ICP-MS Automation Script

Code Author: Mohamed Harhash, PhD student in Environmental Analytics.
Primary Email: m.harhash@daad-alumni.de
Alternative Email: Harhash007@yahoo.com

Acknowledgment

This code was originally developed by Mohamed Harhash, under the supervision of Dr. Lars Duester, during his Ph.D. at the Federal Institute of Hydrology, Koblenz, Germany.

If you build upon or further develop this code, please acknowledge or cite the original work. For any inquiries, you can contact Mohamed Harhash at the provided emails.

Citation

Harhash, M. (Supervised by Dr. Lars Duester, Federal Institute of Hydrology, Germany). (2024). *Automation solution for ICP-MS* (Version 1.0; Python code). GitHub. Available at https://github.com/Harhash2024/ICP-MS-AUTOMATION.git

Objective:

The main objective was to automate an inductively coupled plasma mass spectrometer (ICP-QMS 7700 / ICP-QQQ- MS 8900) for the monitoring of metal(loids) in river water in a single measurement run with a high time resolution of one hour. To achieve this objective, a self-constructed sample collector/autosampler was constructed to collect samples continuously, 24 hours a day, and an automated ICP-MS instrument was programmed to automatically initiate and perform measurement sequences at scheduled times.

Code capabilities :

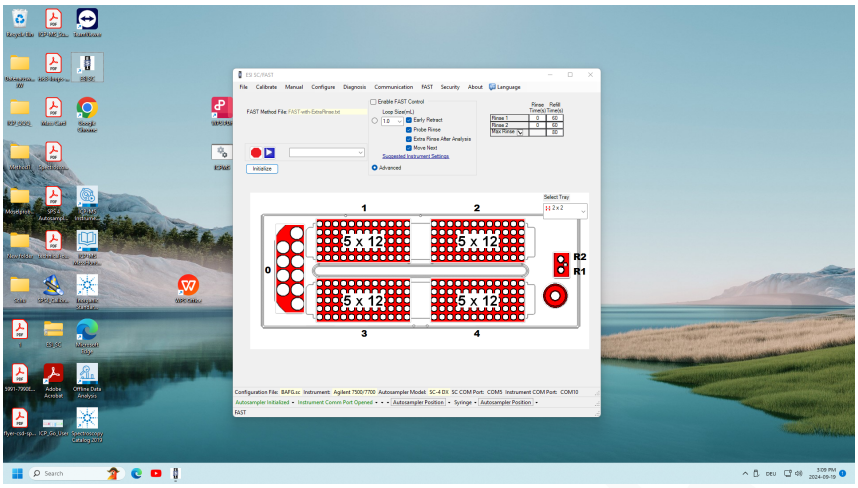
The code is developed to automatically initiate and manipulate the ICP Go software to operate the Agilent ICP-MS 7700 and ICP-QQ-MS 8900 instruments or any other Agilent instrument compatible with the ICP Go software. It carries out measurement sequences and automates the process : 1) logging in to the ICP Go software, 2) turning on the plasma, 3) waiting for warm-up time, 4) selecting and creating batches, 5) performing batch tuning, 6) starting batches and measurements, 7) monitoring errors, 8) sending emails about instrument status, and after measurements, 9) turning off the plasma, 10) logging out and closing the software.

Content

Before you start: 3
Instructions for using ICP Go software 3
Python automation code: 5
Steps to run the code 5
Code Explanation with Snippet 8
1. Open the ICP Go software 8
2. Logging into the ICP Go software. 9
3. Turning on the plasma 10
4. Creating a batch 11
5. Adding the batch to the queue 13
6. Warming up time 14
7. Run the queue and start tuning and the batch 15
8. Waiting tuning of the batch till finish 16
9. Check the error and send an email about the instrument status 17
10. Waiting time for the measurement 18
11. Log out and close ICP Go software. 19

Before you start:

- Make sure that the autosampler software is always opened and minimized.

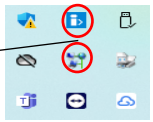


- ICP Go software is an interface to operate and control Agilent ICP-MS machines, and you have to run ICP Go software in the background when running automation code.

Instructions for operating ICP Go software

1. **Double-Click to Initiate ICP Go Software:**
   - Locate the ICP Go software icon  on your Desktop.
   - Double-click the icon to start the program.
2. **Running in the Background:**
   - Once the ICP Go initiated, the software will continue to run in the background, and It will remain active until you manually close it.

Make sure that the Masshunter service and ICP Go software is running in background

