Dependencies:

Graphic user interface depends on streamlit for creating web application;
on matplotlib for plotting graphs;
and on pandas for creating tables

Structure of software:

Urania is operated by 2 independent programs that communicate one with another through JSON files.

One program is responsible for interactions with mass spectrometer, another one is responsible for data visualisation and graphic user interface.

Structure of libraries:

The code consists of the following libraries: AbnormalityReaction.py; Functions.py; GUI_Settings_Menu.py; JSONoperators.py; main.py; RGA_comms.py; VSC_comms.py; StreamlitGUI.py;

AbnormalityReaction.py library is responsible for finding abnormalities in mass spectrometer readings and report them to user

FindAbnormalityInSpectrum function inputs a single spectrum as dictionary; date and time of this spectrum; filename of log; filename of file with quality standarts and options

This functions compares each PPM reading for integer molar masses in spectrum to tolerance boundaries set in the quality standarts file.

This function may run in ambient and on-demand mode, depending whether DoReportToJSON is True or False. If True, function is supposed to run right after spectrum is obtained. It reports all abnormalities to JSON log. If False, function is supposed to run on-demand via GUI; and display all abnormalities to user via GUI

Functions.py library contains functions that do not fit into any other library.

get_time_list function inputs list of spectrums and returns list of corresponding moments of time; at which those spectrums were recorded

plot_mass function inputs list of spectrums and desired index; and outputs the list of PPM's corresponding to given index. For example, it can return all third elements from list of lists.

GUI_Settings_Menu.py library is responsible for settings menu that runs through graphic user interface. It contains functions to modify JSON settings config that other pages rely on.

reset_to_default function erases currently used JSON config and overwrites it with factiory default JSON config

apply_page_settings function replaces given line in JSON config with desired line. Different lines in JSON config correspond to different pages of GUI. It reads old JSON config and makes full copy of it with exception of desired line; then, overwriting old config with this copy.

modify_* functions input Settings dictionary for given line (obtained before) and Settings filename. Those functions let user change given parameter, and then call **apply_page_settings** function to make changes to JSON config

Settings_Page function is a ready–to–use Settings menu that runs through Streamlit

JSONoperators.py library is responsible for working with JSON files.

read_spectrum_json function reads all spectrums from JSON file that contains list of spectrums

read_last_spectrums function reads the specified amount of last spectrums from JSON file. It calls **read_spectrum_json** function and cuts off the unwanted spectrums

read_period_of_time function reads specified amount of spectrums starting from specified moment of time from JSON file. For example, it may read 10 most recent spectrums starting from 01 Jun 2024, 14:00:00.

read_all_page_numbers function reads numbers of all pages that are specified in given JSON config

main.py file runs the main page of graphic user interface

RGA_comms.py library is responsible for sending commands to gas analyser and receiving data from it.

SendPacketsToRGA function inputs a list of str commands. This functions converts those commands to bytes and sends to RGA via TCP protocol.
This function saves all responses from RGA to received_list and returns it.
This function can execute internal commands such as __listen__ or __wait_for_given_mass__. Using those functions, user can control how commands are sent to RGA; and how output is received.
This function raises an error if “ERROR” keyword is detected in output.

GetMassSpectrum function creates a list of prompts for mass spectrometer to receive a mass spectrum with specified initial mass, amount of steps, step length and accuracy. It sends this list of prompts to RGA; receives output, converts the output to required format and

AppendSpectrumJSON function reads specifications (initial mass, amount of steps, step length and accuracy) from metadata in spectrum JSON file; calls **GetMassSpectrum** to get spectrum and writes this spectrum to JSON file

VSC_comms library is responsible for communication with vaccuum control system

StreamlitGUI.py library is responsible for data visualisation through web application.

date_time_input function create widgets for date input and time inputs; allowing user to search for scans made at given moment of time

three_dimentional_spectrum function plots 3d graph with time at X axis; molar mass at Y axis and PPM at Z axis

constant_time_spectrum function plots a graph with molar mass at X axis and PPM at Y axis for specified moment of time. Using slider, user can scroll through different moments of time

constant_time_spectrum_table prints table with numerical value for previous function

constant_mass_spectrum function plots graph with time at X axis and PPM in Y axis for range of specified molar masses. User can input desired molar mass through text input widget

display_one_sample_data function creates a ready-to-use data visualisation page with three different types of graphs displayed. It displays widgets for user to select preferred searching mode (either display most recent spectrums or search for specific moment of time), to select moment of time.

Structure of JSON files:

Spectrum file format

Spectrum files contain results produced from the same pipe over different moments of time in following format:

{"class":"metadata","valve_number":1,"is_a_spectrum":"True","initial_value":1,"amount_of_scans":4096,"step":0.03125}

{"class":"spectrum","time":x,"array":[x1,x2,x3...]}

Time is formatted as seconds since 01 Jan 1970 using standard Python **datetime** library

“Metadata” line contains valve_number (position of multi-inlet valve corresponding for this spectrum), "is_a_spectrum":"True" entry for error prevention; “initial_value”, “amount_of_scans” and “step” enrties correspond to parameters of scanning

Array is formatted as PPM’s for different molar masses starting from “initial_value” with step of “step”

GUI settings file format:

GUI settings file consists of multiple lines; each line corresponds to settings of corresponding page.

Following settings of GUI page are specified in each line of settings file:

- page name
- name of spectrum file
- vertical or horizontal orientation
- default masses to be displayed on const_mass graph
- default amount of spectrums to be displayed
- Options whether to display different types of graphs or no

GUI settings has the following format:

{"page_name": "int", "spectrum_filename": "name", "orientation": "vertical"/”horizontal”, "default_amount_of_spectrums": "int", "default_masses": "int1,int2,int3...", "do_display_3d": "boolean", "do_display_const_mass": "boolean", "do_display_const_time": "boolean"}

GUI settings line specify which page it belongs to; which JSON file is parsed; whether page is displayed horizontally or vertically; how much spectrums are displayed; what masses are displayed on const_mass graph; whether each type of graphs is displayed