Campanile-Carillon Model: Phase II

Design Document Version II - 5/2/19

sddec 19-12

Client: Dr. Tin-Shi Tam

Advisor: Gary Tuttle

Gabriel Stackhouse - Software Lead

Grant Mullen - Integration Manager

Kienan Otto - Report Manager

Ryan Roltgen - Meeting Scribe

Sam Habel - Meeting Facilitator

Yicheng Hao - Power Systems Lead

Email: sddec19-12@iastate.edu

Website: http://sddec19-12.sd.ece.iastate.edu/

Table of Contents

1 Frontal Materials

1.1 List of Figures

1.2 List of Definitions

SCLC - *Student Carillon Leadership Council - responsible for maintenance and transportation of the future campanile carillon model*

MIDI - *Musical Instrument Digital Interface- communication protocol, digital interface, and electrical connector standard [5]*

Carillon - *A musical instrument composed of at least 23 carillon bells, arranged chromatically, and played by a keyboard that allows expression through variation of touch*

MCC - *Mobile Campanile Carillon Model*

2 Introduction

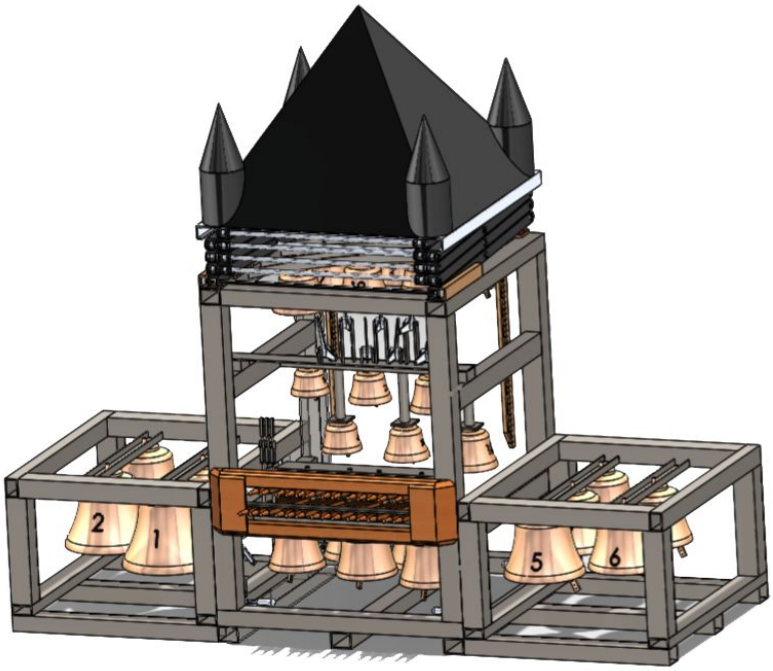

Figure 1: Model of the whole structure

2.1 Acknowledgement

We would like to acknowledge Dr. Tin-Shi Tam and Gary Tuttle for their guidance and support as we work to improve the program and design of the systems to be utilized in the MCC. We would like to thank the multiple ME 415 senior design classes that have worked on the MCC through the past few semesters and providing us with help on space and mounting information. Finally, we thank the Stanton Memorial Carillon Foundation and all the alumni and friends of Iowa State University who are providing the financial support to make this project a reality.

2.2 Problem and Project Statement

The Iowa State Campanile has been a symbol of ISU pride since the tower's construction in 1897 to memorialize Margaret Stanton. [1] However, alumni and friends of ISU can only admire the tower if they are on campus. Moreover, this audience may not be familiar with the beautiful sound of the 50 bells, as the carillon -- the instrument inside the campanile -- is typically only played at noon.

To remedy this, there has been an enormous effort to create a Mobile Campanile Carillon model (MCC). For the past 2 years many groups have been involved in the

realization of this goal, and great progress has been made in that time. This model will in effect showcase the bells, display history and donor information, and implement a playable carillon at ground level. Our task is to design the electrical components of the model. In addition to the widescreen digital display to showcase a “documentary” slideshow, we will design a Guitar Hero style interface to make the carillon playable by a layperson.

