Contrôleurs Régulateurs Embarqués pour des Systèmes Utilisant l'énergie Solaire

Patrick Gatt, Georges Kariniotakis, François-Pascal Neirac

Projet AVENE PME
CRESUS

Rapport : ARMINES / PERSEE (ERSEI)

Contrôleurs Régulateurs Embarqués pour des Systèmes Utilisant l'énergie Solaire

Le 13-01-2016

Patrick Gatt
Ingénieur de recherche
PERSEE – MINES ParisTech
patrick.gatt@mines-paritech.fr

Georges Kariniotakis
Responsable Groupe ERSEI
PERSEE – MINES ParisTech
georges.karinitokis@mines-paritech.fr

François-Pascal Neirac
Professeur
PERSEE – MINES ParisTech
francois-pascal.neirac@mines-paritech.fr
Georges Kariniotakis

Table des matières

1 Préambule

Ce livrable décrit l'ensemble des travaux d'aménagement et de développement une plate-forme expérimentale de PERSEE pour « **C**ontrôleurs **R**égulateurs **E**mbarqués pour des **S**ystèmes **U**tilisant l'énergie **S**olaire » au travers de la réalisation d'une maquette expérimentale d'un contrôleur prédictif pour chauffe-eau solaire individuel CESI.

1.1 Contexte et environnement

INNOVATION :
Les entreprises développant des systèmes utilisant l'énergie solaire (SUS) doivent se différencier en apportant une innovation. Celle-ci prend très souvent la forme d'un contrôleur/régulateur embarqué (CRE) qui peut être suivant les cas un BMS (Battery Management System), un EMS (Energy Management System), un régulateur prédictif, etc.
Le projet CRESUS vise à concevoir une maquette de régulateur prédictif pour chauffe-eau solaire individuel (CESI) qui sera développée et évaluée en utilisant deux CESI placés dans des conditions d'ensoleillement et de soutirage identiques, l'un équipé du régulateur prédictif, l'autre pas.
Ce projet propose un service innovant à toutes les PME/ETI concernées, leur permettant :
Soit de tester une idée en utilisant un environnement de programmation de carte embarquée, et de tester l'ensemble {système solaire/CRE} en conditions réelles (TRL 4).
Soit de faire la preuve de concept d'un CRE innovant développé au sein de l'entreprise en permettant de quantifier ses performances, ou d'en affiner le paramétrage en environnement de test en conditions réelles (TRL 5).
Le principe de base du service CRESUS est de quantifier la performance du dispositif innovant en utilisant le principe du test en parallèle.
Le Centre PERSEE souhaite développer ce service, en utilisant :
D'une part le savoir-faire et le matériel d'acquisition développés sur sa plate-forme de Sophia-Antipolis depuis plus de 30 ans.
D'autre part en développant les moyens logiciels et matériels nécessaires pour permettre le développement et le test de CRE.
Cet environnement comprend tout d'abord un environnement de simulation classique (type Matlab) pour tester des algorithmes de contrôle/régulation sur un système solaire simulé, puis un ensemble d'outils permettant de compiler la partie contrôle/régulation du simulateur sur une carte embarquée (type Raspberry), et enfin tous les éléments nécessaires pour relier les Entrées/Sorties de la carte au monde physique, à savoir le système solaire, des sondes de mesure, ainsi que les prédictions transmises par radio ou internet.

1.2 Objectifs pour la plateforme d'essai CRESUS

Dans ces conditions, nous avons entrepris de développer une nouvelle plateforme d'essais, intégrant d'une part une métrologie autonome avec des outils de contrôle / commande d'autre part un environnement de développement pour contrôleur/régulateur embarqué.

La méthodologie consistera à tester en conditions réelles de fonctionnement l'apport d'un régulateur embarqué prédictif sur les performances d'un chauffe-eau solaire avec appoint électrique par rapport à un chauffe-eau solaire à appoint électrique sans régulateur prédictif.

1.3 Concept de base pour la mise en place et le test d'une première maquette

Une étude préliminaire nous a permis de mettre en évidence que des économies substantielles pouvaient être réalisée si l'on contrôle la gestion d'un système de production à énergie renouvelable avec stockage et appoint en introduisant une connaissance prévisible des ressources disponible pour les 24h avenir.

Le brevet ARMINES 13.01663_PERSEE ci-dessous fait suite à cette étude (voir annexe 6.1).

Réf./ARMINES : 13.01663_PERSEE

Titre officiel : Installation de production énergétique comprenant un dispositif de prédiction météorologique, notamment une installation de chauffe-eau solaire comprenant un tel dispositif

Date de dépôt : 10/07/2013
N° de la demande : 1301663

Les résultats de l'étude sont présentés dans la revue CVC N°880 de sept./oct. 2013
L'article intitulé : Appoint électrique et prévisions météo (voir annexe 6.2).

2 Contraintes et étapes de réalisation

La mise en place de deux chauffe-eau solaire avec leur contrôleur / régulateur nécessite de prendre en compte plusieurs contraintes :

- La situation de l'aire d'essais sur le toit d'un bâtiment le Mines-Paristech.
- La structure du bâtiment.
- La proximité d'une arrivée d'eau de ville et d'une évacuation.
- Les connexions électriques et données numériques.
- Le positionnement de l'instrumentation et son installation sur le matériel expérimental.

Les étapes que nous avons envisagées pour structurer la réalisation du projet :

- Mettre en place un module préfabriqué prolongé d'une plateforme solaire orientée au sud afin d'accueillir l'ensemble du matériel en respectant les contraintes du bâtiment.
- Installer deux chauffe-eau solaire avec leur capteurs et régulateurs standard du fabricant.
- Mettre en place un ensemble des électrovannes pour générer une demande énergétique (consommation d'eau chaude).
- Installer et réaliser un environnement de métrologie autonome.
- Développer et tester un contrôleur prédictif embarqué à base de nano ordinateur type Raspberry en s'appuyant sur l'environnement fourni par MATLAB.
- Concevoir et réaliser les logiciels de contrôle et de commande de la plateforme de test.
- Tester l'ensemble de la réalisation.
- Valider les calculs préliminaires, par des tests expérimentaux.

2.1 Mise en place du module de la plateforme solaire

L'aire experimentale de laboratoire PERSEE se trouve sur le toit du batiment D de Mines Paristech à Sophia Antipolis. Pour accueillir et protéger les installations le choix s'est porté sur la mise en place un module préfabriqué de la société Decotes. Celui-ci est prolongé par une plateforme solaire. Pour respecter les contraintes du batiment le tout est posé sur des sorties de piliers de facon à ce que rien ne repose sur le pavement ou l'étancheité du toit.

Les differentes étapes

1-Le module et la grue

2-La grue prète pour la manutention

3-Levage du module

4-Installation module et plateforme solaire

5-L'ensemble est ensuite raccordé au differents réseau électricite, données, eau, évacuation d'eau.

2.2 Installation de deux chauffe eau solaire

Le choix c'est porté sur du matériel Viessmann qui utilise en standard des regulateurs très diffusé de la société RESOL avec un bus de communication numérique propriétaire Vbus.

Le matériel installé en situation

1-Module et capteurs solaires

2-Les 2 chauffe-eau à tester

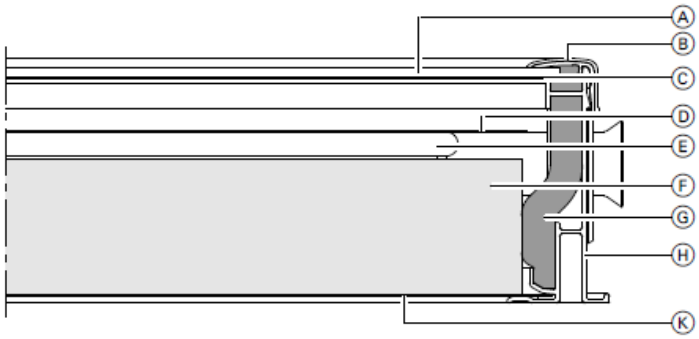
Carateristiques des capteurs solaire Viessmann Vitosol 200-F

Capteur solaire performant version verticale, se composant d'un absorbeur à méandre avec revêtement sélectif en sol-titane, bâti constitué d'un cadre en aluminium plié sur tout son périmètre et joint de vitrage sans raccords, isolation en mélamine ondulée. Face supérieure en verre solaire d'une transparence élevée et résistant à la grêle. Tubage intégré permettant la constitution de batteries d'un maximum de 10 capteurs.

Type SV2A

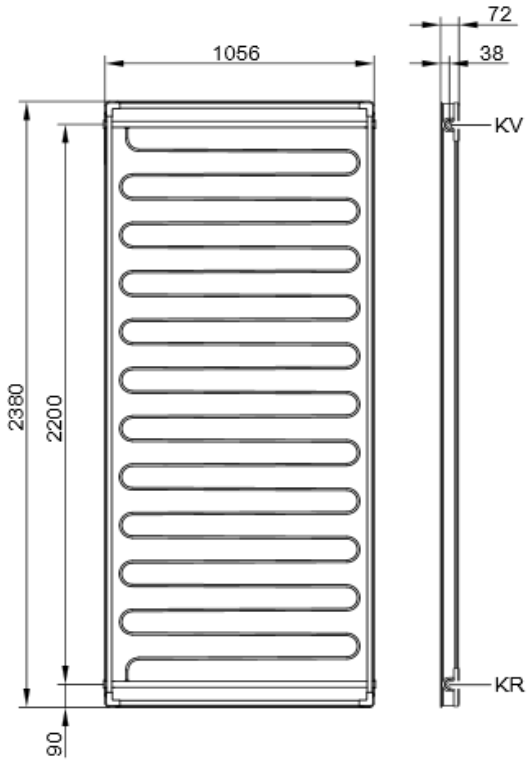
Caractéristiques techniques	
Surface brute	2,51 m²
Surface brute d'absorbeur	2,32 m²
Surface brute d'ouverture	2,33 m²
Largeur	1056 mm
Hauteur	2380 mm
Profondeur	90 mm
Poids (à vide)	41 kg
Contenance en fluide caloporteur	1,83 L
Rendement optique	79,3 %
Déperditions calorifiques k1	4,04 W/(m².K)
Déperditions calorifiques k2	0,0182 W/(m².K²)
Capacité calorifique	5 Kj/(m².K)
Pression de service maximale	6 bar
Température maximale à l'arrêt	186 °C

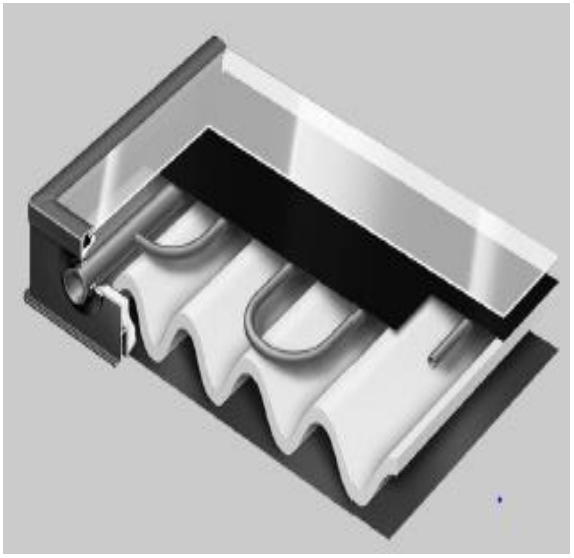
Description



Ⓐ Couvercle de protection en verre solaire, 3,2 mm
Ⓑ Couvre-joint en aluminium
Ⓒ Joint de vitrage
Ⓓ Absorbeur
Ⓔ Tube en cuivre en forme de méandre
Ⓕ Isolation en mousse en résine mélamine
Ⓖ Isolation en mousse en résine mélamine
Ⓗ Cadre en aluminium RAL 8019
Ⓚ Socle en acier avec revêtement zinc et aluminium

Caratéristique dimensionelle de l'absorbeur





Caractéristiques du ballon de stockage électro-solaire Vitocell 100-W CVSA 200L

Version		sans système chauffant électrique	avec système chauffant électrique		
Capacité ballon	**litres**	**200**	**200**	**260**	**300**
Constante de refroidissement Cr	Wh/(l.K.jour)	0,166	0,218	0,217	0,140
Coefficient de pertes thermiques UA	W/K	1,380	1,901	2,348	1,750
V_{es40} (quantité d'eau disponible à 40° C) *1		–	225	380	455
V_{aux}(volume d'appoint)	litres	-	76	122	173
V_{sol} (volume partie solaire)	litres	-	124	140	136

Caractéristiques techniques avec circulateur à haute efficacité énergétique

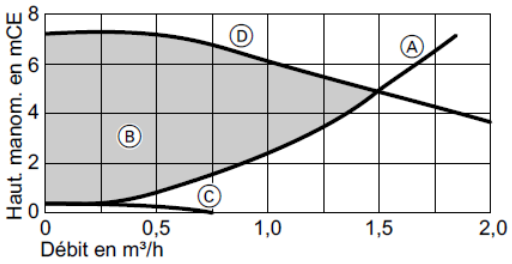
Circulateur (marque Wilo)		Para 15/7
Tension nominale	V~	230
Puissance absorbée		
■ minimale	W	3
■ maximale	W	45
Indicateur de débit	l/mn	2 à 15
Soupape de sécurité (solaire)	bars	6
	MPa	0,6
Température de service maximale	°C	120
Pression de service maximale	bars	6
	MPa	0,6

**1 Valeurs déterminées pour une température de consigne de 60°C avec fonctionnement nocturne et diurne admis en heures creuses.*

Caractéristiques du circulateur du circuit solaire du ballon

Ce circulateur fonctionne en mode PWM « Pulse With Modulation » ce qui permet une transition progressive lors des variations d'ensoleillement et permet un meilleur rendement.

Courbe caractéristique du circulateur à haute efficacité énergétique



Ⓐ Courbe de perte de charge
Ⓑ Hauteur manométrique résiduelle
Ⓒ Puissance minimale
Ⓓ Puissance maximale

Système chauffant électrique EHT

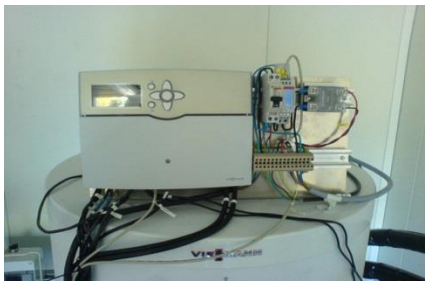
Capacité ballon	**litres**	**200**	**260**	**300**
P_r (puissance nominale)	kW	1,5	2,4	2,7
Tension nominale		1/N/PE 230 V/50 Hz		
Intensité nominale	A	6,5	10,4	11,7
Indice de protection				IP20
Temps de montée en température de 10 à 60° C	h	2,95	2,96	3,73
V_{ap} (quantité d'eau au-dessus du point le plus bas du système chauffant électrique)	litres	76	122	173
V_{es40} (quantité d'eau disponible à 40 °C)	litres	225	380	455

2.3 Installation des régulateurs et instrumentation

Le choix s'est porté sur le modèle **Vitosolic200 SD4** .C'est une version OEM de la société **RESOL** développée pour VIESSMANN. Elle permet à l'aide d'un bus de donnée **VBUS** propriétaire documenté d'accéder à l'ensemble de données mesurées et aux paramètres de commande. Le régulateur a pour rôle de moduler la commande du circulateur solaire en fonction de la différence de température entre la sonde d'entrée et de sortie du capteur solaire. Il met le système en sécurité lorsque la température maximale (paramétrable) est atteinte.

Le régulateur **Vitosolic200 SD4** relève les mesures

- D'ensoleillement.
- De températures.
- De compteur volumétrique de débit solaire et demande d'eau chaude.
- De calorimètre d'énergie entrante et sortante.
- De commande PWM du circulateur.
- La consommation électrique de l'appoint est contrôlée par un wattmètre à sortie numérique MODBUS RS485 indépendant du régulateur.



1-Régulateur solaire VITOSOLIC 200SD4

2-Wattmètre numérique SDM120C

2.4 *Installation de la métrologie / commande*

La métrologie/commande a pour objectif de contrôler le profil de consommation et de permettre le calcul du bilan de performance comparatif entre les chauffe-eau en test.
Elle est composée :

- d'un ordinateur type PC DELL .
- d'une boite d'acquisition de données et de commande type Keysight 34972A, connectés au réseau Ethernet.

L'expérimentation est pilotée par un logiciel d'acquisition développée avec l'environnement Keysight VEE et des fichiers de mesure en partage réseau sur les nano ordinateurs de gestion (RASPBERRY).

1-La métrologie autonome et la commande

2.5 Installation du régulateur non-prédictif et prédictif RASPBERRY

Sur chaque chauffe-eau un nano ordinateur RASPBERRY communique via 2 interfaces USB avec :

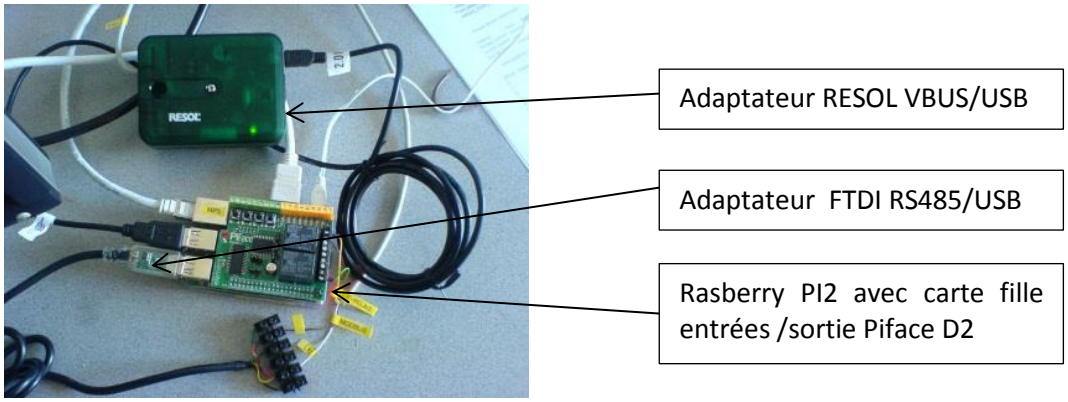
- Le régulateur VITOSOLIC 200SD4 à l'aide d'un adaptateur RESOL VBUS/USB
- Le wattmètre SDM120C à l'aide d'un adaptateur FTDI RS485/USB

Deux cartes filles d'extensions complètent le régulateur :

- Une carte extension entrées /sortie Piface D2
- Une carte extension horloge RTC PIFACE CLOCK
-

Caractéristiques :

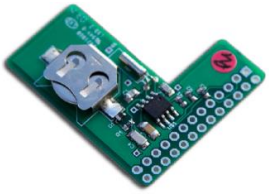
- RASPBERRY PI 2
- RAM de 1Go
- Stockage sur carte mémoire micro SD 32Go



1-Vue d'ensemble du Montage

2-RASPBERRY PI 2

3-PIFACE D2 E/S Numérique



4-Hologe RTC PIFACE Clock

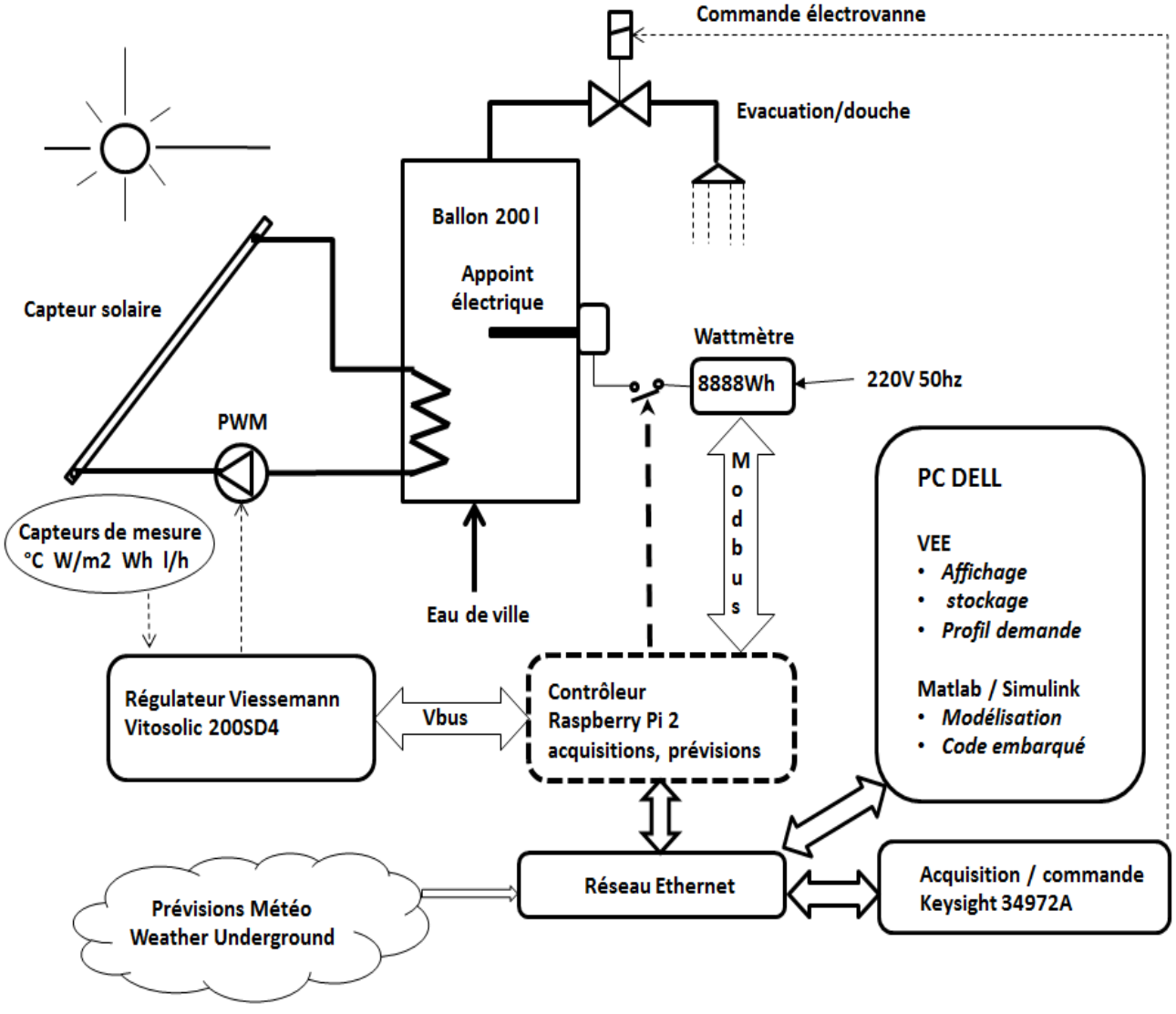
Logiciel et Operating Système

Le Raspberry est équipé d'une carte mémoire mini SD de 32 Go. Elle stocke l'OS, les développements spécifiques en Matlab, Python, Bash, C++ ainsi que les données de mesures et les résultats de calcul. L'Operating Système est RASPBIAN fourni par MATLAB 2015a pour le développement du code du contrôleur prédictif.

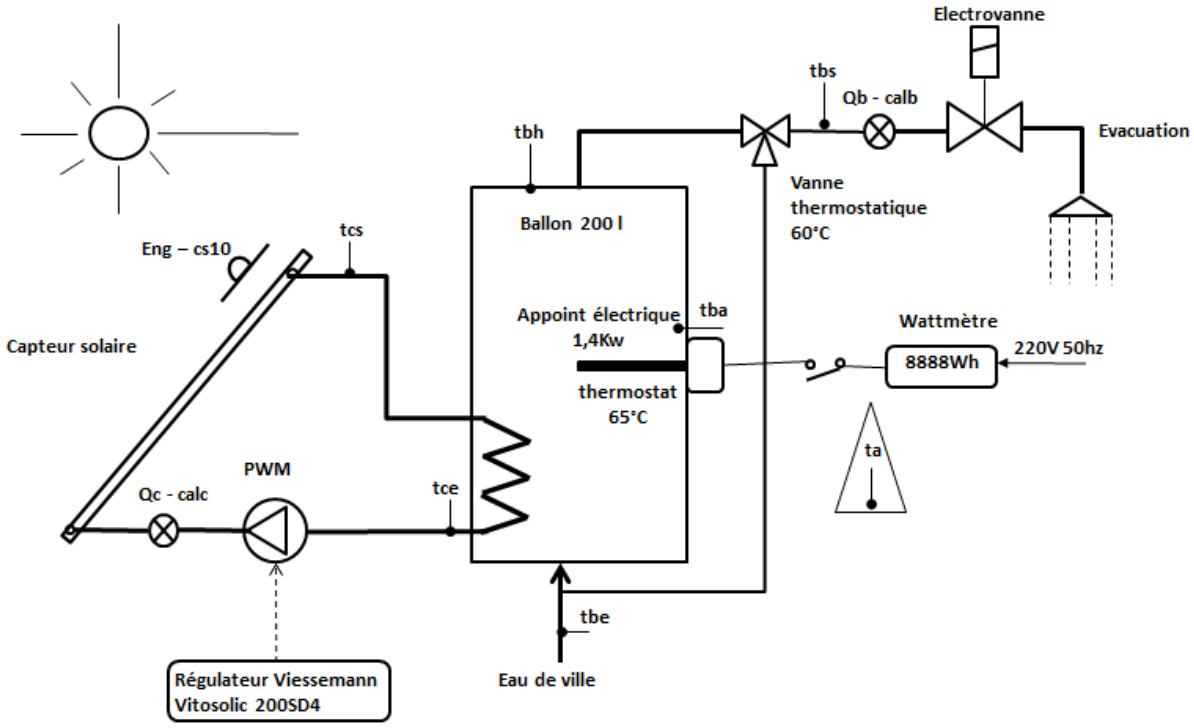
2.6 Principe de fonctionnement de la plateforme

Le réseau Ethernet est au cœur du fonctionnement de la plateforme.
La plateforme est conçue pour utiliser au mieux la communication de données numérique, mais peut si nécessaire accueillir des données analogiques supplémentaires au travers d'interface adaptées.
La maquette embarquée dans le Raspberry PI est constituée d'un ensemble de processus et de code pour le traitement des données en provenance du système à énergie renouvelable et des prévisions météo recueillies sur internet. Un calcul en code Matlab embarqué va périodiquement décider de l'état de l'actionneur, ici un relais de commande pour l'appoint électrique afin d'optimiser la consommation.

Synoptique



2.7 Schéma de l'installation d'un chauffe-eau



Remarque :

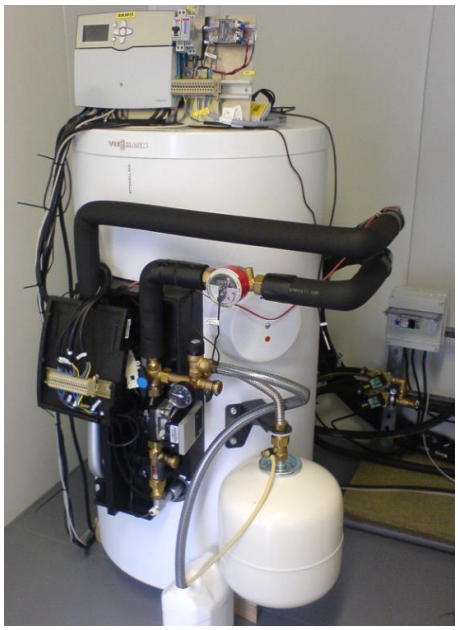
- La vanne thermostatique à 60°C a pour fonction de limiter la température maximum injectée sur le réseau d'eau chaude du logement.
- L'arrivée eau de ville est équipée d'un groupe de sécurité à 7 bars avec vidange, obligatoire pour limiter la pression lorsque le ballon s'élève en température et en amont d'un régulateur de pression à 3bars pour compenser les fluctuations de pression de l'alimentation générale.
- L'appoint est équipe en standard d'un thermostat de régulation réglé à 65°C.

