

To Achieve Secure and Interoperable Transfer of Patient Sensor Data to an Electronic Health Record Database



Harjeet Brar

School of Engineering, University of Warwick

ES327 Technical Report

Supervisor: Dr Georgios Despotou

October 2017 – March 2018

Executive Summary

Currently there is no single fully automated Internet of Things (IoT) system in existence, which acquires health sensor data and securely transfers this to an EHR database system, using an interoperable standard of health codes. Although the number of IoT devices is rapidly increasing, the utilisation of this technology in the healthcare industry is static. This report will therefore explore how the technology can be implemented, to set the foundations that ultimately improve the provision of healthcare.

This report focuses on researching a method of integrating MySignals sensor and hardware equipment onto an Arduino Uno microcontroller, along with an Arduino Ethernet Shield. To then ultimately construct systems capable of securely inserting pseudo sensor data into Relational Database Management Systems (RDBMS). This RDBMS can then interact with external authorised systems, to provide added functionalities to the overall system. Ultimately, the aim is to achieve the secure and semantically interoperable transfer of pseudo sensor data, from the Arduino Ethernet shield to the assigned RDBMS (MySQL).

Design A focuses on utilising a fully-automated system, which functions efficiently and effectively in being able to establish a secure connection between the MySignals HW shield and the sensor device. Following this, pseudo sensor data is declared and transferred from a single upload command of the code in the Arduino IDE. This design however lacks utilisation of encryption protocols, which leaves patient sensor data compromised to attacks from unauthorised systems or personnel. However, the Ethernet connection was preferred for this reason, preventing the chance of an outside attack occurring, due to the secure nature of its connection to the router.

Design B on the other hand, utilised a semi-autonomous design, where the sensor is connected using the same method. Pseudo sensor data is then declared and stored into a XML format, in a secure location. This data was then securely inserted into the MySQL database, using a PHP webpage, which was compiled from the languages: HTML, PHP and jQuery. Due the two-part method implemented in transferring the sensor data, this design could be said to be the more secure, compared to Design A.

A promising future for the healthcare industry can be guaranteed through the successful implementation of an EHR management system. Although Design B is the more secure of the two, Design A offers functionality advantages, through its ability to be implemented with a higher level of feasibility, given its easier functionality. Therefore, the research conducted in this project lays the foundation for an integrated, personal use sensor data storage system. However, extended research

will be necessary to implement functionalities and the necessary security protocols required by individual healthcare service providers, such as the NHS England.

Acknowledgements

I would like to use this section to personally acknowledge my supervisor, Dr Georgios Despotou for the continued support he has provided, through the entirety of this project. Being able to utilise his expertise and passion for the advancement of the healthcare industry, has been extremely helpful in ensuring the project was a success. I would also like to extend my gratitude towards both Dr Despotou and WMG for allowing me to carry out my extensive research project, and use the MySignals equipment as a fundamental component of this.

Table of Contents

1	Introduction	1
1.1	The Problem Being Addressed	1
1.2	Background Information	1
1.3	Design Implementation	2
1.3.1	Design A.....	2
1.3.2	Design B.....	2
1.4	Project Time and Risk Management	3
1.4.1	Time Management	3
1.4.2	Risk Management.....	3
1.5	Project Specification.....	3
1.6	Research Methodology	4
2	Literature Review	5
2.1	Electronic Health Records	5
2.1.1	Overview.....	5
2.1.2	Functionality	5
2.2	Database Management System (DBMS)	5
2.2.1	Structured Query Language (SQL)	6
2.2.2	SQL Database Management Options	6
2.2.3	RDBMS Administrative Tools.....	7
2.3	Server-side Programming.....	7
2.3.1	Internet Protocol (IP).....	7
2.3.2	HTTP	8
2.3.3	Hypertext Pre-processor (PHP) Language	8
2.3.4	Database Connectors and Drivers	9
2.3.5	PHP Extensions	9
2.4	EHR Management Package	10
2.4.1	OpenEMR.....	10
2.5	Software Stack Components	11
2.5.1	XAMPP	11
2.5.2	Apache.....	12
2.6	Data Security	12
2.6.1	Secure Database Connection	12
2.6.2	Encryption	12
2.6.3	Data Protection Act (1998) and Human Rights Act (1998).....	13
2.7	Internet of Things (IoT).....	13

2.7.1	Bluetooth Low Energy (BLE) Technology.....	13
2.7.2	MAC Address	13
2.8	Interoperability.....	14
2.8.1	Foundational, Structural and Semantic Interoperability.....	14
2.8.2	SNOMED-CT (Systemised Nomenclature of Medicine - Clinical Terms)	15
3	Methodology	15
3.1	Software Development Technique.....	15
3.1.1	Spiral Model	15
3.2	Database Design	15
3.3	Project Design Software Modelling.....	16
3.3.1	Unified Modelling Language (UML).....	16
3.3.2	Lucidchart	17
3.3.3	Three Diagram Categories of UML	17
3.4	Data Security Implementation Methods.....	17
3.4.1	SQL Injection (SQLi)	17
3.4.2	Prevention of SQLi using mysqli and PDO	17
3.5	Project Equipment.....	18
	Software	18
3.5.1	Notepad ++	18
3.5.2	Arduino Interactive Development Environment (IDE)	18
	Hardware	18
3.5.3	Arduino Uno	18
3.5.4	MySignals Hardware (HW) Complete Kit	19
3.5.5	Arduino W1500 Ethernet Shield.....	20
3.5.6	Laptop (Windows OS).....	20
4	Results and Analysis	21
4.1	Project Design Results and Analysis	21
4.1.1	Database Design	21
4.1.2	Database Design Analysis	23
4.1.3	Software Modelling	23
4.2	Project Design Implementation.....	29
	Design A – Fully Automated System.....	29
4.2.1	Connecting Sensors to MySignals HW Shield	29
4.2.2	Secure transfer of Sensor Data to MySQL Database	31
	Design B – Semi-Autonomous System	34
4.2.3	Formatting the Sensor Data into an XML File	34

4.2.4	Securely Inserting the XML File into the MySQL Database	34
4.2.5	Additional Sensors Tested	Error! Bookmark not defined.
4.2.6	Arduino Shield Stack Compatibility issues.....	36
4.2.7	Pseudo Data Storage	37
4.3	Comparison of Security Features	37
4.3.1	Encryption	37
4.3.2	Ethernet Compared with WiFi.....	38
5	Conclusion	39
6	Recommendations for Further Work	40
7	References.....	41
8	Appendices	45
8.1	Project Plans	45
8.1.1	Initial Project Plan.....	45
8.1.2	Updated Project Plan.....	45
8.2	Project Risk Assessment	46
8.3	MySQL PHP APIs	47
8.3.1	PHP MySQL Extension	47
8.3.2	PHP mysqli Extension	47
8.3.3	PHP Data Objects (PDO) extension	47
8.4	Operations Conducted by a Software Stack.....	47
8.5	Spiral Methodology	48
8.6	Database Design	48
8.6.1	OpenEMR Database Design.....	48
8.6.2	Database Design of “alternate_mysignals” Database.....	49
8.7	Software Modelling Diagrams in UML for System Design B	50
8.7.1	Use Case Diagram.....	50
8.7.2	Activity Diagram	51
8.7.3	Component Diagram	52
	Design A.....	52
8.8	Retrieving SP02 Sensor Data to Serial Monitor.....	52
8.8.1	Initialising Libraries and BLE module.....	52
8.8.2	Connecting and Initiating SP02 Sensor.....	53
8.8.3	Outputting SP02 and Pulse Values	55
8.8.4	Incorporation of SCTID code	56
8.9	Securely Transferring Sensor Data to MySQL.....	57
8.9.1	GET Request	57

8.9.2	POST Request	59
Design B.....		67
8.10	Formatting of Sensor Data into XML File	67
8.10.1	Initialising SD Libraries and Micro-SD Card	67
8.10.2	Creation of the XML File	67
8.11	Securely Inserting XML File into MySQL Database	68
8.11.1	PHP APIs	68
8.12	Arduino Cable error	72

1 Introduction

1.1 The Problem Being Addressed

Due to the rapid rate at which technology is advancing, it is important that the healthcare industry adapts and introduces innovative solutions, to solve the problems faced, as a result of, backdated network infrastructures in place. Electronic Health Record (EHR) databases systems, will be a key part of a future combined system, where both semantic and foundational interoperability is achieved throughout (see section 2.8). Although, NHS England has vowed to make patient records “largely paperless by 2020” [1] through the incorporation of EHRs, there is a lack of emphasis on creating a single system designed to incorporate EHRs, which are fed by data from portable sensors, where Healthcare Professionals (HPs) and Clinical Decision Support (CDS) systems can provide influential input.

Current personal use sensor devices are available, such as MySignals, but only allow for the personal use and management of sensor data. Without the protected transfer of data to a secure online database management system, where semantic and technical interoperability is achieved, the full utilisation of CDS systems (see section 2.1.1) cannot occur. This will be critical to the progression of Health IT in the future. It is therefore clear to see the viability of exploring this topic further and investigating a method of implementing this form of advanced technology.

1.2 Background Information

This project design aims to implement technology which will improve the fundamental delivery of healthcare. Current systems in place fail to take full advantage of the features contributed by EHR management systems. My design however, focuses on optimising the use of these EHR database management systems, through the effective integration of personal health sensor devices, and Arduino based equipment to ensure the highest level of functionality is achieved.

Given that the United Kingdom has an ever-increasing chronic disease mortality rate, the feasibility of this project can easily be justified. Implementation of such devices to patients suffering from diseases which make mobility difficult, could drastically improve the level of data received from the patients and thus increasing the level of collaboration between the patients and healthcare professionals. Substantial improvements in the quality of the healthcare delivered, would ultimately lead to a decrease in the number of unnecessary deaths.

Furthermore, due to restrictions in the ethical approval of my project, testing with actual sensor data was not permitted. Therefore, throughout the experiment phase, I focused on achieving a secure and

strong connection between the MySignals HW Shield and the sensor device. If this was achieved, in theory it could be said that attaining the sensor data would require a simple follow-up step to this.

1.3 Design Implementation

This project aims to incorporate the connection of portable sensor devices manufactured by the market leader, MySignals, which will be implemented to run coherently within a fully functioning system. Two different design implementations will be made, both offering unique advantages. Initially, a single design system was going to be implemented which utilised an Arduino Uno WiFi microcontroller to process all HTTP requests through. However, when first implementing this system within the School of Engineering laboratories, I was unable to establish a WiFi connection due the security protocol implemented by the University. Therefore, to resolve this issue an Ethernet connection was adapted, which allowed for complete testing of the system, thus justifying its implementation.

1.3.1 Design A

Design A will incorporate the following procedures. First the sensor will be connected to the MySignals HW shield, through the initiation of MySignals BLE module. Following this, pseudo data will be inserted, to replicate the data that would be acquired from the sensor device. A clinical terminology standard such as SNOMED-CT (Structured Nomenclature of MEDicine-Clinical Terms), will then be used to signify if the data is abnormal or normal (see section 2.8.2). This data will then be formatted and securely transferred using a database server, where an Ethernet Shield will provide the function of a client, in this client-server relationship. This design implementation can be said to be fully-autonomous, as the patient user is not required to take any action between the retrieval of the sensor data and the transfer of this data to the database.

1.3.2 Design B

Design B on the other hand, implements the use of securely storing pseudo sensor data into an XML file format, on to a local micro-SD card. The system will be designed to incorporate the functionality to connect the sensor devices to the MySignals HW Shield, as discussed in section 1.2. SNOMED-CT will then be incorporated using the same method. However, this pseudo sensor data is then stored in an XML file format and saved on to a micro-SD card. An Ethernet shield equipped with a micro-SD card slot will provide this functionality. This XML file will then be uploaded into a website which securely transfers the data to a database server. This system design can be described as semi-autonomous, as the procedure requires action to be taken by the patient user, in between the stage of sensor data retrieval and the stage of uploading this data to the database.

1.4 Project Time and Risk Management

1.4.1 Time Management

The initial planning phase began on the 2nd of October 2017, with the final project completion deadline being 15th March 2017. This deadline was however pushed forward two weeks, due to unforeseen circumstances. Implementation of an initial plan was critical to the execution of the project, where updated plans were then implemented to resolve delays within the project timeline (see Appendix 8.11 and 8.1.2). Research has proven the negative impact delays can have on the final result of a project, hence why it is important I implemented methods to alleviate these effects [2]. Having a well-defined plan was critical to ensuring the project, to the highest level, within the constricted time. Therefore, the aim consisted of completing all design and subsequent testing phases, with enough time to evaluate and analyse results achieved.

1.4.2 Risk Management

Carrying out an extensive risk management evaluation, before the initial planning phase was critical, where all risks were identified, monitored and evaluated according to ISO (International Organisation of Standardisation) standards [2]. This will be critical in constructing the risk matrix utilised as part of the project feasibility (see Appendix 8.2), where risks are categorised, evaluated, and rated according to likelihood of occurrence. Ultimately, ensuring all risks can be mitigated effectively.

1.5 Project Specification

This project will aim to produce secure and effective solutions, to challenges faced with the semantic and technical interoperable integration, of personal use sensors into an EHR database system. OpenEMR will be tested as the EHR database storage system, where a test database will be created to store all patient sensor data. Various design implementation methods will be tested, such as the utilisation of an Arduino Uno with a W1500 Ethernet Shield attachment. This has been changed from the initially proposed Arduino Uno WiFi board, to effectively ensure project requirements are fulfilled.

To ensure product quality remains high, it is of utmost importance to maintain control of the “development and maintenance processes” by collecting, examining and analysing both, process and product indicators. **Process indicators** convey the effectiveness of both software development and maintenance activities, whereas, **product indicators** provide a measure to the extent of quality attributes which are observable in a product.

Using a proposed foundational framework, I was able to create and utilise a software development process, which instilled product quality. The framework was inclusive of both technical and operational objectives. Technical objectives are representative of the needs and requirements of the project,

whereas, the operational objectives describe how technical objectives are realised and consequently implemented [3]. It will be important to fulfil these objectives to the highest standard and where an objective was not fulfilled, a detailed explanation will be given, into how this could be improved in the future.

Technical objectives can be detailed as the following:

- Securely connecting the sensor devices to the MySignals HW Shield on the Arduino Uno, to ultimately allow for sensor data to be acquired by a qualified patient user.
- Encrypting and decrypting the pseudo sensor data being transferred between the Arduino microcontroller and the EHR database.
- Securely transferring the pseudo sensor data that has been acquired, ensuring there are no open flaws within the system; this is critical from a security standpoint.

Operation objects include the following:

- Acknowledging and adhering by all standards, rules and regulations, ensuring the system is designed in an adaptive manner.
- System security must abide by all patient data confidentiality laws in place.
- System must ensure all sensor data is transferred and stored using technically and semantically interoperable methods, to allow for CDS systems to ultimately improve the quality of healthcare delivered.

1.6 Research Methodology

Through the reading and observation of findings within scientific journals and scholarly articles, it can be observed which areas research is and is not so prevalent. From this, the necessity for further research can be determined and thus further investigation can be justified. Additionally, previous C++/C coding language knowledge will be applied accordingly, to respective areas of this project.

Therefore, through implementation and analysis of the different proposed designs, well thought out conclusions can be drawn, to consequently improve the quality of healthcare delivered.

2 Literature Review

This literature review will combine theory and previous literature and research to analyse and discuss subjects to an extent that is deemed necessary. This is determined by the relevance these topics will have in carrying out the latter stages of this project.

2.1 Electronic Health Records

2.1.1 Overview

An EHR can be described simply as a digital version of a patient's traditional paper record, which can present data instantly and securely, in a coherent way, to all authorised users. They are built with the intention of making the sharing of information, between healthcare providers, simpler and more secure. Utilising such a function will make acquiring and storing sensor data, a much simpler process.

2.1.2 Functionality

Through research conducted within clinical practices, conclusive findings suggested more effective communication with other healthcare providers and patients was achieved, when implementing the use of EHRs. This led to improved clinical decision making. Thus, allowing healthcare providers to be more involved in the patient care, elevating the level of collaboration; hence, decreasing the likelihood of a simple, avoidable mistake being made. Additionally, EHRs provide a broader view to patient care through, by storing medical and treatment history in a single location. Subsidiary systems can also be implemented to provide further functionality to EHRs. **Clinical Decision Support** (CDS) systems are a good example of this, as they act to support healthcare professionals in making, patient specific decisions; ultimately, providing an all-round better healthcare service.

2.2 Database Management System (DBMS)

A database can be defined as "any collection of data, which has been organised for storage, accessibility and retrieval" [4]. Simple formatting and secure storage are two vital features that must be implemented into the DBMS. Both Relational and Non-relation DBMS solutions will be compared. A Relational Database Management System (RDBMS) uses structured tables to efficiently store data, allowing for easy access and data manipulation. Using foreign keys, tables of data can be linked together, allowing for the required data analysis to be implemented. **Structured Query Language** (SQL) is the standard Data Query Language (DQL) used for RDBMSs, where SQL statements can be used for both interactive and data gathering queries. Additionally, RDBMSs make creating, accessing and expanding databases simple, through administrative tools such as PhpMyAdmin. RDBMSs are applied to problems involving large amounts of the following: "complicated querying, database transactions and routine analysis of data" [5].

On the other hand, data in Non-RDBMSs are represented and structured in JavaScript Object Notation (JSON) documents, where any DQL can be used to manage the formatting; thus, the name “Not Only SQL” (NoSQL). JSON is a minimal, lightweight data-interchange format, used to primarily transmit data between a server and web application [6]. Non-RDBMSs are more tailored towards applications by larger corporations, as compared to the small-scale scope of my project. Therefore, an RDBMS will be utilised as part of this project.

DQLs are computer languages used to make query statements with information statements or databases. SQL is the standardised language used to control functions carried out by RDBMSs, hence it is the only DQL discussed below.

2.2.1 Structured Query Language (SQL)

SQL is used to communicate with databases and is the adopted standard for RDBMSs. Simply the language can be used to perform tasks such as: create a table within a database, update data within a database or retrieve data from a database. SQL statements can be classified into the following four categories [7]: Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL) and Transaction Control Language (TCL). However, as the database being created as part of this project will be created within MySQL Workbench and managed through PhpMyAdmin, knowledge of the SQL language is unnecessary.

2.2.2 SQL Database Management Options

SQL servers were originally designed for Microsoft Operating Systems (OS); hence functionality is optimised for these machines. This technology has now been developed for all major OS's and is not utilised by some of the following database servers such as: MariaDB, MySQL and Microsoft Azure SQL servers.

Microsoft Azure SQL is a RDBMS, cloud-based version of Microsoft's SQL server, with the broadest SQL Server engine compatibility. Microsoft produce various editions of Microsoft SQL server, where each is targeted towards different audiences, with ranging workloads. The Microsoft Azure Basic package is suited for small databases, providing support for one active operation at a given time. Performance, reliability and data protection can be maximised, using built-in intelligence that learns and adapts to application patterns. In-memory technologies allow Microsoft Azure SQL database to achieve performance improvements with workload such as, transactional – Online Transactional Processing (OLTP) [8]. OLTP is the only useful workload related towards the context of this project, hence will be discussed further. Being a profit-based organisation means only limited features, with limited usage are available for free, for a 30-day period, thus, limiting its feasibility as part of this project [8].

Online Transaction processing (OLTP) is the most predominantly used use case for RDBMSs, as the design of RDBMSs focus on data normalisation, to allow for extremely fast query processing speeds, whilst maintaining data integrity. OLTP is characterised by many short online transactions such as SLQ's DML statements.

MySQL is currently “the most popular open source SQL database management system”, providing the functionality to add, access and process data [9]. The database is structured as a RDBMS, following an American National Standards Institute (ANSI) – standard schema of structuring data in: tables, columns, views, triggers and cursors. This allows for a more organised layout to be implemented, from which file processing speed can be optimised. This is of great importance to the functionality of my healthcare database, due to the requirement of processing many small single on-line transactions [10].

MariaDB is an extremely popular open source database solution, which was developed as a fork to MySQL, after MySQL was sold to the Oracle Corporation. Therefore, MariaDB is also an RDBMS and uses the same database structure and indexes, allowing for the easy transfer of data between the pair. MariaDB offers a few unique features when compared to MySQL, such as improvements in: storage efficiency, power efficiency and CPU utilisation. Additionally, MariaDB is shaped by its users, allowing collaborative innovation to occur at much faster speeds. This has led to MySQL steadily falling behind MariaDB, in terms of product development [11]. However, due to MySQL's compatibility advantages with OpenEMR compared to MariaDB, it is viable to adopt MySQL, since the task of simply transferring sensor data will not be utilising MariaDB's key improvements on MySQL.

2.2.3 RDBMS Administrative Tools

Database administration is the important function of maintaining and managing a given DBMS software. Heavily used DBMS software's such as MySQL required ongoing management, to ensure functionality is at an optimum level. MySQL therefore requires an administrative tool which can pass statements written in SQL, carrying out various important management functionalities.

PhpMyAdmin is a free open source portable web application, and currently the most popular MySQL administrative tool. This web application offers the following features: a secure web interface, MySQL database management inclusive of managing tables, views, fields and indexes for the safe storage of data, as well as the ability to import and export data into various formats [12].

2.3 Server-side Programming

2.3.1 Internet Protocol (IP)

An **IP address** is simply a numerical label which is uniquely assigned to each device connected to a computer network, which is utilising the IP for communication purposes. The IP address was created

to serve the two following principle functions: “host or network interface identification and location addressing” [13]. This will serve as an integral part to establishing a connection between the Arduino Ethernet shield and the Apache web server (see section 2.5.2).

The **IPv4** is the fourth version of the Internet Protocol (IP) and one of the core protocols implemented to control the basic internetworking of the internet. IPv4 is the most commonly used to route internet traffic today [14]. The connectionless protocol operates to optimise, using a “best effort delivery model”. This addresses aspects such as data integrity, to ensure upper layer transport protocols are implemented, such as the Transmission Control Protocol (TCP). When utilising the given web server, the IPv4 protocol will be implemented to ensure that data is achieved using the most secure method.

2.3.2 HTTP

This application protocol sets a foundation of rules for the formatting and transfer of files, on the World Wide Web (WWW). Additionally, HTTP defines the actions carried out by web servers and browsers, in response to different commands. Being a stateless protocol allows commands to be executed independently, with no relation to prior commands. **HTTP verbs** instruct web servers on how to act, indicating the desired action to be performed on a given resource. Using HTTP requests-response protocols, such as GET and POST, I will be able to transfer pseudo sensor data through a web server to a given database server, such as MySQL [15].