2.3 Operational Environment

As the name suggests the intention of the model is to make the carillon mobile. In addition to mobility, it will need to be able to withstand extended periods of being both indoors and outdoors. This means that the components we select must be water resistant and able to handle temperatures in both the summer and winter months. Additionally, since the model will routinely be transported, it will also need to be durable and retain continuity during transport.

2.4 Intended Users and Uses

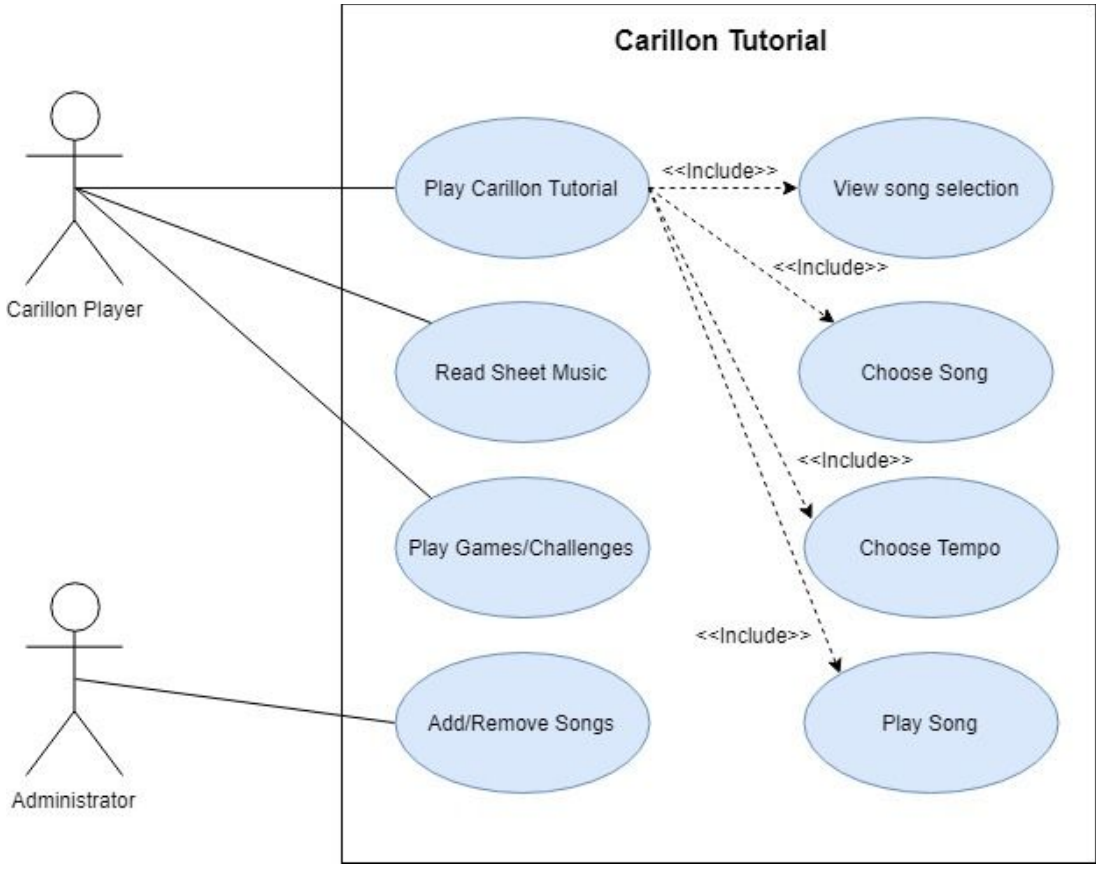

Figure 2: Use Case diagram of system

There are two distinct groups who will be interacting with the MCC: curious onlookers who aren't familiar with playing an instrument, and musicians. Since the model will be displayed at many different venues, the client wants anyone to be able to interact with the model and make beautiful music, no matter what their musical background may be. To achieve this, we must design an interface that makes playing the instrument easy, intuitive, and most importantly, fun. However, because musicians will also be using the instrument, we need to be cautious that none of our solutions interfere with the expected response from the instrument that the Carillonneur expects.

