The Collation and Graphical Representation of Magnetometer Data to Track Solar Activity Using a Web-Based Application

Final Report for CS39440 Major Project

Author: Oliver Roy Thomas Earl (ole4@aber.ac.uk)
Supervisor: Mr. Dave Price (dap@aber.ac.uk)

8th May 2018

Version 1.2 (Release)

This report is submitted as partial fulfilment of a BSc degree in Computer Science with German (G4R2)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, United Kingdom

Declaration of Originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work, I understand and agree to abide by the University's regulations governing these issues.

Name Mr. Oliver R.T. Earl

Date 24/04/2018

Consent to share this work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Department of Computer Science, Aberystwyth University.

Name Mr. Oliver R.T. Earl

Date 24/04/2018

Acknowledgements

I dedicate this work to my Mum, to my Dad, to my Nan, to my brother Thomas, to my sister Phoebe, to my three pug-dogs Toby, Toffee, and Luna, and naturally to the farthest reaches of my extended family.

I am grateful to academic and technical staff from the Institute of Mathematics, Physics, and Computer Science at Aberystwyth University for taking the time out to talk to me about my project, most notably regarding retrieving measurements from amateur radio-based equipment and the magnetometer itself.

Additionally, I am thankful for the patience and understanding of academic staff from both the Department of Computer Science and the Department of Modern Languages at Aberystwyth University as I attempted to juggle both major and minor disciplines of my undergraduate degree over these last few years, especially when both my appearance and outward behaviour were oftentimes dictated by a combination of sleep deprivation, and perhaps slightly excessive caffeine consumption.

Furthermore, I wish to express my sincerest gratitude to Dave Price who enthusiastically agreed to be my supervisor for this Major Project. Despite my stress, which was sometimes overwhelming, he was incredibly approachable and reassuring, a refreshing source of encouragement, particularly towards the end, and granted me a large amount of flexibility with this endeavour.

Thank you to my colleagues at Aberystwyth University, Aberystwyth University Students' Union, Aberystwyth Nightline, and the Nightline Association for their pertinacious support and sense of purpose that they have bestowed upon me during my time as an undergraduate.

Finally, I wish to extend my deepest and most heartfelt thanks to my beloved friends cheering me on, or at least photoshopping my face onto memes, either here with me in Wales or wherever they might be in the world.

Thank you very much.

Vielen herzlichen Dank.

Diolch yn fawr iawn.

Abstract

The Institute for Mathematics, Physics, and Computer Science (IMPACS) at Aberystwyth University make use of an FGM-3 magnetometer (Appendix D), currently based, at this time of writing, on Frongoch Farm in Aberystwyth, free of any electromagnetic interference. The purpose of the instrument is to periodically take a measurement of the Earth's magnetic field – data that can be analysed by researchers and students during solar events and disturbances such as solar flares and coronal mass ejections, to explore the impact these events may have on the Earth's magnetosphere [1].

The aim of this project is to produce a web-based application that can retrieve data from the remotely-hosted magnetometer instrument and make it more readily available to the aforementioned students and staff, both in the form of a web-based application where data can be explored and graphically represented, as well as programmatically, by the use of an API (Application Programming Interface) [2] capable of converting and returning desired data into a significantly more processible form, such as JSON. (JavaScript Object Notation) [3]

This is a far preferable outcome to the current process of data retrieval: consisting of manually connecting to a file share hosted by the computer directly linked to the magnetometer, retrieving unlabelled CSV (Comma Separated Value) files, and manually extracting the data. For second year Physics undergraduate students, who often undertake assignments involving this data (Appendix C), this is incredibly timesaving.

The web application, coined *Borealis*, can establish a connection to the computer, and retrieve new entries from the magnetometer each day when the log is complete. This data is then refined, collated, and stored in a relational database. While the core application is complete and functional, additional functionality involving connectivity to further data sources were originally planned and could be implemented in further iterations of the software.

Keywords: magnetometer, magnetosphere, magnetic field, solar activity, web application, Chart.js, API, graphing

Contents

Table of Figures

1. Background, Analysis & Process

1.1. Background

1.1.1. Introduction

The Institute of Mathematics, Physics, and Computer Science (henceforth IMPACS) have installed a system, consisting of a FGM-3 magnetometer sensor (Appendix D) connected to a Windows workstation in an isolated area of Frongoch Farm, Aberystwyth - off-campus and away from any sources of electromagnetic interference that could affect accuracy. The magnetometer takes measurements approximately every minute – recording data to a CSV (Comma Separated Value) file containing a timestamp, a measurement of the Earth's magnetosphere, and the temperature of the instrument in Celsius. This type of timestamp is, according to D. Langstaff: 'A timestamp indicating the number of seconds elapsed since 12:00am Friday, January 1st, 1904.' (Appendix C). This makes them out to be LabVIEW format timestamps (also notably used by the Apple HFS+ (Hierarchical File System) filesystem) since they use a different epoch (reference date; era) to the more common UNIX format. [4]

The readings taken by the magnetometer are used by researchers and students of IMPACS primarily for following patterns in the Earth's magnetosphere in response to solar events or disturbances, particularly solar flares [5] – defined by NASA as a 'sudden, rapid, and intense variation of brightness.' [6] Similar events of note are solar eclipses, and importantly coronal mass ejections – which are described by the National Oceanic and Atmospheric Administration as a 'large expulsion of plasma and magnetic field from the Sun's corona.' [7]

The aforementioned researchers and students are currently required to manually establish a connection to the computer, running Windows 10, directly linked to the FGM-3 by means of mounting a hosted SMB (Server Message Block – a network protocol used for sharing files and printers [8]) share that software running on the computer saves the previously described CSV data files. There is currently no alternative means to programmatically retrieve data. As demonstrated in Appendix C, IMPACS undergraduates are routinely tasked with retrieving data and graphing their findings by hand as part of their learning.

1.1.2. Personal Motivation

When it became time to decide upon a Major Project topic, two criteria were used to govern the consideration process.

The first is that the project topic would be interesting. Whilst this might be in the eyes of many an obvious and almost mandatory criterion, many undergraduate students both within this year group and of previous years often undertake projects that they believe will highly benefit their academic degree, or their career path, with no genuine interest for the field their topic lies in.

It was of tremendous importance for a topic to be interesting. As noted by R. Boyd in the context of the Scrum development methodology: 'If teammates aren't enthusiastic, no process or methodology will help.' [9]

The second criterion is that the topic has the potential to have an impact following its completion, regardless whether it takes the form of a software engineering, or research project. It is intended for this project to potentially be of use to academic staff and students within IMPACS, or to provide the foundation for future research involving the use of magnetometers and additional data sources to take and graph readings, and to determine visual trends in the Earth's magnetosphere.

The flexibility of the project and broadness of the project area was incredibly attractive. Since there were no predetermined languages, frameworks, or use cases specifically mandated for this project, besides namely the use of the FGM-3 magnetometer (Appendix D) and providing a means to make the data collected by the instrument (and any additional data sources) more accessible or manipulated in a way that would prove useful to those who wish to use it.

To summarise, the project area demonstrated a great area of flexibility and allowed for a great deal of self-determination in what software was to be delivered, as well as the functional requirements constituting a finished piece of software. Furthermore, it pertains to an incredibly fascinating area of science, without also having a strong Mathematics or Physics background as a prerequisite.

1.2. Project Analysis

1.2.1. Project Aims

The primary aim of this project is to produce a web-based application that will retrieve measurement data from the FGM-3 magnetometer and make it available to researchers and students of IMPACS both in the form of a web interface – providing data in the form of graphs and tables that can be explored in detail, as well as programmatically, by means of an API that can be requested to return data in the form of JSON (JavaScript Object Notation [3]) and/or XML (Extensible Markup Language [10]) that would be significantly easier to work with in a scientific or data processing context.

As the application is only intended for use internally within the institute, it should be built in as lightweight a form as possible, without the need for heavy frameworks and tools, so that it may run on a web server, such as on the Users subdomain (commonly referred to as 'Central') - a shared web space run by Aberystwyth University, that may have limited resources and potentially critical web hosting features disabled, or on a virtual machine where access to specific technologies and runtimes may not be guaranteed.

Additionally, as the application would be used within the institute, it will be necessary for the application to have localisation support – and be capable of switching between both English and Welsh language mediums in accordance with the Welsh Language Scheme of Aberystwyth University, which in turn is written in accordance to the Welsh Language Act of 1993. [11]

Although not core intended functionality, it would be highly beneficial if the program is capable of retrieving, collating, and graphing data from other sources that could be displayed alongside and/or compared with that retrieved from the magnetometer, for academic and educational purposes. This data should also be made available via the API. Potential candidates include online databases containing readings of magnetosphere or ionosphere strength or activity – one such example is the Geomagnetic Data Master Catalogue made available by the World Data Centre for Geomagnetism. (Edinburgh) [12]

Another possibility was the inclusion of data from amateur radio-based equipment. The phenomena of the Earth's ionosphere being affected by geomagnetic disturbances such as solar flares are well-documented, in particular such events can allow for very low frequency (VLF) radio waves to reflect off of (or 'bounce') from the consequently highly ionised ionospheric layers. [13] Data could be collected by monitoring the latent energy on a radio antenna, by monitoring a spectrum of frequencies, or by monitoring a specific frequency, and correlating the data with that collected from the magnetometer to determine trends and whether similar behaviours can be observed with HF (High Frequency) and VHF (Very High Frequency) radio waves and frequencies. The software involved would likely come from the open source GNU Radio (GNU's Not UNIX) suite. [14]

1.2.2. Foreseen Constraints

As it is impossible to know fully where the web application could be run, engineering the software to run with a minimal number of prerequisites as possible as previously stated is an intelligent design choice. However, this will have implications when determining programming languages and frameworks that can be considered for developing the application with.

In the case of Central, it is not possible to enable the use of `mod_rewrite`, a URL (Unicode Resource Locator) rule-based rewriting engine module used by the Apache 2.x web server software, [15] which typically is enabled at the directory level using `.htaccess` configuration files. [15] The significance of this, is that it is a prerequisite for many commonly used web frameworks, such as the very popular Laravel framework for PHP (PHP Hypertext Preprocessor) – workarounds exist, but issues may crop-up with routing functionality. [16]

This functionality is supposedly disabled for security reasons. It could be speculated that the decision behind this specific configuration was due to the sheer number of content management system installations, such as the popular WordPress, for blogging and easy to modify websites. While convenient, abandoned

or otherwise outdated, insecure installations can pose significant security risks to the server [17] and are commonly exploited by malware looking to hijack the Apache rewrite engine to propagate spam; [18] with this functionality disabled, the risk is partially mitigated.

The primary challenge that will be encountered during the development of the software lies within the core functionality of the software – the retrieval of data from the magnetometer. Previously as evidenced by Appendix C, it was previously possible to retrieve data by means of FTP (File Transfer Protocol) but following a complete overhaul of the adjoined computer system, including a fresh installation of the Microsoft Windows 10 operating system to replace the archaic Windows XP configuration, this functionality was superseded by SMB filesharing. While this proves more user friendly for those possibly intimidated by accessing files via FTP, since SMB shares are more easily mounted through the Microsoft Windows File Explorer, [19] it comes at a cost for programmatic access. Whilst most web languages have some form of support for the FTP protocol [20], functionality for establishing connections to SMB shares is not included and would require a third-party library if one exists, or additional programming.

The computer connected to the magnetometer will be on the university network, and therefore behind a security firewall – completely inaccessible to those without VPN (Virtual Private Network) access, or to computers that are within this network. This creates further necessity that the program is run on Central, or on a computer or virtual machine within the network and cannot be stored on a third-party hosting service or PaaS (Platform as a Service) configuration.

Moreover, retrieving data from a computer connected to a radio antenna, should that additional functionality be implemented, would pose a similar challenge. Unless the computer is connected to the network with a comparable file sharing system set up, a means of connecting to and downloading data from it would be required. Alternatively, it was informally suggested by academic staff within the Department of Physics, that a script or small application could be set up that would run on the computer and send data to the web application, such as by means of the HTTP (Hypertext Transfer Protocol) POST request method. Receiving data in this manner however would require some form of token authentication system to ensure data received is authentic and does not pose a security threat to the application and its underlying system. [21]

1.3. Process Methodology

1.3.1. Initial Plans and Scrum

Due to experience working as a member of a Distributed Scrum team in industry, resultantly fostering a familiarity and comfort with the agile methodology, Scrum was originally intended to be the procedure utilised for this project. Scrum is an agile methodology, as described by G. Moer of Microsoft specifically as 'a framework used by teams to manage their work' and that it consists of an iterative

lifecycle characterised by fixed time-periods known as 'sprints'. The list of deliverable items is stored on the 'product backlog' and what is worked on is determined at the beginning of each sprint. Members of the team communicate with each other daily in a 'Scrum meeting or stand-up'. [22] Scrum is so named as its approach differs greatly to the traditional 'Waterfall' style methodologies that at the time dominated software engineering; an agile approach that how like in rugby, 'the ball gets passed within the team as it moves as a unit up the field.' [23] Distributed Scrum is simply Scrum where development teams are geographically dispersed. [24]

While a powerful and popular agile methodology, Scrum did not feel like the right fit for this project. Scrum, like many other related methodologies place a great deal of emphasis on collaboration and communication with other members of a development team to maximise 'agility', the ability to embrace inevitable change and produce deliverables to their deadlines. R. Kasturi reminds us that one of the key phrases of the Agile Manifesto is 'Individuals and interactions over processes and tools', [25] further reaffirming this importance of working together.

As this project involved working independently, there would be nothing besides the methodology itself to ensure 'discipline' – ensuring adherence to the agile process and not simply reverting to 'hacking' – throwing together code in the shortest amount of time and neglecting important processes like code reviews. Scrum teams even include such a role for ensuring the process is adhered to and that the team operates in the most efficient way possible, the Scrum Master, defined by F. Attanasio as the 'framework custodian.' [26] Furthermore, Scrum places a great deal of emphasis on customer satisfaction in the form of repeated software delivery. As this is not a priority for this project, the benefits from fixed iterations are mostly redundant. Acknowledging from experience the intrinsic difficulties of remaining disciplined in software development, chiefly whilst developing solo, a more flexible approach that focused more on functionality as opposed to fixed iterations and teamwork was necessary.

1.3.2. Migration to Feature-Driven Development

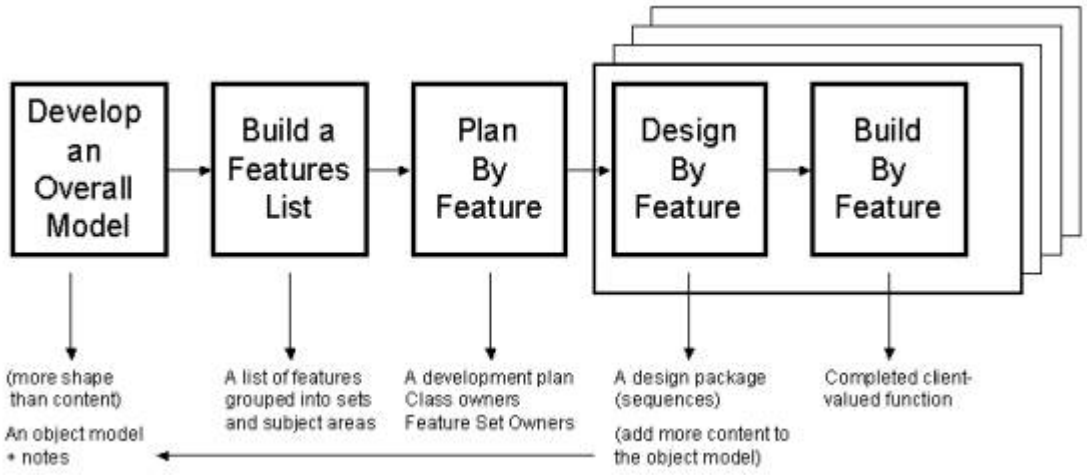


Figure 1-1 Feature-Driven Development Process Model [27]

Feature-Driven Development was decided upon as the agile methodology to be used in this project. As the requirements for the project are well understood and that it would not prove problematic attempting to develop an overall model at the start of development, FDD appeared to be the appropriate middle-ground between discipline and agility – traditional plan-heavy models such as Waterfall and agile methodologies like Extreme Programming. The only major alteration to the methodology however is that all major and supporting roles were taken on by a single person. [28] The five processes outlined in Figure 1.2 are as follows:

1: *Develop an overall model*
This primarily consists of class and sequence diagrams, often referred to as a 'high-level walkthrough' of the system. [29]

2: *Build a features list*
Features for the project are expressed in the form of <action> <result> <object>. These are broken down into smaller features if the actions would take a significant amount of time. [29]

3: *Plan by feature*
This involves the order in which features are developed, but since the project is being developed by a single developer, there is no need to distribute items to developers to maximise development efficiency and to make use of developers' strengths. [29]

4: *Design by feature*
The overall model is improved with the additional information gained so far. This also involves producing additional sequence diagrams for individual features. [29]

5: *Build by feature*
The implementation of classes and their underlying methods and functionality, carrying out unit tests where possible (substituted with manual testing where tests were not created) and pushing to the project repository. [29]

2. Design

2.1. Programming Languages, Frameworks, and Libraries

2.1.1. Introduction
There are numerous programming languages that are well-suited for the task of providing a web-based interface and graphing tool, interfacing with and retrieving data from the magnetometer, (also potentially additional sources) and an API to which requests can be sent.

2.1.2. PHP
PHP is one of the oldest server-side open source programming languages for the web, originally created in 1994 by Rasmus Lerdorf in the C programming language. [30] Since then it has grown dramatically and is used on more than hundreds of millions of domains across the Internet. Most notably it is used for the popular content management system WordPress, and countless web frameworks such as CodeIgniter and Laravel. The programming language offers an incredible degree of flexibility thanks to its large number of built-in functions that usually do not require additional modules to be installed [31], providing the language with a stellar quantity of possible use cases, ranging from small command line tools and dynamic webpages, to enterprise level web applications and RESTful (Representational State Transfer) services of which can be written in a procedural or object-oriented style.

While the programming language is in its seventh major revision, only the latest version of PHP 5.6 is available on Central. While this poses no major security threat at this time of writing as there is continued support for this iteration of the language until January 2019, [32] it nevertheless poses a strong limiting factor in what frameworks can be used to develop the application.

A strong example is the Laravel framework; incredibly well-suited to this task due in part to its MVC (Model-View-Controller) [33] architectural pattern. Tasks such as fetching data from varied sources can be implemented using services, processed, and saved using models and controllers, and so forth. [34] Unfortunately, it requires PHP 7.x.x to run.

2.1.3. Python
The Python scripting language created in the early 1990s by Guido van Rossum from the Netherlands [35] is the language of choice for many scientific and data processing applications thanks to being easy to learn, powerful, lightweight, and yet have access to countless libraries and frameworks to further extend

functionality. Python is so popular that it came first in the IEEE Spectrum Top Programming Languages of 2017 survey. [36] The most prevalent backend framework is Django, that in likeness to many other web frameworks, makes use of the MVC architectural pattern. [37]

2.1.4. Node.js

One of the fastest growing backend technologies is Node.js, an open source environment based on the Google Chrome V8 JavaScript engine that executes JavaScript on the server-side. While JavaScript as we know it has existed since the 90s, Node.js itself is a far newer technology, having made its debut in 2009. [38] As the usage of Node.js allows for JavaScript to be on the backend in addition to the frontend of a web application, quite often in the form of a MEAN (MongoDB, Express, Angular, Node.js) stack of technologies, this combination permits for a high-performance system with a wealth of opportunities for code reuse and an adherence to the DRY (Don't Repeat Yourself) principle. [39]

2.1.5. Ruby

Ruby is a programming language developed in 1995 by Japanese software engineer Yukihiro 'Matz' Matsumoto, who wanted to produce a programming language that placed emphasis on a natural, beautiful syntax, as opposed to a simple one. Rails is a rapid web framework built for Ruby in 2003 by David Hansson – it too utilises an MVC architectural pattern. [40] In likeness to Django and Laravel, Rails is perfectly suited for developing RESTful services and enterprise level applications. Noteworthy examples of apps developed in Rails include pre-2009 iterations of the social networking site Twitter, and the Three Rings volunteer management suite [41].

2.1.6. Miscellaneous Technologies

While not under a substantial amount of consideration, it is important to note the existence and popularity of numerous other technologies for the web, including Java, Microsoft ASP.net, (Active Server Pages) Perl, and ColdFusion, each with their own strengths and unique selling points. [31]

2.1.7. Frontend Languages and Frameworks

Decision making and research for the frontend of the web application is more straightforward. The most complete graphing/charting JavaScript library is Chart.js, featuring full responsive design support, multiple chart types and makes use of the HTML5 (Hypertext Markup Language) canvas functionality for maximum cross-browser support. [42] This will serve as the library that powers the program's data graphing functionality. There exist numerous options for HTML and CSS (Cascading Style Sheets) frontend frameworks that will govern the appearance and styling of the web application – the most popular candidates of which are Twitter's Bootstrap, and Zurb's Foundation. Due to multiple years of experience designing websites with the former, this was opted as the ultimate choice, although either would have proven more than satisfactory. Both frontend frameworks require the use of the jQuery JavaScript library to facilitate interaction with the DOM. (Document Object Model) [42] [43]

2.1.8. Final Decision

The ultimate decision to use PHP for development was taken due to two primary factors. The first of which is familiarity; with around fourteen years of experience scripting with PHP it abolishes the potentially lengthy process of familiarisation - not just in terms of language syntax but also coding standards, development environment, and workflow tools such as automated testing and dependency management. It could be argued that other languages, particularly Python, are more suited to the project's aims due to the wide availability of data processing libraries and pre-existing widespread usage as a scientific scripting language, but on the other hand K. Sierra argues that 'when we talk about "the best tool for the job" we should look not only at "best for the task" but also "best for those who must use it"'. [44]

The second deciding factor is compatibility. As previously explored, the software is likely to run on an IMPACS provided computer or virtual machine, or on Central. Central only supports PHP 5.6 and does not carry runtimes for any of the other programming languages, so a virtual machine or dedicated server would be required.

To mitigate the possibility of the software running on an environment as restrictive as Central, the decision was made to write the software in standard 'vanilla' PHP, making use of only necessary frameworks and libraries that will be compatible with available database technologies and the version of Apache web server running. Should a far better environment present itself at a later point, the program can be straightforwardly refactored to take advantage of PHP 7.x.x functionality.

2.2. Persistent Storage

The application, after retrieving data and carrying out any necessary conversion or processing from a select data source then saves entries as records in a relational database.

PDO (PHP Data Object) is the interface used for accessing databases in PHP. It is database agnostic, meaning that the same functions for issuing queries, etc. can be used without prior knowledge of the type of relational database used – this is taken care of by a PDO database driver, of which there are many, including support for MySQL, (Structured Query Language) PostgreSQL, Microsoft SQL, and SQLite. [45]

For development, an IMPACS-hosted MySQL database will be used, but it is intended that the type of database can be specified in a settings or configuration file, so that any supported database by the environment's PHP installation may be used.

2.3. Development Environment

Application development will be carried out primarily on two systems. The JetBrains PhpStorm IDE (Integrated Development Environment) is a powerful cross-platform software package specifically designed for PHP software development that provides debugging, code analysis, refactoring, version control, and Composer dependency management support out-of-the-box. [46]

In addition, the cross-platform Visual Studio Code editor by Microsoft is also used for making quick modifications to source code files and text documents. Microsoft Word 2016 for both Windows and macOS is used wherever possible for rich text editing. If any image editing is required, Adobe Photoshop CS6 for macOS will be used wherever possible.

The technical specifications of the computers used for development are as follows:

Desktop:

- Mac Pro 3,1 (Early 2008) running macOS 10.13.x High Sierra
- 2x 2.8GHz Quad-Core Intel Xeon E5462
- 32GB 667MHz DDR2 RAM
- NVIDIA GeForce GTX1050 Ti
- Solid-State Drive

Laptop:

- Custom-built laptop running Windows 10 Education x64
- 2.5GHz Intel Core i7-4710MQ
- 8GB 1066MHz DDR3 RAM
- NVIDIA GeForce 860M
- Solid-State Drive

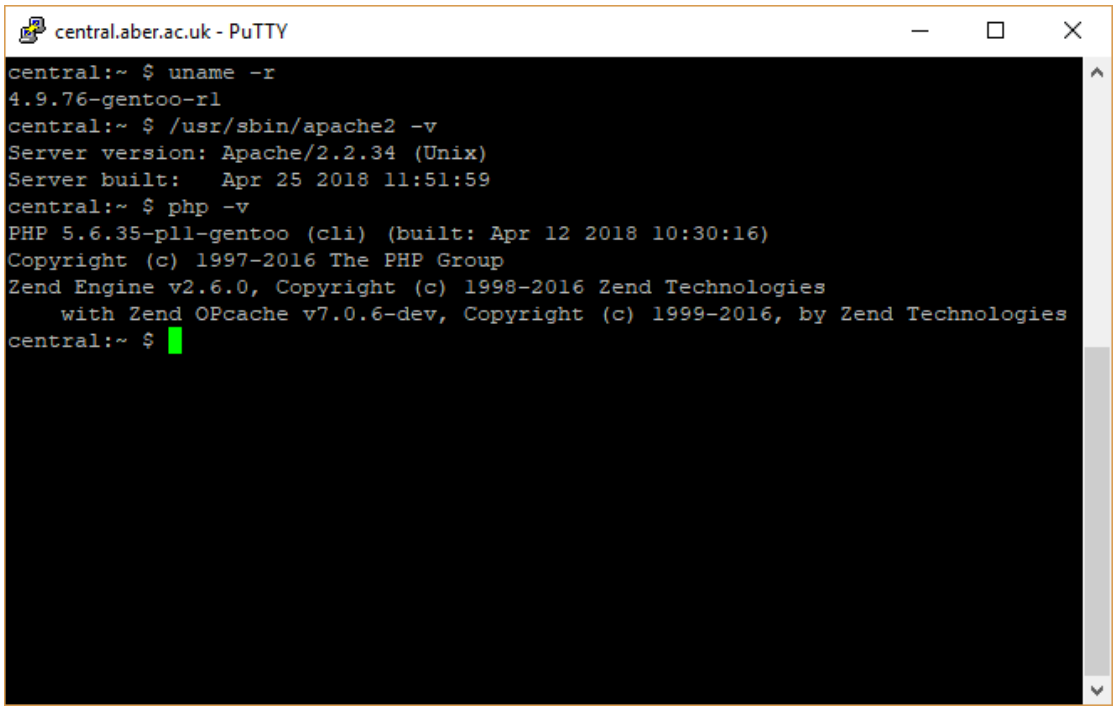


Figure 2-1 Central system information as viewed from PuTTY on Windows.

It can be seen from the above figure that at this time of writing, Central appears to be running the Gentoo GNU/Linux operating system, with the Apache 2.2.34 web server, and PHP 5.6.35.

2.4. Security Requirements

The application, while interfacing with both a relational database and with data sources which both require authentication to download data therefrom, is not planned to include any functionality to create, update, delete, or otherwise modify data. As this data is not confidential and will be used by a wide number of students and academic staff, authentication with the application itself is not necessary for viewing its content. As measurements will be taken to protect even non-identifying / confidential data, this will fully comply with the Data Protection Act 1998.

Nevertheless, precautions must be made to ensure that common cyberattacks against web applications are mitigated. The inclusion of prepared statements, and both client-side and server-side validation and sanitisation of incoming data can help prevent SQL injection attacks where an attacker may attempt to inject SQL queries or script into a HTML form in order to modify, delete, or add malicious entries to the underlying database. [47]

Additional security measures applicable to this type of web application include CSRF (Cross-Site Request Forgery) protection on forms to help prevent malicious POST requests [48] and the inclusion of integrity parameters for JavaScript or CSS includes that are retrieved from CDNs, (Content Distribution Networks) that prevent content to be loaded if there is a suspicious mismatch in file checksums.

2.5. Version Control

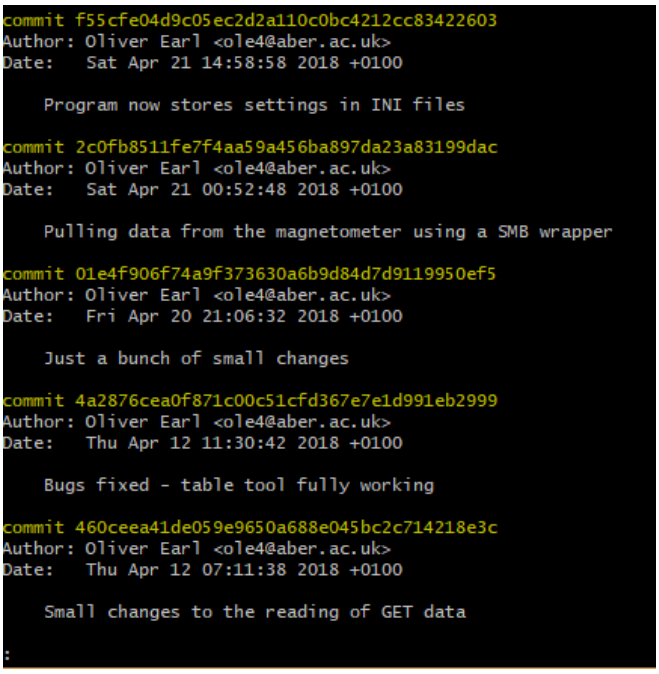


Figure 2-2 Example commits within the project Git repository.

The source code in its entirety for the project is stored within a Git repository. Git is a widely used, open source distributed version control system (DVCS) that was originally developed by Linus Torvalds in 2005 for use with the Linux kernel and its source code. [49] Changes to the source code are uploaded ('pushed') to an academic Bitbucket account (the repository's remote repository) where they can be browsed or downloaded ('pulled') by both the developer and the project supervisor. Bitbucket is a Git repository management system by Atlassian. [50]

As a complementary precautionary measure, the repository is securely kept within a personal Dropbox account, a cross-platform cloud-based storage solution that uploads and synchronises changes to files and folders within it to all associated computers and mobile devices running the software, allowing for seamless development across multiple platforms. [51]

2.6. Overall Design

2.6.1. Introduction

The application is split into three main controller modules, Magneto, Renderer, and Retriever. Magneto is the internal name for the main class of the application, as that is what the application was referred to internally and serves as the root namespace for the application. Eventually the program was renamed to 'Borealis' since the program was intended to retrieve data from more than just magnetometers, but the internal name stuck. This class could be considered the main 'app' class, or the primary controller of the application and is the first to bootstrapped once autoloading is complete.

The Renderer class is the controller that handles the view of the application and uses a templating system to render the view from template files; it separates the view from the rest of the application by only injecting what the view needs into it, such as fetched data from the model, sanitised data from the user, locale data, etc. It is also responsible for routing requests, rendering the correct views, and injecting required data accordingly.

The Retriever class is a controller responsible for retrieving data from data sources. As only FGM-3 magnetometer data retrieval was successfully implemented on time, the Retriever class was only designed to retrieve data from it, but future refactoring could be an option to enable further support for alternative sources, such as those from amateur radio and sources from the Internet.

Additional supporting classes are called upon by these controllers for smaller functions, such as the model for magnetometer, retrieving and saving program settings, and functions pertaining the program's localisation functionality.

2.6.2. Architectural Pattern

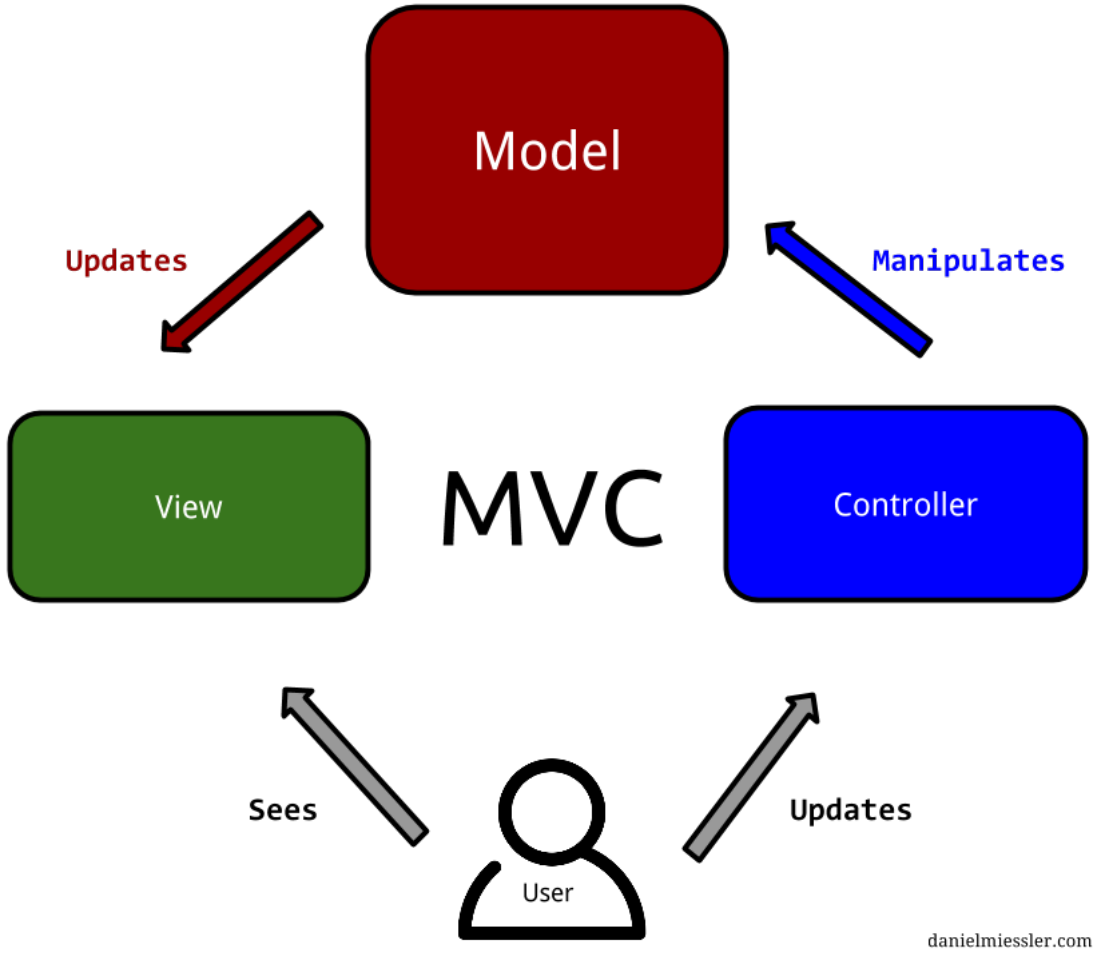


Figure 2-3 Model-View-Controller Architectural Pattern [51]

Like many web applications, the MVC architectural pattern is likely to fit the application best. This separation of concerns will separate the complex project into three distinct areas:

- The **model**, or domain, which concerns data, such as object models that represent data stored in the database.
- The **view**, which provides the template and interface which the user interacts with
- The **controller** which manipulates the model and as put by D. Miessler, 'facilitates content being moved back and forth between them [the Model and View]' [52]

2.6.3. Intended Structure

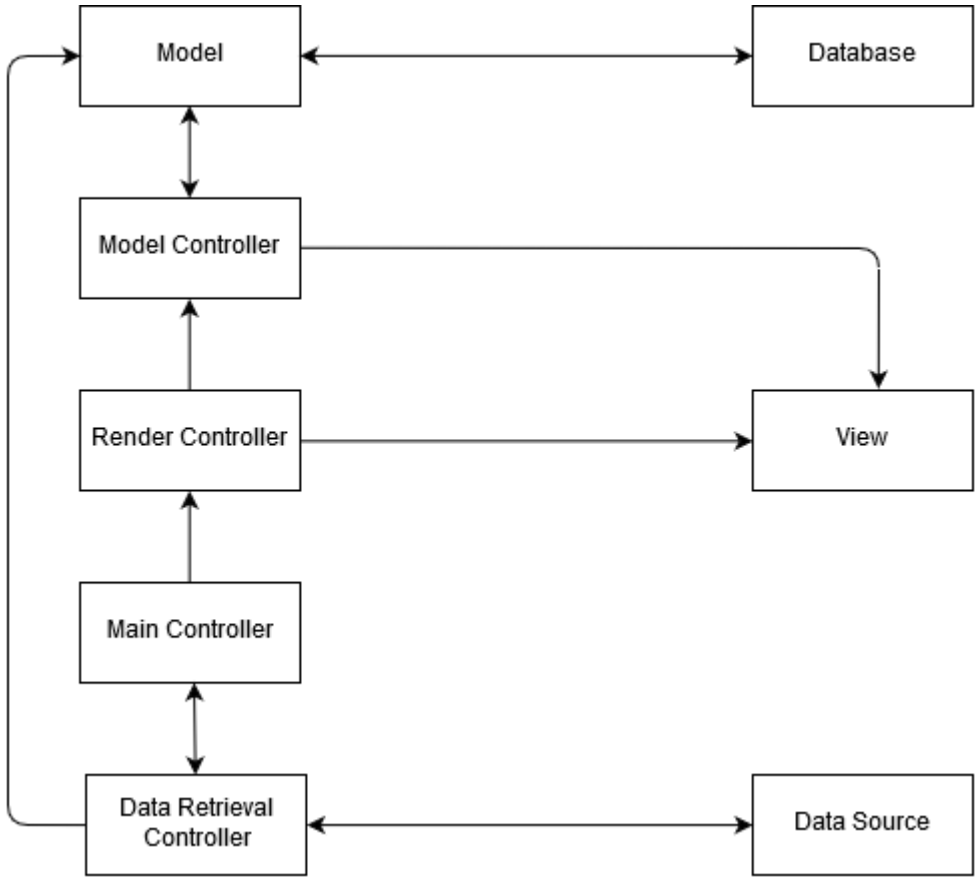


Figure 2-4 Rudimentary initial UML (Unified Modelling Language) diagram demonstrating data flow and separation of concerns.

