## ~~4.2.3 Radio Frequencies (optional)~~

Chapter left blank intentionally.

## 4.8 Software Design

In the following section, an overview of the software modules that will be use is provided.

**4.8.1 Purpose**

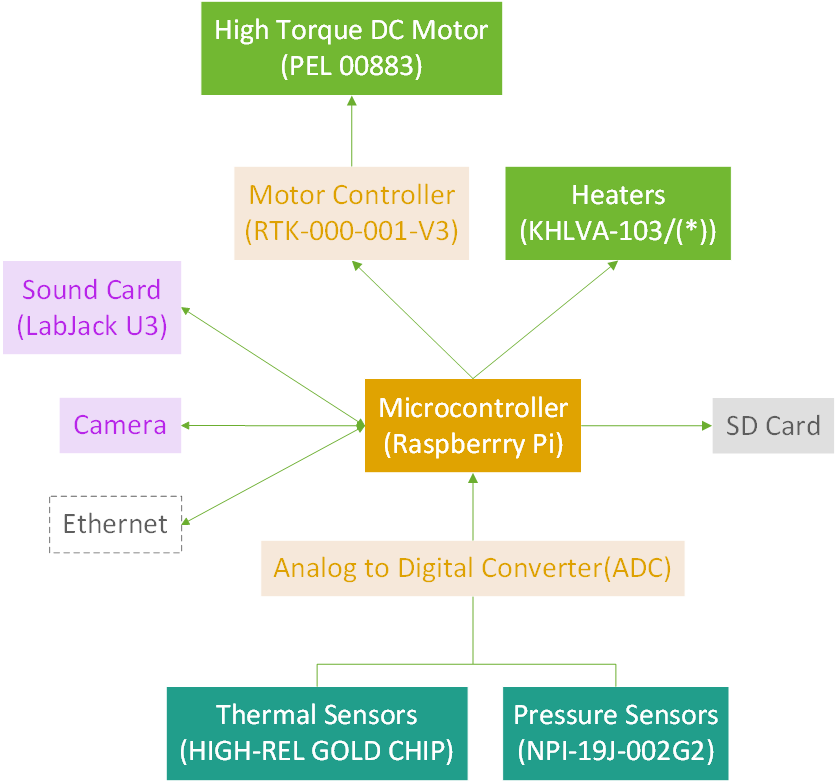
Software both on the on-board part of the experiment and the ground station is going to be used for different reasons.

* The **on-board system** is going to :
  + Monitor the heaters’ temperature (when to turn on and off)
  + Be responsible for the movement of the emulsification motor
  + Turn on and off the I-VED technique
  + Store the data from the I-VED technique in a logfile in the SD card
  + Store the data from the pressure and thermal sensors in a logfile in the SD card
  + Create a collective logfile where all of the data is going to be stored in the SD card
  + Control the Camera and store the video data in the SD card
  + Send a sample of the data of the sensors and the I-VED technique to the ground station
  + Receive commands from the ground station on when each phase of the experiment has started in order to control the motor’s movement and the I-VED technique (TBD)
* The **ground station** **system** is going to :
  + Receive the sample data from the on-board system using the rocket’s Downlink
  + Include a GUI for the visualization of the data during the flight
  + Command the on-board system to turn on and off the I-VED and start or slow down the motor (TBD)

**4.8.2 Design**

**4.8.2.1 Process Overview**

As seen in the diagram [*NUMBER\_OF\_DIAGRAM*] below, the microcontroller will receive data from the A2D (Analog to Digital) Converter which will convert the data from the two sensor sets (3 thermal sensors + 3 pressure ones) and according to them it will adjust the Heaters’ temperature while also storing their data in the SD card. Meanwhile, It will also be able to control the motion of the DC Motor using the Motor Controller and communicate with the Camera and Sound Card (for the I-VED technique) in order to turn them on and store their data in the SD Card. The Ethernet connectivity will be used in order to send a sample of data to the ground station and, in case telecommand will be implemented, it will be able to send the commands to the Microcontroller.(TBD)

****

**4.8.2.2 General and safety related concepts**

As shown in the [NUM\_OF\_PARAGRAPH] Risk assessment paragraph, the probable errors or safety issues that Software subteam has to be concerned over are :

* Microcontroller failure
* Connection Loss between the rocket and the ground station
* Sensor Software Failure

*4.8.2.2.1 Microcontroller Failure or Breakdown*

This error’s probability is extremely low, but if something like that happens the consequences are critical for the experiment. Without the microcontroller the data from the sensors, the control of the motor and the sound card are non-existent, and depending on the timing of the occurrence, it could be completely fatal for the experiment since the camera could never be signaled to start filming.

