The GNOME Experiment Manual (GNOME Book)

The Budker Group, Dr. Samer Afach

Updated on December 24, 2016

Preface

This document is intended to be a comprehensive book on the GNOME experiment. It is written by Samer Afach. If you have any questions and/or suggestions, please send an e-mail afach@uni-mainz.de.

Part I

The physics of the GNOME experiment

Chapter 1

The physics of the GNOME experiment

Articles, documents and presentations that discuss the physics of the GNOME experiment are:
http://dx.doi.org/10.1103/PhysRevLett.110.021803
http://dx.doi.org/10.1002/andp.201300061
https://indico.mitp.uni-mainz.de/conferenceDisplay.py?confId=32

Part II

How to be a part of the GNOME experiment - Manual

Chapter 2

The GNOME Data Standard

2.1 Introduction. For whom is this chapter?

This chapter is for those who would like to join GNOME, but would not like to use our default equipment. You're free to create the HDF5 files that you'll upload to the server. This might be necessary in case you don't have a magnetic field station. In case you do, you're also free to reinvent the wheel and redo everything we did for you. It's, however, very not recommended, and very risky. Remember that other people will have to use your data. The more it complies to the standard, the easier it becomes.

Anyway, the data is checked before being uploaded to the server. In case a file doesn't comply to the standard, it will be rejected. We apologize for this harsh method, but we don't have a choice.

If you're willing to use our acquisition box and software, please refer to the next chapters for more information on that.

2.2 Why have a standard?

The GNOME experiment consists of collecting data from multiple stations around the world. Each station consists of an atomic magnetometer of some sort. Other kinds of sensors can get involved, such as atomic interferometers and atomic clocks. The Budker AG has taken the responsibility of storing the data and providing backups for this data for a very long time that could be over 10 years. With all these stations and data arriving, there should be a method to ensure that all this data is accessible in an ordered fashion, such that checks can be done as easily and systematically as possible, and certain information can be read from the data in a systematic way that does not depend on the station. It would not make sense to have every station write the data their own way. This will make the analysis process orders of magnitude more difficult, which is a waste of resources. Good organization saves money and time and leads to less errors and more reliability.

2.3 Data structure and format

2.3.1 Data format

The standard format that was chosen for the GNOME experiment is the HDF5 format. The following are the reasons for choosing this data format:

1. Accessible from all programming languages. The HDF5 Group has libraries for every programming language and every scripting language
2. The data is stored in binary form, making diskspace consumption minimal
3. Highly organized, with possibility to use multidimensional data
4. One could add header and meta information, called "Attributes"
5. Data viewers are available and easy to use
6. HDF5 files consist of datasets, and one could add as many datasets as necessary, making it easy to initiate a standard without limiting the data that can be stored in the files

HDF5 files can be viewed using the program HDFView, which is available online for free.

2.3.2 Data directory and files

Data is identified by time of recording it; i.e., by year, month, day, hour, minute, second and millisecond. Every day's data is stored in one folder, and every file contains the data of one minute. If the data directory is D:\GNOMEData, then a file of the date February, 15th 2015 should be stored in the folder

D:\GNOMEData\2015\02\15\

For a file that stores data that starts at 15th of February, 2015, at time 13:45:20.123, where 13 is the hour, 45 is the minute, 20 is the second and 123 is the milliseconds, and the station username is mainz01, then the file name is

mainz01_20150215_134520.h5

Notice that the first numerical code indicates the date, and the second indicates the time in seconds precision. The upload/synchronization software looks for all the files in the directory, and reads its contents to decide the file name on the server. Therefore, it is necessary that the files contain valid information **from the inside**. The upload software does not care about the file name.

