# Front Page

School of Computing and Mathematics

Teesside University

Middlesbrough

TS1 3BA

SCIPA

Software Controlled Industrial Process Automation

BSc (Hons) Computing

[Date of Completion]

William Whitehead

M2006199

Supervisor: Jim Longstaff

Second Reader: [Second Reader]

# Abstract

Process automation is a key system in most of today’s industry; it’s able to reduce costs, maximise efficiency, improve safety and, amongst many other benefits, is able to ultimately improve the profitability of a process.

This project follows the design and development of a supervisory control and data acquisition system (SCADA) that is able to monitor and self-control a set process loop via a user defined configuration. The project also contains an investigation into the implementation of a ‘big data’ database using MongoDB.

# Acknowledgements

## Teesside University

First and foremost, I’d like to thank Jim Longstaff and John Goodge whose support as my supervisors has been second to none, respectively this year and last. Without their help, final year would have been much more of a challenge.

I’d also like to thank Mansha Nawaz, whose obscure teaching and mentoring style has proved to be invaluable. His support through the Advanced Database Systems and Computing Project modules has been thorough, throughout and very much appreciated.

## DuPont Teijin Films

Many thanks to both Andrew Doonan and Andrew Taylor whose perfectionisms, intolerance to ignorance and intrigue to new ideas was passed down to me during my time on placement. Process automation is a field of computing I take great interest in, and without my time with the company, I wouldn’t be aware of any of the concepts of control theory, never mind be able to design and implement a ‘big-data ready’ SCADA system.

For their guidance, support and friendship, I thank them both.

# Contents

[Abstract i](#_Toc447544649)

[Acknowledgements ii](#_Toc447544650)

[Teesside University ii](#_Toc447544651)

[DuPont Teijin Films ii](#_Toc447544652)

[Contents iii](#_Toc447544653)

[Table of Figures v](#_Toc447544654)

[1. Introduction 1](#_Toc447544655)

[1.1. A Brief Overview of Process Control and Control Theory 1](#_Toc447544656)

[1.2. Process Control via Process Loops 1](#_Toc447544657)

[2. External Hardware and Software 2](#_Toc447544658)

[3. Project Methodology 3](#_Toc447544659)

[4. Problem Domain and Example Deliverable Solution 4](#_Toc447544660)

[5. Research Stage 5](#_Toc447544661)

[6. System Specification 6](#_Toc447544662)

[7. System Design 7](#_Toc447544663)

[8. Version Control 8](#_Toc447544664)

[9. Project Development 9](#_Toc447544665)

[9.1. Internal Projects and Solution Structure 9](#_Toc447544666)

[9.2. Inbound Data Handling and Reading 9](#_Toc447544667)

[9.3. Controlling the Process with Rules 9](#_Toc447544668)

[9.4. Outbound Data Handling and Writing 9](#_Toc447544669)

[9.5. Data Storage and Management 9](#_Toc447544670)

[9.5.1. Relational Databases within SQL Server 9](#_Toc447544671)

[9.5.2. NoSQL Databases within MongoDB 9](#_Toc447544672)

[9.6. Business Intelligence Tools 9](#_Toc447544673)

[9.7. Backend and Automated BI 9](#_Toc447544674)

[9.8. Frontend and User-Focussed BI 9](#_Toc447544675)

[10. Testing 10](#_Toc447544676)

[10.1. Acceptance Testing 10](#_Toc447544677)

[10.2. Unit Testing 10](#_Toc447544678)

[10.3. Component Testing 10](#_Toc447544679)

[10.4. Performance Testing 10](#_Toc447544680)

[11. Project Evaluation 11](#_Toc447544681)

[11.1. Project Goals 11](#_Toc447544682)

[11.2. Next Steps and Future Improvements 11](#_Toc447544683)

[11.3. Recommendations 11](#_Toc447544684)

[12. Critical Review 12](#_Toc447544685)

[13. Conclusion 13](#_Toc447544686)

[14. References 14](#_Toc447544687)

[15. Appendices 15](#_Toc447544688)

[15.1. Project Proposal 15](#_Toc447544689)

[15.2. Poster Presentation 24](#_Toc447544690)

[15.3. Arduino Sketch Source Code 26](#_Toc447544691)

[Trending Application 26](#_Toc447544692)

[Input Acceptance Application 26](#_Toc447544693)

[Basic IO Application 26](#_Toc447544694)

[Other 26](#_Toc447544695)

# Table of Figures

# Introduction

## A Brief Overview of Process Control and Control Theory

Process automation is a fundamental element in most of today’s industry. The ability to actuate hardware based on a user-defined configuration and real-world sensory data is a topic both businesses and mathematicians have invested significant amounts of time and money in, especially where such systems can be used to enhance safety, efficiency and profitability.

Control theory is the principle of controlling a process based on external factors and ‘rules’. The purpose of such is to “achieve optimal process operation despite the presence of significant uncertainty about the plant behaviour and disturbances” (Engell, 2007, p.203). Control theory, therefore, requires any application of such to allow unexpected changes in the incoming data and be able to handle the change appropriately. An example may be the sudden drop in a temperature reading of a vessel, the control system must be able to interpret the unexpected change and automatically implement a remedy via any preconfigured routes, be that software based or mechanical.

