S.A.T. – Simple Audio Thing

Senior Project Proposal

Megan Francis

Abstract

Abstract Here
purpose of the **Abstract** is to give a quick summary of the entire paper, so readers can decide if they want to read the rest of your paper. It's NOT the first paragraph of the rest of your paper.

This paper details the process of my senior project.

The **Introduction** will be an overview of the project. It should place the project in context of your interests and goals for what you want to know when you graduate and pursue in the future. It should also explain how the project fits into the broad field of computing.

Introduction

SAT, or Simple Audio Thing, is a functional MP3 player designed for the Raspberry Pi. It has the basic functions of an mp3 player as well as a way to find and import songs to the music library. This project falls into several different categories of computing: front-end application design, very small database design (music library), programming (Integrating my Python code with existing programs and hardware), and operating systems (working with configuration files & Linux functions).

I have a serious love for music and a curiosity in physical computing and my project was really an attempt to merge the two interests. Unfortunately, due to complications, it ended up taking more of a turn toward application design than physical computing. Fortunately, application design is a field which I find very interesting and is an area I would like to pursue as a possible career in the future. I am still very interested in pursuing the original idea for this project, though I will likely switch the controller to being my laptop.

The **Background** section will likely be similar to your background section from your proposal. It should include a review of previous work in this area and how your project fits in.

Background

This section will contain projects that inspired and directed both my original idea and are related to how the project ended up. The first section covers related Raspberry Pi projects and the second section is a broader overview of related projects.

Raspberry Pi Audio System Projects

There were several Raspberry Pi projects that I found relevant to my project, many of which were audio systems using the Pi as a streaming device in multiple rooms and using a laptop, phone, or tablet as the controller for the device. These systems ranged from having Bluetooth speakers to wired speakers and often used the Apple program AirPlay to connect the devices.

With SAT I had hoped to skip making the laptop the controller and have the Raspberry Pi handle the music and multi-room capabilities. I had enough time to make it efficiently handle music that already exists in a music directory on the Pi, but it does not handle streaming music from websites or devices attached to it, nor does it handle multiple rooms although the GUI does have a section included in it (currently commented out) that contains the outline for handling multiple rooms using buttons.

Another audio system project I encountered was the PiPod. In the forums from raspberrypi.org a person with the username peterburk used a Raspberry Pi and the case of an old iPod Classic to create his own iPod.

This is project was an inspiration to me as I learned recently that Apple is no longer making the iPod classic. It also has some similarities to the way my project ended up as it has a GUI controlled audio system that handles a wired speaker or headphones. They are both controlled via hardware input (SAT by keyboard and the PiPod by iPod hardware).

Other Related Projects & Programs

There are several similar programs including iTunes, Windows Media Player, VLC Media Player, and many more that can play mp3 files in a similar manner to SAT. Most of these players are pretty intense and have lots of features and options as well as ways to play other audio files and even video files. For use on the Raspberry Pi, these systems would likely bog down a bit as the processor on the Pi is much smaller than your average computer.

XBMC is also one such system that is built specifically for the Raspberry Pi and is meant to boot directly into this software. It is mostly a streaming software that runs directly on the Raspberry Pi and has methods to play music, video, movies, and much more. It is similar to my project in that it is built for the Raspberry Pi and handles music playback.

Most of the programs that exist that are similar to my project have much more functionality and are less specialized whereas my project is just a simple mp3 player.

The **Methods** sections will provide the details of your project. They should cover what you intended to do, and some of the technology and issues you worked with. You may also note significant changes that the project took when you encountered roadblocks or other issues. Methods will note what technical choices you made and why you made them.

Methods

This section contains several sub-sections describing details of how the project came to be. The first section describes the original intention of the project and the goals I had for it. The second section details the technology used in the project and why I chose those technologies. The third section explains the roadblocks I encountered when working on the project and how that shaped what the project ended up being. The final section is a quick overview of what the project turned out to be.

