Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2021

Stream Hoppers
The Stream Hopper

Seth Jaksik
Justin Erdmann
Kevin Chawla
Dominic Kotzer
Alexander Isaula

Revision History

Revision	Date	Author(s)	Description
0.1	2.01.2021	SJ	document creation
0.2	2.15.2021	JE, KC, SJ, DK, AI	complete draft

CONTENTS

List of Figures

1 Introduction

The Stream Hopper is a customizable IOT Hub Device to enhance viewer's experience and streamer's interaction with their viewers. The IOT Hub can be customized to have unique events to the streamer's preference. The IOT Hub has three different types of connections to supported devices: USB, GPIO, and WiFi connections. The IOT Hub will support various devices with different connection types from Twitch or StreamLabs. The Stream Hopper will provide "in-real-life" notifications for streamers based on stream events. The intended user audience of the Stream Hopper is any streamer who wants to grow their stream by conveniently increasing the entertainment value of their stream.

The Stream Hopper will be an IOT Hub that connects to the user's Twitch and reads live data from their stream through the Twitch and Streamlabs API. The user will be able to select the events to trigger their devices and they will be able to add their own custom devices.

2 System Overview

Our device will be a universal hub where multiple IOT devices can connect and interface with the Twitch website and activate on various Twitch events. First, an event will happen on the userâs Twitch channel, such as a follow, donation or subscription, and through the API, a message will be sent to the Raspberry Pi. Based on the message from Twitch, an event will be triggered through the General Purpose Input/Output pins on the Pi or via the internet to an IOT device. The user will be able to customize these devices through a Graphical User Interface and they will be able to add or delete their own unique electronic devices. In addition to customizing the devices, the user will be able to decide what events will trigger their devices.

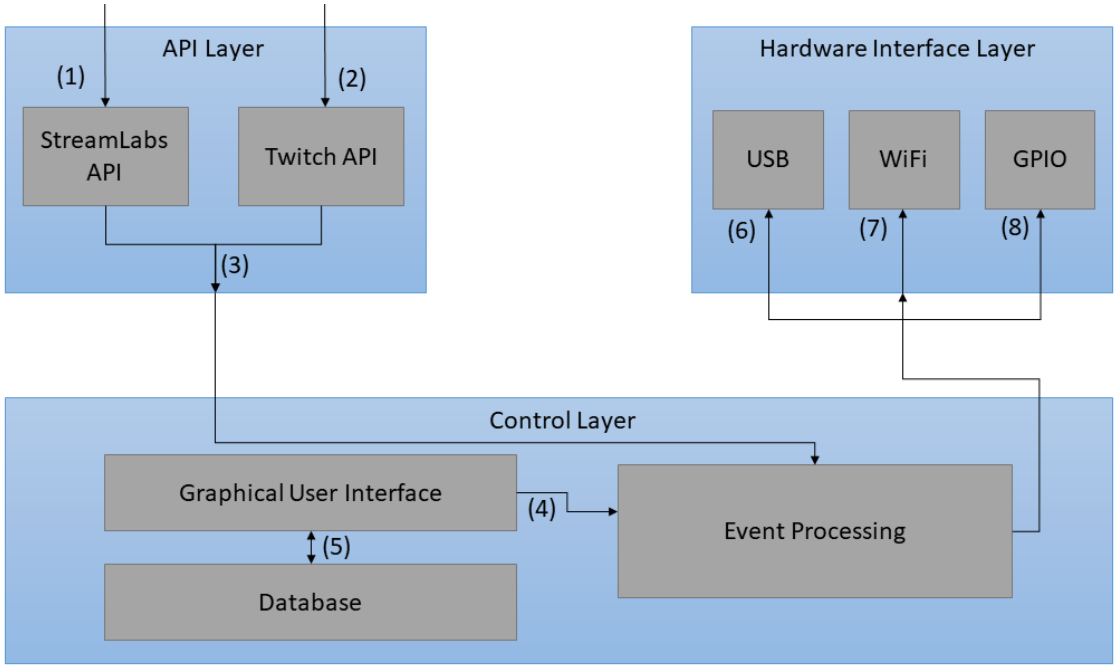

Figure 1: System architecture

3 API Layer Subsystems

The API Layer contains the Streamlabs API and Twitch API subsystems. These systems interact with their respective API interfaces to get the event data that will be used as input for the Control Layer.

