Table des matières

1 Introduction

1.1 Préambule

L'automatisation de systèmes utilise une place prépondérante dans les habitations d'aujourd'hui. La commande de chauffage est un élément essentiel au confort des habitants. Toutefois, il en ressort qu'en périodes d'entre-saison, des facteurs externes ont pour conséquence une augmentation ou une diminution de température désagréable au bien-être de l'occupant.

1.2 Interface de commande de chauffages

C'est là qu'est née l'idée de fausser les valeurs de commande de chauffage (de type mazout, électrique, pompe à chaleur ou chauffe-eau) de manière à contrer les facteurs externes et garder une température constante et agréable dans la résidence. Le chauffage est contrôlé par un régulateur qui lit une ou plusieurs sondes de température. La figure 1 démontre de manière schématique le système étudié lors de ce travail.

L'appareil faussé est une sonde de température extérieure. Un automate programmable industriel (API), appareil flexible et très utilisé pour l'automatisation de systèmes, ainsi qu'une carte électronique de type PCB permettront d'imiter le comportement d'une sonde.

La carte électronique est imaginée, dimensionnée et créée, en se basant sur le travail de diplôme d'un ancien étudiant. Elle fait office d'interface entre l'API, la sonde de température et le régulateur de chauffage. De l'électronique est utilisée pour simuler le comportement de la sonde de température extérieure. Un relais permet de choisir si la sonde réelle ou l'électronique (simulation de la sonde) sera branchée au régulateur de chauffage.

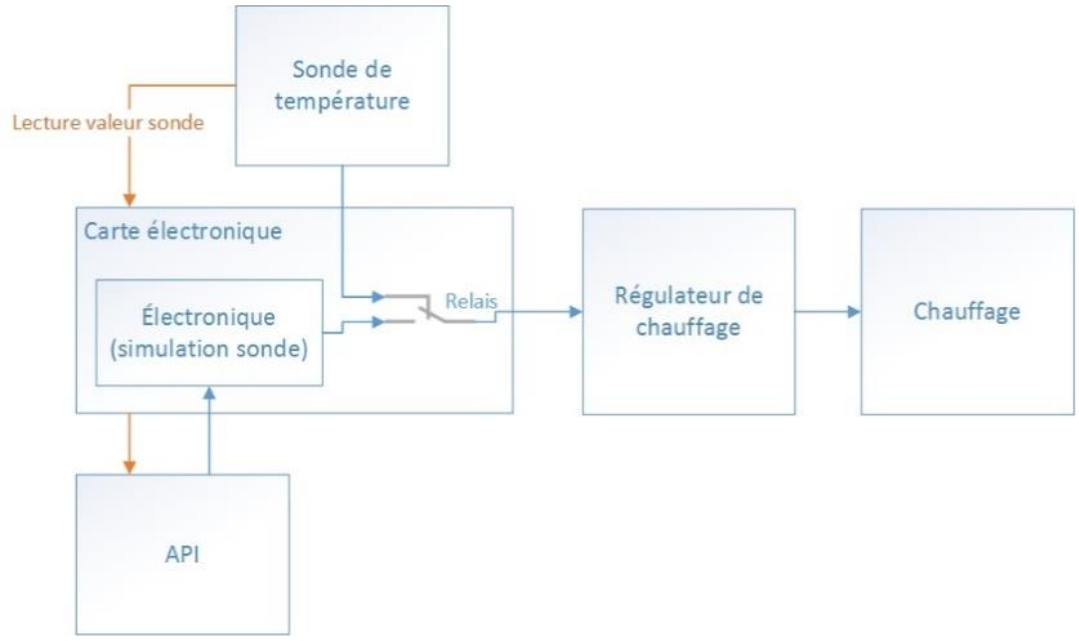


Figure 1 : Rôle de la carte électronique

1.3 Serveur d'alarme

Dans une seconde partie, un serveur d'alarme est créé sur un automate Wago nouvelle génération PFC 200. Ce sera l'API MASTER. Ce serveur d'alarme se situe dans les locaux d'une entreprise de la région. Il contient une base de données des clients ainsi que leurs numéros de téléphone. Si une alarme d'un client survient (en provenance d'un API SLAVE), un ou plusieurs SMS sont envoyés aux numéros de téléphones mobile associés à ce client.

Une visualisation web est aussi créée, sur ce même API, pour permettre d'ajouter, supprimer ou modifier les informations des clients.

1.4 Mise en commun de la partie gestion des alarmes avec la partie pilotage de chauffages

Si un problème ou quelque chose d'anormal survient lors du pilotage de chauffage, une erreur est transmise depuis l'automate qui commande la carte électronique (automate considéré comme si c'était celui d'un client de l'entreprise régionale) en direction de l'API serveur d'alarme. Ce dernier enverra donc un sms au propriétaire de l'habitation contenant la nature du défaut.

1.5 Serveur météo

Ensuite, si le temps à disposition le permet, un serveur météo sur automate Wago est à réaliser. Des requêtes sont envoyées auprès du serveur *openweathermap* pour obtenir les données météo. Ces données sont ensuite stockées dans des variables, puis retranscrites dans un fichier XML généré par l'API. Les automates des clients envoient une requête auprès de l'automate MASTER pour récupérer le fichier XML contenant les données météo. Ainsi, si le serveur *openweathermap* n'est plus disponible, il suffit d'adapter la programmation de l'API MASTER.

2 Interface de commande de chauffages – Partie hardware

2.1 Situation de départ

Monsieur Jérôme Catteeuw, ancien étudiant à la HES-SO de Sion, a développé lors de son travail de diplôme une plaque électronique de type PCB utilisée pour piloter une pompe à chaleur. Cette carte électronique interagit avec un automate programmable industriel (API) de marque Wago.

Avant que M. Catteeuw n'intervienne, la pompe à chaleur étudiée fonctionnait avec le principe suivant :

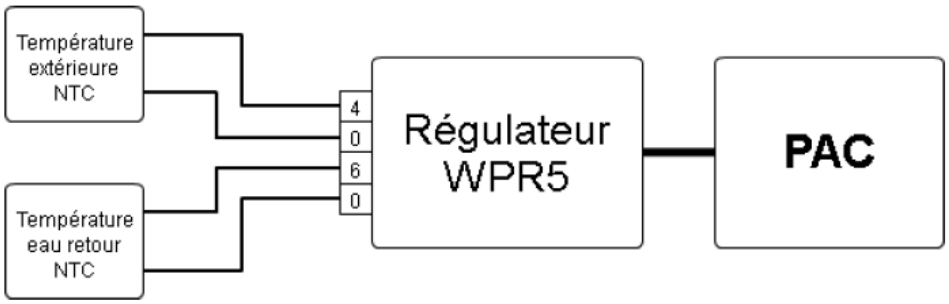


Figure 2 : Régulation de base d'une pompe à chaleur donnée

« Deux sondes de type NTC sont connectées au régulateur de la pompe à chaleur pour mesurer la température extérieure et la température retour de l'eau. Comme le montre la figure 2, le régulateur de la PAC[1] ne mesure pas la température ambiante de l'habitation. »[2]

Le présent régulateur compare les valeurs de ces deux thermistances et, le cas échéant, allume ou éteint la pompe à chaleur.

A noter qu'il existe d'autres régulateurs de chauffages qui mesurent la température ambiante de l'habitation.

[1] PAC : Pompe à chaleur
[2] Source : Travail de bachelor API_PAC, Diplôme 2016, Jérôme Catteeuw, cf. p.11

Après les modifications effectuées par M. Catteeuw, la régulation de la pompe à chaleur fonctionne comme suit :

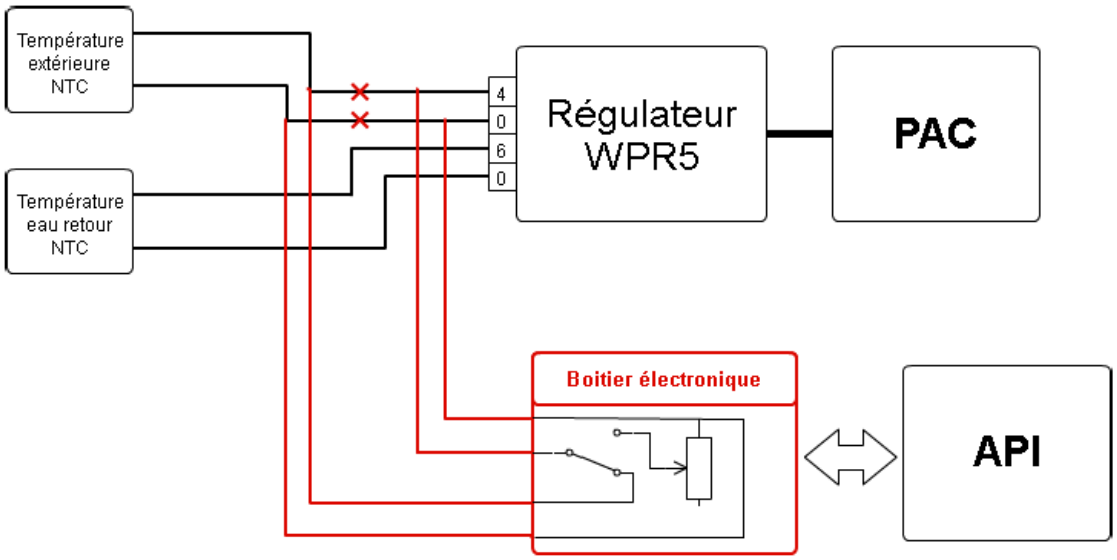


Figure 3 : Implantation de la carte électronique dans le système PAC existant

Le régulateur ne reçoit plus la valeur directe de la sonde NTC extérieure mais reçoit la valeur du potentiomètre électronique appartenant à la plaque électronique PCB. Ce qui permet de donner au régulateur WPR5 une valeur de température extérieure faussée, de manière à piloter indirectement la pompe à chaleur. En effet, si la température faussée est bien inférieure à la température extérieure réelle, le régulateur donnera l'ordre à la PAC de s'enclencher. L'intelligence est dans l'automate programmable industriel.

Si une coupure de courant survient, la sonde de température extérieure est immédiatement reconnectée au régulateur, tandis que la plaque électronique est déconnectée. Cela permet d'avoir un système sûr.

Le schéma électronique de cette carte électronique est disponible en annexe 2a.

2.1.1 Système proposé en automne 2016

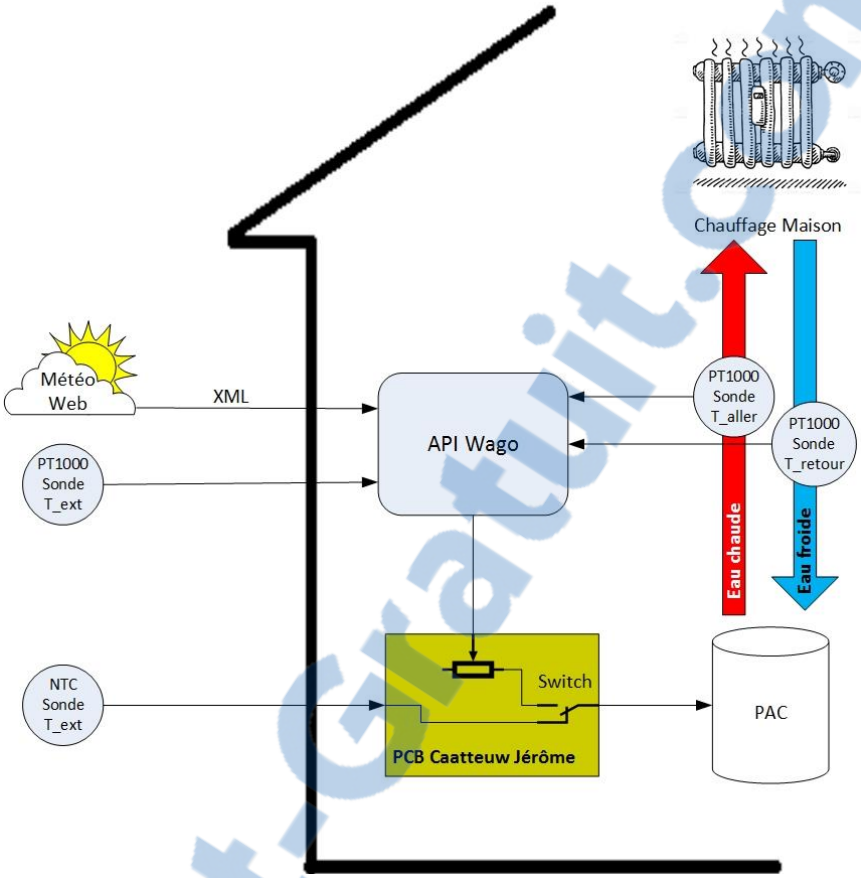


Figure 4 : Système implémenté en automne 2016, par M. Catteeuw, à Ayent

L'automate Wago envoie une requête vers un serveur météo pour connaître les prévisions météo du lieu de l'habitation. Un fichier XML est alors généré et reçu par l'automate, via Ethernet. En ayant ces prévisions météo ainsi que les températures extérieures et les températures aller (chaud) et retour (froid) de l'eau du chauffage, l'automate choisit la valeur du potentiomètre sur la carte électronique PCB et la transmet à la pompe à chaleur. Cette dernière est persuadée que cette valeur reçue est celle de la sonde NTC de température extérieure.

°C	-20	-15	-10	-5	0	5	10	15	20	25
k-Ohm	14,6	11,4	8,9	7,1	5,6	4,5	3,6	3,0	2,4	2

°C	30	35	40	45	50	55	60	65	70	75
k-Ohm	1,66	1,38	1,1	0,97	0,82	0,7	0,59	0,51	0,43	0,37

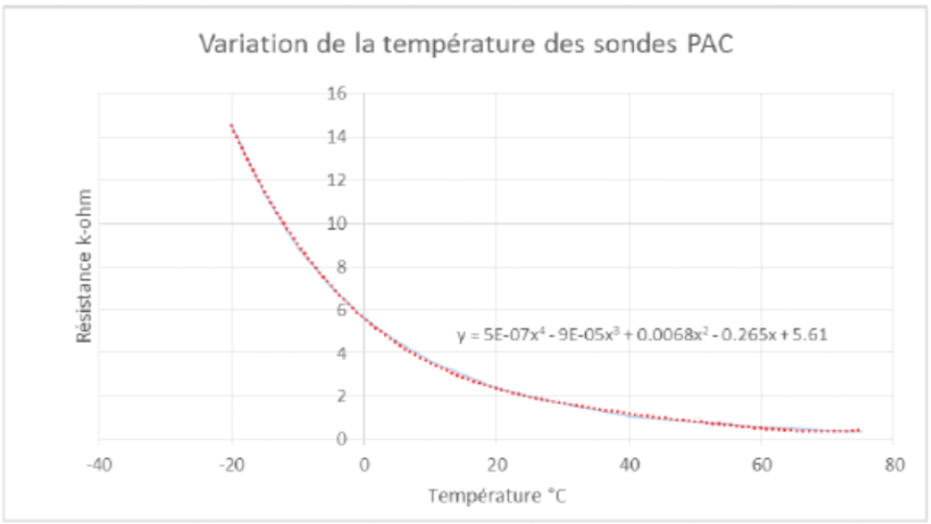


Figure 5 : Sonde de température NTC 2kΩ[3]

La thermistance de la sonde NTC 2kΩ varie entre 370 Ω et 14.6 kΩ, il a donc fallu dimensionner les composants de la carte PCB de manière à reproduire les mêmes effets que la sonde NTC.