During the initial planning of the application, one of the first things considered was how the flow of data would work, ensuring that concerns remained separate and that logic designed for one specific area of the program did not overflow into the next to maximise code efficiency and maintainability.

As demonstrated by Figure 2-4, multiple controllers are used to communicate with the model layer depending on whether that data is being retrieved for rendering into a view that the user can interact with, or whether data is incoming from a data source, such as a magnetometer.

Although not always the norm in MVC, in this instance the view is given model data from a dedicated controller that is responsive for formatting data returned from the model into something expected by the view.

This overall model does not include functionality outside what is considered core (explored more in Feature List 2.7) to the program, such as the workings of localisation.

2.7. Feature List

In line with the second process in the Feature-Driven Development methodology, 'Build a Feature List' [29], the following is a full feature list intended for the application. In addition, each of the features has been given a classification of 'Core' and 'Complimentary'. This is a simplified derivative of MoSCoW analysis commonly used in software engineering, particularly the Dynamic Systems Development Method methodology (DSDM), where the 'must haves, should haves, could haves, and won't haves' are identified in a software project. [53] The full method would be unnecessary to implement however, as 'won't haves' are redundant due to there being no additional development cycles following the deadline of the project, and since there are so few pieces of core functionality, it could be considered trivial to differentiate between 'must have' and 'should have'.

To reiterate, the format of identified features is <action> <result> <object>. If a feature is too large, it is broken down into subsections. [29]

- **Magnetometer Retrieval**
  - Connect to and retrieve data from the magnetometer. (Core)
  - Determine when data requires retrieving. (Complimentary)
  - Allow forced data retrieval by means of an override. (Complimentary)
- **Frontend Interface**
  - Navigate and choose data to render in a graph (Core)
  - Specify desired data using HTTP GET parameters (Complimentary)
  - Display stored data in the form of a graph. (Core)
  - Display stored data in the form of tables. (Complimentary)
  - Display all data at once in the form of a graph. (Core)
  - Display latest data in the form of a graph. (Core)
  - Add means to export currently displayed data as JSON. (Core)
  - Add means to export currently displayed data as XML (Complimentary)
  - Add a Settings page to change basic settings of the frontend. (Complimentary)
  - Add an About page with information regarding the project and program (Complimentary)
  - **Localisation**

        - Add a means to change the language between English and Welsh. (Complimentary)
        - Add a means to detect whether the user's web browser language is English or Welsh and choose language accordingly. (Complimentary)
        - Website locale is stored within files that is printed onto the page according to what language is loaded. (Complimentary)
- **Program Settings**
    - Program settings are saved to and retrieved from a persistent file.
- **Security**
    - Provide sanitisation to incoming data (Core)
    - Provide client-side validation on inputs (Core)
    - Provide basic CSRF protection (Complimentary)
- **API**
    - Return JSON of requested data entries using GET or POST (Core)
    - Return JSON of latest data entry (Complimentary)
    - Return JSON of all data entries (Complimentary)
    - Return XML in place of JSON for all the above (Complimentary)
    - Handle errors and malformed requests accordingly (Complimentary)
    - Receive and process commands such as manual magnetometer retrieval (Complimentary)
- **Other Data Source Retrieval (*Unimplemented*)**
    - **Amateur Radio / Radio Antenna Retrieval**
        - Receive data from amateur radio / radio antenna connected computer (Complimentary)
    - **Internet Retrieval**
        - Receive data from Internet-based sources (Complimentary)
        - Manually specify choice of Internet source (Complimentary)

2.8. Detailed Design

2.8.1. Introduction

This section will go into detail each of the major classes of the application, their notable methods and will elaborate upon interaction with third-party libraries where applicable.

This section overlaps very heavily with the Implementation side of the report, particularly due to significant discussion of code, functions, classes, and language functionality, such as the wealth of built-in functions within PHP. It was felt that this overlap, although unorthodox, would help efficiently and fluently describe the application, the algorithms and flow of execution therein, and design choices taken due to working knowledge of the programming language and its feature set.

The program features two primary entry points, `index.php` and `api.php`, the former autoloads, and bootstraps the Magneto class responsible for setting up the

application, whereas the latter file following autoloading carries out a different route of execution specifically for returning data programmatically.

2.8.2. Composer Autoloader

Composer is a dependency management tool for PHP that relies on a JSON file, named `composer.json` to install any third-party libraries, frameworks, or other dependencies that the application needs so that they can be installed or updated by issuing a single command, making deployment of the application simple, without the need of storing hundreds of megabytes or more of third-party code within a program's repository. Dependencies can be installed on a global basis, such as for helpful workflow tools, or more commonly on a per-project basis. Similar popular dependency managers for other languages include npm for Node.js and Bundler for Ruby. [54]

Composer produces an autoload file, `autoload.php` that is included within both entry-points of the program to automatically make available the classes and namespaces of third-party libraries, eliminating the need of painstakingly including each individual file as they are needed. In this project, Composer is also used to PSR-4 autoload the namespaces pertaining to this application, so that use commands can be issued, and classes resolved without complex require chains. [55] PSR-4 is the fourth PHP Standard Recommendation enacted by the PHP Framework Interop Group, who describe this recommendation as a 'specification for autoloading classes from file paths.' [56] This mapping of directories to namespaces in PSR-4 style can be seen inside the `composer.json` file as follows:

```
"autoload": {
  "psr-4": {
    "ole4\\Magneto\\": "src/",
    "ole4\\Magneto\\Config\\": "src/Config",
    "ole4\\Magneto\\Functions\\": "src/Functions",
    "ole4\\Magneto\\i18n\\": "src/i18n",
    "ole4\\Magneto\\Models\\": "src/Models",
    "ole4\\Magneto\\Tests\\": "tests/"
  }
```

Figure 2-5 Autoload subsection of the `composer.json` file.

This autoload functionality is crucial for the correct development of object-oriented applications with PHP.

2.8.3. Magneto

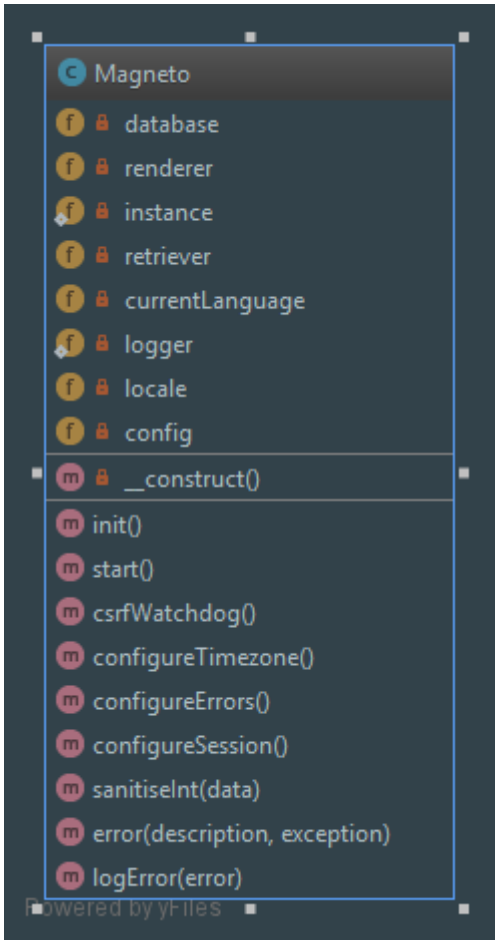


Figure 2-6 Autogenerated Magneto Class Diagram.

A public static `init()` method is called from the `index.php` entry point to begin bootstrapping the program. This method is responsible for setting up the program time-zone, session handler, and error handling. Following this, it instantiates and returns the application, which then begins to set up other areas of the program. During the setup of error handling, the program configures the popular Monolog/Logger third-party logging library, used for logging application events and error messages to logfiles. [57]

First the configuration array is retrieved from a static method in the Config class, then the program language and its associated locale file are retrieved from the Locale class. Then a PDO instance is retrieved from Connector. Finally, the Renderer and Retriever are instantiated and the `start()` method is called.

The `start()` method is the primary method for the class that carries out any required functionality Therein lies calls to methods in the Renderer and Retriever classes to begin their subsequent functionality, covered in detail within their subsequent sections, but not before the program calls an internal method, `csrfWatchdog()`, which determines whether any POST data has come from an authentic user request, and if not due to a suspected cross-site request forgery attack, throw an exception that ceases program execution.

```
private function csrfWatchdog()
{
    if (empty($_SESSION['token'])) {
        $_SESSION['token'] = base64_encode(openssl_random_pseudo_bytes(32));
    }

    if (!empty($_POST)) {
        if (empty($_POST['csrf_check']) || !hash_equals($_POST['csrf_check'],
$_SESSION['token'])) {
            Magneto::error('csrf_violation', 'CSRF Violation');
        }
    }
}
```

Figure 2-7 CSRF Watchdog Private Method.

This method inserts a cryptographically secure 32-bit token into the user's session if one does not already exist. The token is also inserted into all forms contained in the program's views. Although C. Shiflett recommends the combination of PHP's MD5 algorithm function in conjunction with the built-in `uniqid()` and `rand()` functions [48], we now know these to be cryptographically insecure, in part due to the following:

- MD5 is considered 'broken and unsuitable for future use' by the Carnegie Mellon University Software Engineering Institute [58]
- `rand()` is widely considered predictable, detailed as such in a heavily detailed report by J. Roper about cracking random number generators. (RNG) [59]
- `uniqid()` only adds a set amount of entropy (lack of predictability; chaos) by default and is considered inappropriate for cryptography by The PHP Group itself [60]

Consequently, the more reliable randomness provided by a combination of `base64_encode()` and `openssl_random_pseudo_bytes()` that are used to safely generate a token. If POST data exists, then the token is checked. If a mismatch with that stored in the session is found, or if it merely does not exist, the program calls the internal error function.

This function, a public static method, is one of the most notable in the class due to its program-wide usage, is called in catch blocks for exceptions and when an error or unexpected condition is encountered anywhere in the program.

```
public static function error($description, $exception)
{
    try
    {
        if ($description === 'csrf_violation') {
            new Exception('CSRF violation');
        }
        $locale = Locale::getLocale();
        $localisedException = $description;
        if (isset($locale[$description])) {
            $localisedException = $locale[$description];
        }
        $_SESSION['errors'][] = $localisedException;
        self::logError("{$localisedException} {$exception}");
    }
    catch (Exception $exception)
    {
        trigger_error("Unrecoverable error. Please contact developer.
<br>Exception: {$exception}",
            E_USER_ERROR);
    }

}
```

Figure 2-8 Error Public Static Method.

The only exception to its normal pattern of execution is for when the CSRF protection is tripped, causing the method to throw an exception itself and end the program. For the purpose of ensuring that even error messages themselves are localised, the method will retrieve locale files itself. If there are matching localised error messages it will use them, otherwise defaulting to those provided by the method arguments.

The error method will add the exception to session storage which is injected into the view and displayed as a Bootstrap styled alert. It will also use the Monolog library to write an error message to the dedicated error logfile in the storage directory.

Finally, it is of note that the Magneto class also contains a static method for sanitising integers, which is used frequently through the application particularly for sanitising ID values used for retrieving magnetometer records.

2.8.4. Config

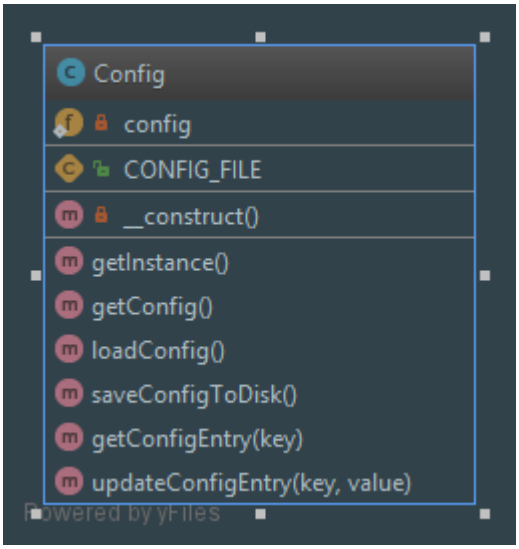


Figure 2-9 Autogenerated Config Class Diagram.

Config is simply a wrapper class for the program's settings, providing the necessary functionality for making modifications to it during runtime, fetching specific values, saving changes to disk as well as the initial loading of the configuration.

It was originally intended that the program's configuration would be saved in INI format, a common plaintext file format used for storing program initialisation values that can be easily parsed by PHP into associative arrays. Eventually this implementation was replaced with a modest config array file technique that is explored along with the reasoning behind this choice and the details behind its exact implementation its subsequent section 3.2 in the Implementation chapter. The program configuration values are as follows, with a brief description: (The keys to chiefly important config entries have their keys highlighted in **bold** and are elaborated upon further following the table. Entries in *italics* cannot be edited anywhere in the program and are considered constant values.)

Config Array Key	Data Type	Array Value	Description
debug	Boolean	true	Debug mode flag. Determines whether certain operations such as error reporting behave differently.
appName	String	'Borealis'	Application name

appVersion	String	'1.4.0'	Version of the program
appDescription	String	'Data collation and graphing tool with API for magnetometer data.'	Description of the program
appAuthor	String	'Oliver Earl'	Author of the program
appAuthorEmail	String	'ole4@aber.ac.uk'	The email address that belongs to the author of the program
maxElements	Integer	4	Number of magnetometer entries that are displayed in the Navigator view
hostname	String	'db.dcs.aber.ac.uk'	Hostname used for database connectivity
dbName	String	'ole4'	Name of the database
username	String	'ole4'	Username used for database authentication
password	String	'REDACTED'	Password used for database authentication
dbType	String	'mysql'	Type of relational database, can be any supported database type for which PDO has access to a driver, such as 'mysql' or 'postgres'
magnetometer_hostname	String	'imapspc0017.imaps.aber.ac.uk'	Hostname for the computer connected to the magnetometer
magnetometer_username	String	'imaps\ole4'	Domain and username used

			for SMB authentication
magnetometer_password	String	'REDACTED'	Password used for SMB authentication
magnetometer_share	String	'magdata'	Remote share that is to be mounted containing magnetometer data
magnetometer_latest	String	'111'	Three-digit code representing the day of the year, used to determine latest retrieval

`maxElements` determines how many input boxes are displayed in the Navigator view, the view that allows the user to choose what data entries they would like loaded into the Graphing Tool, or in tabular format.

`magnetometer_latest` is updated whenever the program retrieves data from the magnetometer. It is a three-digit code that represents the day of the year. This will be covered in more detail in the Retriever subsection.

If the `debug` flag in the program settings is set to true, errors and exceptions will be raw printed to the screen instead of just being rendered as Bootstrap alerts, in addition to being logged, and an additional debugger becomes available in the Graphing Tool view.

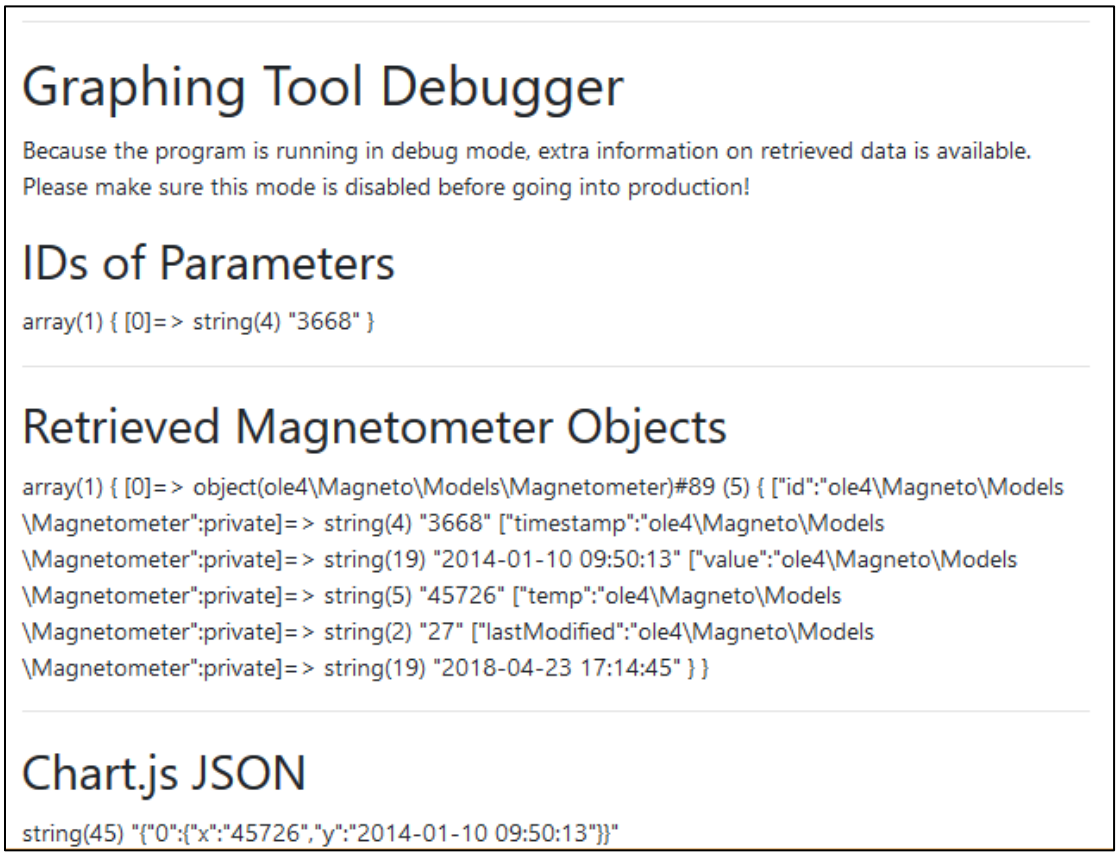


Figure 2-10 Graphing Tool Debugger that becomes available when the debug flag is set.

This debugger shows data into the view pertaining to saved entries from the database, their unique IDs, and the generated JSON necessary for rendering that has been injected into Chart.js, the JavaScript charting framework.

2.8.5. Connector

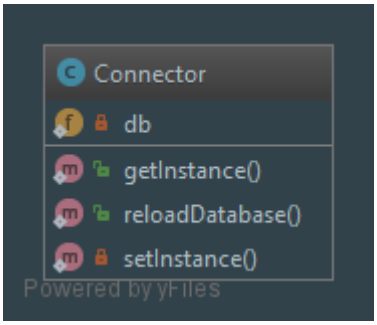


Figure 2-11 Autogenerated Connector Class Diagram.

The Connector class initially sets up a PDO object with aforementioned values stored in the configuration file and returns it. It also has the ability to destroy the database object so that it can be reloaded – a functionality that was originally intended for supporting changing database config values live.

```
private static $db;

public static function getInstance()
{
    if (!isset(self::$db)) {
        self::setInstance();
    }
    return self::$db;
}
```

Figure 2-12 Private database property and characteristic singleton instance Public Static Method.

The PDO object static private property `$db` exists as part of a singleton, a creational design pattern used for ensuring that only one instance of a class exists at one time by privatising the constructor, preventing external instantiation outside of a dedicated public static method. [61] The method `getInstance()` will launch the necessary private static `setInstance()` method if the database object has not yet been instantiated, otherwise it returns the one already in use. This helps prevent instantiating more than one unnecessary database connections, and in the case of ever being able to change the database settings, more than one database connection at a time.

This database functionality was originally self-contained within the Magneto class, but it was abstracted to its own class so that it could be more readily injected into other classes that required a connection to the database.

2.8.6. Locale

The Locale class has two main jobs. The first is to determine and keep track of the program's language, which is either English or Welsh. The second is to provide localisation files containing localised versions of application text strings.

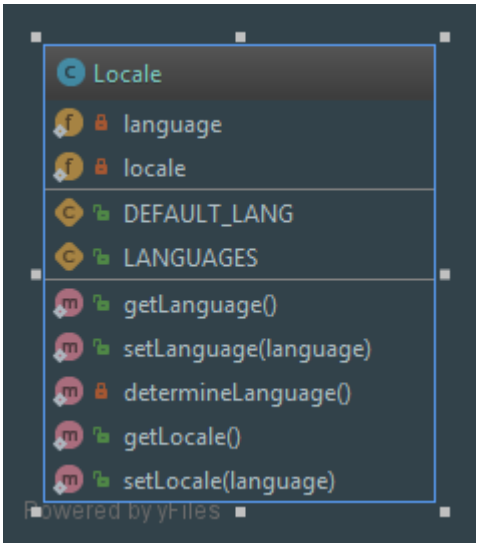


Figure 2-13 Autogenerated Locale Class Diagram.

`setLanguage()` is one of the two key methods in this class as it will first determine whether the language is already pre-set by calling `determineLanguage()` – this first checks the session for language data, if this data does not exist, the program will determine the user's web browser language

and use that value. Should it be neither English or Welsh, it defaults to English. Ultimately if the user has submitted a language choice by clicking on a link or specifying in the URL as a GET parameter under 'language' (i.e. `index.php?language=cy`), this will be chosen as the program's language.

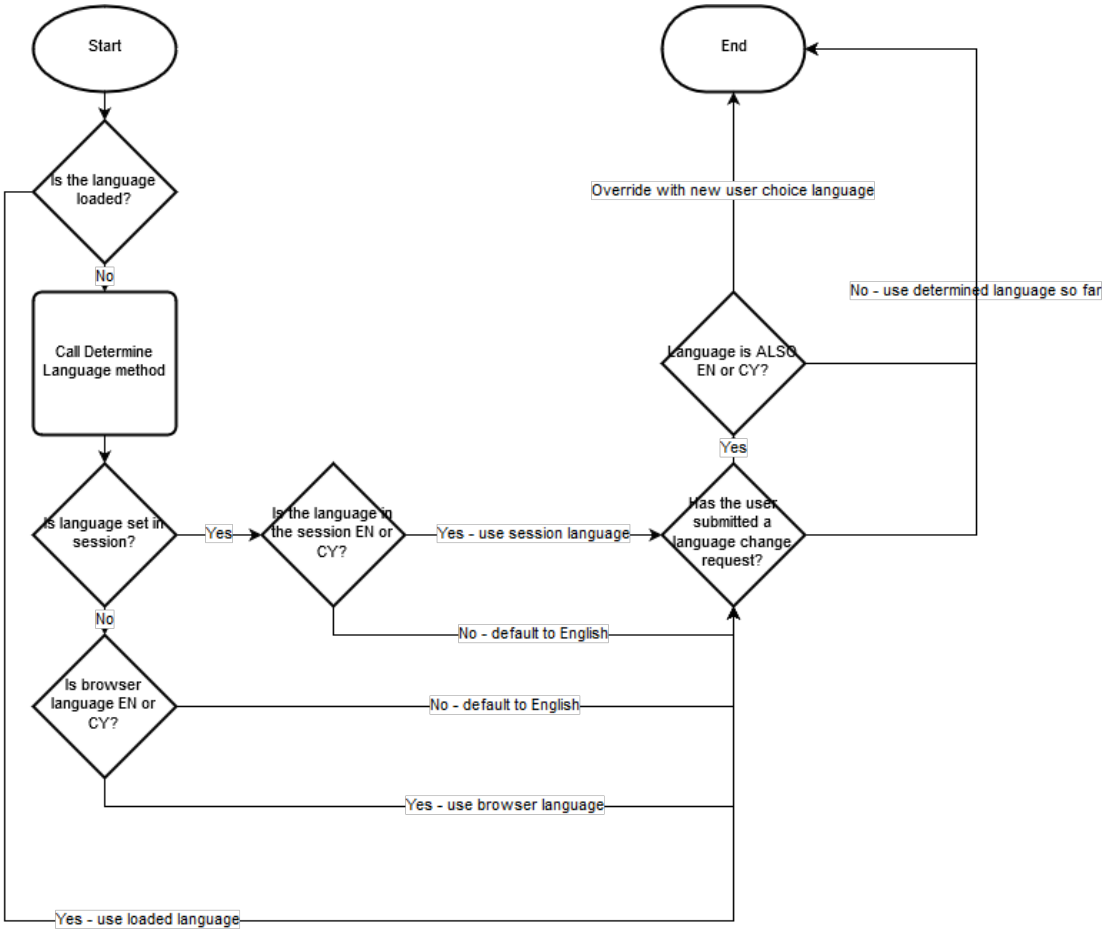


Figure 2-14 Sequence Diagram for Language Determination algorithm.

As demonstrated by the sequence diagram, the language determination algorithm outlined in `setLanguage()` and `determineLanguage()` determine at multiple levels if a predominant language has been established and makes attempts to detect it through the user's browser or session data, using English as a failsafe should there be any suspicious anomalies or an unsupported language being used. Any language request sent through, provided it is a supported language however will take precedence in order to facilitate the switching of application localisations.

The next step once the user's language has been determined is to load locale files. As opposed to a Gettext internationalisation system that is frequently used in PHP projects to provide multilingual support, [62] the program uses a cruder but easier to implement directory containing JSON files for each supported locale. The JSON file contain identical keys, except with values representing localised text strings. This JSON file is decoded into an associative array that is injected into views and error messages. The design choice was due to the complexity of configuring Gettext

in comparison, and as localisation was a Complimentary functionality, the KISS ('Keep it Simple, Stupid') principle [63] was kept heavily in consideration.

```
public static function setLocale($language = null)
{
    if (is_null($language)) {
        $language = self::DEFAULT_LANG;
    }

    $dataset = "locale/{$language}.json";

    if (file_exists($dataset)) {
        $json = json_decode(file_get_contents($dataset), true);

        if (is_null($json)) {
            Magneto::error('locale_json_failure', json_last_error_msg());
        }
        self::$locale = $json;
        return;
    }
    Magneto::error('locale_json_missing', json_last_error_msg());
    self::$locale = null;
}
```

Figure 2-15 Public Static Method for setting program locale.

`setLocale()` takes the current language as a parameter when called, but defaults to English as a failsafe. It then checks to see whether the appropriate locale file exists prior to attempting to gets its contents and decode the contained JSON into an associative array. If the JSON is malformed, or the locale file is missing or inaccessible, errors are issued.

2.8.7. Renderer

The Renderer is one of the three main controller classes in the application and plays the role of developing and rendering the view, injecting necessary data into the view, and also takes on routing responsibilities. Routing involves handling necessary GET queries to direct the user to the correct webpage for the application.

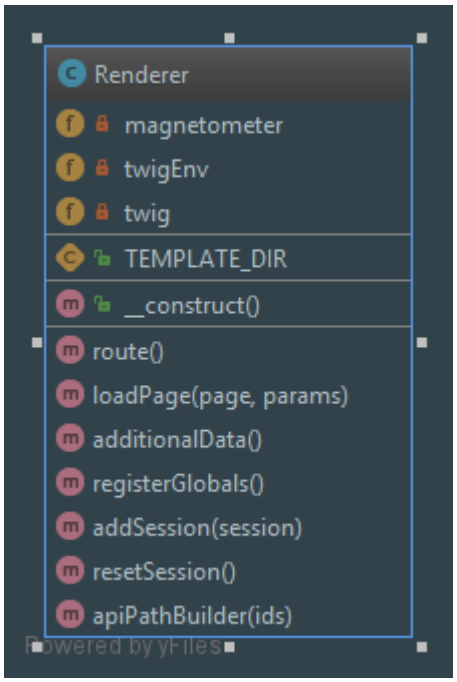


Figure 2-16 Autogenerated Renderer Class Diagram.

The method serves as a wrapper for the Twig templating engine library, commonly known for its use within the Symfony framework. [64] This is initialised when the class is initialised, where it is most importantly passed the path to the templates directory where Twig templates for the views are stored. The `loadPage()` method acts as a wrapper for Twig's `render` method, passing in the page that will be rendered along with an array of arguments that Twig expects as data injected into the view. This data is amalgamated and injected during routing, and global data accessible by all views is configured in `registerGlobals()`.

`registerGlobals()` registers configuration data, a database connection, the current program language, the loaded localisation array, and a generated CSRF token for forms into the view using Twig's `addGlobal()` method. [65] Additionally, arrays containing errors and success messages are injected, but this has no effect on the view if they are empty as they are by default.

The `route()` method is large and made up of a very complex switch statement in order to load the correct pages and both retrieve and inject the correct information into the view based on a page GET (i.e. `index.php?page=about`) in lieu of clean or 'pretty' URLs [66] generated by rewrite engine functionality enjoyed by many web frameworks. While large or excessive switch statements are considered a code smell in object-oriented programming languages as they indicate that functionality could most likely be implemented using polymorphism instead, [67] it is important to consider that there is a trade-off between having a simple but 'smelly' router that works for a relatively simple application, and a more complicated router whose implementation is much more complex, coming back to the aforementioned KISS principle. [63] Polymorphism is eloquently described by J. Skeet as something that ' allows the expression of some sort of contract, with

potentially many types implementing that contract in different ways, each according to their own purpose.' [68]

```
case 'graph':
    $entries =      $this->additionalData();
    $objects =      $this->magnetometer->getObjectsFromIds($entries);
    $graphJson =    $this->magnetometer->getGraphJsonFromObjects($objects);
    $apiPaths =     $this->apiPathBuilder($entries);
    $this->loadPage('graph', [
        'entries'   => $entries,
        'objects'   => $objects,
        'json'      => $graphJson,
        'jsonApi'   => $apiPaths,
        'xmlApi'    => $apiPaths . 'xml=true'
    ]);
    break;
```

Figure 2-17 `route()` switch case for graph, demonstrating what methods are called and what data is injected into the view prior to rendering.

As seen in the above figure, a number of values are retrieved before being inserted into the array of arguments to be passed into the view. For displaying entries in the Graphing Tool view, magnetometer entry parameters must be harvested, done in the `additionalData()` method. Once this data is collected, the `$entries` variable can be used to collect an array of Magnetometer objects from the database, which consequently can be used to generate JSON for use with the Chart.js library. Lastly, paths to the API entry-point for both JSON and XML are generated using the entry IDs and the unsophisticated `apiPathBuilder()` method.

```
private function resetSession()
{
    if (isset($_SESSION['errors']) || isset($_SESSION['successes'])) {
        if (isset($_SESSION['timer'])) {
            if ($_SESSION['timer'] !== time()) {
                unset($_SESSION['errors'], $_SESSION['successes'],
$_SESSION['timer']);
            }
        } else {
            $_SESSION['timer'] = time();
        }
    }
}
```

Figure 2-18 Reset Session Private Method

The `addSession()` and `resetSession()` methods are for the displaying of Bootstrap alerts – errors and success messages. A timer is used to ensure the message is displayed at least once before being unset. This takes place before views are rendered.

Twig templates, including their syntax, are elucidated in greater detail in 2.12 User Interface Design. Additional insight into the use of custom Twig filters is also elaborated upon in Implementation 3.5.

2.8.8. Retriever

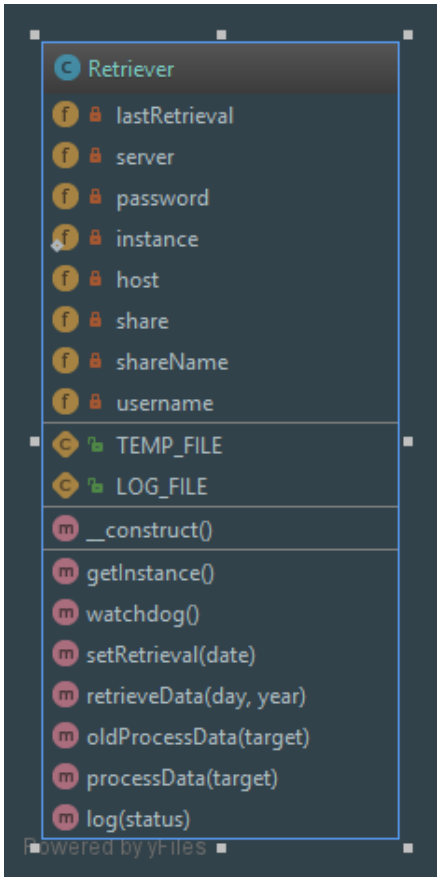


Figure 2-19 Autogenerated Retriever Class Diagram.

It is a reasonable assumption to make that the Retriever class contains the most important functionality in the application. It is responsible for the retrieval and storing of data from data sources, namely the magnetometer. It was originally proposed that this class remains agnostic from which source data is retrieved from by making use of the previously described abstract class or interface, to propose a standard model that all data sources will either inherit from or implement. This would maximise the opportunity for code reuse and polymorphic functionality regardless of what children classes or implementations are used.

As only magnetometer functionality was implemented, that will be the scope of further discussion regarding this class, its methods, and its properties.

