University of York

Second Year Project

Embedded Systems Project Written Report

Author:
Douglas Parsons

Supervisor:
Dr. M.J. Freeman

April 27, 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 potential for individual extensions to the main body of the project, allowing each member to showcase 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 audio on a user selectable channel corresponding to data being sent down a Controller Area Network bus (CAN bus) [1]. The individual component was not specified, and it was instead left up to each individual to decide on an extension project, research, and implement it.

My group's solution was very complete, not only matching the full specification, but also extending the implementation to a high degree, producing a user-friendly end product with a high quality audio output. My individual extension provided the device with a more user-friendly interface than the rudimentary keypad attached to the MBED board through the implementation of a shell type interface. This 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 effectively as a group was imperative to the success of the project. Throughout the ten week period we worked effectively as a team through a number of different means. First of all we held regular group meetings. This allowed us to keep track of each group member's progress and enabled each member to feedback any potential problems they had encountered, as well as ensuring we were meeting the specifications of the project. Secondly, each contribution to the group solution was passed through a code review prior to inclusion. This ensured that each member was only adding high quality contributions, and enabled group members to learn from each other. In addition, each member of the group agreed to use the version control system, git, coupled with github. This enabled a much easier collaboration of code throughout the project, and made communication between individuals much simpler. Through the combination of these methods, my group was able to remain highly productive throughout the course of the project. Furthermore, the high levels of communication enabled each individual to work primarily on their areas of interest, removing the need to continually worry about to the overall progress of the project. Overall I feel we worked very well together as a team, and the project served as a valuable insight into a fluent working environment as a team.

1.3 Expectations and actual outcomes

Due to my personal inexperience of programming embedded systems, and unfamiliarity with the LPC1768 board used, it was very difficult to predict the outcome of the project. However, due to the competency and eagerness of my group's members, it was expected from a very early stage that we would rapidly be able to match the specification and extend the project to a high degree. The majority of the project went very fluently, however, certain aspects of the project took much longer than initially expected. Notably the CAN bus in the labs was not working at the beginning of the ten week period, and consequently producing working code for reading the CAN packets took longer than expected. Furthermore, handling the wide range of possible cases for user input, and interaction between the keyboard and shell input methods took much longer than expected. The end product that we implemented was capable of matching each point of the specification, and surpassed the specification within the group project, as well as individual extension project. Therefore I feel the group's time management skills worked very well, and the end product lived up to expectations.

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 member's 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 social 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.

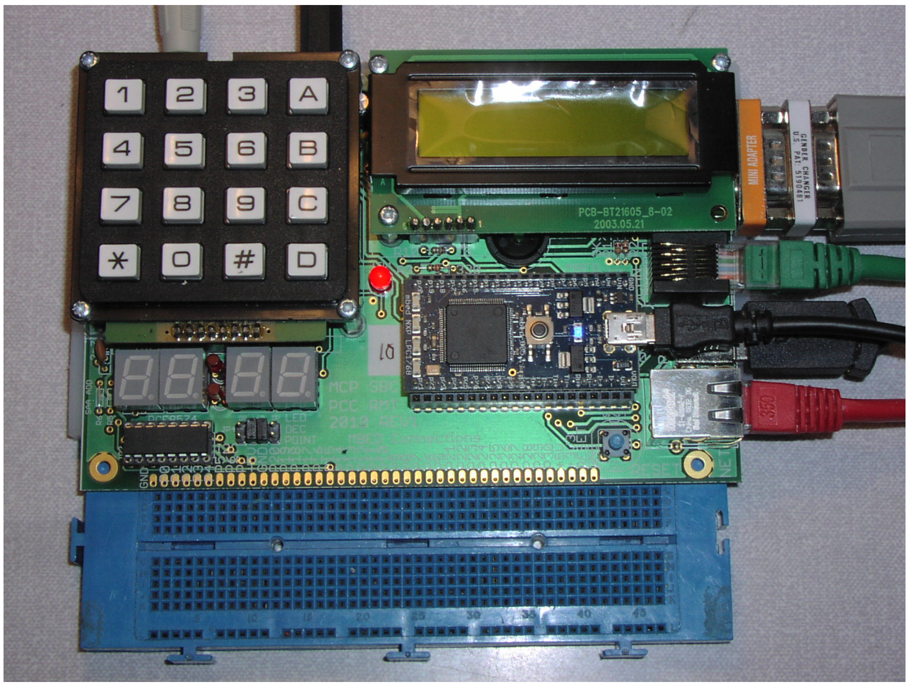
Chapter 2

Technical Description of Problem

2.1 Description of Problem and Requirements

The Embedded Systems Project consisted of two main sections: a group solution that was required to meet a set specification, and an individual extension to the group's solution that was 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 project 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 [2]. The MBED board is also sat on a board of peripheral accessories, enabling interaction with a variety of different components. The board is pictured below [3].



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 connector, audio in and out jacks, a TCP/IP connection, as well as off board connections to user definable hardware. Interaction with these devices is achieved directly through the manipulation of the MBED board's pins, therefore enabling a high level of communication with relative ease [4]. The range of devices provides a large variety of possible methods for meeting the specification aims, and enables a large scope for individual creativity with extensions to the specification.

The group task consisted of multiple different challenges, with the concept that the MBED board would continually receive data from a Controller Area Network bus (CAN bus). A CAN bus is a type of network, commonly used for rapid communication between embedded systems in situations such as aeroplanes and cars. It could be simply stated as "a network technology that provides fast communication among micro-controllers up to real-time requirements" [5]. Following the successful retrieval of CAN packets from the network bus, the group solution must decode any packets received, filter out those that do not correspond to the selected channel, and play notes corresponding to the frequency, volume, and duration specified in the received packet. In addition the user must be able to view useful information, select the channel ID, and adjust the volume of the device. The full requirements for the group solution are as follows [6]:

S.1. A solution using MBED for live monitoring of CAN bus data packets. Ideally this should include the ability to show all data and also to show data filtered according to a configurable instrument channel ID.

