1. Install Required Library

Install requests if it’s not already installed.

pip install requests

2. Set Up

Get your Hue Bridge IP address. We’ll use 192.168.8.238 in the example code

Generate a user token by pressing the Hue Bridge's physical button and making an authenticated API request.

Here is the code to run to make an authenticated API request

AuthenticatedApiControl.py

import requests

import json

bridge\_ip = "192.168.0.238"

url = f"http://{bridge\_ip}/api"

payload = {"devicetype": "my\_hue\_app"}

response = requests.post(url, json=payload)

if response.status\_code == 200:

    print("Response:", response.json())

else:

    print("Failed to create username:", response.status\_code, response.text)

If successful, the response should include a new username in a format like:

[{"success":{"username":"new-username-generated"}}]

Save the information into a file named config.json

config.json

{

    "bridge\_ip": "192.168.0.238",

    "username": "new-username-generated"

}

3. Python Script to Discover Light IDs

HueGetLightSettings.py

import requests

import json

import os

# Get the directory of the current script

script\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

# Construct the full path to config.json

config\_path = os.path.join(script\_dir, "config.json")

# Load configuration from config.json

with open(config\_path, "r") as config\_file:

    config = json.load(config\_file)

bridge\_ip = config["bridge\_ip"]

username = config["username"]

url = f"http://{bridge\_ip}/api/{username}/lights"

response = requests.get(url)

if response.status\_code == 200:

    lights = response.json()

    for light\_id, light\_info in lights.items():

        print(f"Light ID: {light\_id}")

        print(f"Name: {light\_info['name']}")

        print(f"State: {'On' if light\_info['state']['on'] else 'Off'}")

        print(f"Brightness: {light\_info['state']['bri']}")

        print()

else:

    print("Failed to retrieve lights:", response.status\_code, response.text)

This script fetches all lights connected to the Hue Bridge and displays each light’s ID, name, state, and brightness.

Put the information into the HueLights.json file

The order doesn’t matter

HueLights.json

{

    "7": "Living Room 1",

    "9": "Living Room 2",

    "5": "Living Room 3",

    "8": "Living Room 4",

    "6": "Living Room 5",

    "10": "Living Room 6"

}

4. Control The Hue Lights

Review the contents of

Colors.csv

and

Amazon Alexa Color Names and Examples and RGB code (.docx or .htm)

Those files will give you names and example of the available color names.

To view the HTM file in Visual Studio Code:

Install the "HTML Preview" Extension:

Open the Extensions view.

Search for HTML Preview and install it.

Preview the .htm File:

Open your .htm file in VS Code. You will be viewing the .htm code

Press ctrl+shift+v to activate the HTML Preview.

View the Rendered Page:

The preview will appear within VS Code.

A screenshot of a computer

Description automatically generated

You can now run HueControl.py to control the lights.

At the end of the script, specify the light name and the color you wish to change it to.

HueControl.py

import requests

import csv

import os

import json

# Load bridge configuration from config.json

config\_file = os.path.join(os.path.dirname(\_\_file\_\_), "config.json")

with open(config\_file, "r") as file:

    config = json.load(file)

bridge\_ip = config["bridge\_ip"]

username = config["username"]

# Load light IDs and names from HueLights.json

hue\_lights\_file = os.path.join(os.path.dirname(\_\_file\_\_), "HueLights.json")

with open(hue\_lights\_file, "r") as file:

    light\_ids = {v: int(k) for k, v in json.load(file).items()}

# Function to read RGB values from Colors.csv

def get\_rgb\_from\_color\_name(color\_name):

    csv\_file = os.path.join(os.path.dirname(\_\_file\_\_), "Colors.csv")

    with open(csv\_file, mode="r", encoding="utf-8-sig") as file:

        reader = csv.DictReader(file)

        for row in reader:

            if row["Name"].strip().lower() == color\_name.strip().lower():

                return int(row["Red"]), int(row["Green"]), int(row["Blue"])

    return None

# Function to convert RGB to XY (Hue color space)

def rgb\_to\_xy(red, green, blue):

    r, g, b = [x / 255.0 for x in (red, green, blue)]

    r = pow((r + 0.055) / 1.055, 2.4) if r > 0.04045 else r / 12.92

    g = pow((g + 0.055) / 1.055, 2.4) if g > 0.04045 else g / 12.92

    b = pow((b + 0.055) / 1.055, 2.4) if b > 0.04045 else b / 12.92

    x = r \* 0.664511 + g \* 0.154324 + b \* 0.162028

    y = r \* 0.283881 + g \* 0.668433 + b \* 0.047685

    z = r \* 0.000088 + g \* 0.072310 + b \* 0.986039

    return round(x / (x + y + z), 4), round(y / (x + y + z), 4)

# Function to set specified light to specified color

def set\_light\_to\_color(light\_name, color\_name):

    light\_id = light\_ids.get(light\_name)

    if not light\_id:

        print(f"Light '{light\_name}' not found.")

        return

    rgb = get\_rgb\_from\_color\_name(color\_name)

    if not rgb:

        print(f"Color '{color\_name}' not found.")

        return

    xy = rgb\_to\_xy(\*rgb)

    url = f"http://{bridge\_ip}/api/{username}/lights/{light\_id}/state"

    payload = {

        "on": True,

        "xy": xy,

        "bri": 254  # Maximum brightness

    }

    response = requests.put(url, json=payload)

    if response.status\_code == 200:

        print(f"Set '{light\_name}' to '{color\_name}'")

    else:

        print(f"Failed to set '{light\_name}' to '{color\_name}': {response.text}")

# Example usage

set\_light\_to\_color("Living Room 1", "Aqua")