The Project:

Public Interactive Installation to Demystify Sensors and “SMART” Technology.

By Edward Clarence Deaver, IV
For the City of Syracuse Innovation Team

Project Advisor: Dr. Voorhees
Project Manager: Sam Edelstein

Table Of Contents:

Definitions

Term/Acronym	Definition
Transparency	"How something is open enough to allow things to be deeply observed from different perspectives."[1]
Processing (Java - framework)	A Java framework "built for the electronic arts, new media art, and visual design communities"[2]
OpenFrameworks (C++ - Framework)	"About." openFrameworks, November 30, 2019. https://openframeworks.cc/about/.
NodeJS	JavaScript based server that is designed for async behavior by default.
Express	Minimalist web framework.[3] Flask is to Python, as Express is to Node.
Socket IO	Library built on top of web sockets to make working with them easier.
FadeCandy Server	A server package to accept commands for the Fade Candy.
FadeCandy	A USB Teensy based interface between a computer and NeoPixel strips.[4]
OPC	Open Pixel Control protocol is an open-source protocol to control LEDs. The FadeCandy server accepts messages formatted into this structure.
Adafruit HUZZAH ESP8266	An Arduino like device that is based on the ESP8266.
Ultrasonic Distance Sensor	This is an ultrasonic sensor that allows you to trigger and listen for a response to obtain distance data.

[1] Cysneiros, Luiz Marcio, and Vera Maria Benjamim Werneck. "An Initial Analysis on How Software Transparency and Trust Influence each other." In *WER*. 2009.http://www.inf.puc-rio.br/~wer/WERpapers/artigos/artigos_WER09/cysneiros.pdf
[2] "Processing." Visualising Information for Advocacy. Accessed December 1, 2019. https://visualisingadvocacy.org/node/725.html.
[3] "Node.js Web Application Framework." Express, December 4, 2019. https://expressjs.com/.
[4] Scanlime. "Scanlime/Fadecandy." GitHub, January 6, 2019. https://github.com/scanlime/fadecandy.

Raspberry Pi	A mini computer that runs Linux, and has GPIO pins.
Mac Mini	Small Apple computer.

Project Plan

The project plan was originally created with the assumption that Twitter or Twilio would be used in the project. Due to conversations that reevaluated the Twitter strategy Internet based interactivity was cut in favor of solely using distance sensors (and other physical sensors) connected to a Raspberry Pi. This subsequently cut the image manipulation part of the previous project plan as well. Also, Twitter has denied our application. The Raspberry Pi was cut in favor of using a Mac Mini due to software limitations. The project will only be using 2 distance sensors.

The project started as of August 24, 2019, with the application to have a Twitter Developer Account. The Initial Parts stage of tasks started September 2, 2019 and ended September 19th. The Lighting stage of tasks started September 9, 2019 and ended November 21, 2019. The External Data Input stage started September 24, 2019 and ended October 21st. Real Clear Information stage started October 21st and ended November 21, 2019. The Audio stage started as a last minute add on November 22, 2019 and ended December 6, 2019. The Networking spiral (a last minute addon) of the project started November 26, 2019 and is currently in progress. The Networking spiral still needs the AES parts to be implemented and tested. The Website, and Installation spirals have not been started. As of the midterm OpenFrameworks has replaced Processing, and a Mac Mini has replaced a Raspberry Pi 4.

The project uses a Trello board for project management:
https://trello.com/invite/b/4cKgckkG/653c713386306cc274f7aca1076d212b/research-project
The Spirals in this document are more up to date than the Trello board.

The code for the project can be found here:
https://github.com/EdwardDeaver/SyracuseInnovationLEDProject

The software that runs on the local machine is in the localhost folder, and the NodeJS files are in the Heroku folder.

The OpenFrameworks source code is here in ofApp.cpp
https://github.com/EdwardDeaver/SyracuseInnovationLEDProject/tree/master/LocalHost/OpenFrameworks/apps/myApps/PublicInstallationProject/src

The Arduino source code is here in ReadingInDataFromUltraSonic.ino:
https://github.com/EdwardDeaver/SyracuseInnovationLEDProject/tree/master/LocalHost/Arduino/ReadingInDataFromUltraSonic

The Python UDP Server script is here in UDPSERVER.py:
https://github.com/EdwardDeaver/SyracuseInnovationLEDProject/tree/master/LocalHost/Python

The Node JS main file is here in index.js:
https://github.com/EdwardDeaver/SyracuseInnovationLEDProject/blob/master/Heroku/NodeJS/index.js

Trello board color scheme:

- Pink - Software
- Navy Blue - Logistics
- Green - Hardware
- Yellow - Milestone
- Orange - WantsNotNeeds - This is something I would like to happen but it's not needed for the core project.

Gantt Chart color scheme:

- Pink - Software
- Purple - Logistics
- Green - Hardware

Spiral Stages: Planning

Objective

To understand the scope of the project and obtain materials to continue with the project.

Activities

These activities are necessary for the project to move forward.

#	Name	Scope
1	Tax Exempt Status	Obtain tax exempt status from Adafruit and Amazon.
2	Obtain tools from Adafruit	Obtain distance sensors, lights and miscellaneous tools.
3	Obtain tools from Amazon	Obtain Raspberry Pi 4, FadeCandy, and miscellaneous sensors.
4	Revize Trello board.	The project is managed in a Trello board, and TeamGhantt.

Spiral Stages: Lighting

Objective

To control the LEDs via hardware input.

Activities

These activities are to create the LED interactivity. This stage overlaps with the GPIO stage.

#	Name	Scope
5	Setup FadeCandy server	Setup the FadeCandy server and wire the lights to the FadeCandy.
6	External Data Input to Graphics to OPC	Create a representation of the LEDs in a grid pattern, and feed the distance sensor data into the graphics to modify the lights.
7	Animations	Solidify animations that are triggered when the distance sensors input data.
8	Stress testing	Stress test the graphics capabilities of the Mac Mini.
9	Revise Trello board.	The project is managed in a Trello board, and TeamGhantt.

Spiral Stages: External Data Input

Objective

To obtain input from 2 distance sensors and feed them into an OpenFrameworks program.

Activities

These activities are to create the LED interactivity. This stage has been completed.

#	Name	Scope
10	Connect Ultrasonic Distance Sensor to ESP8266 Arduino	Successfully read in distance data into Arduino
11	Determine accurate data collection from sensors	Stress test the number of inputs that the Raspberry Pi can handle.
12	Determine how to send data over serial	Should the Arduino program use serial print or serial write.
13	Determine how to read byte data and convert it to ASCII in OpenFrameworks	Successfully read in and convert byte data from the Arduino to usable data input.
14	Revise Trello board.	The project is managed in a Trello board, and TeamGhantt.

Spiral Stages: Real Clear Information

Objective

To explain how the system operates in common vernacular without losing accuracy. There is a high chance not all these points will be completed by December.

Activities

These activities focus on communications, graphic design and web design.

#	Name	Scope
15	Create clear definitions of the material parts of the system	Define each part of the system in clear English.
16	Test definitions on people.	Ask people with varying levels of technology literacy if they understand the definitions.
17	Revize Trello board.	The project is managed in a Trello board, and TeamGhantt.

Spiral Stages: Audio

Objective

Produce tone for each range of movement.

Activities

These are extra steps not needed for core functionality.

