|  |
| --- |
| Technical Report |
| GLOBLOCK |
| BSHCE4 NM BSc in Computing |

|  |
| --- |
| Alex Quigley - x10205691  5/19/2014 |



# Document Control

## Project Information

|  |  |
| --- | --- |
| Project Title: | Globlock |
| Course Code: | BSHCE4 |
| Course Title: | BSc (Hons) in Computing |
| Course Specialisation: | Network and Mobile Technologies |
| Student Name: | Alex Quigley |
| Student Number: | 10205691 |
| Date: | 19th May 2014 |

## Project Information

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Scope | Prepared | Date |
| 1.01 | Create | Alex Quigley | 19th May 2014 |

## Distribution

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Title | Version | Date |
| Paul Hayes | Project Supervisor | 1.01 |  |
| Jonathan McCarthy | Module Co-Ordinator | 1.01 |  |

## Definitions, Acrobyms and Abbreviations

See appendix <<x>>

Contents

[Document Control 1](#_Toc388311860)

[Project Information 1](#_Toc388311861)

[Project Information 1](#_Toc388311862)

[Distribution 1](#_Toc388311863)

[Definitions, Acrobyms and Abbreviations 1](#_Toc388311864)

[Executive Summary 7](#_Toc388311865)

[Introduction 8](#_Toc388311866)

[Intended Audience 8](#_Toc388311867)

[Background 8](#_Toc388311868)

[Initial Idea 8](#_Toc388311869)

[Repositories 8](#_Toc388311870)

[NFC/RFID 8](#_Toc388311871)

[Globlock Intended Audience 8](#_Toc388311872)

[Physical Tokens 8](#_Toc388311873)

[Aims 9](#_Toc388311874)

[Concurrency Control 9](#_Toc388311875)

[Tokenisation 9](#_Toc388311876)

[Repository Management 9](#_Toc388311877)

[Technologies Used 10](#_Toc388311878)

[Hardware 10](#_Toc388311879)

[Arduino 10](#_Toc388311880)

[RFID 10](#_Toc388311881)

[ID-12LA 10](#_Toc388311882)

[ID12LA Breakout Board 10](#_Toc388311883)

[Client PC and Server Host 11](#_Toc388311884)

[Software 11](#_Toc388311885)

[Github 11](#_Toc388311886)

[Trello 11](#_Toc388311887)

[Development Environments 11](#_Toc388311888)

[Arduino 11](#_Toc388311889)

[Visual Studio 11](#_Toc388311890)

[Notepad++ 11](#_Toc388311891)

[Supporting Frameworks and Technologies 11](#_Toc388311892)

[XAMPP 11](#_Toc388311893)

[Apache HTTP 11](#_Toc388311894)

[PHP 11](#_Toc388311895)

[Database Technologies 12](#_Toc388311896)

[MySQL 12](#_Toc388311897)

[SQLite 12](#_Toc388311898)

[Programming Languages 12](#_Toc388311899)

[Client Application 12](#_Toc388311900)

[C# 12](#_Toc388311901)

[JSON 12](#_Toc388311902)

[Client Device (Arduino) 12](#_Toc388311903)

[Server Side API 12](#_Toc388311904)

[Management Website 12](#_Toc388311905)

[Structure 13](#_Toc388311906)

[System 14](#_Toc388311907)

[Requirements 14](#_Toc388311908)

[Features Overview 14](#_Toc388311909)

[User Permissions Management 14](#_Toc388311910)

[Repository and Document Management 14](#_Toc388311911)

[Globe Management 15](#_Toc388311912)

[Reporting 15](#_Toc388311913)

[System Features 15](#_Toc388311914)

[User Management 15](#_Toc388311915)

[Document Management 15](#_Toc388311916)

[Document Upload 15](#_Toc388311917)

[Document Download 15](#_Toc388311918)

[Globe Management 15](#_Toc388311919)

[Create Globe 15](#_Toc388311920)

[Archive Glove 15](#_Toc388311921)

[Globe Access 15](#_Toc388311922)

[Open Globe 16](#_Toc388311923)

[Reporting 16](#_Toc388311924)

[Non-Functional Requirements 16](#_Toc388311925)

[Performance/Response time requirement 16](#_Toc388311926)

[Recoverability & Reliability Requirement 16](#_Toc388311927)

[Maintainability & Re-usability Requirement 17](#_Toc388311928)

[Design and Architecture 18](#_Toc388311929)

[Overview 18](#_Toc388311930)

[Block Device 18](#_Toc388311931)

[Client Application 20](#_Toc388311932)

[Hosted Server 20](#_Toc388311933)

[Management Web Application 21](#_Toc388311934)

[Globlock API 21](#_Toc388311935)

[Implementation 22](#_Toc388311936)

[Database Schema 22](#_Toc388311937)

[Server Side 22](#_Toc388311938)

[Server DB Schema in SQL format 24](#_Toc388311939)

[Client Side 25](#_Toc388311940)

[Client DB Schema in SQL format 25](#_Toc388311941)

[The Globlock API 26](#_Toc388311942)

[Class diagram overview 26](#_Toc388311943)

[Package File Types 26](#_Toc388311944)

[Classes 26](#_Toc388311945)

[Handler Concept 26](#_Toc388311946)

[Broker Concept 26](#_Toc388311947)

[Support 27](#_Toc388311948)

[Test Files 27](#_Toc388311949)

[API & Package 27](#_Toc388311950)

[API Main File 27](#_Toc388311951)

[Package 27](#_Toc388311952)

[Request Broker 27](#_Toc388311953)

[Construction 28](#_Toc388311954)

[Sanitation 28](#_Toc388311955)

[JSON 29](#_Toc388311956)

[API Interaction 29](#_Toc388311957)

[Point of Entry 29](#_Toc388311958)

[Handling Requests 29](#_Toc388311959)

[Request Types 30](#_Toc388311960)

[HANDSHAKE 30](#_Toc388311961)

[Request Table 31](#_Toc388311962)

[Sessions 31](#_Toc388311963)

[SET, FORCE & DROP 32](#_Toc388311964)

[PULL & PUSH 32](#_Toc388311965)

[Client Applications 34](#_Toc388311966)

[BrokerManager 34](#_Toc388311967)

[Client Application Class Diagram Overview 35](#_Toc388311968)

[Client GUI 35](#_Toc388311969)

[Management Web Application 36](#_Toc388311970)

[Homepage 36](#_Toc388311971)

[Globes 37](#_Toc388311972)

[Documents 37](#_Toc388311973)

[Groups 38](#_Toc388311974)

[Users 38](#_Toc388311975)

[Testing & Error handling 39](#_Toc388311976)

[Error handling 39](#_Toc388311977)

[Exceptions Caught and Thrown 40](#_Toc388311978)

[Buffering 41](#_Toc388311979)

[User Testing Evaluation 42](#_Toc388311980)

[Conclusions 43](#_Toc388311981)

[API 43](#_Toc388311982)

[Frameworks 43](#_Toc388311983)

[Arduino 43](#_Toc388311984)

[Apprehensions 43](#_Toc388311985)

[Time Management 43](#_Toc388311986)

[Further development or research 45](#_Toc388311987)

[Research 45](#_Toc388311988)

[References 46](#_Toc388311989)

[Appendix A - Project Proposal 47](#_Toc388311990)

[Appendix B - Requirement Specification 48](#_Toc388311991)

[Appendix C - User Manual 49](#_Toc388311992)

[Appendix C - Monthly Journals 50](#_Toc388311993)

[Month: September 50](#_Toc388311994)

[My Achievements 50](#_Toc388311995)

[My Reflection 50](#_Toc388311996)

[Intended Changes 50](#_Toc388311997)

[Month: November 50](#_Toc388311998)

[My Achievements 50](#_Toc388311999)

[My Reflection 51](#_Toc388312000)

[Intended Changes 51](#_Toc388312001)

[Month: December 51](#_Toc388312002)

[My Achievements 51](#_Toc388312003)

[My Reflection 51](#_Toc388312004)

[Intended Changes 52](#_Toc388312005)

[Month: January 52](#_Toc388312006)

[My Achievements 52](#_Toc388312007)

[My Reflection 52](#_Toc388312008)

[Intended Changes 52](#_Toc388312009)

[Month: February 52](#_Toc388312010)

[My Achievements 52](#_Toc388312011)

[My Reflection 52](#_Toc388312012)

[Intended Changes 52](#_Toc388312013)

[Month: March 52](#_Toc388312014)

[My Achievements 52](#_Toc388312015)

[My Reflection 53](#_Toc388312016)

[Intended Changes 53](#_Toc388312017)

[Month: April 53](#_Toc388312018)

[My Achievements 53](#_Toc388312019)

[My Reflection 53](#_Toc388312020)

[Intended Changes 53](#_Toc388312021)

[Appendix F - Arduino Schematic 54](#_Toc388312022)

[Other Material Used 55](#_Toc388312023)

# Executive Summary

Repositories are often very complicated to use for untrained or unskilled professionals. They come with a large overheard, and require a significant amount of time to implement and use. Repositories have been part and parcel of the development environment since revision and historical code management become standard practice.

It is clear that the benefits of a repository, holds value in any organisation and not just at a development level. Repositories come at a cost. The cost is an additional layer of complexity and the requirement for implementation and maintenance on the system.

Without a repository, there can be many different versions of documents. Multiple uncontrolled revisions with changes overlapping and causing confusion, risk and loss, can lead to a very costly experience.

Globlock is the solution that takes away the difficulty and complication by providing a facility for documents to be managed and archived, securely, secretly and successfully, will little or no requirement for tacit knowledge of complicated repository systems.

Added to this the concurrent access controls which prevent multiple version being accessed and edited by multiple users.

The system itself is intuitive, and provides an abstraction of the documents by giving them a physical form. This allows users of the system to feel as if the object in their hand contains the files, instead of the system that it is used with.

This feeling carries weight in the real world and the physical hand over of a document or group of documents is no longer a part of the working environment.

# Introduction

This document is a deliverable for a final year project as part of an Honours Degree in computing with the National College of Ireland.

This document is a technical report for the Globlock project. The report will discuss the background to the products inception from initial idea through to completion.

## Intended Audience

The intended audience for this document includes but is not limited to, the lecturers and faculty members of the National College of Ireland (listed in 'Distribution List' above) as well as external examiners tasked with marking the project.

The document is also intended for potential customers or users of the software and hardware elements of the product, upon project competition, and will also be made available to potential employers as an example of my work, capabilities and skills.

## Background

### Initial Idea

The idea for Globlock came about during the ‘dot conf’ I.T. conference held at the National College of Ireland. I attended the event in 2012 and had been given a small stress ball from Grafton Media as part of the welcome pack. I was researching the ‘Internet of Things’, home automation systems and RFID for personal projects at the time and something struck a chord with me.

### Repositories

Having worked with repositories in the past I have always found them to require technical knowledge to make full use of them and found them to be out of reach of unskilled or untrained I.T. professionals. Particularly around reducing duplication, the potential benefits are quite obvious.

### NFC/RFID

The use of NFC has become increasingly popular among phone manufacturers, payment systems and widespread in supply logistics, products production and manufacturing. The increased tech world’s focus on the ‘Internet of things’, has removed people fears over this type of technology to a certain extent and clearly identified a potential market for IOT, pervasive and ubiquitous products and services.

### Globlock Intended Audience

The purpose of 'Globlock' is to provide a means for unskilled or professionals untrained in I.T., as well as those trained, but without the necessary infrastructure, to gain access to and utilise the benefits of a repository management system.

In order to reduce potential costs to customers, open source, freely available software and architecture was used throughout, both at a software and hardware level.

It is important to note that 'Globlock' is not a replacement for existing repository systems and

services. It is intended for use in environments where knowledge and training in concept and use of

repository systems is unavailable or unsuitable. In particular it is directed at small to medium sized offices where concurrency control and revision reposition is needed or would add value to the business processes.

### **Physical Tokens**

'Globlock' provides a mechanism to tokenise files and projects, and allow a physical abstraction of these digital objects. This physical abstraction will allow strict concurrency control over the associated files/projects.

The software and hardware interfaces required for this abstraction and control are all encapsulated

in 'Globlock'. Projects and their associated documentation can be linked to a physical object, and the system will allow users to review and maintain a repository of versions, review changes, re-assign files or projects to different physical objects, and report on progress and updates

## Aims

### Concurrency Control

Physical possession of globe objects is required, as well as a username and password. This 2 factor authentication aims to deliver a system with robust concurrency control on a single file or group of files, with a revision history and reporting capability. The system would be implemented to support a team of users who would have a business need to prevent the possibility of multiple versions of a file or files in circulation.