S.2 An MBED solution which has the following specifications :-

- Receives CAN bus data continuously
- Has the ability to preset a local 'ID'
- Can optionally filter the data according to it matching the preset ID
- For each data element, will generate an audio tone corresponding to frequencies, durations, and volume levels if specified in the data stream
- Has a visual display on the MBED host board indicating useful information
- Has the ability to receive user input via keypad, or an alternative
- Has a volume control function

In addition to the group solution, each individual was required to extend their groups implementation. This extension did not have to match any set specification, and was instead left up to each individual to decide. For my individual extension, I implemented a 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 both more intuitive and more responsive than using the 16 digit keypad, from the MBED peripherals board.

For our group solution, the specification was broken down into three separate areas. This allowed us to focus our expertise and interests on the aspects best suited to each group member. 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).
- User interaction with the device in order to display information and adjust settings.

2.2 Discussion of Technical Aspects and Challenges

My group deconstructed the specification into three major areas requiring technical innovation. These three main areas, as seen in the previous section, equally correspond to the main technical aspects and challenges faced in meeting the specification for the group project.

The first of these challenging areas regarded the retrieval of data from the CAN bus, and the decoding of any packets received to extract their content. Data transmitted on the CAN bus does not necessarily arrive at regular intervals, and it is impossible to determine when packets may arrive. Due to the rapid communication the utilisation of the CAN bus enables, it is possible for a large number of packets to arrive in a very short space of time. The ability to receive and process each packet, without any being missed, is crucial for the correct functionality of the device. This therefore requires a highly optimised, interrupt based solution, capable of receiving and managing a large amount of data transfer in a very short space of time. Furthermore, the extraction of data from any received packets presents an additional technical challenge. Received packets may be of a variety of different lengths and types, so a robust system must be implemented in order to retrieve their data.

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. No information is known about the audio output apart from the frequency, duration and volume of each note. The implementation of an audible output therefore requires the generation and manipulation of a wave. The most logical solution to generating a wave would be the use of a wave look-up table, giving values for a range of different values. However, this presents the additional challenges of storing the wave-table in memory on the device, and interpolating between values accurately and rapidly. Further areas of technical challenge arise in the production of a more authentic sounding product. In order to increase the realism of the implemented sound samples, additional features such as 'Attack Decay Sustain Release' models of a waves lifespan [7], equal loudness curves [8], and sound synthesis [9] must also be implemented into the solution, vastly increasing the complexity. Sound synthesis is a particularly difficult problem, due to the possibility of containing expensive mathematical calculations. These calculations could slow down the processing of audio samples, resulting in a strange sounding end product. As a result, the efficiency of sound synthesis is a vital consideration, and a significant challenge.

The third technical challenge presented is the ability for a user to interact with 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. There are many different methods of interaction with the device: the most notable being communication via serial from a desktop computer, or interaction through the peripheral accessories provided, such as the 16 digit keypad. The user's interaction is not predictable or consistent, and similar to the CAN bus data, a large amount of data may be transferred in a short period of time. Further challenges are faced as any user interaction with the device requires parsing, and executing the required user interaction without having any significant effect on the audio output of the system. Furthermore, the challenge of implementing a user interface is made increasingly difficult by the required ability to interact with many other aspects of the project. Aspects such as the audio synthesis code and the CAN bus code must be carefully manipulated and tweaked in order to provide a level of user interaction for settings such as the overall output volume, and filtering data from the CAN bus. Furthermore, any user interface must be both intuitive and user-friendly, requiring careful design considerations.

Chapter 3

Description and Discussion of Team-based Solution

3.1 Description of the Team Solution

The team solution following the completion of the project was a successful working product. Every point on the specification was fully met, and the solution incorporated additional features such as a user friendly shell style input, or the ability to synthesise multiple channels of music. This chapter sets out a full technical description of each different aspect of the finished product, explaining in detail how each feature is implemented.

Controller Area Network Bus Processing

The first technical challenge faced in the implementation of the group's solution was the retrieval of data from the Controller Area Network bus (CAN bus). The implemented solution uses an advanced system to ensure that no transmissions of data packets are missed. Each time data is transmitted on the CAN bus the MBED board generates an interrupt. The interrupt service routine copies the data transmitted on the CAN bus into the devices memory, ready for processing at a later stage. By avoiding the processing of any data in the interrupt handler, it ensures that no race conditions can occur, and therefore data transmissions cannot be missed.

The data packets are stored in memory as a queue. This enables a first-in-first-out approach to processing of the data to occur. The CAN data packets are decoded according to the given specification. There are two possible types of packets received: a basic data packet containing information for the frequency, volume, channel and control data of a note; or a text packet containing information for a track name, beats per minute of the track, and channel numbers and names [10]. If the packet received is a data packet, it interacts directly with the audio code: turning on, off, or adjusting notes corresponding to the information contained. Alternatively, if the packet is a text packet, it requires further processing to extract the information contained.

Text packets may arrive in multiple packets due to the length of text contained. Following the arrival of text packets, a check-sum is sent. If the length of the text received does not match the value of the check-sum, the text is ignored. This ensures all packets were successfully received, and that no packets have been corrupted. If the check-sum value is correct, and the text packets have been correctly received, they are parsed in order to extract their content into a more usable format. This is achieved through a state based text-parser which draws inspiration from a stack-overflow post [11]. The text parser iterates through the text, alternating state depending on whether the current location is within a word, inside quotation marks or inside white-space. This breaks the text into smaller sections and removes quotation marks, enabling it to be presented in a useful format, such as displaying the track name on the LCD display or on the computer screen.

Shell User Interface

To provide a user-friendly and simplistic user interface, enabling a high level of control and interaction with the device, a shell style user interface was created. Typically used for interaction with a computer, a shell is a device that takes a user input, and converts the input (command) into the desired functionality. Simply put, a shell is "a program that takes your commands from the keyboard and gives them to the operating system to perform" [12]. For the MBED system, communication with the device occurs via serial, with a computer running GNU screen (terminal) used for both input, and feedback from the device. This enables the user to enter commands via the computer keyboard, or other input method, and the MBED device to then execute these commands.

