**ptimization and Job **P**arallelization via a **U**ser-Optimal **S**cheduler

Hyuk Jun Yoo,[1,2] Kwan-Young Lee,[2*] Donghun Kim,[1*] and Sang Soo Han[1*]

[1]Computational Science Research Center, Korea Institute of Science and Technology, Seoul 02792, Republic of Korea

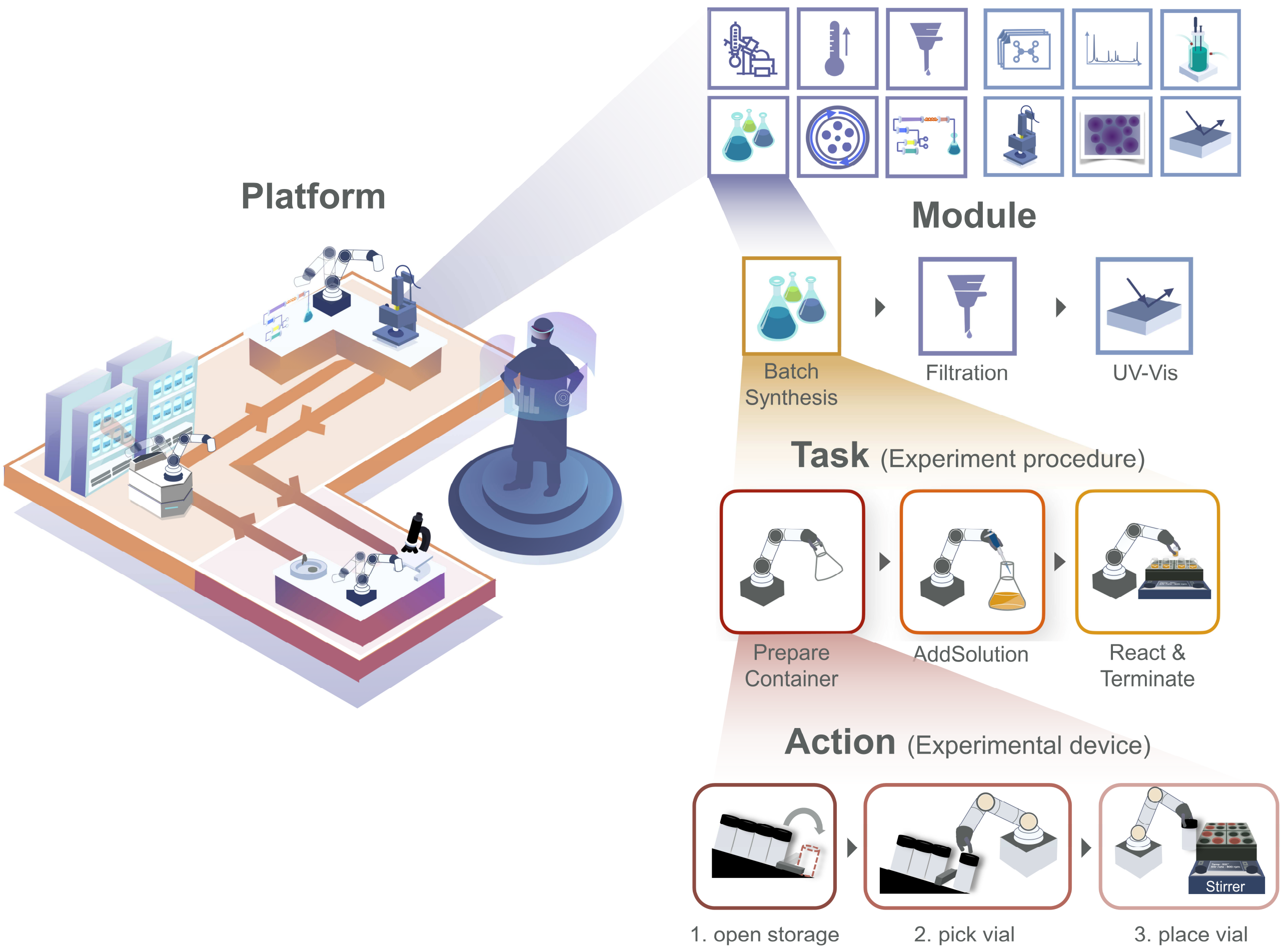
[2]Department of Chemical and Biological Engineering, Korea University, Seoul 02841, Republic of Korea

*Correspondence to: sangsoo@kist.re.kr (S.S.H.); donghun@kist.re.kr (D.K.); kylee@korea.ac.kr (K.-Y.L.).

Table of contents

1. Terminology definition in OCTOPUS

Supplementary Figure S1. Definition of words



2. Job submission via the interface node and job scheduler of the master node

Supplementary Figure S2. Detailed job script structure for manual and automated experimentations

The "metadata" key represents information about the experiments, including the subject, group name and log level. The "algorithm" key represents the process recommendation, including the model name, the total number of experiments and the model hyperparameters. The "process" key represents the experimental process information, including the module and task sequences and the fixed experimental conditions for each module.

Supplementary Figure S2a. Job script for manual experimentations

```
{
    "metadata" :
    {
        "subject":"Manual Experiment",
        "group":"KIST_CSRC",
        "logLevel":"DEBUG"
    },
    "model":
    {
        "modelName":"Manual",
        "totalExperimentNum":2,
        "inputParams":[
            {
                "AddSolution=AgNO3_Concentration" : 0.0125,
                "AddSolution=AgNO3_Volume" : 1000,
                "AddSolution=AgNO3_Injectionrate" : 100
            },
            {
                "AddSolution=AgNO3_Concentration" : 0.0175,
                "AddSolution=AgNO3_Volume" : 800,
                "AddSolution=AgNO3_Injectionrate" : 300
            }
        ]
    },
    "process":
    {
        "Synthesis":{
            "BatchSynthesis":{
                 "Sequence":["AddSolution_Citrate","AddSolution_H2O2","AddSolution_NaBH4","Stir",
                 "Heat","Mix", "AddSolution_AgNO3", "React"],
                "fixedParams":
                {
                    "AddSolution=H2O2_Concentration" : 0.375,
                    "AddSolution=H2O2_Volume" : 1100,
                    "AddSolution=H2O2_Injectionrate" : 100,
                    "AddSolution=Citrate_Concentration" : 0.04,
                    "AddSolution=Citrate_Volume" : 1100,
                    "AddSolution=Citrate_Injectionrate" : 100,
                    "AddSolution=NaBH4_Concentration" : 0.01,
                    "AddSolution=NaBH4_Volume" : 3000,
                    "AddSolution=NaBH4_Injectionrate" : 100,

                    "Stir=StirRate":1000,
                    "Heat=Temperature":25,
                    "Mix=Time":300,
                    "React=Time":3600
                }
```

```
            },
            "FlowSynthesis":{}
        },
        "Preprocess":{
            "Washing":{},
            "Ink":{}
        },
        "Characterization":{
            "UV":{}
        },
        "Evaluation":{
            "RDE":{},
            "Electrode":{}
        }
    }
}
```

Supplementary Figure S2b. Job script for automated experimentations

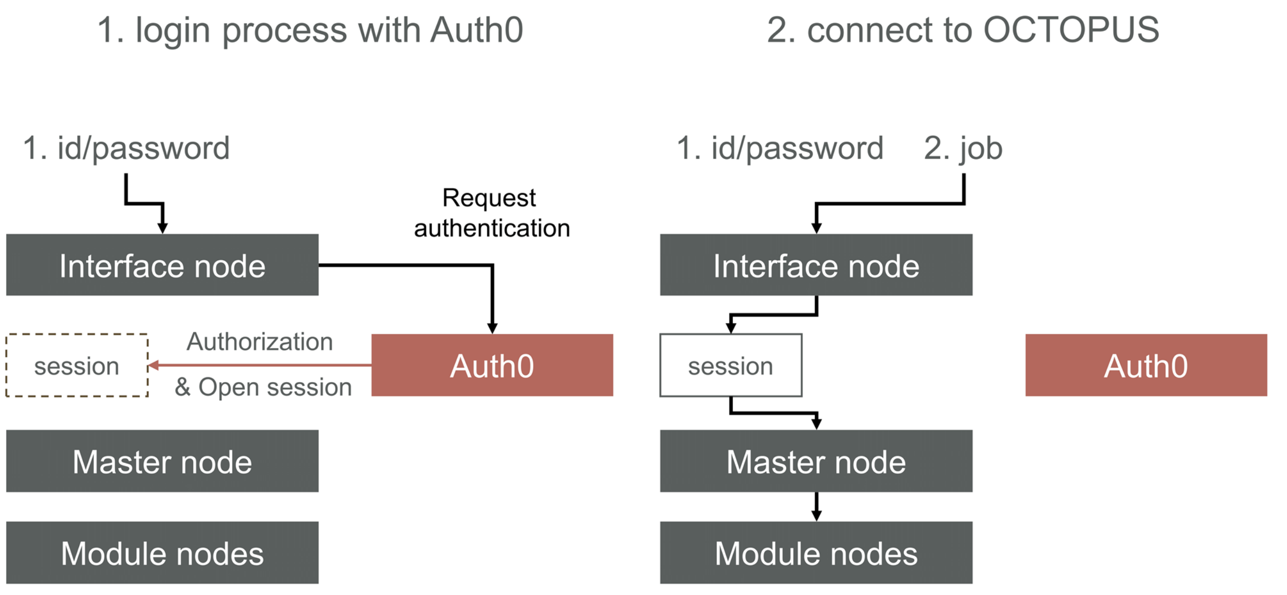
```
{
    "metadata" :
    {
        "subject":"Automated Experiment",
        "group":"KIST_CSRC",
        "logLevel":"DEBUG"
    },
    "model":
    {
        "modelName":"BayesianOptimization",
        "batchSize":6,
        "totalCycleNum":3,
        "verbose":0,
        "randomState":0,
        "sampling":{
            "samplingMethod":"latin",
            "samplingNum":20
        },
        "acq":{
            "acqMethod":"ucb",
            "acqSampler":"greedy",
            "acqHyperparameter":{
                "kappa":10.0
            }
        },
        "loss":{
            "lossMethod":"lambdamaxFWHMintensityLoss",
            "lossTarget":{
                "GetAbs":{
                    "Property":{
                        "lambdamax":573
                    },
                    "Ratio":{
                        "lambdamax":0.9,
                        "FWHM":0.03,
                        "intensity":0.07
                    }
                }
            }
        },
        "prange":{
            "AddSolution=AgNO3_Concentration" : [25, 375, 25],
            "AddSolution=AgNO3_Volume" : [100, 1200, 50],
            "AddSolution=AgNO3_Injectionrate" : [50, 200, 50]
        },
        "initParameterList":[],
```

```json
        "constraints":[]
    },
    "process":
    {
        "Synthesis":{
            "BatchSynthesis":
            {
                 "Sequence":["AddSolution_Citrate","AddSolution_H2O2","AddSolution_NaBH4",
                 "Stir","Heat","Mix", "AddSolution_AgNO3", "React"],
                "fixedParams":
                {
                    "AddSolution=AgNO3_Concentration" : 1250,
                    "AddSolution=H2O2_Concentration" : 375,
                    "AddSolution=H2O2_Volume" : 1200,
                    "AddSolution=H2O2_Injectionrate" : 200,
                    "AddSolution=Citrate_Concentration" : 20,
                    "AddSolution=Citrate_Volume" : 1200,
                    "AddSolution=Citrate_Injectionrate" : 200,
                    "AddSolution=NaBH4_Concentration" : 10,
                    "AddSolution=NaBH4_Volume" : 3000,
                    "AddSolution=NaBH4_Injectionrate" : 200,

                    "Stir=StirRate":1000,
                    "Heat=Temperature":25,
                    "Mix=Time":300,
                    "React=Time":7200
                }
            },
            "FlowSynthesis":{}
        },
        "Preprocess":{
            "Washing":{},
            "Ink":{}
        },
        "Characterization":{
            "UV-Vis":
            {
                 "Sequence":["GetAbs"],
                "fixedParams":
                {
                    "UV=Hyperparameter_WavelengthMin":300,
                    "UV=Hyperparameter_WavelengthMax":849,
                    "UV=Hyperparameter_BoxCarSize":10,
                    "UV=Hyperparameter_Prominence":0.01,
                    "UV=Hyperparameter_PeakWidth":20
                }
            }
        },
        "Evaluation":{
            "RDE":{},
            "Electrode":{}
        }
    }
}
```

Examples of job scripts for manual and automated experiments. The job script is based on the JSON (JavaScript Object Notation) format. Other job script formats were uploaded to the GitHub repository. (https://github.com/KIST-CSRC/Octopus)

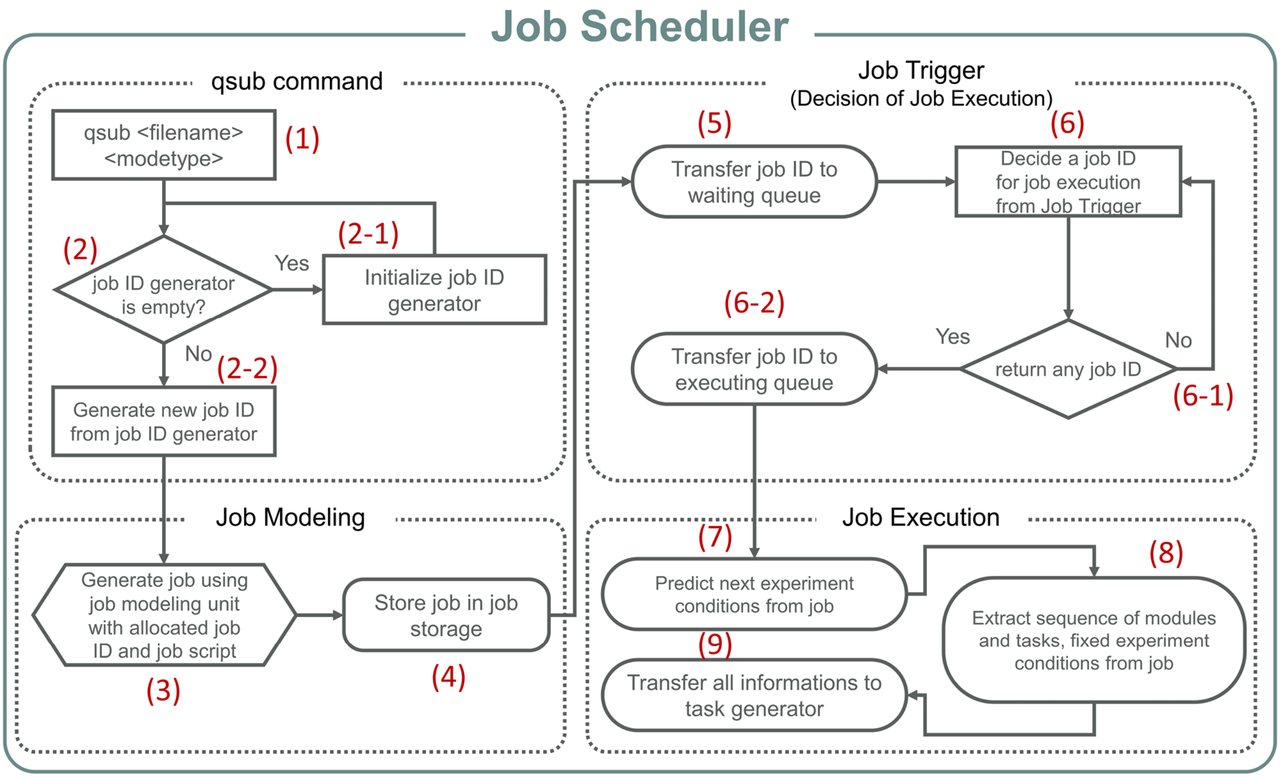
Supplementary Figure S3. Login process with Auth0 in the interface node of OCTOPUS



Supplementary Figure S4. Commands definitions for clients and administrators

Command	***Description***	***Example***
qstat	(Client) Monitor job in whole queue	qstat
qsub <jobscript filename> <mode>	(Client) submit job script file to server	qsub synthesis_test virtual qsub UV_test real
qdel <job ID>	(Client) delete job in hold queue	qdel 0
qhold <job ID>	(Client) hold job in execution queue	qhold 0
qrestart <job ID>	(Client) restart job in hold queue	qrestart 0
qlogout	(Administrator) logout from server	qlogout
ashutdown <module name or “all”>	(Administrator) shutdown module node or all of it	ashutdown UV ashutdown BatchSynthesis
areboot <module name or “all”>	(Administrator) reboot module node or all of it	areboot UV areboot all
updateNode <module name or “all”>	(Administrator) update module node for the newest physical device settings	updateNode BatchSynthesis

Supplementary Figure S5. Workflow of job submission in job scheduler



Supplementary Figure S6. Examples of job management via a command line interface

Supplementary Figure S6a. Command line interface examples of master node and module node initializations

(Module, BatchSynthesis) Module node on → heartbeat & update device information

```
[BatchSynthesis] Waiting...
[Emergency Stop] Waiting...
2023-10-05 19:15:23,577 - BatchSynthesis::INFO -- [BatchSynthesis] : packet information list:['info', 'BATCH', 'None', 'all', 'virtual']
2023-10-05 19:15:23,911 - BatchSynthesis::DEBUG -- [IKA_RET_0 (heartbeat)] : Hello World!! Succeed to connection to main computer!
2023-10-05 19:15:25,631 - BatchSynthesis::DEBUG -- [PUMP (heartbeat)] : [PUMP] heartbeat action success
2023-10-05 19:15:26,699 - BatchSynthesis::DEBUG -- [DS_B (heartbeat)] : [DS_B] heartbeat action success
2023-10-05 19:15:27,821 - BatchSynthesis::DEBUG -- [PIPETTE (heartbeat)] : [PIPETTE] heartbeat action success
2023-10-05 19:15:28,938 - BatchSynthesis::DEBUG -- [Science_town (heartbeat)] : Hello World!! Succeed to connection to Linear Actuator computer!
2023-10-05 19:15:31,005 - BatchSynthesis::DEBUG -- [VialStorage (heartbeat)] : Hello World!! Succeed to connection to main computer!
2023-10-05 19:15:31,020 - BatchSynthesis::DEBUG -- [Module Node (BatchSynthesis)] : {'Stirrer': {'Stirrer_0': {'Port': 'COM5', 'DeviceName': 'IKA_RET_0',
'Temperature': 25}}, 'Pump': {'AgNO3': {'SolutionType': 'Metal', 'PumpAddress': 1, 'PumpUsbAddr': '/dev/ttyPUMP2', 'Resolution': 1814000, 'Concentration':
 0.0125, 'SyringeVolume': 5000, 'DeviceName': 'CavroCentris'}, 'H2O': {'SolutionType': 'Solvent', 'PumpAddress': 0, 'PumpUsbAddr': '/dev/ttyPUMP0', 'Resol
ution': 1814000, 'Concentration': 0, 'SyringeVolume': 5000, 'Connection': {'H2O': 5, 'AgNO3': 6, 'NaBH4': 7, 'H2O2': 8, 'Citrate': 9}, 'DeviceName': 'Cavr
oCentris'}, 'NaBH4': {'SolutionType': 'Reductant', 'PumpAddress': 2, 'PumpUsbAddr': '/dev/ttyPUMP2', 'Resolution': 1814000, 'Concentration': 0.01, 'Syring
eVolume': 5000, 'DeviceName': 'CavroCentris'}, 'H2O2': {'SolutionType': 'Oxidant', 'PumpAddress': 3, 'PumpUsbAddr': '/dev/ttyPUMP1', 'Resolution': 1814000
, 'Concentration': 0.375, 'Density': 1.45, 'MolarMass': 34.0147, 'SyringeVolume': 5000, 'DeviceName': 'CavroCentris'}, 'Citrate': {'SolutionType': 'CA', '
PumpAddress': 4, 'PumpUsbAddr': '/dev/ttyPUMP1', 'Resolution': 1814000, 'Concentration': 0.04, 'SyringeVolume': 5000, 'DeviceName': 'CavroCentris'}}, 'DS_
B': {'Id': 'dsr01', 'Model': 'm0609', 'NETWORK': '192.168.137.100', 'WorkRange': 900}, 'Pipette': {'PVP': {'SolutionType': 'Surfactant', 'PumpUsbAddr': '/
dev/ttyUSB0', 'DeviceName': 'Dynamixel'}}, 'LinearActutator': {'Stirrer_location_list': [{'x': 807000, 'y': 305420}, {'x': 716240, 'y': 305420}, {'x': 877
030, 'y': 375370}, {'x': 647830, 'y': 375370}, {'x': 877030, 'y': 465330}, {'x': 647830, 'y': 465330}, {'x': 807190, 'y': 534460}, {'x': 716240, 'y': 5344
60}], 'move_z': {'up': 125000, 'down': 150000}, 'Center_location_list': {'x': 450000, 'y': 0, 'z': 125000}}, 'VialStorage': {'DeviceName': 'VialStorage',
'Port': 'COM3', 'BaudRate': 9600}}
```

(Module, UV-Vis) Module node on → heartbeat & update device information

```
[UV] Waiting...
[Emergency Stop] Waiting...
2023-09-24 22:21:35,189 - UV-Vis::INFO -- [UV] : packet information list:['info', 'UV', 'None', 'all', 'virtual']
Send :  b'heartbeat,status'
2023-09-24 22:21:37,195 - UV-Vis::DEBUG -- [USB2000plus (heartbeat)] : Hello World!! Succeed to connection to main computer!
2023-09-24 22:21:37,197 - UV-Vis::DEBUG -- [Module Node (UV-Vis)] : {'Spectrometer': {'DeviceName': 'USB2000+', 'DetectionRange': '200-850nm', 'Solvent':
{'Solution': 'H2O', 'Value': 2000, 'Dimension': 'μL'}}, 'LightSource': {'DeviceName': 'DH-2000-BAL', 'DetectionRange': '210-2500nm', 'Lamp': 'deuterium(25
W) and halogen lamps(20W)'}}
```

(Master) Master node on → Resource manager send heartbeat & update device information from modules

```
2023-09-24 22:03:04,142 - JobServer::INFO -- [ResourceManager] : receive information of BatchSynthesis
2023-09-24 22:03:10,169 - JobServer::INFO -- [ResourceManager] : receive information of UV-Vis
[JobServer] Server on at :5555.
[JobServer] Waiting...
```

When the master node is executed, it sends a heartbeat to the activated module nodes to check their connectivity. If all module nodes are successfully connected, the resource manager retrieves the device information from each module node. In the two examples mentioned above, this information includes the experimental device information associated with each module node.

