Electronic Supplementary Material (ESI) for Reaction Chemistry & Engineering.

Electronic Supplementary Information

OpenFlowChem - a platform for quick, robust and flexible automation and self-optimisation of flow chemistry

Nikolay Cherkasov[1,2], Yang Bai[2], Antonio Exposito[2], Evgeny V. Rebrov[1,2,3]

[1] School of Engineering, University of Warwick, Coventry CV4 7AL, UK;
[2] Stoli Catalysts Ltd., Coventry CV3 4DS, UK;
[3] Department of Biotechnology and Chemistry, Tver State Technical University, Nab. A. Nikitina 22, 170026 Tver, Russia

S1. Additional details on the OpenFlowChem architecture

We need to implement modularity and high reusability of the code and use as many features already available in the commercial software associated with the instruments to accelerate and simplify the creation of the automation system. Figure 1 shows the architecture of the OpenFlowChem platform. It contains three major layers: (i) device monitors that handle communications with individual units, (ii) system integration module to provide interaction between the individual instruments, (iii) optional external safety devices to ensure an additional system-independent safety margin. The systems may be externally controlled or interact horizontally for additional flexibility and modularity.

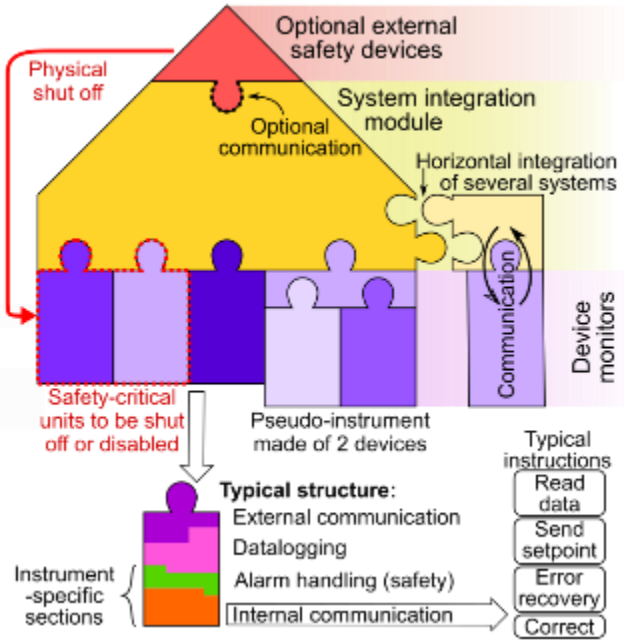

Figure 1. Overview of the automation platform solution which consists of optional safety devices, system integration module responsible for the system logic and oversight which is connecter to device monitors.

The device monitor modules communicate to the reactor components such as pumps, mass flow controllers, analytical equipment via RS232, Ethernet, etc. The analogue instruments are communicated via the digital-to-analogue converters (DAC) and the corresponding device monitors handle digital communications with converters. In this case, it is beneficial to use separate digital to analogue converters for various instruments to allow for rapid rearrangement of the system with no additional efforts. However, if several units are logically connected and should act in a synchronised fashion, they can be combined to a single DAC or even digitally-communicating units can be combined into a pseudo-device, the example is section 3.3 of the paper in details. The devices may be connected physically (wired or wirelessly) to the computer running the system module, or via the internet implementing the internet of things approach – the particular way should be selected based on the particular system requirements and the implementation efforts.

The device monitors are modular self-sufficient units that can be run as standalone applications to control the required operation of the instrument. The device monitors incorporate the standard components such as a datalogger, an external communication module, error and internal alarm handling modules (Figure 1). The alarm handler provides a two-level alarm about to the operation of the unit with the level 1 acts as a warning to the operator about abnormal (but not safety-critical) operation and level 2 leads to a safe shutdown of the instrument. The examples of the alarms include a significant deviation of the flow rate (level 1) or overpressure of the unit (level 2 – immediate shutdown). The internal communication part takes into account the possibility of communication errors, provides error recovery or returns the level 2 alarm in case of a critical communication error such as a lost connection.

The structure of the device monitors, although not conventional, allows for programming the device monitors quickly for a broad range of various instruments substituting only the instrument-related communication parts and the alarm handling. The approach ensures the robustness by the extensive utilisation of the checked and debugged sections of the code from other device monitors. Handling instrument-specific problems at the devi ce monitor level removes the need to manage possible problems at a higher system level where the complexity may be already high. The self-sufficient device monitors allow for fully manual operation of the system with the device monitors for each instrument simply running in parallel with no additional development time – the approach suitable for the quick proof of principle studies. The idea using separate programming parts for each instrument is established.[1–4] However, having fully self-sufficient device monitors, as presented in the work, provides a possibility not only to sufficiently simplify the system module making it more transparent and reliable but also provides an additional safety and troubleshooting possibility by ensuring that the device datalogging proceeds even if the system module malfunctions.

The next level in Figure 1 contains the system integration module that incorporates all the device monitors, sends the instructions to them and provides safe operation with the required interlocks. The device monitors do not normally communicate with each other, but only to the system integration module. Using this approach, every device can be “unplugged” from the larger system or a new instrument added with only modifications required only at the system level. The communication between the device monitors and the system may include turning into a safe-shutdown mode all the devices in case a level 2 alarm is received from any devices, getting to a certain setpoint or doing analysis or sampling. Considering that the most instrument-specific operations were already handled on the device monitor level, the system integration module is easy to program implementing a system logic block scheme. A part of the system module includes the optional graphical user interface loop that can

be either adjusted to create a maximally visual environment or quick programming – the balance to be determined by the application.

The next part of the system includes the optional external safety devices which may be connected as the device monitors. In case of hydrogenation, for example, the safety systems may include the hydrogen sensors to detect a leak with the ability to stop the hydrogen supply and effectively disable the most safety-critical component, the hydrogen flow controllers. In the case of a flow reactor operating with the hydrogen shutoff, the system will receive no information about hydrogen signal or the fact that the hydrogen line is closed. Therefore, level 2 alarm on the hydrogen mass flow controllers may be the absence of the gas flow for a minute which would provide a signal for the safe shutdown to the whole flow reactor without the digital connection with the hydrogen safety system. However, the possibility of adding such safety systems as device monitors and the ability to monitor the feedback (but not to stop safety interlocks) may provide an additional level of responsiveness. In the previous example of a hydrogen sensor noticing a leak from the reactor, the system can be shut down immediately when the leak is noticed without waiting for exhaustion of hydrogen from the feeding pipes. A relay with watchdog functionality is another example the external safety system – the unit which has the ability to disable safety-critical units (pumps, mass-flow controllers and heaters) in case the watchdog receives no digital communication from the control system. The watchdog provides the possibility to ensure safe operation even if the control system freezes or malfunctions and does not require any physical problem to develop before the safe shutdown. However, it is worth noting that hazard and operability study (HAZOP) analysis or similar has to be applied to the system to ensure safe operation even in the case of the control system or power failure.

