A Report for Little Toy Blue's Microprocessor-Based Toy Project, Stratum, a Musical Synthesizer

Project Dates
6 March 2018 to 14 April 2018

Team Pear
Educational Toy Division Engineers

Austin Wong
Erik Sangeorzan
Jared Grimes
Kevin Zentner

Date Prepared: 14 April 2018

Prepared For
Little Toy Blue
Nikola Tesla
Vice President for Innovation

Executive Summary

The purpose of this report is to document the progress our team has made in constructing a prototype of a microprocessor-based, educational toy.

Little Toy Blue recently asked its Educational Toy Division engineers to examine the current state of the microprocessor-based educational toy market, propose a solution for future entry into this market, and create a working prototype of a proposed solution by the assigned deadline. Little Toy Blue does not currently have any presence in the microprocessor-based educational toy market and believes a successful entry into that market this summer will help the growth of their business.

Our team, Team Pear, was tasked with three criteria regarding the construction of our prototype. These criteria took the form of a technical constraint, a time constraint, and a design constraint. We fulfilled the technical constraints of creating our prototype on the E100 processor and an Altera DE2-115 development board, supporting a wide variety of input/output peripherals. We fulfilled the time constraint by successfully developing a working prototype within the allotted time period of one month and delivering our final presentation, with a demonstration included, by April 14, 2018. Throughout the past month, we punctually delivered a proposal presentation regarding our idea and turned in a progress report. This final written report is due by April 16, 2018. Lastly, our idea, Stratum, is a unique toy that will help with Little Toy Blue's entry into the microprocessor-based, educational toy market.

Our concept, Stratum, is an easy-to-use musical synthesizer intended for children. Our prototype includes various instruments (a basic triangle wave, grand piano, acoustic guitar, and drum kit) with multiple octaves. We have included the option to record and playback up to three, ten-second-long tracks and a metronome with options of 80, 120, and 160 beats per minute. We have also created an intuitive user interface to interact with the various features we have implemented.

We were inspired to delve into the music educational due to the lack of a sustainable, cheap synthesizer on the market, as well as the education benefits from learning about music. Currently, the market does not have any cheap synthesizers. The only other options are complex websites which most children, and even adults, would have a hard time learning, or actual physical synthesizers that are very expensive. Our product would provide a cheap, simple solution for music amateurs. Furthermore, learning music at a young age has also been shown to provide many future benefits to those who studied it. With this in mind, we hope that our customers, most being young children, will benefit greatly from our product once it is released in stores.

Table of Contents

1 INTRODUCTION

The following section will explain the purpose for this project, establish context, and describe how we went about fulfilling the requirements.

1.1 Background and Context

Little Toy Blue currently has no presence in the microprocessor-based educational toy market. As a result, earlier this year our team was assigned to Little Toy Blue's Educational Toy Division as engineers. Little Toy Blue wants to enter the microprocessor-based educational toy market, and after receiving funding, Nikola Tesla assigned our team to create a toy to help Little Toy Blue enter this market.

In order to prepare for this task, first we individually visited local toy stores to research the current state of the educational toy market. We took note of the various input/output devices, and using the information that we collected, we individually proposed a unique toy which would help Little Toy Blue stand out from the other toy companies. We then decided on one of our four designs and began creating a prototype. As a result of our efforts into design and development, we had created a prototype of our initial idea, Stratum.

1.2 Meeting the Criteria and Constraints

Stratum, which will be detailed further in section 1.3, is an inexpensive, intuitive musical synthesizer with advanced capabilities comparable to that of a professional synthesizer. The three main criteria and constraints were to have a unique toy in mind to help Little Toy Blue's eventual entry into the microprocessor-based educational toy market, build our prototype using an Altera DE2-115 development board and the E100 processor, and create a functional prototype by the April 14, 2108 deadline. We fulfilled all three of these requirements and used the following input/output devices: a keyboard, touch screen, VGA display, SD card, and speaker.

1.3 Overview of Stratum

Stratum is an intuitive, functional, and kid-friendly synthesizer. Music is played via keyboard input, and timbre of sound can be modified by the user from the touchscreen. For our prototype, four different timbres of sound — grand piano, plucked acoustic guitar, triangle wave, and drums — can be chosen. The current octave can be modulated up and down for a given instrument through keyboard input, and an integrated metronome allows the user to keep time while playing. Stratum also features multi-track layering, allowing for layer recording, playback, and deletion. In the prototype, this feature is implemented for up to three layers.

