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

Architectural Design Specification
CSE 4316: Senior Design I
Fall 2020

Stream Hoppers
The Stream Hopper

Seth Jaksik
Justin Erdmann
Kevin Chawla
Dominic Kotzer
Alexander Isaula

Revision History

Revision	Date	Author(s)	Description
0.1	10.30.2020	KC	Document Creation
0.2	11.13.2020	SJ, JE, DK, AI, KC	Complete Draft 1
1.0	11.15.2020	SJ, JE, DK, AI, KC	Release 1

CONTENTS

List of Figures

List of Tables

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.

1.1 Purpose and Use

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.

1.2 Intended Audience

The Stream Hopper is intended to be used by streamers who want to grow their channel and add entertainment value to their stream. If this product were sold commercially, it would be popular among Twitch Streamers who want an active viewing experience for their viewers. In addition the Stream Hopper is intended to be used by individuals who are technologically competent as it will be much more customizable and useful for these individuals.

2 System Overview

The Stream Hopper IOT Hub will be divided into three separate systems that will communicate with each other to trigger IOT Devices on Twitch and Streamlabs events. The API Layer will query the Twitch and Streamlabs API and retrieve relevant live data. The Control Layer will receive the data from the API Layer and trigger the IOT Devices based on presets obtained from the user via the Graphical User Interface. The Hardware Interface Layer will receive commands from the Control Layer and will activate the IOT Devices to produce live Audio and Visual stimulus.

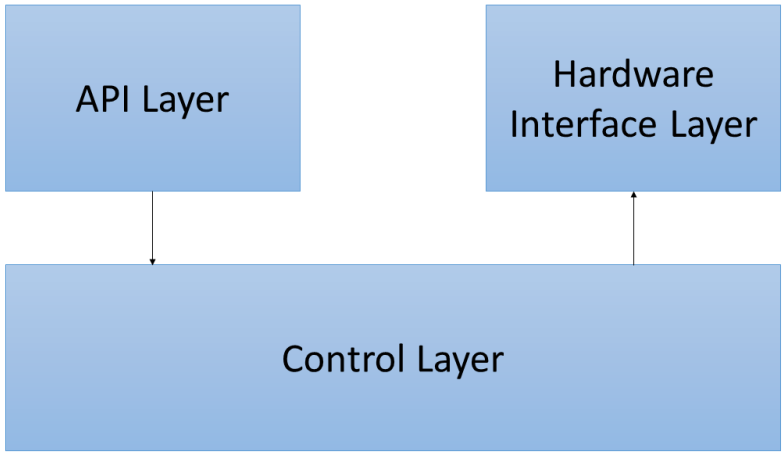

Figure 1: Architectural Layer Diagram

2.1 API Layer Description

The API Layer will communicate directly with Twitch and Streamlabs in order to retrieve all relevant live data used to trigger the IOT Devices. This data will arrive as JSON objects from the Twitch and Streamlabs servers and will be parsed into the necessary data type according to the information received. For example, if the data received is the number of followers the user currently has, then the data will be parsed into an integer. The data will then be sent to the Control Layer.

2.2 Control Layer Description

The Control Layer will be the main processing layer for the Stream Hopper IOT Device. It will receive data from the API Layer and query the Graphical User Interface for the Current System Preset, Trigger and IOT Device. It will then create the execution request and will send the data to the Hardware Interface Layer via WiFi, GPIO or USB.

2.3 Hardware Interface Layer Description

The Hardware Interface Layer will contain all the IOT Devices. These devices will be connected to the IOT Hub via WiFi, GPIO or USB and this layer will receive data from the Control Layer. The data will arrive as execution requests specific to an IOT Device. These requests will be executed by the IOT Devices and then the devices will be ready for a new request from the Control Layer.

