University of York

Second Year Project

Embedded Systems Project Written Report

Author:
Douglas Parsons

Supervisor:
Dr. M.J. Freeman

April 11, 2015

Contents

Chapter 1

Introduction

1.1 Summary of project work

The Second Year Embedded Systems Project was a ten week project, undertaken for the duration of Spring Term 2014-2015. The project was a primarily a group task, completed in groups consisting of four members. However, there was also the capability for individual extensions to the main body of the project, allowing each member to showcase individual competency and creativity.

The group project involved programming an ARM LPC1768 'MBED' microcontroller, situated on top of a board of peripheral accessories, in order to generate and play music on a user selectable channel corresponding to data being sent down a Controlled Area Network bus (CAN bus). The individual component was not specified, and was instead left up to each individual to decide on an extension project, research, and implement it.

The groups solution was very complete, not only matching the full specification , but also extending the implementation to a high degree, producing a user friendly, good sounding end product. My individual extension provided a more user friendly interface than the rudimentary keypad attached to the MBED board through the implementation of a shell type interface. This shell style interface allows the user to input commands via a computer keyboard in order to change modes, adjust settings, or display useful information.

1.2 Experiences as a team

The capability to work successfully as a group was imperative to the triumph of the project. Throughout the ten week period we had regular group meetings, allowing us to keep track of each person's progress, and we had strong levels of feedback on contributions to the solution through regular code reviewing sessions. This enabled us to remain productive and on track throughout the project. Furthermore, the high levels of communication enabled each individual to work on their preferred areas, or areas of interest without ever straying too far away from the desired end goal. Each member of the group had a strong drive to provide a high quality finished product, and eagerness to develop their personal skills. This resulted in a large amount of time being spent in the labs outside of the scheduled practicals, enabling us as a group to progress much further than we otherwise might have done. Overall I feel we worked very well together, and

the exercise served as a valuable insight into a fluent working environment as a team.

1.3 Expectations and actual outcomes

At the beginning of the ten week period it was very difficult to predict what would be possible to achieve due to a combination of inexperience with programming embedded systems, and an uncertainty regarding the limitations of the LPC1768 board used. However, it was expected that during the course of the ten weeks my group would be capable of meeting the specifications of the project, as well as implementing our individual solutions. Certain aspects of the project took longer than initial expectations would have suggested, for example, it took a significant portion of time to implement the various different user control methods present in our final solution. Overall, our time management skills worked out very well though, and our completed solution lives up to, if not exceeds, the expectations of the group.

1.4 What is in the report

This report sets out to provide a description of our team solution to the set assignment, as well as highlighting areas of individual contribution throughout. The report will begin with a technical description of the set problem, including a discussion of requirements and technical challenges that may be faced in the meeting of these requirements. Following a description of the problem, the group solution will be discussed. The discussion will highlight how each section of the requirements has been met by the implemented solution, as well as detailing how the problem has been broken down for each individual member, presenting each members implementation and technical innovation. After a discussion of the group's solution, the appropriate testing strategies and methods that were used throughout the project will be examined in detail, providing feedback on how they have proved useful, and incorporating a discussion on a professional/social/ethical/environmental aspect of the solution. The report will then focus critically on my individual implementation, detailing the technical innovation, implementation, and testing undergone for my contribution. Finally the report will conclude with a reflective summary of work undertaken, considering which aspects of the project went smoothly, which areas did not go as smoothly, how this might be improved in future, and what lessons can be taken away from the completion of the project.