2.3.3 Hypertext Pre-processor (PHP) Language

PHP is a widely used, open source, server-side scripting language, most commonly used to carry out web development tasks. PHP has the unique advantage of being easily embedded into HTML code and carrying out a task. The code is then executed on the server, generating and processing HTML requests, which are then sent to the client, thus establishing the client-server relationship. PHP establishes and connects with MySQL and a given web server, such as Apache (see section 2.5.2), using Application Programming Interfaces (APIs). The version of PHP being utilised as part of this project can be seen in figure 1.

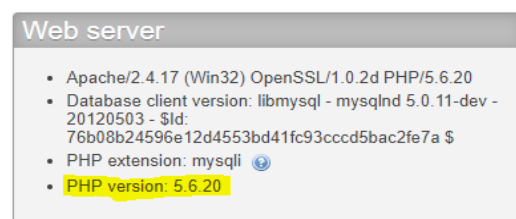


Figure 1: PhpMyAdmin web and database server details

APIs are essentially a contract provided by one software to another, where a structured request and response is recorded. Desired tasks can be carried out using APIs [16]. PHP extensions incorporate the necessary APIs required to communicate with the MySQL database server and will be discussed further below.

2.3.4 Database Connectors and Drivers

MySQL documentation simply refers to **connector** as the line of code which when implemented, allows an application to connect to the MySQL database server. PHP code will therefore allow a developer to “connect to the database server, query with the database and carry out other database-related functions”. Software generally known as the connector, is required to provide the API the PHP application will use, whilst handling communicational interactions between the application and the database server [16]. It is vital a secure connection method is adopted here, to ensure security related objectives are fulfilled.

A **driver** may call libraries such as, the MySQL Client Library or the MySQL Native Driver, which implement software designed to communicate with specific types of database servers. Additionally, libraries such as these “implement the low-level protocols required to communicate with the MySQL database server”. Also, in MySQL documentation, the term “driver” defines the part of the software “that provides the database-specific part of the connector package” [16].

2.3.5 PHP Extensions

PHP code is structured of a core, plus optional extensions to these core functionalities. On occasion, a PHP extension can be used to expose an API to the PHP developer, which then allows its functionalities to be used within the software. The PHP Data Objects (PDO) MySQL driver extension instead “provides an interface to the PDO layer above it”.

The three most popular **MySQL PHP APIs** used to connect to a MySQL database server, are as follows: PHP’s MySQL extension, PHP’s mysqli extension and PDO. Further details of these extensions can be found in appendix 8.3.

All three PHP MySQL extensions use a low-level library that implements the required protocol to allow them to function effectively. PHP’s **MySQL native driver, mysqlnd**, was developed as an alternative to outdated libraries such as MySQL Client Library. Therefore, the PHP system can fully utilise the design of mysqlnd, which has various speed and memory enhancements. Additionally, with this native driver now being part of the standard distribution, MySQL installation is not required on the developer’s machine, to build or to run PHP database applications. Various advantageous gains in efficiency are fulfilled, due to the driver being compiled as a PHP extension, which utilises the PHP memory management system [17]. The version of mysqlnd being utilised as part of this project can be seen in figure 1.

2.4 EHR Management Package

A study conducted in 2014, involving numerous open source EHR packages, such as: FreeMED, Hospital

1. Health information and data.
2. Result management (e.g. images, clinical dashboard, alerts).
3. Order entry and management (e.g. computerised provider order entry).
4. Decision support (e.g. drug interactions and prevention).
5. Electronic communication and connectivity (e.g. email and integrated records).
6. Patient support (e.g. patient education support).
7. Administration processes (e.g. patient scheduling billing).
8. Reporting and population health management (e.g. quality indicator, national registries).

Figure 2: Eight widely adopted functionalities

Software	Learnability	Efficiency	Satisfaction
FreeMED	4	3	3
GNUmed	3	3	4
GNU Health	4	3	3
Hospital OS	2	1	1 ^a
HOSxP	1 ^a	1 ^a	1 ^a
OpenEMR	4	4	4
OpenMRS	3	4	4
OSCAR	3	3	3
THIRRA	3	3	3
WorldVista	3	3	3
ZEPRS	3	2	2
ClearHealth	3	2	2
MedinTux	1 ^a	1 ^a	1 ^a

^a Non-English interface.

Table 1: Comparison of learnability, efficiency and

Software	1	2	3	4	5	6	7	8
FreeMED	✓	x	✓	x	✓	x	✓	✓
GNUmed	✓	✓	x	✓	✓	x	✓	✓
GNU Health	✓	x	✓	x	x	x	✓	✓
Hospital OS	✓	x	✓	x	x	x	✓	✓
HOSxP	✓	✓	✓	✓	x	x	✓	✓
OpenEMR	✓	x	x	✓	✓	✓	✓	✓
OpenMRS	✓	x	✓	x	✓	x	✓	✓
OSCAR	✓	x	✓	✓	✓	x	✓	✓
THIRRA	✓	✓	✓	✓	x	x	✓	✓
WorldVista	✓	✓	✓	✓	x	x	✓	✓
ZEPRS	✓	x	x	✓	x	x	x	✓
ClearHealth	✓	x	x	✓	x	x	✓	✓
MedinTux	✓	✓	✓	x	✓	x	✓	✓

Table 2: Comparison of EHR systems with the widely adopted functionalities

OS and OpenMRS. These were tested with the goal of finding the most suitable EHR management solutions. Test subjects of varying abilities were tested, where packages were ranked on a scale of 1-5 in the following categories: learnability, efficiency and satisfaction. It was conclusively proven that OpenEMR was the most effective EHR software package, in all the given categories, through the following scores (see table 1) [18]. Various risks can therefore be mitigated, as a result of using an EHR package with a lower learnability risk. Below are a set of eight core widely adopted functionalities (see figure 2), that have been compared with the other open source EHR packages as shown in table 2.

It can therefore be concluded that, OpenEMR again shows its wide range of functionality, hence why it is said to be “most popular open source EHRs and medical practice management solution” [19]. Thus, only OpenEMR will be discussed further, and the different software’s it incorporates.

2.4.1 OpenEMR

OpenEMR is an Office of the National Coordinator (ONC) and Open Source Initiative (OSI), completely certified EHR management systems solution. Having successfully received the ONC certification, allows the EHR management system to store data in a structured format allowing healthcare providers to easily retrieve and transfer patient data, in a way that can contribute to improving patient care. The cross-platform software utilises the MySQL database servers, to control and manage EHRs, through

the effective use of superior features, beyond the scope of this project [19]. My EHR management system and database design, will be designed around the features incorporated with OpenEMR.

2.5 Software Stack Components

“**Stacks**” are effectively bundles of software that are required to access and send data to a database, including everything from “operating systems and web servers to APIs and programming frameworks” [20]. Bundling the software’s together, allows for an easier download and deploying of the components at once, where compatibility is provided by a layer from each component in bundle. Components can either be generalised or targeted towards very specific purposes [20].

The Activity diagram depicted in section 4.1.3.3, effectively explains the operations conducted by a stack. Additionally, refer to Appendix 8.4 for a diagram depicting greater detail [20]. Software stacks are compiled of **four tiers**, of which the later three are server-side. The four tiers are as follows:

- **The client tier** – representative of the only component in the browser.
- **The web tier** – web server, handles HTTP requests.
- **The business tier** – application server, inclusive of the server-side programming languages, such as PHP.
- **The database tier** – chosen database server, in the case of this project, MySQL.

Due to OpenEMR and its adoption of XAMPP, it was viable for this project to utilise XAMPP as the designated software stack package.

2.5.1 XAMPP

XAMPP can be broken down in to the following: Cross-platform (**X**), **A**pache, **M**ySQL, **P**HP and **P**erl, where XAMPP is a variation of LAMP (**L**inux, **A**pache, **M**ySQL, **P**HP), but can be used in more operating systems; thus, providing the developer with greater flexibility, if required. The following features are consistent in all LAMP variations: “flexible, customisable, easy to develop and deploy and secure”. Additionally, this product is open-source and is therefore inclusive of a large supportive community of developers [20]. The XAMPP control panel is shown in figure 3, allowing users to start and stop the web server (Apache) and the database server (MySQL).

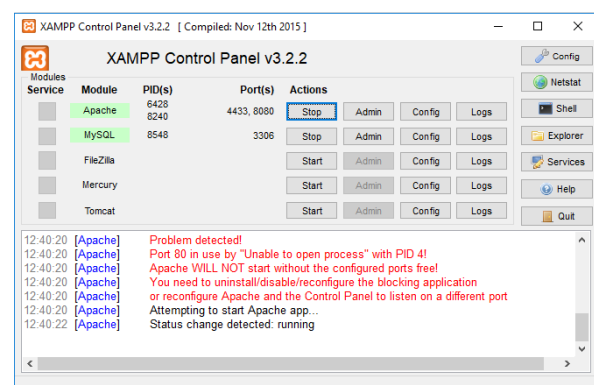


Figure 3: XAMPP control panel

2.5.2 Apache

Apache is a fast, reliable and secure web server application used as part of XAMPP, which runs 67% of all web servers in the world. Apache provides a secure and efficient web server, which maintains in sync with current HTTP standards [21]. Simply, a web server stores software and website component files, such as HTML documents. Through its connection to the internet, physical data interchange with other devices connected to the web can be supported. PHP was discussed above in section 2.3.3.

2.6 Data Security

Due to the confidential nature of the data being transferred and stored within my system, I must ensure I abide by all **laws** and satisfy all **regulations** in place, to ensure patient data is kept safe and secure. Man-In-The-Middle (MITM) and SQL injection (SQLi) attacks are becoming more prevalent and harder to prevent, with advanced technology being readily available to cyber criminals. Data security features are applied to prevent the unauthorised access to computers, databases and websites, whilst also preventing the corruption of data. This therefore validates the need for both a secure database connection protocol and a high-level encryption standard implementation. Various laws and regulations must also be abided by, such as the Data Protection Act and the Human Rights Act, ensuring health records are kept confidential. This is due to the importance of secure and meaningful sharing of health information, between healthcare professionals and other healthcare systems [23].

2.6.1 Secure Database Connection

It is fundamental to the security of the data within the database, that a secure connection be achieved before concentrating on further details. mysqli and PDO are the desired choice given there advanced and updated functionalities, allowing for security related improvements to be achieved. This is in comparison to mysql which is outdated and is now being deprecated from newer versions of PHP. This is shown in figure 4 where an error message has occurred due this depreciation. It would be beneficial for the user in the foreseeable future, if a standardised PHP extension was utilised [24]. This is discussed further in section 3.4

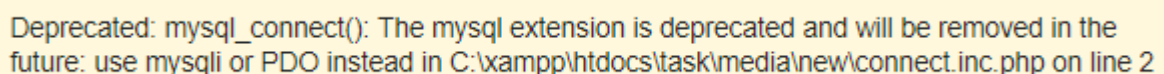


Figure 4: mysql_connect depreciation error

2.6.2 Encryption

Encryption is defined as the operation of converting data from a plaintext format, to an encoded version which can only be decoded by a stakeholder with access to a decryption key. There are various encryption standards widely adopted, but due to restrictions in the Arduino Uno's power and memory, higher demanding encryption standards such as SSL (Secure Socket Layers) cannot be adopted. Block-

ciphers such as AES and XTEA, offer forms of encryption which can be adopted by an Arduino Uno microcontroller, through extended downloadable zip libraries.

2.6.3 Data Protection Act (1998) and Human Rights Act (1998)

The Data Protection Act was implemented to fulfil these requirements and ensure data is kept safe and secure. When abiding by this act, everyone responsible in the handling of data must follow strict rules in place called “data protection principles”. This insures that the data is: “used fairly and lawfully; used for limited, specifically stated purposes, to ultimately improve a given patients healthcare; used in a way that is adequate, relevant and not excessive; accurate; kept for no longer than is necessary; handled according to people’s data protection rights and kept safe and secure” [25]. Furthermore, the Human Rights Act is fundamental in the protection of all patients interacting with my system. It is important my system doesn’t take away a given patients fundamental rights [26].

2.7 Internet of Things (IoT)

It is predicted by 2020 there will be over 50 million interconnected devices, which will profoundly affect the way businesses and societies interact [27]. This project aims to incorporate IoT aspects through the effective and secure networking of medical sensors, into a system utilising a DBMS to the most optimum level. Therefore, researchers have concluded “things” incorporated within an IoT structure, should refer to an “inextricable mixture of hardware, software, data and services” [28].

2.7.1 Bluetooth Low Energy (BLE) Technology

BLE technology was adapted from Bluetooth technology by Nokia, to provide **low energy** and effective **cost usage**, whilst maintaining similar characteristics to “standard” Bluetooth technology. This allows BLE devices to transmit less data, over a shorter distance, consuming less power than “standard” Bluetooth, in the process.

A BLE device, such as the SP02 BLE sensor (a part of the MySignals HW kit), is incorporated with a BLE beacon. This is a small battery or USB powered device, which emits a Bluetooth Low Energy signal. Each BLE device has a unique Media Access Control (MAC) address, which can be found using a smartphone application such as nRF Connect [29].

2.7.2 MAC Address

MAC addresses uniquely identify devices which have been assigned to Network Interface Controllers (NICs), by the manufacturer. NICs simply communicate within the data link layer (second layer) of the Open Systems Interconnection model, to allow for a connection between a computer device and a computer network to be achieved [30]. MAC addresses are also used as network addresses for the majority of IEEE 802 network technologies, such as Ethernet and WiFi. IEEE 802 is part of a family of

IEEE (Institute of Electrical and Electronic Engineers) standards, which focuses on Local Area Networks (LANs) and Metropolitan Area Networks (MANs) [31]. Declaring MAC Addresses for the Ethernet shield and the BLE sensor devices will be vital to the process of establishing a secure connection.

2.8 Interoperability

Interoperability describes the “extent to which systems can exchange data and interpret the shared data” as defined by HIMSS (Healthcare Information and Management Systems Society). For interoperability to be achieved between two systems, data must be both exchanged and subsequently, understood by a user. Healthcare interoperability focuses on the “ability of different information technology systems and software applications to communicate, exchange data” and consequently utilise the information that has been exchanged. The ultimate goal is to create a system which leads to advancements in the delivery of healthcare. Health information technology interoperability can be defined as the three levels below, in order of increasing understanding being achieved between the systems [32].

2.8.1 Foundational, Structural and Semantic Interoperability

Foundational interoperability focuses on the data exchange from one information technology system to another, but does not necessitate that the receiving information system should be able to interpret the data [33]. This will be achieved through the validation and processing of HTTP requests, using an Arduino Ethernet shield to provide this functionality.

Structural interoperability is an intermediate level between foundational and semantic interoperability. This form of interoperability defines the exchange of data with the preservation of structure and format, using message format standards such as mysqli or PDO. Achieving this level of interoperability ensures “data exchanges between systems can be interoperated at a data field level”.

Semantic interoperability provides the highest level of interoperability achievable between two systems, where both the ability for two systems or more to exchange information and to utilise the information that has been exchanged, is achieved [32]. This relies on both, the “structuring of data exchange and the codification of the data”, through vocabulary so the receiving system can interpret and understand the data. Achieving this level of interoperability supports the electronic exchange of patient medical sensor data among healthcare professionals, through EHR systems, which allows for improved: “quality, safety, efficiency and efficacy of healthcare delivery”. To achieve semantic interoperability, I will be utilising SNOMED-CT and its provided terminology to incorporate codification into my data.

2.8.2 SNOMED-CT (Systemised Nomenclature of Medicine - Clinical Terms)

SNOMED-CT is a “comprehensive, multi-lingual and global clinical healthcare terminology”, which is used to encode patient sensor data in a standardised way, allowing semantic interoperability to be achieved with EHRs [34]. There are several benefits in favour of the adoption of SNOMED-CT, namely the fact that it is adopted by NHS England. Also, becoming acquainted with the nomenclature and the SCTID codes is a very simple in comparison to other vocabularies such as HL7, thus eliminating various risks. This simplicity factor is incorporated through the SNOMED CT Browser, which makes it easy for an SCTID code to be located and utilised in as necessary [35]. SCTIDs are representative of SNOMED-CT components. Additionally, SNOMED-CT is continuously evolving, with new versions being released every six months, where new terms and relationships are incorporated.

3 Methodology

3.1 Software Development Technique

As discussed in the project feasibility report, an **agile approach** to software development was implemented, throughout the course of carrying out this project. This is due to the large scale of my project and the positive results shown towards such an approach, from previous research conducted on the topic. This technique focuses on a value-driven approach, which allows developers to deliver a high-quality level of work. This involves the advocacy of the following properties: “adaptive planning, evolutionary development, early delivery and continuous iterations of improvement” which ultimately results in the project having the ability to rapidly adjust to changes that may occur over the projects plan [36].

3.1.1 Spiral Model

The spiral methodology adopts an approach where 4 phases are iterated in the following ongoing process: “determining objectives, evaluating risks, developing and testing of the next prototype and planning the next iteration” [37]. A visual representation of this diagram is shown in Appendix 8.5. This method was designed and adapted for large software project, where large scale planning and risk management is essential. This would therefore allow me to complete the project with a greater level of flexibility, whilst eliminating and avoiding risks as necessary. Additionally, this method leaves the final product highly customisable, which will allow me to edit the software as required, to find the best possible solution [38].

3.2 Database Design

To create a database design the **MySQL Workbench** software was used, as this is a MySQL developed software, hence will incorporate all the necessary tools required to create a sufficient database design.

The software also provided the following **functionalities**: “data modelling, SQL development and comprehensive administration tools for server configuration, user administration, backup, plus much more” [39]. The tool automates numerous error prone and time-consuming tasks, improving the eventual database design. This therefore allows for the resolution of issues with the design before major investment of time and resources have been made. MySQL Workbench also enables **model-driven** database design, which is proven to be the most efficient methodology for creating all round, well-functioning databases. There is also the incorporation of a degree of flexibility to allow for evolving project requirements.

Furthermore, “model and schema validation utilities enforce the best practice standards and MySQL-specific physical design standards” for the effective generation of efficiently functioning MySQL databases. MySQL Workbench incorporates numerous techniques such as: forward and reverse engineering, change management and database documentation [40]. The model-driven architecture (MDA) database design will be documented in the results section 4.1.1.

3.3 Project Design Software Modelling

Modelling of a software development project is a vital step in the design procedure before the implementation of any code. Modelling languages allow for the blueprints of a project to be visually represented, which can be an effective tool in ensuring a project’s success. It also allows developers to ensure all project specifications have been accounted for before the implementation of code makes this difficult. Various surveys have proven that a larger proportion of software projects fail, in comparison to the amount that are successful. Therefore, it is important to mitigate the risk of failure, and increase the odds of success, through the implementation of an effective software modelling language.

3.3.1 Unified Modelling Language (UML)

The Unified Modelling Language (UML) is an approved ISO (International Organisation of Standardisation) standard and is one of the most widely adopted modelling language, where it is used by over 10 million developers [41]. UML allows developers to “specify, visualise and document modules of software systems, including their structure and design, in a way that meets all of these requirements” [42]. The language is built upon the fundamental object-oriented programming concepts but can alternatively be used to model non-object-oriented languages. Version 2.0 of UML is the most popular, with readily available online resources. There are numerous UML-based tools available on the market, where Lucidchart will be the online tool I adopt as part of this project.

3.3.2 Lucidchart

Lucidchart, a UML diagramming tool, will be used to create all diagrams, in a clear and industry standardised manner. The web-based user interface also incorporates various other diagram types and is supported by web standards such as HTML5 and JavaScript. Additionally, the online tool is supported by Google Chrome, the web browser I will be using to tackle technical aspects of this project. Utilising Lucidchart's free 30-day trial will allow me to create all necessary UML diagrams, thus justifying its viability to be used as part of this project.

3.3.3 Three Diagram Categories of UML

There are thirteen types of diagrams exclusive to UML, which can be divided into the following three categories: **structural**, **behavioural** and **interactional** diagrams. Each diagram has an independent purpose.

Behavioural diagrams can be represented by the following three diagrams: Use Case diagram, Activity diagram and State Machine Diagram. I will be utilising both the **Use Case** and **Activity diagrams**, to effectively model behavioural aspects of my project. All diagrams will be depicted in the following section 4.1.3, along with explanations of what is occurring the diagrams themselves. Given the dynamic representation provided by the Activity diagram, interactional diagrams will not be utilised.

Structural diagrams can be modelled by diagrams such as: class diagrams, object diagrams and **component diagrams**, where the later will be utilised for this project.

3.4 Data Security Implementation Methods

3.4.1 SQL Injection (SQLi)

Due to high level of severity this attack could have on a given database, it is important to implement methods of preventing or limiting the severity of damages. SQL Injection (SQLi) refers to an injection attack, where an attacker executes malicious SQL statements with the intent to manipulate data stored within a MySQL database [22]. Given the right circumstances, attackers can bypass authentication and authorisation protocols, and retrieve contents of entire databases or even impersonate users. To achieve such authorisation, an attacker

can, insert a payload which will be

included as part of the SQL query (see

```
1. username = request.POST['username']  
2. password = request.POST['password']
```

figure 5) and ran this against the database server.

Figure 3: SQLi attack example

3.4.2 Prevention of SQLi using mysqli and PDO

Both mysqli and PDO are suitable solutions to the security issues that must be solved, thus I will be utilising both extensions and discussing the features in terms of security in the analysis section of this

report. It should be noted that the PDO extension can be used on 12 different database drivers, whereas mysqli is limited to only mysql. Additionally, both extensions implement SQL injection security, preventing a hacker from injecting malicious GET or POST requests. The best solution to implement would be a prepared statement, which is detailed in the results section for mysqli (section 4.2.2.1) and PDO (section 4.2.4.1) respectively [43].

3.5 Project Equipment

In the carrying out of this project, both software and hardware components were used. The literature review provided details on software related to server-side programming, hence for details on XAMPP, refer to section 2.5.1. Additionally, software's implemented for the design process have already been discussed, such as Lucidchart (section 3.3.2) and MySQL Workbench (section 3.2).

Software

3.5.1 Notepad ++

This software enables software to be compiled in a very easy to see layout and colour scheme. This makes debugging the code significantly easier, as the colour scheme will change if there is a syntax issue. With the support of 84 languages, including PHP, HTML and jQuery, it is easy to understand how the source code highlighter has been widely adopted, winning various prizes as a result of this popularity. Thus, proving its ability to be easily adopted as part of this project.