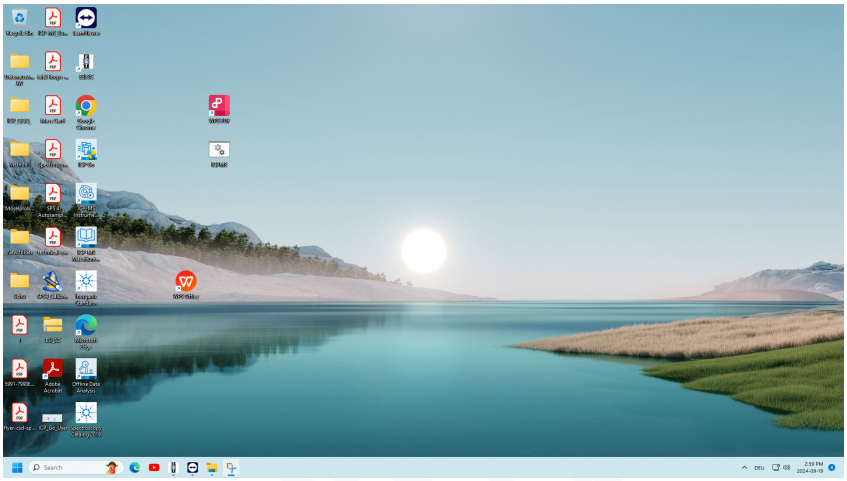


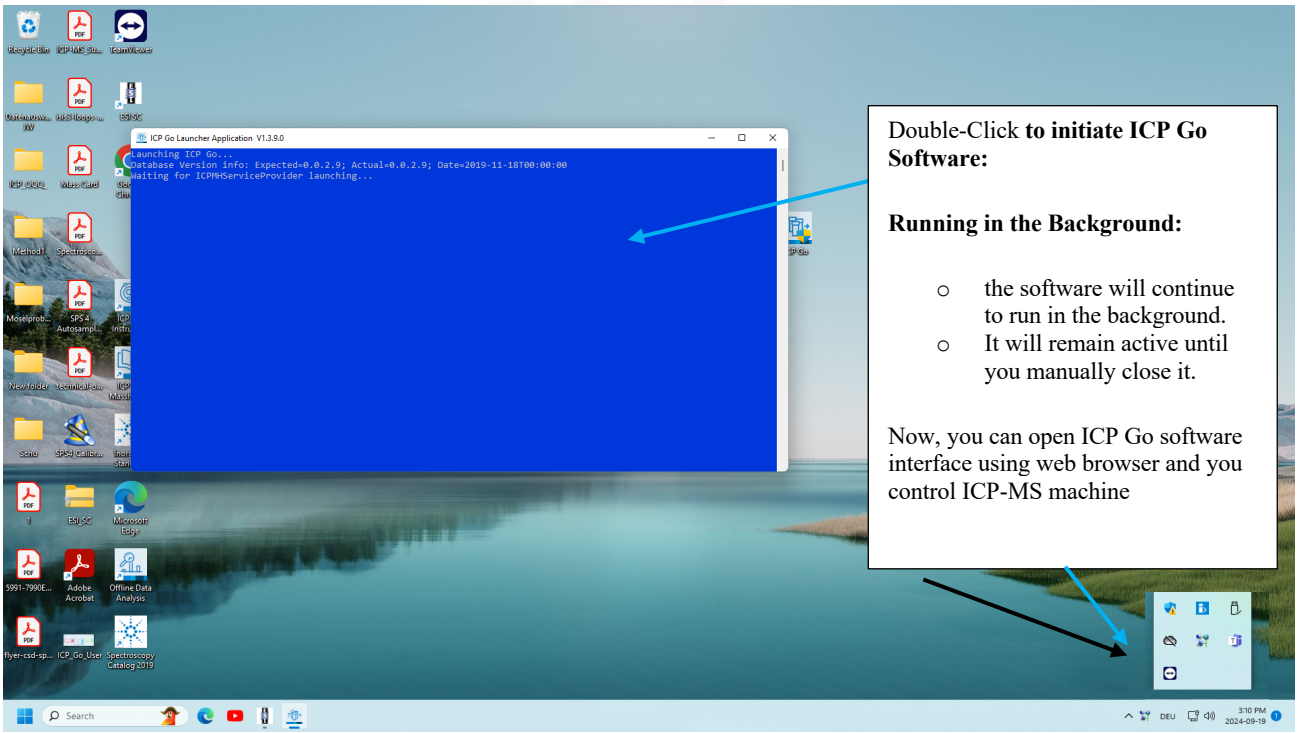
3. **Accessing the ICP Go Software Interface:**
   - Double-click the ICP Go icon on your computer. This will automatically open your web browser to the correct URL (*http://xxx.xxx.xxx.xxx:5000*) to access the ICP Go interface.
   - If needed, you can also copy this ***URL*** from ICP Go and paste it into any web browser's address bar to open the software manually.

- The ICP Go user interface will open in the browser window, allowing you to control the ICP-MS machine. Save this URL from ICP Go and replace it in the ICP-MS automation code.

4. Operating the ICP-MS machine:

- Use the web-based interface ICP Go software to perform tasks such as turning plasma on, creating batches, auto-tuning, starting analyses, monitoring instrument status, and adjusting settings. It has all control options which are provided by MassHunter software.





Python automation code:

Steps to run the code

1. Install python 3.x

To install Python through Anaconda:

- Go to the official anaconda website and download it.
- Choose the version suitable for your operating your system, and anaconda includes Python 3 by default.
- Windows: Install anaconda by double click *anaconda***.exe** file and follow the instructions.
- After installation, you can verify installation by opening Anaconda **prompt** and typing this commend `conda --version` to know the conda version.
- To keep python projects organized, create separate environment for each project.

```
Conda create -n myenv python=3.9   #### you can replace myenv with the name of your environment and your python version.
```

- Activate the environment

```
conda activate myenv
```

- Launch Python or Jupyter
  - You can do this through the Anaconda interface.
  - To start a Jupyter Notebook, click the Launch button next to Jupyter Notebook.

2. Download the code:

a) You can download the code directory from this GitHub repository.

https://github.com/Harhash2024/ICP-MS-AUTOMATION.git

OR

b) Clone the repository:

Install Git:

Download Git from the official website: https://git-scm.com/downloads.

Run the installer and follow the prompts to complete the installation.

Restart your **command prompt** to ensure the new git command is recognized.

Command prompt:

```
git clone https://github.com/Harhash2024/ICP-MS-AUTOMATION.git
cd ICP-MS-AUTOMATION
```

1) Install Python dependencies:

Install the necessary Python packages using the `requirements.txt` file:

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

3. Set up ChromeDriver: Download ChromeDriver from

https://developer.chrome.com/docs/chromedriver

Ensure that the ChromeDriver is either in your system's `PATH` or provide the full path in the script:

```
python

driver =
webdriver.Chrome(executable_path='C:/path/to/chromedriver.exe')
```

4. Configure Email Notifications:

Update the script with your email credentials to enable the error notification feature:

```
python

gmail_email = "your-email@gmail.com"
gmail_password = "your-app-password"
recipient_email = "recipient-email@example.com"
```

5. Ensure ICP Go Software Accessibility:

Ensure that the ICP Go software is accessible via the web interface at *`http://xxx.xxx.xxx.xxx:5000`*. ***Update this in the code as well***.

To run the script, navigate to the directory containing it and execute the following steps:

1. Place your Python script (***`code_name`***`.ipynb`) in a preferred folder.
2. To schedule and run the code, create a `.bat` file (a `.bat` file is used to execute commands in Windows).
3. In the notebook file, copy and paste the following command lines, modifying the paths to match the locations of your folders.

- Commend lines:

```
@echo off
```

```
:: Path to the Jupyter executable
"C:\path\to\jupyter.exe" nbconvert --to notebook --execute
"C:\path\to\your_notebook.ipynb" --output
"C:\path\to\output_notebook.ipynb"
```

- Name this notebook file as you prefer `e.g., ICP-MS_Automation.bat`

4. Schedule the .bat File in Task Scheduler

   - Press **Windows Key + R** on your keyboard to open the **Run** dialog box.
   - Type ***taskschd.msc*** and press **Enter**. This will open the **Task Scheduler**.

     a. Open **Task Scheduler** and create a new task.
     b. In the **Actions** tab, select **Start a program**.
     c. Browse to the .bat file you created (e.g., ICP-MS_Automation.bat).
     d. Set the **Triggers** and other settings as needed, then save the task.

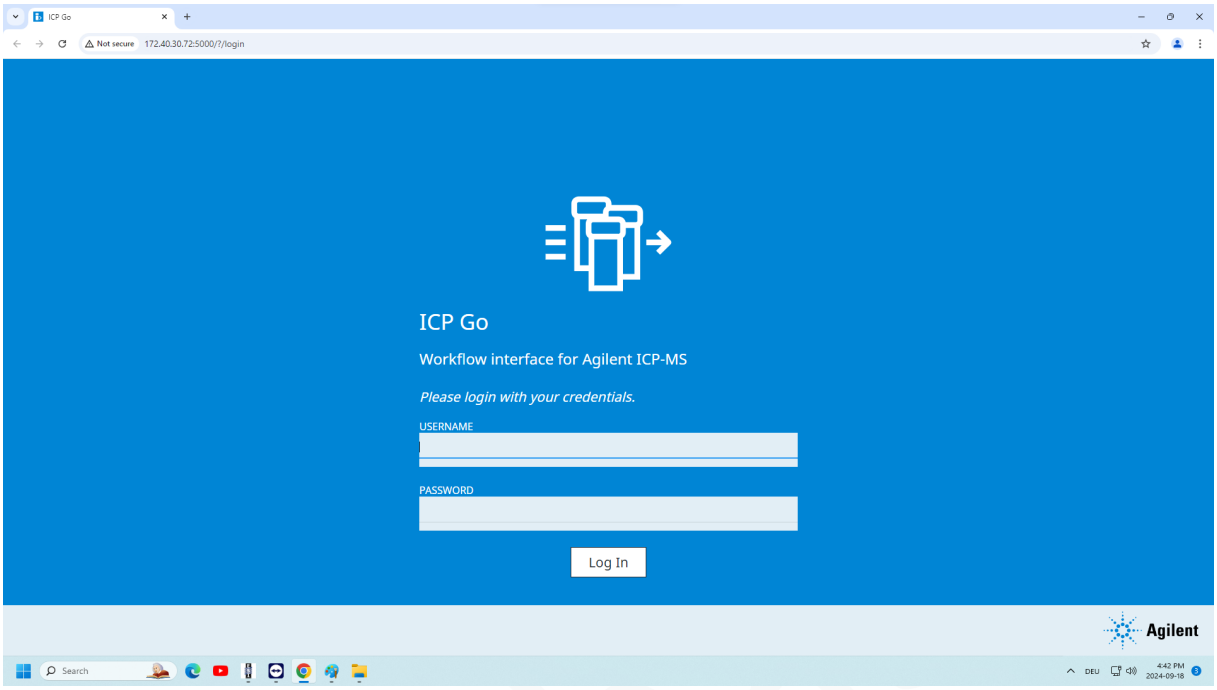
With this setup, your .bat file will directly execute your Jupyter Notebook file each time it's run.