This project focussed on the development of SCIPA, a Supervisory Control and Data Acquisition system, often abbreviated to ‘SCADA’. SCADA is a term that “generally refers to an industrial computer system that monitors and controls a process” (Subnet, 2015) where a process contains one or more process loops.

Industrial processes often implement specialist hardware and control devices in order to maintain a steady production environment which are, at both a manufacturer and a vendor level, often controlled via Programmable Logic Controllers, often referred to as PLCs.

## Process Control via Process Loops

A process loop is the term used to describe a single algorithm that runs repeatedly within a process. A basic thermometer, for example, has a single process loop that loops over the following algorithm repeatedly:

* Read ‘local temperature’;
* If ‘local temperature’ is less than the ‘set point’ send ‘ON’ command to boiler;
* If ‘local temperature’ is more than or equal to the ‘set point’ send ‘OFF’ command to boiler.

Industrial process often consist of many process loops, with the more complex systems housing collections of process loops within PLCs, which in turn are controlled within a parent process loop.

A process loop within this project is taken to mean a ‘closed loop’ in that “an operation, process, or mechanism is regulated by feedback” (Merriam-Webster, 2015). By allowing both data input and output from the application, SCADA systems like SCIPA are able to allow users to define ‘rules’ in order to manage and automate actions taken by a computer for process automation.

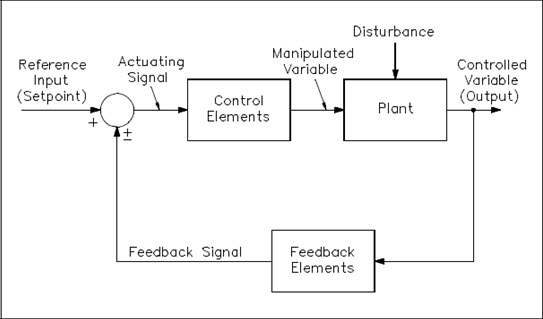


Figure 1.2.1 Closed Process Loop (TechTransfer, 2014).

Figure 1.2.1 shows the process in which a process loop is executed. A reference input, commonly referred to as a ‘set point’, is passed to the system alongside live process data, shown as ‘feedback signal’. The control element of the loop, which in terms of this report is a SCADA system, performs one or more actions in order to manipulate and control the variable which is then output back to the ‘plant’ or process environment. The loop will repeat forever, with the SCADA system controlling the values output to the plant in an organised manor.

## Project Outcome

The aim of this project is to design and build a fully working process control system that has the ability to read live data, output data or commands to both software and hardware based components based upon process rules and provide users with useful and meaningful ways to investigate and visualise process data.

The key targets are to:

* Read data via common industrial methods, such as Serial or Database;
* Allow users to create complex rule sets for data;
* Output either process data or commands based upon the rules;
* Store live data in a relational database for fast-access and reporting;
* Store historical data using a long-term data repository for data archiving;
* Grant access via an industrially-design HMI.

This report is split down to cover all of these aspects in detail, focussing on the real-world practises and the academic benefits of working with such.

# External Hardware and Software

Process control systems are often able to read data from and write data to a selection of platforms. In order to emulate a real-world and industrial environment, SCIPA has been designed to communicate with live data from databases that allow access via OLE, SQL or ODBC drivers, as well as flat files based in Unicode, ANSI and ATF-8, and serial data, including devices that require RTS (Request To Send) and DTR (Data Terminal Ready) signals.

In terms of hardware connections, the prototype application can successfully communicate with a multitude of hardware devices via RS32 and USB connections, with testing completed using a collection of Arduino Uno devices to simulate serial process data and recievers.

For acceptance and integration testing with external software, basic C# applications have been developed to simulate process data handlers using flat text files based on the local development computer.

Further implementation of SCIPA was conducted using the Microsoft Azure IaaS (Infrastructure as a Service) platform for hosting SQL Server databases, with NoSQL being handled by a local instance of MongoDB.

MongoDB has been chosen as the long-term data store for this project because it is the fourth most popular database engine in the world, ranking first in the world as a non-relational database engine (gmbh, 2016). By deciding to implement MongoDB, the project has used an industry-standard tool and acted as a strong academic introduction to such technology.

# Project Methodology

The development of this project’s artefact was conducted, for the most part, using the Agile methodology. The principle of the Agile development method is that projects are completed by using an iterative cycle of production. Each element of the developed system will be designed, built and tested in its own right.

“Individuals and interactions over processes and tools  
Working software over comprehensive documentation  
Customer collaboration over contract negotiation  
Responding to change over following a plan

That is, while there is value in the items on  
the right, we value the items on the left more.”

(Agilemanifesto.org, 2015)

There are several implementations of the Agile methodology, all of which must follow the above quote which lays out the requirements for what Agile should mean to developers. For this project, the Scrum technique has been used as far as was reasonably possible. Scrum is usually implemented for projects where developers are able to work within teams of six or seven, however, the ‘sprint’ technique was used as a strong foundation on the iterative development of the system. Each of the required elements of the system, as laid out in the project proposal formed the product backlog, with each ‘product’ housing a set of sprints. Sprints were completed within the allotted time and committed to the repository.

