Sisyphus Senior Design Team: Final Report

Members:
Robert Burkhardt, Joseph Casper, Stuart Enters, Grace Fleming, & Andrew Wojciechowski

Advisor:
Dr. Chris Taylor

Partner:
Sisyphus Industries

Year:
2020-2021

Executive Summary

The sisyphus table is a kinetic art table that uses a magnet on a robotic arm beneath the table's surface to manipulate a metal ball atop the table's surface to trace intricate patterns in the sand, creating spirograph designs that take shape over time. A strip of LEDs runs around the circumference of the table. The lights' purpose is to illuminate the sand to sharpen the shadows created by ridges in the sand which adds depth to the spirograph designs, and add color to an otherwise monochrome device. The Sisyphus table is controlled through a mobile app. This mobile app allows for selection of which design the ball is currently tracing in the sand as well as control of the speed of the ball. Additionally, the app provides the user with the ability to control the color of the LED lights, and gives the user several patterns for the lights to select from.

The purpose of the MSOE senior design team's work over the past year was to expand upon the table's overall ambience. The team spent considerable time exploring potential methods of doing this, from developing new lighting patterns to developing a new track for the ball to trace, to constructing a mobile application to change the lights of the table when a user received a phone notification. Eventually, the team decided to implement a method of "mood lighting" by placing a microphone on the table, listening to audio, classifying the 'mood' of the audio, mapping this mood to a corresponding color, and displaying this color on the sisyphus table. An additional set of controls was added to the app to account for enabling/disabling this feature, and options for users to change which colors were associated with which mood were also implemented.

Project Summary

In order to allow the Sisyphus table to reflect a color based on ambient sound, a few critical components needed to be put in place. First and foremost, the table needed to be equipped with a way to record audio, which was accomplished by adding an USB microphone onto the existing hardware via use of a USB hub. Once the table could “hear”, a submodule was developed on the table that, when the mood lighting feature was enabled, would record 5 second audio segments to analyze for mood. Due to computational constraints of the table, this could not be done on the table itself, and so a cloud hosted server was developed to do the computation. The server then sent back a set of coordinates, which the table mapped to color.

The existing software architecture already used multiple modules to accomplish the different parts of the work needed, so spawning a new one was not difficult. Once the submodule, named “sis-listen” in keeping with existing naming conventions, was spawned, it would go dormant. The user interface of the Sisyphus mobile app was changed so the user could enable mood lighting from their settings. At that point, the dormant sislisten module would then awaken and begin recording audio. Sislisten managed broadcasting audio data to the server, receiving coordinantes, mapping these coordinates to colors, then forwarding them to existing endpoints in another submodule to change the lights themselves.

The server that converted audio to emotion was based upon an existing project to classify music by mood (link | code). This machine learning model analyzed the piece based upon the values for valence and arousal, which can be used to relatively determine the emotion of a piece. Instead of fixing this to a set value, a coordinate on a color wheel was returned, so

the sislisten client could use user defined settings to get the color. This allowed users to define what color they associate with specific emotions, personalizing the table to their expectations.

Plan Summary

1. Plan summary, including:
    - A list of completed PBI's (separated by sprint) — include a brief summary if the PBI title is not enough to clearly convey what was accomplished
    - Total time spent for each team member

Sprint 1

Sprint Goal: Finish off any zero-day items for setup and develop simple yet creative light sequences to demonstrate setup completion.

Completed PBIs

- #11 - Setup linking for Python and JS
    - Summary: The existing Sisyphus projects were written in Python and JS and we wanted to setup linting for that code so that we can more easily catch errors as we made changes to the codebase as well as to keep the code in a clean maintainable state
- #22 - Spike: Setting up automated testing and CI
    - Summary: The existing Sisyphus codebase had no automated tests in it. We wanted to set up the ability to write automated tests for any areas that we changed and for any new code that we wrote in the Sisyphus codebase.
- #12 - 3-Hand Clock Light Pattern
    - Summary: This PBI involved creating a light pattern on the table where it looked like the lights were a 3-Hand Clock.
- #24 - Spinning / Bouncing Lights
    - Summary: This PBI involved creating a light pattern where the lights on the table spinned / bounced around the table.
- #16 - Change Lights based on time
    - Summary: This PBI involved creating a light pattern where the lights on the table changed based upon what time of day it was.

Time Spent:

Andy	31.83h
George	26.08h
Grace	28.67h
Joe	22.50h

Stuart	48.33h

Sprint 2

Sprint Goal: Continue exploring different light patterns and begin investigating additional inputs to affect lighting schemes.

Completed PBIs:

- #28 - Tech Debt: Fix eslint warnings
    - Summary: This PBI continued on the setup done in the first sprint. Since we had the linters setup in the first sprint we wanted to go through and cleanup the existing code to make it more easily maintainable.
- #23 - Create Lateral Erase Track
    - Summary: When transitioning between tracks an erase track plays to erase the previous pattern on the table. The table already had one which erased the pattern in circles. We wanted to create another erase track to erase the pattern using horizontal lines instead of circles.
- #17 - Lights based on Weather
    - Summary: In this PBI we created a new LED light pattern for the table which contacted a weather API and made the table's lights respond based on the current weather.
- #20 - Spike: Sync table with phone notifications
    - Summary: In this PBI we investigated to see if we can have the table's lights flash when we receive a notification to our phones.

