Cairo University

Faculty of Computers and Information

Graduation Project 2017/2018

"System to decrypt and mine the telemetry data of satellites"

Supervised by: Prof. Ali Fahmy

Prof. Abou ella Hasanien

Prepare by:

Nourhan Mahmoud Mohamed	20140298	CS-IS
Ahmed Hesham	20140048	IT-IS
Ahmed Mostafa	20140045	IT-IS

Table of Contents

LIST OF ABBREVIATIONS

OOP	Object oriented programming
ERD	Entity Relationship diagram
UI	User interface
IDE	Integrated development environment
ADCS	The Attitude determination and control System
OBC	On Board Computer System
PS	Power system
CDHS	Command and Data Handling Subsystem
APID	Application Process Identifier
DTMF	Dual Tone Multi-Frequency

DB	Database
CS	Communication System
SVM	Support vector machine

LIST OF FIGURES

Figure

Chapter 1: Introduction

1.1 introduction to the main area of the project

Space industry is our domain as its one of the most important industries in the modern age and used to measure the advancement of nations in the world. Egypt will launch the first satellite created and manufactured by Egyptian hands.
In this project, a computer program will be developed to decrypt the data sent from the satellite to the ground unit.
This program will unpack the decrypted telemetry packets sent by the satellite in each session and show the data sent in readable form to the responsible engineers.
The Graphical User Interface will show these sensor reads and make some charts and graphs that demonstrate the health of the satellite through the sensors

1.2 Motivation

Satellites have allows us to see large areas of Earth at one time. This ability means satellites can collect more data, more quickly, than instruments on the ground.

Satellites also can see into space better than telescopes at Earth's surface. That's because satellites fly above the clouds, dust and molecules in the atmosphere that can block the view from ground level.

Before satellites, TV signals didn't go very far. TV signals only travel in straight lines. So they would quickly trail off into space instead of following Earth's curve. Sometimes mountains or tall buildings would block them. Phone calls to faraway places were also a problem. Setting up telephone wires over long distances or underwater is difficult and costs a lot.

But there are some restrictions in the main operation of the satellite one of these restrictions is the communication between the satellite and the ground station as shown in fig.1.

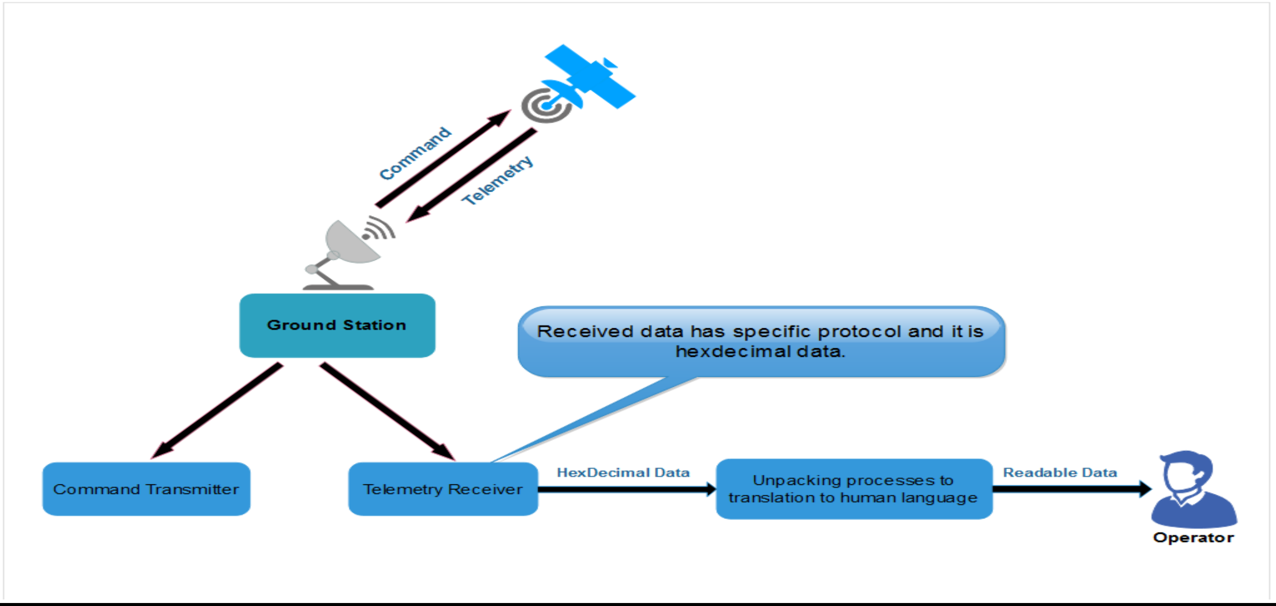

Command
Telemetry
Ground Station
Received data has specific protocol and it is hexdecimal data.
Command Transmitter
Telemetry Receiver
HexDecimal Data
Unpacking processes to translation to human language
Readable Data
Operator

Fig. 1 Satellite and ground station communication problem

To solve this problem we need to create a program that will decrypt the telemetry packets sent from the satellite to the ground unit. This program will unpack the telemetry packets sent by the satellite in each session and illustrate the data sent in readable and understandable way to the operator

With satellites, TV signals and phone calls are sent upward to a satellite. Then, almost instantly, the satellite can send them back down to different locations on Earth.

1.3 Problem definition

The main problem is the unpacking and the understanding of the telemetry data sent by the satellite and processing it into readable and understandable information.
The satellite contacts the ground station on average 2 times a day. On contact the satellite send enormous amount of packets which represent a huge amount of data which is impossible to be monitored by eye the led to the need of processing and accurate unpacking and storing in the database and fast retrieving too.

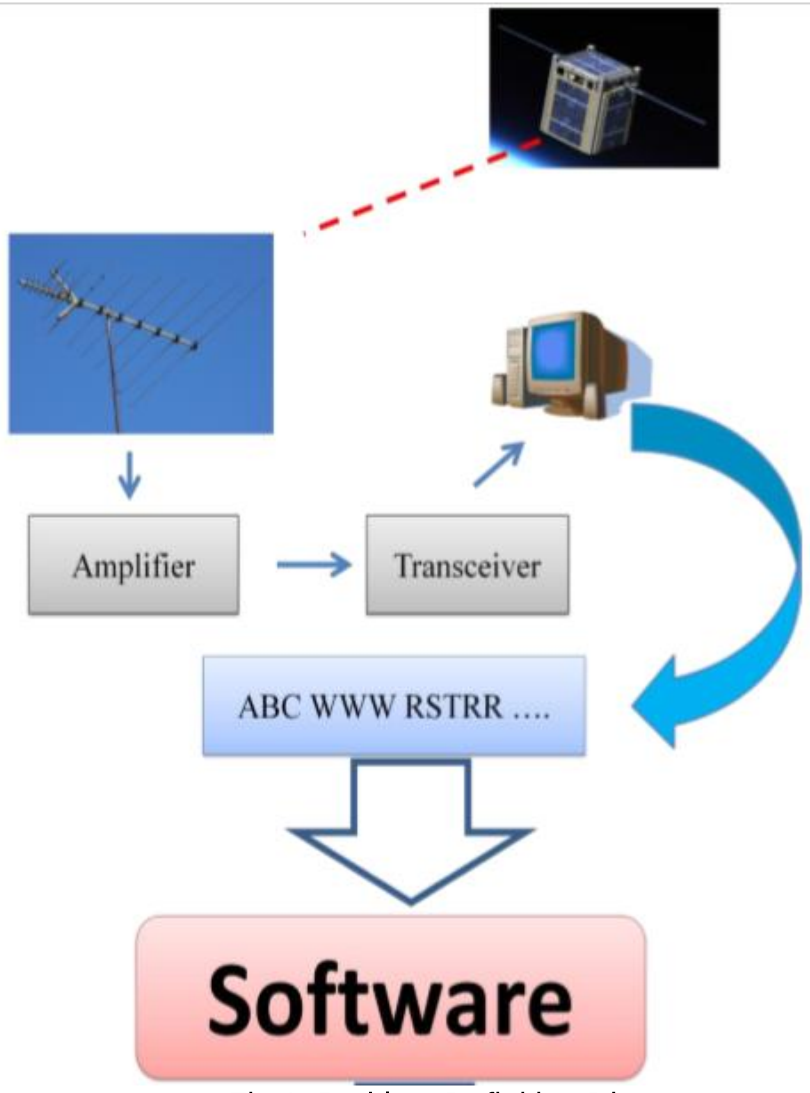


Fig.1. Problem Definition Diagram

1.3.2 Satellite Architecture

A satellite consists of a payload, which is the mission-specific equipment, and a collection of subsystems is called bus. Satellite bus is a group of components that support a common function. There is a difference between the payload and the rest of the satellite bus, because the payload is typically unique for a given mission, whereas the bus may be able to support different missions.
A bus typically consists of Power Subsystem (PS), Communication Subsystem (CS), Attitude Determination and Control Subsystem (ADCS), Thermal Control Subsystem, Structure Subsystem, Command and Data Handling Subsystem (CDHS), Telemetry, Tracking and Command Subsystem (TT&C).

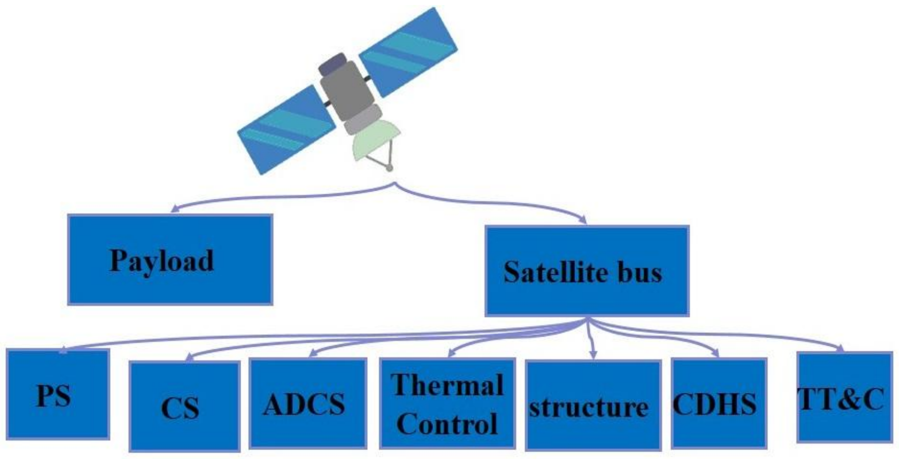


Fig. 2. Satellite Architecture

1. The power system is responsible for the management of the power generated by the solar cells and stored in the batteries in the satellite to certain levels to maintain the availability of power when the satellite needs it.

2. The communication system is responsible for the communication with the ground station on earth.

3. The Attitude determination and control System (ADCS) is one of the most important systems in the satellite as it responsible for many tasks:
    3.1. In the first launch it stabilizes the satellite by damping the angular velocity and initializing the construction of satellite attitude.

    3.2. Changing the attitude and orientation needed to capture image.

    3.3. Keeps the orientation to stay in touch with the command with low accuracy and low power consumption.

    3.4. The execution of the ground station order of the desired attitude and orientation. The determination of the position of the satellite is done in the ADCS system using Magnetometer, Sun Sensor and Star Sensor. In addition to this the changing in the attitude or the orientation of the satellite is done in the ADCS by the Magneto Torque (MT) and The Reaction Wheel. There several operational modes of this system which are IAA (DE tumble mode) (Damping mode), SB (Stand By mode), PTM (Imaging mode), HAAC and EM (Emergency mode).

4. The Telemetry system is responsible for combining all the reads of all the system into the form of the packet sent from the satellite to the ground station.

5. The Structure system is responsible for measuring the health of the components of the satellite.

6. The On Board Computer System (OBC) is the brain of the satellite which manages all the systems above and it controls the satellite by the orders sent from the ground station.

7. The propulsion system is responsible for the control of the satellite by the external thrust generated for the satellite movement to change speed or to maintain certain condition.

1.3.3 Telemetry, Tracking and Command (TT&C) Subsystem

A communication between the satellite and the ground station is needed. The primary goal of it is to provide a link to transfer the health status information to the ground station then returns back to the satellite with a proper command or task from ground station. These functions are performed by Telemetry, Tracking and Control (TT&C), which is the brain subsystem of satellite provides a connection between the satellite itself and the facilities on the ground. The TT&C subsystem is required for all satellites regardless of the application to ensure the satellite performs correctly.

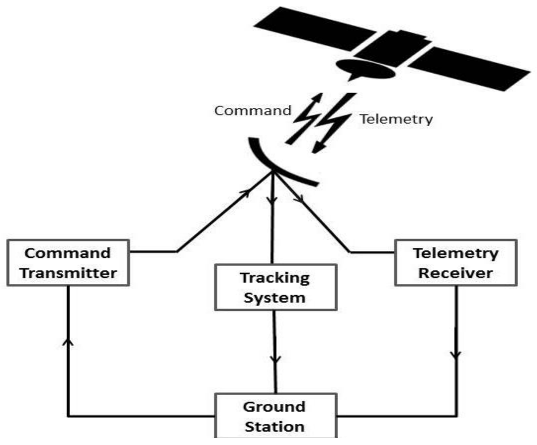


Fig. 3. Main Functions of Telemetry, Tracking and Control Subsystem

1.3.4 Telemetry Data

During the artificial satellite operational lifetime the ground station is receiving the telemetry data, is non-stationary time series dataset contains thousands of sensor measurements from various subsystems, which contains the wealth information related to the health and status of the entire satellite and all its subsystems which reflect the operational status and payload of satellites. The health and status measurements of the satellite include the status of resources, the health and mode of operation for each subsystem and environmental data like values of sun and radiation or like star trackers. The telemetry data is analyzed in the ground control station for the health monitoring purposes such as failure diagnostic or prognostic, and anomaly detection.