3.5.2 Arduino Interactive Development Environment (IDE)

To support the Arduino microcontroller, the Arduino IDE was used to compile, store, verify and upload software to the Arduino microcontroller board. This compiler also allowed for the necessary libraries to be stored and then recalled for a given purpose. This was effective in the programming of the MySignals HW shield, where certain libraries were required to call functions, to program certain components on the shield effectively.

Hardware

This section provides details on the hardware components used throughout this project.

3.5.3 Arduino Uno

The Arduino Uno is a microcontroller board with an ATmega328P System on Chip (SoC), which acts as a Microcontroller Unit (MCU). The ATmega328P chip (figure 5) is a high-performance 8-bit microchip, which combines 32KB in-system Programming (ISP) flash memory, with read and write capabilities. There are 14 digital input/output pins, of which 6 are Pulse Width Modulation (PWM) outputs and 6 are analog inputs. The figure 5 below labels all major components of the Arduino Uno, which contains everything required to adequately support the microcontroller.

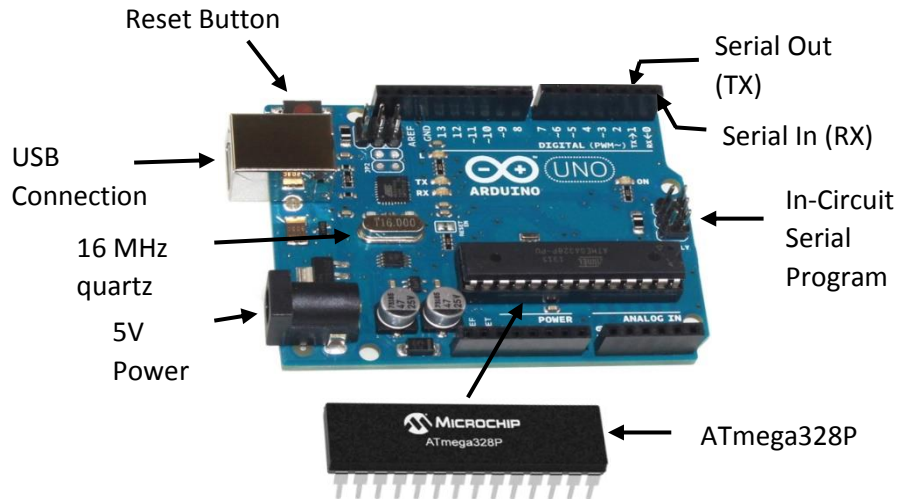


Figure 5: Arduino Uno labelled

3.5.4 MySignals Hardware (HW) Complete Kit

Medical aspects of this project were carried out using the MySignals Hardware (HW) complete kit (figure 6). MySignals is simply a “development platform for medical devices and eHealth applications”. The kit is inclusive of 17 sensors, some of which utilise BLE technology (see section 2.7.1) and a MySignals HW Arduino shield. The sensors can measure over 20 biometric parameters, where the data can then be processed by the Arduino Uno microcontroller.



Figure 6: MySignals Equipment Kit

3.5.4.1 MySignals HW Arduino Shield

Firmware for the MySignals board can be found online, which includes “high level library functions” allowing for the easy management of the 16 sensor devices. Due to flash memory limited to 32KB in the Arduino Uno, all features cannot be used at the same

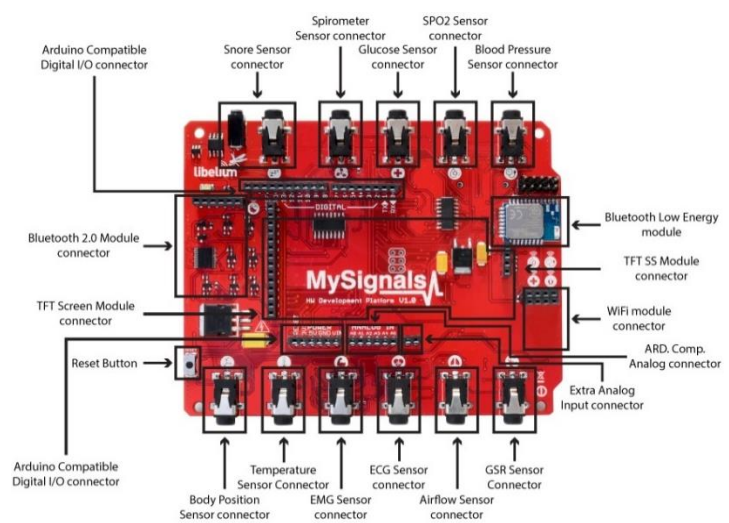


Figure 7: MySignals HW Shield

time. Successful connection between the shield and BLE sensors, is achieved through the in-built BLE module on the shield.

3.5.5 Arduino W1500 Ethernet Shield

This Arduino Uno R3 shield is an alternative to the Arduino Ethernet shield R3 [44], which is now a retired product. Therefore, given the scope of this project it was viable to adopt this product, which is inclusive of the following features: supports the latest version of Ethernet capabilities and Arduino's official Ethernet libraries, whilst integrating a W5100 controller and a micro-SD card slot. Additionally, the shield supports both TCP/IP, hence streams of information can be checked for errors before being distributed over the network. Finally, the shield is stack capable, hence the MySignals shield can be connected above this.

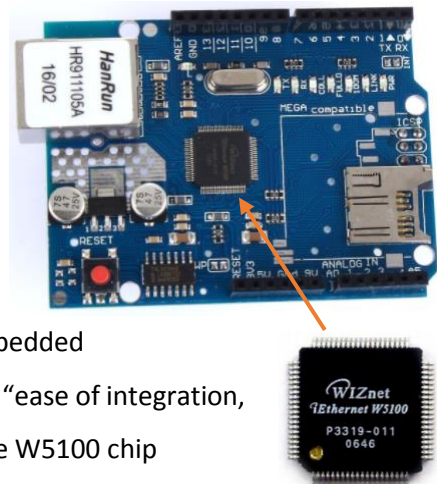


Figure 8: Arduino W1500 Ethernet Shield

The **W1500 Ethernet shield** is integrated with a fully-featured and internet enabled single-chip, designed to be used in embedded applications where the following requirements are necessary: “ease of integration, stability, performance, area and system cost control” [45]. The W5100 chip can also implement internet connectivity without the use of an OS, drastically reducing the required programming needs.

The hardwired TCP/IP stack also supports “TCP, UDP and IPv4”, which is necessary for the given application of using the shield as a client and as a micro-SD card port.

Both a **WiFi router** with internet connectivity and an **Ethernet cable** were required to fully utilise the Ethernet shields capabilities, to ultimately process the HTTP requests.

All the equipment stacked together, is depicted in the following section.

3.5.6 Laptop (Windows OS)

The laptop I used as part of this project was a Windows 10 build, with the following very basic specifications:

- Processor: Intel® Atom™ x5-Z8300 CPU @ 1.44GHz 1.44GHz
- Random Access Memory: 2.00GB.

These low-range specification increases the systems adoptability.

4 Results and Analysis

This section will document all result achieved throughout the course of this project, and concisely analyse these results.

4.1 Project Design Results and Analysis

4.1.1 Database Design

The database design depicted in figure 9 below was constructed in **MySQL Workbench**. Initially the MySQL Server was connected to the MySQL Workbench software, using an in-built tool within workbench. Following this, an SQL editor window was opened, in which all SQL functions can be applied to MySQL databases. The Data Modelling windows was ultimately opened, in which all database design procedures take place. Hence, this is where the model-driven database designs were constructed.

Initially, the OpenEMR database design was reverse engineered to understand and comprehend the high-level design features incorporated in such a database. This complex database design can be found in appendix 8.6.1. Given the scope of my project, it was not necessary to incorporate all these features, thus why I adopted the simplified database design shown in figure 9. This database was

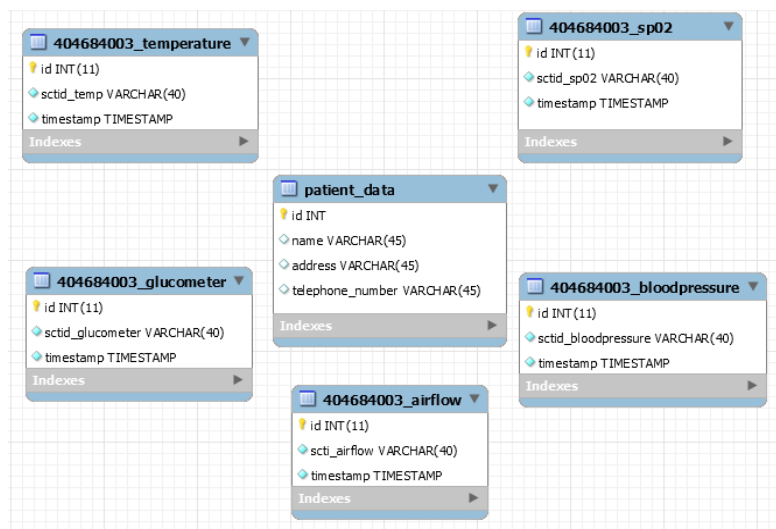


Figure 9: Design A - Model-driven database design

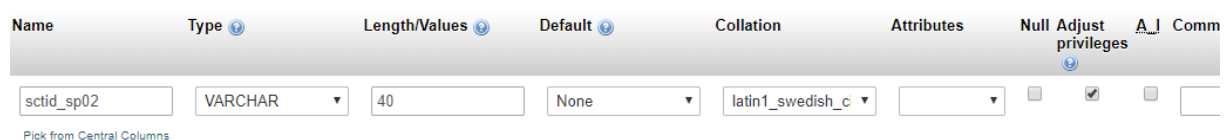
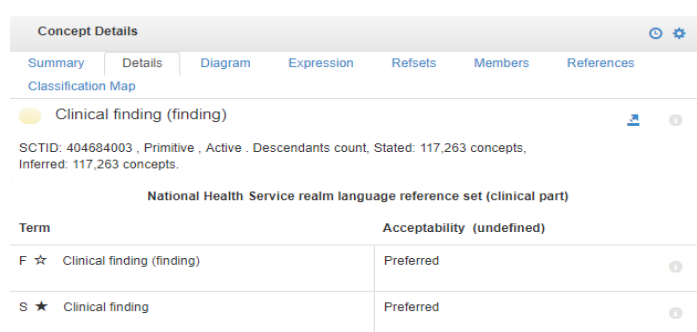


Figure 10: Column editing tool in PhpMyAdmin

named “**mysignals**”, thus will be referred to as this from here on. It should also be noted, table names and column names are case sensitive, hence why I decided to keep everything in lowercases. It was important to include a table storing all vital data necessary to a given healthcare professional. This can be seen in the middle of the database design, where the table named “patient_details” is constructed including: a name, address, and telephone number. Each container within the database design

represents a table implemented inside the database. The “Types” of data are declared and can be seen to the right of the “Name”, where figure 10 shows how this is declared in PhpMyAdmin. The majority of “Types” can be classified as one of the following: integer (**INT**), variable character (**VARCHAR**), text (**TEXT**) and timestamp (**TIMESTAMP**). Within each table, an id integer is declared to hold a positional value of the data value being stored. This is set to autoincrement and is the primary key of each table, thus will be present in the first column of the table and will automatically increase with each data value inserted. This will allow authorised users or systems easy access to a particular sensor data value.

Tables storing sensor data are positioned around the “patient_details” table. The method adopted to store the pseudo sensor data uses figure 11, where the SCTID code defined for a **clinical finding** is implemented into the title of the given table. The SCTID code is shown as “4046844003”, thus all the table names



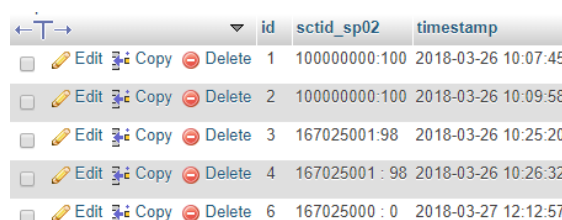
The screenshot shows the 'Concept Details' page for 'Clinical finding (finding)' in the SNOMED-CT web browser. It includes tabs for Summary, Details, Diagram, Expression, Refsets, Members, and References. The SCTID is 4046844003. Below, a table shows the term 'Clinical finding (finding)' with a preferred status.

Term	Acceptability (undefined)
F ☆ Clinical finding (finding)	Preferred
S ☆ Clinical finding	Preferred

Figure 11: SCTID for a clinical finding

begin with this integer value, followed by the name of the sensor being used [35]. All SCTID codes can be found using the SNOMED-CT web browser, as shown in figure 11. The column storing the data implements the VARCHAR type, as both numbers and special characters, in colons, will be utilised. Refer to table 3 for further details. Furthermore, table 3 shows the structure of the spO2 sensor data storage table in PhpMyAdmin, located within the mysignals database. Table management tools are also depicted, as well as pseudo data storage in the second column. The database administration tool, PhpMyAdmin is accessed using the following URL prompt, <http://localhost:8080/phpmyadmin/>. Localhost represent the generic IP of the web server active on your system (127.0.0.1). It is often better to use the IPv4 address to access the given web and database servers.

Another viable option is to store the sensor data and the SCTID codes in two separate columns, as this will allow authorised individuals or systems easier access to the values required. This database will be named “**alternate_mysignals**” where the table structure is shown in table 4. This table design will be referred to as the **alternate_mysignals design**. The alternative database design is depicted in Appendix 8.6.2.



The screenshot shows a table with columns: id, sctid_spO2, and timestamp. It contains five rows of data, each with an edit, copy, and delete icon.

	id	sctid_spO2	timestamp
<input type="checkbox"/> Edit Copy Delete	1	100000000:100	2018-03-26 10:07:45
<input type="checkbox"/> Edit Copy Delete	2	100000000:100	2018-03-26 10:09:58
<input type="checkbox"/> Edit Copy Delete	3	167025001:98	2018-03-26 10:25:20
<input type="checkbox"/> Edit Copy Delete	4	167025001 : 98	2018-03-26 10:26:32
<input type="checkbox"/> Edit Copy Delete	6	167025000 : 0	2018-03-27 12:12:57

Table 3: Single column table for sensor data storage



The screenshot shows a table with columns: id, sctid, spO2, and timestamp. It contains one row of data, each with an edit, copy, and delete icon.

	id	sctid	spO2	timestamp
<input type="checkbox"/> Edit Copy Delete	1	167025001	98	2018-03-28 01:57:00

Table 4: Alternate table design

Both the, mysignals and alternate_mysignals database designs showcase simplistic features, where either design could be adopted depending on the secondary systems requirements.

4.1.2 Database Design Analysis

Given the scope of this project, the role fulfilled by the database is to simply store the data being retrieved from the sensor devices. Therefore, only the semantically interoperable SCTID codes and the data values need to be inserted into the database. Also, it is beneficial to make the design as simple as possible, ensuring the data can be easily passed on to further authorised systems or individuals. It could therefore be said that storing the sensor data and SCTID codes in separate columns would better the delivery in this aspect; thus the alternate_mysignals design as shown in appendix 8.6.2. The **security of the database** is outside of the scope of this project however if necessary, a password can be added. Additionally, there is the option to encrypt the data inside the database, but this would make the simple task of viewing the data difficult, as a decryption protocol will have to be implemented each time the user attempts to view the database tables. Moreover, it can be argued that limitations in the Arduino Uno's memory would be benefit from the use of the single column database design (mysignals database), as only a single POST request is processed.

4.1.3 Software Modelling

As discussed in the design section, UML diagrams will be utilised as the modelling language for the software development aspects of this project. All system Design A diagrams will be depicted in this section with in-depth analysis of the various incorporated features. System Design B diagrams can however be found in Appendix 8.7.

4.1.3.1 Use Case Diagram

The purpose of the Use Case diagram is to demonstrate the various ways, users (actors) may interact with a system. An effective Use Case diagram would **represent** the following:

- **Situations** the given system interacts with any of the following: people, organisations and external systems. These entities are referred to as actors.
- **Goals** that the system will help the actors achieve.
- The overall **scope** of the system.

Therefore, this allows developers to provide a high-level overview of the relationships that exist between “use cases, actors and systems” [46]. A Use Case diagram is compiled of systems, actors, use cases and relationships.

In the case of this project, **system** is singular as each use case diagram will be representing a single design. The system's boundaries are represented by the rectangle which helps to define the scope of the system, where anything inside of this is carried out by the project system.

Actors are represented through the stick figures outside of the system. They are either someone or something that uses the system to achieve a goal, such as a person, organisation, another system or an external device. See figure 12 for Design A and Appendix 8.7.1 for Design B. Actors can be either primary or secondary. Primary actors initiate the use of the system and secondary actors react to actions taken by the primary actors. It is noticeable that primary actors are to the left of the system and secondary actors are to the right. As shown in figure 12, the patient user is the former and the CDS systems acts as the later.

A **use case** represents an action that accomplishes a given task within the system, thus describing the fundamentals of the system. Use cases are depicted through the oval shapes organised within the rectangle, as the actions occur within the system.

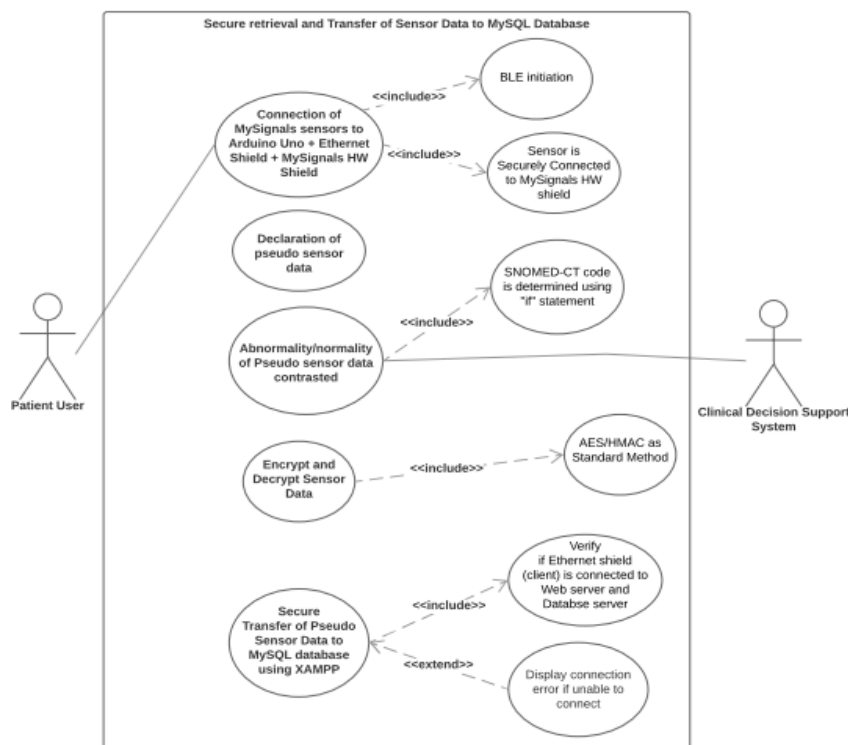


Figure 12: System Design A Use Case Diagram

Relationships within use case diagrams can occur in the following forms: **association**, **include** and **extend**. An association relationship is symbolised through a black line to signify a basic communication or interaction between actors and use cases. Include and extend relationships are represented through the dashed lines, where an include relationship shows dependency between a “base use case” and an “included use case”. An extend relationship on the other hand, will only occur if certain criteria is met,

showing dependency between a “base use case” and an “extend use case”. A noticeable difference is the direction in which the arrows point, which provides visual reinforcement of the relationship being displayed.

Figure 12 displays the use case diagram for **Design A**. Each use case is defined on the premise of the definition provided above. Thus, the uses cases below are set in order to accomplish objectives set by the system. Additionally, include and extend relationships are incorporated to add functionality to the additional actions carried out by the system. For example, the first use case focuses on the establishment of a secure connection between the sensor device and the MySignals HW Shield. In order for this to occur, BLE initiation of the MySignals HW shield must take place, allowing for the BLE sensor to be detected, consequently connected to the Arduino platform. Without this action, a connection cannot be established, thus why it is depicted as an include relationship. The following use case below, focuses on the declaring an pseudo sensor data value, which is then utilised with another function determining whether the sensor data is of an abnormal or normal value, and displaying this using an SCTID code. Thus, incorporating semantic interoperability and as a result, achieving a set operation objectives. Encrypting and decrypting of the sensor data was detailed as the following use case, which would incorporate either of the technologies briefly introduced below. Finally, the last use case involved the secure transfer of the pseudo sensor data to the MySQL database using XAMPP as the server stack software. This also incorporates include and extend relationships, where each time the systems attempts to transfer data to the database, all client-server relationship connections are tested and verified. If one server-side connection fails, the system will be unable to insert data into the database, thus a display error should be raised in the possible occurrence of this given event. As this event will only occur given the server connection is not established, it is depicted as an extend relationship.

The use case representing the system **Design B** is depicted in Appendix 8.7.1, where the same actors are utilised and the same top and bottom use cases are used. This is due to the initial and end functionality of this system is the same as that in Design B. the major difference is found in the pseudo sensor data is formatted into a XML file and how this file is then stored on a micro-SD card. These functions incorporate the use of the Arduino SD library, hence why the included relationship is depicted.

4.1.3.2 Activity Diagram

An Activity diagram essentially showcases the flow of activities performed by the system. They are also considered to be under the behaviour diagram subset, as they describe “what is happening within the system being modelled” [47]. Figure 13 below showcase the flow of activities within my project.

Activity diagrams offer numerous beneficial features to the developer, where an Activity diagram can represent the following:

- Showcase the logic of the steps occurring within a system.
- Provide a description of the steps performed within the UML Use Case diagram.
- Give an illustrative representation of “workflow between users and the system”.
- Simplify and provide explanation for any complicated use cases.