When an input is entered into the computer terminal, an interrupt is generated on the host MBED board. The interrupt handler first determines whether the user's input is a special case character (BACKSPACE, or ENTER), or a regular character. If the input is a regular character, it is stored in memory for processing at a later stage, as well as being transmitted over serial to be displayed on the computer screen. If the input character is BACKSPACE, the last character entered is removed from the input stored in the devices memory, and a special character sequence (\x8\x8) is sent to the computer in order to delete the displayed characters from the screen. This will only occur if the user has an input to delete. Finally, if the user inputs the ENTER key, a flag is toggled on, indicating that the user's input requires processing. Processing of the user's input is not done within the interrupt handler in order to avoid unpredictable behaviour through the potential occurrence of race conditions.

Processing of the user's input occurs in three different stages. First the input is parsed, where it is split up into separate words. This simplifies the logic needed to determine the correct functionality to perform. Parsing is done using a similar method as the CAN bus text packets [11], however modifications have been made to set a limit on the word count of the user's input. This prevents the situation where an input consisting of a large number of separate words results in excessive memory consumption, causing the device to crash from writing into unassigned memory. Following the parsing of text, the user's input is matched to possible input commands using `strcmp` for string comparisons. This determines the correct functionality of the user's input. Once the correct functionality has been determined, the user's command is executed. Depending on what this command is, the execution of the user's input may involve alterations being made to variables, states of the system, or may interface with the audio, or CAN bus segments of the group solution.

Keypad and LCD interaction

The keypad and the LCD additionally provide useful tools for interaction with the device. The specification states that the user must be able to display useful information on the host board, explicating the need to incorporate the LCD display into the solution [6]. The keypad was configured to work on an interrupt based system, as opposed to a poll-based system. This generated an interrupt each time a key was pressed. The interrupt handler first determined whether the user input was a valid key, filtering out any undesired interrupts generated from the row and column scanning of the keypad. Furthermore, a debounce system was generated, only allowing one successful interrupt to occur once every half a second. This was incorporated using SysTick to determine the difference in time between inputs. Following a successful input, the correct functionality was then determined and executed.

The LCD display was heavily utilised by the group solution and had a wealth of features. The LCD code enabled writing to only a section of the display, allowing different information to be displayed on each line. This was achieved by manipulation of the device drivers, writing to only a section of the displays memory prior to updating the display. Furthermore the LCD display was further extended, adding the capability for scrolling text. This was implemented using pointer manipulation to incrementally iterate through a text string and periodically update the display with this string using a system timer (SysTick).

Audio Processing

The audio processing aspect of the group solution was very thorough, and incorporated a vast array of different features to produce a high quality audio output from the end product.

No information is known about the desired audio output, and consequently wave-forms must be generated from first principles. This is implemented using a look-up table. In order to store a look-up table on the device, a tool was created to convert an audio sample into a list of values. This enables the possibility for alternate waves tables such as a piano samples to be incorporated into the solution, in addition to the more simplistic sine waves, or the square waves incorporated in our teams solution. This list of values was then stored in the flash memory on the device. For each note requiring audio playback, the wave-table is stepped over, and interpolated to provide readings between different values. This value at each moment in time then determines the output voltage. Only one wave table is used for all the audio playback, different frequency notes are simulated by adjusting the rate at which the wave-table is looped over. This is achieved by manipulating the step-size: stepping through the wave-table at a faster rate corresponds to playback of a higher frequency note. The step-size for each note is determined as each note is turned on by simply dividing the frequency of the note by the frequency of the wave-table. This method for the generation of notes is known as an oscillator [13].

In order to allow playback of more than one note at any one time, a method of audio synthesis known as Additive Synthesis was implemented. This is a simplistic method of representing multiple notes and involves adding together the wave-forms of each notes to create a composite wave [9]. In the implemented solution, it is not possible to combine wave tables together as we only use one. Instead, the values generated by the oscillator method at each moment in time are added together. In order to create an output with consistent volume, the end value of the additive synthesis is also divided by the total number of notes that are playing at that moment in time.

In order to enable each individual note to have a volume level, the value read from the wave-table is multiplied by a number between 0 and 1 corresponding to the volume level of the note. However, the human ear does not perceive all frequencies equally, and further modification to the volume of each note has to be made in order to compensate for this. Equal volume loudness curves describing how volume levels perceived by human ears varies for different frequencies are described by the ISO 226:2003 standard [14]. Data points from this curve are stored in the device's flash memory, and interpolated to provide an accurate correction value for each frequency. This is applied to each individual note to produce a consistent sounding output.

To further the accuracy of the output audio sample, an attack-decay-sustain-release model is implemented for each note. This simulates the effect of a musical instrument's variation in volume and sound over time by incorporating a state based system. As the note is initially turned on it enters the attack phase. In the attack phase the volume rises from 0 to 100 (maximum volume) over a short duration of time. Following the attack phase, the note enters a decay phase, where the volume level is gradually lowered down to the sustain level specified in the CAN bus packets. When a note gets turned off, it exits the sustain phase and enters the release phase. In the release phase the volume level of the note drops from the sustain value to zero over a small period of time [7].

In order to achieve a highly optimised result, the mathematics involved in the implementation of the above features avoids the use of floating point types. This is due to the inefficient implementation of floating number systems on the host LPC1768 board, caused by a lack of floating point hardware [15, 16]. Instead, a fixed point library was created. This uses the first 18 bits of a 32 bit number as the integer value, and the last 14 bits as the decimal value, allowing accurate decimal calculations to be performed rapidly. The implementation of a fixed point arithmetic library therefore provided a significant improvement to the capabilities of the system.

Finally, once the audio signal is generated, it is outputted to the Digital to Analogue Converter (DAC). This occurs through the use of Direct Memory Access (DMA). This removes the requirement to process any information: the DAC is given direct access to the memory locations needed for outputting audio sound. Rather than requiring any processing, the DAC simply reads the values whenever required to create the desired sample rate (20,000Hz). There is a significant increase in performance caused by the use of Direct Memory Access, due to the removal of interaction with the devices central processing unit (CPU) [17]. In our group solution, a speaker is connected directly to the output of the DAC to provide audio playback. The speaker is created using an SSM2305 amplifier circuit [18]. This amplifier is a low power device capable of running off the MBED board's 5 Volt power rails, meaning it does not require an external power source.