Retriever implements the SMB wrapper third-party library Sgrabaum/SMB, which is a fork of Icewind1991/SMB [69]. The primary reason for utilising this fork over the original library is due to the ease in which it can be installed using the Composer dependency manager without sacrificing functionality or being forced to use initialise and pull from the legacy PEAR (PHP Extension Application Repository) repository, [70] for the purposes of communicating with the magnetometer, mounting shares, and downloading data. The library has methods built-in such as `get()` for retrieving files, as well as other methods for reaching

file-specific metadata, such as last modified dates and file sizes. The library is configured using `magnetometer_` prefixed values passed in from the Config configuration array.

The Retriever is also a singleton class. When the `watchdog()` method is called, it checks whether it should attempt to pull data from the magnetometer by checking whether the current day differs from the value of the `magnetometer_latest` key stored in the configuration file, or whether there is an override flag in GET. (`index.php?override=yes`)

When either of these criteria are met, the `retrieveData()` method is invoked.

```
public function retrieveData($day = null, $year = null)
{
    try {
        if (is_null($day)) {
            $day = sprintf('%03d', date('z') + 1);
        }

        if (is_null($year)) {
            $year = date('Y');
        }
        $name = "DATA{$day}.csv";
        $tempFile = self::TEMP_FILE;
        $dir = $this->share->dir($year);

        foreach ($dir as $file) {
            if ($file->getName() === $name) {
                $this->share->get($file->getPath(), $tempFile);
                return $this->processData($tempFile);
            }
        }
        $_SESSION['errors'][] = 'record_missing';
        return null;
    } catch (Exception $exception) {
        Magneto::error('magentometer_failure', $exception);
        self::$instance = null;
        return null;
    }
}
```

Figure 2-20 Retrieve Data Public Method. Used to begin capturing data from the magnetometer.

The method accepts both a day of the year and year as optional parameters, deriving them from today's date and year if they are not provided. Files stored on the magnetometer are stored in the format `DATA123.csv` – where `123` represent a three-digit number corresponding to the day of the year, including trailing zeros, which make it necessary to format numbers as specific strings using PHP's built-in `sprintf()` and `date()` functions.

When the file has been found, the file is downloaded using the SMB library's built-in `get()` method, storing the file in a temporary CSV file in the storage directory. The pathway to this temporary file is then passed to the processing stage of data retrieval, `processData()`.

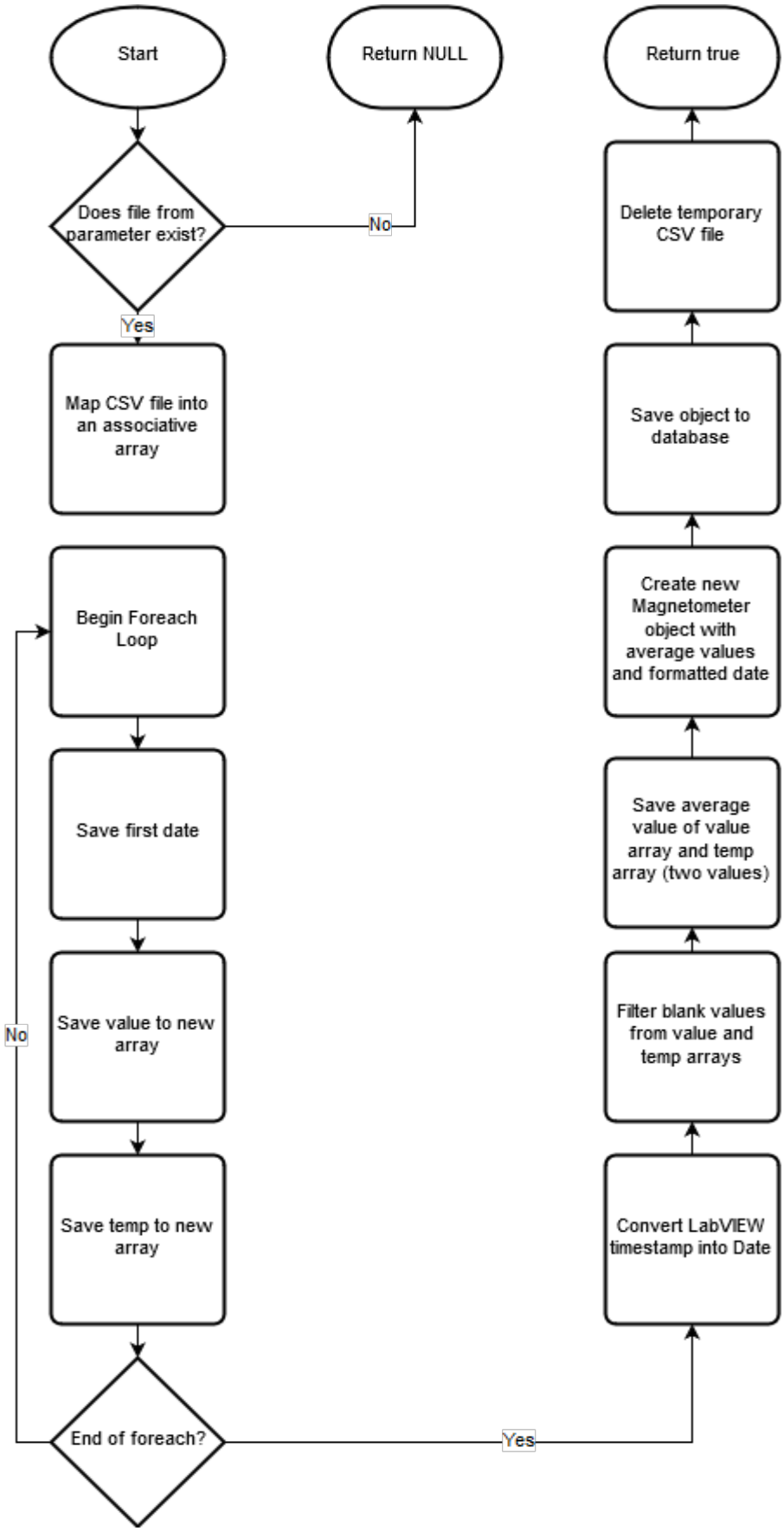


Figure 2-21 Process Data Sequence Diagram demonstrating data processing algorithm for retrieved magnetometer data stored in CSV file.

By keeping the data processing functionality separate and requiring a path to a CSV file as a parameter, this makes for reusable code that can be recycled if the program were to be further developed in the future. It maps the entirety of the CSV contents to an associative array – with each entry in the multidimensional array representing a row that was in the file. The magnetometer value and instrument temperature are popped from the array as it is iterated through by a foreach loop and stored in separate arrays for future processing. A timestamp is also retrieved. At the end of processing, any blank entries are filtered out of the new arrays and new values are calculated by finding the average measurement for the magnetometer value and the temperature.

This information is used to instantiate a new Magnetometer object – keeping `$id` and `$lastModified` defaulted to `NULL` as they will be filled in by the relational database. The Magnetometer model class contains a method used to save this information to the database which is called, but not before checking whether the record already exists in the database. If it does, the data used so far is discarded and an error is thrown. Regardless of execution path, the temporary CSV file created earlier is then destroyed.

Specific design choices for filtering out blank values and using average values is expanded upon in Implementation 3.4.

2.8.9. Magnetometer

The Magnetometer class is a part of the model layer of the program. It represents a daily entry from the FGM-3 magnetometer in the database. Its private properties are as follows:

Property Name	Data Type	Description
$id	Integer	Unique ID
$timestamp	String (Date/Time)	Date of entry creation
$value	Integer/Float	Reading in arbitrary value
$temp	Integer	Instrument temperature
$lastModified	String	Last Modified Date

As explained in the previous section, when Magnetometer objects are instantiated, the `$id` and `$lastModified` properties can be left as `NULL` values so that they are filled in automatically by the database. In addition, although LabVIEW format timestamps are originally used by the magnetometer, they are converted to standard date string for database stored entries.

Besides mutator methods, there are three methods pertaining to date conversion. `dateToUnix()` converts parses date strings into UNIX timestamps, and the two remaining methods are for converting timestamps between the respective

timestamp formats; making use of the epoch difference stored as a constant used for the numerical conversion. This value is `2082844800`. [4]

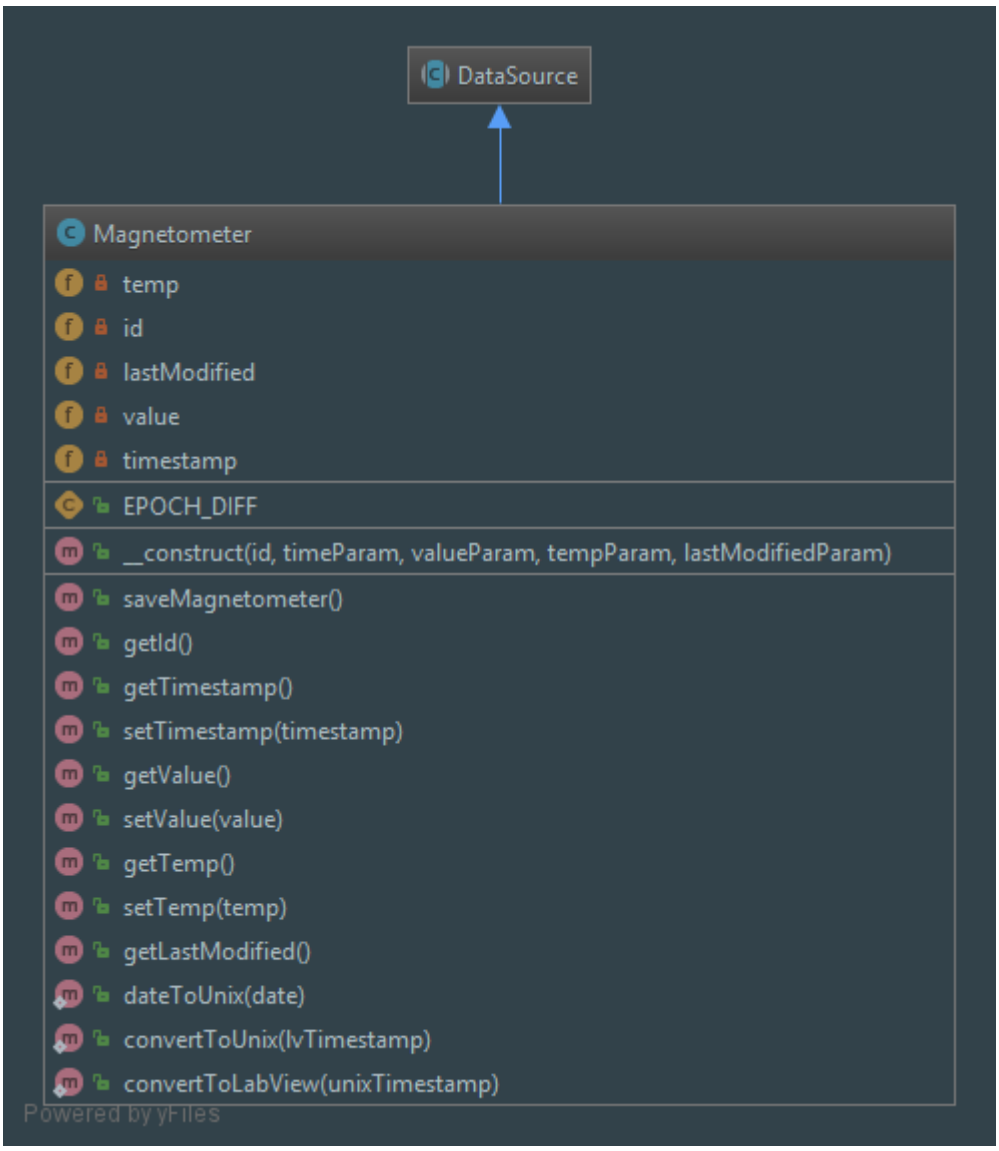


Figure 2-22 Autogenerated Magnetometer Class Diagram.

The `saveMagnetometer()` public method is used for saving the current object to the database with the Connector PDO instance. The usage of prepared statements is the one of the most notable characteristics amongst this and all other database-interactive methods in the program – as described by D. Marcus, prepared statements work by preparing 'an empty SQL query with empty values as placeholders' and then to 'bind values or variables to the placeholders. [71] Prepared statements help protect against SQL injection attacks. [47]

2.8.10. MagnetometerController

The MagnetometerController class is a controller responsible chiefly for accessing the database, fetching data, and returning arrays containing newly constructed Magnetometer objects. One of the other key functionalities contained within is the generation of JSON intended to be inserted into Chart.js.

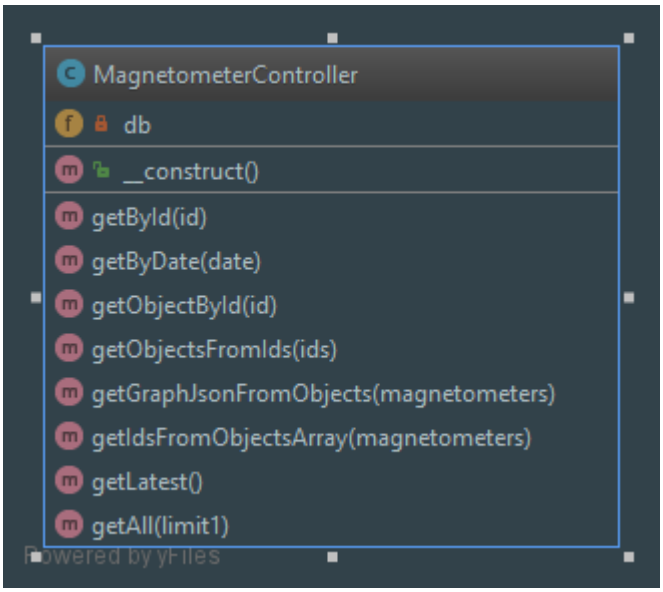


Figure 2-23 Autogenerated MagnetometerController Class Diagram.

`getById()` and `getByDate()` retrieve a single magnetometer entry from the database by parameter and return it in the form of an associative array. `getObjectById()` will however return a Magentometer object.

```
public function getObjectsFromIds($ids)
{
    $magnetometer = null;
    $array = [];

    foreach ($ids as $id) {
        $magnetometer = $this->getObjectById($id);
        if (!is_null($magnetometer)) {
            array_push($array, $magnetometer);
        }
    }
    return $array;
}
```

Figure 2-24 Get Objects from IDs Public Method.

This behaviour is made use of in other methods, such as `getObjectsFromIds()` which accepts an array of ID integers used for repeatedly calling `getObjectById()`, building an array containing Magnetometer objects which is subsequently returned.

For the Graphing View use case Chart.js utilises X and Y coordinate values for plotting points onto a graph. The `getGraphJsonFromObjects()` public method takes an array of Magnetometer objects, whereby it iterates through each object, further getting the object's date to be used as the Y coordinate and the recorded value for the X coordinate. It ignores any blank or `NULL` entries to help avoid poisoning graphs with blank or corrupted data.

Rather than repeating code in best adherence with the DRY principle of software engineering [39], the `getAll()` and `getLatest()` methods attempt to make efficient use of their sibling methods to retrieve all entries stored in the database,

or the very latest added, so far as to make use of one-another for this purpose as `getLatest()` merely calls `getAll()` but includes the optional Boolean parameter `$limit1` set to true. The algorithm for requesting all, or just one magnetometer entry from the database is illustrated by the following sequence diagram:

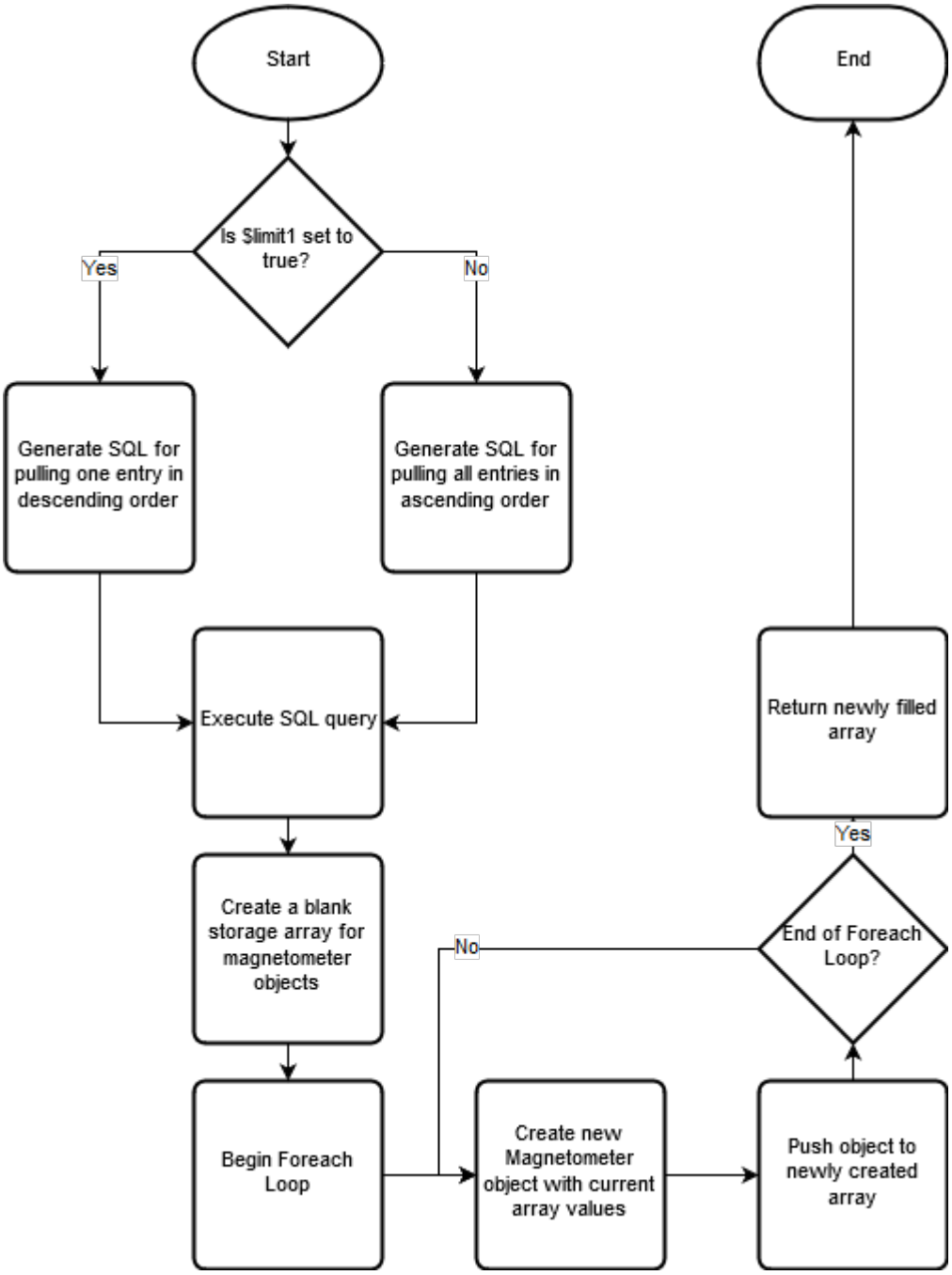


Figure 2-25 Get All Public Method Algorithm Sequence Diagram.

The significance of the difference in order in both paths of execution is that when hunting for the latest entry in the database, in reality, the program queries for all the entries, but returns only the first it finds ordered by unique ID, if this were to be in descending order, it would return the oldest entry in the database. This is not completely fool-proof and relies on the assumption that entries are inserted

into the database in chronological order and therefore the larger the ID value, the newer the magnetometer entry. This should be refactored to look for dates, but this could potentially take longer. In this battle of simplicity versus robustness, simplicity on this occasion was decided to be the more beneficial factor, which in turn corresponds with the KISS principle. For all entries, entries are returned in chronological order so that they can be read from top-to-bottom in tabular data, and elsewhere in the program to ensure data is rendered left-to-right in Chart.js, unless the user intentionally chooses data in an unorthodox order in the Navigator view.

2.8.11. API

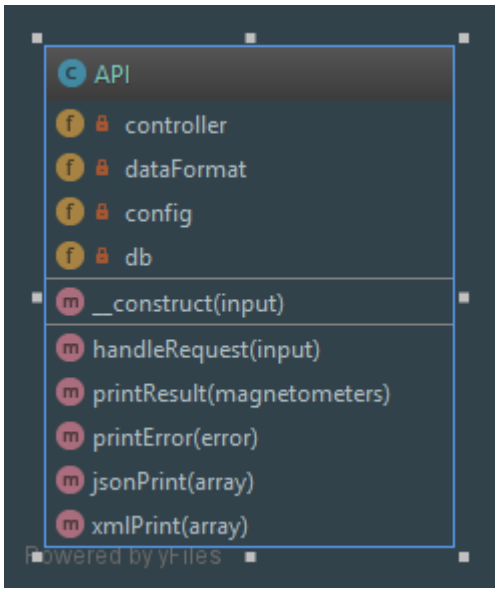


Figure 2-26 Autogenerated API Class Diagram.

The API is a means of accessing stored database magnetometer entries programmatically. The class is outside of the ole4\Magneto namespace, as the file exists also as an entry-point with procedural script that returns a HTTP 400 (Bad Request) [72] if there is no included GET or POST information to be handled, otherwise, an API object is instantiated with the GET or POST super-arrays as parameters. If they are empty however, or at any point the new `$api` object encounters an error, the program attempts to return a HTTP 400 message, where possible with an associated error message.

The construction of a new API object calls `handleRequest()`, which is the primary method of the application, responsible for sanitising requests, invoking a MagnetometerController instance for the retrieval of data, and finally calling the `printResult()` helper method, which acts as a gateway to the `jsonPrint()` and `xmlPrint()` methods accordingly, depending on whether an XML flag has been set in the request. The default response is in JSON format.

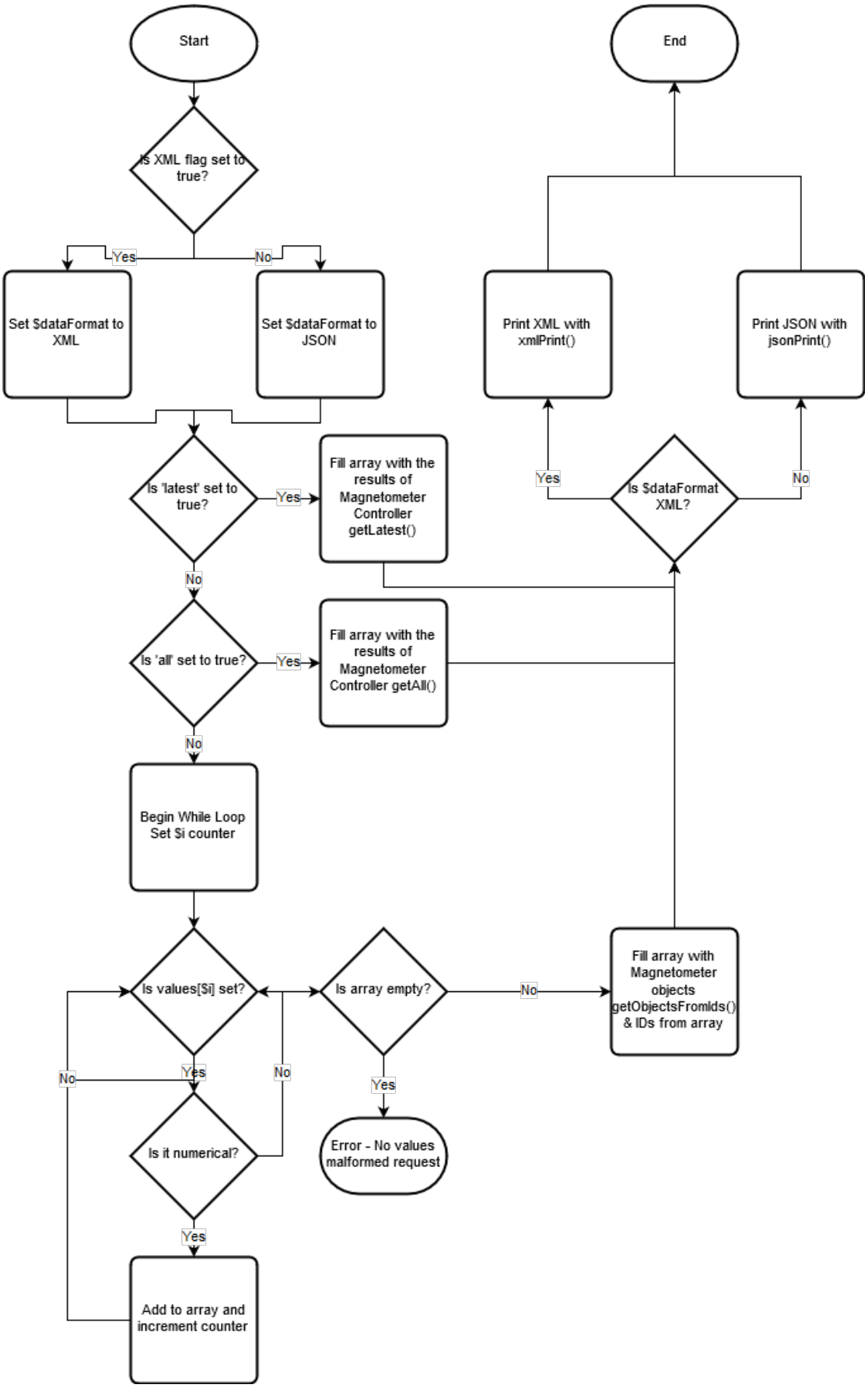


Figure 2-27 Handle Request Algorithm for API Sequence Diagram.

Both `jsonPrint()` and `xmlPrint()` follow a similar pattern of iterating through Magnetometer objects, but their approach is slightly different. `jsonPrint()` iterates through each object and uses its getter methods to build a new multidimensional associative array. This is converted and printed as JSON using the built in PHP method `json_encode()`.

```
private function xmlPrint($array)
{
    header('Content-type: text/xml');
    echo '<?xml version="1.0" encoding="UTF-8"?>';
    if (isset($array['error'])):
        echo "<error>{$array['error']}</error>";
    else:
        echo '<magnetometers>';
        foreach ($array as $magnetometer):
            if (!empty($magnetometer)):
                echo '<magnetometer>';
                echo "<id>{$magnetometer->getId()}</id>";
                echo "<timestamp>{$magnetometer->getTimestamp()}</timestamp>";
                echo "<value>{$magnetometer->getValue()}</value>";
                echo "<temp>{$magnetometer->getTemp()}</temp>";
                echo "<lastmodified>{$magnetometer-
>getLastModified()}</lastmodified>";
                echo '</magnetometer>';
            endif;
        endforeach;
        echo '</magnetometers>';
    endif;
}
```

Figure 2-28 XML Print Private Method, complete with PHP alternative logic syntax.

`xmlPrint()` on the other hand has embedded XML within the method which is echoed out with embedded values included using getter methods. This mixture of logic and mark-up is discouraged in web development unless it is necessary such as in templates. [73] Additionally, it uses PHP alternative control structure syntax to maintain the highest level of readability [74] despite being an assortment of code and in lieu of a dedicated templating language e.g. Twig that is used elsewhere in the program (2.12 User Interface Design). Also note the alternative use of single versus double quotes – for marginally more efficient code, double quotes should be used where necessary or where variable evaluation (detecting variables in strings) is required, such as in this method. [75] Nevertheless, as the method is single-purposed, well documented, and functions as intended, this should not prove problematic, even with large API queries.

2.8.12. DataSource and InternetData

DataSource is an abstract class that Magnetometer inherits from. It was intended that the majority of behaviour could be shared and would be stored within this abstract class that could be called upon or overridden by its children classes and for the polymorphic usage of data sources elsewhere in the program.

InternetData is a stub class that also inherits from DataSource. These two classes are explored in more detail in 3.7 Additional Data Sources.

2.8.13. Directory Structure

The directory layout and structure of the application is as follows:

- Root Directory
  - assets
    - fonts
    - images
    - scripts
    - styles
  - doc
  - locale
  - src
    - Config
    - Controllers
    - Database
    - i18n
    - Models
  - storage
    - csv
    - logs
      - magnetometer
    - settings
  - templates
    - partials
  - tests
    - Config
    - Controllers
    - Database
    - i18n
    - Models
  - vendor

The assets folder stores static resources used by the application, such as stylesheets, JavaScript scripts, images, and fonts.

Doc contains documentation generated by the phpDocumentation tool, which generates comprehensive HTML documentation based on the application's PHPDoc. (Explored more in Implementation)

Locale contains JSON files for each supported language – English and Welsh. The values contain localised versions for each key.

Src and its subdirectories contain PHP classes that make up the application.

Storage contains several types of data. The csv directory temporarily stores CSV files as they are processed by the Retriever class. The program error logs can be

found inside the logs directory, with magnetometer retrieval activity being logged into its own subdirectory and logfile.

Twig template files that are explored in 2.12 User Interface Design are stored in templates, with partial template files (includes) being stored in partials so that they can be reused by other views.

The tests subdirectory directly corresponds with the src subdirectory – containing PHPUnit test classes for each program class that has unit tests written for it. This is looked over in detail within 4.2.1 Unit Testing.

The vendor folder contains third-party libraries, tools, and dependencies retrieved by Composer, including but not limited to Logger, the Twig templating engine, and the SMB connectivity library.

2.9. Development Order

The order of development begins with core functionality within Magneto, Renderer, Config, and Connector classes to ensure a working prototype is ready in time for the mid-project demonstration. All other behaviour such as Retriever can be implemented afterwards, as where dummy data can be seeded and used beforehand since Renderer is intended to behave the same regardless of where data comes from. Testing is intended to be done alongside project implementation. (See fourth chapter, Testing)

2.10. Relationships and Dependencies

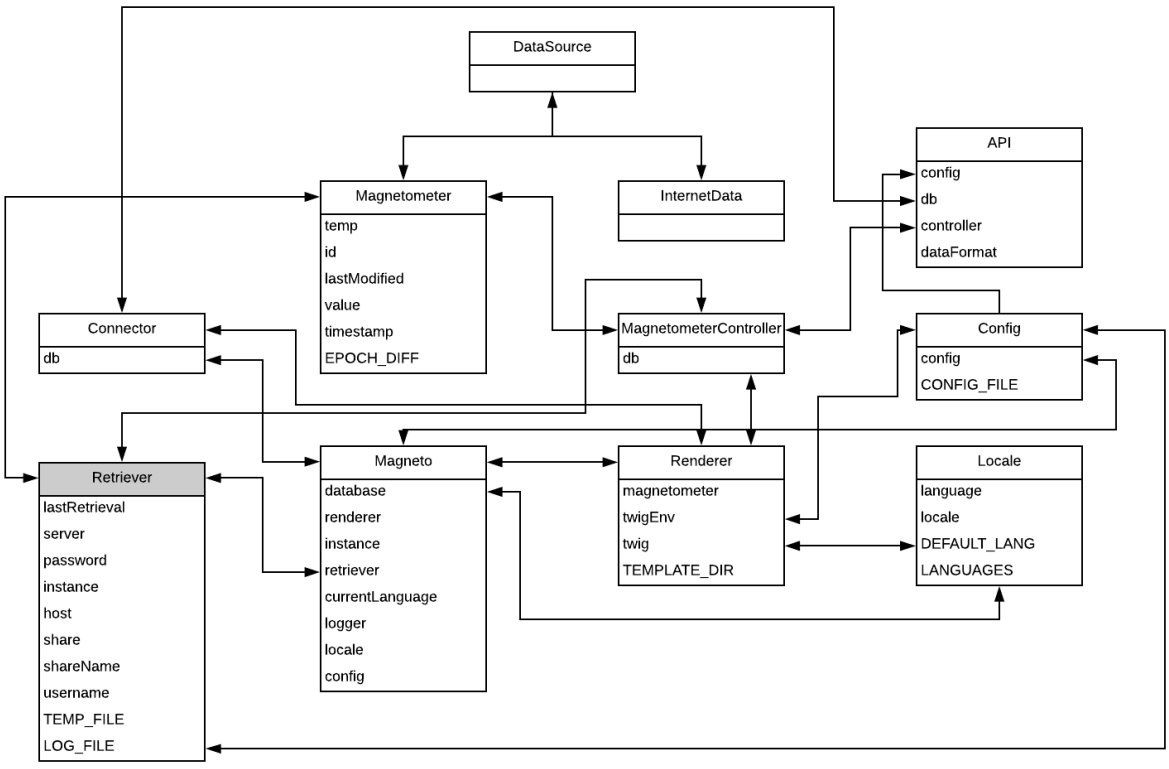


Figure 2-29 Class Diagram demonstrating relationships between classes.

This figure provides a glimpse at the dependencies of the application and how other classes depend on each other. Although it would appear at first glance to be chaotic, each relationship can be broken down into reasoning as to why it exists. The following table aims to do establish each class' dependencies, including third-party libraries not included in the previous diagram. This does not include dependencies on built-in PHP classes, such as PDO, or Exception.

Depending Class	Dependency	Purpose
Magneto	Connector	Initial database setup
	Locale	Initial language and locale setup
	Config	Initial config setup
	Retriever	Begin watchdog services
	Renderer	Begin routing services
	Logger Classes	Configure logging services
API	Config	Initial config setup
	Database	Initial database setup
	MagnetometerController	Database retrieval
Config	Magneto	Error handling
Connector	Magneto	Error handling
Locale	Magneto	Error handling
MagnetometerController	Magneto	Error handling, sanitisation
	Magnetometer	Corresponding model class
	Connector	Database retrieval
Magnetometer	Connector	Saving to database
	DataSource	Parent class
InternetData	DataSource	Parent class
DataSource	None	N/A
Renderer	Config	Injecting config into view
	Connector	Injecting instance into view
	Locale	Injecting locale into view
	MagnetometerController	Retrieving data and injecting into view
	Twig Classes	Templating Engine and rendering views
Retriever	Config	Updating latest entry in config file
	MagnetometerController	Querying database for existing records
	Magnetometer	Saving to database

	SMB	Magnetometer services and SMB connection

2.11. Use Cases

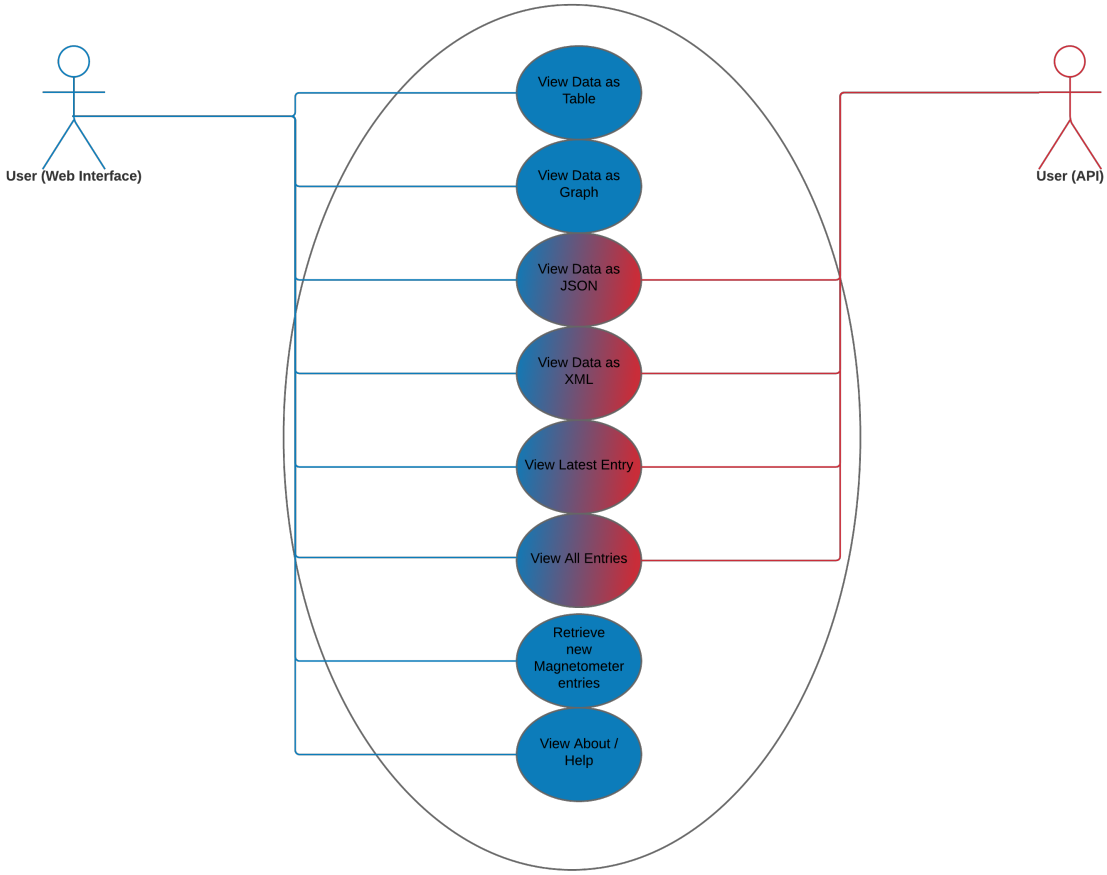


Figure 2-30 Web Interface vs API Use Case Diagram.