2.3.3 File contents

2.3.3.1 HDF5 capabilities

Before proceeding with describing the content of files, you should be familiar with what HDF5 files can do and what information they can take. If you're already familiar with this, you could skip it. The following are the components that create an HDF5 file

Datasets

A dataset in an HDF5 is simply an array of data. It can have any dimensions. It can be 1-dimensional, i.e., an array. It can be 2-dimensional, i.e., a table. It can be higher dimensional. Unless you're willing to use complex data structures, a dataset can take only one type. For example, a dataset can only have 32-bit integers. Or it can have only 64-bit floating point numbers (doubles). It can't have both. If you wish to save 2 arrays, each with different data types, then you should save them to two datasets.

Note: It is possible to have multiple datatypes in a single dataset; this is called a *compound dataset*. **Datasets in GNOME data are not allowed to be of compound datatypes.**

Attributes

Attributes are individual pieces of information that can be inserted either globally in an HDF5 file, or in datasets. You can think of them as header information. A single attribute can take any binary type (integers, floats, doubles, strings, etc...), and you can store as many attributes as you wish anywhere.

Groups

You can think of groups as directories in an HDF5 files. You can put a few datasets in a group. For GNOME, I find groups very unnecessary.

Playing around with HDF5 files

HDF5 files have a very steep learning curve, especially when used in low-level languages, such as C++. If you want to familiarize yourself with HDF5 files, you could use Python (the module is called h5py). I highly recommend this, as it'll teach you how this works, and will teach you what you can and cannot do. Examples are way more than you can imagine on the web.

2.3.3.2 Content description

The file contents have to fulfill certain requirements to be accepted for upload. Beside these requirements, you are completely free to add as many datasets and attributes as you feel necessary.

Every file to be uploaded requires minimal amount of information to be accepted. This information is both attributes and datasets.

Every file is expected to have data of length 60 seconds.

Must-exist global attributes

The following global attributes must exist in every HDF5 file you create

Attribute name	Type
DataModel	String
DefaultDataset	String
DefaultMainEquation	String
DefaultMainEquationVarName	String
DefaultMainEquationVersion	String

DataModel: This is a string that you should agree with the GNOME collaboration about. For example, the default magnetic field data model is called "MagneticField_Default". The data model defines what expectations should be made when reading the file.

DefaultDataset: The name of the default dataset. The default dataset contains the most important set of data, and contains also other attributes that are very important, such as the date and time of the data.

DefaultMainEquation: The concept of the equation is simply that a few datasets can be combined to construct a meaning physical quantity. This field contains the **name of the attribute in the default dataset** that contains the equation.

DefaultMainEquationVarName: The result of the equation, what would you call it? Magnetic field? Frequency? Whatever you call it, this name should go here. This name is less prioritized than an optional name that can be put in the equation itself. This will be discussed later in Sec. 2.3.4.

DefaultMainEquationVersion: This is 1.0, unless a newer version of the equation is used in the future.

Default dataset

Every file must contain a dataset that we call the default dataset. In it, the most important data can be put. The default dataset **can be only 1 or 2 dimensional**.

The default dataset contains primary information about the file and its data. The default dataset must contain the following attributes:

Attribute name	Type
Altitude	64-bit float
Latitude	64-bit float
Longitude	64-bit float
<main equation name>	String
Date	String
t0	String
t1	String

Altitude, Latitude and Longitude: The GPS coordinates of your station

<main equation name>: Don't take these triangular brackets literally. The name of this attribute must be equal to the value of the global attribute "DefaultMainEquation" that was mentioned before.

Date: the date of the starting time in the format yyyy/MM/dd. For the date Sep, 14 2015, this date value will be: 2015/09/14

t0: contains the time of the first point in the format hh:mm:ss.zzz, for the time 13 hours, 15 minutes, 17 seconds and 123 milliseconds, the correct string is 13:15:17.123

t1: contains the time of the expected t0 of the next file.

The "SanityChannel" dataset

The sanity channel is strictly one-dimensional, with sampling rate of 1 Hz. Hence, for a file with 60 seconds, only 60 data points should be there. Each data point is a Boolean value that represents the sanity of the data. Along with the data of the file, the sanity channel has the value True, if the data is sane/reliable/valid, and False otherwise, so if the data is corrupt/unreliable/unusable for any reason. Preferably, an automated set of electronics automatically determine this value objectively[1]. It is very not recommended to just use human judgment to decide the value of this channel. The GNOME collaboration appreciates handling data professionally as little human intervention as possible.