Time Spent:

Andy	33.08h
George	28.25h
Grace	28.75h
Joe	28.08h
Stuart	25.50h

Sprint 3

Sprint Goal: Finish off carryover from sprint 2, and explore methods of input for lighting dependent on the physical environment.

Completed PBIs:

- #25 - Have all members set up the Sisyphus project on a Raspberry PI
    - Summary: Since we only had one production table to test with we had all team members set up the Sisyphus table firmware to run on a local Raspberry PI to make development easier. We all had Raspberry PIs that we used in another class so we bought new SD cards for our PIs and a light strip that we can use for working with the table's lighting code.
- #13 - Change Lights based on Holidays/Calendar Events
    - Summary: In this PBI we created a new LED light pattern for the table which made the lights respond based on if the current day was a holiday or not.
- #15 - Knowledge Acquisition: Sync Table Lights with Audio Track
    - Summary: In this PBI we investigated to see if we can make the table's lights respond to audio. This is the PBI where we found the initial off the shelf machine learning algorithm to translate audio to moods and served as the basis for our mood lighting feature which we worked on for the rest of the year.
- #30 - Light Mode: Magnetic
    - Summary: In this PBI we created a new LED light pattern for the table where the lights got brighter depending on the edge of the table that the ball was the closest to.

Time Spent:

Andy	31.83h
George	19.25h
Grace	39.33h
Joe	22.50h
Stuart	16.33h

Sprint 4

Sprint Goal: Investigate mapping audio to colors and how to integrate a Python ML model with the existing code base.

Completed PBIs:

- #40 - Spike: Mood Lighting & Performance Implications
    - Summary: When we first found the off the shelf machine learning algorithm we ran it on a Raspberry PI and found the CPU usage to be at 100%. We wanted to

investigate the performance implications of this algorithm and eventually decided that streaming the audio to a server would be the best option performance wise.

- #44 - Spike: Mood Lighting & Audio Interfacing
    - Summary: In this PBI we investigated the hardware changes that we needed to make to the table in order to have the table listen for audio.
- #41 - Spike: Mood Lighting & Integration
    - Summary: In this PBI we investigated how to add a toggle to the Sisyphus app to enable this mood lighting feature.

Time Spent:

Andy	21.58h
George	24.50h
Grace	28.25h
Joe	28.75h
Stuart	30h

Sprint 5

Sprint Goal: Determine where the modified (volume-sensitive) algorithm for translating audio to color will run.

Completed PBIs:

- #45 - Spike: Mood lighting & Mobile as a Artificial Intelligence Computation Platform
    - Summary: In this PBI we investigated if we could run the machine learning algorithm on a mobile phone. We determined that this was not feasible with the remaining sprints that we had left.
- #46 - Create First-Pass Architecture Draft
    - Summary: In this PBI we wanted to create an architecture diagram of the Sisyphus software system showing how our new systems interact with each other and how they interact with the existing systems.
- #48 - Add Volume Feature to Algorithm
    - Summary: In this PBI we wanted to add the ability for the machine learning algorithm to respond to how loud the audio was and to change the brightness on the table accordingly.

Time Spent:

Andy	25.33h

George	16.92h
Grace	35.83h
Joe	26.25h
Stuart	23.50h

Sprint 6

Sprint Goal: A user will be able to enable mood lighting and see its effects on the lab Sisyphus table.

Completed PBIs:

- #49 Spike: Mood Lighting: Stream Audio to AI Service
  - Summary: Attempted to create a prototype server that would receive an audio stream, process said stream, and return color/coordinate results.
- #41 Spike: Mood Lighting & Integration
  - Summary: Attempted to integrate the full solution, client and server, which is enabled by the web app setting
- #55 Document Final Architecture
  - Summary: Organize a formal first-draft architecture to serve as a communication basis for the rest of the team and as an artifact for the project
- #56 Handle Sisyhpus Table Audio Streaming Request
  - Summary: Allow the table to record audio and package it into an HTTP request that could be accepted by the server
- #57 Handle Sisyphus Table Audio Streaming Response
  - Summary: Allow the table to receive a response from the AI service and modify the table state as a result (i.e. update the current LED color)
- #58 Mood Lighting Audio Processing - Smooth Transition from ML to Table
  - Summary: Attempt to make the color output be more consistent while a song is playing by both modifying the output of the AI service model and enabling smooth transitions on the table
- #61 Create a user-reachable toggle to enable new table functionality
  - Summary: Attempt to spike out a solution for modifying the Sisyphus web app and presenting a new setting to the user

Time Spent:

Andy	20.50h
George	23.50h

Grace	25.75h
Joe	22h
Stuart	17.50h

Sprint 7

Sprint Goal: Expand functional proof-of-concept mood lighting to production-ready prototype.

Completed PBIs:

- #63 Enable End-To-End Mood Lighting on Production Platform
    - Summary: Set up a full implementation of the architecture and have the table send audio to the service and have the service respond with colors which the table uses to then update the current LED color.

Time Spent:

Andy	11h
George	11.50h
Grace	17.75h
Joe	15.50h
Stuart	12h (Est.)

Sprint 8

Sprint Goal: Support user settings and finish carried over PBIs from last sprint

Completed PBIs:

- #65 Make AI Service Return Coordinates
    - Summary: In order to allow for user settings, have the AI service return raw coordinates rather than a full color.
- #62 Look into deploying the AI service on a optimal deployment platform
    - Summary: After running into issue with current platform Heroku - investigated AWS, Google Cloud, and other cloud service providers to see which would return results with good latency, low cost, good security, and maximum uptime.