Nomenclature

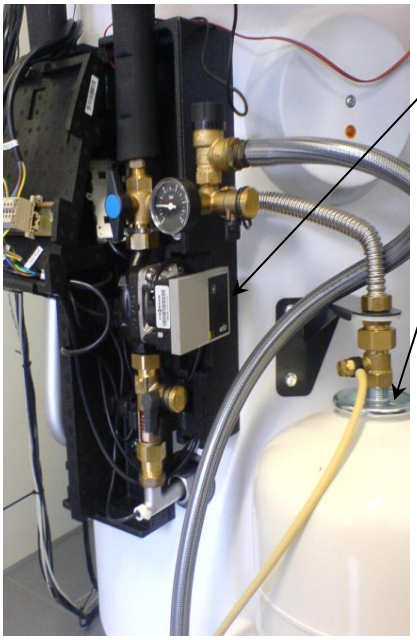
	Désignation	Voies Vito200	Références	Unités
1	tcs	S1	Température capteur sortie	°C
2	tce	S2	Température capteur entée	°C
3	tbh	S3	Température ballon haut	°C
4	tba	S4	Température ballon appoint	°C
5	ta	S5	Température ambiante intérieure	°C
6	tbs	S11	Température ballon sortie	°C
7	tbe	S12	Température ballon entée	°C
8	Eng – cs10	13	Ensoleillement global dans le plan du capteur	W/m²
9	Qc	14	Compteur volumique	Litre
10	Qb	15	Compteur volumique	Litre
11	PWR1	16	Commande circulateur capteur	%
12	Wattmètre	**Sdm120C**	Consommation appoint	Wh

Calorimétrie	Désignation	Compteur	Sondes	cp	Unités
Circuit capteur	Calc	Qc	tcs - tce	Eau glycolée	Wh
Circuit distribution	Calb	Qb	Tbs - tbe	eau	Wh

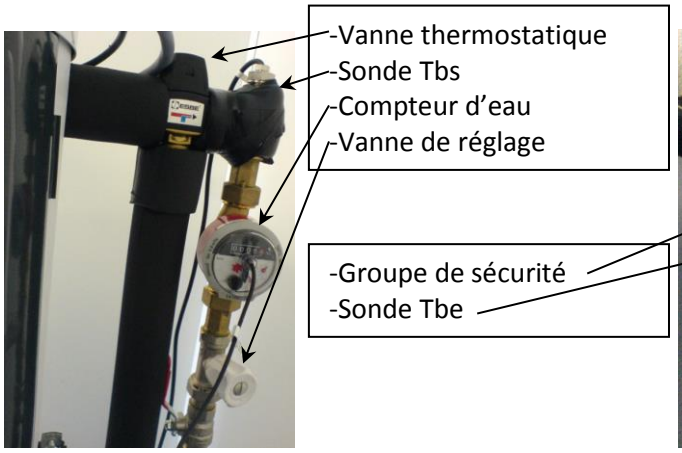
Le matériel installé



1-Ballon vue d'ensemble



2-Détail circulateur et vase d'expansion



3-Sortie ballon

4-Entrée Ballon

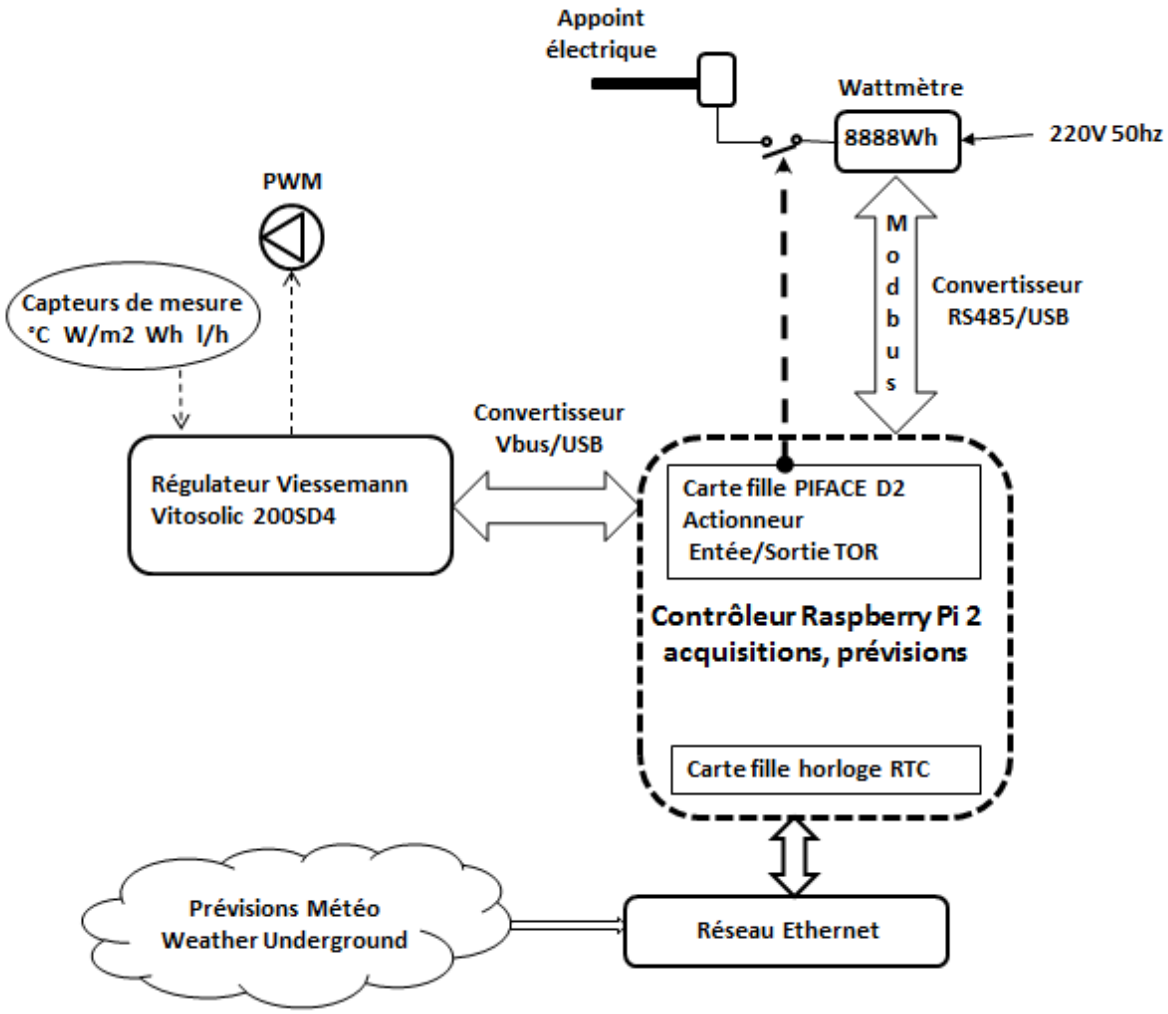
5-Electrovannes et arrivée d'eau

6-Sortie capteur. Pyranomètre et sonde sortie

2.8 Schéma fonctionnel du régulateur avec contrôleur prédictif ou non prédictif

L'environnement développé est appliqué ici au contrôle d'un chauffe-eau solaire mais reste applicable à tout autre système de gestion de production avec stockage à énergie renouvelable.
L'objectif est d'optimiser la commande de l'appoint électrique.

Vue d'ensemble



On retrouve l'ensemble des éléments d'un système de contrôle/commande à savoir :

- Une horloge sauvegardée.
- Une métrologie.
- Des actionneurs.
- D'un code exécutable de prédiction pour la commande développé avec Matlab/Simulink.
- Une connexion Ethernet pour le partage et la récupération de données.

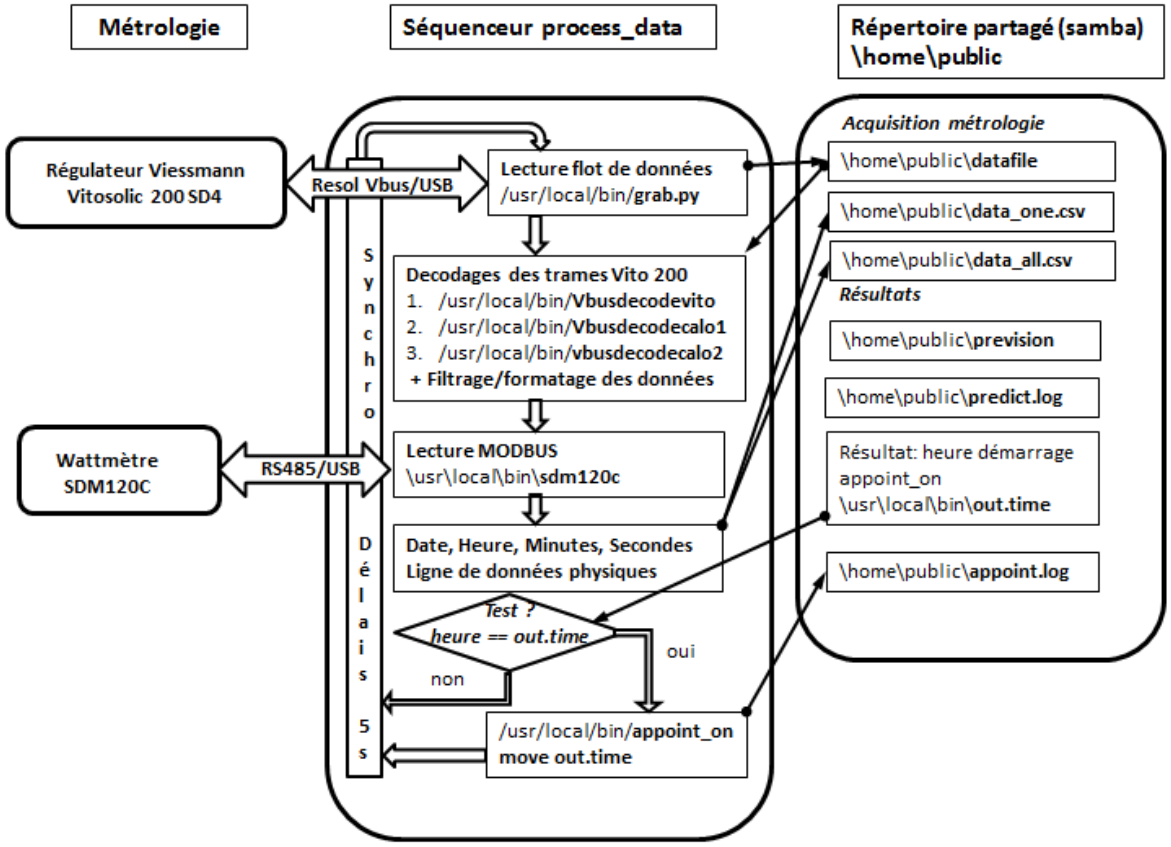
Architecture logicielle du contrôleur
(Voir annexe 6.3)
Les ressources qui ont inspirées ce qui suit sont :

- Pour le protocole vbus du régulateur Vitosolic 200 SD4
  https://piamble.wordpress.com/tag/vbus/
- Pour le protocole MODBUS du wattmètre sdm120c
  https://github.com/gianfrdp/SDM120C

Objectifs :

- Il s'agit de mettre en place un automate de contrôle/commande sur une plateforme Linux Raspbian fournie par Matlab.
- L'automate est composé de 2 séquenceurs qui démarrent des taches périodiques.
- Un séquenceur logiciel avec une période de 10 secondes **process_data** cadence les tâches d'acquisition à exécuter ainsi que la mise en marche calculée de l'appoint.
- Un séquenceur système, crontab, gère les tâches journalières à effectuer à heures fixes.

Organigramme du séquenceur process_data



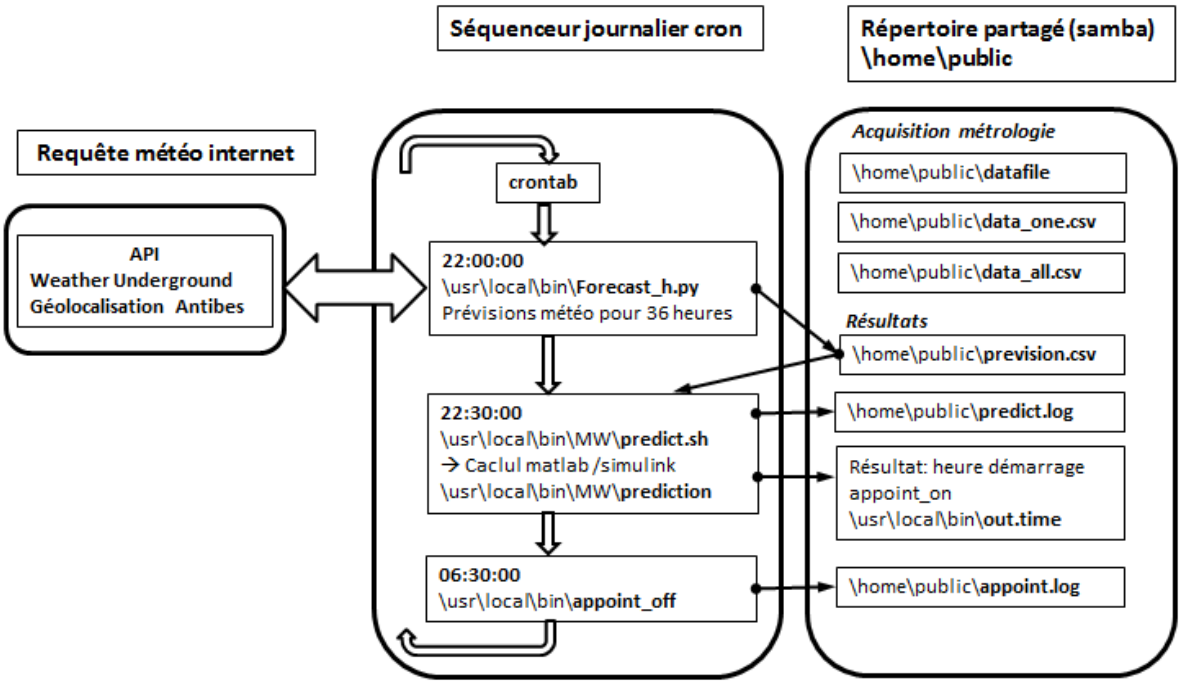
Fonctionnement

Le séquenceur **process_data** est un script bash qui s'exécute en boucle infinie.

1. La tâche **grab.py** lit un flot de données hexadécimales calibrées issues du vbus et le stocke en sortie dans le fichier **datafile**.
2. Un ensemble de tâches vbusdecode permet d'extraire du fichier datafile les trames :
   - De données de mesure standard **vbusdecodevito**
   - De données du calorimètre 1 pour le circuit capteur **vbusdecodecalo1**
   - De données du calorimètre 2 pour la sortie ballon **vbusdecodecalo2**
   - Les données sont formatées en une ligne csv en vue de leur stockage

3. La tâche **sdm120c** envoi des requêtes et lit les données issues du MODBUS.
4. La date et du temps sont relevés
5. Si aucune erreur de transmission n'est détectée la ligne de mesure est écrite dans les fichiers
   - **data_one.csv** (une seule mesure pour l'ensemble des capteurs)
   - **data_all.csv** (concatenation de data_one.csv)
6. Le **test** du fichier **out.time** et de son contenu (heure de démarrage de l'appoint) va déclencher la commande **appoint_on** ou **non**
   - **Si** l'appoint est enclenché **alors** pour permettre un nouveau cycle d'appoint **out.time** est déplacé puis renommé.
7. Un **délai** de 5 secondes d'attente va permettre une synchronisation de la période de fonctionnement à 10 secondes.

Organigramme du séquenceur système prédictif cron
(Voir annexe 6.4)



Fonctionnement

Le séquenceur **cron** est un service de l'operating système **linux Raspbian** installé sur le **Raspberry**. L'application **crontab** permet sa configuration.

1. L'application forecast_h.py de Weather Underground
   http://www.wunderground.com/weather/api/d/docs
   fournie les 36 prévisions horaires futures et les stocke dans le fichier **prevision.csv.**
2. Le script bash predict.sh lance le calcul **Matlab/Simulink prédiction,** le fichier **out.time** est créé. Il stocke l'heure de démarrage de l'appoint**.** L'historique est stocké dans le fichier **predict.log**
3. A la fin des heures creuses EDF soit 6h30, la commande **appoint_off** stoppe l'appoint. L'historique des commandes d'appoint sont stocké dans le fichier **appoint.log**

Remarque :
Pour le **chauffe-eau non prédictif,** le calcul de prédiction est remplacé par la commande **appoint_on**. Le thermostat de l'appoint est alors chargé de faire la régulation en température de la partie haute du ballon (au-dessus de la résistance électrique) durant la période des heures creuses.

2.9 Code de calcul embarqué de prediction Matlab/Simulink.

(Voir annexe 6.5)

Pourquoi utiliser Matlab pour développer un contrôleur prédictif ?

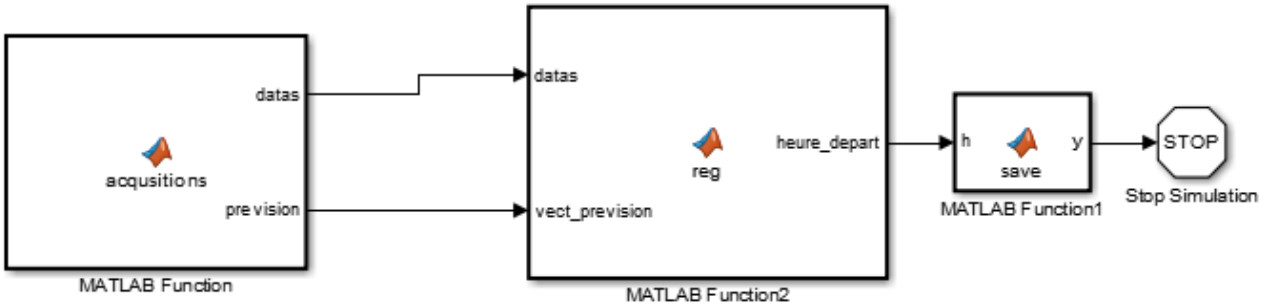
Le but de la présente maquette à travers l'exemple du contrôle prédictif d'un système à énergie renouvelable est d'envisager une plateforme plus générique à vocation expérimentale et pédagogique.
L'aspect pédagogique de la plateforme nécessite de la rendre accessible à des élèves de tout horizon.
Ce qui correspond bien avec l'utilisation Matlab qui est un outil très rependu dans l'enseignement.
Ce choix va permettre par exemple de créer un modèle du système à énergie renouvelable, ici un chauffe-eau solaire, dans lequel il sera aisé d'extraire le code de calcul du contrôleur prédictif.
Ainsi le contrôleur prédictif testé en environnement de simulation pourra aisement être validé sur le chauffe-eau solaire réel.
Ce choix n'empêche pas que la plateforme reste par sa conception très ouverte à l'utilisation d'outil de programmation autre tel que le Python et le C++.

Préalable bibliographique et initiation:

Il est necessaire d'acquérir quelques connaissance de base sur le développement de code embarqué avec Matlab/Simulink.
Voir :
http://fr.mathworks.com/help/supportpkg/raspberrypiio/examples/getting-started-with-matlab-support-package-for-raspberry-pi-hardware.html?s_tid=gn_loc_drop

La prise en main à travers quelques exemples fournis par Mathworks permet de maitrisé les différentes opérations de test de compilation et de recherche de bug.
Voir :
https://www.youtube.com/watch?v=J3ZpB0SfbNA
http://makerzone.mathworks.com/raspberry-pi/

Organigramme Simulink du contrôleur prédictif



Le but de du code à générer et à exécuter sur le Raspberry en mode «autonome » est de calculer de l'heure de démarrage de l'appoint sachant l'heure de fin 6h30.
Le code exécuté qu'une fois par jour est composé de 3 fonctions :

Acquisitions :

- Lit les fichiers dynamiques au format caractère du répertoire partagé du Raspberry data_one.csv et previsions.csv.
- Stocke les données dans une structure type tableau ou record.

Remarque : Simulink ne possède pas de fonction pour la lecture de fichier d'entrées sur Raspberry. Il nécessaire d'utiliser des fonctions en C pour réaliser cette opération.

Reg :

- Constitue le contrôleur. Elle effectue le calcul de l'heure de démarrage de l'appoint en fonction des données d'entrée et de la prévision météo et du créneau des « heures creuses » EDF soit 22h30 à 6h30.

Save :

- Effectue la sauvegarde du résultat du calcul dans le fichier out.time.

2.10 Métrologie

Elle utilise toute la puissance que nous offre linux à savoir :

- Une connexion Ethernet.
- Le partage de fichier Linux/Windows avec un **serveur de fichier Samba** sur chaque Raspberry.

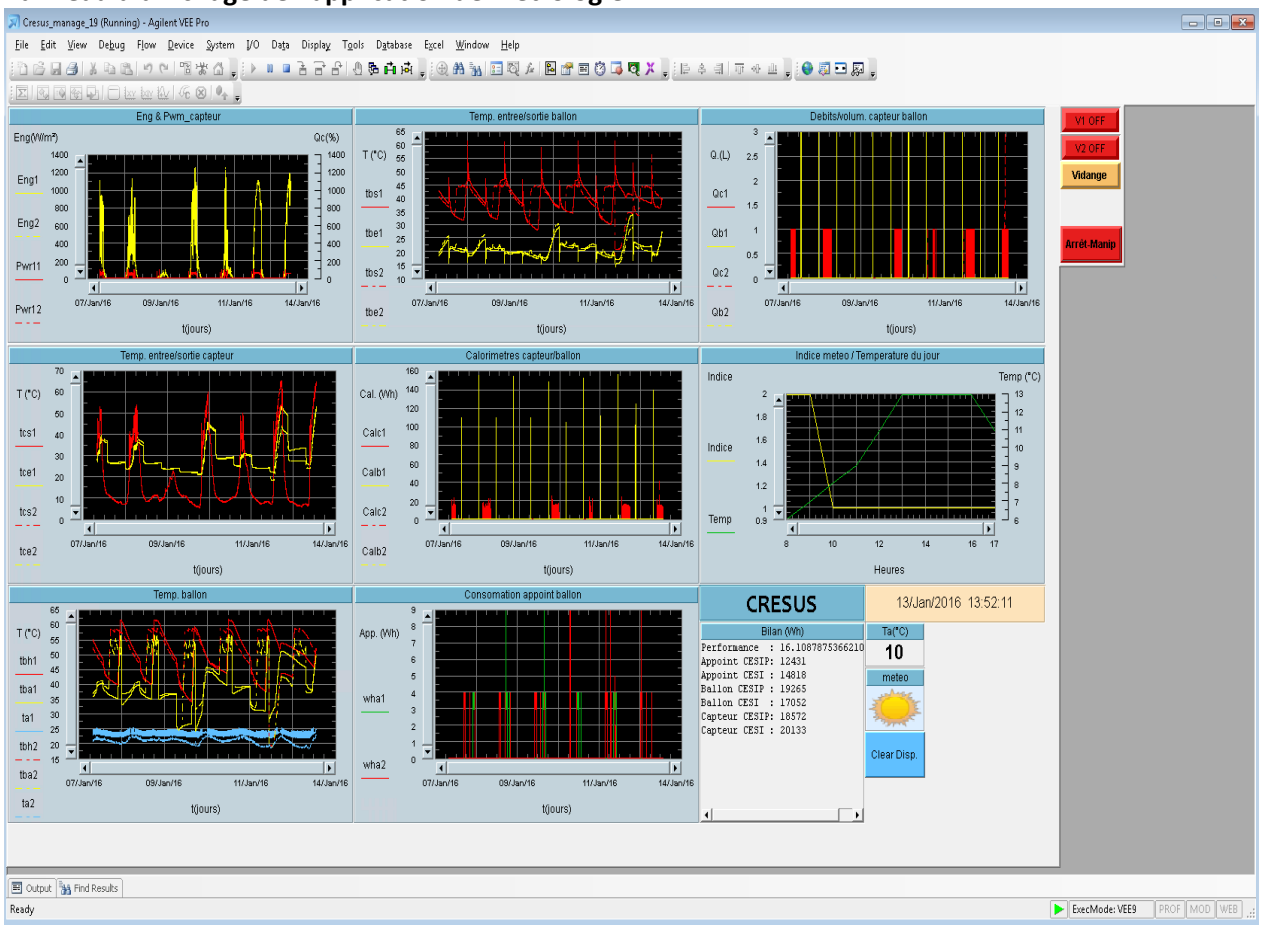
Dans ces conditions la récupération des données mesurées et des prévisions se résume à la lecture des fichiers data_one.csv et prévision.csv sur les répertoires partagés des 2 Raspberry.
La ligne de mesure du fichier **data_one.csv** a pour entête :

Date et heure ; tcs ; tce ; tbh ; tba ; ta ; tbs ; tbe ; Qc ; Qb ; Pwr1 ; eng ; calc ; calb ; wattmetre
(Voir §2.7)

Une application Keysight VEE, **cresus_manageXX.vee** permet :

- De paramétrer un modèle d'expérimentation.
- De contrôler un multiplexeur d'acquisition /commande Keysight 34972A.
- De piloter les électrovannes manuellement.
- De gérer un profil de soutirage automatique pour chaque chauffe-eau.
- De comparer les données en temps réel et de les stockées après traitement.
- De rejouer rapidement une séquence de plusieus jours.
- De calculer les bilans de comptage d'énergie sur une séquence donnée.

Panneau d'affichage de l'application de métrologie



3 Développement d'un régulateur prédictif par simulation

3.1 Objet

Le test d'un régulateur prédictif suppose que l'on connaisse à priori quelle est la loi qui permet de décider du comportement du régulateur en fonction des prévisions météorologiques du lendemain.

Nous avons pour cela développé un simulateur du comportement thermique d'un dispositif de production d'eau chaude sanitaire au pas de temps horaire. Ce simulateur reprend les caractéristiques du CESI Viessmann Vitosol 200-F telles que décrites au § 2.2 . Nous entreprendrons 4 types de simulation :

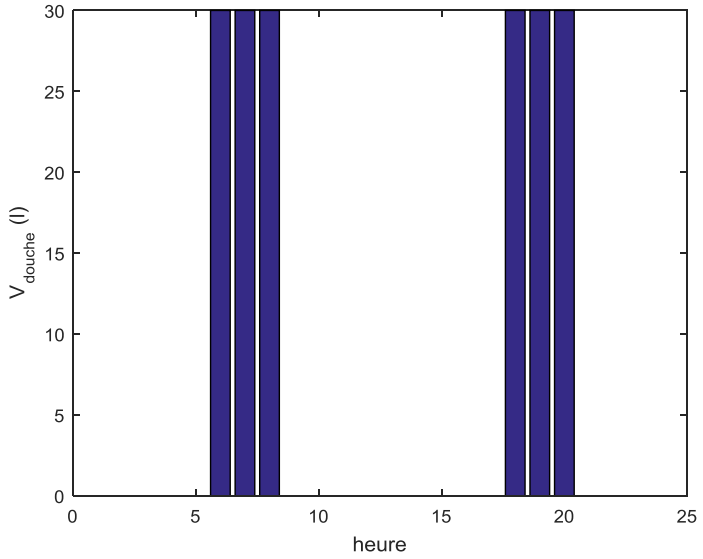
- Simulation d'un Chauffe-Eau Individuel sans capteur solaire (CEI)
- Simulation d'un Chauffe-Eau Individuel avec capteur solaire, régulation non prédictive (CESI)
- Simulation d'un Chauffe-Eau Individuel avec capteur solaire, régulation prédictive optimale (CESI-PredOpt)
- Simulation d'un Chauffe-Eau Individuel avec capteur solaire, régulation prédictive simplifiée (CESI-PredS)

Le but de ces simulations est d'une part d'établir une loi de commande prédictive simple à embarquer dans le régulateur Rapsberry, d'autre part de comprendre et de quantifier les performances de la prédiction en termes de bilan énergétique.