Supplementary Figure S6b. Command line interface examples of module node updates

(Client) Update device information from all modules

```
input commands (if terminate: input 'qlogout'): updateNode
2023-10-05 19:30:20,738 - JobClient::INFO -- [HJ] : request to update Node Server
2023-10-05 19:30:38,178 - JobClient::INFO -- [HJ] : succeed to update module node information:{'BatchSynthesis': {'Stirrer': {'Stirrer_0': {'Port': 'C
OM5', 'DeviceName': 'IKA_RET_0', 'Temperature': 25}}, 'Pump': {'AgNO3': {'SolutionType': 'Metal', 'PumpAddress': 1, 'PumpUsbAddr': '/dev/ttyPUMP2', 'R
esolution': 1814000, 'Concentration': 0.0125, 'SyringeVolume': 5000, 'DeviceName': 'CavroCentris'}, 'H2O': {'SolutionType': 'Solvent', 'PumpAddress':
0, 'PumpUsbAddr': '/dev/ttyPUMP0', 'Resolution': 1814000, 'Concentration': 0, 'SyringeVolume': 5000, 'Connection': {'H2O': 5, 'AgNO3': 6, 'NaBH4': 7,
'H2O2': 8, 'Citrate': 9}, 'DeviceName': 'CavroCentris'}, 'NaBH4': {'SolutionType': 'Reductant', 'PumpAddress': 2, 'PumpUsbAddr': '/dev/ttyPUMP2', 'Res
olution': 1814000, 'Concentration': 0.01, 'SyringeVolume': 5000, 'DeviceName': 'CavroCentris'}, 'H2O2': {'SolutionType': 'Oxidant', 'PumpAddress': 3,
'PumpUsbAddr': '/dev/ttyPUMP1', 'Resolution': 1814000, 'Concentration': 0.375, 'Density': 1.45, 'MolarMass': 34.0147, 'SyringeVolume': 5000, 'DeviceNa
me': 'CavroCentris'}, 'Citrate': {'SolutionType': 'CA', 'PumpAddress': 4, 'PumpUsbAddr': '/dev/ttyPUMP1', 'Resolution': 1814000, 'Concentration': 0.04
, 'SyringeVolume': 5000, 'DeviceName': 'CavroCentris'}}, 'DS_B': {'Id': 'dsr01', 'Model': 'm0609', 'NETWORK': '192.168.137.100', 'WorkRange': 900}, 'P
ipette': {'PVP': {'SolutionType': 'Surfactant', 'PumpUsbAddr': '/dev/ttyUSB0', 'DeviceName': 'Dynamixel'}}, 'LinearActutator': {'Stirrer_location_list
': [{'x': 807000, 'y': 305420}, {'x': 716240, 'y': 305420}, {'x': 877030, 'y': 375370}, {'x': 647830, 'y': 375370}, {'x': 877030, 'y': 465330}, {'x':
647830, 'y': 465330}, {'x': 807190, 'y': 534460}, {'x': 716240, 'y': 534460}], 'move_z': {'up': 125000, 'down': 150000}, 'Center_location_list': {'x':
 450000, 'y': 0, 'z': 125000}}, 'VialStorage': {'DeviceName': 'VialStorage', 'Port': 'COM3', 'BaudRate': 9600}}, 'UV': {'Spectrometer': {'DeviceName':
 'USB2000+', 'DetectionRange': '200-850nm', 'Solvent': {'Solution': 'H2O', 'Value': 2000, 'Dimension': 'µL'}}, 'LightSource': {'DeviceName': 'DH-2000-
BAL', 'DetectionRange': '210-2500nm', 'Lamp': 'deuterium(25W) and halogen lamps(20W)'}}}
```

(Module, BatchSynthesis) Resend heartbeat & update new device information & transfer that information to Master

```
2023-10-05 19:30:22,244 - BatchSynthesis::INFO -- [BatchSynthesis] : packet information list:['info', 'BATCH', 'None', 'all', 'virtual']
2023-10-05 19:30:22,524 - BatchSynthesis::DEBUG -- [IKA_RET_0 (heartbeat)] : Hello World!! Succeed to connection to main computer!
2023-10-05 19:30:24,225 - BatchSynthesis::DEBUG -- [PUMP (heartbeat)] : [PUMP] heartbeat action success
2023-10-05 19:30:25,328 - BatchSynthesis::DEBUG -- [DS_B (heartbeat)] : [DS_B] heartbeat action success
2023-10-05 19:30:26,400 - BatchSynthesis::DEBUG -- [PIPETTE (heartbeat)] : [PIPETTE] heartbeat action success
2023-10-05 19:30:27,472 - BatchSynthesis::DEBUG -- [Science_town (heartbeat)] : Hello World!! Succeed to connection to Linear Actuator computer!
2023-10-05 19:30:29,551 - BatchSynthesis::DEBUG -- [VialStorage (heartbeat)] : Hello World!! Succeed to connection to main computer!
2023-10-05 19:30:29,565 - BatchSynthesis::DEBUG -- [Module Node (BatchSynthesis)] : {'Stirrer': {'Stirrer_0': {'Port': 'COM5', 'DeviceName': 'IKA_RET_0',
'Temperature': 25}}, 'Pump': {'AgNO3': {'SolutionType': 'Metal', 'PumpAddress': 1, 'PumpUsbAddr': '/dev/ttyPUMP2', 'Resolution': 1814000, 'Concentration':
 0.0125, 'SyringeVolume': 5000, 'DeviceName': 'CavroCentris'}, 'H2O': {'SolutionType': 'Solvent', 'PumpAddress': 0, 'PumpUsbAddr': '/dev/ttyPUMP0', 'Resol
ution': 1814000, 'Concentration': 0, 'SyringeVolume': 5000, 'Connection': {'H2O': 5, 'AgNO3': 6, 'NaBH4': 7, 'H2O2': 8, 'Citrate': 9}, 'DeviceName': 'Cavr
oCentris'}, 'NaBH4': {'SolutionType': 'Reductant', 'PumpAddress': 2, 'PumpUsbAddr': '/dev/ttyPUMP2', 'Resolution': 1814000, 'Concentration': 0.01, 'Syring
eVolume': 5000, 'DeviceName': 'CavroCentris'}, 'H2O2': {'SolutionType': 'Oxidant', 'PumpAddress': 3, 'PumpUsbAddr': '/dev/ttyPUMP1', 'Resolution': 1814000
, 'Concentration': 0.375, 'Density': 1.45, 'MolarMass': 34.0147, 'SyringeVolume': 5000, 'DeviceName': 'CavroCentris'}, 'Citrate': {'SolutionType': 'CA', '
PumpAddress': 4, 'PumpUsbAddr': '/dev/ttyPUMP1', 'Resolution': 1814000, 'Concentration': 0.04, 'SyringeVolume': 5000, 'DeviceName': 'CavroCentris'}}, 'DS_
B': {'Id': 'dsr01', 'Model': 'm0609', 'NETWORK': '192.168.137.100', 'WorkRange': 900}, 'Pipette': {'PVP': {'SolutionType': 'Surfactant', 'PumpUsbAddr': '/
dev/ttyUSB0', 'DeviceName': 'Dynamixel'}}, 'LinearActutator': {'Stirrer_location_list': [{'x': 807000, 'y': 305420}, {'x': 716240, 'y': 305420}, {'x': 877
030, 'y': 375370}, {'x': 647830, 'y': 375370}, {'x': 877030, 'y': 465330}, {'x': 647830, 'y': 465330}, {'x': 807190, 'y': 534460}, {'x': 716240, 'y': 5344
60}], 'move_z': {'up': 125000, 'down': 150000}, 'Center_location_list': {'x': 450000, 'y': 0, 'z': 125000}}, 'VialStorage': {'DeviceName': 'VialStorage',
'Port': 'COM3', 'BaudRate': 9600}}
```

(Module, UV-Vis) Resend heartbeat & update new device information & transfer that information to Master

```
2023-10-05 19:30:33,668 - UV-Vis::INFO -- [UV] : packet information list:['info', 'UV', 'None', 'all', 'virtual']
Send :  b'heartbeat,status'
2023-10-05 19:30:35,687 - UV-Vis::DEBUG -- [USB2000plus (heartbeat)] : Hello World!! Succeed to connection to main computer!
2023-10-05 19:30:35,687 - UV-Vis::DEBUG -- [Module Node (UV-Vis)] : {'Spectrometer': {'DeviceName': 'USB2000+', 'DetectionRange': '200-850nm', 'Solvent':
{'Solution': 'H2O', 'Value': 2000, 'Dimension': 'µL'}}, 'LightSource': {'DeviceName': 'DH-2000-BAL', 'DetectionRange': '210-2500nm', 'Lamp': 'deuterium(25
W) and halogen lamps(20W)'}}
```

If the client makes changes to the device settings, or adds additional devices to a module node, the master node must be updated due to the allocation of device action via the new settings. The client can use the 'updateNode' command in the prompt interface to refresh the latest device information of the module node.

Supplementary Figure S6c. Command line interface examples of login process, job submission and job status

(Client) login success/qsub → qstat

```
Please enter your id: : HJ
Please enter your password: : 123123
Server: login success
input commands (if terminate: input 'qlogout'): qsub figure/job_id_0 virtual
2023-09-18 14:42:29,413 - JobClient::INFO -- [HJ] : request to submit job : USER/HJ/job_script/figure/job_id_0.json
2023-09-18 14:42:29,413 - JobClient::INFO -- [Client] : path error : [Errno 2] No such file or directory: 'USER/HJ/job_script/figure/job_id_0.json'
input commands (if terminate: input 'qlogout'): qsub figure/jobid_0 virtual
2023-09-18 14:42:57,613 - JobClient::INFO -- [HJ] : request to submit job : USER/HJ/job_script/figure/jobid_0.json
2023-09-18 14:42:57,663 - JobClient::INFO -- [HJ] : succeed to submit job, userName:HJ, jobID:0
input commands (if terminate: input 'qlogout'): qstat
2023-09-18 14:43:05,693 - JobClient::INFO -- [HJ] : request to check job status
+----------+-------------------+-------+----------------+---------+-------+----------------------------------+----------+
| userName |      jobTime      | jobID |  jobFileName   | current | total |              status              | modeType |
+----------+-------------------+-------+----------------+---------+-------+----------------------------------+----------+
|    HJ    | 2023-09-18 14:42:57 |  0  | figure/jobid_0 |    6    |   18  | 6_0/18:Batch-->PrepareContainer | virtual  |
+----------+-------------------+-------+----------------+---------+-------+----------------------------------+----------+
2023-09-18 14:43:05,698 - JobClient::INFO -- [HJ] : succeed to check job status
input commands (if terminate: input 'qlogout'):
```

(Client) login failure

```
Please enter your id: : HJ
Please enter your password: : 12341234
Server: login failure
```

(Master) login success/qsub → qstat

```
2023-09-24 22:05:40,095 - JobServer::INFO -- [JobServer] : ('127.0.0.1', 47886) is connected.
2023-09-24 22:05:44,533 - JobServer::INFO -- [JobServer] : login status:login success, username:HJ
2023-09-24 22:05:55,695 - jobID_0_AI::INFO -- [JobExecution] : [jobID=0] Algorithm, model : BayesianOptimization
2023-09-24 22:05:55,696 - jobID_0_AI::INFO -- [JobExecution] : ######### Cycle start #########
2023-09-24 22:05:55,696 - JobServer::INFO -- [JobServer] : HJ ('127.0.0.1', 47886): qsub?HJ?figure/jobid_0?{'metadata': {'subject': 'jobID_0_AI', 'group': 'KIST_CSRC', 'logLevel': 'DEBUG'}, 'algorithm': {'model': 'BayesianOptimization', 'batchSize': 6, 'totalCycleNum': 3, 'verbose': 0, 'randomState': 0, 'sampling': {'samplingMethod': 'latin', 'samplingNum': 20}, 'acq': {'acqMethod': 'ucb', 'acqSampler': 'greedy', 'acqHyperparameter': {'kappa': 10.0}}, 'loss': {'lossMethod': 'lambdamaxFWHMintensityLoss', 'lossTarget': {'GetAbs': {'Property': {'lambdamax': 573}, 'Ratio': {'lambdamax': 0.9, 'FWHM': 0.03, 'intensity': 0.07}}}}, 'prange': {'AddSolution=AgNO3_Concentration': [25, 375, 25], 'AddSolution=AgNO3_Volume': [100, 1200, 50], 'AddSolution=AgNO3_Injectionrate': [50, 200, 50]}, 'initParameterList': [], 'constraints': []}, 'process': {'Synthesis': {'BatchSynthesis': {'fixedParams': {'BatchSynthesis=Sequence': ['AddSolution_Citrate', 'AddSolution_H2O2', 'AddSolution_NaBH4', 'Stir', 'Heat', 'Mix', 'AddSolution_AgNO3', 'React'], 'AddSolution=AgNO3_Concentration': 1250, 'AddSolution=H2O2_Concentration': 375, 'AddSolution=H2O2_Volume': 1200, 'AddSolution=H2O2_Injectionrate': 200, 'AddSolution=Citrate_Concentration': 20, 'AddSolution=Citrate_Volume': 1200, 'AddSolution=Citrate_Injectionrate': 200, 'AddSolution=NaBH4_Concentration': 10, 'AddSolution=NaBH4_Volume': 3000, 'AddSolution=NaBH4_Injectionrate': 200, 'Stir=StirRate': 1000, 'Heat=Temperature': 25, 'Mix=Time': 300, 'React=Time': 7200}}, 'FlowSynthesis': {}}, 'Preprocess': {'Washing': {}, 'Ink': {}}, 'Characterization': {'UV': {'fixedParams': {'UV=Sequence': ['GetAbs'], 'UV=Hyperparameter_WavelengthMin': 300, 'UV=Hyperparameter_WavelengthMax': 849, 'UV=Hyperparameter_BoxCarSize': 10, 'UV=Hyperparameter_Prominence': 0.01, 'UV=Hyperparameter_PeakWidth': 20}}}, 'Evaluation': {'RDE': {}, 'Electrode': {}}}}?virtual
2023-09-24 22:05:55,696 - jobID_0_AI::INFO -- [BayesianOptimization] : batchSize=6
2023-09-24 22:05:55,706 - jobID_0_AI::INFO -- [Algorithm [0]] : next_point 0: {'AddSolution=AgNO3_Concentration': 25, 'AddSolution=AgNO3_Volume': 100, 'AddSolution=AgNO3_Injectionrate': 50}
2023-09-24 22:05:55,706 - jobID_0_AI::INFO -- [Algorithm [1]] : next_point 1: {'AddSolution=AgNO3_Concentration': 225, 'AddSolution=AgNO3_Volume': 650, 'AddSolution=AgNO3_Injectionrate': 150}
2023-09-24 22:05:55,711 - jobID_0_AI::INFO -- [Algorithm [2]] : next_point 2: {'AddSolution=AgNO3_Concentration': 150, 'AddSolution=AgNO3_Volume': 900, 'AddSolution=AgNO3_Injectionrate': 50}
2023-09-24 22:05:55,721 - jobID_0_AI::INFO -- [Algorithm [3]] : next_point 3: {'AddSolution=AgNO3_Concentration': 350, 'AddSolution=AgNO3_Volume': 1100, 'AddSolution=AgNO3_Injectionrate': 150}
2023-09-24 22:05:55,732 - jobID_0_AI::INFO -- [Algorithm [4]] : next_point 4: {'AddSolution=AgNO3_Concentration': 300, 'AddSolution=AgNO3_Volume': 400, 'AddSolution=AgNO3_Injectionrate': 100}
2023-09-24 22:05:55,742 - jobID_0_AI::INFO -- [Algorithm [5]] : next_point 5: {'AddSolution=AgNO3_Concentration': 125, 'AddSolution=AgNO3_Volume': 500, 'AddSolution=AgNO3_Injectionrate': 100}
2023-09-24 22:05:55,787 - jobID_0_AI::INFO -- [ProcessGenerator] : Allocate all of process based on job script!
2023-09-24 22:05:55,792 - jobID_0_AI::INFO -- [JobExecution] : ######### [jobID_0_AI-HJ] Cycle 1/3 is running #########
2023-09-24 22:05:55,813 - jobID_0_AI::INFO -- [TaskScheduler] : check location: {'BatchSynthesis': {'Stirrer': ['?', '?', '?', '?', '?', '?', '?', '?'], 'vialHolder': ['?', '?', '?', '?', '?', '?', '?', '?']}, 'UV': {'vialHolder': ['?', '?', '?', '?', '?', '?', '?', '?']}}
2023-09-24 22:05:55,823 - jobID_0_AI::INFO -- [TaskScheduler] : BatchSynthesis is started, allocate location: {'Stirrer': [0, 1, 2, 3, 4, 5], 'vialHolder': [0, 1, 2, 3, 4, 5]}
2023-09-24 22:05:55,834 - jobID_0_AI::INFO -- [TaskScheduler] : check location after allocation: {'BatchSynthesis': {'Stirrer': [0, 0, 0, 0, 0, 0, '?', '?'], 'vialHolder': [0, 0, 0, 0, 0, 0, '?', '?']}, 'UV': {'vialHolder': ['?', '?', '?', '?', '?', '?', '?', '?']}}
2023-09-24 22:05:58,939 - jobID_0_AI::DEBUG -- [Batch] : storage_empty_to_stirrer is started.
2023-09-24 22:05:58,950 - jobID_0_AI::DEBUG -- [Batch] : vial_num_list: [0, 1, 2, 3, 4, 5]
2023-09-24 22:05:58,960 - jobID_0_AI::DEBUG -- [Batch] : line_num_list: [0, 0, 0, 0, 0, 0]
2023-09-24 22:05:58,970 - jobID_0_AI::DEBUG -- [Batch] : vialHolder_list: [0, 1, 2, 3, 4, 5]
2023-09-24 22:05:58,980 - jobID_0_AI::DEBUG -- [Batch] : Start Robot Queue : storage_empty_to_stirrer is started.
```

(Master) login failure

```
2023-10-05 19:42:48,794 - Master::INFO -- [Master] : ('127.0.0.1', 41186) is connected.
2023-10-05 19:42:52,999 - Master::INFO -- [Master] : login status:login failure, username:HJ
```

The client tries to log in to the master node via their ID and password. If the client ID or password does not match, the master node notifies the client of a login failure. After the login process is successful, the client can submit a job script using the 'qsub' command. The master then converts to an activated job and conducts the experiment based on the information in the job script. The client can monitor the status of their job using the 'qstat' command. The job status table includes the client name, job submission time, filename of job script, number of current experiments, number of total experiments, job status and mode type of the job.