Time Spent:

Andy	21.50h
George	11.75h
Grace	18h (Est.)
Joe	12h (Est.)
Stuart	12h (Est.)

Sprint 9

Sprint Goal: Finalize mood lighting color settings and smoothing and enable beat detection.

Completed PBIs:

- #64 Add User Settings to Sislisten
    - Summary: Allow sislisten to accept user mood-color mappings and translate coordinate from the AI service into an RGBW which will be set as the new primary color on the Sisyphus table
- #47 Color Axis Editor in Mobile App
    - Summary: Add the ability for users to set the color of each mood and send those mappings to sislisten on the table. Majority of work was completed in other sprints but was not able to merge until this sprint due to platform issues.
- #66 Handle User Settings Feedback
    - Summary: Test that users settings are utilized when mapping music to colors with a series of manual tests as well as use previously acquired knowledge about color smoothing to produce more consistent results.

Time Spent:

Andy	13.08h
George	10.50h
Grace	20h (Est.)
Joe	12h (Est.)

Stuart	12h (Est.)

Demonstration of Engineering Principles

Requirements

Functional Requirements

F1: The LED strip shall turn red (as a default) when the audio corresponds to anger.

- Description: When mood lighting is enabled for the first time by a user we should provide an initial configuration of the feature so that users can get an idea of how the feature works.
- Rationale:Red is a color that people commonly correlate to anger so red correlating to anger will make a good default to the Mood Lighting feature.
- Originator: Andrew Wojciechowski
- Fit Criterion: When mood lighting is enabled for the first time via the Sisyphus mobile app one quadrant of the grid shall show as red with the emotion anger.
- Customer Satisfaction: 7
- Customer Dissatisfaction: 5
- Priority: Medium
- Dependencies: None
- History: Last modified 2/8/2021

F2: The LED strip shall turn blue (as a default) when the audio corresponds to sadness.

- Description: When mood lighting is enabled for the first time by a user we should provide an initial configuration of the feature so that users can get an idea of how the feature works.
- Rationale:Blue is a color that people commonly correlate to sadness so blue correlating to sadness will make a good default to the Mood Lighting feature.
- Originator: Andrew Wojciechowski
- Fit Criterion: When mood lighting is enabled for the first time via the Sisyphus mobile app one quadrant of the grid shall show as blue with the emotion sadness.
- Customer Satisfaction: 7
- Customer Dissatisfaction: 5
- Priority: Medium
- Dependencies: None
- History: Last modified 2/8/2021

F3: The LED strip shall turn green (as a default) when the audio corresponds to calmness.

- Description: When mood lighting is enabled for the first time by a user we should provide an initial configuration of the feature so that users can get an idea of how the feature works.
- Rationale:Green is a color that people commonly correlate to sadness so green correlating to sadness will make a good default to the Mood Lighting feature.
- Originator: Andrew Wojciechowski
- Fit Criterion: When mood lighting is enabled for the first time via the Sisyphus mobile app one quadrant of the grid shall show as green with the emotion calmness.
- Customer Satisfaction: 7
- Customer Dissatisfaction: 5
- Priority: Medium
- Dependencies: None
- History: Last modified 2/8/2021

F4: The LED strip shall turn yellow (as a default) when the audio corresponds to happiness.

- Description: When mood lighting is enabled for the first time by a user we should provide an initial configuration of the feature so that users can get an idea of how the feature works.
- Rationale:Yellow is a color that people commonly correlate to happiness so yellow correlating to happiness will make a good default to the Mood Lighting feature.
- Originator: Andrew Wojciechowski
- Fit Criterion: When mood lighting is enabled for the first time via the Sisyphus mobile app one quadrant of the grid shall show as yellow with the emotion happiness.
- Customer Satisfaction: 7
- Customer Dissatisfaction: 5
- Priority: Medium
- Dependencies: None
- History: Last modified 2/8/2021

F5: The mood lighting mode shall be enabled via the Sisyphus mobile app.

- Description: An option needs to be available for the user to enable the mood lighting feature.
- Rationale:All options for the Sisyphus table are controlled via the Sisyphus mobile app so we should put the option to enable mood lighting where all of the other options for the table are. Also, since there are other light patterns users shall be able to enable/disable the feature on demand in order to select other light patterns.
- Originator: Andrew Wojciechowski
- Fit Criterion: When a user navigates to the settings page in the Sisyphus mobile app an option to enable mood lighting shall be available to the user as a checkbox.
- Customer Satisfaction: 8
- Customer Dissatisfaction: 3
- Priority: High
- Dependencies: None

- History: Last modified 2/8/2021

F6: The table shall collect audio samples when the mood lighting mode is enabled.

- Description: When mood lighting is enabled samples shall be collected in order to determine what color the LEDs on the table should be turned to.
- Rationale: The only option that requires audio input is mood lighting so audio samples should only be collected when mood lighting is enabled.
- Originator: Andrew Wojciechowski
- Fit Criterion: When mood lighting is enabled the table shall collect 5 second samples to use for the mood lighting feature.
- Customer Satisfaction: 8
- Customer Dissatisfaction: 10
- Priority: High
- Dependencies: None
- History: Last modified 2/8/2021

F7: The color shall be changed through already existing externally facing APIs

- Description: The table has APIs for setting the colors already and the mood lighting feature should take advantage of those existing APIs.
- Rationale:It will be easier to use the existing APIs to change the color instead of writing our own APIs.
- Originator: Andrew Wojciechowski
- Fit Criterion: When the mood lighting algorithm has calculated a color to set on the table. The feature shall use the existing light APIs on the table to set the primary color of the lights on the table.
- Customer Satisfaction: 3
- Customer Dissatisfaction: 3
- Priority: Low
- Dependencies: None
- History: Last modified 2/8/2021