3.1 Layer Hardware

The Raspberry Pi 4B is the main hardware component of the Stream Hopper, in which all software processes will be run from.

3.2 Layer Operating System

The API Layer will be running on Raspbian 5.4.

3.3 Layer Software Dependencies

The layer will be dependent on many common JavaScript libraries to support proper interaction with the Control Layer.

3.4 Streamlabs API Subsystem

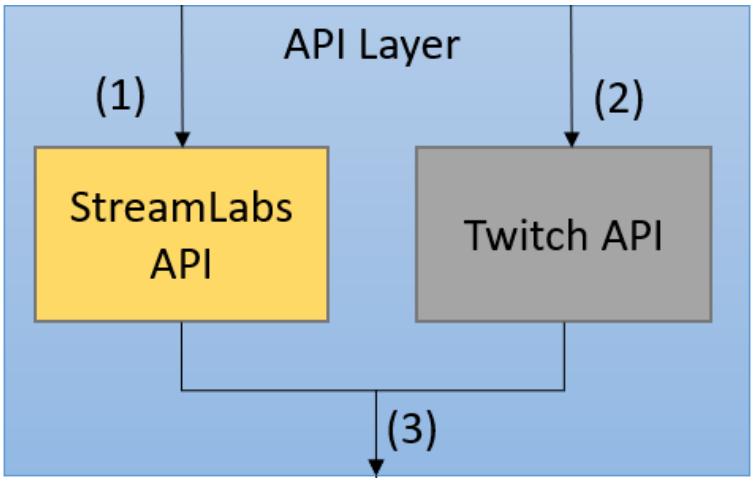

Figure 2: API Subsystem diagram

3.4.1 Subsystem Software Dependencies

The Streamlabs API Subsystem will require a JSON-RPC library for NodeJS to send and receive information from the Streamlabs servers, as well as the Streamlabs API wrapper.

3.4.2 Subsystem Programming Languages

The Streamlabs API Subsystem will be written in NodeJS.

3.4.3 Subsystem Data Structures

The Streamlabs API will use a FIFO to recieve the incoming API calls, which will then be processed by the Data Processing subsystem, and then placed into an outgoing FIFO to be sent off to the Control Layer.

3.4.4 Subsystem Data Processing

The Streamlabs API Layer will receive an API response from the StreamLabs API and extract the important data for the particular request. Then the Subsystem will package the Data as a JSON object and send it to the Control Layer to be processed and executed.

3.5 Twitch API Subsystem

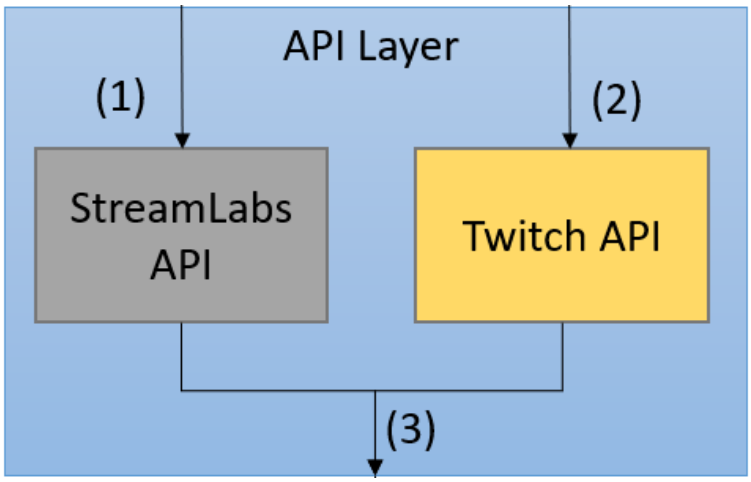

Figure 3: Example subsystem description diagram

3.5.1 Subsystem Software Dependencies

The Twitch API Subsystem will require a JSON-RPC library for NodeJS to send and receive information from the Twitch servers, as well as the Twitch API wrapper.

3.5.2 Subsystem Programming Languages

The Twitch API Subsystem will be written in NodeJS.

3.5.3 Subsystem Data Structures

The Twitch API will use a FIFO to recieve the incoming API calls, which will then be processed by the Data Processing subsystem, and then placed into an outgoing FIFO to be sent off to the Control Layer.