There are two types of telemetry data:

- Housekeeping Data: Instrument health, safely and state parameters, are transmitted to ground in periodic telemetry data.
- The data for which the satellite was launched, such as photographs of some of the areas that were claimed, mineral, etc.

1.3.5 Ground Station:

The earth station is the ground part of the satellite system. Its main functions are transmitting and receiving traffic signals to and from satellites. Note that transmission from an earth station to the satellite is called uplink (Command), while transmission from the satellite to the earth station is called downlink (Telemetry).

After reception of the earth station for satellite data in the period of communication between the station and the satellite, which can be more than once in some industrial satellites and at most once in others, and this data is a frame, frame from a number of packets. Each satellite has its own protocol to connect to the earth station and to send data to the ground to be protected. This data has its own structure even if it is received from anyone who is not responsible cannot benefit from it and know that information.

The problem facing the ground station is that it wants to decode the structure of each packet and convert it to the concept information of the space engineers and draw some on the charts.

1.3.6 Case Study (EGYSAT1):

EGYSAT1 Architecture:

Packet Format:

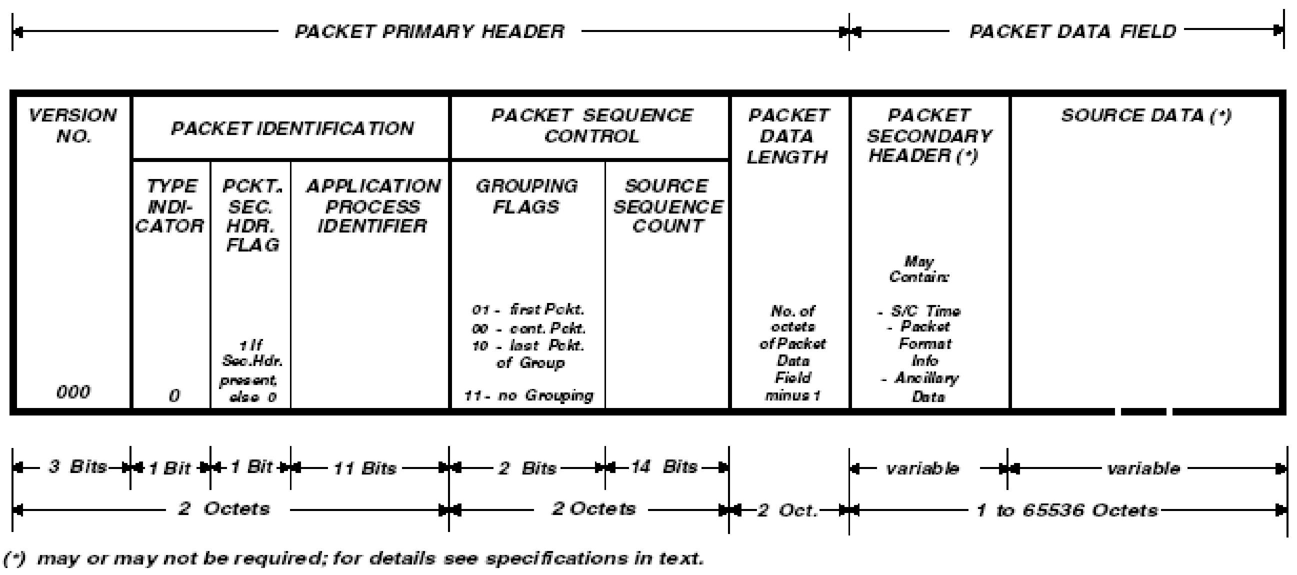


Fig. 4 EGYSAT1 telemetry packet format

As shown in figure 3:

- Version Number – «000» binary;
    - Type Indicator – «1» – Tele command Packet, «0» – Telemetry Packet;
    - Packet Secondary Header Flag – "1", if a Packet Secondary Header is present in the packet; "0", if a Packet Secondary Header is not present;
    - Application Process Identifier – identifier of an application process, which issues data (packets). Application Process Identifier field contains of 11 bits;
    - Grouping Flags. The grouping flags shall be set as follows:

· "01" for the first source packet of a group;
· "00" for a continuing source packet of a group;
· "10" for a last source packet of a group.

For a source packet not belonging to a group of source packets the grouping flags shall be set to "11".

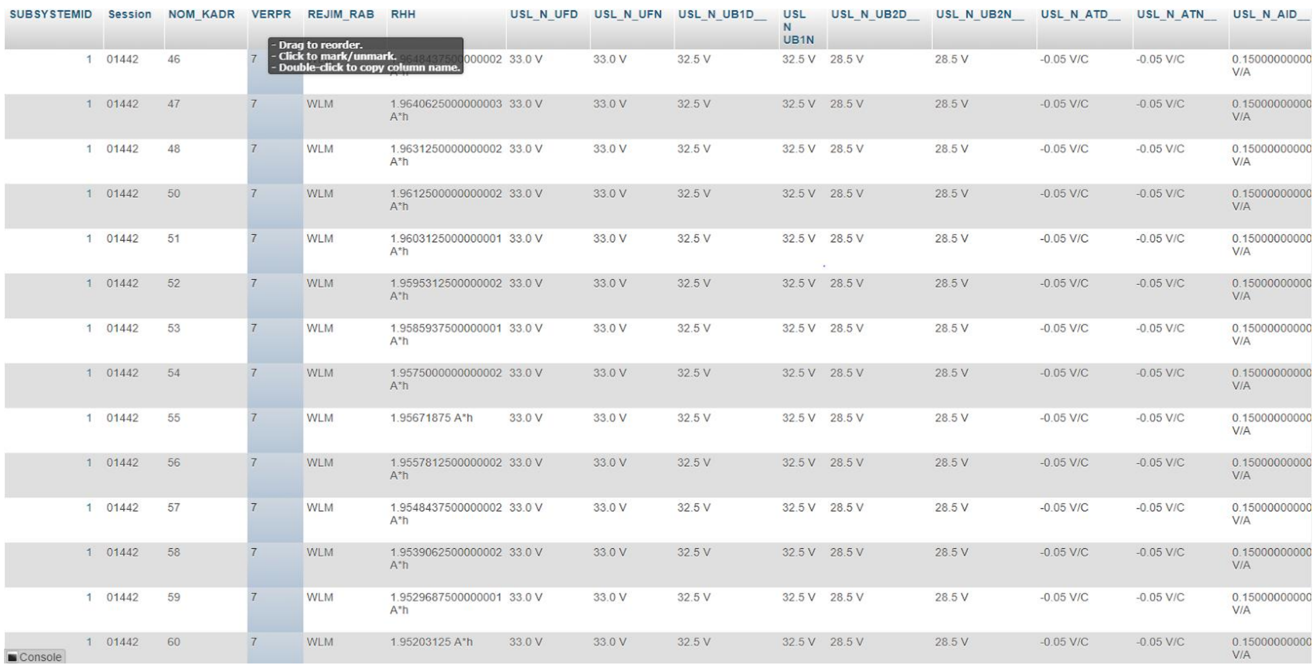
- The Source Sequence Count provides the sequential binary count of each source packet generated by an application process identified by a unique application process identifier. The source sequence count shall be continuous, modulo 16384.
- Packet Data Field Length – binary number equal to the number of octets in the packet data field minus 1.

TEST DATA

We used in the test data a 3 months telemetry data each file is a single session consisting of huge number of packets.

TEST RESULTS

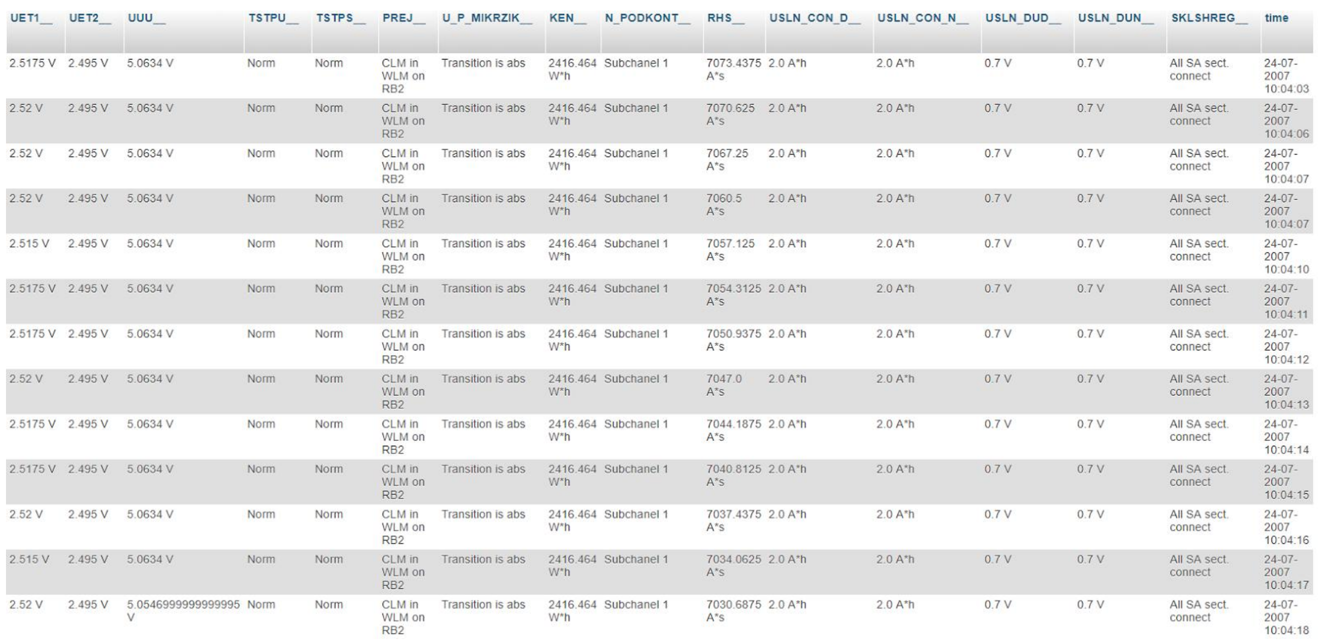
We have unpacked these files successfully and stored the unpacked packets in the database. These 3 months of telemetry data made about 106,000 power unpacked packet stored in the database. Each row has the session number as shown in fig.5 and the timestamp of each packet as shown in fig.6.



- Drag to reorder.
- Click to mark/unmark.
- Double-click to copy column name.

SUBSYSTEMID	Session	NOM_KADR	VERPR	REJIM_RAB	RHH	USL_N_UFD	USL_N_UFN	USL_N_UB1D__	USL N UB1N	USL_N_UB2D__	USL_N_UB2N__	USL_N_ATD__	USL_N_ATN__	USL_N_AID__
1	01442	46	7	[illegible]	[illegible]9648437500000002	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	47	7	WLM	1.9640625000000003 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	48	7	WLM	1.9631250000000002 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	50	7	WLM	1.9612500000000002 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	51	7	WLM	1.9603125000000001 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	52	7	WLM	1.9595312500000002 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	53	7	WLM	1.9585937500000001 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	54	7	WLM	1.9575000000000002 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	55	7	WLM	1.95671875 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	56	7	WLM	1.9557812500000002 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	57	7	WLM	1.9548437500000002 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	58	7	WLM	1.9539062500000002 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	59	7	WLM	1.9529687500000001 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A
1	01442	60	7	WLM	1.95203125 A*h	33.0 V	33.0 V	32.5 V	32.5 V	28.5 V	28.5 V	-0.05 V/C	-0.05 V/C	0.15000000000 V/A

Fig. 5 Screenshot of unpacked packet of EGYSAT1 with session number



UET1__	UET2__	UUU__	TSTPU__	TSTPS__	PREJ__	U_P_MIKRZIK__	KEN__	N_PODKONT__	RHS__	USLN_CON_D__	USLN_CON_N__	USLN_DUD__	USLN_DUN__	SKLSHREG__	time
2.5175 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7073.4375 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:03
2.52 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7070.625 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:06
2.52 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7067.25 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:07
2.52 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7060.5 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:07
2.515 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7057.125 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:10
2.5175 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7054.3125 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:11
2.5175 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7050.9375 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:12
2.52 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7047.0 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:13
2.5175 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7044.1875 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:14
2.5175 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7040.8125 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:15
2.52 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7037.4375 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:16
2.515 V	2.495 V	5.0634 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7034.0625 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:17
2.52 V	2.495 V	5.0546999999999995 V	Norm	Norm	CLM in WLM on RB2	Transition is abs	2416.464 W*h	Subchanel 1	7030.6875 A*s	2.0 A*h	2.0 A*h	0.7 V	0.7 V	All SA sect. connect	24-07-2007 10:04:18

Fig. 6 Screenshot of the unpacked packet of EGYSAT1 with its own timestamp

1.4 Project Objective

We solved this problem by developing software that unpacks the packet received from the satellite and a database that we store the unpacked data in. The database is designed that we can retrieve the desired sensor reads using the session number and the specific time. Finally the GUI (graphical user interface) that shows the responsible engineer the sensors' reads charts and graphs to facilitate the satellite monitoring.

In the case of EGYSAT1 the Ukrainian model was a solution but on the other hand it was a black box to us. The advantage was that it was compatible with the EGYSAT1 and it had a lot of options. The disadvantage was that we couldn't do more that was permitted by the Ukraine.

It's not secure to have satellite software designed by other country and it's a black box to us.

1.4.1 SYSTEM ARCHITECTURE:

The System architecture is a 3 tier architecture consisting of 3 tiers as shown in fig 10. These tiers are presentation tier, logic tier and data tier.

1. The data tier is the data stored in the database.

2. The logic tier is considered as the processing unit of the program responsible for the calculations and manipulating the data between the other 2 tiers. It's also the layer containing all the back end software controlling the unpacking process and the mining process as well.

3. The presentation tier is mainly the GUI (Graphical User Interface) which is always in contact with the operator of the program.

One of the main advantages of this architecture is the security maintained by these layers as anyone can't access the data without going through the 3 tiers with their security. It's also useful in the isolation property and the ease of modifications in the future.