3.2 CEI

3.2.1 Profil de soutirage

Chaque jour, 3 douches d'un volume V_{douche} = 30 litres sont programmées matin et soir, suivant le profil suivant :



Chaque douche demande de l'eau à une température T_{douche} = 35 °C. Cette température est obtenue en mitigeant de l'eau soutirée dans le ballon (volume S_{douche} à température T_{ballon}) avec de l'eau du réseau (à température T_f). S_{douche} est obtenu par :

$$S_{douche} = V_{douche} \frac{T_{douche} - T_f}{T_{ballon} - T_f}$$

Nous faisons l'hypothèse d'un stock brassé (ballon à température uniforme), de telle sorte que la perte enthalpique correspondante au niveau du ballon vaut :

$$Q_{douche} = S_{douche} \left(T_{ballon} - T_f\right) \rho \, C_p$$

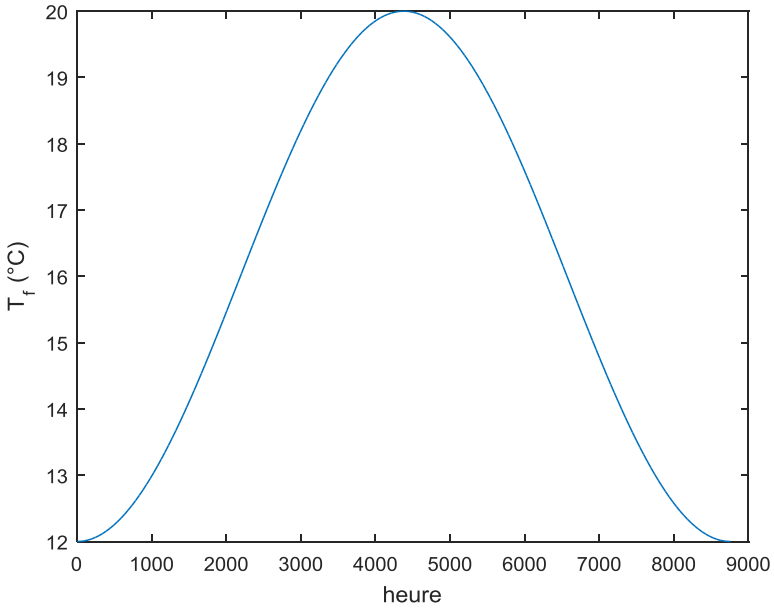
(ρ et C_p désignant la masse volumique et la chaleur massique de l'eau)

Cette équation est valable à tous les pas de temps, sachant que s'il n'y a pas soutirage ($S_{douche} = 0$), on a simplement $Q_{douche} = 0$.

Précisons que nous avons volontairement choisi de simuler une famille "économe" : la consommation annuelle correspondante est de 365*180 l soit 65 m^3, alors que l'Ademe estime cette consommation entre 120 et 200 m^3 par foyer.

3.2.2 Température d'eau froide

La température d'eau froide évolue selon une sinusoïde qui va de 12 °C en hiver à 20 °C en été :



3.2.3 Energie d'appoint

L'énergie d'appoint est fournie par une résistance électrique. Celle-ci s'enclenche chaque jour à 23 heures (démarrage des heures creuses) et peut durer jusqu'à 7 heures du matin (fin des heures creuses). La puissance nominale (1500 W) est apportée au ballon jusqu'à ce que sa température dépasse une consigne $T_{consigne}$ fixée à 55 °C.

3.2.4 Résultats

Le bilan énergétique du CEI sur une année s'établit comme suit :

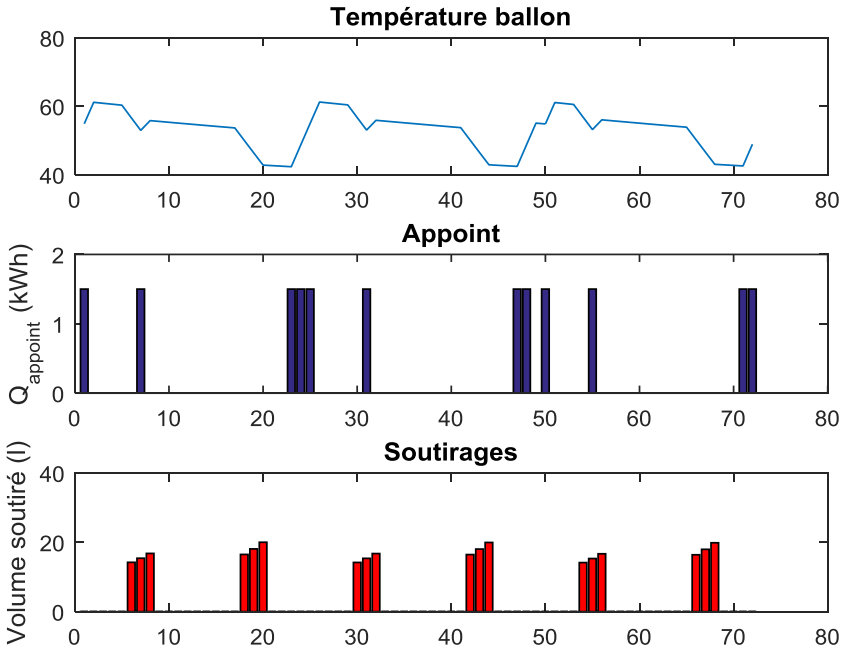
```
CEI :

Conso Electrique : 1890.000000 kWh
Besoins pour la douche 1437.696221kWh
Déperditions : 452.206322 kWh (23.926260 % de la consommation)
Nbre de douches froides : 0
```

On remarque :

- Malgré le profil "économe" du foyer simulé, la facture demeure significative (> 200 € à 11 c€/kWh, tarif heures creuses hors abonnement)

- Les déperditions sont loin d'être négligeables, représentant près du quart de la consommation d'énergie, bien que nous ayons supposé une température ambiante de 20 °C pour le calcul de ces déperditions.

Si nous zoomons sur une séquence particulière de 3 jours (journées 20, 21, 22), on observe le comportement suivant :



La température du ballon est caractérisée :

- Par des parties à pente négative faible, correspondant aux déperditions (environ 2 °C en 9 heures)
- Par des parties à pente négative forte, correspondant aux soutirages
- Par des parties à pente positive forte, correspondant à la mise en marche de l'appoint.

3.3 CESI

3.3.1 Modélisation du capteur

Le rendement du capteur est donne par la formule normalisée :

$$\eta = a_0 - \frac{a_1(T_m - T_{ext})}{I} - a_2\ 800\left[\frac{T_m - T_{ext}}{I}\right]^2$$

Avec T_{ext} = température extérieure et $T_m = \frac{T_e+T_s}{2}$ température moyenne du capteur dans lequel le fluide caloporteur entre à T_e et sort à T_s. I est l'irradiation dans le plan du capteur (W/m^2).

Les coefficients a_0 (rendement optique), a_1 et a_2 sont fournis dans la documentation du constructeur (cf § 2.2).

L'énergie captée vaut :

$$Q_{cap} = \eta\ I\ A$$

Où A est la surface du capteur.

Cette énergie est transférée au ballon par l'intermédiaire d'un échangeur dont le rendement est pris égal à 0,95 :

$$Q_{sol} = Q_{cap} * 0.95$$

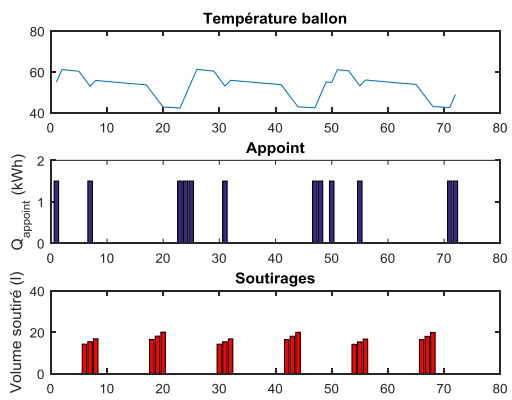
La température du ballon est limitée pur des raisons de sécurité à 70 °C.

3.3.2 Résultats

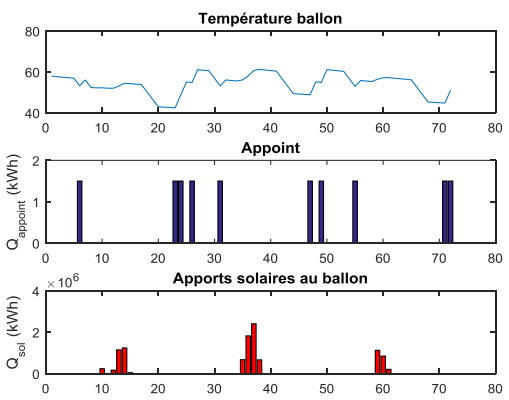
Dans des conditions de simulation identiques à celles du CEI, nous obtenons le bilan suivant :

	CEI	CESI
Conso. Appoint électrique (kWh)	1890	669
Apport solaire au ballon (kWh)	-	1338
Déperditions ballon (kWh)	452	572
Nbre douches froides	0	0

Comme on le voit, le capteur solaire diminue la consommation électrique de 65 %. Cependant, on remarque également que les déperditions augmentent sensiblement. Ceci s'explique simplement en comparant le comportement du chauffe-eau sur la séquence des jours 20-21-22 :



CEI



CESI

Nous constatons bien que le recours à l'appoint est nettement diminué dans le cas du CESI. Par contre on remarque également que la température du ballon, notamment le jour 2 caractérisé par un fort apport solaire, augmente sensiblement. Cela va permettre de réduire le recours à l'appoint le jour suivant, mais en contrepartie les déperditions du ballon sont également majorées.

3.4 CESI_PredOpt

3.4.1 Fonctionnement

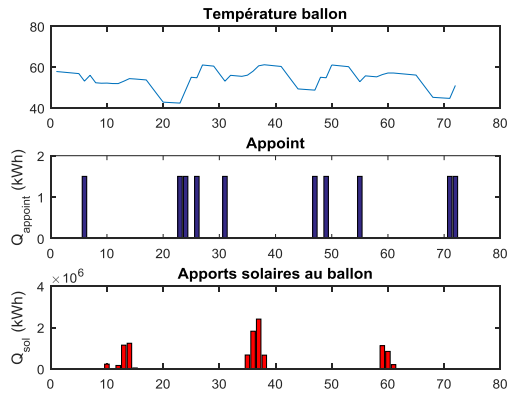
Le CESI prédictif optimal n'est imaginable qu'en simulation. Il est basé sur le principe suivant :

- Un CESI non prédictif est caractérisé par un appoint qui s'enclenche tous les jours à 23 h, et un apport solaire qui maintient dans le ballon un niveau de température inutilement haut. La seule contrainte en termes de confort est que chaque jour on dispose au moment de chaque douche d'une température ballon supérieure ou égale à T_{douche} (35 °C). Or on obtient dans le cas du CESI des températures très souvent supérieures à 40, voir à 50 °C pendant toute une journée.
- Le CESI prédictif optimal va utiliser un algorithme récursif pour limiter cette température. Sachant que la période d'heures creuse se termine à 7 h :
    - On lance chaque jour à 23 h une simulation portant sur toute la journée du lendemain (la météo est connue puisque nous sommes en environnement simulé)
    - Si {$T_{ballon} \geq T_{douche}$ *pendant toutes les heures de soutirage*}, alors c'est qu'il n'y avait pas besoin de recourir à l'appoint. On valide cette option et on passe à la journée suivante.
    - Si ce n'est pas le cas, on relance une simulation en utilisant l'appoint pendant une durée DA = 1 h sur le créneau de fin des heures creuses [6h – 7h].
    - On renouvelle le test {$T_{ballon} \geq T_{douche}$ *pendant toutes les heures de soutirage*} en incrémentant à chaque fois DA=DA+1 s'il n'est pas satisfait.

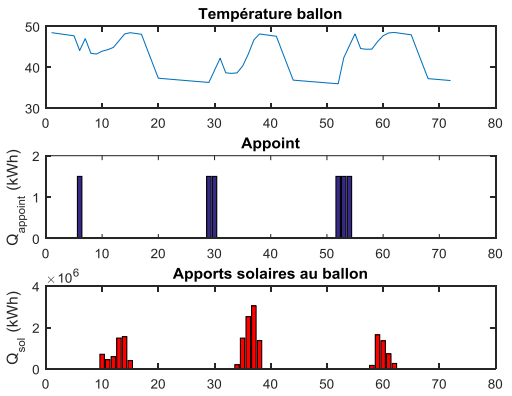
On parvient ainsi à une utilisation optimale de l'appoint électrique :

	CEI	CESI	CESI_PredOpt
Conso. Appoint électrique (kWh)	1890	669	262
Apport solaire au ballon (kWh)	-	1338	1673
Déperditions ballon (kWh)	452	572	504
Nbre douches froides	0	0	0

La consommation d'énergie électrique est réduite à son minimum (262 kWh, soit 28 €), et les déperditions, tout en restant supérieures à celles du CEI, sont nettement diminuées par rapport à celles du CESI.



CESI



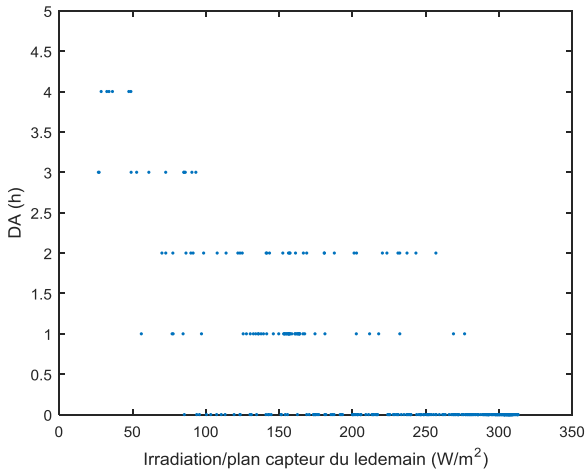
CESI_PredOpt

Comme on le voit sur la séquence des jours 20-21-22, la température du ballon est ramenée à sa valeur minimum (35 °C pendant les soutirages), et la consommation d'appoint électrique est fortement réduite. On remarque également que les apports solaires sont nettement augmentés, ce qui est dû au fait qu'une température plus basse du ballon va se traduire par une température plus basse de la température d'entrée dans capteur T_e , ce qui augmente le rendement de captation de ce dernier.

3.4.2 Analyse

La version optimale du régulateur prédictif se traduit par un calcul de la durée d'utilisation de l'appoint DA calculée de façon récursive. Dans le but de pouvoir développer une version simplifiée de ce régulateur, analysons la façon dont DA dépend des prédictions du lendemain.

Dans notre cas, DA s'échelonne de 0 à 4 heures. Il est logique de penser que le principale variable explicative soit l'ensoleillement prévu le lendemain. Toutefois si l'on regarde cette dépendance en détail, on obtient la figure suivante :

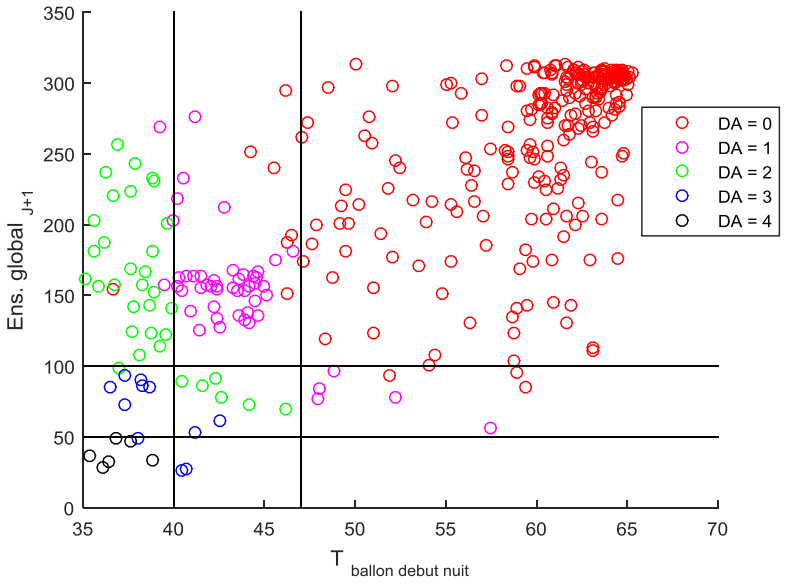


Si l'on observe globalement une tendance logique à ce que DA soit plus élevée pour de faibles valeurs de l'irradiation du lendemain, on observe également que cette valeur prend les valeurs 1 ou 2h de façon indifférenciée. On conçoit que cela soit dû au fait que cette valeur soit liée non seulement au futur (quelle sera la valeur de l'ensoleillement du lendemain ?) mais également au passé : une séquence de mauvaises journées va inévitablement provoquer un épuisement progressif de la charge du ballon, de telle sorte que même si un fort ensoleillement est prévu le lendemain, il faudra utiliser 1 ou 2 h d'appoint pour éviter toute douche froide.

Nous pouvons établir une cartographie de la valeur optimale de DA en fonction des 2 variables explicatives :

- T_{ballon} en début de la période d'heures creuses (23 h) : cette variable est représentative de l'état de charge du ballon
- Irradiation moyenne dans le plan du capteur prévue pour le lendemain

On obtient la figure suivante pour l'ensemble des 364 points :



- Les points rouges en haut à droite correspondent aux points où l'ensoleillement du lendemain est élevé et le ballon est chargé en début de nuit : l'algorithme prévoit DA = 0, c'est-à-dire pas d'utilisation de l'appoint.
- A l'inverse les points noirs en bas à gauche sont des points à faible ensoleillement au jour J+1 et avec stockage déchargé en début de période heures creuses.

Entre ces 2 extrêmes, on peut classifier l'ensemble des points en utilisant un tableau à 9 zones :

Valeur optimale de DA (h)		T_{ballon} (23 h.)		
		< 40 °C	[40 – 47] °C	> 47 °C
Ens. Global moyen (J+1) (W/m^2)	> 100	**2**	**1**	**0**
	[50-100]	**3**	**2**	**1**
	< 50	**4**	**3**	**2**

Il est maintenant possible de tester un CESI prédictif simplifié, c'est-à-dire un CESI qui ne va pas utiliser d'algorithme d'optimisation récursif.

3.5 CESI_PredS

Le CESI prédictif simplifié utilise donc chaque jour à 23 h uniquement 2 valeurs :

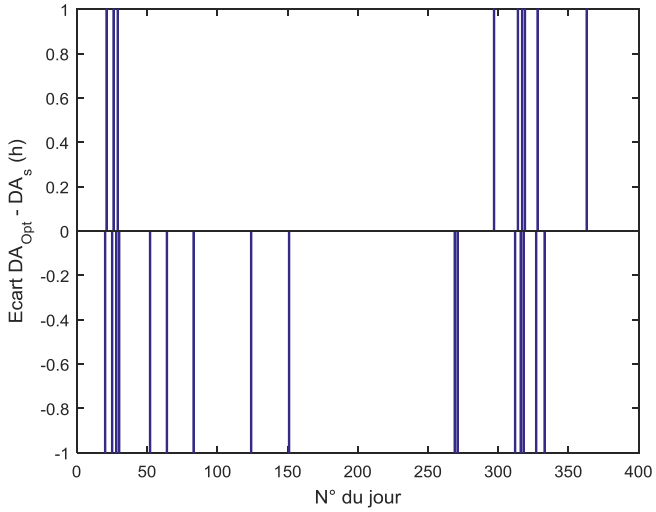
- Une valeur issue de la prédiction météo, et qui classifie simplement le jour J+1 en 3 catégories. On peut par exemple utiliser la nébulosité (forte, moyenne, ou faible)
- La mesure de la température du ballon.

On obtient dans ces conditions le bilan suivant :

	CEI	CESI	CESI_PredOpt	CESI_PredS
Conso. Appoint électrique (kWh)	1890	669	262	271
Apport solaire au ballon (kWh)	-	1338	1673	1665
Déperditions ballon (kWh)	452	572	504	506
Nbre douches froides	0	0	0	2

Comme on peut le constater, le CESI prédictif simplifié a des performances quasi-identiques à celle du prédictif simplifié. Par rapport au CESI non prédictif, il réduit l'énergie d'appoint de 40 % contre 41 % pour le prédictif optimal.

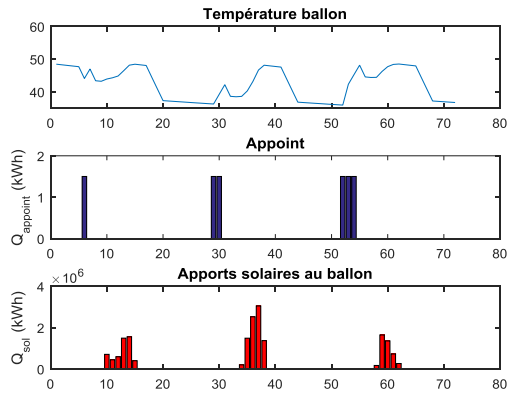
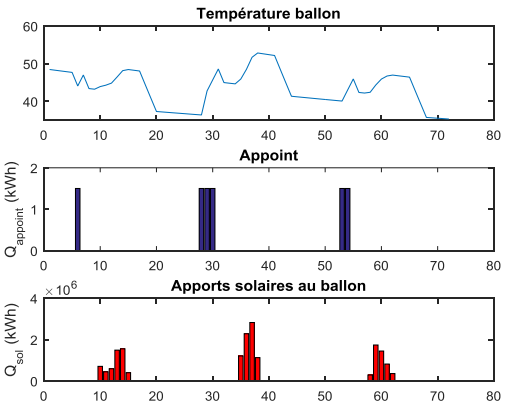
Il y a au total 25 jours où le régulateur predictif simplifié calcule une valeur de la durée d'utilisation de l'appoint DA différente de celle du régulateur prédictif optimal. Si l'on appelle respectivement DA_{Opt} et DA_S les valeurs calculées par les 2 régulateurs, l'écart entre ces 2 valeurs se répartit comme suit :



Il vaut +/- 1 heure. Cet écart n'a pas d'incidence notable sur le bilan, par contre il est la cause de 2 "douches froides" :

```
Jour : 298 heure : 20 Douche froide à : 34.737978 °C
Jour : 364 heure : 20 Douche froide à : 32.267895 °C
```

Si l'on compare le comportement des 2 régulateurs sur la période des jours 20 à 22, on obtient :

	
CESI_PredOpt	**CESI_PredS**

- Le jour 21, CESI_PredS utilise une durée d'appoint de 2 heures contre 1 heure pour CESI_PredOpt. De ce fait, la température ballon va se trouver plus élevée, et par conséquent les apports solaires plus faibles du fait de l'abaissement du rendement.
- Le jour 22, la température plus élevée du ballon va conduire CESI_PredS à utiliser l'appoint pendant 2 heures contre 3 heures pour CESI_PredOpt. C'est la situation inverse du jour précédent, avec notamment un apport solaire plus fort pour CESI_PredS que pour CESI_PredOpt.

Ce comportement est important à analyser : les erreurs de prédiction ne s'accumulent pas, elles peuvent au contraire dans certains cas se compenser, ce qui explique que le régulateur simplifié obtienne en termes de bilan énergétique une performance quasi-égale à celle du régulateur optimal. Le risque encouru est celui de 2 douches froides sur les 2190 (6*365) que comporte la simulation.

4 Validation expérimentale

Le régulateur utilisé par notre modèle est défini par le tableau ci-dessous.
Définition du régulateur en fonction des contraintes du modèle et des données de prévisions

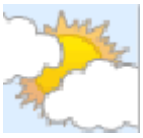
Valeur optimale de DA (h)		T_{ballon} (23 h.)		
		< 40 °C	[40 – 47] °C	> 47 °C
Ens. Global moyen (J+1) (W/m^2)	> 100	**2**	**1**	**0**
	[50-100]	**3**	**2**	**1**
	< 50	**4**	**3**	**2**

Weather Undergroud nous fournit des prévisions météorologiques de nébulosité heure par heure sur 36h. Nous retiendrons uniquement les previsions du lendemain entre 8h et 17h pour effectuer le calcul de la durée de fonctionnement de l'appoint du chauffe-eau prédictif.

Prévisions métérologique :


1


2 : > 100 W/m²

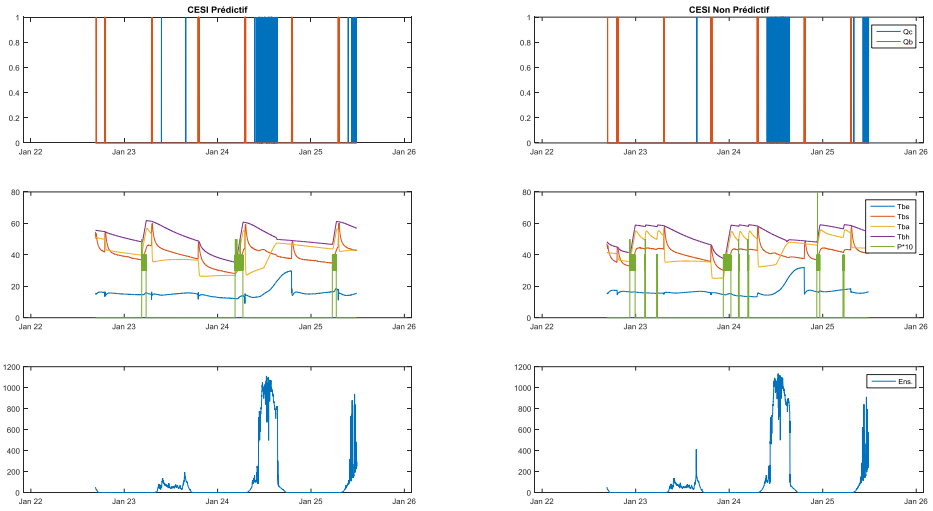

3 : [50-100] W/m²


4 : < 50 W/m²

Le régulateur embarqué dans le code executable Matlab du Raspberry sur le chauffe eau prédictif utilise le tableau modifié suivant :

Valeur optimale de DA (h)		T_{ballon} (23 h.)		
		< 40 °C	[40 – 47] °C	> 47 °C
Prévision moyenne	<2	**2**	**1**	**0**
	>=2 & <=3	**3**	**2**	**1**
	>3	**4**	**3**	**2**

4.1 *Analyse d'une sequence courte*



La figure ci-dessus illustre le comprtement des 2 CESI sur une séquence de 3 jours.

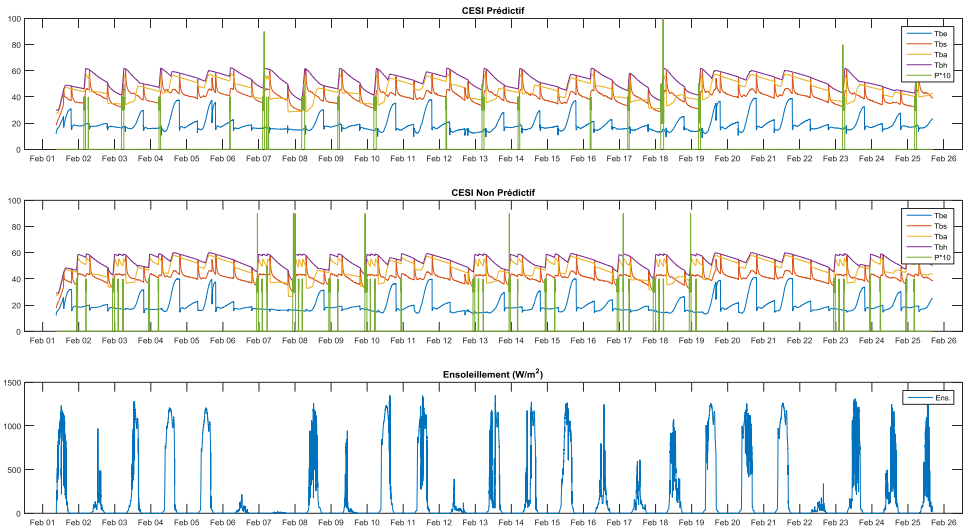
- Les figures de la 1ère rangée donnent l'évolution d'une part du débit de fluide dans les capteurs (Qc, en bleu), d'autre part du débit de soutirage dans le ballon (Qb, en rouge). Qc reste quasi-nul durant la 1ère journée, par manque de soleil, alors ue le circulateur fonctionne pendant la journée du 24 janvier, entre 9h30 et 16 h.
- Les figures de la rangée intermédiaire donnent pendant ces journéesl'évolution de Tbe, Tbs, Tba, Tbh, et P (suivant la nomenclature définie en 2.7). La différence de comportement des 2 chauffe-eau est caractérisée par la valeur de P, puissance d'appoint électrique (en vert). Si on analyse par exemple la première journée :

    - Pour le CESI non prédictif, l'appoint s'enclenche le 23 janvier à 22h30. Par la suite, un cyle de séquences marche-arrêt de la résistance permet de maintenir la température de référence de l'appoint (Tba) entre 50 et 55 °C. Il a consommé 2555 Wh durant la nuit.
    - Le CESI prédictif, quant à lui, choisit une durée de fonctionnement DA de 2 heures. Il s'enclenche donc à 4h30 du matin, et s'interromp une heure plus tard lorsque Ta atteint 55 °C. Sa consommation nocturne s'élève à 1615 Wh.