F8: The user shall be able to select custom colors that map to different moods

- Description: Customization shall be available in order for users to change which colors coordinate to which moods.
- Rationale: People may correlate moods to color differently so a feature shall be provided in order to select custom color to correlate to different moods.
- Originator: Andrew Wojciechowski
- Fit Criterion: When mood lighting is enabled in the Sisyphus mobile app the color axises shall be displayed and the colors of each of the quadrants shall be customizable in the Sisyphus mobile app.
- Customer Satisfaction: 8
- Customer Dissatisfaction: 5

- Priority: High
- Dependencies: None
- History: Last modified 2/8/2021

F8: The user shall be able to enable and observe beat detection

- Description: The user can enable beat detection via the app and the lights will respond accordingly
- Rationale: People may want the table to respond to ambient audio in different ways according to more concrete metrics (beats per minute vs. mood) and this mode does not affect color but rather brightness
- Originator: R. George Burkhardt
- Fit Criterion: When beat detection is enabled in the Sisyphus mobile app the lights shall brighten and dim at the same rate as the beats per minute (BPM) of the audio being played.
- Customer Satisfaction: 8
- Customer Dissatisfaction: 2
- Priority: Low
- Dependencies: None
- History: Last modified 5/13/2021
- **NOT MET**

Nonfunctional Requirements

NF1: The audio system shall be installable within 20 minutes, by a user who has minimal experience with technology and should require similar technical skills to that necessary for installing the Sisyphus LED Kit.

- Description: The modification necessary to add audio recording and processing to the table should be minimal so that Sisyphus industries can market this addition as an upgrade kit to their users
- Rationale: Keeping install time below 20 minutes means that the user can use the new functionality quickly and does not require more of their time then necessary (since we assume most Sisyphus table owners have minimal technical experience)
- Originator: R. George Burkhardt
- Fit Criterion: The installation of the audio module (i.e. microphone, Pi processor) should take no more than 20 minutes. The upgrade kit shall be of a similar technical level to the LED install kit.
- Customer Satisfaction: 8
- Customer Dissatisfaction: 10
- Priority: High
- Dependencies: N/A
- History: Last Modified 2/9/2021

NF2: The system shall alert the user that any and all audio recordings are not saved.

- Description: A disclaimer should be available to the user saying than no audio will be recorded and stored on external servers if any web services are used
- Rationale: In keeping with ethical software practice, the system should ensure the user is aware of how the audio is used.
- Originator: R. George Burkhardt
- Fit Criterion: A disclaimer is immediately available to the user upon installing the audio upgrade kit
- Customer Satisfaction: 7
- Customer Dissatisfaction: 3
- Priority: High
- Dependencies: F5
- History: Last Modified 2/9/2021
- **NOT MET**

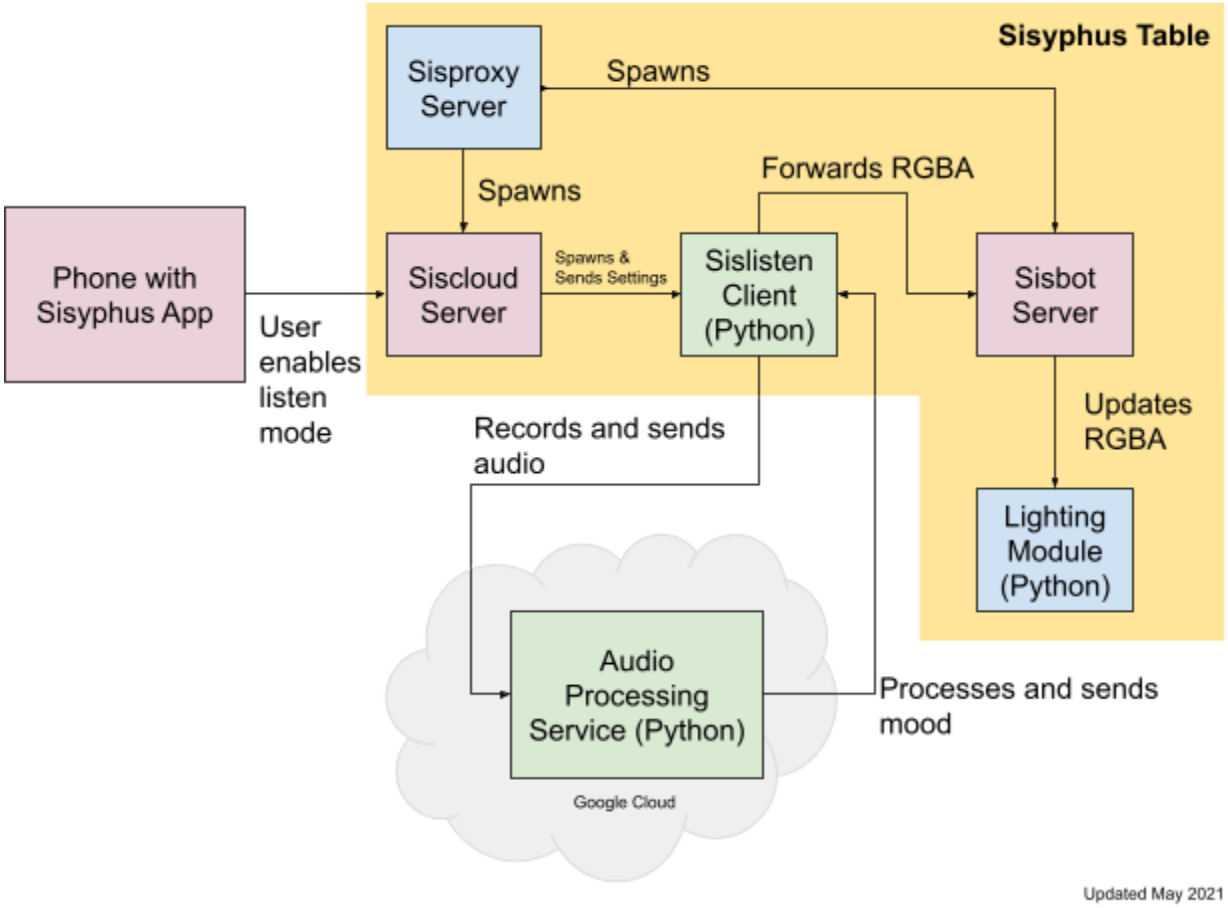
NF3: Any new additions to the UI will abide by the current UX design patterns and standards
- Description: Any new UI components (e.g. dropdowns, icons, etc.) should be styled in similar ways and any interactions should match how the app currently operates (e.g. checkboxes for toggling options)
- Rationale: The app should provide a cohesive user experience between old features and new features. Changes in styles and patterns can be jarring to users, thus it should be avoided or kept to a minimum
- Originator: R. George Burkhardt
- Fit Criterion: New UI elements will be styled with the same colors, fonts, and icon packs that are currently in the app. New UI elements will be kept to a minimum.
- Customer Satisfaction: 5
- Customer Dissatisfaction: 5
- Priority: Medium
- Dependencies: F6, F8
- History: Last Modified 2/9/2021