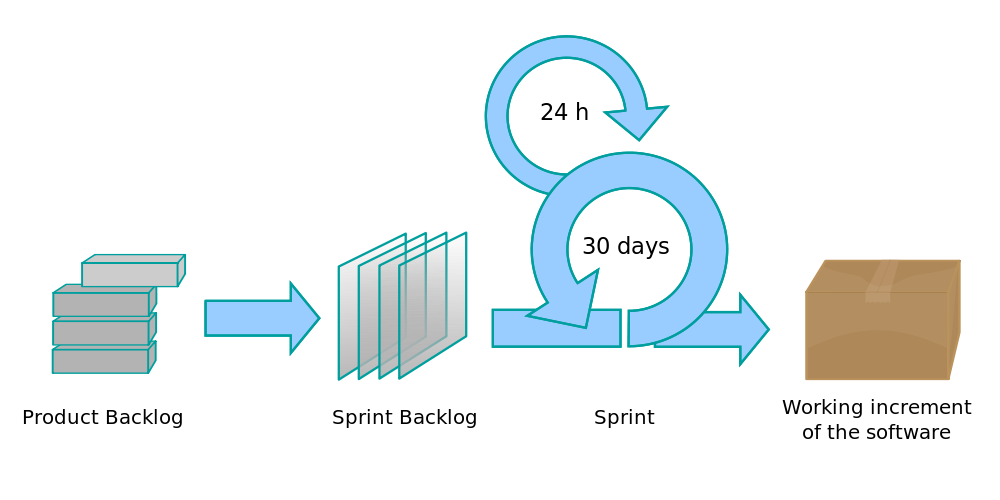


Figure 3.0.1 Scrum development methodology (Lakeworks, 2009).

The key idea of products and sprints within Scrum is that each product is a self-contained, fully tested and working element of the application, made up over one or more sprints.

## Analysis

As process control is often considered a topic more suited to students of Masters or Doctorate courses focussed on mathematics or chemical engineering, the analysis stage of this project focussed on reading materials surrounding basic mathematical principles of control theory as well as engineering practises in terms of process development in the real world.

Steve Mackay, the foundation Dean of Engineering for the Engineering Institute for Technology, gives complimentary online seminars which formed a more visual and interactive basis for researching the requirements of a SCADA system. These seminars formed the foundations for the project’s research, giving insight into the necessary engineering principles and industrial-expectations for Human Machine Interfaces (HMIs).

Further research was conducted to fully investigate best-practises for multithreaded applications, especially when working with several data sources and backend services within a time critical environment.

Alongside the research with regards to multithreading, several applications were developed both for the development computer to simulate database and flat file communication on both the inbound and outbound channels, as well as Arduino sketches, small looping applications designed to output trending values within a given range. The sketches designed for the Arduinos submitted serial data every second with values trending either positively or negatively and were used to research and test both Windows’ and C#’s ability to accept data from several sources simultaneously.

The more technical programming challenge of working across threads was combatted via the further research into method invocation which was required to ensure that the User Interfaces could show live data without becoming overwhelmed with data processing, rendering them unfit for purpose.

## Design

As the design of the application architecture directly impacts the efficiency of the program, the layout of the project played an important and integral part of the design phase. The methodologies used to fully plan, articulate and progress the design of the system

Discussing the elements of such, Learn.org states that “Design methodology stresses the use of brainstorming to encourage innovative ideas and collaborative thinking to work through each proposed idea and arrive at the best solution” (2003). Brainstorming, basic sketches and technical drawings were all used to design the architecture of the application as a whole, the functionality for each respective user interface.

A more key element of the design process for this project was the algorithm to be used in order to effect a process loop. An “algorithm is a statement about how a problem will be solved” (Beech, no date), with the problem in this instance being the understanding of the live process value. The required algorithm must implement many of the algorithmic design concepts, including sequence, decision and repetition constructs in order to prove effective and fit-for-purpose. Given that an algorithm is the process of solving a problem in a finite number of steps, the design of the algorithm is key in removing unnecessary required processing of data.

The product is designed to be distributed over several logical tiers so as to allow cleaner separation of concerns and reduce unnecessary dependencies. This approach, known as N-Tier, is designed to “separate processing into discrete tiers that are distributed between the client and the server” (Microsoft, 2016) and is a modern approach to application development.

## Implementation

Implementation of this followed the project’s methodology as a whole, using Scrum. Each incremental build of the product was a further step towards the completion of one or more of the products from the product backlog, and each step was in accordance with the relevant product’s design.

As the project was designed to use a layered architecture, the individual projects that make up SCIPA as a whole were initially implemented in isolation of all other layers so as to remove any unrequired dependencies being formed. As the system grew ‘tighter’ and projects required access to one another in order to be of intended use, projects were conglomerated into a single solution.

During both the isolation and ‘packaged’ phase for each project within the solution, the system was stored on a secure, cloud-based backup platform with automated version control so as to ensure that any changes made could be easily ‘rolled back’ within a matter of seconds. Formal versioning and version control was also part of the implementation phase, in that each implementation iteration was committed to a Git repository to provide a further layer of developmental security.