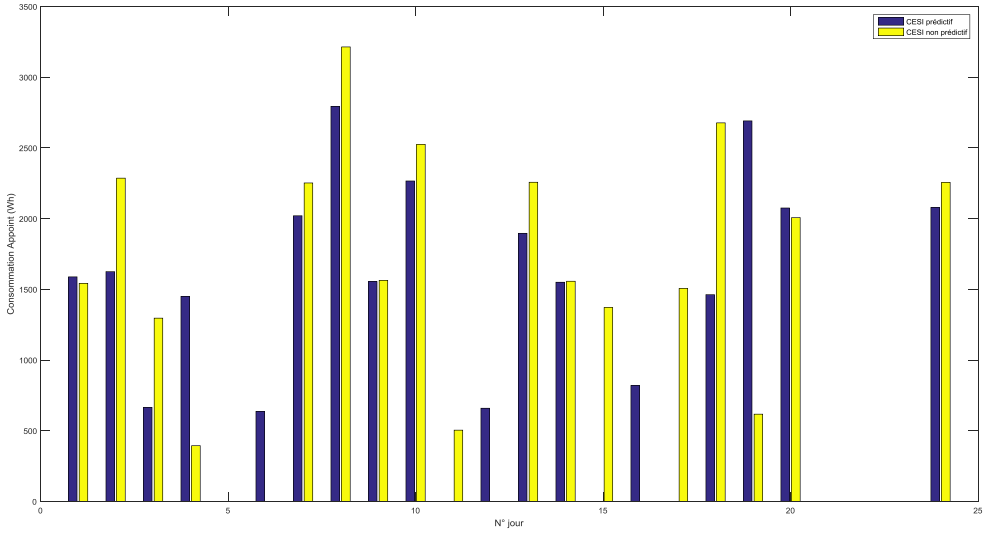
C'est donc sur cet exemple une économie de 37 % de l'énergie d'appoint qu'entraîne le recours à la prédiction.

4.2 *Analyse sur une séquence de 25 jours*

La séquence suivante est caractérisée par une dominance de journées très ensoleillées (> 1000 W/m^2 dans le plan des capteurs pendant 16 jours sur 25).



On observe sur cette séquence un comportement beaucoup pus varié. Si l'on fait tracer jour par jour les consommations des deux ballons, on obtient la figure suivante :



- Certains jours, le CESI prédictif réalise son économie d'appoint comme décrit au paragraphe précédent. Cette économie va typiquement de la fourchette 10-15 % (jours 7, 8, 10, 13, 24, …) jusqu'à 49 % (jour 3).
- A l'inverse, il est des jours où le CESI prédictif consomme plus que le CESI non prédictif. Cela s'explique simplement parce que, en maintenant une température moyenne de ballon plus basse que le non prédictif, le CESI prédictif va parfois se retrouver en fin de nuit amené à remonter la température du ballon, alors que le non prédictif aura une température plus élevée due au fait qu'il a dans les jours précédents dépensé plus d'énergie.

Au total sur l'ensemble de la séquence le CESI prédictif réalise une économie de 8 %. En termes de tendance, on remarquera que cette économie représente environ 3,5 kWh. Ramenée à une année, on parvient à une économie annuelle de 50 € (à 15 c€ le kWh).

Toutefois ce chiffre n'a d'autre intérêt que de fournir un ordre de grandeur. Les tests doivent se poursuivre sur des périodes plus longues. En outre nous avons utilisé un rélage simplifié du régultaeur prédictif, qui peut certainement être amélioré, car il y a dans la séquence analysée certains jours où il aurait été possible d'utiliser moins d'appoint sans risquer pour autant la douche froide.

5 Résultats et conclusion

Pour des raisons évidentes l'expérimentation devra être effectuée sur une durée suffisamment longue de plusieurs mois avec de nombreuses alternances de periode météo à nebulosité variable. On peut toutefois considérer que les 2 principaux objectifs de la maquette ont été atteints :

- L'environnement hard d'acquistion/commande et de contrôleurs Rapsberry interfacés avec une prédiction météo internet est totalement fonctionnel.
- Les performances mesurées du régulateur prédictif confirment les résultats obtenus par simulation.

6 Annexes et documentations techniques

6.1 Brevet

Réf./ARMINES : 13.01663_PERSEE

Titre officiel : Installation de production énergétique comprenant un dispositif de prédiction météorologique, notamment une installation de chauffe-eau solaire comprenant un tel dispositif

Date de dépôt : 10/07/2013
N° de la demande : 1301663

DESCRIPTION

DOMAINE TECHNIQUE

La présente invention concerne une installation et un procédé de production énergétique qui comprend pour ladite production énergétique un moyen de production d'énergie à énergies renouvelables ainsi qu'un moyen de production énergétique à énergies non renouvelable.

Dans la présente demande, on entend par moyen de production à énergies renouvelables un dispositif qui utilise des énergies renouvelables telles que le soleil, le vent ou l'eau pour produire de l'énergie thermique ou électrique. Ces moyens de production d'énergies renouvelables étant à titre d'exemple et de façon non limitative du type capteur solaire thermique ou photovoltaïque, éolienne, centrale hydroélectrique ou usine marée motrice.

A l'inverse, on entend par moyen de production à énergies non renouvelables un dispositif qui n'utilise pas des énergies renouvelables telles que le soleil, le vent ou l'eau pour produire de l'énergie thermique ou électrique. Ces moyens de production d'énergies non renouvelables étant à titre d'exemple et de façon non limitative des moyens d'apport d'énergie électrique reliés à un fournisseur d'énergie électrique ou tirant leur énergie à partir d'énergies fossiles du type groupe électrogène.

ART ANTERIEUR

Il est connu de l'homme du métier une installation de production énergétique qui comprend un moyen de production à énergies renouvelables (1) du type capteur solaire ainsi qu'un moyen électrique d'apport d'énergie (5) du type résistance électrique avec sonde de régulation (Th), cette installation étant caractérisée en ce qu'elle comporte un calculateur (CA) du type contrôleur de température différentiel qui met en marche un circulateur (4) pour transporter l'énergie accumulée dans les capteurs solaire (1) vers un dispositif de stockage

d'eau chaude sanitaire (2) du type ballon par l'intermédiaire d'un fluide caloporteur lorsque l'écart entre la température du fluide circulant dans les capteurs solaires (Ts) et la température du fluide disposé dans le ballon d'eau chaude sanitaire (Tb) a atteint un seuil prédéterminé.

Une telle installation de production énergétique est illustrée en figure 1, elle comprend un calculateur (CA) différentiel et fonctionne selon un procédé qui comporte des étapes consistant à :

a) capter l'énergie solaire pour chauffer un fluide caloporteur qui est disposé dans des capteurs solaires (1),

b) comparer la température de sortie du capteur (Ts) avec la température au point bas du ballon (Tb),

c) mettre en marche un circulateur (4) dès que $Ts - Tb \geq \Delta t$, Δt étant un seuil prédéterminé de température, Δt étant préférentiellement choisi tel que $0 \leq \Delta t \leq 10°C$,

d) transporter au moyen d'un fluide caloporteur l'énergie captée au niveau des capteurs solaires (1) vers un échangeur (7) disposé dans un ballon d'eau chaude sanitaire (2) de sorte à chauffer de l'eau disposée dans celui-ci,

e) arrêter le circulateur (4) dès que $Ts - Tb \leq 0°C$,

f) activer la résistance électrique (5) lorsque la plage tarifaire (HT) est en « heures creuses » et que la sonde de régulation (Th) est en dessous d'un seuil prédéterminé, le seuil étant compris entre 60 °C et 85°C.

On entend ici par « heures pleines » une plage tarifaire durant laquelle le coût facturé de l'électricité est désavantageux. Par opposition on désigne par « heures creuses » une plage tarifaire dont le coût est avantageux.

Ce type d'installation présente un inconvénient principal. En cas de succession de journées présentant un apport

énergétique renouvelable faible, telles que des journées nuageuses, suivies d'une ou plusieurs journées présentant un bon apport énergétique telles que des journées ensoleillées, le moyen électrique d'apport d'énergie (5) est fortement sollicité pendant la période de faible ensoleillement, ce qui implique lors de la transition entre les 2 périodes, un niveau énergétique dans le dispositif de stockage trop élevé, ce qui est défavorable pour minimiser l'énergie disponible dans le dispositif de stockage (2). Ainsi, si les transitions entre les journées à faible et fort ensoleillement avaient pu être prévues, l'utilisation du moyen d'apport d'énergie électrique (5) aurait pu être réduite préalablement à l'arrivée d'une journée fortement ensoleillée en fonction du besoin utilisateur estimé, de sorte à minimiser le coût électrique de l'installation.

Afin de proposer une solution à cet inconvénient, les documents JPH04155156, JPS6269063, JP2011247513 et JP2006153383 proposent d'intégrer aux installations de l'art antérieur un dispositif de prévision météorologique. Toutefois, la prédiction météorologique utilisée dans ces documents ne permet de prédire de façon simple et fiable la production à venir du moyen de production d'énergie à énergies renouvelables.

BUT DE L'INVENTION

L'invention propose de résoudre ces inconvénients en intégrant dans une installation du type précité un dispositif de prédictions des conditions météorologiques à venir à partir d'au moins 4 niveaux de nébulosité et un procédé de gestion de l'alimentation du moyen électrique d'apport d'énergies intégrant lesdites prédictions.

A ce but l'invention propose une installation de production énergétique comprenant un moyen de production d'énergie utilisant des énergies renouvelables, un moyen de production énergétique à énergie non renouvelable, un dispositif de stockage d'énergie, un circuit de distribution pour satisfaire un besoin utilisateur et un calculateur qui reçoit des données représentatives de la consommation énergétique et de la production énergétique à venir du moyen de production d'énergie utilisant des énergies renouvelables pour commander la production du moyen de production énergétique à énergie non renouvelable de sorte à satisfaire un besoin utilisateur, les données représentatives de la production énergétique à venir intégrant des données météorologiques, caractérisée en ce que les données météorologiques sont reçues sous la forme d'au moins 4 différents types de niveau de prédiction de sorte à estimer la production du moyen de production d'énergie utilisant des énergies renouvelables, le moyen de production énergétique à énergie non renouvelable étant mis en marche ou arrêté de sorte à minimiser son utilisation tout en maintenant dans le dispositif de stockage d'énergie le niveau d'énergie minimal qui permette de satisfaire le besoin des utilisateurs.

Ainsi, selon l'invention la quantité suffisante d'énergie pour l'utilisateur est stockée dans le dispositif de stockage d'énergie ce qui permet également d'éviter la perte par dissipation du surplus de production énergétique dans l'installation, notamment au niveau du dispositif de stockage d'énergie.

Selon le mode de réalisation préféré de l'invention mais de façon non limitative, l'invention concerne une installation de production énergétique, destinée à la production d'eau chaude sanitaire comprenant: un capteur solaire thermique, un moyen électrique d'apport d'énergies, un dispositif de stockage d'énergie thermique, un échangeur, un circulateur, un système de canalisation pour la circulation d'un fluide caloporteur

qui relie le capteur solaire thermique à un échangeur thermique, un circuit de distribution d'eau chaude sanitaire et un calculateur qui reçoit des données d'une horloge tarifaire et/ou des données de besoins d'utilisateurs et/ou des données de température et/ou de débit et/ou des données météorologiques pour commander la production du moyen électrique d'apport d'énergies caractérisée en ce que les données météorologiques sont reçues sous la forme d'au moins 4 différents types de niveau de nébulosité de sorte à déterminer un profil d'irradiance globale journalier permettant d'estimer la production du capteur solaire thermique, le moyen électrique d'apport d'énergies étant mis en marche ou arrêté de sorte à minimiser son utilisation tout en maintenant dans le dispositif de stockage d'énergie thermique le niveau d'énergie minimal qui permette de satisfaire le besoin des utilisateurs (BU).

Ainsi, l'utilisation d'un profil d'irradiance globale journalier définit en fonction d'un des 4 niveaux de nébulosité permet pour une localisation donnée de l'installation d'estimer de façon simple et fiable la production énergétique à venir.

Avantageusement, l'échangeur thermique est disposé dans un ballon d'eau chaude sanitaire.

L'intégration de l'échangeur dans le ballon d'eau chaude sanitaire permet de limiter les pertes par échanges thermiques avec le milieu extérieur de l'installation à énergies renouvelables. Toutefois, dans une variante de réalisation, un tel échangeur pourrait être disposé en dehors du ballon d'eau chaude sanitaire.

Selon l'invention, le calculateur intègre un microprocesseur et un système d'exploitation qui présente un logiciel de prédiction relié à un serveur météo à partir duquel il

recueille les données météorologiques à venir selon au moins 4 niveaux de nébulosité.

L'utilisation d'un serveur météo permet ainsi de recueillir une ou plusieurs fois par jour des prévisions météorologiques géo localisées pour le site de l'installation de production d'eau chaude sanitaire.

L'invention concerne également un procédé de minimisation de l'apport en énergie non renouvelable pour une installation de production énergétique destinée à satisfaire un besoin utilisateur utilisant des énergies renouvelables et non renouvelable consistant à :

a) recevoir au niveau d'un calculateur des données représentatives de la consommation et de la production énergétique de l'installation,

b) recevoir au niveau d'un calculateur des données météorologiques selon au moins 4 niveaux de nébulosité,

c) utiliser les données reçues par le calculateur pour prévoir la production à venir du moyen de production énergétique à énergie renouvelable,

d) mettre en marche ou arrêter le moyen de production énergétique à énergie non renouvelable de sorte à minimiser son utilisation tout en maintenant dans le dispositif de stockage d'énergie le niveau d'énergie minimal qui permette de satisfaire le besoin des utilisateurs.

Avantageusement, les données représentatives de la consommation et de la production énergétique de l'installation sont des données d'une horloge tarifaire et/ou des données de besoins d'utilisateurs et/ou des données de température et/ou de débit.

Avantageusement, le moyen de production énergétique à énergie non renouvelable est mis en marche si l'heure correspond à une période tarifaire à coût avantageux et/ou si la quantité

d'énergie disponible dans le dispositif de stockage d'énergie est insuffisante pour couvrir le besoin des utilisateurs et/ou si les prévisions météorologiques à venir sont défavorables pour la production énergétique du moyen de production énergétique à énergie renouvelable.

A l'inverse, le moyen de production énergétique à énergie non renouvelable est arrêté si l'heure correspond à une période tarifaire à coût désavantageux et/ou si la quantité d'énergie disponible dans le dispositif de stockage d'énergie est suffisante pour couvrir le besoin des utilisateurs et/ou si les prévisions météorologiques à venir sont favorables pour la production énergétique du moyen de production énergétique à énergie renouvelable.

Dans le mode préférentiel de l'invention un tel procédé est utilisé pour une installation de production énergétique telle que décrite précédemment.

Grâce à ce procédé, la demanderesse a pu constater un gain global de 60% sur la consommation de la résistance électrique d'une installation de chauffe-eau sanitaire. En effet, en observant les enregistrements d'irradiance globale sur un site donné pendant plusieurs années on peut constater que la connaissance des prévisions météorologiques selon 4 niveaux de nébulosité permet d'anticiper la consommation d'apport d'énergie électrique et par là même de minimiser le coût énergétique de l'installation.

D'autre buts et avantages apparaitront à la lumière de la description détaillée de modes de réalisation de l'invention et des dessins annexés dans lesquels :

BREVE DESCRIPTION DES FIGURES

La Figure 1 représente les différents éléments techniques qui constituent une installation à énergies renouvelables du type capteur solaire thermique conventionnelle.

La Figure 2 représente les différents éléments techniques qui constituent une installation à énergies renouvelables du type capteur solaire thermique selon l'invention.

La Figure 3 représente les éléments techniques essentiels d'une installation de production énergétique comprenant un moyen de production énergétique à énergie renouvelable et un moyen de production énergétique à énergie non renouvelable selon l'invention.

La Figure 4 représente un exemple de réalisation d'algorithme pour le logiciel du calculateur pour une installation selon l'invention

La Figure 5a représente la récupération des données de prévisions météorologiques concernant la nébulosité à 4 niveaux pour la journée du lendemain sur un serveur informatique spécialisé ou un système de diffusion hertzien par ondes radio ou satellites.

La Figure 5b représente l'obtention de l'irradiance globale pour chaque jour de l'année en fonction du niveau de nébulosité journalier et d'un modèle de référence d'irradiance globale annuelle.

La Figure 5c représente la prévision d'irradiance globale cumulée pour le jour suivant qui permet d'estimer l'énergie qui sera disponible dans le ballon de stockage le jour suivant.

La Figure 5d représente une séquence d'irradiance globale sur plusieurs jours avec une journée à forte nébulosité, dans laquelle on constate un faible écart entre les données réelles mesurées et les prévisions calculées.

MODE DE REALISATION DE L'INVENTION

Dans la suite de la demande, on utilise la même référence pour désigner deux éléments qui présentent une fonction similaire.

La figure 2 présente une installation à énergies renouvelables qui comporte des moyens de production d'énergie

thermique (1) du type capteur solaire, ainsi qu'un moyen électrique d'apport d'énergie thermique (5) du type résistance électrique.

Dans le mode de réalisation préféré de l'invention, le capteur solaire (1) est du type capteur solaire thermique plan simple vitrage ou sous vide.

Préférentiellement et de façon non limitative, la résistance électrique est du type thermoplongeur en épingle sur bride ou stéatite positionnée à mi-hauteur du ballon de stockage.

Cette installation de production d'énergie thermique est notamment destinée à être utilisée pour chauffer de l'eau chaude sanitaire stockée dans un dispositif de stockage (2) d'eau chaude sanitaire tel qu'un ballon cylindrique vertical. La taille du dispositif de stockage et de captation d'énergies renouvelables devant être adaptée à la consommation en eau chaude sanitaire de l'installation.

Afin de distribuer l'énergie thermique produite au niveau du capteur solaire (1) vers le dispositif de stockage de l'eau chaude sanitaire (2), l'installation comporte un système de canalisation (3), allant des capteurs solaires (1) vers un système de stockage de l'eau chaude sanitaire (2), ainsi qu'un circulateur (4) qui permet de transporter un fluide caloporteur des capteurs (1) vers le dispositif de stockage d'eau chaude sanitaire (2), le fluide caloporteur étant un mélange d'eau et d'antigel qui présente une caractéristique de fonctionnement en température comprise entre -25°C et 200°C. Généralement, le fluide caloporteur contient également des inhibiteurs spécifiques qui permettent de protéger les dispositifs métalliques utilisés dans le système de circulation du fluide caloporteur de l'installation solaire. Ces inhibiteurs étant également des moyens de protection efficaces contre la corrosion, les boues et la formation de dépôts.

Après avoir été chauffé au niveau des capteurs solaires (1) et transporté par le circulateur (4), le fluide caloporteur arrive à l'intérieur du dispositif de stockage d'eau chaude sanitaire(2), de sorte qu'un échangeur thermique (7) disposé à l'intérieur du dispositif de stockage (2) puisse échanger l'énergie thermique dudit fluide caloporteur avec l'eau chaude sanitaire disposée dans le dispositif de stockage (2). Ainsi, le fluide caloporteur chauffe l'eau chaude sanitaire disposée dans le dispositif de stockage (2) en se refroidissant.

Dans ce mode de réalisation préféré, l'échangeur thermique (7) est disposé à l'intérieur du dispositif de stockage (2) de l'eau chaude sanitaire. Toutefois, l'homme du métier pourrait concevoir un tel échange thermique avec un échangeur disposé à l'extérieur du dispositif de stockage d'eau chaude sanitaire sans pour autant sortir du cadre de l'invention.

Selon l'invention, le deuxième moyen de chauffage d'eau chaude sanitaire disposé dans le moyen de stockage d'eau chaude sanitaire (2) est un moyen électrique d'apport d'énergies constitué par une résistance électrique (5) disposée au-dessus d'une sonde de température (Th) immergée à mi-hauteur du dispositif de stockage (2). Ainsi la résistance électrique (5) est à distance de la sonde de température (Th) de sorte à mesurer la température d'eau chaude sanitaire au centre du ballon de stockage.

Afin d'atteindre le but de l'invention qui est de limiter les dépenses financières liées à l'utilisation de la résistance électrique (5) tout en fournissant à l'utilisateur l'eau chaude dont il a besoin, des mesures de température régulières sont effectuées au niveau de la sonde de température (Th) disposée à mi-hauteur du dispositif de stockage de l'eau chaude sanitaire (2), la température de l'eau chaude sanitaire disposée dans le ballon de stockage

(Th) étant contrôlée de façon régulière de sorte à ne pas être inférieure à une température de consigne fixée pour l'installation.

Ici, on entend par régulières des périodes de 60 secondes à plusieurs minutes de façon à pouvoir comptabiliser l'énergie thermique stockée.

Préférentiellement, la résistance électrique (5) est activée lors des périodes tarifaires « heures creuses » et si la température du ballon d'eau chaude sanitaire (Th) est en dessous d'un seuil prédéterminé ou si le calcul de prédiction de la production énergétique à partir des prévisions météorologiques (DM) selon 4 niveaux de nébulosité sont défavorables ou si la consommation à venir estimée par le calculateur (CA) est supérieure au stock d'énergie disponible, le calculateur (CA) ayant pour fonction de moduler le fonctionnement de la résistance d'appoint (5) de sorte à satisfaire l'ensemble des contraintes décrites ci-dessus tout en évitant de produire un surplus non consommé d'énergie.

Selon l'invention mais de façon non limitative, le contrôle du fonctionnement de la résistance électrique (5) est effectué par un calculateur (CA) qui décide d'arrêter l'appoint d'énergie électrique (5) lorsque la durée calculée de fonctionnement nécessaire est atteinte et/ou lorsque les conditions d'activation ne sont pas remplies.

Ainsi, lors du fonctionnement de l'installation, la connaissance des prévisions météorologiques (DM) selon 4 niveaux minimum de nébulosité pour les prochaines heures permettent de calculer l'irradiance globale future, d'en déduire l'énergie renouvelable disponible dans le ballon de

stockage et de moduler préalablement la quantité d'énergie d'appoint à injecter, de diminuer le niveau de température dans le ballon de stockage et ainsi de limiter les pertes en maîtrisant l'apport énergétique de la résistance électrique d'appoint (5).

Dans le mode de réalisation préféré de l'invention, le dispositif de prédiction intègre une carte à microprocesseur ou un calculateur (CA) ainsi qu'un système d'exploitation et un logiciel qui constitue la logique du contrôleur. Ainsi, le dispositif utilise des données d'entrées physiques telles que des mesures de températures et de débit, des signaux de tarification du fournisseur (HT) du type énergie électrique ainsi que des données numériques telles que des prévisions météorologiques (DM), les prévisions étant téléchargées régulièrement depuis les services disponibles sur des serveurs de données informatiques via un support hertzien ou filaire et les serveurs fournissant des prévisions météorologiques, de température, d'humidité, de la nébulosité, de l'irradiance globale, en fonction de la géo localisation du lieu de l'installation à énergies renouvelables.

Avantageusement et de façon non limitative, les signaux de tarification du fournisseur du type énergie électrique constituent une horloge tarifaire (HT) qui présente en mémoire les périodes « heures pleines » et « heures creuses » du fournisseur d'énergie, de sorte que les informations de ladite horloge tarifaire (HT), les prévision météorologiques (DM) et les données du besoin énergétique à venir (BU) des utilisateurs sont combinées par le calculateur (CA) pour calculer l'utilisation de la résistance électrique (5) qui minimise le coût financier de l'installation tout en satisfaisant le besoin utilisateur.

De par cet aspect avantageux, le logiciel calcule en temps réel la commande à appliquer aux sorties du calculateur (CA) et active ou désactive le circulateur et/ou la résistance de chauffe (5) pour l'appoint dans le ballon de stockage (2).

De façon non limitative, le dispositif de prédiction (DM) comporte une carte à microprocesseur ayant des périphériques et une logique de fonctionnement constituant un logiciel de gestion implémenté dans le système d'exploitation de la carte, la carte à microprocesseur étant composée d'un calculateur (CA) présentant des caractéristiques techniques connues par l'homme du métier ainsi que les périphériques suivants :

- au moins 4 voies de mesures avec convertisseur analogique ou digital pour transformer en données numériques compatibles avec le calculateur les mesures de températures et de débit avec une précision de +/- 0,1°C sur la plage de températures allant de 0 à 150 °C. et +/-1% sur la plage de débit,
- une entrée « Tout ou Rien » isolée ou numérique pour récupérer à partir du compteur d'énergie du fournisseur d'électricité les informations tarifaires du fournisseur d'énergie électrique,
- une sortie par relais statique pour la commande de la résistance d'appoint soit en « Tout ou Rien », soit en « modulation de train d'onde ».
- une sortie configurable soit en analogique ou en « Tout ou Rien » pour la commande du circulateur soit en variation de fréquence soit en marche/arrêt.
- la récupération des données de prévisions météorologiques (DM) étant effectuée par une connexion réseau filaire ou sans fil informatique type Ethernet ou un récepteur radio permettant de recueillir les signaux hertziens ou satellites, le récepteur étant équipé d'un convertisseur de protocole adapté au service météo correspondant de sorte à extraire à partir du signal hertzien les données de prévisions météo (DM) souhaitées.

Un exemple d'algorithme de fonctionnement qui intègre des données météorologiques dans une installation selon l'invention est illustré en figure 4, il comprend les étapes consistant à :

1) A la mise en service les paramètres du système sont configurés dans le calculateur notamment le dimensionnement et l'orientation du système de captation (1), le volume et la température de consigne de fonctionnement du ballon de stockage (2).

2) Les données physiques du système telles que les données de températures Tb, Ts, Th et de débit Q sont acquises et mémorisées. Le calcul en temps réel à partir des données physiques de l'énergie produite dans le ballon (2), permet par cumul itératif de déduire l'énergie stockée disponible à chaque instant.

3) Les données de prévisions métrologiques à 4 niveaux sont téléchargées depuis un serveur. Elles permettent de calculer la prédiction d'irradiance globale pour la journée à venir pour déduire l'énergie disponible dans le ballon (2) dans les prochaines heures en fonction du besoin des utilisateurs (BU) à venir si la différence entre l'énergie mesurée et l'énergie prédite est inférieure à une limite prédéterminée, alors on passe à l'étape suivante, sinon l'appoint est arrêté.

4) Le calculateur évalue l'énergie d'appoint à fournir

5) En fonction de la tarification du fournisseur d'énergie électrique « heures pleines » ou « heures creuses » de l'horloge tarifaire (HT) l'appoint est mis en marche selon la durée et la puissance estimée à l'étape précédente par le calculateur (CA).