1.4 Uniqueness of Our Idea

We chose to make Stratum because of the numerous benefits music can have on the academic development of children. Research has shown that music improves social skills, grades, standardized test scores, and acceptance rate to numerous programs including medical school. We believe that once a final version of our product reaches store shelves, children will love using their creativity to make and play music on Stratum.

2 DESIGN DESCRIPTION

The following section will describe our initial, barebones prototype, the hardware we used to construct our final prototype, and how our prototype meets the given criteria and constraints.

Before starting any work on developing our prototype for Stratum, our team met and devised overall goals for the development for the prototype and final version of Stratum. Since we wanted to create a completely functional product by the given deadline, we had to create a prototype that reasonably pushed our limits. We did not want to take on a task that was too difficult to complete, but also hoped for an adequate challenge that would help each member in our team grow as engineers.

Next, we wanted to make our prototype usable by and useful for all people, especially younger children. The purpose of this project was to create a microprocessor-based, educational toy, and audience for most educational toys is children. We knew that the intuitivity of the design would have a major impact on the user experience of children, so we took this into deep consideration in the midst of development.

This leads us into the final goal we had while developing Stratum, which is to make it as intuitive as possible. We know many professional musical synthesizers can have an overwhelming amount of features. Rather than making a complicated user interface with tons of customizability, we decided to take a few of these advanced features from professional synthesizers and recreate them in a much simpler Stratum. With these design and technical preferences in mind, we began to design and develop our barebones prototype.

2.1 Barebones Prototype

Our first goal was to construct a barebones prototype, which we did by March 23, 2018. Our working, barebones prototype included the ability to play within a full octave from the keyboard. The prototype also includes the ability to stop the repetition of sound waves upon key release. This is necessary for the success of Stratum because that is how all synthesizers handle key presses. Having completed a basic, working form of our prototype, we then continued to add more features to our prototype.

2.2 Development Board and Input/Output Devices

We developed our prototype using an Altera DE2-115 development board and an E100 processor. Our prototype requires a VGA display, keyboard, touchscreen, SD card, and speaker. The VGA display and touch screen are needed for displaying an interactive graphical user interface, so that the user can easily access the different prototype capabilities. The user interface allows users to effortlessly layer music, select an instrument to play, and initiate a metronome tempo. The keyboard is used to actually play the notes and change octaves at the press of certain buttons. Instead of being used to interact with the user, the SD card is used as storage for the necessary data, such as sound samples and image files, that our prototype needs to function properly. Specifically, the actual hardware used to create and test the prototype are a 640 x 480 VGA monitor, Dell Mono Desktop Speaker, and Dell PS2 Keyboard. Despite the barebones functionality provides in our prototype, we plan to improve the hardware used in the final version of Stratum.

2.3 Meeting the Criteria

Our prototype meets the various criteria assigned to our group at the beginning of this project. We developed our prototype using the provided Altera DE2-115 development board and the E100 processor and used the default hardware given to us. Our prototype was completed by the April 14, 208 deadline and this report and a corresponding presentation was finished by the April 16, 2018. Additionally, our team promptly completed memoranda, ethics reports, progress reports, and delivered a presentation for our initial idea throughout the entire process of this project.

Our current user interface is very simple and intuitive, but, when designing the final product, we would make modifications to the way a user interacts with the layering. The current way we layer contains ten buttons for three layers, which could initially be quite overwhelming and confusing. Instead, we would add a layer selector which would greatly reduce the screen real estate used by the layering feature. Figure 1 on the next page shows the graphical user interface for the prototype that appears on a touchscreen.

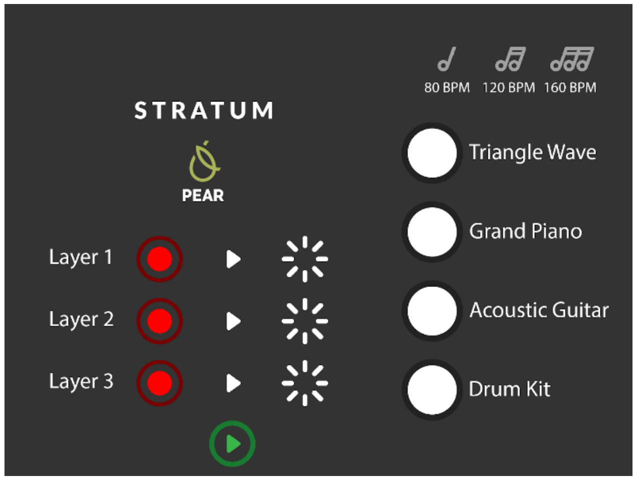

Figure 1: Graphical User Interface of our Prototype

This simplistic graphical user interface seen above makes it easy for the user to change instruments, change the beats per minute (BPM) of the metronome, and record, erase, or playback one or multiple layers.