Original Project Idea

The original concept for my project was a cost effective, fully functioning audio system that responds to voice commands and plays music across several rooms when requested. Using the Raspberry Pi as a controller, it would take commands in through a microphone. From there the command would be processed and the audio (music file playing, volume, rooms being played in, etc.) would be changed based on that command. If time permitted I intended to make the system very adaptable and be able to play music from several sources such as an iPod, a smart phone, and even possibly internet radio like Pandora.

I also had some personal goals such as learning more about the Raspberry Pi and gain some experience working with hardware and the software that will directly interact with that hardware.

Technology and Choices

Raspberry Pi: I used this small computer as the brains of the operation because it is small, very transportable, simple, and inexpensive.

Voice Recognition Software: Ultimately this did not pan out, but I had three programs picked out to try, Jasper, Voice Control by Steven Hickson, and Voice Recognition by Oscar Liang. All of which were built specifically for the Raspberry Pi and had good reviews. See the subsection 'Roadblocks' below to see why this changed.

Python: I chose Python to be my main programming language. I chose this language as I am fairly comfortable with Python, more so after this project, and I knew that learning how to use the Raspberry Pi and controlling hardware on the system would be enough of a challenge.

Raspbian Jessie: This is the operating system I used with the Raspberry Pi. I chose this as it was the newest release of Raspbian and recommended by the Raspberry Pi foundation.

Hardware: Additional hardware I did use were wired and Bluetooth speakers. I had also intended to use a microphone, but that ended up being unnecessary when the voice recognition software did not work. Again, see the 'Roadblocks' section below for more detail.

Tkinter: I used Python's GUI package Tkinter to create the user interface for my program. I decided on this after doing some research as it sounded pretty simple and easy to both learn and to use. This piece of the project was added two months in after the voice recognition software failure, and I wanted to make sure I had enough time to learn it, so I chose a simple one.

PyGame Mixer: PyGame is a Python module used for writing video games, it has a very simple and the Mixer module in PyGame is used to play sounds and music. I chose this as it had very simple methods to load, play, pause, and stop a music file.

Roadblocks

Downloading Software: The initial roadblock was just simply downloading the software and dependencies of the software for the operating system and programs that I needed for my project. I had planned for this to take a couple of days and it took a couple of weeks. Attempting to download the voice recognition software alone was fairly intensive and difficult and every time I thought I had managed to download it correctly I would run the program and errors would occur. Eventually I found all the dependencies that I needed to make the software run without errors.

Voice Recognition Software: After having successfully downloaded a voice recognition software I thought the worst was over and I finally had something I could work with. When I tried it a few times I wasn't able to have the program recognize anything I said. I spent the next two months working on this. I tried all three of the different voice recognition softwares. I attempted to mess with configuration files in the operating system, and broke the operating system a couple of times, had to re-download that. Tried different operating systems.

When none of that worked I thought maybe I would attempt to create my own voice recognition software using a speech-to-text engine. I tried one by Google and one called Sphynx and wasn't able to have either translate a single word into text. During every step I was still attempting to adjust audio configuration files or find the right file to edit, but could not find the right one.

At this point I had a discussion with the professor supervising my project and we both decided it was time for a change in goals. We agreed upon working toward using a GUI instead of voice control for

the project and if I had time to come back to this then I would. Unfortunately, I ran out of time before I was able to come back to this as a part of my project.

Multi-Room Capabilities: After I had a GUI working, I thought I would attempt to re-visit the multi-room problem. I didn't have much time left at this point, but I had at least one idea that I had time to pursue and that was Bluetooth speakers. I successfully paired and ran music through two of my Bluetooth speakers, but was never able to run them both at the same time. When that quandary led me back into the audio configuration files of the Raspberry Pi I knew it would be running me around in circles again for as long as I was willing to let it. So I abandoned that track and decided to optimize my GUI further and make it more user friendly with the limited amount of time I had left.

Bluetooth Speakers: As stated in the previous section I was able to pair multiple speakers with the Pi and I even had my program running music through them, however, the day before the code freeze I plugged in a wired speaker and I couldn't get another Bluetooth one to work with my program again. I was able to have another media player on the Pi play through a Bluetooth speaker while running my program through a wired speaker, but that was as close as I was able to get to making it run through a Bluetooth speaker again before the code freeze.