Several systems may be connected horizontally to existing commercial systems. Example 3.2 from the paper includes using Matlab as an external computational unit to calculate the next experimental conditions to implement. The horizontal arrangement compared to the widely used incorporation of the Matlab code into the Labview systems provides the possibility for the easy modification or even substitution with other systems. Also, the possibility to split the overall system into semi-independent subsystems provides the possibility to use the internet of things for their interfacing. In contrast to the excellent solution presented by Fitzpatrick et al.[4], our architecture provides the possibility for the cloud-based chemistry but does not incorporate it by default. We believe that our approach increases security, simplicity and reduces the implementation costs for the systems that do not require cloud-based capabilities. When the cloud-based operation is carried out using our approach, a limited number (often one) of internet interaction channels need to be managed and secured compared to the plurality of units with their inevitable vulnerabilities connected to the internet.

S2. Structure of a device monitor module

We took an HPLC pump, Knauer P4.1S equipped with a pressure transducer to demonstrate an example of a device monitor module. The device monitor contains the components responsible for (i) external communication, (ii) internal communication, (iii) alarm handler, and (iv) a datalogger. The pump represents a typical flow chemistry unit which has the main function (provide a flow) and returns several parameters back (the setpoint flow and an additional output of the operating pressure). The same idea applies to many other units where there might be several or no additional output or the readout value might be the observed flow, not a setpoint value.

The actual device monitor files for several instruments used in this paper are provided online. However, we have used the devices only for the small-scale laboratory work and always used the external safety devices. Therefore, we cannot guarantee absolute robustness of the units and while using them all the safety risks (including the software malfunction) must be adequately assessed and addressed. The presented device monitors also serve for the illustration purposes mainly and may be modified according to the requirements of the task at hand.

Considering the shallow learning curve and speed and efficiently of programming, we disregarded some good Labview practices of keeping the VI block diagrams below 1024x768. Although in principle we agree that such a convention is useful, the utilisation of the larger diagram allows for faster and visual understanding of the main units involved. In the following discussion, we use some of the Labview common terminology and do recommend studying the Labview Core 1, Core 2 and Core 3 courses. Although these are aimed at rather different tasks and approaches, the background knowledge provided there covers all the requirements of the OpenFlowChem platform.

The structure of the block diagram for the device monitor example is shown in Figure 2. It contains the variable initialisation part – the section which zeros all the variables to ensure the proper start. Then the test if the external control variables are entered correctly. If correct, the corresponding flag is set to show that the external control works. Then 3 parallel loops work on datalogging at a set interval, main operation of the unit and handling the external communication requests. All the sections are described in details further.

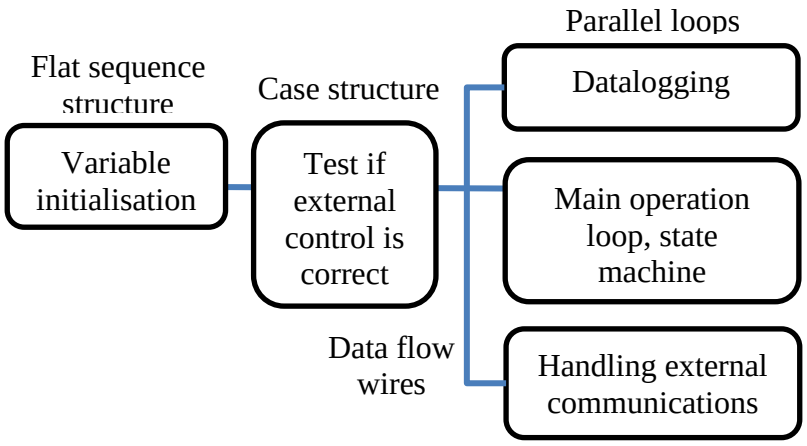

Figure 2. Scheme of the block diagram of a typical device monitor.

Figure 3 shows the variable initialisation and external control test sections of the block diagram. Here, the data flow forces the sequential execution of the code and the external control section checks if the variables provided agree with the expected format. Here we use cluster controls for IN and OUT controls separately. The reason for using 2 controls is that the OUT control is the most often needed to check the status of the instrument and its main operating parameters – in this case, the OUT control contains the Boolean indicators for the level 1 and level 2 alarms, the Ready flag that shows that the instrument can accept new commands, current operational parameters such as the flow rate setpoint readout and the current pressure as well as the string with the recent event. The IN control is needed

only either to set a new setpoint of stop the instrument completely and contains the setpoint value, the stop button and the button to enact the setpoint change.

The combination of these two controls into one is possible, however, it would have increased a chance of data writing collision – simultaneous writing of the data (for example entering the recent event string or stopping the system) into the same variable which results in unpredictable operation (prevent stopping the equipment, for example). When the IN and OUT variables are separated, this chance is eliminated provided the new IN data are entered only when the instrument is ready to accept (The Ready flag in the OUT variable).

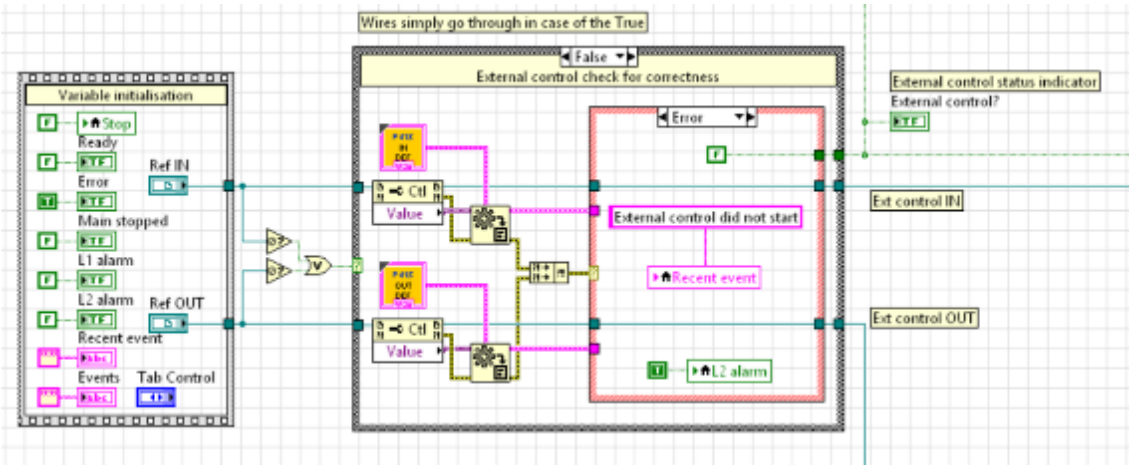

Figure 3. Variable initialisation and external control test sections of the device monitor block diagram.