Script Workflow:

1. **Login**: The script logs into ICPMS Go software using stored credentials in the code.
2. **Plasma Control**: The plasma is turned on.
3. **Batch Creation**: The script creates and saves a new batch using the appropriate template based on the current date and time.
4. **Warm-up**: It waits for the plasma to warm up.
5. **Batch Queueing**: Adds the created batch to the queue for measurement.
6. **Tuning**: Performs any necessary tuning for the measurement.
7. **Measurement**: The script waits for the measurement to complete.
8. **Error Handling**: If any errors are encountered, an email notification is sent, and the error is logged.
9. **Turn off the plasma**
10. **Logout and Shutdown**: Once the batch is complete, the script logs out of ICPMS Go and safely closes the software.

Code Explanation with Snippet

1. Open the ICP Go software

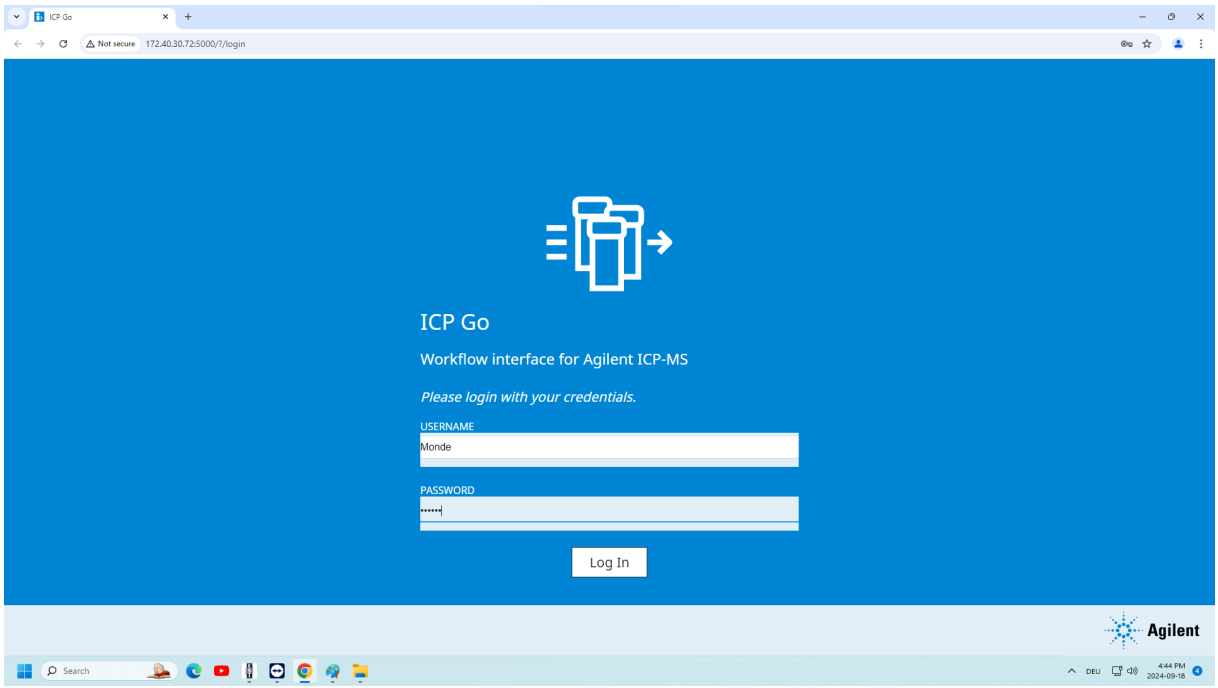


Code snippet:

```
#1- Opening ICPMS Go software

driver = webdriver.Chrome()
driver.get("xxx.xxx.xxx.xx:5000")
### Check this IP address if it is the same for your ICP-MS. it is provided
with the software
driver.maximize_window()
```

2. Logging into the ICP Go software.

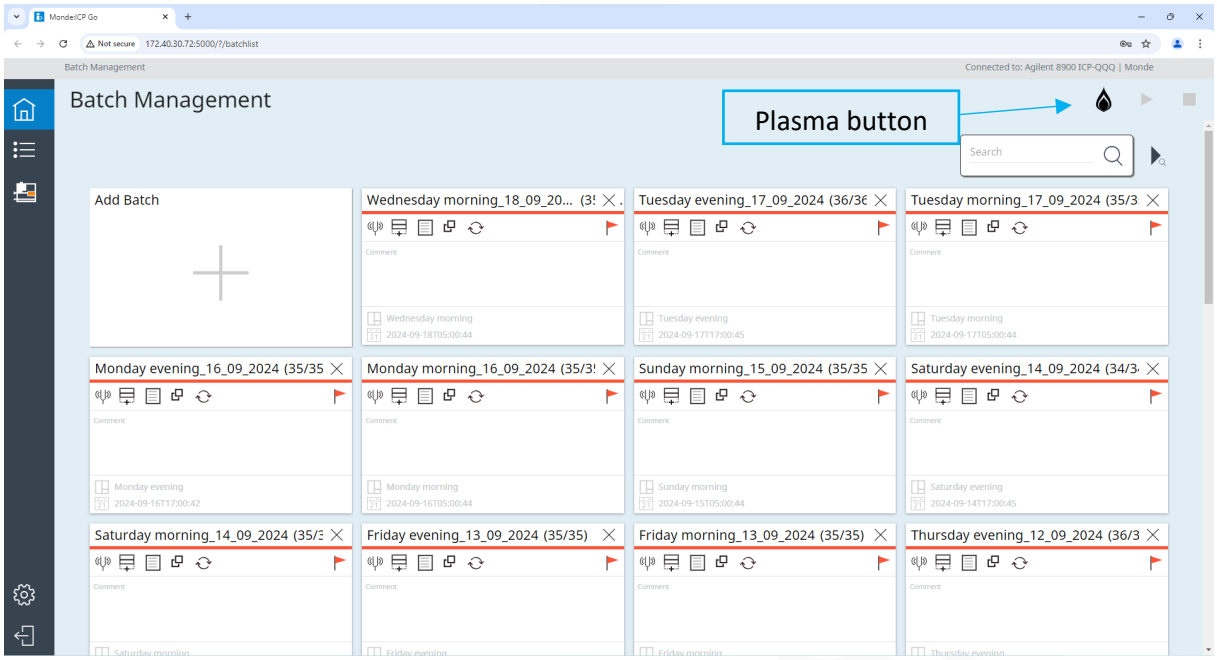
- You can modify ***user name*** and ***password*** according to your system