[1]The Fribourg group has designed a device that takes care of this. Please contact the GNOME collaboration for more details.

Every other dataset

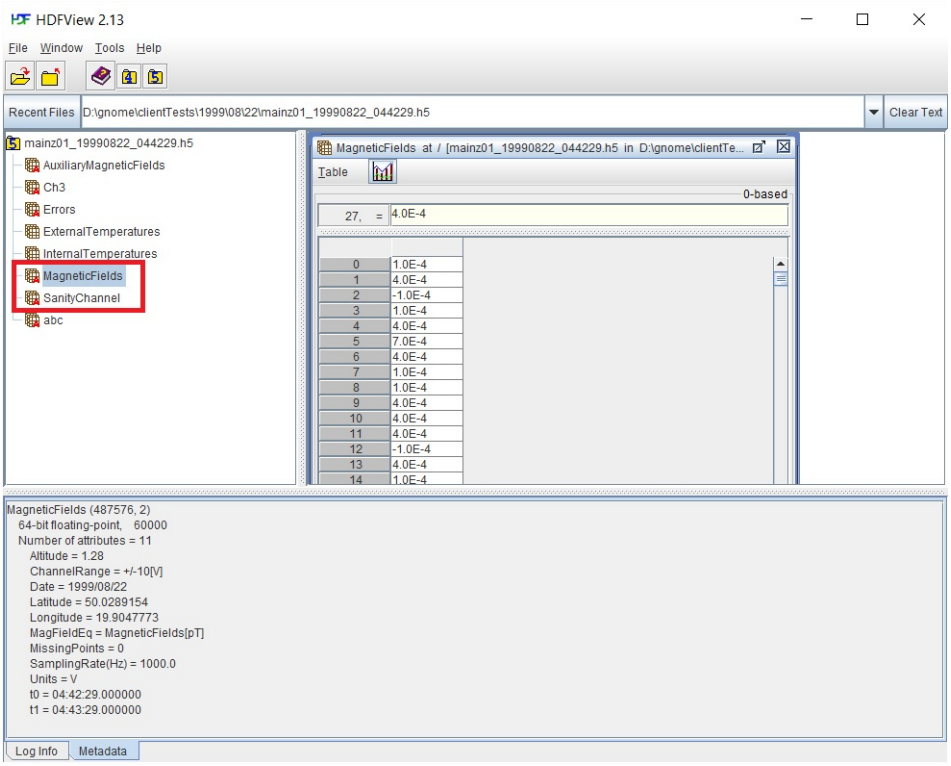
Other datasets that are expected to be included in the analysis (hence, primarily involved in the Main Equation) must be 1 or 2 dimensional, too. Any dataset that has higher dimensionality is not supported by the GNOME collaboration software, and cannot be plotted in the website, and cannot be involved in the standard data analysis. Please avoid using higher dimensional data, and preferably, discuss other options with the collaboration before proceeding.

Inside every dataset, these attributes have to be defined, be careful that attribute names **are case sensitive**

1. "SamplingRate(Hz)", which is a 64-bit or 32-bit floating point number that contains the sampling rate of the data
2. "Units", which is a string that contains the units of the raw data measured. It could be "V" for volts, or anything you like.

Datasets in GNOME data are not allowed to be of compound datatypes.

The following is a screenshot of HDF5 with a valid file. The file is viewed using HDFView (free software). On the top left are the datasets names, and on the top right is the default dataset, called in this case "MagneticFields". The buttom part shows the available attributes, which include all the information required.