After the external control test section, the data flows further into 3 parallel loops of the datalogging, main loop and external control handler (Figure 2).

Figure 4 shows the datalogging section of the device monitor block diagram. The overall function is rather obvious – the loop writes the file every given interval. The data (setpoint flow and the current pressure) are taken from the main loop, and any logging error is treated as a critical level 2 error and results in the stop of the whole device monitor. The leftmost part determines the logging file location and creates the file with the header in the subfolder if the file does not exist. Every day, a new datalog file is created, and the data can only be appended to the file avoiding the possibly of accidental log file damage.

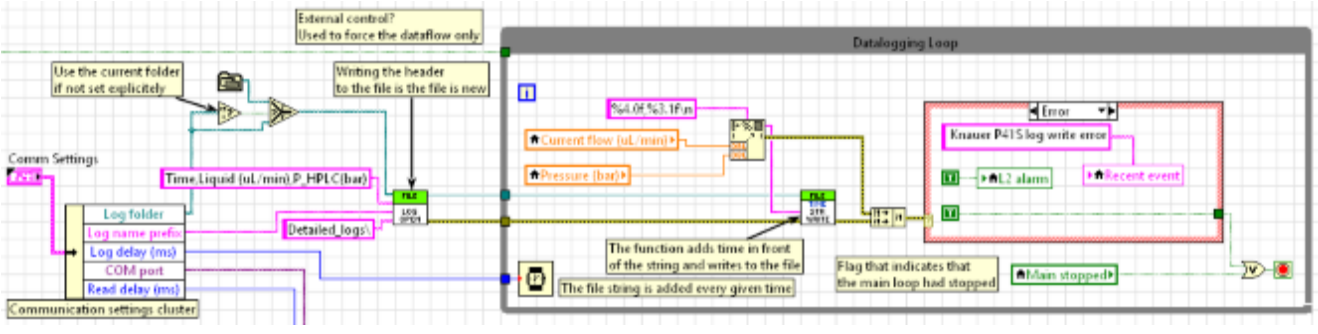

Figure 4. Datalogging section of the device monitor block diagram.

The main loop is built using the state machine structure explained in details by the National Instruments, the Labview creator, http://www.ni.com/tutorial/7595/en/. Figure 5 shows the block scheme of the main loop.

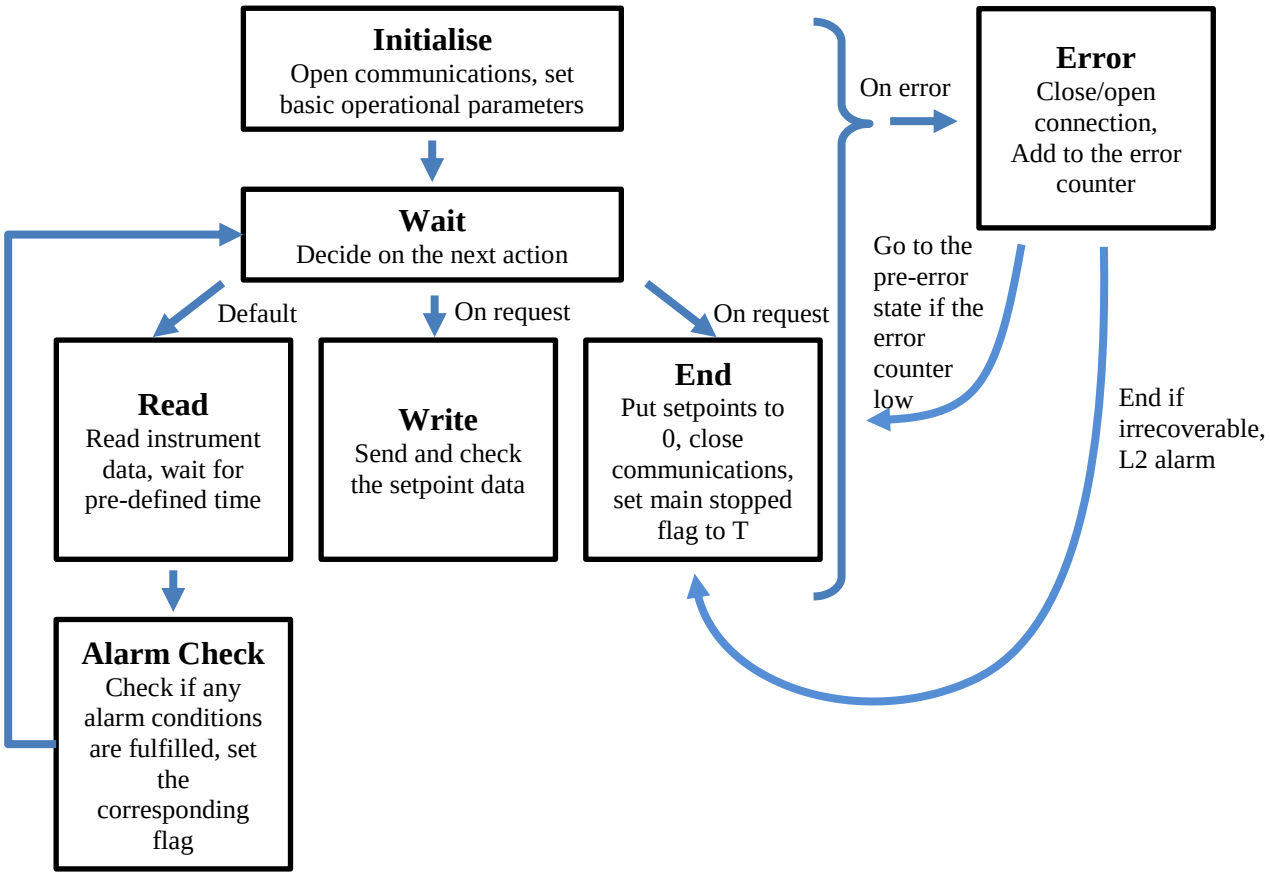

Figure 5. Block scheme of the main loop code of the device monitor block diagram.

Figures 6 - 13 show the screenshots of the main loop and the handling external communications loop.