La régulation biaisée se fait donc avec seulement la valeur d'une sonde « faussée » que le régulateur perçoit. Aucune modification n'est effectuée entre le régulateur et la pompe à chaleur. Cela permet de garder la garantie du régulateur et de la pompe à chaleur.

2.2 Problématique

La carte électronique PCB conçue par M. Catteeuw fonctionne pour ce type de régulateur et pour ce type d'application. Il est alors impossible de l'utiliser pour un autre régulateur utilisant une sonde ayant une gamme différente de valeurs de résistances.

Une nouvelle carte électronique est donc créée afin d'être utilisée dans un plus large champ d'applications pour le chauffage de bâtiments.

[3] Source : Travail de bachelor API_PAC, Diplôme 2016, Jérôme Catteeuw, cf. p.12

2.3 Nouveau cahier des charges

La nouvelle carte électronique doit être plus flexible afin de répondre aux objectifs suivants :

- Élargir les possibilités d'utilisation de la carte en proposant plusieurs gammes de résistances variables
- Fournir des points de mesures pour effectuer les tests à l'aide d'appareils de mesures
- Offrir à l'API une possibilité de lecture des potentiomètres, pour pouvoir ajuster la valeur de la résistance voulue

Ses composants doivent cependant être à un prix raisonnable.

2.4 Equipement

Pour trouver la solution la plus adéquate et flexible possible, il est important de bien comprendre le fonctionnement de chaque appareil (c'est-à-dire l'API, la sonde et le régulateur) qui interagit avec la plaque électronique PCB.

2.4.1 Sonde de température extérieure

Plusieurs types de sondes existent sur le marché. Les plus répandues sont les PT100, PT1000, Ni1000, Ni1000 TK5000, NTC 1kΩ, NTC 2kΩ, NTC 10kΩ, NTC 20kΩ.

Temp °C	PT100 Ω	PT1000 Ω	Ni1000 Ω	Ni1000 TK5000 Ω	NTC 1kOhm Ω	NTC 2kOhm Ω	NTC 5kOhm Ω	NTC 10kOhm kΩ	NTC 20kOhm kΩ	KTY 81-210 Ω
-50	80	803	743	791	32886	77977	333914	668	1668	1030
-40	84	843	791	831	18641	43040	167835	336	813	1135
-30	88	882	842	872	10961	24651	88342	177	415	1247
-20	92	922	893	913	6662	14615	48487	97	221	1367
-10	96	961	946	956	4175	8947	27649	55	122	1495
0	100	1000	1000	1000	2961	5642	16325	33	70	1630
10	104	1039	1056	1045	1781	3657	9952	20	42	1772
20	108	1078	1112	1091	1205	2431	6247	12	25	1922
25	110	1097	1141	1114	1000	2000	5000	10	20	2000
30	112	1117	1171	1138	834	1655	4028	8	16	2080
40	116	1155	1230	1186	589	1151	2662	5	10	2245
50	119	1194	1291	1235	424	816	1800	4	7	2417
60	123	1232	1353	1285	310	590	1244	2	5	2597
70	127	1270	1417	1337	231	434	876	2	3	2785
80	131	1309	1483	1390	175	324	628	1	2	2980
90	135	1347	1549	1444	134	246	458	1	2	3162
100	139	1385	1618	1500	104	189	339	1	1	3392
110	142	1422	1688	1557	81	147	255	1	1	3607
120	146	1461	1760	1615	65	116	194	0	1	3817
130	150	1498	1883	1675	52		150	0	0	4008
140	154	1536	1909	1736	42		117	0	0	4166
150	157	1573	1987	1799	34		92	0	0	4280

Tableau 1: Comparaison des sondes de température[4]

« Le choix de la méthode de connexion et du type d'interface électronique dépend de la précision de mesure recherchée pour une installation donnée. Une sonde de température peut être utilisée selon 3 modes de connexion : 2 fils, 3 fils, ou 4 fils. »[5]

Pour les sondes supérieures à 1 kΩ, la méthode à 2 fils est la plus utilisée car la résistance du câble influence très peu la résistance de la sonde. De plus amples informations sont disponibles en annexe 1a.

[4] Source : http://lampopumput.info/foorumi/index.php?topic=18236.15
[5] Source : http://www.technetea.com/PT100.html

2.4.2 Régulateur

Le régulateur est l'intelligence du système de chauffage. Il active la charge (PAC, boiler, radiateur électrique, chauffage à mazout, …) lorsqu'il fait froid et la désactive lorsqu'il fait chaud.

La mesure de la valeur de la sonde se fait en appliquant une tension continue connue en direction de la sonde, puis en lisant le courant qui y revient. La valeur de la résistance est alors connue en appliquant la loi d'ohm.

$$Loi\ d'ohm : U = R * I \qquad => \qquad R = \frac{U}{I}$$

Il existe deux types de régulateurs fonctionnant de deux manières différentes :

1) Régulateur tout ou rien, ON/OFF avec hystérèse
2) Régulateur inverter, élève gentiment la puissance lors de changement de température

2.4.3 Cartes d'automates : entrées analogiques

La mesure de la sonde de température extérieure est lue par un automate qui doit être équipé d'une carte d'entrée analogique de type RTD[6]. Cette carte doit évidemment être compatible avec le type de sonde utilisé.

Pour les deux automates les plus répandus en Suisse, Siemens et Wago, les différentes cartes de lecture de résistance sont fournies en annexe 1b.

[6] Resistor Temperature Detector : résistance qui varie en function de la température

2.5 Solutions proposées

2.5.1 Première possibilité : source de courant

Le régulateur applique une tension continue connue en direction de la sonde ; il lit le courant qui y revient. En modifiant ce courant, le régulateur est « trompé ».
Il faut bien évidemment avoir le choix de la valeur du courant qui sera fournie au régulateur, afin de simuler des valeurs de résistances différentes (explication donnée au chapitre §2.4.2). Une source de courant est donc nécessaire.

Une carte de sorties analogiques est alors indispensable. Une tension sort de cette carte (0-10V). Elle correspond à la valeur de « fausse » température désirée. Cette tension (nommée « e » sur le schéma 1) est alors convertie en courant « i » avec des amplificateurs opérationnels.

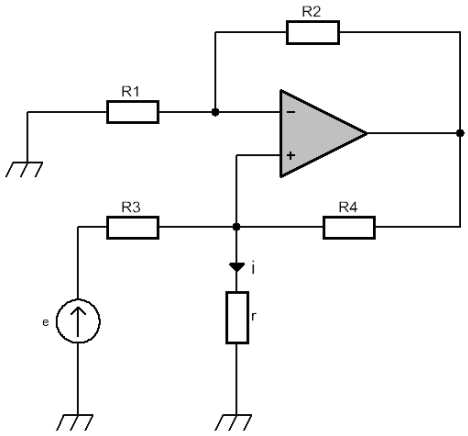


Schéma 1: Convertisseur tension-courant[7]

La charge « r » (régulateur) est toujours parcourue par le même courant « i ». $Loi\ d'ohm : \ i = \frac{e}{R3}$

Le problème est le suivant : pour chaque changement de température souhaité, il faut que la résistance R3 change de valeur pour que le régulateur reçoive un autre courant.

2.5.2 Deuxième possibilité : potentiomètres électroniques

Un ou plusieurs potentiomètres électroniques configurables à l'aide d'impulsions sont utilisés. De cette manière, il est possible de recréer le fonctionnement identique d'une sonde de température. En cas de coupure de courant, le potentiomètre utilise sa mémoire non-volatile. Cette dernière stocke la valeur de la résistance avant que la panne intervienne.

Le seul désavantage de cette possibilité est le suivant : le courant traversant le potentiomètre est limité à une certaine valeur.

[7] Source: http://electronique.aop.free.fr/AOP_lineaire_NF/9_convertisseurTC.html

2.6 Solution adoptée

L'utilisation de potentiomètres électroniques se profile comme la solution la plus adéquate, car la flexibilité est plus grande que d'utiliser des amplificateurs opérationnels. Les potentiomètres utilisés sont les MAX5483EUD+ et MAX5484EUD+ de Maxim Integrated. Leurs valeurs maximales sont de 10 kΩ pour le premier et 50 kΩ pour le second.

Le but est de copier le fonctionnement de plusieurs types de sondes. Des potentiomètres différents sont donc utilisés pour avoir de nombreuses plages de résistances.

Cependant, il faut faire attention au courant maximum que peut supporter le potentiomètre (voir tableau 2).

	I_max sur la borne mobile W	**R_min interne**
Potentiomètre 10 kΩ	5 mA	70 Ω
Potentiomètre 50 kΩ	1 mA	110 Ω

Tableau 2: Courant maximum des différents potentiomètres

Une résistance faisant office de chute de tension est placée devant le potentiomètre afin de limiter le courant en dessous de la valeur critique admissible.

La résistance est choisie comme suit pour une tension de 5V du régulateur, avec le potentiomètre de 10 kΩ :

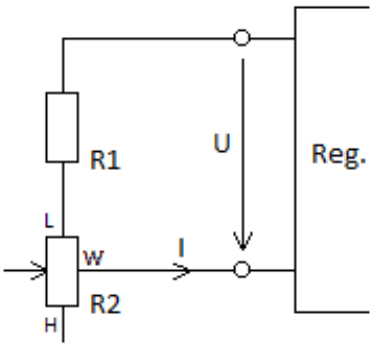


Schéma 2: Choix de la résistance de limitation de courant

$$Loi\ d'ohm: R1 + R2_{min} = \frac{U}{I_{max}}$$

En modifiant l'équation ci-dessus, $R1 = \frac{U}{I_{max}} - R2_{min} = \frac{5}{5*10^{-3}} - 70 = 930\ \Omega$

La valeur de la résistance R1, selon la table de normalisation de résistances E24 et en prenant la valeur supérieure la plus proche, doit alors être de 1 kΩ.

Le système qui sera démontré dans le paragraphe §2.7 fonctionne pour une tension de régulateur inférieure à 12 VDC (de manière à garder le courant en dessous de la limite de courant).

2.7 Carte électronique

2.7.1 Composants

Une carte électronique est imaginée et dimensionnée avec la plupart des composants de la carte électronique qu'avait réalisé M. Catteeuw (annexe 2a), à l'exception des composants suivants :

- Relais Axicom V23105A5505A201
  - Taille plus petite que le relais Panasonic précédemment choisi afin de ne pas surdimensionner la plaque électronique.
- Diode 1N4148
  - Pour faire effet de diode de roue libre lors de l'ouverture brusque des relais.
- Potentiomètre MAX5484EUD+
  - Pour avoir de hautes plages de résistances, allant jusqu'à 50 kΩ par potentiomètre.
- Bouton Knitter MFP-120
  - Pour choisir la configuration des potentiomètres : série ou parallèle, selon la configuration utilisée.
- Jumpers
  - Pour choisir quel sera le type de potentiomètre utilisé : 10 kΩ ou 50 kΩ.
- Résistances et condensateurs
  - Type SMD pour une question de place surtout que leur puissance dissipée est minime (<125 mW).
- Points de mesures

Le schéma électrique est fourni en annexe 2b.

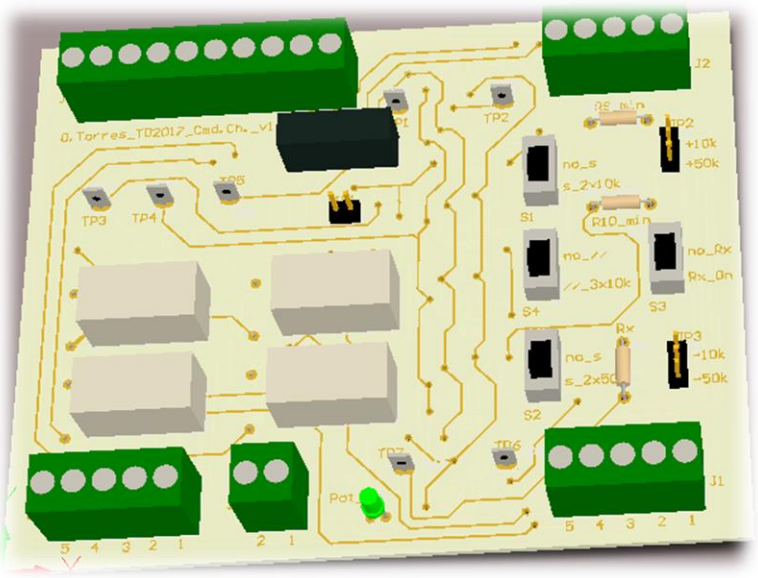


Figure 6 : Plaque électronique PCB : Vue de dessus

2.7.2 Potentiomètres

Sur cette plaque électronique, 4 potentiomètres de 10 kΩ (dont un qui est utilisé pour la configuration série et parallèle) et 2 potentiomètres de 50 kΩ sont utilisés de la manière suivante :

- 1 potentiomètre 10 kΩ
- 2 potentiomètres 10 kΩ en série
- 3 potentiomètres 10 kΩ en parallèle
- 1 potentiomètre 50 kΩ
- 2 potentiomètres 50 kΩ en série

Avec les configurations ci-dessus, il en ressort que la plaque électronique est flexible et offre les gammes de résistances suivantes pour une tension de régulateur de 5 VDC :

- Choix d'une résistance fixe de valeur désirée
- Plage de résistance 320 Ω … 3.7 kΩ
    - Sondes PT500, PT1000, NTC 1kΩ, Ni1000 et Ni1000 TK5000
- Plage de résistance 1 kΩ … 11 kΩ
- Plage de résistance 1 kΩ … 21 kΩ
    - Sondes NTC 2kΩ
- Plage de résistance 5.7 kΩ … 55.7 kΩ
- Plage de résistance 5.7 kΩ … 105.7 kΩ
    - Sondes NTC 10kΩ et NTC 20kΩ

Selon l'annexe 2b, la borne J3_8 est utilisée pour commuter deux relais K3 et K4 qui ont la fonction suivante :

- Relais désactivés : lecture de la sonde de température
- Relais activés : lecture de la valeur des potentiomètres électroniques

En lisant la valeur des potentiomètres avant de les appliquer au régulateur, il est possible de diminuer de manière logicielle l'erreur que peuvent avoir ces potentiomètres, jusqu'à avoir une erreur maximale de 10 Ω pour le potentiomètre 10 kΩ ou 50 Ω pour celui de 50 kΩ. Cela permet d'ajouter quelques impulsions pour être le plus proche possible de la valeur de résistance que l'on désire fournir au régulateur.