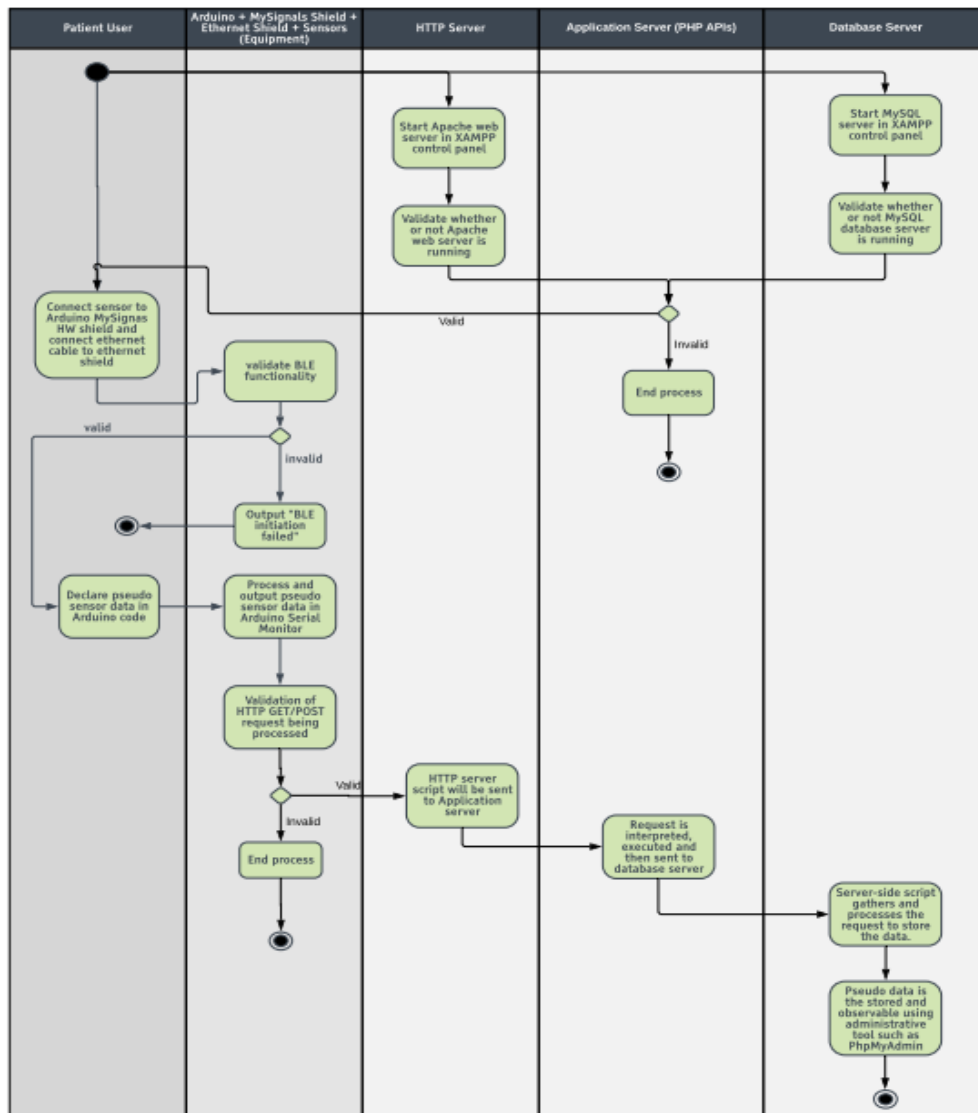


Figure 13: Design A Activity Diagram

- Lastly, allows software architecture elements to be modelled, including the following: method, function and operation.

The **filled black circle** represents the initial or starting node, where **black lines with arrow heads** are used to represent the direction of flow of activities within the system. Actions or activities are shown through the **green boxes** within the diagram, representing actions carried out by the system itself. The

diamond shape, which has been used on several occasions below represents a merge, which combines edges that have been separated by a decision. In my activity diagram below for design A, merges are used to validate actions carried out by the system. Finally, the **filled black circles with a black bordered outline** are used to represent the end node, thus the end of the activities.

Therefore, the activity diagram displayed above details the flow of actions carried out by the system detailed as **Design A**. The purpose of the Activity Diagram is to expand on the high-level aspects detailed in the use case diagram above. To begin, the initial step involves the initialising of the server-side technology, where if this is not successful the activity diagram ends (see top right portion of Activity Diagram). When successful the system initialises the BLE module on the MySignals HW shield, ensuring sensor connectivity can be achieved. Following this, pseudo sensor data is declared and outputted in to the serial monitor, along with the SCTID corresponding to the finding and the category this finding falls under. This pseudo data is then compiled into a HTTP request in either the form of a POST request or a GET request, which is then validated and then processed. If this request is invalid due an error such as API incompatibility, the activity diagram ends. When validated, the HTTP server script is sent to the Application server. This compiled PHP script, allows for sensor data received from the Arduino to be inserted into the MySQL database. This request is then interpreted, executed and then sent to the MySQL database server. The SQL server-side script then gathers and processes the request to store the data and can be observed using a database administration tool such as PhpMyAdmin for MySQL. **Design B** shown in Appendix 8.7.2 on the other hand incorporates a step to increase the security of the system. Firstly, a new column is incorporated into the Activity Diagram representing the XML file created, which incorporates the function of creating and storing the XML file. This file format is the validated to ensure compatibility with the PHP API and the application server. Once validated the process involved in processing this HTTP POST request can be visually described as being the same as that shown in Design A. However, differences do lie in the method used with the PHP API to process this POST request.

4.1.3.3 *Component Diagram*

It can be said that component diagrams are integral to the effective building of a given software system. They allow for visual aspects of system structures to be better understood. The purpose of such a diagram is to simply show the relationship between components in a system. The term in “component” refers to a “module of classes, representative of systems or subsystems which have the ability to be interfaced with the rest of the system” [48]. Component-based development (CBD) is a development approach designed and optimised to allow the developer to identify the necessary components, so the system functions as intended. Component diagrams also offer the following benefits by helping a developer:

- Visually display the systems **physical structure**.
- Display **relationships** between the system's components.

The structure of a given component diagram is as follows. The **components** can be depicted in either or both of the following ways. A “component symbol” in the top right corner of a component box can be used. See figure 16 of Design A for visual guidance. Component can also be written between the double angle brackets as follows, <<component>>, and inserted inside a given component box.

Additionally, the following components of the Component Diagram, are representative of the interfaces interacting between two components. These interfaces are as follows.

Provided Interface: The symbol to the right, figure 14, represents the interface representative of information produced by a component and is then used by the “required interface on another component” [48].



Figure 14: Provided interface

Required Interface: symbol shown in figure 15 to the right, is where a component is required to acquire information from another component, to perform it necessary function.



Figure 15: Required interface

Using these benefits and components, the Component diagram below (figure 16) effectively provides visual aid to allow for better understanding of the system, as well as the relationships between components in the system. Thus, constructing the Component Diagram for Design A. Initially, sensor selection is introduced as a component of the system, which is inclusive of the corresponding Arduino (.ino) file required to program the Arduino

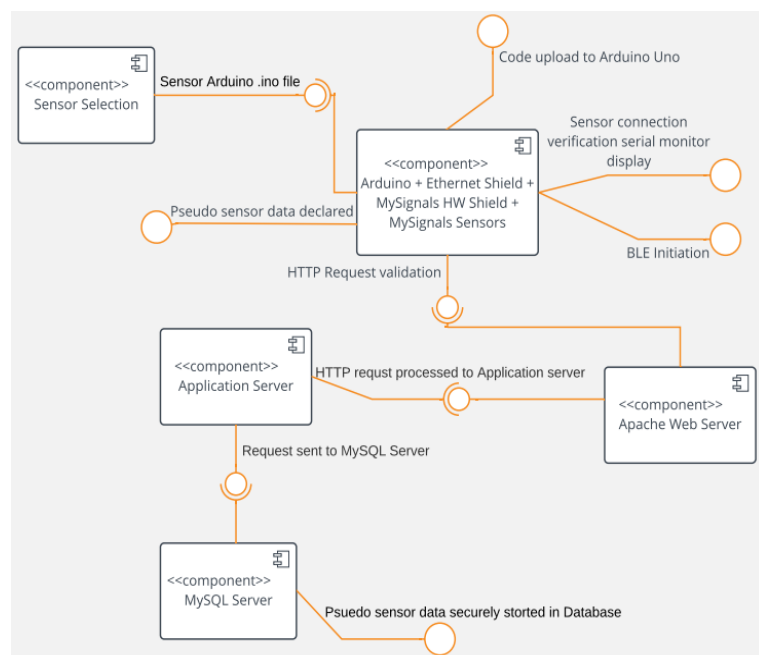


Figure 16: System Design A component diagram

Uno and all shields attached. This component is therefore provided to the Arduino Uno and all the shield devices connected, including the Ethernet Shield and the MySignals HW shield, which acts as another component as part of this system. This component also provides interfaces such as upload of code to the Arduino Uno, Arduino IDE, through the initial BLE initialisation phase. Pseudo sensor data is then declared within the Arduino IDE. Additionally, an HTTP request is validated, processed and

provided to the Apache web server, another component. This web server then processes this HTTP request to the Application server component. Ultimately, the request is processed (sent) to the MySQL server component and hence, the pseudo data is securely stored within the MySQL database. Appendix 8.7.3 represents the component diagram for Design B, where the explanation is provided in the Appendix.

4.2 Project Design Implementation

This section will go through each step in the process to achieving all objectives set in the Project Specification (section 1.5). All design implementations will be discussed, with an explanation into how each system design is able to incorporate successful techniques to deliver on all objectives. All design implementation in this section utilise the sp02 BLE sensor module. Code to achieve the same level of functionality with another other four sensor modules will also be included in the Appendices. These systems will then be tested against these objectives, where an analysis will explore how effectively areas of the system design were able to fulfil set objectives.

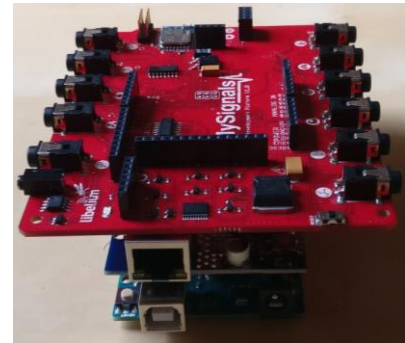


Figure 17: Arduino Shield equipment stack top angle

Design A – Fully Automated System

4.2.1 Connecting Sensors to MySignals HW Shield

All designs incorporated the same method of connecting the sensor devices to the MySignals HW Shield. This is due to the same MySignals libraries providing the same BLE initiation functionality, thus being utilised. Therefore, the methodology of how the sensors were connected will only be explained in detail in this section and will be referenced back in sections further on in the report. The code for initialising the

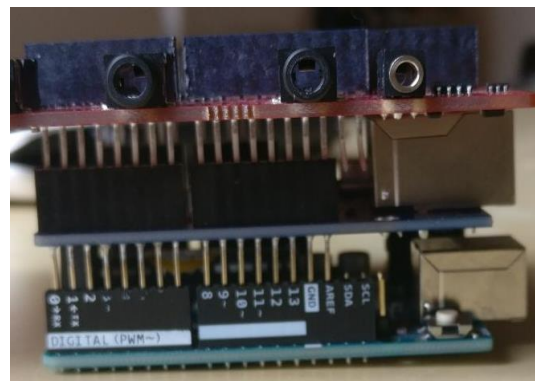


Figure 18: Arduino shield equipment stack side view

MySignals HW shield can be found in Appendix 8.8.1. All sensor initiation code will also be included in the Appendices. Below, I will provide an in-depth analysis of the main functions operating within the code. Also, the setup of the equipment used to effectively connect the sensor devices was the same throughout the conduction of this project. This is shown in figures 17 and 18, and was used as it allowed the Arduino Ethernet Shield access to the SPI pins on the Arduino Uno microcontroller (see figure 5), thus allowing for a successful internet connection to be produced. This then allowed for the MySignals HW Shield to be stacked on top of the Arduino Ethernet shield, where all essential pins were

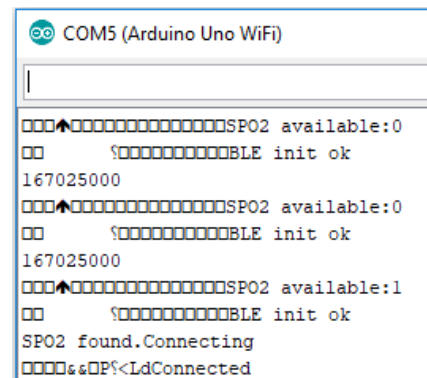
connected. This connection was deemed successful in that the BLE module was initiated, thus in theory it could be assumed the sensor device would be able to establish a successful connection.

To begin, two libraries must be declared and included in the first lines of the code (see figure 19), to ensure all MySignals functions are able to complete required functionality. These libraries must therefore be downloaded [49] and included as zip folders in the Arduino IDE being used [50], as the fundamental code required to initialise the BLE module on the MySignals shield and consequently connect to the sensor, is contained within these libraries. All code assembled within the Arduino IDE used this as a foundation and built upon the core functions MySignals implemented as part of their libraries.

```
#include <MySignals.h>
#include <MySignals_BLE.h>
```

Figure 19: MySignals libraries

Following this, the BLE module of the MySignals HW shield must be initialised. This is where the second library shown in figure 19 is utilised. All code utilised for this stage is implemented from the libraries provided by MySignals. Refer to Appendix 8.8.1, lines 22 to 50. Additionally, the MAC address of the sensor device must be found and inputted into line 4 of the code, shown in Appendix 8.8.1 to ensure the BLE module connects with the wireless sensor. This was found using the application discussed in the Literature Review, nRF connect. Lines 33 and 37 are respectively critical in displaying to the user whether the BLE module has successfully initiated or not.



```
COM5 (Arduino Uno WiFi)
SP02 available:0
BLE init ok
167025000
SP02 available:0
BLE init ok
167025000
SP02 available:1
BLE init ok
SP02 found.Connecting
LdConnected
```

Figure 20: Available number of sensors in Serial Monitor

The next step involves connecting to the SP02 sensor. First the number of SP02 sensors available is declared, through line 51 within the code. This value is then outputted using a Serial.println() function (lines 54 and 55), so the user can be aware of whether the sensor has connected. Figure 20 shows how this is then outputted to the serial monitor. The MAC_SP02 variable was declared on line 4, representing the MAC address of the SP02 sensor device. The BLE module will connect to the sensor device with the declared MAC address and only this device, see Appendix 8.8.2, line 65. Following this, the SP02 sensor can be programmed using a method developed by MySignals, to collect the patients SP02 level and Pulse (Appendix 8.8.2 lines 90 to 109). Ultimately, the due to successful nature of the connection achieved, it can be assumed the sp02 sensor will be able to output both the SP02 and pulse values to the serial monitor, using the code displayed in Appendix 8.8.3, lines 111 to 122. Lines 123 to 165, ensure various error messages are relayed back to the user in the case of an error occurring during the eventual step of acquiring the sensor data.

Following this, pseudo sensor data is utilised to ensure ethical viability of the project is justified as discussed in the section 1. This pseudo data is simply declared using the “float” and “u_int32” functions to respectively declare the spO2 and SCTID variables (see Appendix 8.9.1.2, lines 14 and 15).

Additionally, SCTID codes are incorporated through Appendix 8.8.4, lines 167 to 179. This defines the integer values taken up by the abnormal and normal SP02 terms. These SCTIDs are located using the SNOMED CT Browser (see section 2.8.2). Through the incorporation of an “if” and “else” statement, the SCTID associated with the pseudo sensor data value can be printed to the serial monitor. It should also be noted that the pulse was attempted to be incorporated, however due to the Arduino Uno R3 being limited to 32 bits (section 3.5.3), this variable could not be declared. The SCTID for a patient’s pulse level is a 64-bit integer, and therefore could not be incorporated due to this hardware limitation.

4.2.2 Secure transfer of Sensor Data to MySQL Database

Sending the sensor data to the MySQL database utilised the validation and processing of HTTP request. To ensure that the HTTP requests were being processed effectively, I adopted an initial method of testing with fake data (Appendix 8.9). When this functioned effectively, I transferred the same techniques and integrated them with the MySignals HW shield (Appendix 5.2.4), to ensure the ultimately effective functionality was achieved. This method could then be adopted as necessary of any given sensor. It can be argued both POST and GET are equally secure. However, a POST request does not declare the data values in the URL (Uniform Resource Locator) being processed to insert the data, thus hiding the values from potential attackers. Therefore, I decided to implement the **POST** request method. The code implemented for the GET request is shown in Appendix 8.9.1, as this was fundamental in the development of my principle PHP knowledge.

4.2.2.1 PHP POST APIs

Two PHP APIs were implemented to ensure secure connection was established with MySQL database and the table within the database, where data is being stored. The PHP files are called config.php and update.php respectively. They are stored in the following directory C:/Users/XAMPP/htdocs/update. It is important the API files be stored within the **htdocs** folder inside XAMPP, as this allows the Apache web server and MySQL database server to access and connect to these files, thus serving the intended purpose of the files. The first PHP API, **config.php** can be seen in appendix 8.9.2.1. This API establishes a secure connection with the database and web server, to allow for data to be exchanged using a client-server relationship. The following variables must first be declared: the host name, the database user name, the database password and the database name. In PHP, variable names are declared using the “\$” at the start. The mysqli_connect function is then implemented (line 9 and 10), to securely connect the PHP API to the declared database.

The second PHP API file, **update.php** (see appendix 5.2.2), instantiates the config.php file to ensure database server connectivity is established. Following this, line 6 declares a variable called \$query, which runs an insert function on the database. This insert function declares the name of the table within the database that the sensor data is being inserted into, whilst also declaring the HTTP request method being implemented to insert the data. This PHP API incorporates the POST method, as can be seen on line 7. The following mysqli_query function is then used, where the variable \$query is declared within its parameters (line 11). Running this variable, therefore implements the insert function on the MySQL table.

The mysql extension **mysqli** was adopted due to its enhanced security features, which allowed the PHP API to securely connect to a given database. It should be noted that the “i” in the mysqli extension, is an acronym for “improved”.

4.2.2.2 PHP GET API

An alternative PHP API was implemented to process the GET request. This is shown in appendix 8.9.1.1. The major difference can be seen in how the insert function is implemented, where the mysql_real_escape_string() function is used (line 6). Inside these parameters, the GET request is declared, \$_GET[''], where the value inside of these parentheses disclosed in the square brackets is used in the URL implementation. The code showing the GET requests validation and processing from the Arduino Ethernet shield is shown in appendix 8.9.1.2. The following section will explain in further detail how the Arduino Ethernet shield was programmed to function as a server.

Arduino Ethernet Shield Request

4.2.2.2.1 Processing HTTP GET Request Using Test Data

Using a GET request offered a different dynamic to design, through the option to insert multiple columns of data through one request. This method could therefore be implemented in the alternate_mysignals database design, where SCTID codes and data values are stored in separate columns (Table 4). The Arduino code implemented for this method is shown in appendix 8.9.1.2, where the major difference lies in the declaration of the PHP API, the HTTP request being utilised and the method of used to declare the URL functions (see lines 36 to 59).

4.2.2.2.2 Processing HTTP POST Request Using Test Data

In order to process this data on this the Arduino Uno microcontroller, the following code displayed in Appendix 8.9.2.3, was used once

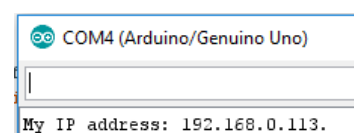


Figure 21: Arduino Ethernet Shield

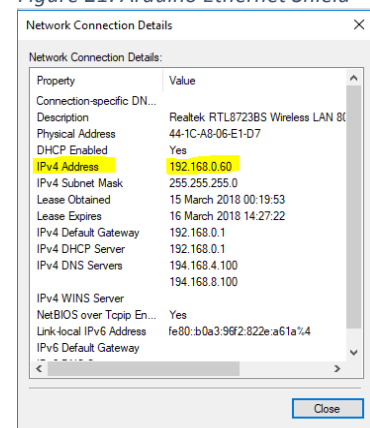


Figure 22: Network connection details in Windows OS

again. First, the two libraries were declared as shown on lines 1 and 2, to allow for the Arduino to program the Arduino Ethernet Shield. Next a variable for the MAC address must be created, which can either be defined to the manufacturers specifications or independently defined (see line 4). As I was unaware of the Ethernet Shield MAC Address, I assigned my own. Also following this, two IP addresses were defined using the “char server[]” variable. One represented the Arduino ethernet shield (see line 6) and another, represented the IPv4 Address. The Arduino Ethernet shield IP address can be found using an example Arduino sketch provided by the developers, such as DchpAddressPrinter.ino, where the output can be seen in figure 21.

The **IP address** of the given system being used is found through the following method on a windows OS. Firstly, opening the control panel and selecting the “Network and Internet” heading, and then following up by selecting the “Network and Sharing Centre”. Then selecting the connected connection being utilised, and then selecting the details section for further information, figure 22 is displayed. In this, the IPv4 Address can be seen highlighted, and will differ for each system, but should be specified for the system on which the web server is being activated from.

Next a uint32_t variable is defined for the SCTID (line 13), along with the sensor data being acquired (line 15). As in the PHP Post API the value referred to inside the square brackets is “sp02”, in the defined POST request (Appendix 5.2.2 line 7). Therefore, the “yourdatacolumn” variable is defined as this (line 16), to ensure the POST request is processed for the declared value. The string sp02 is then defined as shown on line 32, incorporating all variables in the correct format, including the colon. This spo2 string is then used later as a client.println() function, to ultimately process the sensor data. Before this, the PHP API update.php location is defined (line 39), along with the host IPv4 address and the Apache webserver port being utilised (line 42). In this example, this is declared as 192.168.0.60 and 8080, respectively. It is also critical the version of HTTP being used is declared, as well as the connection close function (line 44) and the content type (line 45).

4.2.2.2.3 Processing HTTP POST Request Using MySignals Data

Appendix 8.9.2.4 shows the implementation of this same method but utilises the option for MySignals sensor data, rather than the pseudo data. Line 18 to 22, shows how the variables were declared to hold all data values being collected from the sensor. As before, a string holding the value “=sp02” is declared to carry out the POST request (line 24). Now in processing the request, an “if” and “else” statement is created (lines 200 to 209), which determines the SCTID code dependent on the sp02 value. Consequently, the following code uses this determined SCTID code and the sp02 value, and implements this to be processed as a HTTP POST request (lines 211 to 233).

Design B – Semi-Autonomous System

This system focuses on the acquiring of sensor data, then formatting this data into an XML file, which is then inserted into an online webpage, which ultimately inserts the XML file into the MySQL database table. The sensor data is acquired from the MySignals HW shield using the same method shown in section 4.2.1.

4.2.3 Formatting the Sensor Data into an XML File

Due to restrictions with the Arduino microcontroller, the XML file can only be created and stored onto a micro-SD card, through the utilisation of the Arduino Ethernet Shield. The code shown in Appendix 8.10 uses fake data, where Appendix 8.10 focuses on utilising the MySignals sensor data.