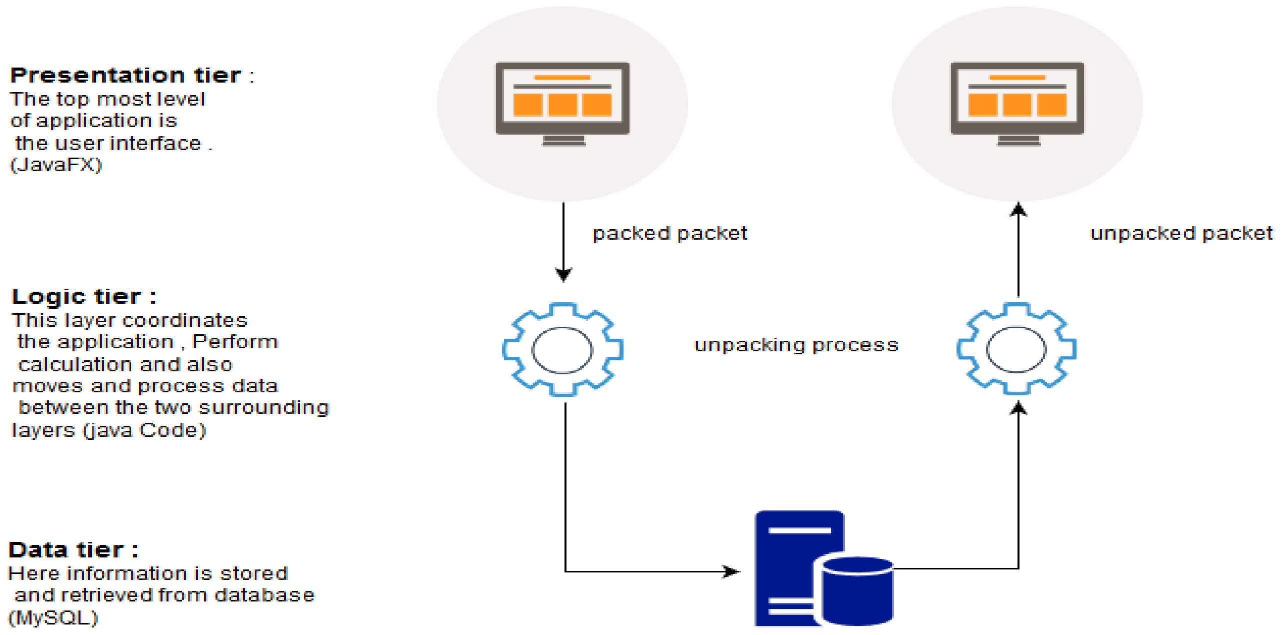

Presentation tier :
The top most level of application is the user interface . (JavaFX)
Logic tier :
This layer coordinates the application , Perform calculation and also moves and process data between the two surrounding layers (java Code)
Data tier :
Here information is stored and retrieved from database (MySQL)
packed packet
unpacked packet
unpacking process

Fig. 7 System architecture diagram

1.4.2 **Data mining**

We have used the data mining algorithms to the unpacked packets to get useful information out of it. We used the R language to apply these algorithms for the reasons mentioned recently. The purpose of the data mining in this system is to calculate the actual capacity of the battery using the sent data and the then estimate the capacity of the battery for the next sessions. That can be used to estimate the time of the failure of the satellite.

Algorithms applied:

We have used the SVM algorithm to predict the future capacity of the battery. We used the following equation to calculate the capacity of the battery fig. 17.

$$C = \sum (I_{disch} - I_{ch}) \cdot \Delta t$$

C	: Charged/Discharged capacity (Ampere Hour)
I_{disch}	: Battery current during discharge (Ampere)
I_{ch}	: Battery current during charging (Ampere)
Δt	: Time step (i.e. 10 sec.)

Fig8 Formula of calculating the capacity of battery

We use the formula above to calculate the current capacity and then use it on the predicted packets to get an estimation of the future efficiency of the battery.

1.5 Gantt chart of project time plan

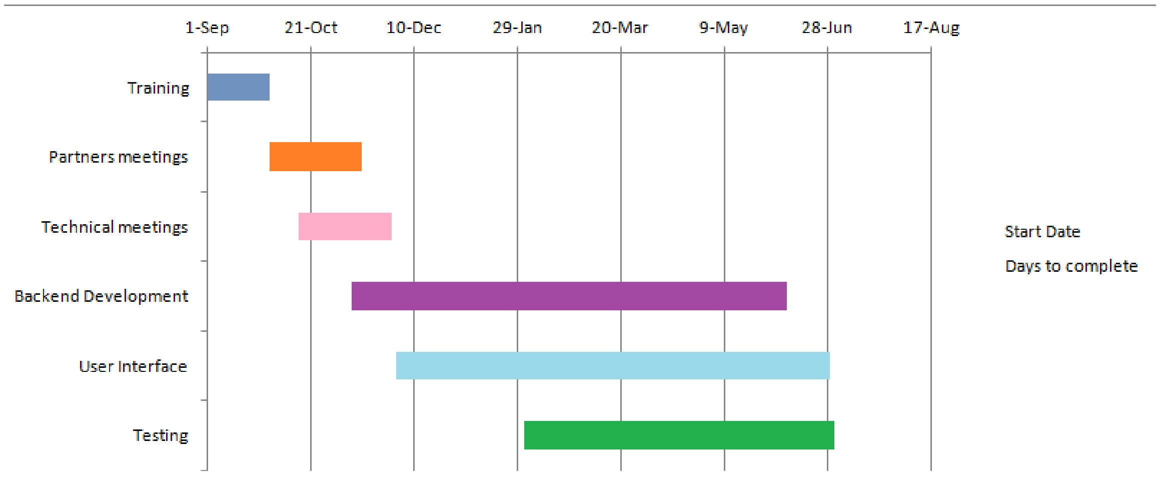


Fig.9. Gantt chart of the project time plan

1.6 Project development methodology

Waterfall methodology:

Waterfall model is a classical model used in system development life cycle to create a system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion. This model is divided into different phases and the output of one phase is used as the input of the next phase.

1.7 The used tools in the project

We have used multiple tools to create this system.

- java for the backend code with NetBeans IDE 8.2. for many reasons:
  - As it is compatible with all platforms without the need for recompilation.
  - It allows creating modular maintainable applications and reusable code.
  - Java is platform-independent
  - One of the most significant advantages of Java is its ability to move easily from one system to another.
  - The ability to run the same code on many different systems using.
- MYSQL for the creating the database developing it using EasyPHP Devserver 16.1.1. for many reasons:
  - Data Security as MySQL is globally known for being the most secure and reliable database management system used in popular web applications like WordPress, Drupal, Joomla, Facebook and Twitter.
  - High Performance as MySQL features a distinct storage-engine framework that facilitates system administrators to configure the MySQL database server for a flawless performance.
  - The Flexibility of Open Source.
- Using JavaFx to design UI
  - Many companies are now seeing the benefits of JavaFX, and are migrating their existing Swing applications to JavaFX applications. While there is definitely a learning curve for JavaFX, many developers are relative fast up to speed. JavaFX API's are very well aligned with the other API's that are part of the Java SDK.
- We used also the R language to apply the data mining algorithms using Rstudio for many reasons

- R is the most comprehensive statistical analysis package available. It incorporates all of the standard statistical tests, models, and analyses
- R is a programming language and environment developed for statistical analysis by practicing statisticians and researchers.
- The graphical capabilities of Rare outstanding, providing a fully programmable graphics language that surpasses most other statistical and graphical packages.
- R is free and open source software, allowing anyone to use and, importantly, to modify it.
- R has no license restrictions.

1.8 Report Organization

The document was organized in four chapters. This is the introduction chapter. It gives general idea about the project, its objectives, methodology and introduces the next chapters. The rest of the thesis is organized as follows:
Chapter two covers the nearest examples of the project and the main differences between them.
Chapter three shows all steps of the software development, starting from the assumptions made, requirements, design, coding and ending with the test plan.
Chapter four is devoted to the software design, shows System Component diagram, System Class Diagrams, sequence Diagrams, project ERD
And also system UI design as a screenshots of the software

Chapter 2: Related work

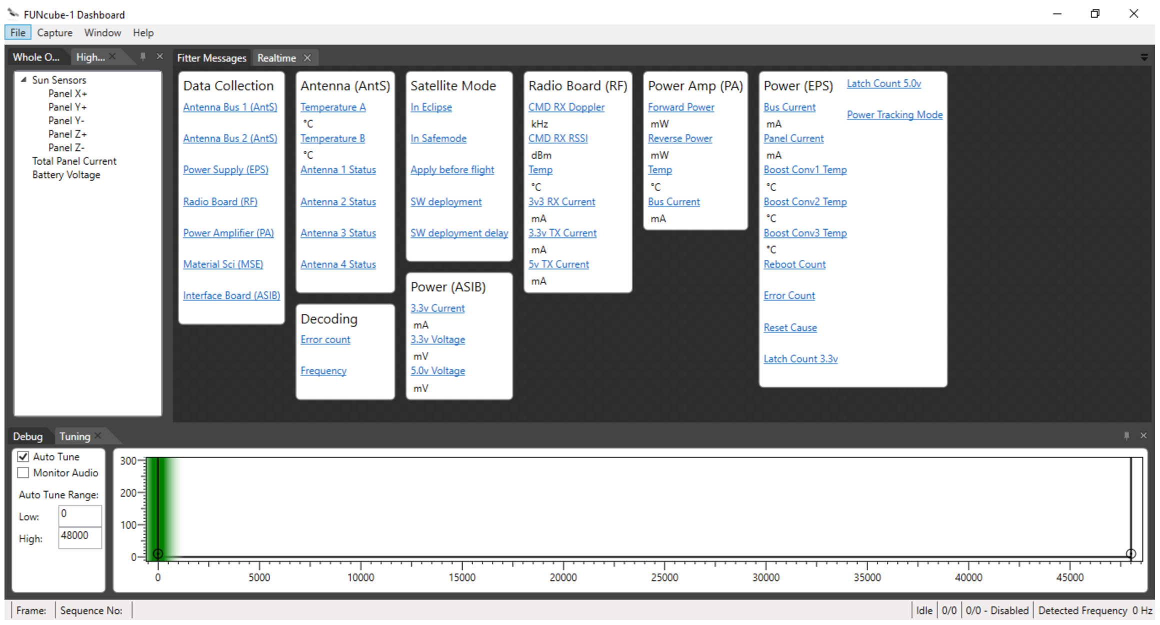
2.1 FUNcube-1

2.1.1 Introduction

The FUNcube-1 Dashboard user interface enables the display of telemetry, debug data and Fitter messages from the FUNcube-1 (FC1) spacecraft. It can also upload the received data to the FUNcube Data Warehouse over the internet.

The Dashboard can accept live inputs from a directly connected FUNcube Dongle (both Pro and Pro+ models) or by audio from another SSB radio fed into the computer soundcard. It can also display previously recorded data from IQ “WAV” files or from the “.funcubebin” recordings that it can create during operation.

The first screen you will see is shown in Figure 4. There are eleven separate sets of telemetry data displayed on the initial Dashboard screen along with a summary set (“Telemetry Decoding”) which gives details on how successful the data decoding is progressing. The last field to appear is usually the “Power Tracking Mode” field at the bottom of the EPS frame on the right hand side or may be the Telemetry Decoding details.



[✓] Auto Tune

Auto Tune Range:

Low: 0

High: 48000

Detected Frequency 0 Hz

Fig.10. FUNcube-1 Dashboard

2.1.2 The main difference

The FUNcube-1 Dashboard is working on static subsystems and sensors as user can't change anything in the number of subsystems or sensors

And also can't change sensors information or add more sensors in subsystem

But FUNcube-1 Dashboard has more functionality like debug and tuning and too many related graphs

2.2 Another Examples:

2.2.1 PolySat

In 1999, a team designed, constructed, and tested CP1, Cal Polly’s first satellite. CP1 communicated on amateur radio frequencies using a combination of Morse Code and Dual Tone Multi-Frequency (DTMF) to encode data. Morse Code is used to identify transmissions while allowing operators to tune to the correct frequency, and DTMF data is sent at 15 characters per second. Despite its relative simplicity, the CP1 communication system was highly efficient .

2.2.2 AAU1 CubeSat

It was developed by students of Aalborg University in Denmark. The system used a 9600 baud rate for communications. This satellite used a Mobitex packet encoding scheme underneath standard AX.25. These packets contained telemetry data, but could not be decoded by regular amateur radio operators due to the Mobitex packet encoding. This satellite beaconed every two minutes if the on-board computer was not functioning, and every four minutes in a low battery situation. Ground stations only received about 1 kB of data .

2.2.3 CUTE-1

CUTE-1 is a Japanese CubeSat developed by sixteen graduate and undergraduate students at Tokyo Institute of Technology Laboratory for Space Systems. One of the primary mission goals was to test two different implementations of downlink communication protocols.

Chapter 3: System Analysis

3.1 Project specification

3.1.1. Functional requirement

- Decode text file with several packets to readable form
- Sensors values displayed using charts.
- Initialize section enable the user to initialize the sensors in every subsystems
- Data saved into the DB so that user can view them anytime.
- Displaying values of the sensors of all subsystems
- Design window to display specific sensors
- Show data mining result

3.1.2 Non-functional requirement

1- **Usability:**
UI enable the user to see all sensors values in the required packet in specific subsystem or in chosen sensors needed
Can also make charts of that values to make it more readable and make decisions on it

2- **Modifiability:**

System can be generic for more than one satellite
As all needed parameter will entered by the user

3- Security:

Using 3 Tier architecture makes User doesn’t have direct access to the database and logic layer

4- Performance:

Performance involves things such as throughput of information through the system, system response time (which also relates to usability), recovery time, and start-up time.

5- Robustness:

The ability of a system to maintain a function. Even if the user enters a wrong input, the program is not crash and handle this situations. Even if there are changes in the environment.