### Tokenisation

File or files would be tokenised. Each project would have a physical form; A Globe Object. The soft copy or digital representation of the file on the system (whether they be stored locally or remotely) would be represented by a physical token in the real world, in the form of a small durable ball, however any device capable of rfid communication at 125Mhz would be able to provide this abstraction, so cards, tags, buttons and fobs would allow this physical tokenisation.

By tokenising the file or files, it is then possible to prevent multiple user change access and multiple revisions at any given time. Only the user in physical possession of the token, having granted them access through the system prior to creating the token, would be able to make changes to the file or files.

The physical handoff of the token in itself is also something meaningful at a psychological level such as the passing a baton from one person to another.

### Repository Management

The robust central API delivers a secure and tightly controlled document management system and repository. Each token and their associated files would be supported by administration of user profiles and access rights/permissions, and would allow high level users (super users or admins) to assign files to the physical Globe Objects, review change comments and current status.

# Technologies Used

## Hardware

### Arduino

*“Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists and anyone interested in creating interactive objects or environments.”*

(arduino.cc, 2014)

Arduino was used in place of any other programmable hardware elements as it would allow a level of interaction that would not be possible with other systems. It would also ensure complete control over source code, and allow a high level of extensibility.

Arduino is also open source hardware and the components required for a bare bones hardware implementation could be sourced at a very low cost, and should mass production be required, the hardware elements could be very economical.

Arduino allowed interaction such as handshaking with the device and also allowed a pre formatted response from the device which improved the robustness of the system and prevented potential issues such as slow buffering and serial communication loss. It also allowed for scope for additional components such as LCD displays etc..

### RFID

RFID is core to the entire concept and allows the necessary tokenisation of files by providing a unique identification for each Object which can then be associated with a particular project.

*“Radio frequency identification, or RFID, is a generic term for technologies that use radio waves to automatically identify people or objects. There are several methods of identification, but the most common is to store a serial number that identifies a person or object, and perhaps other information, on a microchip that is attached to an antenna (the chip and the antenna together are called an RFID transponder or an RFID tag). The antenna enables the chip to transmit the identification information to a reader. The reader converts the radio waves reflected back from the RFID tag into digital information that can then be passed on to computers that can make use of it.”*

(rfidjournal.com, 2012)

MIFARE was used due to the small form factor of the tags and this was necessary for embedding the rfid capability in the globe objects (small, soft durable sponge balls).

### ID-12LA

ID-12LA rfid reader chip was used in conjunction with the Arduino.

Designed and developed by ID-Innovations (http://id-innovations.com), this product is widely available and reads tags at a frequency of 125Mhz which was ideal for the Globlock system and MIFARE style RFID tags.

### ID12LA Breakout Board

The ID20 Breakout board is a small PCB available from Sparkfun Electronics (sparkfun.com), which allows the ID-12LA (or ID20) to be mounted to the PCB and in turn mounted to a prototyping breadboard for testing and development.

### Client PC and Server Host

There was no specific requirement for a PC but any PC/Laptop device capable of running Windows 7 and connecting to the internet or a local network was required for the client, and any system capable of hosting an apache web service, MySQL and PHP4.5 or higher was required for the back end of the system. This architecture will be described in more detail later in the document.

## Software

### Github

Throughout the development process Github was used as a code repository for the Globlock system, supporting references, libraries and documentation.

Github allowed code to be pushed and pulled from different PC’s and devices while still maintaining a master copy.

### Trello

Trello was used to manage the early deliverables, tasks and milestones of the project. Once coding began, this was dropped in favour of github where all files and associated documents were maintained.

## Development Environments

### Arduino

The Arduino IDE, which is an open source software IDE based on processing, allowed the use of RFID and software serial libraries. Individual code segments, called sketches are compiled into native C and uploaded to the Arduino directly from the IDE.

### Visual Studio

Visual Studio was used for the development of the client application which was developed to demonstrate the interaction between a unique RFID tag read from the Arduino and sent to the client application, which in turn communicates with the server. Visual Studio was chosen for its extensive libraries and support for serial communications.

### Notepad++

For PHP scripts, as well as HTML, CSS and initialisation files, notepad++ was the only development environment used. This prevented added ‘baggage’ found with other IDE’s designed for web based technologies such as PHP, HTML5 and CSS3.

## Supporting Frameworks and Technologies

### XAMPP

XAMPP, again an open source product, was used to port a LAMPP stack on a windows based machine. A portable version of XAMPP loaded to an USB flash drive was used to allow the deployment and publishing of PHP, HTML5 and CSS3 files from the different machines I developed the system on. In a production environment the same could be used.

### Apache HTTP

The Apache Software Foundations Apache HTTP Server was used to publish and manage server files over the web. Apache was chosen as an open source alternative, due to its highly configurable environment and widely available knowledge bases and free support network on the internet

### PHP

Although the client application was developed in C# in Visual Studio .NET, PHP an open source language and framework, was chosen as it allows greater portability for the system. PHP has greater flexibility around OS choices and this would allow the entire Globlock system to appeal to a greater audience.

## Database Technologies

### MySQL

MySQL, an open source database technology, was chosen due to its scalability, flexibility and cost. MySQL also comes packaged with the LAMPP or XAMPP stack and this grouping of technologies has proved very successful in the past.

### SQLite

A small portable lightweight DB was required for the client to manage transactions and users of the client application. SQLite was chosen for this and due to the fact it is again open source, and the language similarity to MySQL would reduce development time.

## Programming Languages

### Client Application

Visual Studio .NET was used for the client application which primarily used C#.

### C#

C sharp was chosen for the client application in order to make use of the SerialPort libraries as well as provide a smooth and tasteful look and feel for the client application. Having worked with Java before I was confident the transition to C# would cause little difficulty.

### JSON

*“JSON is short for JavaScript Object Notation, and is a way to store information in an organized, easy-to-access manner.”*

(Lengstorf, 2014)

Java Script Object Notation was used as a return string message from the Server Side API interaction as it is becoming an increasingly common message broker. It was chosen in place of XML. A broker object (discussed later) is encoded in JSON format on the server prior to being received by the client application.

### Client Device (Arduino)

C++:The Arduino is written entirely in C++. When compiled in the Arduino IDE it is compiled to native C before being uploaded as a sketch to the target Arduino device.

### Server Side API

The Server Side API is coded almost entirely in PHP (decribed earlier) and messages encoded in JSON are also collected from the client application.

A single language for the server API allowed for a more robust and focused approach to its development.

### Management Website

#### HTML5 & CSS3

HTML5 is used for the development of the Management Website. CSS3 is also used and the management website is a fully responsive desingn, which allows segmentation on smaller devices which improves the user experience when interacting with the system.

#### PHP

PHP was also used for the management of the website to provide the necessary table content for the different components of the system such as users, groups and documents.

#### Javascript

Javascript is used to add some sugar to the user interface and improve the look and feel of the website.

## Structure

The remaining chapters of the report for the most part will outline the technical aspects to the project.

Firstly we will look at the system features and their intended implementation.

# System

## Requirements

### Features Overview

Below is a brief overview of the system features. See Appendix of SRS document for full Use Case Diagram.

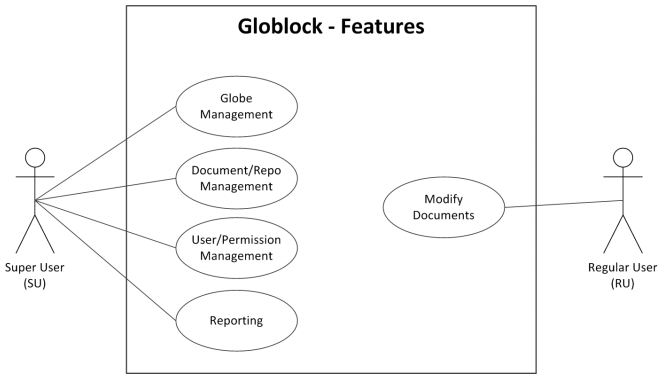


Figure 1

### User Permissions Management

A very important feature of the initial specification ‘Globlock’ system was the ability to manage user accounts. This is still the case and has been implemented. A high level user essentially controls other user accounts and permissions over globes created and modified within the system.

Due to time constraints, the ability to assign individual permissions on specific projects is not possible however the system has been developed to allow this implementation at a later stage.

A basic user will exist which will interact with the globes and make changes to the associated files. At a later stage, these basic users will be restricted from accessing globes that they have not been given privilege to by the high level user. Privileges of read or write may be assigned to each user on each globe, but at present all users can interact with Globe Projects.

### Repository and Document Management

As each globe is manipulated with by basic users and changes made to its associated files and documents, each iteration or revision of these will be stored and recorded. A central repository located on the server, the transparency of which will be hidden from the user, will store each revision or version. This in essence will provide the archiving functionality and management of documents. The location of the storage is configurable through the initialisation files on the server.

Future scope will allow a high level user to have a facility to take a revision or version of a single document, or entire group of documents, and extract from the central archived repository and associate with a new or existing Globe. This will essentially allow multi-threaded versions (branches) of documents.

### Globe Management

Each Globe will be managed by high level users of the system. A function of the software will allow the high level users to assign individual as well as groups of users access rights to the Globes created.

These permissions will be basic read/write but will allow users to be assigned to Globes for information only. Users without write permissions will not be able to update or make changes to revisions of the Globe or its associated files or documents. As stated all users currently have read/write access to all files.

### Reporting

Another feature of the system in future revisions will be basic reporting functionality which will allow high level users to track changes to Globes, files and documents and also track what users have interacted with the Globes.

Each main section of the Management website currently lists all documents, globes, groups and users maintained within the system. If an addition is made, the particular sections form data is posted to a server side php script, which re-directs to the original location on successful completion.

Functional requirements

## System Features

### User Management

A function of the system is the facility to allow a SU to create user accounts so that they may interact with the system. This is completed through the Management Website, hosted on the server.

### Document Management

As the goal of the system is to provide a repository and archive for document management, the ability to upload and download documents is a functional requirement.

### Document Upload

When uploaded to the system, the documents are stored in a predefined location based on a combination of configurations set out in the initialisation files and the unique identity of the document in the database.

### Document Download

Upon successful interaction with a globe object and user has been validated by the system, the document is downloaded locally to the client PC. Again the location of the storage is based on a combination of configurations set out in the initialisation files and the unique identity of the associated globe object.

### Globe Management

Globe management is an encapsulation of all the functional requirements that will be needed for

documents to be tokenised and accessible from a RU or group of users. In the current state the system does not allow documents to be assigned individually to globe projects.

### Create Globe

A globe can be created and added to the database. This is completed through the Management Website, hosted on the server.

### Archive Glove

When a user uploads a new version of a document, the previous version is copied to an archive folder and replaced in a current directory with the newly uploaded document.

### Globe Access

After Globes have been created, documents uploaded and associated and users given access to the necessary tools, there then exists a requirement for users to open the newly created Globes and modify their contents. This is made possible by the client applications interaction with the server side API.

### Open Globe

When a user places the globe device at the reader, the client application receives its unique id, requests the associated documents from the server, and downloads them locally.

### Reporting

A very basic functionality from the system is the ability to report on current and archived Globes that have been created, and the changes that have occurred to them. Currently, each section of the management website lists the specific tables of entries, whether it be users, groups, globes, documents and the date they were added to the system.

## Non-Functional Requirements

### Performance/Response time requirement

#### Client Application

Windows service must not use more than 5% of system resources during idle periods and no more than 75% during non-idle for more than 15 seconds. This is to prevent the Globlock system being viewed as a burden or hindrance to the user.

In the current state, the application uses approximately 11.25 megabytes of system memory while the application is in an idle ‘listening’ state.



Figure 2

#### Server Response

Response from the server must not exceed 60 seconds for requests for primitive data queries. Download times should not exceed 5 minutes per document.

In the current state, response from the server after client application requests takes less than a second for any and all request types.

#### Reader Response

It should not take longer 10 seconds to read Embedded RFID in Globes from the Block device. A certain amount of buffering is required in order to allow serial communications, however in the current state this is still taking less than a one or two seconds.

#### Server Availability

Server must have minimum of 99% uptime. This will allow 6.72 hours over a four week period to allow for scheduled changes or unscheduled down time.

If server is unreachable, documents must be stored locally until such time that the server becomes available.

In the current state there has been no uptime failure, and documents are stored on the client pc. The location of the storage is based on a combination of configurations set out in the initialisation files and the unique identity of the associated globe object. If server response fails, the documents are still maintained locally.

### Recoverability & Reliability Requirement

#### Globe Loss

As a Globe is a physical object it may become lost. A facility to archive globes already exists and will allow previous documents associated with a lost Globe to be assigned to a new Globe.

The API has been designed in such a way as to allow globe objects to be assigned to projects by FORCE, in that the previous asset assigned to it is overwritten.

