SpaceOps-2023, ID # 170

Telemetry Data Management System for Kuwait's First CubeSat: QMR-KWT

Nourah Alfialy[a, b]*, Kassem Saleh[a]

[a] *Department of Information Science, Kuwait University, P.O. Box 5969, Safat, Kuwait 13060,* nourah.alfialy@gmail.com
[b] *Orbital Space, 221 Al Maktoum Rd - Port Saeed, Dubai, UAE,* info@orbital-space.com
* Corresponding Author

Abstract

Satellite telemetry data is used to monitor the health of the satellites for the duration of their missions. Such data accumulates very quicky and over time turns into a massive volume of data. We report on the design of a telemetry data management system for Kuwait's first CubeSat named QMR-KWT which means "Moon of Kuwait" in Arabic.

QMR-KWT is an educational CubeSat launched into Sun Synchronous Orbit (SSO) at 525 km in June 2021. It operates in the armature UHF frequency band and its mission objective is to inspire students from the all over world to learn about satellite technology. The command-and-control ground station for QMR-KWT is located in Dubai, UAE where its telemetry data is received. Since it is an armature CubeSat, other ground stations around the world can also receive its telemetry data. An open-source network of satellite ground station known as SatNOGS can be used to gather more telemetry data.

We designed and tested a telemetry data management system that gathers raw satellite data from different ground stations, decode it, and then store the data in a structured database for subsequent data visualization and analysis. An open-source data visualization tool known as Grafana was used to create a dashboard to monitor the health of QMR-KWT cubesat.

The system's backend was written in Python programming language to read and decode the raw data received by the ground stations as the CubeSat passes over them. The processed data is then fed to a MySQL-based database. The system's frontend includes a graphical user interface to view and verify the data before it is stored in the database and a dashboard to display the data from the database in real-time.

Keywords: nanosatellites, ground-segment operations; decoder software, agile software development process, object-oriented design.

Acronyms/Abbreviations

Orbiting Picosatellite Automatic Launcher (OPAL), Space System Development Laboratory (SSDL), Continuous Waveform (CW), Onboard Computer (OBC), Ultra High Frequency (UHF), Sun Synchronous Orbit (SSO), Solar Panel (SP), Electric Power System (EPS), Attitude Determination & Control System (ADCS), Dual Tone Multi-Frequency (DTMF), Frequency Modulation (FM), Frequency Shift Keying (FSK), Gaussian Frequency Shift Keying (GFSK), Battery Charge Regulator (BCR), Low Earth Orbit (LEO), Object-Oriented Design (OOD).

1. Introduction

Kuwait's first CubeSat named QMR-KWT which means "Moon of Kuwait" in Arabic is an amateur CubeSat with an educational mission to allow students from all over the world to learn about satellite technology. QMR-KWT was successfully launched and deployed into a Sun Synchronous Orbit (SSO) at 525 km in summer of 2021. It was intended to serve as the platform for "Code in Space" initiative which gives the opportunity to students to develop and test their own software code on a satellite in orbit. This is a first of its kind initiative that

gives students a real hands-on learning opportunity to learn about the fundamental concepts in satellite operations [1].

QMR-KWT mission was designed planned in Kuwait, integrated by Orbital Space in collaboration with its partner in Bulgaria, and registered as a space object under the UAE. It is a standard 1U CubeSat bus with additional UHF transceiver and OBC as payload. The CubeSat command and control is operated by Orbital Space satellite ground station located in Dubai Silicon Oasis. The ground station is open to all students of all ages to engage and interact with the CubeSat. QMR-KWT achieved some of its mission objectives but unfortunately didn't achieve its main mission objective. For unknown reasons, the electrical power subsystem didn't generate enough power supply. The CubeSat operated in low power mode for about 8 months before it went silent [1]..

In general, after a CubeSat is launched to space, its bus OBC gathers telemetry data from the different subsystems and send it through the communications subsystem to its mission control on Earth. CubeSats usually transmit a beacon signal at regular intervals (every 1 minute or 5 minutes or another time interval) based on the design and mission of the CubeSat. Beacons usually contain the CubeSat identification details and data about its health which is known as housekeeping telemetry data. This signal is usually sent encoded, then it's decoded at the ground station using a decoder software [2].

The primary purpose of this project is to develop an operations software to manage the received raw telemetry data from QMR-KWT. The secondary purpose is to create a telemetry dashboard to analyse, visualize, and monitor data received from the CubeSat. The project objectives include:

1. Decode the CubeSat beacon signals
2. Create a user interface to view telemetry data
3. Create a static Data Structure (Database) to be used as a data source for the telemetry dashboard