Code snippet:

```
#2-Logging into ICPMS Go
wait = WebDriverWait(driver, 20)
username_input = wait.until(EC.element_to_be_clickable((By.NAME,
"username")))
username_input.click()
username_input.send_keys("your_username") ##### replace it with your own user
name
password_input = driver.find_element(By.NAME, "password")
password_input.send_keys("Monde1") #### replace it with your own password
login_button = driver.find_element(By.ID, "submit")
login_button.click()
## waiting 7 second to laod the page
time.sleep(7)
…
```

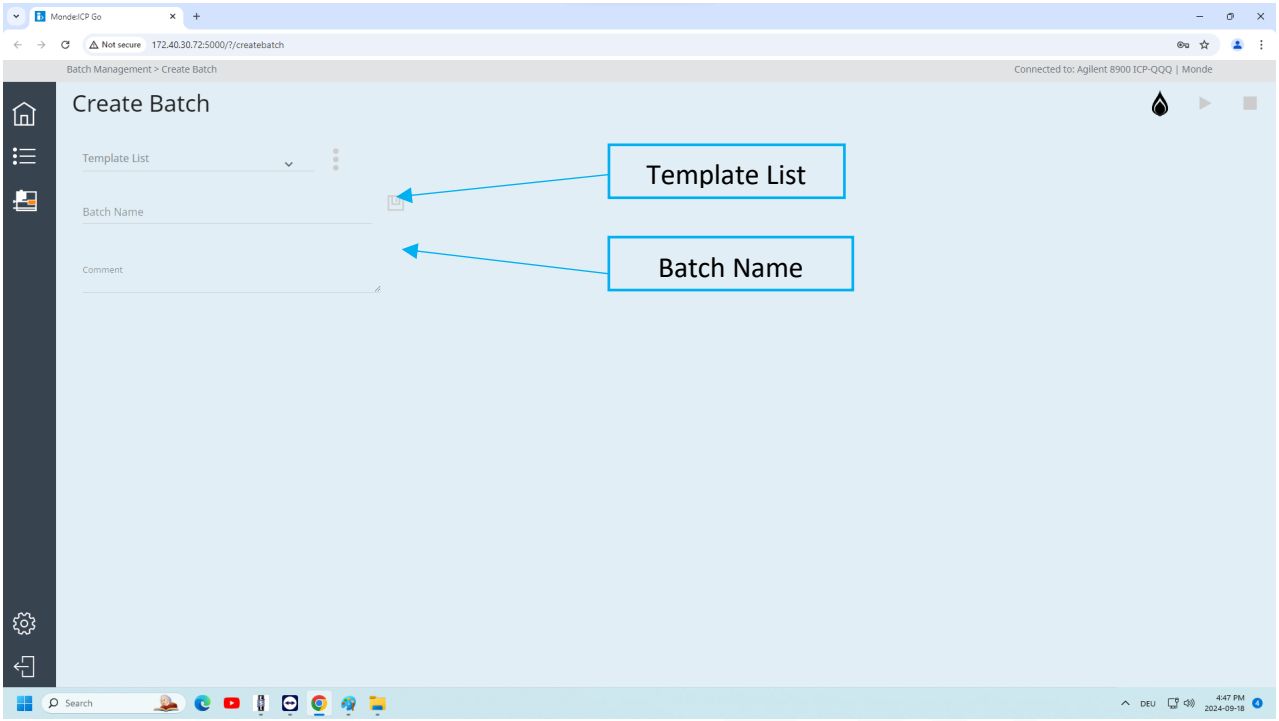
3. Turning on the plasma



Code snippet:

```
#3-Starting the plasma
driver.find_element(By.CSS_SELECTOR, "#plasma .pe-ripple__mask").click()
time.sleep(2)
driver.find_element(By.CSS_SELECTOR, "#yes .pe-ripple__mask").click()
```