#	Name	Scope
18	Play audio	Load an audio file into the program and play it.
19	Control audio panning	Create audio for left and right channels.
20	Play audio using sensor input	Pass sensor data into the audio function to play tones per category.
21	Testing	Does the audio successfully play without clipping? Do the audio files play for the correct categories? Do the audio files continue to play after rapid change in input value?

Objective

Send data of sensors to a web server.

Activities

These are extra steps not needed for core functionality.

#	Name	Scope
22	Create NodeJS webserver	Create an Express web server that can accept a post request on glitch.com.
23	Implement input validation	Validate sensor inputs on NodeJS.
24	Move the project to Heroku	Move the project to be hosted on Heroku from Glitch.com.
25	Implement SocketIO server	Create a Socket IO server that will emit data when the post request is completely validated.
26	Create Socket IO Client	Create client-side JavaScript to create a socket io connection to the server.
27	Create valid post request in Python	Create a proof of concept that can successfully send a post request using Requests to the NodeJS server.
28	Create a UDP server in the same Python file	Create a UDP server that can accept connections and send data to the NodeJS server via a http post request.
29	Use a UDP socket in OpenFrameworks	Successfully send sensor data to UDP server. Note: The ofxlibwebsockets library throws architecture errors when compiling in my openFrameworks app. So, I pivoted.
30	Create a sending data rate limit	Limit the sending of data from OpenFrameworks to 1 piece of data sent per 10 seconds.
31	Send data to server	Test if the UDP server still sends data to the server.
32	Test	Does the system currently end connection if any of the validation checks are failed?

		Does the connection allow non-correct data to be sent to the client? Do each of the if statements work?
33	Validate Python UDP server input	Make sure python server is receiving numbers.
34	Implement AES encryption on the Python side	Encrypt the secret key using AES.
35	Implement AES decrypt / Encrypt on the NodeJS side	Decrypt the incoming message and test if it matches the stored key. If not reject connection.
36	Test	The security aspect of this part will require continuous testing.

Spiral Stages: Website

Objective

Make a front end website to explain the project

Activities

These are extra steps not needed for core functionality.

#	Name	Scope
37	Client-side validation of Socket IO	Implement client-side validation of the incoming Socket IO data.
38	What this project does section	Create a section that explains why this is important.
39	Why this project exists section	Create a section that explains why this exists.
40	How this project works section	Create a section that explains how this works.

Spiral Stages: Installation

Objective

Send data of sensors to server.

Activities

These are necessary for presenting the project.

#	Name	Scope
41	Portable install design	How can this be installed on a cart.
42	Print labels for each part of the project	Print the simple definitions onto labels for each part.

Design Document

Project Overview

The purpose of the project is to demystify smart technology by providing physical interaction for them. The user will be able to see a clear relationship between physical action and digital reaction. They will be able to see how sensor inputs effect digital things, the lights.

Requirements

Functional:

1. Control lights using distance sensor data.
2. Distance sensors data controls animations in OpenFrameworks.
3. Power supply should have redundant failsafes.
4. The program needs to stop lights from being set to all white for 5 minutes or more. This is due to heat (RGB LEDs have individual LEDs per color channel, and to make white all of them are on at full brightness).

Non-Functional:

1. The performance needs of the system are the ability to not have notable frames per second drops.
2. The exhibit should be accessible during business hours.
3. All power supplies should be UL rated.

Design Rationale

The application uses the OpenFrameworks GUI interface as the main source of control because it will provide the easiest form of applying advanced animations to a grid of LEDs by creating what can be thought of as a low-resolution screen. The FadeCandy is used because it is the easiest tool to control NeoPixels at scale in a Do-It-Yourself fashion.

How does this convey an understanding of #SmartNotMagic

First it should be understood why citizen's knowledge of "smart" things is important. Governments are currently having vendors try to sell them facial recognition software, ways to monetize citizens data, and artificial intelligence solutions.[5] [6] This system itself will not produce a complete understanding of "smart" things by the entire public, but it can provide a jumping off point to explore ideas. First, the project will try to help the understandings of "smart" through the use of radical transparency of the system. Radical transparency is a business philosophy based on total openness.[7] It is similar in some respects to the open-source ethos.[8] Second, I have created technology definitions for parts of the system that are defined using the lowest technology literacy rate of a given area to create simple definitions. Though I do not know how to define technology literacy rates I have made sure the definitions use metaphors that relate to devices the reader probably has interacted with. The project relies on two these parts to accomplish the mission because when transparency increases in a [governmental] system it acts as a multiplier on preconceived notions of government to reinforce those held ideas.[9] [10] Researchers at the Utrecht University Netherlands conducted a study(N=658) to test the link of transparency of government and trust. The researchers found that due to preexisting ideas of government that people had the effects of transparency were not prominent in increasing trust.[11] [12]

It is possible to change these preconceived notions. In advertising, agencies have been doing it for years. Agencies like DoSomething.org and the AdCouncil participate in advocacy or education related advertising campaigns.[13] [14] [15] At the federal government level there has been the creation of code.gov to show what open-source projects the federal government has made and is currently sharing.[16]

[5] Sam, Edelstein. Interview by Edward Deaver. Personal interview. Syracuse,December 3, 2019.
[6] ACLU, "How to Stop 'Smart Cities' From Becoming 'Surveillance Cities'". https://www.aclu.org/blog/privacy-technology/surveillance-technologies/how-stop-smart-cities-becoming-surveillance-cities
[7] Fishman, Marina. "How to Do Radical Transparency Right as a Manager." Copper. Copper, August 20, 2019. https://www.copper.com/blog/radical-transparency.
[8] "What Is Open Source?" Opensource.com. Accessed December 4, 2019. https://opensource.com/resources/what-open-source.
[9] Grimmelikhuijsen, Stephan. (2012). Linking Transparency, Knowledge and Citizen Trust in Government: An Experiment. International Review of Administrative Sciences - INT REV ADM SCI. 78. 50-73. 10.1177/0020852311429667.
[10] Grimmelikhuijsen, Stephan. (2010). Transparency of Public Decision-Making: Towards Trust in Local Government?. Policy & Internet. 2. 10.2202/1944-2866.1024.
[11] Grimmelikhuijsen (2012).
[12] Grimmelikhuijsen (2010).
[13] Wolff, Martijn, and W E Biernatzki. "The Social and Cultural Effects of Advertising ." *Communication Research Trends* 14, no. 1 (1994): 34–34. http://cscc.scu.edu/trends/v14/V14_1.pdf.
[14] "Let's Do This!" Let's Do This! | DoSomething.org. Accessed December 4, 2019. https://www.dosomething.org/us.
[15] "Ad Council." AdCouncil. Accessed December 4, 2019. https://www.adcouncil.org/.
[16] "Code.gov." Code.gov. Accessed December 4, 2019. https://code.gov/.

Dependencies

Localhost software dependencies:

1. SiLabs CP210x USB to UART Bridge VCP Drivers for Mac OS.
   Use: To be able to read in serial data from the ESP8266.
   a. https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers
2. Fade Candy Server
   a. https://github.com/scanlime/fadecandy
3. Python UDP server:
   a. Python version 3.
   b. Socket (Native Python package)
   c. Json (Native Python package)
   d. python-dotenv==0.10.3
   e. requests==2.22.0
4. Caffeine - an app that forces the Mac computer to never sleep.