3.5.4 Subsystem Data Processing

The Twitch API Layer will receive an API response from the Twitch API and extract the important data for the particular request. Then the Subsystem will package the Data as a JSON object and send it to the Control Layer to be processed and executed.

4 Control Layer Subsystems

4.1 Layer Hardware

The Raspberry Pi 4B is the main hardware component of the Stream Hopper, in which all software processes will be run from.

4.2 Layer Operating System

The Control Layer will be running on Raspbian 5.4.

4.3 Layer Software Dependencies

The layer will be dependent on the Streamlabs and Twitch.tv APIs in order to communicate with the rest of the Layers. The Control Layer will also be dependent on many common JavaScript libraries in order to represent common control flow of the program.

4.4 Graphical User Interface Subsystem

The Graphical User Interface layer provides the user the tool needed to create the combination of devices and event with trigger according to the appropriate needs. Also, here the user has the option to create presets for different needs.The GUI Subsystem will be a web application.

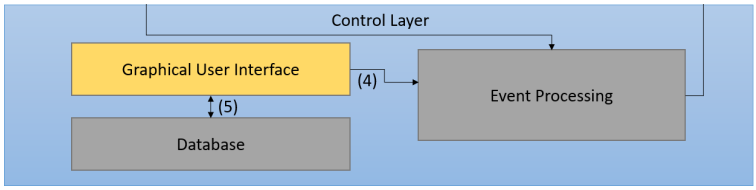

Figure 4: Control Layer Subsystem diagram

4.4.1 Subsystem Hardware

The GUI will be using the Raspberry Pi touchscreen hardware to allow the user interactions. It will also run on the Raspberry Pi as described in the Layer.

4.4.2 Subsystem Operating System

The Event Processing Subsystem operating system is the same as the Layer Operating System

4.4.3 Subsystem Software Dependencies

This system will be designed using the React framework of JavaScript. The subsystem will require HTML5 compatibility browser that can support the cdn dependencies in the system including bootstrap4, Jquery and CSS5. The React framework will follow the ECMAScript 6 standard.

4.4.4 Subsystem Programming Languages

GUI is programmed using HTML, CSS, JavaScript, Bootstrap and Jquery

4.4.5 Subsystem Data Structures

The Devices, presets and triggers configurations will be stored in tables and that data will be transferred/stored in the MySQL database for the user to refernce later and to make sure that the current configuration is not lost. The tables will be stored in the JSON format.

4.4.6 Subsystem Data Processing

The data processing will be through the GUI will be done using REST API and HTTP protocol.

4.5 Event Processing Subsystem

The Event Processing Subsystem will control the overall dataflow of the Stream Hoper IOT device. The subsystem will receive configuration details from the GUI subsystem and store them for later retreival and comparison. On receiving a message from the API layer, the subsystem will determine if the message received matches any of the triggers set by the GUI subsytem, and send the respective hardware triggers off to the Hardware Layer.

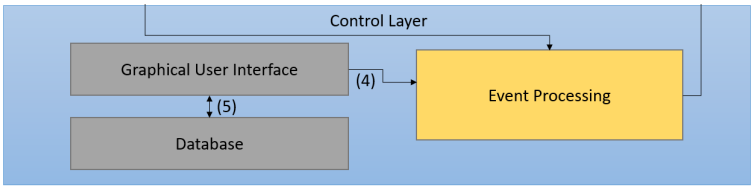

Figure 5: Event Processing Subsystem diagram

4.5.1 Subsystem Operating System

The Event Processing Subsystem operating system is the same as the Layer Operating System.

4.5.2 Subsystem Software Dependencies

The Event Processing Subsystem will depend on the API we are using to read in and interpret the Twitch.tv and Stream Labs APIs as well as the JSON format that will be needed by the Hardware Layer subsystems to determine which hardware devices to send messages to.

4.5.3 Subsystem Programming Languages

The Event Processing Subsystem will be using Javascript.

4.5.4 Subsystem Data Structures

A description of any classes or other data structures that are worth discussing for the subsystem. For example, data being transmitted from a micro-controller to a PC via USB should be first be assembled into packets. What is the structure of the packets?