Supplementary Figure S6d. Command line interface examples of job submission, hold, restart and deletion

(Client) qsub → qhold → qrestart

```
input commands (if terminate: input 'qlogout'): qstat
2023-09-24 22:29:12,389 - JobClient::INFO -- [HJ] : request to check job status
+----------+---------------------+-------+----------------+---------+-------+---------------------------------+----------+
| userName |       jobTime       | jobID |  jobFileName   | current | total |              status             | modeType |
+----------+---------------------+-------+----------------+---------+-------+---------------------------------+----------+
|    HJ    | 2023-09-24 22:28:39 |   0   | figure/jobid_0 |    6    |   18  | 6_2/18:Batch-->PrepareContainer | virtual  |
+----------+---------------------+-------+----------------+---------+-------+---------------------------------+----------+
2023-09-24 22:29:12,395 - JobClient::INFO -- [HJ] : succeed to check job status
input commands (if terminate: input 'qlogout'): qhold 0
2023-09-24 22:29:19,453 - JobClient::INFO -- [HJ] : request to hold job (jobID:0)
2023-09-24 22:29:21,554 - JobClient::INFO -- [HJ] : succeed to hold job (jobID: 0)
input commands (if terminate: input 'qlogout'): qstat
2023-09-24 22:29:26,646 - JobClient::INFO -- [HJ] : request to check job status
+----------+---------------------+-------+----------------+---------+-------+-----------------------------------------+----------+
| userName |       jobTime       | jobID |  jobFileName   | current | total |                  status                 | modeType |
+----------+---------------------+-------+----------------+---------+-------+-----------------------------------------+----------+
|    HJ    | 2023-09-24 22:28:39 |   0   | figure/jobid_0 |    6    |   18  | Holding&6_2/18:Batch-->PrepareContainer | virtual  |
+----------+---------------------+-------+----------------+---------+-------+-----------------------------------------+----------+
2023-09-24 22:29:26,652 - JobClient::INFO -- [HJ] : succeed to check job status
input commands (if terminate: input 'qlogout'): qrestart 0
2023-09-24 22:29:51,758 - JobClient::INFO -- [HJ] : request to restart job (jobID:0)
2023-09-24 22:29:52,835 - JobClient::INFO -- [HJ] : succeed to restart job (jobID: 0)
input commands (if terminate: input 'qlogout'): qstat
2023-09-24 22:29:55,189 - JobClient::INFO -- [HJ] : request to check job status
+----------+---------------------+-------+----------------+---------+-------+---------------------------------+----------+
| userName |       jobTime       | jobID |  jobFileName   | current | total |              status             | modeType |
+----------+---------------------+-------+----------------+---------+-------+---------------------------------+----------+
|    HJ    | 2023-09-24 22:28:39 |   0   | figure/jobid_0 |    6    |   18  | 6_2/18:Batch-->PrepareContainer | virtual  |
+----------+---------------------+-------+----------------+---------+-------+---------------------------------+----------+
2023-09-24 22:29:55,194 - JobClient::INFO -- [HJ] : succeed to check job status
```

(Module) qsub → qhold → qrestart

```
2023-09-24 22:29:15,071 - BatchSynthesis::INFO -- [BatchSynthesis] : packet information list:['qhold', '0']
2023-09-24 22:29:15,071 - BatchSynthesis::DEBUG -- [Module Node (UV-Vis)] : qhold:jobID 0 is holded
2023-09-24 22:29:16,269 - BatchSynthesis::INFO -- [BatchSynthesis] : packet information list:['0', 'DS_B', 'storage_empty_to_stirrer', '0,2', 'virtual']
holded jobID :  0
current input_qhold_jobID_list :  [0]
2023-09-24 22:29:47,386 - BatchSynthesis::INFO -- [BatchSynthesis] : packet information list:['qrestart', '0']
2023-09-24 22:29:49,258 - BatchSynthesis::DEBUG -- [DS_B (callServer)] : [DS_B] storage_empty_to_stirrer action success
2023-09-24 22:29:54,501 - BatchSynthesis::INFO -- [BatchSynthesis] : packet information list:['0', 'LA', 'center', 'null', 'virtual']
2023-09-24 22:29:54,517 - BatchSynthesis::DEBUG -- [Science_town (virtual)] : start moving to specific location:center
2023-09-24 22:29:54,517 - BatchSynthesis::DEBUG -- [Science_town (virtual)] : location:center
```

(Client) qsub → qhold → qdel

```
input commands (if terminate: input 'qlogout'): qstat
2023-09-24 22:36:39,641 - JobClient::INFO -- [HJ] : request to check job status
+----------+---------------------+-------+----------------+---------+-------+---------------------------------+----------+
| userName |       jobTime       | jobID |  jobFileName   | current | total |              status             | modeType |
+----------+---------------------+-------+----------------+---------+-------+---------------------------------+----------+
|    HJ    | 2023-09-24 22:36:31 |   0   | figure/jobid_0 |    6    |   18  | 6_0/18:Batch-->PrepareContainer | virtual  |
+----------+---------------------+-------+----------------+---------+-------+---------------------------------+----------+
2023-09-24 22:36:39,647 - JobClient::INFO -- [HJ] : succeed to check job status
input commands (if terminate: input 'qlogout'): qhold 0
2023-09-24 22:36:54,547 - JobClient::INFO -- [HJ] : request to hold job (jobID:0)
2023-09-24 22:36:56,636 - JobClient::INFO -- [HJ] : succeed to hold job (jobID: 0)
input commands (if terminate: input 'qlogout'): qstat
2023-09-24 22:37:04,715 - JobClient::INFO -- [HJ] : request to check job status
+----------+---------------------+-------+----------------+---------+-------+-----------------------------------------+----------+
| userName |       jobTime       | jobID |  jobFileName   | current | total |                  status                 | modeType |
+----------+---------------------+-------+----------------+---------+-------+-----------------------------------------+----------+
|    HJ    | 2023-09-24 22:36:31 |   0   | figure/jobid_0 |    6    |   18  | Holding&6_1/18:Batch-->PrepareContainer | virtual  |
+----------+---------------------+-------+----------------+---------+-------+-----------------------------------------+----------+
2023-09-24 22:37:04,721 - JobClient::INFO -- [HJ] : succeed to check job status
input commands (if terminate: input 'qlogout'): qdel 0
2023-09-24 22:37:06,673 - JobClient::INFO -- [HJ] : request to delete job (jobID:0)
2023-09-24 22:37:07,733 - JobClient::INFO -- [HJ] : succeed to delete job (jobID:0)
input commands (if terminate: input 'qlogout'): qstat
2023-09-24 22:37:12,696 - JobClient::INFO -- [HJ] : request to check job status
+----------+---------+-------+-------------+---------+-------+--------+----------+
| userName | jobTime | jobID | jobFileName | current | total | status | modeType |
+----------+---------+-------+-------------+---------+-------+--------+----------+
+----------+---------+-------+-------------+---------+-------+--------+----------+
2023-09-24 22:37:12,702 - JobClient::INFO -- [HJ] : succeed to check job status
input commands (if terminate: input 'qlogout'):
```

(Module) qsub → qhold → qdel

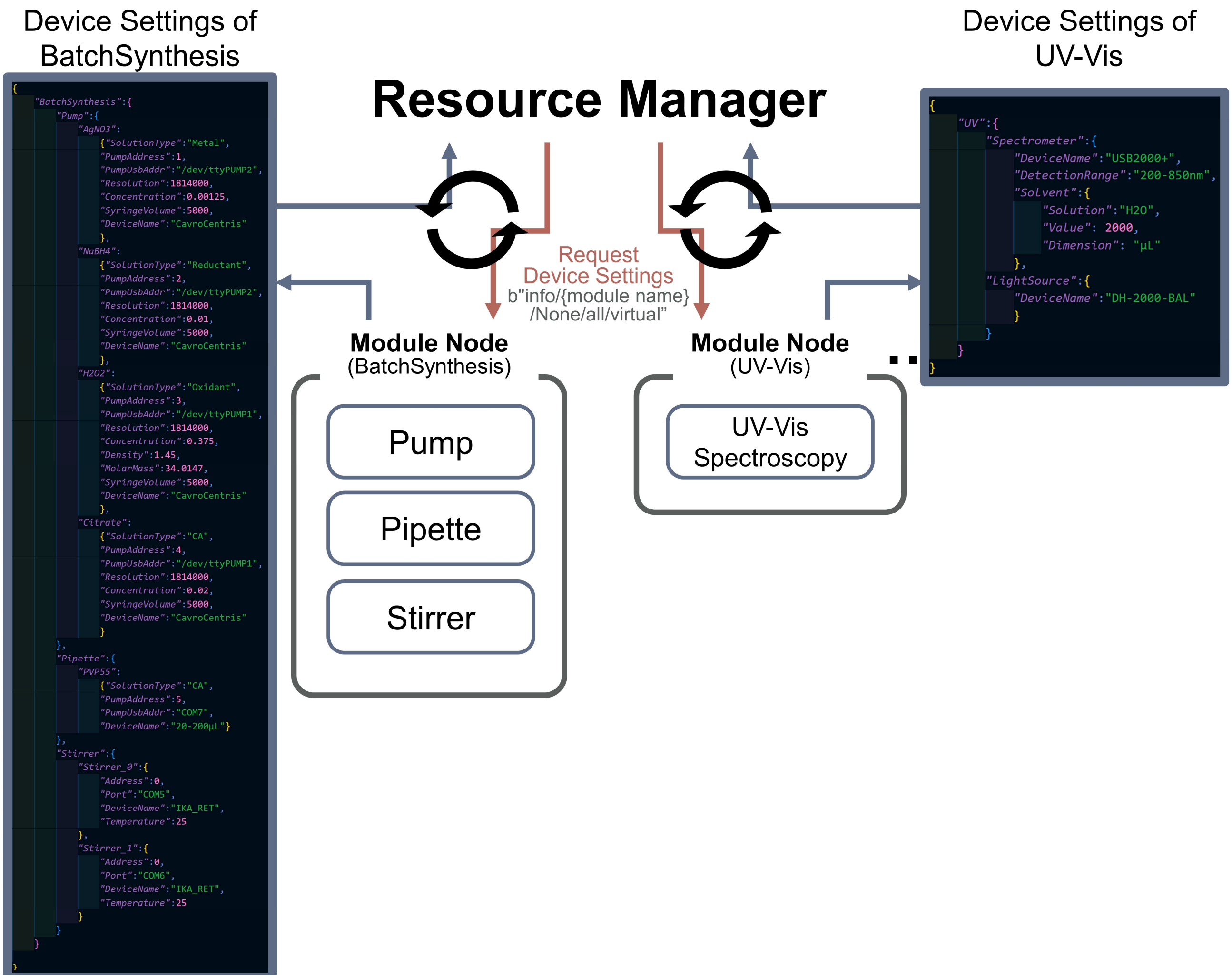
```
2023-09-24 22:36:50,168 - BatchSynthesis::INFO -- [BatchSynthesis] : packet information list:['qhold', '0']
2023-09-24 22:36:50,168 - BatchSynthesis::DEBUG -- [Module Node (UV-Vis)] : qhold:jobID 0 is holded
2023-09-24 22:36:54,042 - BatchSynthesis::INFO -- [BatchSynthesis] : packet information list:['0', 'DS_B', 'storage_empty_to_stirrer', '0,1', 'virtual']
holded jobID :  0
current input_qhold_jobID_list :  [0]
2023-09-24 22:37:02,296 - BatchSynthesis::INFO -- [BatchSynthesis] : packet information list:['qdel', '0']
```

The client can temporarily pause the submitted job or execute the job using the 'qhold' command. When the client enters 'qhold', the master node sends the job ID to the module node due to a job pause. Then, the module node stores the job ID and holds all the device actions of that job ID. If the client enters the 'qrestart' command, the paused job resumes. However, if the client enters the 'qdel' command, the paused job will be deleted.

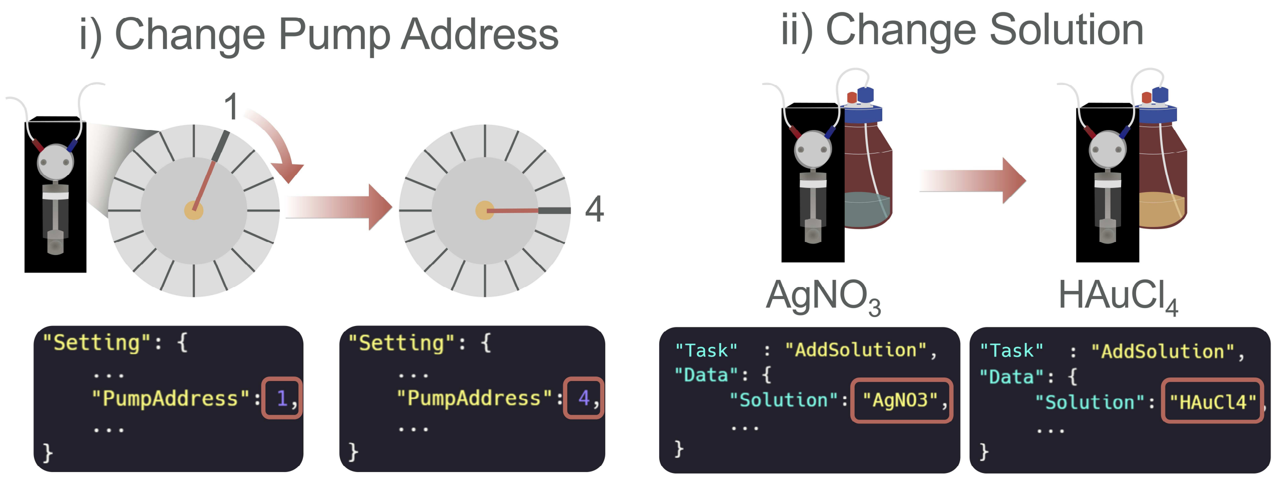
3. Job executions in the master node

Supplementary Figure S7. Functions of the task generator

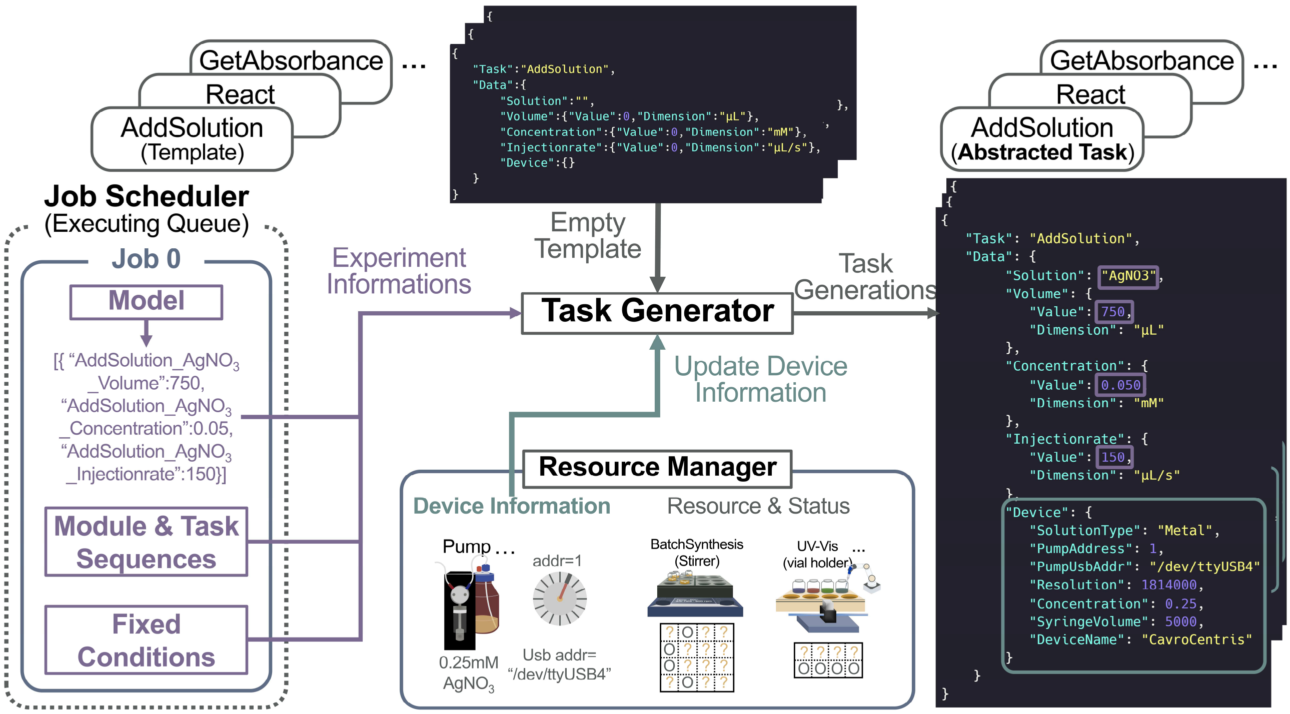
Supplementary Figure S7a. Detailed workflow of update device settings in resource manager



Supplementary Figure S7b. Task reflection depending on latest device information



Supplementary Figure S7c. Detailed workflow of the task generator



Supplementary Figure S7d. Examples of task templates

```
# Robot
self.MoveContainer_template={
    "Task":"MoveContainer",
    "Data":{
            "From":"",
            "To":"",
            "Container":"",
            "Device":{}
        }
}
# BatchSynthesis
self.PrepareContainer_template={
    "Task":"PrepareContainer",
    "Data":{
            "From":"",
            "To":"",
            "Container":"",
            "Device":{}
        }
}
self.AddSolution_template={
    "Task":"AddSolution",
    "Data":{
        "Solution":"",
        "Volume":{
            "Value":0,"Dimension":"μL"
            },
        "Concentration":{
            "Value":0,"Dimension":"mM"
            },
        "Injectionrate":{
            "Value":0,"Dimension":"μL/s"
            },
        "Device":{}
    }
}
self.Stir_template={
    "Task": "Stir",
    "Data": {
        "StirRate": {
            "Value": 0,
            "Dimension": "rpm"
        },
        "Device":{}
    }
}
self.Heat_template={
```