The other users we must consider are the members of the Student Carillon Leadership Council -- the individuals responsible for upkeep and transportation of the MCC. Any system we implement must maximize reliability, so the model does not require constant maintenance. In addition, any components with reasonable chance of failure must be documented so that individuals with no electrical expertise can diagnose and replace the components in question.

2.5 Assumptions and Limitations

Assumptions

1. Although the MCC will need to withstand outdoor conditions. It will not be left outside during extreme conditions. Such as extreme cold, heavy rain/snow, or dangerous winds
2. We expect the MCC to be fabricated in the summer of 2019 so that our designs can be implemented before the Fall of 2019

Limitations

1. Each component of the design shall be reliable and minimize failure rate.
2. Although no discrete financial limitations were specified, the project should minimize unnecessary costs while maximizing reliability and durability.
3. The design should be simple enough to be repaired and operated by a layperson. To assist in this, we will create documentation to help repairs and operation of the system.
4. The design should be in compliance with the goal to remain portable, and modular.
5. There will be limited software support from the team after we graduate.

2.6 Expected End Product and Deliverables

We expect to develop two independent final products requested by Dr. Tin-Shi Tam. Each will be used as marketing material for Iowa State University to take to various events to represent the school. The deliverables are:

1. A functional tutorial guide on how to play a song on a 27-key carillon. The hardware will be located inside the campanile model and will have a battery life of at least 8 hours. This program will be able to read and display songs imported by the SCLC, and will allow for different levels of difficulty depending on the skill of the operator.
2. A standalone, battery operated information station which will display information about the university as selected by Dr. Tin-Shi Tam. The station will also be able to display interactive 3D models of structures on campus, such as the Iowa State University Campanile. The battery life of this station will match the campanile system's, lasting at least 8 hours.

2.7 Nature of Content

As our program is intended to help users play music on a mobile carillon, we had to take certain precautions. Our software needs to be simple enough to not encounter issues when attempting to run in a non-development environment, and our hardware systems need to be low-powered enough to avoid losing power in the middle of an event. Our available space in the campanile is very low, so we are unable to include larger or additional batteries to prolong battery life on more powerful hardware.