#### Software Error Response

All errors should be captured and in the event the system cannot commit changes the user will be clearly and effectively informed.

All errors are managed and stored to a logfile on the server in the case of API errors and all transactions are written locally on the client side sqlite database.

#### Hardware Failure

In the event of hardware failure the management application should continue to allow the management of users and documents, until such time that the Globe can be docked, loaded and undocked.

In the current state, all management of the system is carried out through the management website, entirely separate to the client. This allows complete separation in the event of a hardware failure.

#### User Access

Users without previously assigned access should not be able to access the files. Users with only read privileges should not be able to write or make changes. User accounts, both super (SU) and regular (RU) will require passwords. Passwords complexity will be defined the ISO27001 standard for Information Security.

In the current state all users of the client system require password identification. The password is encrypted using SHA1.

#### Communication Transport

A method of encryption (TBD) will be used to encrypt data packets and files while being transported to prevent breaches.

In the current state, the system does not encrypt entire packet information, however session tokens are generated from a random string and encrypted with the addition of a salt value, using SHA1. User passwords are also encrypted using SHA1. It would be recommended that a SSL certificate be used on the server if system was implemented in a production environment.

### Maintainability & Re-usability Requirement

Updates to platform of systems should be easy to manage and maintain with no additional training by the users.

The client application is a standalone executable, with an extremely simple user interface and as such would not require a great deal of training in order to interact with it.

#### Re-Usability

The Globes should be interchangeable and re-usable in that a super user may decide to assign new or archived information to a Globe, to prevent making the Globe objects obsolete.

The API has been designed to allow a FORCE server call which will overwrite the existing object reference with that of a new one.

#### Code Re-use

Code shall be written in such a way as too allow as much flexibility as possible around it’s re-use. Throughout the design and development of the system code re-use was high on the agenda and object orientation, segmentation and encapsulation facilitate this.

# Design and Architecture

## Overview

The diagram below gives a very high level overview of the separate components of Globlock.

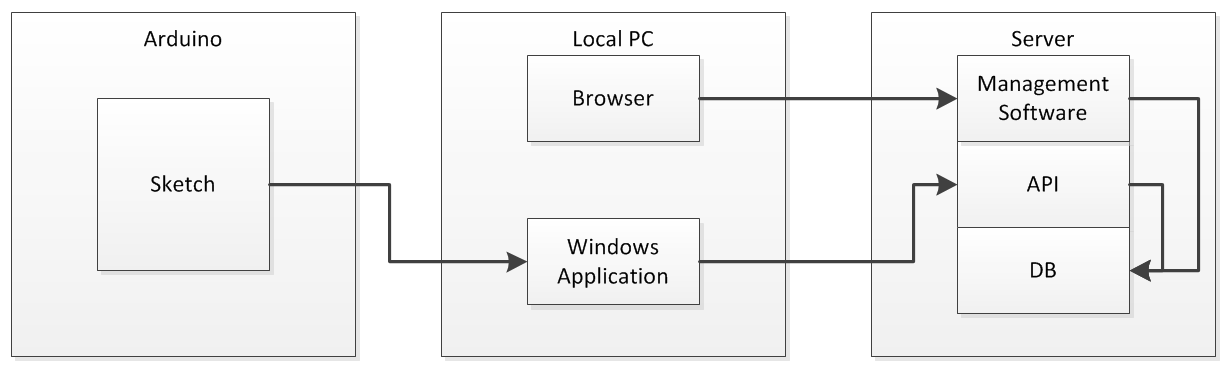


Figure 3

The Arduino device communicates tags to the client windows application. This application processes server requests, and returns the results to the client or carries out the desired client side functions depending on the result.

The API acts as middleware between the client and the database. The client may also interact with the management software which is delivered in the form of a website. Again, the website acts as middleware between the user and the database.

## Block Device

The original block device design and schematic allocated some digital pins on the Arduino to allow the device to interact with an LCD display. There was also room for the addition of a buzzer to alert the user. This worked fine in the early stages of testing but added a level of complexity later on which slowed development.

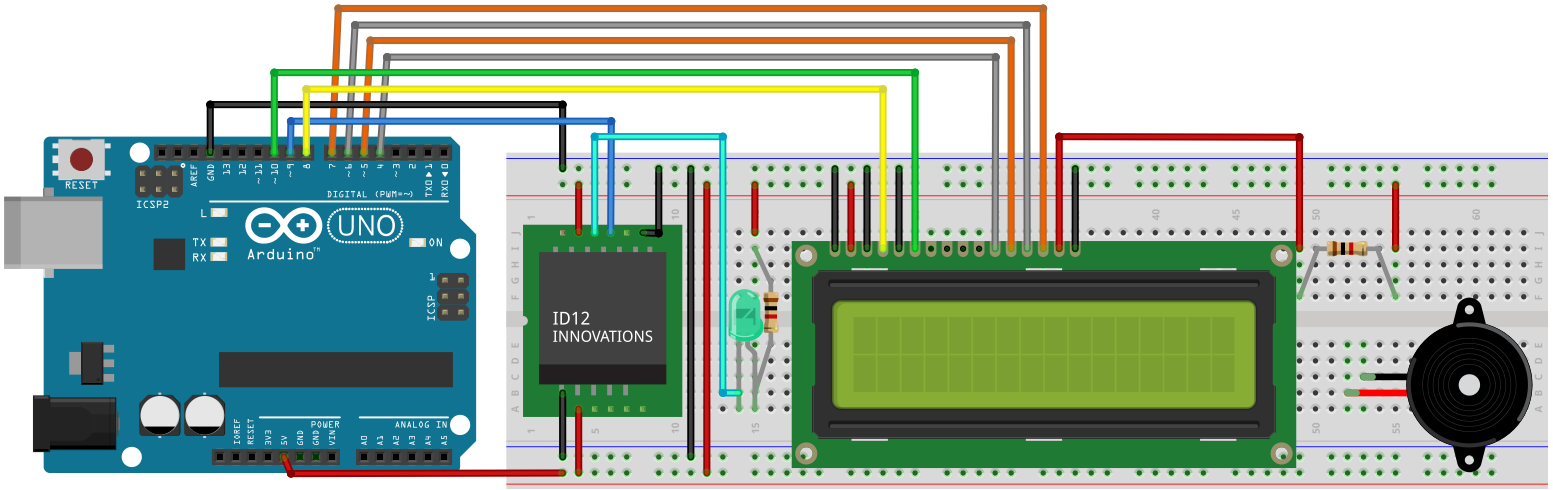


Figure 4

This additional LCD component would have provided some additional user experience, and relay information visually and aurally to the user such as system states (‘Connecting’, ‘Handshaking’,’ Listening’), however due to time constraints this additional functionality was dropped in favour of a simpler design schematic.

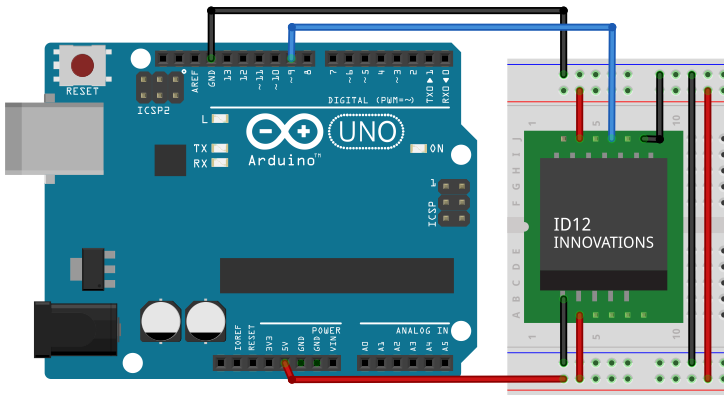


Figure 5

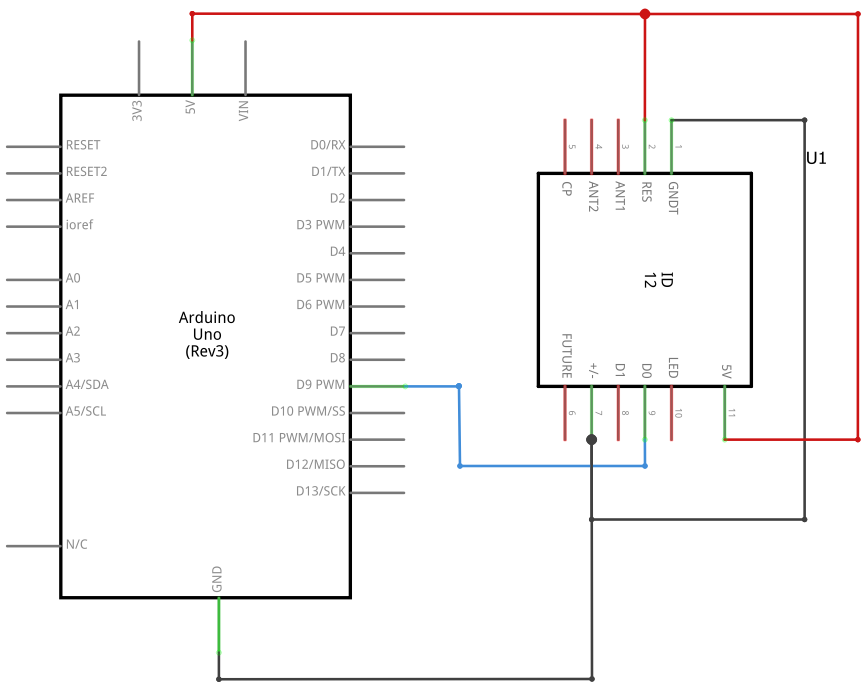


Figure 6

The diagram and schematic above were created using Fritzing (fritzing.org). The table below represents some important information contained in the figures:

|  |  |  |
| --- | --- | --- |
| **Object** | **Colour** | **Description** |
| Wire | Red | Live (+5v). Carries 5 volts from Arduino device to breadboard and connected components. |
| Wire | Black | Ground (-GND). Carries ground back to Arduino device |
| Wire | Blue | Carries tag id to the Arduino device using a virtual serial port (software serial). |
| Component | Black/Green | ID-12LA. Device to read rfid tags. |
| Carries tag ID using ASCII over I2C. |
| Draws approx. 65mA of current |
| Range approx. 120mm. |
| Pin 9 | - | Used for capturing data from ID-12LA |
| Pin 13 | - | Used to flash for data received |

## Client Application

The client application uses a SerialPort library to send and receive commands from the block device through the devices USB port.

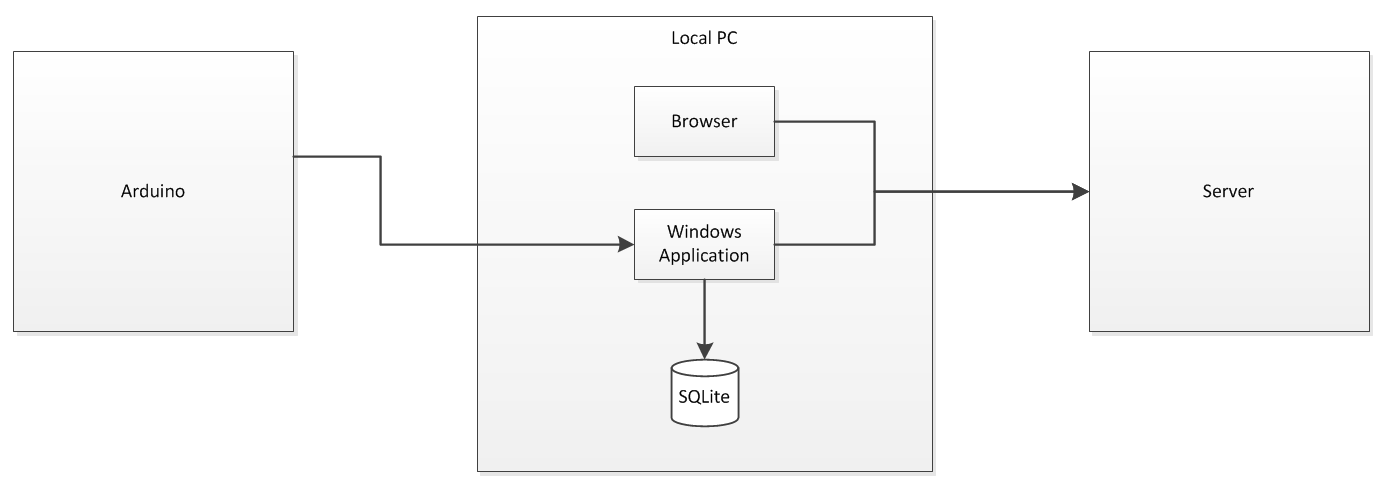


Figure 7

Depending on the state and data received, it communicates with the server. To do this it makes use of the Web Client Library. Transaction information and login information is stored in a local SQLite database.