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 an aspect of the project that they found interesting, and throughout the project we had regular feedback on all our work to ensure 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.

Douglas Parsons (Myself)

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-decay-sustain-release curve to improve the realism of the output sound. Additionally, Liam implemented 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, and a speaker circuit.

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: 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 required 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, Shivam discovered that the device contains an additional 32Kb of memory, typically reserved for communicating to a USB or Ethernet device [16]. 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.

Douglas Parsons (Myself)

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 ability 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's interest in music technology meant that he was eager to work on the audio code for the project. He implemented a simplistic system, allowing musical notes to be generated by looping, 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 sped 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 [19]. In addition, to create a more realistic sounding output from the device, an attack-decay-sustain-release state model was incorporated for each note. This gives an initial spike 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 [7].

Chapter 4

Evaluation and Testing of Team-based Solution

4.1 Description of the Team's Testing Strategy

Throughout the duration of the project, my group was thorough and rigorous in our continual testing of the device through a variety of means. Through use of the version control software, git, new features were added as branches to the main project. This enabled new features to be tested externally to the main body of the project, and furthermore enabled strict code reviews to be performed on each contribution to the group solution. In addition, a debugging library was implemented, allowing messages to be output for debugging purposes only, and easily removed from the solution. Furthermore, the group solution underwent extended testing and was continually running in the laboratories throughout the duration of the project.

Individual Testing

As our team used the version control system, git, we were able to produce new branches of the project for each new feature. This not only enabled each new features to be tested separately, but enabled different group members to work on alternate features simultaneously, with no possibility of interference. After the completion of a new feature, prior to the inclusion within the group solution, each section of the feature underwent individual tests. For a greater ease in testing the project, a debugging library was created, enabling serial messages to print diagnostics. The debugging library was only included in the binaries installed on the device if the make command included it. This means, for the purposes of testing it was possible to use `make debug` and have messages containing text, or values printed out over serial. However, once sufficient testing was done, no alterations had to be made to the code to remove these statements, and instead the project could be installed by calling `make`. This therefore provided a simplistic tool, effectively creating two distinct modes of operation: a debugging mode, where information was output over serial, and a production mode, where normal operation ensued.

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 solution running for long periods of time within the labs, and continually checking to ensure nothing unexpected had occurred at any point. In addition, each new addition to the solution was fully tested by each group member on its incorporation within the project.

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

The group's testing strategy proved very efficient as capable. Through the use of individual branches for new features, any issues caused by new additions would not affect other parts of the group solution, and enabled a much simpler debugging process. Through the testing strategy many different issues were identified, and dealt with without before becoming significant problem. Through leaving the group solution running for long periods of time, a large number of different test cases and situations were explored, such as the audio playback for many different tracks. This allowed us to ensure that the functionality of the end product was consistent. 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 Aspect of the Group Solution

The group solution presented has the potential to solve an issue that may arise when considering the usability of the device interface. The host board has very limited interaction capabilities, with only the 16 digit keyboard providing a potential method for user input. For many users, in particular those with physical disabilities, this method of interaction may not provide the most user-friendly experience, and a lack of alternative modes of operation could cause a fundamental issue. Our solution to the issues that may arise from the lack of diversity of input methods comes through the incorporation of an alternative method of communication with the device: the group solution implements interaction through a computer via a shell style interface, therefore making it more accessible to a wider range of users.

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, such as speech recognition software. These alternative methods for user interaction with the device may provide a much more user-friendly experience for many users. Methods of Human-Computer interaction for disabled users have been the focus of a large number of studies, such as Simpson and Stephanidis [20, 21]. The improved usability of computer interaction provided through methods described in their works could extend to the MBED board through the group's implementation of multiple interface methods. Hence, the group's solution prevents a potential social issue that could occur for physically limited users.

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. However, my individual contribution also consisted of the modification and extension of previously existing LCD screen code in order to achieve a much greater degree of control over the attached LCD display.

The shell style interface was set up using an interrupt based system from the computer. This enables users to enter commands into GNU Screen running on the computer through any input method. These inputs form commands that are then processed and executed on the device to perform tasks such as displaying information on the host board, adjusting device settings, or allowing users 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 entering '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

The user is therefore provided with a high degree of control over the device's functionality. The user interface provided by the shell therefore enables the user to display useful information on the computer screen, on the device, enter and display custom messages, adjust the devices mode of functionality (printing packets, or playing audio), and also gives the user control over the audio output. Interaction with the onboard LCD display can be clearly seen through the above commands, hence explicating the necessity for alterations made to LCD display code. The high degree of control over the LCD additionally enables additional features such as scrolling text, or writing multiples times without clearing the LCD display.

5.2 Discussion of Technical Innovation and Implementation

The implementation of the shell style interface and the improvement of the LCD display code encountered many different technical challenges throughout. The first technical challenge faced was setting up a system enabling user input to be transmitted to the device. Communication with the MBED board occurs via serial. As text is entered into the terminal window of the computer, an interrupt is generated. This input transfers the text entered in the terminal into the devices memory for processing at a later stage, as well as printing it out on the display so the user can see it. Setting up this interrupt provided a significant technical challenge. Once this was set up a strong level of communication with the device was enabled. A further challenges was then presented with the allocation of memory to store the user's input. The simplest solution for allocating memory was used: a set input limit is defined, and any user input past this limit is not stored in memory, or printed to the computer monitor - effectively it is ignored. Furthermore, special case characters such as BACKSPACE required additional management to remove both the characters displayed on the monitor, and those stored in memory.

When the enter key is pressed (\r\n received on the interrupt), 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 white-space. This breaks the user's command into its 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 dissection 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 occurring 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.

In order to successfully perform the desired functionality, settings are altered within many different areas of the project. The implementation of the shell therefore required a fully detailed understanding of how each individual component of the project functions in order to manipulate and alter existing code. Modifications were made throughout the project, and included: the addition of a settable ID within the CAN bus code, the addition of a filter in the CAN bus to ignore unwanted data according to the preset ID, the addition of a volume control in the synth code, and a major overhaul of existing LCD code to allow a much greater degree of control over the display.