Chapter 2

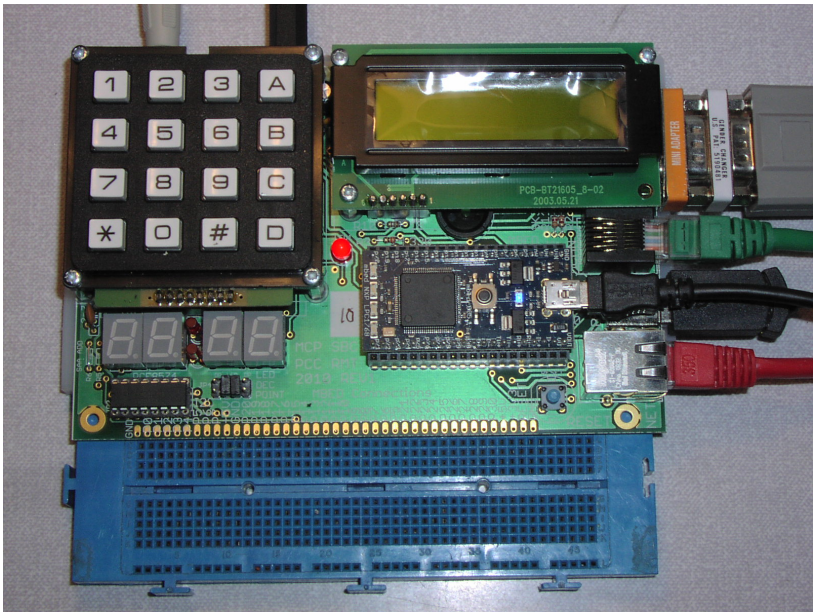
Technical Description of Problem

2.1 Description of Problem and Requirements

The Embedded Systems Project consisted of two main sections: a group solution to a predetermined problem, and an individual extension to the group's work, left up to each person to determine. Both sections of the project involved embedded systems software development in the C programming language.

The target platform of the development is an ARM-based microcontroller, situated on a board of peripheral accessories. The microcontroller consists of an interface board coupled with the ARM cortex-M3 based LPC1768, providing easy interaction via USB cable to transfer binaries, and simplifying the process of serial communication [how-mbed-works].

The MBED board is sat on a board of peripheral accessories, as pictured below.



The board contains multiple accessories that the MBED board can interface with in various ways to fulfil the requirements, and allowing many options for

individual extensions. The accessories on the board include: a 16 key keyboard, a 16 by 2 line LCD, four seven segment displays, a micro SD slot, a USB host connection, CAN bus connecter, audio in and out jacks, a TCP/IP connection, as well as off board connections to users hardware.

The group task consisted of multiple different challenges, with the concept that the MBED board would continually receive CAN packets, filter out those that do not match a user selectable id, decode the CAN packets, and play notes corresponding to the frequency, volume, and duration that were specified from the CAN transmissions. In addition the user must be able to view useful information, select the channel id, and adjust the volume.

In addition there was an individual component. This did not have to match any specification, and was instead left up to each person to decide on their own extension to the group project. For my individual extension, I implemented a Linux shell style user interface. This allows user to interact with the device by typing commands into a terminal on their computer, effectively adding an additional element of user control that is more intuitive and responsive than using the 16 digit keypad from the MBED board.

For our group solution, the specification was broken down into three separate areas, allowing us to focus our expertise and interests on what suited us best. The breakdown of the three sections was as follows: The receiving and decoding of CAN packets (CAN bus), generation of audio samples corresponding to frequency, duration and volume (Audio), and the ability for the user to interact with the device in order to adjust settings (User Interface).

2.2 Discussion of Technical Aspects and Challenges

As previously mentioned, my group deconstructed the specification into three main areas of technical challenge. The first of these challenging areas regarded the receiving of data from the CAN bus, and the decoding of the packets received to extract their content. As the CAN packets are not arriving at regularly timed intervals, the solution to this problem is required to be interrupt based. However, due to the nature of the CAN packets, a very large number can arrive in a comparatively short space of time. This therefore requires a highly optimised solution, in order to prevent packets either being missed, or race conditions within the interrupt handler from occurring.

The second technical challenge of the project is the generation of audio tones corresponding to frequency, duration and volume levels as specified by the CAN packets. The CAN packets do not contain information detailing what each note should sound like. As a result, all musical notes need to be generated from first principals, for example using a look up table detailing values for a note at each moment in time. In combination, to make the notes sound more realistic, each note must go through at least three stages. The first of these stages defines the

initial playing of a note, plucking a string or pressing a key. This is known as 'attack'. The second stage corresponds to the continual noise made from the note until it is stopped and is known as 'sustain'. The final stage details the note as it dies down and stops and is known as 'release'. This presents additional problems as each note requires multiple states, and needs to be tracked in its progress through each. Furthermore, it may be an additional desire to play more than one note at any one moment in time. This requires synthesis of multiple different notes in order to produce an appropriate output. The mathematics of this can be expensive to compute, and furthermore, an increase in the number of notes can drastically increase the processing time. A delay in the output of the notes can lead to a strange sounding finished product, so the efficiency of any synthesis calculations are paramount to the end product.