Through the browser, the user may make changes and create and upload new Documents, Globes, Groups and Users. This management web application is hosted on the server.

### Hosted Server

Whether a local host or a remote host, the server provides the backbone of the entire system, and hosts a number of PHP files which are the core of the Globlock API.

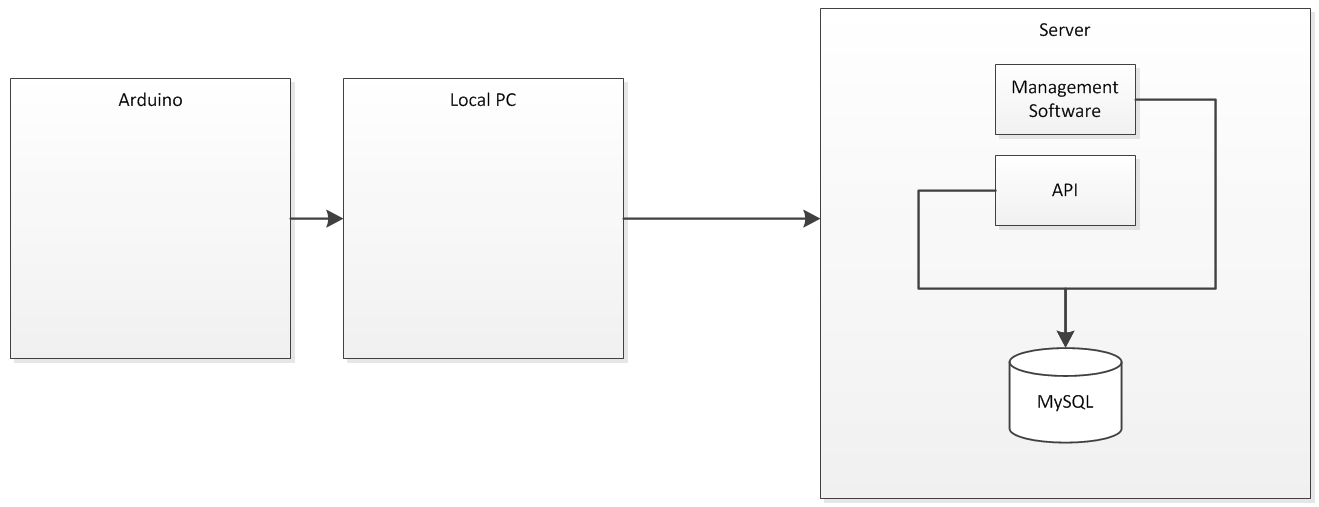


Figure 8

## Management Web Application

From the management web applications view, the framework of the server the diagram below:

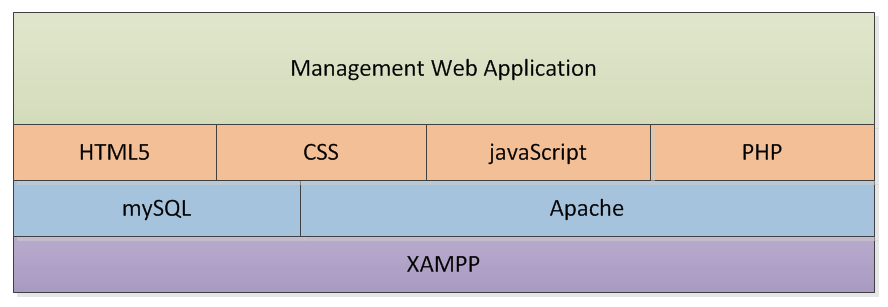


Figure 9

On top of the XAMPP stack sits the Apache HTTP server and a MySQL database. Both of which are open source and allow for great configurability.

On top of this is the web deployment such as HTML5 for the server pages, CSS3 to provide visually appealing interface components and html elements, Javascript to further add to this user experience element and finally some PHP scripts to allow server functionality.

## Globlock API

From a purely API standpoint the framework is very similar.

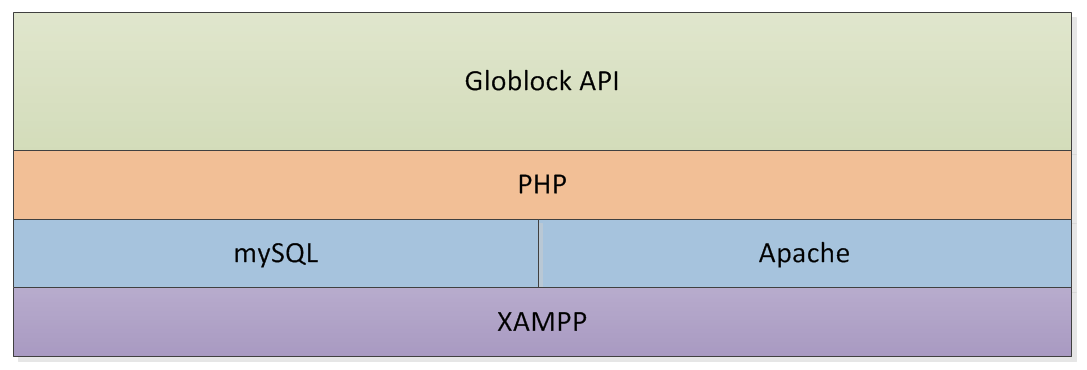


Figure 10

# Implementation

## Database Schema

### Server Side

The current system state from a server perspective is outlined below.

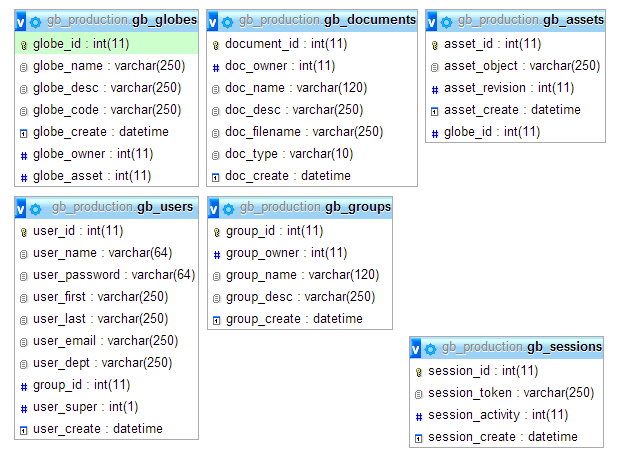


Figure 11

There are 6 tables in total in the ‘gb\_production’ database, so named after ‘Globeblock’ production system.

Each table has a unique primary key, and also maintains a default datetime stamp which allows the system to accurately identify when the particular record was inserted into the database.

#### Sessions Table

The session table (gb\_sessions) is the most frequently accessed and updated table and for this reason the schema for the table was kept to a minimum. The activity defines whether the session information is at the correct activity with the system.

The first entry point to the API is the ‘HANDSHAKE’ request, but after this if a client wishes to interact with the system they must first complete a ‘SESSION’ request. This will be discussed in greater detail later but essentially a session token is generated and inserted to the database.

The activity is then set to 1. Each time the session token is used, the activity is increased. A session token may only be used twice. The first of which must be to ‘VALIDATE’ a globe object .

All subsequent actions require the activity to be at 2 in the database, otherwise the requests will be aborted.

After an action is requested on the server at activity level 2, the session activity is set to -1, which represents an inactive session token. The session token can then no longer be used.

#### Assets T***a***ble

The assets table (gb\_assets) contains the references to individual unique globe objects and their association to globe projects (FK: globe\_id). Revision information for each asset is also maintained in this table.

#### User Table

The user table (gb\_users) maintains user information, as well as group association (FK: group\_id) and whether the user is a high level user (user\_super: 1/0) or not.

#### Globe Table

The Globe table (gb\_globes) maintains globe information, such as name, description and short code. A globe owner (globe\_owner) was put in place for future scope to allow permission around updates and changes to individual globes.

#### Documents Table

When a user uploads a document on the system, the document information is created here (gb\_documents). Again, an owner column is present for future scope to allow permission around updates and changes to individual documents.

#### Groups Table

The groups table (gb\_groups) maintains information related to the different groups with the system. As with Globes and Documents, an owner attributes allows for future scope to around permissions and updates to individual groups.

### Server DB Schema in SQL format

-- Database: `gb\_production`

--

CREATE DATABASE IF NOT EXISTS `gb\_production`

USE `gb\_production`;

-- --------------------------------------------------------

-- Table structure for table `gb\_assets`

CREATE TABLE IF NOT EXISTS `gb\_assets` (

`asset\_id` int(11) NOT NULL AUTO\_INCREMENT,

`asset\_object` varchar(250) NOT NULL,

`asset\_revision` int(11) NOT NULL,

`asset\_create` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`globe\_id` int(11) NOT NULL,

PRIMARY KEY (`asset\_id`)

);

-- --------------------------------------------------------

-- Table structure for table `gb\_documents`

CREATE TABLE IF NOT EXISTS `gb\_documents` (

`document\_id` int(11) NOT NULL AUTO\_INCREMENT,

`doc\_owner` int(11) NOT NULL DEFAULT '0',

`doc\_name` varchar(120) NOT NULL,

`doc\_desc` varchar(250) NOT NULL,

`doc\_filename` varchar(250) NOT NULL,

`doc\_type` varchar(10) NOT NULL,

`doc\_create` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`document\_id`)

);

-- --------------------------------------------------------

-- Table structure for table `gb\_globes`

CREATE TABLE IF NOT EXISTS `gb\_globes` (

`globe\_id` int(11) NOT NULL AUTO\_INCREMENT,

`globe\_name` varchar(250) NOT NULL,

`globe\_desc` varchar(250) NOT NULL,

`globe\_code` varchar(250) NOT NULL,

`globe\_create` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`globe\_owner` int(11) NOT NULL,

`globe\_asset` int(11) DEFAULT NULL,

PRIMARY KEY (`globe\_id`)

);

-- --------------------------------------------------------

-- Table structure for table `gb\_groups`

CREATE TABLE IF NOT EXISTS `gb\_groups` (

`group\_id` int(11) NOT NULL AUTO\_INCREMENT,

`group\_owner` int(11) NOT NULL DEFAULT '0',

`group\_name` varchar(120) NOT NULL,

`group\_desc` varchar(250) NOT NULL,

`group\_create` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`group\_id`)

);

-- --------------------------------------------------------

-- Table structure for table `gb\_sessions`

CREATE TABLE IF NOT EXISTS `gb\_sessions` (

`session\_id` int(11) NOT NULL AUTO\_INCREMENT,

`session\_token` varchar(250) NOT NULL,

`session\_activity` int(11) NOT NULL,

`session\_create` datetime DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`session\_id`)

);

-- --------------------------------------------------------

-- Table structure for table `gb\_users`

CREATE TABLE IF NOT EXISTS `gb\_users` (

`user\_id` int(11) NOT NULL AUTO\_INCREMENT,

`user\_name` varchar(64) NOT NULL DEFAULT '1',

`user\_password` varchar(64) NOT NULL DEFAULT '0',

`user\_first` varchar(250) DEFAULT 'undefined',

`user\_last` varchar(250) DEFAULT 'undefined',

`user\_email` varchar(250) DEFAULT 'undefined',

`user\_dept` varchar(250) DEFAULT 'undefined',

`group\_id` int(11) DEFAULT '0',

`user\_super` int(1) NOT NULL DEFAULT '0',

`user\_create` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`user\_id`)

);

-- --------------------------------------------------------

### Client Side

The current system state from a client database perspective is outlined below.

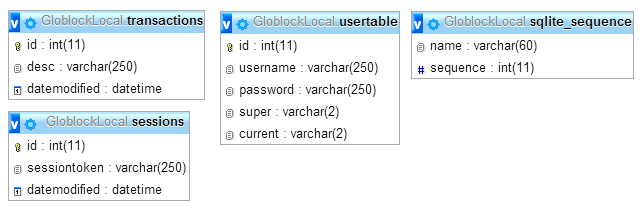


Figure 12

#### Transactions

The transaction table allows information about the continuous running success of the client application to be written to the local sqlite database.

#### UserTable

The user table maintains username and password (encrypted using SHA1) information for users who wish to remain logged into the application after the application is shut down or restarted.

#### Sessions

The sessions table maintains session token information for use in the application for server requests, and removes the need to store token information in memory.

#### SQLITE\_SEQUENCE

This table is a boilerplate sqlite table that maintains the auto increment sequence information for other tables in the database.