The overhaul of the LCD display code provided a significant technical challenge. The initial desire was simply to allow scrolling text on the display: however, it soon became clear that further alterations were required. Initially the LCD code would clear the display prior to every write command. To enable text to be written to separate lines of the display independently, or to enable scrolling text on one or both of these lines it was therefore necessary to remove this behaviour, and discover how to write only to a section of the LCD display. This required accessing only certain memory addresses of the LCD display buffer in order to prevent clearing parts of the display. Following the successful implementation of the capability to write to each individual line of the LCD display, the next target was to give the ability to scroll text across the display, in order to effectively increase the number of characters that could be read. This was more straightforward and was achieved using pointers to manipulate the current position in a string, and a system timer (SysTick) to update the relevant section of the display at regular intervals.

5.3 Testing Strategy, and the Results

Due to the nature of the extension project, testing was relatively straightforward. Using the debugging library, it was very easy to check the effects of any command that was entered to ensure correct behaviour. Therefore, any errors or undesirable behaviour were rapidly discovered. Furthermore, the interface became a central aspect of the group solution, and as a result was used by each member of the group on a regular basis to interact with the device. Through each individual's interaction many test cases were encountered, and therefore my individual extension underwent a continual and thorough testing scenario.

The results of the testing strategy were very positive. The shell style interface had a large number of features by the end of the project, and worked very well. Furthermore, my individual contribution provided a very valuable tool throughout the project due to the high level of interaction that it provided.

Chapter 6

Summary and Conclusions

6.1 Reflective Summary of Team Work

Team work was an essential part of the project, and an area in which my group excelled. Each individual was allowed to select their own areas of the project to work on, and furthermore regular meetings and code review sessions both ensured that no individual was overwhelmed, and ensured that contributions to the solution were of the highest quality.

The group solution achieved every point of the specification in an elegant manner, and had a wealth of additional functionality through additions to the project. Examples of these additions include: the ability to synthesise music and play back multiple notes on each channel and multiple channels simultaneously; an improved user interaction via a shell style interface complete with the ability to adjust settings and change mode of operation without re-installing the device; and a more realistic sounding audio playback through the use of attack-decay-sustain-release models.

6.2 Reflective Summary of Individual Work

My individual extension project was the implementation of a style shell user interface, enabling user interaction with the MBED board through interaction with a computer. This occurred primarily through serial communication between the MBED device and the computer. The addition of the shell style interface provided a wide range of valuable tools throughout the course of the project, and the end result was a great degree of control over the host board. Examples of the control given include:

- The ability to play notes
- The ability to change the device volume
- The ability to write custom text to the LCD display
- The ability to change which channel of music is being listened to
- The ability to view useful information, such as the track name, channel, or what notes are currently being played
- The ability to view CAN packets as they are received on the CAN bus
- The ability to scroll text on the LCD display

This enabled a great degree of control over the host board, and provided invaluable tools throughout the course of the project. This extension project was initially very challenging and was an ambitious task to undertake due to the scale of the implementation combined with any desired features. However, the end result was very rewarding and one that I can be proud of.

6.3 What went well

My group had a very successful end product, and this was the result of many aspects of the project going very well. From the beginning of the project, it was clear that each member of the group was very enthusiastic about producing a high quality end result, which enabled a positive working atmosphere where each group member was free to work on whatever aspect they desired as the project moved on. The use of the version control software, git, was a particular strength of the project. This enabled group members to work on separate branches for individual contributions, and allowed everything to be neatly organised, and kept track of.

Furthermore, we were aware of the limitations of the host MBED board prior to the start of the project, and therefore continually conscious of performance. Each feature implemented into the product was carefully considered, and its implementation was carefully monitored to ensure that it did not require excessive processing. Coupled with careful study of the data sheet for the host board, we were able to extract a great amount from the device, and produce a high quality finished product that was well optimised.

6.4 What went poorly

While the vast majority of the project went very smoothly and according to plan, there were two occasions where significant issues were encountered. The first problem we encountered was in the first week of the project. After spending a week attempting to receive data from the CAN bus and failing, we decided to use the online tool-chain to set up a device to send CAN packets, and plugged this into our existing code. This worked perfectly, and led to the discovery that the CAN bus was not working everywhere within the lab. Because of this, a lot of time within the first week of the project was spent attempting to correct code that was already working.

The second issue that was faced was the inconsistent use of version control software. The majority of the group (Myself, Shivam and Liam) all used git, and github as a central location allowing us to effectively work as a team on the project. Mingzhao, despite being shown how to use git many times, chose not to use any version control software. This made interaction with his contributions to the project much more difficult, and keeping up to date with his progress a challenging task.

6.5 What could have been improved

Due to the nature of the project, there is a large amount of extension to the basic functionality that can be added. Our group solution extended the base implementation a long way, however additional features were nearly implemented that could have made a good improvement to the finished product. There are three examples of features that were not implemented in the final version of our project, but would have contributed to an overall improvement of the project.

The first improvement would be the incorporation of multiple wave tables. Our group solution used square waves to generate audio samples, however, there is scope for allowing any wave to be used. Through use of the shell it would be possible to alternate between different wave tables without the need to re-install the binary on the device. This would allow a further degree of control over the device, and could be used to alternate to more pleasurable sounding outputs for certain channels.

The second potential improvement that could have been made links with the first, and is the implementation of drum samples to create a more complete sounding audio track. On the CAN bus sixteen channels of audio data are sent corresponding to the different instruments. Channel 11 typically represents a drum track, and for our project this channel is being ignored. This therefore is a potential improvement that could have been made.

The third potential improvement is a greater level of interaction between the shell and the keypad user input methods. As the keypad interaction was only implemented into the group solution near the end of the project, the interfacing between the two methods is not flawless, and it is possible for minor discontinuities to occur. For example, if the user changes the volume using the keypad the new volume level is displayed on the LCD display. Changing the volume once again via the shell does not update the value displayed on the LCD display. The finished product could therefore be improved by improving the interactions between the two devices.