There are two identifiable primary use cases for the application. Although the program as previously discussed is written for students and researchers of IMPACS, whether the program is being accessed through its web interface or programmatically via its API presents different functionality, albeit all functionality present on the API is also present on the web interface.

The above diagram shows tasks in blue that are available only from the web interface: viewing data in tables, viewing data in graphs, retrieving new magnetometer entries, and viewing about / help.

Those with a red-blue gradient are available through both entry-points and are viewing data as JSON, viewing data as XML, viewing the latest entry, and viewing all entries.

2.12. User Interface Design

2.12.1. Introduction

The criteria for designing an interface were decided on early in development – the web interface must be simplistic, informative, and as responsive as possible. Responsiveness is a web design approach that was originally defined by E. Marcotte back in 2010 as a design that responds to the device and screen resolution the user is using. [76] Although the design did not change much throughout the project's lifecycle as designs could not be dictated by client desires, the design has changed slightly nevertheless.

2.12.2. Twig Template Engine

As deliberated beforehand, the Twig templating engine is a popular PHP templating engine developed by those behind the Symfony framework. [64] The reason behind adopting this engine for the project is due in part to familiarity with producing Twig templates, but also due to compatibility, as Twig 1.x will run on PHP 5.6 and does not require any frameworks, including Symfony. It will happily operate alongside vanilla PHP if initialised correctly. [77]

While there are numerous benefits for using a templating system over standard PHP or HTML files for views, ranging from caching functionality to simply easier to read syntax, [78] but the primary criteria of interest are template inheritance and forced separation of concerns. [79] Template inheritance means that template files can be easily reused and not just included but also inherited from using {% extends %} syntax. Blocks can be assigned into templates that will be filled in by included or child templates that share those blocks. For example, the following code is taken from the homepage template, which inherits the layout from the master template, such as the HTML `<head>` and its child tags, footer scripts, navbars, etc. This filling-in behaviour is additionally incredibly useful for `<title>` tags.

```
{% extends "master.php.twig" %}
{% block title %}{{ locale.about.title }}{% endblock %}
{% block content %}
    <div class="row">
        <main role="main" class="col-md-9 ml-sm-auto col-lg-10 pt-3 px-4">
            <div class="d-flex justify-content-between flex-wrap flex-md-nowrap
align-items-center pb-2 mb-3 border-bottom">
                <h1 class="h2">{{ locale.home.title }} <small> {{ config.appName }}
{{ config.appVersion }}</small></h1>
            </div>
            <h2>{{ locale.home.subtitle }}</h2>
            <p>{{ locale.home.introduction }}</p>
        </main>
    </div>
{% endblock %}
```

Figure 2-31 Homepage Twig Template.

If a developer wishes to access data within Twig templates, it must be passed as an entry in an array of parameters when Twig renders the view or assigned as a Twig global. Twig does not allow PHP to be executed in Twig files, nor does it allow

access to the global scope. From the above figure, the arrays `locale` and `config` can clearly be seen, whose contents are being printed onto the webpage to facilitate localised content.

For clarity, Twig commands sit in-between `{% %}` tags. To merely print the result of a command or variable, double curly braces `{{ }}` are used. For comments, double curly braces containing hashes are used. For example: [64]

```
{# There is a block coming up, and this is a comment! #}
{% block %} {{ variable }} {% endblock %}
```

2.12.3. Bootstrap 4 and Theme

When working as a solo developer covering all aspects of an application, both backend and frontend, 'business' logic and presentation, it is important to be as time efficient as possible. As designs were initially conceptualised it was already clear what kind of layout was desired for the application, which thanks to Bootstrap is incredibly straightforward to develop, pre-existing code, mark-up and styling were downloaded from the Bootstrap 4 Dashboard example. [80] This pre-designed template was of considerable interest as it is demonstrated working alongside Chart.js, and meets the established criteria of responsiveness, simplistic, and informative, as there is plenty of room for additional buttons and controls, as well as tabular data.

Responsiveness is made simple with the use of the Bootstrap 4, whose grid layout system is based on the new CSS grid system to maximise efficient use of screen 'real estate' (screen space) and adapt to changing resolutions, different device viewports, and browsers as necessary. [81]

2.12.4. Conceptual Designs

Two rough designs were produced on by hand on paper in pencil early in development (note the old internal name for the program present on the drawings, *Magneto*) once the Bootstrap 4 Dashboard template had been found, both were used in the initial brainstorming of the Graphing Tool view, and the Navigator view. Due to their large resolution (scanned A4 paper), they have been included as Appendix E.

2.12.5. Final Designs

2.12.5.1. Homepage and About

The homepage and about page views share mostly identical templates apart from their textual content.

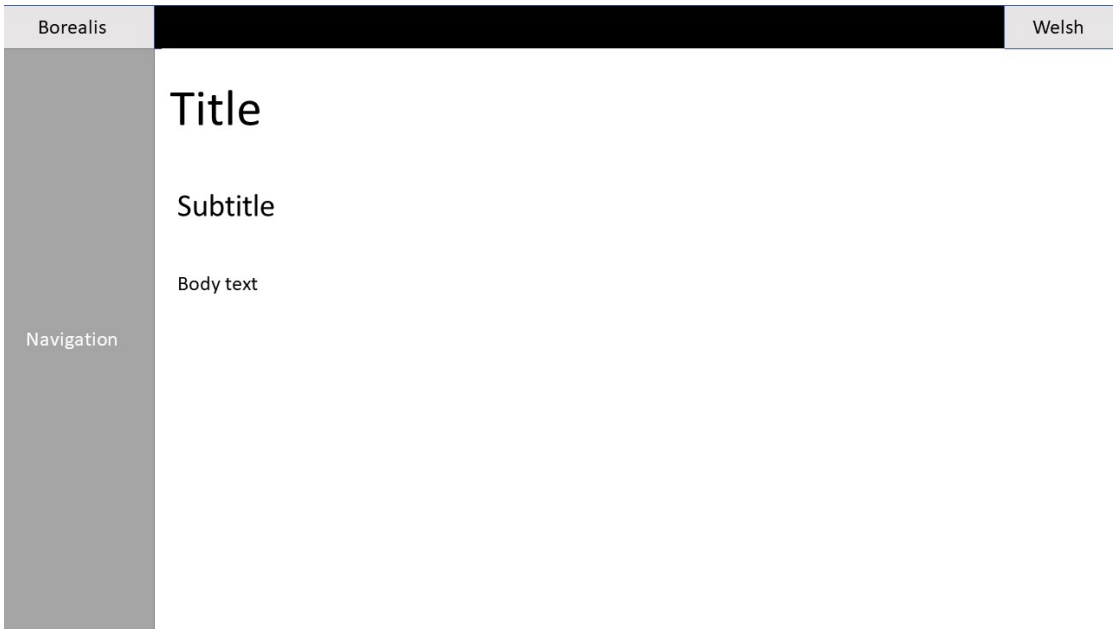


Figure 2-32 Homepage/About page Design

2.12.5.2. Navigator

The Navigator view is the view responsible for displaying all magnetometer entries available on the database, and as many entry selectors as defined by the config key `maxElements`. Originally, this was going to update by means of jQuery DOM manipulation, where the user could dynamically add and delete elements to choose as many entries as they wanted without going to the Settings page, but this was changed due to time constraints. Nevertheless, it is reflected on the final design. Buttons for adding amateur radio and Internet (ESA / European Space Agency) sources are also present, albeit remain as unimplemented functionality in the software.

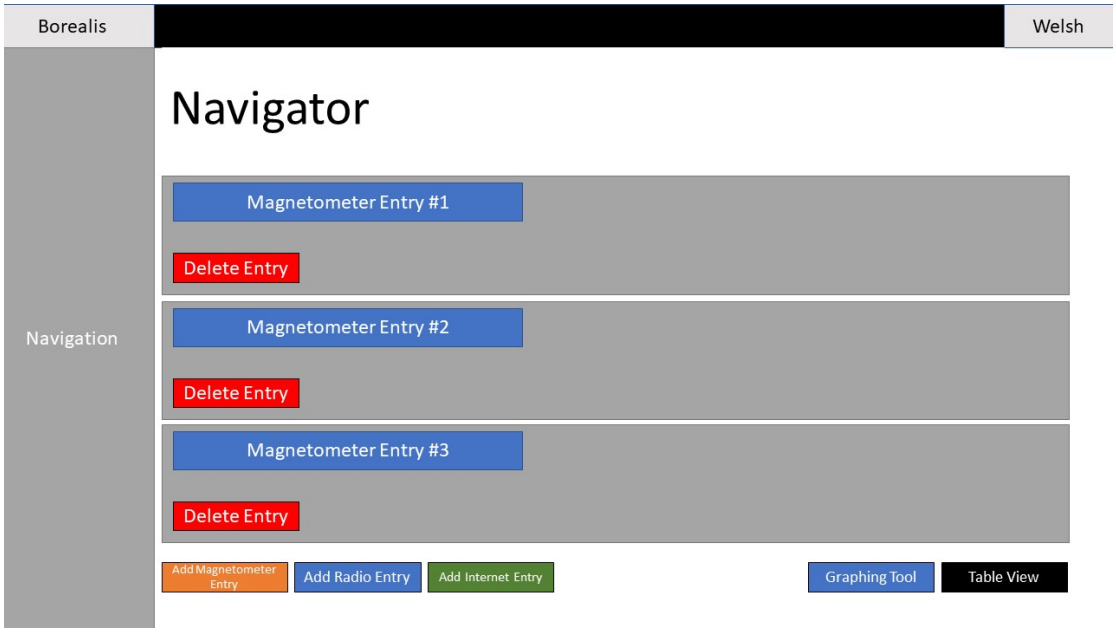


Figure 2-33 Navigator Design

The Navigator and the Graphing Tool contain a creative trick to ensure that the application is only ran with JavaScript support enabled. An inline style is placed on the containing element for the main body of the program, the navigator form in this case, or the Graphing Tool container in the case of the latter, rendering it invisible. A small line of JavaScript follows it that removes the property from the element, so the process is instantaneous and is unlikely to be noticed, except by those with scripting disabled. Instead, they are treated to a message requesting they reenable JavaScript by means of the `<noscript>` tag.

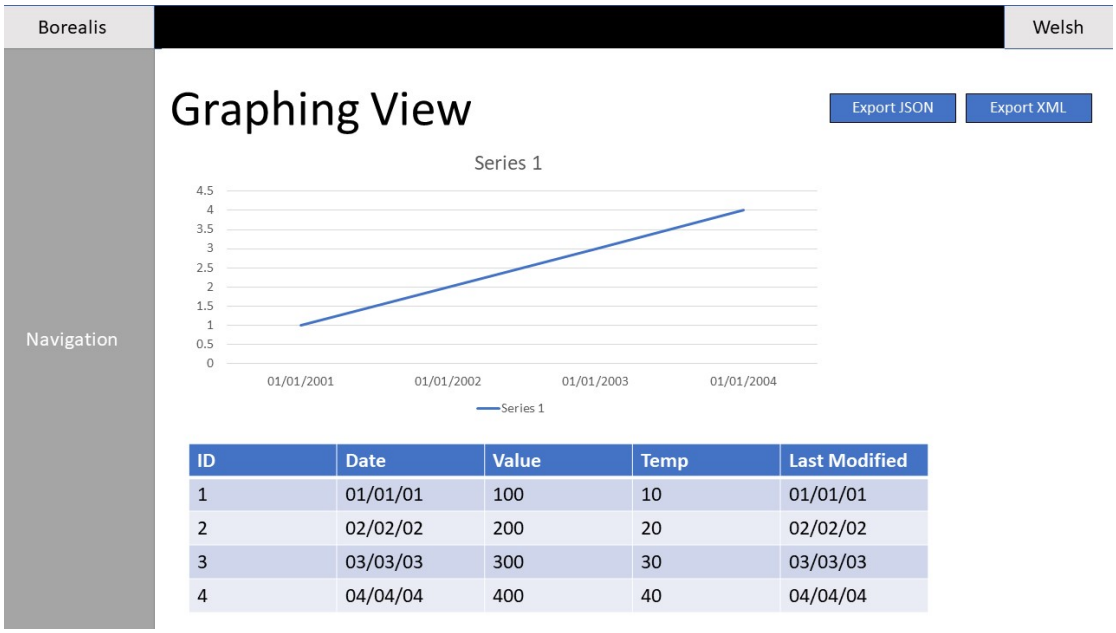
```
<div class="row">
    <div class="col-md-9 ml-sm-auto col-lg-10 pt-3 px-4">
        <noscript class="alert alert-danger">{{ locale.errors.javascript
}}</noscript>
        <form id="navigator" class="form-control" action="index.php?page=graph"
method="POST" style="display: none">
            <script type="application/javascript">

document.getElementById('navigator').style.removeProperty('display');
            </script>
```

Figure 2-34 Code snippet from Navigator Twig template demonstrating JavaScript enforcement.

2.12.5.3. Graphing Tool View

The Graphing Tool is one of the primary functions of the *Borealis* application – graphically rendering data using the Chart.js JavaScript library. The majority of screen real-estate is given to the chart, with the Tables view embedded underneath, and buttons to facilitate the exporting of data to JSON and XML format. Although not present in this final graph, it was decided later in development to add the functionality to switch between line and bar graph formats dynamically and an extra button was added to facilitate this.



ID	Date	Value	Temp	Last Modified
1	01/01/01	100	10	01/01/01
2	02/02/02	200	20	02/02/02
3	03/03/03	300	30	03/03/03
4	04/04/04	400	40	04/04/04

Figure 2-35 Graphing Tool Design

The script responsible for rendering the Chart.js graph is kept in its own Twig template file that is included onto the Graphing Tool view. JSON is inserted

directly into a JavaScript variable on the server-side before being sent to the user's browser.

```
<script type="application/javascript">
    var ctx = document.getElementById("chart");
    var data = JSON.parse('{{ json | raw }}');
    var values = [];
    var dates = [];
    for (var i in data) {
        if (data[i].x && data[i].y) {
            values.push(data[i].x);
            dates.push(data[i].y);
        }
```

Figure 2-36 Code snippet from Chart Twig Template, demonstrating code insertion

As visible from the above code snippet, the contents of the Twig array `json`, containing JSON generated by the MagnetometerController is inserted directly into the JavaScript variable `data`, processed by JavaScript's built-in JSON parsing function. The raw filter is used here to properly escape the JSON as Twig prints it to the DOM.

2.12.5.4. Tables View

The Tables view can be viewed independently or as part of the greater Graphing Tool view as additional tabular data underneath the graph. This is facilitated by either directing Twig to render the Tables View directly and by using Twig to insert it within the Graphing View template as an include.

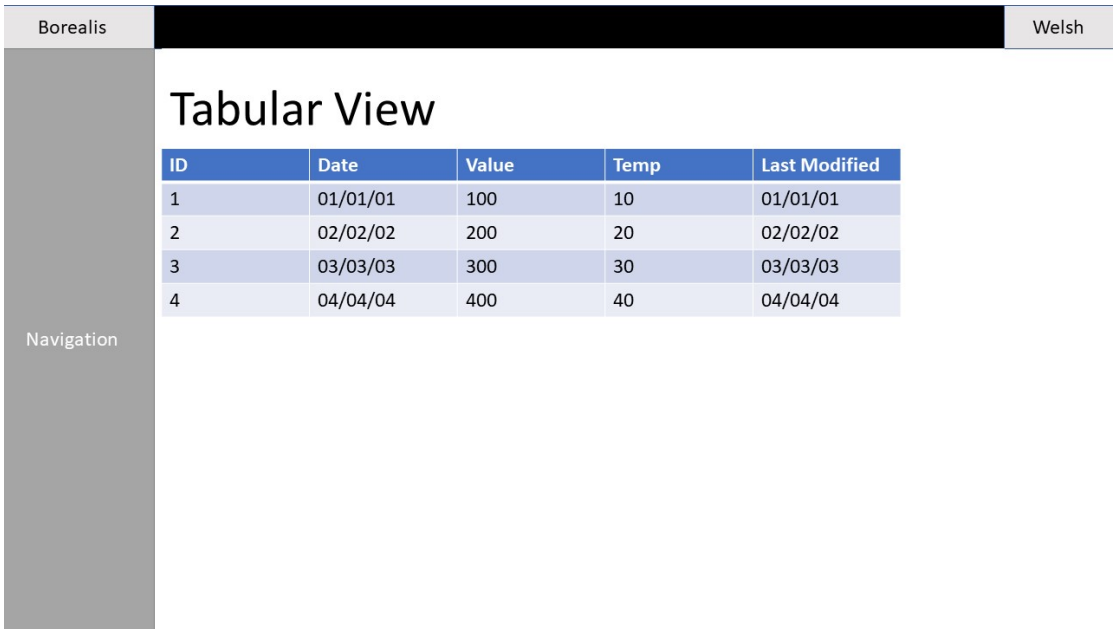


Figure 2-37 Tables View Design

2.12.5.5. Settings

The settings view currently serves to adjust the number of elements (`maxElements`) displayed in the Navigator view. Although additional behaviour was initially concepted and is explored in more detail in Implementation 3.2

Configuration Changes. Malicious tampering is mitigated with the inclusion of both client-side and server-side validation, and CSRF protection.

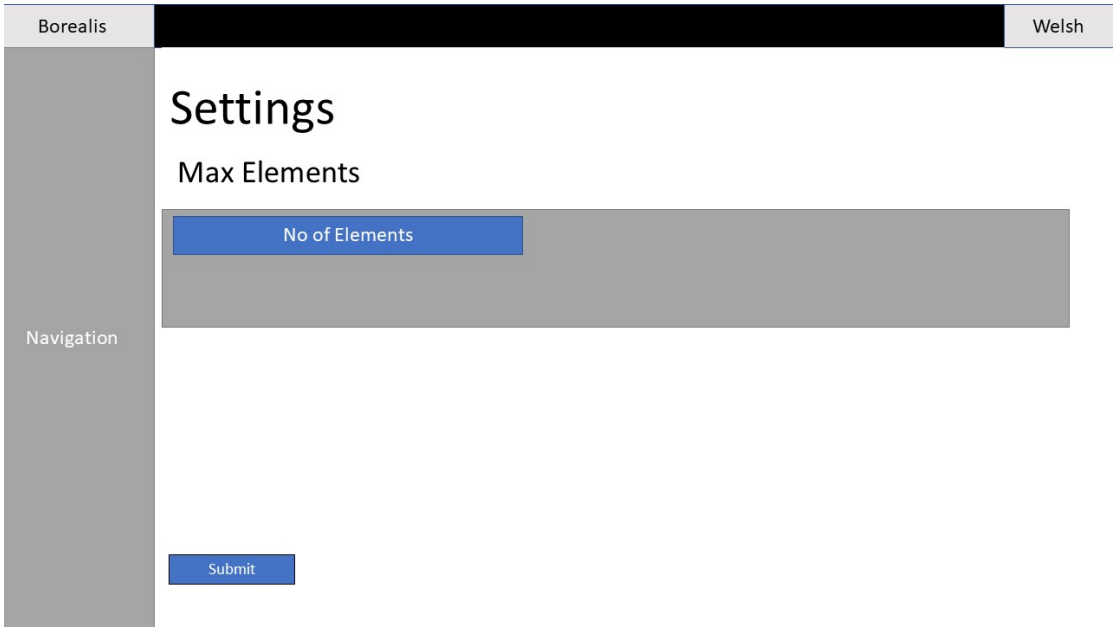


Figure 2-38 Settings Design

2.12.5.6. API

A miniscule amount of design went into the appearance of the API segment of the program, as it was intended just to return plain JSON that would be rendered by the web browser or captured by an API client or terminal-based application. Similarly, XML returned by the API was not intended to contain any XSLT (Extensible Stylesheet Language Transformations) [82] to format it. However, a simple mock-up was designed to show what data returned by it may appear like in a web browser:

```
Content-type application/json
{
'id' => '1',
'timestamp' => '01/01/01',
'value' => '100',
'temp' => '10',
'lastmodified' => '01/01/01'
}
```

Figure 2-39 API Design

3. Implementation

3.1. Introduction

In this section of the report, challenges faced during the implementation stage of the project as well as significant diversions from planning will be explored in detail. As previously covered in 2.8 Detailed Design, there is considerable overlap between the sections of Design and Implementation, however this section will focus more on specific issues and diversions from design that justify their own subsections.

3.2. Configuration Changes

As briefly touched upon in the Config subsection of Design 2.8.4, it was originally intended for the application to store its configuration changes in an INI file, a plaintext format file that is widely used to store settings for computer programs of all mediums, including web applications. [83] PHP has built-in functionality for the parsing of such a file into an associative array, but writing entries, or updating existing entries is a more challenging procedure and involves manually locating and writing to entries that would be required when working with any kind of plaintext file. [84]

Not wishing to write a substantial quantity of potentially unreliable code for writing to such a file in order to continue adhering as closely as was possible with the KISS principle, [63] a search for an existing library or code package that would provide this functionality was initiated. After a short while, a library known as Magicalex/Writeinifile was discovered and had all the required capabilities for working with INI files whilst also being compatible with PHP 5.6. [85]

Unfortunately, a lot of difficulty was encountered when trying to get the library to work as intended. It was easy to install using the Composer dependency manager, however writing to INI files repeatedly left missing values or malformed file structure. While this was possibly down to programmer error, there existed a drought of documentation available for the bite-sized library outside of its README document and frankly insufficient time available to painstakingly debug. In the end, a decision was made to refactor the Config module to use a far more crude, but knowingly effective approach at reading and writing to a configuration file. The use of a dedicated associative array PHP file.

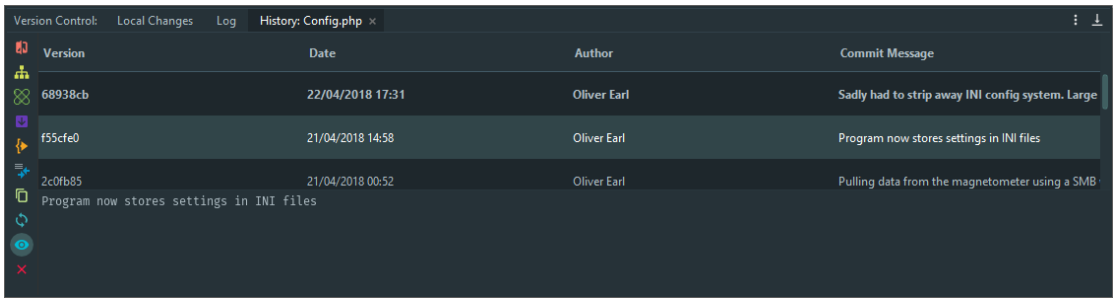


Figure 3-1 Git commits showing addition and removal of Writeinifile library displayed in JetBrains PhpStorm.

This method is widely used and liked by experienced developers such as C. LaViska who published an article on his company's website describing the step-by-step procedure and the benefits of using this technique. [86] However, the way in which data is written to it following changes is significantly more rudimentary, involving exporting the contents of the currently in-memory configuration and manually writing the remaining file syntax to the file itself.

```
private static function saveConfigToDisk()
{
    try
    {
        file_put_contents(self::CONFIG_FILE, '<?php return ' .
var_export(self::$config, true) . ';');
        return true;
```

Figure 3-2 Code snippet from Config Save Config to Disk Private Static Method

While this use of file writing and concatenation is a code smell, it was an unavoidable evil during development to ensure that a consistent pace could be maintained, and that more essential functionality would not be left unimplemented.

3.3. SMB Connectivity

During the initial stages of the project, prior to the refurbishing and system update of the computer connected to the magnetometer, data was accessed by means of FTP, making programmatic access simple, since PHP includes built-in functionality for the protocol. [20] When the system changed to incorporate SMB as its primary form of access, this meant that a great deal of research had to be carried out to figure out how retrieval was going to happen, and whether it was even possible.

Despite having redundancy plans including configuring software on the connected computer to POST data to the application directly, or to simply request that IMPACS restored access via FTP protocol, a silver bullet was discovered that made them unnecessary. A means of programmatically accessing SMB shares via PHP. This was the open source SMB library by S. Grabaum and R. Appelman, available for install via Composer. [69]

A spike work project was promptly assembled using sample code from the library's README, the magnetometer share details, and my user credentials to see whether it would work – the expected output was to see all available files within a chosen directory printed to the webpage. Overjoyed jubilation was to follow as the attempt was a success, and work promptly began to construct the Retriever class using this library, fully assimilating it into the project.

```
$auth = new \Icewind\SMB\BasicAuth($user, $workgroup, $password);
$serverFactory = new \Icewind\SMB\ServerFactory();

$server = $serverFactory->createServer($host, $auth);

$share = $server->getShare($share);

$files = $share->dir('2018');
foreach ($files as $file) {
    echo $file->getName() . "\n";
}
```

Figure 3-3 Code snippet from spike work using the SMB library.

If the SMB library had not worked as intended and the project following a more traditional Waterfall, design-heavy approach to software engineering, the project would have gone down a completely different route and would have required substantial design alterations to accommodate, particularly if FTP access could not be re-established. Thankfully in part due to methodology decisions, the project was always ready to adapt to differing means of data retrieval, especially given that this was not planned for concretely.

3.4. Retrieving Magnetometer Data and Data Depth

One unexpected difficulty came up whilst developing the Retriever class and the overall functionality that governs retrieving data from the magnetometer that had a pronounced impact on development.

As previously elaborated, the magnetometer records data to CSV files, with entries approximately once a minute. Each line in the CSV file, each record, contains a LabVIEW timestamp, a measurement in arbitrary units and temperature in Celsius. Files are named `FILExxx.CSv` where `xxx` represents a three-digit number, from 001 to 365, representing the day of the year, including trailing zeroes. As of this time of writing, there exists four directories on the magnetometer share, dating back to 2014, each containing a file for most days of the year, the result is a maximum of 1424 CSV files, each containing up to potentially 1440 entries, for each minute of the day. During a database seed, this may have meant that up to over 2 million entries would need to be parsed and entered into a database. This was in the developer's opinion to be too much data, especially given that for each entry, the timestamp would need to go through a two-step conversion first into a UNIX timestamp, and then into a standard date that would be recognised by the database.

This would also be difficult to represent at the interface. After just one day of database retrieval, one HTML selection box contained over at thousand entries for just one day. It is unlikely that a user would want to painstakingly select which minutes of the day they want to represent graphically. Rather, it is safe to assume they would rather select data for the entire day.

While this would be more than feasible to represent, and still retrieve data in its entirety, it was felt a more elegant solution was needed that would still accurately represent patterns and shifts without the need to save such a quantity of raw data. The algorithm was changed to determine the average value and temperature for each day (each CSV file) and record a single Magnetometer entry with these averaged values. It is hoped that despite this, trends and patterns in readings will still be pronounced when rendered in graphs.

```
private function oldProcessData($target) {
    // Need to go through file line-by-line
    if (!file_exists($target)) {
        return null;
    }
    $entries = array_map('str_getcsv', file($target));
    foreach ($entries as $entry) {
        $formattedDate = date('Y-m-d H:i:s',
Magnetometer::convertToUnix($entry[0]));
        $magnetometer = new Magnetometer(
            null,
            $formattedDate, // Date
            $entry[1], // Value
            $entry[2], // Temperature
            null
        );
        // Save it
        $magnetometer->saveMagnetometer();
    }
    return true;
}
```

Figure 3-4 Code snippet Previous algorithm for retrieving magnetometer data.

During the first attempted full seed of the database using this algorithm, the sheer amount of data transfer temporarily caused the web server to be knocked offline, forcing further development to be done on a different working environment. While it was never formally confirmed that the software was at fault for the downtime, no downtime or maintenance had been planned, and later telephone conversations with staff at the responsible body for IT (Information Technology) services at Aberystwyth University, Information Services, yielded further uncertainty as to the cause. In any case, on the 21st of April as the incident occurred, an apology email was sent to Information Services explaining the situation and sincerely apologising for any inconvenience caused. (Appendix F)

3.5. Overcoming PHP 7.x.x Conditioning

Giving up functionality that a developer might be used to in newer iterations of a programming language or framework is a challenge within itself. S. Ford at the Agile Alliance 2015 conference said that 'working with legacy code is almost

inevitable in today's development landscape'. [87] Although PHP 5.6 is still supported by The PHP Group and receives active security updates as previously explored [32], this mainly serves to support legacy applications and environments. The truth is that web development is a fast-moving environment that is dictated by the constant evolution of web technologies spanning the entire stack, as evidenced by the rapid evolution of the World Wide Web itself over the past few decades. [88]

PHP is a dynamically-typed language, allowing for the unprompted casting of data from one type to another, such as integers into strings, and vice versa. Many argue that this lack of strict typing allows for more flexibility when developing, but this lack of verbosity can lead to unintended side effects not easily picked up by a debugger or by an IDE – the trade-off manifesting itself as there being less syntax errors but harder to find logic errors. [89]

Scalar type hinting was a welcome addition to PHP 7, allowing for a wider range of possible type hinting. Similarly, the steep rise in popularity of TypeScript, a strongly-typed superset of JavaScript that transcompiles back into JavaScript for running in the web browser [90] is evidence that developers enjoy the benefits of type safety to develop efficient, safe code. Not making use of this functionality, as well as return type hinting, as it is missing in PHP 5.6 was difficult to cease.

Furthermore, the null coalesce operator makes an enormous difference when it comes to writing code. `isset()` and `is_null()` are two of the most used built-in PHP functions used throughout the application due to the importance of checking for `NULL` values and responding accordingly. In PHP 7, this is made incredibly more straightforward:

PHP 5: `$result = isset($foo) ? $foo : 'Not set';`

PHP 7: `$result = $foo ?? 'Not set';`

In short, programming was made more tedious by the continuous need to be aware that a legacy version of PHP is being used, and to be more rigorous with testing when data type cannot be guaranteed.

3.6. Template Debugging

For Twig versions 1.5 and above, an included dump filter allows the user to dump the contents of a Twig variable or results of a function to screen. This functionality was unavailable due to being restricted to version 5.9 of PHP, where the first major release of Twig was the only option. Resultingly, a custom filter bound to PHP's own `var_dump()` functionality was required to be created to handle this functionality manually. While this was not a major issue, it was a minor inconvenience caused by legacy software.

In addition, due to the minimal debugging functionality that was available within Twig, a debugging tool in the Graphing View was added and is only displayed when the program's debug flag is enabled.

3.7. Additional Data Sources

It was intended from the beginning of the project's inception that retrieving and displaying data from alternative data sources alongside magnetometer entries would be planned, but Complementary, functionality. When it became clear that this would not be able to be delivered even in a modest form without compromising the overall integrity of the program within the short time frame of the project, the primary focus was set on the magnetometer retrieval, and fragments of these abandoned features such as the abstract class DataSource and the stub child class InternetData can be found in the source code. These classes were initially intended, potentially alongside a Radio class, to become fully functional data classes with shared functionality and polymorphic logic.

The refocus on a singular data source however was not executed very efficiently, causing most of the code within what were intended to be agnostic modules, such as Retriever, to become primarily focused with the magnetometer classes and those alone. It would not be impossible to refactor the application to make stronger use of the abstract class and derivative children data source classes once again, but it will take longer and require more reverse-engineering than it would have done if implementation was done more carefully.

3.8. Navigator jQuery

As previously discussed in the Navigator Design subsection, it was originally intended for data selection panels to be dynamically added or removed using jQuery. This partially-working functionality was ultimately deleted in favour of manually specifying the number of elements to display when it became apparent that there was insufficient time to get the functionality finished.

3.9. PHPDoc

PHPDoc (sometimes stylised as PhpDoc) is an adaptation of Javadoc to help document PHP code. [91] Due to the belief in the importance of clear but well-commented code even in an agile workflow, all important classes, methods, and constructors have been provided with a PHPDoc block. Instructions are also provided in the project README on how to compile this into HTML using the phpDocumentor tool.

3.10. Review of Implementation Stage

The implementation was a blend of successes and things that could have gone better. While the majority of the program was completed to more than just an acceptable standard, despite the aforementioned issues that were encountered along the way, aspects could have been done better in issues ranging from maintainable code authorship to adherence to agile methodology through the implementation stage itself. Overall, the implementation was a success in spite of

the challenges faced by solo development. Further in-depth analysis will be provided within the Critical Evaluation section of this report.

4. Testing

4.1. Overall Approach to Testing

Testing is an integral part of software development and is a core activity of Feature-Driven Development's five-step process. Following Build by Feature, unit testing and inspection is typically carried out to ensure features are as bug-free as possible and work as intended before being merged into the software build. [29]

Testing throughout the project can be broken down into two main components. Software-aided testing, and manual testing. Software-testing consists of the use of unit tests and code linters to ensure functionality is not broken as features are implemented, and that code remains relatively clean. Manual tests consist of carrying out tests by-hand with both expected and extreme inputs on both a program-wide and individual module basis; the test, expected outcome and results of which were stored in a testing table. Testing on the API was also carried out.

4.2. Software-Aided Testing

4.2.1. Unit Testing

Unit tests were developed for the majority of public classes for each method of the application. The unit testing framework used is PHPUnit, originally developed by Sebastian Bergmann based on the xUnit architecture shared with the popular Java testing framework JUnit with the purpose of helping to quickly identify code regressions as code is committed. [92]

The structure of the testing directory, which contains PHPUnit tests mimics that of the source 'src' directory in this project, with tests in place of where a class file would be. Test classes have taken the same name as the class they are testing, but with a 'Test' suffix. Similarly, the test counterparts for class methods contain a 'test' prefix. An example is for the class Magnetometer, the testing class MagnetometerTest also exists, and for its method `saveMagnetometer()`, there exists a testing method called `testSaveMagnetometer()`.