3 MAJOR DESIGN DECISIONS AND TRADEOFFS

The following section will detail the choices made in regards to software and hardware utilized in the prototype, as well as desired software and hardware for the final product.

3.1 Instruments in the Prototype and Final Product

Choosing which instruments to include (and how many notes for each) in the prototype was a challenge. Due to memory and time limitations, we could only implement 76 different sound files, but we did not want to diminish Stratum's core ideas of being versatile and fun.
In the final product, we seek to include more instruments as mentioned previously, with more octaves for each depending on the instrument's range. However, there was not time nor space for these all to be added to the prototype.

Instead, we chose three classic instruments (besides the triangle wave option) that, added together, cover a broad range of musical capabilities. The first was a grand piano, which has a versatile, strong sound that can be used for bass lines or melodies. The second was a guitar, another very popular instrument that adds a softer, more mellow option for melodies. The last was a drum kit, which acts as the backbone of a tune and makes the feel of the song more concrete.

3.2 Hardware in the Prototype and Final Product

On the hardware side of this project, we chose equipment that we felt made the prototype as intuitive as possible, and we plan to implement similar, but higher quality, hardware in the final toy.

3.2.1 Keyboard

A standard PS2 keyboard was chosen for the prototype's note input because the layout of the keyboard is most similar to the layout of a piano, a hugely popular and most recognizable layout of notes. This was also chosen to provide a tactile response when a key is pressed, similar to that of a piano, and something the user would miss out on if inputting notes through a touch screen. The layout resembles an actual octave of keys as they would appear on a typical piano. Figure 2 below shows the keyboard layout.

Figure 2: Keyboard Layout Showing How Keys Correspond to Notes

This figure shows the simplicity of our prototype's keyboard layout. The colored letters on the keyboard in figure 2 shows the musical note to which that key corresponds. This is very easy for a user to get used to, especially since the F key and the G key line up with the musical notes F and G.

A similar keyboard would be implemented in the final product. Though we considered instead using a piano keyboard, using a computer keyboard makes the toy attractive to a broader audience, as there are more people that have used a computer than people that have played a piano. Intuitiveness is crucial for our toy, and using the most commonly recognized keyboard layout is the best option.

3.2.2 Graphical User Interface

To display the Stratum interface and provide another method of input, a touch screen was used in the prototype.

This display allows the user to easily view their different layer and instrument options. It also allows the user to change between instruments, record, playback, and delete specific layers, and choose a tempo at which the metronome will function. The main reason for this addition was that putting all of these inputs on the keyboard would have made it feel very cluttered. The keyboard was meant to somewhat emulate a piano, and having it do anything more than input notes and change octaves would have taken away from that purpose. Instead, shifting these functions to a separate screen significantly improved the ease of interaction with Stratum. We desired a method of displaying information about the user's musical creations in an interactive fashion, and this was the best hardware for that.

The final product would also utilize a touch screen, though a higher quality, more responsive unit. This would be even more useful in the final product than in the prototype because sliders to adjust note modulations, which were not completed for the prototype and will be added to the final product, are easiest to use with a touch screen. We would use a smaller touch screen with higher resolution to reduce wasted space, and in turn, cost, while also providing a much sharper and more engaging display.

3.2.3 Speaker

A speaker was crucial to the success of the prototype. The speaker used in the prototype was an inexpensive, mono-channel speaker; it was only able to output one channel of sound, unfortunately making the music feel flat. This was our only option and it slightly took away from the fun of the toy.

For the final product, an upgrade to stereo-channel speakers would be highly beneficial. Stereo sound is much more engaging as it broadens the width of the sound, sending multiple streams of music towards the user, as opposed to one. This would open up numerous possibilities for the development and use of the toy, as well. Users could, upon our implementation, record different sounds to different channels, so the resulting music would have certain instruments playing out of certain speakers, allowing a brand new level of musical depth for the user to explore.

3.2.4 SD Card

Finally, an SD card, which is capable of storing a fixed amount of data, was used as the storage for the aforementioned sound files. This opened up the use of .wav files, which are discussed more in the next section. Our SD card was initialized with a concatenated file containing all of our instrument sound files, and this was crucial to implementing different instruments.