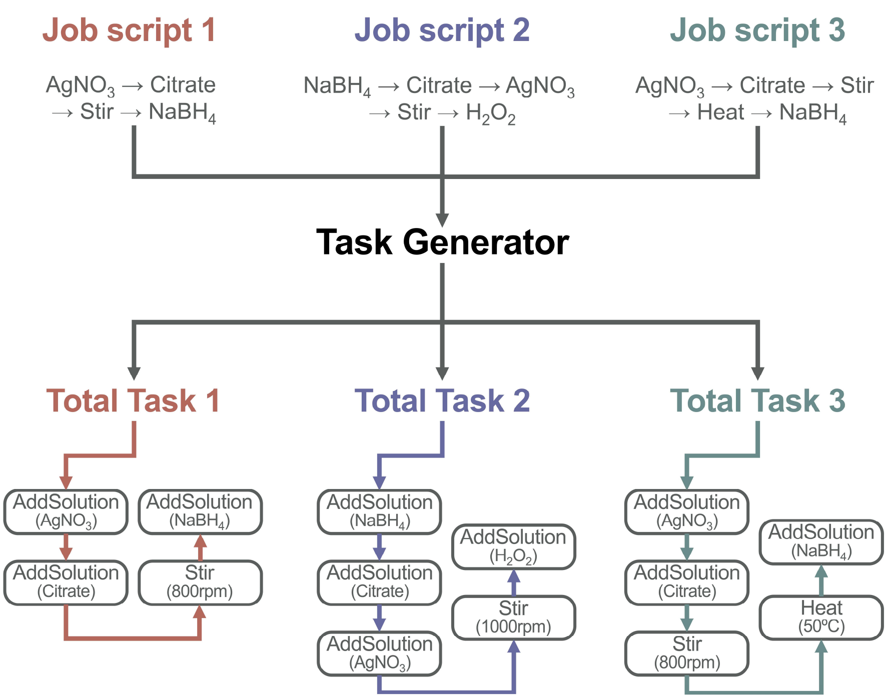
```
    "Task": "Heat",
    "Data": {
        "Temperature": {
            "Value": 0,
            "Dimension": "ºC"
        },
        "Device":{}
    }
}
self.Mix_template={
    "Task": "Mix",
    "Data": {
        "Time": {
            "Value": 0,
            "Dimension": "sec"
        },
        "Device":{}
    }
}
self.React_template={
    "Task": "React",
    "Data": {
        # "To": "",
        "Time": {
            "Value": 0,
            "Dimension": "sec"
        },
        "Device":{}
    }
}
```

```
# UV-Vis
self.GetAbs_template={
    "Task":"GetAbs",
    "Data":{
        "Device":{},
        "Hyperparameter":{
            "WavelengthMin":{
                    "Description":"WavelengthMin=300 (int): slice wavlength section
depending on wavelength_min and wavelength_max",
                "Value": 0,
                "Dimension": "nm"
            },
            "WavelengthMax":{
                    "Description":"WavelengthMax=849 (int): slice wavlength section
depending on wavelength_min and wavelength_max",
                "Value": 0,
                "Dimension": "nm"
            },
```

```
            "BoxCarSize":{
                "Description":"BoxCarSize=10 (int): smooth strength",
                "Value": 0,
                "Dimension": "None"
            },
            "Prominence":{
                  "Description":"Prominence=0.01 (float): minimum peak Intensity for
detection",
                "Value": 0,
                "Dimension":"None"
            },
            "PeakWidth":{
                "Description":"PeakWidth=20 (int): minumum peak width for detection",
                "Value": 0,
                "Dimension": "nm"
            }
        }
    },
}
```

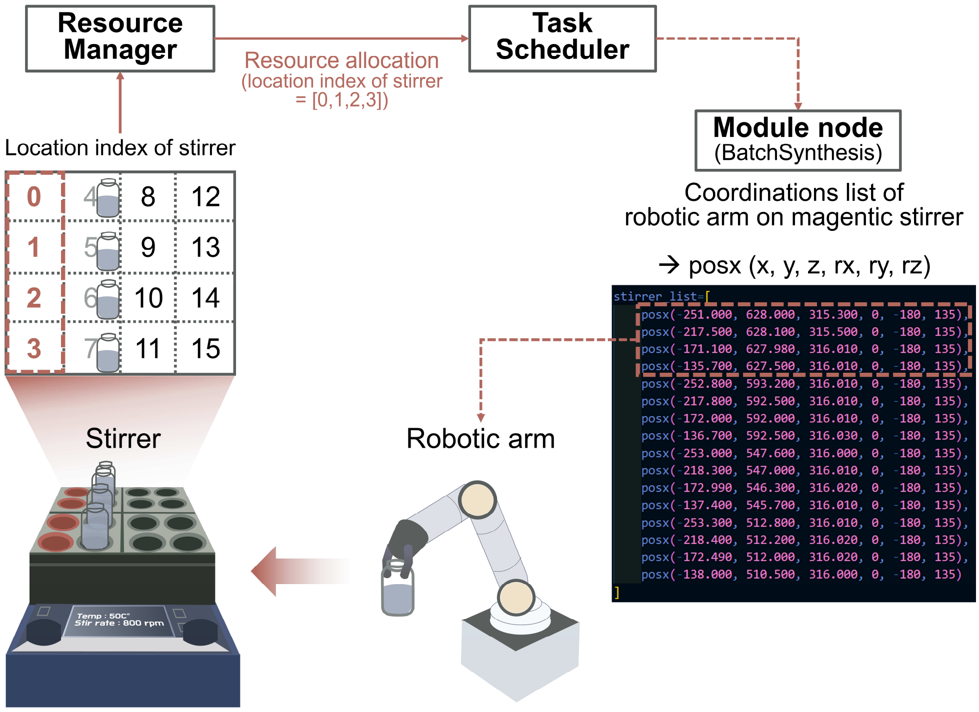
The origin of task template has empty value. The examples of template uploaded in Github repository. (https://github.com/KIST-CSRC/Octopus)

Supplementary Figure S7e. Conversion of job script in terms of task sequences in the task generator

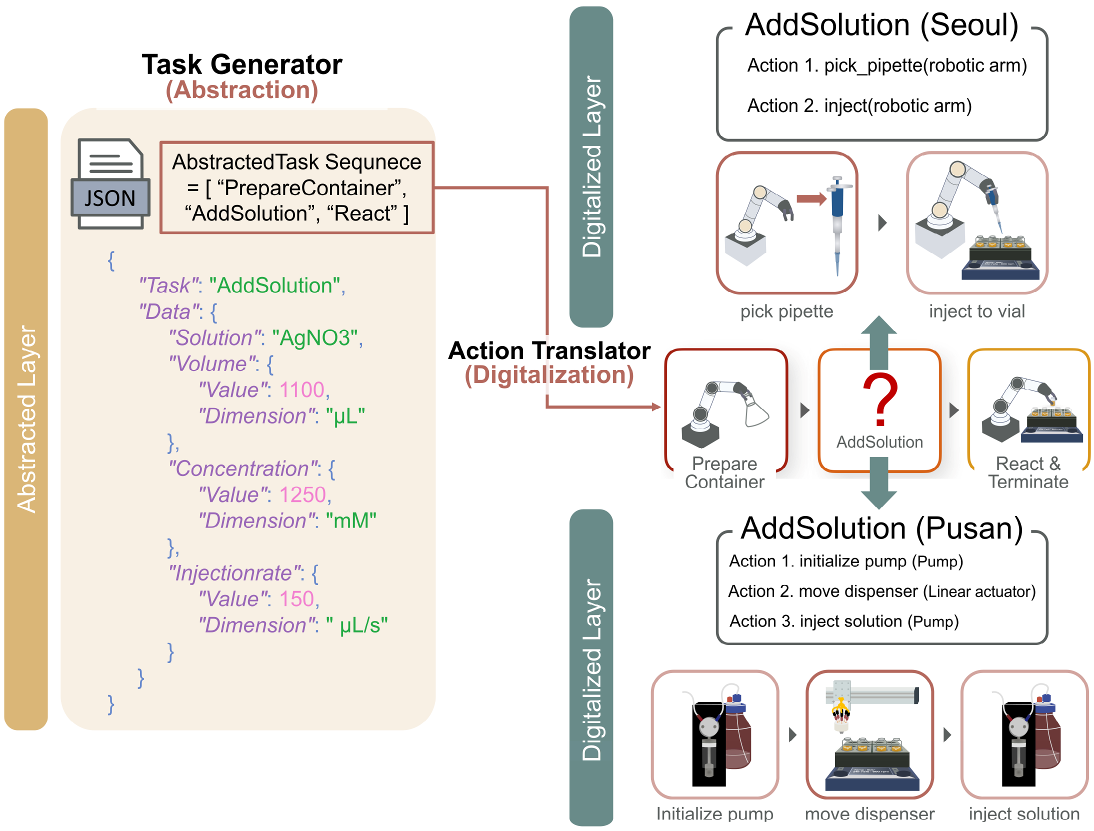


A task generator allows clients to create process recipes based on their desired experimental conditions and process sequences. To execute various process sequences for each job script, we would present the generated recipe in the JSON data format shown in the GitHub repository.

Supplementary Figure S8. Examples of resource allocation in batch synthesis module



Supplementary Figure S9. The role of action translator for abstraction and digitalization

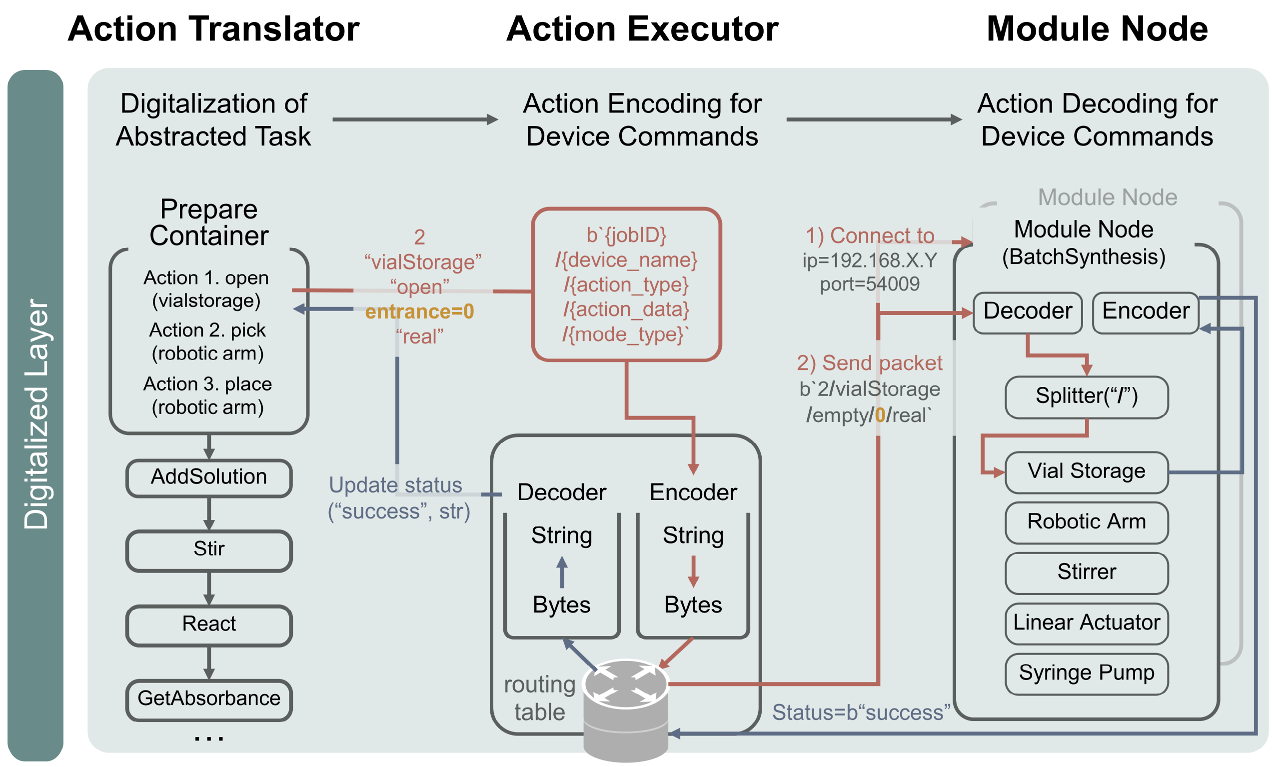


Supplementary Figure S10. Functions of the action executor

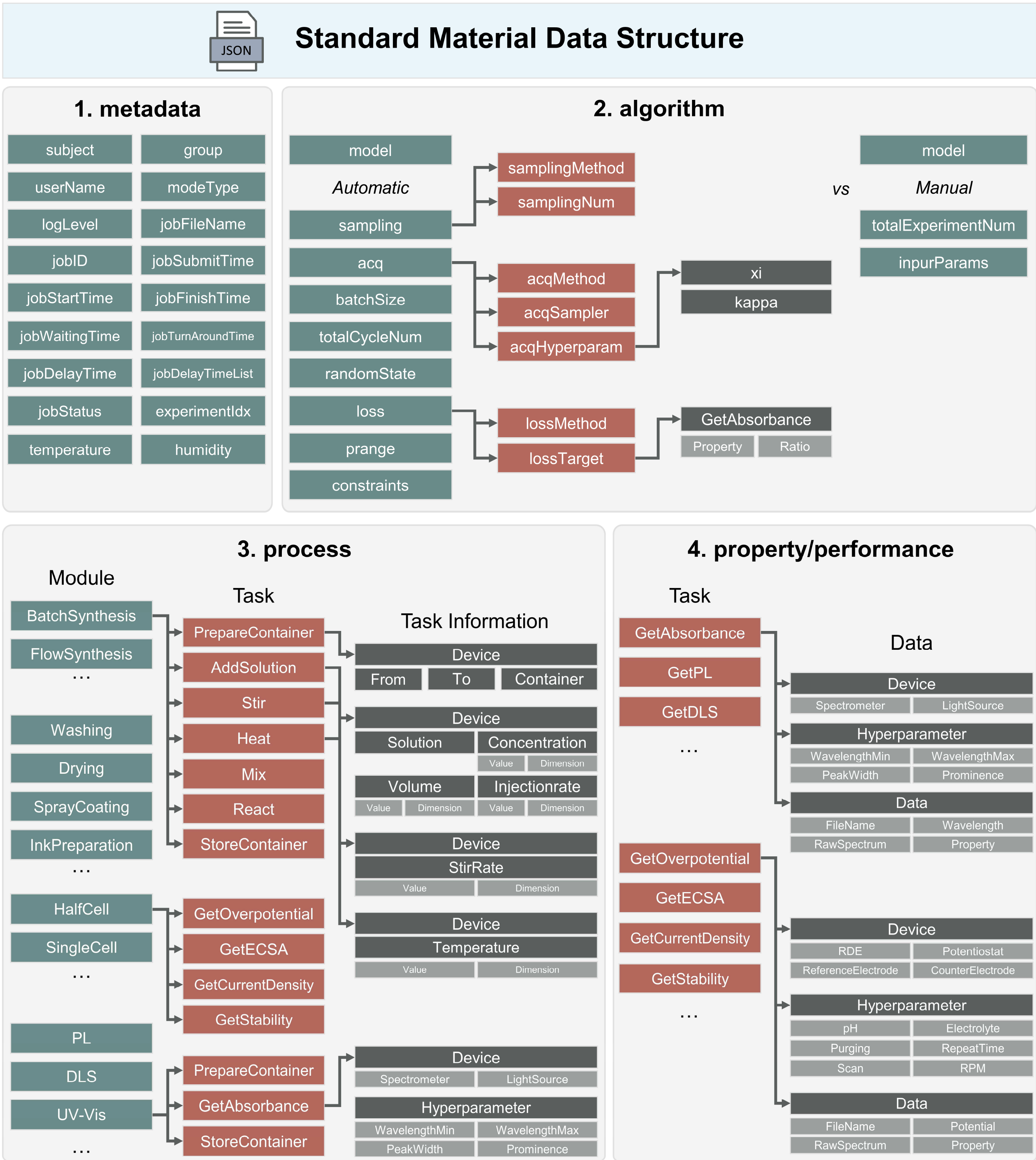
Supplementary Figure S10a. Predefined device commands of the action executor

Device	Pre-defined data transfer formats	Example
Robotic Arm (DooSan_Robotics)	{jobID}/DS_B/{pick_and_place type}/{pick_num,place_num}/{mode_type}	0/DS_B/storage_empty_to_stirrer/0&2/real
Vial Storage	{jobID}/STORAGE/open/1/{mode_type}	0/STORAGE/open/5/real
Pump	{jobID}/PUMP/single/{solution_name,volume,concentration,injection_rate/{mode_type}	4/PUMP/single/AgNO3&1500&0.05&100/real
Stirrer	{jobID}/STIRRER/{stir_type or heat_type}/{stirrer_address,temperature}/{mode_type}	3/STIRRER/Heat/0&25/real
LA	{jobID}/LA/{move_type}/{location_index}/{mode_type}	1/LA/down/0/real
UV-Vis Spectroscopy	{jobID}/SPECTROSCOPY/Abs/{element}/{mode_type}	6/SPECTROSCOPY/Abs/Ag/real
Pipetting Machine	{jobID}/PIPETTE/sample/{pipette_volume,inject_volume, tip_loc,pump_in_loc,pump_out_loc,mixing_time}/{mode_type}	2/PIPETTE/sample/&2-20&2&5&0&3&3/real
…	…	…

Supplementary Figure S10b. Detailed workflow of the action executor



Supplementary Figure S11. Hierarchy structure of the generated material data



The second auxiliary function is to store the results of the experiments in individual JSON files, and is conducted during the job cycle. (Figure S12a, Supplementary Information) The reason for choosing JSON is its flexible, computer-readable structure. In the MAP for chemical experiments, there are various modules, each containing diverse tasks, device types and device settings. Given this variability, storing the material data in a structured format within a relational database would be challenging. In other words, tabular data with a relational database to store the task information in MAPs, inevitably face quite a sparse table due to the tremendous diversity of the attributes and methods from the various device settings and module components.

Therefore, we implement a nonstructured data format with inclusivity for the data structure of MAP. The most significant feature of JSON is its hierarchical structure. A well-defined JSON format enhances the flexibility of storing attributes/methods of diverse tasks and aids in data readability, making it easier for clients to understand the processes. Furthermore, the JSON format is a commonly used data specification on the web, so it helps to easily convert CLI to a web-based interface. We designate the standard material data structure with four main hierarchical keys—metadata, algorithms, processes and property/performances—as the highest-level categories. To enable the utilization of the accumulated data for AI-driven experimental planning in the future, we implement MongoDB, which stores and manages the JSON data.

