Technological Feasibility

Team: TITAN

10/3/2020

Astrophysical Materials Lab Data Logging and Instrument Control Tool

Sponsored by: Dr. Will Grundy & Dr. Jennifer Hanley

Team Mentor: Volodymyr Saruta

Quinton Jasper

Hui Wan

Elijah Anakalea-Buckley

Hao Liu

Table of Contents

1. Introduction

The goal of studying astrophysical materials is to provide a greater understanding of how and why different structures within this universe react in the ways that they do. Planets within the solar system are encompassed by strange extraterrestrial environments, differing radically from the comparatively comfortable environment humans are familiar with on planet Earth. The key elements that establish the definition of an environment are the materials that reside on a planetary body, coupled with a source of energy, or a lack thereof, which changes and mobilizes those materials. But, in order to understand all the bizarre realities taking place amongst the universe, researchers are left to conduct their research down here. The materials of outer planets consist of a variety of gases, some of which include nitrogen, carbon monoxide, methane and ethane; and the study of these materials, and how they react with low temperatures, is spearheaded by the efforts of Lowell Observatory on the NAU campus. Lowell is one of the oldest observatories in the United States, and is greatly responsible for research in astrophysics and astrochemistry. Here, Dr. Will Grundy and Dr. Jennifer Hanley are astronomers that research, study and experiment specifically with tools that simulate these extraterrestrial environments.

Dr. Grundy and Dr. Hanley conduct their research in a lab on the Northern Arizona University campus that collaborates with NAU students, using complex hardware and software to tightly monitor materials in extremely low temperatures. These temperatures simulate the conditions of extraterrestrial bodies much further from the sun, such as the dwarf planet Pluto, or Saturn’s moon Titan. The current system consists of a desktop user interface which controls and monitors the cryo-chamber equipment. The program also keeps track of all data flowing in from the hardware, and user interactions. This data is then dumped into a log file for later analysis.

The problem with the system in place is that it's operating on both outdated software technologies, and limited functionalities. It’s a homebuilt solution established by the clients. Their solution has the core essentials for their research in mind, but it still leaves much room for improvements. The software places limits on hardware controllability, and workflow efficiency. As such, the client seeks a full redesign and an upgrade to a more professional-grade solution.

The mission is to provide the clients with a modern graphical interface solution, implemented on modern technologies and modular architecture.

The solution provided to the client will be, at its core, a complete upgrade of their system. The goal is to provide a clean, modular program architecture that will allow Titan, or the Ice Lab team, to add or remove features without sacrificing the integrity of the code's core functionality.

By providing these researchers with a stronger tool for collecting and analyzing this data, the work done by Titan will provide subsequent researchers a better system to conduct ground breaking research in the astrophysical field. By providing this utility refresh, research can be done with more flexibility, familiarity and efficiency. The software provided will be reliable and configurable to new sensors and hardware.

2. Technological Challenges

The User Interface (UI)

This section will outline some of the different challenges the team expects to encounter with the user interface/user experience component of this project. These include, but are not limited to:

- Displaying a real-time plot of sensor data to the user interface
- Allowing the user to manipulate axis scaling from within the UI
- Use as much open-source software as possible
- Designing a well documented user interface "control panel" that is intuitive to all users
- Allow for the window to be resized without cluttering the interface
- Responsive control over the client's lab hardware, providing accurate feedback to the user.
    - Toggle control over which sensors are active, sending data to the program, and plotting the data in real-time
- Give users control of the data logging solution.
- Establishing a modular solution for the user interface's layout
    - Using this modular system to help design other more specific visual components of the interface (eg. alarm pop-up windows, submenus)
    - Ensure that the infrastructure for the UI is separate from the logic of the main program.
- Ensuring the application environment on the client's device is capable of operating with updated UI frameworks.
- The ability to interact with an alarm module, which will conduct several operations once a specified criteria has been met (eg. certain temperature has been reached within the chamber)
- Allowing for command-line use of the application

Data Logging

Next, outlining some of the different expected challenges with the Data Logging and Database Organization component of this project. These include, but are not limited to:

- Determining best option for data logging with client(show pros and cons of current network based system vs reliable standalone USB system)
- Understanding the full scope of all data being recorded and the method that the system currently uses to communicate with the current software.
    - If conversion from network to USB is needed, the team will need to fully understand how the data logger transfers information and its format
    - Establishing connectivity from the database to the new system in place, deep analysis of the old system is the first step to this process
    - Research for how an effective way to process and connect the database to the UI is needed
- Creating User friendly and accessible queries that satisfy the requirements put forth by the clients
    - Through communication with the client, the team will be able to determine specific queries that are are commonly formed to make an easily accessible function
    - Working cohesively with the UI/UX developer, integrating an editable query system that is usable from the interface
    - Having grad and lab students in the department use the system to provide feedback and for optimisation
- Determining and creating structure for a more effective logging system/optimizing current Data logging to process information how the client would like
- Modulating the Data Logging system to have the adaptability to add new hardware and effectively log its new data
- Effectively communicating data to plotting libraries for users(Combination of Data Logging and UI/UX)

Hardware connectivity and Network compatibility

This section provides an outline of anticipated challenges regarding the hardware connectivity and network compatibility component for this project. These include, but are not limited to:

- Connect laboratory hardware with main computers
  - The key component of this section is to create a kind of local network where the data can be shared with each other. So laboratory hardware and main computers can access data easily together and operate them by the local network
  - The application should also be prepared to handle multiple kinds of data input methods. There is already research being conducted for both local IPv4 network sockets, and USB serial data streams.
  - The network also should express emphasis on security and privacy, to avoid the case of data leakage
- The software should be able to handle abrupt disconnections from the hardware. Such as the case of power failure. The software should be capable of resetting data connections, referring to the data logs, and resuming data collection from where it left-off.

3. Technology Analysis

User Interface Design

For this project, the designing of a reliable, practical, and coherent user interface (UI) stands as one of the most important factors when it comes to the client's first impressions, as well as among the most important in determining whether the team can deem the solution a success in the long-term. Previously, this document has outlined several challenges that the team is anticipating during the design process within this project's UI component, many of which include cross-compatibility with other project components. This section will go into greater detail to further explain these design preferences and issues, and some ways which they can be overcome.

It would be optimal to have this new implementation written using an up-to-date version of a graphical user interface (GUI) toolkit as a base. In this case, the research conducted proves GTK3 is the best candidate, which is optimal for the clients Linux environment. At this point the team could then begin implementing core functionality that the client requires. The first item in the list would be real-time line-graph data modelling, by which the scale of the graph axes can be manipulated by users from within the UI. This, followed by a more reactive user interface system.

The UI of the alarm system should establish a good abstraction for different alert methods that are built-in to the Linux system. This would include the ability to, when alerting users that are on-site, have elements of the user interface change in color, text content or create custom pop-up windows, which will push critical information to the user. For users who are not on-site, this functionality would allow users to have alerts emailed based on the parameters of the alarm. Lastly, the alarm system interface should allow for a greater selection of alarm-triggering parameters. Triggers should include: chamber pressure changes, power outages, measurements reaching outside a set threshold, or a simple countdown timer.

Currently, many of these functionalities are written by hand using GTK2/Python library bindings. While these current implementations work for this software system, there are concerns about longevity and scalability. For designing the user interface as a whole, the team has identified a GTK+ interface designer called Glade. Glade is capable of providing real-time feedback of interface design, and provides methods of exposing UI control elements which can be captured and monitored by Python logic. The team has also identified some external libraries such as ‘matplotlib’, ‘bokeh’, and ‘plotly’ which would be used for abstracting, and perhaps increase the performance of, the real-time data modelling process.

Data Logging Design

The primary problem with the current system is the outdated methods and the problematic issue of finding eventful data that was recorded in the past. Traversing the current database and logging system is not quite efficient or user friendly. The solution proposed to the client was possibly establishing a database that logs data anomalies and reactions, and having the ability to “pin” these events so that they can be distinguishable for later analysis. This will aid in providing a traversable database designed for parsing specific events.