4.5.5 Subsystem Data Processing

The subsystem will receive data packets from the API layer containing the API calls the device will need to trigger on. These will be in JSON format that will be slightly restructured from the Twitch.tv and Streamlabs APIs to ensure they are compatible together. The subsystem will also send formatted JSON data packets to the Hardware layer in order for the devices to be triggered correctly. The subsystem will receive JSONs from the GUI that contain the current informatin on devices, presets, and triggers being used. These will be stored in lists and referenced using ID numbers in order to identify each packed individually.

4.6 Database Subsystem

The Database Subsystem will contain the database with all the user's preferences and presets. This database will be run directly on the Raspberry Pi and the user will be able to add, delete and edit database items from the physical touch-screen GUI or the online GUI.

4.6.1 Subsystem Hardware

The Database Subsystem hardware is the same as the Layer Hardware.

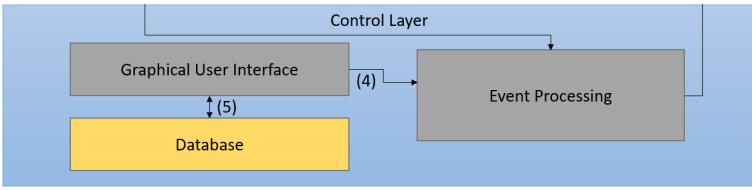

Figure 6: Database Subsystem diagram

4.6.2 Subsystem Operating System

The Database Susbsystem Operating System is the same as the Layer Operating System.

4.6.3 Subsystem Software Dependencies

The MySQL-Client package will be utilized for Raspbian OS and the Pi will programmatically make queries to the Database via Python.

4.6.4 Subsystem Programming Languages

The Database Subsystem will use MySQL to maintain and query the database and the Raspberry Pi will use Python to connect and communicate with the database.

4.6.5 Subsystem Data Structures

The Database will be queried via MySQL commands and send responses as JSON objects. The Raspberry Pi will then parse these commands via Python.

4.6.6 Subsystem Data Processing

The Database subsystem will receive commands and queries from the Control Layer via JSON objects. Then, the Database Layer will parse these messages and distribute commands to the database via Python.

5 Hardware Layer Subsystems

The Hardware Interface Layer will consist of all the Physical Hardware Devices that produce visual/audio stimuli for the audience. These devices will be connected to the StreamHopper via WiFi, USB and GPIO pins on the Raspberry Pi. The WiFi devices will typically be used via their respective APIs and the USB and GPIO devices will typically be triggered via on/off electrical signals from the Raspberry Pi.

5.1 Layer Hardware

The Raspberry Pi 4B is the main hardware component of the StreamHopper, in which all software processes will be run from. Starting from the initial input from the mutiple APIs, the Raspberry Pi will take in the input and use the information to enact the user selected response and add the selected configuration to the database. Other hardware components are the GPIO pins of the Raspberry Pi which will send electrical signals to the connected device to create a desired response. In addition, the USB connections will be used to connect to other devices such as speakers to create user selected responses. Lastly the final component of the hardware layer is the WiFi component in which communication between connected devices will be through the internet and router.

5.2 Layer Operating System

Raspbian 5.4 will be used on the Raspberry Pi 4B to configure and apply the new StreamHopper application as well as able to utilize the GPIO, USB, and WiFi components. The components however do not require an operating system.

5.3 Layer Software Dependencies

To control GPIO pins of the Raspberry Pi, we will use the Python programming language and the GPIO Zero library to control GPIO pins. To communicate through USB with the Python programming language we will use the uhubctl library. Finally for the WiFi communication, we will use socket module from Python for communication.

5.4 USB Subsystem

The USB Subsystem will consist of all the devices connected to the StreamHopper via USB. These devices will connect directly to the Raspberry Pi via the Pi's onboard USB ports. They will be triggered via on/off electrical signals on Twitch/StreamLabs events that the user has chosen.