Localhost hardware dependencies:

1. 2x ESP8266 3.3v Arduino board.
2. 1x FadeCandy device
3. 4x WS2811 leds 60 per strand. (NeoPixel)

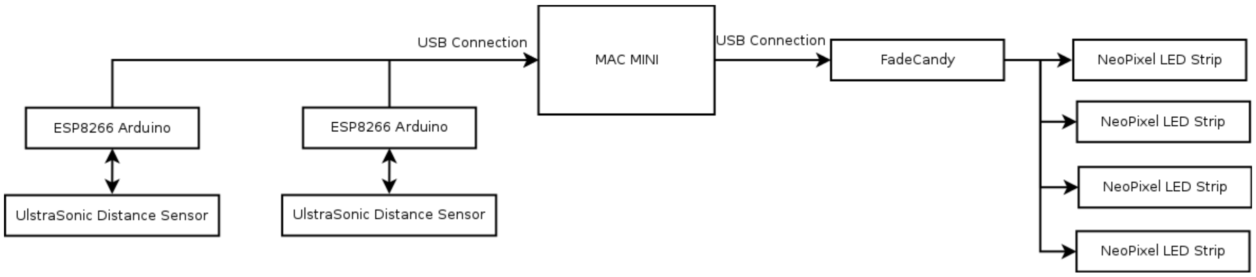
Development package depencies:

1. OpenFrameworks 0.10.1
   a. ofxNetwork
   b. ofxOPC
      i. https://github.com/DHaylock/ofxOPC
   c. ofxGui
   d. ofxNetwork

Node JS Webserver dependencies:

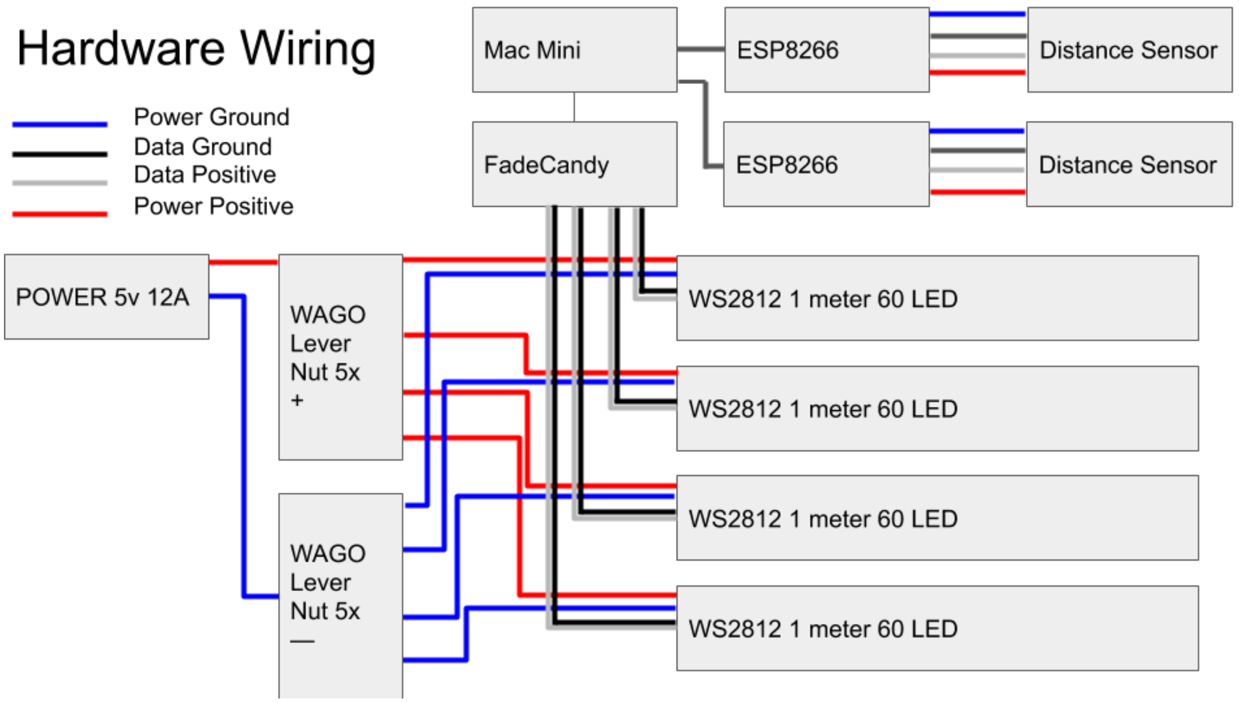
1. **NodeJS == 12.X**
2. **"ejs": "^2.5.6"**
3. **"express": "^4.15.2"**
4. **"body-parser": "1.19.0"**
5. **"http-server": "0.12.0"**
6. **"socket.io": "2.3.0"**
7. **"request": "^2.81.0"**
8. **"tape": "^4.7.0"**

System Architecture (Hardware)



The hardware system architecture is comprised of two ESP8266 Arduinos, Mac Mini, FadeCandy, Ultrasonic Distance Sensor and Neopixel LED Strips. The Arduinos are connected to the Mac Mini via a USB cable. The FadeCandy is connected to the Mac Mini via a USB Cable. The FadeCandy is connected to the NeoPixel LED Strips via 2 data wires. The changing software stack caused the hardware change from the Raspberry Pi 4 to Mac Mini. The move towards OpenFrameworks from Processing was because when the Raspberry Pi 4 was assumed to be the final hardware I discovered Processing can not produce a microsecond delay which is required to use the ultrasonic distance sensors. Those sensors were going to be connected to the GPIO pins on the Raspberry Pi 4. Because of the inability to use them with Processing, this resulted in the move towards using external Arduinos and sending that data over serial. The Raspberry Pi 4 was replaced by the Mac Mini due to complications that arose in compiling OpenFrameworks on Debian Buster for the Raspberry Pi 4. Also, having the ultrasonic distance sensor data parsed externally frees up resources on the Mac Mini and allows future input from multiple sensors on one Arduino. The FadeCandy is used to control the lights because they require a specific data timing that the Raspberry Pi 4 could not meet (the Mac Mini does not have GPIO pins), and it allows easy control of NeoPixel LEDs. Using a desktop system allows using a screen to show how data inputs are parsed into physical outputs: the LEDs.

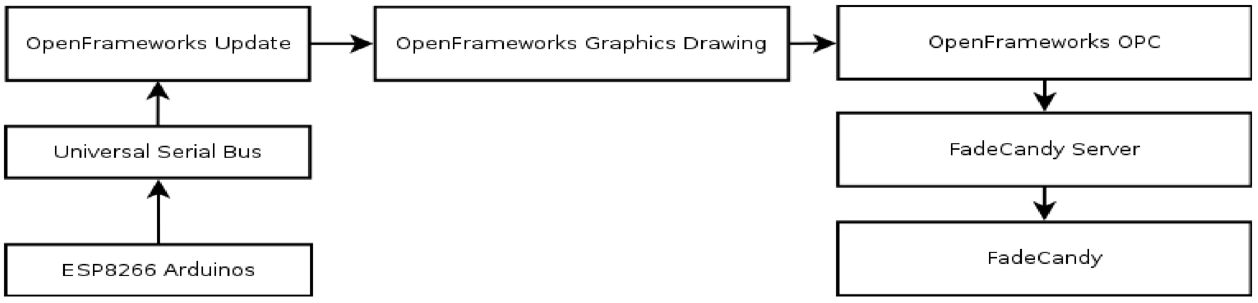
System Architecture (Hardware-wiring)



The power for the LED strips is supplied via a 5-volt 12-amp power supply then split into parallel lines supplying power to the individual led strips. The LED strips are individually wired to the FadeCandy as well. I wired female JST SM connectors to the FadeCandy and created a male-to-male JST SM 6-foot extension cable for each led strip, with one end plugging into the Fade Candy and the other end the LED strips. The power leads for the LEDs were also extended but due to the potential 3.6 amp draw of an individual LED strip I could not use the JST SM connector because they are rated for a max amperage of 3 amps, and the LED strips have the potential to draw 3.6 amps.[17] The Fade Candy connects to the Mac Mini via a mini-USB cable. One ESP8266 connects to the Mac Mini via a micro-USB cable, and the other via a USB to TTL Serial Cable. The distance sensors are connected to the ESP8266s via 4 wires, 2 for power and 2 for data.