The ways to mitigate the probability of this happening are :

* Buying the product from a well known and trusted vendor, in order to avoid badly made soldering or PCB printing, mistreatment of the product, and in order to have possible customer support in case something is needed,
* Testing it under extreme conditions and avoid mistreating or endangering it, so that it we can be as sure as possible that the microcontroller won’t misbehave during the flight.

*4.8.2.2.2 Connection Loss*

This includes both total and temporary connection loss between the rocket and the ground station. The severity of this happening, highly depends on whether the telecommand option will be used.

In case it will not be implemented, then the only consequence, that this might bring, is that the people in the ground station during the campaign will have no or partial visibility of the data and the status of the experiment.

In case it will be, even a partial connection loss of a few seconds might be a problem for the experiment, as the people monitoring it won’t be able to send instructions for whatever error might happen or even the standard process. Anything more than 3-5 seconds (TBD), during a critical time or near

The probability of such a thing happening is quite high but manageable up to a point with Go-Back protocols and algorithms like that.

The actions that can be taken in order to avoid these problems are:

* Testing the code extensively in order to make sure that the problem is not created by itself,
* Testing different kinds of situations and try to make up error handling for these,
* Testing with the rocket system mounted to see if some of the problems are created because of other connectivity issues besides the code itself or whether some problems are created because of different protocols.

*4.8.2.2.3 Sensor Software Failure*

Sensor software failure can be considered highly unlikely but its consequences can range between highly dangerous to the experiment to just a partial distortion of the final data. Therefore it creates an issue that should be addressed.

There are very few ways to mitigate the possibilities of such a thing happening, mainly because software vendors are taking care over these stuff and publish new updates regularly, but these are some of the ways :

* Testing the code extensively for possible unknown mistakes or exceptions happening,
* Using well known libraries in order to avoid instabilities in the code.
* Update libraries frequently and test the code on the newest updates

**4.8.2.3 Interfaces**

In terms of the interfaces:

* All of the experiments’ components will be wired-soldered together except for the camera whose connection will be defined later.
* The experiment will connect to the Downlink of the rocket using an RS-422 port. The data bandwidth usage is described below in the following tables.

| **Data to be sent through the Downlink** | | |
| --- | --- | --- |
| Data | bits | Frequency |
| Thermal Sensor 1 | 16 | 3 |
| Thermal Sensor 2 | 16 | 3 |
| Thermal Sensor 3 | 16 | 3 |
| Pressure Sensor 1 | 16 | 3 |
| Pressure Sensor 2 | 16 | 3 |
| Pressure Sensor 3 | 16 | 3 |
| Motor Status | 8 | 1 |
| Heater Status | 1 | 1 |
| Sound Card Status | 1 | 1 |
| Camera Status | 8 | 1 |
| Error | 8 | 1 |
| Sum : | 122 | Max : 3 |

| Sync1 | Sync2 | MsgID | MsgCNT | Data | CSM | CSM | CRC | CRC |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 8 | 8 | 8 | 122 | 8 | 8 | 8 | 8 |

| Baud Rate (bits/s): | 558 |
| --- | --- |

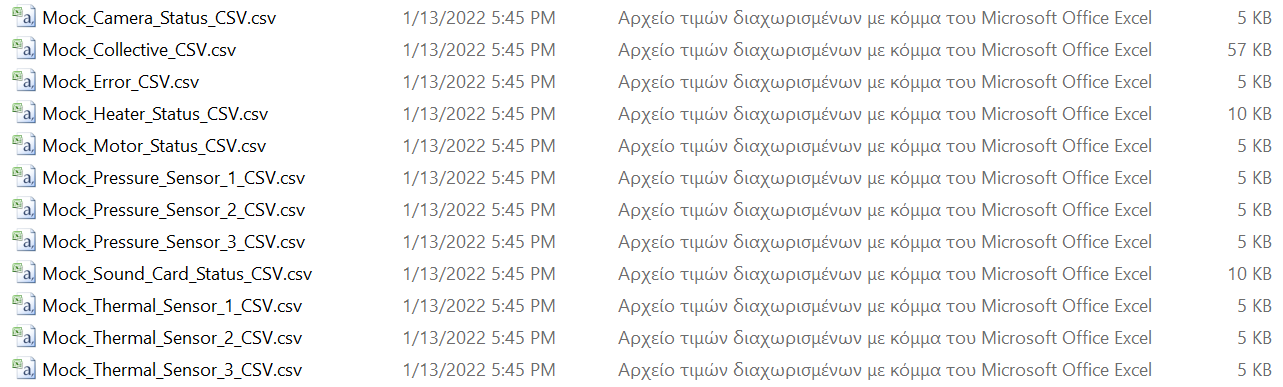
Which derived using the the following formula:

In kB/s that is roughly 0.07kB/s as a maximum Bandwidth.

**4.8.2.4 Data acquisition and storage**

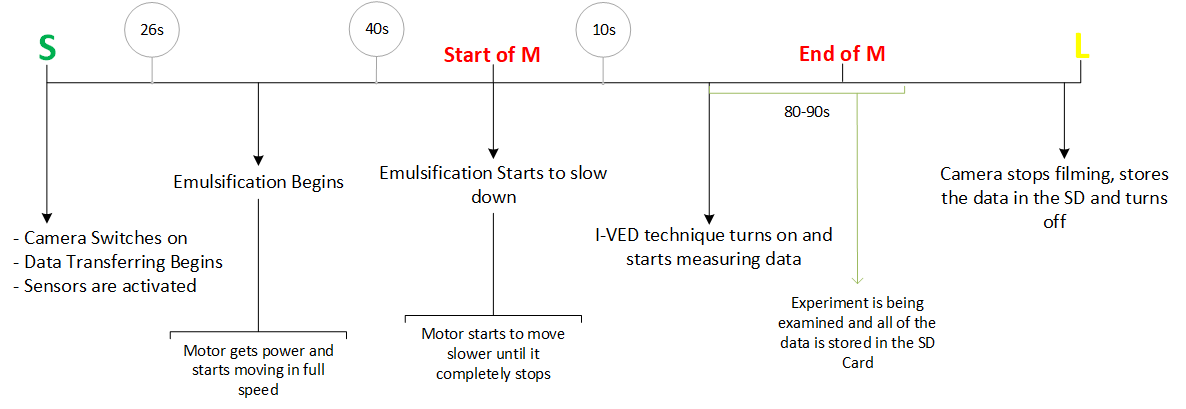
The data collected are going to be split into 3 different categories:

* Video data, which are going to be stored separately inside the camera in order to avoid unnecessary connectivity issues or breakdowns in the connection,
  + The video file size is approximately going to reach 4-5 GBs, but further research is needed because the size depends on many factors, mainly on the technical details of the camera.
* Sound Card data, of the I-VED technique, which are going to be stored inside the the SD card of the Raspberry Pi,
  + The file size of this file ranges between 164 and 52,626 kBytes because the frequency in which the Sound Card will be functioning hasn't been defined yet. But it definitely can be stored inside the SD of the Raspberry Pi.
* Sensor and Event data, which are going to be stored in the same folder as the Sound Card data but will not have any correlation with them.
  + In the image below we can deduct that the total storage size used up by the logfiles is 117kB.