The desired characteristics of the product would be centered on a well organized database that can be edited to find anomalies in all tests taken in a certain time period. This can also be achieved through search queries that will be commonly used by the testers. Optimizing the database to act specifically with the needs of the lab would be the general goal. The further details and developments would come from communicating with the clients and testers. Interactions between the two parties have raised some key concepts like those mentioned above but through the development process, the team will be able to determine more specifics on how the clients would like the users to interact with the data logged.

Local development of the software is ideal. Because the overall goal of modularity and self containment, the system would ideally be run on the local machine and not need access to other sources. This method would also preserve the modularity aspect. The clients would have a complete rundown of the code and be able to make adjustments when implementing new hardware and sensors. That being said, data logging has been optimized over the years. For example the database language SQL, MySQL, SQLite, Maria DB and Oracle are all alternatives that work extremely efficiently with Python. Queries can be easily formed and edited to find data throughout the system. If SQL does not achieve the desired outcome, CSV is another data format scheme that is known to work well, and has already been implemented in the client's current system. If in the case all database systems fail to produce a viable result, that would leave the team to optimize the current plain-text output method. Dedicating research toward open source data logging systems could provide a better end product. Another alternative includes not simply using open source software but recreating it to have a mixture of optimized software while implementing modularity locally.

Hardware connectivity and Network compatibility Design

The first problems that need attention are the methods of connecting the lab equipment to the software. If the original software still works, that may provide a useful reference for designing the new implementation, though it may be out of date. The project requires that the system is able to extract data from the lab equipment such that the UI can render data results to the screen, while the data-handling component organizes and saves that data into a database structure. If the lab equipment were to run into a critical error, such as power failure, network outage, or some other external failure, recovering all the data by reloading the log files is a feature slated for implementation. After communicating with the clients and testers, the software should be able to record data in many situations. Spare equipment that has backup power may need to be put on the back burner so that it can record the current data into the log files created in the case the system is crashed. After the system recovers, the data will not be lost.

The next problem that needs attention is to connect laboratory hardware with main computers. Implementing a local network specifically for the laboratory that others cannot get access, and help to prevent data leakage. The laboratory equipment and the main computers can share data with each other. So the laboratory can get the same data analysis along with the main computers for better user communication. Cleaning up the client's home built hardware drivers and updating the log files will provide faster transfer speeds. Moreover, making sure the system is still fully functional when placed under new hardware environments, such as the case of hardware upgrades and replacements, is crucial.

Approach Analysis

Within this section, each characterized subsection will contain a score-chart, which will present research the team has conducted for discovering useful third-party tools. After the team took the time to understand the challenges, and grew better acquainted with multiple solutions to these challenges, the team has provided an analysis of which external solutions meet the highest expectations.

Real-time Data plotting

Here, **Table 1** visualizes utility rankings of different plotting frameworks for Python. While Bokeh is advertised as being capable of real-time data visualization, it appears that this library is best for web-application use cases. Similarly, even though Plotly has a Python distribution, it is still primarily designed for a web-app interface. The Matplotlib documentation, as well as community documentation have already proven that it is capable of embedding itself into a Gtk window through the Python/GTK3 API.

Table 1. Comparison of plotting frameworks*

Name	**Documentation**	**Compatibility**	**Stability**	**Overall Utility**
Matplotlib	**5**	**4**	**5**	**5**
Bokeh	**4**	**2**	**5**	**3**
Plotly	**5**	**2**	**5**	**3**
HandwrittenGTK	**5**	**5**	**2**	**2**

**Rankings are scored on a 1-5 scale (1 = Poor | 5 = Excellent).*

User Interface Development Platform

There is not much to choose from in this regard, QT and GTK reign as the dominant "scalable" options for user interface design on Linux. The only other option is Tk/Tcl, which is a Python-native library. While very versatile, Tk/Tcl does not tout the same support, development environments, and additional abstraction libraries that other frameworks (QT and GTK) have. So, Tk/Tcl is ruled out. Between the final two toolkits, the possibility of using a QT-based framework in the case of emergency shouldn't be ignored. However, because the client's computing environment is based on the Gtk framework, as well as having the original program written in an older version of GTK, it is more suitable to use GTK as the primary UI library. These rankings and assessments can be visualized below in **Table 2**.