The first requirement is the initialisation of the SPI and SD libraries (lines 1 and 2). Following this, a constant integer variable “chipSelect” is declared, which is utilised later within the code, to ensure the SD card slot on the Ethernet shield is functioning. Lines 6 to 9, declare all pseudo sensor data values, which will be inserted into the XML file and consequently into the database. Line 21, then utilises the SD.begin() function within an “if” statement to initialise the SD port, ensuring the user that there is a card inserted. If not, line 22 is implemented to output this error to the serial monitor. As a result, the SD.remove() function can be used to create the designated file name (see line 30). Line 32 then opens this file and begins the process of writing to it. Given the that the file is available (line 35) and the declaration of File as MyFile, all commands involving writing to the file,

```
<?xml version="1.0"?>
- <patient>
- <sensor>
  <sp02> 167025001:100 </sp02>
</sensor>
</patient>
```

Figure 23: XML file created and saved to micro-SD

must be written as MyFile.print() or MyFile.println(). Thus, lines 37 to 55 implement the writing of the sensor data to the micro-SD card in an XML format. The XML file created can be seen in figure 23. If the Arduino was unable to open the file an “else” statement is used to output the error that has occurred to the serial monitor (line 64).

4.2.4 Securely Inserting the XML File into the MySQL Database

4.2.4.1 PDO Extension Utilisation

Inserting this XML file into the MySQL database involves several steps, to validate the file and format and ultimately process the request. The first step is the establishing of the database server and connecting to the given database. This is achieved through the import.php file shown in Appendix 8.10, where this PHP file incorporates the use of PDO extension, rather than mysqli. First the file extension is declared in capitals, due to the how the Arduino formats the file extension. This was important to incorporate (line 8) as file extensions have a case sensitive nature. Following this, the explode function is used (line 9) to break down the string into an array, where the first parameter declares the separator,

hence the full-stop after the file name. Thus, the file name is separated at the full-stop and both portions of the name are outputted into a string. Line 10 then locates the later portion, thus the file extension name and declares this in the `$file_extension` variable. Following this, an “if” statement is incorporated to ensure both the declared file extension and the proposed file extensions are the same.

The `simplexml_load_file` function is then used to interpret the XML file into an object, where this object is then declared into the `$data` variable (line 13). Next, the database connection is established, using the PDO extension. Line 14 exhibits this and line 15 explains the different components being incorporated. Prepared statements are used on lines 17 and 18. Following this, the insert function is utilised and declared into the `$query` variable (lines 20 to 24), which is then executed from line 26 to 36.

The website created to carry out this insert function is named `index.php`, refer to appendix 8.11.1.2. The code for this portion was sourced from the following reference [51], as learning to code in HTML and jQuery was not part of the proposed project plan, thus various risks could be eliminated by using a template for my code. A basic template was adopted to incorporate

the necessary headings, boxes and buttons. The main title “Import XML Data into Mysql Table Using Ajax PHP” (line 7), is written between `html <title>` and `</title>` tags. HTML is divided into various levels of

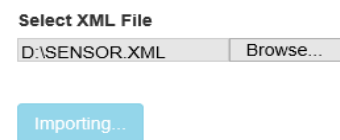


Figure 24: XML file upload website

tags, where for example the `<html>` is used to begin all contact with the html document. Then `<head>` and `</head>` tags can be incorporated, where everything between is part of the html documents heading (lines 6 to 10). The fundamental tag is `<body>` and `</body>`, as everything between this is visible on the webpage. Thus, all webpage headings and textboxes must be included between these tags (line 11 to 32). The webpage created as a result of this is shown in Appendix 8.11.1.2.3. Figure 24 shows the design of the website and how the given user can browse for their designated XML file, which can then be upload. Once inserted, a success message is displayed to make user aware.

Following the creation of the viewable webpage, focus was on incorporating functionality and insuring the desired actions the website was created to carry out, could then be carried out. To do this, the **jQuery JavaScript library** is used, which has the purpose of making the use of JavaScript much simpler. jQuery is currently the most popular and most extendable JavaScript framework, incorporating some of the following features: “HTML/DOM manipulation, CSS manipulation, HTML event methods, Effects and animations, AJAX and Utilities”[52].

4.2.4.2 Explanation of jQuery Code

The code utilising the functionality of the jQuery library is shown in Appendix 8.11.1.2.2. Line 35 of the code detects the state of readiness of the code, using the `$(document).ready(function()` function. The code included beyond this point will only run once the page “Document Object Model (DOM) is ready for the JavaScript code to execute” [53]. Following this, a method is implemented to prevent the default action of the event from being triggered [54]. Line 39 of the code, `$.ajax()` allows the asynchronous HTTP requests to be performed, where the details of this request are declared between the parameters, from line 40 to 49. The `beforeSend` function ensures the necessary actions are carried out before the POST request is processed. If the HTTP request is then successfully validated, the data can then be included and ultimately processed (lines 50 to 56).

4.2.5 Arduino Shield Stack Compatibility issues

In order for this design to be fully functional certain requirements must be fulfilled. Firstly, the design function when using pseudo data, thus the PHP APIs and the Arduino code can be said to be functioning effectively. The MySignals HW shield is then connected to test if the system is able to function with all shields stacked together. The output of the BLE initiation being successful suggests the system will be able to connect to the sensor device. However, this is not achieved, instead the MySignals shield reoccurring outputs the number of spO2 sensor device available as 255. Possible issues with power limitations were resolved by powering the Arduino with a mains 5V supply, along with the serial cable.

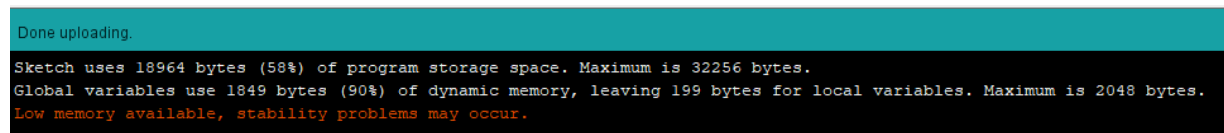


Figure 25: Deficiency in Arduino Memory

The occurrence of this error can however be explained with an added possible solution.

Figure 25 above shows the error occurring the Arduino IDE. This message states that there is “low memory available” within the microcontroller and that “stability problems may occur” as a result of this. These functionality problems can then explained through the Arduino Uno’s low flash memory availability of only 32k bytes. This can therefore explain why all three shields are unable to exhibit complete functionality between each other. To resolve this, a more powerful microcontroller could be used, with a greater flash memory such as the Arduino Mega2560.

It also worth noting of another Arduino cable error which occurred regularly, with the cause of the error associated with the overuse of the Arduino microcontroller (see appendix 8.12). During the occurrence of this, the code would fail to upload to the Arduino, thus preventing the use of the sensors.

Due to the spontaneous nature of this error, it is hard to predict, therefore limiting the use of the Arduino for extended periods of time, would be beneficial.

4.2.6 Pseudo Data Storage

The Pseudo data is stored into the MySQL database, which is then managed by PhpMyAdmin. Table 5 shows this table to the right. It is firstly worth noting that advantages of including the **timestamp** column. This was helpful in the testing phase of the project, but would also be beneficial to both the patient and the healthcare professional, to determine when and how regularly sensor data is uploaded. Furthermore, it can be said this database table will be more efficient in reducing computing power, when inserting data into only one column, as only one HTTP request is required to be processed. However, as explained before the alternative_mysignals database will be more effective in being able to transfer and extract the required data, as each column stores a separate data value.

id	sctid_sp02	timestamp
1	100000000:100	2018-03-26 10:07:45
2	100000000:100	2018-03-26 10:09:58
3	167025001:98	2018-03-26 10:25:20
4	167025001 : 98	2018-03-26 10:26:32
6	167025000 : 0	2018-03-27 12:12:57

Table 5: mysignals database table

4.3 Comparison of Security Features

Design A implemented the mysqli extension, whereas Design B focused on the incorporation of the PDO extension. Both extensions function well in preventing SQLi attacks, using prepared statements (see Appendix 8.11.1.1 lines 17 and 18). They also establish secure connections between the database servers, thus justifying the use of either in this aspect. However, in the situation the user would like to adopt a different database server, the PDO extension would be more viable, allowing for the easy transfer of data across databases. It can also be argued that mysqli, is native to MySQL thus features developed for the extension are likely to function with less glitches, as the library will be updated more often with unique features.

4.3.1 Encryption

An import feature missing from both design systems is the incorporation of an interoperable encryption standard. It can be argued that the system is secure without any form of encryption, however this layer of added encryption would prevent the severity of MITM attacks, whilst also ensuring that the patient sensor data is secure throughout all stages of system. The main issue faced when incorporating this function into the Arduino Uno microcontroller and the Arduino Ethernet shield, was the failure to establish interoperability between the Ethernet shield and the PHP API. This was mainly due to lack of resources and understanding in how the PHP API would be able to decrypt the data being sent to it. Both AES (Advanced Encryption Standard) and XXTEA are standardised methods. **AES** is a symmetric-key algorithm which operates as a block cipher, thus the same key is used

to both encrypt, and decrypt given data, where keys can be utilised in three different length: 128, 192 and 256 bits. **XXTEA** is also a block-cipher, where block sizes increment in multiples of 32-bits, thus providing greater level of flexibility in size of data being encrypted. The block-cipher method focuses on encrypting the entire block of data, rather than encrypting one bit at a time [55].

4.3.2 Ethernet Compared with WiFi

Initially, the focus of my project was on utilising a secure WiFi connection to transfer the data from the Arduino to the MySQL database. However, due to a lack of documentation on the Arduino Uno WiFi, which was a result of the product being discontinued by Arduino, difficulties arose in implementation of the on-board ESP8266 WiFi enabling microchip [56]. Also, with a lack of libraries and HTTP POST or GET request examples, I found it difficult to implement the same methodology I was able to in the Ethernet shield. Additionally, after conducting thorough research into the connection types, conclusive findings resulted in the Ethernet connection being the preferred choice.

4.3.2.1 Reliability

WiFi signals are subject to a larger amount of interference, due to layout of homes and the signal obstruction. This can then result in the signal being dropped completely, which would then disconnect the Arduino from the database server. As Ethernet connections are not affected by these issues, it can therefore be said they are more the reliable of the two.

4.3.2.2 Convenience

Given the initial scope of the project the positioning of the Arduino device was relevant, hence why WiFi would have been the preferred method of adoption. However, given that it is unlikely the patient will need to move around very often with the devices and the sensors, it is viable to utilise an Ethernet connection. Furthermore, due to restrictions on where the project equipment could be used, it was not possible to establish a WiFi connection, thus a secure Ethernet connection was the more viable option, as Ethernet ports are located throughout the School of Engineering laboratories.

4.3.2.3 Security

The first advantageous feature of Ethernet connectivity is the security associated with a physically secure connection, which prevents unauthorised personal from viewing the details between the connections. A wireless connection on the other hand can be intercepted with significantly less hassle than the Ethernet connection. Therefore, in terms of the adopting a connection with a secure nature, an Ethernet connection is also more viable.

5 Conclusion

To summarise, this technical project has effectively implemented two different system designs. Both designs were successful in being able to establish connections between the MySignals HW Shield and the sensor devices. Five sensor devices were tested with to ensure successful connections were established. These sensor devices Arduino IDE code required to securely connect the sensors to the MySignals HW shield, can be found in libraries using the following link [49]. These Arduino sensor connection code is displayed Appendix 8.12. There was however, a lack of functionality when combining the Arduino hardware shields together, where the sensor devices were unable to connect to the MySignals HW shield. This lack of functionality could simply be explained through the error message that occurred on upload to the Arduino. Nevertheless, both designs functioned beyond this point and were able to successfully and securely insert pseudo sensor data into the MySQL database. This pseudo data was then stored in a structured formatted, thus providing the system with structural interoperability. Additionally, features were incorporated through use of SNOMED-CT to achieve semantic interoperability between the MySignals sensor device and the EHR management database, successfully fulfilling a vital operational objective of the project. Therefore, it can conclusively be said that complete interoperability was achieved between the two systems the pseudo sensor data is being transferred between.

Design A used a fully-autonomous approach, as pseudo sensor data was first declared in the Arduino software. This pseudo data was then declared within the parameters required to process a HTTP POST request using the `client.print()` function. This function also implemented the connection with the PHP API, allowing for a secure connection to the designated MySQL database server to be established. Utilising this connection, the POST request could be validated and then processed to allow for the data to then be inserted into the specified database table. This design is the preferred design choice due to its autonomous nature, which alleviates work required on the patient user's behalf, therefore adding to the viability of the implementation of such a system. However, due to the lack of an encryption mechanism between the Arduino Ethernet Shield and the PHP application server, the system could be vulnerable to an attack from an unauthorised system or individual.

Design B was therefore implemented to resolve security related issues by creating a semi-autonomous system, which can manipulate the format of pseudo sensor data into an XML file, safely stored onto a micro-SD card located on the Ethernet Shield. This XML file is then inserted into to the PHP webpage (`index.php`) and then securely inserted into the MySQL database table as specified. By eliminating the step of processing the HTTP POST request through the Arduino Ethernet Shield and instead processing this request through the `index.php` file webpage, the system can conclusively be described as the more secure. This is due to the security features implemented through the PDO code as explained in section

4.2.4.1, as well as the increased security protocols imbedded in a computer device, in comparison the security features of the Ethernet shield. It can therefore be concluded that the vulnerability of the system has been drastically lessened, as a result of establishing this system as being more viable in its security aspects.

Given the initial requirements of the project and final designs implemented, it can conclusively be said that the specification was affected. Thus, the EHR management system was built with the functionality to achieve both technical and semantic interoperability, but was unable to incorporate high-level encryption mechanisms. Thus why Design B was implemented, showing a more secure alternate design method.

6 Recommendations for Further Work

From the output of the results achieved it can be derived that the issues are not associated with the software aspects of this project. Hence, it can be recommended that a more powerful Arduino microcontroller device is tested to attempt to achieve complete functionality with the sensors.

Furthermore, further investigation into encryption mechanisms is feasible given the systems inability to incorporate these features. It can be recommended that either AES or XTEA be used as the most viable mechanisms to implement, given that libraries are associated with both the Arduino Uno and PHP scripts. FHIR (Fast Healthcare Interoperability Resource) was briefly researched for this purpose of providing an added layer of security, thus could be a potential option for implementation. FHIR however utilised Health Level 7 rather than SNOMED-CT as its semantic interoperability mechanism.

It can also be recommended that an ESP2866 module be tested with. The availability of resources of this WiFi enabled module is much greater than that of the Arduino Uno WiFi. This can therefore eliminate various risks associated with the difficulties of the tasks. However, to implement this method, either an independent WiFi network will have to be provided or the researcher will have to be granted with access to the take certain equipment home, which will allow for the complete system functionality to be tested.

Finally, it would be recommended that before further research is conducted, that a full ethical approval is applied for, to allow for complete sensor testing. This would also for a tasks and objectives to be achieved at a quicker rate, as both the software design and testing phases can occur simultaneously.

7 References

- [1] - A paperless NHS: electronic health records. (2018). [ebook] House of Commons, p.4. Available at: <http://researchbriefings.files.parliament.uk/documents/CBP-7572/CBP-7572.pdf> [Accessed 4 Mar. 2018].
- [2] - Project Management and Engineering Research. (2018). [ebook] José Luis Ayuso Muñoz, pp.11, 124. Available at: <https://0-link-springer-com.pugwash.lib.warwick.ac.uk/content/pdf/10.1007%2F978-3-319-51859-6.pdf> [Accessed 7 Mar. 2018].
- [3] - Arthur, J., Nance, R. and Balci, O. (1993). Establishing software development process control: Technical objectives, operational requirements, and the foundational framework. *Journal of Systems and Software*, [online] 22(2), pp.117-128. Available at: <https://www.sciencedirect.com/science/article/pii/016412129390090K> [Accessed 2 Mar. 2018].
- [4] - Health Catalyst. (2018). *The Healthcare Database: Purposes, Strengths, and Weaknesses*. [online] Available at: <https://www.healthcatalyst.com/healthcare-database-purposes-strengths-weaknesses> [Accessed 4 Mar. 2018].
- [5] - Pluralsight.com. (2018). *Relational vs. non-relational databases: Which one is right for you?*. [online] Available at: <https://www.pluralsight.com/blog/software-development/relational-non-relational-databases> [Accessed 5 Mar. 2018].
- [6] - Json.org. (2018). *JSON*. [online] Available at: <https://www.json.org> [Accessed 28 Feb. 2018].
- [7] - MySQL, A. and Mankad, S. (2018). *A Simple Guide to Database Design in MySQL - Open Source For You*. [online] Open Source For You. Available at: <http://opensourceforu.com/2011/05/a-simple-guide-to-database-design-in-mysql/> [Accessed 28 Feb. 2018].
- [8] - Docs.microsoft.com. (2018). *Monitor XTP in-memory storage*. [online] Available at: <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-in-memory-oltp-monitoring> [Accessed 25 Feb. 2018].
- [9] - Oracle.com. (2018). *MySQL | The Most Popular Open-Source Database | Oracle UK*. [online] Available at: <https://www.oracle.com/uk/mysql/index.html> [Accessed 5 Feb. 2018].
- [10] - Sarig, M. (2018). *MariaDB vs MySQL: In-Depth Comparison*. [online] Blog.panoply.io. Available at: <https://blog.panoply.io/a-comparative-vmariadb-vs-mysql> [Accessed 5 Feb. 2018].
- [11] - Infinite-blue.com. (2018). [online] Available at: http://infinite-blue.com/blog/wp-content/uploads/2016/07/MariaDB_MySQL_Whitepaper_060116.pdf [Accessed 6 Feb. 2018].
- [12] - Docs.phpmyadmin.net. (2018). *Introduction — phpMyAdmin 4.9.0-dev documentation*. [online] Available at: <https://docs.phpmyadmin.net/en/latest/intro.html> [Accessed 5 Feb. 2018].
- [13] - Anon, (2018). *What is the function of an IP address?*. [online] Available at: <https://www.quora.com/What-is-the-function-of-an-IP-address> [Accessed 6 Feb. 2018].
- [14] - Bgp.potaroo.net. (2018). *AS65000 - BGP Table Statistics*. [online] Available at: <http://bgp.potaroo.net/as2.0/bgp-active.html> [Accessed 6 Feb. 2018].
- [15] - VersionOne Blog. (2018). *Hello HTTP: How to GET What You Want and POST What Ya Got on the Web*. [online] Available at: <https://blog.versionone.com/hello-http-how-to-get-what-you-want-and-post-what-ya-got-on-the-web/> [Accessed 10 Feb. 2018].

- [16] - Php.net. (2018). *PHP: Overview - Manual*. [online] Available at: <http://php.net/manual/en/mysqli.overview.php> [Accessed 11 Feb. 2018].
- [17] - Dev.mysql.com. (2018). *MySQL :: MySQL PHP API :: 6.1 Overview*. [online] Available at: <https://dev.mysql.com/doc/apis-php/en/apis-php-mysqldb.overview.html> [Accessed 11 Feb. 2018].
- [18] - Seffah, A., Donyaee, M., Kline, R. and Padda, H. (2018). *Usability measurement and metrics: A consolidated model*. [online] Available at: <https://link.springer.com/article/10.1007%2Fs11219-006-7600-8> [Accessed 14 Feb. 2018].
- [19] - Open-emr.org. (2018). *OpenEMR*. [online] Available at: <https://www.open-emr.org> [Accessed 14 Feb. 2018].
- [20] - Home, H., Development, W. and Stack, C. (2018). *What is a Software Stack? Choose the right stack for your project*. [online] Hiring | Upwork. Available at: <https://www.upwork.com/hiring/development/choosing-the-right-software-stack-for-your-website/> [Accessed 15 Feb. 2018].
- [21] - Group, D. (2018). *Welcome! - The Apache HTTP Server Project*. [online] Httpd.apache.org. Available at: <https://httpd.apache.org> [Accessed 16 Feb. 2018].
- [22] - Acunetix. (2018). *What is SQL Injection (SQLi) and How to Fix It*. [online] Available at: <https://www.acunetix.com/websitesecurity/sql-injection/> [Accessed 17 Feb. 2018].
- [23] - Healthit.gov. (2018). *What is an electronic health record (EHR)? | FAQs | Providers & Professionals | HealthIT.gov*. [online] Available at: <https://www.healthit.gov/providers-professionals/faqs/what-electronic-health-record-ehr> [Accessed 28 Feb. 2018].
- [24] - mysql_connect(), D. (2018). *Deprecated: mysql_connect()*. [online] Stackoverflow.com. Available at: <https://stackoverflow.com/questions/21797118/deprecated-mysql-connect> [Accessed 17 Feb. 2018].
- [25] - Gov.uk. (2018). *Data protection - GOV.UK*. [online] Available at: <https://www.gov.uk/data-protection> [Accessed 18 Feb. 2018].
- [26] - Equalityhumanrights.com. (2018). *The Human Rights Act | Equality and Human Rights Commission*. [online] Available at: <https://www.equalityhumanrights.com/en/human-rights/human-rights-act> [Accessed 19 Feb. 2018].
- [27] - More than 50 billion connected devices. (2018). [ebook] ericsson. Available at: http://www.akos-rs.si/files/Telekomunikacije/Digitalna_agenda/Internetni_protokol_ipv6/More-than-50-billion-connected-devices.pdf [Accessed 20 Feb. 2018].
- [28] - Contracting for the 'Internet of Things': Looking into the Nest. (2018). [ebook] Available at: https://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID2725913_code1175289.pdf?abstractid=2725913&mirid=1 [Accessed 21 Feb. 2018].
- [29] - Play.google.com. (2018). [online] Available at: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp&hl=en_GB [Accessed 21 Feb. 2018].
- [30] - m.dhgate.com. (2018). *HEXIN PCI-E NIC Desktop Gigabit Ethernet NIC PXE Diskless Desktop Network Card Independent network card 2018 from hkmurata*. [online] Available at:

- <https://m.dhgate.com/product/hexin-pci-e-nic-desktop-gigabit-ethernet/177904890.html>
[Accessed 22 Feb. 2018].
- [31] - Ieee802.org. (2018). *LMSC, LAN/MAN Standards Committee (Project 802)*. [online] Available at: <http://www.ieee802.org/> [Accessed 21 Feb. 2018].
- [32] - IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. (2018). [ebook] Available at: <http://ieeexplore.ieee.org/document/182763/> [Accessed 21 Feb. 2018].
- [33] - Uniform Data Standards for Patient Medical Record Information. (2018). [ebook] Report to the Secretary of the U.S. Department of Health and Human Services. Pages 21-22. Available at: <https://www.ncvhs.hhs.gov/wp-content/uploads/2014/08/hipaa000706.pdf> [Accessed 25 Feb. 2018].
- [34] - Agrawal, A. and Revelo, P. (2015). Analysis of the consistency in the structural modeling of SNOMED CT and CORE problem list concepts. *Journal of the American Medical Informatics Association*. [online] Available at: <http://ieeexplore.ieee.org/document/8217666/authors> [Accessed 27 Feb. 2018].
- [35] - Termbrowser.nhs.uk. (2018). *NHSDigital SNOMED CT Browser*. [online] Available at: <https://termbrowser.nhs.uk/?perspective=full&conceptId1=404684003&edition=uk-edition&release=v20171001&server=https://termbrowser.nhs.uk/sct-browser-api/snomed&langRefset=999001261000000100,999000691000001104> [Accessed 24 Feb. 2018].
- [36] - Agile Alliance. (2018). *What is Agile Software Development?*. [online] Available at: <https://www.agilealliance.org/agile101> [Accessed 20 Feb. 2018].
- [37] - Mohamed Sami. (2018). *Software Development Life Cycle Models and Methodologies*. [online] Available at: <https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/> [Accessed 27 Feb. 2018].
- [38] - Kienitz, P. (2018). *The pros and cons of Spiral Software Development | DCSL Software Ltd*. [online] DCSL Software Ltd. Available at: <https://www.dcssoftware.com/pros-cons-spiral-software-development/> [Accessed 9 Mar. 2018].
- [39] - Mysql.com. (2018). *MySQL :: MySQL Workbench*. [online] Available at: <https://www.mysql.com/products/workbench/> [Accessed 10 Mar. 2018].
- [40] - Mysql.com. (2018). *MySQL :: MySQL Workbench: Visual Database Design*. [online] Available at: <https://www.mysql.com/products/workbench/design> [Accessed 12 Mar. 2018].
- [41] - 19501:2005, I. (2018). *ISO/IEC 19501:2005 - Information technology -- Open Distributed Processing -- Unified Modeling Language (UML) Version 1.4.2*. [online] Iso.org. Available at: https://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=32620 [Accessed 20 Mar. 2018].
- [42] - Uml.org. (2018). *What is UML | Unified Modeling Language*. [online] Available at: <http://www.uml.org/what-is-uml.htm> [Accessed 20 Mar. 2018].
- [43] - Marjanovic, D. (2018). *PDO vs. MySQLi: Which Should You Use?*. [online] Code Envato Tuts+. Available at: <https://code.tutsplus.com/tutorials/pdo-vs-mysqli-which-should-you-use--net-24059> [Accessed 16 Mar. 2018].
- [44] - Store.arduino.cc. (2018). *Arduino Ethernet Shield 2*. [online] Available at: <https://store.arduino.cc/usa/arduino-ethernet-shield-2> [Accessed 16 Mar. 2018].