6) L'algorithme est mis en veille pendant une période de 60s à plusieurs minutes puis le calcul est relancé automatiquement.

Les figures 5a à 5d, représentent l'utilisation de 4 niveaux de nébulosité pour une prédiction météorologique fiable.

A la mise en service du procédé de prédiction il sera nécessaire de configurer la géo localisation du site de l'installation, l'orientation, le dimensionnement du système de captation (1), et le dimensionnement du ballon de stockage (2). Préférentiellement, un serveur de données météorologiques figure 5a délivre 4 niveaux de nébulosité : Beau temps, Ciel Couvert, Temps Pluvieux, Temps Orageux, pour les heures à venir à des intervalles réguliers. Par réguliers on entend une période de 1 heures à 6 heures. Ces données étant téléchargées et mémorisées dans le calculateur (CA) à microprocesseur, un modèle d'irradiance globale annuelle corrigée de la météo par beau temps implémenté dans le calculateur (CA) permettant de déterminer par pondération des 4 niveaux de nébulosité, 4 niveaux d'irradiance globale annuelle du site géo localisé et orienté. Cette construction pour chaque jour de l'année des 4 niveaux possibles d'irradiance globale est représentée en figure 5b.

Par la suite illustrée en figure 5c en connaissant la date du jour et le niveaux d'irradiance globale annuelle du site géo localisé et orienté, le calcul de la prévision de l'irradiance globale cumulée est effectué de sorte à déduire l'énergie prévue pour le jour suivant dans le ballon de stockage (2), la résistance électrique d'appoint (5) étant alors mise en marche, notamment en heure creuse si l'énergie prévue le jour suivant dans le ballon de stockage (2) n'est pas suffisante pour subvenir au besoin des utilisateurs (BU) et à l'inverse la résistance électrique d'appoint (5) étant arrêtée si l'énergie prévue le jour suivant dans le ballon de stockage (2) est suffisante pour subvenir au besoin des utilisateurs (BU). Dans le mode de réalisation préférentiel de l'invention, après avoir été mise en marche en raison d'un

manque d'énergie prévue dans ballon de stockage (2) pour satisfaire le besoin des utilisateurs (BU) du jour suivant la résistance électrique d'appoint (5) est arrêté dès que l'énergie suffisante pour couvrir le besoin des utilisateurs (BU) du jour suivant est obtenue.

Le graphique qui exprime l'évolution de l'irradiance globale mesurée et prévue heure par heure sur plusieurs journées est illustré en figure 5d. Les résultats obtenus mettent en évidence de faibles écarts entre les valeurs mesurées et les prévisions à 4 niveaux de nébulosité et valident la qualité et la fiabilité du procédé de l'invention.

L'invention ne se limite pas aux modes de réalisation décrits précédemment, l'homme du métier pourrait concevoir des variantes de réalisation sans pour autant sortir du cadre de la protection de l'invention conférée par les revendications.

NOMENCLATURE

1 : moyen de production d'énergies thermiques

2 : dispositif de stockage d'énergies thermiques

3 : système de canalisation de la circulation d'un fluide caloporteur

4 : circulateur

5 : moyen électrique d'apport d'énergies thermiques

6 : arrivée d'eau froide sanitaire

7 : échangeur thermique

8 : circuit de distribution hydraulique d'eau chaude sanitaire

Th : sonde de température

HT : horloge tarifaire

DT : données de températures du fluide caloporteur

Q : données de débit du fluide caloporteur

DM : données météorologiques

BU : Besoin utilisateur

CA : calculateur

Ts : température du fluide caloporteur au niveau du moyen de production d'énergies thermiques

Tb : température de l'eau au point bas du dispositif de stockage d'énergies thermique

Revendication

1) Installation de production énergétique comprenant un moyen de production d'énergie utilisant des énergies renouvelables (1), un moyen de production énergétique à énergie non renouvelable (5), un dispositif de stockage d'énergie (2), un circuit de distribution pour satisfaire un besoin utilisateur (BU) et un calculateur (CA) qui reçoit des données représentatives de la consommation énergétique et de la production énergétique à venir du moyen de production d'énergie utilisant des énergies renouvelables (1) pour commander la production du moyen de production énergétique à énergie non renouvelable (5) de sorte à satisfaire un besoin utilisateur (BU), les données représentatives de la production énergétique à venir intégrant des données météorologiques (DM), caractérisée en ce que les données météorologiques (DM) sont reçues sous la forme d'au moins 4 différents types de niveau de prédiction de sorte à estimer la production du moyen de production d'énergie utilisant des énergies renouvelables (1), le moyen de production énergétique à énergie non renouvelable (5) étant mis en marche ou arrêté de sorte à minimiser son utilisation tout en maintenant dans le dispositif de stockage d'énergie (2) le niveau d'énergie minimal qui permette de satisfaire le besoin des utilisateurs (BU).

2) Installation de production énergétique selon la revendication 1, destinée à la production d'eau chaude sanitaire comprenant: un capteur solaire thermique (1), un moyen électrique d'apport d'énergies (5), un dispositif de stockage d'énergie thermique (2), un échangeur (7), un circulateur (4), un système de canalisation pour la

circulation d'un fluide caloporteur (3) qui relie le capteur solaire thermique (1) à un échangeur thermique (7), un circuit de distribution d'eau chaude sanitaire (8) et un calculateur (CA) qui reçoit des données d'une horloge tarifaire (HT) et/ou des données de besoins d'utilisateurs (BU) et/ou des données de température (DT) et/ou de débit (Q) et/ou des données météorologiques (DM) pour commander la production du moyen électrique d'apport d'énergies (5) caractérisée en ce que les données météorologiques (DM) sont reçues sous la forme d'au moins 4 différents types de niveau de nébulosité de sorte à déterminer un profil d'irradiance globale journalier permettant d'estimer la production du capteur solaire thermique (1), le moyen électrique d'apport d'énergies (5) étant mis en marche ou arrêté de sorte à minimiser son utilisation tout en maintenant dans le dispositif de stockage d'énergie thermique (2) le niveau d'énergie minimal qui permette de satisfaire le besoin des utilisateurs (BU).

3) Installation selon la revendication 2, caractérisée en ce que l'échangeur thermique (7) est disposé dans un ballon d'eau chaude sanitaire (2).

4) Installation selon les revendications 2 ou 3, caractérisée en ce que le calculateur (CA) intègre un microprocesseur et un système d'exploitation qui présente un logiciel de prédiction relié à un serveur météo à partir duquel il recueille les données météorologiques (DM) à venir selon au moins 4 niveaux de nébulosité.

5) Procédé de minimisation de l'apport en énergie non renouvelable pour une installation de production énergétique destinée à satisfaire un besoin utilisateur (BU) utilisant

des énergies renouvelables (1) et non renouvelable (5) consistant à :

e) recevoir au niveau d'un calculateur (CA) des données représentatives de la consommation et de la production énergétique à venir de l'installation,

f) recevoir au niveau d'un calculateur (CA) des données météorologiques (DM) selon au moins 4 niveaux de nébulosité,

g) utiliser les données reçues par le calculateur (CA) pour prévoir la production à venir du moyen de production énergétique à énergie renouvelable (1),

h) mettre en marche ou arrêter le moyen de production énergétique à énergie non renouvelable (5) de sorte à minimiser son utilisation tout en maintenant dans le dispositif de stockage d'énergie (2) le niveau d'énergie minimal qui permette de satisfaire le besoin des utilisateurs (BU).

6) Procédé selon la revendication 5, caractérisé en ce que les données représentatives de la consommation et de la production énergétique de l'installation sont des données d'une horloge tarifaire (HT) et/ou des données de besoins d'utilisateurs (BU) et/ou des données de température (DT) et/ou de débit (Q).

7) Procédé selon la revendication 6, caractérisé en ce que le moyen de production énergétique à énergie non renouvelable (5) est mis en marche si l'heure correspond à une période tarifaire (HT) à coût avantageux et/ou si la quantité d'énergie disponible dans le dispositif de stockage d'énergie (2) est insuffisante pour couvrir le besoin des utilisateurs (BU) et/ou si les prévisions météorologiques (DM) à venir sont défavorables pour la production énergétique du moyen de production énergétique à énergie renouvelable (1).

8) Procédé selon la revendication 6, caractérisé en ce que le moyen de production énergétique à énergie non renouvelable (5) est arrêté si l'heure correspond à une période tarifaire (HT) à coût désavantageux et/ou si la quantité d'énergie disponible dans le dispositif de stockage d'énergie (2) est suffisante pour couvrir le besoin des utilisateurs (BU) et/ou si les prévisions météorologiques (DM) à venir sont favorables pour la production énergétique du moyen de production énergétique à énergie renouvelable (1).

9) Utilisation du procédé selon l'une quelconque des revendications 5 à 8 pour une installation de production énergétique selon l'une quelconque des revendications 1 à 4.

Installation de production énergétique comprenant un dispositif de prédiction météorologique, notamment une installation de chauffe-eau solaire comprenant un tel dispositif.

L'invention concerne une installation de production énergétique destinée à satisfaire un besoin d'utilisateurs

(BU) comprenant des moyen de production d'énergie utilisant des énergies renouvelables (1) et non renouvelables (5) ainsi qu'un calculateur (CA) qui reçoit des données de prédictions météorologiques (DM) pour réduire le coût énergétique de fonctionnement de l'installation, les données météorologiques (DM) étant reçues sous la forme d'au moins 4 différents types de niveau de nébulosité de sorte à déterminer un profil d'irradiance globale journalier permettant d'estimer la production du moyen de production d'énergie utilisant des énergies renouvelables (1), le moyen de production énergétique à énergie non renouvelable (5) étant mis en marche ou arrêté de sorte à minimiser son utilisation tout en maintenant dans le dispositif de stockage d'énergie (2) le niveau d'énergie minimal qui permette de satisfaire le besoin des utilisateurs (BU).

L'invention concerne notamment une installation de chauffe-eau solaire qui utilise de telles prédictions météorologiques.

Figure 2

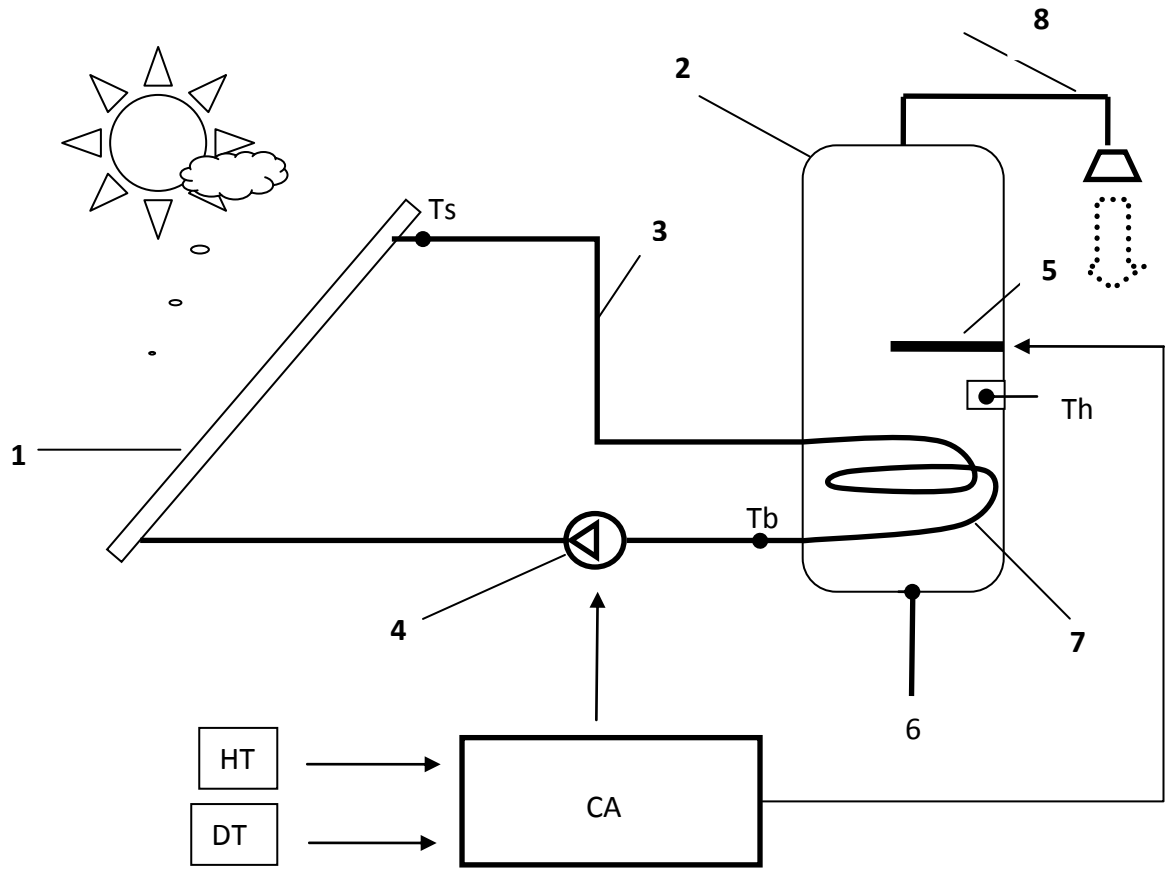


Figure 1

8

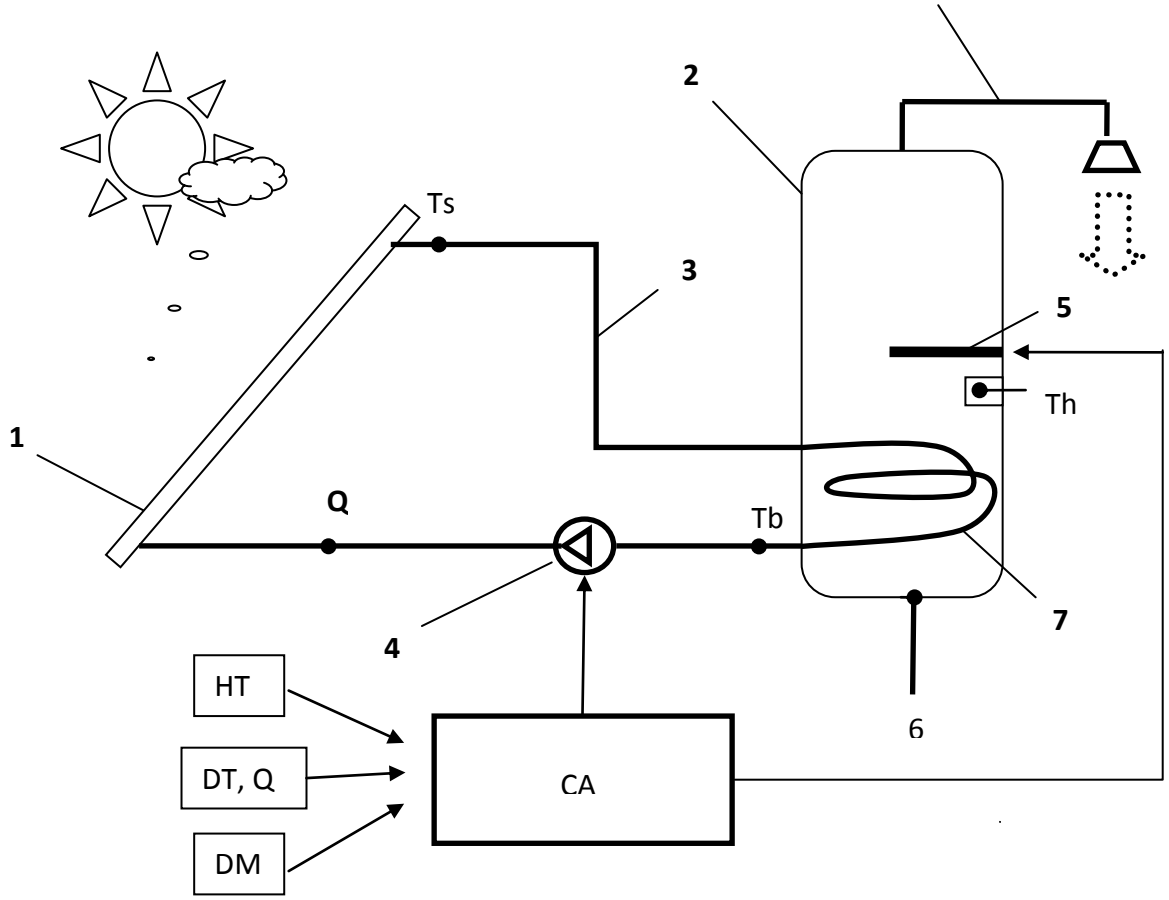


Figure 2

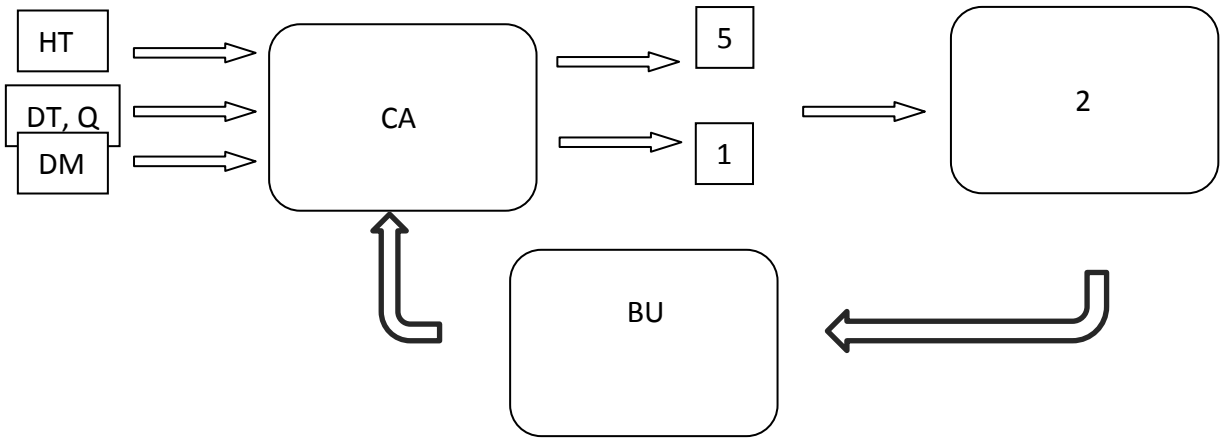


Figure 3

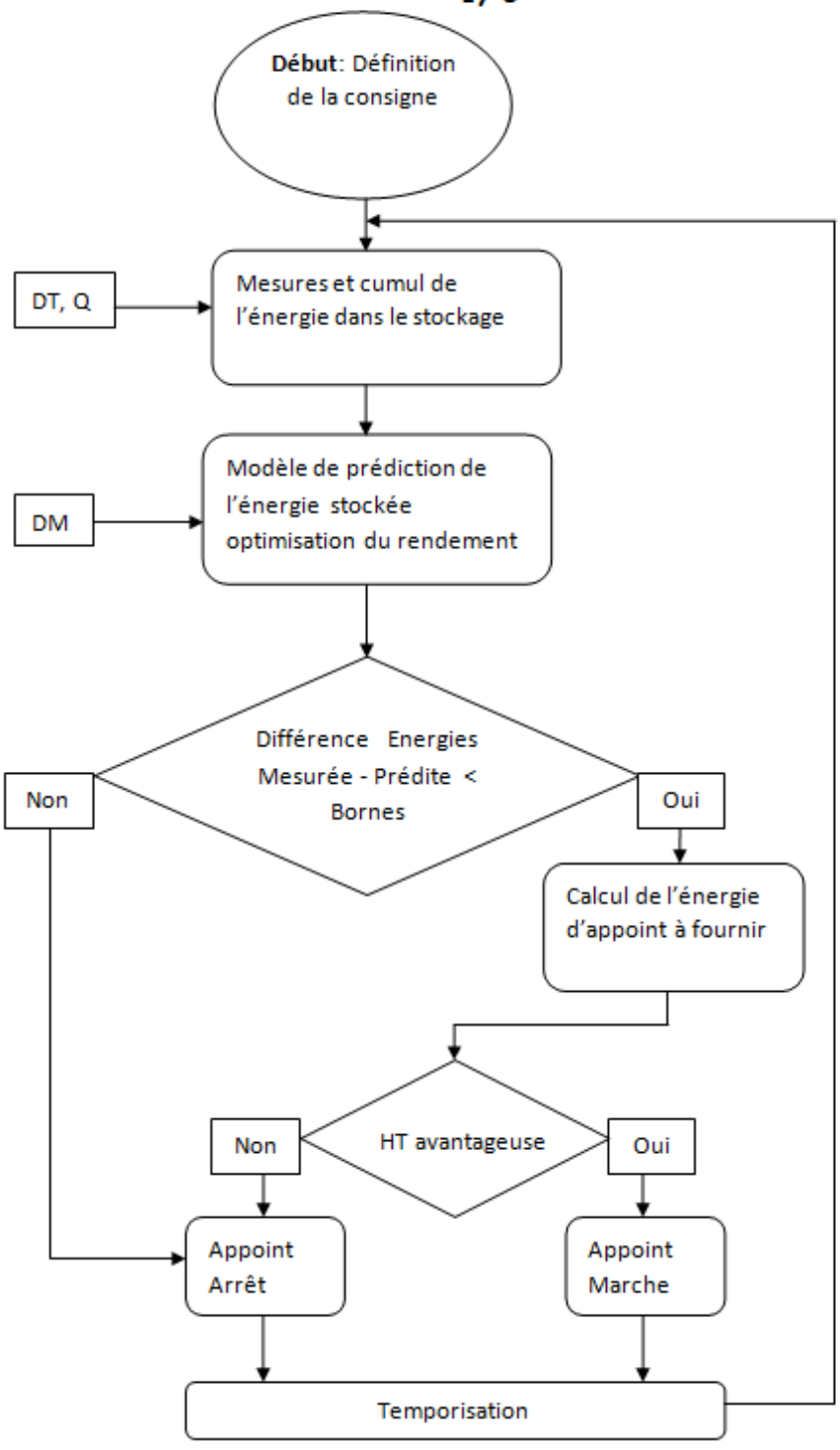

Début: Définition de la consigne
DT, Q
Mesures et cumul de l'énergie dans le stockage
DM
Modèle de prédiction de l'énergie stockée optimisation du rendement
Différence Energies Mesurée - Prédite < Bornes
Non
Oui
Calcul de l'énergie d'appoint à fournir
Non
HT avantageuse
Oui
Appoint Arrêt
Appoint Marche
Temporisation

Figure 4

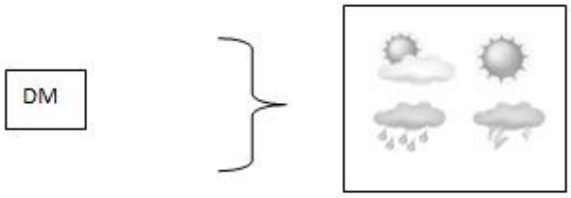


Figure 5a

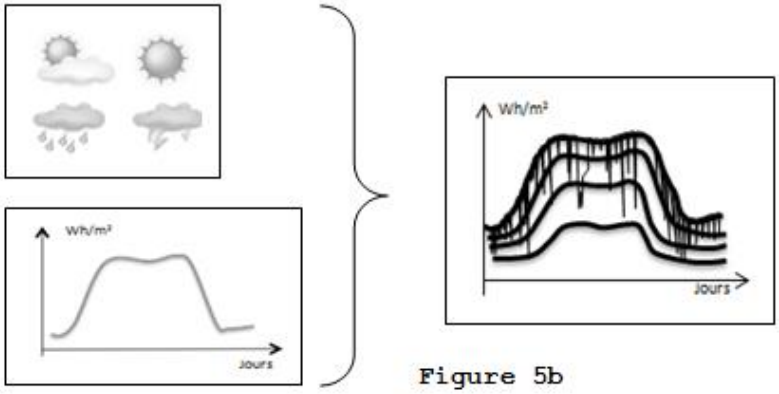


Figure 5b

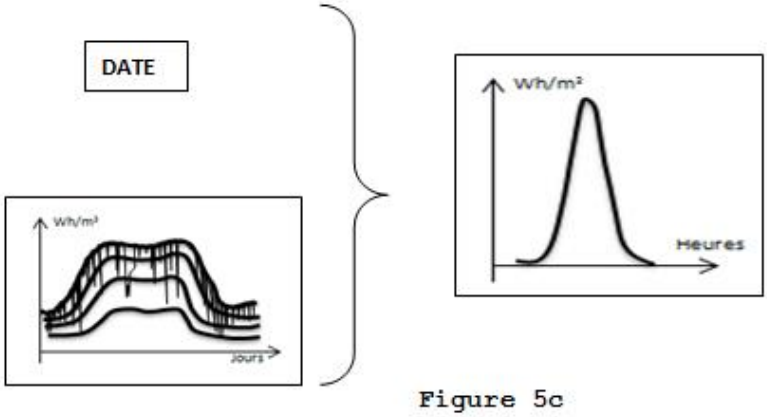


Figure 5c

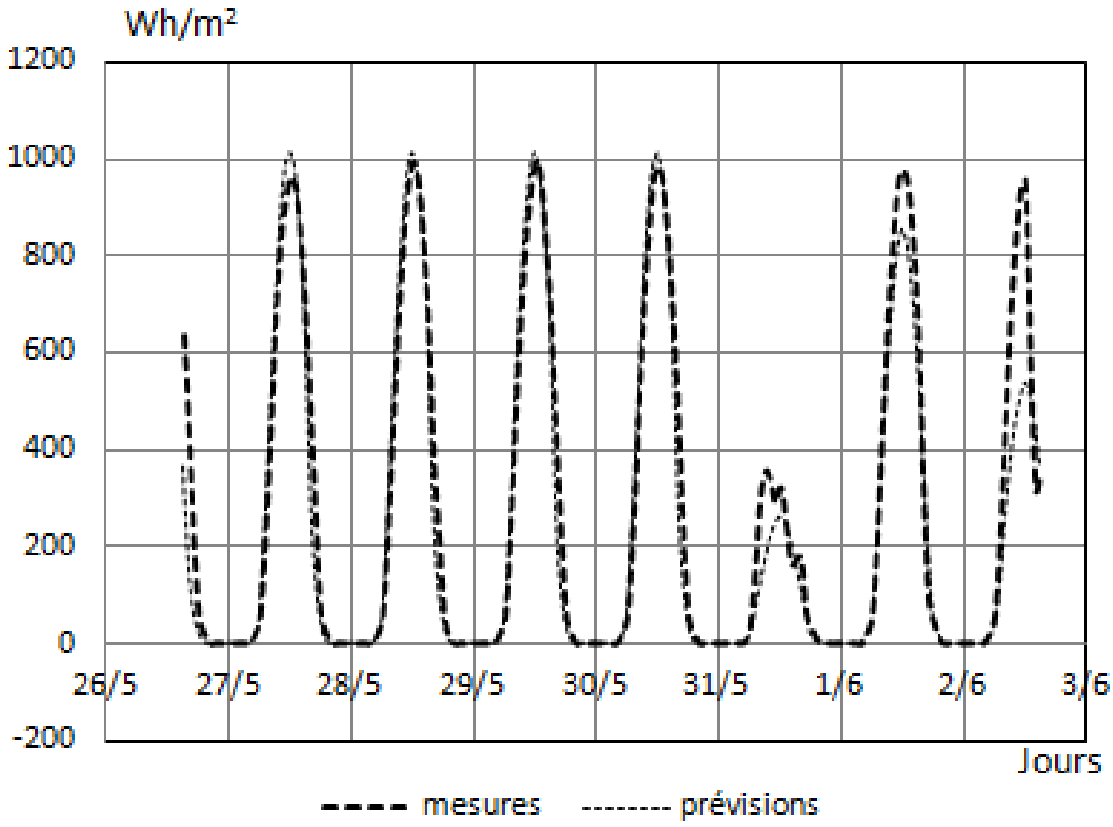

Wh/m²
1200
1000
800
600
400
200
0
-200
26/5
27/5
28/5
29/5
30/5
31/5
1/6
2/6
3/6
Jours
mesures
prévisions

Figure 5d

6.2 *Article : Appoint électrique et prévisions météo paru dans la revue CVC N°880*

PROFESSION ÉTUDES

Appoint électrique et prévision météo

Modèle de référence

Modèle prévision météo parfaite

Modèle prévision météo plus réaliste

Par Briac Piriou, Arnaud Pascal, François Pascal Neirac, Patrick Gatt, Mines Paristech

Les capteurs solaires à appoint électrique permettent d'assurer l'approvisionnement en eau chaude sanitaire, l'apport solaire couvrant entre 50 et 90 % des besoins. Notre étude vise à réduire la part de l'appoint électrique en intégrant des prévisions météo dans son asservissement.