NF4: The audio feature shall not negatively impact any existing functionality
- Description: Any changes made should be done in such a way that they do not regress the system
- Rationale: Any changes will need to exist side by side tables that do not have audio/lighting functionality thus any changes must be atomic as to not ruin the product experience for other users
- Originator: R. George Burkhardt
- Fit Criterion: The table will be able to function normally with or without audio input
- Customer Satisfaction: 8
- Customer Dissatisfaction: 10
- Priority: High
- Dependencies: N/A
- History: Last Modified 2/9/2021

The project did not succeed in meeting all requirements, however, a large headway was made in attempting to complete those requirements. If this team (or any other) were to continue the project - those requirements would be top priority and the wealth of information and experience generated by this team in their attempt would serve well as a project basis.

Software Architecture and Modeling

The team added a new module to the project called sislisten, which holds the code that runs the mood lighting features. The sislisten client records and sends audio to an external mood lighting service. This service returns the color coordinate. Sislisten then places that coordinate on a color map, then sends the corresponding color to the led lights on the table. Sislisten also accepts custom user settings for personal color mappings. This allows users to map moods to colors as they see fit.



The system was designed with modularity in mind. In other words, each component of it will be easy to interchange with a different component. If a better mood lighting algorithm is found, the current one can be replaced. Also, one could put in a different algorithm that has nothing to do with music or mood lighting, and the system would work with very few modifications. The mood lighting server is easy to run on different hosts. Currently, it is hosted

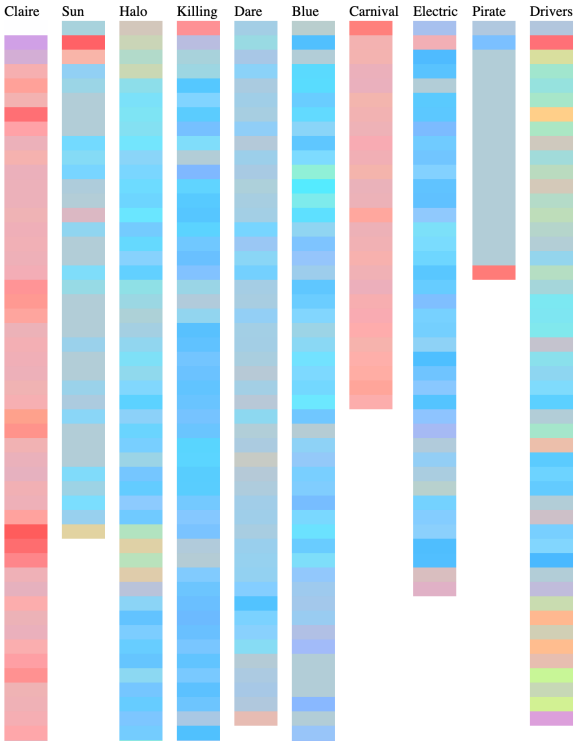
on Google Cloud, but it could very easily be moved to run on a Raspberry Pi, computer, or different webhost.

In addition to being modular, it was designed to interfere with existing code as little as possible, in order to not compromise existing functionality. Siscloud was only modified to have additional buttons to have the user turn on the mood lighting mode, and sisproxy was only modified to spin up sislisten with the rest of the sisbot code. This allows sislisten to be added or removed easily.