4. Network-protocol-based modularization

Supplementary Figure S12. Process modularization for homogeneity via device server

Supplementary Figure S12a. An actual example of heterogeneous environments

Module Node

Device Server

Module Node = BatchSynthesis

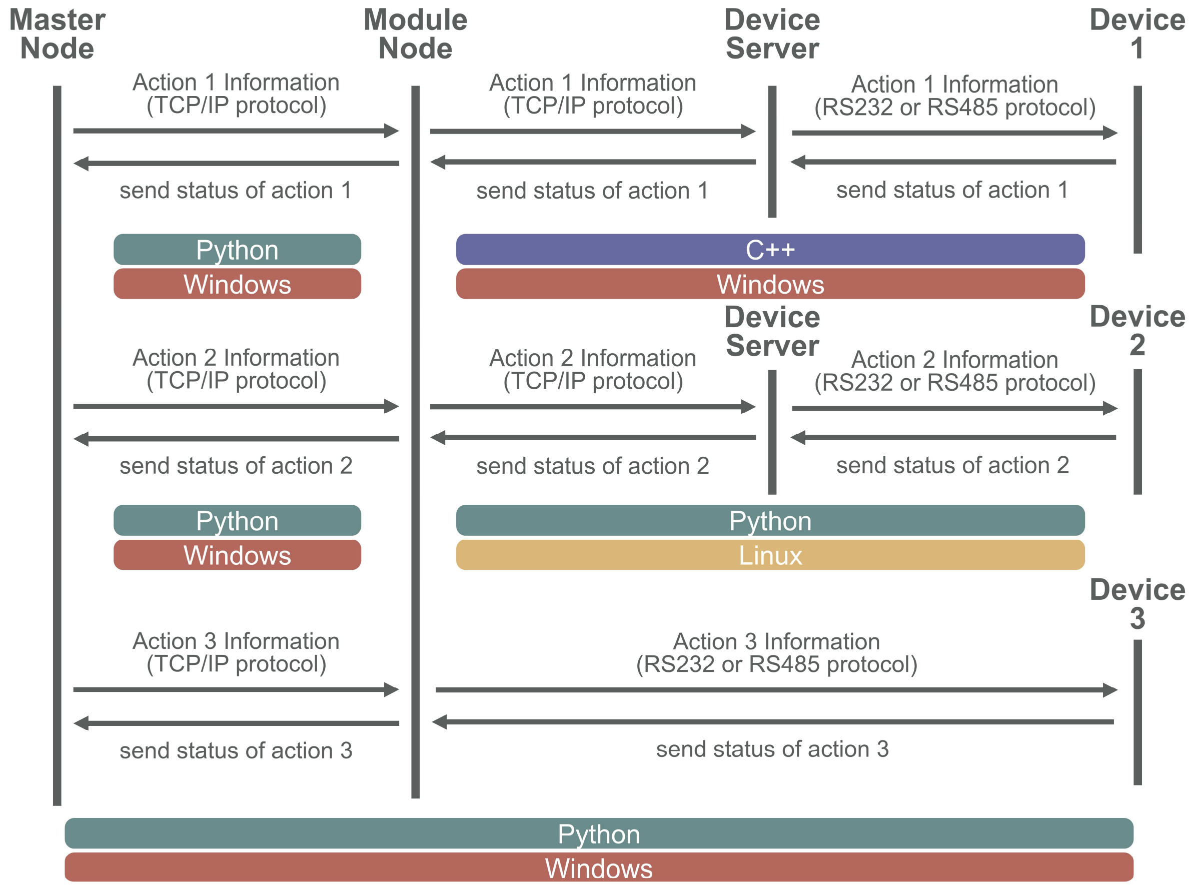
Operating System	Doosan Robotics (Robotic Arm)	XYZ Linear Actuator	Vial Storage	Stirrer	Syringe Pump (Solution Dispenser)
Linux	O	X	O	O	O
Windows	X	O	O	O	O
Programming Language	-	-	-	-	-
Python	O	X	O	O	O
C++	O	O	X	X	X

Module Node = UV-Vis

Operating System	Doosan Robotics (Robotic Arm)	uArm Swift Pro (pipetting machine)	Linear Actuator (pipette)	UV-Vis Spectroscopy
Linux	O	O	X	O
Windows	X	O	O	O
Programming Language	-	-	-	-
Python	O	O	O	O
C++	O	X	X	O

We configured the module node using Python based on the Windows OS. Communication between the environment of the module node and other devices is designed to set up a device server, enabling interaction with the module node. The table represents the configuration of our module node, as presented in our recent research.[3]

Supplementary Figure S12b. Virtual workflow of the network protocol with a hierarchical structure



This workflow depends on device settings of modules.

Supplementary Figure S13. Process modularization for scalability via internal and external network

Supplementary Figure S13a. Utilization of the internal network protocol in the routing table

Process	Internal IP address
Synthesis	192.168.1.X
Preprocess	192.168.2.X
Evaluation	192.168.3.X
Characterization	192.168.4.X
Database	192.168.5.X

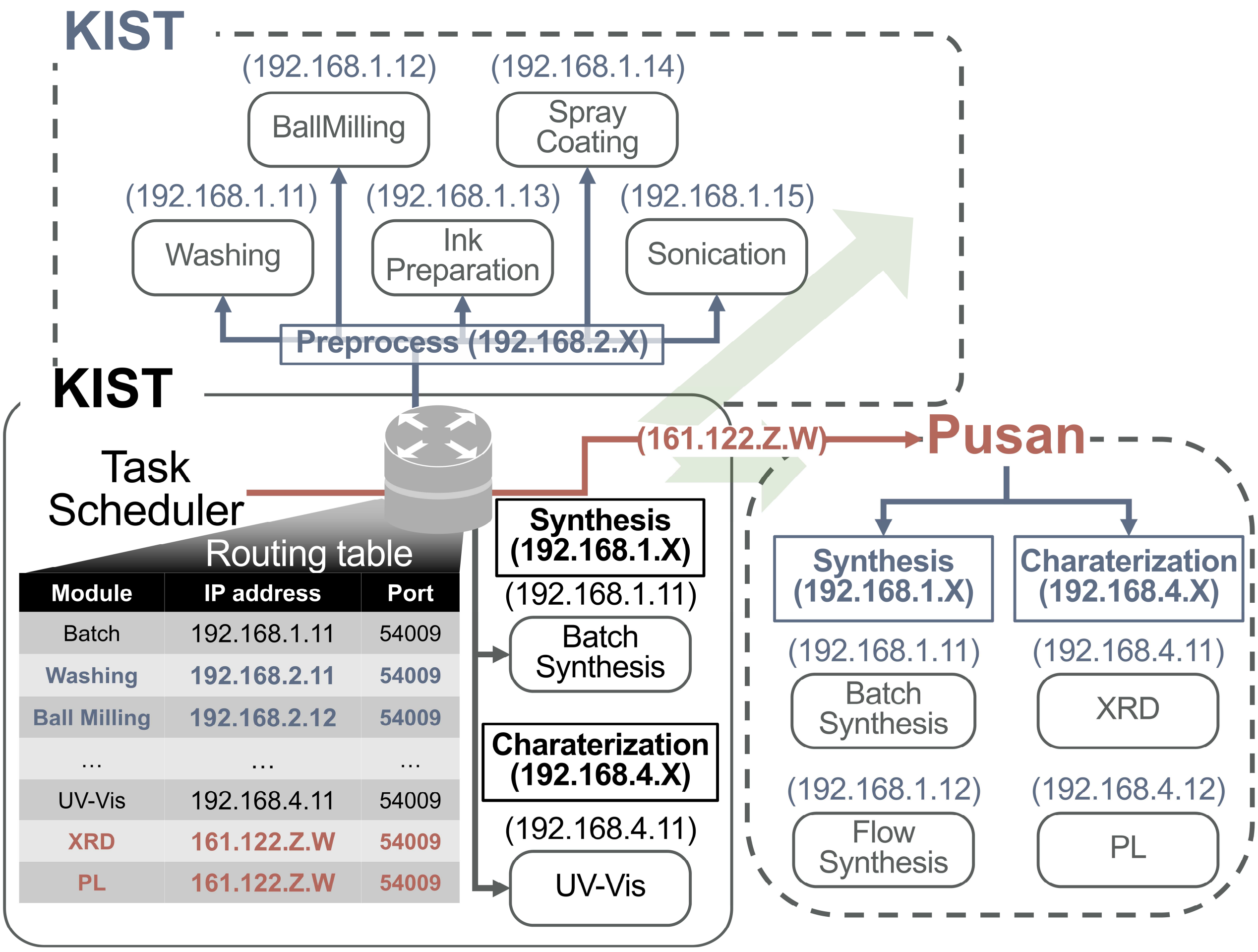
Module (Preprocess)	Internal IP address
InkPreparation	192.168.2.11
BallMilling	192.168.2.12
Washing	192.168.2.13
…	…

Module (Preprocess → Washing)	Internal IP address	Internal PORT
Pipette (Module PC)	192.168.2.13	54009
Sonication (Module PC)	192.168.2.13	54009
Centrifuge (Device Server 1)	192.168.2.13	54011
Robotic arm (Device Server 2)	192.168.2.13	54012
…	…	…

The third number represents the type of experimental process, and the fourth number represents the module information. The types of experimental processes include synthesis, preprocessing, evaluation, characterization and database, which are commonly defined in chemical experiments. These experimental processes were represented by numbers from 1 to 5, making it easy for researchers to identify which process they belong to via the third number alone. Starting from the fourth number, different modules are included in the same process type. For example, within the same preprocessing category, various modules might be grouped, such as processes for washing, ball milling, ink preparation, spray coating and sonication. In the fourth part of the internal network address, the presence of '1' represents the gateway. Consequently, modules should be sequentially recorded in the routing table starting from '11'.

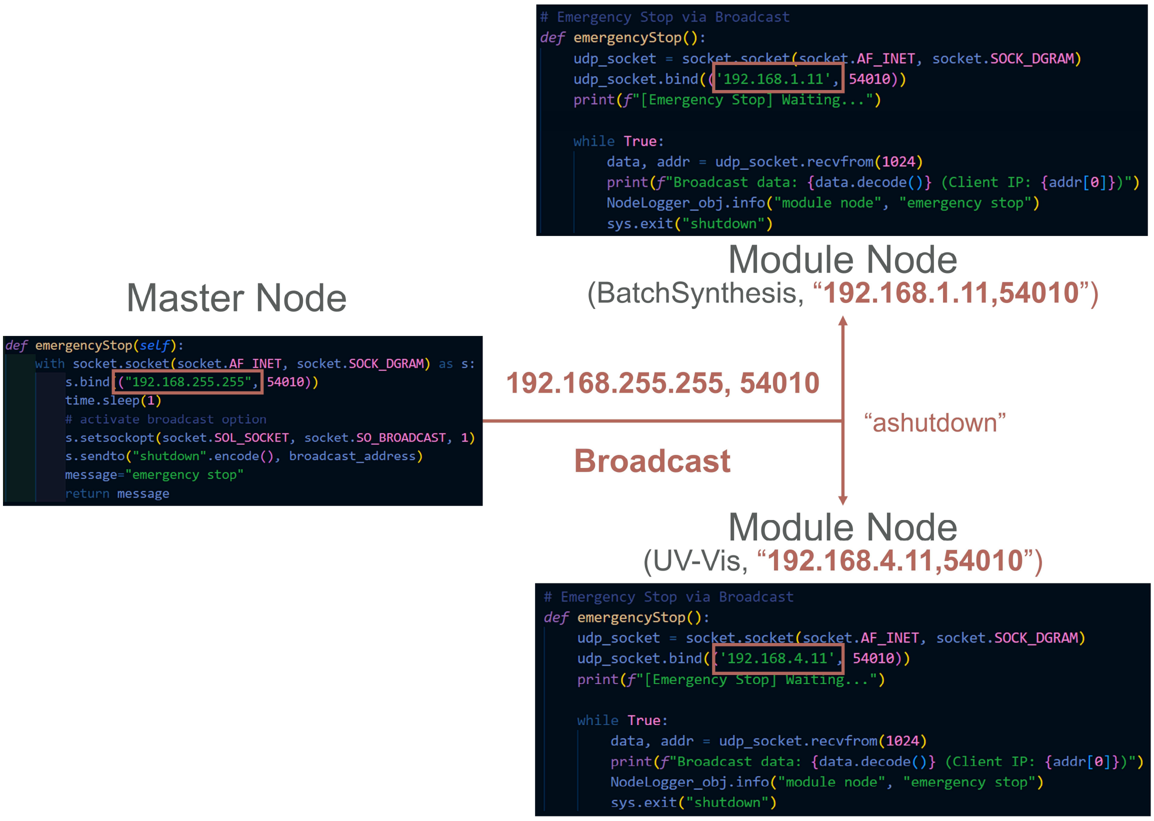
In network protocols, broadcast refers to a method of sending messages or data packets across a network to multiple computers or network devices simultaneously via a single transmission. This means that the information is sent to all the devices on the network without specifying a specific target computer or device. Broadcast is typically used for network management, debugging, or when the same message needs to be sent to multiple devices. We set the internal network address of the emergency stop to “192.168.255.255”. The term “255” enables broadcasting via an internal network protocol.

Supplementary Figure S13b. Virtual examples of scalable autonomous experiment platform based on internal and external network



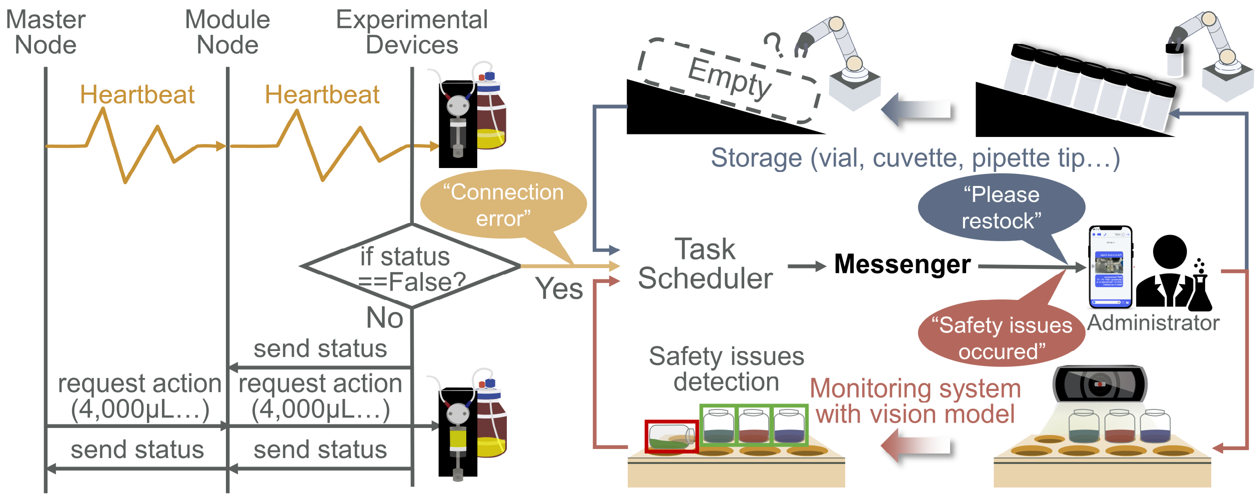
Supplementary Figure S14. Process modularization and utilization for safety.

Supplementary Figure S14a. Broadcast-enabled module node shutdown

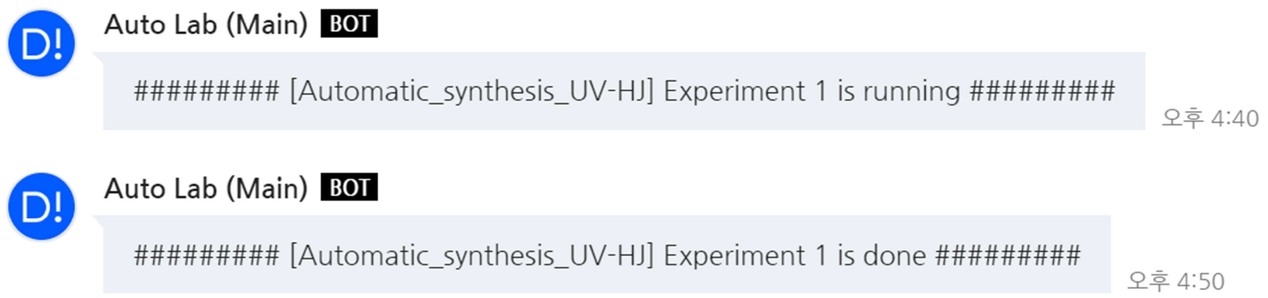


In network protocols, broadcast refers to a method of sending messages or data packets over network to multiple computers or network devices simultaneously via a single transmission. This means that the information is sent to all devices on the network without specifying a specific target computer or device. Broadcast is typically used for network management, debugging, or when the same message needs to be sent to multiple devices. We set internal network address of emergency stop to "192.168.255.255". "255" enables broadcast in internal network protocol.

Supplementary Figure S14b. Workflow of the safety alert system

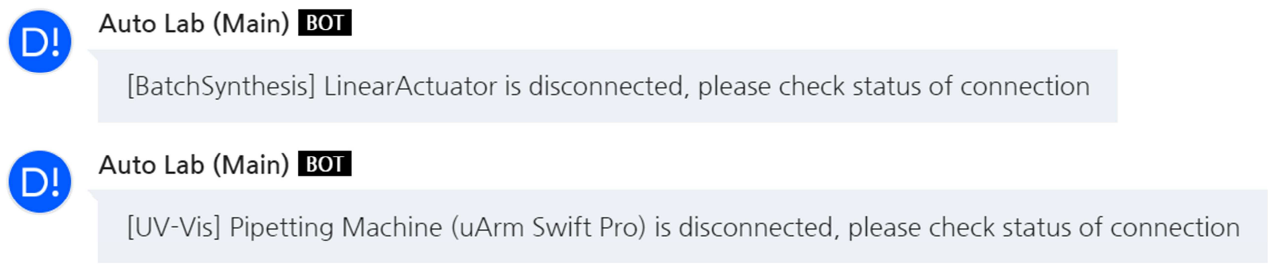


Supplementary Figure S14c. Real messages of the alert system showing the experiment progress



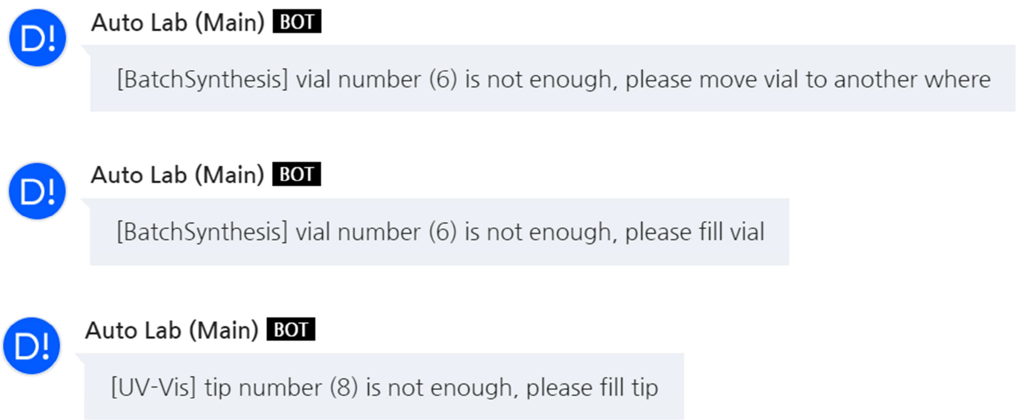
This message is the result of communication through Dooray Messenger.[4]

Supplementary Figure S14d. Real messages of the alert system for device disconnection via heartbeat



This message is the result of communication through Dooray Messenger.[4]

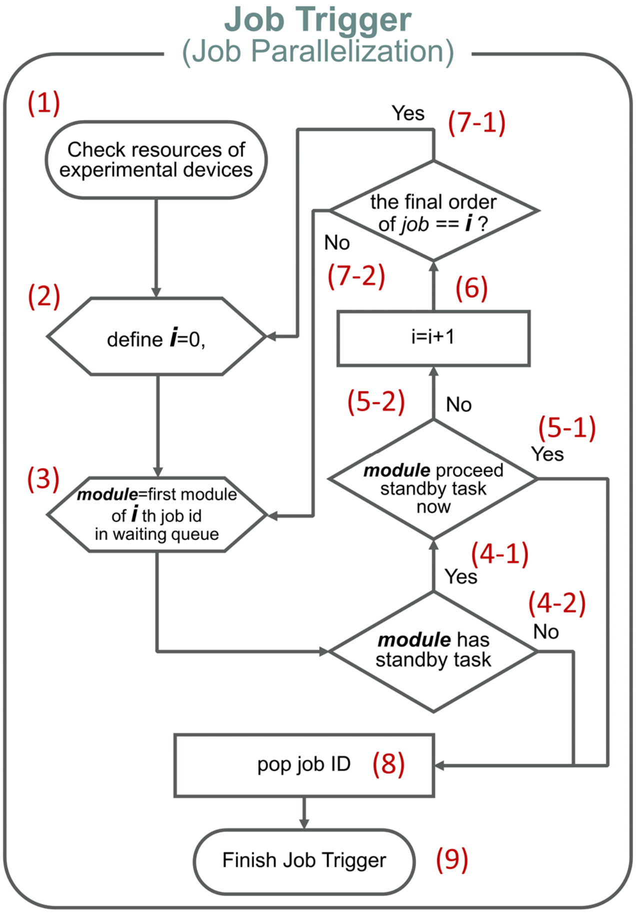
Supplementary Figure S14e. Real messages of the alert system via the restock function for chemical vessels



This message is the result of communication through Dooray Messenger.[4]