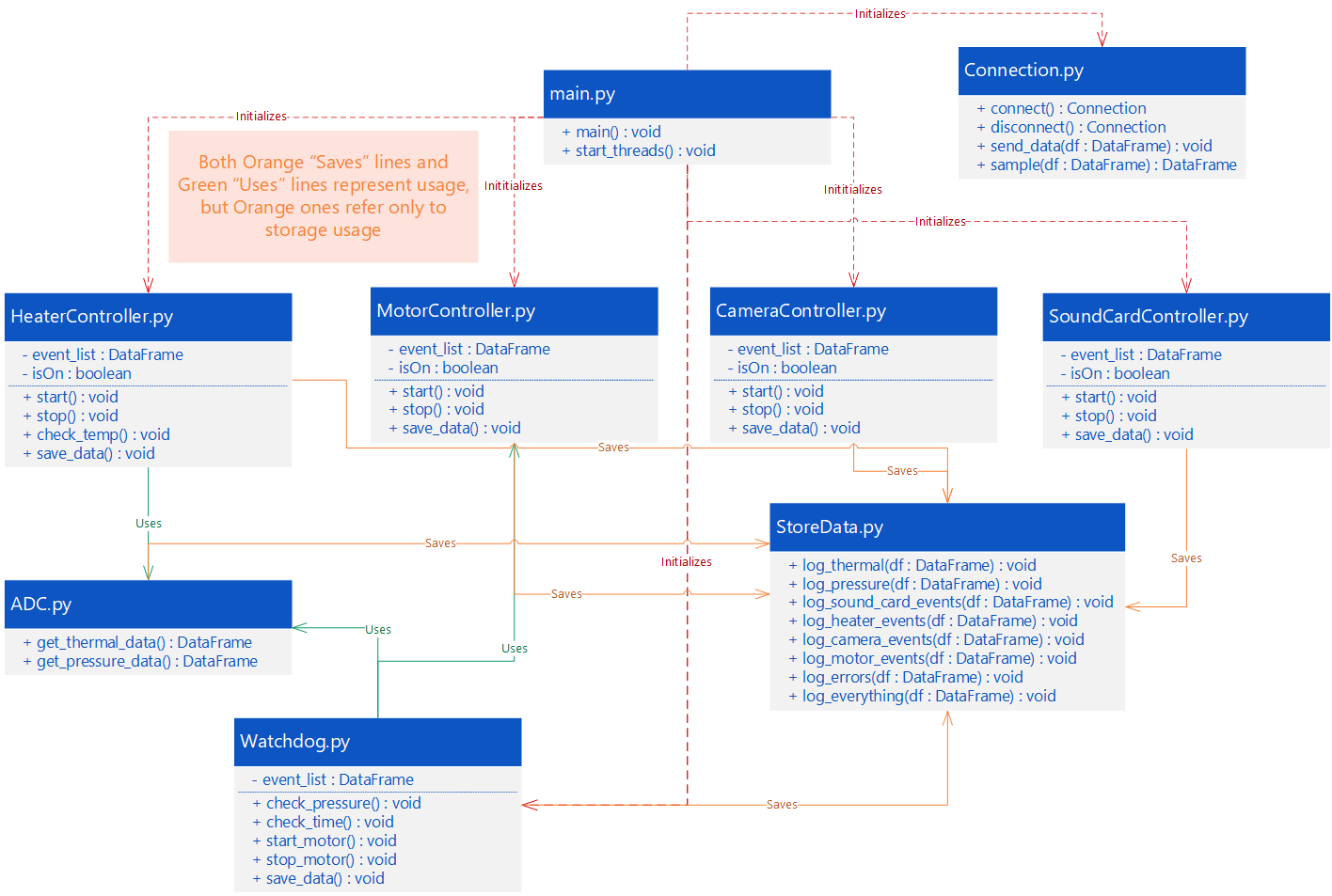
[*NUM\_OF\_IMG*] A simulation of the final data logged in the Raspberry Pi storage