The Milestone project management methodology was used throughout the project to provide a solid basis and backup ‘guide’ during the implementation. The milestone approach ensures that every milestone that is met is committed to a form of permanent backup storage.



Figure 3.3.1 Milestone implementation methodology.

For this project, a slight modification was made to the chosen methodology so as to better incorporate the other backup and versioning systems in place.



Figure 3.3.2 Modified Milestone implementation methodology.

The revised implementation of the milestone methodology allows for committals to the repository, which is a much more frequent exercise than permanent backups of all of the stable versions.

## Testing

Testing has been a continuous and rigorous element within this project, with unit and observation testing being conducted at regular intervals for each product, and often sprint, within the relevant Scrum backlog.

Unit Testing forms an entire project file within the overall solution, and is used to ensure that individual components of the application return values as expected. Expected values are prepared prior to the execution of the application via a manual run through of that element’s algorithm. The unit tests will pass should the system return the same result as the expected value, otherwise these will fail.

Observational testing is the second form of testing that was used during development. This method requires visible input data, often by hand, being entered into the application with any associated debugging messages or printed statements being checked to ensure the values are consistent. The project has been implemented with a built-in logging service which assists in this testing type.

Acceptance testing is an expansive term used to describe a multitude of testing types. The definition that acceptance testing has been taken to mean as part of this project is as follows:

“A technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users” (Acceptance testing, 2016).

In this respect, SCIPA’s acceptance testing focussed primarily on the interoperability of each tier within the solution, especially those that have a direct reference, and thus, dependence on another tier.

# Problem Domain and Example Deliverable Solution

# Research Stage

text

# System Specification

Text

# System Design

Text

C:\Users\Will Whitehead\Downloads\SCIPA Loop Diagram (1).png

# Version Control

text

# Project Development

Text.

## 9.1. Internal Projects and Solution Structure

Text

## Inbound Data Handling and Reading

Text

## Controlling the Process with Rules

Text.

## Outbound Data Handling and Writing

Text.

## Data Storage and Management

Text.

### Relational Databases within SQL Server

Text.

### NoSQL Databases within MongoDB

Text.

## Business Intelligence Tools

Text.

### Backend and Automated BI

Text.

### Frontend and User-Focussed BI

Text.

# Testing

## 10.1. Acceptance Testing

## 10.2. Unit Testing

## 10.3. Component Testing

## 10.4. Performance Testing

# Project Evaluation

## 11.1. Project Goals

## 11.2. Next Steps and Future Improvements

## 11.3. Recommendations

# Critical Review

# Conclusion

# References

Paste the relevant references here.

# Appendices

## 15.1. Project Proposal

**Working Title**

SCIPA – Software Controlled Industrial Process Automation. *Pronounced ‘Skipper’.*

**Overview**

Process automation is a key system in most of today’s industry. The ability to actuate hardware based on a user-defined configuration and real-world sensory data is a topic I take great interest in, especially when the data collected can be used to improve process and/or business efficiency, as well as safety.

While a fully functional, industry-ready Process Control System is unfeasible during the timeframe, it is the aim of this project to develop a system that is a working example that can effectively interact with the real world in order to control a set workflow, otherwise known as a process loop. Such systems are referred to as SCADA systems, an acronym for Supervisory Control and Data Acquisition.

Data is at the heart of this proposal. The collection, analysis, control and acting upon such is the key aim of the project, and the final product will essentially evaluate data from a range of sources and, based on a user-configured instruction set, will actuate hardware or software based upon those readings.

A very primitive example would be the use of a thermostat to control room temperature. If the temperature reading for the room drops below the user-set value, the heating system is activated.

**Project Rationale**

Collecting, using and storing data is of great personal interest, with the majority of my placement year being focussed on industrial data and the optimisation of database and query structures, to better provide the business with efficient use of its process and quality data.

Process control systems start small but can scale up to be highly distributed environments in which large, complex engineering workflows take place. At each stage, real-world data is fed back to the controller to ensure each control loop operates as desired.

As companies look for better ways to make their own process more efficient and cost-effective, it is important that they’re able to interact with the data collected from their equipment to improve their flow. Designing, building and improving a process control system, in the form of a SCADA, will help me better understand the interaction between control loops and the real world, as well as improving personal knowledge on effective use of the available SQL servers and data reporting; an area in which I’ll be seeking future employment.

Process control, process dynamics and Process Control Systems are often the focus of Chemical Engineering and/or Mathematic students, and as such, this project will aim to effectively use sensory data to evaluate and interact with the real world on a basic level, building up simple Process Loops, with the use of advanced mathematics and complex engineering principles not being considered as part of the work.

A ‘Process Loop’, for this project at least, will be taken to mean a basic set of Input/Output from the application. For example, one loop may be reading the temperature of a water tank; if the temperature is below a certain level, a heater must be switched on. “A closed control loop exists where a process variable is measured, compared to a setpoint, and action is taken to correct any deviation from setpoint” (Process Control Fundamentals, 2006). Open control loops allow automatic intervention at set intervals where process values are irrelevant. For example, a valve may be opened at 12 minutes into a process to allow cool water to flow to the cooling system to prevent overheating, regardless of the temperature of the process fluid.