3 Specifications and Analysis

3.1 Proposed Design

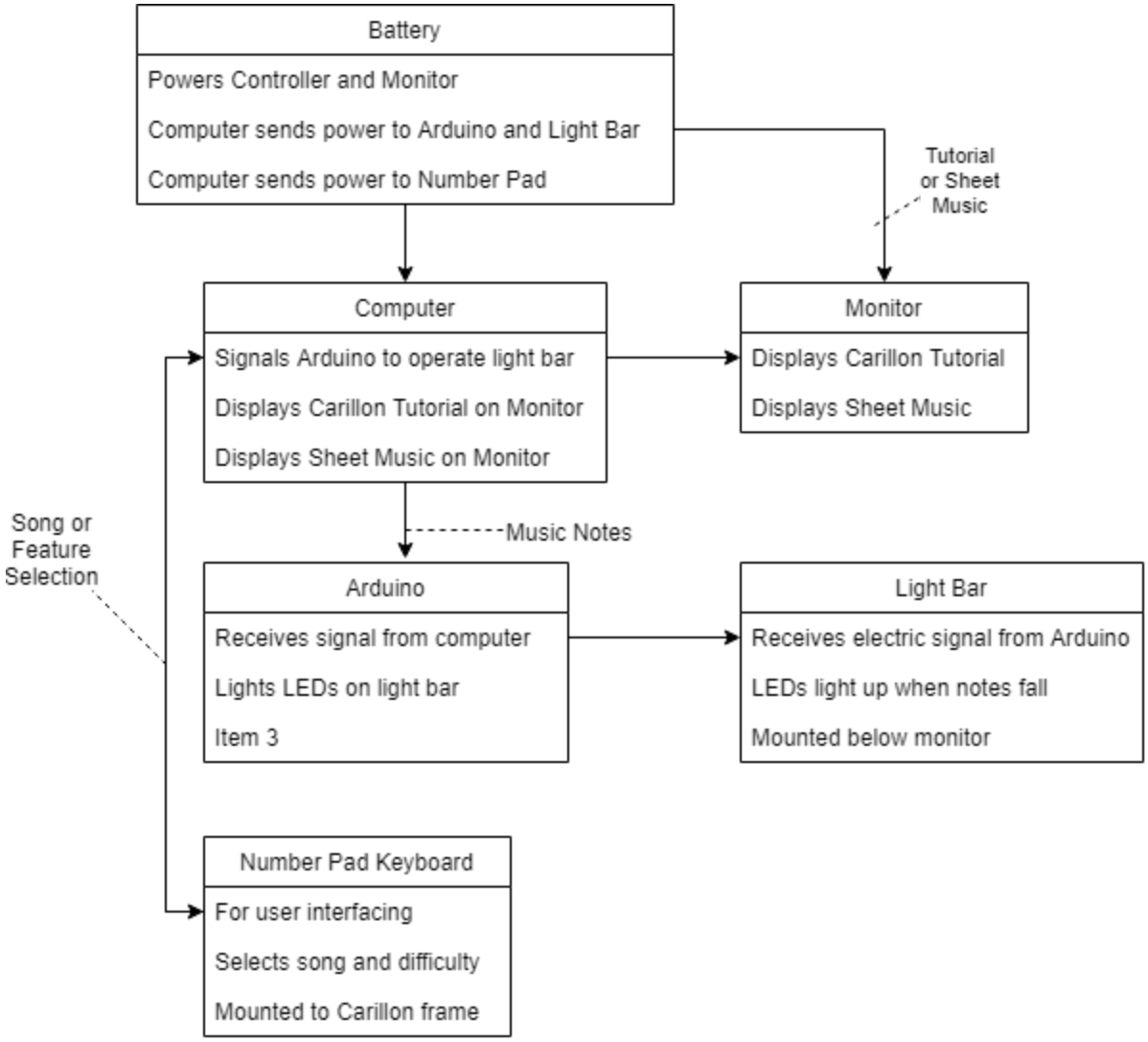

Figure 3: Block diagram of system

Our plan is to have a screen on the MCC that will display falling notes onto a keyboard that mimics the layout of the carillon batons (keys of the carillon). We have a program from the previous group that we are working on debugging to run reliably and efficiently. There is also an LED bar that perfectly lines up with the batons on the carillon that will turn on the light over the note that should be played. This is for the user to have a secondary method to see what note should be played next. To control the LEDs and run

the software we will use a Windows-based machine that is powerful enough for the program to run properly.

The group has made a decision on a monitor that meets the specifications of the available space while still being wide enough to have a one-to-one layout for each baton on the carillon. To interact with the program, we are planning on getting a keypad for the device to move through the menu screens and to select what song to play. Once the song starts playing, the user doesn't need to provide any additional input until they decide to choose a new song.