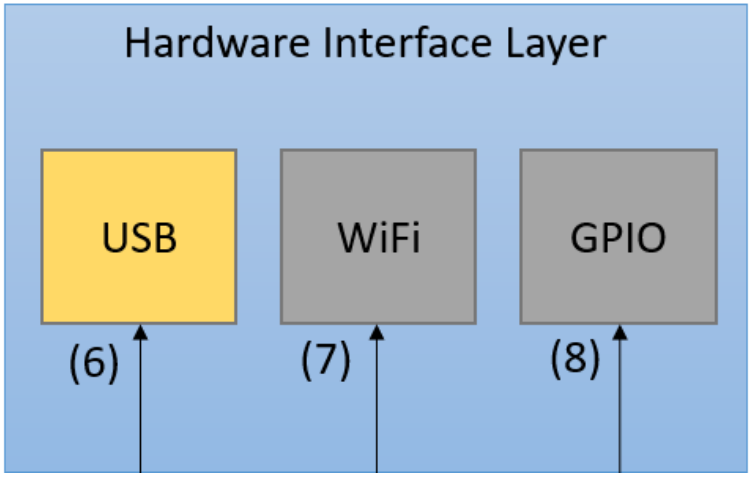

Figure 7: USB Subsystem diagram

5.4.1 Subsystem Hardware

The main hardware components of the USB Subsytem will be any device chosen by the user that is operated via USB. The user will be able to customize the StreamHopper's USB ports and can configure any USB powered device to any USB port on the Raspberry Pi.

5.4.2 Subsystem Operating System

The USB Devices will be run via on/off electrical signals from the Raspberry Pi. The devices will be operated via the device manufacturer's specifications and may or may not have an Operating System. The electrical signals will be controlled by the Raspberry Pi via the Raspbian OS.

5.4.3 Subsystem Software Dependencies

The USB devices will be triggered via signals from the Twitch/Streamlabs API via the Raspberry Pi. These devices do not have any software dependencies by themselves however the electrical signals from the Pi will be controlled using the 'uhubctl' utility.

5.4.4 Subsystem Programming Languages

The USB devices will be run using the manufacturer's specifications and will not have any software running on them that is operated by the StreamHopper and, therefore, will not be utilizing any programming languages. The Raspberry Pi will utilize Python to call the 'uhubctl' utility.

5.4.5 Subsystem Data Structures

The USB subsystem will be run via the onboard USB ports on the Raspberry Pi. Therefore they will not be utilizing any data structures as the individual devices will be run via the manufacturer's specifications.

5.4.6 Subsystem Data Processing

The USB devices will not be using any software and will only receive an on/off electrical signal from the Raspberry Pi's USB ports so there will be no data processing in the USB Subsystem.

5.5 WiFi Subsystem

The WiFi Subsystem will consist of all the WiFi connected devices that receive commands from the Raspberry Pi. A router will be configured and all the WiFi devices will be connected to the Raspberry Pi via the local router. These devices will receive commands from the router via data packets communicated over the internet.

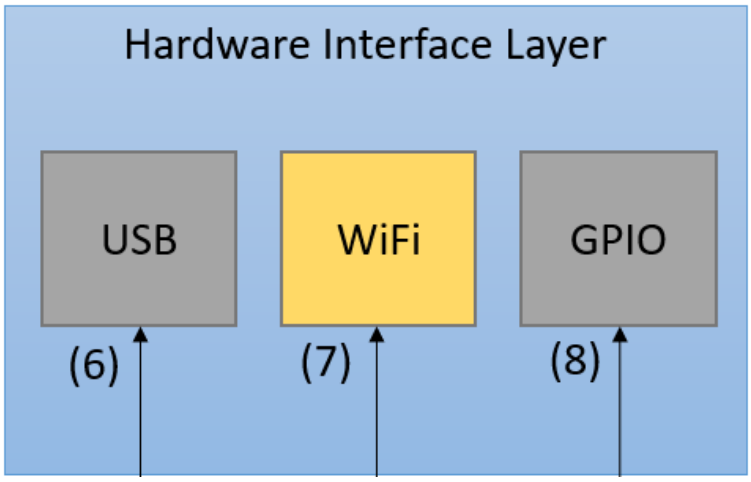

Figure 8: WiFi Subsystem diagram

5.5.1 Subsystem Hardware

The Subsystem will contain WiFi-controlled devices with an open API. In particular, we will provide the user with preconfigured versions of the Lifx Color Bulb, Lifx LED Strip and Weemo Smart Plug. In addition, a WiFi router will also exist in this subsystem to allow communication between the Raspberry Pi and the WiFi devices.