### Client DB Schema in SQL format

-- Database: `GloblockLocal.db`

--

--------------------------------------------------------

-- Table structure for table `Sessions`

CREATE TABLE IF NOT EXISTS Sessions(

id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

[sessiontoken] VARCHAR(250),

[datemodified] datetime

);

--------------------------------------------------------

-- Table structure for table `Transactions`

CREATE TABLE IF NOT EXISTS Transactions(

id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

[desc] VARCHAR(250),

[datemodified] datetime

);

--------------------------------------------------------

-- Table structure for table `UserTable`

CREATE TABLE IF NOT EXISTS UserTable(

id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

[username] VARCHAR(250),

[password] VARCHAR(250),

[super] VARCHAR(2),

[current] VARCHAR(2)

);

--------------------------------------------------------

-- Table structure for table `aqlite\_sequence`

CREATE TABLE IF NOT EXISTS sqlite\_sequence(

name,

seq

);

INSERT INTO sqlite\_sequence VALUES('Transactions',0);

INSERT INTO sqlite\_sequence VALUES('UserTable',0);

# The Globlock API

## Class diagram overview

The class diagram below illustrates how the core API of the system is packaged together, and identifies file types (discussed later) as well as their interaction with systemsuch as database file access [  ] or file access and manipulation[  ].

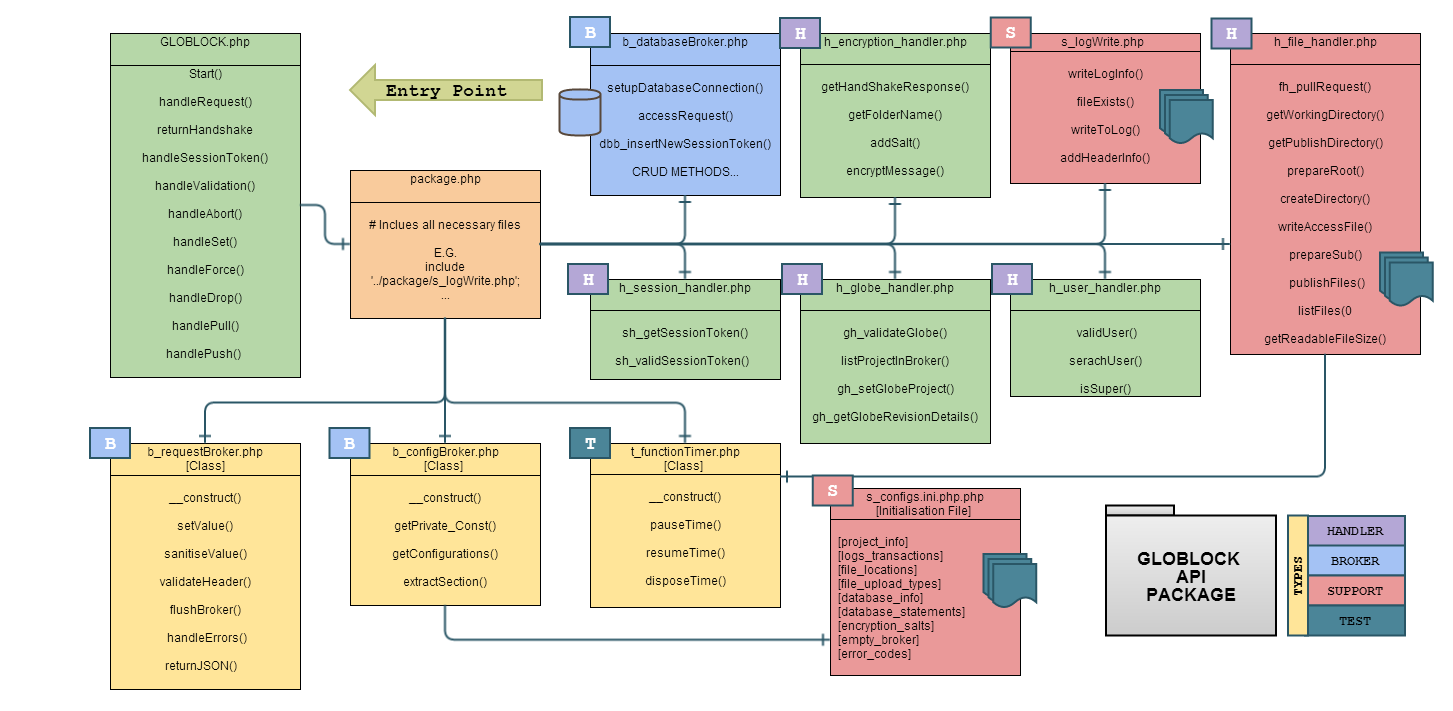


Figure 13

## Package File Types

### Classes

There are three PHP object classes in the server side API package, in the Request broker, the configurations broker, and the function timer. Of the three, the request broker is of most value to the API’s interaction with supporting files and client responses.

### Handler Concept

The concept of a handler is similar to a broker in that it takes away the processing required the core API file and allows the API file to concentrate solely on responding to requests. Unlike the broker, however, no information is maintained and once the desired method is accessed from the core API file, the handler is disposed.

Essentially handlers act as servants for the core API file, carrying out the desired actions and returning the requested responses.

### Broker Concept

The concept of the broker is to allow a go between from one area of the API’s functionality to another, and at the same time carry or maintain some information relevant to the particular request. The Request Broker as an example is integral to the entire API and in fact the entire system. The configurations broker allows configurability of the system and the database broker acts as a form of middleware between the API’s distinct components and the database itself. All database transactions are carried out by the database broker. Similarly, all custom configurations are managed by the Configurations broker.

### Support

Support types are in place to support the API, outside of basic functionality and request responses. The log write support script writes processing information to a log file in order to track unseen or uncaptured errors, as well as track transactions on the core API and supporting files.

The ‘configs.ini’ file supports the system administrator/customer in configuring the system, and allows settings to be modified and updated.

### Test Files

An example of a test file is the function timer class which starts a timer when an instance is constructed and has methods to pause, stop or dispose of the object, upon doing so, returns the interval of time. It was used in the early stages of developed particularly around file handling as there was a fear that the server requests could time out on the client if the time t outside the allowable limits. This proved not to be an issue.

## API & Package

### API Main File

The Globlock API has a single entry point to the system at ‘GLOBLOCK.php’ This file then includes the ‘package.php’ file, which maintains all the necessary include statement for the successful interaction of the API and its components.

The core API file also solely handles all API requests to the server and depending on the particular request, will carry out the appropriate action and when complete, return JSON encoded message string to the requestor. These request types will be discussed in ‘API Interaction’ section later.

### Package

The file entitled package allows a single file to be used for all include statements throughout the API. This allows the API to scale, and each individual file can be extended or change without issue. In essence it allows a central location for all necessary file inclusions.

### Request Broker

Aside from the core API’s structure and functionality, the most important file in the API package is the request broker, and for this reason it will be outlined here.

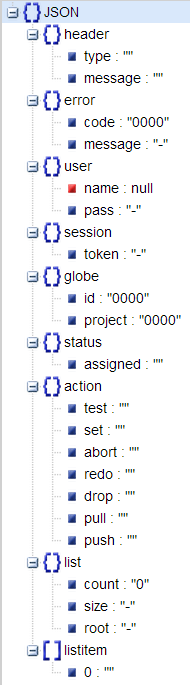


Figure 14

The request broker maintains a class that is utilised through the API. The request broker maintains a catalogue of information and as each action is requested of the API, the instance of the request broker (being passed by reference to the appropriate included method) is updated with the desired information and returned to the requestor as JSON.

### Construction

During construction of an instance of a broker request class, the constructor assigns a default structured value, which is a multidimensional array of values. An empty broker is also created at this time to allow the requestBroker to be flushed at a later stage.

/\*\* CONSTRUCTOR

\* Takes request Header and session user as parameters

\* and using a configuration object, creates and populates an empty broker

\*/ function \_\_construct($request\_header, $session\_user){

$configuration = new configurations();

$this->emptyBroker = $configuration->extractSection("empty\_broker");

$this->brokerData = $configuration->extractSection("empty\_broker");

$this->brokerData['header']['type'] = $request\_header;

$this->brokerData['user']['name'] = $session\_user;

}

### Sanitation

For security reasons, during each update of the brokers data, the data being passed in for update is sanitised, using a series of inbuilt PHP library methods.

/\*\*

\* SANITISE VALUE

\* Takes 'Data' passed as a parameter and sanitises the value, for security

\* Returns the sanitised value

\*/ function sanitiseValue($data){

$data = filter\_var($data, FILTER\_SANITIZE\_STRING, FILTER\_FLAG\_STRIP\_LOW);

$data = filter\_var($data, FILTER\_SANITIZE\_STRING, FILTER\_FLAG\_STRIP\_HIGH);

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

This sanitation of requests ensures that script attacks and malicious injection are avoided immediately, as the data is thoroughly sanitised before accessed from any other method.

### JSON

A return method encodes the request objects array of values to JSON format to allow the API to return JSON as an appropriate response.

function returnJSON($headEncode = 0){

if ($headEncode <> 0) header('Content-Type: application/json');

return json\_encode($this->brokerData );

}

## API Interaction

### Point of Entry

When a request is received by the server, it invariably enters at the point of the ‘GLOBLOCK.php’ core API file. As stated this is the only entry point of the API.

/\* Global Files, Libraries and Declarations \*/

include 'package.php';

/\*\* Creates an Empty Broker Object

\* If Start() returns successful, calls handleRequest

\*/

$broker = new requestBroker("Initialised", null);

if(!start($broker)) exit();

handleRequest($broker);

/\*\* Validates server request type conforms to HTTP POST,

\* If not correct format, updates the brokers error code and message,

\* and echo's the JSON response string message

\*

\*/

function start(&$broker){

if (!($\_SERVER["REQUEST\_METHOD"] == "POST")) {

$broker->handleErrors("NON [POST] TYPE SERVER REQUEST ",121);

echo $broker->returnJSON();

return false;

}

return true;

}

The API’s first port of call is to include the ‘package.php’ which allows the core file to access all the associated classes and methods.

The API then creates an empty ‘Request Broker’ object and from this point on any failure will be returned to the server in the correct format.

The start function is called, which validates the server request method and if not found to be a post exits the API.

If a valid request method is found, the API calls its own internal handleRequest method passing in the newly created broker object.

### Handling Requests

After assignment to the broker each request is validated internally by the broker. Malformed requests are handled by the request brokers error handler.

\*\* HANDLE REQUEST

\* Validates Request Header Type and carries out appropriate action

\*/

function handleRequest(&$broker){

$broker->setValue("header", "type", $\_POST["request\_header"]);

if (!$broker->validateHeader()) {

$broker->handleErrors("BAD REQUEST: UNDEFINED OR MALFORMED HEADER REQUEST", 400);

echo $broker->returnJSON();

return false;

}

switch ($broker->brokerData['header']['type']){

case "HANDSHAKE": //

$broker->setValue('header', 'type', $\_POST["request\_header"]);

returnHandshake($broker);

break;

case "SESSION":

handleSessionToken($broker);

break;

case "VALIDATE":

handleValidation($broker);

break;

case "ABORT":

handleAbort($broker);

break;

case "SET":

handleSet($broker);

break;

case "FORCE":

handleForce($broker);

break;

case "DROP":

handleDrop($broker);

break;

case "PULL":

handlePull($broker);

break;

case "PUSH":

pushFiles($broker);

break;

}

echo $broker->returnJSON();

}

Successful requests lead to internal method calls in the API and as always, passing the broker as a parameter so that it may be accessed from the method called by reference.

### Request Types

Rather than describe each line of code, the HANDSHAKE request is described below and the table in figure 14, gives a brief outline of the inputs required by each action and their response.

### HANDSHAKE

The handshake API request, validates the header message, and returns a handshake response which is generated from a method call to the encryption helper.

function returnHandshake(&$broker){

if (empty($\_POST["request\_body"])){

$broker->handleErrors("LENGTH REQUIRED: MESSAGE REQUEST BODY EMPTY",411);

} else {

$broker->setValue('header', 'message', $\_POST["request\_body"]);

$message = getHandShakeResponse($broker->brokerData['header']['message']);

$broker->setValue('header','type', "HANDSHAKE RESPONSE");

$broker->setValue('header', 'message', $message);

}

}

The method (getHandShakeResponse) in the encryption handler file, takes the header message and encrypts it with a salt using SHA1.

function getHandShakeResponse($message){

writeLogInfo("Handshake Request to :". $\_SERVER['SERVER\_NAME'] ." | From :". $\_SERVER['REMOTE\_ADDR']);

$message = addSalt($message, "handshake");

$message = encryptMessage($message);

return $message;

}

The HANDSHAKE is used to allow requestors to ensure that the server is alive and responding, before attempting to transmit username and password information in a bundled SESSION request.