3.2 Use cases

This section shown the project use cases which summarize some relation between use cases, actors and systems

3.2.1 Use case diagram:

The way a user can interact with the software is shown in the use case diagrams shown in Figure 7. The user can import a file to be decoded, save the decoded data to the database, draw charts, show values of all sensors in their subsystem, choose specific sensors to show them and also initialize the program with new data for every subsystem and their sensors.

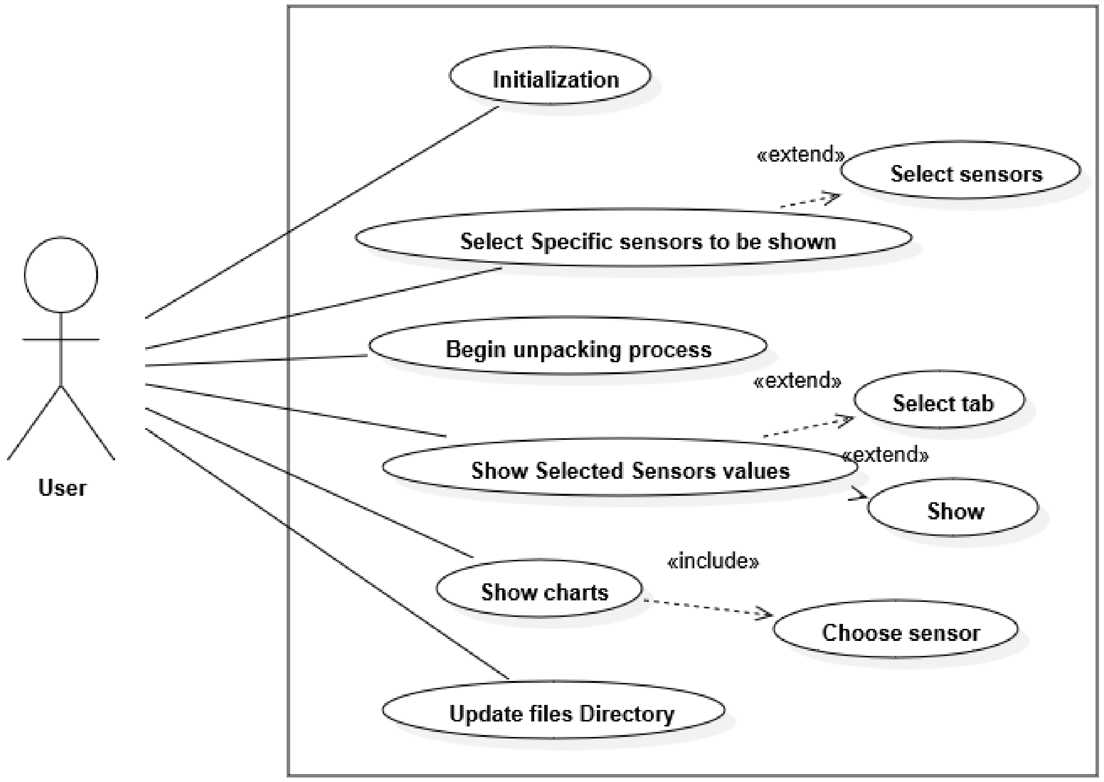


Fig.11 Use case diagram

3.2.2 Use cases table description :

Use Case ID	1	
Use Case Name:	Initialization	
Actors:	User	
Brief Description	This use case allow the user to enter sensors information	
Pre-conditions:	Opening the software	
Post-conditions:		
Flow of events:	**User Action**	**System Action**
	1- Press initialization button	
		2- Ask user to fill the required initialization filed
	3- fill the required initialization filed for every sensor 4 – press button show	
		5 – Show the entered data for every sensor in the table
	6 – Press save	
		7- Save sensors information into DB
Includes:		
Notes and Issues:	1- User can remove sensor after showing it the table 2- Initialization process has limitation on the sensors number if user exceed it warning notification will appear	

Use Case ID:	2	
Use Case Name:	Select Specific sensors to be shown	
Actors:	User	
Brief Description	This use case allow the user to select specific sensors to be shown in the unpacking process	
Pre-conditions:	Opening the software	
Post-conditions:	Make new tab	
Flow of events:	**User Action**	**System Action**
	1- Press select sensors key	
		2- Ask user to fill required filed
	3- Choose sensors 4- Write tab name 5- Press save key	

		6- Save tab name and its sensors in DB
Includes:		
Notes and Issues:		

Use Case ID:	3	
Use Case Name:	Begin unpacking process	
Actors:	User	
Brief Description	This use case show the value of each sensor in the subsystem for all the packets in the selected file	
Pre-conditions:	Update files Directory	
Post-conditions:		
Flow of events:	**User Action**	**System Action**
	1- Press Begin unpacking process key	
		2 – Show the equivalent value for every sensor in each subsystem
Includes:		
Notes and Issues:		

Use Case ID:	4	
Use Case Name:	Show Selected Sensors values	
Actors:	User	
Brief Description	This use case allow the user to show specific sensor that had been selected	
Pre-conditions:	Select Specific sensors to be shown	
Post-conditions:		
Flow of events:	**User Action**	**System Action**
	1- Press Show Selected Sensors values key 2- Choose tab from the table 3- Press show	
		4- Begin unpacking process for only selected tab sensors
Includes:		
Notes and Issues:		

Use Case ID:	5	
Use Case Name:	Show charts	
Actors:	User	
Brief Description	This use case allow the user to show chart for specific sensor	
Pre-conditions:	Begin unpacking process or Show Selected Sensors values	
Post-conditions:		
Flow of events:	**User Action**	**System Action**
	1- Press chart button 2- Choose sensor 3- Press show button	
		4- Show chart for selected sensor
Notes and Issues:		
Includes:		

Use Case ID:	6	
Use Case Name:	Update files Directory	
Actors:	User	
Brief Description	This use case allow the user to enter the file directory for files that have the packets	
Pre-conditions:		
Post-conditions:		
Flow of events:	**User Action**	**System Action**
	1- Press Update files Directory button 2- Press browse key 3- Choose file 4- Press save button	
		5- Save file Directory
Includes:		
Notes and Issues:		

3.3 System test cases.

The following levels of testing done to the software:
1. Unit Testing
2. Integration Testing

- Unit testing

  Here 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. this test must have been done regularly and unit by unit. The following functions tested extensively.

  - Packet Information function:
    This function passed the test if a packet has no time
    In this case function won't calculate the time and set it to 0 without making any error in the unpacking process

  - Split function:
    This function passed the test if a packet APID is not in database, the function will ignore that packet

  - Begin unpacking function:
    This function passed the test if user check to begin unpacking process without selecting a valid text file , the software will pop up notification and wouldn't allow unpacking

  - Initialization function:
    This function must pass the test if user in the initialization section enters more than specific number of sensors in each subsystem, software will pop up notification and wouldn't allow

  - Calibration Function:
    Function grantees there is no type out of the database so there is no sensor type error and calibration for all sensors will be done

- Integration testing

Since each unit is developed independently, a test must be performed when integrating those units. The model used in developing the system, as unit by unit are finished and then new unit is added to the existing system, the integration testing is done by the end of each unit, and each time only one unit is added to the integration. This simplified the testing very much.