A noter que le pas d'un potentiomètre 10 kΩ est de 10 Ω et le pas d'un potentiomètre 50 kΩ est de 50 Ω. Les potentiomètres ont une erreur de ± 25 %.

Les informations concernant les prescriptions d'utilisation et le fonctionnement approfondi de cette carte électronique sont détaillés dans l'annexe 3a.

Le coût des différents composants, de la plaque électronique PCB et du boitier de protection (voir paragraphe $2.8) s'élève à environ 60 CHF par carte électronique. La liste de matériel est disponible en annexe 3b.

2.7.3 Réflexion sur l'entrée `sonde de température extérieure` du régulateur

Dans le tableau ci-dessous, une comparaison est faite entre le fait de lisser ou non la tension que le régulateur applique sur le potentiomètre ainsi que le courant qui y retourne.

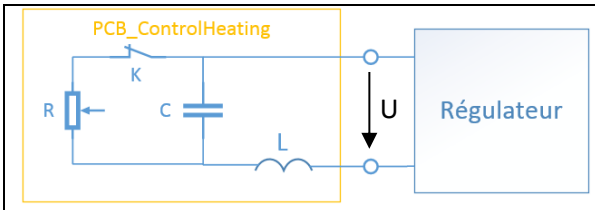
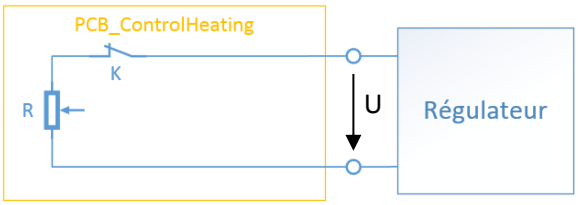
 *Schéma 3: Vue simplifiée - Potentiomètres connectés au régulateur avec filtre* La capacité est présente pour maintenir la tension continue que le régulateur fournit pour lire le courant qui y retourne. L'inductance est utilisée comme inductance de fuite, durant le court instant où le relais K est ouvert lors du changement de branchement sur l'entrée du régulateur, c'est-à-dire au moment où la sonde est débranchée puis le potentiomètre sera ensuite branché.	 *Schéma 4: Vue simplifiée - Potentiomètres connectés au régulateur sans filtre* Durant le court instant où le relais K est ouvert lors du changement de branchement sur l'entrée du régulateur, c'est-à-dire au moment où la sonde est débranchée puis le potentiomètre sera ensuite branché, le régulateur ne percevra aucun courant. Il en déduira qu'il faut augmenter la tension pour recevoir du courant en retour. Jusqu'à ce que la sonde soit connectée. Il faut donc minimiser la durée d'ouverture du relais K pour protéger les potentiomètres lorsque ce relais se refermera.
Avantages : - Pas de pics de tension - Pas de pics de courant	Avantage : - Simplicité
Inconvénient : - Difficulté du dimensionnement de L et de C de manière à être flexible sur un bon nombre de régulateurs	Inconvénient : - Possibles pics de tension et/ou courant

Tableau 3: Réflexion sur la tension du régulateur

Une déduction simple permet de clore cette réflexion. Du fait que c'est une régulation de chauffage, en supposant que ce soit un PID, le régulateur ne sera pas excessivement réactif. En effet, pour du chauffage cela ne sert à rien d'être trop réactif, car l'habitation a une certaine inertie thermique. Le choix de mettre une capacité et une inductance est donc écarté, de plus, il n'y aura pas de dimensionnement compliqué à réaliser.

2.8 Boitier de protection

La carte électronique est protégée par un boitier de protection. Il se fixe dans l'armoire électrique, non loin de l'automate programmable, sur un rail DIN. Le boitier est de marque Wago. Un cache de protection transparent sert à éviter que les composants soient touchés avec les mains lorsqu'il est en utilisation. Ce cache ne referme pas complètement le boitier pour le passage des câbles vers les bornes.

Les bornes supérieures servent à connecter les fils en direction de l'automate et les bornes inférieures pour la connexion en direction du régulateur de chauffage.

Du fait qu'il y a des ouvertures pour le passage des câbles, des insectes peuvent éventuellement pénétrer dans le boitier. C'est pour cela que les composants SMD sont soudés sur la partie inférieure de la plaque électronique.

Sur la partie supérieure du PCB, les pistes seront disposées avec un espace plus grand entre elles. Ainsi les insectes ne pourront pas se poser sur une petite distance entre deux potentiels différents et créer des courts-circuits involontaires.

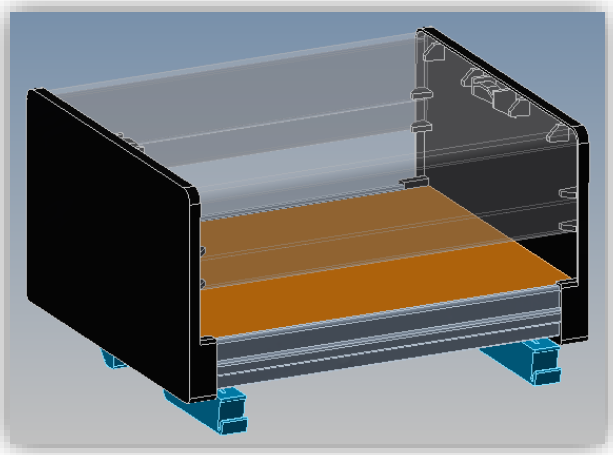


Figure 7: Boitier de protection - PCB

Ce boitier possède un avantage, son cache de protection transparent est facilement détachable car il s'insère dans les rainures des deux pièces dessinées en noir sur la figure 7. De ce fait, il suffit simplement d'enlever ce cache pour accéder au bornier de la plaque électronique.

2.9 Tests électriques

Des tests électriques de la plaque électronique sont effectués en suivant un protocole de tests. Ces derniers se déroulent de la manière suivante :

1) Sans alimentation
   a. Avant soudures : Tests à l'ohmmètre
   b. Après soudures : Tests à l'ohmmètre
2) Avec alimentation
   a. Tests au voltmètre

Ces tests ont permis de dénicher une erreur lors du routage de la carte électronique, mais aussi de remarquer, pour des raisons pratiques, qu'il fallait modifier le branchement de relais afin de ne pas fausser la valeur lue par l'automate (voir annexe 3c).

2.9.1 Modifications après tests

Les tests ont démontré que les relais K3 et K4 ont été mal branchés. En effet, en état de repos, il est plus logique que l'API lise les potentiomètres (schéma 5, chemin vert) et que la sonde de température soit connectée au régulateur (chemin orange).

Pour y remédier, la connexion 8 est inversée avec la connexion 4 et la connexion 9 avec la connexion 3 sur les deux relais entourés en rouge.

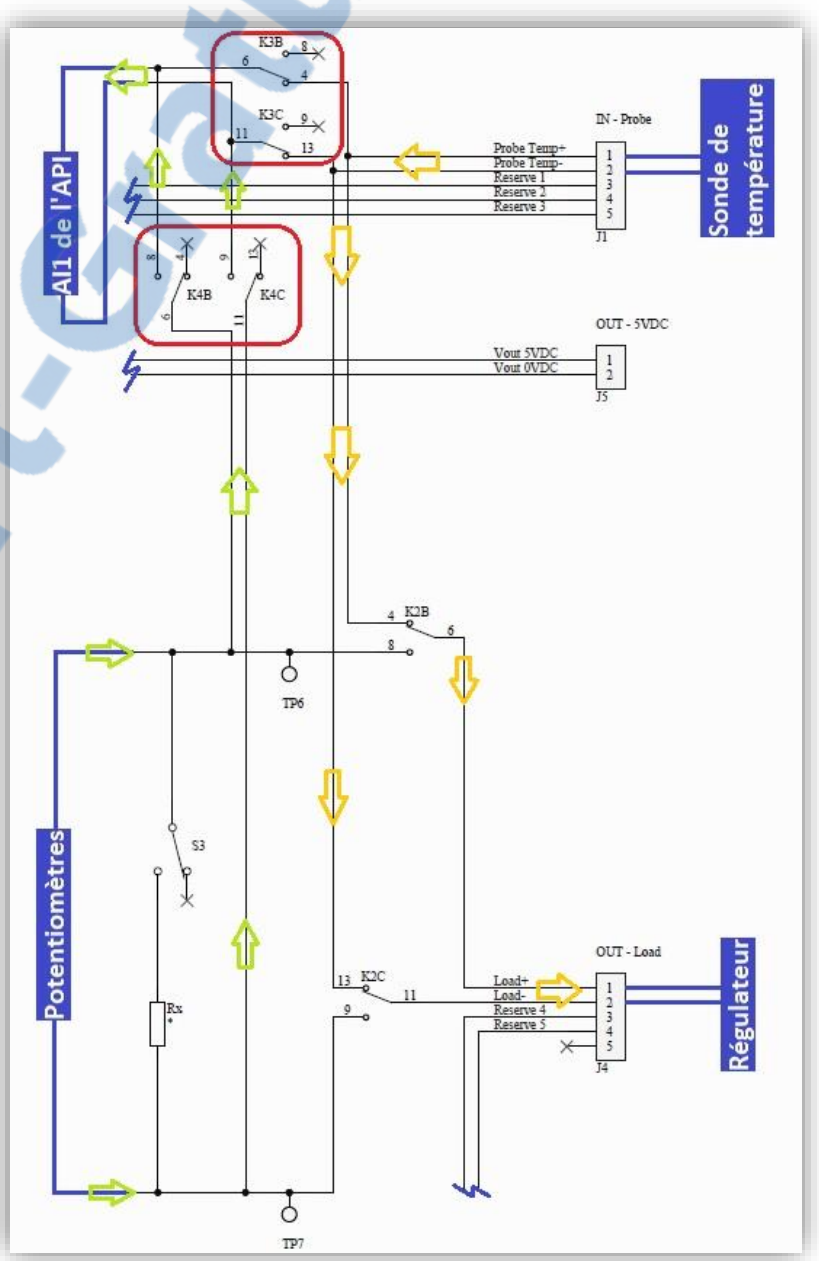


Schéma 5: Modifications - schéma de la plaque électronique PCB

2.9.2 Version low-cost

Après avoir créé une première version de ce PCB, l'idée de proposer une version low-cost de cette plaque électronique est apparue.

Au lieu d'avoir des potentiomètres électroniques, il y a simplement une résistance fixe « température faussée froide » et une autre « température faussée chaude ».

Pour ce faire, 3 relais sont utilisés et servent à :

- K1
  - Appliquer une grande température faussée
  - Appliquer une petite température faussée
- K2
  - Connecter la sonde au régulateur
  - Connecter l'une des deux résistances au régulateur
- K3
  - Fournir la valeur de la sonde de température à l'API
  - Ne rien fournir à l'API

Le schéma électronique de cette version low-cost est disponible en annexe 2c.

2.10 Configuration automate

L'API utilisé possède la configuration suivante :

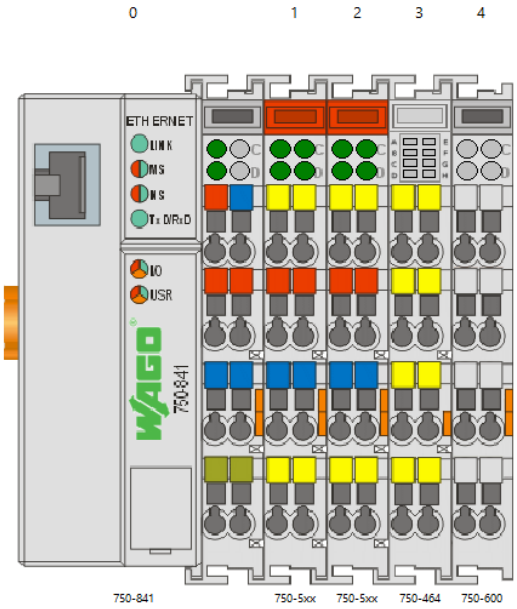


Figure 8: Configuration API - Pilotage de chauffages

Position 0 :	750-841	Contrôleur principal	
Position 1 :	750-531	Carte de 4 sorties numériques	DO 1 à DO 4
Position 2 :	750-531	Carte de 4 sorties numériques	DO 5 à DO 8
Position 3 :	750-464	Carte de 2 à 4 entrées analogiques RTD	AI 1 et AI 2
Position 4 :	750-600	Carte de terminaison de bus	

3 Interface de commande de chauffages – Partie software

La carte électronique PCB doit être pilotée pour, par exemple, commander des relais et avoir l'acquisition de la température extérieure (lecture de la sonde ou des potentiomètres). Un automate programmable est choisi de par sa simplicité de lecture des valeurs de résistances. En premier lieu, cette programmation est faite sur un automate de marque Wago (disponible en annexe 4). Le programme API est codé avec le logiciel Codesys, un logiciel développé par Wago.

Les langages de programmation comprennent les standards CEI 61161-3 :

- LD Langage ladder (schéma à relais)
- SFC Langage séquentiel, proche du Grafcet
- FBD Blocs de fonctions, divisions en réseau
- ST Texte structuré
- IL Liste d'instructions, pseudo assembleur

Langage supplémentaire Wago :

- CFC Blocs de fonctions pouvant être placés librement à l'écran, pas de division en réseau

Pour le présent travail, l'API contient des entrées et sorties de nature différente.

- AI Entrées analogiques *(Analog inputs)* type INT
- DO Sorties digitales *(Digital outputs)* type BOOL

3.1 Utilisation pour d'autres types de sondes ou régulateurs

Une sonde de température extérieure de type PT1000 est utilisée dans le programme créé et expliqué dans l'intégralité de ce chapitre §3. L'annexe 4a présente sous forme de tutoriel comment procéder pour changer des variables ou du code écrit dans les blocs de fonctions. En effet, tous les régulateurs n'utilisent pas les mêmes sondes et ne fournissent pas la même tension (expliqué précédemment dans le chapitre §2.6) afin de lire la valeur des sondes qui y sont branchées.

3.2 Bloc de fonction principal

Le langage de programmation automate choisi pour coder ce bloc est le ST car il est plus flexible et mieux ordonné pour faire plusieurs tâches à la fois.

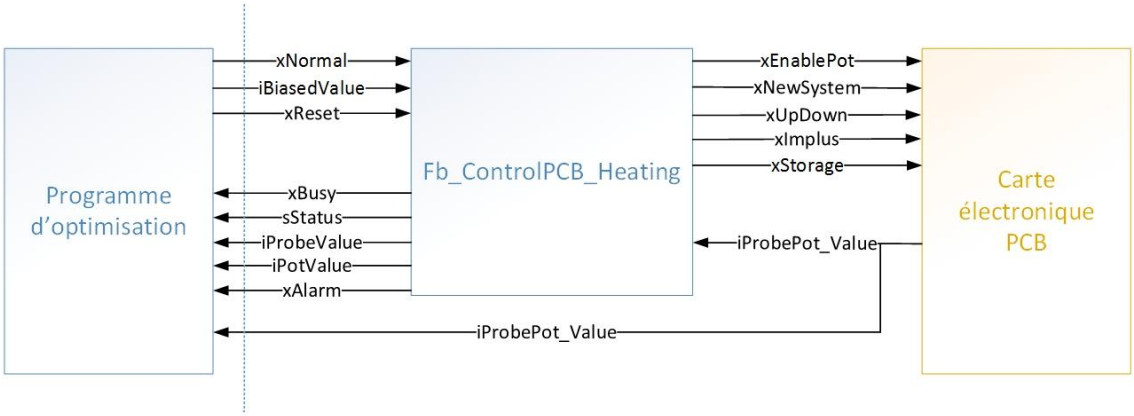


Figure 9: Entrées/sorties du bloc de fonction Fb_ControlPCB_Heating

Sur la figure 9 ci-dessus, la partie software est représentée en bleu, tandis que la partie hardware est représentée en jaune. La ligne en traits-tillés indique la frontière de ce travail de diplôme. Seulement ce qui se situe à droite de cette ligne est effectué.

