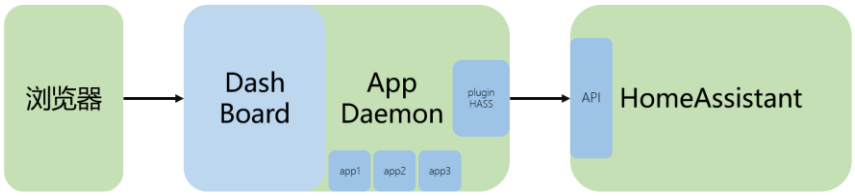
安装、配置与初步运行

【操作步骤】

1. AppDaemon 与 DashBoard 的架构
2. 在 Python 虚拟环境中安装 AppDaemon
3. 配置 AppDaemon
4. 手工运行 AppDaemon
5. 生成第一个 DashBoard 界面

【参考】

- AppDaemon 与 DashBoard 的架构



- AppDaemon 文档网站
  https://appdaemon.readthedocs.io/en/latest/index.html
- 在 Python 虚拟环境中安装 AppDaemon

```
cd
python3 -m venv appdaemon_venv
cd appdaemon_venv
source bin/activate
pip3 install wheel
pip3 install appdaemon
deactivate
```

- AppDaemon 与 DashBoard 的最基础配置

```
appdaemon:
   threads: 10
   plugins:
      HASS:
         type: hass
         ha_url: http://127.0.0.1:8123
         token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJkNDk4NmQ5MzkyM2Y0ZTMzYmRmZjY3NjJkNzc3NjI0NiIsImV4cCI
6MTg1OTc4OTM5MCwiaWF0IjoxNTQ0NDI5MzkwfQ.3I8Sxd242LMQyEsnzEWqnQIChUekfN9rMAkE580qTzs
   hadashboard:
      dash_url: http://0.0.0.0:5050
      dash_password: hachina      #设置访问密码
```

DashBoard 配置(1)

【操作步骤】

1. 将 AppDaemon 加入自启动服务
2. DashBoard 配置样例实操与讲解

【参考】

- AppDaemon 自启动服务配置文件(`/etc/systemd/system/appdaemon@pi.service`)

```
[Unit]
Description=AppDaemon
After=home-assistant@pi.service
[Service]
Type=simple
User=%i
ExecStart=/home/pi/appdaemon_venv/bin/appdaemon -c /home/pi/appdaemon
[Install]
WantedBy=multi-user.target
```

- DashBoard 配置参考

  https://appdaemon.readthedocs.io/en/latest/DASHBOARD_CREATION.html#

- DashBoard 配置样例(example_13_2_1.dash):

```
title: Hello Panel
widget_dimensions: [120, 120]
widget_margins: [5, 5]
columns: 4

my_clock:
  widget_type: clock
  show_seconds: 1
  date_style: "color: #00aaff"
  time_style: "color: #ffaa00"

temperature1:
  widget_type: sensor
  entity: sensor.temperature_158d0001d6daa6
  title:  室内温度
  title2:  小米温湿度传感器
  title_style: "color: #00aaff"
  title2_style: "color: #00aaff"
  value_style: "color: #ffaa00"

temperature2:
  widget_type: temperature
  entity: sensor.temperature_158d0001d6daa6
  settings:
    minValue: 0
    maxValue: 50
    majorTicks: [0,10,20,30,40,50]
    highlights: [{'from':15, 'to': 30, 'color':'rgba(255,170,0,0.3)'}]

layout:
  - my_clock(2x1)
  - temperature1, temperature2
```

DashBoard 配置(2)

【操作步骤】

1. 准备工作：HA 的 darksky 配置
2. 将 Dashboard 样例文件放到 dashboards 目录中
3. 天气页面的配置
4. 控制页面的配置
5. 包含底部导航

【参考】

- DashBoard 配置参考
  https://appdaemon.readthedocs.io/en/latest/DASHBOARD_CREATION.html#
- 样例文件:
  example_13_3_1.yaml —— HomeAssistant 中 darksky 的配置文件
  example_13_3_1.dash —— 天气页面 DashBoard
  example_13_3_2.dash —— 控制页面 DashBoard
  example_13_3_3.yaml —— 底部导航 DashBoard