Chapter 4: System Design

4.1 System Component Diagram

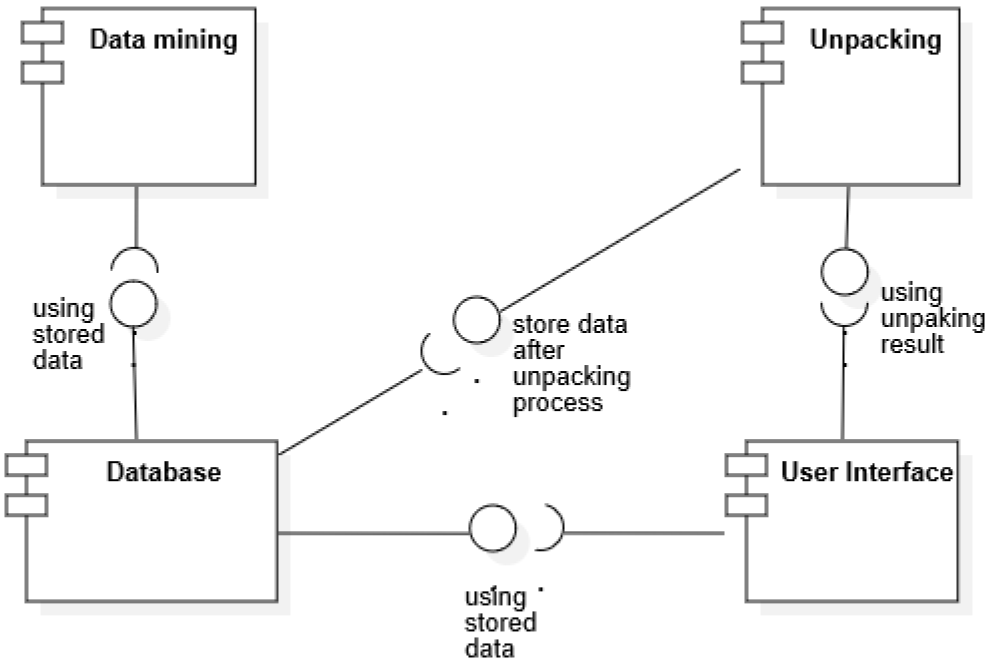


Fig.12 Component diagram

4.2 System Class Diagram

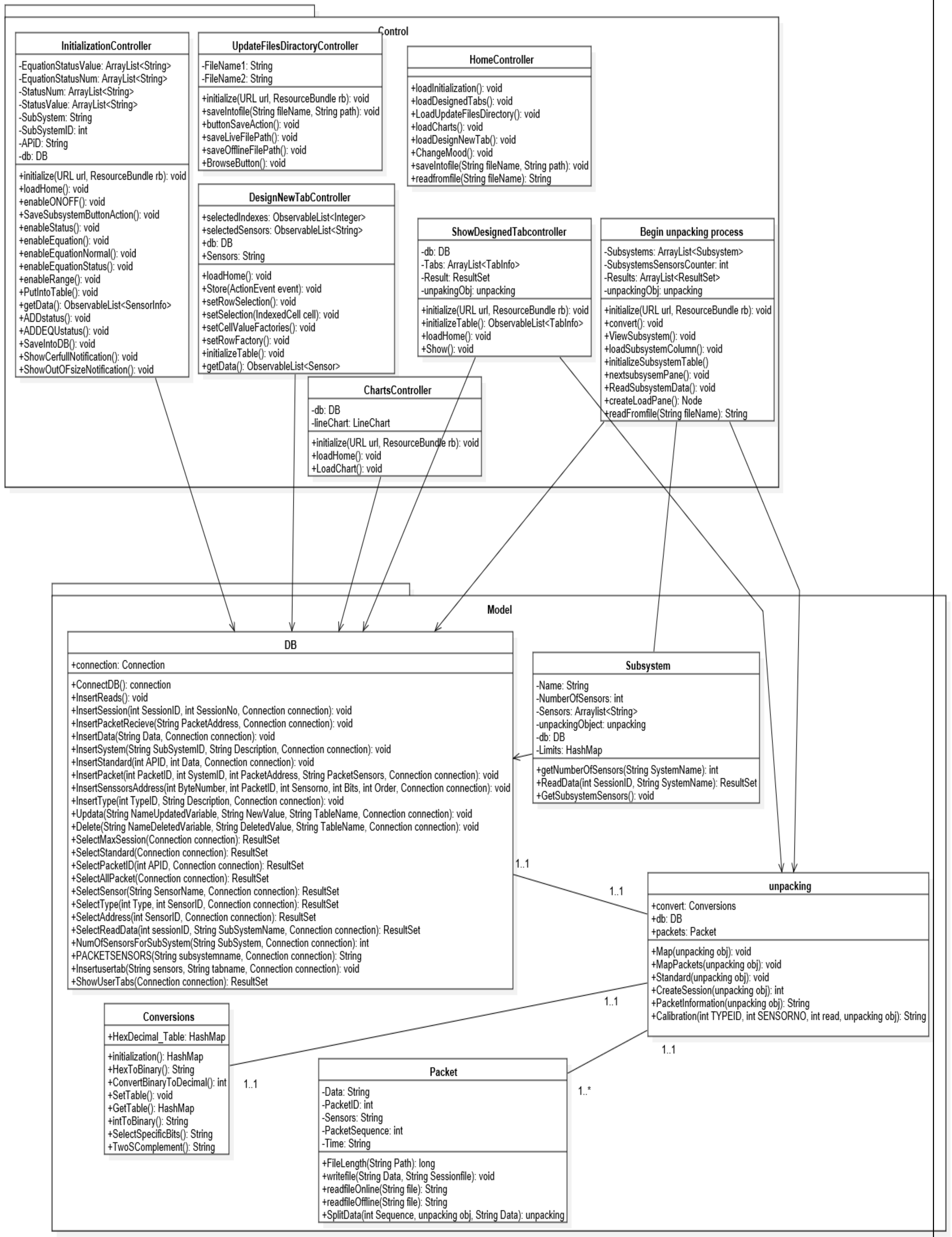

Control
InitializationController
-EquationStatusValue: ArrayList<String>
-EquationStatusNum: ArrayList<String>
-StatusNum: ArrayList<String>
-StatusValue: ArrayList<String>
-SubSystem: String
-SubSystemID: int
-APiD: String
-db: DB
+initialize(URL url, ResourceBundle rb): void
+loadHome(): void
+enableONOFF(): void
+SaveSubsystemButtonAction(): void
+enableStatus(): void
+enableEquation(): void
+enableEquationNormal(): void
+enableEquationStatus(): void
+enableRange(): void
+PutIntoTable(): void
+getData(): ObservableList<SensorInfo>
+ADDstatus(): void
+ADDEQUstatus(): void
+SaveIntoDB(): void
+ShowCerfullNotification(): void
+ShowOutOFsizeNotification(): void
UpdateFilesDiractoryController
-FileName1: String
-FileName2: String
+initialize(URL url, ResourceBundle rb): void
+saveIntofile(String fileName, String path): void
+buttonSaveAction(): void
+saveLiveFilePath(): void
+saveOfflineFilePath(): void
+BrowseButton(): void
HomeController
+loadInitialization(): void
+loadDesignedTabs(): void
+LoadUpdateFilesDirectory(): void
+loadCharts(): void
+loadDesignNewTab(): void
+ChangeMood(): void
+saveIntofile(String fileName, String path): void
+readfromfile(String fileName): String
DesignNewTabController
+selectedIndexes: ObservableList<Integer>
+selectedSensors: ObservableList<String>
+db: DB
+Sensors: String
+loadHome(): void
+Store(ActionEvent event): void
+setRowSelection(): void
+setSelection(IndexedCell cell): void
+setCellValueFactories(): void
+setRowFactory(): void
+initializeTable(): void
+getData(): ObservableList<Sensor>
ShowDesignedTabcontroller
-db: DB
-Tabs: ArrayList<TabInfo>
-Result: ResultSet
-unpakingObj: unpacking
+initialize(URL url, ResourceBundle rb): void
+initializeTable(): ObservableList<TabInfo>
+loadHome(): void
+Show(): void
Begin unpacking process
-Subsystems: ArrayList<Subsystem>
-SubsystemsSensorsCounter: int
-Results: ArrayList<ResultSet>
-unpackingObj: unpacking
+initialize(URL url, ResourceBundle rb): void
+convert(): void
+ViewSubsystem(): void
+loadSubsystemColumn(): void
+initializeSubsystemTable()
+nextsubsysemPane(): void
+ReadSubsystemData(): void
+createLoadPane(): Node
+readFromfile(String fileName): String
ChartsController
-db: DB
-lineChart: LineChart
+initialize(URL url, ResourceBundle rb): void
+loadHome(): void
+LoadChart(): void
Model
DB
+connection: Connection
+ConnectDB(): connection
+InsertReads(): void
+InsertSession(int SessionID, int SessionNo, Connection connection): void
+InsertPacketRecieve(String PacketAddress, Connection connection): void
+InsertData(String Data, Connection connection): void
+InsertSystem(String SubSystemID, String Description, Connection connection): void
+InsertStandard(int APID, int Data, Connection connection): void
+InsertPacket(int PacketID, int SystemID, int PacketAddress, String PacketSensors, Connection connection): void
+InsertSenssorsAddress(int ByteNumber, int PacketID, int Sensorno, int Bits, int Order, Connection connection): void
+InsertType(int TypeID, String Description, Connection connection): void
+Updata(String NameUpdatedVariable, String NewValue, String TableName, Connection connection): void
+Delete(String NameDeletedVariable, String DeletedValue, String TableName, Connection connection): void
+SelectMaxSession(Connection connection): ResultSet
+SelectStandard(Connection connection): ResultSet
+SelectPacketID(int APID, Connection connection): ResultSet
+SelectAllPacket(Connection connection): ResultSet
+SelectSensor(String SensorName, Connection connection): ResultSet
+SelectType(int Type, int SensorID, Connection connection): ResultSet
+SelectAddress(int SensorID, Connection connection): ResultSet
+SelectReadData(int sessionID, String SubSystemName, Connection connection): ResultSet
+NumOfSensorsForSubSystem(String SubSystem, Connection connection): int
+PACKETSENSORS(String subsystemname, Connection connection): String
+Insertusertab(String sensors, String tabname, Connection connection): void
+ShowUserTabs(Connection connection): ResultSet
Subsystem
-Name: String
-NumberOfSensors: int
-Sensors: Arraylist<String>
-unpackingObject: unpacking
-db: DB
-Limits: HashMap
+getNumberOfSensors(String SystemName): int
+ReadData(int SessionID, String SystemName): ResultSet
+GetSubsystemSensors(): void
1..1
1..1
unpacking
+convert: Conversions
+db: DB
+packets: Packet
+Map(unpacking obj): void
+MapPackets(unpacking obj): void
+Standard(unpacking obj): void
+CreateSession(unpacking obj): int
+PacketInformation(unpacking obj): String
+Calibration(int TYPEID, int SENSORNO, int read, unpacking obj): String
1..1
Conversions
+HexDecimal_Table: HashMap
+initialization(): HashMap
+HexToBinary(): String
+ConvertBinaryToDecimal(): int
+SetTable(): void
+GetTable(): HashMap
+intToBinary(): String
+SelectSpecificBits(): String
+TwoSComplement(): String
1..1
1..1
Packet
-Data: String
-PacketID: int
-Sensors: String
-PacketSequence: int
-Time: String
+FileLength(String Path): long
+writefile(String Data, String Sessionfile): void
+readfileOnline(String file): String
+readfileOffline(String file): String
+SplitData(int Sequence, unpacking obj, String Data): unpacking
1..*