La petite lettre minuscule au début de chaque variable représente le type de variable :

INT: i BOOL: x STRING: s

Le programme d'optimisation de chauffage n'est pas effectué. Il devra fonctionner de la sorte :

xNormal = TRUE	Sonde de température appliquée au régulateur, lecture de la valeur des potentiomètres
xNormal = FALSE	Potentiomètres appliqués au régulateur, lecture de la valeur de la sonde de température
iBiasedValue	Valeur de consigne pour les potentiomètres
Impulsion de *xReset*	Redémarrage du système. Pendant ce temps, la sonde est appliquée au régulateur

Tableau 4: Entrées du bloc de fonction en provenance du programme d'optimisation

Lors de l'initialisation du bloc de fonction, la sonde de température est appliquée au régulateur de chauffage pendant que le potentiomètre est appliqué à l'entrée de l'automate, selon le schéma 6 ci-dessous.

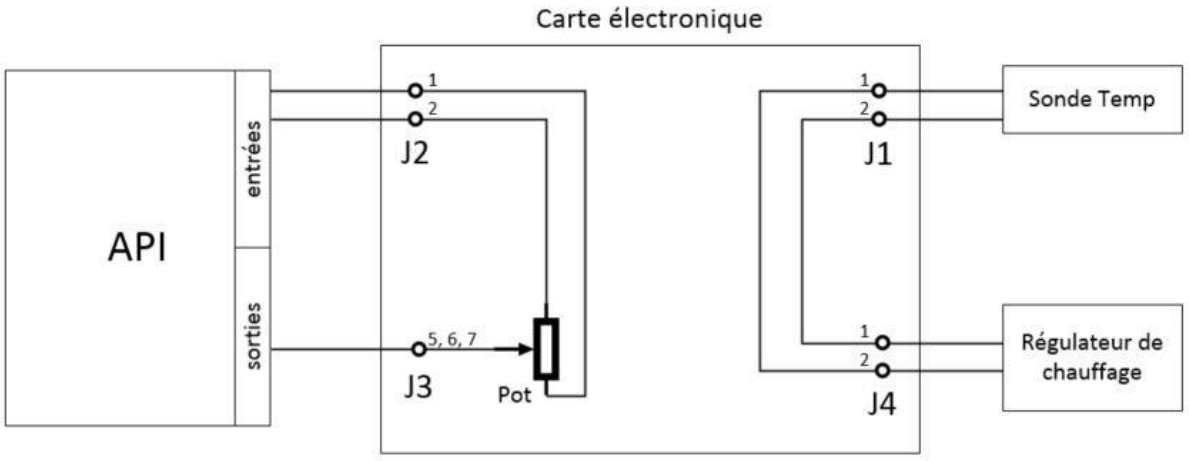


Schéma 6: Configuration de pilotage de chauffage par sonde de température

Une fois le potentiomètre initialisé et s'il a la même valeur que l'entrée *iBiasedValue*, le changement est effectué. Le potentiomètre est appliqué au régulateur de chauffage pendant que le la sonde de température est appliquée à l'entrée de l'automate, selon le schéma 7 ci-dessous.

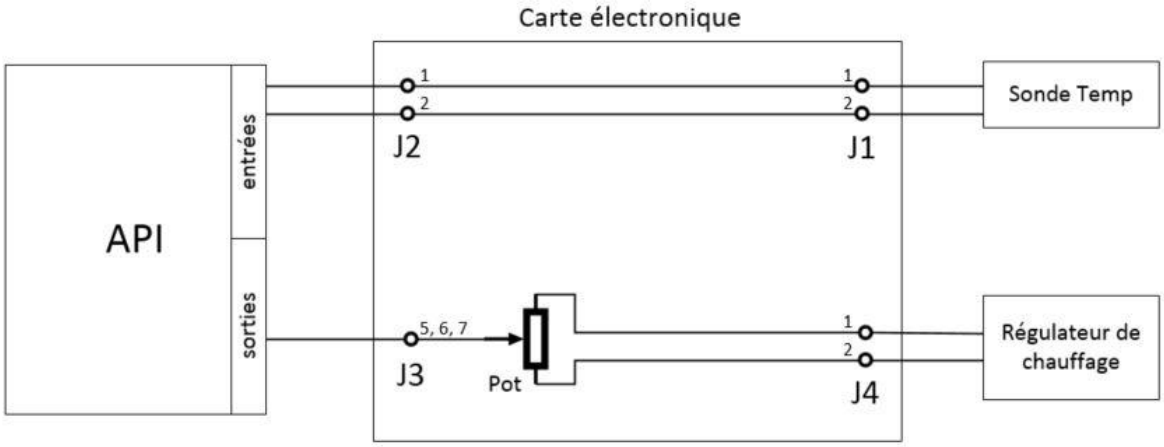


Schéma 7: Configuration de pilotage de chauffage par potentiomètres électroniques

Le bloc de fonction *FB_ControlPCB_Heating* est principalement une machine d'état. Il y a quelques lignes de code avant et après la machine d'état. Le bloc de fonction travaille cycliquement avec un temps de 15ms. C'est-à-dire qu'il se passe 15ms entre un état et un autre pour effectuer le reste du code en dehors de la machine d'état. Entre chaque état, du code est effectué comme ci-après :

Code début -> Machine d'état : Etat X -> Code fin

Les instructions qui doivent être effectuées en tout temps se déroulent avant ou après la machine d'état. Lors d'un changement de sonde pour un autre type de chauffage, il faut bien entendu modifier des constantes dans ce bloc de fonction.

3.3 Visualisation

Une visualisation est conçue afin de simuler les actions que le programme d'optimisation doit fournir (Figure 10, entourée en bleu).

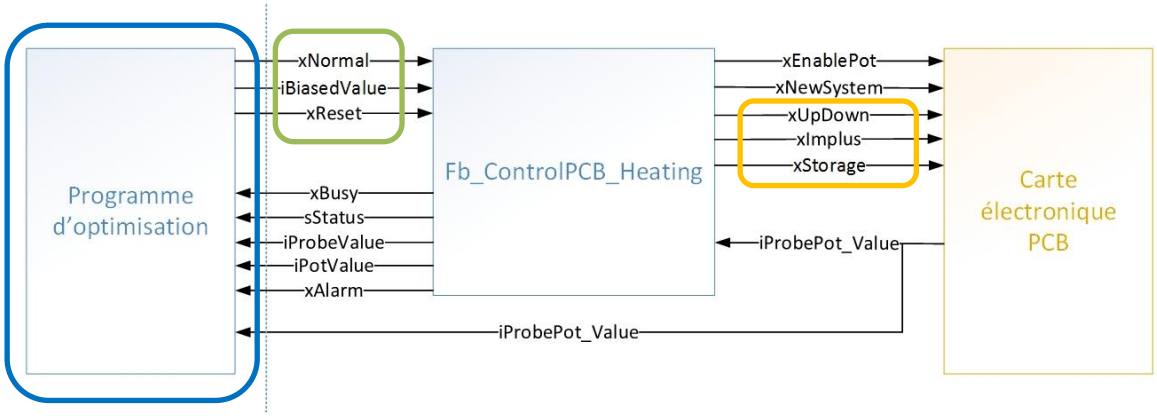


Figure 10: Entrées/sorties du bloc de fonction Fb_ControlPCB_Heating, utilisés pour la visualisation

Ce programme d'optimisation (non effectué durant ce travail de bachelor) doit récupérer les prévisions météo pour pouvoir prévoir un changement climatique brusque et, par conséquent, piloter le système de chauffage pour contrer cet effet qui pourrait prétériter le confort des habitants.

En périodes d'entre-saison, juste avant que le soleil se couche, le capteur de température extérieur peut percevoir une température extérieure largement supérieure à la température réelle si la sonde reçoit des rayons de soleil. Dans ce cas, il faudrait fausser cette température afin de démarrer le système de chauffage en prévoyance d'un coup de froid.

iPotValue est la valeur des potentiomètres qui sont appliqués au régulateur de chauffage. Pour changer la résistance des potentiomètres, les sorties *xUpDown*, *xImpuls* et *xStorage* sont utilisées (entourées en orange).

3.3.1 Fausser la température

Afin d'utiliser les potentiomètres pour piloter le régulateur de chauffage, il faut que le programme d'optimisation fournisse les conditions suivantes :

- *xNormal* doit être à FALSE, c'est-à-dire en couleur grise
- Attribuer une valeur à *iBiasedValue*, tant qu'elle se situe au-dessus de la limite inférieure (ici : -30°C) et en dessous de la limite supérieure (ici : 50°C)
- *xReset* doit être à FALSE, c'est-à-dire en couleur grise
- Aucune alarme ne doit être signalée

Vu que ce programme d'optimisation n'est pas effectué, l'utilisateur peut le simuler en modifiant les valeurs d'entrées du bloc sur la visualisation (entouré en vert sur la figure 10 vue au chapitre §3.3).

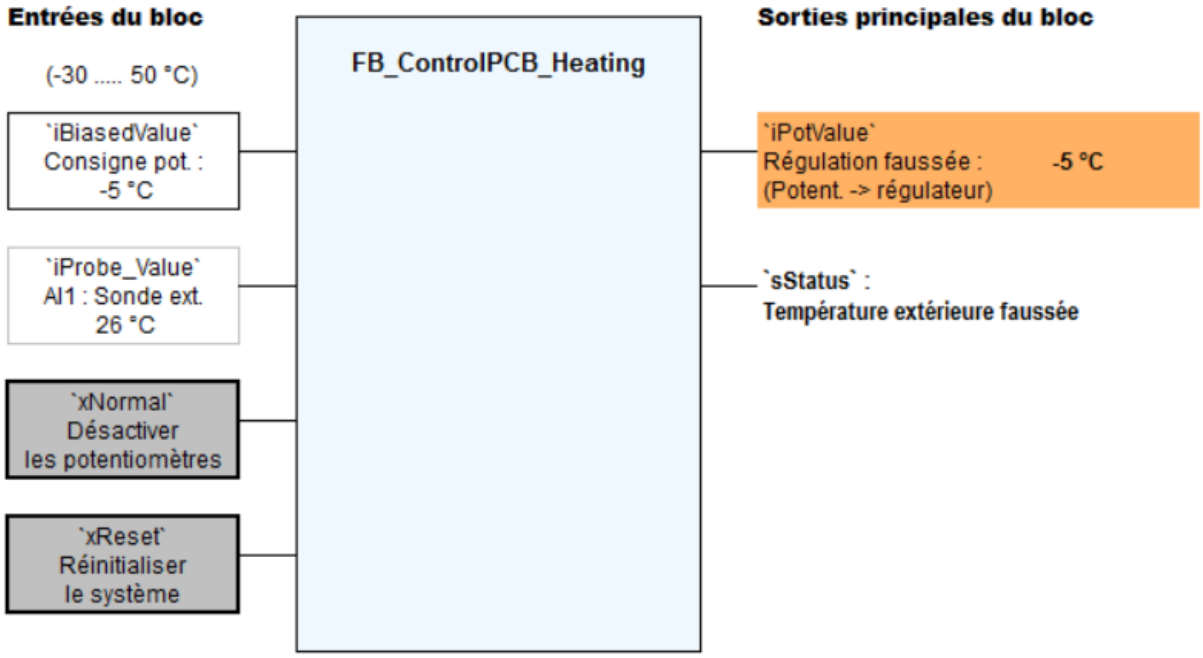


Figure 11: Visualisation - Fausser la température à une valeur de -5°C, appliquer les potentiomètres au régulateur de chauffage

3.3.2 Utiliser la sonde de température extérieure pour piloter le régulateur

Afin d'utiliser la sonde de température extérieure pour piloter le régulateur de chauffage, il faut que les conditions suivantes soient remplies :

- La variable *xNormal* doit être à TRUE, c'est-à-dire en couleur verte
- Peu importe la valeur assignée à *iBiasedValue*, tant qu'elle se situe au-dessus de la limite inférieure (ici : -30°C) et en dessous de la limite supérieure (ici : 50°C)
- La variable *xReset* doit être à FALSE, c'est-à-dire en couleur grise

Lors du signalement d'une alarme pour cause de défaut, la sonde est automatiquement appliquée aux bornes du régulateur tant que cette alarme n'est pas supprimée et quittancée, même si les conditions ci-dessus ne sont pas forcément remplies.

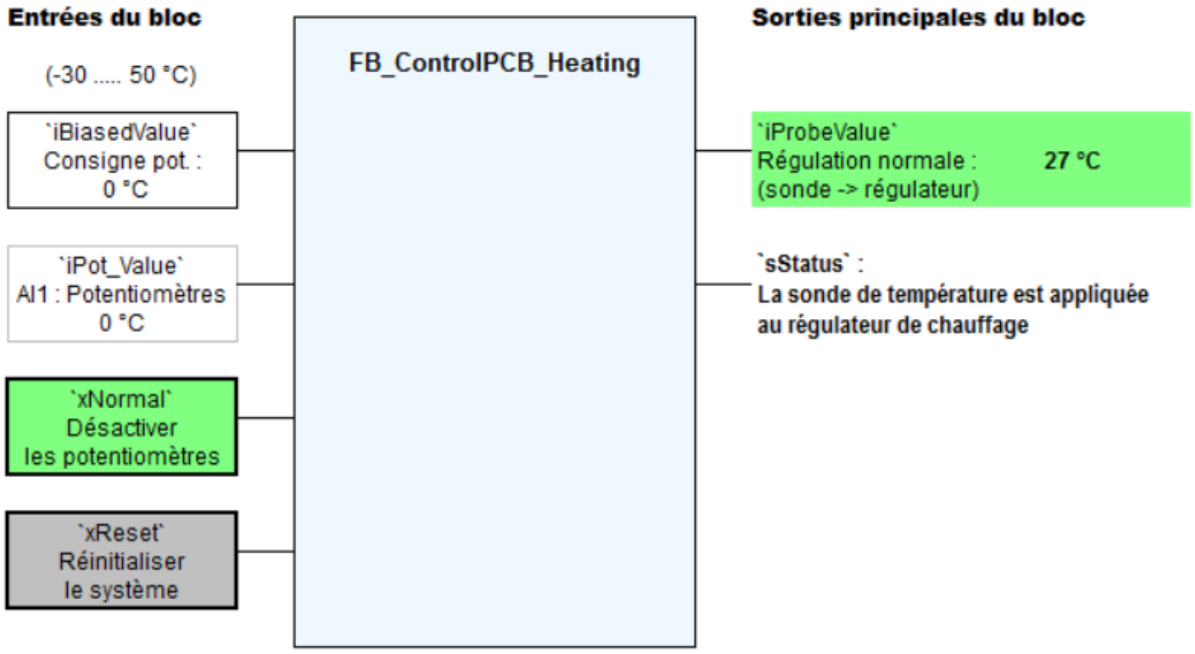


Figure 12: Visualisation - Appliquer la sonde de température extérieure au régulateur de chauffage

3.4 Tests effectués

3.4.1 Bloc de fonction sur la durée

Pour valider le bon fonctionnement du bloc de fonction *Fb_ControlPCB_Heating*, le programme *Test_the_code* est réalisé (disponible en annexe 4d). Il effectue différentes combinaisons de valeurs des variables provenant du programme d'optimisation (*iBiasedValue*, *xNormal* et *xReset*).