Final Project Result

After that many changes, the original project is fairly unrecognizable from the final project. Instead of a multi-room voice controlled system, I successfully created a single speaker, GUI controlled, mp3 player. Further detail will be provided in the subsection 'Project Components & Functionality' in the 'Results' section below.

The **Results** section describes what you accomplished and how the final project evolved. It has details of your finished project, its components, functionality, and should be documented with screen shots or photographs (if you used hardware).

Results

This section contains the details of how my project changed and what it is today. The first section will discuss the evolution of the project. The second section will go over the project components and functionality, the final section will be what I have learned and accomplished from doing this project.

Project Evolution: From ATLAS to SAT

As discussed in the Original Project Idea subsection of 'Methods', the original idea for my project was an affordable, multi-room, voice-controlled audio system, or ATLAS (Audio Triggered Labile Audio Device). The reason for the changes in this system are located in the 'Roadblocks' subsection of Methods above. Through those roadblocks, SAT (Simple Audio Thing) was born. SAT is a simple, GUI operated mp3 player created for the Raspberry Pi. For more details on how it works and what functionality it has, see the 'Project Components & Functionality' subsection below.

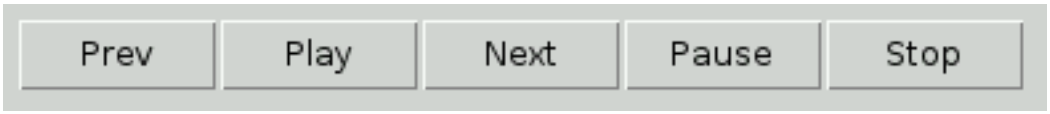
Project Components & Functionality

This section contains screen shots of my project as well as a description on all buttons and features in detail. The first picture below is the main screen. This is the screen that the program loads to and will allow the user to play music files from the music directory.

Layout:

Now Playing: This section exists in the upper left corner of the main window. It will display the artist and song title for the song that is currently being played in the music player. If there is no song playing, the text to the right of 'Artist:' and 'Song:' will simply be blank.

Volume Slider: Located on the right side of the main screen, this slider controls the volume of the song that is currently being played in the music player. The volume starts at 100, slide the button up or down for higher or lower volume respectively. The exact volume number will be displayed to the left of the slider button.



First row of buttons, just below the now playing section.

Previous: This is the button with the text 'Prev'. It will play the last song that has been played if it is available. If no song has yet been played or the Stop button has been pressed, this button will do nothing. This is implemented by using a stack where every time a new song is played, the previous song is pushed onto that stack, if the previous button is pushed, it pops a song from the stack and places the song that was playing into the beginning of the array being used for the Next button.

Play: If nothing is currently playing, this button will select a song at random from the music library to play as well as populate a music queue for the Next button to cycle through. Otherwise it will resume playback of a paused song, or do nothing if a song is already playing.

Next: This button will play the next song lined up in a music queue. When interacting with the Prev button, it will push a song onto the previous stack when that song has finished. Pressing the previous button will place the song that was playing at index 0 of the music queue for the Next button.

Pause: This button pauses playback of the song that is currently playing. If nothing is playing than the button will do nothing.

Stop: This button will do nothing if there is not currently a song playing. Otherwise it will stop music playback and clear both the music queue and the Previous stack.

Second row of buttons, just below the first.

Search: This button will pull up a search window in order to search for a song in the music library. This is detailed later in the section 'Using the Search Window' below.

Randomize: This button will play a random song from the music library at any time during song playback or even before a song has started playing on the music player. It will also populate the music queue if nothing is playing and repopulate it based on the song selected if there was a song playing.

Loop: This button is initialized as on as you can see by its color (black button with white text). When on it will loop through the music library so music is continuously playing. This button does not loop a single song.