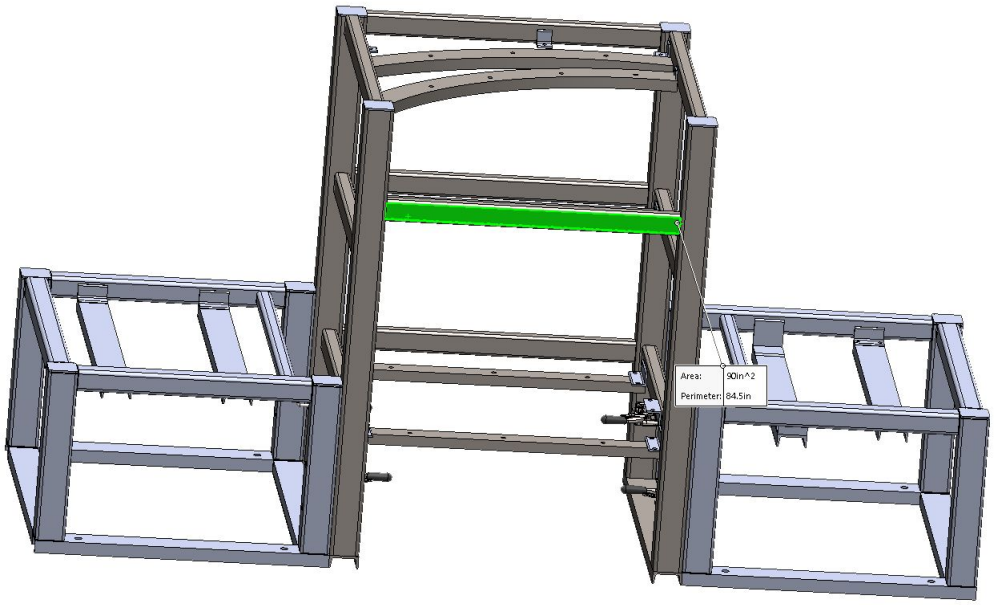

Figure 4: Monitor mount location on structure

We have recently discovered that the Raspberry Pi used by the previous group wasn't powerful enough to run the program at the desired resolution. The CPU was consistently overheating, and the program was suffering significant slowdowns due to inadequate hardware. For now we are proceeding with the backup Linux laptop that was purchased for the project. In the future, once our program is finalized and finalize the necessary hardware specifications, we will order a new device to run the program. It was also recently requested by the client that we port the program to Windows to provide more ease-of-use, so the new device we choose will likely be running Windows.

Our plan is to design and build a power system that will power our display, the Windows laptop, and any other peripherals needed. To accomplish this, we will use lead acid batteries, an inverter to power the monitor and the laptop, and a high amperage battery

charger. We will likely make use of 300W to 400W inverters in order to have plenty of headroom to power all system components without overheating. The capacity (in aH) of our lead acid battery will be determined by our desired battery life. The capacity could always be increased at a later date by wiring additional batteries in parallel. The batteries we choose will likely be deep cycle marine batteries, similar to many batteries used in solar systems. Last, we will make use of a high amperage battery charger in order to charge the batteries to full capacity in a short amount of time. We currently have a bill of materials, but we are waiting to order until we get the go-ahead from the client.

One of the peripherals we are considering implementing is a battery monitoring system that will take the battery voltage and estimate our current battery life. This could be put onto a custom pcb alone with our microcontroller.

3.2 Design Analysis

On the software side of the project, we have designed the project in the SFML library with a C++ codebase, with the end-goal of easy expandability. Abstract classes and inheritance gives the advantage of easily adding new features to the program on a whim, which has proven to be beneficial throughout the semester as the client thinks up of new things they'd like added. Additionally, the project is fairly client independent; while it certainly took some work to complete the port from Linux to Windows, the fact that SFML and C++ run on both operating systems made it a much more manageable task. Last, both SFML and C++ are known for being extremely lightweight, which will save the client a considerable amount of money on the hardware to run the program.

The main disadvantage of the software comes from using a gaming framework (SFML) instead of a full-blown engine, such as Unity. While SFML is certainly more lightweight, it requires *far* more implementation of basic gaming concepts -- things that Unity would have out of the box. This slows development considerably, and has the potential to produce a less marketable product. Unfortunately, due to strict time constraints of the project, we were forced to build upon the previous group's SFML implementation rather than rebuild the project from scratch in Unity. Despite this, we believe that we have the project in a state where additions are easy to implement, so any disadvantages that SFML brings can be effectively mitigated.

Hardware-wise, our system has five main components: the Windows PC that runs the program, an input device to control it (still TBD), the monitor, LED light bar, and an Arduino Uno to communicate between the PC and the light bar. The low amount of components brings many advantages, such as ease of setup, ease of diagnosing any

issues, and a low rate of failure. It also makes upgrading the system very easy: if a more powerful computer is needed, for example, one simply needs to drop in the new PC in place of the old one. Simplicity is key in hardware design, and we believe we have achieved that here.