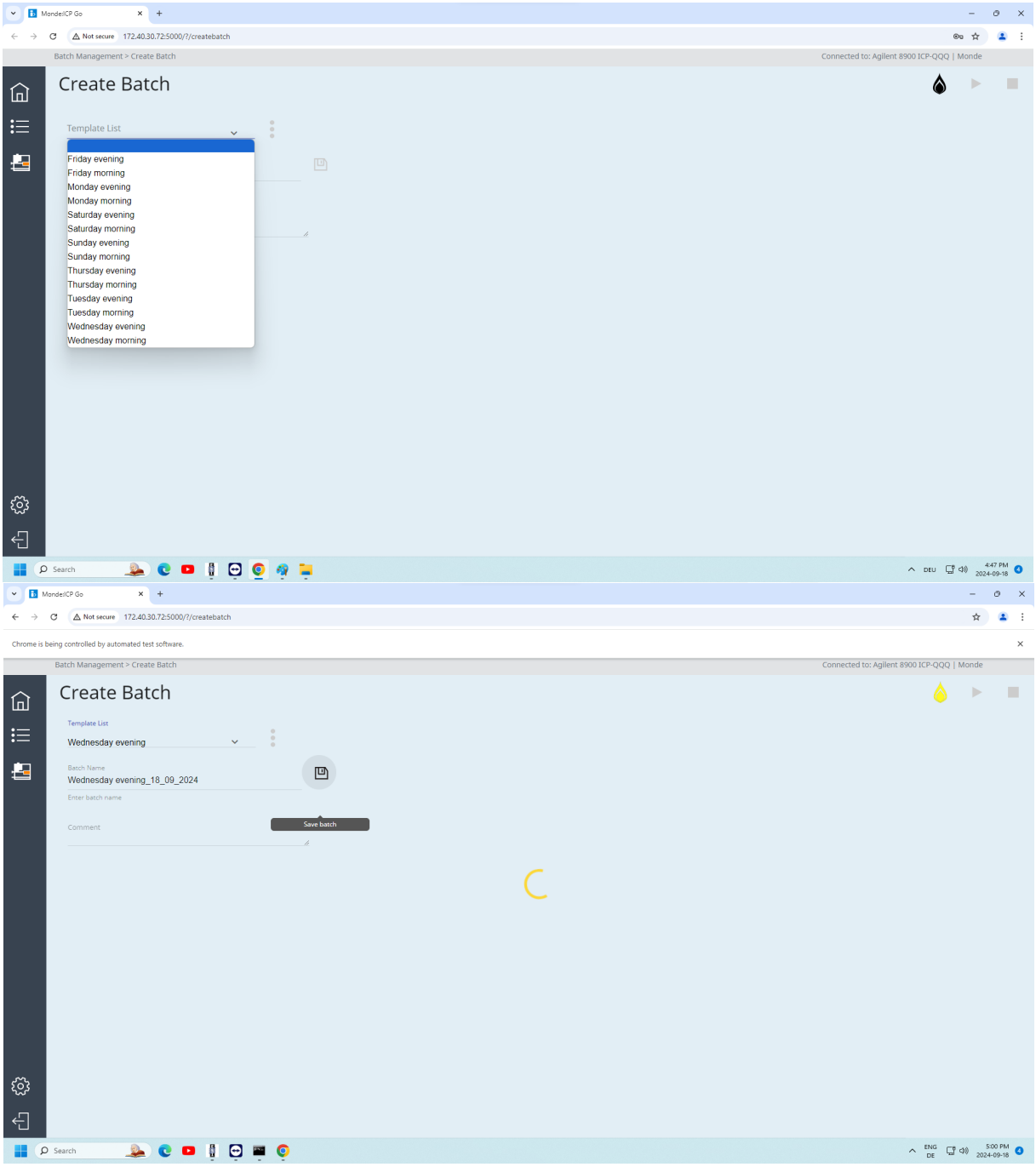
4. Creating a batch



Code snippet :

```
#4-Creating and managing new batches based on templates
driver.find_element(By.CSS_SELECTOR, "svg").click()
driver.find_element(By.ID, "template").click()

# determine the current date,time and day of the week
current_day_index = datetime.datetime.now().weekday()
current_time = datetime.datetime.now().time()
current_time_period = "morning" if current_time <
datetime.datetime.strptime("12:00:00", "%H:%M:%S").time() else "evening"
day_names = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday"]
selection = f"{day_names[current_day_index]} {current_time_period}"

# click dropdown list for selecting proper batch for the current day and time
time.sleep(4)
dropdown_element = driver.find_element(By.ID, "template")
dropdown_element.click()
dropdown = Select(dropdown_element)
dropdown.select_by_visible_text(selection)

# Naming the created new batch as day name 'evening' or 'morning' and date
timestamp = datetime.datetime.now().strftime("%d_%m_%Y")
element_name = f"{selection}_{timestamp}"
print(element_name)
driver.find_element(By.NAME, "batchname").click()
time.sleep(5)
driver.find_element(By.NAME, "batchname").send_keys(element_name)
time.sleep(10)

# Saving the batch
driver.find_element(By.CSS_SELECTOR, "#savebatch .pe-ripple__mask").click()
time.sleep(10)
```

- Open the list of batch templates and select the appropriate batch.



Dynamic creation and management of batches

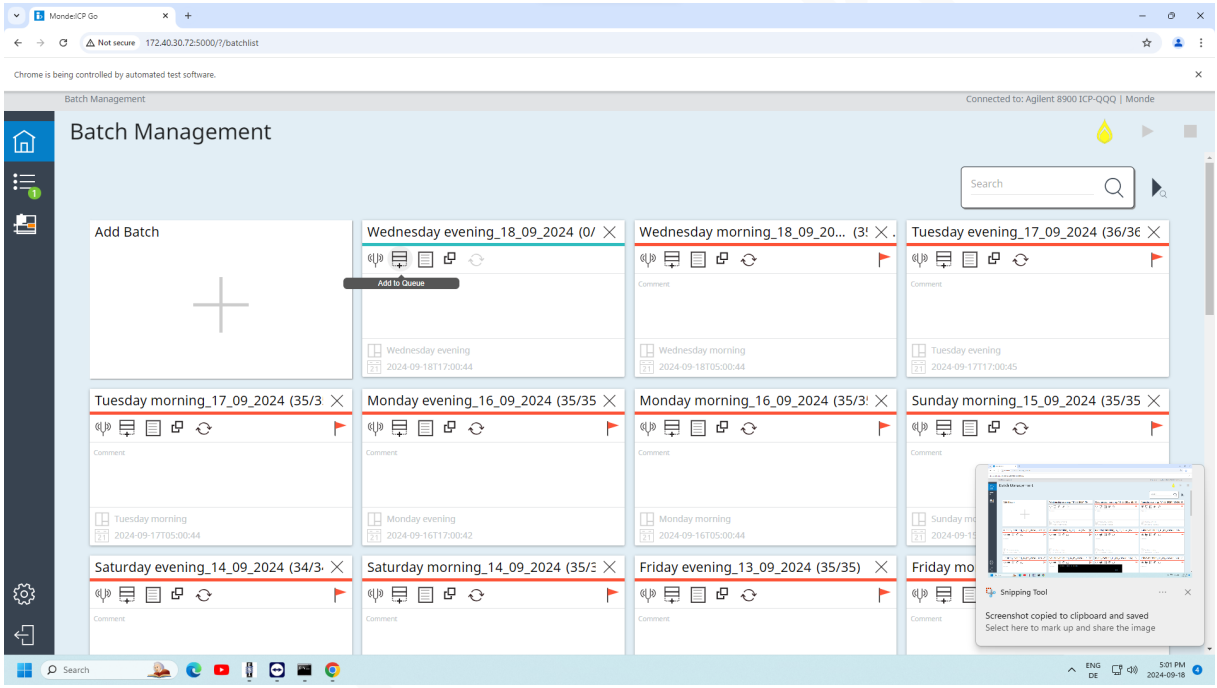
- As the automation is designed for daily monitoring, the ICP-MS machine will run twice a day, morning and evening, to measure multiple elements in river water. Therefore, two batch templates are created for each day of the week.
- Each batch is configured for the appropriate location of the sample vials and the two blanks based on the corresponding day.
- The evening batch is configured to measure samples in Rack 1 and Rack 2, which contain samples collected between 7:00 and 18:00.

- The morning batch is configured to measure samples in Rack 3 and Rack 4, which contain samples collected between 19:00 and 6:00 on the previous day.
- For clarification, a sample named '7:00' represents the mixed sample collected from 7:00 to 8:00.

Naming System of created batch.

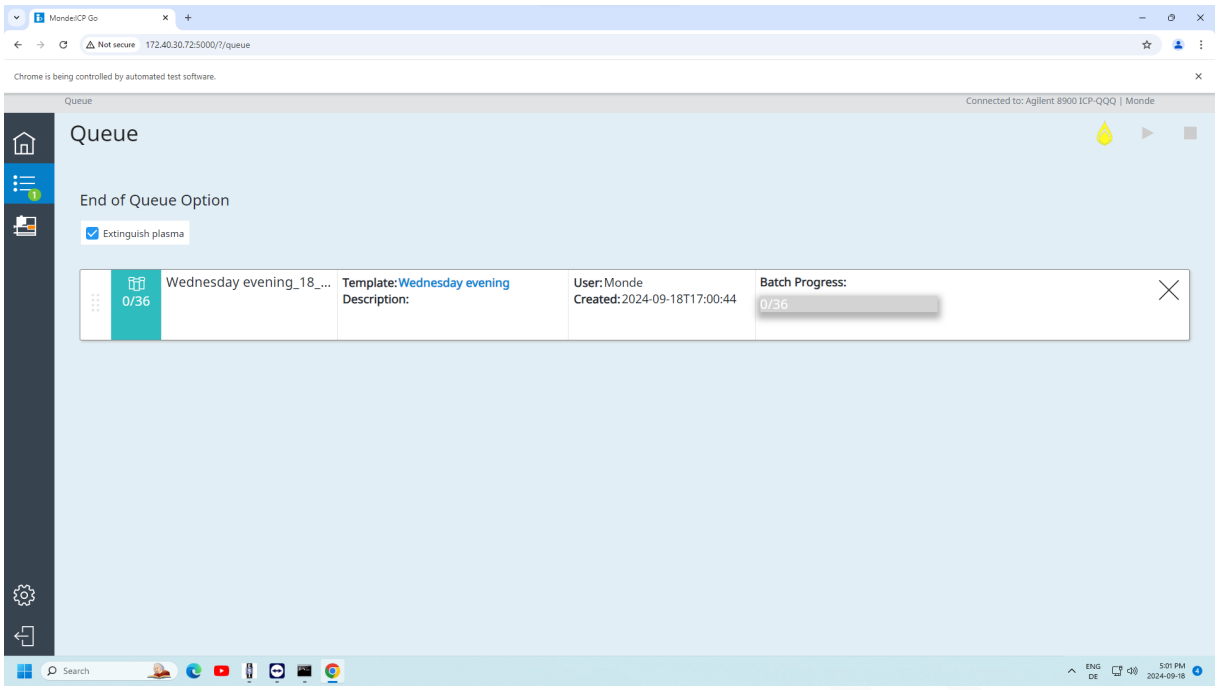
The code will first determine the current date, time, and day of the week (e.g., Monday, Tuesday…). It will then identify whether it is morning or evening, after 12:00 is considered evening, and before 12:00 is considered morning. Based on this information, the code will select the appropriate batch, such as "Monday evening" from the batch list, and name it as 'Monday evening _date of measurement'.