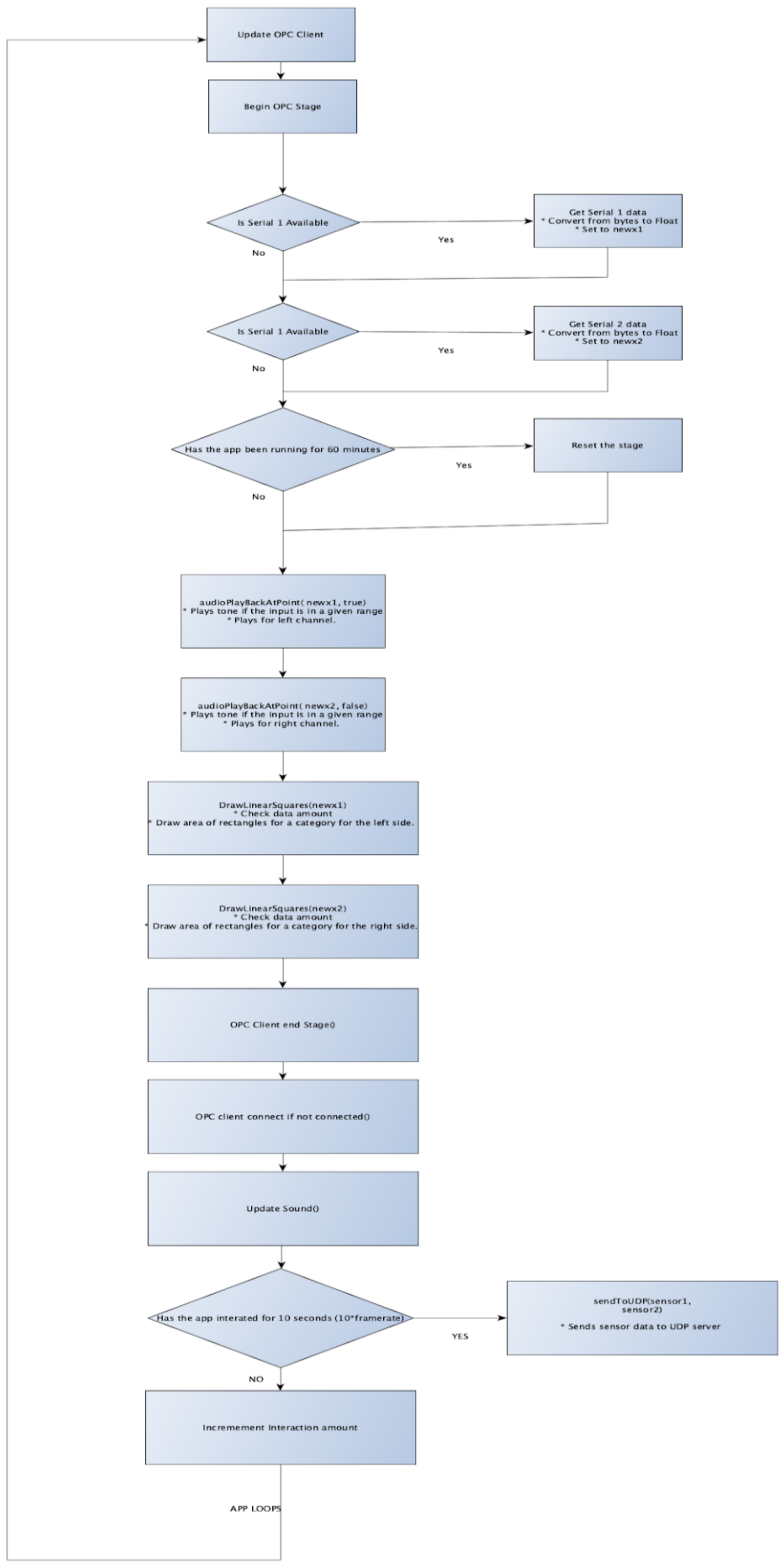
[17] "SM Connector." JST SM Connector. Adafruit. Accessed December 5, 2019. https://cdn-shop.adafruit.com/datasheets/JSTSM.pdf.

System Architecture (Software - Data High Level Overview)



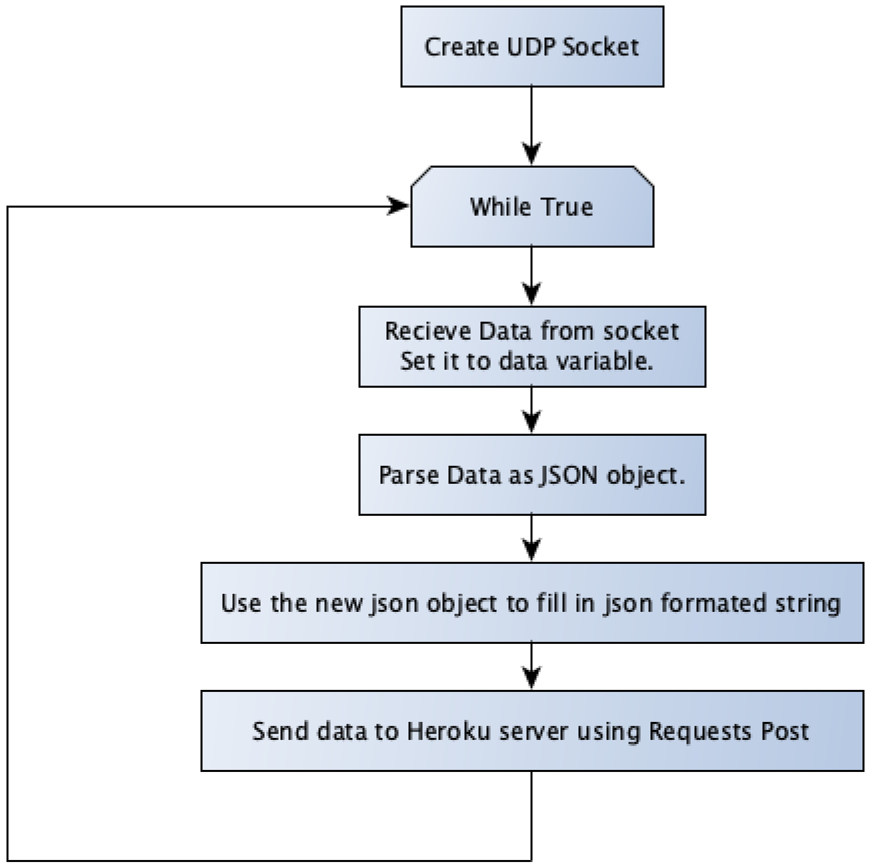
The software system architecture is comprised of Arduino C++ code, and OpenFrameworks C++ code. The Arduino code is used to control the specific timings of the distance sensors and reduce the computation time done by the main OpenFrameworks program. The OpenFrameworks library is used to easily create graphics from the serial bus data. Using OpenFrameworks allows access to a large Addon community. This allows me to use a similar package, OPC library, created by David Haylock, and in this application modified by me, Edward Deaver.