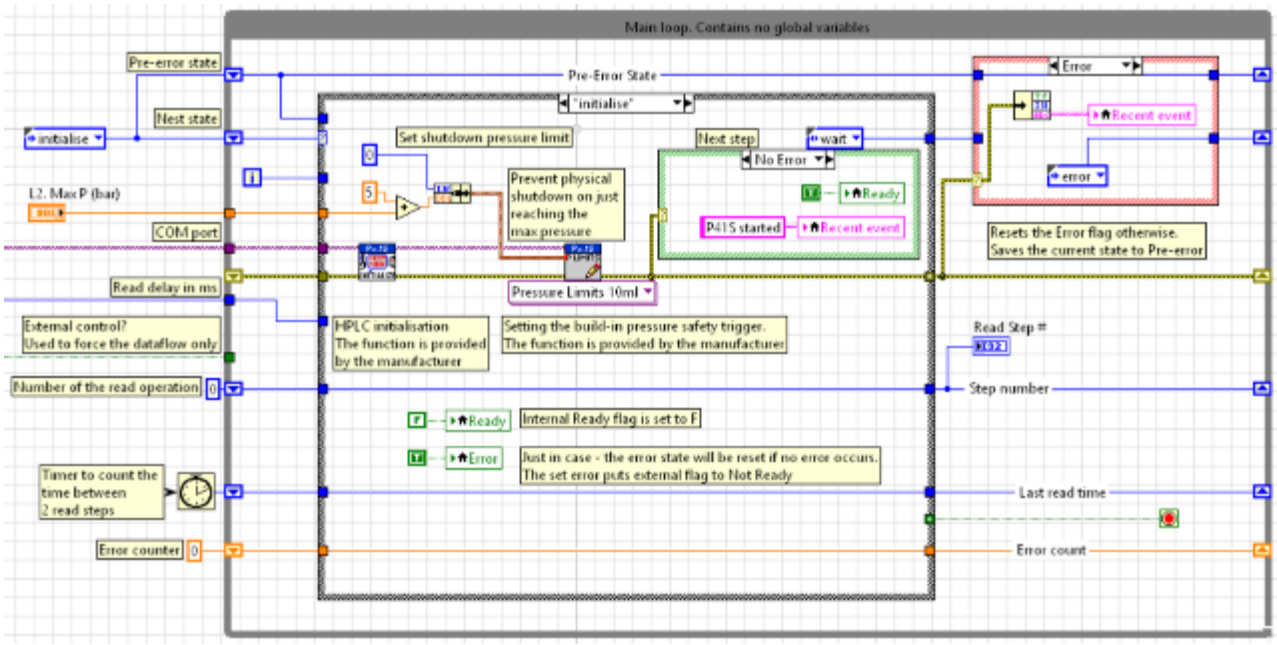

Figure 6. The Initialise section of the main loop of the device monitor block diagram

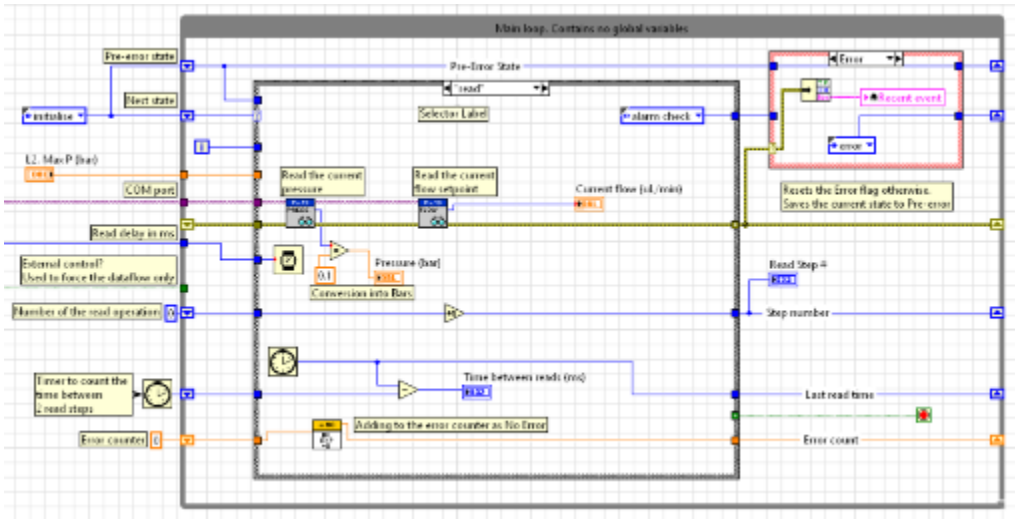

Figure 7. The Read section of the main loop of the device monitor block diagram

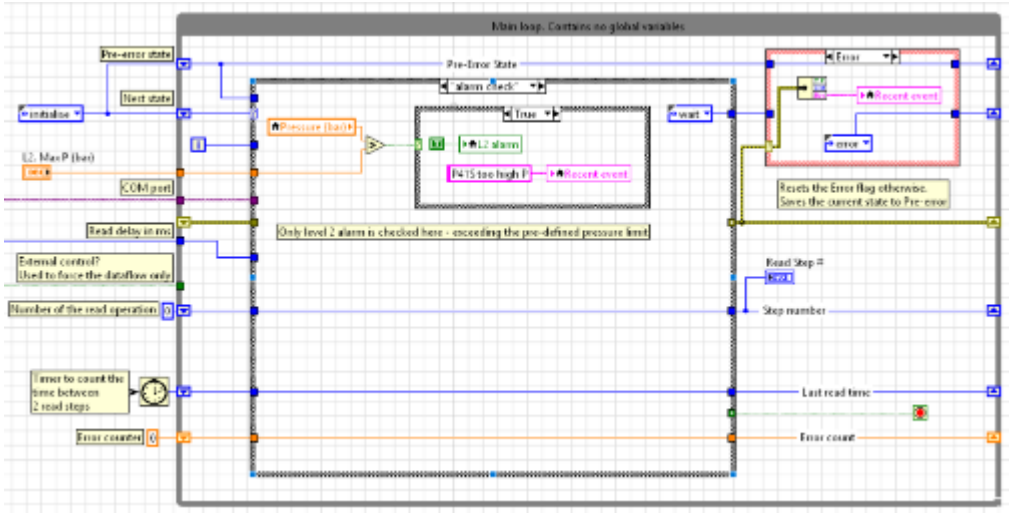

Figure 8. The Alarm Check section of the main loop of the device monitor block diagram