PHPUnit tests can be run at any time with the command 'composer test' within the project directory.

Full unit test results can be found under Appendix G.

4.2.2. Linting

Linting is the automatic analysis of code for potential errors, poor stylistic and logical choices, and code smells in source code. This is an especially crucial step for scripting languages such as PHP or JavaScript as they are not compiled – a process that will often alert programmers to potential errors and poor structure.

For PHP code, the linting tool used is PHP Mess Detector, which will provide a wealth of information in the terminal window. It can be run using the command 'composer lint-php' – this is already preconfigured to run on the 'src' directory.

It was originally planned to include CSS and JavaScript linting tools, but due to the small amount of code in the aforementioned languages it was decided to omit them. They would also require to be installed using the Node.js npm dependency manager, which would add unnecessary complications to the program structure and ease of program configuration.

Full linting results can be found under Appendix H.

4.2.3. Difficulties Encountered

Tests were intended to be run on both on one of the development computers as well as on Central. This was necessary due to the firewall constraint explored within the first chapter of this report – tests ran on a development computer, even whilst connected to the campus network were unable to succeed in both establishing a connection to the magnetometer and to the database instance used. However, when attempting to run PHPUnit in the Central environment further problems ensued. As Composer was not installed globally, it was necessary to run it locally, but due to a problem seemingly involving the Linux system configuration Composer refused to run altogether. Attempts to manually run PHPUnit resulted in fatal PHP errors to do with directory resolution. As there was simply not enough time to diagnose and resolve these issues, only the set of results from a development computer have been included.

Linting revealed a multitude of stylistic and structural issues in the program source code, which should be rectified as soon as possible in future revisions of the program.

4.3. Manual Testing

4.3.1. Graphing Interface Testing

Manual testing of web application functionality was carried out. The behaviour was carried out using 'expected' data – data within the typical ranges that the program expects to receive.

The results have been recorded in Appendix I.

4.3.2. Stress Testing

Additional manual testing using parameters or scenarios that the program does not expect to encounter was also carried out – this is to test the program's fault tolerance and ability to cope with potential tampering, including CSRF violations.

The results have been recorded in Appendix J.

4.4. API Testing

Specific testing to the `api.php` application entry-point have been conducted on the program – this is stored in its own testing table with both expected and unexpected values. This testing was carried out using APITester.com, which supports both GET and POST requests supported by the program and can record the responses that are returned.

The results have been recorded in Appendix K.

4.5. Known Bugs

There are three serious known bugs that affect functionality and behaviour within the program that have are yet to be rectified and were identified using software testing. They may be fixed prior to project delivery, but as this is not guaranteed, they are documented here and are also reflected on in the testing results.

The first is regarding the magnetometer retrieval algorithm. Due to a potential flaw with how the application works with dates – formatting for them for use with magnetometer files, and then saving the date to the program's config file to mark the current day's retrieval as complete, the program may sometimes retrieve data more than once for a given day. This can lead to multiple entries, or more commonly errors indicating that a magnetometer entry already exists – this is an error message only intended to be seen when the user issues a manual override retrieval command and the latest entry already exists within the database. This bug presents no issues to program stability, but rather data integrity and must be rectified as soon as possible. It was found through manual testing and application usage.

The second bug identified is on the Graphing View. When working with a large amount of entries, the 'Export JSON' and 'Export XML' buttons can sometimes generate 'Request URI Too Large' errors. This is due to a flaw in the exporting algorithm. API paths are constructed using GET in mind, and too many entries can generate an exceedingly long URL that cannot be handled by the web server. To fix this, the buttons must pass information to the API using POST instead of GET. This bug was discovered during manual testing on the Graphing View using 'Display All'.

The third and final bug discovered pertains to the API. Even with properly constructed POST requests, the API almost always returns Request 400 Bad Request, which it is programmed to do when receiving malformed request. Additional refactoring is required in the API to ensure that this functionality remains working. This was discovered during API testing and is shown in Appendix K.

5. Critical Evaluation

5.1. Introduction

In this section of the report, a critical analysis and evaluation of the success of the project, from the initial research, through the combined design and implementation of the software application, as well as the efforts taken to test the program throughout all stages of development. Additionally, the agile methodology that underpins the project will be examined in detail to identify whether it was applied correctly and suited the objectives of the project.

Finally, both the notable achievements and shortcomings of the project will be identified and explored in detail, before providing an overall conclusion to the project and personal evaluation and mark estimation of the work carried out.

5.2. Identification of Requirements

Although the project began as flexible, broad topic area where requirements and use cases could be established by the developer themselves, provided certain criteria involving the usage of the FGM-3 magnetometer were met, the project's requirements were both quickly and thoroughly established.

It was quickly identified that the project should benefit the pre-existing users of the magnetometer and the data it provides; making it easier for said data to be accessed, explored, and/or manipulated. The data was quickly found to be cumbersome to work with, so the program should make it simple to access programmatically for use by scientific applications and data processing tools.

Throughout the initial identification and objective analysis however there consisted a slight grey area that caused some unnecessary lack of clarity. This was determining under what environment or computer system the application would be running under – whether it would be running on a system such as Central, or whether the application would be able to be run on a dedicated computer or virtual machine that still had unhindered access to a database instance and to the magnetometer itself by being located also behind the university firewall – which was one of the biggest constraints discovered in the process.

While this lack of clarity was very well worked around and the most safe cause of action was taken, and ultimately a good, working piece of software was produced, had these factors been more concretely investigated, software could have been built to leverage technologies that might have been available on the aforementioned dedicated platform, such as PHP frameworks like Laravel, or other programming languages perhaps better suited, such as Python with Django, or Ruby on Rails, in their entirety, rather than developing it in a legacy distribution of a language without fundamental dependencies on modern third-party frameworks.

5.3. Design and Implementation Retrospective

The design of the program went considerably well but was kept to the minimum mandated by Feature-Driven Development. After an initial plan and feature list were constructed, with basic plans for larger algorithms and patterns of behaviour constructed, Design by Feature could take place, with smaller algorithms and planned class-wide behaviour being modelled. Consequently, this report features plenty of detailed sequence and class diagrams elaborating the program's design and intended functionality. There is a large amount of overlap between Design and Implementation, namely due to the continuously evolving design that continued to model the actual implementation as development proceeded – this is a major causation for the significant overlap between the two segments within this report.

The implementation of the program solely on the other hand was not without its shortcomings. Although the program is functional and meets the requirements that were set forth in the initial stages of the project, code and the functional layout of the program were not consistently developed with sustainability, ease-of-maintenance, and futureproofing in mind. While some areas of the program, such as the Renderer, would continue to work in a continued way as the program evolved and was expanded upon in potential future releases, other areas such as the Retriever especially would require significant refactoring to accommodate other data sources. It was initially planned that development would concentrate on polymorphic design, so that all classes derived from DataSource would be supported, but as development pigeonholed onto magnetometer retrieval specifically, this ceased to be the case.

In conclusion, although development was successful and produced code of a sufficient, industry-level quality, and a functional application, more could have been done to ensure the implementation went smoothly and was more consistent throughout. Additionally, the design phase of the program should have documented changes as they evolved throughout implementation, rather than solely existing in the state they are now.

5.4. Suitability of Tools

The PHP programming language has been used for decades as a backend web application language, ranging from content management systems and blogging platforms such as WordPress, to RESTful services, and enterprise level applications.

Granted, many of these applications will run with a framework that leverage the power of the programming language whilst making complex applications easier to develop by means of control inversion and inclusion of complicated functionality. It is important to remember however that under the hood; the technology remains the same. PHP continues to be a popular language for the web, and while not as previously discussed as specifically designed for data processing as Python-based or Ruby-based solutions, the choice of language was sound.

Furthermore, JavaScript is the dominant language on the web, and by making use of the extensive Chart.js library and the HTML5 canvas, stunning, dynamic graphs can be generated using data retrieved from the backend.

Overall it is believed that the choice of technologies on all aspects of the application were solid.

5.5. Code Quality

Every effort was taken during implementation to produce excellent quality code. While the presence of some code smells is an inevitability even in large software companies, a sizable amount of effort was taken to ensure that code was self-explanatory, in adherence to the PSR coding standards as closely as possible, (in the case of PHP code) and was properly commented for future developers or maintainers to fully understand the purpose of each important function and class.

The use of linting tools brought to light additional issues with code that should be refactored as early as possible, particularly with the overuse of static methods in methods, which is a common code smell. Nevertheless, the program operates fine minus a few identified bugs. HTML mark-up within templates has been formatted with great care and conforms to the HTML5 standard as closely as possible. No issues have been identified with CSS or JavaScript code within the project.

It can be safely concluded that while there are some persistent smells in the source code of the project, the quality is particularly good, with a prominent level of documentation that is expected of enterprise or academic level software.

5.6. Project Achievements

5.6.1. Short Analysis of Data Findings

Following the completion of the Borealis application, interest was quickly garnered at the visual representation of magnetometer data it displayed. It was immediately obvious upon initial inspection of graphed data that while data was mostly consistent, there were notable events that are hypothesised to correlate to real-world events.

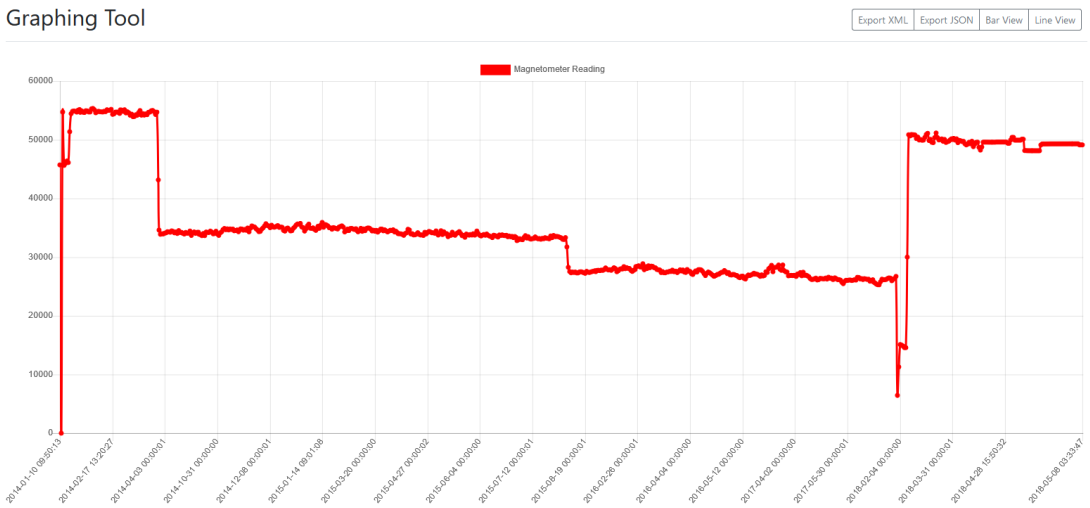


Figure 5-1 Display All View - Displaying All Magnetometer Data

The first notable period, besides an anomalous entry reading of zero is according to D. Price of IMPACS during the initial usage period of the magnetometer whilst it was being hosted in an astronomy hut in Frongoch Farm.

Eventually the magnetometer was moved to the Physical Sciences building on Penglais campus, Aberystwyth University. The decline in readings is reportedly due to electromagnetic interference present from the building, most notably from the building's elevator, causing the accuracy of magnetometer readings to decline substantially. Interestingly, after moving the instrument to another area of a laboratory, the reading declined again significantly, unbeknownst to researchers at the time.

The reasoning for the following short-term plummet in readings is not concretely known but is believed to be due to moving the device to another laboratory, potentially the underground Intelligent Systems Laboratory (ISL) – where it and the coupled computer system were refurbished and provided with a new operating system.

Following refurbishment, the entire system was returned to Frongoch Farm, where readings several years on are more reminiscent of the readings initially taken after its first deployment as the researchers knew by this point in time that more accurate readings would be available in an environment free of electromagnetic interference. The reason for the decline in value following re-deployment compared to its initial deployment is currently unknown.

The significance of these findings is that it validates one of the objectives of the application, and one of the desires for undertaking the project in the first place – providing substantial academic and/or scientific value to those working with the magnetometer or looking to track trends in magnetospheric activity in relation to solar activity, or perhaps more humorously, its location within a laboratory.

5.6.2. Adaptability

The project, as explored in-depth throughout this report has presented a large amount of challenges, particularly during the implementation stage. Due to the adherence of the Feature-Driven Development agile methodology and not the use of concrete, Waterfall-like methodologies, rapid adaptations were able to be made when circumstances changed, such as the indefinite postponing of additional data source implementation due to time constraints and difficulties with the magnetometer functionality.

As such, rebounding from issues such as non-working libraries and working within the bounds of such tight software constraints and nevertheless producing a piece of software that successfully fulfils its desired features and functional requirements demonstrates a great deal of agility and adaptability; themes that are personally believed to be present in all stages of the project and contributed substantially to its success.

5.6.3. Emotional Resilience and Self-Motivation

J. Atwood, most notable as one of the cofounders of the popular programming discussion message board Stack Overflow warns us of the failings of solo development in his blog: 'Working alone means complete control over a software project, wielding ultimate power over every decision. But working on a software project all by yourself instead of being empowering, is paradoxically debilitating.' [93]

An academic undertaking such as this project is not an easy feat. Unlike most real-world scenarios where groups of software engineers will work together, quite often in pairs or agile teams to accomplish work, working on something independently means that one person is solely responsible for ensuring that work is continuously delivered and that all features or functional requirements are met. Furthermore, said person is responsible for all aspects of the project, from analysis, to design, through implementation and testing.

Throughout the project, despite difficulties balancing its necessary workload with additional responsibilities, namely assignment work issued from the Department of Modern Languages, part-time employment, voluntary undertakings, and the day-to-day routine and requirements of undergraduate life, the project has been delivered in a punctual manner, and the developer has remained resolute, self-motivated, and enthusiastic throughout.

5.7. Project Shortcomings

5.7.1. Discipline in Agile Methodology

Although Feature-Driven Development was argued to be a more appropriately fitting methodology for this project over Scrum, which was initially proposed as the working methodology, the adherence to the methodology was not perfect.

While the overall adherence to the methodology was mostly successful and was as previously described a major contributing factor to the general success of the project, there were lapses in developer discipline, where the principles and guidelines of the methodology were not consistently followed. As time constraints became more pronounced and supplementary functionality was forced to be put on hold, there became more of an emphasis on producing working code and ensuring functionality was delivered, rather than strictly abiding by the agreed development methodology. A noteworthy example of this was the inconsistency of unit testing and the implementation of testing in its entirety.

In Feature-Driven Development, unit tests are expected to be produced along with features in the Build by Feature stage of an FDD iteration. Although some smaller classes do have comprehensive unit test files built in PHPUnit, some of the larger, more significant classes do not have as thorough counterparts. In defence of the implementation process however, this was partially due to a limit of public methods that could be tested, and not having anything more than a beginner's understanding on how to set up tests to work with protected or private methods provided further limitations. As tests were produced lackadaisically and almost always retroactively, this hindered obedience to the methodology and hurt development overall.

On an additional note, further research has indicated that agile methodologies specifically adapted for use by solo developers exist, almost all of which are derived from Extreme Programming, where the focus on pair programming is shifted toward aggressive refactoring and working through problems systematically. [94] One such suggestion was made to incorporate rubber duck debugging – the act of talking aloud, often to a rubber duck or other inanimate object to help work out the cause of a problem, to mitigate the need for pair programming. [95]

5.7.2. Workflow and Continuous Integration

It was initially planned to make use of JavaScript workflow tools, particularly task runners such as Grunt in development – taking care of the minification (and transcompilation if using superset languages like TypeScript) of JavaScript and CSS code, the automatic linting and tidying of PHP code, and using Git triggers to ensure that unit tests pass before code can be committed to the repository to help prevent code regressions.

The use of a fully-pledged IDE negated the need to use this kind of workflow as it itself can be configured to carry out tasks automatically and to run tests or use its built-in linter prior to committing changes. This meant that a Node.js installation wasn't required, or an associated `package.json` file was not required to use now redundant JavaScript workflow tools. Furthermore, CDN usage was opted over the use of downloading and minifying (or transcompiling) of frontend frameworks, particularly Bootstrap, which ultimately makes the project easier to maintain.

While this all would appear to be good news, it did mean the project had no Continuous Integration (CI) tools or configuration in place. This means that unless PhpStorm with the correct project configuration files were in use, which are files that are almost always excluded from Git repositories, then there would be nothing preventing a developer from committing inadequate quality code or regressions to the repository.

This did not have a massive impact on the project as it was developed by a solo developer, but it could have been catastrophic for larger or growing software projects. In the future, developers or maintainers should consider adopting workflow tools such as task runners or watchdogs like Grunt or Gulp to carry out automatic tasks, and to use Git triggers to ensure certain criteria are met prior to allowing code to be committed.

In essence, the developer did good to make effective usage out of the powerful tools provided by the IDE but should have taken more time to set up the aforementioned tools in order to ensure development remains sustainable, particularly if it is taken on by other developers in the future.

5.7.3. Further Improvements to Testing

Aside from the previously discussed inadequacies in consistent unit test production, more types of tests could have been produced to help constitute a more thorough approach to application testing.

It was established that the targeted user for the program are academic researchers and students within IMPACS. Acceptance testing carried out with the target audience would have provided valuable information as to whether the current state of the system and its user interface were acceptable in their eyes.

Further to the discussion of user interface, testing tools such as Cucumber could be used to programmatically test user interfaces, where tests are written to emulate user behaviour when interfacing with a GUI.

In conclusion, while testing was sufficiently thorough for the project, much more could have been implemented by the developer to provide more comprehensive testing data, and what was carried out could have been accomplished in a far more efficient, agile manner.

5.8. Project Conclusion

The title of the project is 'The Collation and Graphical Representation of Magnetometer Data to Track Solar Activity Using a Web-Based Application' – broken down into its key components, the following can be established:

- The project collates and graphically represents magnetometer data
- This data tracks solar activity, or corresponds to solar activity
- The application is web-based

The project goes beyond the collation and graphical representation of data; providing additional means to programmatically access data purely for the benefit of academic researchers and students. While the choice of the term 'solar activity' is perhaps in hindsight questionable, the magnetometer exists almost exclusively for this purpose – recording the Earth's magnetosphere in response to solar events and activity. The application does not itself provide any correlation or comparison of recorded data to solar activity, nor does it record information regarding such events – it purely records data from its data source. If the program had implemented other data sources, as may be the case in future revisions, it would record data from those also. The Borealis application is thereby an extension of the instrument's purpose – making data recorded for monitoring solar activity more accessible, or graphically visible. The application is also of course, web-based, and thus this criterion is also met.

If it were possible to start the project again, more focus would be put into ensuring disciplined adherence to the chosen agile methodology, which would likely be one of the previously discussed flavours of Extreme Programming better suited for solo development.

If it were possible to extend this project by another month or so, an aggressive refactoring to eliminate code smells and difficult to maintain code would be pursued; opting to go back to the initial polymorphic design incorporating abstract classes.

If the outcome of the marking of the project was at the developer's discretion, it would be graded as first class. The reasoning is as follows:

- Although the loss of additional data sources has sizably shortened the complexity of the application, the intricacy of the algorithms behind downloading, processing, storing and later reusing data from the magnetometer are interesting, well-documented, and effective.
- The development process has adhered to the agile process as much as possible, whilst also incorporating common software engineering principles such as KISS.
- The software, while being as self-documenting as possible, is additionally well documented.
- This report has hopefully been an interesting and inciteful read into the ability and habits of the software engineer responsible.
- The project has the capability of providing future use to IMPACS.

To finalise this write-up, the application was a fulfilling and enjoyable experience to implement and provided an invaluable learning experience. This project was in the eyes of the developer an undeniable success and fulfilled all criteria that it needed to meet. The undertaking was the perfect contrast between challenging and occasionally stressful, whilst also a strong opportunity to learn more about object-oriented web program development, APIs, and working in such confined conditions and legacy code that are highly likely to be real-world scenarios in industry. The steps that were taken to accomplish this project were careful, even if they were not always meticulously planned.

While at the beginning of the project feelings ranged from mild apprehension to what could only be described as downright terror at the thought of attempting to construct a program and produce designs of this magnitude, but in hindsight they were simply natural anxiety manifesting because of undertaking something new when it was important to also succeed and to produce a high-quality end-product.

6. Appendices

A. Third-Party Code and Libraries

Bootstrap 4.0.0; Dashboard Template
License: MIT
Description: HTML/CSS Framework

jQuery 3.3.1
License: MIT
Description: JavaScript library for DOM interaction.

Chart.js 2.7.2
License: MIT
Description: JavaScript charting/graphing library.

Sgrabaum/SMB
License: MIT
Description: PHP wrapper for smbclient and libsmbclient-php

Twig 1.35.3
License: BSD-3-Clause
Description: Flexible, fast, and secure template engine for PHP.

Monolog/Composer 1.23.0
License: MIT
Description: Logging and logfile library for PHP.

Composer / Composer Autoloader
License: MIT
Description: PHP Dependency Manager and autoload generator.

PHP Mess Detector (phpmd) 2.6.0
License: BSD-3-Clause
Description: PHP Linting tool

PHPUnit 5.7
License: BSD-3-Clause
Description: Unit testing framework for PHP

B. Ethics Submission

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
ole4@aber.ac.uk

Full Name
Oliver Earl

Please enter the name of the person responsible for reviewing your assessment.
Reyer

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department
cs

Module code (Only enter if you have been asked to do so)
CS39440

Proposed Study Title
The Collation and Graphical Representation of Magnetometer Data to Track Solar Activity Using a Web-Based Application

Proposed Start Date
29/01/2018

Proposed Completion Date
04/05/2018

Are you conducting a quantitative or qualitative research project?
Mixed Methods

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
The project is designed to make the retrieval of data from a magnetometer owned by IMPACS stationed near Penglais Farm substantially easier, by autonomously retrieving data from it and making it available through its web-based graphing interface, as well as through an API that can provide data in the form of JSON.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Not applicable

Will appropriate measures be put in place for the secure and confidential storage of data?
Yes

Does the research pose more than minimal and predictable risk to the researcher?
No

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?
No

Please include any further relevant information for this section here:
Not applicable.

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.
Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.
Yes

Please include any further relevant information for this section here:
Not applicable.

C. Undergraduate Classwork by D. Langstaff

This appendix is a piece of university coursework aimed at second year Physics undergraduate students at Aberystwyth University by D. Langstaff. The worksheets come from the module PH25520 Experimental Physics in 2014.

The appendix is three pages long and begins on the next page.

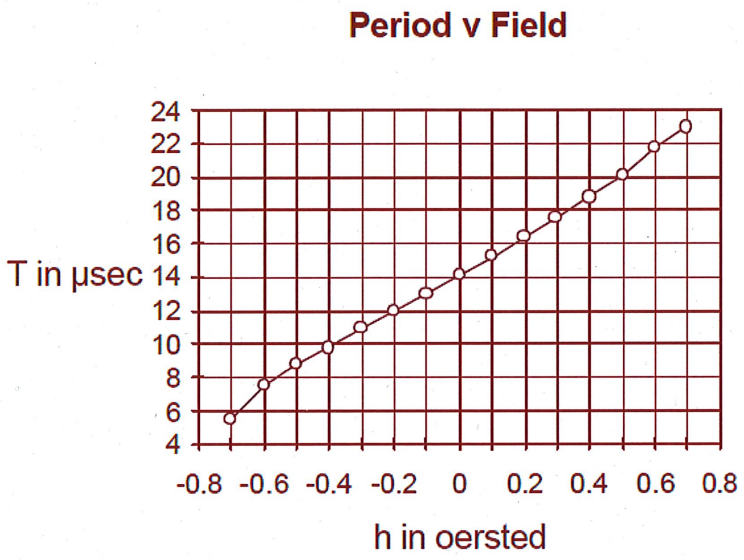
PH25520 Experimental Physics

Measuring the Earth's Magnetic Field

The aim of this experiment is to measure fine variations over time as they occur in the Earth's magnetic field and to attempt to correlate this to measurements taken by other researchers.
A magnetometer has been produced using a simple FGM-3 magnetic sensor, full details of which may be obtained from the manufacturers website http://www.speakesensors.com/ .

The magnetometer is installed in the astronomy hut at Frongoch farm, to minimise interference from stray magnetic fields.

The output from the sensor is in the form of a 0 to 5 volt square wave, the period of which is proportional to the magnetic field, as shown in the graph below.



Please note: 1 oersted $= \frac{10^3}{4\pi}$ A/m. The magnetic field should be measured using S.I. units. In this graph the Field intensity h is measured in an old unit which is no longer in use.

There are a few snags with using an inexpensive sensor such as this. Firstly, given that in order to study fluctuations in the Earth's magnetic field a resolution in the order of nanoteslas is required, from the graph it can be deduced the period would need to be measured to a resolution of around 10 picoseconds; which is not easy (or cheap). Secondly, if the output from the sensor is observed on an oscilloscope it becomes clear that the period of the waveform is by no means stable enough to measure with any degree of accuracy due to noise produced by local electrical activity. Nevertheless, these problems can be overcome by taking an average of the period of the waveform over a relatively long time; in this instance 3 seconds. Over this period of time, the random noise will average out to something approaching zero, hopefully leaving just the original signal. The equipment that you will

be using achieves the averaging process by means of a digital system, the internal workings of which it is not necessary for you to understand.
The Digital system interface with a computer via a minilab pod connected via USB. Data is collected via a Labview program installed on a dedicated computer used for the experiment. The data from the digital circuitry comes in the form of a 16-bit binary number at a rate of one sample every 3 seconds is saved to a computer by a LabVIEW program which is already provided. The data has been scaled so that the linear part of the field is covered within the range of the 16-bit number; where 0 equates to an output frequency from the sensor of 120 kHz and 65534 (216) to just under 50kHz. The program also provides a real time display of the raw field data as well as the temperature, which is recorded at the same time as each field sample.

Data logging

Recording to file is optional; if it is selected, data will be written into a .csv file in C:/magdata/<year>/. To keep the size of the files reasonable, a fresh file is created each midnight. The filenames are of the format "DATAxxx.csv", where xxx is the day of the year, with 001 being January 1st. the filename of the current output file is displayed on the front panel of the instrument.

The file format is as follows:
Each line of the file contains the readings from a single timepoint.
The values in the file are separated by commas. The fields within each line are:

1. A timestamp, indicating the number of seconds elapsed since 12:00 a.m., Friday, January 1, 1904.
2. An integer representing the observed magnetic field intensity in arbitrary units.
3. The temperature of the instrument, in °C

Data may be retrieved from the instrument by using WinSCP or some other FTP client. The instrument machine is at the address "ugexppc15.dph.aber.ac.uk" and logging in with the username "Magnetometer" and password "magnetometer".

Data may also be emailed to experimenters as each file is closed at midnight. To do this, simply put a list of email identifiers in the appropriate box on the "Setup" tab of the Labview instrument.

The instrument PC has been set up to allow remote desktop connection, with your usual username and password.

Temperature Compensation

Because the resolution is now in the nanoteslas range (the exact resolution is for you to determine), temperature drift within the sensor is much more significant. Thus the data which the magnetometer provides will need considerable postprocessing. In order to obtain the resolution from the instrument necessary to study variations in the earth's magnetic field, it is necessary to correct the output of the sensor for variations in temperature in the instrument's environment. In order to do this, data from several days' observations should be combined together and used to produce a plot of indicated field vs temperature is produced. If it is assumed that variations in the magnetic field will average out over time, this curve can then be used to produce a function which will enable the field data to be corrected for temperature. The magnetic sensor contains a single bipolar junction transistor and consequently the relationship between temperature and the raw field data is an inverse log law.
Once the data has been corrected for temperature, information from the manufacturer of the FGM3 magnetic sensor can be used to convert the arbitrary numbers indicating the output frequency of the sensor into conventional units, such as nanoteslas.
You should note that changes occurring in the data are more significant than the absolute value of the field at any time.

Comparison with other data

There are a large number of stations around the world monitoring the earth's magnetic field and some of these produce data in a form suitable for download.

Suitable starting points for internet searches to find appropriate data are the World data centre for Geomagnetism (http://www.wdc.bgs.ac.uk/), the United States Geological Service National Geomagnetism Program (http://geomag.usgs.gov/) and Intermagnet (http://www.intermagnet.org)

You should find data from one of the ground stations closest to Aberystwyth (Hartland seems a good bet!) and try to get data in a format you can then process to find similar features to that shown in your local data.

Topics for Background Research in Report

In addition to carrying out the experiments detailed above, your experimental report should also contain some background research into some or all of the following topics: Flux gate magnetometer, geomagnetic field, magnetosphere, ionosphere, fluctuations in earth's magnetic field, SWARM.

D. FGM-3 Magnetometer Technical Documentation

This appendix is a comprehensive technical document explaining the technical underpinnings of the FGM-3 magnetometer sensor.

While it is twenty-two pages long, it has been included for the reader's convenience should they wish to learn more on the inner workings and functionality of the instrument, particularly since the materials are difficult to find online. It begins on the next page.

Speake & Co Llanfapley
6, Firs Road * Llanfapley * Abergavenny * Monmouthshire * NP7 8SL * U.K.
Tel/Fax 01600 780150 *
email * billspeake@btconnect.com

FGM-series
Magnetic Field Sensors

SCL

- +5 volt operation
- Three Terminal Devices
- DC to 20 KHz Bandwidth
- Low Temperature Sensitivity
- High Intrinsic Sensitivity
- Built-in feedback coils

Description

The FGM-X series of devices are very high sensitivity magnetic field sensors operating in the ±50 microtesla range (±0.5 oersted). They are simple, essentially three terminal devices, operating from a single +5 volt supply, the connections being ground, +5v and output. The output is a robust 5 volt rectangular pulse whose period is directly proportional to the field strength, (giving a frequency which varies inversely with the field), making it very easy to interface to a computer or micro controller. The typical period swing for the full range of an FGM-3 is from 8.5 μs to 25 μs (~120 KHz to ~50KHz), a clear indication of its high sensitivity.

The FGM-1 is essentially a miniature version of the FGM-3 with a slightly lower sensitivity.

The FGM-2 is a single package containing two FGM-1 sensors at right angles to one another.

Unlike Hall Effect field sensors, which are virtually unusable in this range because of their large temperature sensitivity, the FGM-series has a very low temperature coefficient.

Since the lowest effective Nyquist sampling rate is ~50 KHz, appropriate filtering can provide an AC field bandwidth from DC to ~20kHz.

Maximum Supply Voltage	**7 volts**
Recommended supply Voltage	**5 volts ±0.5 volts**
Typical Supply Current	**12 mA**
Operating Temperature Range	**0 - 50 °C**

1

Typical Applications

Since the range covers the earth's field magnitude, multiple sensors can easily be arranged to provide compass orientation or full three-dimensional orientation systems, using the local earth's field as a reference, (-gimballed compass or virtual reality helmet devices.)

Other applications include conventional magnetometry, earth field magnetometry, ferrous metal detectors, internal vehicle re-orientation alarm sensors, external vehicle or ship passage sensors, wreck-finders, non-contact current sensing or measurement, conveyor belt sensors or counters and in conjunction with small permanent magnets, movement and proximity sensors and ferrous impurity detectors for non-magnetic alloys.

Conventional magnetometry would include magnetic material measurement, BH loop measurement, Preisach displays and archaeological artifact assessment.

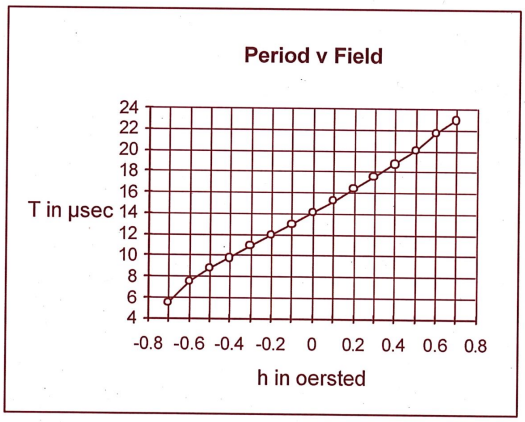
For use with applications requiring a larger range of field strength, external feedback winding techniques can increase both the sensor's maximum range and its linearity. This approach is described in more detail in the separately available "Application Notes" but basically consists of using an overwound solenoidal coil in a negative feedback loop which continuously attempts to zero the field seen by the sensor. By this means the range and the linearity cease to be a function of the sensor characteristics and depend only on the feedback current through the coil.

For added convenience, the miniature versions, FGM-1 and FGM-2, have such a coil built into their normal structure.

Output Period as a Function of Field Strength

Typical Characteristics

The chart below shows the typical response of the larger size sensor of the range, the FGM-3



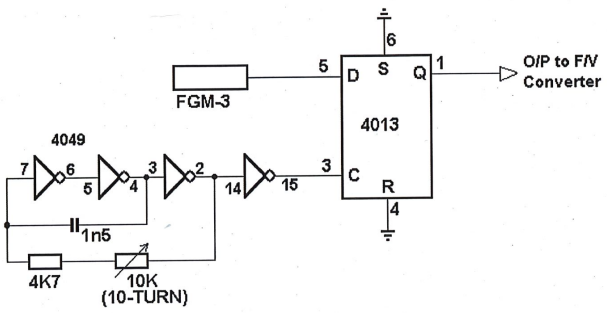
Between ±0.5 oersted (±50 µtesla) the non-linearity is about 5.5%.

2

Digital Heterodyning

Where the full range of the sensor is not required, such as in ferrous material detection or the measurement of small field fluctuations, a technique of digital heterodyning is useful. This is not a true heterodyne process but has close similarities when used over small frequency ranges. It is more akin to the production of aliasses by undersampling and gives very high sensitivity to small signal fluctuations. This makes it very for the remote detection of moving ferrous objects or for the measurement of the earth field fluctuations which accompany magnetic storm activity.

The technique requires a stable but adjustable source of clock pulses of similar frequency to the sensor output. These are used to undersample the sensor output and produce a much lower frequency square wave. One easy way to achieve this is to use the sensor output as the D-input of a D-type bistable and the clock source as the trigger input as shown below.



Digital Heterodyne

The sensor is used in a fixed position and the clock signal is set to a frequency close to the sensor frequency. The output of the bistable is a square wave of frequency equal to the difference between sensor and clock frequencies, similar to a heterodyne mixer.

A small percentage change in the sensor frequency becomes a large percentage change in the bistable frequency. This can be converted to a voltage as before, for meter or chart recorder, but gives a large increase in apparent sensitivity without the need for high gain amplifiers.

The configuration of the CMOS oscillator above is more stable than most and is suitable for such applications as ferrous metal detectors. Some care is needed in decoupling both the oscillator and the sensor to prevent a tendency to frequency lock, if the highest sensitivity is required. However such an arrangement has successfully detected passing vehicles and a measure of its sensitivity can be obtained from the fact that it could pick up a motorcycle in the far lane of a three lane motorway from the grass verge.

The more critical applications of earth field or materials magnetometry need a more stable oscillator, such as a crystal controlled type, but a fixed frequency is normally adequate, since such instruments are not usually mobile.