One disadvantage of our hardware design is that, despite having a low point of failure, one component failing theoretically *could* hinder the entire system. For example, if the Arduino Uno fails, the light bar will no longer work as designed. The Windows PC and/or monitor failing means the system can't be used at all. We can mitigate this the best we can by providing backup hardware options where possible, but the risk is still there nonetheless. In addition, our design currently includes very low-powered hardware, which may prove to be insufficient. We already have had to upgrade the Raspberry Pi to a Windows PC, for example, and it's looking increasingly likely that the Arduino Uno will have to be upgraded next. Accounting for better hardware options from the beginning may have been a better option.

Our battery system design includes a 12 volt battery charger, a 12 volt deep cycle lead acid battery, and a 12 volt 300W inverter. A advantage to this type of system is that it is simpler and does not require switching delivered power between the inverter and the wall outlet. The simpler system is much easier to implement. Because of the reduced part number, it takes up less space and reduces costs. The disadvantage to this type of system is that if there is an outlet available nearby, then the system will be turning 120V AC to 12V DC back to 120V AC. This isn't ideal because of the power and thermal inefficiencies. However, given that the carillon likely won't have wall power readily available most of the time and other constraints, this was compromise we made.

Since the carillon model is designed to be mobile, we will have to be considerate of the space available for our system to fit in. This could cause problems if we rush on buying materials for the battery without ensuring that, when assembled, they will all fit in the allotted space. To help with this, we are meeting with the ME 415 team and referencing CAD files that have the exact dimensions of the model. Both will be extremely important as we continue to design and build the battery solution that will have the right balance between high capacity vs. taking up a small amount of space.

4 Testing and Implementation

4.1 Interface Specifications

The core of our group's carillon tutorial system will be a single computer. Initially, we will design everything to operate on a laptop which will provide power to the Arduino and light bar, as well as provide data to both the monitor and Arduino. All data will be

processed onboard the central computer. Our original design called for a touch-screen monitor for user-interfacing, but because of the high cost of a touchscreen monitor that fits specifications, the design has been altered to use a small touchpad or keypad to navigate menus within the program.

4.2 Standards

- IEEE 1625-2004: *IEEE Standard for Rechargeable Batteries for Portable Computing [4]*
    - Guidelines to design, test, and evaluate different battery components to create well-rounded solution
    - Will use when designing battery solution for electrical components
- IEEE 14764-2006: *Standard for Software Engineering – Software Life Cycle Processes – Maintenance [4]*
    - Guidelines for testing and maintaining software in the long term
    - Will use as we implement the software, ensuring that any team that follows us has an easy time of maintaining the software
- IEEE 2003-1997: *Requirements and Guidelines for Test Method Specifications and Test Method Implementations [4]*
    - Best practices for software test-cases, including types, readability, complexity, etc.
    - Will use while writing test cases concurrently with designing the program

4.3 Hardware and Software

A low-power laptop is being used to test our code, which is being written entirely in C++. Our final design will not use a laptop to run the program, but our final hardware solution has not been decided yet. In the past, our team has used a Raspberry Pi 3 Model B for testing, but it does not have enough processing power to run the program smoothly. To test the final hardware solution, it must be able to run the program smoothly at 30 frames per second or higher for all songs. Our minimum operable frames per second is 24.

The code will be tested mainly through edge cases and routine operational tests. The edge cases will be empty MIDI files as well as MIDI files with a large amount of notes to put strain on the system. The routine test involves uploading a MIDI file and displaying it without errors. [5] The program must also be able to display uploaded sheet music as a PDF.

The battery will be tested by running the system for 8 hours or more, which is the requested battery life of the system.

4.4 Functional testing

Our team has a few major points for our functional testing:

We are focusing our testing on a few of the major components, with the chief focus being placed on the light bar. The light bar is custom built with addressable LEDs, so we are testing for accuracy and synchronization. The LEDs light up in conjunction with a note falling down the screen. Specifically, an LED will light up when a note in its corresponding column reaches the bottom of the screen and turn off when the note leaves the screen. We will test over multiple songs and see how accurate the light bar is when hitting the notes. Our tests must conclude that the LEDs accurately function in coordination with the on-screen notes.