6.6 Lessons Learned

The project has served as a valuable learning experience throughout its duration in a variety of ways. First of all I have developed a much stronger understanding of the technical intricacies of programming in C, in particular with regards to embedded systems - while the language itself is very minimal and straightforward to learn, there are many small details that require careful thought and consideration, such as the use of pointer manipulation. In addition, concepts such as interrupts, SysTick, or direct memory access are not entirely unfamiliar, but the implementation desired features using these tools has been a new experience. The project has also greatly developed my understanding of music technology. Through meeting the project's requirements to generate audio samples and our own desire to make these samples sound more realistic, a lot of research has been done into this field. It has been fascinating to research and learn about all the different aspects involved in our group's solution, such as sound synthesis, ADSR curves, and equal volume adjustment. In addition, I have not worked on a group project before where version control software has been so strongly used. Throughout the project I have gained a much better understanding of how a group should function, and I have developed a much better understanding of the communication between group members and how collaboration works as a result of this.

Overall there is a huge amount that I can take away from the Embedded Systems Project, and the project has provided a valuable opportunity to further develop my knowledge of embedded systems programming, working as a group, and music technology.

Bibliography

[1] Wikipedia. (2015) Can bus — wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/w/index.php?title=CAN_bus&oldid=656136296 [Accessed: Apr. 15, 2015]

[2] ARM.mbed. How mbed works. [Online]. Available: http://developer.mbed.org/handbook/How-mbed-works [Accessed: Apr. 15, 2015]

[3] P. Cooper. Mbed resources. [Online]. Available: module.cs.york.ac.uk/empr/mbed_resources/graphics/MPC_TNG_2 [Accessed: Apr. 15 2015]

[4] ——. Mbed pins. [Online]. Available: http://www-users.cs.york.ac.uk/ pcc/MCP/MbedPins.html [Accessed: Apr. 15, 2015]

[5] W. Voss, *A Comprehensible Guide to Controller Area Network*. Copperhill Technologies Corporation, 2005. [Online]. Available: https://books.google.co.uk/books?id=PU6ppO3XbUwC [Accessed: Apr. 15, 2015]

[6] U. of York Department of Computer Science. (2014/2015) Empr assessed project, open group and individual assessment. [Accessed: Apr. 16, 2015].

[7] W. Pirkle, *Designing Software Synthesizer Plug-Ins in C++: For RackAFX, VST3, and Audio Units*. 2 Park Square, Milton Park, Abingdon, Oxon: Focal Press, 2014. [Online]. Available: https://books.google.co.uk/books?id=oaYgBQAAQBAJ&pg=PA286 [Accessed: Apr. 15, 2015]

[8] A. Kefauver, *The Audio Recording Handbook*, ser. Computer Music and Digital Audio Series. A-R Editions, 2001. [Online]. Available: https://books.google.co.uk/books?id=bbNfr_bwUXwC [Accessed: Apr. 15, 2015]

[9] E. Miranda, *Computer Sound Design: Synthesis techniques and programming*. Taylor & Francis, 2012. [Online]. Available: https://books.google.co.uk/books?id=ajr4_nS3X0gC [Accessed: Apr. 15, 2015]

[10] P. Cooper. (2014/2015) Hapr 2014-2015 basic data packet. University of York Computer Science Department. [Online]. Available: http://www-users.cs.york.ac.uk/ pcc/HAPR/2014-15 [Accessed: Apr. 15, 2015]

[11] (2012) Parse string into array based on spaces or “double quotes strings”. Stack Overflow. [Online]. Available: http://stackoverflow.com/questions/9659697/parse-string-into-array-based-on-spaces-or-double-quotes-strings [Accessed: Apr. 15, 2015]

[12] J. William E. Shotts. What is ”the shell”? [Online]. Available: http://linuxcommand.org/lc3_lts0010.php [Accessed: Apr. 17, 2015]

[13] M. Puckette, *The Theory and Technique of Electronic Music.* World Scientific Publishing Company, 2007. [Online]. Available: https://books.google.co.uk/books?id=TCtnWBfyhbwC [Accessed: Apr. 17, 2015]

[14] *International Organization for Standardization*, ISO Standard 226:2003 Acoustics – Normal equal-loudness-level contours, 2003.

[15] C. Kormanyos, *Real-Time C++: Efficient Object-Oriented and Template Microcontroller Programming.* Springer, 2013. [Online]. Available: https://books.google.co.uk/books?id=xTNEAAAAQBAJ&pg=PA202 [Accessed: Apr. 17, 2015]

[16] *LPC176x/5x User Manual*, NXP Semiconductors, 4 2014, rev. 3.1. [Online]. Available: http://www.nxp.com/documents/user_manual/UM10360.pdf [Accessed: Apr. 17, 2015]

[17] M. Barr, *Programming Embedded Systems in C and C++*, ser. O'Reilly Series. [Online]. Available: https://books.google.co.uk/books?id=a2Q6U0b36rMC&pg=PA62 [Accessed: Apr. 17, 2015]

[18] (2008) Ssm2305 filterless high efficiency mono 2.8 w class-d audio amplifier. Analog Devices, Inc. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/SSM2305.pdf [Accessed: Apr. 18, 2015]

[19] M. Rafiquzzaman, *Microprocessors and Microcomputer-based System Design.* Boca Raton, FL, USA: CRC Press, Inc., 1990. [Online]. Available: https://books.google.co.uk/books?id=s2KPvKCQm7sC&pg=PA31 [Accessed: Apr. 15, 2015]