5. Adding the batch to the queue

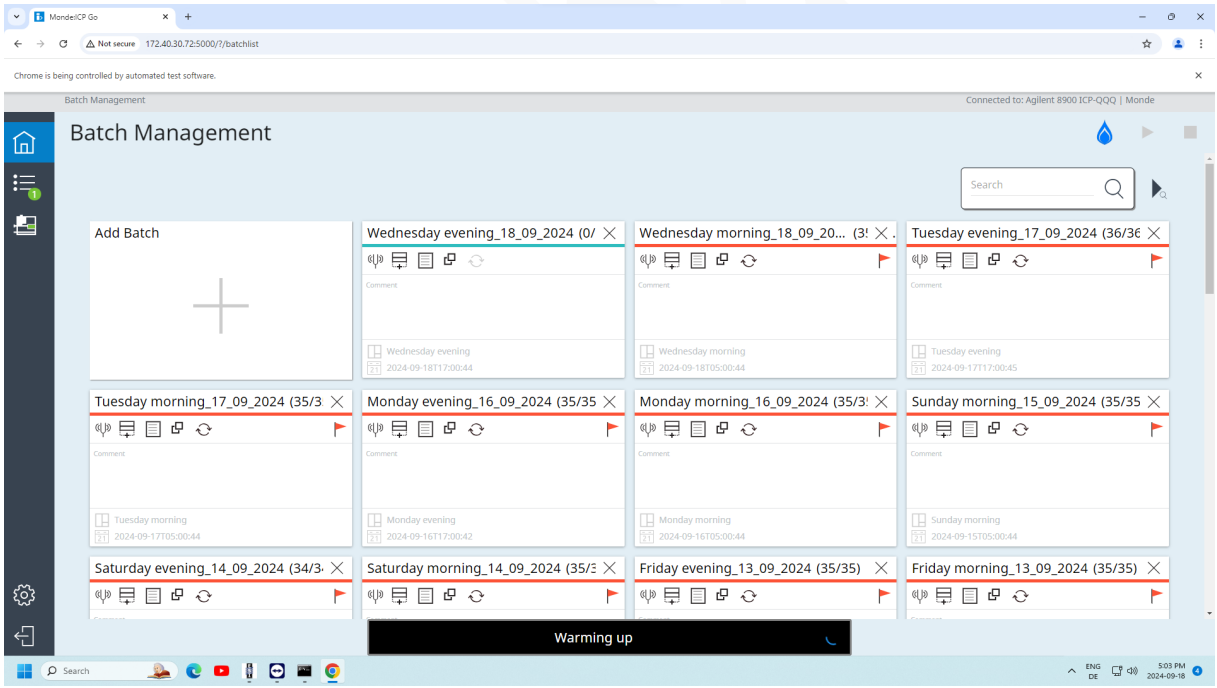


Code snippet :

```
#Add the created batch to queue

time.sleep(25)
xpath_selector = f'//*[@id="queuebutton_{element_name}"]/div'
element = driver.find_element(By.XPATH, xpath_selector)
element.click()
```