- [45] - W5100 Datasheet Version 1.1.6. (2018). [ebook] Spark Fun. Available at:
https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf
 [Accessed 21 Mar. 2018].
- [46] - Lucidchart. (2018). *UML Use Case Diagram Tutorial*. [online] Available at:
<https://www.lucidchart.com/pages/uml-use-case-diagram> [Accessed 20 Mar. 2018].
- [47] - Lucidchart. (2018). *Activity Diagram Tutorial*. [online] Available at:
<https://www.lucidchart.com/pages/uml-activity-diagram> [Accessed 19 Mar. 2018].
- [48] - Lucidchart. (2018). *Component Diagram Tutorial*. [online] Available at:
<https://www.lucidchart.com/pages/uml-component-diagram> [Accessed 19 Mar. 2018].
- [49] - Anon, (2018). *MySignals HW SDK V1*. [online] Available at: http://www.cooking-hacks.com/media/cooking/images/documentation/mysignals_hardware/MySignals_HW_SDK_V1.0.2.zip [Accessed 15 Jan. 2018].
- [50] - Arduino.cc. (2018). *Arduino - Libraries*. [online] Available at:
<https://www.arduino.cc/en/Guide/Libraries> [Accessed 6 Mar. 2018].
- [51] - Webslesson.info. (2018). *How to Insert XML Data into Mysql Table Using PHP with Ajax*. [online] Available at: <http://www.webslesson.info/2017/09/how-to-insert-xml-data-into-mysql-table-using-php.html> [Accessed 18 Mar. 2018].
- [52] - W3schools.com. (2018). *jQuery Introduction*. [online] Available at:
https://www.w3schools.com/jquery/jquery_intro.asp [Accessed 18 Mar. 2018].
- [53] - jquery.org, j. (2018). *\$(document).ready() | jQuery Learning Center*. [online] Learn.jquery.com. Available at: <https://learn.jquery.com/using-jquery-core/document-ready/> [Accessed 24 Mar. 2018].
- [54] - jquery.org, j. (2018). *event.preventDefault() | jQuery API Documentation*. [online] Api.jquery.com. Available at: <https://api.jquery.com/event.preventDefault> [Accessed 24 Mar. 2018].
- [55] - wolfSSL. (2018). *What is a Block Cipher? - wolfSSL*. [online] Available at:
<https://www.wolfssl.com/what-is-a-block-cipher> [Accessed 10 Mar. 2018].
- [56] - Web.archive.org. (2018). *Arduino UNO WiFi*. [online] Available at:
https://web.archive.org/web/20170711112529/http://www.arduino.org/products/boards/arduino-uno-wifi#board_usb_overcurrent_protection [Accessed 15 Mar. 2018].

8 Appendices

8.1 Project Plans

8.1.1 Initial Project Plan

	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1	🔍	Phase 1: Project Feasibility report	23 days	Wed 04/10/17	Fri 03/11/17	
2	🔍	Literature Review	1 wk	Wed 04/10/17	Tue 10/10/17	
3	🔍	Create architecture of the system	1 wk	Wed 11/10/17	Tue 17/10/17	
4	🔍	Create Use case diagram	1 wk	Wed 18/10/17	Tue 24/10/17	
5	🔍	Complete 1st draft of feasibility report assessment	1 wk	Wed 25/10/17	Tue 31/10/17	
6	🔍	Finalise the feasibility report for submission on the 03/11/17	6 days	Fri 27/10/17	Fri 03/11/17	
7	🔍	Phase 2: Getting sensor data onto Arduino board	31 days	Sat 04/11/17	Mon 18/12/17	1
8	🔍	Go over basics of C/C++ language and completing Audacity's C++ Basics Programme	10 days	Sat 04/11/17	Thu 16/11/17	
9	🔍	Ensure that the Arduino can interperate data from one sensors, starting with Glucometer sensor (utilising ADC cababilities)	1 wk	Thu 16/11/17	Wed 22/11/17	
10	🔍	Code which ensures all sensor data will be outputted into a similar and transferrable format	1 wk	Wed 22/11/17	Tue 28/11/17	
11	🔍	Incorporate standards to ensure the system complies with all rules and regulations	4 days	Wed 29/11/17	Sun 03/12/17	
12	🔍	Incorperate SNOMED CT as the clinical vocabulary to achieve semantic interoperability	11 days	Mon 04/12/17	Mon 18/12/17	
13	🔍	Phase 3: Encryption of sensor data	25 days	Mon 18/12/17	Fri 19/01/18	7
14	🔍	Learnt the basics of cryptography in Arduinos and how standard libraries can be implemented to achieve this (HOLIDAY PERIOD)	3 wks	Mon 18/12/17	Fri 05/01/18	
15	🔍	Encrypt the common format of sensor data, using a form of cryptography such as XTEA	2 wks	Mon 08/01/18	Fri 19/01/18	
16	🔍	Phase 4: Constructing Technical report & project and time management evaluation (preparation for Assessment B)	47 days	Wed 10/01/18	Thu 15/03/18	
17	🔍	Begin report based on research conducted to this point & during the holidays		Wed 10/01/18		
18	🔍	First draft	0 days	Mon 05/02/18	Mon 05/02/18	
19	🔍	Have final report ready at this point for amendments	0 days	Tue 27/02/18	Tue 27/02/18	
20	🔍	Deadline for final report		Thu 15/03/18		
21	🔍	Phase 5: Secure and wireless transfer of sensor data (technical interoperability)	20 days	Mon 22/01/18	Fri 16/02/18	
22	🔍	Add Arduino WIFI Shield and code to allow the Arduino to wirelessly and securely transfer data to EHR system	3 wks	Mon 22/01/18	Fri 09/02/18	
23	🔍	Data transfer must interact with APIs (Application Programming Interfaces) of EHR system	1 wk	Mon 05/02/18	Fri 09/02/18	
24	🔍	Ensure that this data transfer is technically interoperable with the EHR system	1 wk	Mon 12/02/18	Fri 16/02/18	
25	🔍	Phase 6: Decrypting data (semantic interoperability)	10 days	Mon 19/02/18	Fri 02/03/18	21
26	🔍	Data must be decrypted by EHR system	1 wk	Mon 19/02/18	Fri 23/02/18	
27	🔍	Ensure the data is semantically interoperable between CDS and the EHR system and use of SNOMED CT	1 wk	Mon 26/02/18	Fri 02/03/18	
28	🔍	Phase 7: Meeting Assessment (Preparation for Assessment D)	28 days?	Thu 15/03/18	Mon 23/04/18	25
29	🔍	Begin preparing for Assessment		Thu 15/03/18		
30	🔍	Be ready for this day			Mon 23/04/18	

8.1.2 Updated Project Plan

	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1	🔍	Phase 1: Project Feasibility report	23 days	Wed 04/10/17	Fri 03/11/17	
2	🔍	Literature Review	1 wk	Wed 04/10/17	Tue 10/10/17	
3	🔍	Create architecture of the system	1 wk	Wed 11/10/17	Tue 17/10/17	
4	🔍	Create Use case diagram	1 wk	Wed 18/10/17	Tue 24/10/17	
5	🔍	Complete 1st draft of feasibility report assessment	1 wk	Wed 25/10/17	Tue 31/10/17	
6	🔍	Finalise the feasibility report for submission on the 03/11/17	6 days	Fri 27/10/17	Fri 03/11/17	
7	🔍	Phase 2: Getting sensor data onto Arduino board	31 days	Fri 22/12/17	Fri 02/02/18	1
8	🔍	Go over basics of C/C++ language and completing Audacity's C++ Basics Programme	10 days	Fri 22/12/17	Thu 04/01/18	
9	🔍	Ensure that the Arduino can interperate data from one sensors, starting with Glucometer sensor (utilising ADC cababilities)	1 wk	Fri 22/12/17	Thu 28/12/17	
10	🔍	Code which ensures all sensor data will be outputted into a similar and transferrable format	1 wk	Fri 29/12/17	Thu 04/01/18	
11	🔍	Incorporate standards to ensure the system complies with all rules and regulations	4 days	Sat 06/01/18	Wed 10/01/18	
12	🔍	Incorperate SNOMED CT as the clinical vocabulary to achieve semantic interoperability	11 days	Fri 19/01/18	Fri 02/02/18	
13	🔍	Phase 3: Encryption of sensor data	14 days	Sat 06/01/18	Thu 25/01/18	7
14	🔍	Learnt the basics of cryptography in Arduinos and how standard libraries can be implemented to achieve this (HOLIDAY PERIOD)	3 wks	Sat 06/01/18	Thu 25/01/18	
15	🔍	Encrypt the common format of sensor data, using a form of cryptography such as XTEA	2 wks	Mon 08/01/18	Fri 19/01/18	
16	🔍	Phase 4: Constructing Technical report & project and time management evaluation (preparation for Assessment B)	47 days	Wed 10/01/18	Thu 15/03/18	
17	🔍	Begin report based on research conducted to this point & during the holidays		Wed 10/01/18		
18	🔍	First draft	0 days	Mon 05/02/18	Mon 05/02/18	
19	🔍	Have final report ready at this point for amendments	0 days	Tue 27/02/18	Tue 27/02/18	
20	🔍	Deadline for final report			Thu 15/03/18	
21	🔍	Phase 5: Secure and wireless transfer of sensor data (technical interoperability)	20 days	Mon 29/01/18	Fri 23/02/18	
22	🔍	Add Arduino WIFI Shield and code to allow the Arduino to wirelessly and securely transfer data to EHR system	3 wks	Mon 29/01/18	Fri 16/02/18	
23	🔍	Data transfer must interact with APIs (Application Programming Interfaces) of EHR system	1 wk	Mon 12/02/18	Fri 16/02/18	
24	🔍	Ensure that this data transfer is technically interoperable with the EHR system	1 wk	Mon 19/02/18	Fri 23/02/18	
25	🔍	Phase 6: Decrypting data (semantic interoperability)	10 days	Mon 26/02/18	Fri 09/03/18	21
26	🔍	Data must be decrypted by EHR system	1 wk	Mon 26/02/18	Fri 02/03/18	
27	🔍	Ensure the data is semantically interoperable between CDS and the EHR system and use of SNOMED CT	1 wk	Mon 05/03/18	Fri 09/03/18	
28	🔍	Phase 7: Meeting Assessment (Preparation for Assessment D)	53 days?	Thu 08/02/18	Mon 23/04/18	25
29	🔍	Begin preparing for Assessment		Thu 15/03/18		
30	🔍	Be ready for this day			Mon 23/04/18	
31	🔍		5 days	Thu 08/02/18	Wed 14/02/18	

8.2 Project Risk Assessment

Category	What are the risks	Planning for the risk (Avoid, Reduce, fall-back or Accept)	Mitigating actions required	Risk Evaluation			Risk Rating Low, Medium, High or Extreme
				Severity (1 – 4)	Probability (1 – 3)	Risk Ranking	
Management	Falling behind on module work and hence showing in grades	Reduce – Plans are far in advance as possible to predict the occurrence of such a risk. Make use of Gantt chart and other management tools to be on top of both workloads.	Put modules work first as worth more of overall grade, but draw up a plan to help eventually getting back on track for both, the project and the modules.	2	1	4	Medium
	Loss of data	Avoid – Keep multiple copies of work backed up regularly in the following forms: USB, online storage (google drive/email), university documents and both home laptops. Back up regularly at the end of every week.	Result to continuing from the most recent back-up.	2	1	4	Medium
	Unable to meet with project supervisor due to unforeseen circumstances	Fallback – In the case I am unable to arrange a meeting with my supervisor, I will have a pre-arranged plan to see Dr Richard Lillington and overcome this issue.	Will have to either wait to be assigned a new project supervisor or ask personal tutor for assistance.	3	1	6	Medium
	Task taking a significantly longer period of time than first estimated	Avoid – plan with expertise knowledge of given tasks and the time taken by researchers in the past to achieve these tasks. Always leave a buffer, to overcompensate for any given task.	Seek assistance from a lecturer that will be able to provide most support. As I am coding in C++, this will most likely be Dr. S. Fahmy.	2	2	5	Medium
Compliance	Caught not complying with PPE rules and regulation in place by the Engineering Department	Avoid – Make use of pre-paid locker to always have access to PPE on campus. Therefore, I can never make the mistake of forgetting to bring it with me.	The risk is not worth the reward in the case I am unable to access the lockers. Therefore, on that given day I will sensibly skip the lab session and catch up in convenient time.	1	1	1	Low
	Being unable to abide by all standards in place or finding out too late	Avoid – In literature review of Project assignment A, go through all possible standards in place that project must comply with.	Amend project work with the help of Dr Despotou, to make sure project complies with the standard that was missed.	1	2	2	Low
Technical	Mistake made either by myself or an individual to severely impact my equipment in labs. Risk can affect both hardware and software	Fallback – accept the mistake and the fact that there was a limited number of options available to you, to ultimately prevent the risk from occurring. Have back up board ready for use with the use of backed up code.	Begin by drawing up a plan, on how I will get back on track to the position I was in before. Work on the new device available to me, with my backed up piece of code.	3	1	6	Medium
	Cyber data flaw, where data is not transferred in a secure manner.	Avoid – Thoroughly research the various methods to securely transfer data and make sure the data transfer complies with all rules and regulation. This is to ensure patients data is protected from various breaches in security that could occur.	If data is found to be unprotected, protect the patient data by stopping the transfer of data. Work to resolve the issue through the help of a friend who is well experienced in dealing with securities.	2	2	5	Medium
Financial	Project going over budget or changes to proposed budget	Avoid – By using WMG's equipment, I can easily avoid any conflicts in the budgeting of this project.	In the case this does occur I will make use of WMG's equipment	1	1	1	Low

8.3 MySQL PHP APIs

8.3.1 PHP MySQL Extension

This is the original but slightly outdated extension, which allows for the development of applications that will interact with a MySQL database. Using a procedural interface, this extension is only intended to be used by MySQL versions older than 4.1.3. Therefore, it is implied that only limited functionality will be available when used with MySQL versions 4.1.3 and later. Example code is as follows,

```
mysql_connect($host, $user, $password);
```

8.3.2 PHP mysqli Extension

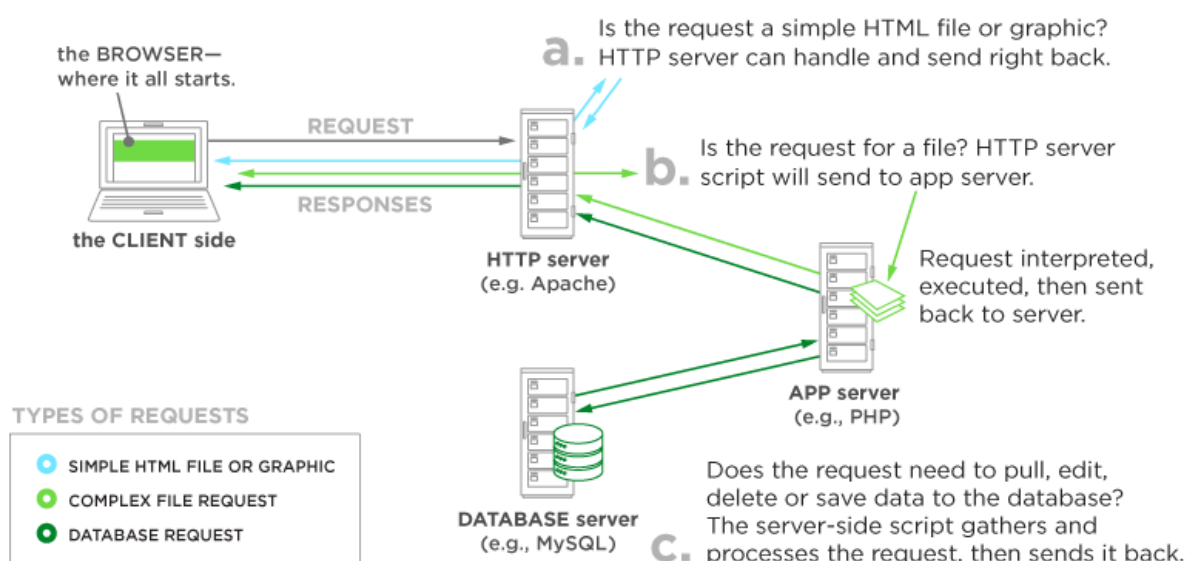
It should be noted that the “i” in the mysqli extension, is an acronym for “improved”. It was developed to optimise use of new features found in MySQL version 4.1.3 or newer. Key enhancements include the following: “Object-oriented interface, Support for Prepared Statements, Support for Multiple Statements, Support for Transactions, Enhanced debugging capabilities and Embedded server support” [<https://dev.mysql.com/doc/apis-php/en/apis-php-mysqli.overview.html>]. Example code is as follows,

```
mysqli_connect($host, $user, $password, $dbname);
```

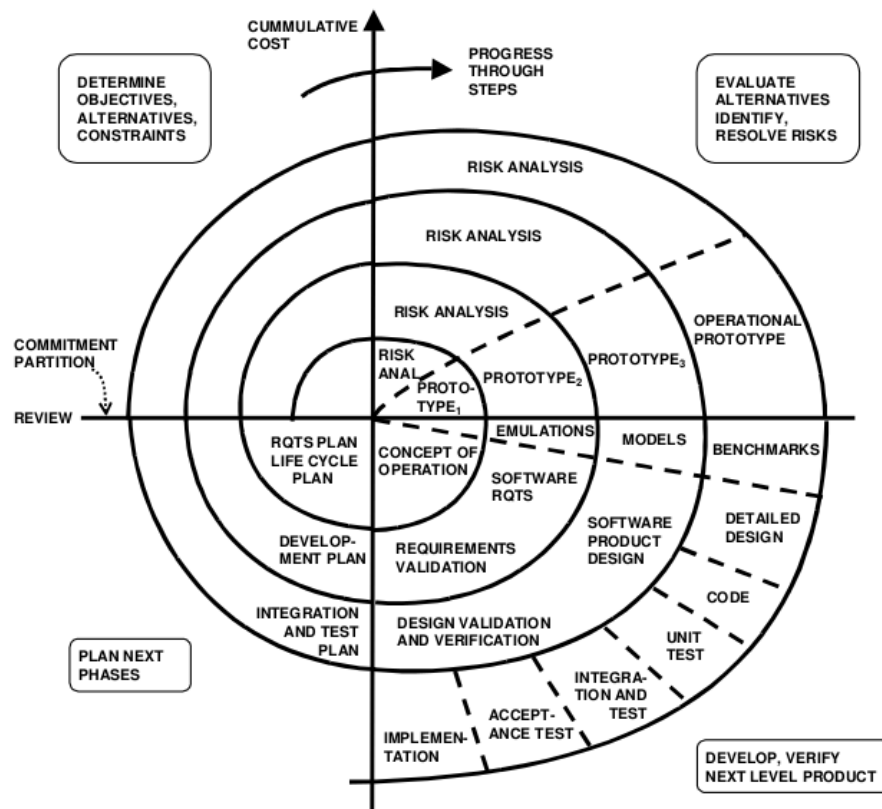
8.3.3 PHP Data Objects (PDO) extension

Using a PDO will provide a consistent API, regardless of the database server the application is connecting to. Thus, in theory, using such an extension allows developers to switch database server, by only making minor changes to the PHP code. Its advantageous features include being a simple, clean and portable API. However, it doesn’t fully utilise the advanced features available in the latest MySQL server. Example, `new PDO('mysql:host=localhost;dbname=test', $user, $pass);`