Ces combinaisons varient dans un temps progressif. C'est-à-dire qu'au départ les variables ne restent que quelques minutes à la même valeur, puis au fil du temps, changent de moins en moins rapidement. Ainsi, le bloc de fonction est testé avec différentes constantes de temps, comme le ferait le programme d'optimisation.

Ce test est effectué durant 3 jours et n'a activé aucune alarme.

3.4.2 Alarmes

Les alarmes prévues lors de défauts ou de valeurs de variables non attendues sont aussi testées et simulées.

Ces alarmes sont les suivantes :

- Alarme 1 Consigne en dehors des limites durant un temps trop long
- Alarme 2 Fil de la sonde de température extérieure coupé
- Alarme 3 Trop de temps pris pour fausser la température extérieure
- Alarme 4 Trop de temps pris pour relier la sonde de température au régulateur

3.5 Perspectives

3.5.1 Tests sur un système de chauffage

Des tests sur un système de chauffage fonctionnant avec régulateur et sonde extérieure peuvent être réalisés. De cette manière l'interface de mesure/commande de chauffages pourra être validée. Des paramètres sur la programmation doivent évidemment être modifiés selon le type de sonde et de régulateur. Ces informations se trouvent en annexe 4e.

3.5.2 Programme d'optimisation

Un algorithme d'optimisation de chauffage peut ensuite être écrit. Il choisira, en utilisant des prévisions météorologiques et un modèle thermique de l'habitation, à quel moment il faudra fausser la température extérieure.

3.5.3 Bloc de fonction pour autres type d'automates

Le bloc de fonction de pilotage de la carte électronique PCB peut être développé pour d'autres types d'automates tels que Siemens, Beckhoff ou Schneider Electric. Il suffit de recopier le contenu et les variables du bloc de fonction pour l'adapter au logiciel de programmation de ces API.

Ces différentes marques d'API ont un point en commun, elles utilisent les mêmes langages de programmation (les standards CEI 61161-3).

4 Serveur d'alarme Wago – Etude des possibilités

4.1 Introduction

Cette partie du travail de diplôme est mandatée par l'entreprise **Atelier R2D2 sàrl** à Anzère.

L'Atelier R2D2 propose principalement des solutions domotiques à ses clients, dont un service d'alarme lors d'infraction du domicile ou de détection de fumée en cas d'incendie. Cette entreprise s'occupe aussi de la maintenance de la centrale de chauffage à distance d'Anzère (CBA). Lors d'un défaut chez un client, dans cette centrale de chauffage ou dans l'une de ses sous-stations, des alarmes sont envoyées par e-mail.

4.2 Problématique

Actuellement, les envois d'alarmes fonctionnent selon la figure 13 ci-dessous :

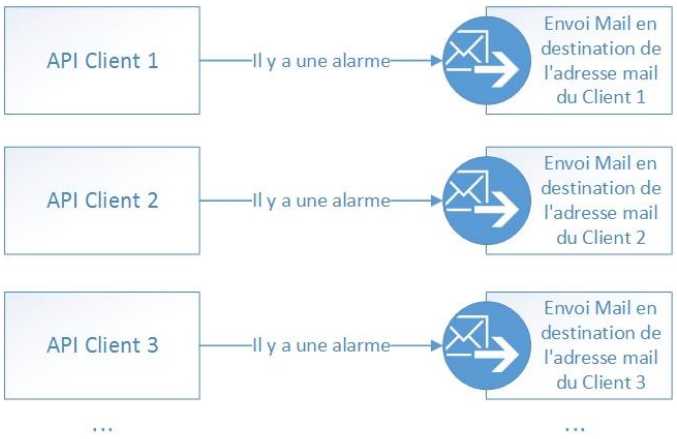


Figure 13: Envois d'alarmes par mail, état actuel

Lorsque l'algorithme présent dans l'automate programmable du client observe un défaut majeur, il le signale en envoyant une alarme par mail au client.

Quelques clients (mais aussi l'Atelier R2D2 pour la maintenance de la CBA) aimeraient aussi recevoir un sms lorsqu'une alarme apparaît. Actuellement, pour proposer ce service, deux possibilités se distinguent :

1) Installer un module (ToPass) Wago dans chaque habitation. Ceci est un service onéreux ; de plus, cette technologie devient gentiment obsolète.
2) Utiliser un service, tel que IFTTT[8], permet d'envoyer des SMS lors de la réception d'un mail. Il contient les désavantages suivants :
    a. Si ce service n'est plus gratuit ou n'existe plus, il faudra en trouver un autre et l'installer chez chaque client.
    b. Il y a une limite du nombre d'envoi de SMS par mois. Si cette limite est atteinte, le client ne recevra plus de téléalarme.

[8] Davantage d'informations sont disponibles sur : https://ifttt.com/applets/1110p-send-an-sms-when-a-new-gmail-is-from-a-specific-email-address

4.3 Cahier des charges

La proposition de l'Atelier R2D2 est de créer un serveur d'alarme sur un automate Wago de nouvelle génération (PFC200). Il sera installé dans les bureaux de l'entreprise à Anzère. Le module principal de cet automate contient un emplacement de carte SIM pour permettre l'envoi et la réception de SMS.

Une base de données contenant les numéros de téléphones portables de tous les clients doit être créée. Lors de la réception d'une alarme de la part d'un automate client, l'automate « serveur d'alarme » devra chercher le(s) numéro(s) de téléphone portable pour transmettre cette alarme par SMS au client.

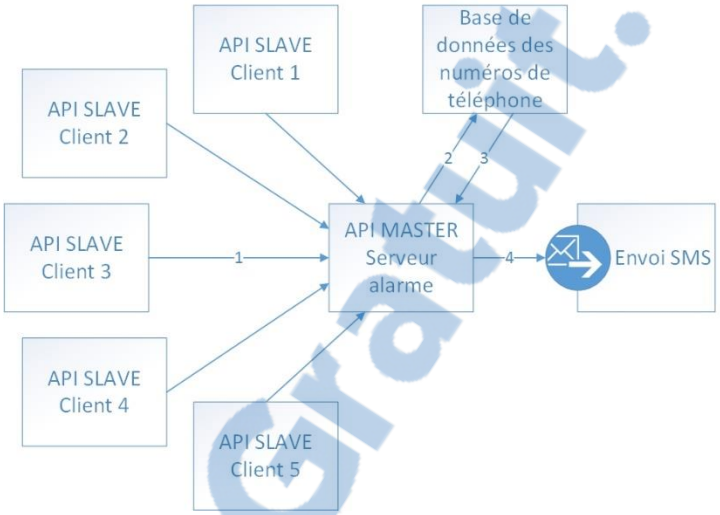


Figure 14: Centralisation des alarmes et envois de SMS[9]

Un exemple du système désiré est démontré sur la figure 14. Les numéros ci-après correspondent aux numéros des flèches.

1. L'API du client 3 (SLAVE) perçoit une quelconque alarme. Il envoie l'information à l'API serveur (MASTER) via e-mail.
2. L'API MASTER fait une requête à la base de données pour obtenir les numéros de téléphones du client 3.
3. La base de données renvoie les numéros de téléphone correspondants
4. L'API MASTER envoie un ou plusieurs SMS d'alarme aux différents numéros de téléphone

Cette base de données devra être accessible au moyen d'une visualisation qui aura les fonctionnalités suivantes :

- Ajout/suppression/modification de client et de numéros de téléphone mobile
- La recherche d'un client dans la base de données pour pouvoir afficher ses attributs (numéros de téléphone mobile)
- Enregistrement cyclique de cette base de données

[9] Remarque : Dorénavant et pour la suite de ce présent rapport, la dénomination MASTER sera utilisée pour nommer l'API Serveur d'alarme et la dénomination SLAVE pour un des API Client.

4.4 Solutions proposées

4.4.1 Première solution : variables API

L'enregistrement de la base de données se fait avec des variables et des tableaux de variables.

Certaines variables doivent être déclarées avec les attributs « *retain persistant* » de manière à garder leurs valeurs lors d'une éventuelle coupure de courant et lorsqu'une nouvelle version du programme est envoyée sur l'API.

Cette méthode possède un désavantage. Si l'automate ne fonctionne plus, toute la base de données est perdue.

4.4.2 Deuxième solution : carte mémoire

L'enregistrement sous forme de fichier « .csv » est enregistré dans une carte mémoire. Les différentes colonnes de la base de données seront séparées par des caractères spéciaux (par exemple par un point-virgule « ; »).

Cette solution a le même désavantage que la première : si la carte mémoire ne fonctionne plus, la base de données est perdue.

4.4.3 Troisième solution : serveur SQL

L'enregistrement sous forme de base de données SQL[10] est stocké sur un serveur. La modification ou la lecture de cette base de données s'effectue par envoi et réception de requêtes SQL.

Dans un premier temps, le serveur fonctionne sur un ordinateur dans le bureau de l'Atelier R2D2. Ensuite, il serait envisageable de le faire « tourner » dans un NAS[11] relié au réseau internet local (LAN).

4.4.4 Comparaison des solutions

	Avantages	**Inconvénients**
Première solution	- Simplicité : pas d'élément externe à la programmation API	- Programmation passablement compliquée - Si l'automate ne fonctionne plus : la base de données est perdue
Deuxième solution	- Enregistrement sur carte mémoire SD - Possibilité de faire des copies des fichiers sur un disque dur	- Il faut retirer la carte SD de l'automate pour faire une copie sur un disque dur d'ordinateur par exemple - Si l'automate ne fonctionne plus : ce qui est fait après la dernière copie est perdu
Troisième solution	- Questionnement de la base de données par requêtes - Enregistrement de la base de données directement sur disque dur - Exportations de la base de données possible	- Si le serveur est éteint cela ne fonctionne pas

Tableau 5: Comparaison des solutions pour la création d'une base de données

Le tableau ci-dessus permet de comparer ces trois solutions et de choisir la meilleure d'entre elles. Le choix se porte donc sur la base de données SQL, car c'est la solution ayant le plus d'avantages, le moins d'inconvénients et permet une communication facilitée entre les API SLAVE et MASTER. De plus, dans le paragraphe §4.6 ci-dessous, les logiciels utilisés sont gratuits et open source.

[10] Structured Query Language est un langage informatique servant à exploiter des bases de données avec des requêtes structurées

[11] Network Attached Storage est un serveur de stockage en réseau

4.5 Réception des alarmes des automates clients

Dans le cahier des charges, il est prévu de recevoir les alarmes via un e-mail envoyé depuis l'automate du client. Après recherches, il s'est avéré impossible de recevoir et déchiffrer le contenu d'un e-mail avec un API Wago. Il a donc fallu trouver un autre système pour pouvoir échanger des informations entre automates distants.

L'envoi et la réception de données s'est donc fait par le biais d'une base de données SQL (Plus précisément MySQL). Elle se trouve dans un PC dans le bureau de l'Atelier R2D2. Elle pourra être atteignable par dynDNS (serveur web) ou VPN (réseau privé virtuel).

De manière à centraliser les informations, utiliser une base de données SQL pour l'enregistrement des numéros de téléphone des clients paraît comme la solution la plus appropriée. De plus, en termes de sécurité, la troisième solution (voir chapitre §4.4.3) semble la plus convenable.

4.6 Solution adoptée

4.6.1 Application PhpMyAdmin

« PhpMyAdmin est une application web de gestion pour les systèmes de gestion de base de données MySQL. C'est l'une des plus célèbres interfaces pour gérer une base de données MySQL sur un serveur PHP. »[12]

Cette application est open source et gratuite.

Pour faire fonctionner cette application, il faut que l'ordinateur utilise un logiciel contenant :

- Apache (serveur http)
- PHP
- MySQL

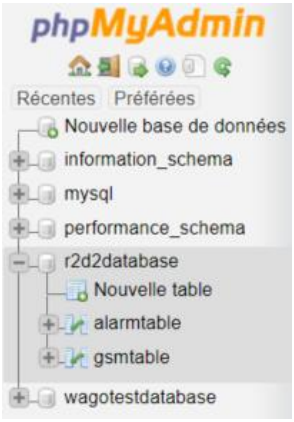
4.6.2 Logiciel UwAmp

Figure 15: Affichage du logiciel UwAmp

Le logiciel choisi (contenant Apache, PHP et MySQL) est UwAmp (gratuit), car il est facile d'utilisation. Il suffit de le lancer et tous les serveurs sont démarrés automatiquement. Ce logiciel utilise le port 80. Si une autre application utilise ce port, il faudra lui en assigner un autre.

[12] Source : https://fr.wikipedia.org/wiki/PhpMyAdmin

4.6.3 Base de données MySQL



La base de données MySQL se trouve sur le serveur PhpMyAdmin. Pour y accéder, il faut insérer la commande suivante dans le navigateur internet :

localhost/mysql ou 192.168.1.93/mysql[13]

Dans le cas présent, la base de données se nomme `*r2d2database*`. Elle contient deux tables de données. La première est la table de gestion des clients et de leurs numéros de téléphone : `*gsmtable*` et la seconde est la table des alarmes : `*alarmtable*`.

Figure 16: Base de données - hiérarchie

Le bureau de l'Atelier R2D2 est équipé d'un automate et d'un PC fixe. Le PC contient les bases de données MySQL et l'API gère les alarmes reçues depuis `*alarmtable*`. L'API fera des requêtes pour questionner ou modifier cette base de données.