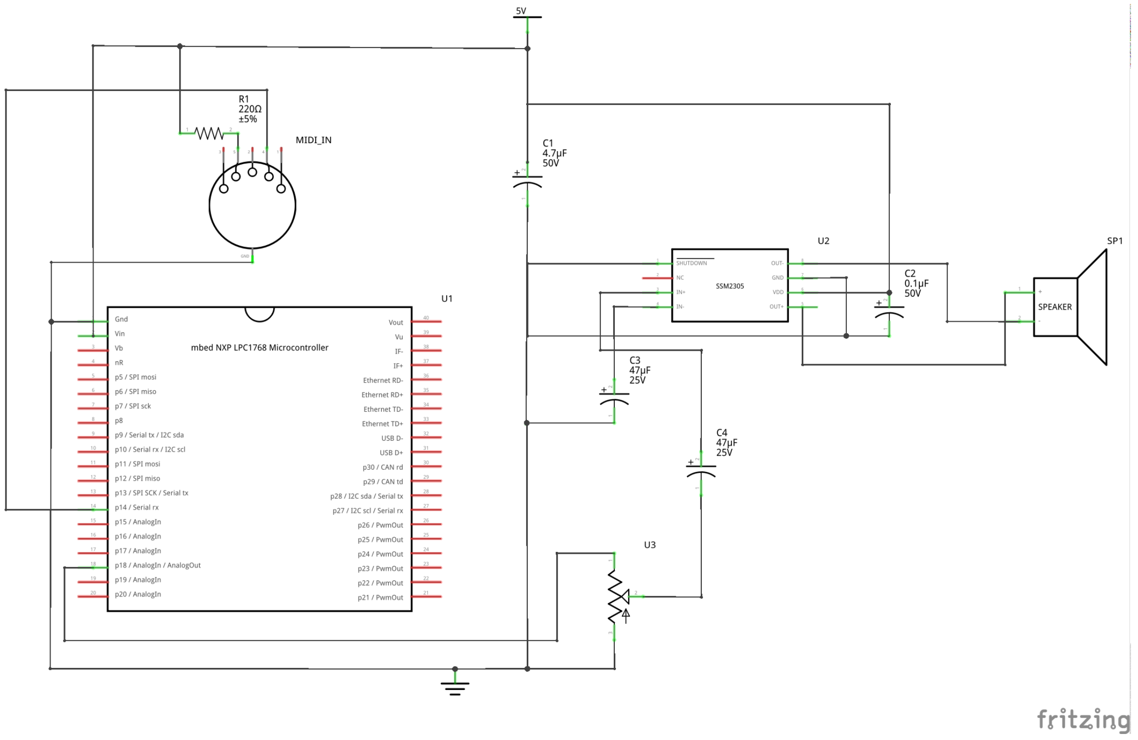
[20] R. C. Simpson, *Computer Access for People with Disabilities: A Human Factors Approach.* Boca Raton, FL, USA: CRC Press, Inc., 2013. [Online]. Available: books.google.co.uk/books?id=3H3NBQAAQBAJ [Accessed: Apr. 15, 2015]

[21] C. Stephanidis and M. Antona, *Universal Access in Human-Computer Interaction: Design Methods, Tools, and Interaction Techniques for eInclusion: 7th International Conference, UAHCI 2013, Held as Part of HCI International 2013, Las Vegas, NV, USA, July 21-26, 2013, Proceedings*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, no. pt. 1. [Online]. Available: books.google.co.uk/books?id=HJW5BQAAQBAJ [Accessed: Apr. 15, 2015]

Appendix A

Specified Documents

A.1 Speaker Circuit



A.2 Meeting minutes

EMPR Group 10 Meeting
Beginning of Week 2 (12/01/2015)

Meeting Chair: Shivam Mistry
Secretary: Liam Fraser
Members present: Shivam Mistry, Liam Fraser, Douglas Parsons, and Mingzhao Zhou(Edward)

- Set up private GitHub repository. Ensure everyone has access
- Discuss mandatory code review. Decide on feature branches and pull requests when merging into master. Code review can be done at this stage.

Tasks:
- DMA Synth code using Wavetables / Fixed point library - Liam
- Screen and Keypad - Edward
- Can Bus - Shivam and Doug

EMPR Group 10 Meeting
Beginning of Week 3 (19/01/2015)

Meeting Chair: Douglas Parsons
Secretary: Shivam Mistry
Members present: Shivam Mistry, Liam Fraser, Douglas Parsons, and Mingzhao Zhou(Edward)

Task assessment:

- Initial synth code /w fixed point library - done over weekend by Liam
- Can Bus - spent the previous week in the lab trying to get can bus working. Eventually figured out that the can bus was broken. Was fixed by Friday of week 2.
- Implement basic keypad support - Edward

Further Tasks:

- Now that Can bus is working, look at processing data and text packets - Shivam
- Start working on individual project: a shell for the MBED - Doug
- Continue work on screen and keypad - Edward
- Add sample support for drums and begin on individual project which is piano mode. This requires multiple wavetables for one instrument, support for ADSR curves and a MIDI input - Liam

EMPR Group 10 Meeting
Beginning of Week 4 (26/01/2015)

Meeting Chair: Liam Fraser
Secretary: Edward White
Members present: Shivam Mistry, Liam Fraser, Douglas Parsons, and Mingzhao Zhou(Edward)

Task assessment:

- Can Bus working - can process text and data packets
- Piano mode and dependencies for that well underway - Liam
- Working on volume control with keypad - Edward

Further Tasks:

- Implement a Queue and checksum checking for CAN bus - Shivam
- Continue work on Piano mode, also look at building a couple of speakers for in the lab - Liam
- Continue work on screen and keypad - Edward

EMPR Group 10 Meeting
Beginning of Week 5 (02/02/2015)

Meeting Chair: Mingzhao Zhou
Secretary: Douglas Parsons
Members present: Shivam Mistry, Liam Fraser, Douglas Parsons, and Mingzhao Zhou(Edward)

Task assessment:

- Piano mode is working although sounds like an organ. Parts for speakers ordered - Liam
- Shell is mostly implemented. Waiting to merge in new Synth code before continuing - Doug
- Queue and Checksum for CAN bus implemented: can now play the tracks in a reliable fashion - Shivam
- Working on the visual display function on LCD - Edward

Further Tasks:

- Build some speakers and continue work on Piano mode - Liam
- Shivam: write memory allocator using the upper 32K of SRAM typically reserved for peripherals such as Ethernet and USB
- Doug: Fix scrolling text on shell and improve the LCD code
- Continue work on keypad - Edward

EMPR Group 10 Meeting
Beginning of Week 6 (09/02/2015)

Meeting Chair: Shivam Mistry
Secretary: Liam Fraser
Members present: Shivam Mistry, Liam Fraser, Douglas Parsons, and Mingzhao Zhou(Edward)