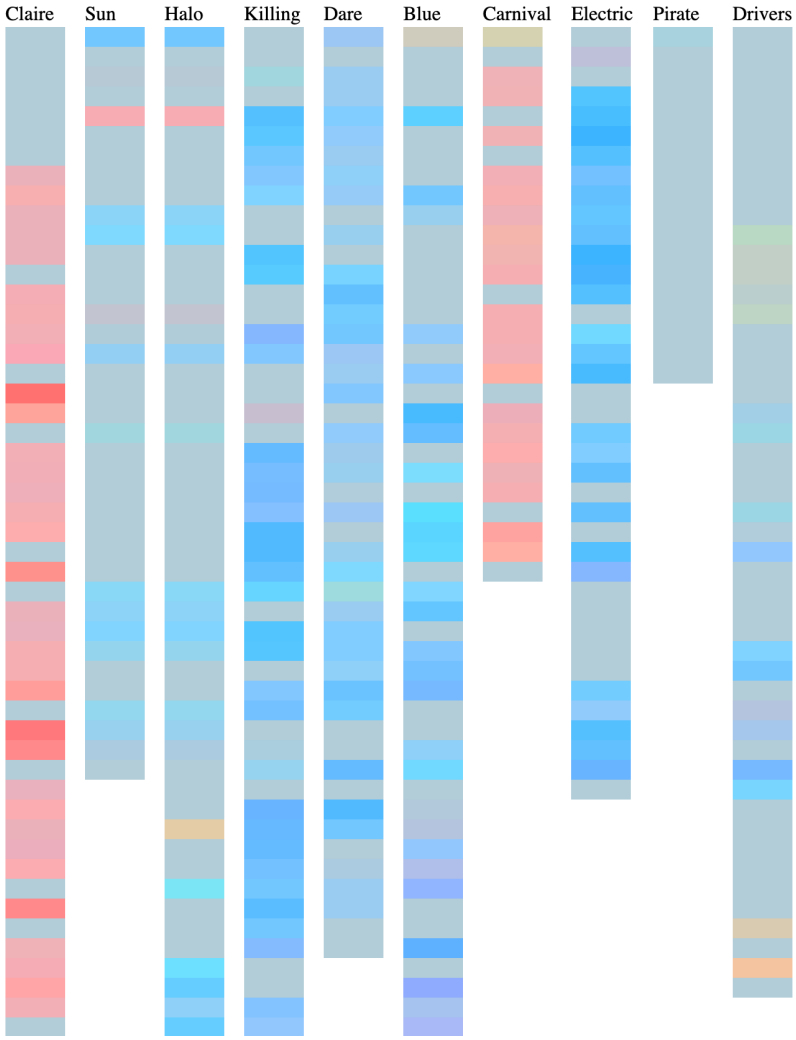
Testing

Some unit tests were written for the sislisten module. One test verifies that a valid response is returned when the server is called. Another test verifies that the server returns a valid color when it is given mood lighting data. These tests were initially written on development machines, then moved onto the Pi to make sure that it ran correctly on the target hardware. After verifying that these tests worked, they were added to a CI pipeline on GitLab that ran each time someone made a merge request. Lastly, An integration test was written for the mood lighting server to test that all of the components worked together without throwing any errors. All in all, it would have been good to have more tests for the system, but the team was very rushed to finish the project.

There were some manual tests done to check how consistent the colors the server returned were. One test was done on ten different songs before a smoothing algorithm was added, and after the smoothing algorithm was added to see if the new code made the colors more consistent.



Test before smoothing



Test after smoothing

This test shows that with the added smoothing algorithm, the colors returned by the table were much more consistent.

Tools

Aligning with industry standards, our team made use of tools to aid our development process. These included:

- **Gitlab :** Served as a source and version control server, stored the wiki, ran continuous integration pipelines, and kept track of issues for the team.
- **Docker :** Tool used by gitlab-ci to provide a consistent, lightweight testing environment for running tests.
- **PyTest :** Served as the testing framework for *sislisten*.
- **Google Cloud :** The cloud deployment platform for the APS (Audio Processing Service).
- **Putty & MobaXTerm:** SSH clients for Windows used to SSH into the production table and development raspberry pi's

For further details on these tools, see the Tools section of the Team Components Report.

Third Party Components

Third-Party components are designated as the parts of our final product that originate from other sources. In brief, third-party components of a software nature include:

- **PyAudioAnalysis** : A library developed by a researcher at another university to extract audio features from a given audio clip. The ASP (Audio Processing Service) utilizes this library to extract features from recorded audio.
- **PyAudio :** A well-supported Python library for recording audio on Linux. Plays an important role in the sislisten repository.
- **Flask :** A server-wide web framework that powers both the ASP and sislisten.
- **Numpy / Scipy :** Key 3rd-party, official Python packages that handle machine learning and linear algebra. These are used in sislisten and the ASP o process audio and determine mood, respectively.

Those components that involve hardware consist of the following.

- **USB Hub** : A usb hub is required to expand the USB capabilities of the Sisyphus table, which currently runs off a raspberry pi model 3A, which only has 1 USB port.
- **USB Microphone** : A USB microphone is required to collect audio samples of music playing.

For more information, see the Third Party Components and Tools section of the Team Components Report.

Experimentation and Prototyping

There were many aspects of our goal that concerned us in terms of technical feasibility. Spikes and prototypes were used to confirm or combat these concerns. Below is a list of these major spikes.