Figure 1.1 - "Code in Space" satellite Ground Station opening in Dubai

Figure 1.2 - "Code in Space" satellite Ground Station in Dubai

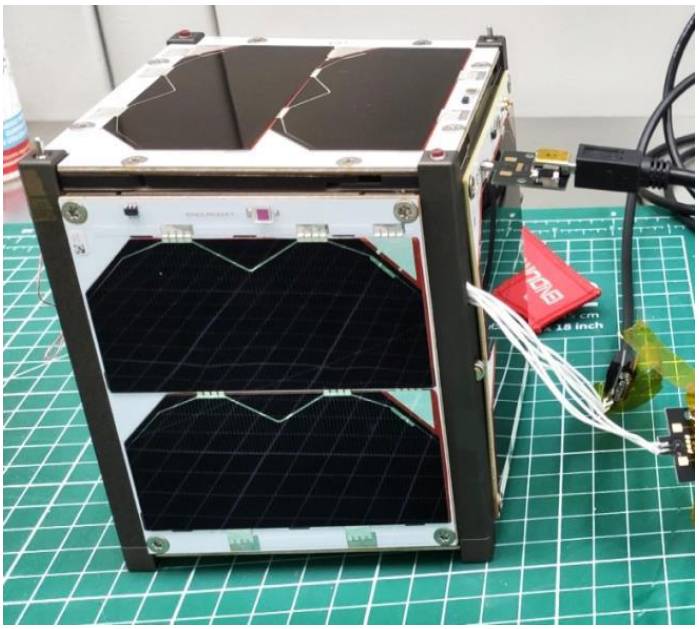

Figure 1.3 QMR-KWT CubeSat

2. Material and methods

Telemetry data which contains information about the spacecraft's health status consists of data gathered by various sensors over a time, such as temperature, voltage, currents, and so on. The data is transmitted from QMR-KWT through its communications subsystem which comprises of a triple redundant deployable circularly polarized omnidirectional UHF antenna and a UHF transceiver type II with a frequency range (Tx/Rx) of 430 to 440 MHz. On the other side (on Earth), the ground station tracks the CubeSat and receives

its signals. The essential and main functions performed and done at the ground station in support of an operational spacecraft include the following tasks:

- Data processing operations to deliver all engineering and scientific data in the formats necessary for the mission's success.
- Telemetry operations, which collect and store satellite data and health status.
- Satellite tracking to identify the satellite's orbital position.
- Commanding operations to interrogate and control the satellite's numerous functionalities.
- Controlling operations and loading the OBC, as well as determining orbital parameters and scheduling all satellite passes.
- Data connections with other worldwide ground stations.

The Orbital Space ground station consists of UHF antenna, a UHF transceiver type II, data recorder, control consoles, and a computer. When the CubeSat passes over the ground station, data from the housekeeping telemetry frames are received. The data is a mix of status flags, such as subsystem on/off, and engineering parameter values. Extraction of housekeeping telemetry data for quality control and health assessment, data processing, orbit determination, and data analysis are among the immediate tasks after the CubeSat pass is over. The data are then passed to QMR-KWT dashboard for further detailed analysis and visualization which includes graphics facilities tools.

The sssumptions and methodology followed in designing and developing this project are first the CubeSat will send a packet every minute so the system should be able to manage a file that contains all the data sent during the day. The signals are assumed to be encoded first using an encoding process so signals received in the local ground station entered into a computer through the sound card as shown in figure 2.1. Then, a sound decoder program is used to convert the sound signals to text so the parser code can decode this text by converting the raw hex AX.25 packets to ASCII, parse and save the output data into a text file (logs). This log file can be then imported into QMR-KWT ground software, where the software can read this file, search for decoded beacons, display all valid beacons in the screen, and then the user can request to save this output values into a database. The user can also request to download all the observations of QMR-KWT CubeSat from SatNOGS network. This request to download is received as an Excel file of type CSV file (comma-separated values) which is a text file that has a specific format that allows data to be saved in a table structured format. The user can also import this Excel file (CSV) that is being downloaded from SatNOGS network into the software, read this file line by line to search for valid decoded beacons, convert raw hex AX.25 packets to ASCII, parse them into a readable text format using the parser code attached to the software. Displays all the decoded beacons to the user and save them it into a database. This software was developed using Python programming language and had been connected to MySQL database which is an open source relational database management system. QMR-KWT telemetry dashboard created using Grafana tool which allows user to query, visualize, and analyze metrics and logs.

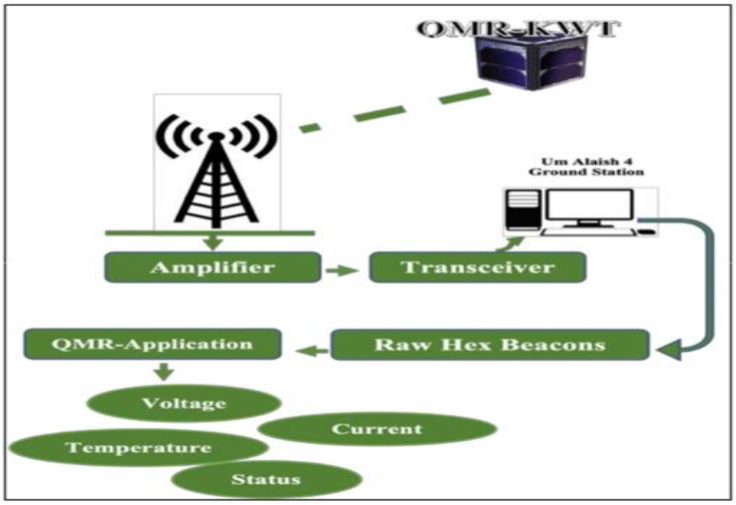

Figure 2.1 Flow Chart of the Ground Station System

2.1 Software Development Methodology

Software development methodology is a framework for structuring, planning, and controlling the development of an information system in software engineering. This project followed the Agile software development process, so whole software was implemented by the plan of Agile development. The software was broken into a small units so each unit was built and tested until it is completed. Then another unit is developed, tested, and merged with the previous ones, and so on. This technique enables us a developer to create a functional software in a short time and allow us add additional features to it in the future. Furthermore, breaking the system into small units reduces the development difficulties.