System Architecture (Software - OpenFrameworks Update Function)


Update OPC Client
Begin OPC Stage
Is Serial 1 Available
Yes
No
Get Serial 1 data
* Convert from bytes to Float
* Set to newx1
Is Serial 1 Available
Yes
No
Get Serial 2 data
* Convert from bytes to Float
* Set to newx2
Has the app been running for 60 minutes
Yes
No
Reset the stage
audioPlayBackAtPoint(newx1, true)
* Plays tone if the input is in a given range
* Plays for left channel.
audioPlayBackAtPoint(newx2, false)
* Plays tone if the input is in a given range
* Plays for right channel.
DrawLinearSquares(newx1)
* Check data amount
* Draw area of rectangles for a category for the left side.
DrawLinearSquares(newx2)
* Check data amount
* Draw area of rectangles for a category for the right side.
OPC Client end Stage()
OPC client connect if not connected()
Update Sound()
Has the app interated for 10 seconds (10*framerate)
YES
NO
sendToUDP(sensor1, sensor2)
* Sends sensor data to UDP server
Incremement Interaction amount
APP LOOPS

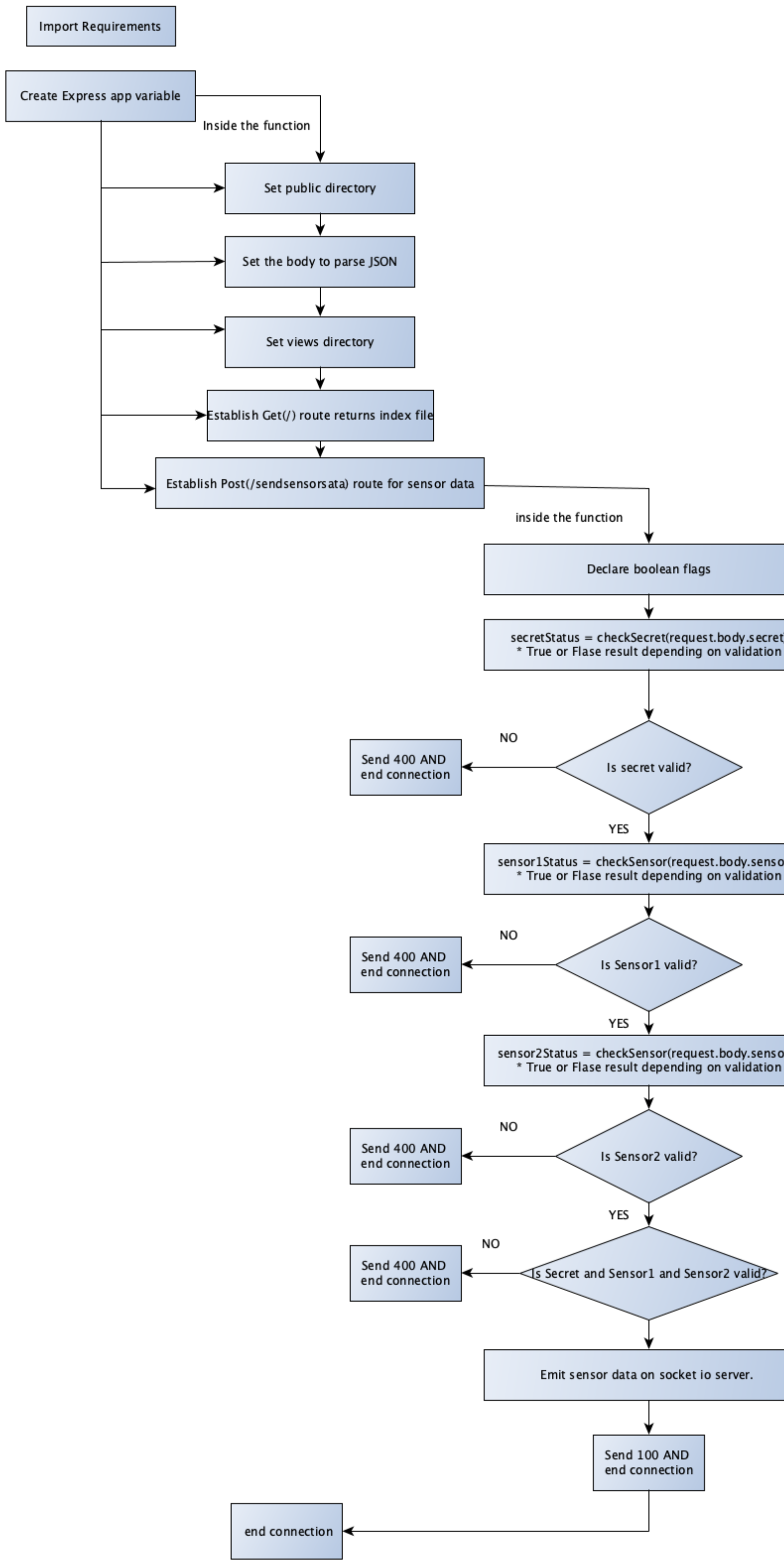
The update function is where data is updated in the OpenFrameworks framework. The application first runs the OPC update function which resets errors and timer data. Next, an if statement checks if serial1 is available, if it is listenToSerial1(dataDivisionAmount) is ran and ***newx1*** is set to its value. dataDivisionAmount is used to scale data for the screen, it is currently set to 1. Then, if serial2 is available it will run listenToSerial2(dataDivisionAmount) and ***newx2*** is set to its value. Then resetTimeAndSpace(60*numberOfMinutesTillReset) is ran to check if the program has been running for 60 minutes. If it has it clears the screen, and resets its timer. Then, audioPlayBackAtPoint(newx1, **true**) and audioPlayBackAtPoint(newx2, **false**) are called to produce sounds for a given range. If the second value is true it plays audio for the left audio channel, if it is false, it plays audio for the right channel audio. After that DrawLinearSquares(newx1,0, topSquareHeight, **true**) and DrawLinearSquares(newx2,0, topSquareHeight, **false**) are called. The last value determines it draws rectangles for the left or right led strips, true is for the left side and false is for the right side. Then OPC EndStage is ran which reads the pixel values from the screen into an external frame buffer the OPC library creates. Then OPC Client writes that pixel data to different lighting channels. Then, the app runs ofSoundUpdate() which is an OpenFrameworks internal function that updates the sound engine. Next, if 10 seconds has passed the application will call sendToUDP, to send the sensor data.

