

Intégration SolisArt

Avertissement d'usage

Je ne suis pas un professionnel du développement, ni un expert de Home Assistant. J'ai développé cette intégration pour mon usage personnel, et je vous la partage « en état ».

Elle fonctionne chez moi depuis octobre 2024 sans incident.

Si vous choisissez de l'installer, c'est sous votre entière responsabilité. Je ne pourrais être tenu d'aucune responsabilité de dommages ni matériels ni immatériels consécutif à l'usage direct ou indirect de ce développement.

Je ne vous garantis pas ma disponibilité, ni ma compétence pour en assurer support ou assistance.

Je partage ce code à titre entièrement gratuit, vous êtes libre de l'utiliser à titre non commercial, sans en tirer aucun profit.

Introduction

Cette intégration est conçue pour Home Assistant, néanmoins il sera assez facile de l'adapter à d'autre système de domotique (voir le chapitre principe de fonctionnement)

Pour en faciliter l'installation et la maintenance, cette intégration est fournie sous forme d'un package HomeAssistant.

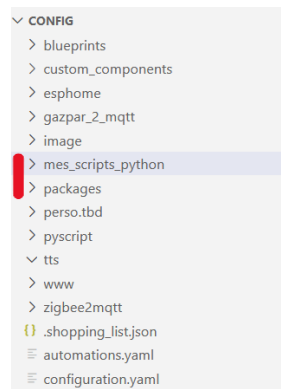
Préparation

Si vous ne l'avez déjà fait vous devez d'abord effectuer les opérations suivantes :

Dans le répertoire « **config** » créer les répertoires

1. « **packages** »
2. « **mes_scripts_python** »

Note : vous pouvez choisir un autre nom SAUF « pyscript » ou « python_script » qui sont reconnus et interprétés par HA

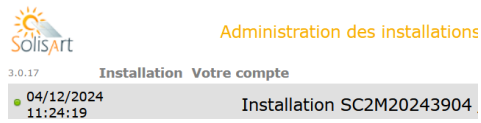


3. Dans le « config.yaml » : Ajouter la ligne

```
homeassistant:
  packages: !include_dir_named packages
```

Installation des fichiers

4. Copiez le fichier « **solisart.yaml** » dans le répertoire « **Packages** »
5. Copiez le fichier « **Solisart.py** » dans le répertoire « **mes_scripts_python** »
6. Ouvrez le fichier « **Solisart.py** » et renseignez vos login et password de connexion à l'application SolisArt, ainsi que le n° de votre installation



Ne faites aucune autre modification dans ce fichier

7. Enregistrer et fermer ce fichier

Vérification

- Redémarrer home assistant
- Aller dans : Outils de développement > Etats > dans « filtrer les entités », tapez: **solisart** ➔ vous devriez voir apparaître les Sensors créés de base.

Extrait :

Outils de développement		
YAML	ÉTATS	ACTIONS MODÈLE ÉVÉNEMENTS STATISTIQUES ASSIST
États actuels de l'entité		
Définir l'état		
Entité	État	Attributs <input checked="" type="checkbox"/>
<input type="text" value="filtrer les entités solisart"/>	<input type="text" value="filtrer les états"/>	<input type="text" value="filtrer les attributs"/>
<input type="checkbox"/> binary_sensor.solisart_chaudiere1 <input type="info"/> solisart Chaudiere1	off	friendly_name: solisart Chaudiere1
<input type="checkbox"/> sensor.solisart <input type="info"/> SolisArt		t1: 10.8 t2: 19.7 t3: 21.6 t4: 44.0 t5: 16.3

A ce stade le système fonctionne. Vous devez maintenant l'adapter à votre installation.

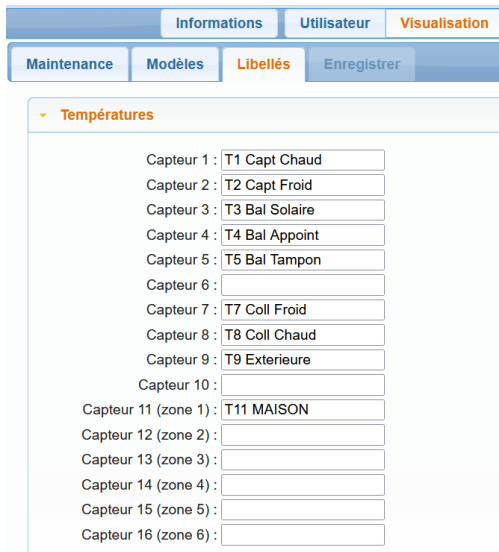
Adaptation à votre installation SolisArt

Toutes les modifications s'effectuent uniquement dans le fichier **solisart.yaml**.