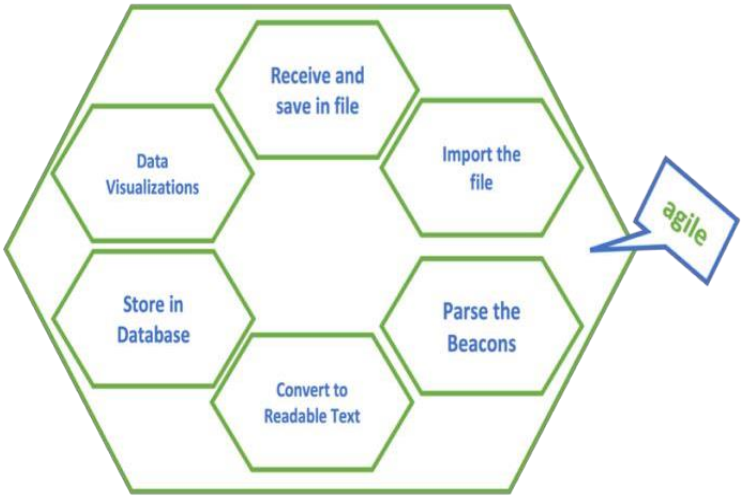

Figure 2.2 Software Development Methodology used

2.2 Software Requirements

The requirements of the software development were derived directly from the project objectives. The following are the functional requirements:

- The raw hex AX.25 packets are previously saved in a CSV or text file. Then, the software is to read this file and decode data to its corresponding readable text values using the parser code.
- The parser program already evaluates whether the data is valid or not. So, in case that there is no valid beacons data inside the imported file, the program exits with no output.
- After decoding and processing of raw telemetry data they are to be displayed to the user though Graphical User Interface (GUI).
- If data received during one or more days were saved in one text file, the software is to be able to read this file with a good performance and without unacceptable delays.
- Data must be saved into a Database so that user can view it at anytime.
- For the “Save” option, sometimes the user may want not to save the output data values into the database. The software is to allow the user to do this.
- Voltages, currents, power, temperatures and other telemetry data are to be displayed using visualization tools such as Grafana.
- The total number of beacons data in the imported file should be displayed to the user.
- If the information indicated that the CubeSat may be in danger, the dashboard is to alarm the user.
- It would be good if all the visualizations were displayed in the same page. Other times it may be better to view each one them in separate page.
- The colors used for the software and dashboard design must be comfortable and attractive to the user.
- To achieve high reliability, all bugs that may influence the code safety and issues with the software components should be eliminated.

3. System Design and Testing

The process of defining and evaluating the architecture and functionality of the system to ensure that it meets the specified requirements is outlined in this section.

3.1 Design

The design of the software was done according to the project objectives. The method followed in the design of the software was Object-Oriented Design (OOD). The problem is divided into classes and objects of these classes are used. In addition, a user-friendly design of the graphical user interface was implemented to the software.

3.2 Software Coding

The code of the software was developed based on the design mentioned above. Python programming language was used for coding. The reasons for choosing Python are:

- It is an Object oriented Programming Language.
- It is a language that can deal with strings. Since the data to be parsed are in text format, Python was the best choice.
- Programs written in Python can be run under any operating system.
- PyCharm is used and it's considered integrated development environment (IDE) used in computer programming, specifically for the Python programming language.

3.3 System Testing

The following levels of testing were done to the developed software:

3.3.1 Unit testing

In unit testing, each unit must be tested independently. The main purpose of this test is to make sure that there are no syntax errors or logical errors. Since the software units development were done using agile, this test must be done regularly in unit by unit. The following functions were tested extensively.

- Import Function

  This function must pass two tests:
  1. If a non text file was imported.
  2. If the selected file contains data that doesn't belong to QMR-KWT.

- Save Function

  This function must pass the following tests:
  1. If the database is not found.
  2. If the database cannot be opened.
  3. If in a "Save" option, a file that is previously saved was specified must not be allowed to re-save it again to avoid duplicate data.

3.3.2 System Testing

System testing is done when the development of the software was completed. This testing was done as a black box which must be provided with inputs and brings out the outputs.

4. Results

4.1 Developed Software

The software developed has a GUI which consists of four main buttons, “Start Import” button in the welcome screen. The other button such as “Import”, “Next”, and “Save” are in the main screen. Text boxes are also used to display the output values.

4.1.1 How to use the Software

When the user starts executing the software by clicking on the QMR-KWT icon in the desktop, the welcome screen is displayed as shown in Figure 4.1. The name of the software, logo, and the “Start Import” button will be displayed to the user.

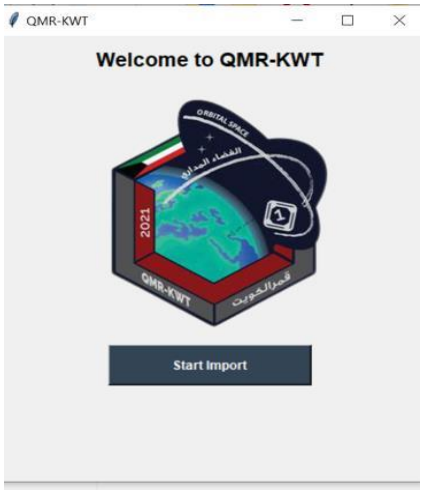

Figure 4.1 shows the welcome screen of the Software

In case the user decides to click on the “Start Import” button, the main screen will be displayed to as shown figure 4.2.