### Request Table

|  |  |  |
| --- | --- | --- |
| **REQUEST TYPE** | **IN** | **OUT** |
| HANDSHAKE | [Header], [Message] | [Encrypted message response] |
| SESSION | [Header], [User:Pass] | [Session Token] |
| VALIDATE | [Header], [Session Token], [Globe ID] | [Action list] |
| ABORT | [Header], [Session Token] | [Success result] |
| SET | [Header], [Session Token], [Globe Project], [Globe ID] | [Success result] |
| FORCE | [Header], [Session Token], [Globe Project], [Globe ID] | [Success result] |
| DROP | [Header], [Session Token], [Globe Project], [Globe ID] | [Success result] |
| PUSH | [Header], [Session Token], [Globe ID], [Files] | [Success result] |
| PULL | [Header], [Session Token], [Globe ID] | [File Information], [File list] |

Each request to the API has an internal call to an associated handleMethod, each of which are in the class diagram.

### Sessions

Once the requestor is assured that the server is alive, in order to interact with the API they must request a session taken which has a limited lifespan. This session token allows interaction to the API for all other stages of the system. Sessions will only be granted to valid users of the system.

#### Session Token

The session token is generated by creating a datetime value in combination with a random digit between 1-1000, and the salt value configured in the configs.ini file. This value is then encrypted using SHA1.

$randString = addSalt(date("Ymdhis") . rand(1,1000), "session");

$token = strtolower(encryptMessage($randString));

A random generated token ensures the system cannot be fooled, a sequence interpreted or reverse engineered.

In order to receive a session token, a valid username and password are required, but only for SESSION requests. Subsequent interaction with the API does not require username or password information, instead only a valid SESSION token is needed, providing it has not expired or reached its full activity cycle.

#### Activity

A SESSION requests returned session token is only valid for 2 activities. The first of which MUST be a VALIDATION request. After each request on the back of a specific session token, the session token’s activity flag is updated. This update occurs during session token validation. This is a security failsafe by design. If somehow the token was intercepted, without knowing the current activity cycle (after validation or before) the session token is voided if an incorrect activity is requested.

#### Action List

Validation takes a session token and a globe object id as input and returns an action list if successful. This action list identifies the next possible course of action on the part of the requestor and after receiving this, the last remaining activity on the session token can be accessed.

#### Abort

At this stage it is possible to abort a session token, which sets its activity cycle to ‘-1’ in the database or ‘inactive’.

if (sh\_validSessionToken($broker, 2)) {

$broker->setValue('header','message', "ABORT SUCCESSFUL");

}

Abort does not require any update as merely validating the session will update the activity, and as it is at stage 2 in the cycle, will be marked ‘inactive’ in the database

### SET, FORCE & DROP

Set takes a session token, globe project and object id as input and validates all the header information before calling the gh\_setGlobeProject method which internally in the globe handler, validates all the globe related information and updates the broker with the success of the operation.

function handleSet(&$broker){

try{

$broker->setValue('header','type', "SET RESPONSE");

if (!(isset($\_POST["session\_token"])))

throw new Exception("Exception Thrown (EMPTY POST): ");

$broker->setValue("session", "token", $\_POST["session\_token"]);

# Activity updated to -1

if (sh\_validSessionToken($broker, 2)) {

$broker->setValue("header", "message", "Valid Session");

gh\_setGlobeProject($broker);

} else {

$broker->setValue("header", "message", "Invalid Session");

}

}catch(Exception $e){

$broker->handleErrors($e."UNABLE TO SET GLOBE OBJECT ",121);

}

}

Similarly, Force will overwrite an existing globe project, and drop will remove its association entirely.

### PULL & PUSH

PULL and PUSH request are essentially the meat and veg of the entire API request stack.

PULL on successful token validation will create a root directory on the server that is accessible through HTTP requests, if it does not already exist. In this directory it a ‘.htaccess’ file is created which denies access to the Apache HTTP server to prevent the folders parent structure being published.

function writeAccessFile($type, $directoryTo){

try{

$filename = ".htaccess";

$fullname = $directoryTo .'/'. $filename;

if (file\_exists($fullname)) return true;

switch ($type){

case "root":

$first = "Deny"; $second = "Allow";

break;

case "sub":

$first = "Allow"; $second = "Deny";

break;

}

$fileContents = "Order ". $first .",". $second ."\n". $first ." from all";

if(file\_put\_contents($fullname, $fileContents)){

Then a copy of the associated globes projects files are moved to a newly created publish directory within the root directory. This folder is generated through the encryption handler to ensure it is non sequential, similar to session token creation.

function getPublishDirectory(&$configs){

$publish\_directory = $configs["file\_locations"]["publish\_directory"];

$sub\_Dir = strtoupper(encryptMessage(

addSalt(date("Ymdhis") . rand(1,1000), "folder"))

);

$full\_Directory = $publish\_directory .'/'. $sub\_Dir;

return $full\_Directory;

}

Finally a ‘.htaccess’ file is created with the opposite to the root directory, and the entire file and folder details are inserted to the broker for JSON response to requestor.

PUSH requests on successful validation, move a copy of the existing associated documents to an archived revision folder, update the database to increment the asset revision and copy the newly uploaded file to the current working directory of the associated globe project.

# Client Applications

The client application developed in C# allows interaction with the block device, and subsequent requests to the server, whether it be local or remote.

In order to decode the JSON message content from the server request responses, JSON.NET library from Newtonsoft (<http://james.newtonking.com/json>) was used.

When a request is made, the response is decoded and passed to a BrokerRequest object.

namespace Globlock\_Client {

public class BrokerRequest {

public Header header { get; set; }

public Error error { get; set; }

public User user { get; set; }

public Session session { get; set; }

public Globe globe { get; set; }

public Status status { get; set; }

public Action action { get; set; }

public List list { get; set; }

public List<string> listitem { get; set; }

public void updateError(string code, string message) {

this.error = new Error();

this.error.code = code;

this.error.message = message;

}

}

public class Header {

public string type { get; set; }

public string message { get; set; }

}

public class Error {

public string code { get; set; }

public string message { get; set; }

}

public class User {

public object name { get; set; }

public string pass { get; set; }

}

public class Session {

public string token { get; set; }

}

public class Globe {

public string id { get; set; }

public string project { get; set; }

}

public class Status {

public string assigned { get; set; }

}

public class Action {

public string test { get; set; }

public string set { get; set; }

public string abort { get; set; }

public string redo { get; set; }

public string drop { get; set; }

public string pull { get; set; }

public string push { get; set; }

}

public class List {

public string count { get; set; }

public string size { get; set; }

public string root { get; set; }

}

}

Throughout the application the broker concept is used.

## BrokerManager

A broker manager is created on application start. This broker manager, allows the application to store and maintain information and objects required for the application to process requests and interact with the reader device.

BrokerManager brokerM = new BrokerManager();

## Client Application Class Diagram Overview

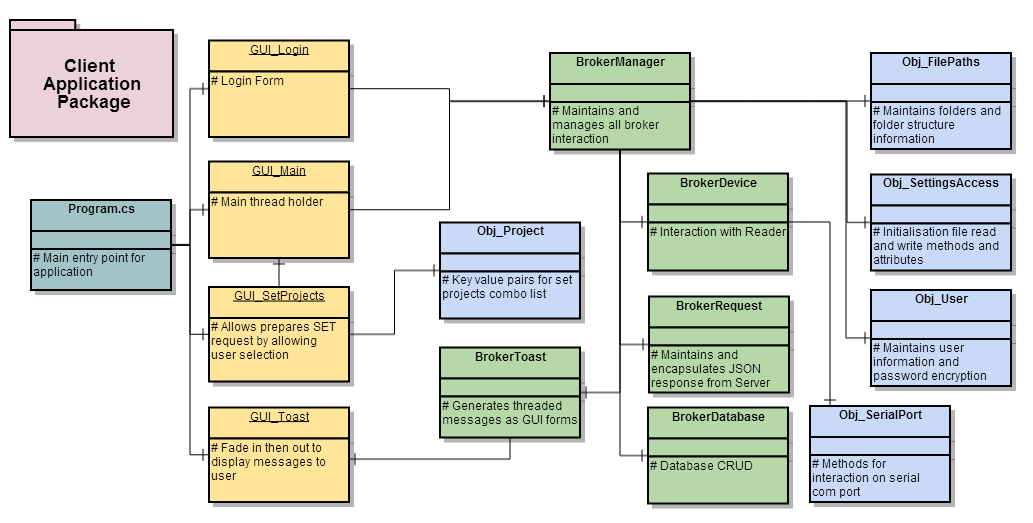


Figure 16

The central starting point of the client application is the ‘Program.cs’ file. This file creates the broker manager object. On construction of the broker, a number of methods are called which go about defining and creating objects to support the client applications interaction with the device and the client applications interaction with the API.

prepareINIFile();

prepareDatabase();

prepareWebClient();

prepareRequest();

As can be seen from the class diagram of the Client Application Package, The broker manager manages a number of other broker objects, as well as objects used to maintain information about specific components such as Users, file paths, serial ports and settings.

### Client GUI

There are 4 main GUI elements in the client application. The low number is due to the fact that as little architecture visible and system interaction needed the better, to allow a streamlined highly abstracted interaction from a user’s standpoint.

#### Main

The main form (GUI\_Main) is there to allow future scope for interactions with the client such as modifying system settings and preferences. In the current state this GUI merely holds the main thread of the application and carries out the necessary actions upon a new globe object RFID being read through the serial port or a response from the server in the form of JSON.

#### Login

Again a simple design approach was used for the login form to prevent unnecessary baggage and interaction on the system. Some sugar has been added that

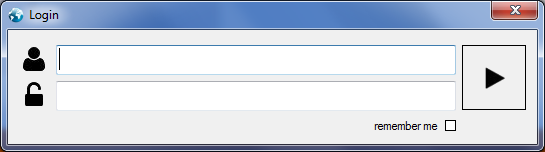


Figure 17

#### Set Projects

The set projects GUI (GUI\_SetProjects) is a simple pop-up form that allows the user to select and set a particular project to associate with the recently read globe object. A list of all the unassigned projects is presented to the user.

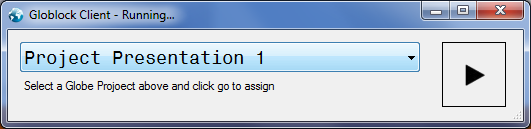


Figure 18

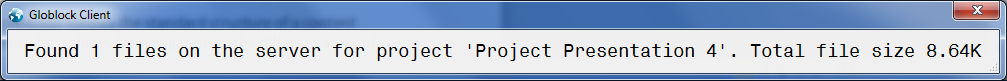


Figure 19

## Management Web Application

The management web application is a web portal that allows the super user to interact and modify the individual components of the system such as users, groups, documents and globes. The website utilises a combination of HTML5, CSS3, Javascript and JQuery.

By using HTML5 and CSS3, the design of the management web application ensures that it follows the practice of repossive design.

*“Responsive Web design is the approach that suggests that design and development should respond to the user’s behavior and environment based on screen size”*