**4.8.2.5 Process Flow**

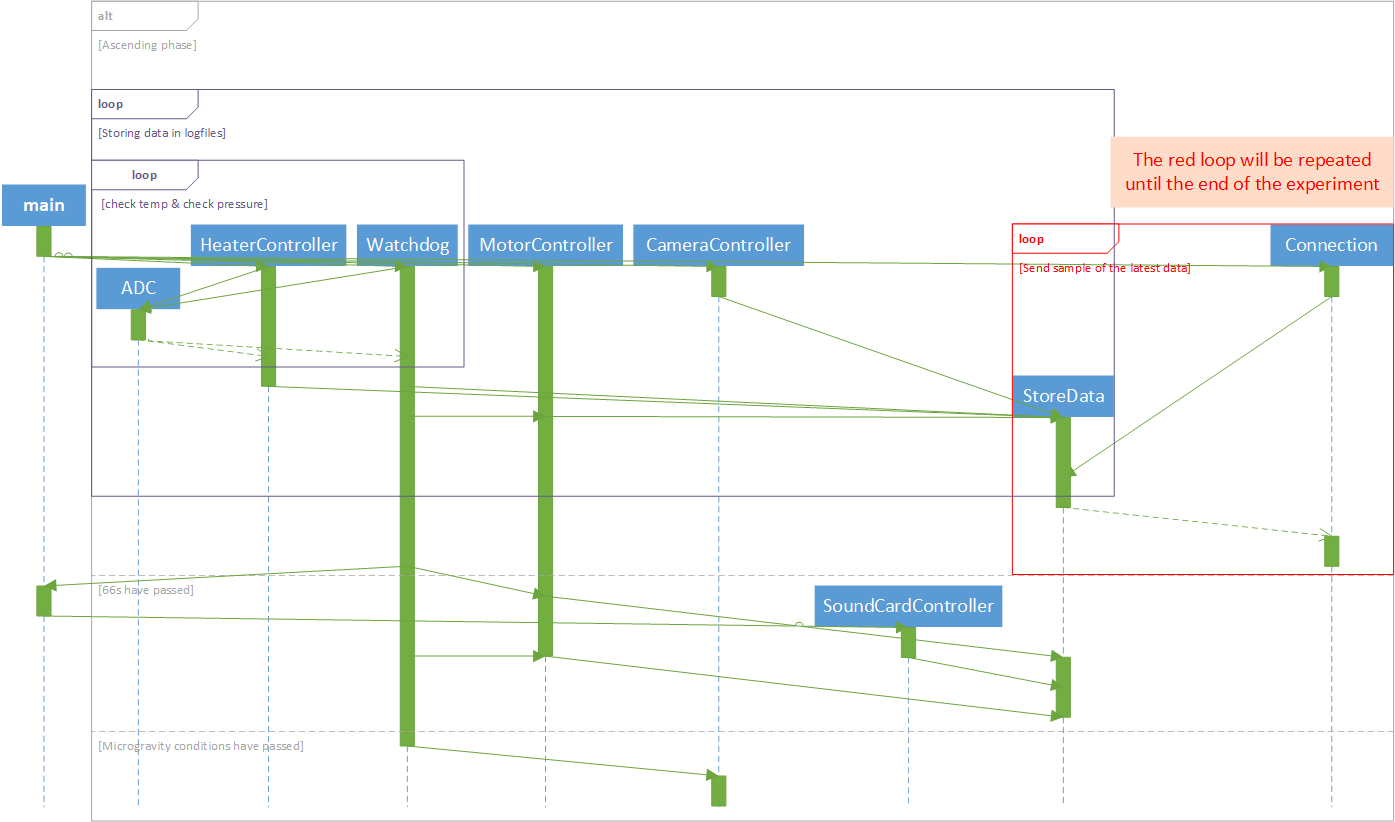
****

**4.8.2.6 Modularisation and pseudo code**

The code of the experiment is going to be separated into several different modules which can be seen in the *[NUM\_OF\_IMG]* diagram. The `main.py` will initialize all the component Controllers, the Connection and the `Watchdog.py` in different threads. These threads will be able to draw and message data, when needed, to each other and they will all use the `StoreData.py` module in order to save their data in the SD card. `Connection.py` module will get a sample of this data from `StoreData.py` in order to send it to the ground station.

****

*[NUM\_OF\_IMG] UML of the relations between different modules*



*[NUM\_OF\_IMG] Sequence UML that explains the usage timeline of each module*

The UML Sequence diagram above describes the usage of each class mentioned before, according to the Timeline represented in the **4.8.2.5** paragraph

**4.8.3 Implementation**

Regarding the implementation, the programming language Python will be used, since it’s easy to write, very rich and it is compatible with all the tasks the Software subteam is called to work on. It is one of the main languages that are used for Data Analysis, the main language that Raspberry Pi uses and it can create very rich GUIs when used correctly. Other lower-level languages like C or Java might also be used in order to implement the Data Link protocols as seen in paragraph **4.8.2.3** but that remains to be defined after the STW session.

To aid the development, Mu, Spyder, Visual Studio Code (setup for Python) and PyCharm IDEs will be used each for different stages of the development as they offer different tools for debugging or projecting information.

The Libraries and Frameworks that are definitely going to be used are Anaconda, Django or Flask (to be defined when the research on GUI will be done), pandas, matplotlib, scipy and numpy. Some libraries, like the adafruit\_blinka, gpiozero, RPi.GPIO and spidev, will have to be studied before use in order to choose which one is better or more safe to use.

In terms of Operating Systems, Raspbian is going to be used for the Raspberry, so the knowledge of Linux kernels and Debian Linux is vital, and everything is going to be developed in Microsoft Windows.

## 7.1 Data Analysis Plan

* I-VED measurements will be plotted in Conductivity-Time diagrams.
* Direct analysis of the video captured.
* Analysis of specific frames of the video in the BubblesEdit software so as to extract the size distribution of the droplets.
* Comparison of the data from the two diagnostics.