System Architecture (Software - Python UDP Server)


Create UDP Socket
While True
Recieve Data from socket
Set it to data variable.
Parse Data as JSON object.
Use the new json object to fill in json formated string
Send data to Heroku server using Requests Post

The Python application creates a UDP socket on the localhost. It continuously is listening for a message. Once it receives a message it creates a json formatted string of the data with the secret key inside it. Then it sends a post request to the Heroku server. The key will need be encrypted using AES in the future, and input validates. Then the loop goes back to the top of the while loop. Input

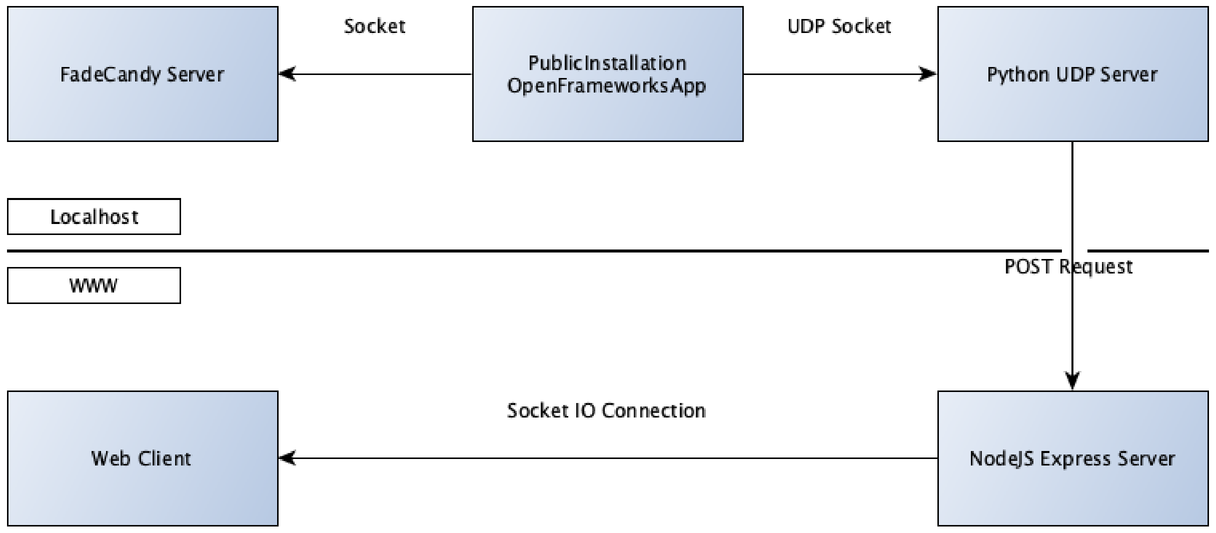
System Architecture (Software - NodeJS Web Server)



Import Requirements
Create Express app variable
Inside the function
Set public directory
Set the body to parse JSON
Set views directory
Establish Get(/) route returns index file
Establish Post(/sendsensorsata) route for sensor data
inside the function
Declare boolean flags
secretStatus = checkSecret(request.body.secret)
* True or Flase result depending on validation
NO
Is secret valid?
Send 400 AND end connection
YES
sensor1Status = checkSensor(request.body.sensor1);
* True or Flase result depending on validation
NO
Is Sensor1 valid?
Send 400 AND end connection
YES
sensor2Status = checkSensor(request.body.sensor2);
* True or Flase result depending on validation
NO
Is Sensor2 valid?
Send 400 AND end connection
YES
NO
Is Secret and Sensor1 and Sensor2 valid?
Send 400 AND end connection
Emit sensor data on socket io server.
Send 100 AND end connection
end connection

The NodeJS Express web server, hosts the post request endpoint and the Socket IO server. This description will focus on the post request endpoint. When the server receives a post request on the /sendsensorsata endpoint it will first establish local boolean variables for the status of each of the validation methods for sensor and secret variables. The secret is validated to see if it matches with the one stored as a NodeJS environment variable. If it doesn't match it's associated boolean value is set to false, and the connection is ended. If it does match its boolean value is set to true, and sensor 1 is checked if it is not undefined and is a number. If its value fails, its boolean value is set to false and the connection is ended. This process is the same for sensor2. If all three boolean values are true, then the sensor data is emitted from the socket io server. The post function also has a function to end the connection outside of all the if statements.

Network Architecture (Dataflow)



The OPC library creates a socket connection to the FadeCandy server in order to tell the FadeCandy board what to do. My application creates a UDP Socket to a Python UDP server to get the data from openFrameworks to the outside world. I am using a UDP socket instead of a TCP implementation due to the potential TCP will slow my application down if packets are dropped. Due to the nature of this data, it is not of high value so it is not an issue if some data points are lost. The Python server then sends the NodeJS server a Post request. Once the NodeJS validates the incoming data it will emit that to the web client over a Socket IO connection.

Network Architecture (Messages)

The application uses JSON formatted messages to communicate with every part of the system except the FadeCandy server which the OPC library sends messages using the OPC protocol.

The message from the OpenFrameworks application to the UDP Server is:

- {“sensor1”: Float, “sensor2”: Float, “timesent”: SystemTimeInMillisSeconds }

The message from the UDP Server to the NodeJS server:

- { "sensor1": Float , "sensor2": Float, "secret": SECRETVALUE}'

The message from the NodeJS server to the SocketIO client:

- {“sensor1”: Float, “sensor2”: Float}

Folder Structure (openFrameworks)

OpenFrameworks Folder Structure:

- OpenFrameworks Root
  - Addons
  - Apps
    - myApps
      - PublicInstallationProject
        - Src
          - main.cpp
          - ofApp.cpp
          - ofApp.h
        - Bin
          - Data
            - OpenSans-Bold.ttf
            - tone83.mp3
            - tone130.mp3
            - tone174.mp3
            - tone261.mp3
          - PublicInstallationProjectDebug.app
  - Docs
  - Examples
  - Libs
  - Other
  - Scripts

Folder Structure (Python)

OpenFrameworks Folder Structure:

- Python UDP Server
    - Requirements.txt
    - UDPSERVER.py

Folder Structure (NodeJS)

NodeJS Folder Structure:

- **NodeJS** Root:
    - **app.json**
    - **Index.js**
    - **> node_modules (folder)**
    - **package-lock.json**
    - **package.json**
    - **Procfile**
    - **> public (folder)**
    - **README.md**
    - **Test.js**
    - **> views (folder)**

File Descriptions

Application:
Pre-condition: fcserver is currently running, and UDPSERVER.py is running.

FILE: ofApp.h
Description: Intialitizes the variables and classes for the project that are used in the ofApp class. The variables and classes are separated into a public and private scope.

FILE: ofApp.cpp
Description: This runs the application. All data processing goes on here.

FILE: main.cpp
Description: Main method to run the app in ofApp.cpp.

FILE: ofxOPC.cpp
Description: OPC object supplied in ofxOPC library. I modified it to allow the stage background to be reset or held. Also, I added a string value to set the file path for the font used in the file.

FILE: ReadingInDataFromUltraSonic.ino(Arduino):
Description: This reads the distance of data from the ultrasonic distance sensor and prints it over serial. The sent data is an average of 30 readings. The board is an Adafruit HUZZAH ESP8266 Breakout. It operates at 3.3v opposed to the Arduino Uno's 5v.

FILE: UDPSERVER.py:
Description: This creates a UDP socket server on the localhost. When messages are received it sends a json formatted post request to the NodeJS server.

FILE: index.js (NodeJS webserver):
Description: This reads serves the project webpage via an Express web server. It also hosts a post request endpoint in which it validates the sensor data inputted, and if it is valid emits it over a socket io connection.

Function Descriptions

FILE: ofApp.cpp

Function: setup()
Description: This sets the variables for the ofApp class. Also, the setup function connects to and establishes the structure of LEDs.
Pre-conditions:

- The function requires the setup function to be declared public in ofApp.h.
- numberOfStrips, movement, lastTime, BackgroundHold, numberOfMinutesTillReset, bSendSerialMessage, nTimesRead, nBytesRead, readTime, hasItRan are declared as a public variable in ofApp.h

Post-conditions:

- Serial1 and Serial2 are created. A GUI window is created.