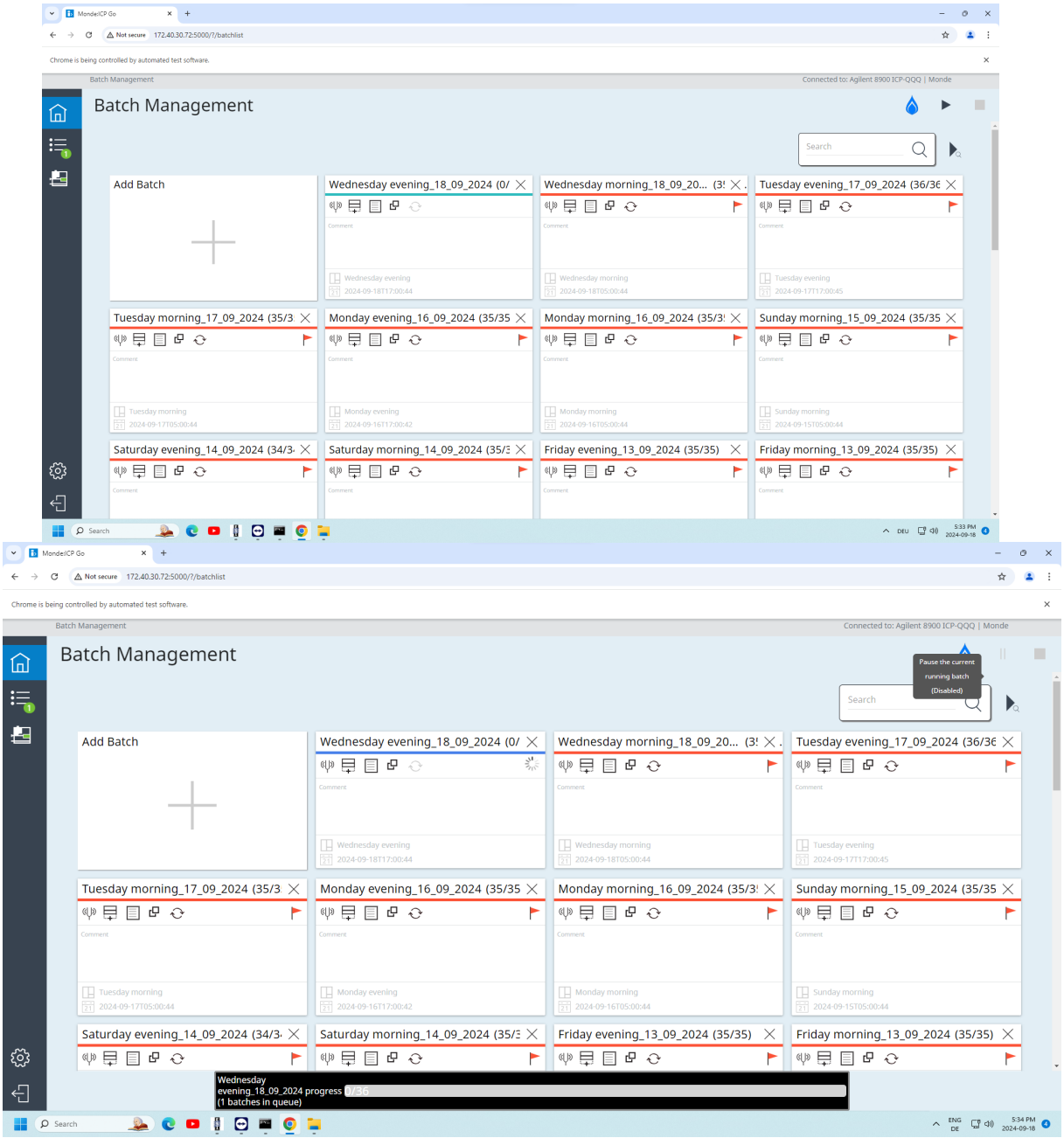
6. Warming up time



Code snippet :

```
#6- waiting warming time

time.sleep(1900)
```

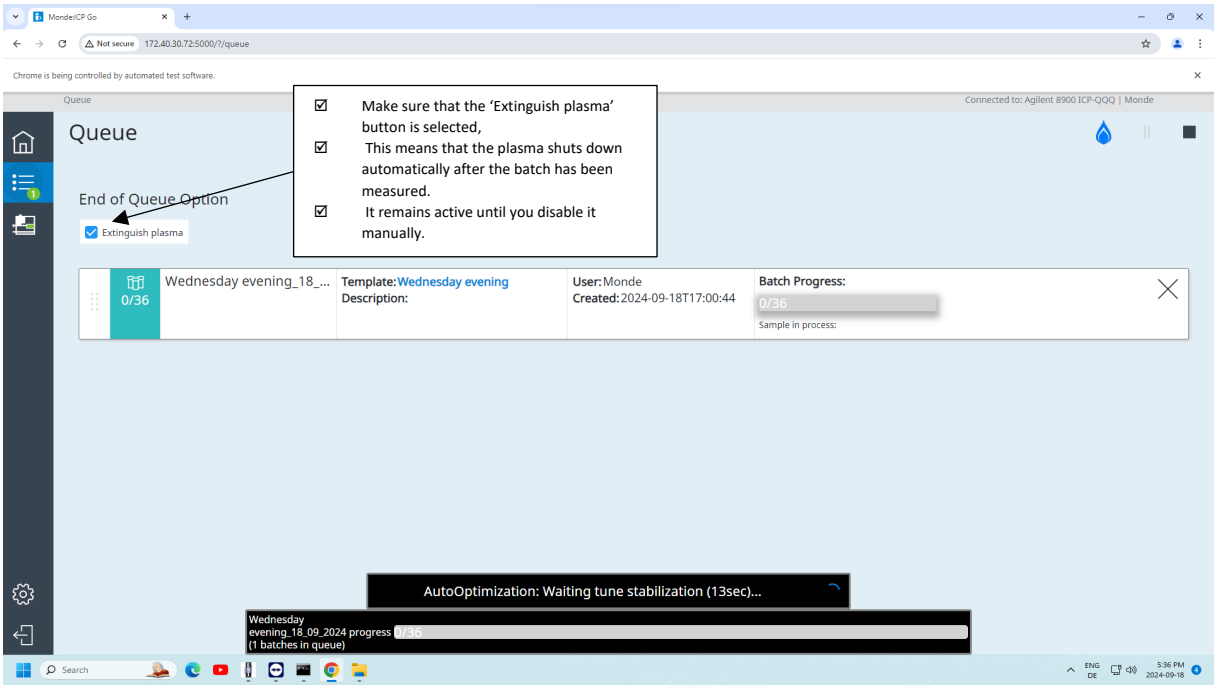
7. Run the queue and start tuning and the batch



Code snippet :

```
# 7- Run the queue and start tuning and the batch

        run_button_selector = "#run .pe-ripple__mask"
        queue_button_selector = "#queue .pe-ripple__mask"

        click_with_retry(driver, run_button_selector)
        time.sleep(20)
        click_with_retry(driver, queue_button_selector)
```

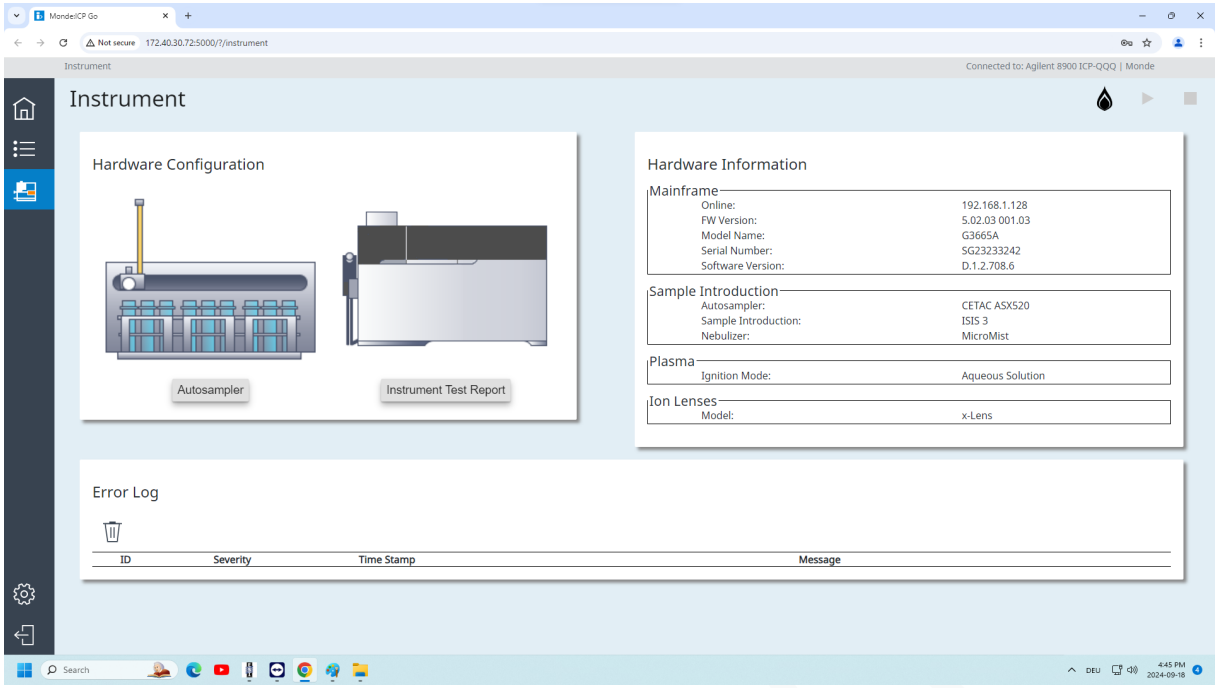
8. Waiting tuning of the batch till finish