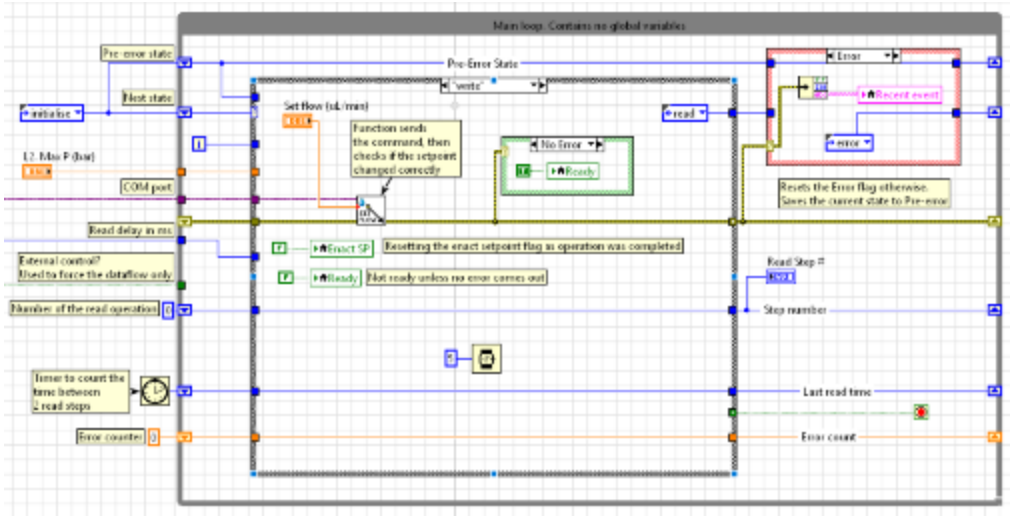

Figure 9. The Write section of the main loop of the device monitor block diagram

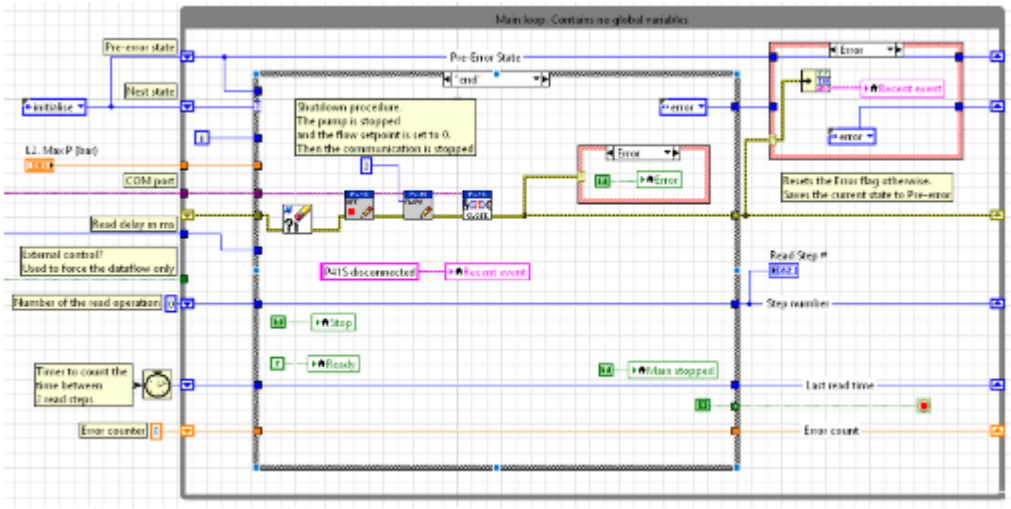

Figure 10. The End section of the main loop of the device monitor block diagram

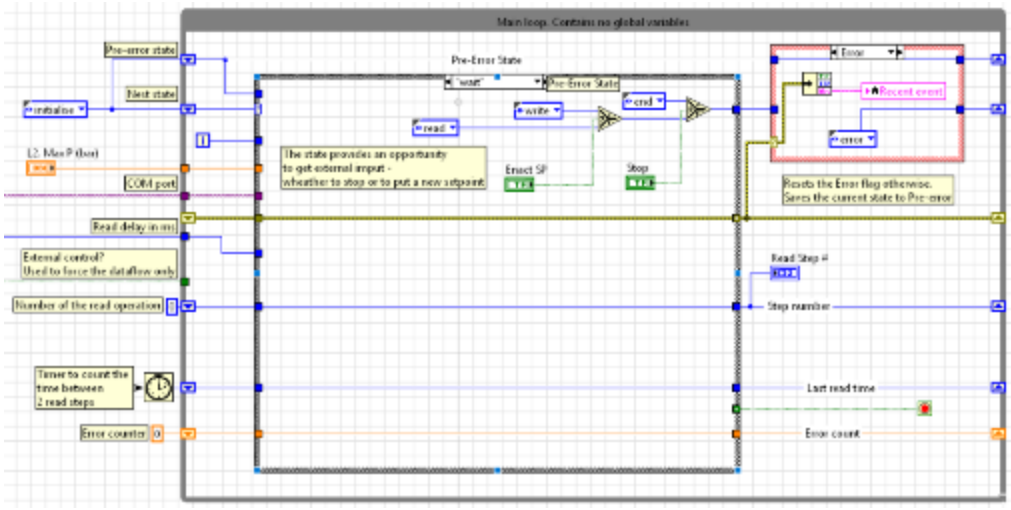

Figure 11. The Wait section of the main loop of the device monitor block diagram

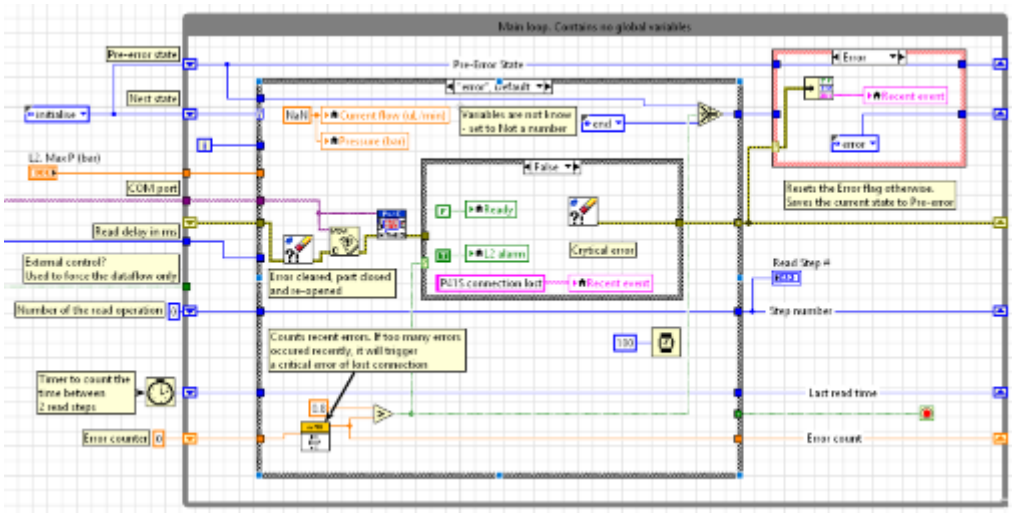

Figure 12. The Error section of the main loop of the device monitor block diagram