- **Choosing an Audio Setup :** This audio spike was used to determine the necessary configuration for adding recording capabilities to the table.
- **Exploring ML on the Pi** : This spike led to the conclusion that running the full recording and machine learning process on a raspberry pi was infeasible, and eventually led to an architecture decision to split out the process into an on-table recording process and a remote web server for audio processing (APS).
- **Exploring ML on the Phone** : This spike led to the conclusion that running a machine learning algorithm natively on the phone was infeasible, and furthered our decision to split out the audio recording and processing pipeline.
- **Exploring ML as a Server :** This spike confirmed that running audio processing remotely was feasible.
- **Mood Lighting Algorithms :** This spike looked to explore other audio algorithms to decipher a song's mood. Our team struggled to find a better algorithm than the one we were using currently; the algorithm did not change in the end.

- **Upgrading the table's Platform OS :** The platform for the Raspberry pi 3 model A required updating in order to utilize the latest version of python 3.7, since python 3.7 was required in order for the scipy / numpy modules (see 3rd party components summary).
- **Selecting a Cloud Deployment Platform :** After deciding to break out the audio processing from the audio recording and run an APS in the cloud, the team needed to find a suitable cloud provider. We explored many different options but ended up selecting google cloud for its built-in security and relatively free costs.

For more information, see the Experimentation and Prototyping section of the Team Components Report.

Documentation

Our team, in keeping with typical agile processes, made sure to keep an organized record of all gitlab issues, sprint retrospectives, a definition of done, and pull requests.

Representative Merge Requests :

- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-sisbot/-/merge_requests/4
    - This merge request was an early request that had a new light pattern to change lights on the table based on time (NOT part of the final mood lighting product, but representative of team effort in merge requests).
- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-siscloud/-/merge_requests/5
    - This merge request allowed for user settings in the mobile application to control and customize mood lighting.
- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-sislisten/-/merge_requests/2
    - This merge request allowed the msoe-sislisten codebase to handle receiving a mood from the mood lighting server, and communicate this color to the table. This was later changed so that sislisten would process coordinates for a mood instead of a color directly, to support user settings.
- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-sislisten/-/merge_requests/7/diffs
    - This merge request allowed the msoe-sislisten codebase to handle receiving coordinates from the mood lighting server.
- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-sislisten/-/merge_requests/10
    - This merge request changed how the msoe-sislisten codebase handled mood coordinates, unifying the colors produced for a given mood.
- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-sisproxy/-/merge_requests/3
    - This merge request allowed sisproxy to start up msoe-sislisten (although it doesn't listen until a specific endpoint is hit) on startup.

Important Gitlab Issues :

- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-sisbot/-/issues/9
    - Setting up local development environments
- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-sisbot/-/issues/10

    - Investigating and making the lab topology accessible from beyond MSOE's campus
- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-sisbot/-/issues/22
    - Setting up CI for our software development process
- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-sisbot/-/issues/46
    - Creating the new architecture to support our goal
- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-sisbot/-/issues/62
    - Cloud deployment spike
- https://gitlab.com/msoe.edu/sdl/sd21/sisyphus/msoe-sisbot/-/issues/61
    - Adding controls for our new feature to the Sisyphus app.

Key Sprint retrospectives:

- https://gitlab.com/groups/msoe.edu/sdl/sd21/sisyphus/-/wikis/Retros/Sprint-2
    - One of our early retros, we helped establish automated testing guidelines and expectations for submitting merge requests. We also highlighted the importance of standardized remote development.
- https://gitlab.com/groups/msoe.edu/sdl/sd21/sisyphus/-/wikis/Retros/Sprint-5
    - This retrospective was probably one of the best our team had. We formally assigned a team member for managing things such as team bonding, weekly report documents and ensuring all members of the project were getting appropriate exposure to the different areas of the project.

PBI completion criteria:

- **Definition of Done:** https://gitlab.com/groups/msoe.edu/sdl/sd21/sisyphus/-/wikis/Process/Definition-of-Done
    - Note that for our definition of done, the "Product Owner" was typically considered to be the entire team by proxy, as the sprint reviews were required to be held after the sprint retrospectives.

Meeting Minutes:

Meeting minutes in the traditional sense were not found to be highly useful from the perspective of the team; rather, meetings on a weekly basis were documented by weekly status reports for the first two terms of the year.

- https://gitlab.com/groups/msoe.edu/sdl/sd21/sisyphus/-/wikis/Reports/F-W6-Status-Report
    - This weekly status report, from week 6 during the fall term, documents completion of our automated testing setup, newly-created 3 light patterns and fully-functional remote environment, complete with live stream via Twitch.
- https://gitlab.com/groups/msoe.edu/sdl/sd21/sisyphus/-/wikis/Reports/S-W1-Status-Report

    - This weekly status report, from during the winter term, documents our progress towards finding an initial architecture.
- https://gitlab.com/groups/msoe.edu/sdl/sd21/sisyphus/-/wikis/Reports/W-W9-Status-Report
    - This was our final report, which documents our team deciding on a “final” architecture.

Project Presentations:

- https://msoe365-my.sharepoint.com/:v:/g/personal/enterss_msoe_edu/EQ0BcT5IbA5EiWIavX2HTeMBf4PKznRGfCPCkDekUhH4kQ
    - This is a link to our final presentation, in a shortened form appropriate for a nontechnical audience.
- https://msoe365-my.sharepoint.com/:v:/g/personal/burkhardtr_msoe_edu/ERXSr2nZhMJIgXIvf1oblm8BOesyhSc4xeCRatpA3xV3JQ
    - This is a link to our final presentation, in a longer and more technical form.

