Description of the Experimental Setup

The emulated process represents the real experimental facilities of the Kiwi-biolab and the way they are operated during cultivation experiments performed there. The experimental activities include a combination of manual labour (such as pre-cultivation of the inoculum, preparation of the solutions and calibration of the robots) and automated tasks (liquid handling, sample preparation and analysis) during the actual execution of the cultivations. The design of the experiment includes a list of activities and conditions to be prepared by the laboratory technicians and an initial experimental plan to be followed by the robots in the form of setpoints (that could be static values or dynamic profiles). The setpoints can be changed during the experiments either by the technician (open-looped operation) or by the optimization and control algorithms (automated operation).

The setup consists of two Liquid Handling Stations (LHS) connected by a custom robot arm. The first LHS handles a system of MBR containing a mutant E. Coli strain that expresses a heterologous protein, performing feeding, induction, sampling and pH control. The samples taken by the first LHS are placed on a tray and transferred by the robot arm to the second LHS, where they are prepared and analysed. Some samples are sent to a high-throughput analyser for additional analytical measurements.

The bioreactor system has 24 MBR arranged in an 8 rows by 3 column lay out. Each MBR has a working volume of 10 ml and starts with cultivation medium, a biomass inoculum and an initial load of glucose. The robot arm of the first LHS manage the cultivation workflow with its 8 pipettes, adding or taking liquid from the MBRs. The cultivation is structured in 3 phases. First, the system operates in the batch phase with no feeding, consuming the initial load of glucose. After that, glucose solution is fed to the MBRs in pulses during the fed-batch phase. In the last part of the experiment, an inductor is added to the MBRs in the induction phase, increasing the production of the heterologous protein while still feeding glucose pulses. During all the phases, samples are taken from the MBR and located in a well plate tray. Every hour, the tray is transferred from the first LHS to the second one where samples are prepared and analytics are performed. In addition to these analytical measurements, each bioreactor has pH and dissolved oxygen tension (DOT) sensors that measure these quantities with high frequency (approximately one sample every 2 minutes).

Samples are taken from all the MBRs of a single column at the same time, and each column is sampled every 20 minutes. The samples are placed in well-plates containing NaOH where the biological activity is stopped. The plates wait in a tray until they are transferred to the second LHS every one hour. Thus, samples from the first column wait approximately 40 minutes before the transfer. In the second LHS, the robotic arm dilutes the samples and perform measurements of optical density (OD), which correlates with biomass concentration, and fluorescence, that indicates the heterologous protein concentration (which is a coloured fluorescent protein like GFP or RFP). Other samples are analysed in the HT analyser after preparation, where they are placed either manually or using a co-bot. In

the HT analyser, glucose and acetate are measured using spectroscopic methods. The analytics take an hour from the moment where the well plate tray is transferred from the first LHS to the second one.

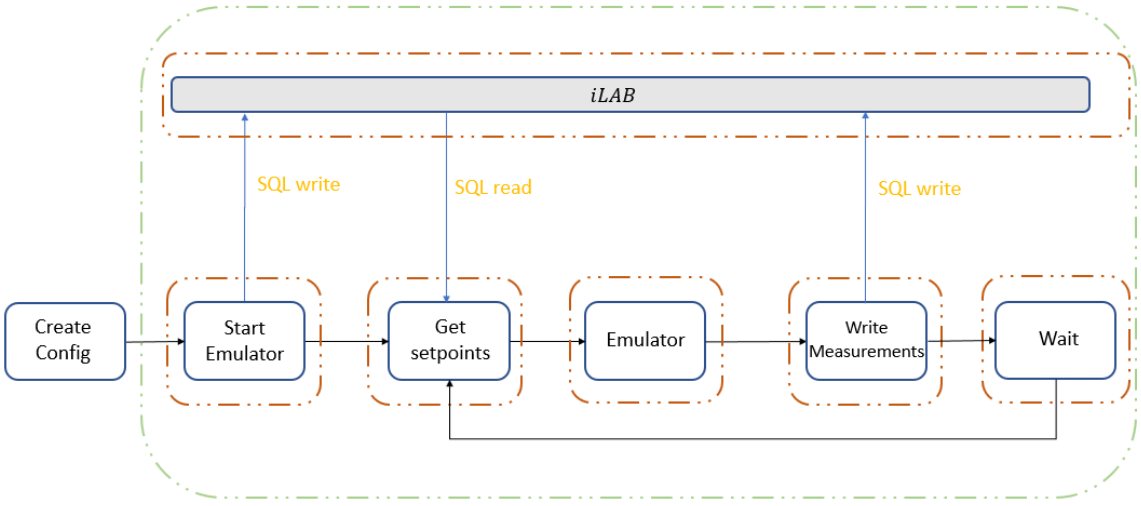
The data from the analytical measurements and from the DOT and pH sensors are written to a SQL database. The SQL database also contains the setpoints for the operation experiment, particularly the feeding profile for each MBR. The first LHS reads the setpoint from the database and internally manages the execution of all its tasks. Communication between the experimental setup and external control and optimization algorithm is done through the SQL database. The methods may query the SQL server to get the data from the measurements (analytical and sensors) and can write the new setpoints directly to it, so they are read by the first LHS. Thus, the internal operation of the robots and the analyser are isolated from the external algorithms and communication is done exclusively via the SQL database.