Un chauffe-eau solaire avec appoint électrique fonctionne selon le fonctionnement suivant. Le capteur solaire thermique permet de récupérer sous forme de chaleur le rayonnement du soleil grâce à un absorbeur. Un fluide caloporteur se réchauffe au contact de l'absorbeur, circule dans le circuit primaire et passe dans un échangeur dans le bas du ballon de stockage pour réchauffer l'eau froide. Le fluide refroidi lors de l'échange thermique repart ensuite vers le capteur pour être à nouveau réchauffé. Un système d'appoint électrique permet de palier au manque de soleil grâce à une résistance électrique située en haut du ballon. L'appoint est régulé par un thermostat permettant le réchauffement du ballon pendant la nuit en cas de manque d'énergie solaire. Le principe du contrôle intelligent est de mettre en place une régulation plus fine sur l'appoint électrique en prenant en compte les prévisions météorologiques.

Modèle de référence

› Explications individuelles des formules

Le ballon est modélisé comme une réserve d'énergie thermique. Les calculs sont faits heure par heure sur l'énergie thermique contenue dans le ballon :

$$Eb(h) = Eb(h-1) + Eappoint(h) + Esolaire(h) - P(h) - S(h) \quad (1)$$

où h est l'heure considérée, Eb est l'énergie du ballon, P les pertes, S les soutirages, Eappoint l'énergie apportée par la résistance électrique d'appoint, Esolaire l'énergie apportée par le capteur solaire.

Pour calculer l'énergie apportée par le soleil nous utilisons la formule du rendement suivante définie par la norme EN 12975-2 (nous prenons un capteur solaire de type Giordano C8S) [3] :

$$\eta(h) = \eta 0 - a1\,Tm(h) - Ta(h)Eng - a2Engstd\,Tm(h) - Ta(h)Eng2 \quad (2)$$

où η_0 est le coefficient optique du capteur Giordano C8S, a_1 et a_2 sont respectivement les coefficient d'ordre 1 et 2 du même capteur, T_m la température moyenne du capteur ($Tm = \frac{Tentrée + Tsortie}{2}$), $T_{entrée}$ la température en entrée du capteur (sortie du ballon), T_{sortie} la température en sortie du capteur (entrée du ballon), T_a la température ambiante, Eng l'ensoleillement global sur le plan du capteur et Eng_{std} un ensoleillement standard conformément à la norme EN 12975-2.

Nous avons une relation simple entre la température du ballon et l'énergie qu'il contient car nous considérons que cette énergie correspond à l'énergie qu'il faut apporter pour élever la température de l'eau depuis la température ambiante jusqu'à la température de l'eau du ballon. On obtient donc la relation suivante :

$$Tballonh = \frac{Ebh}{1Cp * V * \rho} + Teau\,froide(h) \quad (3)$$

où Cp est la capacité calorifique de l'eau, V le volume du ballon et ρ la masse volumique de l'eau.

La température d'entrée dans le capteur est égale à la température de l'eau dans le ballon car dans notre modélisation en énergie, la température dans le ballon est homogène :

$$Tballonh = Tentréeh \quad (4)$$

Les pertes sont modélisées par un coefficient de perte K = 0,6 Wh/(L.K).

Nous supposons alors que l'appoint électrique est réglé pour arriver à une énergie dans le ballon équivalente à 300 l à 50 °C en fin de nuit. Cela correspond à la régulation la plus simple possible sous forme de thermostat.

Les soutirages sont modélisés de telle sorte qu'un quart de l'énergie du ballon est soutirée aléatoirement entre 6h et 9h et un autre quart entre 19h et 22h, toujours de manière aléatoire sur 3h [4]. Considérer les soutirages par rapport à l'énergie contenue dans le ballon permet d'inclure une saisonnalité puisque cette dernière est calculée par rapport à la température ambiante qui dépend de la saison. Il y aura donc moins d'énergie prélevée en été. De plus, une quantité aléatoire, bornée à 20 % de l'énergie prélevée, est ajoutée ou retirée à la quantité totale du soutirage. Cela permet d'ajouter une part d'incertitude. Ces soutirages sont les mêmes pour tous les modèles afin de comparer les performances.

Les données météo utilisées sont celles de la ville de Nice pour l'année 2007. L'ensoleillement est ensuite ramené dans le plan du capteur pour être utilisé dans la formule (2).

beau temps (Clear Sky model) [7] [8]. Nous définissons quatre zones correspondant aux quatre catégories de météo et pouvons estimer l'ensoleillement de la journée en connaissant uniquement la date et la prévision météo. Nous reconstruisons ensuite l'ensoleillement heure par heure en multipliant les différentes composantes horaires d'un pic d'ensoleillement type du mois courant par un coefficient permettant ainsi à ce pic déformé d'avoir la valeur prévue d'ensoleillement totale sur la journée. Cette construction est résumée par la › **Figure 3**.

Les résultats obtenus en intégrant cette prévision météo ne s'éloignent pas beaucoup de ceux de la prévision parfaite. La › **Tableau 1** résume les résultats ainsi que les économies obtenues.

En prenant un prix de 120 € par MWh pour un ménage ayant une consommation moyenne, le dispositif permet de réaliser 240 € d'économies chaque année, à comparer avec le prix d'un tel dispositif, estimé à 200 €. ■ xx-xx

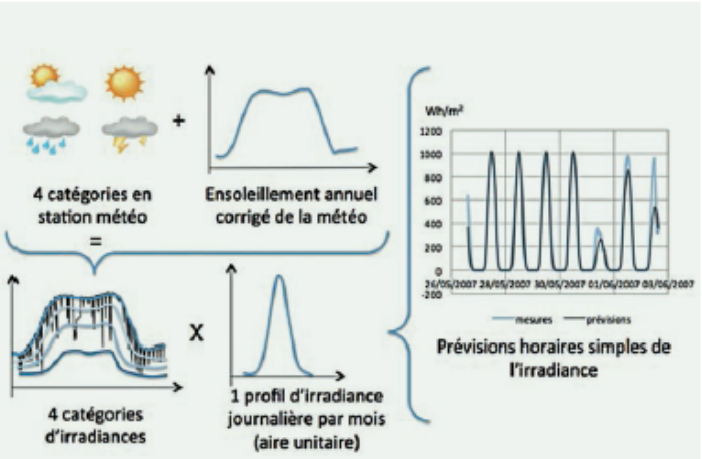


Figure 3 Principe de construction des prévisions horaires réelles.

Tableau 1 Détails des résultats pour les différents modèles

	η	$E_{appoint}$ (10^9 J)	Économie (kWh)	Économies annuelles* (€)
Référence	0,53	10,3	0	0
SCP	0,63	2,8	2080	250
SCR	0,62	3,2	1970	236

SCP : Smart-Control avec prévisions parfaites
SCR : Smart-Control avec prévisions réalistes
* 120 €/ MWh

Bibliographie

[1] Abdunnabi, M.J.R. *"Effect of hot water load patterns on the design parameters of thermosyphon solar water heaters."* IEEE, 2009.

[2] Ching-Tsan Chiang, Yung-Sheng Lee and Xiao Ru Li and Chiung-Chou Liao. *"A RSCMAC Based Forecasting for Solar Irradiance from Local Weather Information."* WCCI 2012 IEEE World Congress on Computational Intelligence , 2012.

[3] ISO. Norme européenne EN 12975-2. UE, ISO - EN.

[4] Kousuke Uchida, Tomonobu Senjyu, Naomitsu Urasaki and Atsushi Yona. *"Installation Effect by Solar Heater System using Solar Radiation Forecasting."* IEEE, 2009.

[5] Lucio Ciabattoni, Gianluca Ippoliti and Sauro Longhi, Matteo Cavalletti and Marco Rocchetti. *"Solar Irradiation Forecasting using RBF Networks for PV Systems with Storage."* IEEE, 2012.

[6] Lucio Ciabattoni, Massimo Grisostomi, Gianluca Ippoliti, Sauro Longhi, and Emanuele Mainardi. *"On Line Solar Irradiation Forecasting by Minimal Resource Allocating Networks."* 20th Mediterranean Conference on Control & Automation, 2012.

[7] Peder Bacher, Henrik Madsen, Bengt Perers, Henrik Aalborg Nielsen. *"A non-parametric method for correction of global radiation observations"*. Accepted 30 October 2012. Communicated by : Associate Editor David Renne

[8] Peder Bacher, Henrik Madsen, Henrik Aalborg Nielsen. *"Online short-term solar power forecasting"* Communicated by : Associate Editor Frank Vignola. Solar Energy 83 (2009) 1772–1783

› Processus itératif

Nous avons donc quatre formules principales qui forment un système d'équations :

- $Ebh = Ebh-1 + Eappointh + Esolaireh - Ph - Sh$ (1)
- $\eta(h) = \eta 0 - a1\ Tm(h) - Ta(h)Eng - a2Engstd\ Tm(h) - Ta(h)Eng2$ (2)
- $Tentréeh = Ebh * 1Cp * V * \rho + Ta(h)$ (3, 4)
- $Tsortieh = Tentréeh + Esolaireh * 1Cp * V * \rho$ (5)

Ces variables étant interdépendantes, il est difficile de passer simplement de l'état h à l'état h+1. Nous avons choisi d'appliquer un système itératif avec une initialisation jusqu'à convergence des valeurs obtenues par ces équations. Il faut cependant savoir dans quel ordre appliquer ces formules.

Nous avons raisonné en fonction des temps caractéristiques de chacune des modifications. Ainsi nous commençons chaque heure par soustraire les soutirages car ce sont les modifications les plus rapides. Nous utilisons pour cela la formule (1) de façon partielle. Ensuite nous pouvons calculer la nouvelle température du ballon à partir de la nouvelle valeur de son énergie (3, 4). Cela nous permet de calculer le nouveau rendement avec cette nouvelle valeur de $Tentrée$ et l'ancienne valeur de $Tsortie$ (2). Cette nouvelle valeur du rendement permet alors de calculer la nouvelle valeur de $Tsortie$ (5). On peut enfin déduire les pertes et ajouter l'apport solaire (1).

Ces premières étapes constituent l'initialisation. Le processus itératif consiste à appliquer successivement (3, 4) puis (2) puis (5) jusqu'à convergence.

Modèle prévision météo parfaite

Nous pouvons maintenant nous intéresser au gain envisageable grâce à la connaissance des prévisions météo [4]. Dans un premier temps, nous supposons une connaissance parfaite de la météo.

Comme on peut le voir sur la › **Figure 1**, il est assez fréquent de ne pas avoir utilisé entièrement l'énergie du ballon en fin de journée. On appelle Δ l'écart entre le plus bas niveau d'énergie de la journée et le niveau de consigne en dessous duquel il ne faut pas descendre. En anticipant l'apport solaire du jour suivant, on peut calculer à l'avance le Δ d'énergie qu'il restera si l'on utilise l'appoint électrique en scénario de référence. Ce Δ d'énergie peut alors être retranché de l'énergie à fournir par l'appoint électrique. Ici se trouve une première source d'économie cependant l'analyse ne s'arrête pas là. En effet, comme l'on aura moins chauffé l'eau durant la nuit, le capteur solaire en verra son rendement (formule (2)) augmenté, le rendement étant une fonction décroissante de la température de l'eau dans le capteur (Tm). On obtient alors un nouveau Δ qui peut à son tour être retranché de l'appoint électrique. Il faut répéter cette opération jusqu'à avoir un delta nul. Le gain réalisé est alors maximal. Nous présentons dans la › **Figure 2** les résultats obtenus grâce à l'intégration des prévisions météorologiques. Nous pouvons voir ici que les gains peuvent être très importants. En effet, l'appoint électrique n'a presque pas été utilisé dans cet exemple.

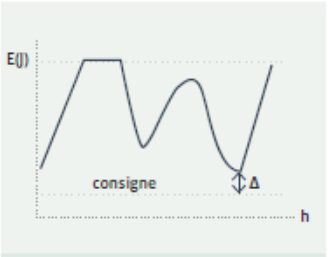


Figure 1 Marge (Δ) par rapport à la consigne, déductible de l'appoint électrique.

Modèle prévision météo plus réaliste

Les résultats précédents bien que très bons ne sont pas réalistes dans le sens où il est difficile d'avoir accès à une prévision parfaite de l'ensoleillement. Nous supposons donc ici que nous n'avons connaissance que d'une prévision journalière indiquant dans quelle situation nous nous trouverons parmi les quatre suivantes : Beau temps, Ciel Couvert, Temps Pluvieux, Temps Orageux [2] [5] [6].

Nous reconstruisons alors la courbe d'ensoleillement heure par heure à partir de cette donnée et de la date. Pour ce faire, nous établissons la courbe d'ensoleillement journalier corrigé de la météo, c'est-à-dire uniquement par

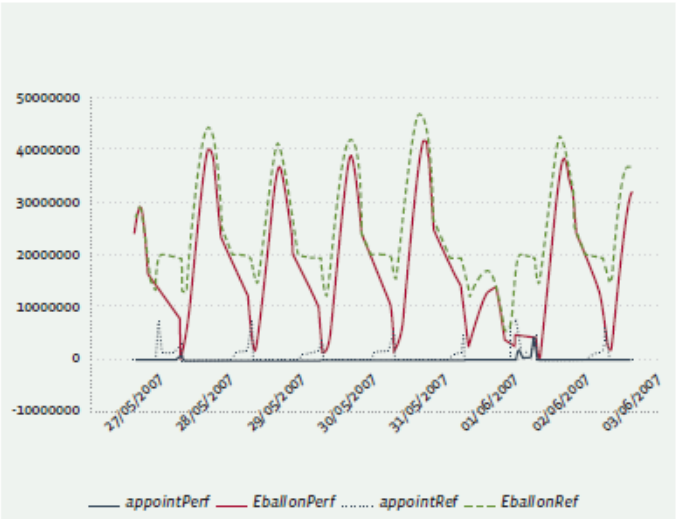


Figure 2 Gains obtenus grâce aux prévisions météo parfaites.

6.3 Logiciels embarqués sur Raspberry

Un séquenceur logiciel écrit en script bash est construit de manière à permettre l'execution de différentes tâches écrites en C, C++, Python ou des outils système tel que le filtre sed ou awk ou la commande date.

6.3.1 Séquenceur process_data

```
#!/bin/bash
DIR=/usr/local/bin
TMP=/home/public
MW_DIR=/usr/local/bin/MW
j=0
while test $j -ne 1
do
   k=0;

# Acquistion du flot de donnee du regulateur Vitosolic 200 à travers l'interface Resol Vbus et stockage danss datafile
   $DIR/grab5.py

# Vitosolic 200 Trame de base  avec le temps       Time     T1     T2     T3        T4         T5         T11        T12 I1     I2     R1     solaire
#    data0=`cat ./datafile | $DIR/vbusdecodevito -c 1 66,15,1,t 0,15,0.1 2,15,0.1  4,15,0.1 6,15,0.1 8,15,0.1 20,15,0.1 22,15,0.1 28,32,1 32,32,1 44,16,1 24,15,1`
# Apres test il est preferable de prendre le temps unix du raspberry

# Vitosolic 200 Trame de base                    T1     T2     T3    T4        T5         T11      T12      I1     I2     R1 solaire
   data0=`cat $TMP/datafile | $DIR/vbusdecodevito -c 1 0,15,0.1 2,15,0.1  4,15,0.1 6,15,0.1 8,15,0.1 20,15,0.1 22,15,0.1 28,32,1 32,32,1 44,16,1 24,15,1`

# Vitosolic 200 Trame calorimetre 1              I1  MWh     KWh    Wh
#   data1=`cat $TMP/datafile | $DIR/vbusdecodecalo1 -c 1 10,15,1  8,15,1 6,15,1`
#    data1=`echo $data1| sed  's/\;/ /;s/\;/ /'|awk '{print $1 $2 $3}'` # Filtrage et construction de la chaine totalisateur
    data1_w=`cat $TMP/datafile | $DIR/vbusdecodecalo1 -c 1 6,15,1 | sed 's/\;/ /'`
    data1_Kw=`cat $TMP/datafile | $DIR/vbusdecodecalo1 -c 1 8,15,1 | sed 's/\;/ /'`
    data1_Mw=`cat $TMP/datafile | $DIR/vbusdecodecalo1 -c 1 10,15,1 | sed 's/\;/ /'`
    let "data1=1000000*data1_Mw+1000*data1_Kw+data1_w"
#    echo "$data1_Mw $data1_Kw  $data1_w \n"
    data1=$data1";"

# Vitololic 200 Trame calorimetre 2              I2  MWh     KWh    Wh
#   data2=`cat $TMP/datafile | $DIR/vbusdecodecalo2 -c 1 10,15,1 8,15,1 6,15,1 `
#    data2=`echo $data2| sed  's/\;/ /;s/\;/ /'|awk '{print $1 $2 $3}'` # Filtrage et construction de la chaine totalisateur
    data2_w=`cat $TMP/datafile | $DIR/vbusdecodecalo2 -c 1 6,15,1 | sed 's/\;/ /'`
    data2_Kw=`cat $TMP/datafile | $DIR/vbusdecodecalo2 -c 1 8,15,1 | sed 's/\;/ /'`
    data2_Mw=`cat $TMP/datafile | $DIR/vbusdecodecalo2 -c 1 10,15,1 | sed 's/\;/ /'`
    let "data2=1000000*data2_Mw+1000*data2_Kw+data2_w"
    data2=$data2";"
#    echo "$data2_Mw $data2_Kw $data2_w \n"
```

```
# Energie d'appoint Smart Process ou EASTRON SDM120C importee en Wh
   data3=`$DIR/sdm120c /dev/ttyUSB0 -i -q|awk '{print $1}'`          # Lecture du totalisateur

# Mesure de la date et du temps
   jour=`date +%d-%m-%y`
   temps=`date +%T`

# Construction de la ligne de mesure compatible .csv
   data="$jour $temps;$data0$data1$data2$data3;"

# Suppression de la derniere messure si erreur sur le wattmetre ou sauvegarde
   case $data3 in
                "NOK")
                    ;;
                  *)
                    echo $data
                    echo $data >> $TMP/data_all.csv  # Sauvegarde
                    echo $data > $TMP/data_one.csv
                        ;;
   esac

# Si out.time existe il donne l'heure de mise en marche de l'appoint
# on a alors delta_t secondes pour activer l'appoint
# puis out.time est renome

now=$(date +"%s")
if [ -f $MW_DIR/out.time ]
then
   h_appoint_on=`cat $MW_DIR/out.time`
   let "delta_t=30"
   depart=$(date "-d$h_appoint_on" +"%s")
    #calcul du delais pour envoyer au moins une commande
   let "limit=depart+delta_t"
   if [ "$now" -ge "$depart" ] && [ "$now" -lt "$limit" ]
   then
#          echo "appoint_on:" $h_appoint_on
           $DIR/appoint_on
           cat $MW_DIR/out.time >> $TMP/out.log
      # on renome le fichier pour avoir 1 seule occurrence
           mv $MW_DIR/out.time $TMP/out.old
   fi
fi

# Attente minimale entre 2 acquisitions
   sleep 5

done
```

6.3.2 Lecture du streaming hexadécimal VBUS grab.py

Le protocle émet sur requête un flot de données qu'il est nécessaire de filtrer pour recupèrer les trames utiles.

```
#!/usr/bin/python
# Reads data from a  Resol BS4 Solar Thermal Heating Controller
#
# Adds data to RRD file.

import os
import serial
import sys

# Set variable DEBUG to 1 for detailed debuggin information to stdout
#DEBUG = 1
DEBUG = 0
# open the Serial port at 9800 baud, with a timeout of 0 seconds
# we are using the GPIO exposed serial port --> /dev/ttyAMA0
# the last line of /etc/inittab which spawns the terminal listener for this
# console has been commented out

#LOOP=150
LOOP=300

# open a binary file for writing
f = open('/home/public/datafile', 'wb')

# the RESOL controller sends data in 9600 8N1 format

ser = serial.Serial(port='/dev/ttyACM0',
            baudrate=9600,
            bytesize=serial.EIGHTBITS,
            parity=serial.PARITY_NONE,
            stopbits=serial.STOPBITS_ONE,
            timeout=3)

if DEBUG: print("Port Opened .... listening") ;

data=[]
cnt=0

while cnt < LOOP:
        w = ser.inWaiting()
        if w > 0:
                data = ser.read(w)
                f.write(data)
                cnt = cnt + w
#               print(cnt)
#               for c in part:
#                       print("0x%02X" % c,sep=" ")

f.close()
ser.close()
```

6.3.3 Décodage des trames de mesure Vitosolic 200 Vbusdecodevito.cpp

Le code vbusdecode.cpp a été adapté au niveau de l'adresse de début de lecture en fonction des trames que l'on veut lire. Le code source est disponible sur
https://github.com/naylogj/hec-display/tree/master/hec/resol/src-vbusdecode

6.3.4 Décodage des trames des calorimètres Vitosolic 200 VbusdecodecaloX.cpp

Par commodité et pour une plus grande robustesse **vbusdecodevito.cpp** dérive 2 autres versions paramétrées pour décoder les trames spécifiques des calorimètres vbusdecodecalo1.cpp et vbusdecodecalo2.cpp

En fonction du la trame à décoder le test de trame de vbusdecode.cpp devient:
Mesures : if (framecount !=0 && themodel1 == 0x10 && themodel2 == **0x60**)
Calorimètre1 : if (framecount !=0 && themodel1 == 0x10 && themodel2 == **0x65**)
Calorimètre2: if (framecount !=0 && themodel1 == 0x10 && themodel2 == **0x66**)

Décodage des trames de mesure VBUS du régulateur Vitosolic 200 Vbusdecodevito.cpp
https://github.com/naylogj/hec-display/tree/master/hec/resol/src-vbusdecode

```
// arguments [-c] [-f filename] [-c count]  fields
// -f filename
//        will put last retrieved value to filename , overwriting anyother contents
//
// -c The number of full frames to decode, then exit
//
// fields
//        4 values , seperated by commas, where value
//                1 = offset from start of data, this value is bytes
//                2 = length, how many bits
//                3 =  if length > 1 this is the multiplier
//                        if length = 1 this is the bit position within the byte
//                                                use for bit fields, like error mask, relay mask etc.
//                4 = format, p = plus , add the returned value to the next field
//                                                t = time , format the output as time
//                                                any other value is ignored.
//                                                future will include c=*C, f=*F, l=logical (bit fields true/false), y = yes/no (bit fields)
// to get this information for your model, check the wiki or look in the Vbus XMl file in Resol lite.
//
// example:
// cat raw.log | ./a.out -f rrdvals -c 1 0,15,0.1 2,15,0.1 4,15,0.1 6,15,0.1 8,7,1 9,7,1 20,16,1,p 22,16,1000,p 24,16,1000000 12,16,0,t 10,1,0 10,1,1
// will give temp of s1,s2,s3,s4 pumpspeed pump1,pump2 and total of watts, formatted system time, r1 and r2 status for a resol deltasol bs plus, put them into a file called rrdvals, and exit after decoding one frame successfully
//45.7 24.6 49.8 29.1 100 0 2609964 14:36 1 0

// updated for RESOL BS4 V2 (RESOL BS 2009).  this emits different data sets we are not interested in all of them
// therefore we need to skip a frame where the first byte aftet the sync byte is in decimal 21
```

```
#include <stdio.h>
#include <iostream>
#include <string.h>
#include <stdlib.h>
using namespace std;

int adding = 0,model=0;
float totals;
string presults;
unsigned char frame[4],allframes[256];
char themodel1=0x00;
char themodel2=0x00;

void giveresults(char parray[])
{
   string formatter;
   char *format = NULL;
   float f;
   char results[24];
   int offset = atoi(strtok (parray,","));
   int length = atoi(strtok (NULL, ","));
   if (length == 1) {
      int bitposition = atoi(strtok (NULL, ","));
      f = (((allframes[offset] >> bitposition) & 0x01 ) * 0x01);
//           ammended line below replaced space with ,
      sprintf(results, "%.0f,",f);
      presults.append(results);
   } else  {
      float multiplier = atof(strtok (NULL, ","));
      if (((length -1) /24) ) {
         f     =     (allframes[offset]     +     (allframes[offset+1]*0x100)     +     (allframes[offset+2]*0x1000)     +
(allframes[offset+3]*0x10000) ) * multiplier;
      } else if (((length -1) /16) ) {
         f = (allframes[offset] + (allframes[offset+1]*0x100) + (allframes[offset+2]*0x1000) ) * multiplier;
      } else if (((length -1) /8) ) {
         f = (allframes[offset] + (allframes[offset+1]*0x100)) * multiplier;
      } else if (multiplier !=0 ) {
         f = (allframes[offset] ) * multiplier;
      }
      format = strtok (NULL, ",");
      if (format != NULL) {
         formatter = format;
         if (formatter == "p" ) {
            adding = 1;
            totals = totals + f;
         } else if (formatter == "t" ) {
            int time  = (allframes[offset] + (allframes[offset+1]*0x100));
            int hours = time/60;
            int mins = time - (hours*60);
            sprintf(results, "%02dh%02d;",hours,mins);
         } else if (formatter == "c1" ) {
               sprintf(results, "C1:%.0f;",f);
             } else if (formatter == "c2" ) {
```

```
                sprintf(results, "C2:%.0f;",f);
               }
      } else if (multiplier < 1) {
         sprintf(results, "%.1f;",f);
      } else sprintf(results, "%.0f;",f);
      if ((formatter != "p") && (adding == 1)) {
         totals = totals + f;
         sprintf(results, "%.0f;",totals);
         adding = 0;
         totals = 0;
      }
      if (formatter != "p") presults.append(results);
   }


}

int decodeheader()
{
   char  buffer[10];
   unsigned char a;
   unsigned char b;
   if
(scanf("%c%c%c%c%c%c%c%c%c",&buffer[0],&buffer[1],&buffer[2],&buffer[3],&buffer[4],&buffer[5],&buffer[
6],&buffer[7],&buffer[8],&buffer[9]) != 1)
   {
      a = buffer[0] + buffer[1] + buffer[2] + buffer[3] + buffer[4] + buffer[5] + buffer[6] + buffer[7];
      b = ~ a;
      a = buffer[8];
//          extra lines below - remove when working
//          printf("** %d ** ", buffer[0]);
//          printf("%d ", buffer[1]);
//          printf("%x ", buffer[2]);
//          printf("%x ", buffer[3]);
            //          printf("%d ", buffer[4]);
            //          printf("** %d **", buffer[5]);
            //          printf("%d ", buffer[6]);
            //          printf("%d ", buffer[7]);
            //          printf("%d ", buffer[8]);
            //          printf("%d ", buffer[9]);
   }

// ammended code here.  if buffer[0]=0x15 then we have a command we want to skip
   if ( buffer[0] == 21 )
   {
     return 0;
   }
// end of ammended code
   if (  a == b )
   {
      model = buffer[3] * 0x100  + buffer[2];
            themodel1=buffer[3];
            themodel2=buffer[2];
```

```
      a = buffer[7];
      return a;
   } else {
      return 0;
   }
}

int decodeframe(int x)
{
   char  buffer[7] ;
   unsigned char a;
   unsigned char b;
   if (scanf("%c%c%c%c%c%c",&buffer[0],&buffer[1],&buffer[2],&buffer[3],&buffer[4],&buffer[5]) != 1)
   {
      a = buffer[0]  + buffer[1]  + buffer[2] + buffer[3] + buffer[4] ;
      b = 0;
      b =  (( ~a | 0x80) - 0x80 ) ;
      a = buffer[5];
      if ( a == b )
      {
         frame[0] =  buffer[0] + (( buffer[4] & 0x01 ) * 0x80 ) ;
         frame[1] =  buffer[1] + (( buffer[4] >> 1 & 0x01 ) * 0x80 ) ;
         frame[2] =  buffer[2] + (( buffer[4] >> 2 & 0x01 ) * 0x80 ) ;
         frame[3] =  buffer[3] + (( buffer[4] >> 3 & 0x01 ) * 0x80 ) ;
      } else {
         frame[0] = 0;
         frame[1] = 0;
         frame[2] = 0;
         frame[3] = 0;
      }
   }
   return a - b;
}

int main(int argc, char* argv[])
{
   char  buffer[2];
   unsigned char a;
   char* arg_dup;
   int framecount,c=0,decodecount=0,decodedcount=0;
   FILE * pFile;
   while (( (decodedcount < decodecount) | (decodecount==0 ) ) && ( fgets(buffer, 2, stdin) != NULL))
   {
      a = buffer[0];
            //printf("%x ", themodel1);
            //printf("%x ", themodel2);
      if ( a ==  0xAA ) {
         framecount  =  decodeheader();
         if (framecount !=0  && themodel1 == 0x10 && themodel2 == 0x60 ) ← Choix de la tame
         {
            for ( int x = 0; x < framecount ; x++ )
            {
               c = c + decodeframe(x);
```

```
            allframes[4*x] = frame[0];
            allframes[(4*x)+1] = frame[1];
            allframes[(4*x)+2] = frame[2];
            allframes[(4*x)+3] = frame[3];
          }
          // if all crcs are ok
          decodedcount++;
          if (c == 0) {
            string filen = "stdout";
            for (int i = 1; i < argc; i++)
            {
              string sw = argv[i];
              if (sw == "-f") {
                i++;
                filen = argv[i];
              } else if (sw == "-c") {
                i++;
                decodecount = atoi(argv[i]);
              } else {
                arg_dup = strdup(argv[i]);
                giveresults(arg_dup);
                free(arg_dup);
              }
            }

            if (filen == "stdout")
            {
              fprintf(stdout,"%s\n",presults.c_str());
            } else {
              pFile = fopen (filen.c_str(),"w");
              fprintf (pFile,"%s\n",presults.c_str());
              fclose (pFile);
            }
            }

          presults = "";
        }
      }
    }
    return 0;
}
```

6.3.5 Décodage de la trame MODBUS du wattmètre EASTON SDM120C sdm120c.c

Le code original https://github.com/gianfrdp/SDM120C n'a pas été modifié.
Pour utiliser le code sdm120C il sera nécessaire d'installer la librairie libmodbus .