2.3.4 Magnetic field equation

The time series from acquisition is saved as a column in a two dimensional data in datasets. Each dataset is expected to be one or two dimensional. Mostly, the variable (e.g., magnetic field) to be evaluated can be estimated by one channel by multiplying it by a constant factor. However, this is not generally the case. Some times, a combination of multiple channels is required to construct the magnetic field. Therefore, for the data analyst of the data, we created a method that can produce meaningful physical quantities without having to manipulate the raw data while acquiring the data.

The method consists of creating a function of all datasets, that will be mathematically used to create the magnetic field. For example, say we have datasets with names: "MagneticFields", "ChX" and "ChY". Each of these datasets consists of a time series with N points (or N rows). All the datasets involved **must have the same length**. We will create an equation that will use each point of these channels, i, that corresponds to time point t_i, to create the physical quantity we are interested in at time t_i. If we call this the magnetic field, then the magnetic fields are created using a function:

$$B_i = f_i\left(MagneticFields_i, ChX_i, ChY_i\right)$$

where B_i is the magnetic field calculated for the point i in time, and the parameters in the parenthesis are datasets names and are variables. An example of such equation:

```
MagneticFields/103+1.5*arctan(ChY/ChX)
```

or:

```
MagneticFields/3498.6211
```

Two-dimensional data

The previous format is valid for 1-dimensional data. If "ChX" and "ChY" are two-dimensional, we can do the following to select the first and second columns

```
MagneticFields/103+1.5*arctan(ChY[[0]]/ChX[[1]])
```

Which is a simple index operator. Notice the double square brackets, and notice also that columns start from 0.

Involving attributes in the equation

You could also add attributes to the equation. Say we have an attribute called "SamplingRate(Hz)" in the dataset "MagneticFields", and we want to multiply it by the last formula we have. What we do is:

```
${MagneticFields/SamplingRate(Hz)}*(MagneticFields/103+1.5*arctan(ChY[[0]]/ChX[[1]]))
```

So basically, we use the format ${dataset/attribute} to access an attribute. Keep in mind also that you can access global attributes with simply ${attribute}.

The final format of the equation

The equation that must be provided is not only the above text, but there's also another part that contains the units and an optional name of the quantity that was produced. This is how it should look like

```
MagneticFields/103+1.5*arctan(ChY[[0]]/ChX[[1]])["Magnetic field",pT]
```

Notice:
The first part is the equation that was discussed before. The second part, in single square brackets, has two parts separated by a comma:

1. The name of the quantity you constructed with this equation, notice the quotation marks. This name is optional. If you include it, then it overrides the name mentioned in the global attribute "Default-MainEquationVarName"
2. The units, which don't use quotation marks (with or without spaces, that doesn't matter). This is mandatory and must be included.

If you would not like to include the name, then do this:

```
MagneticFields/103+1.5*arctan(ChY[[0]]/ChX[[1]])[pT]
```

where in this case, the name will be taken from the global attribute "DefaultMainEquationVarName".

Multiple equations

You can use multiple equations together, which means that you have multiple physical quantities to extract from the data. To do this, separate them with "::", i.e., double colons. For example, to include this equation

```
MagneticFields*10.2["Magnetic field",pT]
```

with this equation:

```
arctan(ChY[[0]]/ChX[[0]])["Angle",Degrees]
```

Do this:

```
MagneticFields*10.2["Magnetic field",pT]::arctan(ChY[[0]]/ChX[[0]])["Angle",Degrees]
```

2.4 Testing your files before committing to their HDF5 format

The GNOME synchronizer (Sec. 3.3) has a test option for the files before uploading them. The testing requires internet connection because part of the data models is retrieved from the server. To test, choose the test option from the combo-box in the bottom, and click on test files.

Chapter 3

How to obtain and use GNOME Acquisition software

Introduction

This is a tutorial on how to install Python and required packages to run the software "GNOME Data Acquisition Software". If you're familiar with Python and you know how to install packages, you may ignore the instructions of the installation and proceed to the git cloning part.