The third technical challenge that can be presented is the user's control over the device. The specification determines that the user must be able to control the volume of the output, alter which channel of notes is being played, and view useful information on the MBED board. This requires a level of interaction, most likely via the 16 digit keypad on the device, however this could be via alternate input methods, such as a computer keyboard. The user's interaction with either device once again may not be continual, but may either be polled, at the cost of system resources, or interrupt based. Further challenges are presented as any possible user input must be stored on the device, parsed on command, and then the desired command must be executed without any significant effect on the audio output. Furthermore, the challenge is made increasingly difficult by the required ability to interface with the entire project: the audio synthesis code and the CAN bus code must be fully understood in order to be manipulated through the user's input.

Chapter 3

Description and Discussion of Team-based Solution

3.1 Description of the Team Solution

The end product of our team solution was a successful working product. Through the incorporation of a Linux style shell into the group solution, we managed to produce a highly configurable single solution that managed to satisfy each section of specification. In addition the group solution contained two custom built speakers, capable of being powered by the 5V voltage rails from the peripherals board. The solution had a strong user interface, allowing the user to type in commands through the computer keyboard to adjust settings and select modes, set an id to filter channels out, and display information, or custom text on the mbed board. In addition it was also possible to adjust volume, channel and display information on the mbed board using the 16 digit keypad. This high level of user interfacing allowed settings to be flexibly turned on and off, and avoided the need to be continually re-installing binaries to the device. By typing in the command "listen", the mbed would then initialise the CAN bus and begin receiving packets. These could be printed out to the screen by issuing a further command, or they could be used to generate audio tones. It is also possible to filter out data according to a preset id by issuing a command, or using the keypad. For each data element received, audio tones can be generated. The solution supports a full synthesis, allowing multiple notes to be played on each channel, and multiple channels to play notes simultaneously. In addition the audio output also implemented an 'attack-sustain-release' model, creating a more realistic sounding solution. The audio code used direct memory access (DMA) to provide a reduced processing cost in outputting the audio samples. The user could also choose to display useful information such as the current song name, the volume, or the channel names on the computer screen or on the MBED board in a variety of ways. For example they could be statically printed on the MBED board, or scrolled across the LCD display, or even printed out to the terminal on the computer.

3.2 How the problem was broken down for individual members

The breakdown of our solution into components individual members could work on was a great strength of our team. Each person worked on as aspect of the project that they found interesting, and throughout the project we had regular feedback on all our work in order to make sure we were meeting our specifications and that nobody was out of their depth. Below I have divided up the work that each individual completed during the duration of the project, and why they were working on that aspect of the project.

Mingzhao Zhao

Each individual was allowed to choose their own area of the project to work on. Mingzhao was not especially confident with the platform, and therefore he decided to target user interaction with the device via the keypad. This provided a possible method of user input, as well as providing a means to optionally filter data, or control the volume output of the device. Following suggestions of other group members he decided to convert the previous keypad polling method incorporated in the mini-projects into an interrupt based system, removing the need for unnecessary CPU cycles to be used polling the device continually.

Shivam Mistry

Shivam worked very strongly as a group member. To begin with he took on the challenge of constructing code to read the data from the CAN bus continually, as well as implementing the base for filtering out data according to a preset ID . Following the construction of the CAN bus code, he worked on optimising the CAN bus code through the implementation of a queue, as well as decoding the CAN packets to extract their data, and the parsing of text packets to extract only the useful information. Furthermore, on realisation of additional device memory, Shivam wrote a memory allocator allowing the additional memory to be used constructively, further optimising the system.

Myself (Douglas Parsons)

Following the mini projects, and from reviewing the specification, it was clear that a strong level of user interaction with the device would provide valuable debugging tools, could potentially allow settings to be changed without re-installing binaries on the device, and could be used for a wide range of useful functions, such as changing the volume or displaying song information. From the mini projects it became clear that the 16 digit keypad attached to the mbed would not provide an intuitive user experience. Rather than relying on the keypad as the sole method of user interaction, I instead focused on allowing a further degree of interaction via the computer keypad. This enabled the user to preset an ID to filter out data, adjust the volume of the output, display any data received on the CAN bus, as well as adjust settings within the project without requiring a full

reinstate. During the implementation of the shell it became clear that this could be a valuable tool for displaying useful information such as the song name, and therefore I additionally worked with the LCD display to enable a much greater degree of control over what was displayed, enabling scrolling text, and writing to each line of the display individually.