Table 2. Comparison of UI rendering frameworks*

Name	**Documentation**	**Compatibility**	**Stability**	**Overall Utility**
GTK3	**5**	**5**	**5**	**5**
QT5	**5**	**4**	**5**	**3**
Tk/Tcl	**5**	**5**	**5**	**1**

**Rankings are scored on a 1-5 scale (1 = Poor | 5 = Excellent).*

Data Logging Solutions

As can be seen in **Table 3**, the top choices on technology implementation are dependent on capabilities, compatibility and access to the technology. SQL databases and its management system is simple in its core but very effective and works well with general Python programs. The queries that can be formed are the biggest advantage and it would be desirable to implement that into the UI, so the researchers can find what they need with ease. As such, this approach is ideal. Comma Separated Value (CSV) is what the system currently uses. This is reliable and simple to understand. The CSV system in place simply dumps data in a plain-text format that must be parsed by the user when searching for specific data points. Because of that, the solution when taking this approach would be optimizing and adding some functionality when the UI calls on its functions. This would include a conceptual update rather than a technical one.

Table 3. Comparison of database/data storage solutions*

Name	**Documentation**	**Compatibility**	**Stability**	**Overall Utility**
MySQL/ SQL	**5**	**5**	**5**	**5**
Maria DB	**5**	**4**	**4**	**3**
CSV	**5**	**5**	**2**	**2**

**Rankings are scored on a 1-5 scale (1 = Poor | 5 = Excellent).*

Hardware Connectivity Solutions

As can be seen in **Table 4**, compatibility and stability are the most important parts in terms of hardware connectivity. It's an unfortunate truth that hardware connectivity is a crucial bottleneck. The integrity of a data-stream depends heavily on the utilities in use to capture it. The ability to capture this data in real-time for visualization, and storage processing. Choosing solutions which have high performance in these two aspects is key. One of the options is "PySerial" which provides a backend for python and encapsulates the access for the serial port. It can run on Windows, Linux, OSD, BSD(possibly any POSIX compliant system). Socket is a python library currently used in the old system. It provides access to the BSD network socket interface and it is also available on all Unix systems, Windows, and MacOS. But the returned sockets are now non-inheritable. Cgi is a library in python which can provide common gateway interface support. It can call HTTP server to utilize the Cgi script so that it places all sorts of information about the request (such as the client's hostname, the requested URL, the query string and data to be processed) in the script's shell environment, executes the script, and sends the script's output back to the client. The script's input is also able to be connected to the client server as well.

Table 4. Comparison of networking and serial data-handling libraries

Name	**Documentation**	**Compatibility**	**Stability**	**Relevance**
PySerial	**4**	**5**	**5**	**5**
Socket	**5**	**5**	**3**	**5**
Cgi	**4**	**4**	**4**	**3**

**Rankings are scored on a 1-5 scale (1 = Poor | 5 = Excellent).*

4. Technology Integration

The three aspects mentioned above are the core aspects of the project. They all must cohesively communicate to ensure a smooth user experience and achieve the goals set out by the clients. Because of the complexity of the system, the process in which they are adapted to the system is essential. The current system uses one file to compile and run the entire system. This was done to self contain and have everything needed connected to a single source. Having a modular system will eliminate the single file architecture and establish a true project tree hierarchy to better organize the source code while keeping it clear and connected.

While the files will be separated, the goal is to continue to have all functions self contained within the system. The separated subfiles will make easily manageable UI, Data Logging and Hardware connectivity files that can be edited and configured when the lab would like a change in the future or if new equipment is being managed.

The first thing that needs to be done is to update the entire system as is into the new reconfigured Python language(Python 3.9). This will be done first and cloned to ensure that if new features somehow break the system, there are still reliable functioning backups in the updated language. From there, the team can begin work with functionality and layout design.