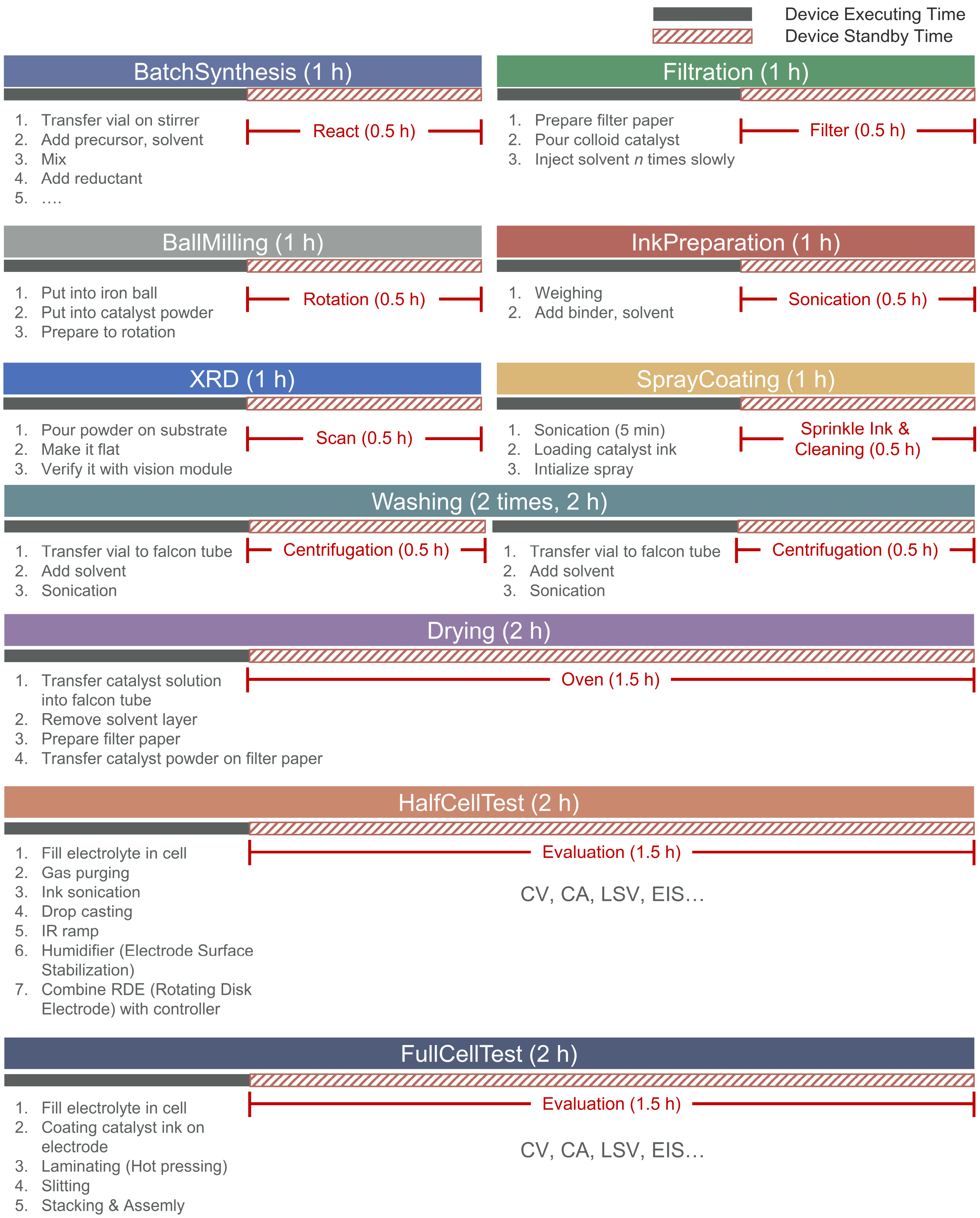
5. Job parallelization to address the module overlap challenge

Supplementary Figure S15. Schematic algorithm of job parallelization

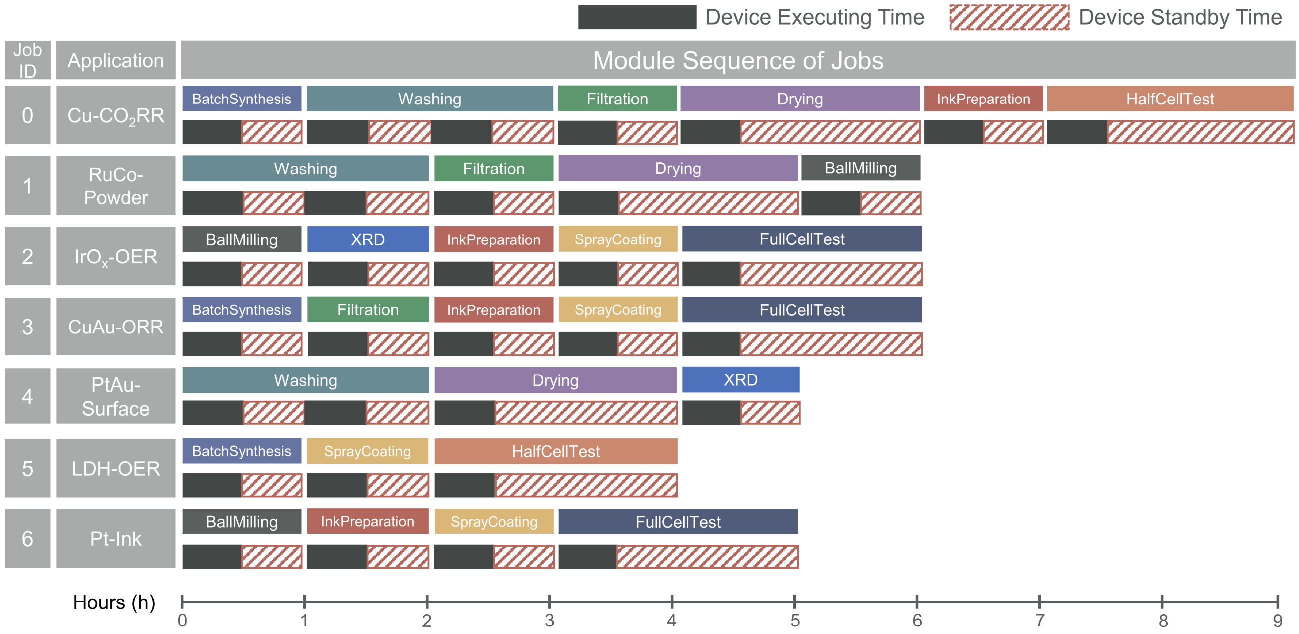


1. Check the resources of the experimental equipment set on the modules.
2. Set an index ***i*** (the index ***i*** represents the position index of the stacked job ID in the waiting queue).
3. The name of the module that is to be executed is retrieved first according to the predefined module execution order from the job, with the job ID corresponding to the ***i***-th position within the waiting queue. This is denoted as a ***"module"*** in the diagram.
4. Does the ***"module"*** own a device standby task?
    4-1. If yes, proceed to (5).
    4-2. If not, proceed to (8).
5. Is the ***"module"*** executing a device standby task?
    5-1. If yes, proceed to (8).
    5-2. If not, proceed to (6).
6. The index ***i*** is incremented by 1.
7. Determine if index ***i*** is the last.
    7-1. If ***i*** is the last order, return to (2) → Reset ***i*** to check the job with the job ID corresponding to the first order in the waiting queue.
    7-2. If ***i*** is not the last order, proceed to (3) → Search for the first module to execute in the job with the job ID corresponding to the next order in the waiting queue.
8. Pop the job ID corresponding to index ***i*** from the waiting queue.
9. End the job trigger.

Supplementary Figure S16. The timeline of the modules used for catalyst development included device standby time



Supplementary Figure S17. Multiple jobs for catalyst application



Supplementary Figure S18. Definition of performance metrics

Supplementary Table S1. Performance metrics of job parallelization

Supplementary Table 1a. Job waiting time between serialization and parallelization

Job ID	Serialization (h)	Parallelization (h)
0	0	0
1	0	0
2	0	0
3	1	0.5
4	4	0.5
5	2	1
6	1	0.5

Supplementary Table 1b. Job turnaround time between serialization and parallelization

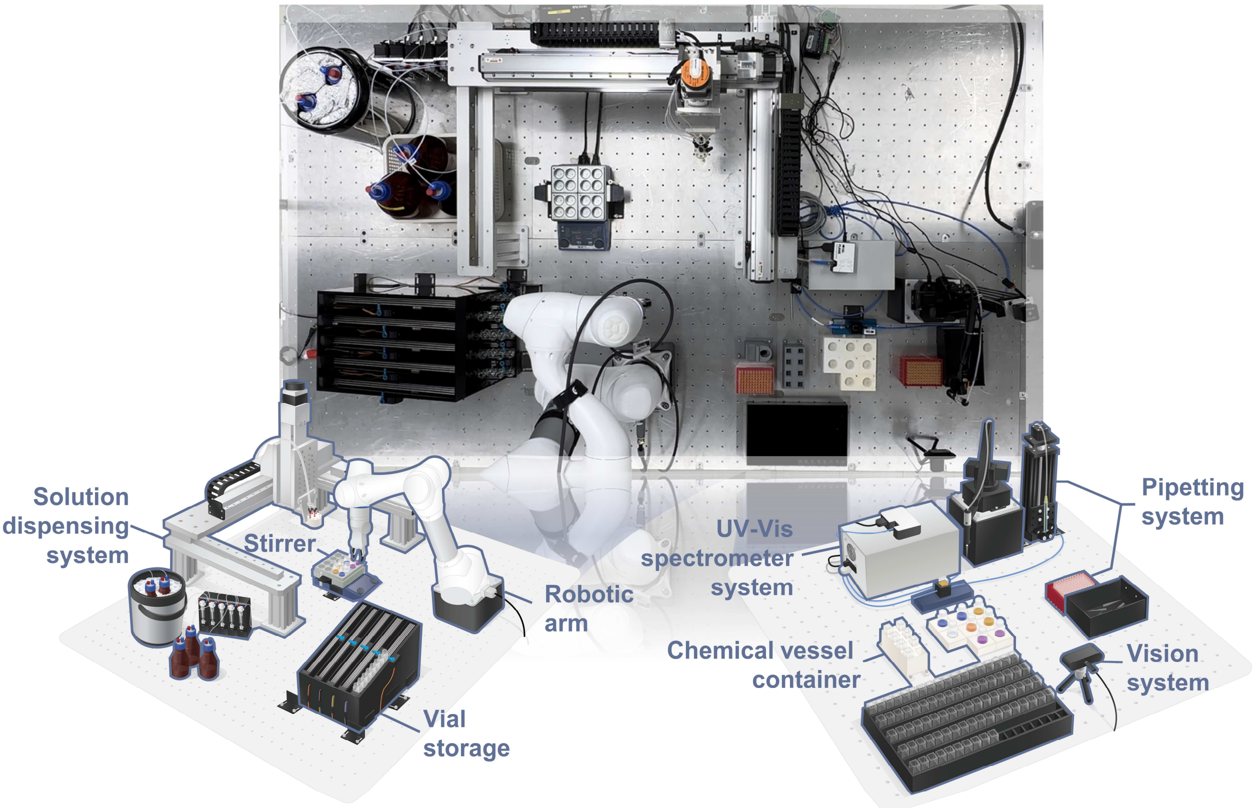
Job ID	Serialization (h)	Parallelization (h)
0	10	9
1	5	5
2	6	6
3	9	6
4	6	5
5	5	4
6	7	5

Supplementary Table 1c. Job total time between serialization and parallelization

Job ID	Serialization (h)	Parallelization (h)
0	10	9
1	5	5
2	6	6
3	10	6.5
4	10	5.5
5	7	5
6	8	5.5

6. Task optimization for preventing device overlaps.

Supplementary Figure S19. A bird's eye view image of modules including "BatchSynthesis" and "UV-Vis"

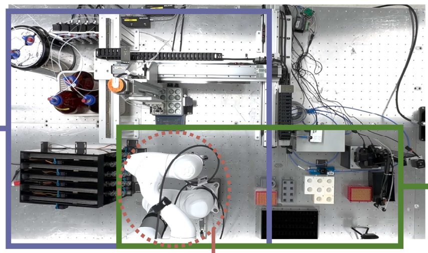


This image is a module used in a previous study[3].

Supplementary Figure S20. Computing results of Boolean operation in python

Boolean	Operation	Boolean	Result
True	* (and)	True	1=True
True	* (and)	False	0=False
False	* (and)	True	0=False
False	* (and)	False	0=False

Supplementary Figure S21. Real examples of masking table



Module: **BatchSynthesis** (Tasks: PrepareContainer, AddSolution) — Device Sharing (Robotic Arm) — **UV-Vis** (Task: PrepareCuvette)

Task / Masking Table

Device	PrepareContainer Mask	AddSolution Mask	PrepareCuvette Mask
Batch_Robotic Arm	T	F	T
Batch_Vial Storage	T	F	F
Batch_Linear Actuator	T	T	F
Batch_Pump	F	T	F
UV_Robotic Arm	T	F	T
UV_Pipette	F	F	T
UV-Vis	F	F	F

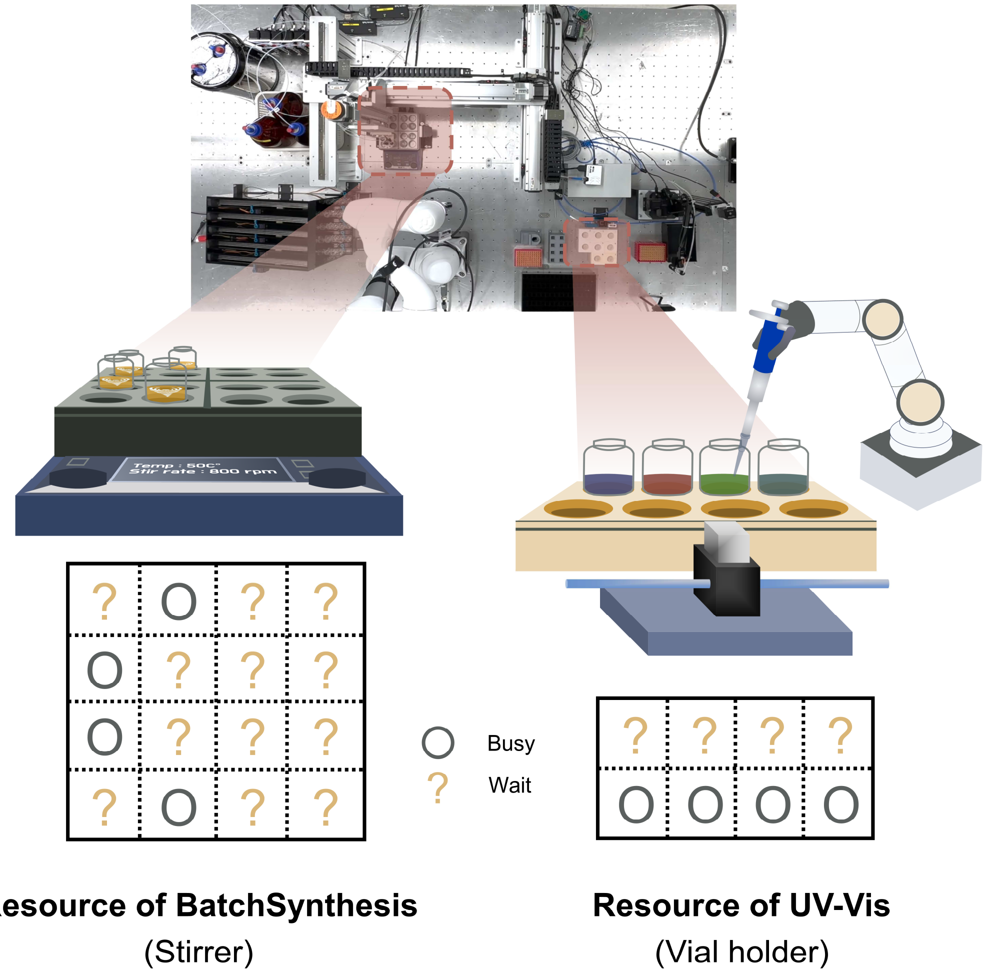
Task / Masking Table

Device	Mix Mask	React Mask	GetAbsorbance Mask
Batch_Robotic Arm	F	F	F
Batch_Vial Storage	F	F	F
Batch_Linear Actuator	F	F	F
Batch_Pump	F	F	F
UV_Robotic Arm	F	F	T
UV_Pipette	F	F	T
UV-Vis	F	F	T

The “BatchSynthesis” module has four resources which include the number of vials that can be processed in the magnetic stirrer. The “UV-Vis” module has also four resources which represents the number of vial holders storing vials for “UV-Vis” spectrum measurements. The image of hardware is a module used in a previous study[3].

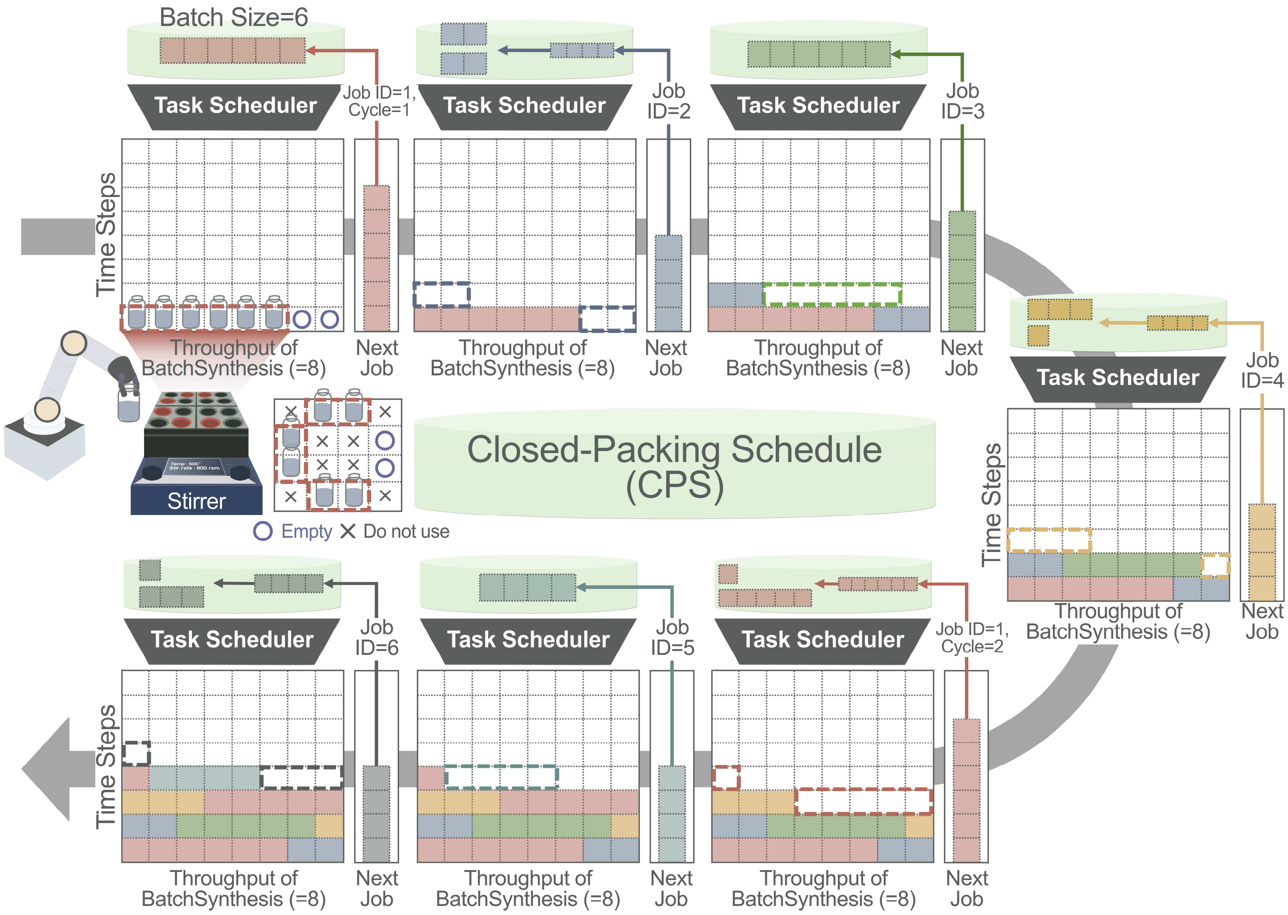
7. The closed-packing schedule for optimizing module resource

Supplementary Figure S22. Definition of resource in realistic platform



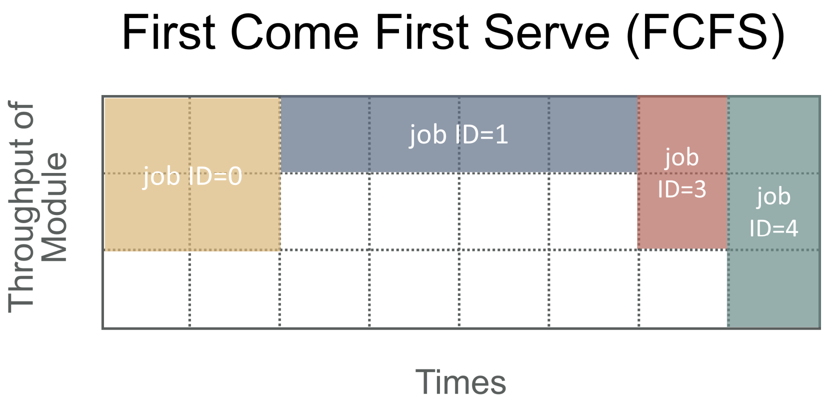
The batch synthesis module has four resources which include the number of vials that can be processed in the magnetic stirrer. The UV-Vis module has also four resources which represents the number of vial holders storing vials for UV-Vis spectrum measurements. The image of hardware is a module used in a previous study[3].