**Technical Overview**

**Key Areas for Investigation**

As part of a project like this, there are several key areas of investigation.

* Process Loops: Control Loops and Feedback Control Loops
* Relational Time Series Databases and Design
* Serial Data Communication
* Open Platform Communication (OPC)
* Programmable Logic Controllers (PLCs)
* Database Servers (including Reporting and/or Analysis Services)

Most Process Control Systems used in industry will be used to interact with PLCs. This communication, for modern systems at least, will be conducted over OPC. By investigating these two topics, it will be possible to evaluate their effectiveness and consider their implementation into the application.

It is likely that the design of an application that allows users to dynamically interact with sensory data and actuators will be a large task, and as such, I’m hesitant to declare that the PLC communication via OPC will definitely be part of the final artefact.

Contrary to the above however, thorough research into the applied side of Process Loops will be included. The data from both the sensors and the actuators, whether they’re hardware or software based, will be stored in a relational database allowing thorough investigation and historical reporting on the process. It may also be interesting to allow retrospective process runs.

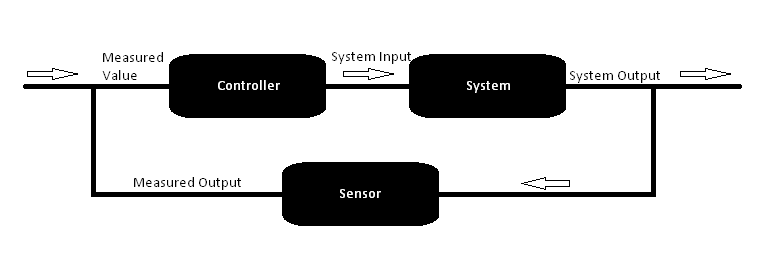
**Background Research**

Process control was first introduced over two thousand years ago, though the first official recording of a feedback control loop was Ktesibios’s water clock from the third century B.C. (Norman, 2015). This clock kept time by regulating the water level in a vessel and thus, by design, the water flow from that vessel, requiring no external input.

Building on this primitive example, control theory in a mathematical form has been built upon heavily, leading to a new era of controlled applications. In 1868, James Maxwell wrote a paper entitled “On Governors” that was able to discuss and prove the instabilities of the “flyball governor” using differential equations (Maxwell, 1867). The “flyball governor” was a speed regulator for steam engines, designed by James Watt in 1788. Following Maxwell’s publishing, the next 100 years was pivotal in the field, progressing stochastic, robust, adaptive and optimal control methods throughout the period between 1950 and 1980.

Following 1980, Control Theory has been recognised as a unique discipline instead of a sub-section of chemical and electrical engineering due to the largescale improvements in profit, safety and efficiency that automated processes can facilitate.

In general a control loop is diagrammed as follows, and this is how my SCADA system will receive, interrogate and actuate changes:



**Literature Review**

A key resource available is ‘Practical Process Control for Engineers and Technicians’ (Altmann and Macdonald, 2005), which focusses on the practical side of process control and SCADA opposed to the mathematical background of the subject. The book explains key phrases in detail, as well as providing real world scenarios, some of which could be used to test the system once developed. Focussing on control concepts opposed to control theory means that it will be possible to quickly understand otherwise difficult concepts without a thorough background in applied mathematics and control theory. However, where further numerical involvement is required, the ‘Data Acquisition and Control Handbook’ (Keithley, 2001) will provide further insight into complex concepts.

The chemical engineering book, ‘Process Dynamics and Control’ (Seborg, Edgar and Mellichamp, 1989), is a leader in Process Control System literature. Providing a firm basis for research, design ideas and this project generally.

The journal, ‘Scientific American’, housed a thorough article on Control Theory (Bellman, 1964) which was referred to when developing the plan for this project, particularly the definitions given for control loops. The journal has since produced published further articles on process control and the theory of such and will therefore become a valuable, up-to-date resource during project development.

**Development Methodologies**

There are several applicable methodologies I could use during the development of the project, namely Scrum, Waterfall and Prototyping (Lindstrom and Jeffries, 2004) (Pree, 1995).

While Scrum is primarily designed for team-based projects, there are aspects that can be well implemented throughout this process. The idea of sprints and ‘daily’ meetings will help keep work on track and focussed both in terms of project completion and scope. A good project plan and an effective use of Scrum will aid the project greatly.

The Waterfall methodology would mean developing the end product in one large ‘sprint’. Whilst this may mean completion would be done sooner, it also opens the project to the possibility of scope creeping and/or unexpected delays, in that there would be no formal review sessions on the work completed and therefore no formal list of priorities in terms of the work to be done.

Whilst Scrum is a member of the Agile family, the agility methodology means that certain aspects of the development cycle are seen as more important than others. Agile software development must follow the official manifesto:

“Individuals and interactions over processes and tools  
Working software over comprehensive documentation  
Customer collaboration over contract negotiation  
Responding to change over following a plan

That is, while there is value in the items on  
the right, we value the items on the left more.”