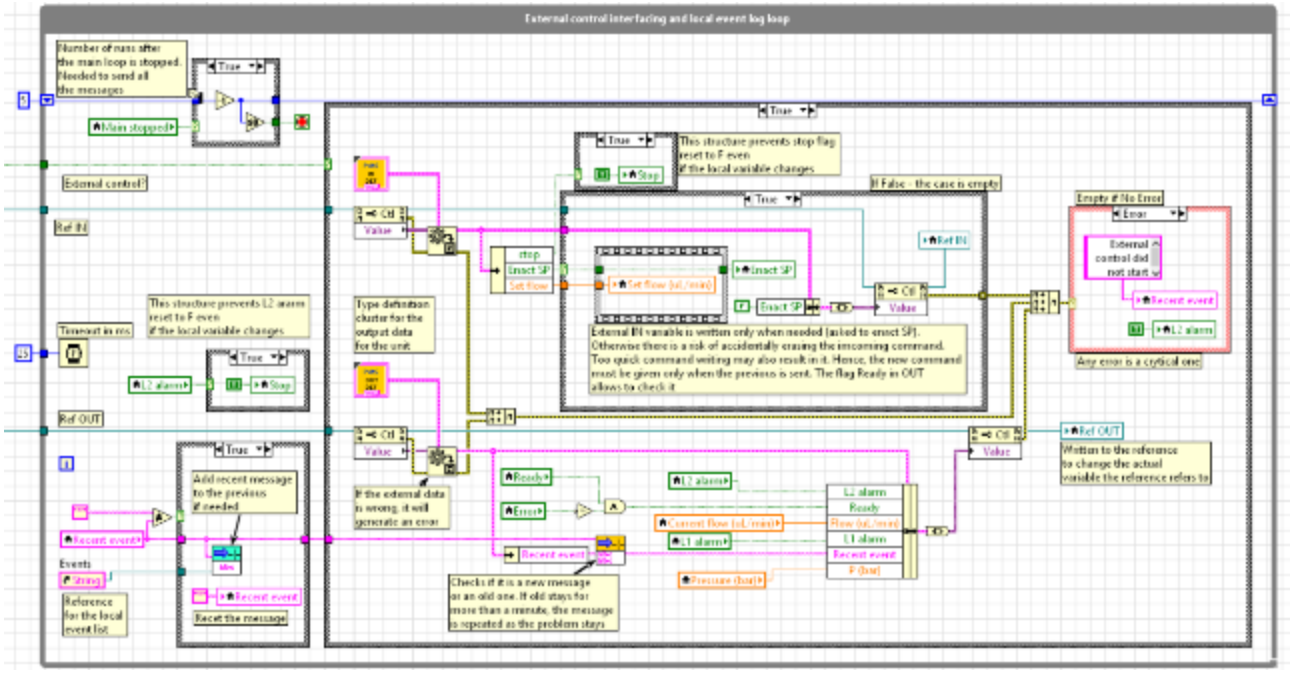

Figure 13. The Handling external communications loop of the device monitor

S3. Running several device monitor units in parallel

Once a device monitor VI (virtual instrument, the name of a program in Labview) is created, it can be cloned, and several instances of the same device can be run in parallel.

To do this, the required number of VIs should be placed in a main system integration unit VI and run keeping in mind that the physical addresses (such as COM ports) for various units must be unique. Figure 14 and 15 show the front panel and the block diagram of a simple example that includes 3 identical device monitors corresponding to a Knauer P120 HPLC pump.

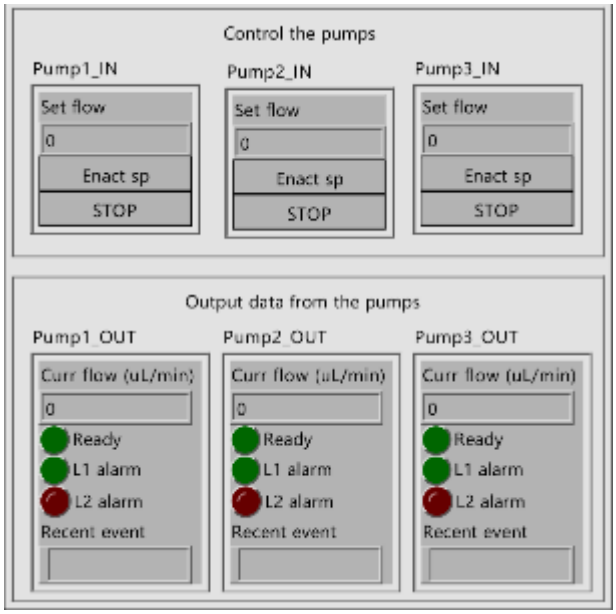

Figure 14. Front panel of the VI that shows how to add several identical units in parallel.