3.1 Why is python used?

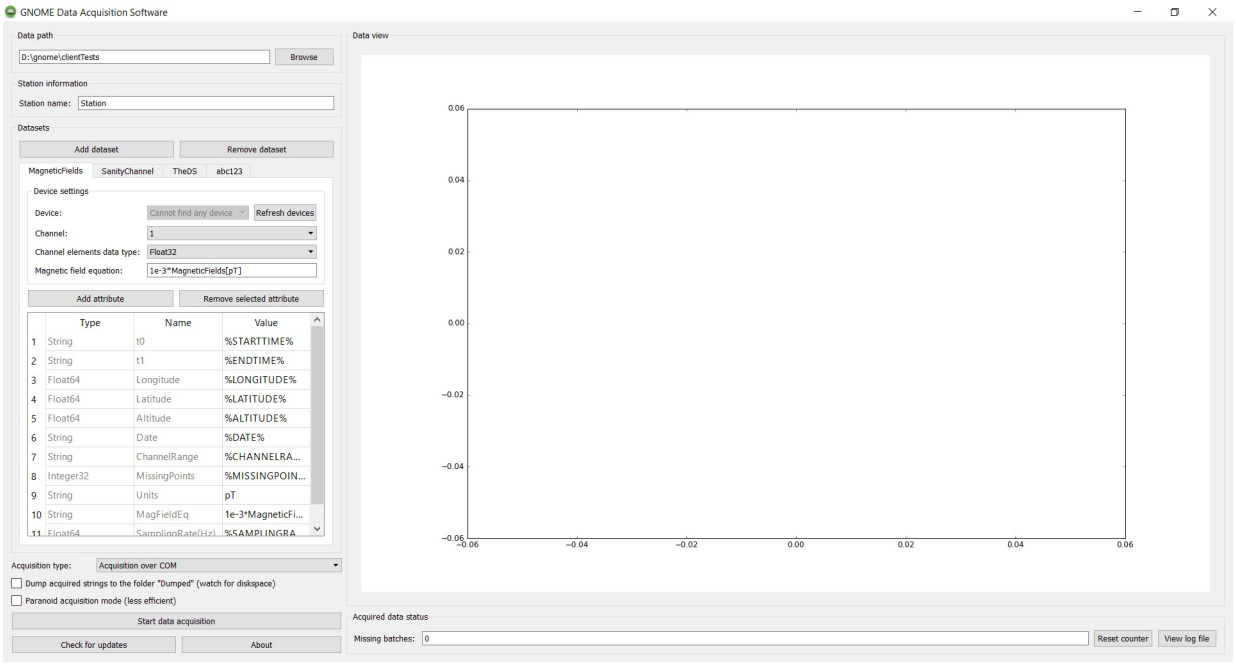
1. It is free and open-source
2. It is very widely used; help and references are everywhere
3. It is object-oriented, meaning that people design modules (or black-boxes) with functionality that can be added to the program
4. It is cross-platform (works on Windows, Linux and Mac) (Note: This tutorial is for Windows)
5. Easy to modify the program in case more users would like to introduce changes (please notify the Budker group in case you need to do changes, perhaps we add them to our version)
6. Tons of free libraries and tools available online for it, and tons of free libraries for scientific requirements and data analysis
7. Easy language and can be learned very fast

3.2 Getting the acquisition software

While the software is written in Python, an executable is created from it and packed in an installer. The source code is available upon request. The installer can be downloaded from the link
https://budker.uni-mainz.de/download/GNOMEAcqInstaller.exe
To install the acquisition software, double click on "GNOMEAcqInstaller.exe", and choose a place to install the software. Please keep in mind that it is recommended to install the acquisition software in a directory that can be modified without root access. If you must install it directly to drive C:, then this condition is not met. In that case, it is important that you run the software with admin rights; otherwise, it is not possible to save your configuration and it will be necessary to reconfigure the software every time it is executed.

3.2.1 Using the software

After the installation, a shortcut on desktop "GNOME Acquirer", which executes the program. The program looks like the follwing.