The final product would also use an SD card, but one with a larger capacity and quicker read and write speeds.

The combination of these upgrades would allow for many more instruments, more notes for each instrument, and less delay between the input of a note from the user and the output of the note through the speakers.

3.3 Software Implementations

Upon deciding to create a music synthesizer, we came up with a few core functions that the toy had to be able to perform. First, we sought to develop the ability to press a key on the keyboard and have a corresponding sound come out of the speaker. This is the fundamental ability of a synthesizer—to allow for key inputs and appropriate sound outputs. We tackled this goal by figuring out how to determine which key on a keyboard was pressed and translating this into the corresponding note. This note is then sent to the speaker for output.

During this step, we decided to limit our note input to one key at a time. This decision was made because we believed that inputting multiple notes at a time is nonessential if the user can record multiple layers to create chords instead. There was also a hardware limitation that would have made multi-note input a huge challenge, and this was not worth the time investment.

Next, we sought to add in different instruments and multiple octaves for each, broadening the spectrum of what the user can experiment with. To accomplish this goal, we learned how to implement .wav files as instrument options. These are files that contain a large number of samples for sounds of any length. We researched our options for usable .wav instrument files and eventually found suitable files. A module was written to transfer these .wav samples from the SD card to SDRAM for quicker read and playback, and another module was written to read the samples in SDRAM and play them back. These modules, which dictate the way different input and output devices behave, are called device drivers, and they are discussed further in section 5.1.

Finally, the third must-have feature was layering, which would allow the user to record the notes they play and play the layers back upon command. These features were crucial to allow for much more creativity and many different combinations of sounds to explore. This took the most time to comprehend and implement. We had to conduct research into how samples, or numbers responding to sounds, worked, and how they can be combined or separated. After figuring this out, we came up with algorithms for both recording input notes and playing them back upon command. Recording boiled down to saving a stream of input notes into a specific block of memory, and playback was a matter of adding together samples from specific memory blocks and playing the sum of these samples out of the speaker individually and sequentially.

3.4 Alternative Designs Considered

The initial concept graphical user interface for Stratum featured sliders that allowed for alteration of various properties of sound. Such properties would have included the frequency, volume, and shape of the sound waves we were creating.
Due to wave modulation never being implemented in the prototype, our team decided to completely change our user interface to make up for these unused features.

When we implemented layering, we needed to add the on-screen buttons necessary to record, play, and erase a specific layer. This resulted in thirteen extra buttons, requiring a massive space allocation required on the user interface. Lastly, we added the various tempos available through the built in metronome to the upper-right corner of the user interface. The result of these design changes was a clean, yet functional graphical user interface that we implemented in our prototype.

4 TESTING

The way our team approached inevitable bugs in our prototype is that we would alternate between coding and debugging. Rather than saving all the debugging for the end, we wanted to fix problems along the way while they were easier to find. The best way to handle this was to not begin new tasks until the current one being worked on was completed without any bugs. This often required a lot of trial and error to finally get our code to be fully-functional. Specifically, we encountered several bugs while working on playing the notes, changing instruments, and layering.

4.1 General Issues

Our team spent about just as much time debugging as we did actually writing code, especially for the programs related to playing notes and changing instruments. These components were the most important parts of our project, but also the hardest to implement. We often had to go back and rewrite a lot of code that was functional initially, but once we added more components, such as layering and switching between instruments, our code was no longer functional. However, from a lot of effort, we were able to fix the part of our code related to playing the correct sound.

4.2 Complications Near the End of Design Process

We structured our design project by making everything independent to begin with. We wanted to get the triangle-wave synthesizer, piano synthesizer, guitar synthesizer, and drum synthesizer to work on their own before tying everything together. The main issue with this was there was a lot of overlap between the files, meaning variable names from different files conflicting.

Our Stratum prototype was made up of so many assembly language files created by different members of our team. Therefore, there was no surprise when we realized we had used the same variable names in our files. We had to spend tedious hours carefully going through all of our files and changing variable names for every instance of the variable in the file we were changing. This was difficult because if we missed one instance of the variable, it could have caused bad errors when trying to run the code. Luckily, with great attention to detail, we were able to successfully complete this part of debugging without running into further issues. Despite many complications and bugs during testing, we managed to fix nearly all of them in order to get Stratum to be fully functional with many great features.