A major concern for our system, as well as any system, is making sure all new changes don't break old, previously-functional code. This requires testing all existing functionality at each iteration of our program. We need to ensure no feature introduces any unintended effects to the program.

Our program should be able to run at a minimum of the full monitor resolution and 30 frames per second. It should remain constant at these specs and show no fluctuations to maintain a smooth, consistent appearance. We will test this by stress testing the program by playing MIDI files with a large amount of notes and seeing how the FPS is affected. [5]

The monitor should be able to display the graphics appropriately in both light and dark settings. This will require a group of people to observe the display from various distances and angles in different brightness situations to determine the correct settings and any changes to the graphics.

The power system should be able to power all the electronics with some headroom in case any minor design constraint changes in the future. Ideally, no more than 50 to 80 percent of the maximum power rating of the inverter would be used nominally.

4.5 Non-functional testing

The non-functional testing is more simple. We have a few major, overarching requirements for our end product:

The product shall:

- Have a tutorial which is usable by any person of any skill level
    - Have a test group see how easy they think it is to play the instrument with our program
- Include an interface for adding additional midi files and sheet music [5]
    - Test if adding multiple songs or multiple pdfs causes any issues when selecting a song or document
- Be easily troubleshooted
    - Create a document and see if our client and others monitoring the program are able to successfully start and navigate through the program

4.6 Process

LED Lights Bar - Tested using the Arduino

- Compatibility tests
- Connection of wires

Monitor - Tested efficiently

- Size sufficient for keyboard
- Testing brightness settings

Battery - Tested using lab instruments

- Power output tests
- Power efficiency tests

Battery Level Indicator - Tested using lab instruments

- Voltage indicate range tests
- Power consumption tests

Software - Tested through the use of a survey

- Usability

- Reliability
- Aesthetic appeal

4.7 Implementation Results and Challenges

Much of the testing will need to be done without the final hardware available. Due to the amount of time needed to acquire the correct monitor and processor, our testing is limited to running the program on a laptop and displaying at a custom resolution on a simple desktop monitor. Debugging the program will be difficult because our team has inherited the majority of the code from the previous group.

If needed, fixing the light bar will be a huge roadblock. We have only one electrical engineer on the team, and the previous group did not provide documentation on their work on the light bar. Our integration with the ME group that is creating the frame for the carillon model may take more time than expected. Our group has struggled to analyze the 3D model of the frame, and we are limited in our ability to plan the space we will need for the various components of the system.

4.8 Simulation and modeling

Regarding simulation, we have done the initial math to calculate our battery life based upon our inverter efficiency, battery capacity, power requirements of our components, and our desired battery life. We also plan to simulate our battery monitoring system before drafting our PCB. This will be in addition to the breadboard and perfboard models we will make. We don’t currently have plans to simulate software, as the high level of interaction makes this difficult.

5 Closing Material

5.1 Conclusion

The tutorial software is functional, but it needs new features and better polish. In terms of functional requirements, there are still bugs present when interacting with the light bar and it cannot currently display PDF sheet music. For non-functional requirements, the user interface needs to be improved aesthetically. The code has been successfully ported to Windows 10 and all our development will be done on that operating system.

The battery solution will use 300W-400W inverters to power system components without overheating. The desired battery life is 8 hours, and the planned capacity of the lead acid battery will fulfill this. We will create a battery life indicator on a custom PCB to monitor the charge level of the battery solution.

Over the summer, the bells will be installed on the frame and the base section of the carillon model will be complete. At that point, we will have a better idea of where our components will be placed, and we can make a final decision for component housing. Our goal is to completely and correctly implement the software and hardware portions of our design by the end of the fall semester.

5.2 References

[1] “Campanile | Iowa State University Admissions.” Admissions, www.admissions.iastate.edu/traditions/campanile.php.

[2] Website of previous group: https://sddec18-01.sd.ece.iastate.edu/

[3] Synthesia https://www.synthesiagame.com/support/faq

[4] IEEE Standards https://standards.ieee.org/

[5] The MIDI Association https://www.midi.org/