8.4 Operations Conducted by a Software Stack

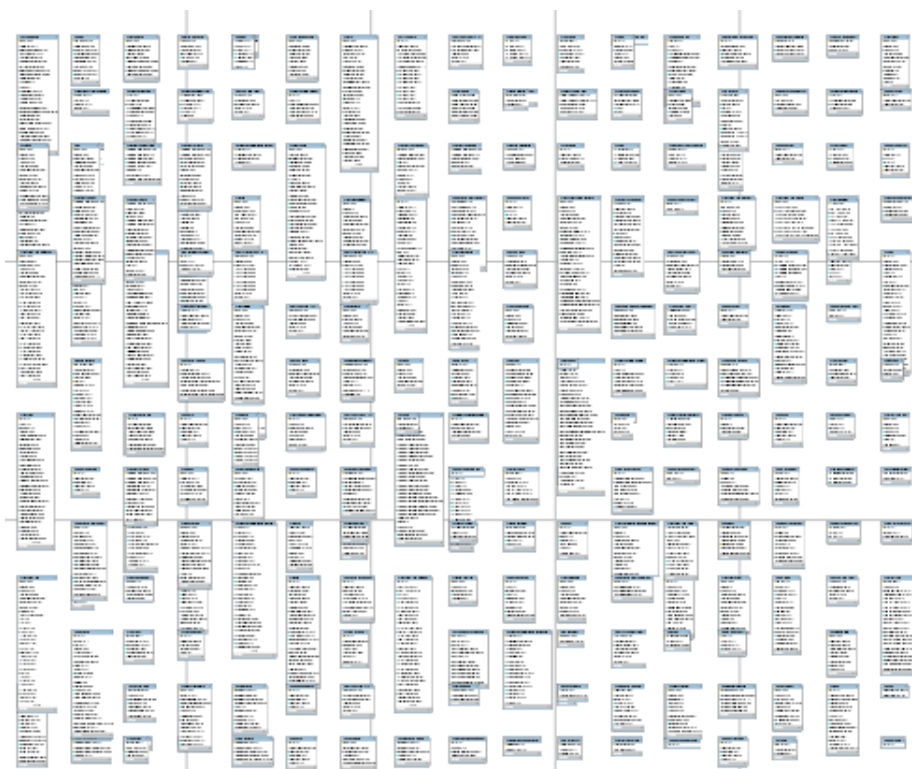


8.5 Spiral Methodology

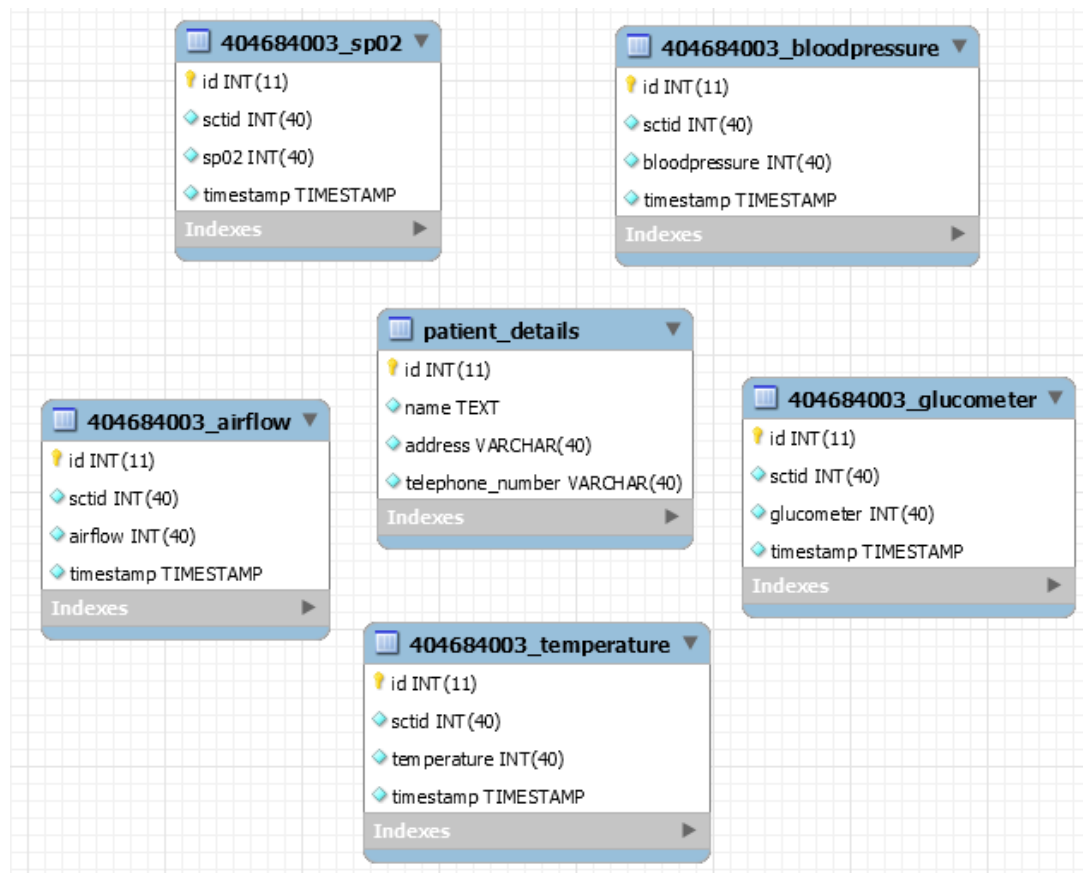


8.6 Database Design

8.6.1 OpenEMR Database Design

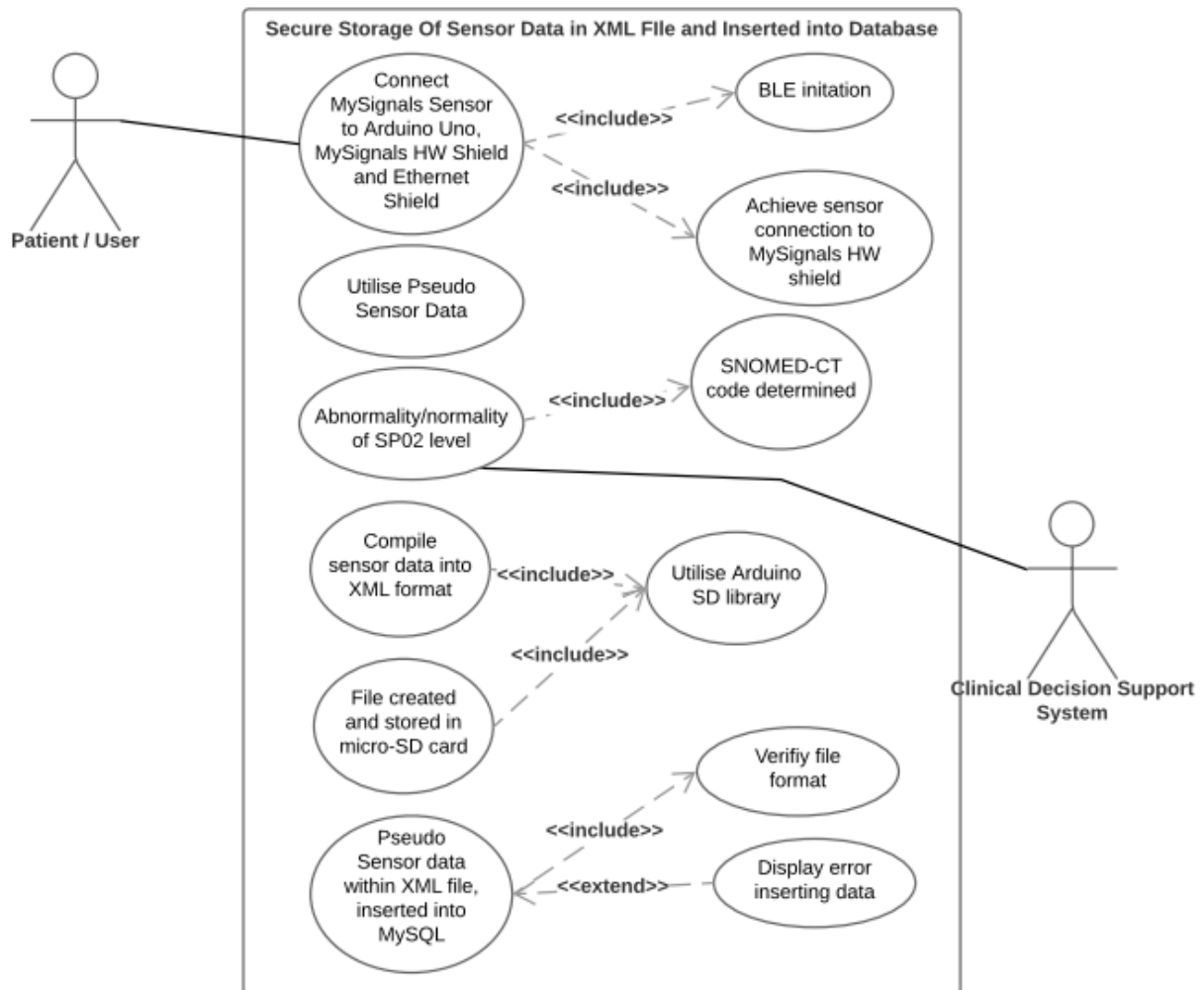


8.6.2 Database Design of “alternate_mysignals” Database

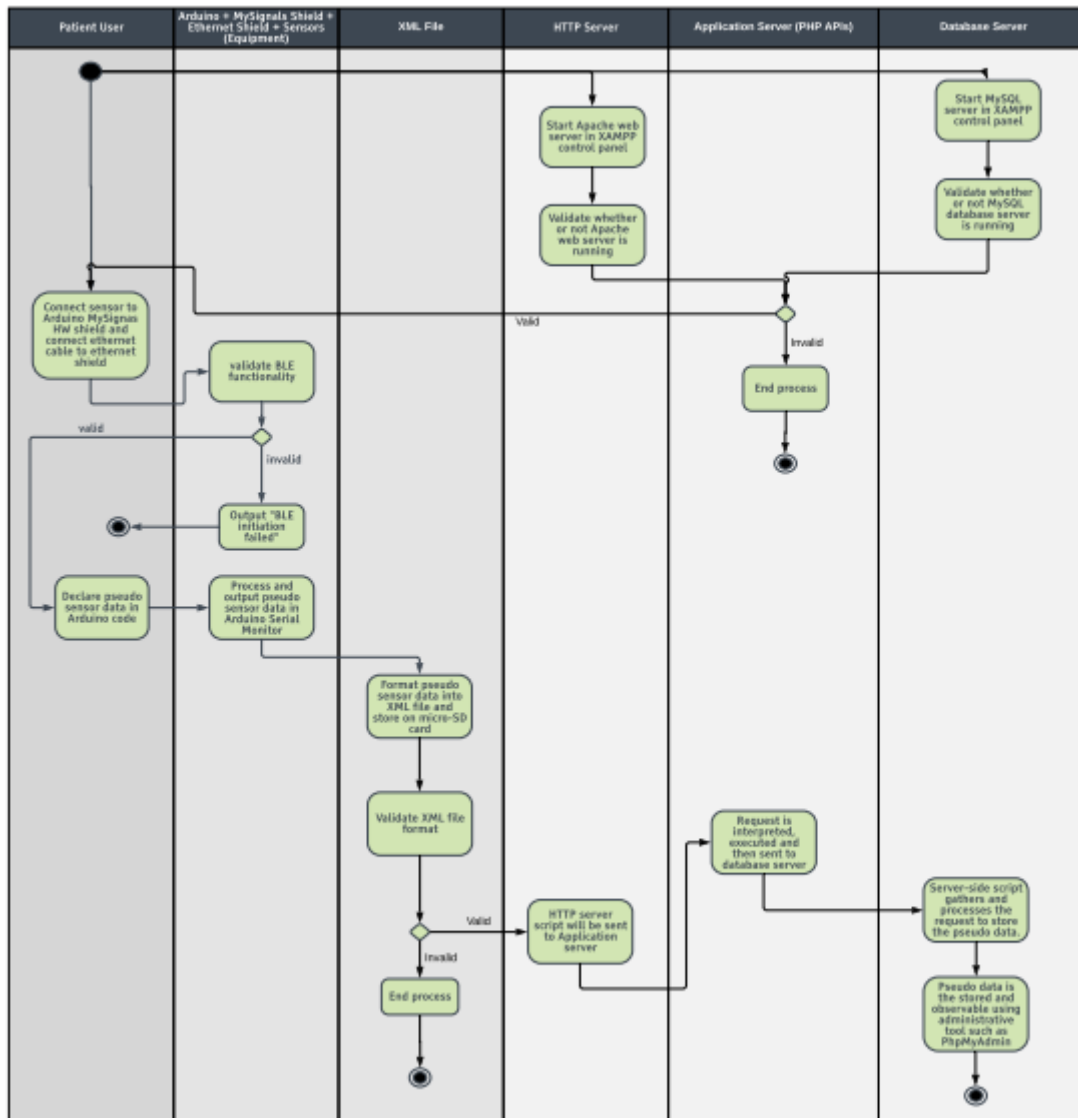


8.7 Software Modelling Diagrams in UML for System Design B

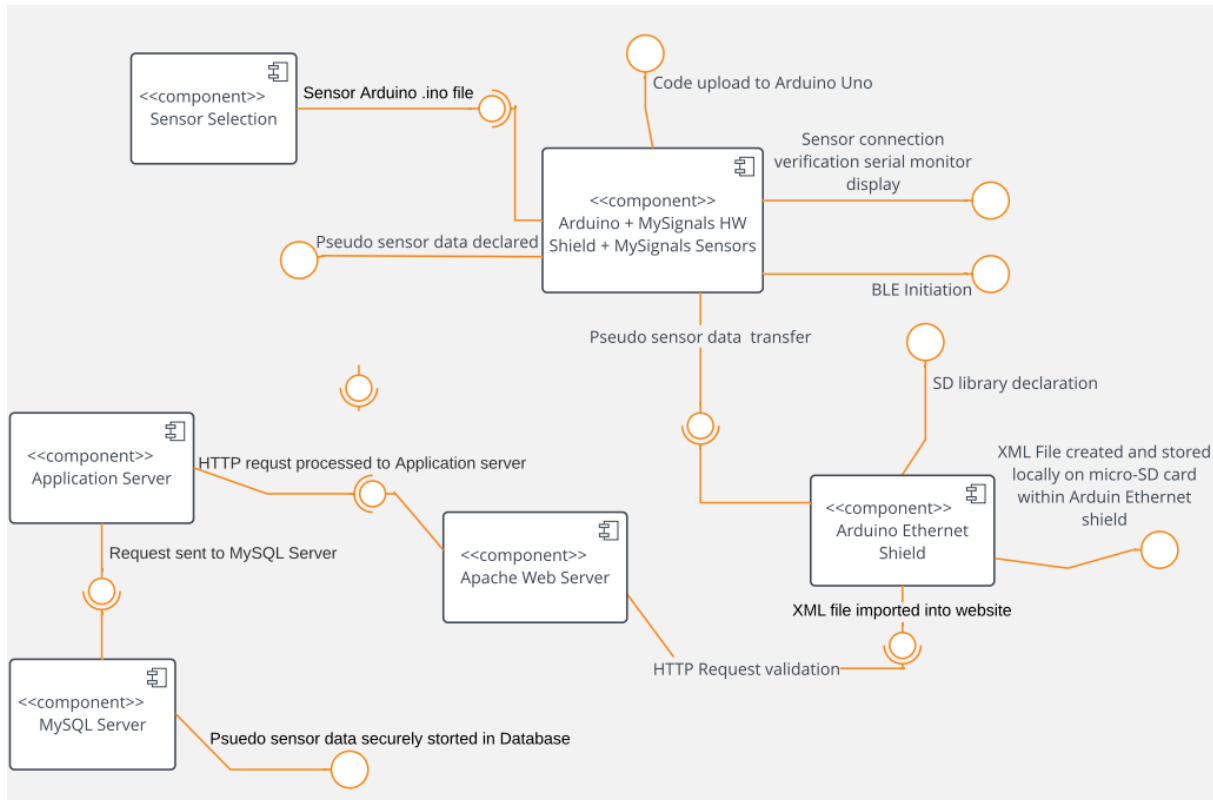
8.7.1 Use Case Diagram



8.7.2 Activity Diagram



8.7.3 Component Diagram



Design A

8.8 Retrieving SP02 Sensor Data to Serial Monitor

8.8.1 Initialising Libraries and BLE module

```
1. #include <MySignals.h>
2. #include <MySignals_BLE.h>
3.
4. char MAC_SPO2[14] = "00A050072626"; // MAC Address of SP02 sensor
   module
5.
6.
7. uint8_t available_spo2 = 0; //variable declared for available
   SP02
8. uint8_t connected_spo2 = 0;
9. uint8_t connection_handle_spo2 = 0;
10. uint8_t pulse_spo2 = 0;
11. uint8_t spo2 = 0;
12.
13. #define SPO2_HANDLE 15
14.
15. void setup()
16. {
17.     MySignals.begin(); //initiates the mysignals HW shield
```

```

18.
19.     Serial.begin(115200); //BAUD rate of arduino set to 115200
20.
21.     MySignals.initSensorUART();
22.     MySignals.enableSensorUART(BLE); //enables BLE module
23.
24.     MySignals_BLE.hardwareReset();
25.     MySignals_BLE.initialize_BLE_values();
26.
27.
28.     if (MySignals_BLE.initModule() == 1)
29.     {
30.
31.         if (MySignals_BLE.sayHello() == 1)
32.         {
33.             MySignals.println("BLE init ok");
34.         }
35.         else
36.         {
37.             MySignals.println("BLE init fail");
38.
39.             while (1)
40.             {
41.             };
42.         }
43.     }
44.     else
45.     {
46.         MySignals.println("BLE init fail");
47.
48.         while (1)
49.         {
50.         };

```

8.8.2 Connecting and Initiating SP02 Sensor

```

51.     available_spo2 = MySignals_BLE.scanDevice(MAC_SPO2, 1000,
    TX_POWER_MAX); //function implemented to scan for available spo2
    sensor devices
52.
53.     MySignals.disableMuxUART();
54.     Serial.print("SPO2 devices available:");
55.     Serial.println(available_spo2); //number of available devices
    corresponding to the MAC Address provided is printed out into
    the serial monitor
56.     MySignals.enableMuxUART();
57.
58.     if (available_spo2 == 1)

```

```

59.     {
60.         MySignals.disableMuxUART();
61.         Serial.println("SPO2 found.Connecting");
62.         MySignals.enableMuxUART();
63.
64.
65.         if (MySignals_BLE.connectDirect(MAC_SPO2) == 1) //MySignals
            shield BLE module attempts to connect to SPO2 sensor with
            declared MAC value
66.         {
67.             connected_spo2 = 1;
68.             connection_handle_spo2 = MySignals_BLE.connection_handle;
69.
70.             MySignals.println("Connected");
71.
72.             delay(6000); //delay of 6 seconds added to ensure
            connection is established
73.
74.             char attributeData[1] =
75.             {
76.                 0x01
77.             };
78.
79.             if (MySignals_BLE.attributeWrite(connection_handle_spo2,
                SPO2_HANDLE, attributeData, 1) == 0) //To connect to the SPO2,
                the SPO2_HANDLE must write "1"
80.             {
81.                 unsigned long previous = millis(); //unsigned long
                    allows for the storage of 32 bit integers and millis() fnc allows
                    for time to be counted for approx 50 days.
82.                 do
83.                 {
84.                     if (MySignals_BLE.waitEvent(1000) ==
                        BLE_EVENT_ATTCLIENT_ATTRIBUTE_VALUE)
85.                     {
86.
87.
88.                         char attributeData[1] = { 0x00 };
89.
90.
91.                         MySignals_BLE.attributeWrite(connection_handle_spo2, SPO2_HANDLE,
                            attributeData , 1);
92.
93.                         uint8_t pulse_low = MySignals_BLE.event[12];
94.                         pulse_low &= 0b01111111;
95.
96.                         uint8_t pulse_high = MySignals_BLE.event[11];
97.                         pulse_high &= 0b01000000;

```

```

97.
98.         if (pulse_high == 0)
99.         {
100.             pulse_spo2 = pulse_low;
101.         }
102.
103.         if (pulse_high == 0b01000000)
104.         {
105.             pulse_spo2 = pulse_low + 0b10000000;
106.         }
107.
108.         spo2 = MySignals_BLE.event[13];
109.         spo2 &= 0b01111111;

```

8.8.3 Outputting SP02 and Pulse Values

```

110.         if ((pulse_spo2 >= 25) && (pulse_spo2 <= 250)
111.             && (pulse_spo2 >= 35) && (pulse_spo2 <= 100))
112.         {
113.             MySignals.disableMuxUART();
114.
115.             Serial.println();
116.             Serial.print(F("SpO2: "));
117.             Serial.print(spo2);
118.             Serial.print(F("% "));
119.             Serial.print(F("Pulse: "));
120.             Serial.print(pulse_spo2);
121.             Serial.println(F("ppm "));
122.
123.             uint16_t errorCode =
MySignals_BLE.disconnect(connection_handle_spo2);
124.
125.             Serial.print(F("Disconnecting error code: "));
126.             Serial.println(errorCode, HEX);
127.
128.             MySignals.enableMuxUART();
129.             connected_spo2 = 0;
130.
131.         }
132.     }
133. }
134. while ((connected_spo2 == 1) && ((millis() - previous)
    < 10000));
135.
136.     connected_spo2 = 0;
137.
138. }
139. else

```

```

140.     {
141.         MySignals.println("Error subscribing");
142.     }
143. }
144. else
145. {
146.     connected_spo2 = 0;
147.
148.     MySignals.println("Not Connected");
149. }
150. }
151. else if (available_spo2 == 0)
152. {
153.     //Do nothing
154. }
155. else
156. {
157.
158.     MySignals_BLE.hardwareReset();
159.     MySignals_BLE.initialize_BLE_values();
160.     delay(100);
161.
162. }
163.
164.
165.     delay(1000);

```

8.8.4 Incorporation of SCTID code

```

166. uint32_t normal_SP02 = 167025001; //normal SP02 level
167. uint32_t abnormal_SP02 = 167025000; //abnormal SP02 level
168. //uint64_t pulse_Snomedct = 297701000000109; //pulse finding
169.
170. delay(10000);
171.
172. if((spo2 >=94) && (spo2 <= 100))
173. {
174.     Serial.println(normal_SP02);
175. }
176. else
177. {
178.     Serial.println(abnormal_SP02);
179. }
180.
181. }
182.
183. void loop()
184. {

