```
import os
import requests
from dotenv import load_dotenv

load_dotenv('credentials.env')

class AutomationBridge:
    """
    Bridge for asnowfix/home-automation logic.
    Supports Home Assistant API and direct Local Discovery.
    """
    def __init__(self):
        self.ha_url = os.getenv("HOME_ASSISTANT_URL") # e.g.,
http://192.168.1.10:8123
        self.ha_token = os.getenv("HOME_ASSISTANT_TOKEN")
        self.headers = {
            "Authorization": f"Bearer {self.ha_token}",
            "content-type": "application/json",
        }

    def toggle_device(self, entity_id, state="toggle"):
        """
        Communicates with the home-automation backend.
        state can be 'on', 'off', or 'toggle'.
        """
        if not self.ha_url or not self.ha_token:
            return "Home Assistant configuration missing, Sir."

        url = f"{self.ha_url}/api/services/light/{state}"
        data = {"entity_id": entity_id}

        try:
            response = requests.post(url, headers=self.headers,
json=data)
            if response.status_code == 200:
                return f"Protocol executed. Device is now {state}."
            return "The automation server rejected the request."
        except Exception as e:
            return f"Network error in automation bridge: {e}"

# Integration for the Main Controller
def handle_automation(query):
    bridge = AutomationBridge()
    if "lights" in query:
        if "on" in query:
            return bridge.toggle_device("light.main_room", "turn_on")
        elif "off" in query:
            return bridge.toggle_device("light.main_room", "turn_off")
    return "I couldn't find a matching automation protocol, Sir."
```