Fig.13. Class diagram

4.3 Sequence Diagrams

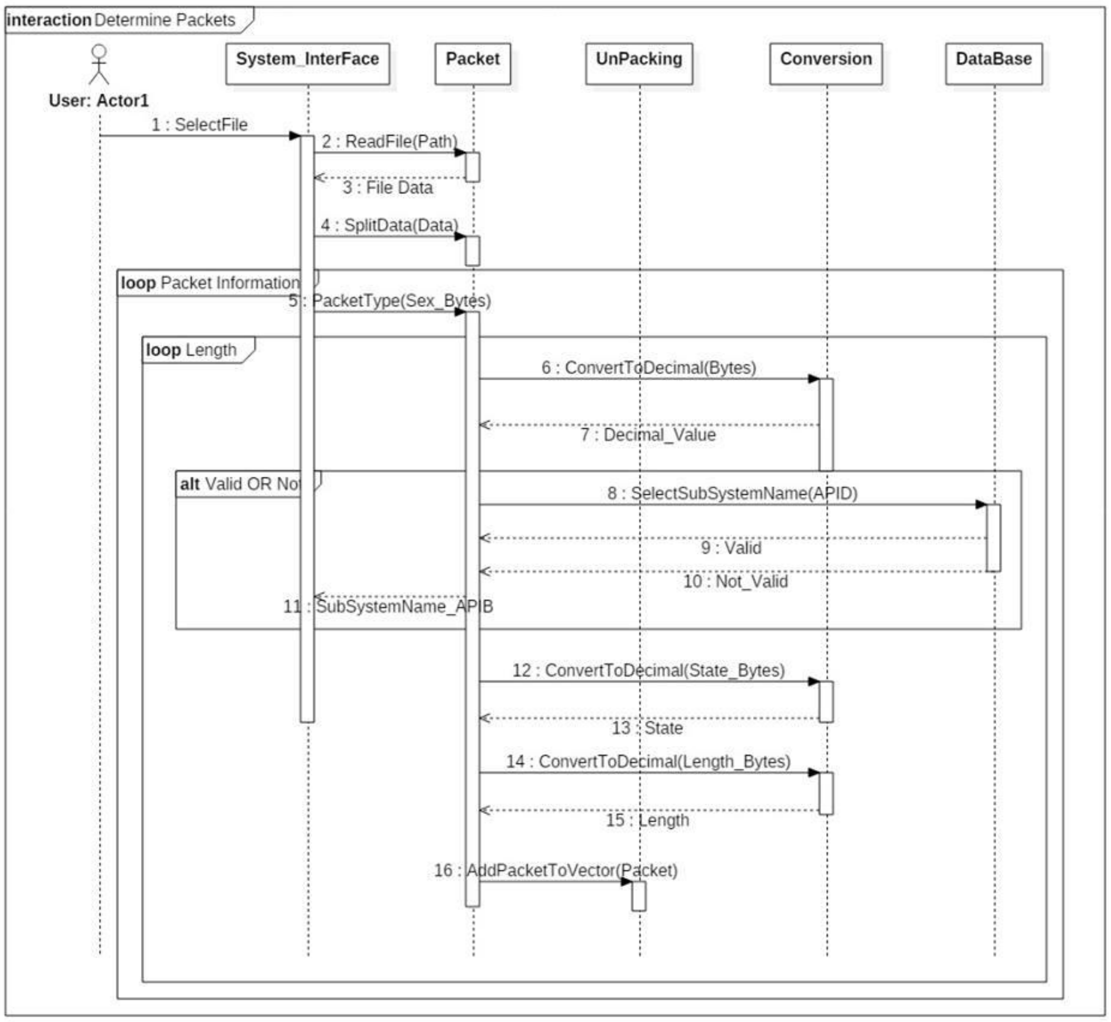


Fig.14.Sequace diagram1

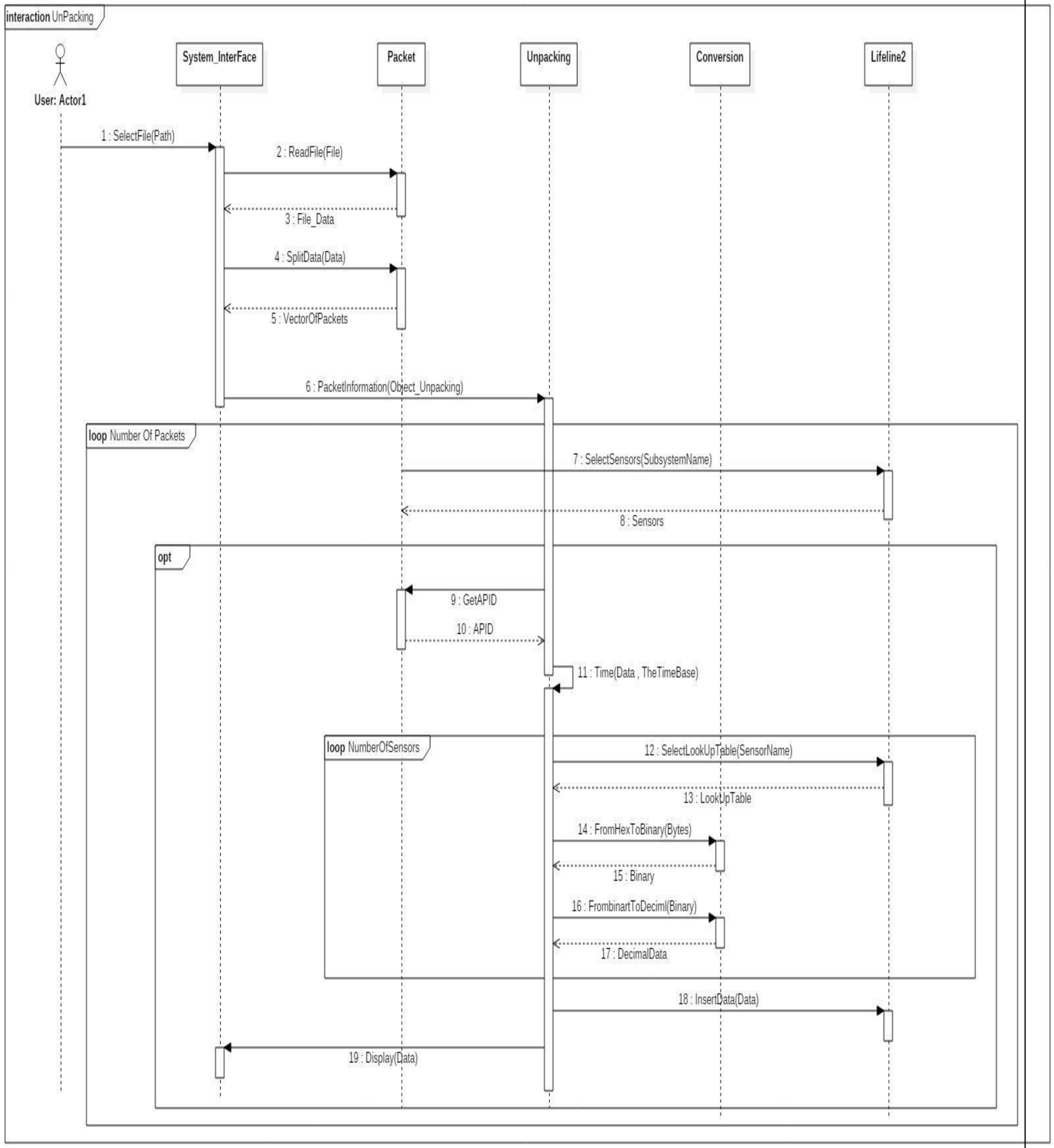


Fig.15.Sequace diagram2

4.4 Project ERD

4.4.1 For (EGYSAT-1) module:

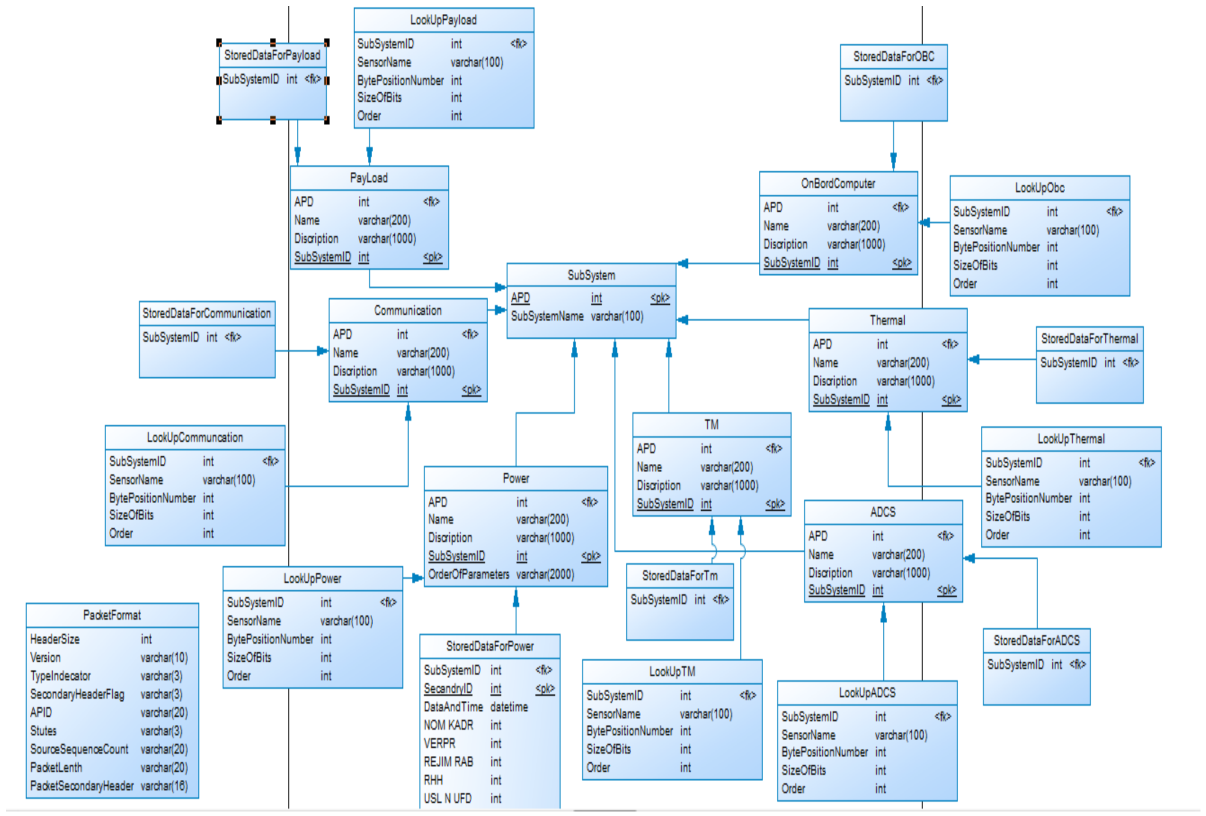


Fig.16. ERD forEGYSAT-1

4.4.2 For the generic module

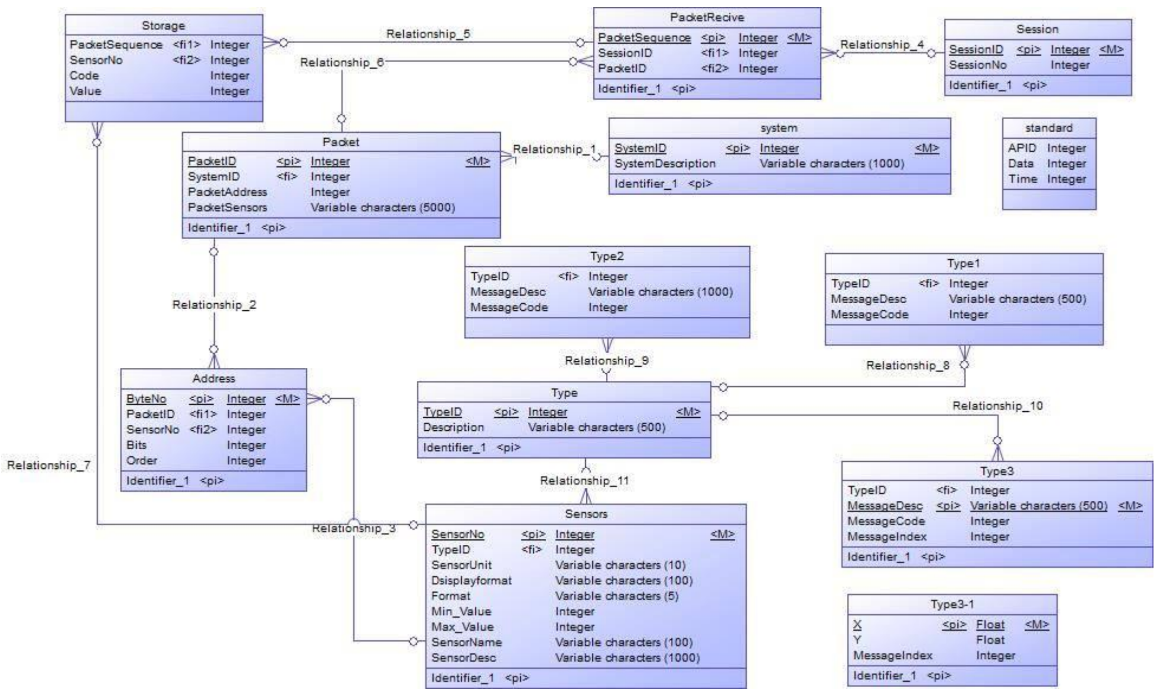


Fig.17. ERD for the generic model

4.5 System GUI Design

The code of the software was written based on the design shown in chapter 3, the system developed was Telemetry unpacking software that has a UI. The UI consists of:

Home page:

Is the first screen you will see is shown in Figure 5. There are seven separate buttons displayed on the initial screen as the menu of the program

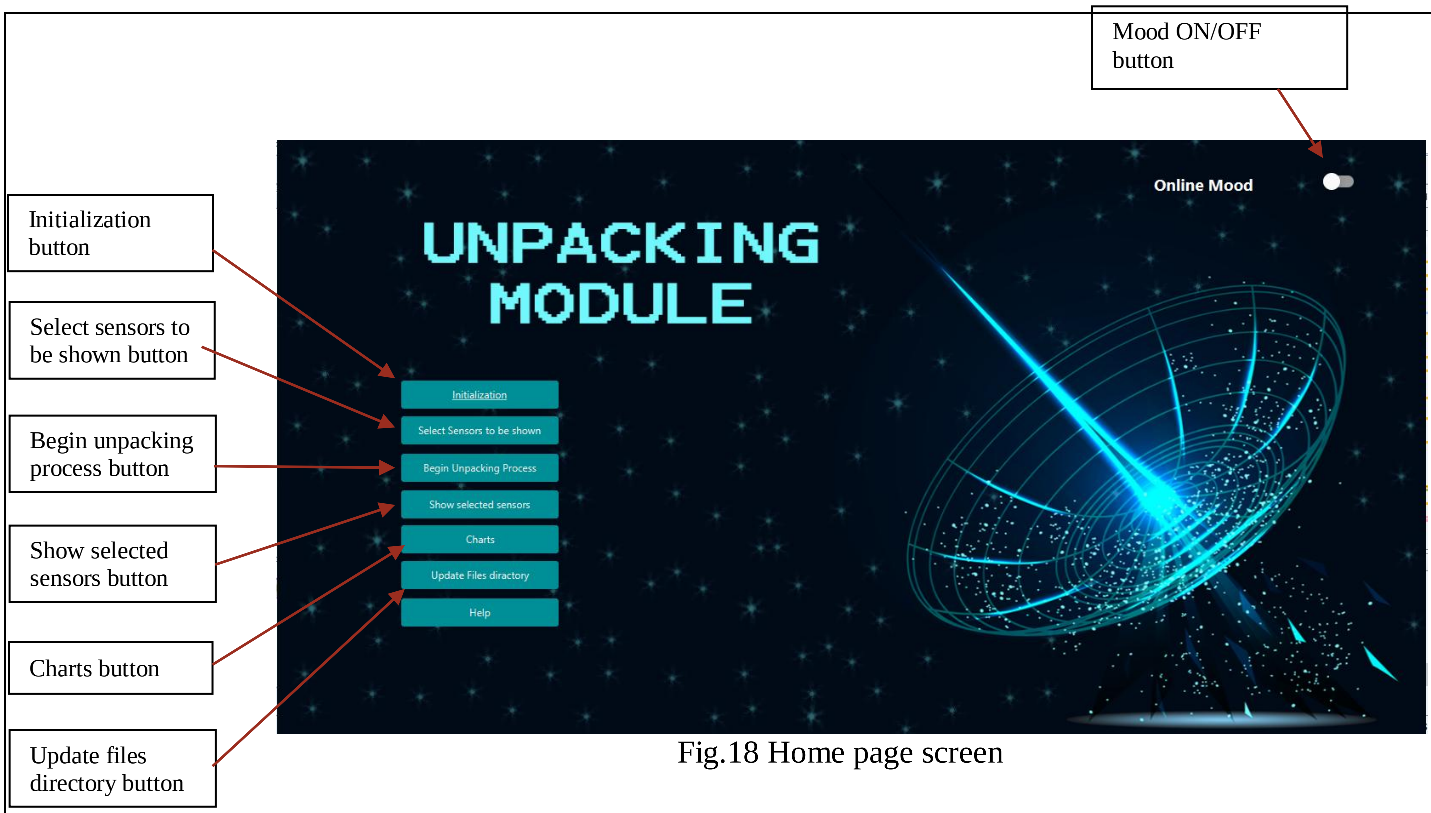


Fig.18 Home page screen

Menu:

1- Initialization

As the last module is a dynamic model initialization section is momentous in our software to enable the user to enter the subsystems information's and its sensors

2- Select Sensors to be shown

Sometimes user don't need to watch all the sensors values during the unpacking process so that section enable the user to choose specific sensors only to be shown and save that sensors in the database under chosen name

3- Begin Unpacking Process

After selecting the files directory software begin to unpack the packets in the chosen file and begin to display it to the user

4- Show selected sensors

In this section user can watch only the selected sensors by choosing the specific tab

5- Charts

User can select sensor and the time of received packets from satellite then software will show the chart of that values

6- Update files Directory

There are two type of unpacking online or offline, every type has a deferent file directory so user can enter the files directory

Software functionality:

Update file directory

It gives the user the ability to make an online session with the satellite or to unpack a previous stored session by choosing which file would be used

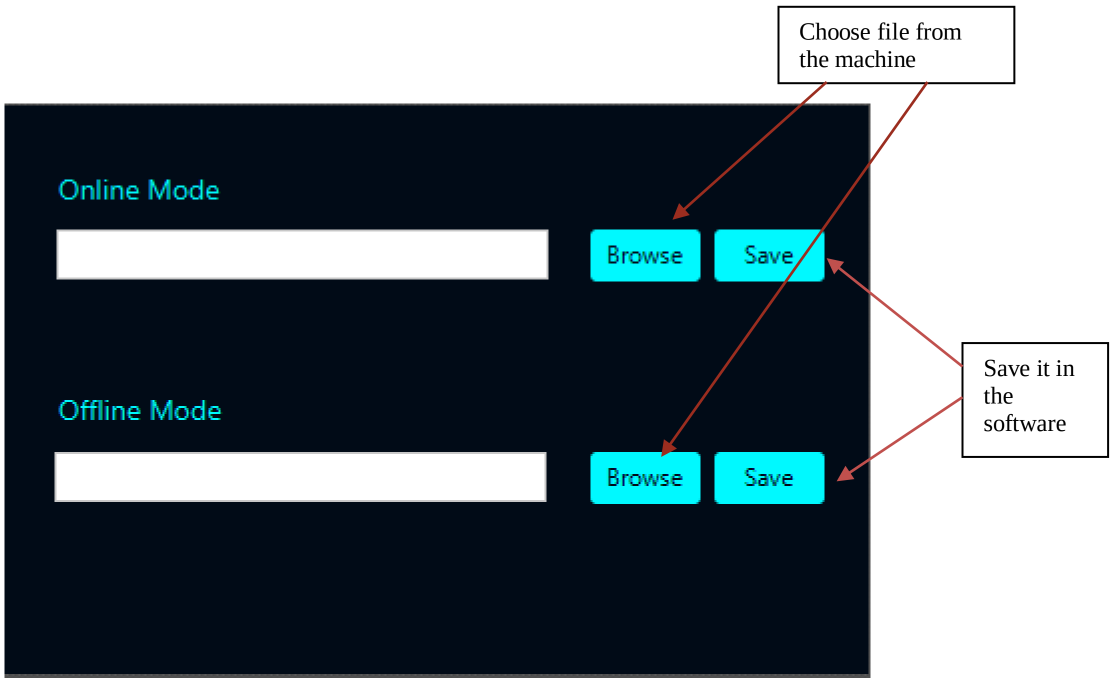


Fig.19 Update file directory screen

■ Initialization

UI enable the user to initialize the sensors for every subsystem by entering their information