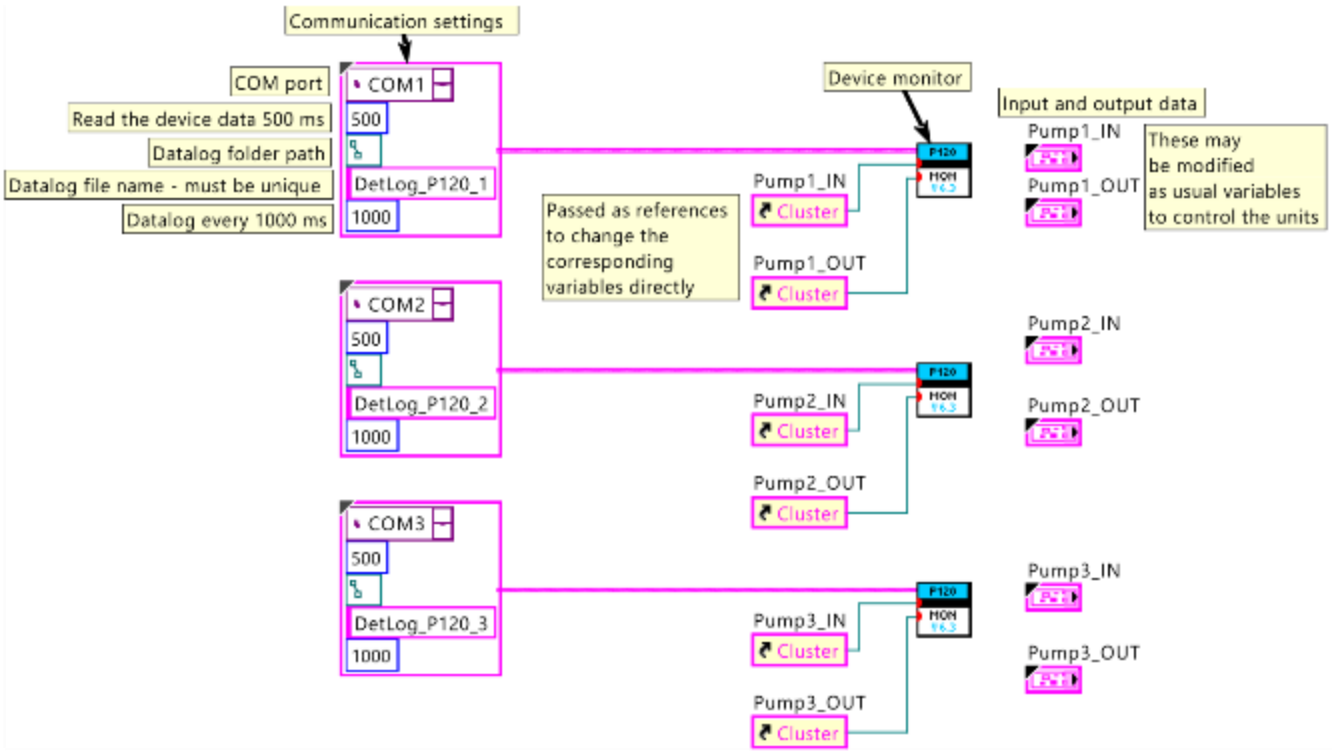

Figure 15. Block diagram of the VI that shows how to run several identical device monitors in parallel.

Additionally, the device monitor VIs must be put into the re-entrant (preallocate) mode in the VI properties settings to allow for fully independent operation of several instances of the device monitor.

More information on Reentrant VIs is provided by the National Instruments, the Labview developer, at the knowledge base http://digital.ni.com/public.nsf/allkb/98847B4E4C715E6D86256C59006B57CC

S4. Comparison of thermal mass-flow meter and optical liquid sensors

Firstly, we have started using a thermal mass-flow controller (Bronkhorst), which was connected to a 500 mL liquid trap prior to the meter to separate the liquid flow. We have introduced various flow rate of isopropanol, combined it with various flow rate of H_2 and measured the resulting H_2 flow using the flow meter. The data presented in Figure 16 show that the gas flow rate measured by the mass-flow meter stabilised only after 3 minutes on stream at the inlet gas flow of 10 mL/min. This time was obviously required to pressurise the liquid trap. The steady-state readings were not achieved in 3 min at a lower inlet gas flow, while the higher inlet gas flow of 20 mL/min resulted in saturation of the mass-flow meter. Moreover, the gas flow rate determined depended significantly on the liquid flow because, obviously, the introduction of liquid displaced gas from the liquid trap. Lastly, even at a low liquid flow rate, the determined gas flow rate did not agree with the inlet gas flow obviously because the solvent vapours affected thermal conductivity of the measured gas.

Therefore, the mass-flow meter was found unreliable and very slow-acting in measurements of the gas flow.

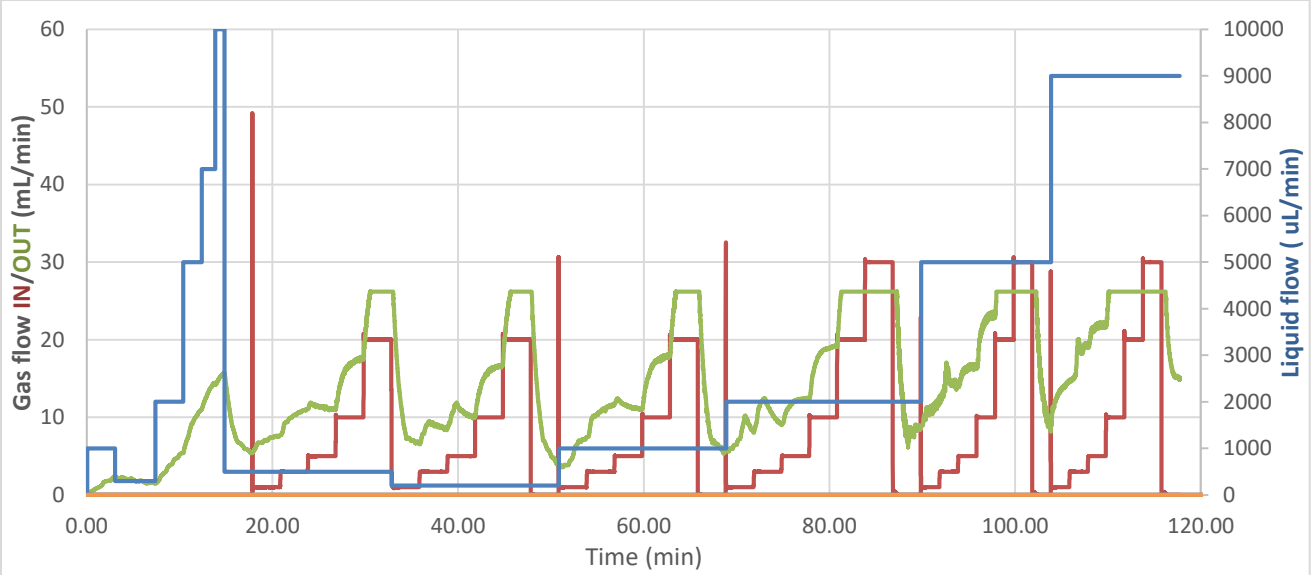

Figure 16. Reading of the mass-flow meter depending on the inlet gas and liquid flow rates.

We have compared three Optek optical liquid sensors connected to (L1) 1/16” OD, 0.5 mm ID FEP tube, (L2) 1/16” OD, 1.0 mm ID FEP tube, (L3) 6 mm OD, 4 mm ID glass tube. The liquid sensor was checking if gas or liquid is present every 100 μs and the liquid fraction was calculated as a fraction of the liquid time and the total observation time. The data in Figure 17 show that the resulting liquid flow rate determined by the optical sensors depends significantly on the tube diameter. The stability of the readings depends on the averaging time reaching acceptable results at the averaging time of 6-30s. Therefore, the optical sensors combine very low price with high stability of the readings and good response time.