制作 App——一个最简单的样例

【操作步骤】

1. 创建 apps 子目录
2. 创建示例 app（hello.py）
3. 创建 app 配置（my_apps.yaml）
4. app 自动重新加载

【参考】

- 样例 app（hello.py）

```
import appdaemon.plugins.hass.hassapi as hass
class HelloWorld(hass.Hass):
    def initialize(self):
        self.call_service("persistent_notification/create",
                          title="来自 AppDaemon 的消息",
                          message="Hello, World!"
                          )
```

- my_apps.yaml

```
hello_world:
  module: hello
  class: HelloWorld
```

- AppDaemon Apps 介绍
  https://appdaemon.readthedocs.io/en/latest/APPGUIDE.html
- 推荐的 Python 教程
  https://www.liaoxuefeng.com/wiki/0014316089557264a6b348958f449949df42a6d3a2e542c000

制作 App——应用 callback

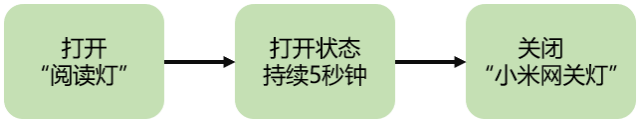
【操作步骤】

1. 停止自动运行的 appdaemon，手工运行
2. 样例一：延时执行任务
3. 样例二：基于状态改变执行任务
4. 样例三：基于事件执行任务

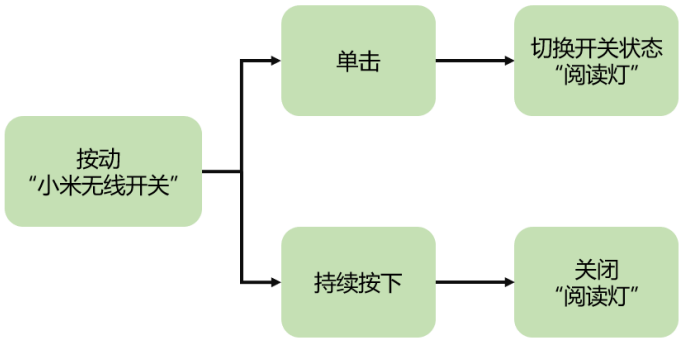
【参考】

- AppDaemon API 参考
  https://appdaemon.readthedocs.io/en/latest/AD_API_REFERENCE.html#
- 延时执行任务（样例一）

- 基于状态改变执行任务（样例二）



- 基于事件执行任务（样例三）



- app 程序

```
import appdaemon.plugins.hass.hassapi as hass

class sub1(hass.Hass):

    def initialize(self):
        self.call_service("persistent_notification/create",
                          notification_id=54321,
                          title="通知",
                          message="准备 5 秒后打开灯"
                          )
        self.run_in(self.lighton_fun, 5)

    def lighton_fun(self, kwargs):
        self.call_service("persistent_notification/dismiss",
                          notification_id=54321
                          )
        self.turn_on("light.gateway_light_7c49eb18e3a7")

class sub2(hass.Hass):

    def initialize(self):
```

```
        self.listen_state(self.lightoff_fun,
                          entity="switch.plug_158d0001703829",
                          old="off",
                          new="on",
                          duration=5
                          )

    def lightoff_fun(self, entity, attribute, old, new, kwargs):
        self.turn_off("light.gateway_light_7c49eb18e3a7")

class sub3(hass.Hass):

    def initialize(self):
        self.listen_event(self.lightcontrol_fun,
                          event="xiaomi_aqara.click",
                          entity_id="binary_sensor.switch_158d000121cf8d"
                          )

    def lightcontrol_fun(self, event_name, data, kwargs):
        if data['click_type']=='single':
            self.toggle("switch.plug_158d0001703829")
        elif data['click_type']=='hold':
            self.turn_off("switch.plug_158d0001703829")
```