Developer Materials :

- https://gitlab.com/groups/msoe.edu/sdl/sd21/sisyphus/-/wikis/Architecture/Final-Pass-Architecture-Draft
    - A final architecture draft to help describe our current codebase as it relates to the existing product.
- https://gitlab.com/groups/msoe.edu/sdl/sd21/sisyphus/-/wikis/Local-Setup-Instructions
    - Walks a developer through setting up an environment to run the SIsyphus codebase on a brand-new raspberry pi.
- https://drive.google.com/file/d/1VZGxCC5UIQvTIJFN0PRCLqooJrSmV0H0/view?usp=sharing
    - This is a link to the final image for the SD card.

Project Post-Mortem

The Sisyphus Table Mood Lighting project makes extensive use of knowledge and skills acquired in earlier course work, and employs realistic constraints that include considerations such as economic, environmental, sustainability, manufacturability, ethical, health, safety, social and political considerations.

Most evidently, the Sisyphus Table Mood Lighting Project employs principles taught in architecture classes surrounding how to construct large and complex software systems in a maintainable, testable, and performant manner. Specifically, the team chose to implement multi-threading, periodic tasks, a microservice and server-client architecture in order to achieve adaptability, extensibility, maintainability and testability. Furthermore, the team utilized techniques learned in verification classes in order to adequately test the microservice created (“sislisten”) in order to prevent against availability and security issues like command injection and DOS attacks. Tools taught in software process courses were used effectively, such as

source/version control access controls to restrict additions to the codebase unless new code could pass certain checks (pipelines and gated merges). Finally, networking knowledge derived from courses such as network protocols was used to create a remote development environment that remained accessible and useful by the student team during the midst of the covid-19 pandemic.

Economic, environment, sustainability, manufacturability, ethical, health, safety, social and political considerations of the project were brainstormed by the team as part of SO-2.3. Moreover, the team implemented controls in the mobile app part of the project specifically to address the ethical implications of the product being used by those with colorblindness so as to make the project interesting and fun for individuals of all abled-ness. Hardware additions to the table including a USB microphone and USB hub were chosen to be generic and use common protocols to communicate with the existing hardware so as to be easily procured, maintained and replaced as cheaply as possible by Sisyphus industries and their customers.

Compared to the original plan when taking on this project, the team goal has shifted somewhat. The shift is largely due to the ambiguity of the project; at first when the project was adopted by the team, there was no clear vision as to what the goal of the project ought to be. It was only after several months of exploring the table's capabilities that the team decided to add another layer of interactivity to the table by adding the capability to process audio and show the results of such processing using the LED's. After deciding on this goal, however, the team made steady progress to this end and did not have any significant redesigns.

Risks encountered while working on the Sisyphus table were constantly taken into account. For the first two quarters, the weekly status reports published by the team required an update on the risks involved with the project. These consistent reports were used by the team to combat risk quickly; for each time a risk was brought to the team's attention, a mitigation was identified ("If we are unable to accurately predict the mood of music through this algorithm, we will use another library to derive the mood or we will switch to simple frequency analysis of audio", for instance). For the third quarter, risk had been minimized so well by the team's previous attentiveness to it, and so much confidence in the technical feasibility of the solution had been created that weekly risk reports were halted with no significant negative impacts on team effort or the fidelity of the final product.

Teamwork and management issues were rare on our team, in part because of each team member's experience with working in a collaborative environment (another skill taught by previous courses and demonstrated relentlessly in this one). Team members traded off speaking roles during the sprint reviews, although the majority of speaking was done by those who did not mind it as much as those who did. There was no "team leader" on our team, although during the final quarter the team did assign a person to be the "WIP Master" for the week, giving that person the responsibility of drawing up outlines for assignment documents prior to meetings and selecting a "team bonding" activity for the team to participate in. This role helped mitigate the collective slacking that seniors in their final term are traditionally susceptible to. Team members behaved with professionalism and respect for the duration of the project as well, with disagreements about architecture resolved through discussion and diagramming. No significant

arguments or ill-will developed between any team member and another, as best the members can tell.

Finally, arguably the most important aspect of a team doing scrum is its ability to make process improvements based on reflection and corrective action. The retrospective ceremony in scrum is the official process by which teams collect and document these ideas. For our team, retros were conducted on Thursday evenings when our sprints were ending, and we followed a template of evaluating how well we implemented improvements identified in the previous retrospective, then discussing improvements to be made based on the last two weeks' performance.

The most notable improvements our team made were:

1) Sending out a calendar invite specifically for each weekly meeting helped team members to remember to attend beforehand (multiple iterations to address this during fall and winter).
2) Breaking activities such as grooming into asynchronous units of work meant to be done outside of meetings (primarily identified during winter).
3) Adding an extra hour to our scheduled meeting invite on the night of a sprint ending to help members plan for more time (identified during winter).
4) Making documenting findings part of a PBI instead of an afterthought (identified in multiple iterations in fall).
5) Exposing more members of the project to more domains, instead of letting team members specialize in one area (identified in winter).
6) Switching to 2-week sprints instead of 3-week sprints in our final term, since this helped our team get in more continuous contact with our stakeholder, and also placed more pressure upon team members to accomplish sprint tasks versus a 3-week sprint which tended to drag.
7) Designating a person to prepare document templates before meetings as well as pick out a bonding activity.
8) Ensuring to test all code on-platform earlier in the sprint, since although development environments were similar to production, they were not exact.

Altogether, the team concurs that our project was a fulfilling and interesting look at how to expand a unique product, and while our final product is not production-ready, we are convinced it is production-near, and look forward to seeing what types of innovations sisyphus industries can derive from our work.