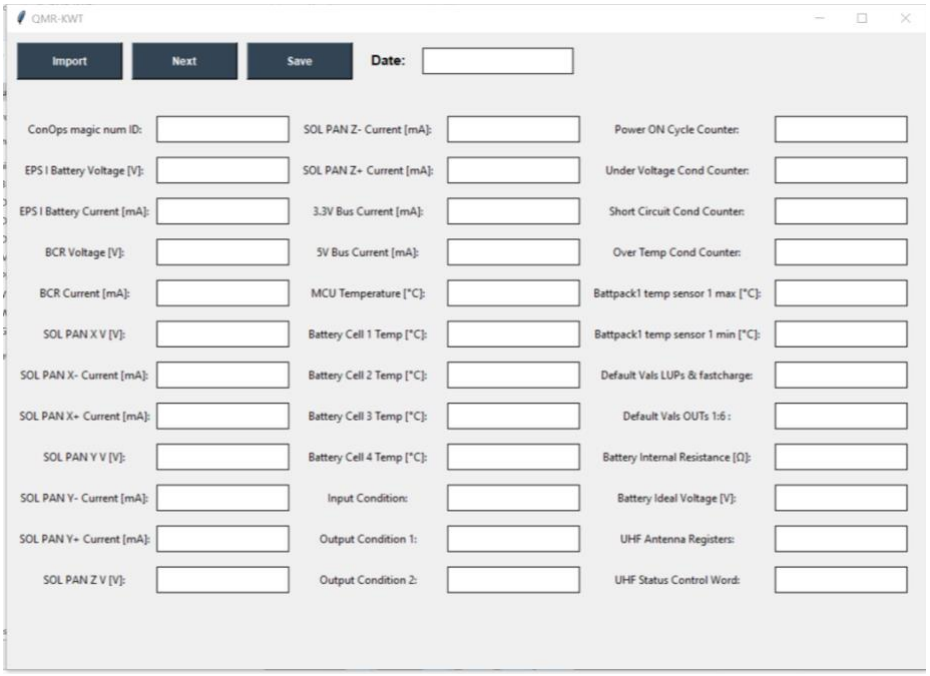

Figure 4.2 shows the main screen of the Software

The main screen contains the following GUI:

- "Import" button, where the user can start import files.
- "Next" button, this button is used when the imported file contains more than one beacon, so in order to move from one output to another and display all the beacons data in this file the user must click on this button until he reaches the last one in the file.
- The total number of beacons in the imported file and the current beacon order number are displayed in the top of this screen as a text.
- "Save" button, the user has the option to store and save the output values into the local database by clicking on this button.
- "X" button, in the upper right corner is used to close the current opened screen.
- Text boxes are used to display the output values in the main screen with a description for each text box.
- The main software function can be started when the user clicks on the "Import" button which displays a file chooser dialogue box as shown in figure 4.3. At this point, the user can choose and select the file name to be imported and then click on "Open" button file chooser dialogue box. In this step, an import function is called first. Then, it calls then the choose file function which returns the directory of the file.

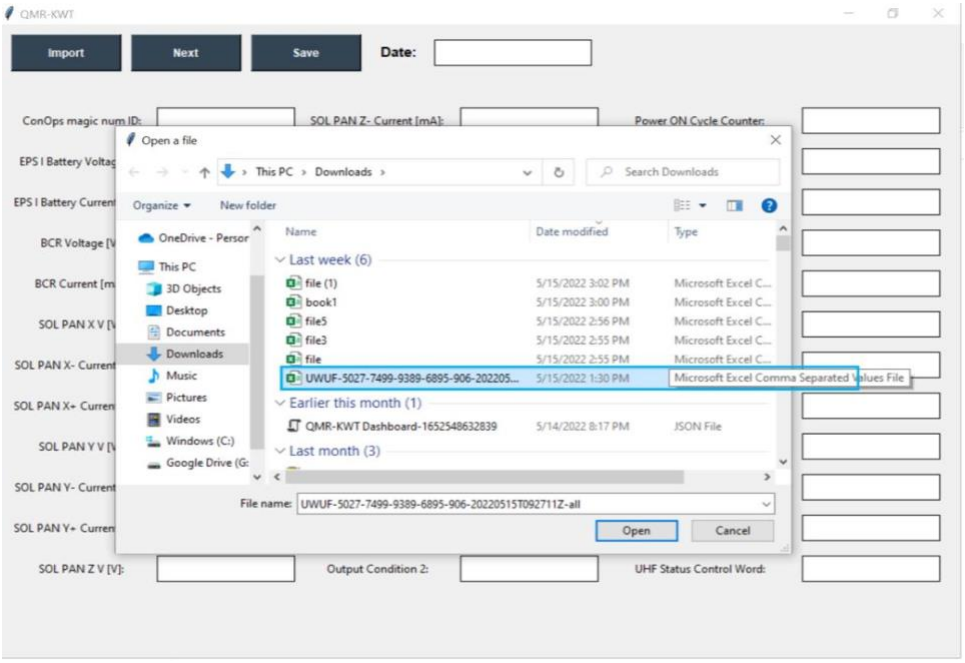

Figure 4.3 displays a file chooser dialogue box

- After the selected file is being imported, the software will read this file line by line to search for beacons no matter if it is already decoded or not. In case the imported file contains raw hex AX.25 packets, these data will be parsed first and decoded to a readable text format. Then, the output values will be displayed to the user on the same screen as shown in figure 4.4.