Function: audioToneSetup()
Description: Loads and sets all ofSound objects for program.
Pre-conditions:

- The sound objects have been created.

Post-condition:

- All data values set. Returns true.

Function: setupOPCLeds(string IPAddress, int port, int stageWidth, int stageHeight, int numberOfStrips, int numberOfLEDS)
Description: Creates NeoPixel objects for each led strip.
Pre-conditions:

- IPAddress is a string.
- port, stageWidth, stageHeight, numberOfStrips, and numberOfLEDS are ints.
- OPCClient is connected.

Post-conditions:

- The LEDS strips are created.

Function Descriptions

FILE: ofApp.cpp

Function: update()
Description: Ingests serial data, runs graphics effects, and sends data to OPC Client. Note that unlike setup, update is run in a loop by the OpenFrameworks library.
Pre-conditions:

- opcClient is connected successfully.

Post-conditions:

- Post-conditions: Data is successfully sent to FadeCandy Server on different channels.

Function: draw()
Description: Draws things to the stage(normally). In this case will draw sensor information and led strips.
Pre-conditions:

- opcClient is successfully connected.
- A GUI window has been created.
- LED strips have been generated.

Post-conditions:

- Sensor data is drawn to the window successfully.

Function: keyPressed(int key)
Description: This is an event listener in OpenFrameworks that listens for a key press. This is not used in this application but without it the program fails to compile.

Function: audioOut(ofSoundBuffer &outBuffer)
Description: This is an event listener in OpenFrameworks that listens for changes in audio.

Function: keyReleased(int key)
Description: This is an event listener in OpenFrameworks that listens for a key press. This is not used in this application but without it the program fails to compile.

Function Descriptions

FILE: ofApp.cpp

Function: exit()
Description: Exits the program and closes connection to fcserver.
Pre-conditions:

- The program is running.
- opcClient is connected to the fcserver.

Post-conditions:

- Application stops running.

Function: resetTimeAndSpace(int secondsToWaitFor)
Description: Reset on screen data by wiping the screen with a rectangle, resets the line object, and resets the timer.
Pre-conditions:

- secondsToWaitFor is an int.
- ofGetElapsedTimef started running at the beginning of the program.

Post-conditions:

- Elapsed time counter is reset.
- Screen is reset.

Function: DrawLinearSquares(int input1, int input2, float topSquareHeight, bool left)
Description: Draws more squares on screen as the user gets closer. If ***left*** is true it draws rectangles on the left side of the screen, if false it draws rectangles on the right side of the screen.
Pre-conditions:

- input1, input2 is an int.
- topSquareHeight is a float.
- Left is either true or false.

Post-conditions:

- None.

Function Descriptions

FILE: ofApp.cpp

Function: audioPlayBackAtPoint(float input, bool left)
Description: Changes volume for audio being played given the input and range. If left is true plays audio for left side. If false, plays for right side
Pre-conditions: The sound objects have been created and set.
Post-conditions: A given volume is set.

Function: DrawSquares(int x)
FUNCTION CUT FROM FINAL BUILD
Description: This uses time to control the coloring of rectangles.
Pre-conditions:
- x is an int.

Post-conditions:
- None.

Function: DrawMeteors(float x1, float x2)
FUNCTION CUT FROM FINAL BUILD
Description: Draws squares that persist on screen that are set at the user input of x1/x2.
Pre-conditions:
- x1/x2 are floats.
- line variable has been created.

Post-conditions:
- None.

Function: listenToSerial1(int divisionAmount)
Description: Listens for input on serial input 1.
Pre-conditions:
- serial1 has been initialized.
- newx1 has been declared a global variable.
- divisionAmount is an int.

Post-conditions:
- newx1 is returned as a float.

Function Descriptions

FILE: ofApp.cpp

Function: listenToSerial2(int divisionAmount)
Description: Listens for input on serial input 2.
Pre-conditions:

- serial2 has been initialized.
- newx2 has been declared a global variable.
- divisionAmount is an int.

Post-conditions:

- newx2 is returned as a float.

Function Descriptions

FILE: ofxOPC.cpp (Modified)

Function: beginStage(bool hasItRan)
Description: Sets the stage of the leds and begins ofFbo object(second frame buffer that can be written to).
Pre-conditions:

- hasItRan is a boolean
- screenCapture has been declared.

Post-conditions:

- The LEDS strips are created.

Function Descriptions

FILE: UDPServer.py

Function: The file is one function.
Description: Creates UDP Socket server and sends json post request to the NodeJS server.
Pre-conditions:

- None.

Post-conditions:

- None.

Function Descriptions

FILE: index.js

Function: post("/sendsensorsata", function(request, response)
Description: Creates a Post request end point for data. The data sent in is then verified. If successfully verified it is sent to the socket io client.
Pre-conditions: NodeJS has the key stored as an environment variable.
Post-conditions: Connection is closed.

Function: checkSecret(secretSent)
Description: Validates the secret key. Checks if environment variable is like the one sent by the server
Pre-conditions: Environment variable set.
Post-conditions: Returns true or false.

Function: checkSensor(sensorData)
Description: Validates Sensor Data. Checks if it is undefined, and a number.
Pre-conditions: None.
Post-conditions: Returns true or false.

Testing

Test 1: How will drawing to the screen without reset effect Frames Per Second?
Outcome: This led to the creation of the resetSpaceAndTime function.

Input	**Expected Results**	**Actual Results**
The program ran the drawMeteor function on a Mac Mini for 36 hours.	The program would freeze the machine.	The program dropped from its initial 30 FPS to 20 FPS and still received input from the Arduino devices.

Test 2: Accuracy rate of ultrasonic sensors.
Outcome: 10+- cm accuracy achieved with 30 millisecond delay per reading and returning an average of 30 readings. Each reading costs 30 milliseconds plus 12 microseconds, 0.030012 seconds. An average result in a time cost of 0.90036 seconds. Unfortunately, this still results in some number jumps which could be fixed by using a “time of flight” sensor.

Input	**Expected Results**	**Actual Results**
Gave a standard distance and changed the values of average amount and delay.	The data output would be accurate but not take as long.	The data was still in accurate at times.

Test 3: Categories range in DrawLinearSquares
Outcome: Due to the inaccuracies in the sensor direct control using the sensor data was cut. Now the effect uses large categories to try to catch data values when they adjust.

Input	**Expected Results**	**Actual Results**
Input sensor data drawLinearEffect and try to directly control the amount of squares on screen.	The program would produce a smooth effect.	The resulting effect was incredibly jumpy and at times almost strobing. This resulted in creating categories to capture the distance sensor even if it was not super accurate.

Test 4: Audio generation quality.
Note: Originally the program used the audio out function to generate audio in real time. This was cut due to its outcome.
Outcome: When the global variable frequency was modified to be higher values than 83 by the drawLinearSquares effect it would create a sin wave that made the audio start to clip. Clipping is when our audio's waveform hits the max volume when the rest of the wave remains under max volume. In OpenFrameworks audio volume is 0 to 1.

Input	**Expected Results**	**Actual Results**
Modified newx1, and newx2 values to count down from 3000. Newx1 = Newx1-2. Newx2 = Newx2 -1.	The program would flawlessly generate tones.	The audio clipped and created a very unpleasant experience.