3 Subsystem Definitions & Data Flow

Beginning from the API Layer, events from Twitch or StreamLabs are received and sent to to their associated API Layer Subsystem both leading to the Event Processing Subsystem. Meanwhile from the Graphical User Interface Subsystem the user may request an action be performed, which leads to the request being sent to the Event Processing Subsystem and the saving of the request for future reference in the Database Subsystem. From the Event Processing Subsystem the request is sent to the Hardware Interface Layer. Additionally the Event Processing Subsystem handles and determines whether the USB, GPIO, or WiFi Subsystem executes the request on the associated connected device.

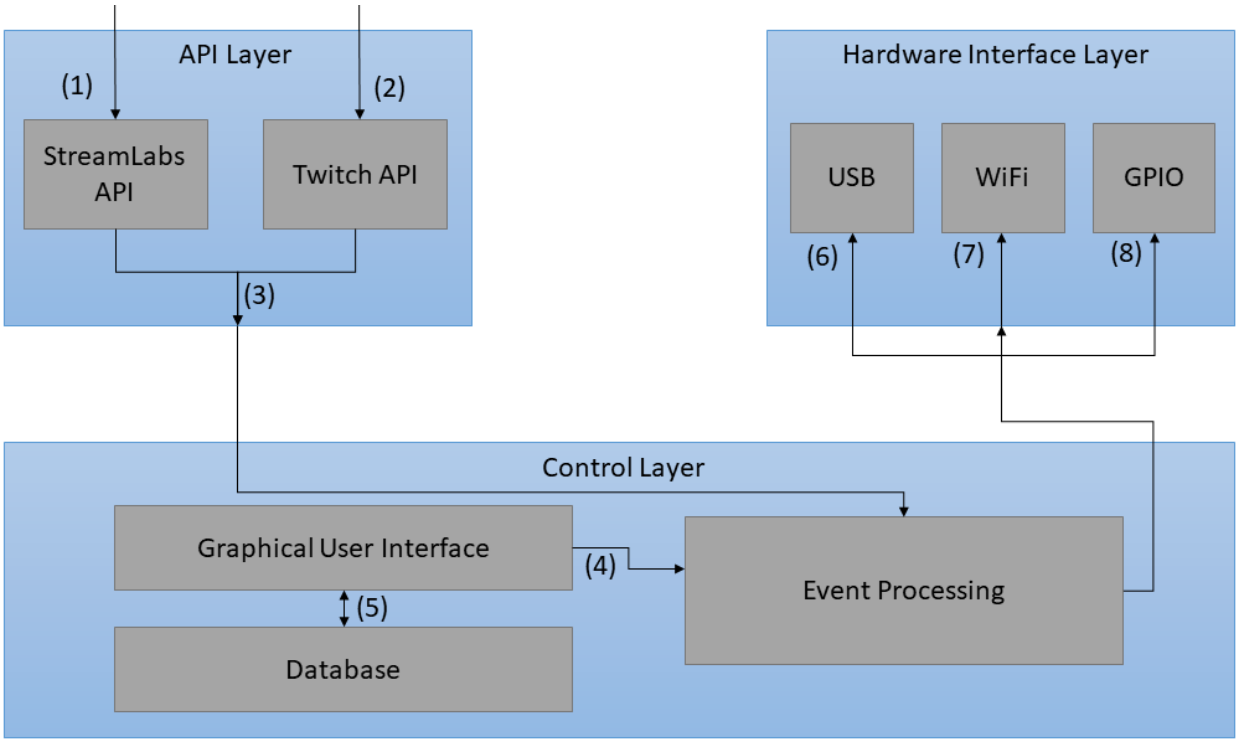

Figure 2: Data Flow Diagram

4 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.

4.1 Streamlabs API Subsystem

This subsystem takes information from the Streamlabs API service. It filters the information retrieved into a useful form for the Event Processing Subsystem in the Control Layer.