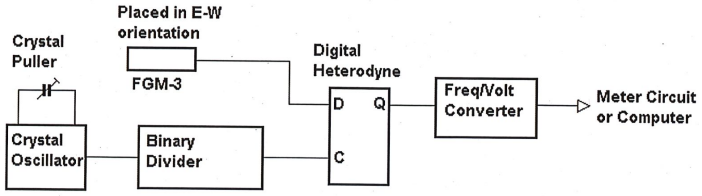
The technique is equally applicable to ferrous object detection or counting of smaller objects passing on a conveyor belt, the sensitivity and range being adjusted to suit the individual system. In this context it is useful to remember that the field produced by a given magnetic moment falls off as the inverse *cube* of the distance, not the inverse square.

All of the above hardware approaches can equally well, if not better, be simulated by software in a computer or microcontroller, often resulting in minimal hardware to achieve sophisticated results.

4

Earth Field Magnetometer

A block diagram of a modest earth field magnetometer is shown below using the type of circuitry described above.



Earth Field Magnetometer Configuration

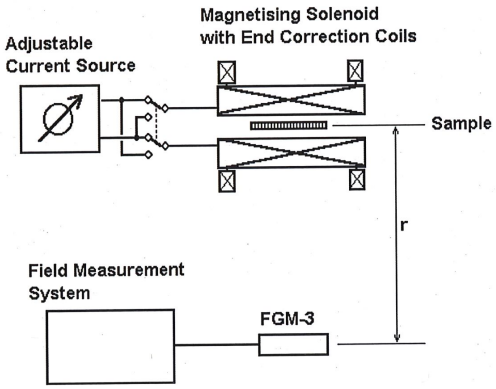
The sensor should be located in an east-west orientation and its mean frequency is measured. A crystal oscillator and binary divider is then selected to produce a frequency around 500 Hz below the sensor frequency. The sensor signal and the divided clock are fed to a digital heterodyne circuit as described earlier. The sensitivity of the FGM-3 is such that this arrangement will give a swing of about 0 to 1000 Hz for a variation in field of around ±500 gamma. (1 gamma $=10^{-5}$ oersted) This gives enough headroom for most magnetic storms likely to be observed. The exact range can, if required be calibrated as outlined earlier.

The output of the digital heterodyne can be taken to a voltage-to-frequency converter for chart recorder use or to a computer to store or plot the data in whatever form is appropriate. If the sensor is calibrated the results can easily be converted to angular or azimuth variations by dividing by the local horizontal component, (which can be measured by a north-south oriented calibrated sensor.) This gives the variation in radians, readily converted to the more suitable minutes of arc.

A magnetometer of this kind needs to be installed in a location far removed from potential sources of magnetic field interference, such as mains transformers and motor vehicles. Fortunately the inverse cube law mentioned previously helps considerably with this aspect. One exception to this rule, however can be exploited in the initial commissioning of the equipment. If difficulty is experienced in finding an appropriate combination of crystal and binary divider to nearly match the sensor output, in the magnetically quiet location needed, the strategic placement of a small ceramic magnet, at a suitable range, can be used to "pull" the sensor frequency instead.

Materials Magnetometer

A setup with this kind of sensitivity is equally capable of being used as the measurement tool in a classical Gauss-type materials magnetometer. The usual configuration of this instrument is illustrated below.



Materials Magnetometer Configuration

This consists of a controllable magnetising arrangement in the form of an air-cored solenoid as shown in the case of low susceptibility specimens or an electromagnet type yoke to reduce demagnetising effects with high susceptibility materials, in conjunction with a field measuring device. The demagnetising field measured by this device is related to the magnetic induction in the specimen.

Taking the magnetising coil through positive and negative cycles large enough to reach saturation in the specimen produces a measured field which displays the hysteresis loop characteristics of the sample specimen. To avoid shearing the hysteresis loop it is necessary to use a specimen with a small demagnetising coefficient or one with a known demagnetisation coefficient which can be corrected for, such as a cylinder. For straightforward quantitative work the distance r should be large compared to the magnetic length of the specimen if induction is to be measured, requiring high sensitivity in the detector. Comparative work and coercive force measurement can be carried out without satisfying this criterion.

Calibration is often carried out using a standard comparison sample of known demagnetisation coefficient and magnetic properties. The arrangement shown can also deal with the process of anhysteretic magnetisation if an alternating current source is superimposed on the direct current supply. The more common ac-driven type of B-H loop tester has difficulty with this since it cannot measure static fields.

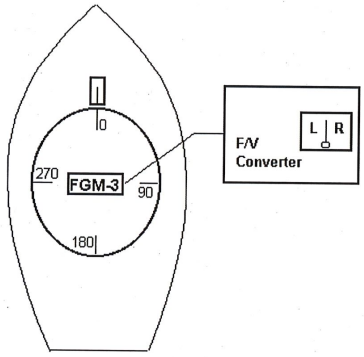
Similar arrangements are used by naval establishments under the name of fixed or portable ranges to determine the efficiency of the various degaussing equipments used to maintain the "magnetic hygiene" of vessels and items taken on board. The other side of this coin is that even more sensitive systems attempt to locate such vessels by detecting the magnetic anomaly caused by their presence.

The latter type of detector, however, needs to be insensitive to its orientation in order to avoid avoid the effects of the earth's field when it is in motion. Provided they can be made orientation insensitive, such detectors can be used as remote wreck-finders by divers

Orientation Measurement Devices

The most well known of these is the common compass, which is used to indicate the direction of the local horizontal component of the earth's field and from this to deduce the heading of the vehicle or vessel in which it is installed.

The simplest is a single axis device known as a steer-on-heading compass or "poor man's autopilot". It consists of a single magnetic sensor mounted on a rotatable disk, marked in degrees around the periphery and fitted with a stationary indicating pointer, as drawn below.



Steer-on-Heading Compass

The output of the sensor is connected to a frequency to voltage converter circuit feeding a centre zero meter display as described previously. In the diagram shown, if the boat veers to the left the meter needle swings to the right, indicating the need to steer in that direction to correct the course. If the rotating disk is turned to a new heading the needle will show the shortest direction to steer in until the new heading is reached when it will return to the centre position. The size of the deflection gives an indication of the amount of correction needed at any time. This type of steering system is said to be easier on the helmsman than having to remember and follow a degree bearing.

The implications of tilt in the sensor will be discussed in detail later but, for this simple system to be practical, the sensor must be gimballed and weighted so as to keep its axis level at all times. Since it is only a single axis device it only needs a single gimbal, provided that the gimbal rotates with the heading disk.

The next level of complexity is a two axis compass and for this it is best to replace the frequency to voltage converter with a microcontroller of some sort as a number of more complex operations need to be carried out especially if a readout display is wanted. Many varieties exist all capable of dealing with the requirements of a compass, but because the sensors have their own analogue-to-digital feature, microcontrollers which have frequency or period determining features built in are the obvious choice in this instance.

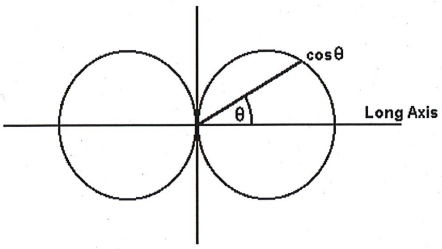
Microcontroller selection has been dealt with better elsewhere and these notes will restrict the subject matter to the compass design principles using the FGM type sensor.

Angular Sensitivity

It is useful to look first at the angular response of an individual sensor. Because of its structure it "sees" the full magnitude of a field which is aligned along its long axis. For any field at right angles to this axis it gives zero output in the sense that its period corresponds to that of a zero field condition. For a field aligned at an angle between these two extremes the response is proportional to the projection of the field on to the long axis of the sensor, therefore to the cosine of the angle between field and sensor.

7

This gives rise to the classic figure-of-eight polar diagram, consisting of two contacting circles or, in the three dimensional case, two contacting spheres.



Polar Response of Sensor

If sensors are aligned along the axes of any two or three axis coordinate system the sensor outputs represent the direction cosines of the field vector with respect to that coordinate system. For convenience the chosen system is usually cartesian but this is not a requirement.

As described under the calibration and linearising techniques it is convenient to normalise the sensor readings by dividing through by the zero-field period. In orientation type devices it is also convenient to then subtract one from these normalised values to yield equal positive and negative ranges about zero. These adjusted values are then proportional, but not yet equal, to the direction cosines of the field vector.

The reason is that no two sensors are exactly alike in absolute sensitivity and must now be calibrated so as to achieve a standard sensitivity. This can be done by the calibration coil method described earlier, after which proportionality constants can be assigned as multipliers to equalise the sensitivities. Alternatively it can be done by aligning the individual sensors in turn along the local earth field vector in the two possible directions, 180° apart and determining the corresponding maxima and minima for each sensor. Proportionality constants are again assigned to equalise the sensitivities.

Two Axis Orientation Sensing

The two axis compass uses twin sensors superimposed at right angles to one another in the same location and both constrained to lie in the horizontal plane. The sensitivity equalising process in this case can be semi automated by rotating the sensors through a full 360° and allowing the software to determine the maxima and minima for both axes.

If the two now standardised values are regarded as the x and y components of the local field vector, **h**, having a modulus equal to $\sqrt{(x^2+y^2)}$, the final normalisation can be realised by dividing each component by this modulus. This gives the true direction cosines of the field vector which together define the unit vector **i**, having the same direction as the field vector, **h**.

This process eliminates the effect of any variation of the absolute magnitude of the measured field, since the sum of the squares of the direction cosines always equals one. Earth field variations are insignificant in this context, but supply or ambient temperature changes are neutralised provided all sensors are equally effected.

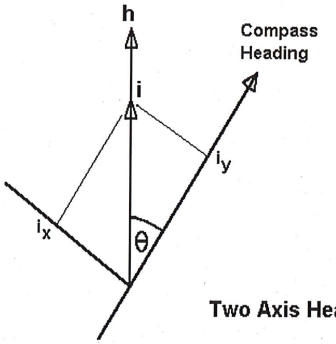
The direction cosines can be readily converted to a more customary representation such as angular heading as follows.

8

Assume that the compass heading indication is aligned with the y-axis and label the components of the unit vector, i_X and i_Y. Then it can be seen from the following diagram that if θ is the conventional heading angle,

$$\tan\theta = -i_X/i_Y \quad \text{and} \quad \theta = \tan^{-1}(-i_X/i_Y)$$

and the compass heading is simply the arctangent of the ratio of the x and y components of the unit vector in the earth's field direction.



Two Axis Heading System

For a three dimensional coordinate axis system with the z-axis at right angles to the other two, there is no conflict with anything that has been said so far, provided that the z-axis remains vertical. In fact this becomes the necessary condition for the successful operation of this type of compass, which needs to be gimballed in two directions and appropriately weighted.

It will be evident that some attention to signs and the possible divisions by zero will be required in considering the full circle of 360°. While this arctangent solution may be possible for a computer with trigonometric functions in a high level language, it is not really appropriate for a lower level of implementation such as a microcontroller, though the underlying principle remains the same in alternative approaches.

The full circle in which the heading vector lies may be segmented into eight 45° octants and the octant occupied by the field vector can be identified by simple non trigonometric tests, easily applied in software.

The rules which do this involve the signs of the i_X and i_Y components and the comparative magnitudes of these components taken as an ordered set. For example if $i_X<0$ and $i_Y>0$ the heading must lie in the first quadrant. If, in addition, $|i_X| < |i_Y|$ it must lie in the first octant between 0° and 45° as in the previous diagram. Other combinations uniquely identify the remaining octants in the fashion illustrated in the chart below.

sign of i_X	sign of i_Y	$\|ix\|>\|i_y\|$ or $\|i_x\|<\|i_y\|$	octant no
negative	positive	less	1
negative	positive	greater	2
negative	negative	greater	3

negative	negative	less	4
positive	negative	less	5
positive	negative	greater	6
positive	positive	greater	7
positive	positive	less	8

The implementation of these rules on their own provide an eight point compass with a ±22.5° accuracy, which while not very precise may be adequate for some undemanding applications. There are other benefits in more sophisticated versions.

9

The first advantage of this technique is the Gray code like way in which the octant rules work. At each octant boundary only one of the rule parameters changes. For example at 45° no sign changes occur but the inequality between i_x and i_y changes direction. At 90° no inequality changes occur and the sign of only i_y changes. This property prevents large scale jitter and confusion which might otherwise occur at the octant transitions if the changes were not totally synchronous.

A second advantage is that in each of the octants, a linear function of either i_x or i_y can be identified which is virtually equal to the desired heading angle, to within a small error. In the 0° to 45° range if ki_x is interpreted as a radian angle it is in fact very little different from the appropriate arctangent for that octant. If k=1.08 the error in doing this is nowhere greater than about 1.25°. If k is ignored and the unit vector x component alone is interpreted as radians the error is never worse than 4.5°, permitting the implementation of a 5° precision compass very easily. Note that for this purpose the modulus of the x component is used, eliminating the need to consider signs. Note also that $|i_x|$ is less than $|i_y|$.

In the next octant, between 45° and 90°, ki_y interpreted as an angle and subtracted from 90° is very close to the correct heading. This pattern repeats around the full circle and leads to the following rule.

Whichever of the direction cosines is the smaller is interpreted as an angle and in odd octants is added to the nearest quadrant boundary, but in even octants is subtracted from the nearest quadrant boundary to obtain the heading. (If as is likely in a software implementation the octants are numbered 0 to 7 rather than 1 to 8, the odd and even should be reversed in the previous statement of the rule.) In conjunction with using k=1.08 this rule will provide almost ±1° precision in a software implementation requiring no trigonometric functions.

Alternatively, since the error is small, a very short lookup table of adjustments to be added to the heading obtained with k=1.0 will improve the precision to a level of around ±0.5°.

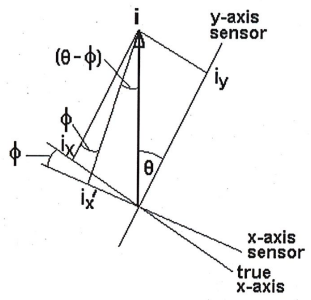
measured angle	added correction
0° - 20°	0°
20° - 29°	0.9°
29° - 33°	1.8°
33° - 37°	2.7°
37° - 39°	3.7°
39° - 41°	4.3°

It should not be assumed that this technique will enable a compass of this accuracy, only that the contribution to the total error budget from this source will be minimised to the extent indicated. Other sources may contribute larger errors in a final design if they are not suitably addressed.

One important potential error is lack of orthogonality in the axes of the two sensors. This can cause a smoothly varying error around the whole compass circle which can be much larger than those discussed above. Fortunately there is a relatively simple correction technique for this as can be seen from the following analysis.

In the diagram below **i** is the unit vector in the field direction, θ is the heading angle and ϕ is the small angular error by which the x-axis sensor departs from the correct right angled position. Also i_x is the true x component of the unit vector, i_y is the true y component and i_x' is the apparent (measured) x component of the sensor in error.

10



Orthogonality Error Diagram

It can be seen from the geometry of the figure that

$$i_y = i \cos\theta$$
$$i_x = i \sin\theta$$
$$i_x' = i \sin(\theta - \phi)$$

Expanding the last relation

$$i_x' = i \sin\theta \cos\phi - i \cos\theta \sin\phi$$

Since ϕ is small $\cos\phi$ may be taken as one and $\sin\phi$ as just equal to ϕ giving

$$i_x' = i \sin\theta - \phi i \cos\theta = i_x - \phi i_y$$

Hence

$$i_x = i_x' + \phi i_y$$

It can be seen from this that the desired x component of the unit vector can be obtained from the apparent measured component, for all angles, by adding a small fixed portion of the y component. The proportion to be added is equal to the orthogonality error in radians.

The value of ϕ can be found, for a standardised and normalised sensor set by rotating the configuration in the earth's field and measuring the angle between the zero-field positions of each sensor. Alternatively the algorithm can be added retrospectively to an otherwise completed compass by checking the error during a full 360° rotation. The value of ϕ can be taken to be the average of the errors at 90° and 270° shown by the digital display. Such a determination needs only to be made once. If the microcontroller has no convenient way of memorising the correction it could alternatively be read from a trimpot value on power-up using RC timing or some other relatively crude analogue input method. The orthogonality then becomes one of the possible adjustments available to the user during the compass "boxing" exercise.

Probably the largest of the final observed errors will arise from failure to constrain the axes of the sensors to the horizontal plane. The errors depend on the direction of tilt and the heading, and on some headings small angular tilts will multiply up to much larger heading errors. For example on a north heading a 1° north-south tilt will produce no error, but a 1° east-west tilt will give rise to almost 2.5° of heading error. There is no simple cure for this other than effective double gimballing, suitably weighted, though short term averaging of multiple readings can improve the stability of the displayed output.

Another more complex alternative is to use a gravity sensor to determine the direction of the gravity vector and use trigonometric calculation to correct for the effects of tilt.

A final aspect of overall accuracy concerns the required precision of sensor readings. Interestingly, this is surprisingly lower than might be thought. Using the type of algorithm described earlier, a full 360° of 1° precision requires only that the measured components be slotted into one of forty-five almost evenly spaced bands. A relatively low six-bit binary measure will cover this. For a 5° precision a miserly four bits is adequate.

In conclusion, for those who may design, build and use a compass, in anger, the illusion of precision created by a 360° digital display may hide a lack of precision which is real. The cautious navigator rarely places total faith in compass accuracy and never trusts it as his sole instrument of navigation.

Three Sensor Systems

The use of three orthogonal sensors permits a three dimensional determination of both the magnitude and direction of the local field vector. This determination, however, is only made with respect to the axis system of the sensor configuration and not in any absolute space. Nevertheless it can provide the basis of many interesting applications other than the compass.

The compass is not the only device which requires absolute referencing. The extension to three dimensions permits in principle, the exploitation of the earth's field in "virtual reality" simulations, with the possible advantage of a "free roving" capability.

The potential to free rove in a large space is a consequence of the fact that the field behaves as a fixed orientation vector everywhere in the space. It can be converted to the forward looking vector of a virtual reality helmet, provided that it can be referenced to some absolute space.

The sensor configuration alone is not adequate for the following reason. For each angle the field can take with reference to the sensor axis system it is possible to rotate the axis system a full 360° around the field vector without any change in the sensor outputs. This ambiguity must be resolved to obtain the desired absolute reference and requires one more fixed orientation vector. The obvious one is the earth's gravity vector which will always provide a local vertical.

While a compass design can make use of a slow response device such as a mercury pool on resistive quadrants or a dielectric bubble on capacitive quandrants, these are useless for virtual reality applications. They usually do not have the angular range and the certainly do not have the speed required to follow rapid head movements.

The minimum requirement would a be speed compatible with a flicker free video image refreshing system, say 70 Hz, though some systems specify a response rate of 250 Hz. To satisfy this kind of requirement calls for something like an accelerometer configuration with a flat bandwidth of this order, which also extends down to DC. Such devices have been recently developed, spurred on by the automobile airbag market, but low-g versions are still very expensive in small quantities. They also have relatively poor signal-to-noise ratios at wide bandwidths. This is not too severe a problem, however, since like the compass algorithms already described, high angular accuracy can be obtained with low binary spans.

This is a fairly complex subject and details are deferred to a later application note.

Pseudo Three Axis Systems

There is a class of systems which use a three axis sensor sytem, but eliminate the need for the gravity vector by an additional constraint on one axis. They have the superficial appearance of three dimensional systems but do not exploit all the possible degrees of freedom. The searchlight is a classic example. It rotates in azimuth around 360° and could rotate in elevation through 180°, but does not have any mechanism for rolling around the remaining axis, since it would be entirely pointless.

If only the human head was satisfied by the same mechanism! Virtual reality would be much easier to implement.

The reason that this works is that as soon as the roll axis is constrained to remain horizontal, the rotational ambiguity around the field vector, mentioned previously, disappears. The trigonometry of the unit vector components is soluble and yields not only the azimuth angles, like a compass, but also the elevation angles.

Gun platforms fall into this category, as do steerable satellite type aerials, some robot mechanisms and any device which needs to point to a direction in space from a horizontal platform. Complex devices of this nature are probably well served by the expensive mechanisms they already employ, but there may be many simpler applications which could benefit from a low cost magnetic sensor configuration and a microchip solution, previously not economic.

One interesting idea may be exploitable by the economy end of the flying sport. Aircraft magnetic compasses are notoriously impossible objects, since even the addition of a gravity vector sensor solves nothing when it indiscriminately combines gravity with the accelerations of manoeuvering. In level flight a gimballed fluxgate compass works well but is useless in turns. Nevertheless it remains a reasonable tool to a power pilot in transit. Since a glider pilot spends a great deal of time in spiral turns, chasing thermals, it is not very appropriate most of the time.

If one axis could be reasonably constrained most of the time, a usable compromise might be achievable. Since the full horizontal rotation of 360° is required and roll angle can be large, the only restriction possible is in the pitch axis. Aircraft do not generally spend very long periods in pitching manoeuvers except during aerobatic activity. They may, however, alter pitch modestly during climbing, descending or turning. During any steady state version of these activities acceleration or deceleration along the line of the fuselage is small or nil.

If a three dimensional sensor configuration were gimballed transversely and suitably weighted, it could perhaps maintain the pitch axis of the sensor set sufficiently horizontal to allow the strategy under discussion to generate a heading and additionally a bank angle of acceptable precision.

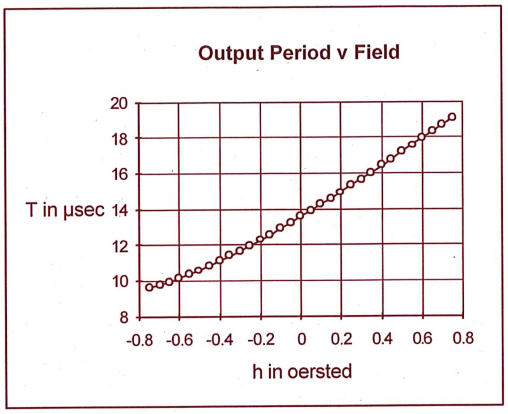
Whatever the precision, it would represent a vast improvement on the conventional fully gimballed compass and add half of an artificial horizon into the bargain, at lower weight and cost than any gyroscopic equivalent.

Three and Two Dimensional Ferrous Detectors

It is possible to elaborate the design of fixed single sensor vehicle detectors described earlier, with advantage, by using a two sensor version. Even when restricted to the horizontal plane, an orthogonal sensor set can provide more information, in the sense that it can provide both angular and magnitude signals for the anomaly caused by the vehicle passage.

An object with a magnetic moment possesses an external pattern of lines of force similar to that of a permanent magnet. This line of force pattern combines additively with the earth's field lines of force which consist locally of straight parallel lines. If the disturbing magnetic moment passes very close to the sensors it produces not only a variation in field magnitude but also large swings in the angular orientation of the detected field. If the passage is more remote from the sensors, not only is the magnitude of the signal reduced, but so also is the total angular swing.

A typical response for the miniature types of sensor, the FGM-1 and FGM-2 is given below.



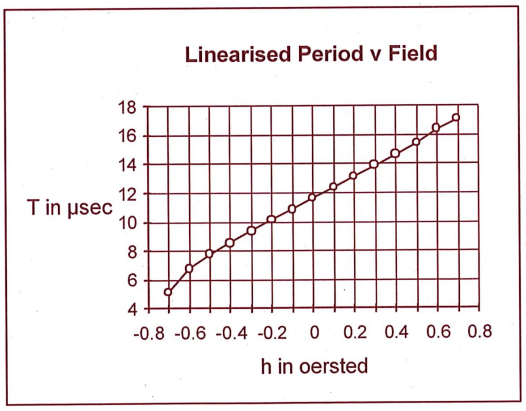
Between ±0.5 oersted (±50 µtesla) the non-linearity is about 4.8%

This non-linearity varies somewhat between individual sensors, but may normally be expected to be in the region of 5%.

A simple strategy will improve this considerably.

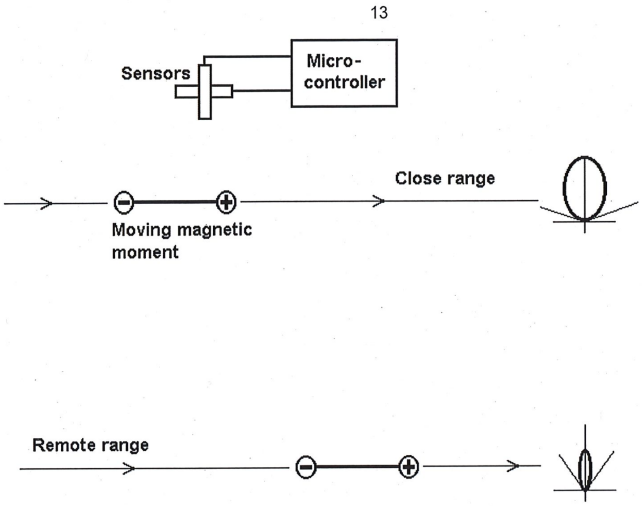
It was stated earlier that the field strength was inversely proportional to the frequency. In practice it will be found that the field strength is more closely inversely proportional to the frequency plus a small constant. If the frequency is measured and a fixed number of kilohertz is added before it is divided into one, to obtain the period, the response curve of period against field will be seen to become much more linear.

The chart below shows the effect of adding a fixed value of 15 KHz to the incoming frequency of an FGM-3 sensor before inverting to obtain the period. (Plot shows T= 1/(f+c), c=15 KHz)



3

While the time variation of these parameters gives some indication of the speed of the passage, if the magnitude of the signal is plotted against the angle in a polar diagram, what results is a time invariant "signature" of the object. In some sense this signature contains information about the range, since for a close passage it will have a large angle polar diagram and for a remote passage a small angle diagram. This range is not absolute as it will also depend on the equivalent magnetic length of the magnetic moment being observed, which is roughly correlated with the size of the vehicle most of the time. The fall off in field strength is proportional to the inverse cube of the ratio of the range to the magnetic length, so the field from large objects falls off more slowly than that from small ones.



Two Axis Ferrous Detector

The actual magnitude and angle variations will be quite small but can be increased to usable size by the digital heterodyne method or, in this case its software equivalent. The polar diagrams shown are oversimplified guesswork and not based on any tests.

However it seems that this is an area worthy of a little more serious research on practical real life situations, since it may resolve the problems of lane separation and vehicle classification in multiple vehicle studies.

Orientation Sensitivity Elimination

The ferrous detection systems discussed so far have been static ones and the fixed large signals produced by the earth's field can be relatively easily eliminated from the desired indications. In situations where the sensor configurations will inevitably be subject to unpredictable movement, the high orientation sensitivity becomes a serious disadvantage in the search for very small signals.

However if we consider a perfectly standardised and normalised, perfectly linearised and perfectly orthogonal sensor set, the problem is easy to deal with since the sum of the squares of the three outputs must always be equal to the field vector modulus squared, a scalar quantity without any orientation. The success of any real implementation will clearly be only a function of how close to perfection the above requirements come.

The mathematics are simple and readily implemented on computer or microcontroller.

The basic sensitivity of the sensors is adequate, matching of the calibrations is more constructive than absolute accuracy, orthogonality correction can be carried out to a fairly high degree, but non-linearity may give the most troublesome source of error.

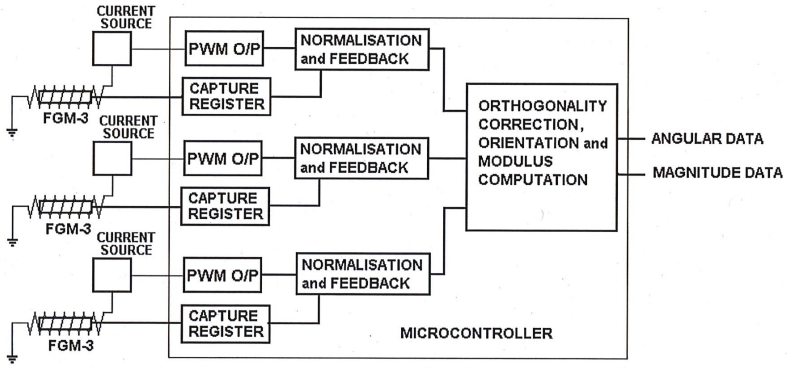
There is a technique which can be very helpful in these circumstances and this is the use of some sort of negative feedback, well known for its ability to improve linearity and stability.

The method consists of overwinding the sensor with a solenoidal coil in which a controlled field can be produced and automatically adjusting this field to cancel out to zero, the local field which the sensor would otherwise experience. The solenoid current giving rise to this cancelling field must be proportional to the local field being cancelled. Since the sensor only ever sees a zero field, its own non-linearity is no longer of consequence and the cancelling current is a direct and linear measure of the local field magnitude.

14

This approach obviously calls for a digital-to-analogue converter to control the current in the cancellation coil, but with a microcontroller, this could be a pulse width modulated, single-bit, output and low pass filter arrangement, as used so successfully in many current low-cost digital audio devices. The software complexity increases but the hardware cost is still held low, probably calling only for a linear current generator of modest current capability.

In any case, total 360° orientation de-sensitising is not always needed and reductions in the angular variation achieved by other means will often considerably improve performance, as for instance in the case of a detector carried in a normally level vehicle or a neutral buoyancy weighted float, trailed just submerged. An error may exist in the output but it remains passably constant.



THREE DIMENSIONAL SENSOR SYSTEM

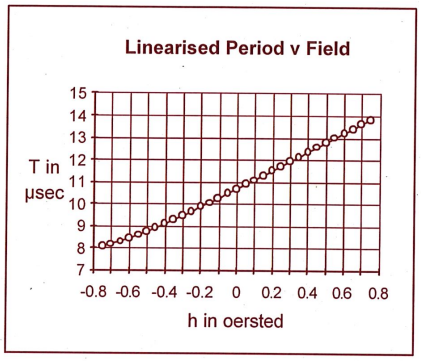
This type of system could find uses as a detector of seabed wrecks in modest depths or as a search tool in archaeological studies. Constructed with sufficient care, it provides a low cost and compact alternative to nuclear magnetic resonance devices in some applications.

15

Apart from the droop at -0.7 oersted the linearity has improved considerably. In fact applying the same definition as before the non-linearity between ±0.5 oersted has been reduced from 5.5% to 0.7%

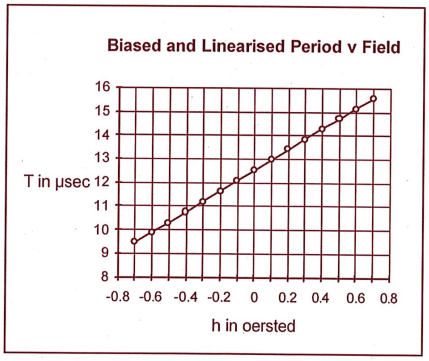
In individual cases varying the 15 KHz figure may produce an even better non-linearity, but in general this value will produce a significant improvement with any production sensor, where individual tailoring is undesirable.

The same technique can be used with the miniature sensors FGM-1 and FGM-2. The chart below shows the effect of adding a constant frequency of 18 KHz to the incoming frequency before obtaining the period by inversion. (Plot shows T=1/(f+c), c=18 KHz)



For the miniature sensors a further improvement in linearity and also maximum range can be obtained by making use of the built in overwound coil intended for feedback systems. This can also be used as a simple biasing coil to alter the zero field period of a sensor. Inspection of the graph above suggests that moving the h scale to the right might improve the linearity.

This is, in fact, the case and the effect of injecting a current of about minus 1 mA (a 4K7 resistor to a -5 volt supply) into the feedback winding pin, can be seen in the graph below.



4

The linear range and linearity can be seen to have improved considerably and applying the same criterion as previously, the non-linearity has been reduced from 4.8% to 0.2% across the increased range of ±0.7 oersted.

If the application is set up in such a way as to measure the period directly, rather than the frequency, followed by an inversion, the linearising technique can still be used.

The method is to use T/(T+cT) as the quantity which is proportional to field strength, where T is in microseconds and c is in kilohertz as before.

For applications requiring greater linearity, the more complex overwound external coil feedback system can provide this together with even more enhanced field strength range if needed.

Supply Voltage Variation

The period (and frequency) of the FGM-series of devices varies with supply voltage, having a coefficient of about 3.5% per volt at the nominal 5 volt supply level. For precise applications good supply regulation is required, but since the transducer's current requirement is low, this is fairly easy to achieve using, for example, single or double regulation with devices from the LM78LXX series.

General Application Notes

Use with Computers and Microcontrollers

The large pulse output gives considerable noise immunity permitting the use of transducers sited at long distances from the main system.

Interfacing is simple in that it requires only one bit of a digital input port per channel of measurement, the technique being to count input pulses for a fixed period to determine the frequency of the incoming signal, from which the field can be calculated. Alternatively, where a faster response is required, the time between successive like edges permits the direct determination of period, from which again the field can be calculated.

With microcontrollers this usually presents no problem, but with systems using many interrupts or extensive multi-tasking it may be necessary to buffer the input signals to deal with the high data rate. However this usually means no more than the addition of a single triple-counter I/O chip even for three-dimensional orientation systems.

For applications such as earth field magnetometry, where readings may only be required at relatively long intervals simple binary division with a single chip 12 or 14 stage divider will reduce the input period to a level where data rate ceases to be a problem to the computer. Alternatively, in such applications where the field variation is extremely small, digital heterodyning with a stable oscillator will also reduce the period but simultaneously maintain the high sensitivity, (in hertz/oersted) to field variations.

For applications which need absolute field magnitude without any orientation sensitivity, it is necessary to use three orthogonal sensors and exploit the fact that the sum of the squares of the three signals is constant regardless of orientation. Provided that the zero offsets, channel sensitivities and linearisation are appropriate to the required absolute sensitivity, this will permit free movement of the sensor head while measuring small changes in absolute field. If the sensor is in constant angular motion, advantage can be taken of this to provide some level of auto-calibration of zero offset and channel sensitivity. (See Application Note - Auto Calibration Algorithm .)

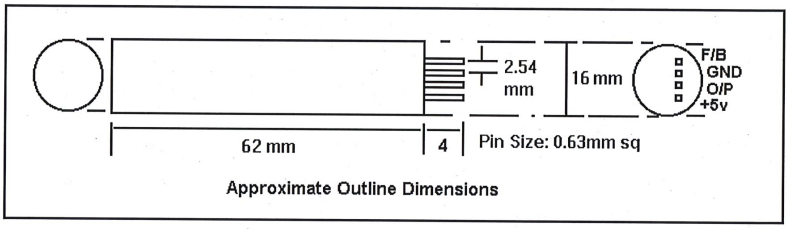
Where the sensor can be permanently fixed, only one sensor is necessary, the zero offset being adjusted to suit the local ambient field strength. This technique is appropriate to fixed ferrous metal detection systems such as conveyor belt counters, vehicle and ship passage detectors and materials magnetometry. A limit to the range of such systems results from the fact that the earth's field itself fluctuates at a low level continuously. The effective range will be a function of the size or likely magnetic moment of the objects being detected, ships generally giving a larger range than vehicles or hand guns. Appropriate filtering of the input frequency variations will enhance range.

Where extremely high sensitivity is required it may be possible to use two sensors in a gradiometer configuration to cancel out the micro-fluctuations of the earth's field. However, this will not always increase range, since the gradiometer sensitivity falls off faster with range than the simple field sensor.

In this context, it should be remembered that the field produced at range by a magnetic moment falls off as the inverse **cube** of the range, so the gradiometer configuration will fall off as the inverse **fourth** power. However such systems may be useful as short range high sensitivity detectors and materials measurement systems. An example might be extremely small magnetic moment inert particles introduced into fluid flow systems for movement detection, such as chemical processing plants or animal internal fluid flow systems in medical research applications.