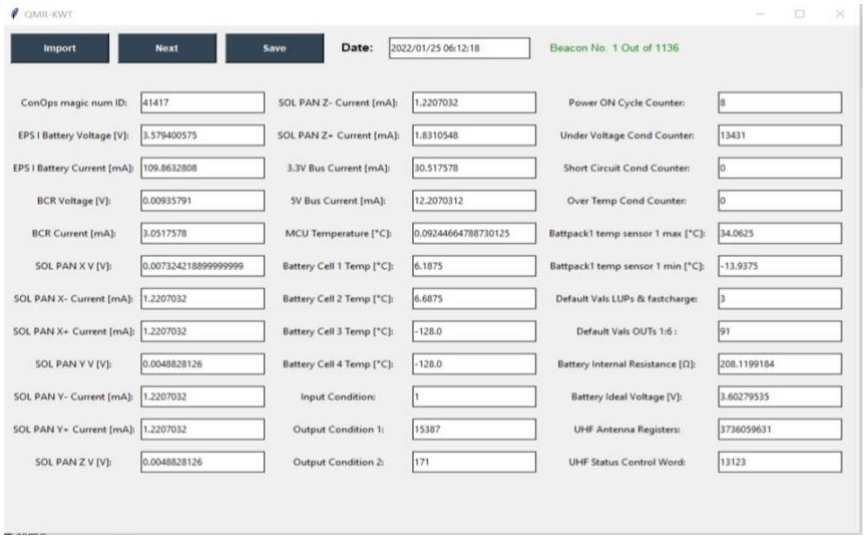

Figure 4.4 display Output Values in the main screen

If the file contains more than one beacon the user can click on the "Next" button to move to the next one. Finally, the user can choose to save the output values from the file into the local database by clicking on the "Save" button. A confirmation message is being displayed as a message box to the user to confirm that the requested save action has been successfully completed as shown in figure 4.5.

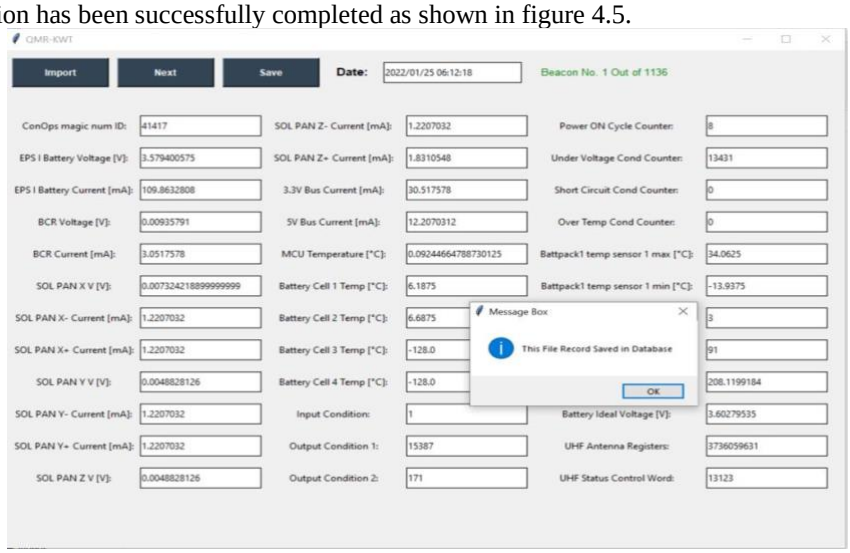

Figure 4.5 File Saved in the Database with a confirmation message

4.1.2 Exceptions and Error Handling

This is an important part that must be carefully handled when developing any software project. In case these actions are not well handled, the system becomes exposed to failure and thus the performance of the system degrades. The system can handle the following actions:

- If the user tried to save before importing any file, the system will ask the user to import a file first.
- If the user tried to save file that is being previously saved into the database, the system will not allow to re-save it and display an error message saying that the is file record already saved in database as shown in figure 4.6.

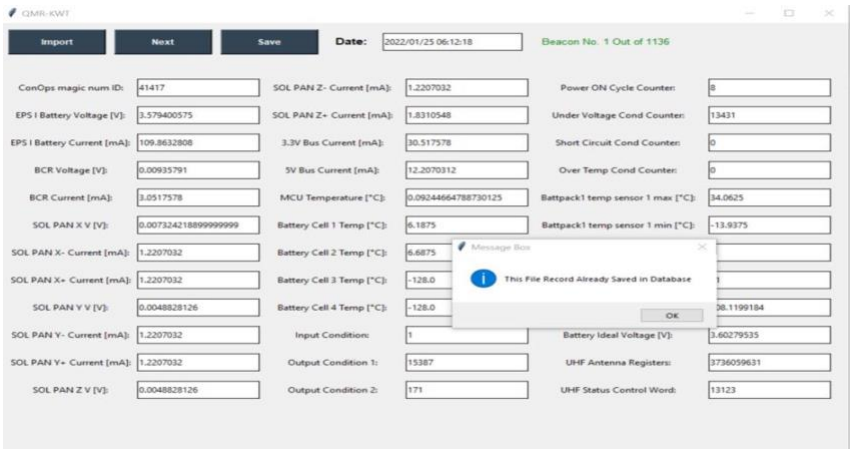

Figure 4.6 An error message is displayed when requesting to save records that is already saved in the database

- If the user selected a text file containing data that doesn't belong to the QMR-KWT CubeSat, the software will display error message saying that this file is incorrect.
- If the user clicks on the "Next" button and the last decoded beacon in the file is already displayed, an info message will be displayed to inform the user that there are no more decoded phoenix beacons in this file as shown in figure 4.7.