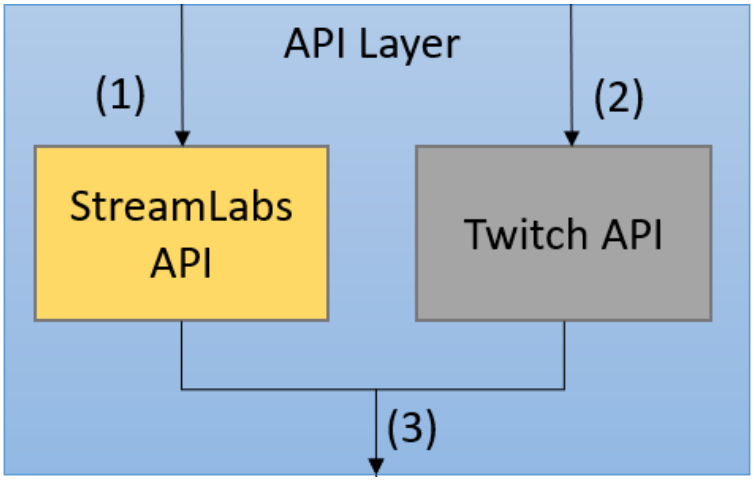

Figure 3: API Layer Subsystems - Streamlabs

4.1.1 Assumptions

The Streamlabs API subsystem will assume that the device is connected to the internet and can send the required requests and receive their responses over the network.

4.1.2 Responsibilities

The Streamlabs API subsystem should communicate with Streamlabs in order to retrieve the live data for use in the Control Layer. It should be parsed into a consistent format based on the type of information received when sent through the Data Formatting Interface to the Control Layer. It should actively communicate at a rate based on recommendations from the Streamlabs API documentation to keep up to date information being fed to the Control Layer.

4.1.3 Subsystem Interfaces

Table 2: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Streamlabs API OAuth 2 Application	JSON-RPC	Server Response
#03	Data Formatting Interface	Raw Data	Formatted Data

4.2 Twitch API Subsystem

This subsystem takes information from the Twitch API service. It filters the information retrieved into a useful form for the Event Processing Subsystem in the Control Layer.

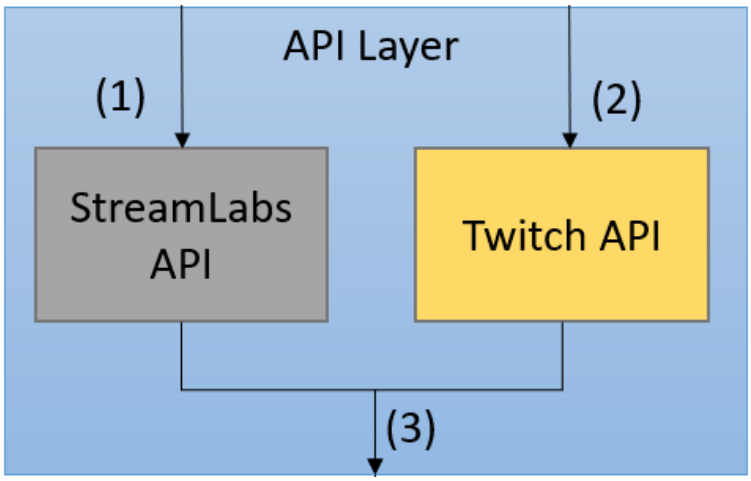

Figure 4: API Layer Subsystems - Twitch

4.2.1 Assumptions

The Twitch API subsystem will assume that the device is connected to the internet and can send the required requests and receive their responses over the network.

4.2.2 Responsibilities

The Twitch API subsystem should communicate with Twitch in order to retrieve the live data for use in the Control Layer. It should be parsed into a consistent format based on the type of information received when sent through the Data Formatting Interface to the Control Layer. It should actively communicate at a rate based on recommendations from the Twitch API documentation to keep up to date information being fed to the Control Layer.