The program uses the structure of an HDF5 file for main options. The user adds datasets, and adds attributes to these datasets.

3.2.2 Default and unchangeable data

As was seen in Sec 2.3, there are mandatory datasets that have to exist, to make the data complaint to the GNOME Data Standard. Other than these, you are free to add as many datasets and attributes as you like. Any piece of information that helps data analysis, must be added. **It is also highly recommended** to discuss anything that may be common among all stations that was not considered yet.

The mandatory datasets, explained before in Sec 2.3, are the "MagneticFields" dataset and the "SanityChannel" dataset.

The "MagneticFields" dataset consists also of the Magnetic Field Equation (please refer to Sec 2.3 for more information). This should be modified within the attributes. The text field is only for easy viewing.

The "SanityChannel" dataset consists of two attributes/parameters that control the method, with which data is converted to Boolean data, as required (please refer to Sec 2.3 for more information). Expected to input is the a voltage, while the output is a Boolean value. Hence, the first parameter is the threshold value. The attribute is called "Threshold(V)". By default, if the voltage value is higher than the threshold value, the value True is assigned to the SanityChannel. Otherwise, the value False is assigned. The second parameter is whether the Boolean decisions mentioned have to be inverted/passed to a NOT gate. This attribute is called "InvertAfterThreshold", and is Boolean. If this parameter's value is set to True, then the SanityChannel's value will go through a NOT gate, before being saved.

3.2.3 Need more assistance?

For information about how the program is used, hover your mouse over something on the program, about which you need more information, and a "tooltip" will appear with more information.

Once the user clicks "Start data acquisition", the software checks for the sanity of the user's input. If the input is sane, then the configuration is saved, so that the user does not repeat the entered information at every execution of the program.

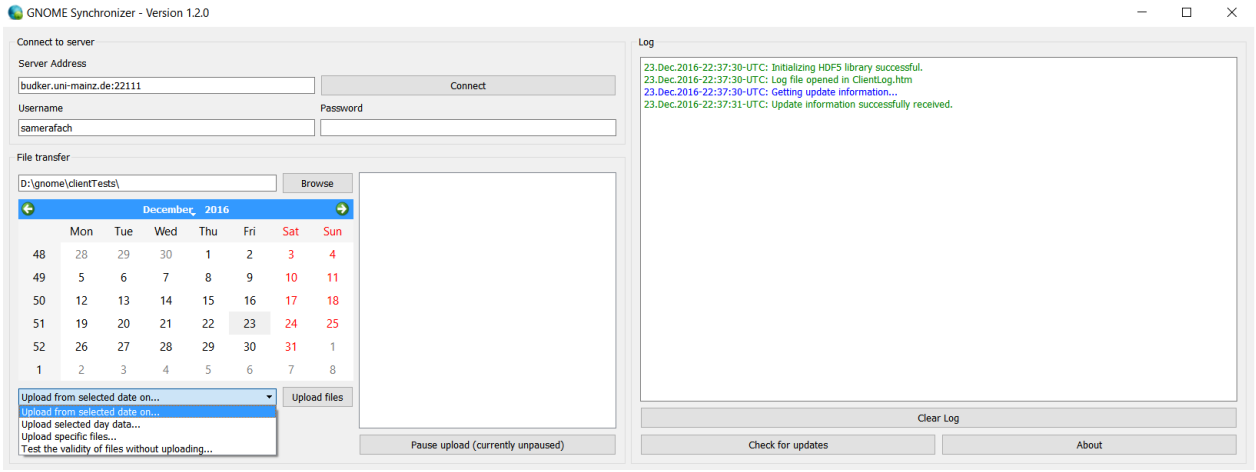
3.3 Upload software: GNOME Data Synchronizer

After acquiring the data from the device, it has to be uploaded to the GNOME Server in Mainz. Uploading the data is done through a software written by the Budker group. The software is called "GNOME Data Synchronizer". The software can be downloaded from the following link:

https://budker.uni-mainz.de/download/GNOMEClientGUI.exe

The program is written in C++ and is compiled statically, such that the executable works by itself. Download the file anywhere you wish on the computer. It is recommended to put the file in a folder, because it generates

a configuration file. The configuration file saves the user the trouble of reentering the information at every execution of the program.