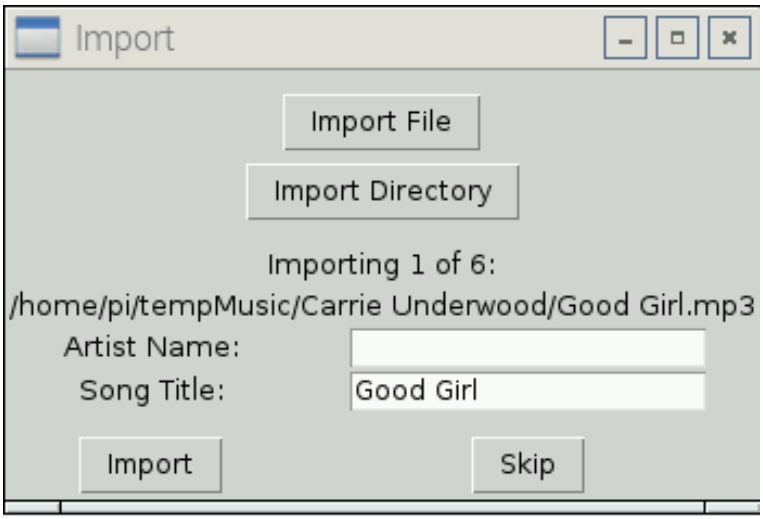
Shuffle: This button shuffles the song in the music queue for songs that will play next. It initializes as on, if turned off, the player will go through the artists in order.

Using the Search Window:

The Search Window appears after the 'Search' button is pressed.

The search button opens a search window where two fields are available for user input. The user can fill out either the artist field, the song field, or both. If both are filled out it will be looking for a specific song by a specific artist. The search window will stay open even after it starts playing a song. Because the search will match a partial artist or song name, if you are unsure of a spelling in the directory, you can type the first few letters and it should pull up in the search results. The search is not case sensitive.

The search window will pull up multiple results as click-able buttons if there is more than one song that matches the search. To play one of those songs, simply click the button with the name of the song you wish to play. If there is only one result, the song will immediately start playing in the music player.



Import Songs Window after a directory has been selected

Importing Songs:

To import a song or directory to the music library, first click File, then Import. This will open the window 'Import' with two button options of 'Import File' and 'Import Directory'. *Caution: This player will only import .mp3 files as that is the only file type it currently plays.*

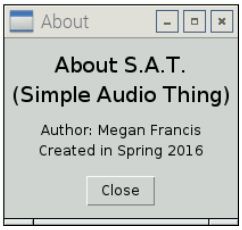
To import a single file, click the 'Import File' button then select the .mp3 file you wish to import. The program will give you two fields to fill out, an Artist Name field and a Song Title field. The Song Title field will automatically populate with the title of the mp3 file, you can change this if it is incorrect. When you are finished editing those fields click import and the program will import the song to the music directory. For details on how these songs are stored in the music directory, check out the section 'The Music Directory' below.

To import a directory/folder of music files, click the 'Import Directory' button and then select the folder you wish to import. Similar to importing a file, the program will give you two fields to fill out: Artist Name and Song Title. The song will auto populate based on the title of the .mp3 file and the artist will populate future songs based on the artist name of the first song you import. In this window it will also tell you how many files it found in the directory, but it will skip any files that are not .mp3 files. You can also skip importing a song by pressing the 'Skip' button in the Import window. When all the songs are imported, an 'Import Complete' message will appear in the window.

The Music Directory:

The default music directory is located in /home/pi/Music. The way this music player stores music in the directory is to have a folder titled with the Artist Name and under that folder all the music files (in .mp3 format) by that artist named with the song title. This allows the player to correctly display the artist name and song title in the Now Playing section of the player.