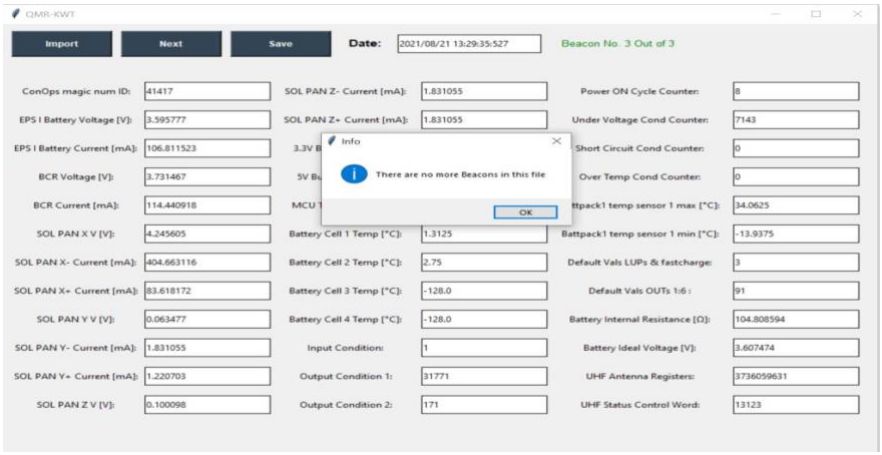

Figure 4.7 Display an info message that there are no more beacons to display

In table 4.1, all the tests performed, and their related actions are shown to ensure that all the actions are well handled to avoid system failures.

	Test	Action
1	Imported file with no valid beacons	Info message displayed as in Figure 4.7
2	If the string passed to the parser it contains undefined symbols or not complete	Skip this line and move to the next one
3	If saving a file record that is already exists in the database	Error message was displayed as shown in Figure 4.6
4	The database file cannot be opened	Error message was displayed.

Table 4-1 Exceptions and Error Handling test table

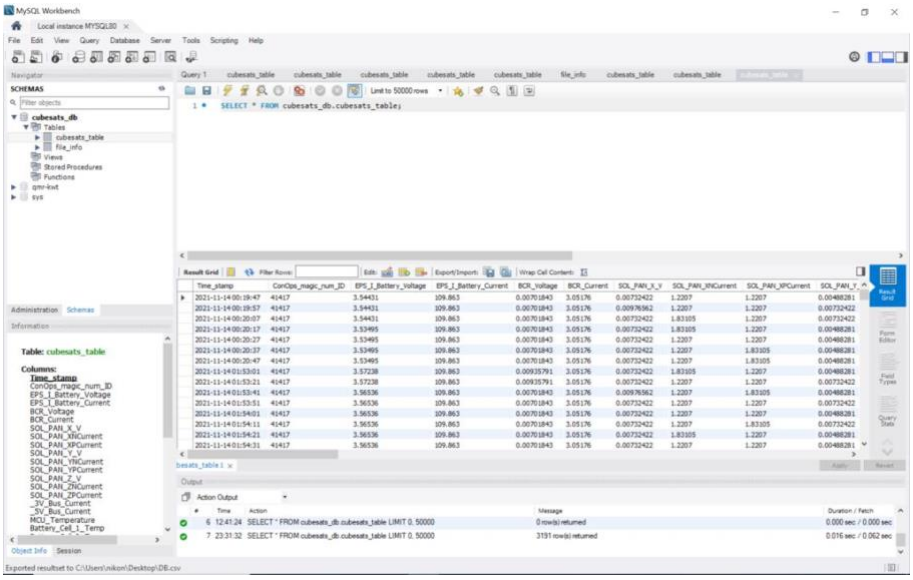

Figure 4.8 Data Saved into cubesats_table

4.1.3 Data Visualization

To facilitate monitoring of CubeSat's status and health condition while the satellite in orbit, it is desirable to compare telemetry data over specific time range and search through previously received data. Grafana was used to create telemetry dashboard for QMR-KWT CubeSat using the output telemetry data already stored in the database through the software developed. This database is connected to dashboard as a data source. Grafana is a visualization and analytics software that is free and open source. It allows us to query, display, alert on, and analyze our metrics, logs, and data from any supported data source. It gives us the tools we need to turn our data from a database into useful graphs and visualizations. Therefore, Grafana was considered a good choice as an analytics and interactive visualization web application. It provides charts, graphs, and alerts when connected to the data source such as MySQL database in our case. Figure 4.9 shows how Grafana tool is used to create the QMR-KWT dashboard [3].

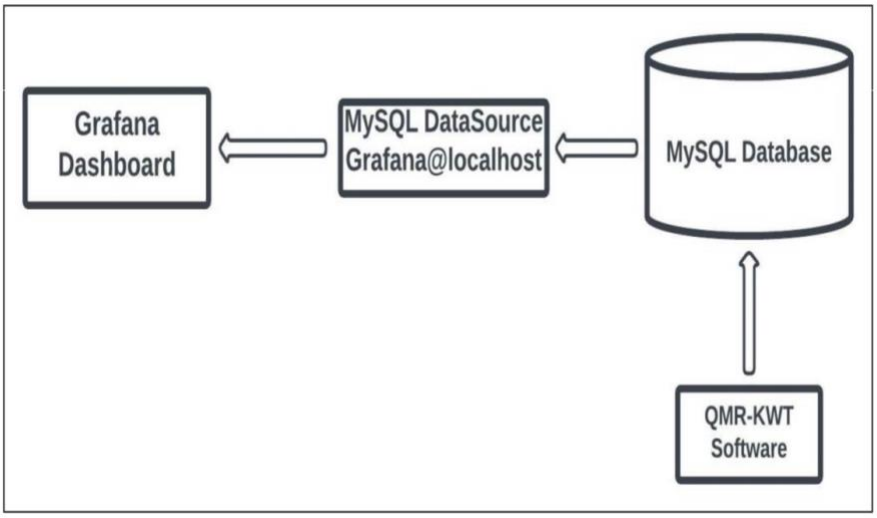

Figure 4.9 Creating Telemetry Dashboard Using Grafana and MySQL Database

Grafana dashboard, shown in Figure 4.10, has several graphs with one or more data series. The user can alter the time range, save the graphs, and export the data to a CSV file and other formats from the upper right corner of Grafana dashboard screen.