5.5.2 Subsystem Operating System

The WiFi devices will not have an operating system as they will run via the device manufacturer's specification. The router will also be run according to the device manufacturer's specifications and may or may not have an operating system.

5.5.3 Subsystem Software Dependencies

The WiFi subsystem will be run via the 'socket' package available for Python. The individual WiFi devices will be given commands via their respective APIs.

5.5.4 Subsystem Programming Languages

As mentioned previously, the WiFi subsystem will utilize Python to communicate via an internet connection.

5.5.5 Subsystem Data Structures

The 'socket' module for Python will be used to send data. This data will be sent via the UDP Internet Protocol and the data will contain API calls to be run by the WiFi devices.

5.5.6 Subsystem Data Processing

The subsystem will utilize common UDP/IP practices which can be found in the 'socket' module for Python. socket.connect() will be used to establish a connection between the Raspberry Pi and the WiFi device. Then socket.send() will be used to send the data from the Raspberry Pi to the WiFi device via the router. This data will be sent according to the LIFX LAN Protocol and will contain an API command for the device to run.

5.6 GPIO Subsystem

The GPIO Subsystem will consist of all the devices connected to the StreamHopper via GPIO Pins. These devices will connect directly to the Raspberry Pi via the Pi's onboard GPIO ports. They will be triggered via on/off electrical signals on Twitch/StreamLabs events that the user has chosen.

5.6.1 Subsystem Hardware

The main hardware components of the GPIO Subsytem will be any device chosen by the user that is operated via Raspberry Pi GPIO. The user will be able to customize the StreamHopper's GPIO pins and can configure any GPIO powered device to any GPIO pin(s) on the Raspberry Pi.

5.6.2 Subsystem Operating System

The GPIO Devices will be run via on/off electrical signals from the Raspberry Pi. The devices will be operated via the device manufacturer's specifications and may or may not have an Operating System. The GPIO pins will be turned on and off via the Raspberry Pi using the Raspbian OS.

5.6.3 Subsystem Software Dependencies

The GPIO devices will be triggered via signals from the Twitch/Streamlabs API via the Raspberry Pi. These devices do not have any software dependencies by themselves. The Raspberry Pi will utilize the 'gpiozero' module in Python to interface with the GPIO pins.

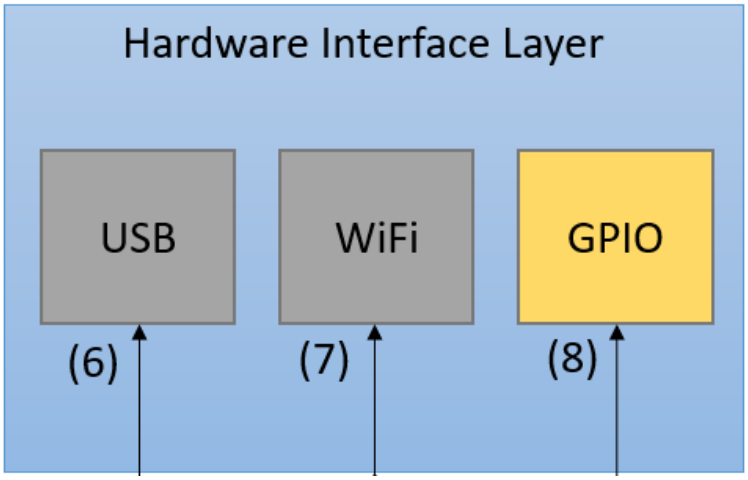

Figure 9: GPIO Subsystem diagram

5.6.4 Subsystem Programming Languages

The GPIO devices will be run using the manufacturer's specifications and will not have any software running on them that is operated by the StreamHopper and, therefore, will not be utilizing any programming languages. The Raspberry Pi will utilize Python to turn the GPIO pins on and off.

5.6.5 Subsystem Data Structures

The GPIO subsystem will be run via the onboard GPIO pins on the Raspberry Pi. Therefore they will not be utilizing any data structures as the individual devices will be run via the manufacturer's specifications.

5.6.6 Subsystem Data Processing

The GPIO devices will not be using any software and will only receive an on/off electrical signal from the Raspberry Pi's GPIO pins so there will be no data processing in the GPIO Subsystem.