The server to connect to is **budker.uni-mainz.de:22111**. The username and password can be obtained from the Budker group. Before connecting, the user should choose the directory where the data is stored. If the acquisition software of Sec. 3 is used, then this is the data path of that program. Otherwise, this directory should contain directories that follow the GNOME data standard, as explained in Chapter 2. In this directory, the folders with years should exist.

Once the user is connected to the server, 3 modes for upload can be chose (from the combo-box shown in the picture):

1. "Upload from selected date on...": This option will upload all the data available in the data directory selected starting from the date selected on the calendar until the present day.

2. "Upload selected day data...": This option will only upload the data of the selected day on the calendar.

3. "Upload a specific file...": This option will open a file dialog, where the user can choose the files to upload.

When file is to be uploaded, it is put on the list on the write, which is the queue for upload.

Part III

Technical details

Chapter 4

The data acquisition software

4.1 Description of the mechanism of data acquisition from the GPS Box provided by Szymon Pustelny

The GPS data acquisition box provides the data in ASCII format through USB in serial. It is possible to connect to the box using standard serial communication with any operating system or through NI VISA. The data acquisition software (Ch. 3) uses standard serial communication to receive the data from the box. The following is an example of the data provided by the box

```
@Header
Date: 2014.12.31
Time: 10.02.01
Week number: 1825
Time of week: 295321
UTC offset: 16
GPS time, GPS PPS, Time from GPS
Receiver mode: 7
Self survey progress: 100%
Warnings
Decoding status: 0x00
Doing fixes Temperature internal [C]: 34.78
Temperature external [C]: --
Latitude [deg]: 50.0287818
Longitude [deg]: 19.9056099
Altitude [m]: 259.13
@Data
Ch1 +/-10 [V], Ch2 +/-10 [V], Ch3 off, Ch4 off
0.0016   0.0012
0.0010   0.0015
0.0010   0.0012
0.0010   0.0012
0.0013   0.0021
0.0010   0.0009
0.0013   0.0012
0.0013   0.0009
0.0010   0.0012
...
0.0013   0.0012
0.0013   0.0009
@Magnetic
X Y Z +/-130 [uT]
-- -- --
-- -- --
-- -- --
...
-- -- --
@End
```

This batch is repeated every second. Triple dots mean there are points in between. The number of given points defines the sampling rate of the system. The acquisition software receives this stream of data, and parses it, and writes it to an HDF5 according to the format discussed in chapter 2.

4.2 Parameter estimation and acquisition algorithms

4.2.1 Storing data batches and accumulating multiple batches

Every batch is stored in an ofject of a class called "DataBatch". The class contains variables for all the information that can be extracted from a single batch. Every batch is saved to a single instance/object of the class DataBatch. After parsing and storing all possible information from the batch in an object, this object is sent to another object called of a class "DataCollected". The class contains a function that takes a DataBatch object, and merges/appends its content to itself, making the process of accumulating data flexible and easily modifiable. For example, in the append function of DataCollected, the data points is concatenated, and the number of missing points is added, and error messages are added if they do not already exist.

4.2.2 Parsing beginning and ending of data

The acquisition software knows the beginning and the ending of the data (and even different parts of the acquired stream using the keywords provided in the stream as shown above. These are mainly the keywords that start with an "@", such as "@Data". Initially, only "@Header" and "@End" are located in the stream to determine a single batch of the data. When found, the data in between is isolated for parsing. The part between @Header and @Data is the header information, which is part on first-words basis.