Liam Fraser

Liam has a strong background in music technology, taking the subject at A-level, and working on personal projects involving music. Furthermore his interest from the beginning of the project in audio processing meant that it was an obvious choice for Liam to look into the generation and playback of music. The specification was exceeded with his contributions, not only did his solution generate an audio tone corresponding to the frequency, duration, and volume levels specified in the data stream, but implemented an attack sustain release curve to improve the realism of the output sound, as well as additive synthesis enabling multiple notes to be played simultaneously, and his solution also used DMA and a fixed point library to improve the efficiency of the system.

3.3 Technical innovation and implementation of each member

Mingzhao Zhao

Mingzhao's work was primarily focused on implementing interaction via the 16 digit keypad on the mbed board. The keypad was set up in the mini-projects to allow user interaction through continually polling the keypad. However, due to the limited speed of the processor it was decided that a better solution would be to use an interrupt based system. However, this caused additional problems with bounce on the switches: multiple interrupts could occur in a very short space of time due to bounces on the switches. Many of these interrupts generated were redundant, or gave nonsensical input values, and furthermore their rapidity caused race conditions to occur within the interrupt handler. Due to the row/column method in which the key pad is scanned, the number of bounces was reduced significantly by filtering out any inputs that did not match one of the keys pressed. The switch bounces were then reduced further by only allowing a single key press every half a second. This method of debounce was implemented using a system timer (SysTick) to count the time between successive key presses. Any key presses that occurred too closely together were disregarded.

Shivam Mistry

Shivam initially targeted the CAN bus, and was able to rapidly achieve a fully functioning prototype. The original solution processed each packet as it was received. However, as the complexity of songs increased it became clear that this simplistic method was not going to be sufficient when many CAN packets

require processing in a short space of time. In order to improve the functionality, a queue was implemented. Packets were added to the queue on being received, and processed whenever there was sufficient processor resources available. This removed the requirement of each packet being processed as it was received, and therefore provided a clean, efficient solution. Shivam's contribution also included the reverse engineering and decoding of any CAN packets received through testing their check-sum, and parsing any received text. As the check-sum method was not specified in any documentation, the first challenge was determining what the check-sum corresponded to. Values that did not match the length specified by the check-sum were not processed, and therefore could not have any impact on the system. Furthermore, the text packets sent prior to each song being played required parsing in order to extract any useful information. Therefore, a text parser, extracting the useful information from these packets to the device was implemented using a state based solution.

Furthermore, following an instance where the physical memory (RAM) on the device was fully used up, Shivam discovered that the device contains an additional 32Kb of memory, typically reserved for communicating to a USB or Ethernet device. However, due to the reserved status of this memory, a memory allocator was required for it to be utilised. Therefore, the group solution contains a memory allocator allowing access to a further 32Kb of memory. This allows a much greater amount of information to be stored on the device. In addition, the look-up table used by the memory allocator was stored in the additional memory, making it a self containing allocator.

Myself (Douglas Parsons)

My personal contribution to the project was primarily targeted at the user interactions with the device. While it was initially set out that Mingzhao was going to work on implementing an interrupt based keypad, I decided that the communication via serial would potentially allow a much greater level of debugging, interaction, and manipulation of settings for the device. From the beginning of the ten week period, I focused on setting up a shell style interface for the device. This would allow the user to communicate with the mbed board by typing in commands on a computer keyboard, and execute the command by pressing the enter key. The input from the keyboard was implemented using interrupts in order to avoid unnecessary computation time from continual polling. Furthermore, the processing of text input required a text-parser to be used. Due to an unknown number of words, and an unknown input length, the parser had to be very carefully adjusted to avoid memory overflow situations. Further difficulty was added to the implementation, as alterations had to be made in many different aspects of the group solution. A detailed understanding of each individuals contributions was required to manipulate their functionality, and continual additions and updates were needed throughout the project as changes were made. In addition to implementing the additional interfacing method of a shell style interface, I focused on customisation of the mini-project LCD screen code in order to allow a greater level of control. I implemented features such as the abil-

ity to write to each individual line of the LCD display, and the ability to scroll text across the screen. The latter of which was particularly difficult, not only requiring pointer manipulation for efficient text processing, but accessing only the required section of the LCD in order to avoid displaying unwanted characters provided a significant technical challenge. Following the implementation of the shell, and the LCD screen code, I worked alongside Mingzhao on combining his code with the rest of the group solution.