Pour faire correspondre les libellés à votre installation ouvrez l'application SolisArt

=> onglet Visualisation > Libellés

Température



Chaque « capteur » correspond au code T

En voici un exemple pour la température T1 (Capteur 1 dans l'appli SolisArt) :

```

- name: "solisart_T1"
  unique_id: "solisart_T1"
  unit_of_measurement: "°C"
  device_class: temperature
  state: "{{ state_attr('sensor.solisart', 't1') }}"

```

En vous référant à votre tableau supprimez ou ajoutez par copier/coller les Sensors température de votre installation.

Circulateurs et vannes



Les circulateurs sont dénommés par SolisArt **trx** (circulateur 1 ⇔ tr1). Ils sont tous décrits dans le fichier, il suffit de commenter ou décommenter chaque bloc de description.

Exemple : sur mon installation il y a un circulateur C6 en 3^{ème} position (tr3), mais il n'y a pas de circulateur en 4^{ème} position (tr4).

```

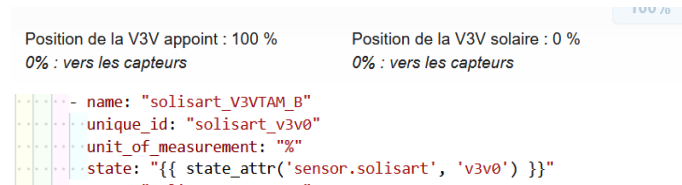
- name: "solisart_cir3 C6 Bal Tampon"
  unique_id: "solisart_tr3"
  unit_of_measurement: "%"
  state: "{{ state_attr('sensor.solisart', 'tr3') }}"
# - name: "solisart_cir4"
#   unique_id: "solisart_tr4"
#   unit_of_measurement: "%"
#   state: "{{ state_attr('sensor.solisart', 'tr4') }}"

```

Les vannes sont dénommées par SolisArt V3V0 à V3V2

Je n'ai pas trouvé de règle claire de correspondance des noms. Normalement, la configuration fournie fonctionne dans la majorité des cas.

Pour vous en assurez, vous pouvez comparer les positions des vannes avec l'application SolisArt.



Note : les tr8 à tr13 représentent les indicateurs de mouvement des vannes. Ces informations sont à mon sens sans réelle utilité pour un système à vanne 3 voies. Je les ai donc commentés.

```
# Mouvement des V3V (probablement Binary sensor - QUELLE UTILITE ???)
# ----- name: "solisart_V3VAPP moving to App"
# ----- unique_id: "solisart_v3v0_to_app"
# ----- state: "{{ state_attr('sensor.solisart', 'tr8') }}"
# ----- name: "solisart_V3VAPP moving to solaire"
# ----- unique_id: "solisart_v3v0_to_solaire"
# ----- state: "{{ state_attr('sensor.solisart', 'tr9') }}"
# ----- name: "solisart_V3VTAM moving to Solaire"
# ----- unique_id: "solisart_v3v1_to_solaire"
# ----- state: "{{ state_attr('sensor.solisart', 'tr12') }}"
# ----- name: "solisart_V3VTAM moving to Tampon"
# ----- unique_id: "solisart_v3v1_to_tampon"
# ----- state: "{{ state_attr('sensor.solisart', 'tr13') }}"
```

Chaudières

SolisArt permet de gérer deux chaudières (Appoint) nommée rl0 et rl4 représentées sous forme de Binary Sensor.

```
# ----- name: "solisart Chaudiere1"
# ----- unique_id: "solisart_rl0"
# ----- state: "{{ state_attr('sensor.solisart', 'rl0') }}"
# ----- name: "solisart Chaudiere2"
# ----- unique_id: "solisart_rl4"
# ----- state: "{{ state_attr('sensor.solisart', 'rl4') }}"
```

Autres Sensor

D'autres Sensor sont fournis dans cette integration

- Débitmètres si votre installation en est pourvue (commentés par défaut)
- LastUpdate : date et heure a eu lieu la dernière collecte des valeurs
- Position des V3V par rapport à la position A (elles sont fournies par défaut en rapport à la position B)

Principe de fonctionnement

Le script Python interroge le serveur SolisArt et génère un Json contenant les paramètres de votre installation.