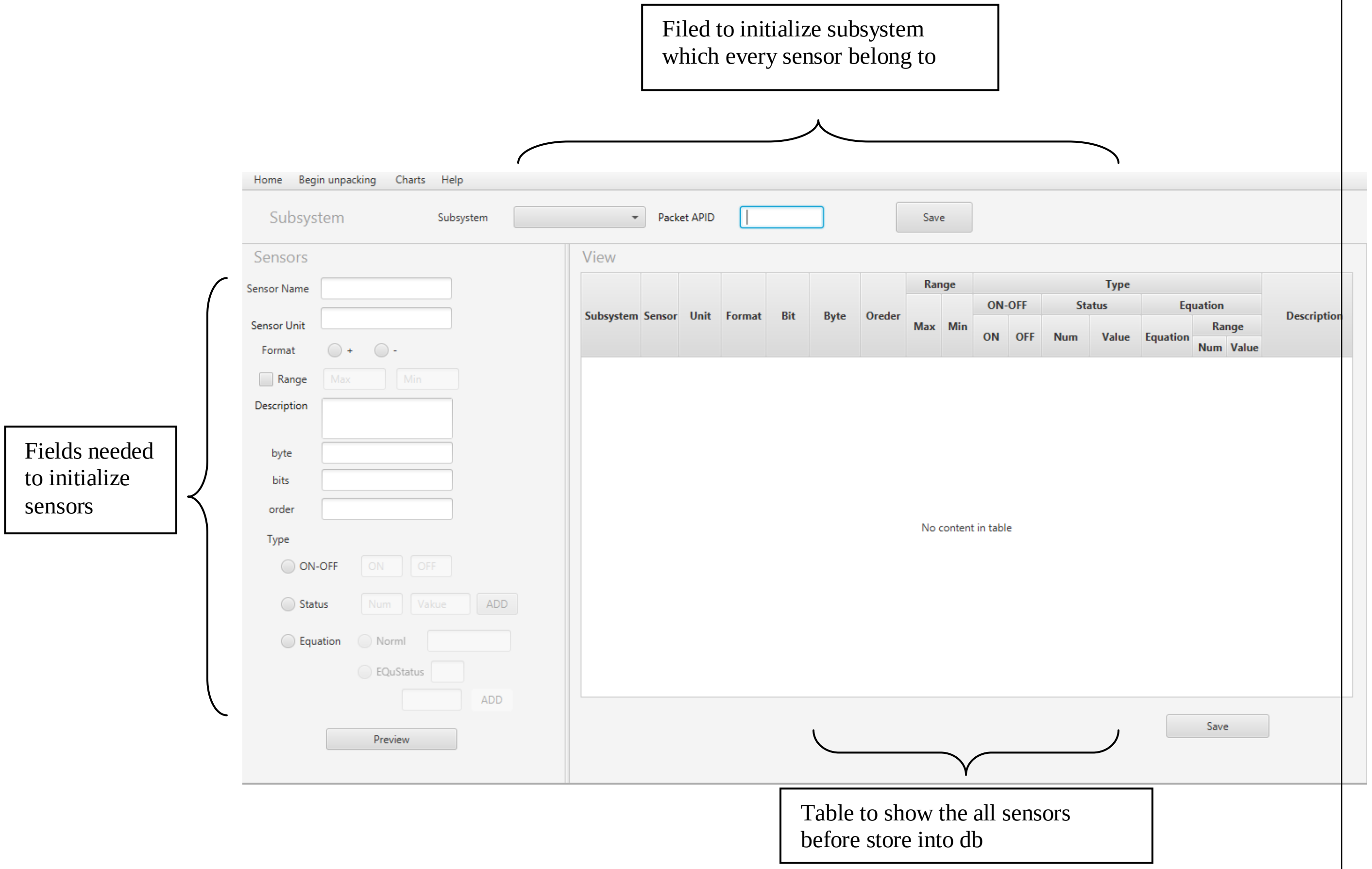
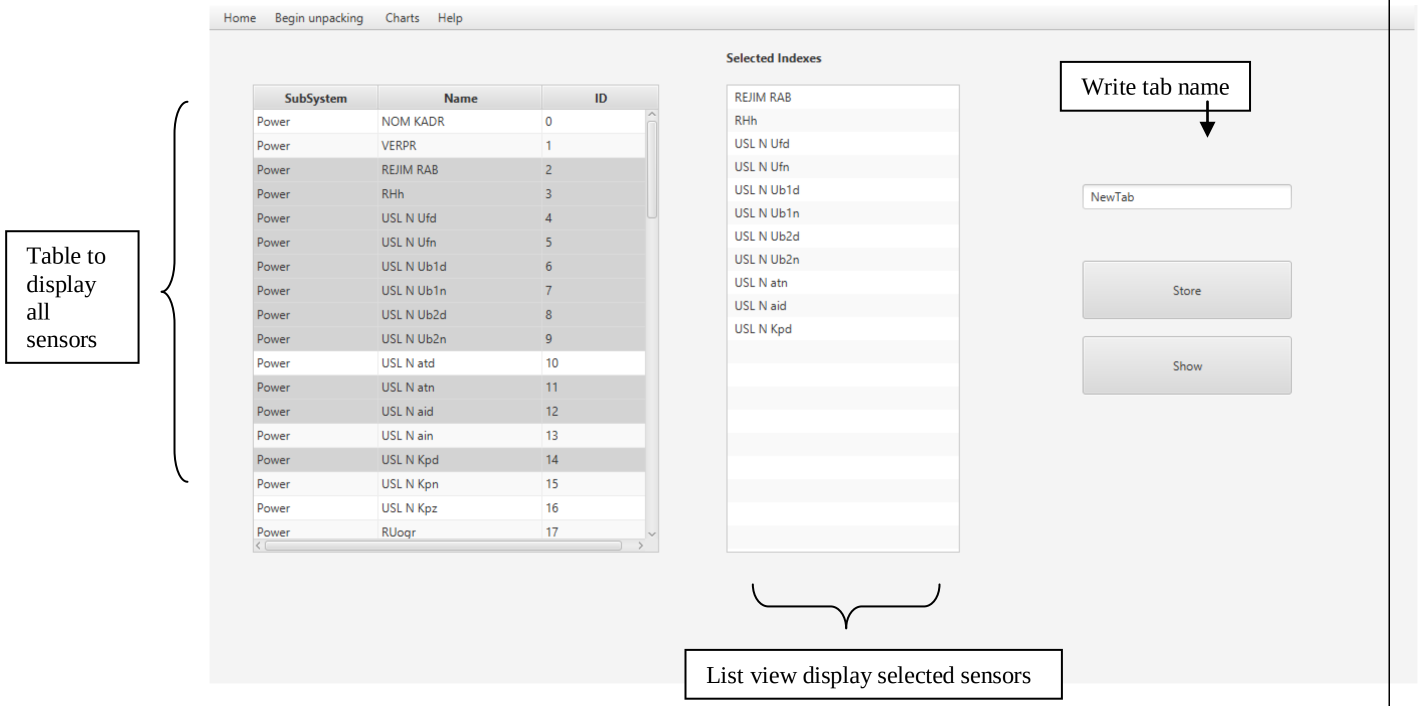


Fig.20 Initialization screen

6- Select Sensors to be shown

Also user can choose specific sensors to watch only their values during unpacking process



SubSystem	Name	ID
Power	NOM KADR	0
Power	VERPR	1
Power	REJIM RAB	2
Power	RHh	3
Power	USL N Ufd	4
Power	USL N Ufn	5
Power	USL N Ub1d	6
Power	USL N Ub1n	7
Power	USL N Ub2d	8
Power	USL N Ub2n	9
Power	USL N atd	10
Power	USL N atn	11
Power	USL N aid	12
Power	USL N ain	13
Power	USL N Kpd	14
Power	USL N Kpn	15
Power	USL N Kpz	16
Power	RUogr	17

REJIM RAB
RHh
USL N Ufd
USL N Ufn
USL N Ub1d
USL N Ub1n
USL N Ub2d
USL N Ub2n
USL N atn
USL N aid
USL N Kpd

Fig.21 Select Sensors to be shown screen

7- Charts

User can view a chart for specific sensor by select sensor and start time, end time and day

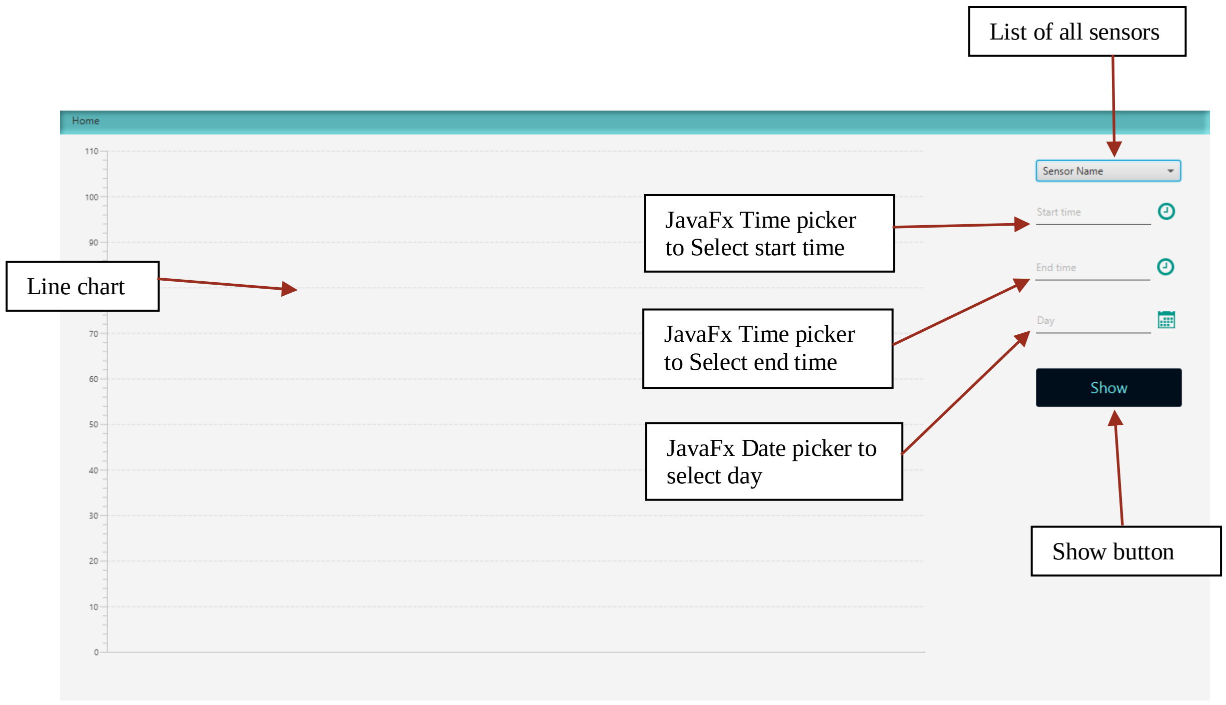
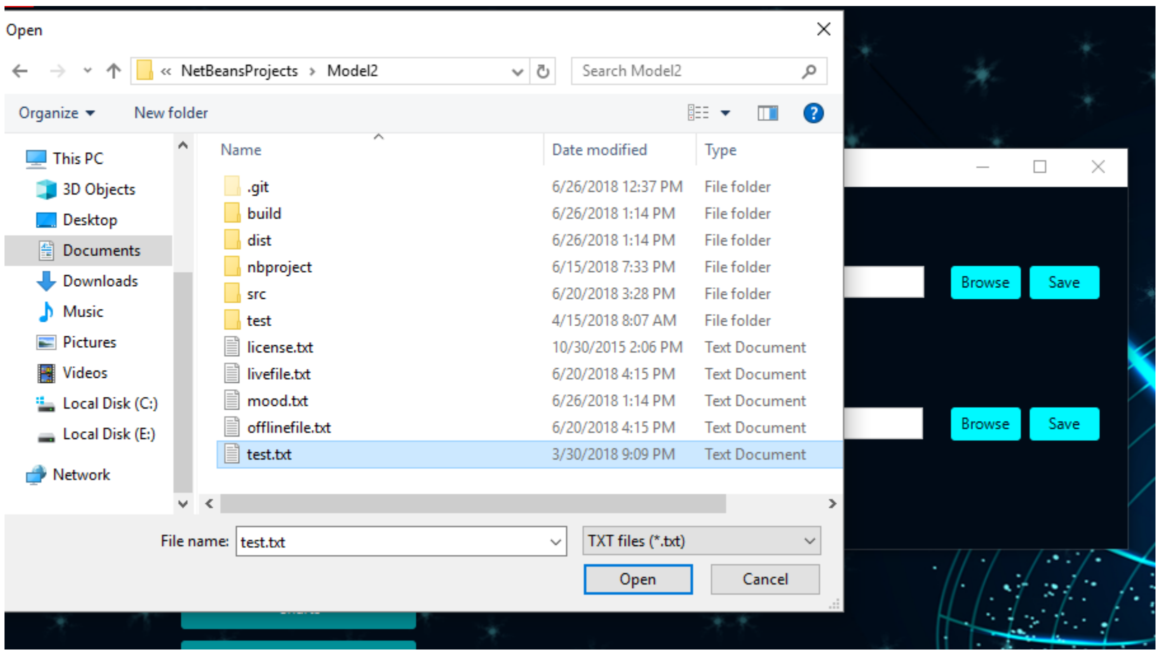

List of all sensors
Home
Sensor Name
Start time
End time
Day
Show
JavaFx Time picker to Select start time
Line chart
JavaFx Time picker to Select end time
JavaFx Date picker to select day
Show button
110
100
90
70
60
50
40
30
20
10
0

Fig.22 show chart screen

How to use the software:

To unpack a file:

1- The user selects the file using the browse button which displays a file chooser dialogue box in the Update files directory window
The file chooser by default displays text files only.