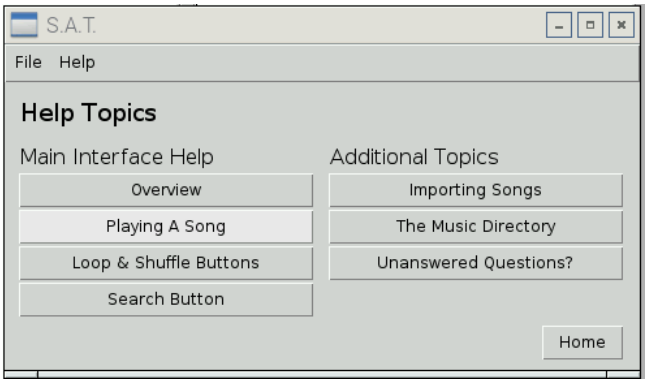
About Window:



This window can be pulled up from the Help Menu under 'About'. It has the author of this program, its title and when it was created.

Help Topics:

Help topics are available in the music player as well. Under the Help Menu, click Help topics and it will load the screen you see below:



From this screen there are a number of available topics covering the main interface and additional topics. Each button pulls up a window with information about the given topic. This design choice was implemented so that a user would be able to directly reference the help information while on any screen in the music player.

This screen does not alter music being played. The player will still play which ever music was playing when this screen was loaded and will continue playing based on what is left in the music queue and the state of the Loop button.

An important subsection in your Results section should be **Lessons Learned**:

- What did you hope to learn in this capstone project?
- What did you learn? Explain in detail.

Lessons learned can be both technical and personal growth (time management, etc.). Provide as much evidence of lessons learned as possible. This section has a large impact on your grade.

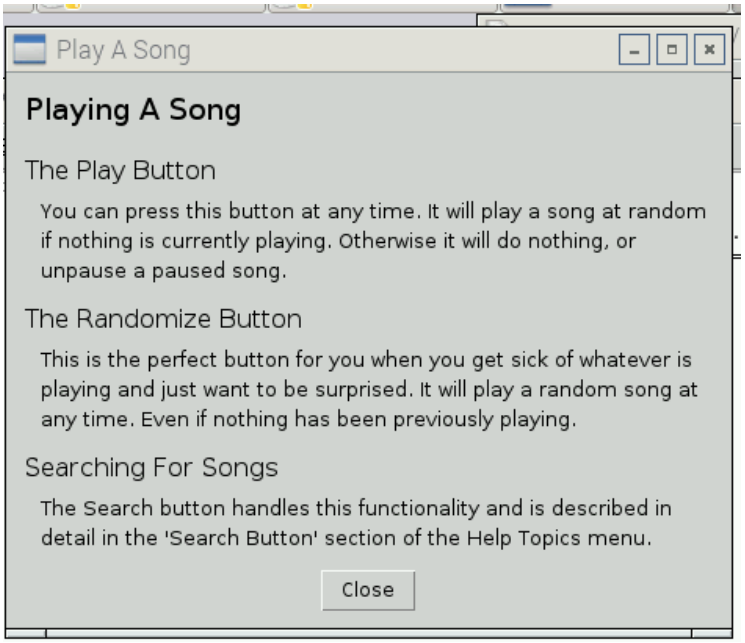
Lessons Learned

When I started this project I was hoping to learn more about physical computing and the Raspberry Pi. I had a curiosity Hoped to learn more about physical computing & the Raspberry Pi.

Learned a lot about programming with a GUI, learned that hardware is difficult, learned that when hitting a roadblock it doesn't help to continually hit your head against it, if you spend a couple weeks and try every angle you can think of, it's time to either get some help or find a new direction. I learned that correct power levels are important when using systems and if you add things that require power to that system, then you're gonna need to give it more power. I learned that it's easier to do these kinds of projects when I'm not the only one I'm responsible to. I learned that the cost of abstraction is adaptability. I learned that it would be a lot nicer if I could do this in a small group. Two sets of eyes and two brains provide different ideas and further reach to find the ideas that you want. Perseverance and flexibility – Mom. Perserverance, determination, problem solving, and dealing with adversity - Josh

The **Conclusions** section describes how the final project compared to the initial ideas, and what you might have changed or would still do if you had more time.