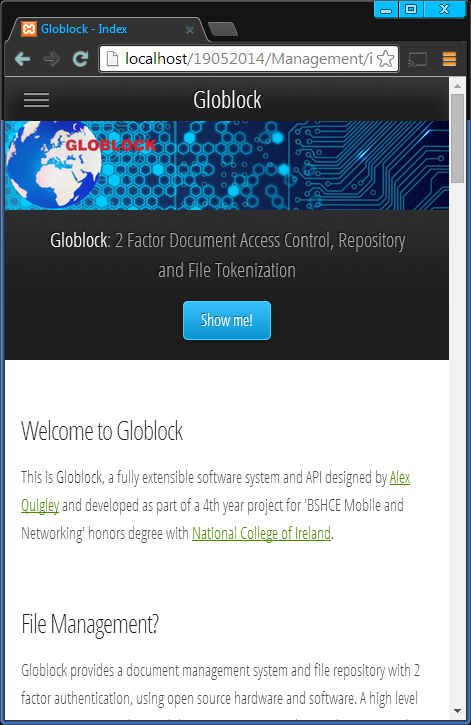
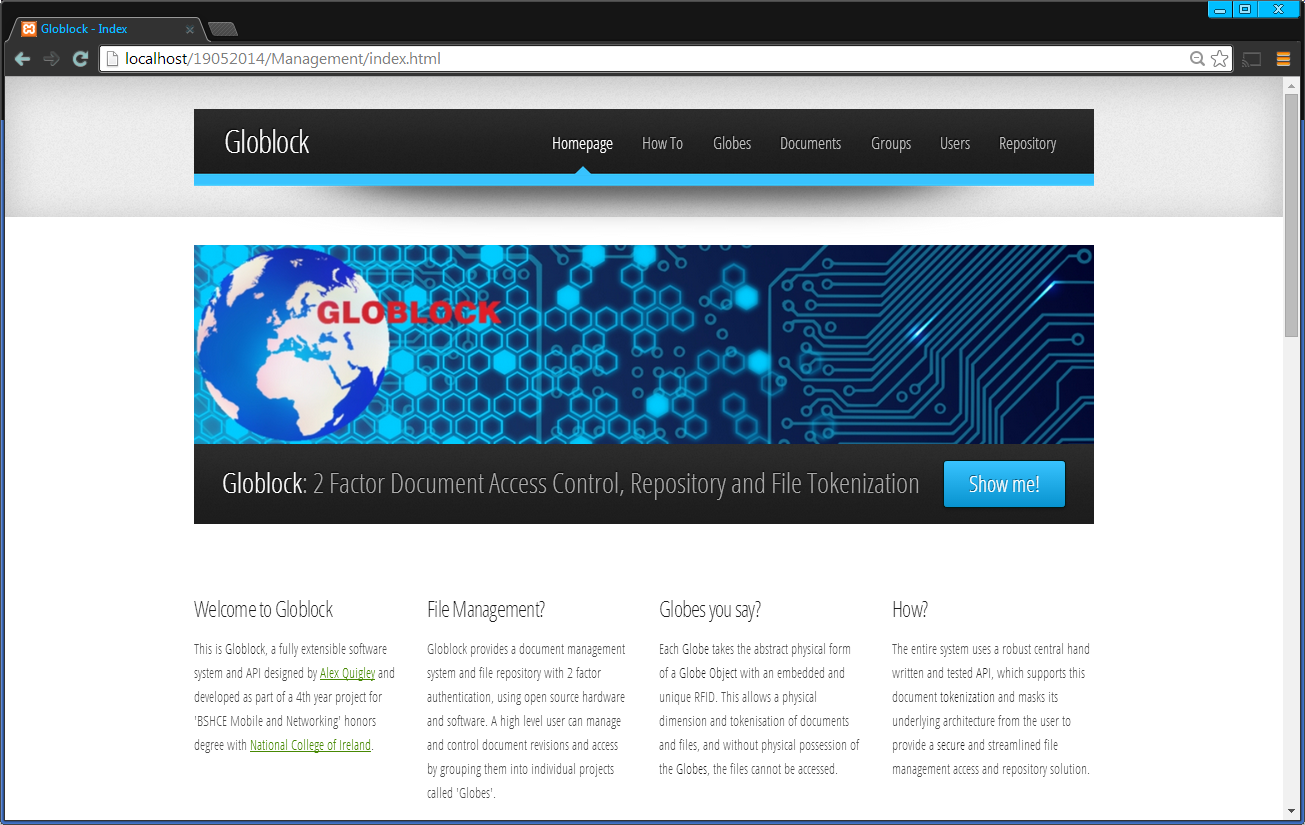
(Knight, 2014)

Depending on the size of the screen the website will look and behave differently.

### Homepage

The homepage to the management web application follows the standard structure of a content management site, allowing for a header menu which allows for navigation to the different areas of the application.

Responsively, the website will reduce its content to edible chunks for the user and this segmentation creates a better user experience when interacting with the application on smaller screens.

|  |  |
| --- | --- |
| Globes The globes pages allow globes to be added and also list current globes that are active on the system. |  |

|  |  |
| --- | --- |
|  | Documents Documents can be created and uploaded here, with a table of all documents currently active on the system. |

|  |  |
| --- | --- |
| Groups User groups can be created and once again groups already added to the system can be viewed in the table also. |  |

|  |  |
| --- | --- |
|  | Users User groups can be created and once again groups already added to the system can be viewed in the table also. |

## Testing & Error handling

The entire development process was completed using an iterative approach. As such, each time a complexity was added to the system, a test case was created.

An example of this is the ‘GLOBLOCK\_test.ph’ file. This file represents a server request and allows calls on all the associated methods within the API to be tested and the results of which are displayed to the tester.

//set up the nested assoc arrays using literal array notation

$header\_array = array("HEAD" => "Response Header", "value" => "");

$error\_array = array("ERROR" => "No Errors", "Error Code" => 0);

As each request was added to the system a method invocation would occur on the test file.

testFunction();

function testFunction(){

otherFunction();

}

This allowed an iterative way of testing files and code additions to the project. For each file that was created a corresponding test file was created and added to the testing stack. Prior to pushing to production these files would be removed.

### Error handling

Again, throughout the application, strict error handling controls are used.

#### API LogWrite

A supporting PHP script was created to allow application information such as transactions to be captured. However, a second more important purpose of this custom script was to write system errors captured by other areas of the API.

In addition to this, depending on the parameters passed into the logwrite method, a security log would be created and written to.

The LogWrite script when initially accessed creates and defines an associative array of values. These values are then used by the system when writing the desired file type and information to the server logs.

$logFiles = array (

"transactions" => array(

"directory" => "../LogFiles/",

"filename" => "transactions.log"

),

"system\_error" => array(

"directory" => "../LogFiles/",

"filename" => "system\_error.log"

),

"security\_err" => array(

"directory" => "../LogFiles/",

"filename" => "security\_err.log"

),

"test\_logging" => array(

"directory" => "../LogFiles/",

"filename" => "test\_logging.log"

)

);

The writeLogInfo method itself takes 2 parameters, 1 of which is optional. A switch statement validates the type and depening on what this type is, will add header information to the error messages or code passed to the method.

/\*\* writeLogInfo

[required] Parameter $info, which defines the information to be written.

[optional] Parameter $type, which defines which log to write to.

'0', Type value, by default, is transaction information log

'1', Type is security information log

'-1', Type is error information log

\*/

function writeLogInfo($info, $type=0){

global $logFiles;

switch ($type) {

case 0:

$info = addHeaderInfo($info, "Transaction");

$location = $logFiles["transactions"]["directory"].$logFiles["transactions"]["filename"];

break;

case 1:

$info = addHeaderInfo($info, "System");

$location = $logFiles["system\_error"]["directory"].$logFiles["system\_error"]["filename"];

break;

case -1:

$info = addHeaderInfo($info, "!!SECURITY!!");

$location = $logFiles["security\_err"]["directory"].$logFiles["security\_err"]["filename"];

break;

case 99:

$info = addHeaderInfo($info, "--TESTING--");

$location = $logFiles["test\_logging"]["directory"].$logFiles["test\_logging"]["filename"];

break;

}

writeToLog($info, $location);

return true;

}

This method also handles which server side log file and location the information should be written to.

When writing the preformatted information to the file, an append is used in order to maintain historic information. The file is also locked for editing using PHP ‘LOCK\_EX’. This locks the file for editing exclusively which ensures an atomic transaction, and prevents any issues and potential misses whereby the potential error/security information is not captured.

/\*\* writeToLog [S06]

Write the information to the logfile. Can be called directly but is unhandled.

[required] Parameter $location, which defines the concatenated location and file name to be written to.

[required] Parameter $info, which defines the information to be written.

\*/

function writeToLog($info, $location){

if(fileExists($location)) file\_put\_contents($location, $info, FILE\_APPEND | LOCK\_EX);

}

### Exceptions Caught and Thrown

Each method in the application if required provides a facility to throw and catch exceptions. Namely in the API this became extremely important due to the limitations of degugging PHP scripts.

if (($fileDetails['globe\_id']==-1)||($fileDetails['asset\_revision']==-1)) {

throw new Exception("Exception Thrown (INVALID ID OR REVISION):");

If an opportunity arose to throw an exception it was used as this is a good practice. It also adds another level of safety and prevents uncaptured errors that are thrown during runtime but not captured, which can lead to catastrophic system failres and prove very difficult to identify and debug.

### Buffering

A considerable issue during development was serial communication not buffereing on time. What is meant here is that although the device is ready to communicate with the client PC, the entire message has not yet been read into memory. In order to allow the process to buffer, a complicated method was needed to allow an entire message to be sent.

This was also a reason for the choice of Arduino over any other reader device. Arduino would facilitate these types of unforeseen system requirements and allow development of workaround and alternative solutions.

/\*

READ SERIAL INPUT DATA

\* Attempts to read serial input from the client and if successful

\* empties the buffer and assigned to 'clientInput'

\* Returns SUCCESS/FAILURE code

\*/ int readSerialInputData(){

int inputComplete = 0;

if(Serial.available()){

if (!(Serial.available()>0)) return GLOBLOCK\_SERIAL\_FAILURE;

do{

bufferedInput += (char)Serial.read();

if (inputByte == GLOBLOCK\_SERIAL\_DATA\_FTR || inputByte == GLOBLOCK\_SERIAL\_DATA\_HDR) inputComplete++;

allowToBuffer(10);

} while((inputComplete<3) && (Serial.available()));

clientInput = bufferedInput;

return GLOBLOCK\_SERIAL\_SUCCESS;

} else {

return GLOBLOCK\_SERIAL\_FAILURE;

}

}

Mid way through the testing of the Arduino device code it was clear that serial communications from the device were not as they should be.

/\*

ALLOW TO BUFFER

\* Delay method that allows the serial to buffer its input and prevent re-read

\*/ void allowToBuffer(int bufferLimit){

delay(bufferLimit);

}

The allow to buffer method was added to delay the system to allow the ID-12LA to read the entire serial communication message. This prevented malformed messages from being read.

On the client application the same issue was occurring. To prevent this, each tag id read from the ID-12LA was encapsulated in a client 3 part object return string or ‘C3PO’.

/\*\*

\* Process Buffer

\* Takes byte list as input and appends the byte stream to the response string

\* if the footer of the message is not read, continue to read

\*/

private void ProcessBuffer(List<byte> bBuffer) {

lastSerialResponse += System.Text.Encoding.ASCII.GetString(bBuffer.ToArray());

if (lastSerialResponse.Length > 10) {

if (lastSerialResponse.Contains("Complete") || lastSerialResponse.Contains("COMPLETE")) {

Console.Beep();

System.Diagnostics.Debug.WriteLine(String.Format("Received Complete: {0}", lastSerialResponse));

STATUS = DEVICE\_STATE\_READ\_COMPLETE;

STATUSMESSAGE = lastSerialResponse;

Console.Beep();

} else {

//Not yet complete, allow to buffer

Console.Beep();

System.Diagnostics.Debug.WriteLine(String.Format("Received Not Complete: {0}", lastSerialResponse));

}

} else {

//Not yet complete, allow to buffer

Console.Beep();

System.Diagnostics.Debug.WriteLine(String.Format("Received Short: {0}", lastSerialResponse));

}

}

When a message thread was envoked on the client application, the byte stream read would be encoded as a string and passed to the processBuffer method. This process buffer method would essentially allow the entire message to be concatenated before taking action.

The C3PO, would contain, Header, Body and Footer information, and until the ‘Complete’ footer information is received, the application would continue as normal.

### User Testing Evaluation

Unfortunately the product did not reach an adequate level of completion and as such real world testing was not completed. However as stated, throughout the development process unit testing and debugging was completed.

The core API which was the goal of the project is fully functioning and has been thoroughly tested using sample data generated.

# Conclusions

Having had a great deal of my time in NCI spent working in Java, I felt C# and C++ would not be a huge leap from what I was used to. My aim with the project was to engross myself in unfamiliar technologies, that are very popular and in a way step out of my comfort zone.

## API

The API was the most difficult part of the entire project and although the current state of the system leaves a lot of room for improvement, the core API itself is a solid piece of well designed and well structured code. Unfortunately getting the API to the state it is in now, have allowed other areas of the project to feel a little unloved.

## Frameworks

I decided not to go with a framework as I felt that the API would be in a better position to carry out the projects underlying goals of abstraction, if designed specifically for my needs. Added to this, I was uncomfortable investing time in framework without a guarantee of results.

On reflection the technologies and languages chosen were not wrong, but the use of a nicely packages framework would have gonbe a long way to improving the final product.

## Arduino

Working with the Arduino gave me great satisfaction particularly when things worked. Prior to the development of the project I was very interested in ubiquitous and pervasive computing. These micro controllers, wearable technology and other ‘IOT’ were everywhere while completing the degree program and when the intial idea came about I jumped at the chance.

I have already set about a number of project I wish to complete, around home automation and control systems, and the project has broken the barriers for me to entire into the brave new world of ubiquitous computing.

## Apprehensions

However small a feat it may be to seasoned professional, I have bridged the gap between software and hardware. I no longer have a fear of the unknown. I no longer have a fear of failure. The process has made me re think my fears about entering into something completely new. I no longer have a fear of HTML5, PHP, C++, C# and all the other technologies used during the project.

I may not have been given the tools necessary to start with a company and immediately make an impact I have however learned a valuable lesson about myself and my abilities.

## Time Management

This has been, without a doubt the greatest difficulty for me during the project. For the past year I have been on temporary assignment with a different group in work. This has been a very demanding and challenging experience, and unfortunately, my otherwise impeccable results from the first three years of college with NCI, and my dedication to giving 100% to the degree has been tarnished.