4.2.3 Estimation of the sampling rate

The program uses a closest-minimum algorithm starting from the known available sampling rates to decide the sampling rate of the system. For example, if the system accepts the following sampling rates: $f_S = \{20, 50, 100, 500, 1000\}$ Hz, then the number of point recorded, N, is subtracted from f_S, with absolute value, i.e., $k = |N - f_S|$. The minimum of the resulting list is located, which would correspond to the sampling rate of the system. If the minimum is zero, this means that no points are lost. If the minimum is not zero, then this number would correspond to the number of points lost during the acquisition.

Of course, this algorithm works only under the assumption that the minimum sampling rate is much higher than the highest number of possible lost points. This is why sampling rates $f_S < 20$ are rejected in the software.

4.2.4 Errors and warnings

If errors or warnings exist or any predictable problem of any kind is found, they are written in a dataset called "Errors" in the attributes.

4.2.5 Writing data points

Data points are considered to start from @Data and end at @Magnetic. The first after data is parsed by stripping spaces and parsing by commas. This gives the ranges of the that are using by the acquisition box.

The data points are taken after the first line. The data is read line by line. Every line is parsed for white-spaces and is saved in an array, and is then transposed to create lists of arrays, where every array contains all the data from a single second. The array is then cast using NumPy to the binary type the user requested.

The acquisition software writes an integer number of batches to the data files. For example, to write one minute of data to a file, and assuming that every batch contains one second of data, 60 batches are written to a file. This will accumulate the number of missing points collected when calculating the sampling rate as discussed in Sec. 4.2.3.

4.2.6 Time stamping

As can be seen from the data format of the data provided by the box, time stamps are only provided once per second. Only two time stamps are provided per data file. The first time stamp is the time stamp of the first batch. The second time stamp is the time stamp of the first data point in the next file. The second time stamp, t_2, is calculated using the formula:

$$t_2 = t_1 + \text{number of collected batches} \times \text{number of batches per second}$$

4.3 Communication problems and how the acquisition deals with data gaps

The acquisition from the DM Technologies box is not perfect and has issues, unfortunately. It's still not settled whether the issues are caused by software or hardware. The issues include ASCII chunks missing from the data stream. The missing data causes a failure in data parsing, and it is, unfortunately, not possible to do a general fix for any amount of missing data, as it could be missing literally from anywhere, including the header, data points, auxiliary magnetic field or even the parts that separate batches making them distinguishable, such as @Header and @End. For this reason, the acquisition software follows a very strict mechanism to ensure that all data is intact.

To resolve this issue, the software was modified to detect batches that would have any kind of problems. If a batch has a problem, the software considers it corrupt and disqualifies it from the main data stream that will be uploaded. Let's call this process Fixing Data Gaps (FDG).

The FDG process comes with a price that is a little more expensive than simply throwing away the corrupt batch alone. To explain this, we have to remember that every data file should have 60 seconds of data. Say that batch number $n = 30$ has a problem (where $n \in [1, 60]$), what does the program do with the other 29 intact batches? The program also throws them away. This is done because the highest priority in the streamed data is:

1. Data per file must be contiguous in time
2. Data per file must contain 60 seconds

Assuming that the frequency of this event (data loss) is not high, this loss is still considered acceptable. Keep in mind that experiments do not normally run 24/7, and issues do happen in them. These problems include maintenance. Such issues in experiments last hours to fix and restore operation; hence, a loss that is less than a minute is acceptable.

Although the data is disqualified for streaming, the program still stores the data on the acquisition computer locally. They are stored in a directory called "CorruptData". The storage happens in two parts. If a failure happens at batch n, the data from batch number 1 to batch number $n - 1$ is stored in an HDF5 file, and batch number n (the corrupt batch) is stored in a text file without parsing it, since the parsing does not work with corrupt batches. The file names correspond to the standard file naming. A special case is the case when the corrupt batch has $n = 1$, i.e., the first batch is corrupted. In that case, the date and time is considered unknown, since parsing the batch is not possible, and no information is available about it. Hence, the file is stored only in 1 text file with a name that contains the UTC time read from the computer doing the acquisition.