Through the development of the UI/UX system, communication with the client is essential. Here, the team will determine the crucial base-level functions that will be implemented or built-upon in the current system. Because data logging is a major aspect of the software refresh, both aspects of UI and data logging will have to operate in cohesion. This, in addition to further client discussion, will aid in determining how researchers would prefer interacting with the software, and obtaining expected data results.

Next comes optimizations to the hardware and software connectivity. There are some possible hardware changes that the client would like to implement, though it is expected that the interactions with the system will be the same. Because of this, the team will focus on the system aspect first, rather than the physical connectivity. Cleaning up the connectivity file on hand

(1408.py which enables hardware-software interaction) and updating the functions will provide faster delivery while leaving out other unneeded information.

The current file in the system works well and simply needs to be updated. This update will come first to ensure it functions with the new system. It will then be set in the background, where the team will focus on extending software functionalities. Optimizing the connectivity as well as the rest of the system will come after meeting the requirements and functions set forth by the clients.

The software developed during this project would be easily maintainable in the future and would ideally provide the comfort of longevity. Because of the modular aspect of the project, when making edits to the system, one would be able to continue doing research while functions are “under construction”. The goal of the final product is to provide the client “ease of mind” as a newly provided software implementation should be able to automate the client’s data-collection routines for research.

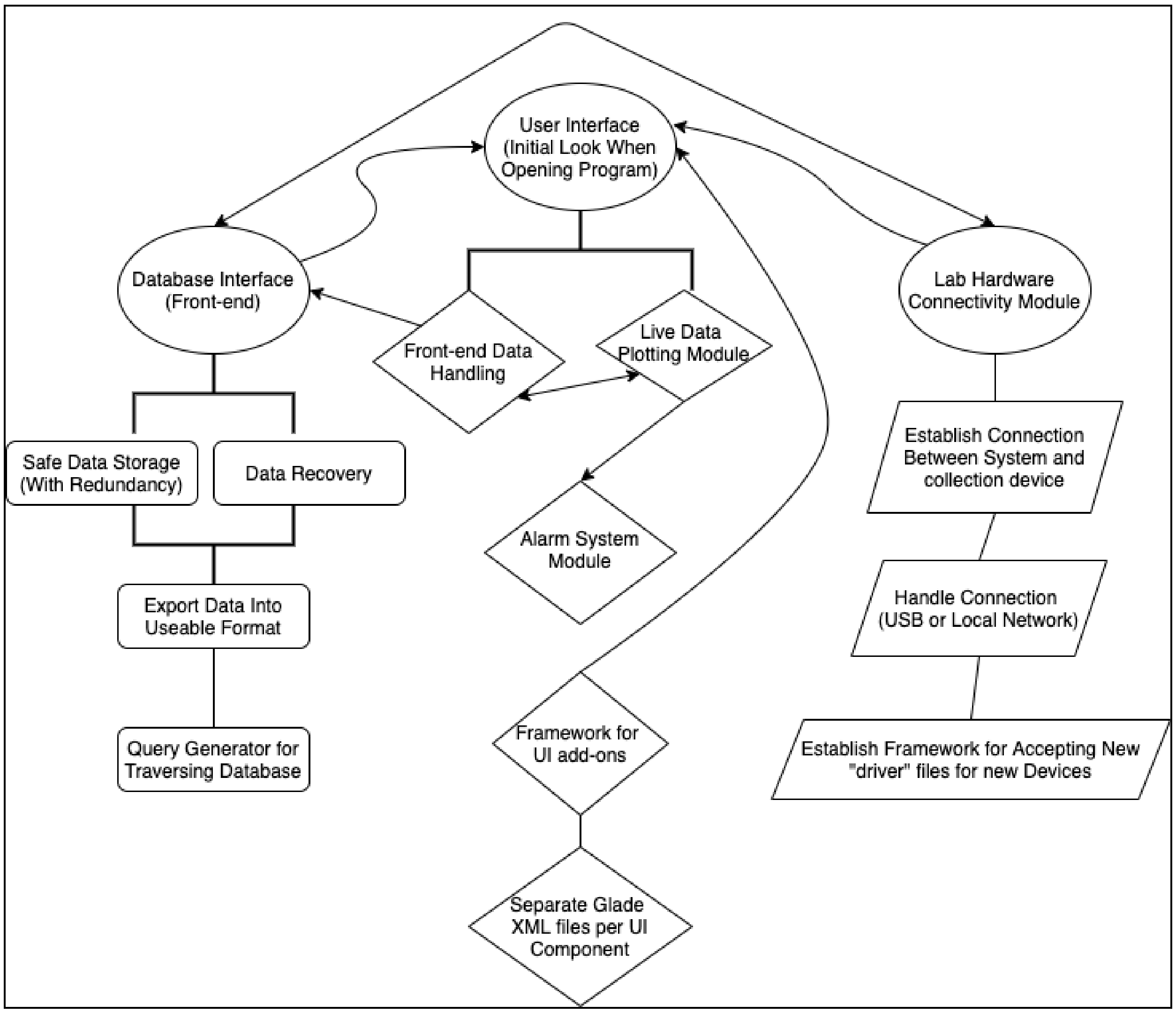
User Interface (Initial Look When Opening Program)
Database Interface (Front-end)
Front-end Data Handling
Live Data Plotting Module
Lab Hardware Connectivity Module
Safe Data Storage (With Redundancy)
Data Recovery
Alarm System Module
Establish Connection Between System and collection device
Export Data Into Useable Format
Handle Connection (USB or Local Network)
Framework for UI add-ons
Query Generator for Traversing Database
Establish Framework for Accepting New "driver" files for new Devices
Separate Glade XML files per UI Component