I had asked at the start to be provided with a supervisor who would give a very small amount of time, on a regular basis as I knew that this would be a huge issue for me. My future goal would be to be in a position as a scrum master or project manager on a medium to large scale project. I have only myself to blame for any shortcoming’s on the project but looking forward it has been a learning experience, and in that regard a very rewarding experience.

I will have to rethink how I will achieve these goals I set out at the start of the degree. However, having all the pieces of a project floating around in my head has been challenging but I think I managed this well which is a valuable skill.

I feel I know everything about how this system ties together. Something I would not have been able to say, had I used a framework or more packaged technology. I think going forward finding a balance is key.

If I could manage my time as well as the ideas, I think I could be a very successful candidate in the future.

# Further development or research

The project itself is far from a complete product and further development could see the product being used as commonly as drop box or some other file repository.

Around the areas of security, permissions and groups could be utilised more if the system were to be used for a large amount of concurrent users.

The client application could also be ported to a Linux environment and used allow for a broader scope of potential users.

## Research

I will continue to research the marriage of hardware and software and how people interact with systems and devices around them.

# References

Massimo Banzi. (2014). *What is Arduino.* Available: http://www.arduino.cc/. Last accessed 2014.

RFID Journal LLC. (2014). *What is RFID.* Available: http://www.rfidjournal.com/faq/show?49. Last accessed 2014

JASON LENGSTORF. (2014). *JSON: What It Is, How It Works, & How to Use It.* Available: http://www.copterlabs.com/blog/json-what-it-is-how-it-works-how-to-use-it/. Last accessed 2014.

Unknown. (2014). *JSON Viewer.* Available: http://jsonviewer.stack.hu/. Last accessed 2014.

Friends-of-Fritzing foundation. (2014). *Homepage.* Available: http://fritzing.org/home/. Last accessed 2014.

John Boxall. (2014). *Arduino tutorial 15a – RFID with Innovations ID-20.*Available: http://tronixstuff.com/2013/02/26/arduino-tutorial-15a-rfid-with-innovations-id-20/. Last accessed 2014.

ID Innovations. (2014). https://www.sparkfun.com/datasheets/Sensors/ID-12-Datasheet.pdf.*ID-12-Datasheet.pdf*. 1 (2), P3.

byteparadigm.com. (2014). *Introduction to I²C and SPI protocols.*Available: http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/. Last accessed 2014.

Kayla Knight. (2014). *Responsive Web Design: What It Is and How To Use It.* Available: http://www.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/. Last accessed 2014.

# Appendix A - Project Proposal

# Appendix B - Requirement Specification

# Appendix C - User Manual

# Appendix C - Monthly Journals

## Month: September

### My Achievements

This is the first entry in the Journal. Up to now I had been attending meetings with colleagues in Intel. The hope was that I would be able to use a project that was posted to the organisations internal developer opportunity tool, requesting from small form factor and mobile I.T. department within Intel. The project was to prevent personnel using handset cameras within their android phones that were rolled out recently, to protect IP while in the cleanroom facility.

Unfortunately the powers that be have decided that there is a skills mismatch and the project would not be suitable for me as the timelines do not coincide with that of the projects deliverables.

Instead I am forced to go with my original project idea which had been originally called file stack. I have decided to go with globlock, and have purchased this domain name as it may be needed for a landing page and or other functionality later on in the projects lifecycle.

### My Reflection

I have to admit I was very disappointed not to be appointed on the companies project however In a way there will be less external pressure. This can be a good thing or a bad thing. I will continue to work hard and develop my own idea in the hope that my IT based colleagues will take notice and this may help me in a future career change to their department

I have the general concept solidified in my head and I am eager to get started with the project planning processes and steps.

### Intended Changes

As stated I have had to change my focus to my own personal idea. This has been clear in my mind since its initial conception back in January but has grown organically since then. I haven’t changed the layout of the initial information attached, but for the project proposal due by the end of the week, I will include all the required changes and functional components needed.

## Month: November

### My Achievements

This month I have narrowed down the technologies that will be used for the development of the front end management suite side of the system. I will mostly be using a combination of PHP, HTML5 and CSS3 to provide the necessary interface to securely upload documents to the server and manage user accounts, globes and file permissions. I have carried out a lot of research on possible frameworks I could use to lessen the amount of code I would need to generate in order to complete the project. However, the frameworks I have found are already a complete solution to file management which would be too bulky for my own requirements.

I have also created a database structure in MySQL which will be used. There are a large number of tables and I was careful to take a considerable amount of time to ensure the DB structure will hold up and require very little change before the end of the project.

I have generated the PHP file upload code and am hoping that what has been developed so far will be a sufficient prototype for the mid-point presentation.

### My Reflection

I have found it very difficult to manage my time successfully. As a part time student, I am also working full time and spare time is not great and there were a number of deliverables last month for a separate module that took preference and prevented me from making a dent in the project and its development.

Also work commitments have also put me in a position whereby when I return from work in the evenings I have not had the drive and commitment to work on the project.

I have met with my supervisor very briefly only once to discuss the project, and it appears that future meetings will not be as beneficial as I had first thought. Going forward I will have to put more effort into the project and manage my time better as I realise time is running out and deadlines approaching and the level of commitment I have shown to date has not been enough, and is well wide of the mark I have set myself in first, second and third year, and the level of commitment I had intended this semester.

I have felt quite deflated with my lack of progress. However, the more code I write, the confident I become and I am hoping that by mid January, I will have the lions share of the code written, and a basic yet fully functioning prototype working, to allow me to carry out some of the tests I stated would be completed in the requirements spec.

### Intended Changes

As stated time management and the amount of work completed to date has been a concern. I plan on addressing this by removing distractions and taking some time off work to work on the project over the next few weeks. Hopefully if I can get a particular section of the system working, it will give me the confidence and drive to maintain the momentum and complete the project as intended.

## Month: December

### My Achievements

This month I have started to design the API that will act as a middle layer and facilitate communications between the client PC and the server. I have also started t work with the hardware and have been able to get both serial communications between a client PC and the Arduino device as well as rfid readability using a iteadstudio NFC shield.

I have however found that the required frequency of the tags needed to be read will require a different type of RFID chip/shield and this has slowed my progress down a little.

Also, the windows application side of the system has shown that a windows service, has recently been changed to prevent access to GUI applications by default which is something I will have to address.

### My Reflection

Although my progress has been slow, since the presentation I have been enjoying working on the project which had previously been a burden. In particular the hardware elements such as the Aduino and a number of components I have been using to get used to the platform, has been a really educational experience and one that after the degree, I will pursue with greater focus.

I started by following some very basic tutorials on-line and found that the platform is very interesting and surprisingly easy to use and get to grips with. On reflection, I think these very accessible hardware components should be encouraged more from the likes of the college as they are a great way to get people involved in electronics and communication from an entry level standpoint.

I am confident I will have a fully functioning system in time to test thoroughly before the approaching deadlines.

### Intended Changes

I have sourced an ID-20 RFID reader chip from sparkfun, and I am awaiting delivery. When I start to get the rfid id sent to a windows application, and that application communicating with the api, I will then be in a better position to address the issue of the windows service being unable to start a windows based application.

I am also looking at different housing for the Arduino device to allow it to be viewed as a more marketable product.

## Month: January

### My Achievements

The API is coming along nicely and the design appears sound from what tests I have implemented. I have also started on the Client application which looks to be more difficult than first anticipated. I have been working with the ID-20 and the ID-12LA and the 12LA appears as though it will suffice for the reader device.

### My Reflection

I am struggling considerably with time management as the new as the current demands from work are causing a strain, and preventing me from moving forward at the pace I had originally planned.

### Intended Changes

I will make a bigger effort to meet the proposed deadlines set out in the project plan. I will also look at using the a local database with the client application to reduce the load on the server requests.

## Month: February

### My Achievements

Device is reading fully and communicating with a basic C# application I have written.

### My Reflection

I feel the hard part is over in terms of the hardware. I had difficulty with the wiring and understanding how the device actually communicates but have made headway. I am struggling considerably with time management as always.

### Intended Changes

I will look at implementing a broker style architecture to ensure the same response is delivered to the client each time regardless of success or failure.

## Month: March

### My Achievements

The API is now fully operational in terms of delivering a session token at the request of the client application. The broker style architecture is a success and JSOn is being returned.

### My Reflection

Looking at the timeframe it looks as though I will not have the allocated time previously thought to design and develop the front end website.

### Intended Changes

I need a method to decode the JSON on the client so that it can be accessed more efficiently.

## Month: April

### My Achievements

I am successfully pulling files and manipulating them on the server as expected.

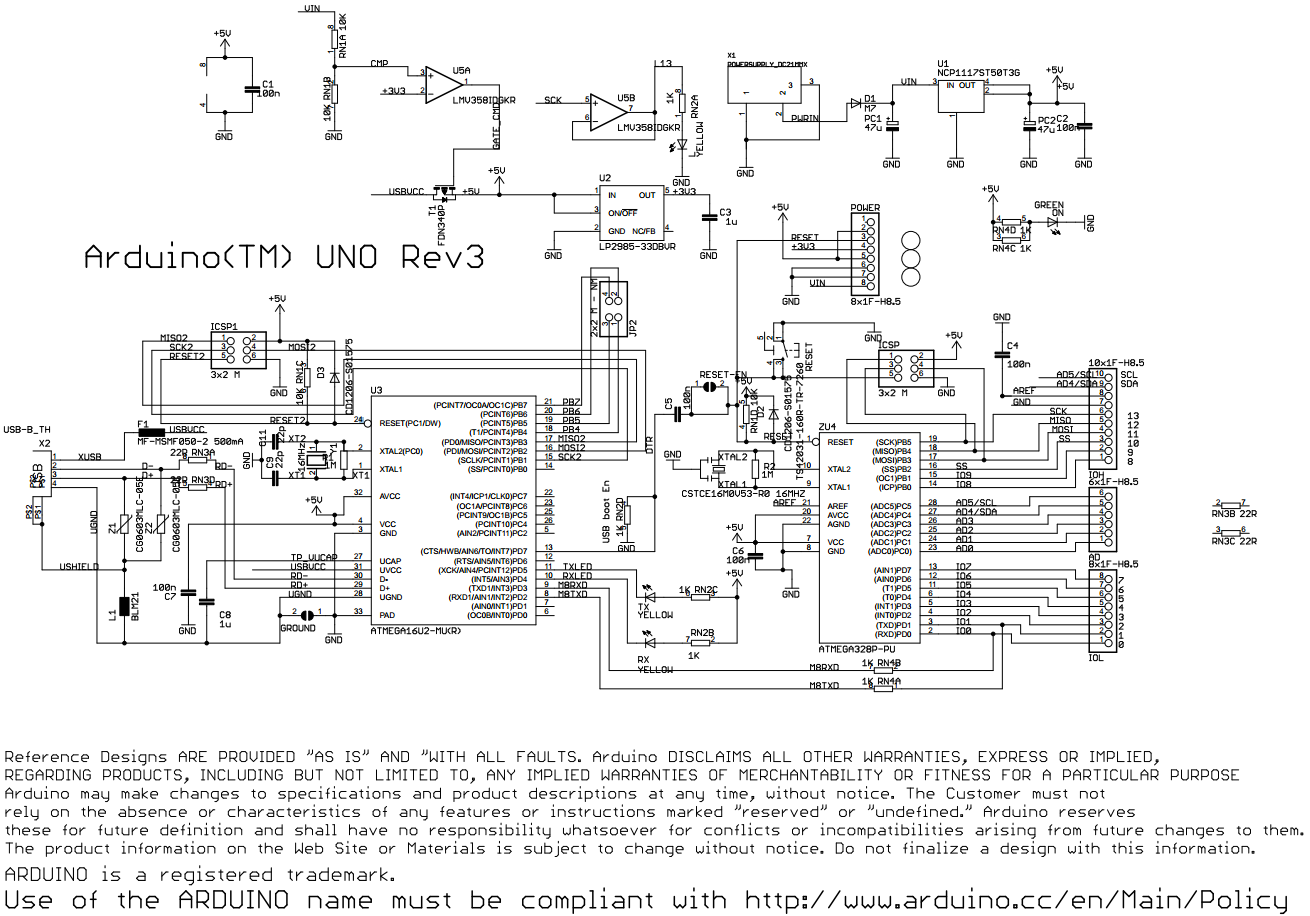
### My Reflection

The API is considerable difficult to test as each iteration requires publishing to the Apache web server prior to testing. I perhaps should have gone with a framework for the API but I feel I have gained some great experience in PHP design and development, that I would not have gained had I used a framework.

### Intended Changes

I will need to start spending more time on the client application.

# Appendix F - Arduino Schematic



Appendix G -

# Other Material Used

References