Note : Il peut donc être utilisé par une autre application domotique ou GTB SCADA.

Le sensor générique « SolisArt » invoque le script Python et « sépare » chaque valeur.

Chaque Sensor s'appuie sur ces valeurs et les enrichie des propriété adéquates.

Affichage des paramètres

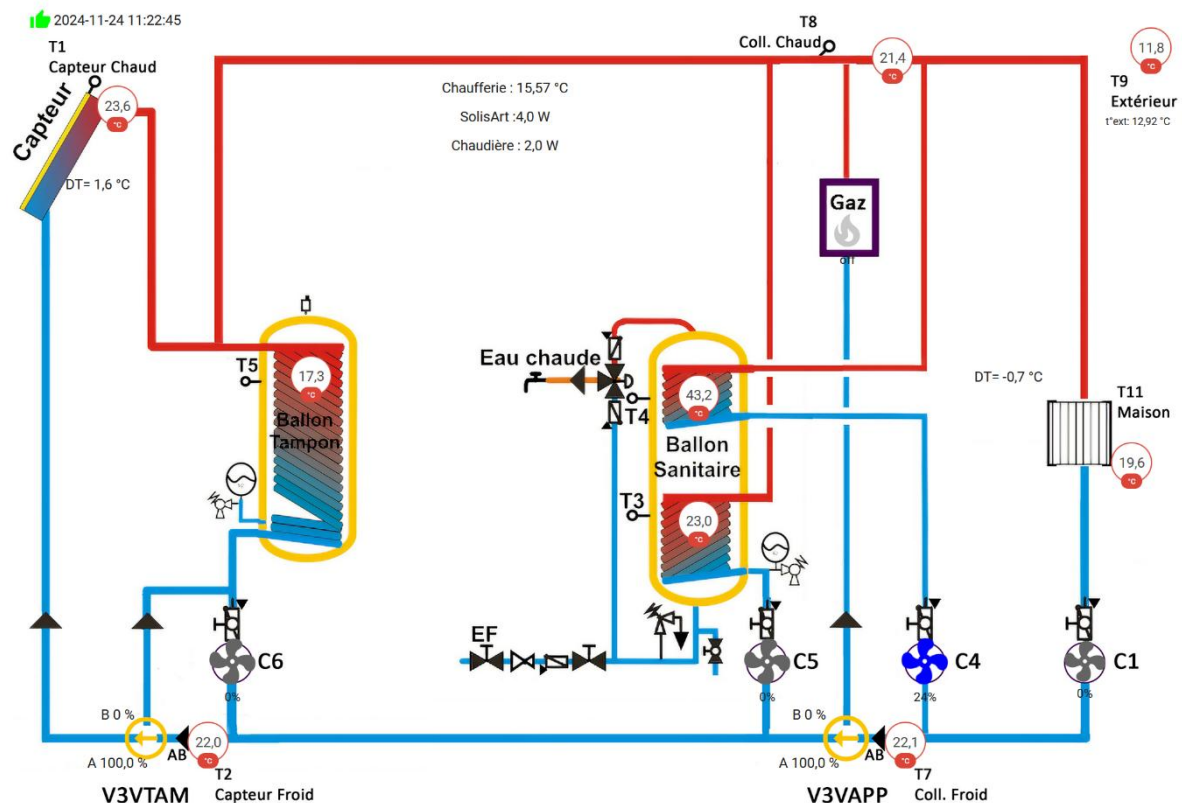
Méthode basique

Je vous recommande de commencer par créer un tableau de bord avec tous les paramètres



J'utilise un tableau de bord dans lequel j'ai toutes les valeurs brutes.

Schéma « animé »



Dans ce schéma les températures sont affichées à leur position dans l'installation.

Lorsque les circulateurs fonctionnent et en fonction de leur vitesse ils tournent +/- vite sur l'animation et change de couleur.

Les vannes 3 voies s'orientent afin de visualiser les circuits.

La flamme de la chaudière devient rouge et clignote, lorsqu'elle est activée.

Pour mon usage, j'ai aussi rajouté des informations provenant d'autres Sansors : t° de la pièce, consommation électrique du SolisArt et de la chaudière, ...