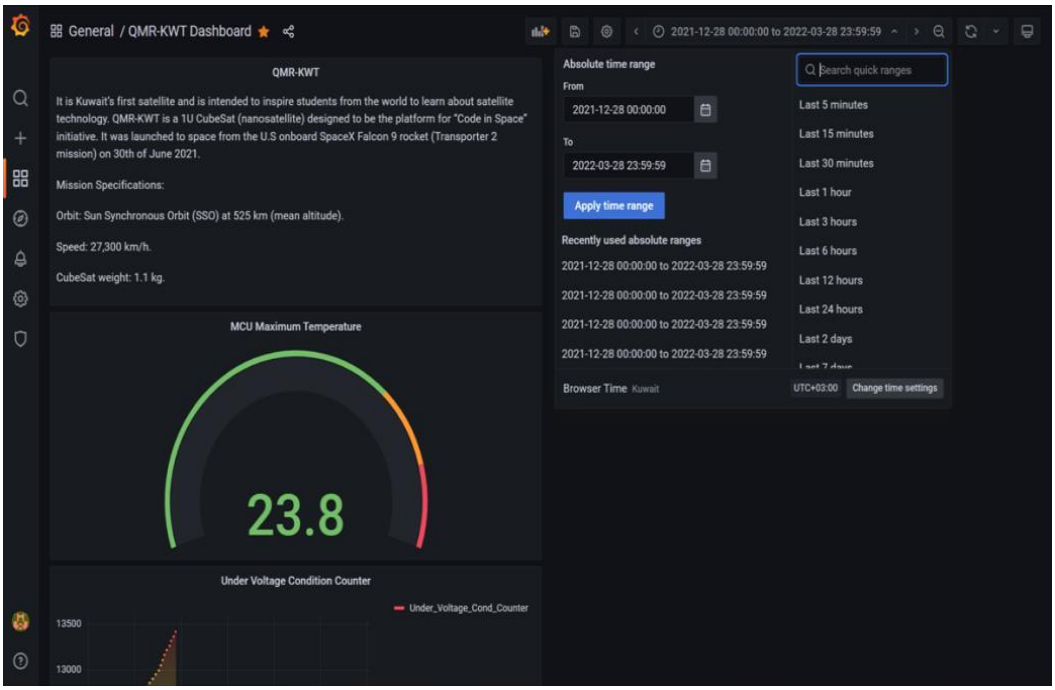

Figure 4.10 Grafana Dashboard

When the user hovers the mouse over one of the plots, the multiple-cursor- label feature displays, providing data labels for all series on the screen at the timestamp corresponding to the cursor position on one graph as shown in figure 4.11 for the EPS Battery Voltage Time Series Graph and figure 4.12 Battery Resistance Chart. This is a very useful feature because it allows the user to compare and reference several data points at the same timestamp.

Figure 4.11 EPS Battery Voltage Time Series Graph

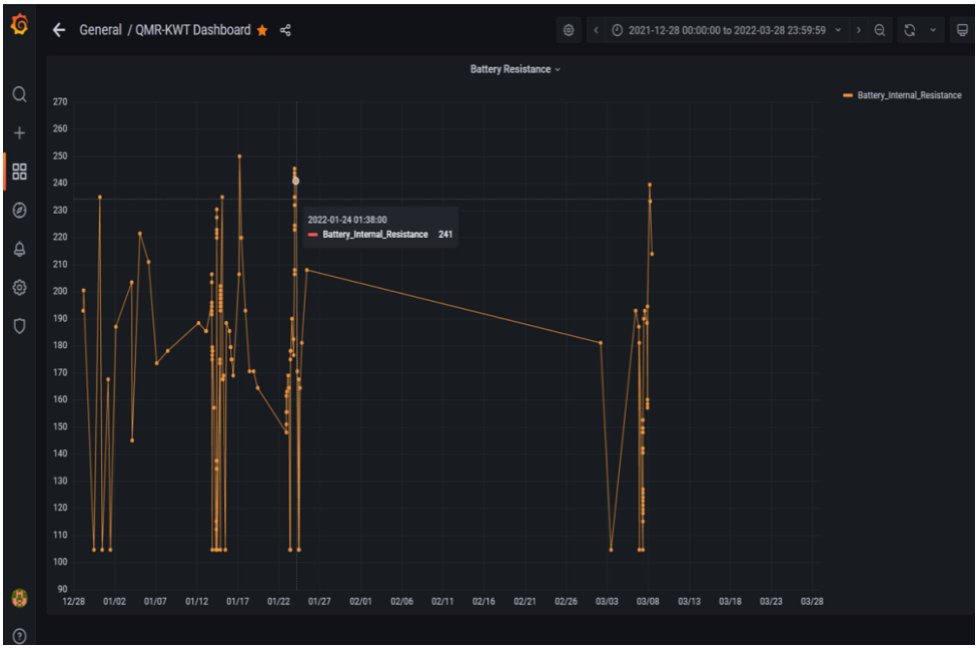

Figure 4.12 Battery Resistance Chart

Figure 4.13 below show BCR Current and Voltage as a table format.