```

```
185. }
```

8.9 Securely Transferring Sensor Data to MySQL

8.9.1 GET Request

8.9.1.1 *Write_data.php*

```
1. <?php
2.
3. $dbconnect = mysql_connect('localhost','root','');
4. $dbselect = mysql_select_db('project');
5.
6. mysql_query("INSERT INTO sp02 (sp02, Pulse, SNOMED_CT_CODE)
   VALUES ('".mysql_real_escape_string($_GET['sp02'])."',
   '".mysql_real_escape_string($_GET['Pulse'])."',
   '".mysql_real_escape_string($_GET['SNOMED_CT_CODE'])."')");
7.
8.
9. if (mysql_query) {
10.     echo "success";
11. }
12. ?>
```

8.9.1.2 *Arduino HTTP GET Request Using Pseudo Data*

```
1. #include <SPI.h>
2. #include <String.h>
3. #include <Ethernet.h>
4.
5. byte mac[] = { 0x90, 0xA2, 0xDA, 0x00, 0x9B, 0x36 };
6.
7. byte ip[] = { 192, 168, 0, 113};
8.
9. byte servidor[] = { 192, 168, 0, 60 };
10.
11. EthernetServer server(8080);
12. EthernetClient cliente;
13.
14. float sp02 = 98;
15. u_int32 sctid = 167025001;
16.
17. String readString = String(30);
18.
19. unsigned long previousMillis = 0;
20. const long interval = 5000;
21.
22.
```



```

23. void setup() {
24.     Ethernet.begin(mac, ip);
25.     Serial.begin(115200);
26. }
27.
28. void loop() {
29.
30.     EthernetClient client = server.available();
31.     unsigned long currentMillis = millis();
32.     if (currentMillis - previousMillis >= interval) {
33.
34.         previousMillis = currentMillis;
35.
36.         if (cliente.connect(servidor, 8080)) {
37.             Serial.println("connected");
38.
39.             sp02 = sp02 + 20;
40.             sctid = sctid + 20;
41.
42.             cliente.print("GET /arduino/saved.php?");
43.             cliente.print("sp02=");
44.             cliente.print(sp02);
45.             cliente.print("&snomedct=");
46.             cliente.print(sctid);
47.
48.
49.             Serial.print("sp02 = ");
50.             Serial.println(sp02);
51.             Serial.print("SNOMED_CT_CODE = ");
52.             Serial.println(sctid);
53.
54.             cliente.println("HTTP/1.1");
55.             cliente.println("Host: http://192.168.0.60:8080");
56.             cliente.println("connection: close");
57.             cliente.println();
58.             cliente.println();
59.             cliente.stop();
60.
61.         }
62.         else {
63.             Serial.println("connection failed");
64.
65.             cliente.stop();
66.         }
67.     }
68. }

```

8.9.2 POST Request

8.9.2.1 config.php

```
1. <?php
2.
3. $dbhost = 'localhost'; //database server
4.
5. $dbuser = 'root'; //database user
6. $dbpass = ''; //database password
7. $dbname = 'project_data'; //database name
8.
9. $conn = mysqli_connect($dbhost, $dbuser, $dbpass) or die ('Error
    connecting to mysql'); //mysql_connect used to connect to mysql
    database
10. ?>
```

8.9.2.2 update.php

```
1. <?php
2. require("config.php"); //implements config php file to ensure
3.                               //connection to database is
    established
4.
5.
6. $query = "INSERT INTO 442082004_sp02(sctid:sp02)
7.           VALUES('$ _POST[sp02]')"; //query used to insert
8.           //POST request used
9.           //data into table inside given database
10.
11. if(!@mysqli_query($query))
12. {
13.     echo "&Answer; SQL Error - ".mysqli_error(); //if
    mysql_query() is not executed
14.
    //it should be returned SQL Error.
15.     return;
16. }
17. ?>
```

8.9.2.3 Arduino HTTP POST Request Test

```
1. #include <Ethernet.h>
2. #include <SPI.h>
3.
```

```

4. byte mac[] = { 0x90,0xA2,0xDA,0x0D,0x0D,0xB1}; //allocate
   ethernet shield with MAC
5. EthernetClient client;
6. IPAddress ip(192, 168, 0, 113); //run and find ethernet shield ip
   address
7.
8. // EDIT: 'Server' address to match the domain of the computer I
   am using
9. char server[] = "192.168.0.60"; //method of finding will be
   explained in the relevant section
10.
11. // This is the data that will be passed into the POST and
   matches your mysql column
12.
13. uint32_t snomedct = 100000000;
14.
15. int yourarduinodata = 100;
16. String yourdatacolumn = "sp02=";
17. String sp02;
18.
19.
20.
21. void setup()
22. {
23.   Serial.begin(9600);
24.   Ethernet.begin(mac,ip);
25.   delay(1000);
26.   Serial.println("connecting...");
27.   postData();
28.
29.   // Combine yourdatacolumn header (yourdata=) with the data
   recorded from your arduino
30.   // (yourarduinodata) and package them into the String
   yourdata which is what will be
31.   // sent in your POST request
32.   sp02 = yourdatacolumn + snomedct + ": " + yourarduinodata;
33.
34.   // If there's a successful connection, send the HTTP POST
   request
35.   if (client.connect(server, 8080)) {
36.     Serial.println("connecting...");
37.
38.     // EDIT: The POST 'URL' to the location of your
   insert_mysql.php on your web-host
39.     client.println("POST /update/update.php HTTP/1.1");
40.
41.     // EDIT: 'Host' to match your domain
42.     client.println("Host:192.168.0.60:8080");

```

```

43.     client.println("User-Agent: Arduino/1.0");
44.     client.println("Connection: close");
45.     client.println("Content-Type: application/x-www-form-
        urlencoded;");
46.     client.print("Content-Length: ");
47.     client.println(sp02.length());
48.     client.println();
49.     client.println(sp02);
50.
51. }
52. else {
53.     // If you couldn't make a connection:
54.     Serial.println("Connection failed");
55.     Serial.println("Disconnecting.");
56.     client.stop();
57. }
58. }
59.
60. void loop() {
61. }
62.
63. void postData() {
64. }

```

8.9.2.4 HTTP POST Request Using MySignals Sensor Data

```

1. #include <MySignals.h>
2. #include <MySignals_BLE.h>
3. #include <Ethernet.h>
4. #include <SPI.h>
5.
6.
7. // Write here the MAC address of BLE device to find
8. char MAC_SPO2[14] = "00A050072626";
9.
10. byte mac[] = { 0x90,0xA2,0xDA,0x0D,0x0D,0xB1}; //Replace with
    your Ethernet shield MAC
11. EthernetClient client;
12. IPAddress ip(192, 168, 0, 113); //ethernet ip address attatched
    to my ethernet shield
13.
14. // EDIT: 'Server' address to match your domain
15. char server[] = "192.168.0.60"; // This could also be
    192.168.1.18/~me if you are running a server on your computer on
    a local network.
16.
17.

```

```

18.  uint8_t available_spo2 = 0;
19.  uint8_t connected_spo2 = 0;
20.  uint8_t connection_handle_spo2 = 0;
21.  uint8_t pulse_spo2 = 0;
22.  uint8_t spo2 = 0;
23.
24.  String yourdatacolumn = "sp02=";
25.  String spo2level;
26.
27.  #define SPO2_HANDLE 15
28.
29.  void setup()
30.  {
31.      MySignals.begin();
32.
33.      Serial.begin(115200);
34.      Ethernet.begin(mac, ip);
35.      delay(1000);
36.      Serial.println("connnecting....");
37.      postData();
38.
39.      MySignals.initSensorUART();
40.      MySignals.enableSensorUART(BLE);
41.
42.      MySignals_BLE.hardwareReset();
43.      MySignals_BLE.initialize_BLE_values();
44.
45.
46.      if (MySignals_BLE.initModule() == 1)
47.      {
48.
49.          if (MySignals_BLE.sayHello() == 1)
50.          {
51.              MySignals.println("BLE init ok");
52.              Serial.println("BLE init ok");
53.          }
54.          else
55.          {
56.              MySignals.println("BLE init fail");
57.              Serial.println("BLE init fail");
58.
59.              while (1)
60.              {
61.              };
62.          }
63.      }
64.      else
65.      {

```

```

66.     MySignals.println("BLE init fail");
67.     Serial.println("BLE init fail");
68.
69.     while (1)
70.     {
71.     };
72.     }
73.     delay(5000);
74.     available_spo2 = MySignals_BLE.scanDevice(MAC_SPO2, 1000,
TX_POWER_MAX);
75.     delay(5000);
76.     MySignals.disableMuxUART();
77.     Serial.print("SPO2 available:");
78.     Serial.println(available_spo2);
79.     MySignals.enableMuxUART();
80.
81.     delay(5000);
82.
83.     if (available_spo2 == 1)
84.     {
85.         MySignals.disableMuxUART();
86.         Serial.println("SPO2 found.Connecting");
87.         MySignals.enableMuxUART();
88.
89.
90.         if (MySignals_BLE.connectDirect(MAC_SPO2) == 1)
91.         {
92.             connected_spo2 = 1;
93.             connection_handle_spo2 = MySignals_BLE.connection_handle;
94.
95.             MySignals.println("Connected");
96.
97.             Serial.println("connected to spo2 sensor");
98.
99.             delay(6000);
100.
101.             //To subscribe the spo2 measure write "1" in SPO2_HANDLE
102.             char attributeData[1] =
103.             {
104.                 0x01
105.             };
106.
107.             if (MySignals_BLE.attributeWrite(connection_handle_spo2,
SPO2_HANDLE, attributeData, 1) == 0)
108.             {
109.                 unsigned long previous = millis();
110.                 do
111.                 {

```

```

112.         if (MySignals_BLE.waitEvent(1000) ==
BLE_EVENT_ATTCLIENT_ATTRIBUTE_VALUE)
113.         {
114.
115.
116.             char attributeData[1] = { 0x00 };
117.
118.             MySignals_BLE.attributeWrite(connection_handle_spo2, SPO2_HANDLE,
attributeData , 1);
119.
120.             uint8_t pulse_low = MySignals_BLE.event[12];
121.             pulse_low &= 0b01111111;
122.
123.             uint8_t pulse_high = MySignals_BLE.event[11];
124.             pulse_high &= 0b01000000;
125.
126.             if (pulse_high == 0)
127.             {
128.                 pulse_spo2 = pulse_low;
129.             }
130.
131.             if (pulse_high == 0b01000000)
132.             {
133.                 pulse_spo2 = pulse_low + 0b10000000;
134.             }
135.
136.             spo2 = MySignals_BLE.event[13];
137.             spo2 &= 0b01111111;
138.
139.             if ((pulse_spo2 >= 25) && (pulse_spo2 <= 250)
&& (pulse_spo2 >= 35) && (pulse_spo2 <= 100))
140.             {
141.
142.                 MySignals.disableMuxUART();
143.
144.                 Serial.println();
145.                 Serial.print(F("SpO2: "));
146.                 Serial.print(spo2);
147.                 Serial.print(F("% "));
148.                 Serial.print(F("Pulse: "));
149.                 Serial.print(pulse_spo2);
150.                 Serial.println(F("ppm "));
151.
152.                 uint16_t errorCode =
MySignals_BLE.disconnect(connection_handle_spo2);
153.
154.                 Serial.print(F("Disconnecting error code: "));
155.                 Serial.println(errorCode, HEX);

```

```

156.
157.         MySignals.enableMuxUART();
158.         connected_spo2 = 0;
159.
160.     }
161. }
162. }
163.     while ((connected_spo2 == 1) && ((millis() - previous)
< 10000));
164.
165.         connected_spo2 = 0;
166.
167.     }
168.     else
169.     {
170.         MySignals.println("Error subscribing");
171.     }
172. }
173. else
174. {
175.     connected_spo2 = 0;
176.
177.     MySignals.println("Not Connected");
178. }
179. }
180. else if (available_spo2 == 0)
181. {
182.     //Do nothing
183. }
184. else
185. {
186.
187.     MySignals_BLE.hardwareReset();
188.     MySignals_BLE.initialize_BLE_values();
189.     delay(100);
190.
191. }
192.
193. uint32_t normal_SP02 = 167025001; //normal SP02 level
194. uint32_t abnormal_SP02 = 167025000; //abnormal SP02 level
195. //uint64_t pulse_Snomedct = 297701000000109; //pulse finding
196.
197. delay(10000);
198.
199.
200. if((spo2 >=94) && (spo2 <= 100))
201. {
202.     Serial.println(normal_SP02);

```



```

203.     sp02level = yourdatacolumn + normal_SP02 + ": " + spo2;
204. }
205. else
206. {
207.     Serial.println(abnormal_SP02);
208.     sp02level = yourdatacolumn + abnormal_SP02 + ": " + spo2;
209. }
210.
211.     if (client.connect(server, 8080)) {
212.         Serial.println("connecting to MySQL database...");
213.
214.         // EDIT: The POST 'URL' to the location of your
            insert_mysql.php on your web-host
215.         client.println("POST /update/update.php HTTP/1.1");
216.
217.         // EDIT: 'Host' to match your domain
218.         client.println("Host:192.168.0.60:8080");
219.         client.println("User-Agent: Arduino/1.0");
220.         client.println("Connection: close");
221.         client.println("Content-Type: application/x-www-form-
            urlencoded;");
222.         client.print("Content-Length: ");
223.         client.println(sp02level.length());
224.         client.println();
225.         client.println(sp02level);
226.
227.     }
228.     else {
229.         // If you couldn't make a connection:
230.         Serial.println("Connection failed");
231.         Serial.println("Disconnecting.");
232.         client.stop();
233.     }
234.
235.     delay(1000);
236. }
237.
238. void loop()
239. {
240. }
241.
242. void postData() {
243. }

```

Design B

8.10 Formatting of Sensor Data into XML File

8.10.1 Initialising SD Libraries and Micro-SD Card

```
1. #include <SPI.h>
2. #include <SD.h>
3.
4. const int chipSelect = 4;
5.
6. uint32_t snomed_ct_sp02 = 100000;
7. uint32_t sp02 = 100;
8. uint32_t snomed_ct_pulse = 20000;
9. uint32_t pulse = 100;
10.
11. void setup() {
12.     // Open serial communications and wait for port to open:
13.     Serial.begin(9600);
14.     while (!Serial) {
15.         ; // wait for serial port to connect. Needed for native USB
           port only
16.     }
17.
18.     Serial.print("Initializing SD card...");
19.
20.     // see if the card is present and can be initialized:
21.     if (!SD.begin(chipSelect)) {
22.         Serial.println("Card failed, or not present");
23.         // don't do anything more:
24.         return;
25.     }
26.     Serial.println("card initialized.");
```

8.10.2 Creation of the XML File

```
27.     // Create/Open file
28.     SD.remove("sensor.xml");
29.
30.     File myFile = SD.open("sensor.xml", FILE_WRITE);
31.
32.     // if the file is available, write to it:
33.     if (myFile) {
34.
```

```

35.     myFile.println("<patient>");
36.     myFile.println("<sensor>");
37.
38.     myFile.println();
39.
40.     myFile.print("<sp02>");
41.     myFile.print(snomed_ct_sp02);
42.     myFile.print(":");
43.     myFile.print(sp02);
44.
45.     myFile.print("</sp02>");
46.
47.     myFile.print("</sensor>");
48.
49.     myFile.println();
50.
51.     myFile.print("</patient>");
52.
53.     myFile.close();
54.     // print to the serial port too:
55.     Serial.println(snomed_ct_sp02);
56.     Serial.println(sp02);
57.     Serial.println(snomed_ct_pulse);
58.     Serial.println(pulse);
59. }
60. // if the file isn't open, pop up an error:
61. else {
62.     Serial.println("error opening sensor.xml");
63. }
64. }
65.
66. void loop() {
67.
68. }

```

8.11 Securely Inserting XML File into MySQL Database

8.11.1 PHP APIs

8.11.1.1 *import.php*

```

1. <?php
2. //import.php
3. sleep(3);
4. $output = '';
5.
6. if(isset($_FILES['file']['name']) && $_FILES['file']['name'] !=
    '')

```

```

7. {
8.   $valid_extension = array('XML'); // file extension declared
9.   $file_data = explode('.', $_FILES['file']['name']); //file
      string broken down
10.   $file_extension = end($file_data);
11.   if(in_array($file_extension, $valid_extension))
12.   {
13.     $data = simplexml_load_file($_FILES['file']['tmp_name']);
14.     $connect = new
      PDO('mysql:host=localhost;dbname=mysignals','root', '');
15.     // connection is established - host ----- database name
      - user - password
16.
17.     $connect->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
18.     $connect->setAttribute(PDO::ATTR_ERRMODE,
      PDO::ERRMODE_EXCEPTION);
19.
20.     $query = "
21.     INSERT INTO 442082004_sp02
22.     (sctid:sp02)
23.     VALUES(:sp02);
24.     ";
25.     // variable query is created to process the insert function
      into the database table
26.     $statement = $connect->prepare($query);
27.     for($i = 0; $i < count($data); $i++)
28.     {
29.       $statement->execute(
30.         array(
31.           ':sp02' => $data->sensor[$i]->sp02, // sp02 variable
      instantiated into insert variable function
32.           // ':sp02' => $data->sensor[$i]->sp02,
33.           // ':snomed_ct_pulse' => $data->sensor[$i]-
      >snomed_ct_pulse,
34.           // ':pulse' => $data->sensor[$i]->pulse,
35.         )
36.       );
37.
38.     }
39.     $result = $statement->fetchAll();
40.     if(isset($result))
41.     {
42.       $output = '<div class="alert alert-success">Import Data
      Done</div>';
43.     }
44.   }
45.   else
46.   {

```

```

47.     $output = '<div class="alert alert-warning">Invalid
      File</div>';
48.   }
49. }
50. else
51. {
52.     $output = '<div class="alert alert-warning">Please Select XML
      File</div>';
53. }
54.
55. echo $output;
56.
57. ?>

```

8.11.1.2 index.php

8.11.1.2.1 HTML Code

```

1. <?php
2. //index.php
3. ?>
4. <!DOCTYPE html>
5. <html>
6.   <head>
7.     <title>Import XML Data into Mysql Table Using Ajax PHP</title>
8.     <script
      src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.mi
      n.js"></script>
9.     <link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstr
      ap.min.css" />
10.   </head>
11.   <body>
12.     <br />
13.     <div class="container">
14.       <div class="row">
15.         <h2 align="center">IMPORT ARDUINO SENSOR DATA TO MYSQL
          DATABASE</h2>
16.         <br />
17.         <div class="col-md-9" style="margin:0 auto; float:none;">
18.           <span id="message"></span>
19.           <form method="post" id="import_form"
            enctype="multipart/form-data">
20.             <div class="form-group">
21.               <label>Select XML File</label>
22.               <input type="file" name="file" id="file" />
23.             </div>
24.             <br />

```

```

25.         <div class="form-group">
26.             <input type="submit" name="submit" id="submit"
                class="btn btn-info" value="Import" />
27.         </div>
28.     </form>
29. </div>
30. </div>
31. </div>
32. </body>
33. </html>

```

8.11.1.2.2 jQuery Code

```

34. <script>
35. $(document).ready(function() {
36.     $('#import_form').on('submit', function(event) {
37.         event.preventDefault();
38.
39.         $.ajax({
40.             url:"import.php",
41.             method:"POST",
42.             data: new FormData(this),
43.             contentType:false,
44.             cache:false,
45.             processData:false,
46.             beforeSend:function() {
47.                 $('#submit').attr('disabled','disabled'),
48.                 $('#submit').val('Importing...');
49.             },
50.             success:function(data)
51.             {
52.                 $('#message').html(data);
53.                 $('#import_form')[0].reset();
54.                 $('#submit').attr('disabled', false);
55.                 $('#submit').val('Import');
56.             }
57.         })
58.
59.         setInterval(function() {
60.             $('#message').html('');
61.         }, 5000);
62.
63.     });
64. });
65. </script>

```

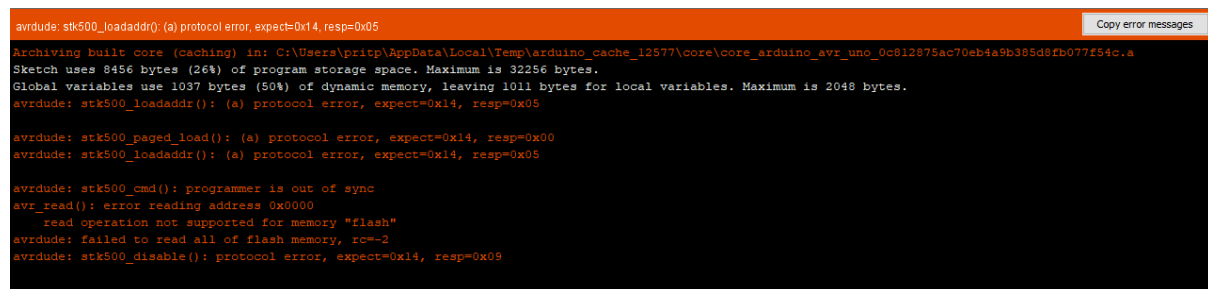
8.11.1.2.3 Webpage Created to Insert XML File Data into MySQL Database



The screenshot shows a web browser window with the title 'Import XML Data into Mysql...'. The main heading is 'IMPORT ARDUINO SENSOR DATA TO MYSQL DATABASE'. Below the heading, there is a section titled 'Select XML File' with a text input field and a 'Browse...' button. At the bottom of this section is a blue 'Import' button.

Inserting

8.12 Arduino Cable error



```
avrdude: stk500_loadaddr(): (a) protocol error, expect=0x14, resp=0x05
Archiving built core (caching) in: C:\Users\priti\AppData\Local\Temp\arduino_cache_12577\core\core_arduino_avr_uno_0c812875ac70eb4a9b385d8fb077f54c.a
Sketch uses 8456 bytes (26%) of program storage space. Maximum is 32256 bytes.
Global variables use 1037 bytes (50%) of dynamic memory, leaving 1011 bytes for local variables. Maximum is 2048 bytes.
avrdude: stk500_loadaddr(): (a) protocol error, expect=0x14, resp=0x05

avrdude: stk500_page_load(): (a) protocol error, expect=0x14, resp=0x00
avrdude: stk500_loadaddr(): (a) protocol error, expect=0x14, resp=0x05

avrdude: stk500_cmd(): programmer is out of sync
avr_read(): error reading address 0x0000
read operation not supported for memory "flash"
avrdude: failed to read all of flash memory, rc=-2
avrdude: stk500_disable(): protocol error, expect=0x14, resp=0x09
```