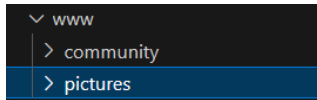
Le schéma de base

Avant de commencer il vous faut le schéma de votre installation que vous pouvez télécharger depuis l'application MySolisArt : Visualisation > Maintenance > Clic droit sur le schéma > enregistrer l'image sous...

Vous pouvez ensuite modifier ce schéma. Pour rendre le schéma plus lisible, j'ai choisi de déplacer la chaudière d'appoint et d'enlever les légendes. C'est un choix personnel.

J'ai aussi remplacé les circulateurs par des cercles vides, et effacé les vannes 3 voies, qui seront remplacés par des icônes animées, un peu plus loin.

Une fois votre schéma terminé, copiez le dans le répertoire « *config/www* ». Je vous recommande d'y créer un sous-répertoire (« pictures » dans mon exemple). Cette étape n'est pas obligatoire mais elle aide à conserver une organisation claire.



Remarques Importantes :

1. Copiez directement votre image dans ce répertoire, n'utilisez pas les fonctions d'importation de HA
2. Si vous modifiez votre schéma il faut changer son nom, sinon HA ne le prend pas en compte.

Avec ces deux remarques vous avez économisé 3 à 4 jours de galère, à chercher pourquoi tout est flou et vos modifications ne sont pas prises en compte...

Création du tableau de bord

Je vous propose de créer un tableau de bord spécifique

- Paramètres > Tableaux de bord > Ajouter un tableau de bord > ajouter un tableau de bord vide

 A screenshot of the 'Ajouter un nouveau tableau de bord' (Add new dashboard) form in Home Assistant. The form has the following fields and options:

- Title*: SolisArt
- Icone: mdi:sun-wireless
- URL*: dashboard-solisart
- Administrateur uniquement: Toggle switch (off)
- Afficher dans la barre latérale: Toggle switch (on)
- CRÉER button

- Ouvrez le tableau de bord ainsi créé, puis ses paramètres

Configuration de la vue SolisArt

PARAMÈTRES ARRIÈRE-PLAN VISIBILITÉ

Mise en page

☐ Sections (par défaut)

☐ Maçonnerie

☐ Barre latérale

☒ Panneau (carte unique)

Titre
SolisArt

Icône

URL

Thème

Sous-vue
Les sous-vues n'apparaissent pas dans les onglets et disposent d'un bouton de retour.

SUPPRIMER LA VUE ENREGISTRER

- Enregistrer
- Ajouter une carte > Élément d'image (Picture card) > option de la carte

Sélectionner Chemin de l'image

☐ Téléverser une image

☒ Chemin local ou URL Web

Chemin de l'image
/local/pictures/schemaHydraulique.png

Attention au chemin de l'image ! c'est bien « */local/pictures* » qu'il faut indiquer et non pas « *www* » (encore un mystères de HA).

C'est aussi ici que vous indiquerez le nom d'une éventuelle nouvelle version de votre schéma.

A ce point vous devez avoir un tableau de bord avec votre schéma, sans aucune valeur.

Peuplement du TdB avec les éléments fixes

Cliquez sur le crayon en haut à droite ; puis sur « modifier » tout en bas à gauche...

- Ajouter un nouvel élément.
Pour afficher les températures j'ai choisi les « badges d'état », pour les autres informations (%vannes par exemple) j'ai utilisé les « labels d'état ». C'est une question de choix personnel.

- Ajoutez par exemple le capteur de température T1

Entité*
solisart_T1

Titre
T1 Chaud Capteur

Comportement lors d'un appui court
Par défaut

Comportement lors d'un appui long
Par défaut

Style

1 left: 11.7%
2 top: 8.4%
3

Le positionnement se fait en indiquant la distance relative de l'élément par rapport à la gauche (left) et au haut (top). Il faut quelques essais pour positionner votre élément exactement là où vous le voulez. Vous pouvez utiliser des nombres décimaux.

- Répétez l'opération autant de fois que vous avez de sensors à positionner sur votre schéma.
- Je préfère passer directement en mode éditeur, et de faire des copier/coller dans le fichier Yaml.

