```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from typing import Optional

app = FastAPI(title="NeuroAIR API")

class AromaSettings(BaseModel):
    aroma_id: str
    intensity: int  # 1-100
    duration: Optional[int] = None  # в минутах

# Mock база данных с состояниями устройств
devices_db = {}

@app.post("/devices/{device_id}/activate")
async def activate_aroma(device_id: str, settings: AromaSettings):
    """Активировать распыление аромата"""
    if device_id not in devices_db:
        devices_db[device_id] = {"status": "off", "current_aroma": None}

    devices_db[device_id] = {
        "status": "on",
        "current_aroma": settings.aroma_id,
        "intensity": settings.intensity,
        "duration": settings.duration
    }
    return {"status": "success"}

@app.post("/devices/{device_id}/stop")
async def stop_aroma(device_id: str):
    """Остановить распыление"""
    if device_id in devices_db:
        devices_db[device_id]["status"] = "off"
    return {"status": "success"}

@app.get("/devices/{device_id}/status")
async def get_status(device_id: str):
    """Получить статус устройства"""
        return devices_db.get(device_id, {"status": "unknown"})
```

```
"""Support for NeuroAIR aroma diffuser."""
import logging
import voluptuous as vol

from homeassistant.components.climate import ClimateEntity
from homeassistant.components.climate.const import (
    HVAC_MODE_OFF,
    HVAC_MODE_FAN_ONLY,
    SUPPORT_PRESET_MODE,
    SUPPORT_FAN_MODE,
)
from homeassistant.const import TEMP_CELSIUS
import homeassistant.helpers.config_validation as cv

_LOGGER = logging.getLogger(__name__)

DOMAIN = "neuroair"
SUPPORT_FLAGS = SUPPORT_PRESET_MODE | SUPPORT_FAN_MODE

AROMA_PRESETS = {
    "relax": "Релакс",
    "energy": "Энергия",
    "focus": "Концентрация",
    "sleep": "Сон",
}

async def async_setup_platform(hass, config, async_add_entities,
discovery_info=None):
    """Set up NeuroAIR platform."""
    # Здесь должна быть логика подключения к вашему API
    api = NeuroAirAPI(config.get("host"), config.get("token"))

    devices = await hass.async_add_executor_job(api.get_devices)

    entities = [NeuroAirDevice(device, api) for device in devices]
    async_add_entities(entities)

class NeuroAirDevice(ClimateEntity):
    """Representation of a NeuroAIR device."""

    def __init__(self, device, api):
        """Initialize the device."""
        self._api = api
        self._device = device
        self._name = device["name"]
        self._state = None
        self._current_aroma = None
        self._intensity = 0

    @property
    def name(self):
        """Return the name of the device."""
        return self._name

    @property
    def hvac_modes(self):
        """Return the list of available hvac operation modes."""
        return [HVAC_MODE_OFF, HVAC_MODE_FAN_ONLY]

    @property
    def preset_modes(self):
        """Return available aroma presets."""
        return list(AROMA_PRESETS.values())

    async def async_set_preset_mode(self, preset_mode):
        """Set new preset mode."""
        aroma_id = next(k for k, v in AROMA_PRESETS.items() if v == preset_mode)
        await self._api.activate_aroma(self._device["id"], aroma_id, 50)
        self._current_aroma = preset_mode

    async def async_turn_on(self):
        """Turn the device on."""
        await self._api.turn_on(self._device["id"])
        self._state = HVAC_MODE_FAN_ONLY

    async def async_turn_off(self):
        """Turn the device off."""
        await self._api.turn_off(self._device["id"])
        self._state = HVAC_MODE_OFF
```

```
from fastapi import FastAPI, Request
from fastapi.responses import JSONResponse

app = FastAPI()

@app.post("/alice/webhook")
async def alice_webhook(request: Request):
    data = await request.json()

    # Анализ команды от Алисы
    command = data["request"]["command"].lower()

    if "включи аромат" in command:
        aroma = extract_aroma(command)
        return respond_with_text(f"Включаю аромат {aroma}")
    elif "выключи нейроэйр" in command:
        return respond_with_text("Выключаю аромадиффузор")

    return respond_with_text("Не поняла команду")

def extract_aroma(text):
    """Извлечь название аромата из текста"""
    aromas = ["релакс", "энергия", "концентрация", "сон"]
    for aroma in aromas:
        if aroma in text:
            return aroma
    return "релакс"  # по умолчанию

def respond_with_text(text, end_session=False):
    return JSONResponse({
        "version": "1.0",
        "session": request_data["session"],
        "response": {
            "text": text,
            "end_session": end_session
        }
    })
```

```
// fulfillment для Google Home
```

```
const {smarthome} = require('actions-on-google');
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
app.use(bodyParser.json());

const neuroair = smarthome();

neuroair.onSync((body) => {
  return {
    requestId: body.requestId,
    payload: {
      agentUserId: '123',
      devices: [{
        id: 'neuroair-1',
        type: 'action.devices.types.AIRFRESHENER',
        traits: [
          'action.devices.traits.OnOff',
          'action.devices.traits.Modes'
        ],
        name: {
          defaultNames: ['NeuroAIR'],
          name: 'Аромадиффузор',
          nicknames: ['Диффузор', 'Ароматы']
        },
        attributes: {
          availableModes: [{
            name: 'aroma',
            name_values: [{
              name_synonym: ['аромат', 'запах', 'эфирное масло'],
              lang: 'ru'
            }],
            settings: [{
              setting_name: 'релакс',
              setting_values: [{
                setting_synonym: ['релакс', 'расслабление'],
                lang: 'ru'
              }]
            }, {
              setting_name: 'энергия',
              setting_values: [{
                setting_synonym: ['энергия', 'бодрость'],
                lang: 'ru'
              }]
            }],
            ordered: false
          }]
        }
      }]
    }
  };
});

neuroair.onQuery((body) => {
  // Запрос состояния устройства
  return {
    requestId: body.requestId,
    payload: {
      devices: {
        'neuroair-1': {
          on: true,
          currentModeSettings: {
            aroma: 'релакс'
          }
        }
      }
    }
  };
});

neuroair.onExecute((body) => {
  // Выполнение команд
  const commands = body.inputs[0].payload.commands;
  const executionResponses = [];

  commands.forEach((command) => {
    command.execution.forEach((execution) => {
      if (execution.command === 'action.devices.commands.OnOff') {
        // Обработка вкл/выкл
      } else if (execution.command === 'action.devices.commands.SetModes') {
        // Установка аромата
        const aroma = execution.params.updateModeSettings.aroma;
        // Вызов API NeuroAIR
      }
    });
  });
  return {
    requestId: body.requestId,
    payload: {
      commands: executionResponses
    }
  };
});

app.post('/fulfillment', neuroair);
app.listen(3000);
```