4.2.3 Subsystem Interfaces

Table 3: Subsystem interfaces

ID	Description	Inputs	Outputs
#02	Twitch API OAuth 2 Application	API Request	Server Response
#03	Data Formatting Interface	Raw Data	Formatted Data

5 Hardware Interface Layer Subsystems

The Hardware Interfaces Layer contains the WiFi, General Purpose Input/Output and USB subsystems. These systems are the endpoint of our data flow and the visual and audio stimulus are produced in this layer. Each subsystem contains a certain type of IOT device and these devices are triggered from the Control Layer.

5.1 WiFi Devices Subsystem

The WiFi subsystem contains all WiFi IOT Devices and it receives data from the Control Layer for various Twitch and Streamlabs Events. These devices will be triggered via WiFi through our local WiFi router.

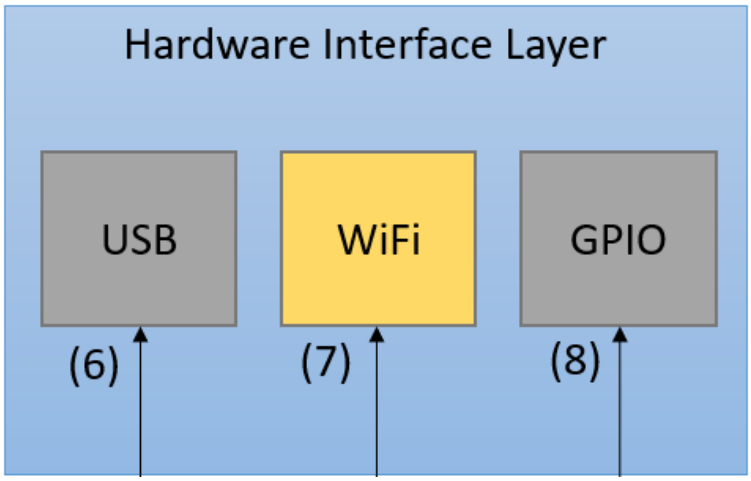

Figure 5: Hardware Interface Layer Subsystems - WiFi

5.1.1 Assumptions

The WiFi subsystem will assume that the Control Layer sends a properly formatted HTTP request. In addition, the WiFi subsystem will assume that the WiFi IOT Devices are connected to the local WiFi router and the IOT Devices have an open source API that will properly process trigger requests.

5.1.2 Responsibilities

The WiFi subsystem should process any trigger request via WiFi and execute that request in a timely manner. When receiving a trigger request, the WiFi subsystem should ensure that it received a properly formatted request. This request should be a JSON Object and it should include the type of WiFi device to be triggered and the action the WiFi device should execute. It should then distribute that request to the proper IOT device. Once the request arrives at the IOT device, the device should execute the request and then standby for future requests from the Control Layer.

5.1.3 WiFi Subsystem Interfaces

Table 4: WiFi Subsystem interfaces

ID	Description	Inputs	Outputs
#07	WiFi Trigger Request	Trigger Request	Visual/Audio Stimulus From Device

5.2 USB Devices Subsystem

The USB subsystem contains all USB Devices and it receives data from the Control Layer for various Twitch and Streamlabs Events. These devices will be triggered via USB directly from the Control Layer.

