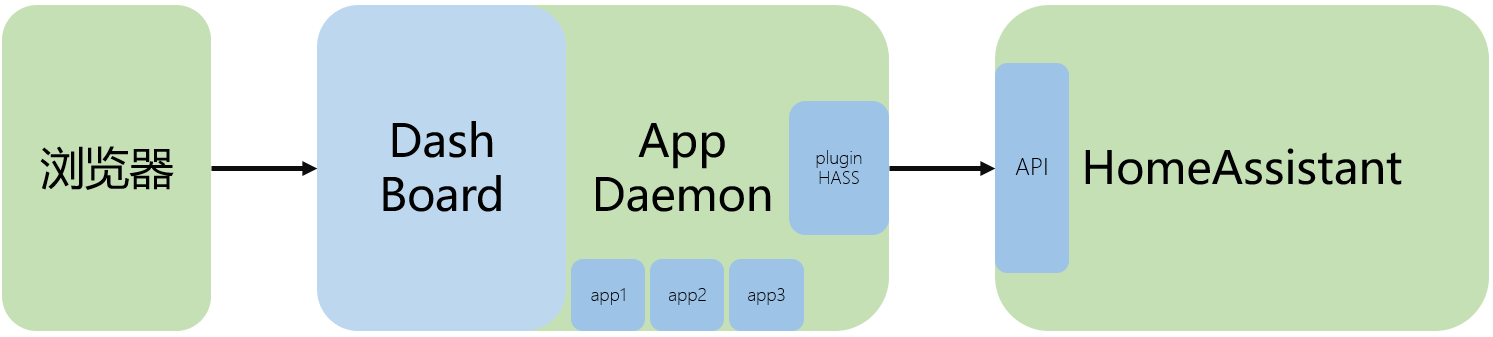
安装、配置与初步运行

【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:

latitude: 31.3016

longitude: 121.5113

elevation: 0

time\_zone: Asia/Shanghai

plugins:

HASS:

type: hass

ha\_url: http://127.0.0.1:8123

token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3……

http:

url: http://0.0.0.0:5050

password: hachina #设置访问密码

hadashboard:

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#](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#](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 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】

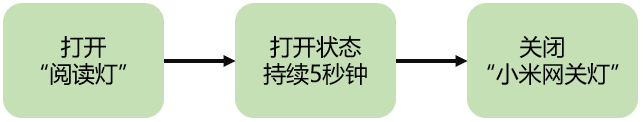
* AppDaemon API参考

[https://appdaemon.readthedocs.io/en/latest/AD\_API\_REFERENCE.html#](https://appdaemon.readthedocs.io/en/latest/AD_API_REFERENCE.html)

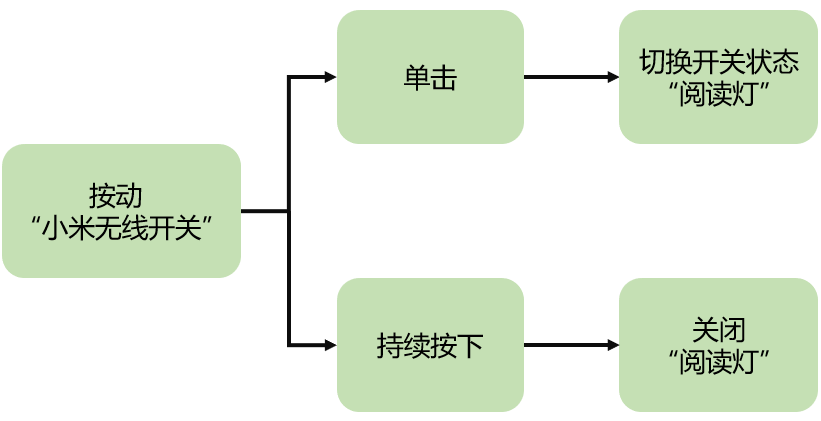
* 延时执行任务（样例一）



* 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 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")