Figure 1 *Broad-scope layout of the newly implemented system. Features have been categorized into three macro-modules: User Interface, Data Management, and Hardware Connectivity. Diagram shows general features implemented in each module, as well as how each module interacts with one another.*

5. Conclusion

Updated libraries, system modularity and newly implemented functions are the plans set in place for the rest of the project. Firstly, updating the current system to the newest Python language version is vital, as the Python Software Foundation mandated that Python2 should reach its end of life. For the sake of security and maintainability, the Python team is recommending that software maintainers migrate to the newest Python3; this will be team Titan's duty to fulfill.

The modularity aspect will contribute to the longevity, organization and maintainability of the system. It will also provide the clients with, essentially, a well-documented API which they can use to establish their own modules for future changes or modifications. The new function implementations that are planned are focused around the data logging and display. Creating a professional experience while having organized and directed access to the database will contribute to the research being done.

The team has done some broad spectrum research on the topics at hand and have a general approach to how to develop the new system. The UI component will be done through extensive communication with the lab researchers as well as the clients to provide an interface that is easily traversable and self-documenting. In addition, operation of the data-logging component is coupled tightly with the UI/UX system. This is where most new features will be introduced, to create a traversable database that can be directed to specific points that will possibly contribute to the research.

As it stands, SQL is what the team is looking to implement, because of its efficiency with creating tables from databases while also being very compatible with the Python language. The methods used to collect data from different lab measurement devices are based on other Python scripts. These Python scripts could be better visualized as "drivers" or as an abstraction level which handles low-level commands to the hardware's API. As was determined by the team, these "driver scripts" are also to be considered "out of date". Providing updates to these modules is the first step. Later optimization will come as the team becomes more acquainted with the system structure, and as more research is conducted on external libraries and utilities.

Overall, the team has been working closely with the client in establishing the needs, the use case, and an understanding of their system, as well as the problems it was designed to solve. Major project topics—User Interface, Data Handling and Storage, Hardware Connectivity—were established and handed out to team members to begin the independent research process. All team members have identified, and introduced samples of different solution approaches outlined above. Though these steps are done in effort to solidify the solution methods, it is still valuable to keep the iterative process in mind, as the image of the final product could change radically as the project progresses. Team members are allowed, and are even encouraged to, make use of these alternatives at their disposal in order to implement, optimize, and achieve the desired results.