(Agilemanifesto.org, 2015)

For this project, it is important to consider that the customer is simulated, and that there is no team to distribute work between. However, the underlying principles of the method will be valuable to the project.

Prototyping is another viable option, in that it would be possible to develop a SCADA system as desired rather quickly, before undergoing testing and improving until all aspects of the plan are eventually met. Whilst this would usually require customer acceptance, this project would simply require all of the project elements to be completed as written within this proposal. This method is most effective when all project requirements are not known from the outset, and as such, is not as useful in this scenario as Scrum.

DMAIC is a methodology used internationally within the Six Sigma project planning course. As part of the course, the methodology is taught and used to ensure that employees are able to effectively and efficiently use time and resources for better management of projects and teams. DMAIC is an acronym for Define, Measure, Analyse, Implement and Control.

In addition to SCRUM’s principles, the addition of DMAIC within each sprint should assist in controlling the workload and time efficiency.

**Potential Ethical Issues**

SCADA systems have potential access to huge amounts of intellectual property in the form of real world, raw process data. This is often valuable for companies both in terms of purchase value and competitor risk. To add to this issue, many SCADA systems are part of a more distributed control system, meaning data from each system is passed via a network to a central controller. The method in which data is sent is a potential attack vector.

Should illegal access to process data be obtained, it is possible for the company to be at risk of losing money, customers or both.

A larger ethical issue, however, is whether or not remote commands (via faulty/modified data or otherwise) can affect the real world in such a way so as to cause severe cost or physical injury to any nearby workers.

Both of these issues are easily controlled for this project as real-world connections will primarily be simulated, however, should physical equipment be used, it will be done so within the university’s lab with qualified operators available and on hand at all times. Similarly, all code produced for this project will be designed with safety in mind, ensuring that code that can affect real world actions operate within a tightly controlled set of boundaries, even when abnormal data is read/calculated.

**Project Deliverables**

The project aims to deliver a desktop C# application that is able to communicate with a range of hardware and software based data sources in order to actuate an output. The application will act as a control (SCADA) system for a basic process loop.

Communication will be conducted either via file writing or serial ports. It is to be assumed that the hardware itself will understand basic commands (such as start, stop, etc…). The end product will merely call upon those basic commands when necessary as part of the developed loop (see Command Structures, above).

The plan is to develop an application that is able to effectively interact with the real world, taking sensory-data and using user-defined configurations to create/cause an output from the system. This will be completed by creating an application that can do the following:

* Simultaneously connect with a variety of data sources;
* Allow a user to build feedback control loops for or around each data source;
* Simultaneously connect with a variety of actuators (software or hardware based);
* Allow a user to build basic command structures for each actuator\*;
* Send commands to actuators based upon those rules and command structures;
* Allow data sources and actuators to be combined as objects;
* Allow a user to set low, high and missing value alarms;
* Allow a user to quickly see and respond to alarms triggered by the process;
* Use a database management system to host a database that can store:
  + Incoming data - either all values *or* only delta values
  + Outgoing data - commands sent to actuators
  + The system configuration
* Allow users to report data from the system
  + Plot/Trend values from any data source
  + Print process reports for a given loop, time period or average values per loop
* Allow a user to interact via an intuitive, ‘finger-friendly’ and scalable user interface;
* Appropriately handle:
  + Disconnected or missing data sources or actuators;
  + Dropped or failed network connections;
  + Faulty, corrupt or unexpected data values;
  + Sudden application terminations;
  + Unavailable connection to database.

**\***Actuators will be assumed to have set command structures. For example, it may have routines for increasing speed, turning on or alarming. The product of this project will call upon and enact those routines safely and where necessary.

**Project Plan**

In addition to this module, it is important to remember that other submission dates will occur through the year from other classes. The deadlines for each of those modules are listed below, with the exception of the *Embedded Systems* second ICA, which is yet to be announced.

|  |  |  |
| --- | --- | --- |
| **Module** | **Deadline Type** | **Date** |
| Computing Project | Proposal Draft Submission | 12/10/2015 |
| Computing Project | Final Proposal Submission | 16/10/2015 |
| Computing Project | Review Session Presentation | 23/11/2015 |
| Embedded Systems | ICA 1 Presentations | 18/01/2016 |
| Advanced Database Systems | ICA 1 Submission | 30/01/2016 |
| Software Architecture | ICA 1 Submission | 08/02/2016 |
| Computing Project | Final Product Submission | 11/03/2016 |
| Software Architecture | ICA 2 Submission | 14/03/2016 |
| Client Focussed Business Solutions | Final Product Submission | 15/03/2016 |
| Computing Project | Final Report Submission | 18/03/2016 |
| ExpoTees | Preparation Start | 11/04/2016 |
| Advanced Database Systems | ICA 2 Submission | 27/04/2016 |
| Client Focussed Business Solutions | Critical Reflection Submission | 28/04/2016 |
| Embedded Systems | ICA 2 Submission | TBA |