Supplementary Figure S23. Detailed workflow of closed-packing schedule in multi jobs



8. Performance test of user-optimal schedulers

Supplementary Figure S24. Schematic design of conventional scheduling algorithm (FCFS)

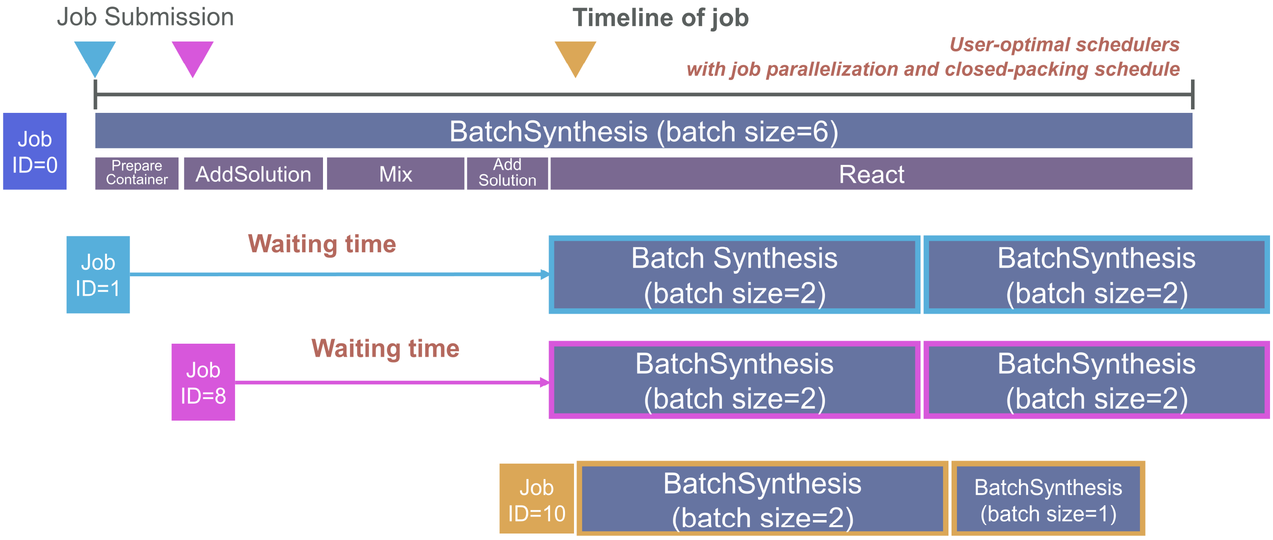


Supplementary Figure S25. Job information for benchmark test of user-optimal schedulers

(job ID) Job Submission Time	(job ID) Job Start Time	(job ID) Job Finish Time	Experiment Type	Module Selection	Batch Size	Numbers of Cycle	Task Time with Device Standby Time	The Number of AddSolution (BatchSynthesis)
(0) 09:00	(0) 09:00	(0) 18:45	Automated (Closed-loop)	Batch Synthesis, UV-Vis	6	3	React (120 min)	4
(1) 09:07	**(1) 09:45**	(1) 11:47	Manual	Batch Synthesis	4	1	React (40 min)	4
(2) 09:54	(2) 09:54	(2) 10:12	Manual	UV-Vis	4	1	-	-
(3) 10:21	(3) 10:21	(3) 11:02	Manual	UV-Vis	5	1	-	-
(4) 10:21	(4) 10:21	(4) 11:10	Manual	UV-Vis	6	1	-	-
(5) 12:47	**(5) 13:00**	(5) 14:38	Manual	Batch Synthesis	2	1	React (80 min)	3
(6) 12:46	**(6) 12:50**	(6) 13:26	Manual	UV-Vis	8	1	-	-
(7) 14:20	(7) 14:20	(7) 14:47	Manual	UV-Vis	6	1	-	-
(8) 15:32	**(8) 16:15**	(8) 17:25	Manual	Batch Synthesis	4	1	React (20 min)	2
(9) 16:53	(9) 16:53	(9) 17:11	Manual	UV-Vis	4	1	-	-
(10) 17:34	(10) 17:34	(10) 18:58	Automated (Closed-loop)	Batch Synthesis, UV-Vis	3	1	React (20 min)	3

The bold text corresponds to "job waiting time" and the underline text corresponds to "job turnaround time".

Supplementary Figure S26. Residual resources-based job split via closed-packing schedule in user-optimal schedulers



Supplementary Table S2. Performance test in realistic platform

Supplementary Table 2a. Result of job waiting time in realistic platform

Job ID	FCFS (h)	User-Optimal Schedulers (h)
0	0	0
1	9.75	0.63
2	11.04	0
3	11.34	0
4	11.72	0
5	12.17	0.72
6	13.81	0.07
7	14.41	0
8	15.16	0.72
9	14.86	0
10	15.92	0

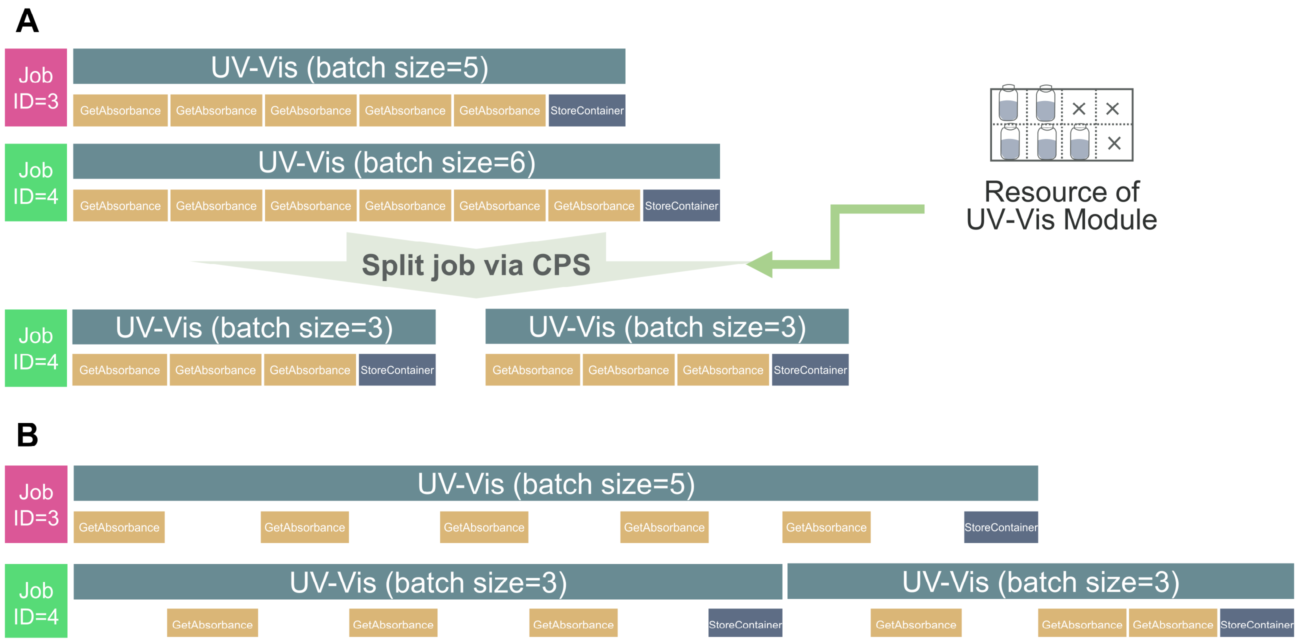
Supplementary Table 2b. Result of job turnaround time in realistic platform

Job ID	FCFS (h)	User-Optimal Schedulers (h)
0	9.75	9.75
1	1.29	2.04
2	0.3	0.3
3	0.38	0.68
4	0.45	0.83
5	1.64	1.64
6	0.6	0.6
7	0.45	0.45
8	0.76	1.18
9	0.3	0.3
10	0.93	1.39

Supplementary Table 2c. Result of job total time in realistic platform

Job ID	FCFS (h)	User-Optimal Schedulers (h)
0	9.75	9.75
1	11.04	2.67
2	11.34	0.3
3	11.72	0.68
4	12.17	0.83
5	13.81	2.36
6	14.41	0.67
7	14.86	0.45
8	15.92	1.89
9	15.16	0.3
10	16.84	1.39

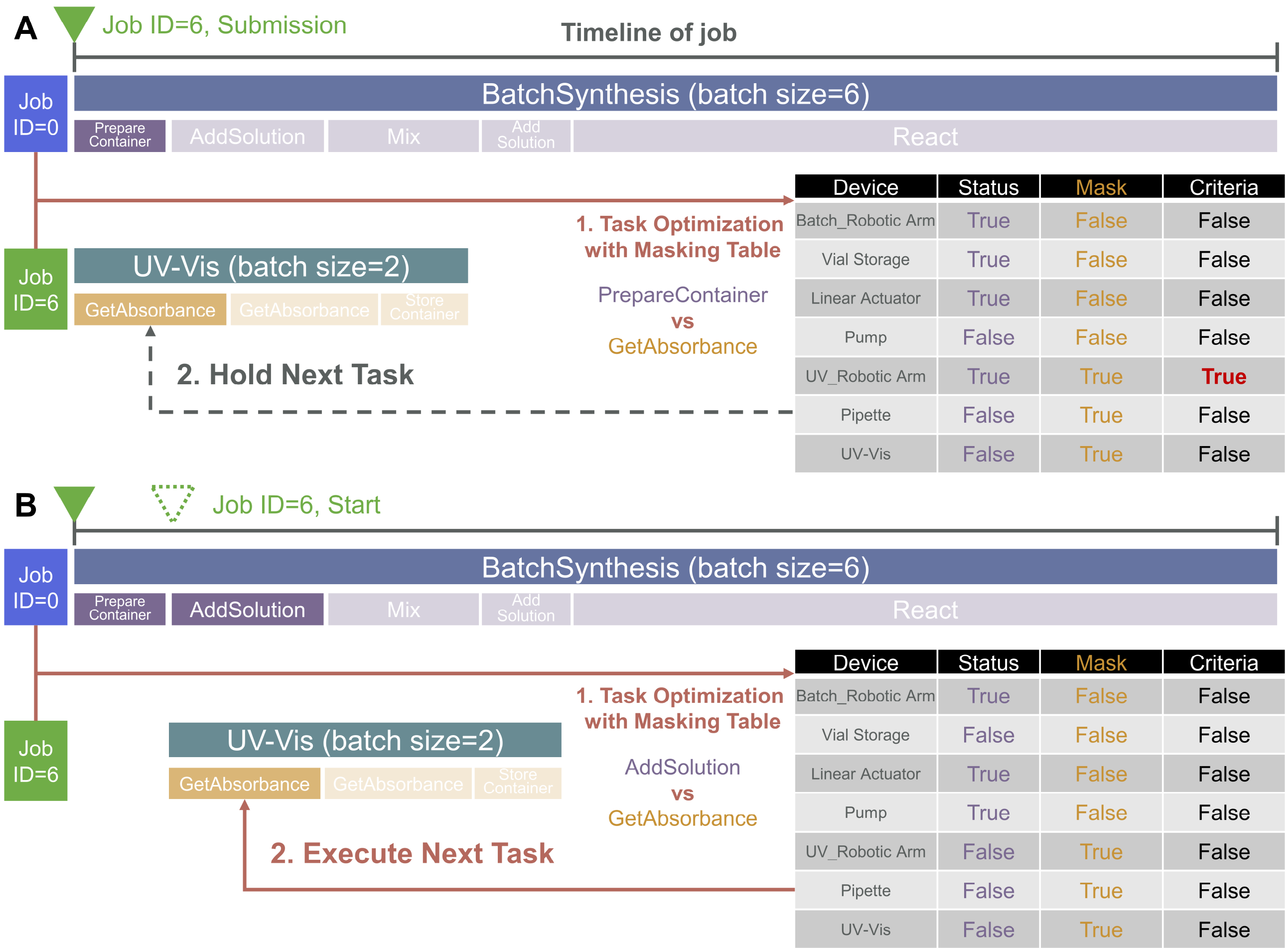
Supplementary Figure S27. Results of closed packing schedule in UV-Vis module



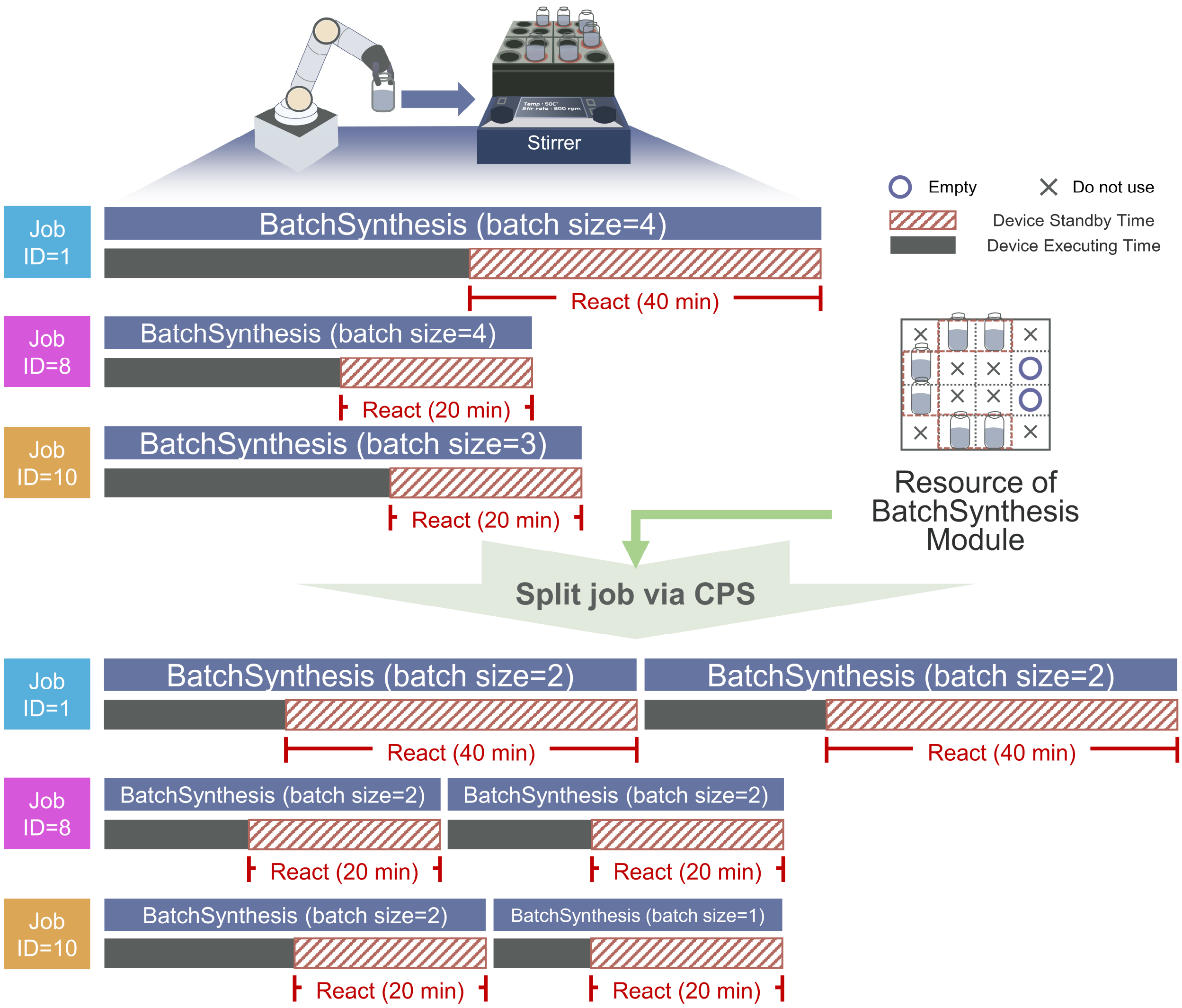
(A) Job split by residual resource-based CPS.

(B) Job parallelization in characterization module without device standby times.

Supplementary Figure S28. Results of task optimization between batch synthesis and UV-Vis module



Supplementary Figure S29. Cause analysis for the delay time of CPS in batch synthesis module



9. Copilot of OCTOPUS

Supplementary Figure S30. Reusability comparison with and without Copilot of OCTOPUS

O = Reusable
▲= Partial modification required
X = Complete modification required

Component	Reusability without Copilot of OCTOPUS	Reusability with Copilot of OCTOPUS
Job scheduler	O	O
Task generator	X	O
Task scheduler	O	O
Action translator	X	▲ (considered robotic-lab setups)
Action executor	▲	O
Resource manager	X	O
Module node	X	▲ (considered robotic-lab setups)

Supplementary Table S3. The description of inputs and outputs of GPT

Inputs in GPT prompt	Outputs as GPT answer
Module name	Actions of each device in module node or device server
All device names in module	
Module name,	Tasks of module
Module description	
Needed task in module information	
Module name	Action sequence for task execution
Generated tasks	
Generated actions	
Module name	Task template and type validation for each task
Generated action sequences	
Generated tasks	

Supplementary Figure S31. Example of action generation for each device in module

Supplementary Figure S31a. GPT prompt example of action generation for each device in module

```
messages=[
    {
        "role": "system",
        "content": """You are administrator of autonomous laboratory, which control AI and robotics for chemical experiments,
        and your role is the module generation for chemical experiments execution with chemical devices.""",
    },
    {
        "role": "user",
        "content":f"""
        chemical actions define as "simple word for devices action, not over 2 words", such as stirrer has "stir", "heat", "stop" actions.
        "pipette draw" or "move to position" is wrong action name.

        please generate actions for controlling chemical devices in {name}, including only {device_str}.
        The first letter of the device name must always be capitalized, and for device names containing spaces,
        express the delimiter with a capital letter instead of a space..
        Don't combine between devices or generate new device name.
        For example, when you get "Robot arm" and "Pipette", don't generate "RobotArmPipette".
        Please save as python json, and key of json is device name, value of json is action list.
        """,
    }
]
```

- ✓ name: module name
- ✓ device_str: all devices in module

Supplementary Figure S31b. Example of generated actions for each device via GPT recommendation and client feedback

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

+-----+---------------------+------------------------------------------------------------------------+
| idx | RobotArmDeviceServer | action types recommended by GPT                                        |
+-----+---------------------+------------------------------------------------------------------------+
| 1   | RobotArm            | ['Move', 'Rotate', 'Lift', 'Lower', 'Release', 'Grip']                 |
| 2   | Pipette             | ['Aspirate', 'Dispense', 'Wash', 'Calibrate', 'EjectTip', 'AttachTip'] |
+-----+---------------------+------------------------------------------------------------------------+
Select the device index you want to modify (enter 'done' to finish, 'back' to return): 1
Do you want to add, modify or delete a task? ('a'/'m'/'d' or 'back' to return): d
Enter the task to delete in RobotArm (or 'back' to return): rotate
Are you sure you want to delete action 'Rotate'? (y/n, 'back' to return): y
Action 'Rotate' deleted from RobotArm.