PHYSICAL CHARACTERISTICS

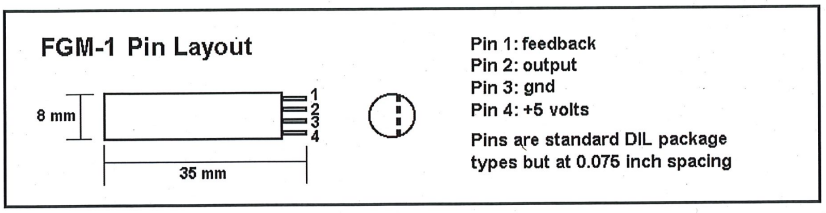
Sensor Outline - FGM-3

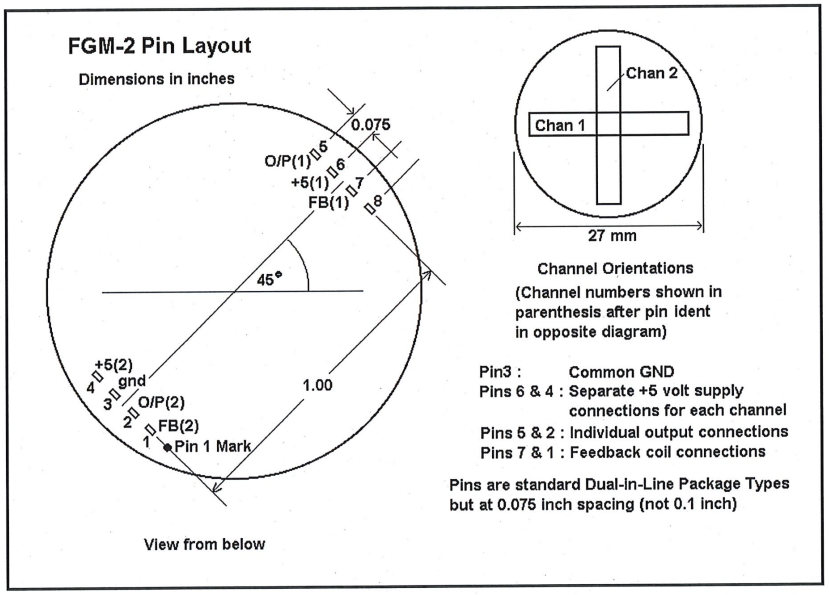


Approximate Outline Dimensions

The sensor has been made with a cylindrical form in order to facilitate the overwinding of external feedback coils. Alternatively, it can be readily inserted into a separately fabricated coil on a tubular core.

As a simple guide, for example, a single layer of 0.2mm wire (0.25mm overall say) wound over 60mm of the sensor length will give the equivalent of 4000 turns/metre. Since 1 oersted is approximately 80 ampere-turns/metre, such a winding will produce a field of around 50 oersted/ampere. Thus with up to 100mA flowing it is possible to offset fields of ±5 oersted, increasing the range of the sensor by x10. Negative feedback also brings all the usual benefits of improved linearity and stability, of course.





CONVERSION TABLES
Magnetic Flux Density

	gauss	tesla	gamma
1 gauss	1	10^{-4}	10^{5}
1 tesla	10^{4}	1	10^{9}
1 gamma	10^{-5}	10^{-9}	1

Magnetic Field Strength

	amp/metre	oersted
1 amp/metre	1	0.01257
1 oersted	79.58	1

NOTE: Technically, the sensor measures flux density, in gauss, but since in vacuum (and virtually in air) the units of flux density are the same magnitude as those of field strength and since the sensor can only really be used in air, oersted have been used in the text and diagrams as equivalent to gauss.

7

Speake & Co Llanfapley
6, Firs Road * Llanfapley * Abergavenny * Monmouthshire * NP7 8SL * U.K.
Tel/Fax 01600 780150 *
email * billspeake@btconnect.com

FGM-series
Magnetic Field Sensors

Application Notes

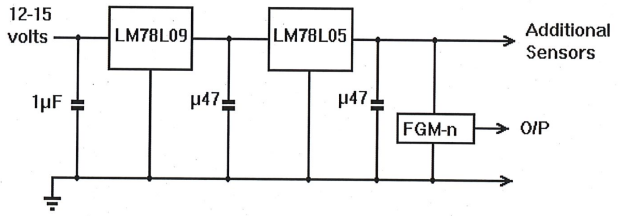
General Application Techniques

Decoupling and Power Supply Regulation

If long leads are used with the sensor it is advisable to provide some local decoupling close to the sensor itself where possible. A 10μF electrolytic capacitor is suitable for this purpose.

Since the sensor itself has a sensitivity of a few percent to power supply variations, it is necessary to provide it with some power regulation in most cases. For many applications, such as orientation devices, a single fixed voltage regulator of the LM78L05 (or equivalent) type is adequate. However for applications such as earth field magnetometry or where extremely small field variations are being studied, supply voltage variation needs to be reduced to a level which permits the temperature coefficient of the device to be the limiting performance factor.

Double regulation from 12-15 volts, first down to 9 volts and then to 5 volts, using the LM78L09 and LM78L05 provides a low cost solution, a typical arrangement being shown below.



Double Regulation Supply

Sensor Calibration

For many applications, such as simple field detection or orientation measurement systems, calibration of the sensors is not necessary.

For those applications which do need to measure field strength, a reasonably accurate calibration can be made using simple equipment. A single layer solenoid can easily be made by close-winding enamelled wire on to a tube having an internal diameter large enough for the sensor to be inserted in it. The field inside a long solenoid is given simply by the product of the current flowing in it and the number of turns per metre with which it is wound. Both these items can be measured reasonably accurately, one with a ruler, the other with an ameter.

For most purposes, a winding at least twice as long as the sensor will give a good calibration, consistent with the likely turns/metre measurement accuracy using a ruler, provided its diameter is no greater than necessary.

Single axis sensors are the easiest in this respect because they have a small diameter themselves and can be inserted into a small diameter tube. The following tables should be helpful in the the design of calibration coils.

First, the field at the centre of a cylindrical coil of the type suggested is given by:

H=geometry-factor x number of turns per meter x current (H in amperes/metre)

where the geometry-factor is shown in the following table.

Length/diameter	Geometry-factor
5	0.9806
6	0.9864
7	0.9900
8	0.9923
9	0.9939
10	0.9950

This permits calibration of the coil centre-field. The correction is small, somewhere between 0.5 and 2 percent, but may be worthwhile in appropriate cases.

Away from the centre of the coil the field falls off towards either end and on the assumption that the coil is twice as long as the sensor, the following table gives a factor for this reduction, at either end of the sensor, for various coil geometries. It also shows, in the third column, the percentage by which the field differs from being uniform along the length of the sensor, assuming that the sensor is centrally placed.

Length/diameter	Reduction-factor	Uniformity
5	0.9788	±1.06%
6	0.9847	±0.77%
7	0.9884	±0.58%
8	0.9910	±0.45%
9	0.9938	±0.31%
10	0.9942	±0.29%

The geometry-factor from the first table should be reduced by this percentage to arrive at a mean calibration factor for the coil, for the most accurate results.

The calibration currents required are modest. Since the usable range of the sensor is around ±0.5 oersted or 40 amperes/metre a single layer winding of 0.5mm enamelled wire (which has an overall diameter of ~0.559mm) will only require 23 mA to reach maximum calibration field strength.

In carrying out such a calibration with a solenoid coil, the coil and sensor should be aligned at right angles to the direction of maximum local field as determined by the sensor alone. Where only relative field measurements are needed this can be done by simply aligning the coil and sensor in an east-west direction. If an accurate zero field calibration point is required the sensor will need to be placed in a zero-field location, such as the inside of a small mumetal container, aligned east-west.

Linearity Correction

In practice, the linearity is much better than the data specification suggests (over the recommended operating range) and few applications really require any correction at all. In a plot of field against period a small concave downwards curvature may be detected which is most simply straightened by adding a small proportion of the period squared to the assumed linear relationship. The proportion to add can be estimated by comparing the full scale negative and positive field periods with the zero field period, obtained from a calibration exercise as described earlier. Substituting these three values for T in the data sheet suggested equation, $H=c_0+c_1(T-T_0)+c_2(T-T_{min})^2$, gives three simultaneous equations from which c_0, c_1, and c_2 can be determined.

If the periods are normalised by dividing by the zero field period value, T_0=1 and the three equations are;

$-0.5=c_0+c_1(T_{min}-1)$ T_{min} measured at -0.5 oersted

$0=c_0+c_2(1-T_{min})^2$ $T_0=1$

$0.5=c_0+c_1(T_{max}-1)+c_2(T_{max}-T_{min})^2$ T_{max} measured at +0.5 oersted

Physically interpreting these coefficients, c_1 is just the usual slope of a linear relationship, c_2 is the small period squared correction previously discussed and c_0 is to offset the disturbance of the zero field period by the squared term and pull T_0 back to to its normalised value of one.

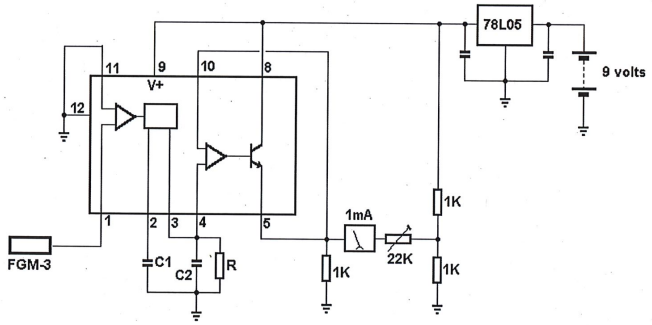
Field Measurement Methods

The simplest method of making field measurements is to use a frequency meter directly on the output of the sensor, set to period measuring mode. In most cases application designers will prefer to dedicate some specific hardware to carry out the conversions automatically and present the results in some acceptable form rather than use the eyeball/pencil and paper method, though this is quite adequate for calibration and familiarisation exercises.

Hardware configurations can vary from minimal, battery powered meter-display detectors through to complex, multiple sensor, computer controlled data collection systems. The following notes describe some useful techniques which may be incorporated into such designs without detailing any complete systems.

Meter or Chart Recorder Outputs

Low cost equipment can be made by using a semiconductor frequency-to-voltage converter such as the LM2907 or equivalent. Many variations of this type of integrated circuit are available in the component catalogues. A suggested circuit for a portable, direct reading instrument is shown below.



C1, C2 and R selected to give required range (see LM2917 data sheet)

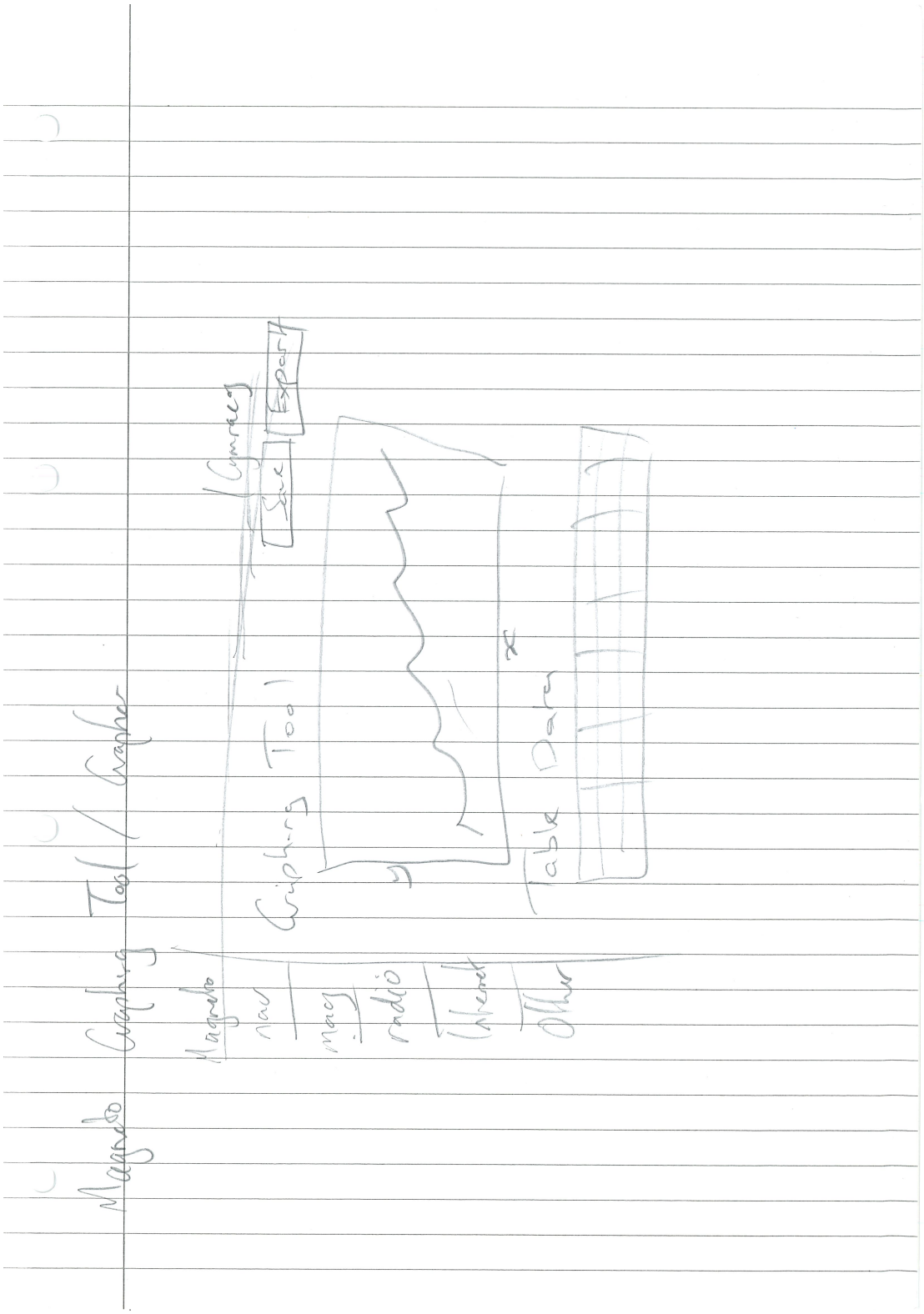
Simple Portable Meter Output Instrument

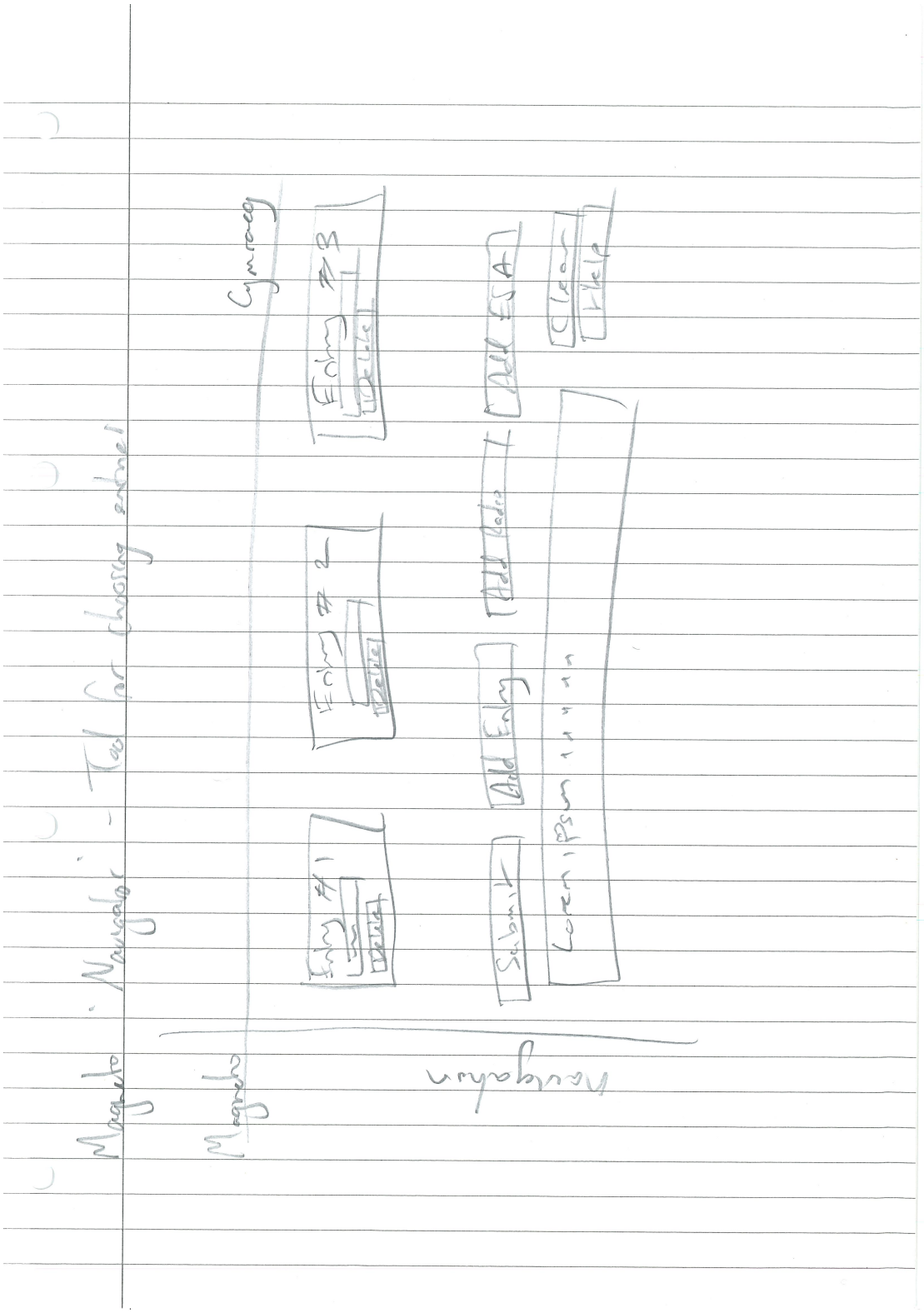
Since the field strength is inversely proportional to frequency the output is not linear but over the ±0.5 oersted range the non-linearity is modest and provides an acceptably spaced meter scale. At higher sensitivities over a more limited full scale range, the non linearity becomes increasingly neglegible and gives an almost evenly spread scale.

Where the converter device is unable to handle the highest frequency output, simple binary division in a prescaler chip is an appropriate method of bringing the sensor output into an acceptable range.

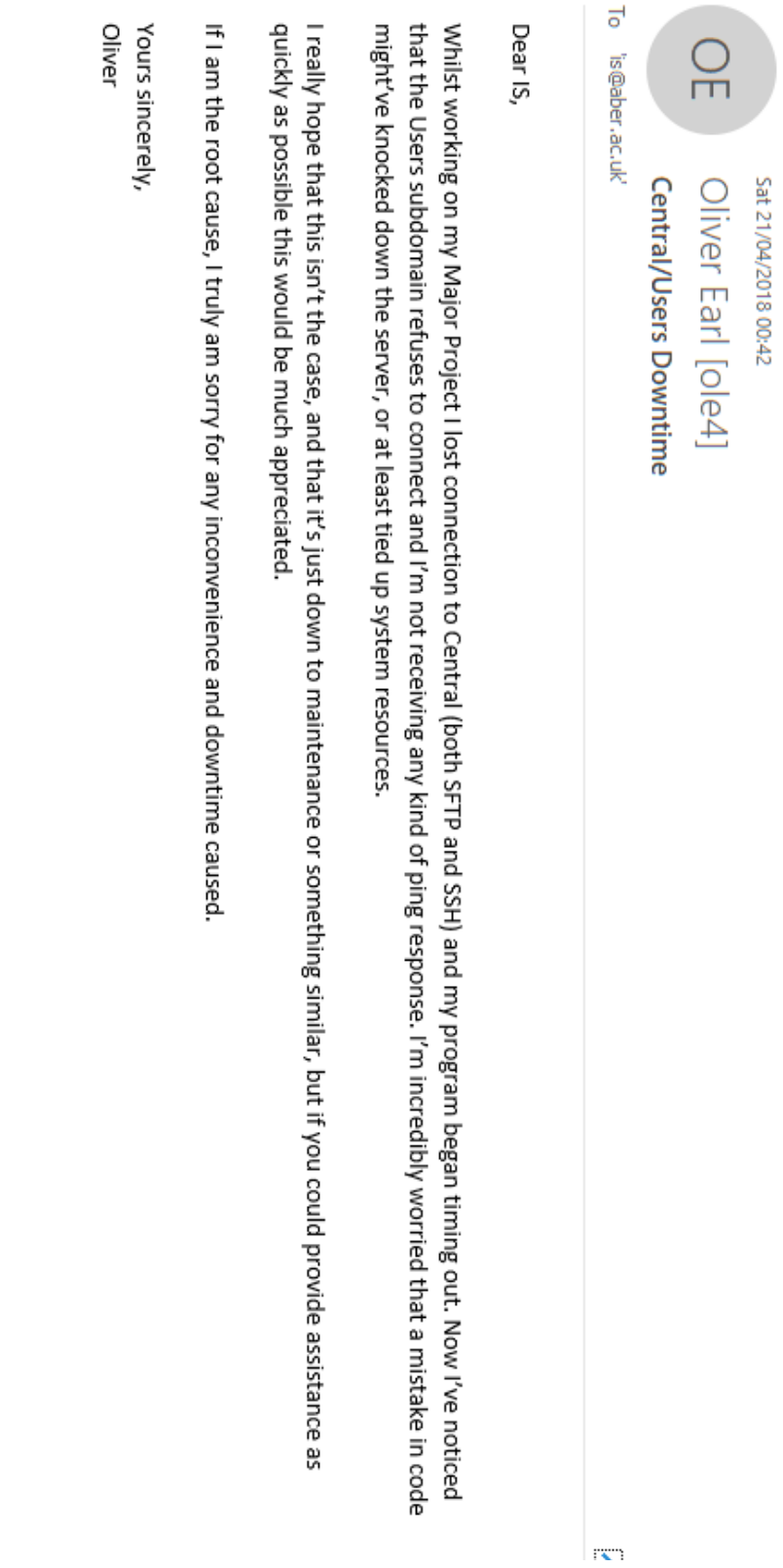
As an alternative to the diode pump type of converter a phase locked loop can also be configured to provide similar performance.

E. Hand-Drawn Conceptual Designs for Graphing Tool and Navigator Views





F. Apology email to Information Services following web server downtime caused by excessive data consumption



OE

Sat 21/04/2018 00:42

Oliver Earl [ole4]

Central/Users Downtime

To 'is@aber.ac.uk'

Dear IS,

Whilst working on my Major Project I lost connection to Central (both SFTP and SSH) and my program began timing out. Now I've noticed that the Users subdomain refuses to connect and I'm not receiving any kind of ping response. I'm incredibly worried that a mistake in code might've knocked down the server, or at least tied up system resources.

I really hope that this isn't the case, and that it's just down to maintenance or something similar, but if you could provide assistance as quickly as possible this would be much appreciated.

If I am the root cause, I truly am sorry for any inconvenience and downtime caused.

Yours sincerely,
Oliver

G. PHPUnit Test Results

Development Machine, Windows 10 x64 Education	
ole4\Magneto\Tests\Config	
Get config	Pass
Get instance	Pass
Get debug entry	Pass
Update config entry	Pass
ole4\Magneto\Tests\Connector	
Get instance	Fail
Reload database	Fail
ole4\Magneto\Tests\Magneto	
Sanitise int with alphanumeric characters	Pass
Sanitise int with HTML tags	Pass
Sanitise int with int	Pass
ole4\Magneto\Tests\Magnetometer	
Save Magnetometer Entry	Fail
Date to UNIX Timestamp	Pass
Convert UNIX to LabView	Pass
Convert LabView to UNIX	Pass
ole4\Magneto\Tests\Retriever	
Get instance	Pass
Set retrieval date	Pass
Retrieve data from magnetometer	Fail
ole4\Magneto\Tests\Locale	
Get language	Pass
Get locale	Pass
Set language	Pass
Set locale	Pass

H. Linting Results

#	File	Line	Problem
1	D:\Dropbox\Major Project\src\Config\Config.php	33	Avoid using static access to class '\ole4\Magneto\Magneto' in method 'getConfig'.
2	D:\Dropbox\Major Project\src\Config\Config.php	55	Avoid using static access to class '\ole4\Magneto\Magneto' in method 'saveConfigToDisk'.
3	D:\Dropbox\Major Project\src\Controllers\MagnetometerController.php	19	Avoid variables with short names like $db. Configured minimum length is 3.
4	D:\Dropbox\Major Project\src\Controllers\MagnetometerController.php	23	Avoid using static access to class '\ole4\Magneto\Database\Connector' in method '__construct'.
5	D:\Dropbox\Major Project\src\Controllers\MagnetometerController.php	26	Avoid variables with short names like $id. Configured minimum length is 3.
6	D:\Dropbox\Major Project\src\Controllers\MagnetometerController.php	28	Avoid using static access to class '\ole4\Magneto\Magneto' in method 'getById'.
7	D:\Dropbox\Major Project\src\Controllers\MagnetometerController.php	56	Avoid variables with short names like $id. Configured minimum length is 3.
8	D:\Dropbox\Major Project\src\Controllers\MagnetometerController.php	129	The method getAll has a boolean flag argument $limit1, which is a certain sign of a Single Responsibility Principle violation.
9	D:\Dropbox\Major Project\src\Database\Connector.php	12	Avoid variables with short names like $db. Configured minimum length is 3.
10	D:\Dropbox\Major Project\src\Database\Connector.php	30	Avoid using static access to class '\ole4\Magneto\Config\Config' in method 'setInstance'.

11	D:\Dropbox\Major Project\src\Database\Connector.php	36	Avoid using static access to class '\ole4\Magneto\Magneto' in method 'setInstance'.
12	D:\Dropbox\Major Project\src\Magneto.php	40	Avoid using static access to class '\ole4\Magneto\Config\Config' in method '__construct'.
13	D:\Dropbox\Major Project\src\Magneto.php	41	Avoid using static access to class '\ole4\Magneto\i18n\Locale' in method '__construct'.
14	D:\Dropbox\Major Project\src\Magneto.php	42	Avoid using static access to class '\ole4\Magneto\i18n\Locale' in method '__construct'.
15	D:\Dropbox\Major Project\src\Magneto.php	43	Avoid using static access to class '\ole4\Magneto\Database\Connector' in method '__construct'.
16	D:\Dropbox\Major Project\src\Magneto.php	46	Avoid using static access to class 'ole4\Magneto\Retriever' in method '__construct'.
17	D:\Dropbox\Major Project\src\Magneto.php	63	csrfWatchdog accesses the super-global variable $_SESSION.
18	D:\Dropbox\Major Project\src\Magneto.php	63	csrfWatchdog accesses the super-global variable $_SESSION.
19	D:\Dropbox\Major Project\src\Magneto.php	63	csrfWatchdog accesses the super-global variable $_POST.
20	D:\Dropbox\Major Project\src\Magneto.php	63	csrfWatchdog accesses the super-global variable $_POST.
21	D:\Dropbox\Major Project\src\Magneto.php	63	csrfWatchdog accesses the super-global variable $_POST.

2 D:\Dropbox\Major 2 Project\src\Magneto.php	63	csrfWatchdog accesses the super-global variable $_SESSION.
2 D:\Dropbox\Major 3 Project\src\Magneto.php	71	Avoid using static access to class 'ole4\Magneto\Magneto' in method 'csrfWatchdog'.
2 D:\Dropbox\Major 4 Project\src\Magneto.php	87	Avoid using static access to class '\ole4\Magneto\Config\Config' in method 'configureErrors'.
2 D:\Dropbox\Major 5 Project\src\Magneto.php	106	configureSession accesses the super-global variable $_SESSION.
2 D:\Dropbox\Major 6 Project\src\Magneto.php	119	error accesses the super-global variable $_SESSION.
2 D:\Dropbox\Major 7 Project\src\Magneto.php	126	Avoid using static access to class '\ole4\Magneto\i18n\Locale' in method 'error'.
2 D:\Dropbox\Major 8 Project\src\Models\Magnetometer.php	18	Avoid variables with short names like $id. Configured minimum length is 3.
2 D:\Dropbox\Major 9 Project\src\Models\Magnetometer.php	24	Avoid variables with short names like $id. Configured minimum length is 3.
3 D:\Dropbox\Major 0 Project\src\Models\Magnetometer.php	38	Avoid using static access to class '\ole4\Magneto\Database\Connector' in method 'saveMagnetometer'.
3 D:\Dropbox\Major 1 Project\src\Models\Magnetometer.php	38	Avoid variables with short names like $db. Configured minimum length is 3.
3 D:\Dropbox\Major 2 Project\src\Renderer.php	52	The method route() has a Cyclomatic Complexity of 10. The configured cyclomatic complexity threshold is 10.
3 D:\Dropbox\Major 3 Project\src\Renderer.php	52	route accesses the super-global variable $_GET.

3 D:\Dropbox\Major 4 Project\src\Renderer.php	52	route accesses the super-global variable $_GET.
3 D:\Dropbox\Major 5 Project\src\Renderer.php	118	The method route uses an else expression. Else is never necessary and you can simplify the code to work without else.
3 D:\Dropbox\Major 6 Project\src\Renderer.php	132	Avoid using static access to class 'ole4\Magneto\Magneto' in method 'loadPage'.
3 D:\Dropbox\Major 7 Project\src\Renderer.php	136	The method additionalData() has a Cyclomatic Complexity of 12. The configured cyclomatic complexity threshold is 10.
3 D:\Dropbox\Major 8 Project\src\Renderer.php	136	additionalData accesses the super-global variable $_POST.
3 D:\Dropbox\Major 9 Project\src\Renderer.php	136	additionalData accesses the super-global variable $_POST.
4 D:\Dropbox\Major 0 Project\src\Renderer.php	136	additionalData accesses the super-global variable $_POST.
4 D:\Dropbox\Major 1 Project\src\Renderer.php	136	additionalData accesses the super-global variable $_POST.
4 D:\Dropbox\Major 2 Project\src\Renderer.php	136	additionalData accesses the super-global variable $_GET.
4 D:\Dropbox\Major 3 Project\src\Renderer.php	136	additionalData accesses the super-global variable $_GET.
4 D:\Dropbox\Major 4 Project\src\Renderer.php	136	additionalData accesses the super-global variable $_GET.
4 D:\Dropbox\Major 5 Project\src\Renderer.php	136	additionalData accesses the super-global variable $_GET.
4 D:\Dropbox\Major 6 Project\src\Renderer.php	140	Avoid using static access to class '\ole4\Magneto\Config\Config' in method 'additionalData'.

47	D:\Dropbox\Major Project\src\Renderer.php	146	The method additionalData uses an else expression. Else is never necessary and you can simplify the code to work without else.
48	D:\Dropbox\Major Project\src\Renderer.php	157	Avoid using static access to class 'ole4\Magneto\Magneto' in method 'additionalData'.
49	D:\Dropbox\Major Project\src\Renderer.php	165	registerGlobals accesses the super-global variable $_SESSION.
50	D:\Dropbox\Major Project\src\Renderer.php	167	Avoid using static access to class '\ole4\Magneto\Config\Config' in method 'registerGlobals'.
51	D:\Dropbox\Major Project\src\Renderer.php	168	Avoid using static access to class '\ole4\Magneto\Database\Connector' in method 'registerGlobals'.
52	D:\Dropbox\Major Project\src\Renderer.php	169	Avoid using static access to class '\ole4\Magneto\i18n\Locale' in method 'registerGlobals'.
53	D:\Dropbox\Major Project\src\Renderer.php	170	Avoid using static access to class '\ole4\Magneto\i18n\Locale' in method 'registerGlobals'.
54	D:\Dropbox\Major Project\src\Renderer.php	177	addSession accesses the super-global variable $_SESSION.
55	D:\Dropbox\Major Project\src\Renderer.php	177	addSession accesses the super-global variable $_SESSION.
56	D:\Dropbox\Major Project\src\Renderer.php	186	resetSession accesses the super-global variable $_SESSION.
57	D:\Dropbox\Major Project\src\Renderer.php	186	resetSession accesses the super-global variable $_SESSION.

58	D:\Dropbox\Major Project\src\Renderer.php	186	resetSession accesses the super-global variable $_SESSION.
59	D:\Dropbox\Major Project\src\Renderer.php	186	resetSession accesses the super-global variable $_SESSION.
60	D:\Dropbox\Major Project\src\Renderer.php	186	resetSession accesses the super-global variable $_SESSION.
61	D:\Dropbox\Major Project\src\Renderer.php	186	resetSession accesses the super-global variable $_SESSION.
62	D:\Dropbox\Major Project\src\Renderer.php	186	resetSession accesses the super-global variable $_SESSION.
63	D:\Dropbox\Major Project\src\Renderer.php	186	resetSession accesses the super-global variable $_SESSION.
64	D:\Dropbox\Major Project\src\Renderer.php	193	The method resetSession uses an else expression. Else is never necessary and you can simplify the code to work without else.
65	D:\Dropbox\Major Project\src\Renderer.php	202	Avoid variables with short names like $i. Configured minimum length is 3.
66	D:\Dropbox\Major Project\src\Retriever.php	38	Avoid using static access to class '\ole4\Magneto\Config\Config' in method '__construct'.
67	D:\Dropbox\Major Project\src\Retriever.php	39	Avoid using static access to class '\ole4\Magneto\Config\Config' in method '__construct'.
68	D:\Dropbox\Major Project\src\Retriever.php	40	Avoid using static access to class '\ole4\Magneto\Config\Config' in method '__construct'.
69	D:\Dropbox\Major Project\src\Retriever.php	41	Avoid using static access to class '\ole4\Magneto\Config\Config' in method '__construct'.

70	D:\Dropbox\Major Project\src\Retriever.php	42	Avoid using static access to class '\ole4\Magneto\Config\Config' in method '__construct'.
71	D:\Dropbox\Major Project\src\Retriever.php	49	watchdog accesses the super-global variable $_GET.
72	D:\Dropbox\Major Project\src\Retriever.php	49	watchdog accesses the super-global variable $_GET.
73	D:\Dropbox\Major Project\src\Retriever.php	49	watchdog accesses the super-global variable $_GET.
74	D:\Dropbox\Major Project\src\Retriever.php	57	Avoid using static access to class 'ole4\Magneto\Magneto' in method 'watchdog'.
75	D:\Dropbox\Major Project\src\Retriever.php	58	The method watchdog uses an else expression. Else is never necessary and you can simplify the code to work without else.
76	D:\Dropbox\Major Project\src\Retriever.php	59	Avoid using static access to class 'ole4\Magneto\Magneto' in method 'watchdog'.
77	D:\Dropbox\Major Project\src\Retriever.php	67	Avoid using static access to class '\ole4\Magneto\Config\Config' in method 'setRetrieval'.
78	D:\Dropbox\Major Project\src\Retriever.php	70	retrieveData accesses the super-global variable $_SESSION.
79	D:\Dropbox\Major Project\src\Retriever.php	94	Avoid using static access to class 'ole4\Magneto\Magneto' in method 'retrieveData'.
80	D:\Dropbox\Major Project\src\Retriever.php	100	Avoid unused private methods such as 'oldProcessData'.
81	D:\Dropbox\Major Project\src\Retriever.php	107	Avoid using static access to class '\ole4\Magneto\Models\Magnetometer' in method 'oldProcessData'.