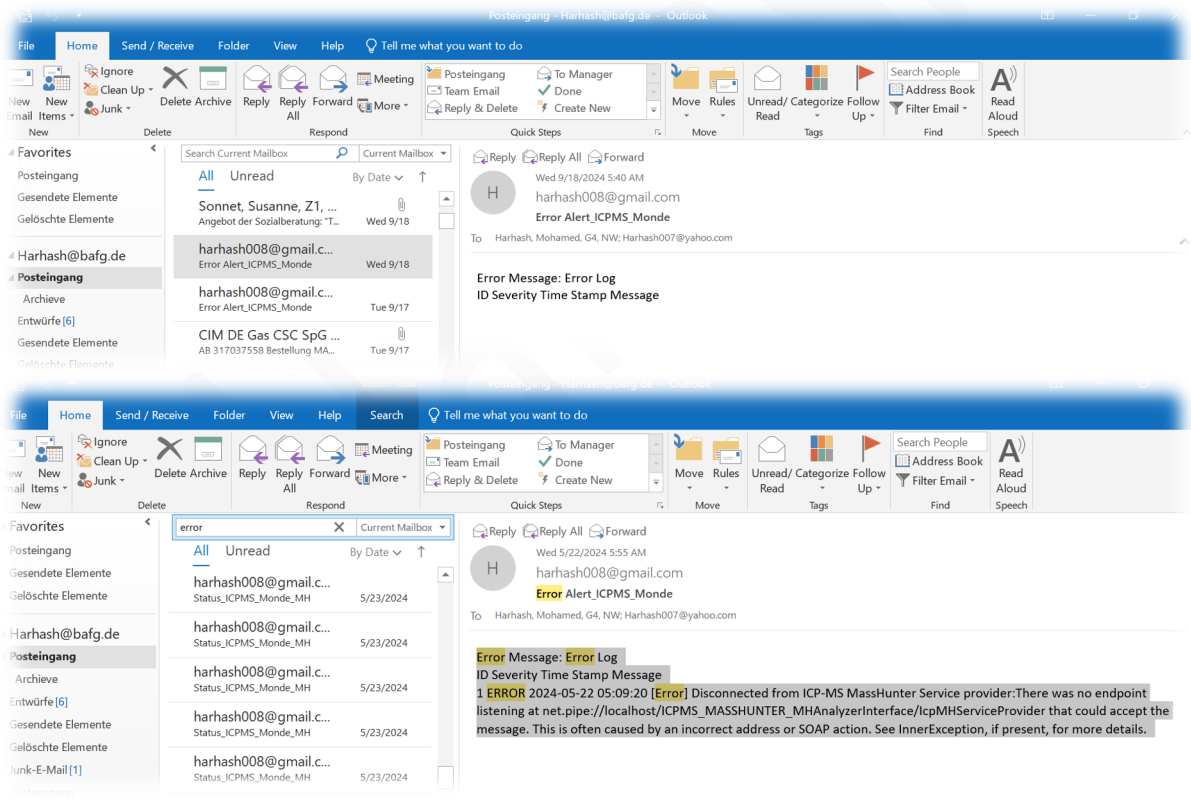


Code snippet :

```
#8- Waiting tuning of the batch till finish
..
time.sleep(180)
…
```

9. Check the error and send an email about the instrument status





Code snippet :

```
#9- check the error and send an email about the instrument status

try:
    instrument_button = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.CSS_SELECTOR, "#instrument .pe-
ripple__mask"))
    )
    instrument_button.click()
except Exception as e:
    print(f"Failed to click the instrument button: {e}")

error_message_selector = "#errorLog > div"
try:
    error_message_element = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.CSS_SELECTOR,
error_message_selector))
    )
    error_message_text = error_message_element.text
    if error_message_text:
        gmail_email = "your email @gmail.com"  #### your E-mail
        gmail_password = "xxx xxx xxx xxx"    ## access code of your E-mail
obtained from you eamil setting ## your-app-password
        recipient_email = " recipient-email@example.com "### here you can use
your E-mail
        subject = "Error Alert_ICPMS_Monde"                    #########
        body = f"Error Message: {error_message_text}"

        msg = MIMEText(body)
        msg["Subject"] = subject
        msg["From"] = gmail_email
        msg["To"] = recipient_email

        server = smtplib.SMTP("smtp.gmail.com", 587)
        server.starttls()
        server.login(gmail_email, gmail_password)
        server.sendmail(gmail_email, recipient_email, msg.as_string())
        server.quit()
except Exception as e:
    print(f"An error occurred: {e}")

print("Email sent successfully.")
```

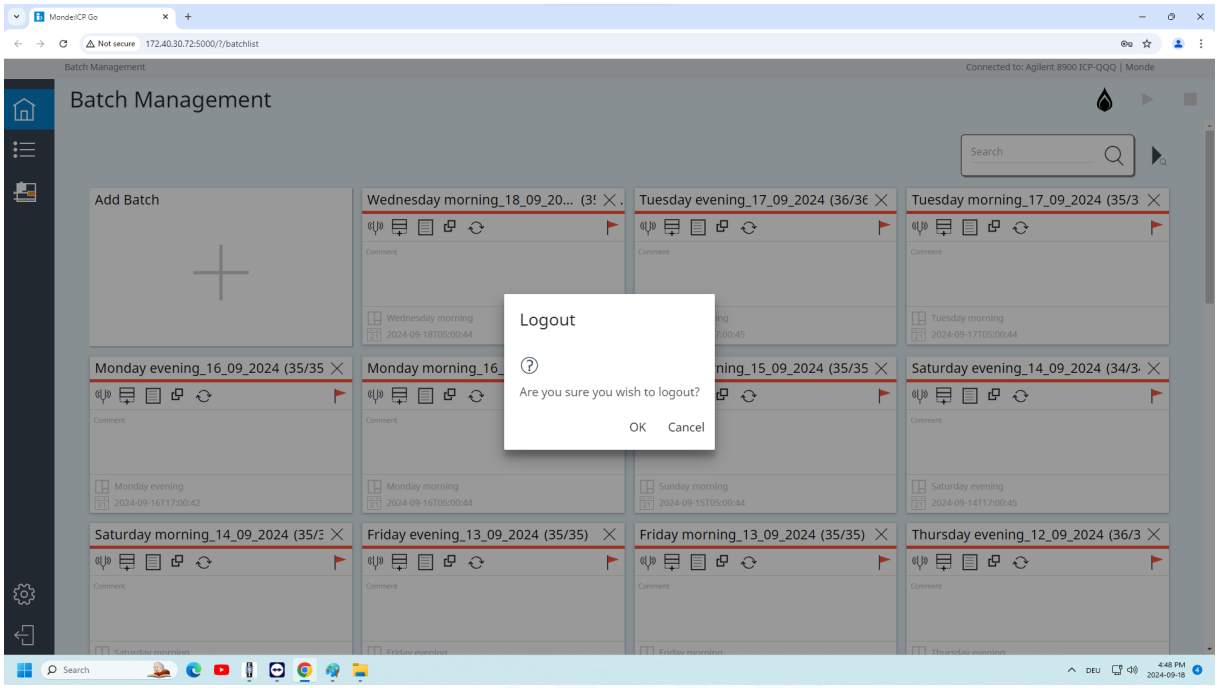
10. Waiting time for the measurement

- You can customize your needs

Code snippet :

```
# 10- Waiting time for the measurement (around 4 hours) and it is more than
our measurment time to ensure enough time after measurment ###

driver.find_element(By.CSS_SELECTOR, "#queue .pe-ripple__mask").click()
time.sleep(16000)
```

11. Log out and close ICP Go software.



Code snippet :

```
# 11-  Sing out  and close ICP Go software


try:
    logout_button = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.CSS_SELECTOR, "#login .pe-
ripple__mask"))
    )
    logout_button.click()
except TimeoutException:
    print("Logout button not clickable. Continuing without clicking.")
try:
    ok_button = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.CSS_SELECTOR, "#ok .pe-ripple__mask"))
    )
    ok_button.click()
except TimeoutException:
    print("Ok button not clickable. Continuing without clicking.")


time.sleep(5)
driver.quit()
```

Alternative Methods to schedule the code initiation

Use a Robust Scheduling System:

- Use a robust scheduling system, such as Windows Task Scheduler, as it is managed by the operating system and can automatically restart failed tasks.
- You can set up triggers like 'on system start-up' or 'on user logon' to ensure the task restarts if it fails. Another advantage is that you don't need to initiate the code or Anaconda each time the computer restarts.

1. **Using windows task scheduler:**
    - The Windows Task Scheduler allows you to schedule scripts, programs, or tasks to run at specific times or intervals.
    - **Steps:**
        1. Open windows task scheduler.
        2. Click on "Create Basic Task" or "Create Task."
        3. Name the task and set a trigger (e.g., daily at a certain time).
        4. In the "Action" tab, select "Start a program" and browse to the script or executable you want to run.
        5. Set any conditions or settings as required and save the task.
    - This method is reliable for scheduling on Windows machines and doesn't require changes to the code.

However, there are also other methods by which the code can be scheduled.

This includes:

2. **Embedded Scheduling in the Code:**
    - You can include scheduling logic directly within the code itself using specific libraries or modules.

    For example, in Python, you can use the schedule or time libraries to set specific times for the code to run.

3. **Using Python Scheduler Libraries:**
    - Libraries such as AP Scheduler (Advanced Python Scheduler) offer flexible scheduling.