Conclusions



Works Referenced

http://www.instructables.com/id/Raspberry-Pi-Screenshots/step2/Instal-Scrot/ - Screenshots on the Pi

http://www.instructables.com/id/Raspberry-Pi-Multi-Room-Audio-MobileTabletPC-Contr/ - Multi-room audio

https://www.raspberrypi.org/forums/viewtopic.php?t=32427&p=342037 – rasp pi PiPod

http://www.apple.com/itunes/

http://www.videolan.org/vlc/index.html

http://windows.microsoft.com/en-us/windows/windows-media-player

https://www.raspberrypi.org/documentation/usage/xbmc/

http://diyhacking.com/best-voice-recognition-software-for-raspberry-pi/ - Voice Recognition Software

https://en.wikipedia.org/wiki/Pygame - PyGame

http://www.pygame.org/docs/ref/mixer.html - PyGame Mixer

Which brings us to my project and what I did accomplish. I successfully created a functional MP3 player. It has your standard pause, play, stop, next, etc. A volume slider, a shuffle button to randomize the music library, a loop button that doesn't loop a song, but the entire library and a search function to find a song in the music library. I created a way to import songs to the music library, a single file at a time, or an entire directory of songs. I also added a help section with information on usage and occasional tips.

EXTRA INTRO STUFFS:

The original idea for my project was much more an integration between hardware and software and I am still very interested in the physical computing side of things, but would be more interested in pursuing that as a hobby. I am still interested in pursuing the original idea for my project though I believe I would instead like to make my computer the controller for the system. Also, now that I have two Raspberry Pis I would love to pursue additional projects more like what Crystal was pursuing with her 8x8x8 cube and such.

have always been interested in physical computing, so to work a bit with that and have nothing that was physical actually work was a very disappointing, and I really wasn't able to quench my interest in the field so that sucks =/ What do I want to know when I graduate and pursue the future?

As for tying it into the very very broad field of computing: well I was working on a raspberry pi (tiny amazing inexpensive COMPUTER), dealing with things like configuration files and different versions of Linux operating systems. The project was done in a programming language (Python) and attempted to integrate the software controlling hardware with software written by me as well as software written by other people (though that didn't pan out). It also ended up being very much a front-end user application with a little bit of backend and data-basing (made my own music library). OH and I actually did integrate my own code with other software like PyGame and Tkinter (though I don't really know if Tkinter counts, cuz that's kinda like python GUI style…

EXTRA BACKGROUND STUFFS:

After seeing these projects, I worried that I would have to modify my original idea beyond recognition and either use or create an application for my phone or laptop to stream through the audio system. But this, I argued with myself, would be no better than just simply setting up Bluetooth speakers around the house and carrying my phone around to them. Why have the Pi at all? And where would voice control come into the system?

From there, I decided on the idea for the most basic version of the system. Instead of connecting the Pi to some device as its controller, or even as its audio input, why not start with music directly on the Pi? The songs would be easier to sift through as I wouldn't have to worry about the hardware gap between the Pi and an iPod and voice control would make more sense as I wouldn't be carrying around the device that controls the system.

What ended up happening?

iTunes with a randomize button and a lot less features on a not as well supported operating system. Or VLC or Windows Media Player or…there are lots of projects in this area that are much more successful and much more wonderful than mine, so it kinda sucks =/

EXTRA METHODS STUFFS:

original project idea and what my project ended up being are probably comparable to the north and south poles. The two are still on the same planet (CS), both have similar climates (audio systems), but that's about as close as they get.

What ended up happening:
I worked with the raspberry pi, PyGame, Tkinter, Rasbian, Bluetooth speakers and programming, speech recognition software, speech-to-text engines, Linux command line and wired speakers, SD card operating systems, wired audio input device, etc.
Significant changes: completely gave up on the voice control after working on it for two months, ended up adding a GUI which was not part of the plan at all, ended up giving up on the multi room problem when I started running out of time and the GUI took longer than I thought it should and the Bluetooth speakers were fucking bitches and >.< nothing worked the way it should so what my project idea was and what my project ended up being feel nearly as close as the north and south poles, still on the same planet (CS), both similar climates (audio systems), but that's about as close as they get.