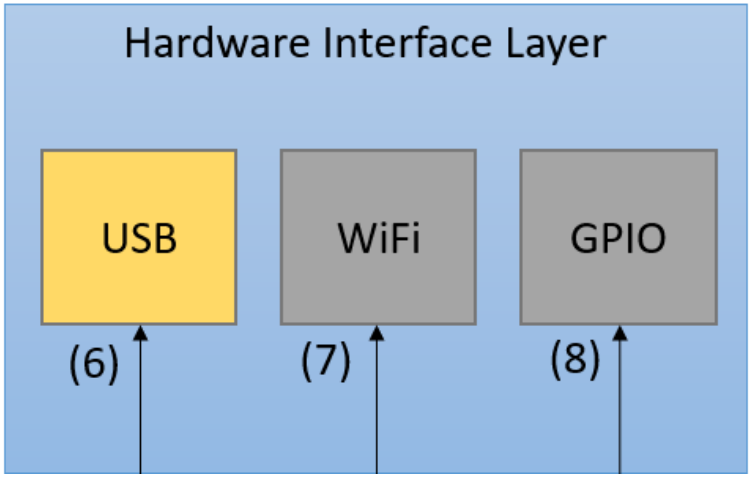

Figure 6: Hardware Interfaces Layer - USB Devices Subsystem

5.2.1 Assumptions

The USB subsystem will assume that the Control Layer sends a properly formatted request. In addition, the USB subsystem will assume that the USB Devices are properly connected to the IOT Hub and the USB devices can be properly triggered via a USB connection either through an open source API or on/off commands.

5.2.2 Responsibilities

The USB subsystem should process any trigger request via USB and execute that request in a timely manner. When receiving a trigger request, the USB subsystem should ensure that it received a properly formatted request. This request should be a JSON Object and it should include the type of USB device to be triggered and the action the USB device should execute. This request could also be an on/off request where the device simply turns on and then turns off after a period of time. This type of request will typically be used for devices such as a Smart Plug or a Smart Light Bulb. Once the request is processed and determined to be valid, it will be sent directly to the proper USB Device via a USB Connection. The request will then be executed and the USB Device should standby for future requests.

5.2.3 USB Devices Subsystem Interfaces

Table 5: USB Subsystem interfaces

ID	Description	Inputs	Outputs
#06	USB Device Trigger Request	Trigger Request	Visual/Audio Stimulus From Device

5.3 GPIO Subsystem

GPIO Subsystem contains any GPIO port connected devices and it receives data from the Control Layer for various Twitch and Streamlabs Events. These devices will be triggered via GPIO ports directly from the Control Layer.

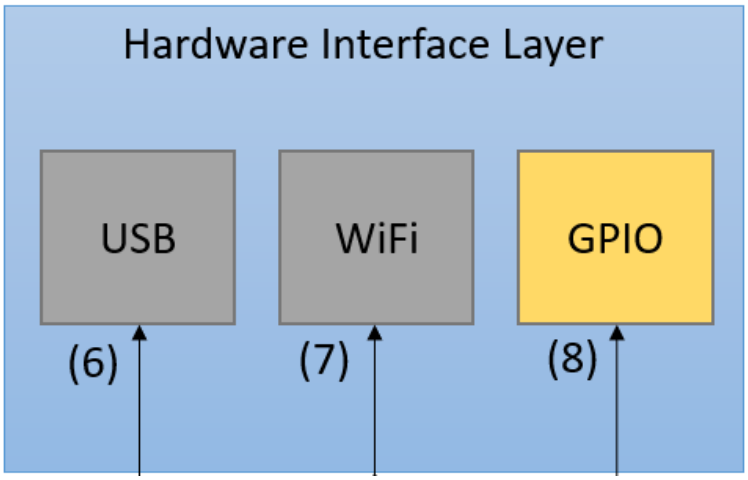

Figure 7: Hardware Interface Layer Subsystems - GPIO

5.3.1 Assumptions

The GPIO subsystem will assume that the Control Layer sends a properly formatted request. In addition, the GPIO subsystem will assume that the GPIO Devices are connected properly to the Stream Hopper either by soldering, external wires, or with the use of adaptors. Additionally the GPIO subsytem will assume the GPIO devices can be triggered by GPIO ports.

5.3.2 Responsibilities

The GPIO subsystem must handle requests from the event processing subsystem and be able to execute them in an expected or reasonable time. This request should be a JSON Object and it should include the type of GPIO device to be triggered and the action the GPIO device should execute. Once a request is received it will identify and execute the request to the proper GPIO device and wait for future requests from the control layer.