Task assessment:

- Piano mode improved, Implemented the sustain pedal for midi input. - Liam
- Begun work on improving the LCD functionality, scrolling text fully implemented across two different lines - Doug
- Begun work on memory allocator, implemented majority of features, but not fully tested - Shivam
- Working on improving interrupt based solution for keypad - Edward

Further Tasks:

- Liam: Work on Equal Loudness to provide a better sounding output
- Doug: Improve efficiency of the LCD scrolling text - currently noisy
- Shivam: Complete full testing on memory allocator
- Edward: Continue work on Keypad Interrupt system

EMPR Group 10 Meeting
Beginning of Week 7 (16/02/2015)

Meeting Chair: Liam Fraser
Secretary: Mingzhao Zhou
Members present: Shivam Mistry, Liam Fraser, Douglas Parsons, and Mingzhao Zhou(Edward)

Task assessment:

- Equal Volume Loudness implemented, and codebase tidied up - Liam
- Vastly improved efficiency of LCD scrolling text - Doug
- Fixed bugs with the memory allocator - Shivam
- Working on improving interrupt based solution for keypad - Edward

Further Tasks:

- Liam: Further work on piano extension
- Doug: Merge in improved Synth code and fix any issues caused
- Shivam: Finishing touches on memory allocator
- Edward: Continue work on Keypad Interrupt system

EMPR Group 10 Meeting
Beginning of Week 8 (23/02/2015)

Meeting Chair: Douglas Parsons
Secretary: Shivam Mistry
Members present: Shivam Mistry, Liam Fraser, Douglas Parsons, and Mingzhao Zhou(Edward)

Task assessment:

- Further improvements made to the Synthesis code - Liam
- Bug fixes for the Shell - Doug
- Extension project work - Shivam
- Working on improving interrupt based solution for keypad - Edward

Further Tasks:

- Liam: Finalize Piano code
- Doug, Edward: Merge in Edward’s keypad code
- Shivam: Finish individual extension project

EMPR Group 10 Meeting
Beginning of Week 9 (02/03/2015)

Meeting Chair: Liam Fraser
Secretary: Douglas Parsons
Members present: Shivam Mistry, Liam Fraser, Douglas Parsons, and Mingzhao Zhou(Edward)

Task assessment:

- Finished implementation of Piano - Liam
- Bug fixes for the Shell, and Edward's code merged in - Doug
- Extension project work almost completed - Shivam
- Helped merge in keypad code into group solution - Edward

Further Tasks:

- Doug, Edward: Debug and test keypad code.
- Everyone: Prepare for group presentation

A.3 Evidence of Preparation

EMPR FORMATIVE FEEDBACK SHEET (MINI PROJECTS)

CANDIDATES Douglas Parsons, Shivam Mistry

MINI PROJECT [✓ 1] 2 3 (ring as appropriate)

PART 1. SELF REVIEW

List the functionality and describe the main features of your solution.

Stage 1: LEDs light up in turn for one second. Process loops 5 times in total.
Stage 2: LEDs display a 4 bit number - each no. displayed for one second + entire process loops twice.
Stage 3: Delays are implemented off an interrupt based timer, rather than idle loop

Given more time, or another attempt, what, if anything, would you add or do differently?

Separate code into libraries so it can be reused at a later date

PART 2. TUTOR APPRAISAL.

	Poor 1 2 3 4 5 Excellent
a. Understanding of the concepts the mini project.	1 2 3 4 [✓ 5]
b. Functional performance of the demonstrated solution	1 2 3 4 [✓ 5]
c. Evidence of originality	1 2 3 4 5

ANY OTHER OBSERVATIONS

Signed: Tutor N.F. [illegible]

Student. [illegible]
[illegible]

Date 16/10/2014

EMPR FORMATIVE FEEDBACK SHEET (MINI PROJECTS)

CANDIDATES Douglas Parsons , Shivam Mistry

MINI PROJECT 1 [✓ 2] 3 (ring as appropriate)

PART 1. SELF REVIEW

List the functionality and describe the main features of your solution.

stage 1: uses serial port to print messages to the terminal
stage 2: initialise i2c bus, and then determine how many devices are connected, as well as their addresses.
stage 3: write a message to the LCD display
stage 4: allow user to enter numbers on the keypad + have them displayed on the LCD
stage 5: added basic calculator func, allowing add, sub, div, mult and clear.

Given more time, or another attempt, what, if anything, would you add or do differently?

The calculator could be refined a large amount, adding add. features or using a different (parsing) approach.

PART 2. TUTOR APPRAISAL.

	Poor 1 2 3 4 5 Excellent
a. Understanding of the concepts the mini project.	1 2 3 [✓ 4 5]
b. Functional performance of the demonstrated solution	1 2 3 4 [✓ 5] –
c. Evidence of originality	1 2 3 4 5

ANY OTHER OBSERVATIONS

Signed: Tutor [illegible] Student. Date 20/11/2014.

EMPR FORMATIVE FEEDBACK SHEET (MINI PROJECTS)

CANDIDATES Douglas Parsons, Shivam Mistry

MINI PROJECT 1 2 [✓ 3] (ring as appropriate)

PART 1. SELF REVIEW

List the functionality and describe the main features of your solution.

Stage 1: Prints out the value of a DC input to the device.

Stage 2: Program uses a lookup table to write a sin wave to the DAC.
Allows use of setfrequency and setamplitude
Sweeps a range of freq. and amplitudes: 100 - 5KHz

Stage 3: Mirrors a sin wave input through the ADC out onto the DAC.

Given more time, or another attempt, what, if anything, would you add or do differently?

Use direct memory access

PART 2. TUTOR APPRAISAL.

	Poor 1 2 3 4 5 Excellent
a. Understanding of the concepts the mini project.	1 2 3 4 [✓ 5]
b. Functional performance of the demonstrated solution	1 2 3 4 [✓ 5]
c. Evidence of originality	1 2 3 4 5

ANY OTHER OBSERVATIONS

Signed: Tutor N E Peeers Student. Date 20/11/2014