➔ Je joins le code complet de mon tableau de bord (**tdb.yaml**), vous y trouverez les éléments de configuration qui vous permettrons de gagner du temps. Faites des copier/coller des sections qui vous intéressent.

```
type: picture-elements
image: /local/pictures/schemaHydrau-3-V06_50pc.png
elements:
  - type: state-badge
    style:
      left: 10%
      top: 15%
      color: transparent
    entity: sensor.solisart_t1
    title: T1 Chaud Capteur
```

Les éléments animés

Nous allons maintenant configurer tous les éléments animés : circulateurs, vannes 3 voies et chaudière.

Les éléments animés sont constitués par des « **Button-Cards** »

1. Installez Button card : <https://github.com/custom-cards/button-card#installation>
je vous recommande la méthode d'installation par HACS, plus intuitive.

Button-card ne dispose pas d'interface graphique, il faut donc le faire directement par le code yaml.

➔ Pour vous faciliter la vie, je vous recommande d'utiliser l'exemple que je fourni (**tdb.yaml**) et de le modifier.

Animation des circulateurs

```
type: custom:button-card
title: C4 Appoint
entity: sensor.solisart_cir1_c4_bal_appoint
name: "[[[ return entity.state + \"%\"]]]"
size: 25%
color_type: icon
icon: mdi:fan
styles:
  card:
    - background-color: rgba(0, 0, 0, 0.0)
label: null
style:
  left: 72.5%
  top: 72.3%
state:
  - value: 50
    operator: ">="
    color: rgb(0,255,0)
    styles:
      icon:
        - animation: rotating 1s linear infinite
        - color: rgb(0,255,0)
  - value: 1
    operator: ">="
    styles:
      icon:
        - animation: rotating 2s linear infinite
        - color: rgb(0,0,255)
  - value: 0
    color: rgb(100,100,100)
```

On utilise l’icône de type ventilateur (*mdi :fan*)

Lorsque la vitesse du circulateur est $\geq 50\%$ sa couleur devient verte (*rgb(0,255,0)*) et l’icône fait un tour toutes les secondes (*animation : rotating 1s linear infinite*)

Lorsque sa vitesse est $\geq 1\%$ (c.à.d. il tourne, mais à moins de 50%) l’icône tourne moins vite : un tour toutes les 2 secondes et est bleue.

A l’arrêt, elle ne bouge pas et est grise.

Vannes 3 voies

```
type: custom:button-card
title: V3VAPP
entity: sensor.solisart_v3vapp
name: "[[[ return entity.state + \"%\"]]]"
size: 25%
aspect-ratio: 1/1
show_name: false
style:
  left: 66.4%
  top: 79.5%
  color: transparent
styles:
  card:
    - background-color: rgba(0, 100, 0, 0.0)
state:
  - value: 95
    operator: ">="
    color_type: icon
    icon: mdi:arrow-up-thin-circle-outline
    color: rgb(219,68,55)
  - value: 5
    operator: ">="
    color_type: icon
    icon: mdi:arrow-top-left-thin-circle-outline
    color: rgb(235,132,25)
  - value: 5
    operator: "<"
    color_type: icon
    icon: mdi:arrow-left-thin-circle-outline
    color: rgb(251,196,0)
```

Pour représenter les vannes 3 voies, on va utiliser les icones de type flèche dans un cercle.

On définit 3 zones de fonctionnement :

- $<5\%$: flèche horizontale jaune
- $\geq 5\%$ (implicitement $<95\%$) : flèche à 45° orange
- $\geq 95\%$ flèche verticale rouge

Vous pouvez changer ces valeurs (et couleurs) comme vous le souhaitez.

Chaudière

<pre>title: Chaudiere type: custom:button-card entity: binary_sensor.solisart_chaudiere1 name: "[[[return entity.state]]]" size: 20% color_type: icon icon: mdi:fire style: left: 66.5% top: 27.6% styles: card: - background-color: rgba(0, 0, 0, 0.0) state: - value: "on" styles: icon: - animation: blink 2s ease infinite - color: rgb(255,0,0) - value: "off" styles: icon: - animation: none - color: rgb(200,200,200)</pre>	<p>Icone flamme <i>mdi :fire</i></p> <ul style="list-style-type: none">• On : icone rouge, clignotante (<i>animation : blink 2s ease infinite</i>)• Off : icone grise, fixe
---	--

Bugs connus

Les différents objets de l’affichage « Schéma » zoom différemment.