Cela fonctionne selon le principe suivant pour un exemple d'une alarme provenant du client 3 :

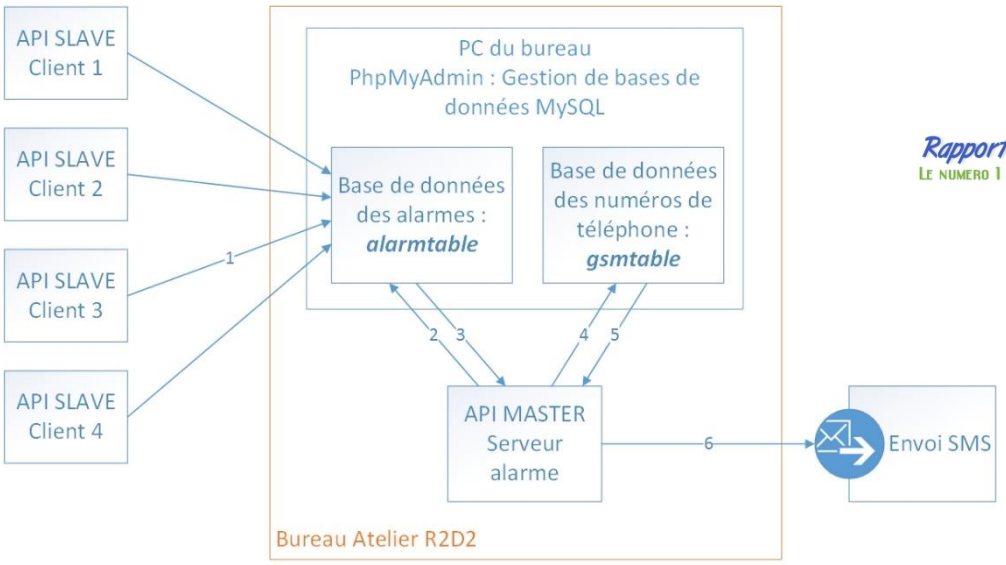


Figure 17: Produit proposé avec bases de données MySQL

1. L'API du client 3 (SLAVE) perçoit une quelconque alarme. Il prépare une requête MySQL et l'envoie en destination du serveur PhpMyAdmin présent dans le réseau local du bureau de l'Atelier R2D2. L'emplacement de l'alarme correspondant est changé à 1 et un message d'alarme apparaît (Colonnes `*AlarmIsActive*` et `*AlarmMessage*`, voir tableau 6).
2. L'API serveur d'alarme (MASTER) fait des requêtes chaque seconde en destination de la base de données des alarmes.
3. Le contenu d'`*alarmtable*` est envoyé en retour. Si une des cases `*AlarmIsActive*` est à 1, l'API va chercher de quel numéro de projet elle provient (Colonne `*ProjectNum*`, voir tableau 6).

[13] Dans le cas présent, 192.168.1.93 correspond l'adresse ip utilisée par l'ordinateur.

4. Une requête est envoyée vers la base de données `*gsmtable*` (voir tableau 7) pour connaître les numéros de téléphone relatifs au numéro de projet.
5. Les numéros de téléphone correspondants sont envoyés en retour.
6. La création d'un SMS ayant pour message le contenu de la case `*AlarmMessage*` est envoyé aux numéros de téléphone.

ProjectNum	AlarmIsActive	AlarmMessage	ResString1	ResString2
2017_1	0			
2016_1	0			
2016_2	1	Intrusion		

Tableau 6: Base de données - alarmtable

ProjectNum	ProjectName	Gsm1	Gsm2	Gsm3	Gsm4	Gsm5	ResString1	ResString2	ResString3
2017_1	NomProjet2017_1	2017111111							
2016_1	NomProjet2016_1	2016111111							
2016_2	NomProjet2016_2	2016222222	2016222216						

Tableau 7: Base de données – gsmtable

Des réserves sont prévues au cas où il faudrait, à l'avenir, faire passer d'autres informations par le biais de ces bases de données.

4.6.4 Requêtes MySQL

Des requêtes[14] sont nécessaires pour modifier la base de données. Ci-dessous, voici un exemple d'une requête pour insérer une ligne dans la table `*gsmtable*`.

```
INSERT INTO `gsmtable` (`ProjectNum`, `ProjectName`, `Gsm1`, `Gsm2`, `Gsm3`, `Gsm4`, `Gsm5`,
`ResString1`, `ResString2`, `ResString3`)
VALUES ('année numéro', 'Nom Prénom', '0790001234', '0780005678', '', '', '', '', '', '');
```

Figure 18 : Exemple de requête MySQL

Afin de ne pas insérer deux lignes avec le même numéro de projet, une sécurité a été mise en place. Chaque nouveau numéro de projet doit avoir une chaine de caractères différente des numéros de projet déjà présents dans la base de données. Le numéro de projet est le pointeur de la table de données. Ainsi lorsque l'API parcoure la table de données, il est certain de retourner les bons numéros de téléphone et non ceux d'un autre client (c'est-à-dire d'une autre ligne).

[14] De plus amples informations sur les requêtes MySQL sont disponibles sur le site web https://dev.mysql.com

4.7 Analyse des risques de panne

Les risques de panne du service d'alarme proposé par l'atelier R2D2 ne sont pas à négliger. C'est pourquoi une analyse de risque est effectuée. Pour une défaillance donnée, les causes peuvent être différentes. Chaque panne est évaluée dans un tableau selon les critères suivants :

- Fréquence de la panne
- Gravité
  - Les gravités sont à considérer en fonction de l'Atelier R2D2. Une panne d'API MASTER est très grave, tandis qu'une panne de l'API SLAVE est moins grave pour l'entreprise.
- Détection
  - Possibilité de prévoir la panne
- Remise en état
  - Facilité à remettre en état facilement et rapidement lorsque la panne survient
- Criticité
  - Multiplication des facteurs critères ci-dessus

Num.	Défaillance	Cause	Fréquence (1…5)	Gravité (1…5)	Détection (1…2)	Remise en état (1…5)	Criticité	Remarques
1a	Panne API MASTER	Câble débranché	1	5	1	1	**5**	Brancher le câble
1b	Panne API MASTER	Pas d'alimentation	2	5	2	2	**40**	Remise en état dépend du fournisseur d'électricité
1c	Panne API MASTER	API MASTER en erreur	1	5	2	1	**10**	Débrancher puis rebrancher API. Le programme dans l'EEPROM sera relancé.
2a	Câble Ethernet entre le PC et l'API MASTER	Câble coupé	1	5	2	1	**10**	Remplacer le câble
3a	Panne PC	Câble débranché	1	5	1	1	**5**	Brancher le câble
3b	Panne PC	Pas d'alimentation	2	5	2	2	**40**	Remise en état dépend du fournisseur d'électricité
3c	Panne PC	Logiciel uWamp arrêté	1	5	1	1	**5**	Relancer le logiciel uWamp
3d	Panne PC	Pare-feu bloque l'entrée depuis le SLAVE	1	5	1	1	**5**	Autoriser l'accès SLAVE -> réseau LAN R2D2
4a	Problème modem bureau R2D2	Pas d'internet (maintenance)	~~3~~ 1	5	2	~~2~~ 1	~~60~~ 10	Remise en état dépend du fournisseur d'internet Vérifier clé USB 4G
4b	Problème modem bureau R2D2	Pas d'alimentation	2	5	2	2	**40**	Remise en état dépend du fournisseur d'électricité
4c	Problème modem bureau R2D2	Pas de Wifi (problème routeur WLAN)	3	1	2	1	**6**	Relancer le routeur
5a	Problème tunnel web	Le service ne fonctionne plus	1	5	2	5	**50**	Trouver un nouveau service de communication, peut prendre du temps
6a	Panne API SLAVE client	Câble débranché ou coupé	1	2	1	2	**4**	Dépend de l'habitation, il faut l'enlever de chemin à câbles pour le remplacer
6b	Panne API SLAVE client	Pas d'alimentation	2	2	2	2	**16**	Remise en état dépend du fournisseur d'électricité
6c	Panne API SLAVE client	API SLAVE en erreur	1	2	2	1	**4**	Débrancher puis rebrancher API. Le programme dans l'EEPROM sera relancé.
7a	Problème modem client	Pas d'internet (maintenance)	3	2	2	2	**24**	Remise en état dépend du fournisseur d'internet
7b	Problème modem client	Pas d'alimentation	2	2	2	2	**16**	Remise en état dépend du fournisseur d'électricité

Tableau 8: Analyse de risques de pannes lors de la transmission d'une alarme - Atelier R2D2

Plus la criticité de la panne est élevée, plus la panne est grave et inversément. Sur la figure ci-dessous, aucune panne n'est supérieure à 60. Le système proposé est déjà relativement sûr. Pour améliorer sa sureté il faut se pencher sur les pannes les plus élevées et trouver un moyen de diminuer le risque qu'elles se produisent.

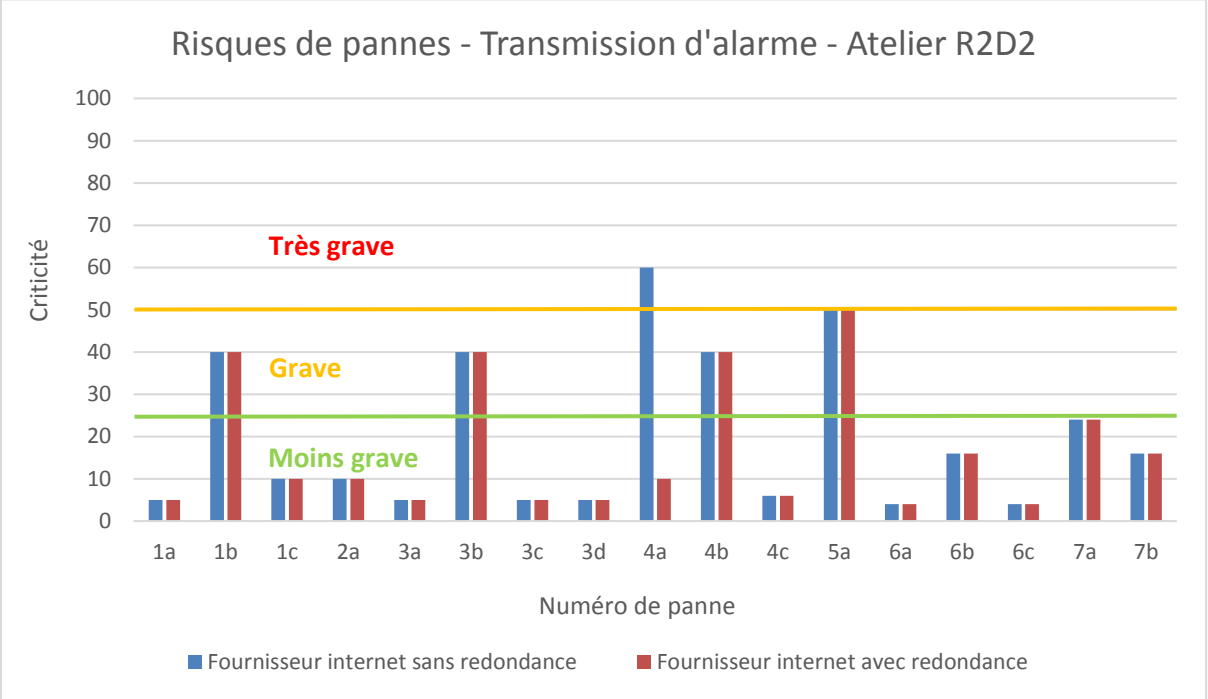


Figure 19: Risques de pannes lors de la transmission d'une alarme - Atelier R2D2

Les pannes ayant une criticité en dessous de 25 sont considérées comme légèrement graves pour le système d'alarmes car la possibilité d'y remédier est simple et rapide. Les numéros de pannes 6 et 7 sont considérées comme moins inquiétantes car elles n'affectent pas le système d'alarme. Il sera toujours opérationnel pour tous les autres clients.

En étudiant la panne ayant la criticité la plus élevée (panne 4a, tableau 8), il est nécessaire d'avoir un abonnement internet avec redondance en cas de maintenance du fournisseur réseau. Swisscom (un fournisseur d'internet parmi tant d'autres) propose un abonnement « *inOne PME* » avec un modem possédant un emplacement USB où l'on branche une clé USB de réseau mobile 4G. En effet, lorsque le fournisseur d'internet est en maintenance, la connexion au web est automatiquement reliée au réseau mobile 4G de manière à toujours être « connecté ».

5 Serveur d'alarme Wago – Partie software

5.1 Programmation API SLAVE

Lorsqu'une alarme sur l'automate programmable d'un client survient, il faut la répercuter sur la base de données du PC de l'Atelier R2D2. Des requêtes sont donc envoyées en destination du serveur PhpMyAdmin vue dans le paragraphe §4.6. La vue générale du programme (disponible en annexe 5a) qui sera intégré dans les automates des clients est démontrée sur la figure ci-dessous :

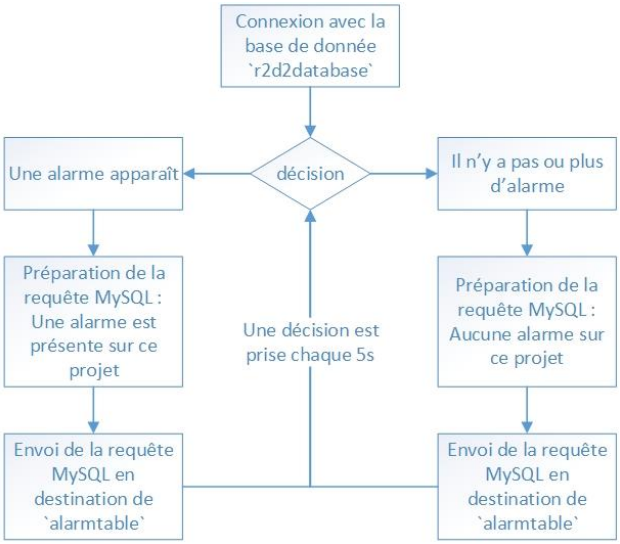


Figure 20: Diagramme de flux - Programmation API Slave

Le rafraîchissement de la table de données `*alarmtable*` est effectué chaque 5 secondes. C'est l'API SLAVE qui gère la quittance des alarmes par le biais de sa propre visualisation ou interface homme-machine.

Pour créer la requête MySQL, une fonction est effectuée. Elle est disponible en annexe 5b. Des blocs de fonctions créés par Wago sont présents dans le projet :

- *fbMySqlLogin* : pour la connexion entre l'API et la base de données
- *fbMySqlLogout* : pour la déconnexion entre l'API et la base de données
- *fbMySqlExecute* : pour exécuter la requête MySQL

5.2 Programmation API MASTER

5.2.1 Programme Alarm_DB

L'automate Wago MASTER de nouvelle génération (PFC 200, 750-8207) se trouve dans le bureau de l'Atelier R2D2. Le bureau de l'entreprise est pourvu de systèmes domotiques contrôlés par cet API. Il a donc fallu rajouter un peu de programmation dans cet automate pour y insérer le serveur d'alarme. Un programme *Alarm_DB*[15] est alors créé.