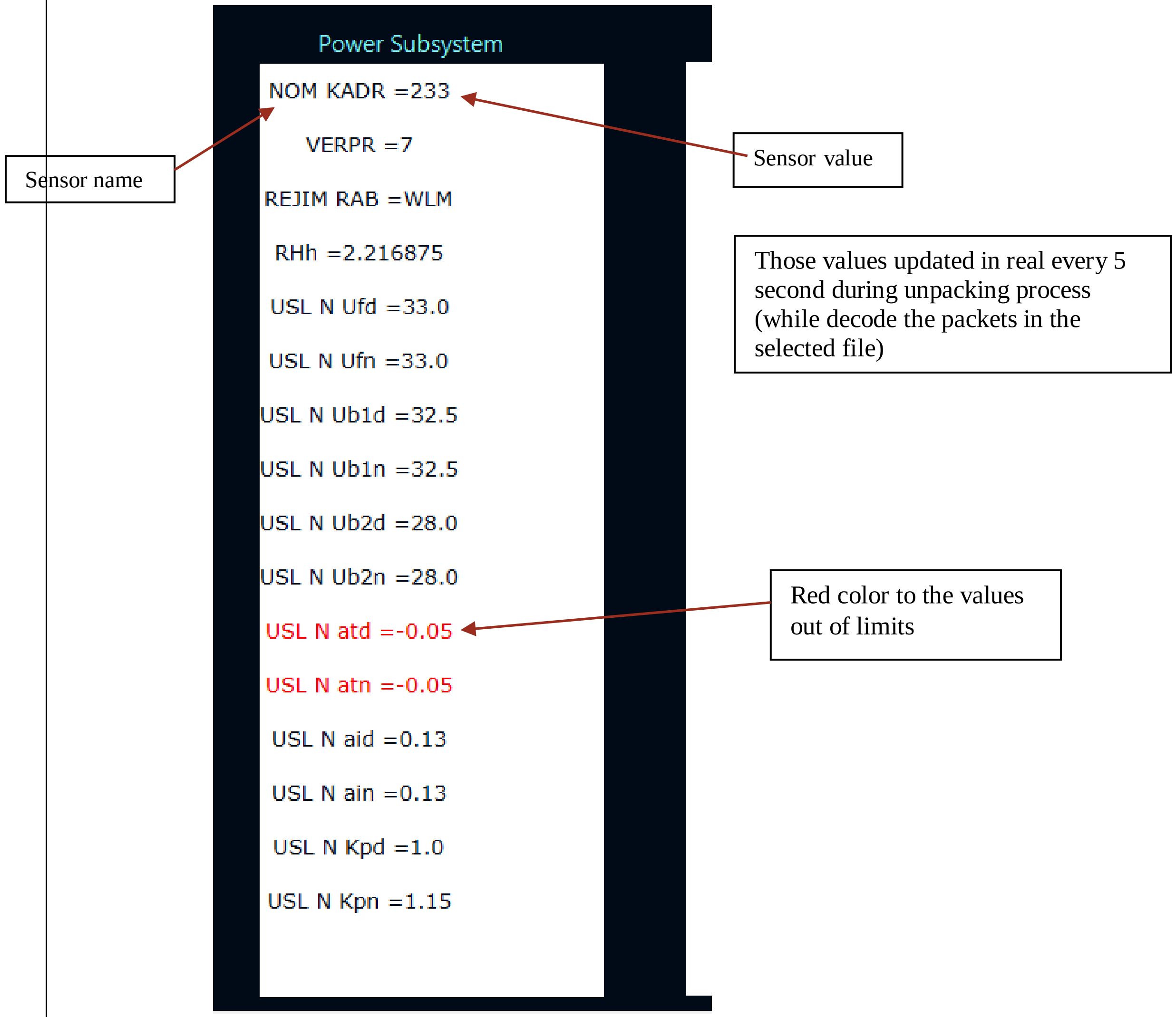


Name	Date modified	Type
.git	6/26/2018 12:37 PM	File folder
build	6/26/2018 1:14 PM	File folder
dist	6/26/2018 1:14 PM	File folder
nbproject	6/15/2018 7:33 PM	File folder
src	6/20/2018 3:28 PM	File folder
test	4/15/2018 8:07 AM	File folder
license.txt	10/30/2015 2:06 PM	Text Document
livefile.txt	6/20/2018 4:15 PM	Text Document
mood.txt	6/26/2018 1:14 PM	Text Document
offlinefile.txt	6/20/2018 4:15 PM	Text Document
test.txt	3/30/2018 9:09 PM	Text Document

File name: test.txt
TXT files (*.txt)

Fig.23 Select file screen

2- Program will be decoded and displayed on the software.
If we take power subsystem as an example

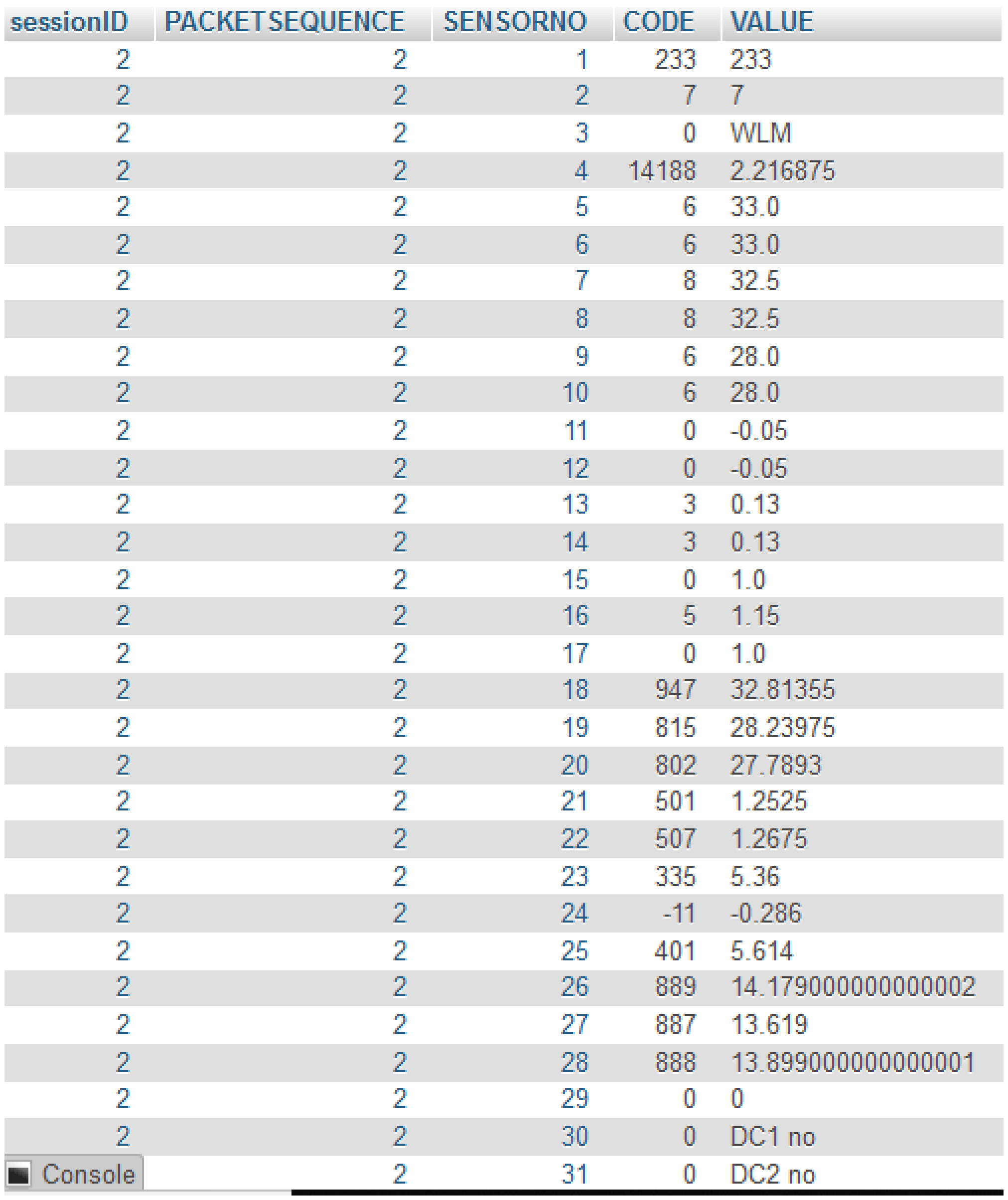

Power Subsystem
Sensor name
Sensor value
Those values updated in real every 5 second during unpacking process (while decode the packets in the selected file)
Red color to the values out of limits

```
NOM KADR =233
VERPR =7
REJIM RAB =WLM
RHh =2.216875
USL N Ufd =33.0
USL N Ufn =33.0
USL N Ub1d =32.5
USL N Ub1n =32.5
USL N Ub2d =28.0
USL N Ub2n =28.0
USL N atd =-0.05
USL N atn =-0.05
USL N aid =0.13
USL N ain =0.13
USL N Kpd =1.0
USL N Kpn =1.15
```

Fig.24 unpacking results screen

3- unpacked data also saved in the database

Sample of data in database



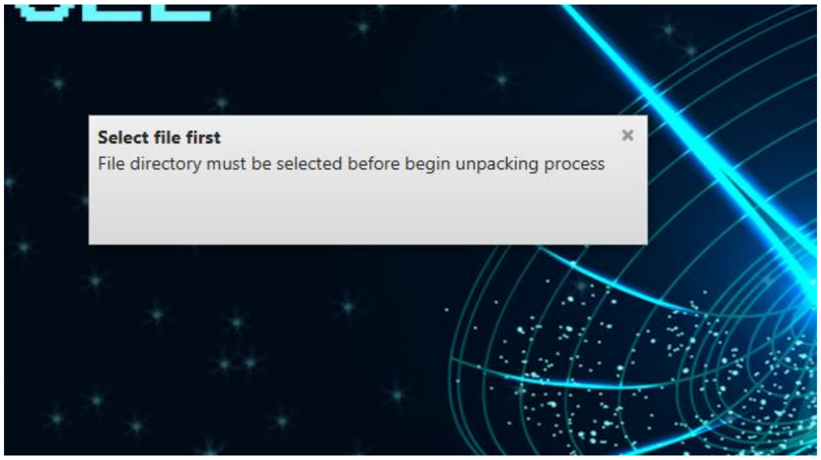
sessionID	PACKETSEQUENCE	SENSORNO	CODE	VALUE
2	2	1	233	233
2	2	2	7	7
2	2	3	0	WLM
2	2	4	14188	2.216875
2	2	5	6	33.0
2	2	6	6	33.0
2	2	7	8	32.5
2	2	8	8	32.5
2	2	9	6	28.0
2	2	10	6	28.0
2	2	11	0	-0.05
2	2	12	0	-0.05
2	2	13	3	0.13
2	2	14	3	0.13
2	2	15	0	1.0
2	2	16	5	1.15
2	2	17	0	1.0
2	2	18	947	32.81355
2	2	19	815	28.23975
2	2	20	802	27.7893
2	2	21	501	1.2525
2	2	22	507	1.2675
2	2	23	335	5.36
2	2	24	-11	-0.286
2	2	25	401	5.614
2	2	26	889	14.179000000000002
2	2	27	887	13.619
2	2	28	888	13.899000000000001
2	2	29	0	0
2	2	30	0	DC1 no
Console	2	31	0	DC2 no

Fig.25 sample of database results

Exceptions and Error Handling:

This is an important part that must be carefully handled when developing software. If it is not well handled the system becomes exposed to failure and thus the performance of the system degrades. The system can handle the following actions:

1. If the user tried to begin unpacking before select any file:
   the system will ask the user to select a file first



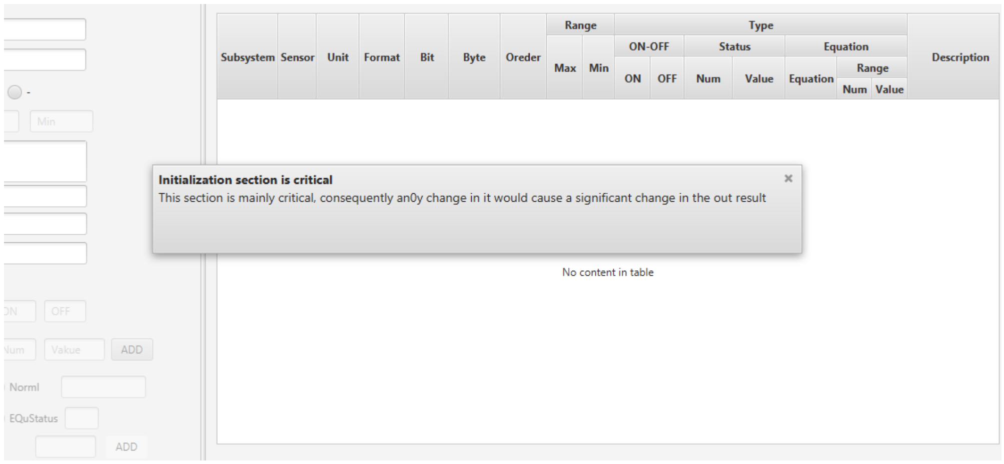
Select file first

File directory must be selected before begin unpacking process

Fig.26 select file notification screen

2. If the user tried to import a non text file:
   By default the file chooser dialog box will display text files only.

3. Using warning notification to show that initialization section is a critical as it can change the behavior of the program



Subsystem	Sensor	Unit	Format	Bit	Byte	Oreder	Range		Type							Description
							Max	Min	ON-OFF		Status		Equation			
									ON	OFF	Num	Value	Equation	Range		
														Num	Value	

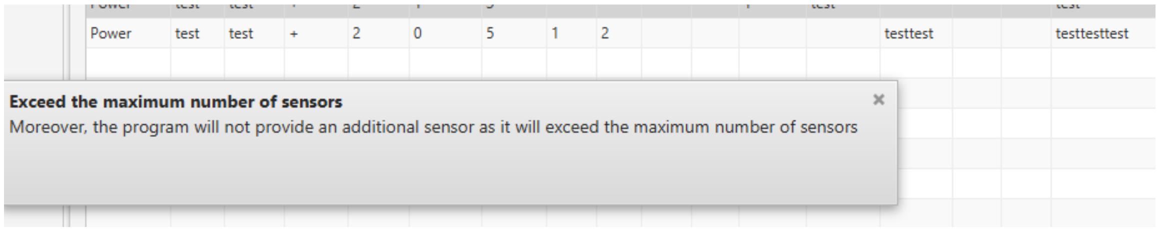
No content in table

Initialization section is critical

This section is mainly critical, consequently an0y change in it would cause a significant change in the out result

Fig27 Initialization notification screen

4. Sensors number has a limit so in initialization section there is a warning notification if user exceed that number and software wont allow the user to enter more sensors



Power	test	test	+	2	0	5	1	2					testtest			testtesttest

Exceed the maximum number of sensors
Moreover, the program will not provide an additional sensor as it will exceed the maximum number of sensors

Fig.28 Exceed the maximum number of sensors notification screen

Finally As the source code is huge to put it in the document
This is the code link on GitHub

https://github.com/Nourhann/GP

References:

wikipedia
https://en.wikipedia.org/wiki/Satellite_system_(astronomy)

waterfall model
https://en.ryte.com/wiki/Waterfall_Model

FUNcube-1
https://funcube.org.uk/

Polysat
http://www.polysat.org/

AAU Cubsat
http://www.space.aau.dk/cubesat/

CUTE-1
https://directory.eoportal.org/web/eoportal/satellite-missions/c-missions/cute-i