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

+-----+---------------------+------------------------------------------------------------------------+
| idx | RobotArmDeviceServer | action types recommended by GPT                                        |
+-----+---------------------+------------------------------------------------------------------------+
| 1   | RobotArm            | ['Move', 'Lift', 'Lower', 'Release', 'Grip']                           |
| 2   | Pipette             | ['Aspirate', 'Dispense', 'Wash', 'Calibrate', 'EjectTip', 'AttachTip'] |
+-----+---------------------+------------------------------------------------------------------------+
Select the device index you want to modify (enter 'done' to finish, 'back' to return):
```

Supplementary Figure S32. Example of task generation for module

Supplementary Figure S32a. GPT prompt example of task generation for module

```
messages=[
    {
        "role": "system",
        "content": """You are administrator of autonomous laboratory, which control AI and robotics for chemical experiments,
        and your role is the task generation for chemical experiments execution with chemical devices.""",
    },
    {
        "role": "user",
        "content": f"""I am going to define tasks for a module, considering tasks as verbs representing experimental processes.
        For example, the tasks for the BatchSynthesis module include "MoveContainer", "AddSolution", "Stir", "Mix", "React", and "Pipette".
        All task name use verb with module name, such as 'BatchSynthesisModule_Stir', 'RDEEvaluationModule_Dispense
        Now, please suggest tasks for the {input_module_name} as python list, including {task_str}.
        If a specific task involves loading measurement or analysis or performance data, ensure the task name ends with "test".
        {description}.
        """
    },
```

- ✓ input_module_name: module name
- ✓ task_str: need to include task name
- ✓ description: module description

Supplementary Figure S32b. Example for generated tasks via GPT recommendation and client feedback

```
###################
[Feedback system (task generation)]: current task List
###################
+-----+------------------------------------+
| idx | current task name                  |
+-----+------------------------------------+
| 1   | SolidStateModule_LoadPowder        |
| 2   | SolidStateModule_AddPowder         |
| 3   | SolidStateModule_WeighPowder       |
| 4   | SolidStateModule_MixPowders        |
| 5   | SolidStateModule_GrindPowder       |
| 6   | SolidStateModule_PressPowder       |
| 7   | SolidStateModule_Calcine           |
| 8   | SolidStateModule_Sinter            |
| 9   | SolidStateModule_Cool              |
| 10  | SolidStateModule_Characterize      |
| 11  | SolidStateModule_XRDtest           |
| 12  | SolidStateModule_TGAtest           |
| 13  | SolidStateModule_DSCtest           |
| 14  | SolidStateModule_BETtest           |
| 15  | SolidStateModule_ParticleSizeTest  |
| 16  | SolidStateModule_DensityTest       |
| 17  | SolidStateModule_PorosityTest      |
+-----+------------------------------------+
Do you want to add, modify or delete this task? ('a'/'m'/'d') (or 'done' to finish):
```

Supplementary Figure S33. Example of the action sequence generation for task execution

Supplementary Figure S33a. GPT prompt example of action sequence generation for task execution

```
messages=[
    {
        "role": "system",
        "content": """You are administrator of autonomous laboratory, which control AI and robotics for chemical experiments,
        and your role is to construct action sequence of chemical task, followed by chemical devices and action type of chemical devices.""",
    },
    {
        "role": "user",
        "content": "I have a task list of {}, {}.\n".format(input_module_name, task_list)+
        """
        The types of chemical tasks we need to perform are listed below:{input_task_list}
        The devices we have and the actions each device can perform are listed below in the form of a dictionary:{input_device_action_dict}
        We called this dictionary, device_action_dictionary.
        The types of devices we have are the keys of the device_action_dictionary.

        Please identify the types of devices needed for each task to perform the chemical tasks.
        Then, create a device_action_dictionary with the task name as the key and a list of strings as the value,
        where each string represents a device and its action concatenated with an underscore "_".
        If there are no devices required for the task, an empty list, [] is used.
        The more detailed the device actions for each task, the higher the accuracy of the task.
        But don't generate new action, just use prior action in {input_device_action_dict}.
        Please save device_action_dictionary as JSON format.
        And, you must remove all annotations in json file.
        """.format(input_task_list=task_list, input_device_action_dict=device_action_dict)
    }
]
```

✓ input_task_list = generated tasks of module

idx	current task name
1	SolidStateModule_LoadPowder
2	SolidStateModule_AddPowder
3	SolidStateModule_WeighPowder
4	SolidStateModule_MixPowders
5	SolidStateModule_GrindPowder
6	SolidStateModule_PressPowder
7	SolidStateModule_Calcine
8	SolidStateModule_Sinter
9	SolidStateModule_Cool

✓ input_device_action_dict = generated actions of devices

device	action list
STIRRER	['Heat', 'Stir', 'Stop', 'heartbeat']
POWDERDISPENSER	['Dispense', 'Stop', 'heartbeat']
WEIGHINGMACHINE	['Stop', 'Tare', 'Weigh', 'heartbeat']
HEATER	['Cool', 'Heat', 'Stop', 'heartbeat']
XRD	['Align', 'Scan', 'Stop', 'heartbeat']
ROBOTARM	['Grip', 'Move', 'Position', 'Release', 'Rotate', 'heartbeat']
PIPETTE	['Aspirate', 'Calibrate', 'Dispense', 'Mix', 'Wash', 'heartbeat']
PUMP	['Decrease', 'Increase', 'Reverse', 'Start', 'Stop', 'heartbeat']

Supplementary Figure S33b. Example of generated action sequences for task execution via GPT recommendation and client feedback

```
###################
[Feedback system (match task-->device-action)]: Final Device:Action List
###################
```

task	device:action list
SolidStateModule_LoadPowder	['ROBOTARM_Grip', 'POWDERDISPENSER_Dispense']
SolidStateModule_AddPowder	['ROBOTARM_Move', 'ROBOTARM_Position', 'ROBOTARM_Grip', 'ROBOTARM_Release']
SolidStateModule_WeighPowder	['WEIGHINGMACHINE_Tare', 'WEIGHINGMACHINE_Weigh']
SolidStateModule_MixPowders	['STIRRER_Stir']
SolidStateModule_GrindPowder	['ROBOTARM_Move', 'ROBOTARM_Position', 'ROBOTARM_Grip', 'ROBOTARM_Release']
SolidStateModule_PressPowder	['ROBOTARM_Move', 'ROBOTARM_Position', 'ROBOTARM_Grip', 'ROBOTARM_Release']
SolidStateModule_Calcine	['HEATER_Heat']
SolidStateModule_Sinter	['HEATER_Heat']
SolidStateModule_Cool	['HEATER_Cool']

Supplementary Figure S34. Example of task template and type validation generation

Supplementary Figure S34a. Prompt engineering of task template generation for type validation using the OpenAI API

Role assignment

```
"role": "system",
"content": """You are administrator of autonomous laboratory, which control AI and robotics for chemical experiments,
and your role is the task template generation for chemical task.""",
```

Prompt

```
task_template_prompt="I have a task list of {}, {}.\n".format(input_module_name, task_list)+"""
And I need to specify the parameters for executing each task in the form of a dictionary template.
The number of hierarchy levels in template is limited to 3, and I won't create any more levels.
The first level in template will have keys "Task" and "Data", such as {"Task":"", "Data":{}}.
The key of second level in template will contain the necessary information for the task.
If some key of second level in template must include a quantitative value, the third level in template must use {"Value":0, "Dimension":""} this format
Dimension must match with the chemical task. For example, AddSolution task should match "μL" or "mL", Press task should match "mPa", "Pa" or "atm".
If some key of second level in template use material, powder, solution or gas, must add "Material" as key and {"Type":""} as value in the second level.
Otherwise, other value of second level in template must include "Value" as key in template, such as {"Type":""}.
If a specific task involves loading measurement, analysis or performance data, must use "Method" as key and {"Type":""} as value in the second level.

More detailed information could increase the reliability of chemical task.
Assign the task template names as variables for each task.
Also, please define all task template without exception in task list.

Define a pydantic class for each task, and don't skip some task, just define all of it.
Finally, you must save task template and pydantic class separately.

if we set task list as BatchSynthesisModule_MoveContainer, BatchSynthesisModule_AddSolution,
BatchSynthesisModule_Stir, BatchSynthesisModule_Heat, BatchSynthesisModule_Mix,BatchSynthesisModule_React,
answer may follow as below example script. Please refer below example code.
```

Example implements for few-shot learining

```
**BatchSynthesisModule.json**
```python
{
    "BatchSynthesisModule_MoveContainer":{
        "Task":"BatchSynthesisModule_MoveContainer",
        "Data":{
                "FromTo":{"Type":""},
                "Container":{"Type":""},
                "Device":{}
            }
    },
```

```
"BatchSynthesisModule_AddSolution":{
    "Task":"BatchSynthesisModule_AddSolution",
    "Data":{
        "Material":{"Type":""},
        "Volume":{
            "Value":0,"Dimension":"μL"
            },
        "Concentration":{
            "Value":0,"Dimension":"mM"
            },
        "Injectionrate":{
            "Value":0,"Dimension":"μL/s"
            },
        "Device":{}
    }
},
```

```
"BatchSynthesisModule_Heat":{
    "Task": "BatchSynthesisModule_Heat",
    "Data": {
        "Temperature": {
            "Value": 0,
            "Dimension": "ºC"
        },
        "Device":{}
    }
},
```
```

Supplementary Figure S34b. Prompt engineering of Pydantic class generation for type validation using the OpenAI API

Role assignment

```
"role": "system",
"content": """You are administrator of autonomous laboratory, which control AI and robotics for chemical experiments,
and your role is the Pydantic generation for chemical task.""",
```

Prompt

```
task_pydantic_prompt="""Also, define a pydantic class followed by json file, and don't skip some task, just define all of it.

if we set task list as BatchSynthesisModule_MoveContainer, BatchSynthesisModule_AddSolution,
BatchSynthesisModule_Stir, BatchSynthesisModule_Heat, BatchSynthesisModule_Mix,BatchSynthesisModule_React,
answer may follow as below example script. Please refer below example code.
```

Examples for few-shot learning

```
**BatchSynthesisModule.py**
```python
from pydantic import BaseModel, Field
from typing import Dict, Any, Optional, Union

class BatchSynthesisModule_MoveContainer_FromTo(BaseModel):
    Type: str = ""

class BatchSynthesisModule_MoveContainer_Container(BaseModel):
    Type: str = ""

class BatchSynthesisModule_AddSolution_Material(BaseModel):
    Type: str = ""

class BatchSynthesisModule_AddSolution_Volume(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "μL"

class BatchSynthesisModule_AddSolution_Concentration(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "mM"

class BatchSynthesisModule_AddSolution_Injectionrate(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "μL/s"

class BatchSynthesisModule_Heat_Temperature(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "ºC"

class BatchSynthesisModule_Mix_Time(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "sec"
class BatchSynthesisModule_Centrifugation_Power(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "rpm"

class BatchSynthesisModule_Centrifugation_Time(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "sec"
```

```
class BatchSynthesisModule_Centrifugation_Power(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "rpm"

class BatchSynthesisModule_Centrifugation_Time(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "sec"

class BatchSynthesisModule_Sonication_Power(BaseModel):
    PowValue: Union[int, float] = 0
    Dimension: str = "kHz"

class BatchSynthesisModule_Sonication_Time(BaseModel):
    Value: Union[int, float] = 0
    Dimension: str = "sec"

class BatchSynthesisModule_MoveContainer_Data(BaseModel):
    FromTo: BatchSynthesisModule_MoveContainer_FromTo
    Container: BatchSynthesisModule_MoveContainer_Container
    Device: Dict[str, Any]

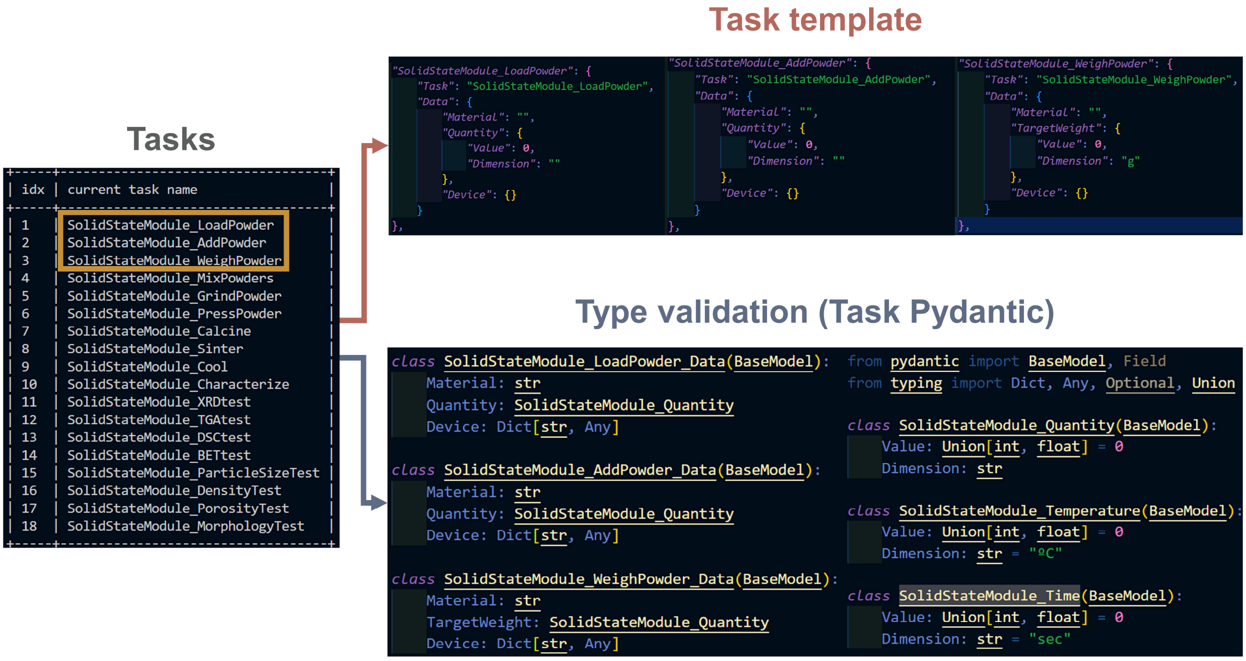
class BatchSynthesisModule_AddSolution_Data(BaseModel):
    Material: BatchSynthesisModule_AddSolution_Material
    Volume: BatchSynthesisModule_AddSolution_Volume
    Concentration: BatchSynthesisModule_AddSolution_Concentration
    Injectionrate: BatchSynthesisModule_AddSolution_Injectionrate
    Device: Dict[str, Any]

class BatchSynthesisModule_Heat_Data(BaseModel):
    Temperature: BatchSynthesisModule_Heat_Temperature
    Device: Dict[str, Any]

class BatchSynthesisModule_Mix_Data(BaseModel):
    Time: BatchSynthesisModule_Mix_Time
    Device: Dict[str, Any]

class BatchSynthesisModule_Centrifugation_Data(BaseModel):
    Power: BatchSynthesisModule_Centrifugation_Power
    Time: BatchSynthesisModule_Centrifugation_Time
    Device: Dict[str, Any]
```
```

Supplementary Figure S34c. Example of generated task template and type validation for each task



Supplementary Table S4. The result of automated code generation/customization via Copilot of OCTOPUS

1st hierarchy	2nd hierarchy	3rd hierarchy		Description
Action	Module	{module name}.py		Action translator script of module
	routing_table.json			JSON file included IP addresses and port number of each module
	ActionExecutor_Class.py			Script of action executor
	ActionTranslator_Class.py			Script of action translator
Analysis	Module	{module name}.py		Preprocess script for raw spectrum or performance data of each module
	Analysis.py			Script of analysis method inherited by Module/{module name}.py
Algorithm	Manual	Manual.py		Script of manual experimentation method
	{algorithm name}	{algorithm name}.py		Script of AI for experiment planning
AutoModuleGeneration	GPT_answer_ generation	{module name} _actions.txt		GPT recommendations for actions and tasks generation saved as text files
	GPT_answer_registration	{module name} _actionsequence.txt		GPT recommendations for action sequences saved as text files
		{module name} _task_pydantic.txt		GPT recommendations for task template generation saved as text files
		{module name} _task_template.txt		GPT recommendations for type validation of task (=Pydantic) saved as text files
	{module name or device server name}	BaseUtils	json_func.py	Functions for socket communications
			TCP_Node.py	Functions for JSON file read/write
		Log	Logging_ Class.py	Logger for recording all actions
		{device_name}		Functions for device controller based on manufacturer's API (Application Programming Interface)
		module_node.py or device_server.py		Interface of module node or device server
DB	DB_Class.py			Script of MongoDB manager
Job	device_standby_time.json			JSON file included tasks with long device standby time for each module
	JobTrigger.py			Script of job trigger
	JobScheduler.py			Script of job scheduler
JobScriptTemplate	{module name}.json			Template of job script for each module
Log	Logging_Class.py			Script of logger
Resource	Module	{module name}.py		Resource allocator for each module
	device_location.json			Device resources (location information) for each module
	device_status.json			Device status table included all devices for each module
	device_masking_table.json			Device masking table for all task of each module
	ResourcAllocator_Class.py			Script of resource allocator inherited by Module/{module name}.py
	ResourcManager_Class.py			Script of resource manager
Task	ActionSequence	{module name}.json		JSON file included action sequence of tasks for each module
	Pydantic	{module name}.py		Script of type validation for each module via Pydantic library

	Template	{module name}.json	JSON file included task templates for each module
	Template_module.json		JSON file included all module templates
	TaskGenerator_Class.py		Script of task generator
	TaskScheduler_Class.py		Script of task scheduler
USER	{client ID}		
UserManager	auth0_config.py		Script of Auth0 configuration for login process
	UserManager_Class.py		Script of Auth0 functions for login process
client.py			Script of client
copilot.py			Script of Copilot of OCTOPUS
master_node.py			Script of master node

* {}: the real name of module, ex) {module name}="SolidStateModule"

Purple boxes represent the core software of OCTOPUS. Green boxes indicate the automated code generation for module operation via Copilot of OCTOPUS, which includes functions for module operation. Red boxes represent the JSON file addressing new module information via Copilot of OCTOPUS.

Supplementary References

[1] L. Parziale, W. Liu, C. Matthews, N. Rosselot, C. Davis, J. Forrester, D. T. Britt, others, *TCP/IP tutorial and technical overview*, IBM Redbooks, **2006**.

[2] A. C. Vaucher, F. Zipoli, J. Geluykens, V. H. Nair, P. Schwaller, T. Laino, *Nat. Commun.* **2020**, *11*.

[3] H. J. Yoo, N. Kim, H. Lee, D. Kim, L. T. C. Ow, H. Nam, C. Kim, S. Y. Lee, K.-Y. Lee, D. Kim, S. S. Han, Bespoke Metal Nanoparticle Synthesis at Room Temperature and Discovery of Chemical Knowledge on Nanoparticle Growth via Autonomous Experimentations. *Adv. Funct. Mater.* **2024**, 2312561. https://doi.org/10.1002/adfm.202312561

[4] NHN Dooray! Corporation, *https://dooray.com/main/service/messenger*.