[15] DB est une abréviation de base de données en anglais

Ce serveur d'alarme fonctionne selon le principe suivant :

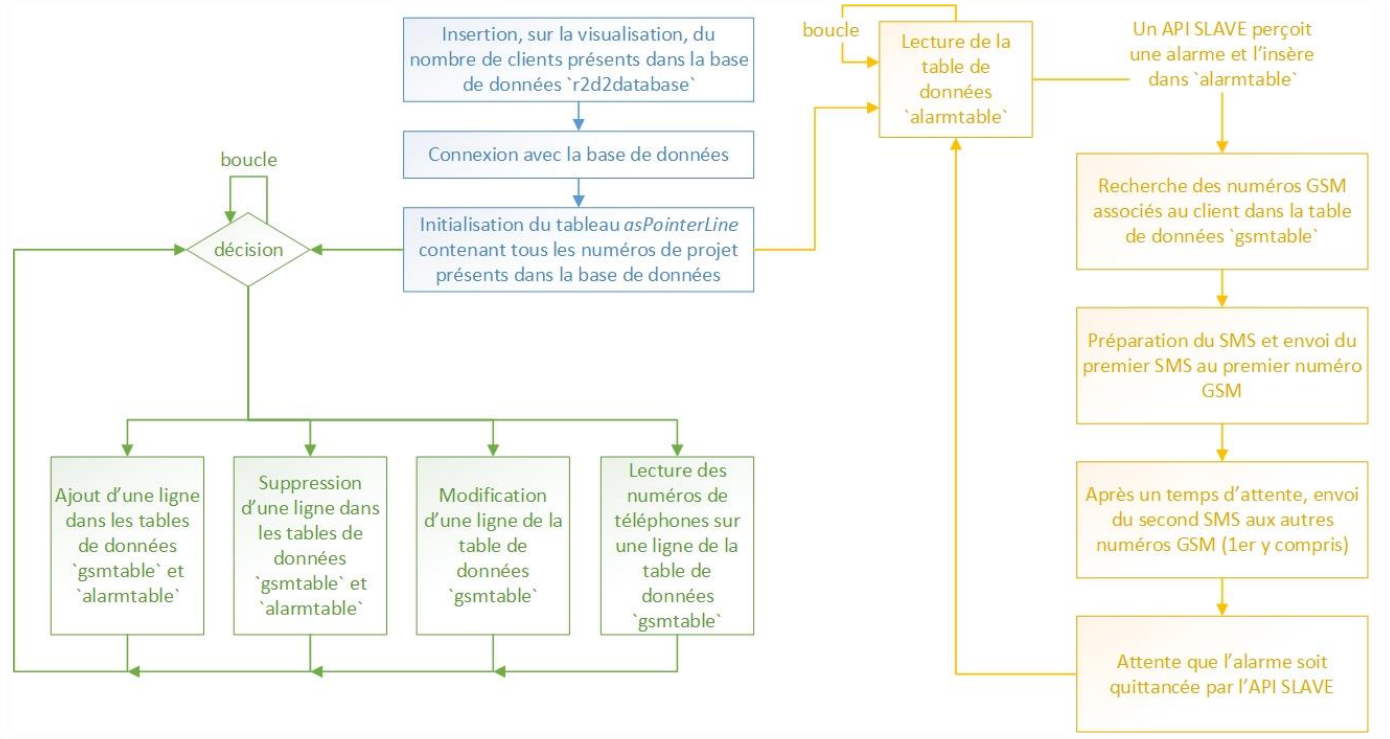


Figure 21: Diagramme de flux - Programmation serveur d'alarme - API MASTER

Pour comprendre ce programme, 3 couleurs distinctes ont été utilisées pour l'illustrer :

- Bleu
  - Phase d'initialisation et de connexion avec la base de données MySQL présente sur le serveur PhpMyAdmin
- Vert
  - Modification et lecture de la base de données par le biais de la visualisation web
- Jaune
  - Gestion des alarmes et d'envoi d'SMS afin d'avertir le client qu'une alarme s'est produite dans son bâtiment
  - Lecture de la table `*alarmtable*` chaque seconde

Ce serveur d'alarme est écrit en langage structuré (ST) afin d'être flexible en cas de modification ultérieure du programme, mais aussi pour cause de simplicité d'écriture.

Ce programme est disponible en annexe 6a.

5.2.2 Fonctions créées

Afin de préparer les différentes requêtes pour questionner et modifier la base de données MySQL, les fonctions suivantes sont implémentées :

- *funMySQLinsert*
  - Fonction d'insertion d'une nouvelle ligne dans les tables de données `*gsmtable*` et `*alarmtable*`
- *funMySQLdelete*
  - Fonction de suppression d'une ligne des tables de données `*gsmtable*` et `*alarmtable*`
- *funMySQLedit*
  - Fonction de modification des numéros de téléphone dans la table de données `*gsmtable*`

Ces fonctions, écrites en langage structuré (ST), sont disponibles en annexe 6b.

5.3 Visualisation API MASTER

L'interface homme-machine du serveur d'alarme est une visualisation web conçue avec le logiciel de programmation e!cockpit. Elle permet de modifier et lire les informations de la table de données `*gsmtable*` et d'avoir un retour sur l'éventuel projet (API SLAVE d'un client) ayant une alarme active.

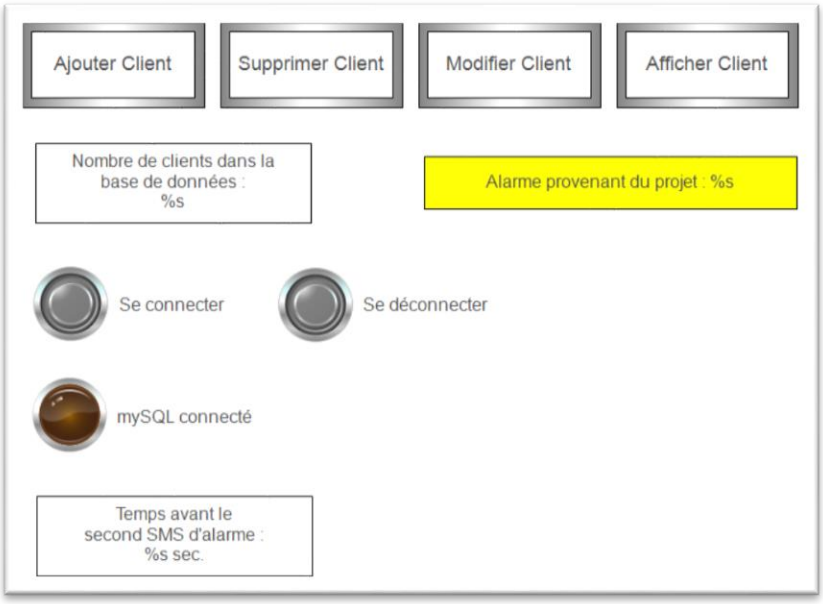


Figure 22: Page d'accueil de la visualisation du serveur d'alarme

Lorsqu'une alarme apparaît d'un API SLAVE, un premier SMS est directement envoyé au premier numéro de téléphone portable associé au client en question. Puis, après un certain temps configurable comme le démontre la figure ci-dessus, un second SMS est envoyé en destination des autres numéros GSM associé au client (y compris le premier numéro de téléphone).

A noter que le « *%s* » signifie qu'une variable de type numérique (INT, WORD, …) ou chaine de caractères (STRING) s'affiche lorsque l'automate MASTER est en fonctionnement.

Les autres pages de cette visualisation sont disponibles en annexe 6c.

5.4 Test du serveur d'alarme : erreur provenant du pilotage de chauffages

Afin de tester le fonctionnement du serveur d'alarme, l'API SLAVE de la partie pilotage de chauffages (vus aux chapitres §2 et §3) est connecté au même réseau Ethernet LAN que l'API MASTER et un PC contenant la base de données MySQL.

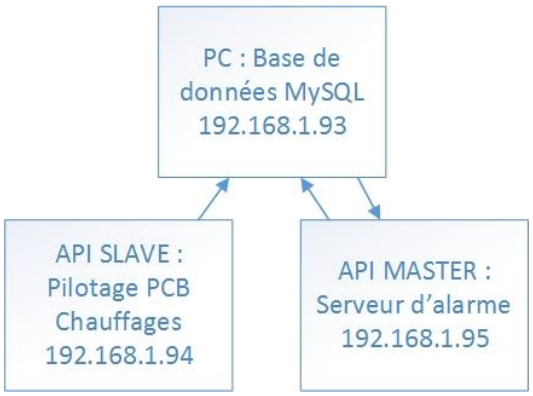


Figure 23: Vue en schéma bloc - test du serveur d'alarme

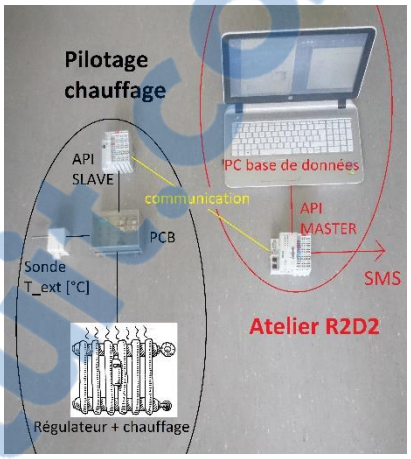


Figure 24: Vue avec composants réels - test du serveur d'alarme

L'alarme part donc de l'API SLAVE pour modifier la table de données MySQL `*alarmtable*`. Ensuite l'API MASTER perçoit cette alarme, recherche les numéros de téléphones portables sur la table de données `gsmtable`, puis enverra le ou les SMS au numéro GSM configuré. Lorsque l'alarme est quittancée par l'API SLAVE, elle est enlevée de la table de données `*alarmtable*`.

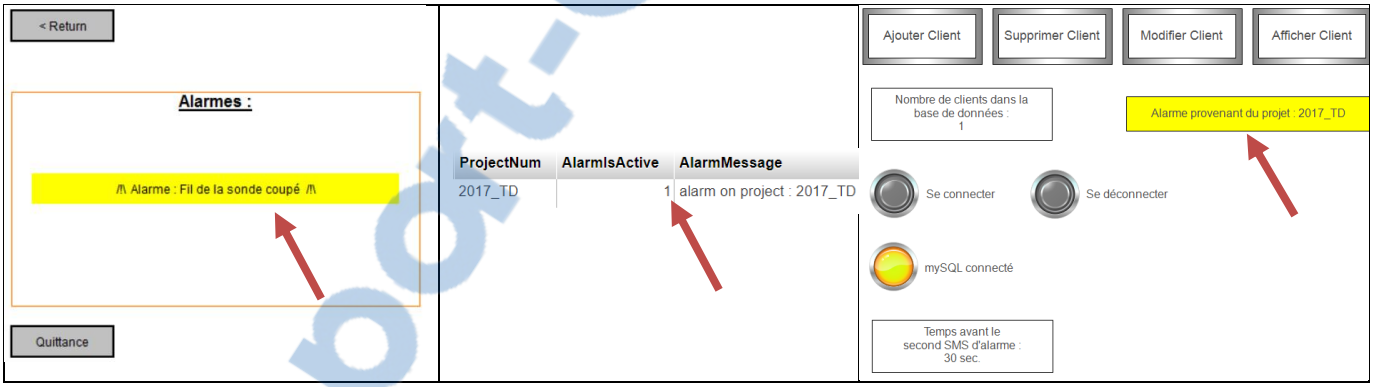


Tableau 9: Parcours de l'alarme

Le chemin parcours de l'alarme fonctionne sans mésaventure, comme le confirme le tableau de captures d'écran ci-dessus.

5.5 Perspectives

5.5.1 Installation dans le bureau de l'Atelier R2D2

Afin de mettre en place ce serveur d'alarme dans les locaux de l'Atelier R2D2 les démarches suivantes sont à effectuer :

- Installer uWamp sur le PC du bureau
  - Ce logiciel contient les serveurs Apache, MySQL et PhpMyAdmin
- Insérer la programmation du serveur d'alarme dans l'API du bureau
- Développer ou installer un logiciel permettant d'atteindre la base de données depuis un réseau internet différent du réseau LAN de l'Atelier R2D2

5.5.2 Atteindre le serveur PhpMyAdmin depuis un réseau extérieur

Pour pouvoir atteindre PhpMyAdmin depuis un réseau internet différent du réseau LAN de l'Atelier R2D2, deux possibilités se distinguent :

- Utiliser DynDNS
  - Service informatique permettant d'associer une adresse IP dynamique à un nom de domaine. Par exemple, associer une adresse IP 10.91.1.27 à une page web http://exemple-r2d2database.ch. Ainsi lorsqu'un appareil désire accéder à l'adresse IP de l'ordinateur pour aller sur le serveur PhpMyAdmin, il peut simplement l'accéder par le biais de l'URL ci-dessus.
- Utiliser VPN
  - Réseau privé virtuel : C'est une sorte de tunnel internet permettant de relier deux réseaux distants. Cela fait croire que les API SLAVE sont dans le réseau internet LAN de l'Atelier R2D2.

5.5.3 Sauvegarde de la base de données

Une exportation des deux tables de données est possible par le biais d'une commande MySQL. Ainsi, si une mauvaise manipulation ou un quelconque problème survient du serveur PhpMyAdmin, des sauvegardes seront disponibles.

La requête suivante permet d'exporter ces deux tables sur le bureau d'un ordinateur (en modifiant *oscar_000* par le nom d'un utilisateur existant sur le PC) :

```
SELECT * INTO OUTFILE 'C:/Users/oscar_000/Desktop/gsmtable.sql' FROM `gsmtable` ;
SELECT * INTO OUTFILE 'C:/Users/oscar_000/Desktop/alarmtable.sql' FROM `alarmtable` ;
```

De plus, cette base de données pourra être exportée sur deux disques durs différents afin d'être certain de ne pas perdre les informations des clients.

5.5.4 SMS pour centrale d'alarme CERTAS

Un SMS avec des caractères spéciaux selon le type d'alarme et le client peut, par la suite, être envoyé à l'entreprise de sécurité CERTAS. Lorsque la centrale d'alarme CERTAS reçoit un SMS, elle appelle le client en question et gère l'incident avec lui. Si le client ne répond pas, une équipe est envoyée à son domicile.

La figure 25 démontre le cas d'une alarme causée par un dérangement sur une installation.

```
1  mess                : STRING := '=760009 E K03&DERANGEMENT INSTALLATION*';
2  destination_no3     : STRING := '41791112233';
```

Figure 25: SMS en destination de CERTAS[16]

Le nombre à 6 chiffres dans le message contient le numéro CERTAS du client. Ainsi CERTAS sait de quel client cela provient en regardant sa propre base de données.

5.5.5 Quittance du client par SMS

Le bloc de fonction *Fb_simpleSMS_8207* créé par Wago permet d'envoyer et de recevoir des SMS. Par conséquent, lorsqu'une alarme est déclenchée accidentellement par le client, il pourrait

[16] Le numéro de téléphone présent sur cette figure est un numéro fictif

répondre par SMS au numéro de la carte Sim insérée dans l'API MASTER du serveur d'alarme de manière à ce que le second SMS ne soit pas envoyé aux autres numéros de téléphone après un certain temps, comme expliqué dans le chapitre §5.2.

Une quittance du client par SMS est possible avec le bloc de fonction Wago

5.5.6 Enregistrement de l'heure d'envoi des SMS

Si un client se retourne contre l'entreprise Atelier R2D2 pour cause de non réception d'un SMS lorsqu'il y a eu une alarme chez lui, il est nécessaire de garder une preuve que le SMS ait bien été envoyé. Un enregistrement de l'heure d'envoi du message dans un tableau de variables permet de fournir la preuve désirée.