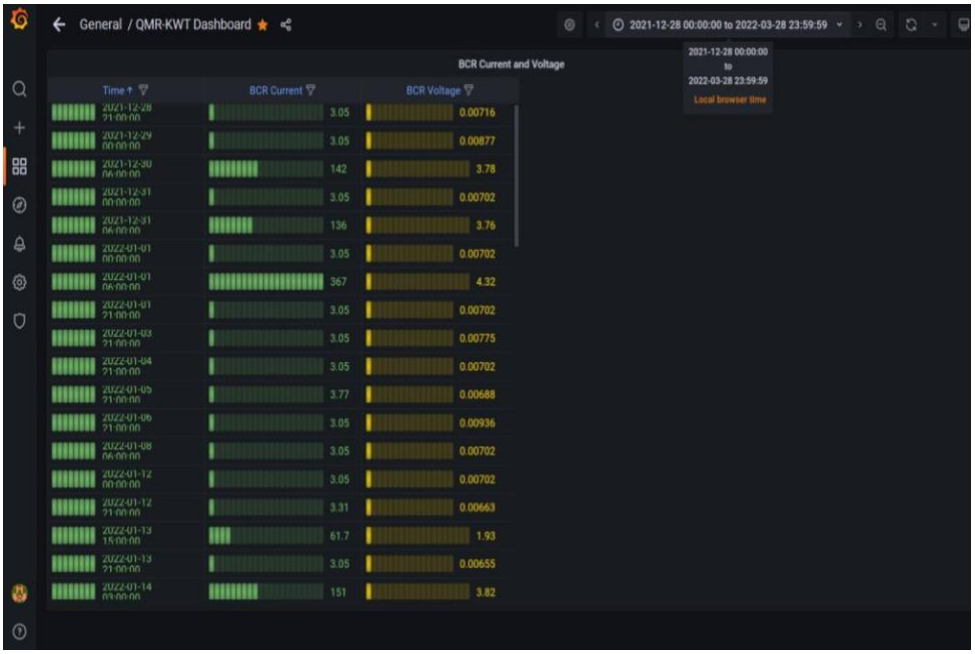

Figure 4.13 BCR Current and Voltage Table

Grafana's time-series graph is the default and main Graph visualization. It could establish alerts as shown in figure 4.14 and 4.15. The limit condition can be specified in a variety of ways, including minimum, maximum, sum, count, last, median, and difference. It may make sure that the telemetry data is above, below, or within a range for each option. It also includes the ability to deliver alerts only once the alert has been active for a certain amount of time.

Figure 4.14 MCU Temperature with Alert displayed in red color

Figure 4.15 BCR Voltage with Alert displayed in red color

5. Discussion

From the telemetry dashboard screen shots mentioned above, we can notice that in the testing we had specified a time range from 28/12/2021 to the midnight of 28/3/2022. Mostly time-series visualizations were used for this dashboard. For example, in the MCU Temperature time series visualization we can notice that the peak value was is 13/1/2022 and it is displayed in red.

In general, this project can be run on any computer, and it is simple to use. The performance of this project is great. Since, this software works with the imported file and goes through it line by line to read the decoded beacon by beacon or packet by packet, it doesn't matter if the file contains 5 or more than 5000 lines. It matters only when visualizations such as graphs are to be displayed, the user have to set the time range in the dashboard.

The software was tested with a file containing more than 5000 AX.25 packets which is more than the expected packets per day. It gave satisfactory performance and no unacceptable delay was recorded. The tasking process begins when a user request to start using the software and click on the “Start Import” Button.

When the parsing code is called, the code performs pattern matching, essentially looking for a couple of specific bytes that indicate that there is indeed a valid beacon inside the file. So, when the program exits with no output is that there is simply no valid beacon data inside the imported file. It means that the parser program evaluates whether the data is valid. In any case the beacon data is in raw bytes format. The byte data is not UTF-8 or ASCII encoded, so whether the user was passing a .txt file or a .bin file does not really matter**.**

The Graphical User Interface was designed and built using Tkinter package, which was a good choice for dealing with GUI. Tkinter isn't the only Python GUI programming toolkit, however, it is the most widely used package.

6. Conclusion

This work presents a real case of a satellite, QMR-KWT CubeSat, which is already designed, integrated, and successfully launched into space. QMR-KWT provided several students with knowledge and capabilities about CubeSat projects.

In this project, a complete study and analysis of QMR-KWT CubeSat’s telemetry data was done and as a result a complete software with a telemetry dashboard were designed and developed.

On the back end of this project, an interface between the database and the software was developed. On the front end, a Graphical User Interface (GUI) was developed to view and modify data in the database and Grafana was set up to visualize telemetry data.

References

1. *Orbital Space, "QMR-KWT", Orbital Space, n.d.,https://www.orbital- space.com/qmrkwt.*
2. *Noe, C., "Design and Implementation of the Communications Subsystem for the Cal Poly CP2 Cubesat Project", in Computer Engineering Department. 2004, California Polytechnic State University: San Luis Obispo.*
3. *Grafana Labs, Grafana Documentation. Available at https://grafana.com/docs/grafana/latest/.*