For each of the weeks available within the project’s time frame, proposed activities for each week have been given below. Where the date is emboldened, there is one or more events from the table above occurring in that given week. The activities proposed are generalised to accommodate any issues that may arise during development, and are subject to change. Each activity, however, has been written in such a way so as to allow it to be completed as part of a sprint.

|  |  |
| --- | --- |
| **Week** | **Proposed Activities** |
| **12/10/2015** | Investigate SCADA systems and write an appropriate proposal. |
| 19/10/2015 | Begin designing class structures. |
| 26/10/2015 |  |
| 02/11/2015 | Design the database and generate documentation for that design (inc. UML) |
| 09/11/2015 |  |
| 16/11/2015 | Finalise class structures and create objects digitally to that effect. |
| **23/11/2015** | Create new C# project and implement each object correctly (including extensions, inheritance, et cetera).  Start writing the report for the first ICA. |
| 30/11/2015 | Build a test application to allow testing of all objects to ensure they act as required and safely handle a range of inputs. |
| 07/12/2015 |  |
| 14/12/2015 | Create a poster for the review session.  Implement the database design in the database management system. |
| 21/12/2015 |  |
| 28/12/2015 | Present the poster at the review session.  Test the database, including all relationships and dependencies. |
| 04/01/2016 | Design a basic user interface to allow creation of device input/output.  Begin writing the second half of the report. |
| 11/01/2016 | Implement the user interface and design a UI to allow creation of loops. |
| **18/01/2016** | Implement the user interface and document the above process. |
| **25/01/2016** | Using SSRS/SSAS, create analytical reports to allow investigation into the controlled process. |
| 01/02/2016 | Implement the connection to simulated hardware so as to allow full process simulation. |
| **08/02/2016** | Using university equipment, control a process and fully document. |
| 15/02/2016 |  |
| 22/02/2016 | Begin full and formal testing, documenting the entire process |
| 29/02/2016 |  |
| **07/03/2016** | Complete final testing, package suitably and submit for marking. |
| **14/03/2016** | Complete the final report and submit for marking. |
| 21/03/2016 |  |
| 28/03/2016 |  |
| 04/04/2016 |  |
| **11/04/2016** |  |
| 18/04/2016 |  |
| **25/04/2016** |  |
| 02/05/2016 |  |
| 09/05/2016 |  |
| 16/05/2016 |  |
| 23/05/2016 | End of university. |

**References**

Agilemanifesto.org, (2015) *Manifesto for Agile Software Development*. Available at: http://agilemanifesto.org/ (Accessed: 11 October 2015).

Altmann, W. and Macdonald, D. (2005) *Practical process control for engineers and technicians*. Oxford: Newnes.

Bellman, R. (1964) 'Control Theory', *Sci Am*, 211(3), pp. 186-200. doi: 10.1038/scientificamerican0964-186.

Keithley, (2001) *Data Acquisition and Control Handbook*. Cleveland, Ohio, USA: Keithley.

Lindstrom, L. and Jeffries, R. (2004) 'Extreme Programming and Agile Software Development Methodologies', *Information Systems Management*, 21(3), pp. 41-52. doi: 10.1201/1078/44432.21.3.20040601/82476.7.

Maxwell, J. (1867) 'On Governors', *Proceedings of the Royal Society of London*, 16(0), pp. 270-283. doi: 10.1098/rspl.1867.0055.

Norman, J. (2015) *The First Truly Automatic Self-Regulatory Device (Circa 250 BCE) : HistoryofInformation.com*, *Historyofinformation.com*. Available at: http://www.historyofinformation.com/expanded.php?id=2306 (Accessed: 11 October 2015).

Pree, W. (1995) *Design patterns for object-oriented software development*. Wokingham, England: Addison-Wesley Pub. Co.