The feeding profiles setpoints are written as the cumulated volume that should be fed at any given time. The LHS has an internal control routine that compares how much volume has been added so far and calculates how much it should feed. Due to the delay in the additions caused by the multiple tasks the arm must perform, and the errors in the amounts due to the precision of the pipettes, using the cumulative volume instead of the volume of the pulses was chosen as the operating variable.

The induction of the protein expression is done by the addition of a chemical reagent (Isopropyl-beta-D-thiogalactoside or IPTG). The microorganism contains a plasmid that can produced a fluorescent protein. While there is some protein production during the normal growth phases (batch and fed-batch), when exposed to the inductor the yield of that protein increases many folds. The addition of the reagent is done manually by an operator.

Description of the emulator/implementation

The emulator consists of several scripts that can be grouped in two categories. The cultivation system is described by a mathematical model written and executed in Python, and the SQL database (which is a copy of the real database), is implemented in MySQL. All programs are run inside Docker containers and managed by Apache Airflow.



The emulator nodes are managed by Airflow and implemented in Docker

The emulator follows the workflow presented in the figure. After initialization using a configuration file that contain variables related to the experimental design and the emulator operation, Node "Start Emulator" creates two other JSON files, the "Emulator_design.json" and "Emulator_state.json" files. From that point on, the emulator executes three steps until the end of the experiment: Node "Get Setpoints", Node "Emulator" and Node "Write Measurements". The "Write" and "Get" scripts communicate with the SQL database; the first one updates the "Emulator_design.json" file while the "Write" script uses the "Emulator_state.json" file to write to the database. The simulation of the system is done with Node "Emulator", which integrates the mathematical model between the last execution and the current time. The script reads the "Emulator_design.json" files and "Emulator_state.json" updating the later with the new values of the system states and with data coming from the second LHS, including measurement errors and time delay in the availability of the samples.

Results can be accessed in the SQL server in port 3306. They are also available as a JSON file in "db_emulator.json", although this is an emulator file and it wouldn't be available in a real experiment.

A monitoring tool can be accessed through a browser at http://localhost:8501/

Running the Emulator

The emulator requires the "EMULATOR_config.json" file. This can be created in different ways, in the repo this is done using the "method_createDesign_Script.py" script. With this file, the emulator can be used from the Docker implementation of Apache Airflow.

1. Run "...\dags\scripts\emulator_dag\emulator2\method_createDesign_Script.py". This will create the "EMULATOR_config.json".
2. Start Docker.
3. In a browser, go to "http://localhost:8080/" to access Airflow
4. Activate the toggle for the Emulator_2.0_DAG and press the play button

Create Design file

The config file of the Emulator can be created with the “method_createDesign_Script.py” script. This can be run with Python without the necessity of using Docker. The design can be modified in the *Config Panel* section o the script.

```
# %% Config Panel
t_duration=16.1 # duration in [h]
```
Duration of the experiment, measured in hours

```
species_list=['Xv','Glucose','Acetate','DOT','Fluo_RFP','Volume']
```
List with the name of the Species used by the model

```
species_IC=[0.18,3,0,100,150,.01] #Initial states for the species listed above
```
List with initial conditions used in the model. For the species list used in the example, the units are [g/l,g/l,g/l,%, UFC/ml, ml]. This values will be applied to all MBR

```
glucose_IC=[5,5,4,4,3,3,3,3]*3 #Initial glucose concentration
```
This overwrites the initial glucose concentration for each MBR. It’s a list of length 24

```
time_pulses=np.arange(4+5/60,t_duration,10/60) #Time in hours
```
Numpy array with the reference feeding pulses for all MBR. This is the initial value written to the database setpoints

```
time_samples_columns={'col1':np.arange(.33,t_duration,1).tolist()+[t_duration],'col2':np.arange(.66
,t_duration,1).tolist()+[t_duration],'col3':np.arange(.99,t_duration,1).tolist()+[t_duration]}
```
Python dict with the analytic sampling time of each column. Each entry of the dict is a numpy array. Sampling times are in hour

```
sampling_rate_DOT=2/60 #Time in hours
```
Sampling rate for the DOT sensor, in hours

```
time_samples_analysis=np.arange(1,t_duration,1).tolist()
```
Numpy array with the times when the samples are sent to the second robot

```
Noise_concentration=5 # in %
```
Noise added to the sampled values, in % of the value
```
Noise_time=1 # in %
```
Noise added to the sampling times (i.e. time_samples_columns), in % of the value

```
mbr_list=np.arange(19419,19443,1) #names of the bioreactors
```
Numpy array with the number of MBR for the SQL database

```
Glucose_feed=[200]*len(mbr_list) # in g/l
```
List with the glucose concentration of the feeding pulses for each MBR. Measured in g/l of glucose

```
Induction_time=[10]*len(mbr_list) #Time in hours
```
List with the induction time for each MBR. Measured in hours
```
Inductor_conc=[1]*len(mbr_list) # 0 to 1 for now
```
List with the inductor concentration. Not used in the current implementation

```
Params=[1.578, 0.43041, 0.6439, 2.2048, 0.1563, 0.1143, 0.1848, 287.74, 0.2586, 1.5874,
0.3322, 0.0371, 0.0818, 7.0767, 0.4242, 1.057]+[750]*len(mbr_list)+[90]*len(mbr_list)
```
List of parameters used by the model in “function_simulation.py”

```
acceleration=1# 1 for real time, otherwise it multiplies time by this factor
```
Acceleration of the Emulator. The time goes faster by this factor. Acceptable values are 1,2,4,8, 60. For higher values, it runs the experiment immediately from beginning to end