Liam Fraser

Liam Fraser, following an interest in music technology was eager to work on the audio code for the project. He implemented a simplistic system, allowing musical notes to be generated by scanning, and interpolating over a look up table. He then looked into generating synthesised music, and implemented additional synthesis into the project, allowing multiple notes to be playing simultaneously. However, on testing his synthesis code on the mbed board, it rapidly became apparent that the use of floating point numbers was not sufficiently quick to produce clean sounding audio. Therefore, a fixed point library was implemented, this allowed much quicker calculations to occur compared to those of floating point numbers. Following this optimisation of the synthesis code, the synth code was further optimised through the use of Direct Memory Access (DMA). This allowed the audio output to work independent of the central processing unit (CPU). This therefore speeds up the processing of any audio, as the CPU does not have to be involved for any sound to be output, only for the generation of audio samples [dma-book]. In addition, to create a more realistic sounding output from the device, an attack sustain release state model was incorporated for each note. This gives an initial rise in volume as the note is turned on, and a fading out of the note as it is turned off, producing a volume curve more typical of an actual instrument [asr-book].

Chapter 4

Evaluation and Testing of Team-based Solution

4.1 Description of the Team's Testing Strategy

Throughout the duration of the project, each section of the project underwent significant reviews, and testing in order to ensure that it was of the highest quality and that it was not laden with any issues causing undesirable behaviour, or significant faults that may have caused the device to crash. Furthermore, once incorporated into the group's main solution, the implemented code underwent a thorough review in order to ensure that the quality of the implemented product was the highest possible, and free from anything that may cause issues in the future. In effect the product underwent many different stages of testing throughout its life-cycle.

Individual Testing

Prior to the inclusion within a greater completed solution, each section of the project underwent individual tests. In order to test each section of the project, a debugging library was created. The debugging library was only included in the binaries installed on the device if the make command included included it. This means, for the purposes of testing it was possible to use `make debug` and have debug messages containing text, or values printed out over serial. However, once sufficient testing was done, no alterations had to be made to the source code to remove these statements, and instead the project could be installed by calling `make`. This enabled statements to be added to the project, that would print out messages over serial.

Code Review

Before each contribution was incorporated into the group's solution, it was first checked over by at least two other group members to ensure the quality of the solution was high. These checks often resulted in re-factoring code to achieve a more streamlined end product, examples of which could include the simplification of logic for writing text to the LCD display through pointer arithmetic, or simplifying code within an interrupt handler in order to avoid race conditions.

Integration Testing

Following the combination of individual components into the group solution, the combined product was then tested thoroughly to ensure no erroneous behaviour could occur. This was primarily done by leaving the boards running for long periods of time within the labs, and listening to make sure nothing unexpected was occurring. In addition, each feature was continually explored in order to make sure everything fully functioned.

4.2 Results of Testing Strategy and How Well This Met the Requirements

At many points throughout the project minor issues were discovered, and corrected. Through the continual testing of the full solution by leaving it running for a long period of time, a large number of test cases and situations were explored. Furthermore, a careful eye was kept at all times to ensure that the product's functionality was as desired, and further testing was done to ensure the requirements of the specification were fully met. Through our thorough and rigorous testing, the end product was remarkably stable. We encountered no situations that would cause the product to produce undesirable behaviour, and furthermore the full requirements were met.

4.3 A Social/Ethical Aspect of the Group Solution

The group solution created has the potential to solve any issues that may arise when considering the usability of the user interface with respect to a disabled user or users. Interaction with the device via the 16 digit keypad that is on the host board may not provide a very effective method of interaction with the device for many users. However, the shell style interface allows user to interact with the device via a computer. Interaction with a computer, and hence the mbed would typically be achieved through use of a keyboard, however, there are many alternative methods of input that would allow interaction with the mbed board in a method better suited for a disabled user. These input methods could include tools such as an improved computer keyboard, speech recognition, or the tailoring of custom tools to allow a greater level of interaction. Such tools are highlighted in many works, examples of which could include *Computer Access for People with Disabilities: A Human Factors Approach*, or *Universal Access in Human-Computer Interaction* [disabled-book, disabled-book2]. The flexibility of input method provides a wide range of possible tools enabling communication with the device. This therefore avoids significantly the problem of interaction for a handicapped or disabled user.