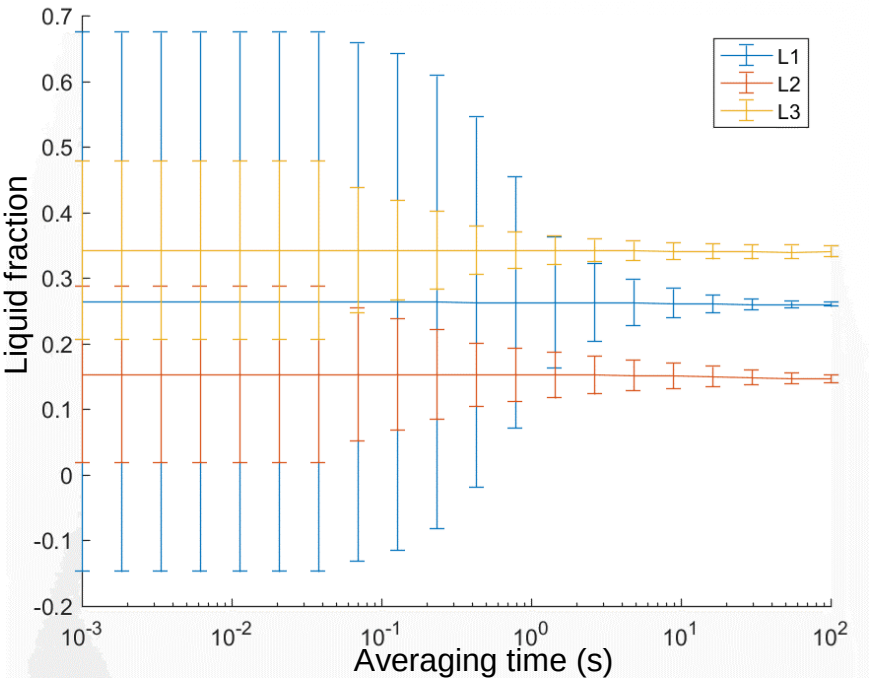

Figure 17. Liquid fraction determined by Optek liquid sensors with tubes of various diameters as a function of the averaging time. (L1) 1/16" OD, 0.5 mm ID FEP tube, (L2) 1/16" OD, 1.0 mm ID FEP tube, (L3) 6 mm OD , 4 mm ID glass tube.

S5. Details of the online GC autosampler

We have attempted to use a conventional approach to sampling using a 6-way valve with the evaporated liquid feed. However, testing the system for the cinnamaldehyde semihydrogenation, we found that the temperature required to evaporate the solution was above 250 °C. The thermal power needed to evaporate the 1 mL/min liquid feed was above 200 W requiring 2 independent temperature controllers – one for the inlet flow and the other to keep the temperature of the valve stable. The biggest problem, came, unsurprisingly, from the formation of minor by-products that were collected in the sampling loop and clogged it within 3-6 hours on stream.

Therefore, we used a standard autosampler but modified the vial holder to have a constantly refillable vial there. Figure 18 shows the scheme of the modified autosampler. Here, we made a custom PTFE block with a 3mm cylindrical hole for the liquid. The inlet gas-liquid flow was coming from the bottom of the cylindrical hole. The gas was quickly moving upwards, while the liquid was moving through a 5 mm OD FEP flexible tubing by gravity. The sampling was done conventionally from this mock vial repeatedly. The volume of the vial was around 50 μL corresponding to very low dead volume. To minimise the chances of collecting gas bubble with the syringe, we set the infusion speed as a medium. However, reproducibility of the area of the peaks was not excellent being ±10% likely because some gas droplets were inadvertently collected by the sampler. Therefore, we used an internal standard. Alternatively, a tub-in-tube degassing might be used to remove gases from entering the sampler.

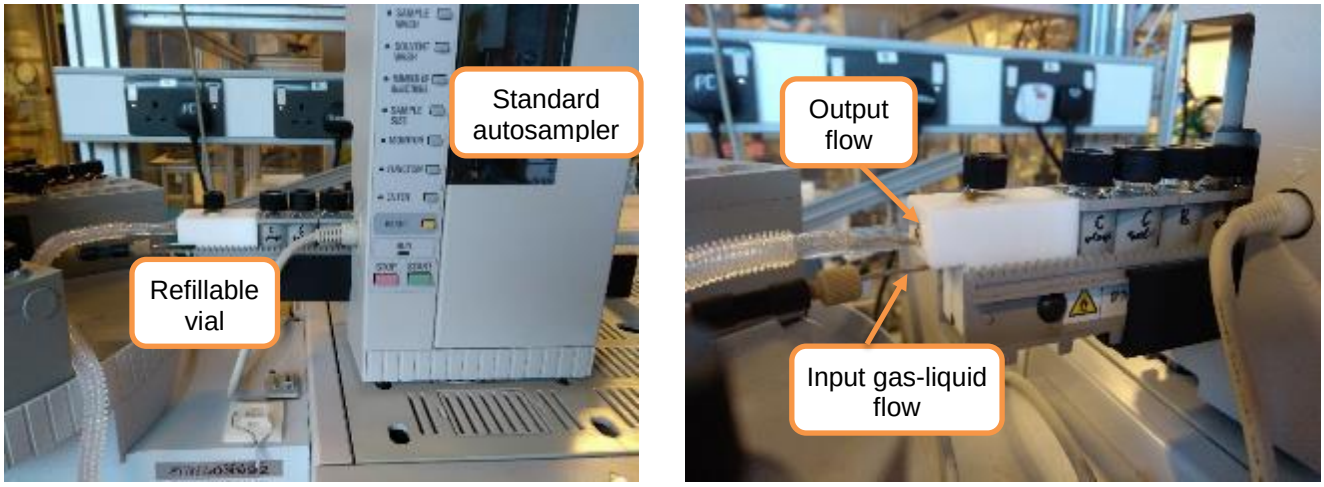

Figure 18. Modification scheme of the autosampler used for the online GC analysis in the current work.

References

1 D. J. Kim, Z. Fisk, D. J. Kim and Z. Fisk, *Rev. Sci. Instrum.*, 2012, **83**, 1–9.

2 A. A. Topalov, I. Katsounaros, J. C. Meier, S. O. Klemm and K. J. J. Mayrhofer, *Rev. Sci. Instrum.*, 2011, **82**, 1–5.

3 C. Wagner, A. Genner, G. Ramer and B. Lendl, *Model. Program. Simulations Using LabVIEW^{TM} Software. InTech.*

4 D. E. Fitzpatrick, C. Battilocchio and S. V. Ley, *Org. Process Res. Dev.*, 2016, **20**, 386–394.