5 REFLECTION

All of the members of our team agree that this was one of the hardest projects any of us have ever had to do. This was the first time building software in assembly language for us all and we had to learn a lot all the way. We learned a lot from this project. Not only did we gain so much knowledge in assembly language programming, but we learned how to work as a team and how to conquer problems that may initially seem near impossible, especially in such a small time frame.

5.1 Technical Knowledge Gained

We began this project being very intimidated of the work we had ahead of us and questioned how we would accomplish creating such a project. In our last lab exercise, we had to create a device driver for an input/output device so our code could interact with these devices. This was the first difficult task we had related to programming in assembly language. Everyone struggled with it, and we were all so relieved when it was successful after hours of work, both in and out of the lab. Later on in our development process, we had to write more device drivers, such as a VGA driver, SD card driver, and SDRAM driver. Writing and testing these drivers, in order to ensure full functionality, were simple yet crucial parts of our development process. When one of our team members had to write the code for one of these drivers, it was done quickly and relatively easily. This showed us exactly how much progress we had all made in developing our assembly-programming skills. We are all now confident in our abilities to create assembly-level software.

5.2 Learning to Work as a Team on a Major Project

Due to the complexity our project, it clearly could not be done mainly by one person. After some contemplation and discussion, we decided that we would try to have each person specialize in one area of the project. Then we planned on all working as a team to bring all of our individual components together once we each finished what we knew we had to get done.

Of course, some of these individual tasks, such as implementing recording and playback features, required two or more of us to work together or that we offer each other help. We were usually sitting next to each other when we were programming our individual components and communicated well to keep each other updated on the progress—or lack of progress—that each of us was making. This allowed us to offer assistance to each other, and helped the development process to progress at a faster rate.

Each of us was able to recognize when another team member ran into a problem that was similar to one that we had previously encountered, and therefore knew how to solve. For instance, while Kevin was working on making the top file that would tie together all components of the prototype, he ran into a problem with the touchscreen. When Kevin told Erik about the issue, Erik reminded Kevin that they had previously made a similar fix to the keyboard device driver when faced with a similar issue. This was a problem that Kevin may have spent much time trying to solve, but thanks to Erik's teamwork skills, it was solved quickly.

Time management was another skill we had to be quick to develop. Originally, we struggled with finding a time for the entire group to meet that worked for all of us.
We each have very busy schedules that often did not line up well for four-person collaboration. We were forced to learn how to get as much work done for our other classes as we could before meeting as a group. We began planning group meetings further in advance to allow for preparations to be made that allowed us to effectively work on our prototype and reports for this project.

5.3 Learning About How Projects Progress

This was the first time we were creating such a high-level project that required so much time and effort. Through making critical decisions that crucially affected design choices, we learned a lot about the reality of the rate of project progress. We observed that our tasks frequently took much longer to complete than we had originally planned. For instance, we had planned on finishing software development by April 8th, utilizing the last week for testing and adding features such as sound wave modulation. In reality, on April 8th, we were still implementing layering, and tying together all three synthesizers (piano, guitar, and drums).

These crucial components were not completed in their entirety until April 12th, which left us working very hard in the final two days before our deadline. Luckily, through great teamwork, we completed the prototype by the deadline, and we are proud of what we created.

6 CONCLUSION

Our assignment was to design a prototype for a microprocessor-based, educational toy for Little Toy Blue that can be developed fully this summer and put out into the market. We had to make the toy simple and mass-marketable. The prototype we created is indicative of the final product we have in mind: enjoyable, intuitive, and beneficial to children. We firmly believe that our work has fulfilled the needs of Little Toy Blue.

Appendix A: Honor Code and Work Distribution

We have neither given nor received unauthorized aid on this assignment, nor have we concealed any violations of the Honor Code. Table 1 shows the work distribution for our team.

Table 1: Workload Distribution

Name	Prototype Development	Final Report	% of Total Work
Austin Wong	Touchscreen response and graphical user interface display	Title Page, Executive Summary, Introduction, and overall editing	25%
Erik Sangeorzan	Layering, additional drivers, memory implementations	Major Design Decisions & Tradeoffs and overall editing	25%
Jared Grimes	Layering and sound file manipulation	Design Description and overall editing	25%
Kevin Zentner	Connecting components and creating instrument synthesizers	Reflection, Conclusion, and overall editing	25%

Signatures: Austin Wong, Erik Sangeorzan, Jared Grimes, Kevin Zentner