Chapter 5

Description, Discussion, Testing and Evaluation of Individual Component

5.1 Description of Individual Component Solution

My individual contribution to the project consisted primarily of a shell style interface for the host board, allowing user interaction via a computer keyboard and monitor, but also consisted of the modification and extension of previously existing LCD screen code, to allow a much greater degree of control over the attched LCD display.

The shell style interface was set up using an interrupt based system from the computer. This enables users to enter commands through any input method for the computer and process them on the device. This method also allows the user to display useful information through the computer monitor.

There were a wide range of possible input commands available through this interface, with many possible options and settings being adjusted on the fly. By typing 'help' and pressing enter, the user could view a list of possible commands, as seen below:

"List of available commands:

playnote note volume : plays the selected midi note
noteoff note : turns off any playing notes
volume ¡vol¿ : sets the output volume to 'vol'
showvol : displays the current volume
write "text" : writes text to the LCD screen
writeline "text" ¡linenumber¿ : writes text to one line of the LCD screen
listen : listens to music on the CAN bus
stoplisten : stops listening to music on the CAN bus
setid ¡channel¿ : Filters out channels that do not match channel number 'channel'. To play all, use "setid all
showid : Shows the id of the current channel
showtrack : Shows the current track name on the LCD
showchan : Scrolls the channels on the LCD
showpacket : Prints CAN packets until a key is pressed

scroll ¨text¡line¿ : displays scrolling text on line ¡line¿ of the LCD screen
stopscroll ¡line¿ : stops scrolling text on the screen. To stop both lines, enter ¡line¿ as 2 or all.
scrollenable ¡line¿ : enables scrolling text on the screen.
shownotes : displays all notes currently being played
cowsay ¨text¨: displays an ASCII cow, saying text
clear ¡line¿ : clears any text that is on the lcd line ¡line¿ Note - you may also have to call stopscroll to fully clear

This therefore provides the user with a high amount of control over the devices output though both audio and visual elements, as well as a high level of control over the devices functionality. Furthermore, the interaction with the on board LCD display can be clearly seen through the above commands, hence explicating the necessity for the alterations made to LCD display code to permit a greater degree of control, and enable features such as scrolling text, and writing to only part of the LCD screen.

5.2 Discussion of Technical Innovation and Implementation

The implementation of the shell style interface and the improvement of the LCD display code consists of many different technical challenges, and required many different technical challenges throughout. First and foremost there was the challenge of setting up a system to enable user input to be recognised. This is achieved using an interrupt based system, such that each new character entered into a screen set up for the board will trigger an interrupt. When the interrput is triggered, the character that was input is transferred to the host board and stored into memory for processing at a later stage. This incorporated several different technical challenges in the implementation, such as figuring out how to set up the interrput, and how to allocate memory for storing any characters input. The simplest solution for allocating memory was used: a set input limit is defined, and any characters that are input past this count are not stored in memory. Other technical challenges involved handling special input cases such as backspace to delete a character from memory, but only if there was any input to delete.

When the enter key is pressed (
r
n received on the interrput), processing of the user's input command begins. First the text is split up into tokens for processing. This requires the use of a text parser. The text parser implemented uses a state based system, with different states corresponding to whether it is interpreting a single word, a phrase (group of words encased in quotation marks), or whitespace. This breaks the user's command into it's constituent components, which can then be individually evaluated. However, it was found that memory allocation problems could occur due to the large range of possible tokens in the input. For example, the user

could input 'a a a a a a', and this would be interpreted as 6 individual tokens, requiring the allocation of six blocks of memory. A solution to this problem was found by simply setting a limit on the number of tokens that can be generated by a single command.

Following the disection of the user's input into components, a string comparison tool (strcmp) was used to determine what the user's input was, enabling the corresponding desired behaviour to be executed. In order to prevent race conditions from occuring within in the interrupt handler, all text processing following the enter key being pressed occurs outside of the interrupt handler using a flag based system.

5.3 Testing Strategy, and the Results

Chapter 6

Summary and Conclusions

6.1 Reflective Summary of Team Work

6.2 Reflective Summary of Individual Work

6.3 What went well

6.4 What went poorly

6.5 What could have been improved

6.6 Lessons Learned

Appendix A

Specified Documents

A.1 Meeting minutes

A.2 Evidence of Preparation