82	D:\Dropbox\Major Project\src\Retriever.php	121	processData accesses the super-global variable $_SESSION.
83	D:\Dropbox\Major Project\src\Retriever.php	121	processData accesses the super-global variable $_SESSION.
84	D:\Dropbox\Major Project\src\Retriever.php	139	Avoid using static access to class '\ole4\Magneto\Models\Magnetometer' in method 'processData'.
85	D:\Dropbox\Major Project\src\Retriever.php	176	Avoid using static access to class 'ole4\Magneto\Magneto' in method 'processData'.
86	D:\Dropbox\Major Project\src\i18n\Locale.php	22	setLanguage accesses the super-global variable $_GET.
87	D:\Dropbox\Major Project\src\i18n\Locale.php	22	setLanguage accesses the super-global variable $_GET.
88	D:\Dropbox\Major Project\src\i18n\Locale.php	22	setLanguage accesses the super-global variable $_GET.
89	D:\Dropbox\Major Project\src\i18n\Locale.php	22	setLanguage accesses the super-global variable $_SESSION.
90	D:\Dropbox\Major Project\src\i18n\Locale.php	26	The method setLanguage uses an else expression. Else is never necessary and you can simplify the code to work without else.
91	D:\Dropbox\Major Project\src\i18n\Locale.php	37	determineLanguage accesses the super-global variable $_SESSION.
92	D:\Dropbox\Major Project\src\i18n\Locale.php	37	determineLanguage accesses the super-global variable $_SESSION.
93	D:\Dropbox\Major Project\src\i18n\Locale.php	37	determineLanguage accesses the super-global variable $_SESSION.
94	D:\Dropbox\Major Project\src\i18n\Locale.php	37	determineLanguage accesses the super-global variable $_SERVER.

95 D:\Dropbox\Major Project\src\i18n\Locale.php	46	The method determineLanguage uses an else expression. Else is never necessary and you can simplify the code to work without else.
96 D:\Dropbox\Major Project\src\i18n\Locale.php	50	The method determineLanguage uses an else expression. Else is never necessary and you can simplify the code to work without else.
97 D:\Dropbox\Major Project\src\i18n\Locale.php	80	Avoid using static access to class '\ole4\Magneto\Magneto' in method 'setLocale'.
98 D:\Dropbox\Major Project\src\i18n\Locale.php	85	Avoid using static access to class '\ole4\Magneto\Magneto' in method 'setLocale'.

I. Manual Testing Table

#	Test	Expectation	Pass?
1	**General Interface**		
1.1	Does the application load when its start point is opened in a web browser?	The homepage view is rendered, and the application is loaded.	Yes
1.2	Does the About page load when the 'About' navigation link is clicked?	The about page view is rendered	Yes
1.3	Does the Settings page load when the 'Settings' navigation link is clicked?	The settings page view is rendered	Yes
1.4	Does the Navigator load when the 'Navigate Entries' link is clicked?	The Navigator page view is rendered	Yes
1.5	Does the latest entry load when the 'Latest Entry' link is clicked?	The latest magnetometer entry is rendered in the Graphing View	Yes
1.6	Does all entries load when 'Display All' is clicked?	All data from the database is rendered in the Graphing View	Yes
2	**Locale**		
2.1	Does the language switch into Welsh when the 'Cymrage' link is clicked?	The website switches into Welsh	Yes
2.2	Switch the language back into English by clicking the 'Saesneg' language link	The website switches back into English	Yes
2.3	The application autodetects browser on language as English on first load	The website loads initially in English	Yes
2.4	The application autodetects browser language as Welsh on first load	The website loads initially in Welsh	Yes
2.5	Entering language='cy' as GET parameter sets language into Welsh	The website loads into Welsh	Yes
3	**Navigator**		
3.1	Click the option box to check whether there is a full range of dates from 2014 to today's date	There is a full range of dates as expected	Yes
3.2	Does the number of entries on the Navigator view correspond to the Max Elements setting on the Settings page?	The number correlates to the setting	Yes

3.3	Are the elements in the Navigator view entries displayed in chronological order?	They are displayed in chronological order	Yes
3.4	Does clicking 'Load Graphing Tool' load the graphing view with the selected entries?	The Graphing Tool is displayed displaying the selected entries	Yes
3.5	Does clicking 'Load Tables Only' display the selected entries as a table?	The Tabular view is displayed displaying the selected entries	Yes
3.6	Does clicking 'Add Magnetometer Entry' add another selection box?	A new magnetometer entry box is added	No – this functionality is unimplemented
3.7	Does clicking 'Add Radio Entry' add a selection box for amateur radio entries?	A new amateur radio entry box is added	No – this functionality is unimplemented
3.8	Does clicking 'Add ESA Entry' add a selection box for Internet data entries?	A new Internet entry box is added	No – this functionality is unimplemented
4	**Graphing Tool (View)**		
4.1	Is the loaded data displayed on a chart, with dates on the X axis and values on the Y axis?	Graph is displayed properly	Yes
4.2	Do the Bar View and Line View change the type of graph?	Graph changes when buttons are clicked	Yes
4.3	Does Export XML render the current data in XML?	XML is presented	Yes
4.4	Does Export JSON render the current data in JSON?	JSON is presented	Yes
4.5	Is the loaded data displayed as a table underneath the chart?	Tabular data is presented underneath the chart	Yes
4.6	When viewing only one entry, does the graph default to bar view?	Graph defaults to bar view	Yes
4.7	In Debug Mode, is the Graphing View Debugger displayed?	Debugger is displayed	Yes
5	**Table View / Tabular Data**		
5.1	Is loaded data displayed in a table?	Table is loaded into a table	Yes
6	**Settings**		
6.1	Does changing and submitting a different Max Elements value take effect?	Value is changed	Yes
7	Magnetometer		
7.1	Retrieves data from magnetometer on daily basis	Data is retrieved	Yes

J. Stress Test Table

#	Test	Expectation	Pass?
1	**General Interface**		
1.1	Does the app load the homepage when multiple page GET values are provided?	Homepage is rendered	Yes
1.2	Does the app load a valid page when two valid pages are provided as GET values?	One of the pages is loaded	Yes – the latter is used
2	**Locale**		
2.1	Does the app successfully load English when the browser's language is neither English or Welsh?	English is loaded	Yes
2.2	Does the app successfully load English when a false language is added to the GET?	English is loaded	Yes
2.3	Does the app load English when the language is tampered with in the session cookie?	English is loaded	Yes
3	**Navigator**		
No additional tests.			
4	**Graphing Tool / View**		
4.1	Does 'Latest Entry' and additional IDs load properly?	Latest Entry takes precedence and other IDs are ignored	Yes
4.2	Does 'Display All' and additional IDs load properly?	Display All takes precedence and other IDs are ignored	Yes
4.3	Graphing View is navigated to without data	Displays 'No data'	Yes
4.4	Does 'Display All' export JSON return the JSON of all entries?	JSON exported	No – URI too long
4.5	Does 'Display All' export XML return the XML of all entries?	XML exported	No – URI too long
4.6	Are invalid / tampered POST data entries displayed in the graph, or cause any issues?	Malformed entries are ignored	Yes
4.7	Are invalid / tampered GET data entries displayed in the graph, or cause any issues?	Malformed entries are ignored	Yes
5	Table View / Tabular Data		
5.1	Table view is navigated to without data	Displays 'No data'	No – displays blank table
5.2	Are invalid / tampered POST data entries displayed in the table, or cause any issues?	Malformed entries are ignored	Yes

5.3	Are invalid / tampered GET data entries displayed in the graph, or cause any issues?	Malformed entries are ignored	Yes
6	Settings		
6.1	Max Elements is non-numerical value	Input rejected	Yes
6.2	Max Elements is non-integer value	Input rejected	Yes
7	CSRF Protection		
7.1	Navigator submitted without CSRF token	App halted	Yes
7.2	Navigator submitted with invalid token	App halted	Yes
7.3	Settings submitted without CSRF token	App halted	Yes
7.4	Settings submitted with invalid token	App halted	Yes

K. API Testing Table

#	Test	Expectation	Pass?
1	GET with valid values	JSON returned	Yes
2	POST with valid values	JSON returned	No – error 400 Bad Request
3	GET with non-existent values	'No magnetometer entries found'	Yes
4	POST with non-existent values	'No magnetometer entries found'	No – error 400 Bad Request
5	GET with valid values, XML flag on	XML returned	Yes
6	POST with valid values, XML flag on	XML returned	No – error 400 Bad Request
7	GET with non-existent values, XML flag on	<error>No magnetometer entries found'</error>	Yes
8	POST with non-existent values, XML flag on	<error>No magnetometer entries found'</error>	No – error 400 Bad Request
9	Access API entry-point with no values	Error 400 Bad Request	Yes
10	Latest flag	Return latest entry as JSON	Yes
11	Latest flag with XML flag	Return latest entry as XML	Yes
12	All flag	Return all database entries as JSON	Yes
13	All flag with XML flag	Return all database entries as XML	Yes
14	GET with both valid and non-existent values	Only valid entries are returned	Yes
15	GET with both valid and malformed values	Only valid entries are returned	Yes
16	POST with both valid and non-existent values	Only valid entries are returned	No – error 400 Bad Request
17	POST with both valid and malformed values	Only valid entries are returned	No – error 400 Bad Request

7. Annotated Bibliography

[1] Z. Gorvett, "The Sun creates 'space weather' that affects us all," BBC, 23 September 2015. [Online]. Available: http://www.bbc.co.uk/earth/story/20150923-the-sun-creates-space-weather-that-affects-us-all. [Accessed 24 April 2018].

An interesting article from the BBC that explains space weather.

[2] MuleSoft, "What is an API? (Application Programming Interface)," MuleSoft, 20 April 2018. [Online]. Available: https://www.mulesoft.com/resources/api/what-is-an-api. [Accessed 24 April 2018].

Helpful overview of what an API is.

[3] J. Lengstorf, "JSON: What It Is, How It Works & How to Use It," CopterLabs, 2 July 2009. [Online]. Available: https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/. [Accessed 24 April 2018].

Overview of the JSON standard.

[4] M. O. D. Coul, "Mac HFS+ Timestamp Converter," EpochConverter.com, 19 February 2016. [Online]. Available: https://www.epochconverter.com/mac. [Accessed 24 April 2018].

Useful tool and background information on converting timestamps from one format into another and on differing epochs.

[5] J. Ratcliffe, "Solar Disturbances," in *An Introduction to the Ionosphere and the Magnetosphere*, London, Cambridge University Press, 1972, p. 14.

An interesting but incredibly in-depth book on the ionosphere and magnetosphere. The introductory chapters were useful on getting some background information, but the remainder was unfortunately too advanced.

[6] G. Holman and S. Benedict, "What is a Solar Flare?," NASA; NASA Goddard Space Flight Center, 23 September 1996. [Online]. Available: https://hesperia.gsfc.nasa.gov/sftheory/flare.htm. [Accessed 24 April 2018].

Insightful but very retro information on solar flares from NASA.

[7] Space Weather Prediction Center, "Coronal Mass Ejections," National Oceanic and Atmospheric Administration, 27 December 2014. [Online]. Available: https://www.swpc.noaa.gov/phenomena/coronal-mass-ejections. [Accessed 24 April 2018].

Interesting, and not too complex information on coronal mass ejections.

[8] R. Sharpe, "Just what is SMB?," Samba, 8 October 2002. [Online]. Available: https://www.samba.org/cifs/docs/what-is-smb.html. [Accessed 24 April 2018].

Background information on the SMB network protocol from the people at Samba.

[9] R. Boyd, "9 Scrum Metrics To Keep Your Team On Track," Pragmatic Marketing, 21 November 2014. [Online]. Available: https://pragmaticmarketing.com/resources/articles/9-scrum-metrics-to-keep-your-team-on-track. [Accessed 24 April 2018].

A helpful read about keeping motivated as a Scrum team.

[10] L. R. Quinn, "XML Essentials," W3C, 2015. [Online]. Available: https://www.w3.org/standards/xml/core. [Accessed 24 April 2018].

An overview and introduction to the XML mark-up language standard.

[11] Welsh Language Services, "https://www.aber.ac.uk/en/cgg/bilingual-policy/welsh-language-scheme/," Aberystwyth University, 20 February 2014. [Online]. Available: https://www.aber.ac.uk/en/cgg/bilingual-policy/welsh-language-scheme/. [Accessed 24 April 2018].

University policy document on Welsh Language provisions and bilingualism.

[12] D. N. Papitashvili, "Geomagnetic Data Master Catalogue," World Data Centre for Geomagnetism (Edinburgh); NASA/GSFC, 1999. [Online]. Available: http://www.wdc.bgs.ac.uk/catalog/master.html. [Accessed 24 April 2018].

An incredibly in-depth and constantly updated data resource for magnetometer readings and geomagnetic data. Makes this program seem extremely tiny.

[13] Stanford University, "Tracking Solar Flares - Ionosphere," 4 June 2009. [Online]. Available: http://solar-center.stanford.edu/SID/activities/ionosphere.html. [Accessed 24 April 2018].

Helpful background information specifically pertaining to the ionosphere.

[14] GNU Radio Foundation, "About GNU Radio," 17 March 2018. [Online]. Available: https://gnuradio.org/about/. [Accessed 24 April 2018].

General information on what GNU Radio is.

[15] The Apache Software Foundation, "Apache Module mod_rewrite," 30 March 2018. [Online]. Available: https://httpd.apache.org/docs/2.4/howto/htaccess.html. [Accessed 24 April 2018].

Technical documentation on how mod_rewrite works by Apache.

[16] A. Mezenin, "Can Laravel 5.1 application run without .htaccess rewrite?," StackOverflow, 10 March 2016. [Online]. Available: https://stackoverflow.com/questions/35924798/can-laravel-5-1-application-run-without-htaccess-rewrite. [Accessed 24 April 2018].

A discussion on Stack Overflow as to whether the Laravel PHP framework can operate without .htaccess – the answer is no.

[17] Google, "Top Ways Websites Get Hacked By Spammers," 22 December 2017. [Online]. Available: https://developers.google.com/web/fundamentals/security/hacked/top_ways_websites_get_hacked_by_spammers. [Accessed 24 April 2018].

Editorial by Google for use by webmasters on how to prevent their websites and content platforms being targeted by exploits, spammers, etc.

[18] D. Dede, "Understanding .htaccess attacks - Part 1," Sucuri, 27 May 2011. [Online]. Available: https://blog.sucuri.net/2011/05/understanding-htaccess-attacks-part-1.html. [Accessed 24 April 2018].

A detailed look as to how malicious code attacks .htaccess to push maliciously generated content.

[19] M. Pinola, "How to Map a Network Drive in Windows 10," Purch, 25 August 2015. [Online]. Available: https://www.laptopmag.com/articles/map-network-drive-windows-10. [Accessed 24 April 2018].

Basic editorial on how to map network drives within Windows 10.

[20] The PHP Group, "FTP Manual," 16 April 2008. [Online]. Available: http://php.net/manual/en/book.ftp.php. [Accessed 24 April 2018].

PHP documentation section on FTP related functionality.

[21] B. Sevilleja, "The Ins and Outs of Token Based Authentication," Scotch.io LLC, 21 January 2015. [Online]. Available: https://scotch.io/tutorials/the-ins-and-outs-of-token-based-authentication. [Accessed 25 April 2018].

Exploring token-based authentication used in RESTful services.

[22] G. Boer, "What is Scrum?," Microsoft Corporation, 2018. [Online]. Available: https://www.visualstudio.com/learn/what-is-scrum/. [Accessed 26 April 2018].

Microsoft's own take on the Scrum methodology. An interesting read.

[23] H. Takeuhci and I. Nonaka, "The New New Product Development Game," Harvard Business Report, January 1986. [Online]. Available: https://hbr.org/1986/01/the-new-new-product-development-game. [Accessed 2018 April 26].

An article from the Harvard Business Report back when Scrum was just emerging in the 80s. Really interesting read.

[24] C. Starr, "Distributed Scrum," Microsoft Corporation; Scrum.org, July 2012. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2013/jj620910(v=vs.120). [Accessed 26 April 2018].

A small article talking about the challenges Distributed Scrum teams face.

[25] R. Kasturi, "Scrum Communication," Scrum Alliance Inc., 14 October 2014. [Online]. Available: https://www.scrumalliance.org/community/articles/2014/october/scrum-communication. [Accessed 26 April 2018].

An article discussing the importance of communication within the Scrum methodology.

[26] F. Attanasio, "Playing the ScrumMaster Role," Scrum Alliance, Inc., 13 December 2013. [Online]. Available: https://www.scrumalliance.org/community/articles/2013/december/playing-the-scrummaster-role. [Accessed 26 April 2018].

A brief overview of the Scrum Master role and what's involved.

[27] NC State University, "The Agile Landscape," 26 November 2011. [Online]. Available: http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2011/ch6_6d_sk. [Accessed 27 April 2018].

An educational document from a wiki that covers several agile methodologies, including Feature-Driven Development.

[28] S. W. Ambler, "Feature Driven Development (FDD) and Agile Modeling," Ambysoft, Inc., 17 January 2018. [Online]. Available: http://agilemodeling.com/essays/fdd.htm. [Accessed 27 April 2018].

In-depth exploration of Feature-Driven development.

[29] L. Karam, "Feature Driven Development (FDD) Processes and Comparison To Other Agile Methodologies," Apiumhub, 5 May 2017. [Online]. Available: https://www.upwork.com/hiring/for-clients/agile-methodologies/. [Accessed 27 April 2018].

The processes and sections of Feature-Driven Development broken down bit-by-bit. Interesting read.

[30] The PHP Group, "History of PHP," 2018. [Online]. Available: http://php.net/manual/en/history.php.php. [Accessed 25 April 2018].

Official PHP documentation on the history of the language and how it came about. Once a very ugly looking language.

[31] K. Yank, "Which Server Side Language Is Right For You?," Sitepoint, 9 October 2001. [Online]. Available: https://www.sitepoint.com/server-side-language-right/. [Accessed 25 April 2018].

A dated but still valid comparison of server-side programming languages.

[32] The PHP Group, "Supported Versions," 2018. [Online]. Available: http://php.net/supported-versions.php. [Accessed 25 April 2018].

The current timeline of supported PHP versions.

[33] T. Reenskaug and J. O. Coplien, "The DCI Architecture: A New Vision of Object-Oriented Programming," Artima Inc, 20 March 2009. [Online]. Available: https://www.artima.com/articles/dci_vision.html. [Accessed 25 April 2018].

Discussing the shortcomings of object-oriented programming and why MVC as an architectural pattern makes sense on the web. Interesting but long read.

[34] T. Otwell, "Laravel 5.6 Documentation," 2018. [Online]. Available: https://laravel.com/docs/5.6. [Accessed 25 April 2018].

Official Laravel 5.6 documentation.

[35] Python Software Foundation, "History of the Software - Python Reference Manual," 22 June 2001. [Online]. Available: https://docs.python.org/2.0/ref/node92.html. [Accessed 25 April 2018].

Dated documentation but a helpful overview of the history of the Python language.

[36] N. Diakopoulos and S. Cass, "Interactive: The Top Programming Languages 2017," IEEE Spectrum, 18 July 2017. [Online]. Available: https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017. [Accessed 25 April 2018].

Incredibly interesting – the top ranked programming languages in multiple categories of 2017.

[37] Django Software Foundation, "Django Documentation," 2018. [Online]. Available: https://docs.djangoproject.com/en/2.0/. [Accessed 25 April 2018].

Official Django framework documentation.

[38] CodeAcademy, "JavaScript and Node.js," 2018. [Online]. Available: https://www.codecademy.com/articles/what-is-node. [Accessed 25 April 2018].

An overview of what Node.js is and how it pertains to JavaScript.

[39] Altexsoft Inc, "The Good and the Bad of JavaScript Full Stack Development," 18 January 2018. [Online]. Available: https://www.codecademy.com/articles/what-is-node. [Accessed 25 April 2018].

Another interesting read on the pros and cons of using JavaScript everywhere on the stack – both on the front and back ends.

[40] S. Basu, "Ruby on Rails Study Guide: The History of Rails," Envato Pty Ltd., 22 January 2013. [Online]. Available: https://code.tutsplus.com/articles/ruby-on-rails-study-guide-the-history-of-rails--net-29439. [Accessed 25 April 2018].

A brief history on the history of Rails, its relationship with Ruby, and its developers.

[41] D. Q, "Dan: Why I Volunteer with Three Rings," Three Rings CIC, December 2016. [Online]. Available: https://www.threerings.org.uk/our-team/volunteer/dans-story/. [Accessed 25 April 2018].

An article by a developer from Three Rings CIC – an Aber alumnus and former volunteer of Aberystwyth Nightline, who wrote the Three Rings application using Ruby on Rails and is now used by hundreds of volunteer organisations across the UK.

[42] Chart.js, "Chart.js Homepage," 2018. [Online]. Available: http://www.chartjs.org/. [Accessed 25 April 2018].

Chart.js official documentation.

[43] M. Otto and J. Thornton, "Introduction - Bootstrap," Twitter Inc., 25 April 2018. [Online]. Available: https://getbootstrap.com/docs/4.1/getting-started/introduction/. [Accessed 25 April 2018].

Bootstrap official documentation.

[44] K. Sierra, "When the "best tool for the job"... isn't," Creating Passionate Users, 1 August 2006. [Online]. Available: http://headrush.typepad.com/creating_passionate_users/2006/08/when_the_best_t.html. [Accessed 25 April 2018].

An incredibly enlightening read arguing striking a balance between finding the right tool for the job and using what feels best to the developer. After all, they have to be the one to use it.

[45] The PHP Group, "PDO Introduction," 25 April 2008. [Online]. Available: http://php.net/manual/en/intro.pdo.php. [Accessed April 25 2018].

Official PHP documentation on the PDO object.

[4 6] JetBrains s.r.o, "PhpStorm - Features," 2018. [Online]. Available: https://www.jetbrains.com/phpstorm/features/. [Accessed 25 April 2018].

Official PhpStorm documentation.

[4 7] G. Menegaz, "SQL Injection Attack: What it is, and how to prevent it," ZDNet, 13 July 2012. [Online]. Available: https://www.zdnet.com/article/sql-injection-attack-what-is-it-and-how-to-prevent-it/. [Accessed 26 April 2018].

A surprisingly detailed overview on SQL injection attacks.

[4 8] C. Shiflett, "Cross-Site Request Forgeries," in *Essential PHP Security*, Sebastopol, CA, O'Reilly Media, Inc., 2006, pp. 24-28.

A helpful guide from a rather dated textbook – still valid provided you cross-reference with materials online.

[4 9] Atlassian, "What is Git," 2018. [Online]. Available: https://www.atlassian.com/git/tutorials/what-is-git. [Accessed 25 April 2018].

Helpful introduction to what Git is by Atlassian.

[5 0] Atlassian, "Bitbucket: What is Bitbucket?," 7 March 2018. [Online]. Available: https://confluence.atlassian.com/confeval/development-tools-evaluator-resources/bitbucket/bitbucket-what-is-bitbucket. [Accessed 252 April 2018].

Official Bitbucket documentation.

[5 1] Dropbox, "What is Dropbox?," (YouTube Video), 21 May 2015. [Online]. Available: https://www.youtube.com/watch?v=QADSH8XYx_A. [Accessed 25 April 2018].

A brief YouTube video that describes what Dropbox is and how it works.

[5 2] D. Miessler, "An MVC Primer," 6 October 2017. [Online]. Available: https://danielmiessler.com/study/mvc/. [Accessed 27 April 2018].

An immensely helpful overview with graphics on how MVC works.

[5 3] DSDM Consortium, "MoSCoW Prioritisation - DSDM Atern Handbook (2008)," Agile Business Consortium Limited, July 2008. [Online]. Available: https://www.agilebusiness.org/content/moscow-prioritisation-0. [Accessed 28 April 2018].

Useful information on how MosCoW works – forms part of a greater handbook on the DSDM/Atern frameworks.

[5 4] N. Adermann, J. Boggiano and C. Community, "Introduction - Composer," 2018. [Online]. Available: https://getcomposer.org/doc/00-intro.md. [Accessed 28 April 2018].

Official Composer documentation.

[55] N. Adermann, J. Boggiano and C. Community, "Autoloading - Basic Usage - Composer," 2018. [Online]. Available: https://getcomposer.org/doc/01-basic-usage.md#autoloading. [Accessed 28 April 2018].

Guide on how the autoloading functionality works in Composer.

[56] P. M. Jones, P. Sturgeon and L. Garfield, "PSR-4 Autoloader," PHP Framework Interop Group, 11 April 2018. [Online]. Available: https://www.php-fig.org/psr/psr-4/. [Accessed 28 April 2018].

An excerpt from the PSR coding standards covering PSR-4 Autoloading. A must-read for any PHP developer.

[57] J. Boggiano, "Monolog - Logging for PHP," 19 June 2017. [Online]. Available: https://github.com/Seldaek/monolog. [Accessed 29 April 2018].

GitHub repository and information for the Monolog library.

[58] C. R. Dougherty, Carnegie Mellon University (CMU) Software Engineering Institute; CERT, 31 December 2008. [Online]. Available: https://www.kb.cert.org/vuls/id/836068. [Accessed 28 April 2018].

Security advisory on the now insecure MD5 algorithm and why it shouldn't be used.

[59] J. Roper, "Cracking Random Number Generators - Part 1," 20 September 2018. [Online]. Available: https://jazzy.id.au/2010/09/20/cracking_random_number_generators_part_1.html. [Accessed 28 April 2018].

An interesting, if lengthy editorial on random number generators, their predictability, and how to beat them.

[60] The PHP Group, "uniqid," 2018. [Online]. Available: http://php.net/manual/en/function.uniqid.php. [Accessed 28 April 2018].

Official PHP documentation.

[61] OODesign.com, "Singleton Pattern," 25 April 2018. [Online]. Available: http://www.oodesign.com/singleton-pattern.html. [Accessed 29 April 2018].

Brief overview of the singleton design pattern/antipattern.

[62] The PHP Group, "Gettext - Human Language and Character Encoding Support," 2018. [Online]. Available: https://secure.php.net/manual/en/book.gettext.php. [Accessed 29 April 2018].

Official PHP documentation.

[63] Interaction Design Foundation, “KISS (Keep it Simple, Stupid) - A Design Principle,” 3 April 2018. [Online]. Available: https://www.interaction-design.org/literature/article/kiss-keep-it-simple-stupid-a-design-principle. [Accessed 29 April 2018].

An in-depth article on the KISS design principle. Useful read.

[64] SensioLabs, “Twig - The flexible, fast, and secure template engine for PHP,” 2018. [Online]. Available: https://twig.symfony.com. [Accessed 29 April 2018].

Official Twig documentation.

[65] CupRacer, “How to pass a php variable to twig template,” StackOverflow, 3 February 2016. [Online]. Available: https://stackoverflow.com/questions/35161380/how-to-pass-a-php-variable-to-twig-template. [Accessed 29 April 2018].

A discussion on Stack Overflow on how to pass PHP variables into Twig.

[66] T. Berners-Lee, “Cool URIs don't change,” W3 Consortium, 1998. [Online]. Available: https://www.w3.org/Provider/Style/URI.html. [Accessed 29 April 2018].

A must-read report from Berners-Lee about fancy/tidy URLs. Except he talked about this in 1998, way before most modern frameworks and rewrite engines.

[67] Refactoring Guru, “Switch Statements,” 27 April 2018. [Online]. Available: https://refactoring.guru/smells/switch-statements. [Accessed 29 April 2018].

Switch statements and how they are often an indicating factor of code smells in OOP languages.

[68] J. Skeet, “Polymorphism - Define In Just Two Sentences,” StackOverflow, 7 November 2013. [Online]. Available: https://stackoverflow.com/questions/409969/polymorphism-define-in-just-two-sentences. [Accessed 29 April 2018].

Possibly one of the best explanations for polymorphism.

[69] S. Grabaum and R. Appelman, “Steffen-99/SMB - SMB,” GitHub, Inc., 11 November 2015. [Online]. Available: https://github.com/Steffen-99/SMB. [Accessed 29 April 2018].

GitHub repository for the SMB library.

[70] The PHP Group, “What is PEAR?,” 29 April 2018. [Online]. Available: https://pear.php.net/manual/en/about.pear.php. [Accessed 29 April 2018].

Official PHP documentation.

[71] D. Marcus, "PHP PDO Prepared Statements Tutorial to Prevent SQL Injection," WebsiteBeaver, 26 November 2017. [Online]. Available: https://websitebeaver.com/php-pdo-prepared-statements-to-prevent-sql-injection. [Accessed 29 April 2018].

Useful information on using PDO prepared statements.

[72] Mozilla; Mozilla Contributors, "MDN Web Docs: 400 Bad Request," 27 November 2017. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/400. [Accessed 30 April 2018].

Mozilla documentation on HTTP response codes.

[73] A. Mourzenko, "Should I include PHP code in HTML or HTML in PHP?," StackOverflow, 4 August 2015. [Online]. Available: https://softwareengineering.stackexchange.com/questions/291819/should-i-include-php-code-in-html-or-html-in-php. [Accessed 1 May 2018].

A discussion shedding light on whether mixing code and mark-up is bad practice or not.

[74] The PHP Group, "Alternative syntax for control structures," 2018. [Online]. Available: http://php.net/manual/en/control-structures.alternative-syntax.php. [Accessed 1 May 2018].

Official PHP documentation.

[75] P. Ajtai, "Single vs double quotes in PHP," PHP.earth, 21 January 2018. [Online]. Available: https://php.earth/docs/faq/misc/single-vs-double-quotes. [Accessed 1 May 2018].

Something that all PHP developers should understand; the usage of single and double quotation marks.

[76] E. Marcotte, "Responsive Web Design," A List Apart, 25 May 2010. [Online]. Available: http://alistapart.com/article/responsive-web-design/. [Accessed 1 May 2018].

The first known instance of the term 'responsive design' being used on the Internet, and an overview of what it means. Must read.

[77] SensioLabs; Twig, "Introduction - Twig," 2018. [Online]. Available: https://twig.symfony.com/doc/1.x/intro.html. [Accessed 1 May 2018].

Official Twig documentation.

[78] Ł. Adamczuk, "7 Reasons to use a PHP template engine for your webpage," Schibsted Tech Polska, 3 June 2015. [Online]. Available: https://www.schibsted.pl/blog/7-reasons-to-use-a-php-template-engine-for-your-webpage/. [Accessed 1 May 2018].

The benefits of using a templating engine such as Twig.

[79] D. Rodger and T. P. D. , "What are the real advantages of templating engines over just using PHP?," StackOverflow, 25 February 2010. [Online]. Available: https://stackoverflow.com/questions/2334643/what-are-the-real-advantages-of-templating-engines-over-just-using-php. [Accessed 1 May 2018].

Further discussion on the benefits of using a templating engine.

[80] Bootstrap; Twitter, Inc., "Dashboard Template," 2018. [Online]. Available: https://getbootstrap.com/docs/4.0/examples/dashboard/. [Accessed 1 May 2018].

Dashboard example from Bootstrap that was later incorporated into the project itself.

[81] C. House, "A Complete Guide to Grid," CSS-Tricks, 25 April 2018. [Online]. Available: https://css-tricks.com/snippets/css/complete-guide-grid/. [Accessed 1 May 2018].

A helpful guide to the new CSS grid.

[82] R. Data, "XML and XSLT," W3schools, 2018. [Online]. Available: https://www.w3schools.com/xml/xml_xslt.asp. [Accessed 1 May 2018].

Basic overview of XSLT and how to format XML.

[83] B. Moon, "Using ini files for PHP application settings," 19 January 2010. [Online]. Available: http://brian.moonspot.net/using-ini-files-for-php-application-settings. [Accessed 1 May 2018].

Interesting guide for using INI files for PHP programs.

[84] The PHP Group, "parse_ini_file Manual," 2018. [Online]. Available: http://php.net/manual/en/function.parse-ini-file.php. [Accessed 1 May 2018].

Official PHP documentation.

[85] Magicalex, "Writeinifile," GitHub, Inc., 21 March 2017. [Online]. Available: https://github.com/Magicalex/WriteiniFile. [Accessed 1 May 2018].

GitHub repository for library originally intended to be used within the program for writing to INI files for configuration.

[86] C. LaViska, "A Better Way to Write Config Files in PHP," ABeautifulSite LLC, 2 May 2016. [Online]. Available: https://www.abeautifulsite.net/a-better-way-to-write-config-files-in-php. [Accessed 1 May 2018].

The config solution that was ultimately implemented to allow for easy writing to file.

[87] S. Ford, “Old Code, New Tricks,” Agile Alliance, 2015. [Online]. Available: https://www.agilealliance.org/resources/sessions/old-code-new-tricks/. [Accessed 1 May 2018].

An excerpt from a talk regarding working with legacy code.

[88] S. Aghaei, M. A. Nematbakhsh and H. K. Farsani, “Evolution of the World Wide Web: From Web 1.0 to Web 4.0,” *International Journal of Web & Semantic Technology (IJWesT),* vol. 3, no. 1, p. 10, January 2012.

Interesting journal entry on the evolution of the Internet.

[89] J. Gros-Dubois, “Statically typed vs dynamically typed languages,” Hackernoon, 29 April 2017. [Online]. Available: https://hackernoon.com/statically-typed-vs-dynamically-typed-languages-e4778e1ca55. [Accessed 01 May 2018].

Basic rundown on the differences between statically typed and dynamically typed languages.

[90] M. Asay, “JavaScript for squares: The incredible rise of TypeScript,” InfoWorld, 21 July 2017. [Online]. Available: https://www.infoworld.com/article/3209652/application-development/javascript-for-squares-the-incredible-rise-of-typescript.html. [Accessed 1 May 2018].

An interesting article outlining the success that TypeScript is currently enjoying, thanks in part to the popularity of the Angular JavaScript framework.

[91] M. Teutsch, “Introduction to PhpDoc,” Sitepoint, 9 January 2012. [Online]. Available: https://www.sitepoint.com/introduction-to-phpdoc/. [Accessed 1 May 2018].

A basic introduction to the PHPDoc standard.

[92] CMARIX TechnoLabs Pvt. Ltd., “Benefits of using PHPUnit,” 14 December 2016. [Online]. Available: http://www.cmarix.com/benefits-of-using-phpunit/. [Accessed 3 May 2018].

The perks of using a unit testing framework like PHPUnit.

[93] J. Atwood, “In Programming, One Is The Lonelinest Number,” Coding Horror, 19 June 2007. [Online]. Available: https://blog.codinghorror.com/in-programming-one-is-the-loneliest-number/. [Accessed 7 May 2017].

An incredibly insightful read on the unexpected dangers of coding independently and the challenges a solo developer might face.

[94] D. McNicol, “Extreme Programming For One,” C2.com, 11 November 2005. [Online]. Available: http://c2.com/xp/ExtremeProgrammingForOne.html. [Accessed 7 May 2018].

A discussion on adapting Extreme Programming for solo developers.

[95] P. Johnson, “Can a lone developer use Agile?,” ITworld, 20 November 2012. [Online]. Available: https://www.itworld.com/article/2714022/it-management/can-a-lone-developer-use-agile-.html. [Accessed 7 May 2018].

An interesting article on how to make use of agile methodologies as efficiently as possible even when working alone.