Notice d'utilisation du code sdm120c

SDM120C
SDM120C ModBus RTU client to read EASTRON SDM120C smart mini power meter registers

It works with SDM120C and SDM220 models

It depends on libmodbus (http://libmodbus.org)

To compile
make clean && make

Usage: sdm120c [-a address] [-d] [-x] [-p] [-v] [-c] [-e] [-i] [-t] [-f] [-g] [-T] [[-m]|[-q]] [-b baud_rate] [-P parity] [-S bit] [-z num_retries] [-j seconds] [-w seconds] [-1 | -2] device
sdm120c [-a address] [-d] [-b baud_rate] [-P parity] [-S bit] [-1 | -2] -s new_address device
sdm120c [-a address] [-d] [-b baud_rate] [-P parity] [-S bit] [-1 | -2] -r baud_rate device
sdm120c [-a address] [-d] [-b baud_rate] [-P parity] [-S bit] [-1 | -2] -R new_time device

where
-a address Meter number (between 1 and 247). Default: 1
-s new_address Set new meter number (between 1 and 247)
-p Get power (W)
-v Get voltage (V)
-c Get current (A)
-f Get frequency (Hz)
-g Get power factor
-e Get exported energy (Wh)
-i Get imported energy (Wh)
-t Get total energy (Wh)
-T Get Time for rotating display values (0 = no rotation)
-d Debug
-x Trace (libmodbus debug on)
-b baud_rate Use baud_rate serial port speed (1200, 2400, 4800, 9600)
Default: 2400
-P parity Use parity (E, N, O)
-S bit Use stop bits (1, 2). Default: 1
-r baud_rate Set baud_rate meter speed (1200, 2400, 4800, 9600)
-R new_time Change rotation time for displaying values (0 - 30s) (0 = no totation)
-m Output values in IEC 62056 format ID(VALUE*UNIT)
-q Output values in compact mode
-z num_retries Try to read max num_retries times on bus before exiting
with error. Default: 1
-j seconds Response timeout. Default: 2
-w seconds tTime to wait to lock serial port. (1-30) Default: 0
-1 Model: SDM120C (defualt)
-2 Model: SDM220
device Serial device, i.e. /dev/ttyUSB0

Serial device is required. When no parameter is passed, retrives all values

Décodage du protocole MODBUS pour le wattmètre EASTON sdm120c.c

https://github.com/gianfrdp/SDM120C

```
#ifdef __cplusplus
extern "C" {
#endif

/*
 * sdm120c: ModBus RTU client to read EASTRON SDM120C smart mini power meter registers
 *
 * Copyright (C) 2015 Gianfranco Di Prinzio <gianfrdp@inwind.it>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

#include <sys/types.h>
#include <time.h>

#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <ctype.h>
#include <getopt.h>

#include <modbus-version.h>
#include <modbus.h>

#define DEFAULT_RATE 2400

#define MODEL_120 1
#define MODEL_220 2

// Read
#define VOLTAGE   0x0000
#define CURRENT   0x0006
#define POWER     0x000C
#define APOWER    0x0012
#define RAPOWER   0x0018
#define PFACTOR   0x001E
#define PANGLE    0x0024
#define FREQUENCY 0x0046
#define IAENERGY  0x0048
#define EAENERGY  0x004A
#define IRAENERGY 0x004C
#define ERAENERGY 0x004E
#define TAENERGY  0x0156
#define TRENERGY  0x0158

// Write
```

```
#define NPARSTOP  0x0012
#define DEVICE_ID 0x0014
#define BAUD_RATE 0x001C
#define TIME_DISP_220 0xF500
#define TIME_DISP 0xF900
#define TOT_MODE  0xF920

#define BR1200 5
#define BR2400 0
#define BR4800 1
#define BR9600 2

#define MAX_RETRIES 100

#define E_PARITY 'E'
#define O_PARITY 'O'
#define N_PARITY 'N'

#define RESTART_TRUE  1
#define RESTART_FALSE 0

int debug_flag     = 0;
int EXIT_ERROR     = 0;
int trace_flag     = 0;

const char *version     = "1.2.0";
char *programName = "sdm120c";
const char *ttyLCKloc   = "/var/lock/LCK..."; /* location and prefix of serial port lock file */

long unsigned int PID;
static int yLockWait = 0; /* Seconds to wait to lock serial port */
char *devLCKfile = NULL;
char *devLCKfileNew = NULL;

void usage(char* program) {
   printf("sdm120c %s: ModBus RTU client to read EASTRON SDM120C smart mini power meter registers\n",version);
   printf("Copyright (C) 2015 Gianfranco Di Prinzio <gianfrdp@inwind.it>\n");
   printf("Complied with libmodbus %s\n\n", LIBMODBUS_VERSION_STRING);
   printf("Usage: %s [-a address] [-d] [-x] [-p] [-v] [-c] [-e] [-i] [-t] [-f] [-g] [-T] [[-m]|[-q]] [-b baud_rate] [-P parity] [-S bit] [-z num_retries] [-j
seconds] [-w seconds] [-1 | -2] device\n", program);
   printf("       %s [-a address] [-d] [-x] [-b baud_rate] [-P parity] [-S bit] [-1 | -2] [-z num_retries] [-j seconds] [-w seconds] -s new_address
device\n", program);
   printf("      %s [-a address] [-d] [-x] [-b baud_rate] [-P parity] [-S bit] [-1 | -2] [-z num_retries] [-j seconds] [-w seconds] -r baud_rate device
\n", program);
   printf("       %s [-a address] [-d] [-x] [-b baud_rate] [-P parity] [-S bit] [-1 | -2] [-z num_retries] [-j seconds] [-w seconds] -R new_time
device\n\n", program);
   printf("where\n");
   printf("\t-a address \tMeter number (between 1 and 247). Default: 1\n");
   printf("\t-s new_address \tSet new meter number (between 1 and 247)\n");
   printf("\t-p \t\tGet power (W)\n");
   printf("\t-v \t\tGet voltage (V)\n");
   printf("\t-c \t\tGet current (A)\n");
   printf("\t-f \t\tGet frequency (Hz)\n");
   printf("\t-g \t\tGet power factor\n");
   printf("\t-e \t\tGet exported energy (Wh)\n");
   printf("\t-i \t\tGet imported energy (Wh)\n");
   printf("\t-t \t\tGet total energy (Wh)\n");
   printf("\t-T \t\tGet Time for rotating display values (0 = no rotation) \n");
   printf("\t-d \t\tDebug\n");
   printf("\t-x \t\tTrace (libmodbus debug on)\n");
   printf("\t-b baud_rate \tUse baud_rate serial port speed (1200, 2400, 4800, 9600)\n");
   printf("\t\t\tDefault: 2400\n");
   printf("\t-P parity \tUse parity (E, N: Default, O)\n");
   printf("\t-S bit \t\tUse stop bits (1, 2). Default: 1\n");
   printf("\t-r baud_rate \tSet baud_rate meter speed (1200, 2400, 4800, 9600)\n");
   printf("\t-R new_time \tChange rotation time for displaying values (0 - 30s) (0 = no rotation)\n");
   printf("\t-m \t\tOutput values in IEC 62056 format ID(VALUE*UNIT)\n");
```

```
   printf("\t-q \t\tOutput values in compact mode\n");
   printf("\t-z num_retries\tTry to read max num_retries times on bus before exiting\n");
   printf("\t\t\twith error. Default: 1\n");
   printf("\t-j seconds\tResponse timeout. Default: 2\n");
   printf("\t-w seconds\tTime to wait to lock serial port. (1-30) Default: 0\n");
   printf("\t-1 \t\tModel: SDM120C (default)\n");
   printf("\t-2 \t\tModel: SDM220\n");
   printf("\tdevice\t\tSerial device, i.e. /dev/ttyUSB0\n\n");
   printf("Serial device is required. When no parameter is passed, retrives all values\n");
}

/*------------------------------------------------------------------------------
   getCurTime
------------------------------------------------------------------------------*/
char* getCurTime()
{
   time_t curTimeValue;
   struct tm timStruct;
   static char CurTime[18] = " ";

   curTimeValue = time(NULL);
   timStruct = *(localtime(&curTimeValue));
   strftime(CurTime,sizeof(CurTime),"%Y%m%d-%H:%M:%S",&timStruct);
   CurTime[sizeof(CurTime)-1] = '\0';

   return CurTime;
}

void log_message(const int debug, const char* format, ...) {
   va_list args;
   if (debug) {
     fprintf(stderr, "%s: ", getCurTime());
     va_start(args, format);
     vfprintf(stderr, format, args);
     va_end(args);
     fprintf(stderr, "\n");
   }
}

/*------------------------------------------------------------------------------
      getMemPtr
------------------------------------------------------------------------------*/
void *getMemPtr(size_t mSize)
{
   void *ptr;
   char *cptr;
   int i;

   ptr = malloc(mSize);
   if (!ptr) {
      log_message(debug_flag, "\nvproweather: malloc failed\n");
      exit(2);
   }
   cptr = (char *)ptr;
   for (i = 0; i < mSize; i++) cptr[i] = '\0';
   return ptr;
}

/*------------------------------------------------------------------------------
   ClrSerLock
   Clear Serial Port lock.
------------------------------------------------------------------------------*/
int ClrSerLock(long unsigned int PID) {
   FILE *fdserlck, *fdserlcknew;
   long unsigned int rPID;
   int bWrite, bRead;
   int errno_save = 0;
```

```
   errno = 0;
   log_message(debug_flag, "devLCKfile: <%s>", devLCKfile);
   log_message(debug_flag, "devLCKfileNew: <%s> ", devLCKfileNew);
   log_message(debug_flag, "Clearing Serial Port Lock (%lu)...", PID);
   fdserlck = fopen(devLCKfile, "r");
   if (fdserlck == NULL) {
      log_message(debug_flag, "%s: Problem opening serial device lock file to clear PID %lu: %s for read.",programName,PID,devLCKfile);
      return(0);
   }
   fdserlcknew = fopen(devLCKfileNew, "w");
   if (fdserlcknew == NULL) {
      log_message(debug_flag, "%s: Problem opening new serial device lock file to clear PID %lu: %s for
write.",programName,PID,devLCKfileNew);
      fclose(fdserlck);
      return(0);
   }
   bRead = fscanf(fdserlck, "%lu", &rPID);
   while (bRead != EOF) {
      if (rPID != PID) {
         errno = 0;
         bWrite = fprintf(fdserlcknew, "%lu\n", rPID);
         errno_save = errno;
         if (bWrite < 0 || errno_save != 0) {
            log_message(debug_flag, "%s: Problem clearing serial device lock, can't write lock file: %s. %s",programName,devLCKfile,strerror
(errno_save));
            fclose(fdserlcknew);
            return(0);
         }
      }
      bRead = fscanf(fdserlck, "%lu", &rPID);
   }
   fclose(fdserlck);
   fclose(fdserlcknew);
   errno = 0;
   if (rename(devLCKfileNew,devLCKfile)) {
      log_message(debug_flag, "%s: Problem clearing serial device lock, can't update lock file: %s.",programName,devLCKfile);
      log_message(debug_flag, "%s",strerror (errno));
   }
   log_message(debug_flag, "done");

   return -1;
}

inline void exit_error(modbus_t *ctx)
{
    modbus_close(ctx);
    modbus_free(ctx);
    //EXIT_ERROR++;
    ClrSerLock(PID);
    //return -1;
    printf(" NOK\n");
    exit(EXIT_FAILURE);
}

inline int bcd2int(int val)
{
   return((((val & 0xf0) >> 4) * 10) + (val & 0xf));
}

int int2bcd(int val)
{
   return(((val / 10) << 4) + (val % 10));
}

int bcd2num(const uint16_t *src, int len)
{
```

```
    int n = 0;
    int m = 1;
    int i = 0;
    int shift = 0;
    int digit = 0;
    int j = 0;
    for (i = 0; i < len; i++) {
        for (j = 0; j < 4; j++) {
            digit = ((src[len-1-i]>>shift) & 0x0F) * m;
            n += digit;
            m *= 10;
            shift += 4;
        }
    }
    return n;
}

#if 0

// unused

int getMeasureBCD(modbus_t *ctx, int address, int retries, int nb) {

    uint16_t tab_reg[nb * sizeof(uint16_t)];
    int rc;
    int i;
    int j = 0;
    int exit_loop = 0;

    while (j < retries && exit_loop == 0) {
      j++;

      log_message(debug_flag, "%d/%d. Register Address %d [%04X]", j, retries, 30000+address+1, address);
      rc = modbus_read_input_registers(ctx, address, nb, tab_reg);

      if (rc == -1) {
        log_message(debug_flag, "%s: ERROR %s, %d", programName, modbus_strerror(errno), j);
        modbus_flush(ctx);
        usleep(2500);
      } else {
        exit_loop = 1;
      }
    }

    if (rc == -1) {
      exit_error(ctx);
    }

    if (debug_flag == 1) {
      for (i=0; i < rc; i++) {
        log_message(debug_flag, "reg[%d/%d]=%d (0x%X)", i, (rc-1), tab_reg[i], tab_reg[i]);
      }
    }

    int value = bcd2num(&tab_reg[0], rc);

    return value;
}

#endif

float getMeasureFloat(modbus_t *ctx, int address, int retries, int nb) {

    uint16_t tab_reg[nb * sizeof(uint16_t)];
    int rc;
    int i;
    int j = 0;
```

```
    int exit_loop = 0;

    while (j < retries && exit_loop == 0) {
     j++;

     log_message(debug_flag, "%d/%d. Register Address %d [%04X]", j, retries, 30000+address+1, address);
     rc = modbus_read_input_registers(ctx, address, nb, tab_reg);

     if (rc == -1) {
      if (trace_flag) fprintf(stderr, "%s: ERROR (%d) %s, %d\n", programName, errno, modbus_strerror(errno), j);
      log_message(debug_flag, "ERROR (%d) %s, %d\n", errno, modbus_strerror(errno), j);
      modbus_flush(ctx);
      usleep(2500);
      if (errno=110) exit_loop = 1;   // Timeout error, retry is not needed
     } else {
      exit_loop = 1;
     }

    }

    if (rc == -1) {
     exit_error(ctx);
    }

    if (debug_flag == 1) {
      for (i=0; i < rc; i++) {
        log_message(debug_flag, "reg[%d/%d]=%d (0x%X)", i, (rc-1), tab_reg[i], tab_reg[i]);
      }
    }

    // swap LSB and MSB
    uint16_t tmp1 = tab_reg[0];
    uint16_t tmp2 = tab_reg[1];
    tab_reg[0] = tmp2;
    tab_reg[1] = tmp1;

    float value = modbus_get_float(&tab_reg[0]);

    return value;

}

int getConfigBCD(modbus_t *ctx, int address, int retries, int nb) {

    uint16_t tab_reg[nb * sizeof(uint16_t)];
    int rc;
    int i;
    int j = 0;
    int exit_loop = 0;

    while (j < retries && exit_loop == 0) {
     j++;

     log_message(debug_flag, "%d/%d. Register Address %d [%04X]", j, retries, 400000+address+1, address);
     rc = modbus_read_registers(ctx, address, nb, tab_reg);

     if (rc == -1) {
      log_message(debug_flag, "%s: ERROR %s, %d", programName, modbus_strerror(errno), j);
      modbus_flush(ctx);
      usleep(2500);
     } else {
      exit_loop = 1;
     }
    }

    if (rc == -1) {
     exit_error(ctx);
```

```
    }

    if (debug_flag == 1) {
      for (i=0; i < rc; i++) {
        log_message(debug_flag, "reg[%d/%d]=%d (0x%X)", i, (rc-1), tab_reg[i], tab_reg[i]);
      }
    }

    int value = bcd2num(&tab_reg[0], rc);

    return value;

}

void changeConfigFloat(modbus_t *ctx, int address, int new_value, int restart, int nb)
{
    uint16_t tab_reg[nb * sizeof(uint16_t)];

    modbus_set_float((float) new_value, &tab_reg[0]);
    // swap LSB and MSB
    uint16_t tmp1 = tab_reg[0];
    uint16_t tmp2 = tab_reg[1];
    tab_reg[0] = tmp2;
    tab_reg[1] = tmp1;

    int n = modbus_write_registers(ctx, address, nb, tab_reg);
    if (n != -1) {
        printf("New value %d for address 0x%X\n", new_value, address);
        if (restart == RESTART_TRUE) printf("You have to restart the meter for apply changes\n");
    } else {
        printf("%s: errno: %s, %d, %d\n", programName, modbus_strerror(errno), errno, n);
        exit_error(ctx);
    }
}

void changeConfigBCD(modbus_t *ctx, int address, int new_value, int restart, int nb)
{
    uint16_t tab_reg[nb * sizeof(uint16_t)];
    uint16_t u_new_value = int2bcd(new_value);
    tab_reg[0] = u_new_value;

    int n = modbus_write_registers(ctx, address, nb, tab_reg);
    if (n != -1) {
        printf("New value %d for address 0x%X\n", u_new_value, address);
        if (restart == RESTART_TRUE) printf("You have to restart the meter for apply changes\n");
    } else {
        log_message(debug_flag, "%s: errno: %s, %d, %d\n", programName, modbus_strerror(errno), errno, n);
        exit_error(ctx);
    }
}

void lockSer(const char *szttyDevice, int debug_flag)
{
    char *pos;
    FILE *fdserlck = NULL;
    long unsigned int rPID;
    char sPID[10];
    int bRead, bWrite, lckCNT;
    int errno_save = 0;
    int fLen = 0;
    char *cmdFile = NULL;
    char *command = NULL;
    char *SubStrPos = NULL;

    pos = strrchr(szttyDevice, '/');
    if (pos > 0) {
        pos++;
```

```
      devLCKfile = getMemPtr(strlen(ttyLCKloc)+(strlen(szttyDevice)-(pos-szttyDevice))+1);
      devLCKfile[0] = '\0';
      strcpy(devLCKfile,ttyLCKloc);
      strcat(devLCKfile, pos);
      devLCKfile[strlen(devLCKfile)] = '\0';
      sprintf(sPID,"%lu",PID);
      devLCKfileNew = getMemPtr(strlen(devLCKfile)+strlen(sPID)+2);          /* dot & terminator */
      devLCKfileNew[0] = '\0';
      strcpy(devLCKfileNew,devLCKfile);
      sprintf(devLCKfileNew,"%s.%lu",devLCKfile,PID);
      devLCKfileNew[strlen(devLCKfileNew)] = '\0';
   } else {
      devLCKfile = NULL;
   }

   log_message(debug_flag, "szttyDevice: %s",szttyDevice);
   log_message(debug_flag, "devLCKfile: <%s>",devLCKfile);
   log_message(debug_flag, "devLCKfileNew: <%s>",devLCKfileNew);

   log_message(debug_flag, "Attempting to get lock on Serial Port %s...",szttyDevice);
   fdserlck = fopen((const char *)devLCKfile, "a");
   if (fdserlck == NULL) {
      log_message(debug_flag, "%s: Problem locking serial device, can't open lock file: %s for write.",programName,devLCKfile);
      exit(2);
   }
   bWrite = fprintf(fdserlck, "%lu\n", PID);
   errno_save = errno;
   fclose(fdserlck);
   fdserlck = NULL;
   if (bWrite < 0 || errno_save != 0) {
      log_message(debug_flag, "%s: Problem locking serial device, can't write lock file: %s. %s",programName,devLCKfile,strerror
(errno_save));
      exit(2);
   }

   rPID = 0;
   lckCNT = -1;
   while(rPID != PID && lckCNT++ < yLockWait) {
      if (debug_flag && lckCNT == 0) log_message(debug_flag, "Checking for lock");
      SubStrPos = NULL;
      fdserlck = fopen(devLCKfile, "r");
      if (fdserlck == NULL) {
         log_message(debug_flag, "%s: Problem locking serial device, can't open lock file: %s for read.",programName,devLCKfile);
         exit(2);
      }
      bRead = fscanf(fdserlck, "%lu", &rPID);
      errno_save = errno;
      fclose(fdserlck);
      log_message(debug_flag, "Checking process %lu for lock", rPID);
      fdserlck = NULL;
      if (bRead == EOF || errno_save != 0) {
         log_message(debug_flag, "%s: Problem locking serial device, can't read lock file: %s.",programName,devLCKfile);
         log_message(debug_flag, "%s",strerror (errno_save));
         exit(2);
      }

      sPID[0] = '\0';
      sprintf(sPID,"%lu",rPID);
      cmdFile = getMemPtr(strlen(sPID)+14+1);
      cmdFile[0] = '\0';
      sprintf(cmdFile,"/proc/%lu/cmdline",rPID);
      cmdFile[strlen(cmdFile)] = '\0';
      log_message(debug_flag, "cmdFile=\"%s\"", cmdFile);
      fdserlck = fopen(cmdFile, "r");
      if (fdserlck != NULL) {
         fLen = 0;
         while (fgetc(fdserlck) != EOF) fLen++;
```

```
        if (fLen > 0) {
          command = getMemPtr(fLen+1);
          command[0] = '\0';
          rewind(fdserlck);
          bRead = fscanf(fdserlck, "%s", command);
          command[strlen(command)] = '\0';
          log_message(debug_flag, "command=\"%s\"", command);
        }
        fclose(fdserlck);
        fdserlck = NULL;
        if (command != NULL) SubStrPos = strstr(command, programName);
      }
      if (cmdFile != NULL) {
        free(cmdFile);
        cmdFile = NULL;
      }
      log_message(debug_flag, "rPID: %lu SubStrPos: %s command: %s %s",rPID,SubStrPos,command, (rPID == PID) ? " = me" : "");
      if (rPID != PID) {
         if (command == NULL) {        // Clear stale only if rPID process is dead (Aurora <= 1.8.8 needs a patch too)
            log_message(debug_flag, "%s: Clearing stale serial port lock. (%lu)", programName, rPID);
            ClrSerLock(rPID);
         } else if (yLockWait > 0)
            sleep(1);
      }
      if (command != NULL) {
        free(command);
        command = NULL;
      }
   }
   if (debug_flag && rPID == PID) log_message(debug_flag, "Appears we got the lock.");
   if (rPID != PID) {
      ClrSerLock(PID);
      log_message(debug_flag, "%s: Problem locking serial device %s, couldn't get the lock for %lu, locked by
%lu.",programName,szttyDevice,PID,rPID);
      exit(2);
   }
}

int main(int argc, char* argv[])
{

   int device_address = 1;
   int model         = MODEL_120;
   int new_address    = 0;
   int power_flag     = 0;
   int volt_flag      = 0;
   int current_flag   = 0;
   int freq_flag      = 0;
   int pf_flag        = 0;
   int apower_flag    = 0;
   int rapower_flag   = 0;
   int export_flag    = 0;
   int import_flag    = 0;
   int total_flag     = 0;
   int baud_rate      = 0;
   int stop_bits      = 1;
   int new_baud_rate  = 0;
   int metern_flag    = 0;
   int compact_flag   = 0;
   int time_disp_flag = 0;
   int rotation_time_flag = 0;
   int rotation_time  = 0;
   int count_param    = 0;
   int num_retries    = 1;
#if LIBMODBUS_VERSION_MAJOR >= 3 && LIBMODBUS_VERSION_MINOR >= 1 && LIBMODBUS_VERSION_MICRO >= 2
   uint32_t resp_timeout = 2;
#else
```

```
    time_t resp_timeout   = 2;
#endif
    int index;
    int c;
    char *szttyDevice  = NULL;
    char *c_parity     = NULL;
    int speed         = 0;
    int bits          = 0;
    int read_count     = 0;

    const char *EVEN_parity = "E";
    const char *NONE_parity = "N";
    const char *ODD_parity  = "O";
    //    char parity           = E_PARITY;
    char parity           = N_PARITY;
    programName       = argv[0];

    if (argc == 1) {
       usage(programName);
       exit(EXIT_FAILURE);
    }

    opterr = 0;

    while ((c = getopt (argc, argv, "a:b:cdefgij:lmnpP:qr:R:s:S:tTvw:xz:12")) != -1) {
       switch (c)
       {
          case 'a':
             device_address = atoi(optarg);
             if (!(0 < device_address && device_address <= 247)) {
                fprintf (stderr, "Address must be between 1 and 247.\n");
                exit(EXIT_FAILURE);
             }
             break;
          case 'v':
             volt_flag = 1;
             count_param++;
             break;
          case 'p':
             power_flag = 1;
             count_param++;
             break;
          case 'c':
             current_flag = 1;
             count_param++;
             break;
          case 'e':
             export_flag = 1;
             count_param++;
             break;
          case 'i':
             import_flag = 1;
             count_param++;
             break;
          case 't':
             total_flag = 1;
             count_param++;
             break;
          case 'f':
             freq_flag = 1;
             count_param++;
             break;
          case 'g':
             pf_flag = 1;
             count_param++;
             break;
          case 'l':
```

```
    apower_flag = 1;
    count_param++;
    break;
  case 'n':
    rapower_flag = 1;
    count_param++;
    break;
  case 'd':
    debug_flag = 1;
    break;
  case 'x':
    trace_flag = 1;
    break;
  case 'b':
    speed = atoi(optarg);
    if (speed == 1200 || speed == 2400 || speed == 4800 || speed == 9600) {
      baud_rate = speed;
    } else {
      fprintf (stderr, "Baud Rate must be one of 1200, 2400, 4800, 9600\n");
      exit(EXIT_FAILURE);
    }
    break;
  case 'P':
    c_parity = strdup(optarg);
    if (strcmp(c_parity,EVEN_parity) == 0) {
      parity = E_PARITY;
    } else if (strcmp(c_parity,NONE_parity) == 0) {
      parity = N_PARITY;
    } else if (strcmp(c_parity,ODD_parity) == 0) {
      parity = O_PARITY;
    } else {
      fprintf (stderr, "Parity must be one of E, N, O\n");
      exit(EXIT_FAILURE);
    }
    break;
  case 'S':
    bits = atoi(optarg);
    if (bits == 1 || bits == 2) {
      stop_bits = bits;
    } else {
      fprintf (stderr, "Stop bits can be one of 1, 2\n");
      exit(EXIT_FAILURE);
    }
    break;
  case 'r':
    speed = atoi(optarg);
    switch (speed) {
      case 1200:
        new_baud_rate = BR1200;
        break;
      case 2400:
        new_baud_rate = BR2400;
        break;
      case 4800:
        new_baud_rate = BR4800;
        break;
      case 9600:
        new_baud_rate = BR9600;
        break;
      default:
        fprintf (stderr, "Baud Rate must be one of 1200, 2400, 4800, 9600\n");
        exit(EXIT_FAILURE);
    }
    break;
  case 's':
    new_address = atoi(optarg);
    if (!(0 < new_address && new_address <= 247)) {
```

```
            fprintf (stderr, "New address must be between 1 and 247.\n");
            exit(EXIT_FAILURE);
        }
        break;
    case 'R':
        rotation_time_flag = 1;
        rotation_time = atoi(optarg);

        if (!(0 <= rotation_time && rotation_time <= 30)) {
            fprintf (stderr, "New rotation time must be between 0 and 30.\n");
            exit(EXIT_FAILURE);
        }
        break;
    case '1':
        model = MODEL_120;
        break;
    case '2':
        model = MODEL_220;
        break;
    case 'm':
        metern_flag = 1;
        break;
    case 'q':
        compact_flag = 1;
        break;
    case 'z':
        num_retries = atoi(optarg);
        if (!(0 < num_retries && num_retries <= MAX_RETRIES)) {
            fprintf (stderr, "num_reties must be between 1 and %d.\n", MAX_RETRIES);
            exit(EXIT_FAILURE);
        }
        break;
    case 'j':
        resp_timeout = atoi(optarg);
        break;
    case 'w':
        yLockWait = atoi(optarg);
        if (yLockWait < 1 || yLockWait > 30) {
            fprintf(stderr, "%s: -w Lock Wait seconds (%d) out of range, 1-30.\n",programName,yLockWait);
            return 0;
        }
        break;
    case 'T':
        time_disp_flag = 1;
        count_param++;
        break;
    case '?':
        if (optopt == 'a' || optopt == 'b' || optopt == 's') {
            fprintf (stderr, "Option -%c requires an argument.\n", optopt);
            usage(programName);
            exit(EXIT_FAILURE);
        }
        else if (isprint (optopt)) {
            fprintf (stderr, "Unknown option `-%c'.\n", optopt);
            usage(programName);
            exit(EXIT_FAILURE);
        }
        else {
            fprintf (stderr,"Unknown option character `\\x%x'.\n",optopt);
            usage(programName);
            exit(EXIT_FAILURE);
        }
    default:
        fprintf (stderr, "Unknown option `-%c'.\n", optopt);
        usage(programName);
        exit(EXIT_FAILURE);
}
```

```
  }

  if (optind < argc) {            /* get serial device name */
    szttyDevice = argv[optind];
  } else {
    /* need at least one argument (change +1 to +2 for two, etc. as needeed) */
    log_message(debug_flag, "optind = %d, argc = %d", optind, argc);
    usage(programName);
    fprintf(stderr, "%s: No serial device specified\n",programName);
    exit(EXIT_FAILURE);
  }

  if (compact_flag == 1 && metern_flag == 1) {
    fprintf(stderr, "Parameter -m and -q are mutually exclusive\n");
    usage(programName);
    exit(EXIT_FAILURE);
  }

  PID = getpid();
  lockSer(szttyDevice, debug_flag);

  modbus_t *ctx;
  if (baud_rate == 0) baud_rate = DEFAULT_RATE;

   ctx = modbus_new_rtu(szttyDevice, baud_rate, parity, 8, stop_bits);

   if (ctx == NULL) {
      log_message(debug_flag, "Unable to create the libmodbus context\n");
      ClrSerLock(PID);
      exit(EXIT_FAILURE);
   }

#if LIBMODBUS_VERSION_MAJOR >= 3 && LIBMODBUS_VERSION_MINOR >= 1 &&
LIBMODBUS_VERSION_MICRO >= 2

   uint32_t old_response_to_sec;
   uint32_t old_response_to_usec;

   modbus_get_response_timeout(ctx, &old_response_to_sec, &old_response_to_usec);

   // Considering to get those values from command line
   modbus_set_byte_timeout(ctx, -1, 0);
   modbus_set_response_timeout(ctx, resp_timeout, 0);

#else

   struct timeval old_response_timeout;
   struct timeval response_timeout;

   modbus_get_response_timeout(ctx, &old_response_timeout);
   response_timeout.tv_sec = -1;
   response_timeout.tv_usec = 0;
   modbus_set_byte_timeout(ctx, &response_timeout);
   response_timeout.tv_sec = resp_timeout;
   response_timeout.tv_usec = 0;
   modbus_set_response_timeout(ctx, &response_timeout);

#endif
```

```
  modbus_set_error_recovery(ctx,                    MODBUS_ERROR_RECOVERY_LINK                    |
MODBUS_ERROR_RECOVERY_PROTOCOL);

  if (trace_flag == 1) {
    modbus_set_debug(ctx, 1);
  }

  int slave = 0;
  slave = modbus_set_slave(ctx, device_address);

  if (modbus_connect(ctx) == -1) {
    log_message(debug_flag, "Connection failed: %s\n", modbus_strerror(errno));
    modbus_free(ctx);
    ClrSerLock(PID);
    exit(EXIT_FAILURE);
  }

  float voltage    = 0;
  float current    = 0;
  float power      = 0;
  float apower     = 0;
  float rapower    = 0;
  float pf         = 0;
  float freq       = 0;
  float imp_energy = 0;
  float exp_energy = 0;
  float tot_energy = 0;
  int   time_disp  = 0;

  if (new_address > 0 && new_baud_rate > 0) {

    log_message(debug_flag, "Parameter -s and -r are mutually exclusive\n\n");
    usage(programName);
    exit_error(ctx);
  } else if (new_address > 0) {

    if (count_param > 0) {
      usage(programName);
      modbus_close(ctx);
      modbus_free(ctx);
      ClrSerLock(PID);
      exit(EXIT_FAILURE);
    } else {
      // change Address
      changeConfigFloat(ctx, DEVICE_ID, new_address, RESTART_TRUE, 2);
      modbus_close(ctx);
      modbus_free(ctx);
      ClrSerLock(PID);
      return 0;
    }

  } else if (new_baud_rate > 0) {
```

```
    if (count_param > 0) {
        usage(programName);
        modbus_close(ctx);
        modbus_free(ctx);
        ClrSerLock(PID);
        exit(EXIT_FAILURE);
    } else {
        // change Baud Rate
        changeConfigFloat(ctx, BAUD_RATE, new_baud_rate, RESTART_TRUE, 2);
        modbus_close(ctx);
        modbus_free(ctx);
        ClrSerLock(PID);
        return 0;
    }

} else if (rotation_time_flag > 0) {

    if (count_param > 0) {
        usage(programName);
        modbus_close(ctx);
        modbus_free(ctx);
        ClrSerLock(PID);
        exit(EXIT_FAILURE);
    } else {
        // change Time Rotation
        if (model == MODEL_120) {
            changeConfigBCD(ctx, TIME_DISP, rotation_time, RESTART_FALSE, 1);
        } else {
            changeConfigBCD(ctx, TIME_DISP_220, rotation_time, RESTART_FALSE, 1);
        }
        modbus_close(ctx);
        modbus_free(ctx);
        ClrSerLock(PID);
        return 0;
    }

} else if (power_flag   == 0 &&
           volt_flag    == 0 &&
           current_flag == 0 &&
           freq_flag    == 0 &&
           pf_flag      == 0 &&
           export_flag  == 0 &&
           import_flag  == 0 &&
           total_flag   == 0 &&
           time_disp_flag == 0
  ) {
   // if no parameter, retrieve all values
    power_flag   = 1;
    volt_flag    = 1;
    current_flag = 1;
    freq_flag    = 1;
    pf_flag      = 1;
```

```
      export_flag  = 1;
      import_flag  = 1;
      total_flag   = 1;
      count_param   = power_flag + volt_flag + current_flag + freq_flag + pf_flag + export_flag + import_flag +
total_flag;
   }

   if (volt_flag == 1) {
      voltage = getMeasureFloat(ctx, VOLTAGE, num_retries, 2);
      read_count++;
      if (metern_flag == 1) {
         printf("%d(%4.2f*V)\n", device_address, voltage);
      } else if (compact_flag == 1) {
         printf("%4.2f ", voltage);
      } else {
         printf("Voltage: %4.2f V \n",voltage);
      }
   }

   if (current_flag == 1) {
      current  = getMeasureFloat(ctx, CURRENT, num_retries, 2);
      read_count++;
      if (metern_flag == 1) {
         printf("%d(%4.2f*A)\n", device_address, current);
      } else if (compact_flag == 1) {
         printf("%4.2f ", current);
      } else {
         printf("Current: %4.2f A \n",current);
      }
   }

   if (power_flag == 1) {
      power = getMeasureFloat(ctx, POWER, num_retries, 2);
      read_count++;
      if (metern_flag == 1) {
         printf("%d(%5.2f*W)\n", device_address, power);
      } else if (compact_flag == 1) {
         printf("%4.2f ", power);
      } else {
         printf("Power: %5.2f W \n", power);
      }
   }

   if (apower_flag == 1) {
      apower = getMeasureFloat(ctx, APOWER, num_retries, 2);
      read_count++;
      if (metern_flag == 1) {
         printf("%d(%5.2f*VA)\n", device_address, apower);
      } else if (compact_flag == 1) {
         printf("%5.2f ", apower);
      } else {
         printf("Active Apparent Power: %5.2f VA \n", apower);
      }
```

```
}

if (rapower_flag == 1) {
   rapower = getMeasureFloat(ctx, RAPOWER, num_retries, 2);
   read_count++;
   if (metern_flag == 1) {
      printf("%d(%5.2f*VAr)\n", device_address, rapower);
   } else if (compact_flag == 1) {
      printf("%4.2f ", rapower);
   } else {
      printf("Reactive Apparent Power: %5.2f VAr \n", rapower);
   }
}

if (pf_flag == 1) {
   pf = getMeasureFloat(ctx, PFACTOR, num_retries, 2);
   read_count++;
   if (metern_flag == 1) {
      printf("%d(%4.2f*-)\n", device_address, pf);
   } else if (compact_flag == 1) {
      printf("%4.2f ", pf);
   } else {
      printf("Power Factor: %4.2f \n", pf);
   }
}

if (freq_flag == 1) {
   freq = getMeasureFloat(ctx, FREQUENCY, num_retries, 2);
   read_count++;
   if (metern_flag == 1) {
      printf("%d(%4.2f*Hz)\n", device_address, freq);
   } else if (compact_flag == 1) {
      printf("%4.2f ", freq);
   } else {
      printf("Frequency: %4.2f Hz \n", freq);
   }
}

if (import_flag == 1) {
   imp_energy = getMeasureFloat(ctx, IAENERGY, num_retries, 2) * 1000;
   read_count++;
   if (metern_flag == 1) {
      printf("%d(%d*Wh)\n", device_address, (int)(imp_energy));
   } else if (compact_flag == 1) {
      printf("%d ", (int) imp_energy);
   } else {
      printf("Import Active Energy: %d Wh \n", (int)(imp_energy));
   }
}

if (export_flag == 1) {
   exp_energy = getMeasureFloat(ctx, EAENERGY, num_retries, 2) * 1000;
   read_count++;
```

```
        if (metern_flag == 1) {
            printf("%d(%d*Wh)\n", device_address, (int)(exp_energy));
        } else if (compact_flag == 1) {
            printf("%d ", (int) exp_energy);
        } else {
            printf("Export Active Energy: %d Wh \n", (int)(exp_energy));
        }
    }

    if (total_flag == 1) {
        tot_energy = getMeasureFloat(ctx, TAENERGY, num_retries, 2) * 1000;
        read_count++;
        if (metern_flag == 1) {
            printf("%d(%d*Wh)\n", device_address, (int)(tot_energy));
        } else if (compact_flag == 1) {
            printf("%d ", (int) tot_energy);
        } else {
            printf("Total Active Energy: %d Wh \n", (int)(tot_energy));
        }
    }

    if (time_disp_flag == 1) {
        time_disp = getConfigBCD(ctx, TIME_DISP, num_retries, 1);
        read_count++;
        if (compact_flag == 1) {
            printf("%d ", (int) time_disp);
        } else {
            printf("Display rotation time: %d\n", time_disp);
        }
    }

    if (read_count == count_param) {
        modbus_close(ctx);
        modbus_free(ctx);
        ClrSerLock(PID);
        printf(" OK\n");
    } else {
        exit_error(ctx);
    }

    /*
    if (compact_flag == 1) {
        printf("\n");
    }
    */

    return 0;
}

#ifdef __cplusplus
}
#endif
```