5.3.3 SUBSYSTEM INTERFACES

Table 6: GPIO Subsystem interfaces

ID	Description	Inputs	Outputs
#07	GPIO Device Trigger Request	Trigger Request	Visual/Audio Stimulus From Device

6 Control Layer Subsystems

In the Control Layer System, both the Graphical User Interface and the Event Processing of the Stream Hoppers occur. In the Graphical User Interface subsystem, the user will be able to control the triggers for the Event Processing subsystem by creating associations between Events and the devices they will trigger. The Event Processing will receive events from the API layer and then send events to the Hardware Interface layer to trigger specific devices based off the settings configured by the Graphical User Interface

6.1 Graphical User Interface

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.

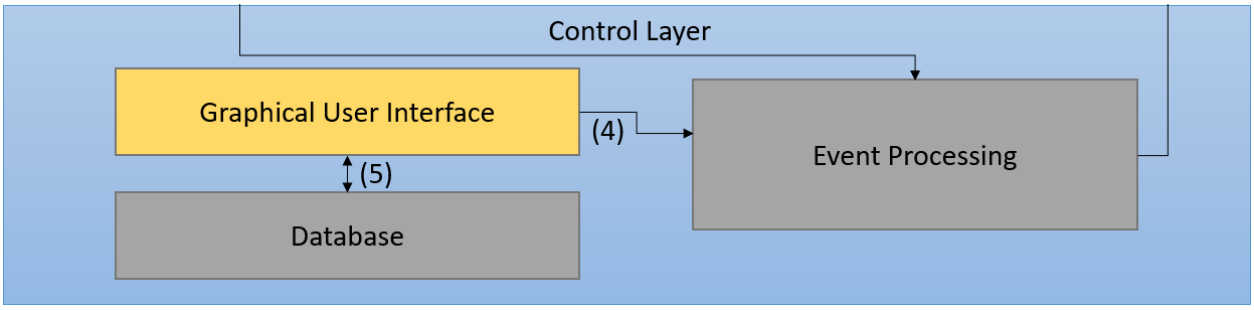

Figure 8: Control Layer Subsystems - Graphical User Interface

6.1.1 Assumptions

The GUI layer assumes that it will receive response from the user and then send it to the event processing layer and store the appropriate information required by the user in the database created.

6.1.2 Responsibilities

The GUI layer is responsible for receiving user input and then processing that into a JSON format to send it to the event processing layer. Additionally, it is responsible to provide the user the functionality to add and make changes according to their own needs and being as flexible as possible with a great experience for their stream.

6.1.3 Graphical User Interface Interfaces

Table 7: Graphical User Interface interfaces

ID	Description	Inputs	Outputs
#4	User Set Triggers	N/A	JSON objects
#5	User Set Triggers, Presets, and Devices	JSON objects	JSON objects

6.2 Event Processing

The Event Processing subsystem consists of the main logic of the Stream Hopper. This subsystem outputs commands to the Hardware Interface layer based off of events received from the API layer.