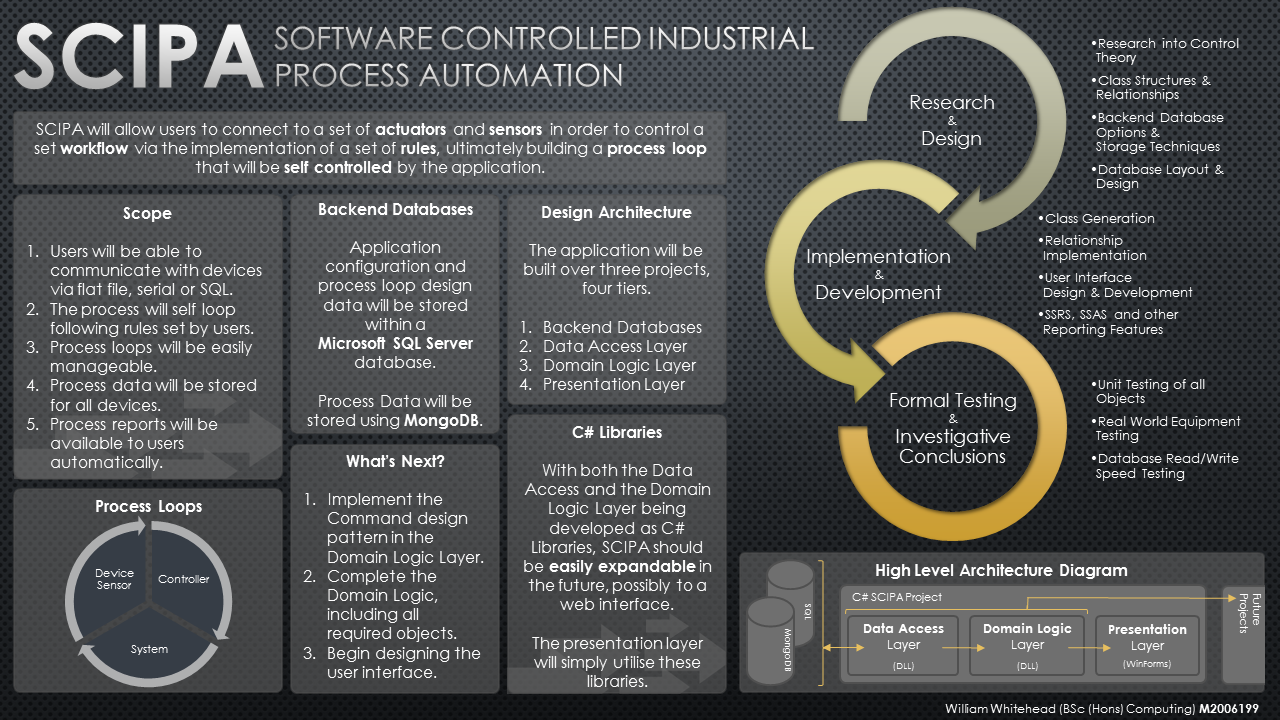
*Process Control Fundamentals*. 1st ed (2006). PA Controls, p. 10. Available at: http://www.eng.unideb.hu/userdir/deak.krisztian/BOOK.pdf (Accessed: 11 October 2015).

Seborg, D., Edgar, T. and Mellichamp, D. (1989) *Process dynamics and control*. New York [etc.]: Wiley.

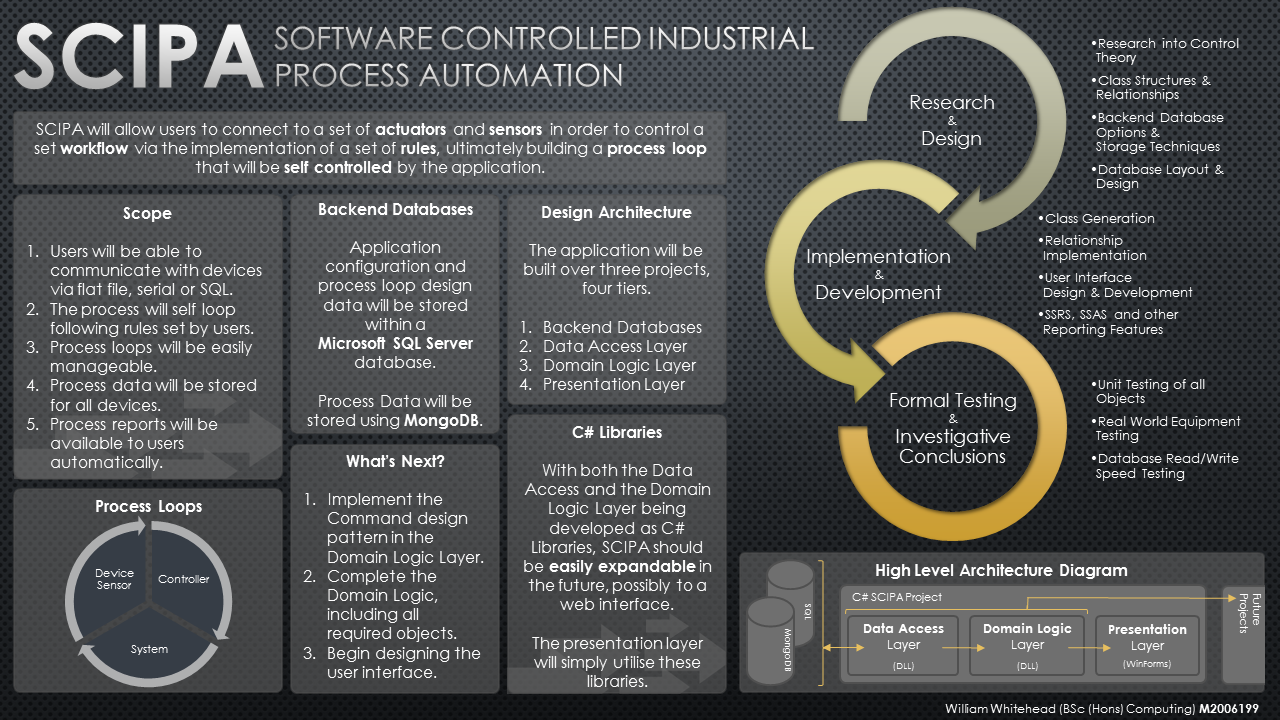
Westphal, L. (1995) *Sourcebook of control systems engineering*. London: Chapman & Hall.

## Poster Presentation

Side A



Side B



## Arduino Sketch Source Code

### Trending Application

### Input Acceptance Application

### Basic IO Application

### Other