Test 5: Audio playback error when input data jumps around.
Note: The audio playback can have issues when the input data jumps very quickly from one extreme to another. I fixed attempted to fix the bug by making sure if any category of audio was played the function immediately ran return to end the function. I also made sure any values outside the desired ranges had no way of playing audio.
Outcome: Using the if statement I made that would rate limit my UDP data transmission I set ***newx1*** and ***newx2*** to be random values with a max of 14000. I also decreased the rate limit from its original 10 second interval to 0.5 seconds. The 14000 max is a number that the sensors have produced and is one that is not within the categories ranges that make sound.

Input	**Expected Results**	**Actual Results**
Using the if statement I made that would rate limit my UDP data transmission I set ***newx1*** and ***newx2*** to be random values with a max of 14000. That value was changed every 0.5 seconds.	The audio should only play when in a defined range.	The audio failed to produce any errors and now works as intended.

Test 6: Does NodeJS reject a connection with a wrong secret key?
Outcome: When sending a json request using curl to the Heroku server with valid sensor 1 and 2 data but a malformed secret key the connection was closed.

Input	**Expected Results**	**Actual Results**
Using Curl to make sure my json requests were correct. I formatted a json request to the Heroku server that had everything correct except the secret key.	The server should close the connection and it immediately should end.	The connection immediately closed.

Test 7: Does NodeJS reject a connection with a non-number sensor1?
Outcome: When sending a json request using curl to the Heroku server with valid sensor 2 and secret data but a non-number sensor 1 the connection was closed.

Input	**Expected Results**	**Actual Results**
Using Curl to make sure my json requests were correct. I formatted a json request to the Heroku server that had everything correct except a non-number sensor 1.	The server should close the connection and it immediately should end.	The connection immediately closed.

Test 8: Does NodeJS reject a connection with an undefined sensor1?
Outcome: When sending a json request using curl to the Heroku server with valid sensor 2 and secret data but an undefined sensor 1 the connection was closed. The server sent back a web page stating there was an error as well "Bad Request". It did not accept the given inputs, so from that perspective it was a success. This is still unexpected behavior due to the error webpage being returned.

Input	**Expected Results**	**Actual Results**
Using Curl to make sure my json requests were correct. I formatted a json request to the Heroku server that had everything correct except an undefined sensor 1.	The server should close the connection and it immediately should end.	The server sent back an error page. It did not accept the given inputs, so from that perspective it was a success. This is unexpected behavior that will need to be addressed.

Test 7: Does NodeJS reject a connection with a non-number sensor2?
Outcome: When sending a json request using curl to the Heroku server with valid sensor 1 and secret data but a non-number sensor 2 the connection was closed.

Input	**Expected Results**	**Actual Results**
Using Curl to make sure my json requests were correct. I formatted a json request to the Heroku server that had everything correct except a non-number sensor 2.	The server should close the connection and it immediately should end.	The connection immediately closed.

Test 8: Does NodeJS reject a connection with an undefined sensor2?
Outcome: When sending a json request using curl to the Heroku server with valid sensor 1 and secret data but an undefined sensor 2 the connection was closed. The server sent back a web page stating there was an error as well "Bad Request". It did not accept the given inputs, so from that perspective it was a success. This is still unexpected behavior due to the error webpage being returned.

Input	**Expected Results**	**Actual Results**
Using Curl to make sure my json requests were correct. I formatted a json request to the Heroku server that had everything correct except an undefined sensor 2.	The server should close the connection and it immediately should end.	The server sent back an error page. It did not accept the given inputs, so from that perspective it was a success. This is unexpected behavior that will need to be addressed.

Test 8: Does NodeJS accept a connection with all valid inputs?
Outcome: When sending a json request using curl to the Heroku server with all valid data It showed the data on the webpage.

Input	**Expected Results**	**Actual Results**
Using Curl to make sure my json requests were correct. I formatted a json request to the Heroku server that had everything correct.	The data should be shown on the website.	The data values were shown on the webpage.

Real Clear Information Definitions

1. Motion detector:
    a. This allows the Arduino to tell how close you are. This is like motion detectors on garage lights.
2. Arduino:
    a. These are “microcontrollers”. They are told to do one thing and they do it forever. They’re like your cell phone but they only run 1 app.
3. Computer:
    a. “When you program, or code, you provide instructions for the computer to follow. Many programmers write code in text, meaning that they type it out on the keyboard.”[18]. On the computer a program is running that listens for the Arduinos to say something. It then draws its response on the computer screen and tells the lights to light up.
4. FadeCandy:
    a. This tells the lights which individual lights to light up and what color.

[18] Google, “CS First Get Started Guide”. https://docs.google.com/document/d/1LHJnxZBPxJXhrWPvnwy5N00T7D59MvVFSpnwyVNlp-o/edit

Bibliography

ACLU, “How to Stop ‘Smart Cities’ From Becoming ‘Surveillance Cities’”. https://www.aclu.org/blog/privacy-technology/surveillance-technologies/how-stop-smart-cities-becoming-surveillance-cities

“Ad Council.” AdCouncil. Accessed December 4, 2019. https://www.adcouncil.org/.

“Code.gov.” Code.gov. Accessed December 4, 2019. https://code.gov/.

Cysneiros, Luiz Marcio, and Vera Maria Benjamim Werneck. "An Initial Analysis on How Software Transparency and Trust Influence each other." In *WER*. 2009.http://www.inf.puc-rio.br/~wer/WERpapers/artigos/artigos_WER09/cysneiros.pdf

Fishman, Marina. “How to Do Radical Transparency Right as a Manager.” Copper. Copper, August 20, 2019. https://www.copper.com/blog/radical-transparency.

Google, “CS First Get Started Guide”. https://docs.google.com/document/d/1LHJnxZBPxJXhrWPvnwy5N00T7D59MvVFSpnwyVNlp-o/edit

Grimmelikhuijsen, Stephan. (2010). Transparency of Public Decision-Making: Towards Trust in Local Government?. Policy & Internet. 2. 10.2202/1944-2866.1024.

Grimmelikhuijsen, Stephan. (2012). Linking Transparency, Knowledge and Citizen Trust in Government: An Experiment. International Review of Administrative Sciences - INT REV ADM SCI. 78. 50-73. 10.1177/0020852311429667.

“Let's Do This!” Let's Do This! | DoSomething.org. Accessed December 4, 2019. https://www.dosomething.org/us.

“Node.js Web Application Framework.” Express, December 4, 2019. https://expressjs.com/.

Sam, Edelstein. Interview by Edward Deaver. Personal interview. Syracuse,December 3, 2019.

Scanlime. “Scanlime/Fadecandy.” GitHub, January 6, 2019. https://github.com/scanlime/fadecandy.

"Processing." Visualising Information for Advocacy. Accessed December 1, 2019. https://visualisingadvocacy.org/node/725.html.

"SM Connector." JST SM Connector. Adafruit. Accessed December 5, 2019. https://cdn-shop.adafruit.com/datasheets/JSTSM.pdf.

"What Is Open Source?" Opensource.com. Accessed December 4, 2019. https://opensource.com/resources/what-open-source.

Wolff, Martijn, and W E Biernatzki. "The Social and Cultural Effects of Advertising ." *Communication Research Trends* 14, no. 1 (1994): 34–34. http://cscc.scu.edu/trends/v14/V14_1.pdf.