6.4 Séquenceur journalier cron

Le service Linux cron se configure et se gère à l'aide de la commande crontab avec des droits administrateur root.

Commande de configuration : #sudo crontab confile_root

Fichier cronfile_root :

00 22 * * * /usr/bin/python /usr/local/bin/forecast_h.py > /home/public/prevision.csv
30 22 * * * /bin/bash /usr/local/bin/MW/predict.sh >> /home/public/predict.log
30 06 * * * /bin/bash /usr/local/bin/appoint_off >> /home/public/appoint.log

Remarque:
Le contrôleur non prédictif ne lance à 22h30 la commande **appoint_on** au lieu de **predict.sh**

6.4.1 Prévisions météo pour les prochaines 36h forecast_h.py

Weather Underground fournie une API qui utilise la géolocalisation en entrée (Antibes).
Forecast_h.py renvoie un tableau de 36 lignes dont l'entête est :

Indice ; Heure ; Prévision de nébulosité ; Prévision de température ambiante

Les données reçues sont stockées dans le fichier **prevision.csv**.

```
#!/usr/bin/python
import urllib2
import json
f = urllib2.urlopen('http://api.wunderground.com/api/be854d38bd0247f7/hourly/q/FR/Antibes.json')
json_string = f.read()

for i in range(0, 36):
    parsed_json = json.loads(json_string)
    date = parsed_json['hourly_forecast'][i]['FCTTIME']['hour']
    prevision = parsed_json['hourly_forecast'][i]['fctcode']
    temp = parsed_json['hourly_forecast'][i]['temp']['metric']
#   print "%s: Date: %s Ciel: %s Temp: %s" % (i, date, prevision, temp)
    print "%s;%s;%s;%s;" % (i, date, prevision, temp)

f.close()
```

6.4.2 Gestion des commandes d'appoint. appoint_on appoint_off

Le but est d'envoyer des ordres de fermeture et d'ouverture sur un des relais de sortie de la carte PIFACE D2. Le code relais.py réalise cet ordre paramétré.

Appoint_on	**Appoint_off**
#!/bin/bash /usr/local/bin/relais.py -o "**1**,0,0,0,0,0,0,0"	#!/bin/bash /usr/local/bin/relais.py -o "**0**,0,0,0,0,0,0,0"

Relais.py

Ce code permet de érer les sorties de la carte fille du Raspberry Pi2 PIFACE D2

```
#!/usr/bin/env python3
#Test relais

import time
import pifacedigitalio as pfio
import os
import sys
import optparse

pfio.init()
leds = [0, 1, 2, 3, 4, 5, 6, 7]

parser = optparse.OptionParser()
parser.add_option('-o','--output',
                  dest="output_filename",
                  default="default.out",
                  )
options, remainder = parser.parse_args()

#rint 'output   :', options.output_filename

liste_sorties = options.output_filename.split(",")

i=0
for valeurs in liste_sorties:
   if int(valeurs) == 1:
      pfio.digital_write(leds[i], 1)
   else:
      pfio.digital_write(leds[i], 0)
   i+=1
```

6.5 Code de calcul de embarqué Matlab/Simulink prediction

Détail de la fonction Acquisition

Toutes les fonctions de base d'entrée / sortie de Matlab ne sont pas utilisable dans l'environnement Simulink il est nécessaire de prendre quelques précautions et de faire des tests de robustesse.
La fonction Acquisitions est une interface qui permet la récupération de données dans les fichiers dynamiques partagés du Raspberry. Plus particulièrement des appels à des fonctions d'entrée/sortie en C natives sont utilisés. Des fonctions particulières Matlab sont utilisées pour orienter la compilation du code sur le Raspberry.

Acquisition

```
function [datas, prevision]= acqusitions()%#codegen
coder.extrinsic('strtok');
% mise en forme des datas
%datas = ones(1,20,'double');
v = ones(1,20,'double');
chaine = readfile('data_one.csv');
index = int32(size(chaine));
token = chaine(1:index(1,2));
remain = strrep(token, '-', ';');
remain = strrep(remain, ' ', ';');
remain = strrep(remain, ':', ';');
datas = read_data(remain, v);

% mise en forme de la prevision
%prevision = ones(36,4,'double');
w = ones(1,144,'double');
chaine = readfile('prevision.csv');
index = int32(size(chaine));
token = chaine(1:index(1,2));
remain = strrep(token, ' ', '');
prevision = read_data(remain, w);

% Conversion chaine en double
function y = read_data(string, s_form)
remain = string;
t = s_form;
l = int32(size(string));
i = 1;
while l(1,2) > 3
   [token, remain] = strtok(remain,';'); %#ok<STTOK>
   l = int32(size(remain));
   t(1,i) = real(str2double(token));
%    fprintf('%3.1f ', t(1,i) );
   i = i + 1;
end
%fprintf('\n');
y = t;

% y = readfile(filename)
% Read file 'filename' and return a MATLAB string with the contents
```

```
% of the file. Internally C functions fopen/fread are used to read
% data from the file.
function y = readfile(filename) %#codegen

% The 'fprintf' function will not be compiled and instead passed
% to the MATLAB runtime. If we choose to generate code for this example,
% all calls to extrinsic functions are automatically eliminated.
coder.extrinsic('fprintf');

% Put class and size constraints on function input.
assert(isa(filename, 'char'));
assert(size(filename, 1) == 1);
assert(size(filename, 2) <= 1024);

% Define a new opaque variable 'f' which will be of type 'FILE *'
% in the generated C code initially with the value NULL.
f = coder.opaque('FILE *', 'NULL');

% Call fopen(filename 'r'), but we need to convert the MATLAB
% string into a C type string (which is the same string with the
% NUL (\0) string terminator).
f = coder.ceval('fopen', c_string(filename), c_string('r'));

% Call fseek(f, 0, SEEK_END) to set file position to the end of
% the file.
coder.ceval('fseek', f, int32(0), coder.opaque('int', 'SEEK_END'));

% We need to initialize the variable 'filelen' to the proper type
% as custom C functions are not analyzed.
filelen = int32(0);

% Call ftell(f) which will return the length of the file in bytes
% (as current file position is at the end of the file).
filelen = coder.ceval('ftell', f);

% Reset current file position
coder.ceval('fseek', f, int32(0), coder.opaque('int', 'SEEK_SET'));

% Initialize a buffer
buffer = zeros(1,65536,'uint8');

% Remaining is the number of bytes to read (from the file)
remaining = filelen;

% Index is the current position to read into the buffer
index = int32(1);

while remaining > 0
  % Buffer overflow?
  if remaining + index > size(buffer,2)
    fprintf('Attempt to read file which is bigger than internal buffer.\n');
    fprintf('Current buffer size is %d bytes and file size is %d bytes.\n', size(buffer,2), filelen);
    break
```

```
    end
    % Read as much as possible from the file into internal buffer
    nread = coder.opaque('size_t');
    nread = coder.ceval('fread', coder.ref(buffer(index)), int32(1), remaining, f);
    n = int32(0);
    n = coder.ceval('(int)',nread);
    if n == 0
        % Nothing more to read
        break;
    end
    % Did something went wrong when reading?
    if n < 0
        fprintf('Could not read from file: %d.\n', n);
        break;
    end
    % Update state variables
    remaining = remaining - n;
    index = index + n;
end

% Close file
coder.ceval('fclose', f);

%fprintf('%s\n', char(buffer(1:index)));

y = char(buffer(1:index));

% Create a NUL terminated C string given a MATLAB string
function y = c_string(s)
y = [s 0];
```

Reg

Version à 1 paramêtre : prevision de nebulosité

```
function heure_depart = reg(datas,vect_prevision)
%datas = ones(1,20,'double');
h = zeros(3,1,'double');
                        %en heure decimale
h_prev = 22;            %heure de releve des previsions pour 36h
h_debut = 8;            %heure de debut de journee ensoleillee
h_fin = 17;             %heure de fin de journee ensoleillee
h_fin_appoint = 6.5;  %appoint_off à 6h30 dans tous les cas
                        %ici 6h27 pour être certain de l'arret
indice_debut = int32(23 - h_prev + h_debut);
indice_fin = int32(23 - h_prev + h_fin);

%reformatage du vecteur prevision
prevision = zeros(36,4,'double');
for i = 0:35
    for j = 1:4
        prevision((i + 1),j) = vect_prevision((i * 4) + j);
    end
end
% moyenne de prevision sur la perdiode d'ensoleillement
cumul = 0.0;
for i = indice_debut:indice_fin
    %Filtrage des previsions
    p = double(prevision(i,3));
       if ( p == 7 ) || ( p == 8)
            p = 2;
        end
        if ( p >= 9) || ( p == 5) || ( p == 6)
            p = 4;
        end
    cumul = p + cumul;

end
nb_previsions = double(indice_fin - indice_debut + 1);
prevision_moy = double(cumul / nb_previsions );

% equation de la droite de transfert (temps de fonctionnement) =
f(nebulosité) appoint y=-2(x-1)
h_dec = h_fin_appoint - 2 * (prevision_moy - 1);
if ( h_dec <= 0.0 )
    h_dec = 24.0 + h_dec;
end
%Mise en forme de heure hh:mm:ss pour le fichier de sortie
h(1) = floor(h_dec);
h(2) = floor((h_dec - h(1)) * 60);
h(3) = floor((((h_dec - h(1)) * 60) - h(2)) * 60);

%affichage debug
%formatSpec = 'prevision= %3.2f cumul= %3.2f nb_previsions= %02.0f
h_depart_dec= %3.2f\n';
%fprintf(formatSpec, prevision_moy, cumul, nb_previsions, h_dec);

%y = datas;
heure_depart = h;
```

Reg

Version à 2 paramêtres : prevision de nebulosité et température du ballon conforme au modèle

```
function heure_depart = reg(datas,vect_prevision)
%datas = ones(1,20,'double');
tba = datas(10);        %temperature ballon au niveau de l'appoint
duree_appoint = 0.0;
h = zeros(3,1,'double');
                        %en heure decimale
h_prev = 22;            %heure de releve des previsions pour 36h
h_debut = 8;            %heure de debut de journee ensoleillee
h_fin = 17;             %heure de fin de journee ensoleillee
h_fin_appoint = 6.5;  %appoint_off à 6h30 dans tout les cas
                        %ici 6h27 pour être certain de l'arret
indice_debut = int32(23 - h_prev + h_debut);
indice_fin = int32(23 - h_prev + h_fin);

%reformatage du vecteur prevision
prevision = zeros(36,4,'double');
for i = 0:35
    for j = 1:4
        prevision((i + 1),j) = vect_prevision((i * 4) + j);
    end
end
% moyenne de prevision sur la perdiode d'ensoleillement
cumul = 0.0;
for i = indice_debut:indice_fin
    %Filtrage des previsions
    p = double(prevision(i,3));
       if ( p == 7 ) || ( p == 8)
            p = 2;
        end
        if ( p >= 9) || ( p == 5) || ( p == 6)
            p = 4;
        end
    cumul = p + cumul;

end
nb_previsions = double(indice_fin - indice_debut + 1);
prevision_moy = double(cumul / nb_previsions );

pm = prevision_moy;
% calcul de la durée appoint en fct de tba et des previsions moyennes
if ( tba < 40 )
    if ( pm < 2)
        duree_appoint = 2;
    end
    if ( pm >= 2) && ( pm <= 3)
        duree_appoint = 3;
    end
    if ( pm > 3)
        duree_appoint = 4;
    end
end

if ( tba >= 40 ) && ( tba <= 47 )
```

```
    if ( pm < 2)
        duree_appoint = 1;
    end
    if ( pm >= 2) && ( pm <= 3)
        duree_appoint = 2;
    end
    if ( pm > 3)
        duree_appoint = 3;
    end
end

if ( tba > 47 )
    if ( pm < 2)
        duree_appoint = 0;
    end
    if ( pm >= 2) && ( pm <= 3)
        duree_appoint = 1;
    end
    if ( pm > 3)
        duree_appoint = 2;
    end
end

% Calcul de l'heure de mise enroute de l'appoint
h_dec = h_fin_appoint - duree_appoint;
if ( h_dec <= 0.0 )
    h_dec = 24.0 + h_dec;
end
%Mise en forme de heure hh:mm:ss pour le fichier de sortie
h(1) = floor(h_dec);
h(2) = floor((h_dec - h(1)) * 60);
h(3) = floor((((h_dec - h(1)) * 60) - h(2)) * 60);

%affichage debug
%formatSpec = 'prevision= %3.2f cumul= %3.2f nb_previsions= %02.0f
h_depart_dec= %3.2f\n';
%fprintf(formatSpec, prevision_moy, cumul, nb_previsions, h_dec);

%fprintf('Previsions moyenne: %02.0f\n', prevision_moy);
%fprintf('tba: %4.2f\n', datas(10));

%y = datas;
heure_depart = h;
```

Save

```
function y = save(h)
h_debut_appoint = 22; %heure depart minimum de l'appoint
h_fin_appoint = 6.5;  %appoint_off à 6h30 dans tous les cas
dt= 0.1;
formatSpec = '%02.0f:%02.0f:%02.0f\n';
h_dec= h(1) + (h(2) / 60);

fprintf(formatSpec, h(1),h(2),h(3));
% le fichier out.time sera cree si l'appoint est necessaire
if (h_dec >= h_debut_appoint) || (h_dec <= h_fin_appoint - dt)
    fileID = fopen('out.time', 'w');
    fprintf(fileID, formatSpec, h(1),h(2),h(3));
    fclose(fileID);
end

y = 1;
```

6.6 Correspondance des indices météo et pictogrammes

Forecast Description Numbers

This table represents fctcode you can encounter in some of the Hourly features. The fctcode correspond to the forecast phrase described in this table. Using the "icon" field in the forecast is a better value to key off than fctcode. This table is just a description of the weather associated with the fctcode.

fctcode	Forecast	Icon Image	fctcode	Forecast	Icon Image
1	Clear		13	Rain	
2	Partly Cloudy		14	Chance of a Thunderstorm	
3	Mostly Cloudy		15	Thunderstorm	
4	Cloudy		16	Flurries	
5	Hazy		17	OMITTED	
6	Foggy		18	Chance of Snow Showers	
7	Very Hot		19	Snow Showers	
8	Very Cold		20	Chance of Snow	
9	Blowing Snow		21	Snow	
10	Chance of Showers		22	Chace of Ice Pellets	
11	Showers		23	Ice Pellets	
12	Chance of Rain		24	Blizzard