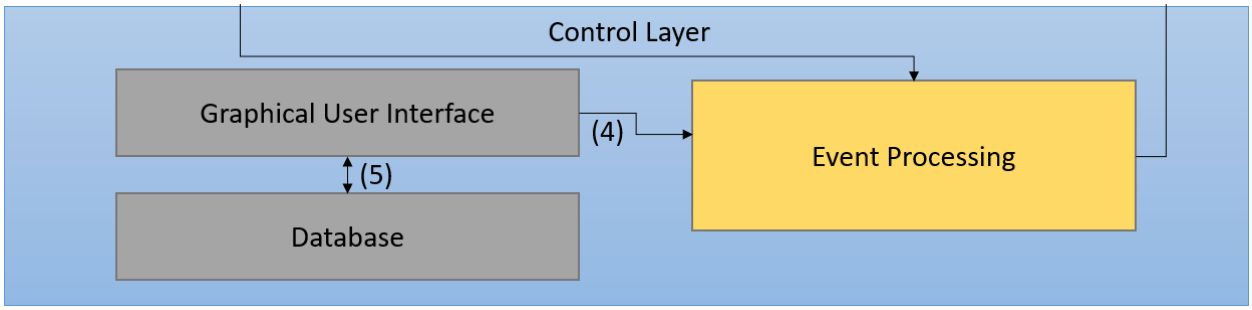

Figure 9: Control Layer Subsystems - Event Processing

6.2.1 Assumptions

The Event Processing subsystem assumes that the API Layer will supply it with events in JSON format and will include the event type as well as any accompanying information from the event such as username, message, or other pieces of information if applicable. The Event Processing subsystem also assumes that it will receive the requested responses from the Graphical User Interface subsystem in order to respond to incoming events how the user wants.

6.2.2 Responsibilities

The Event Processing layer is responsible for outputting messages to the Hardware Interface layer based off of the event and device combinations set in the Graphical User Interface layer. The layer will take input from the API layer in the form of a JSON message. These messages will include the event that occurred, as well as information pertaining to the event. The Event Processing layer then compares the received event to the triggers it has been given from the Graphical User Interface layer. If there is one or more triggers that have been set for that specific event, the Event Processing layer will then send a message to the Hardware Interface layer in the form of a JSON that includes the device to be triggered as well as other information pertaining to the device settings such as light color, message to display, or sounds to play.

6.2.3 Event Processing Interfaces

Table 8: Event Processing interfaces

ID	Description	Inputs	Outputs
#3	API Layer interface	API Event	N/A
#4	User Set Triggers	JSON object	N/A
#6	USB Trigger Message	N/A	JSON object
#7	WiFi Trigger Message	N/A	JSON object
#8	GPIO Trigger Message	N/A	JSON object

6.3 Database

The Database subsystem is the main storage for the Stream Hopper. This subsystem will store the presets, devices, and triggers created by the user in the Graphical User Interface.

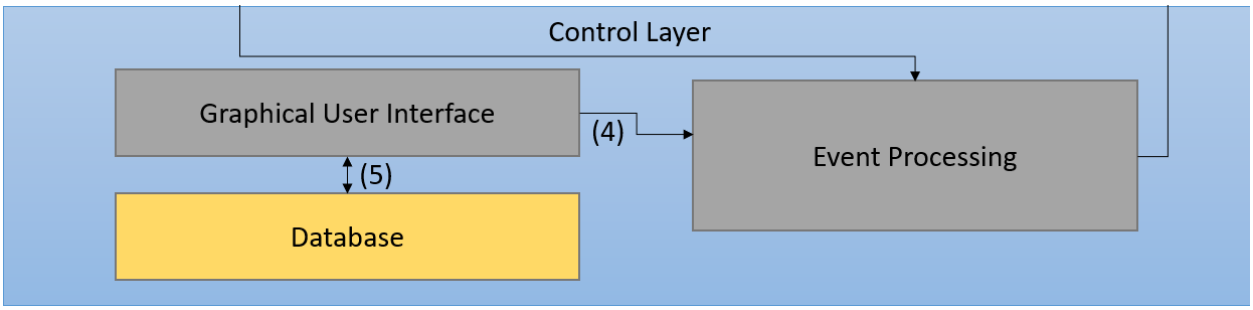

Figure 10: Control Layer Subsystems - Database

6.3.1 Assumptions

The database assumes that the information on the presets, devices, and triggers will be given in JSON objects. The database also assumes that there is enough storage information to hold the needed items in the database.

6.3.2 Responsibilities

The database will be responsible for storing the presets, devices, and triggers created by the user in the Graphical User Interface. The database will also restore the current state of the Graphical User Interface on boot of the Stream Hopper. The database will also give the requested information for loading new presets and triggers into the Graphical User Interface when switching between the current preset and a different one.

6.3.3 Database Interfaces

Table 9: Database interfaces

ID	Description	Inputs	Outputs
#5	User Set Triggers, Presets, and Devices	JSON objects	JSON objects

References