6 Serveur météo

Malheureusement, le temps imparti ne m'a pas permis de développer un prototype d'un serveur météo car la partie de gestion de base de données a pris plus de temps que prévu afin de proposer un serveur d'alarme sûr.

Ce serveur météo a été prévu dans mon planning initial comme le démontre l'annexe 7a : les croix en noir représentent le planning théorique et les croix en rouge le déroulement réel du présent travail de diplôme.

Si ce serveur météo est à réaliser, voici le déroulement proposé :

1. La compréhension d'un bloc de fonction existant pour récupérer les prévisions météo depuis un site web et, dans le besoin, adaptation de ce bloc
2. Des recherches sur la création de fichier XML avec automate programmable Wago
3. La récupération des valeurs météos dans des variables, puis l'inscription de ces valeurs dans un fichier XML
4. Des tests du système développé

7 Conclusion

Ce travail de diplôme propose une manière d'influencer, à distance, le comportement du chauffage de résidences, par exemple en tenant compte de la météo, afin d'améliorer son efficience.

On remarque que pour des systèmes automatisés, il est possible d'assembler plusieurs travaux issus de domaines différents. En effet, ces tâches de nature bien distincte - pilotage de chauffages, serveur d'alarmes et serveur météo - ont pu être étudiées et assemblées pour former un tout.

L'interface de commande de chauffages doit encore être testée sur des chauffages de types différents pour être définitivement validée. Les objectifs sont atteints car le programme fonctionne et pilote de manière correcte la carte électronique PCB. De plus, des alarmes en cas de mauvaise manipulation ou lors d'une erreur du système sont implémentées dans le bloc de fonction de pilotage.

Cette interface est testée en commun avec la seconde partie, le serveur d'alarme. Lorsqu'une alarme survient, un second automate envoie un SMS au numéro de téléphone portable attribué à ce présent projet (mon numéro GSM).

Il restera à installer ce serveur d'alarme et à améliorer le programme de l'automate qui sera dans les bureaux de l'entreprise Atelier R2D2 sàrl. Une liste des tâches futures se trouve dans le paragraphe §5.5.

Le serveur météo n'a malheureusement pas pu être développé mais pourrait par la suite être intégré à ce projet.

À l'avenir, si ce passionnant projet doit être poursuivi, l'informatique, l'automation et l'électronique sont les principales compétences requises.

Sion, le 18 août 2017 Oscar Torres

8 Bibliographie

- Caatteuw Jérôme (Diplôme 2016) API_PAC : Commande prédictive d'une PAC en fonction de la météo
- Documentation MySQL (consultée durant le mois de juin et juillet 2017)
    - https://dev.mysql.com
- Documentation Wago (consultée pendant toute la durée du travail de bachelor)
    - http://www.wago.ch/fr/service/telechargements/assistant-de-telechargement/index.jsp

9 Logiciels

Rapport et présentation		
- Microsoft Office	v2015	Word, Excel, Visio, Powerpoint
Schéma électronique		
- Altium Designer	v17.0.8	Schéma électronique et création de PCB
Wago		
- Codesys	v2.3.9	Programmation API
- e!cockpit	v1.3.1	Programmation API nouvelle génération
- Wago Ethernet Settings	v6.8.2	Adressage d'adresses IP fixes des API
- Wago-IO-Check 3	v3.16.2	Configuration de la carte des AI
Base de données		
- UwAmp	v3.0.2	Serveur PhpMyAdmin, Apache, MySQL

10 Annexes

Annexe 1 : Etude afin de concevoir la carte électronique PCB

a. Lecture de valeur de sonde
b. Cartes d'entrées analogiques API pour automates Wago et Siemens

Annexe 2 : Schémas électroniques PCB

a. Schéma électronique, PCB Jérôme Catteeuw v1, Travail de diplôme 2016
b. Schéma électronique, PCB Oscar Torres v1
c. Schéma électronique, PCB low-cost Oscar Torres v1

Annexe 3 : Carte électronique PCB Oscar Torres v1 - Hardware

a. Mode d'emploi de la carte électronique
b. Liste de matériel de la carte électronique
c. Protocole de test

Annexe 4 : Programmation API – Pilotage de chauffages

a. Tutoriel de programmation pour une sonde de température différente de la PT 1000
b. Programme principal (Main program) : *Control_Heating*
c. Bloc de fonction principal : *Fb_ControlPCB_Heating*
d. Blocs de fonctions de conversion de valeurs : Conversion ohm <-> degrés
e. Programme de test du code : *Test_the_code*

Annexe 5 : Programmation du serveur d'alarme – API SLAVE

a. Programmation de transmission d'alarme à la base de données MySQL
b. Fonction pour faire une requête MySQL pour la table `*alarmtable*`

Annexe 6 : Programmation du serveur d'alarme et visualisation web – API MASTER

a. Programme du serveur d'alarme
b. Fonctions de requêtes MySQL pour l'ajout, la suppression, la modification ou la lecture d'informations de clients pour la table `*gsmtable*`
c. Visualisation web du serveur d'alarme

Annexe 7 : Planning du travail de diplôme

a. Planning Gantt – Travail de Bachelor 2017, Oscar Torres

11 Liste d'illustrations et tableaux

11.1 Figures

11.2 Tableaux

11.3 Schémas

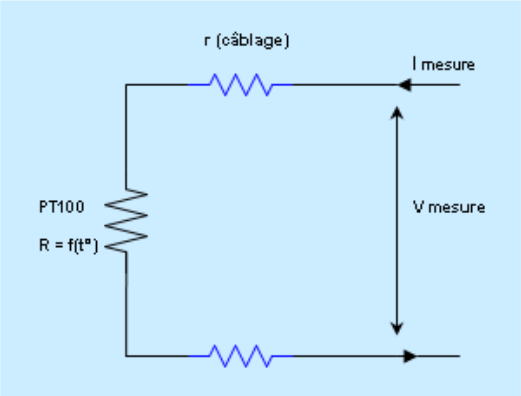
Annexe 1 : Etude afin de concevoir la carte électronique PCB

a. Lecture de valeur de sonde
b. Cartes d’entrées analogiques API pour automates Wago et Siemens

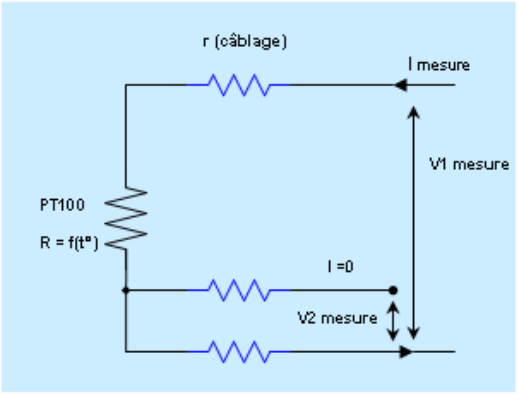
Lecture de valeur de sonde

Mode de connexion et interfaces électroniques

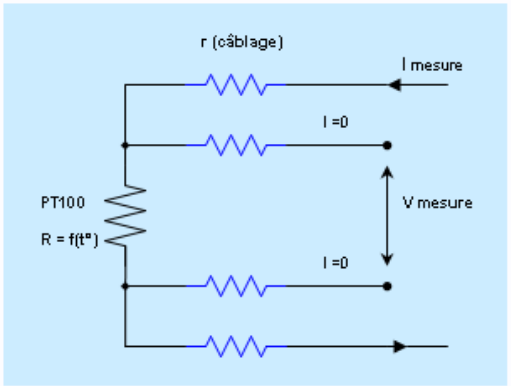
Le choix de la méthode de connexion et du type d'interface électronique dépend de la précision de mesure recherchée pour une installation donnée. Une sonde platine peut être utilisée selon 3 modes de connexion : 2 fils, 3 fils, ou 4 fils.



Interface 2 fils



Interface 3 fils
Une méthode parmi d'autres, ici : V_{Pt} = V1 - 2 x V2



Interface 4 fils

- Le mode "2 fils", le plus simple, n'apporte aucune précision de mesure dès que l'effet induit par la résistance des câbles de connexion devient du même ordre de grandeur que la précision recherchée. A moins d'être pris en compte en tant qu'erreur systématique, un câble standard AWG24 (85 ohms/km) introduit une erreur brute de l'ordre de 0.4°C par mètre de connexion pour une sonde Pt100, mais seulement de 0.04°C pour une sonde Pt1000.

- Le mode "3 fils" assure très souvent une précision suffisante pour les longues distances de câblage, et il existe plusieurs méthodes analogiques ou numériques, toutes basées sur l'hypothèse que les 3 fils de connexion présentent la même valeur de résistance électrique. Par exemple, l'utilisation d'un câble AWG18 (21 ohm/km) tout à fait adapté pour une connexion moyenne distance, introduira en fonction de la qualité de l'interface électronique, moins de 0.4°C d'erreur pour 100 mètres de connexion avec une sonde Pt100.

- Le mode "4 fils" représente en toute théorie le montage apportant la meilleure précision, puisque la mesure de tension étant réalisée directement au niveau de la partie active de la sonde avec une interface haute impédance, les résistances des câbles de connexion n'interviennent plus dans l'erreur de la mesure. Mais dans la pratique, cela ne signifie pas qu'il s'agit forcément là du mode de connexion à privilégier à tout prix.

Source : http://www.technetea.com/PT100.html

SIEMENS

Fiche technique 6ES7531-7KF00-0AB0

SIMATIC S7-1500, MODULE ENTREES ANALOGIQUES AI 8 X U/I/RTD/TC ST, RESOLUTION 16 BIT PRECISION 0,3%, 8 CANAUX EN GROUPES DE 8, 4 CANAUX AVEC MESURE RTD, TENSION MODE COMMUN 10V; DIAGNOSTIC; ALARMES PROCESS INCL. ELEMENT D'ALIMENTATION ETRIER DE BLINDAGE ET BORNE DE BLINDAGE

Informations générales	
Désignation du type de produit	AI 8xU/I/RTD/TC ST
Version fonctionnelle du matériel	FS01
Version du firmware	V2.0.0
• Mise à jour du firmware possible	Oui
Fonction du produit	
• Données I&M	Oui; I&M0 bis I&M3
• Plage de mesure adaptable	Non
• Valeurs de mesure adaptables	Non
• Adaptation de l'étendue de mesure	Non
Ingénierie avec	
• STEP 7 TIA Portal configurable/intégré à partir de la version	V12 / V12
• STEP 7 configurable/intégré à partir de la version	V5.5 SP3 / -
• PROFIBUS à partir de la version/révision GSD	V1.0 / V5.1
• PROFINET à partir de la version/révision GSD	V2.3 / -
Mode de fonctionnement	
• Suréchantillonnage	Non

• MSI	Oui
CiR – Configuration en mode RUN	
Reparamétrage possible en RUN	Oui
Calibrage en RUN possible	Oui
Tension d'alimentation	
Valeur nominale (CC)	24 V
Plage admissible, limite inférieure (CC)	20,4 V
Plage admissible, limite supérieure (CC)	28,8 V
Protection contre l'inversion de polarité	Oui
Courant d'entrée	
Consommation, maxi	240 mA; pour une alimentation de 24 V CC
Alimentation des capteurs	
Alimentation des capteurs 24 V	
• Protection contre les courts-circuits	Oui
• Courant de sortie, maxi	20 mA; max. 47 mA par voie pour une durée < 10 s
Puissance	
Appel de puissance du bus de fond de panier	0,7 W
Puissance dissipée	
Puissance dissipée, typ.	2,7 W
Entrées analogiques	
Nombre d'entrées analogiques	8
• pour mesure de courant	8
• pour mesure de tension	8
• pour mesure de résistance/sonde thermométrique à résistance	4
• pour mesure de thermocouple	8
Tension d'entrée admissible pour entrée de tension (limite de destruction), maxi	28,8 V
Courant d'entrée admissible pour entrée de courant (limite de destruction), maxi	40 mA
Unité technique réglable pour mesure de température	Oui; °C / °F / K
Etendues d'entrée (valeurs nominales), tensions	
• 0 à +5 V	Non
• 0 à +10 V	Non
• 1 V à 5 V	Oui
• Résistance d'entrée (1 V à 5 V)	100 kΩ
• -1 V à +1 V	Oui
• Résistance d'entrée (-1 V à +1 V)	10 MΩ
• -10 V à +10 V	Oui

• Résistance d'entrée (-10 V à +10 V)	100 kΩ
• -2,5 V à +2,5 V	Oui
• Résistance d'entrée (-2,5 V à +2,5 V)	10 MΩ
• -25 mV à +25 mV	Non
• -250 mV à +250 mV	Oui
• Résistance d'entrée (-250 mV à +250 mV)	10 MΩ
• -5 V à +5 V	Oui
• Résistance d'entrée (-5 V à +5 V)	100 kΩ
• -50 mV à +50 mV	Oui
• Résistance d'entrée (-50 mV à +50 mV)	10 MΩ
• -500 mV à +500 mV	Oui
• Résistance d'entrée (-500 mV à +500 mV)	10 MΩ
• -80 mV à +80 mV	Oui
• Résistance d'entrée (-80 mV à +80 mV)	10 MΩ
Etendues d'entrée (valeurs nominales), courants	
• 0 à 20 mA	Oui
• Résistance d'entrée (0 à 20 mA)	25 Ω; plus env. 42 Ohm pour la protection contre les surtensions par CTP
• -20 mA à +20 mA	Oui
• Résistance d'entrée (-20 mA à +20 mA)	25 Ω; plus env. 42 Ohm pour la protection contre les surtensions par CTP
• 4 mA à 20 mA	Oui
• Résistance d'entrée (4 mA à 20 mA)	25 Ω; plus env. 42 Ohm pour la protection contre les surtensions par CTP
Etendues d'entrée (valeurs nominales), thermocouples	
• Type B	Oui
• Résistance d'entrée (type B)	10 MΩ
• Type C	Non
• Type E	Oui
• Résistance d'entrée (type E)	10 MΩ
• Type J	Oui
• Résistance d'entrée (type J)	10 MΩ
• Type K	Oui
• Résistance d'entrée (type K)	10 MΩ
• Type L	Non
• Type N	Oui
• Résistance d'entrée (type N)	10 MΩ
• Type R	Oui
• Résistance d'entrée (type R)	10 MΩ
• Type S	Oui
• Résistance d'entrée (type S)	10 MΩ
• Type T	Oui

• Résistance d'entrée (type T)	10 MΩ
• Type TXK/TXK(L) selon GOST	Non
Etendues d'entrée (valeurs nominales), thermomètres à résistance	
• Cu 10	Non
• Cu 10 selon GOST	Non
• Cu 50	Non
• Cu 50 selon GOST	Non
• Cu 100	Non
• Cu 100 selon GOST	Non
• Ni 10	Non
• Ni 10 selon GOST	Non
• Ni 100	Oui; Standard / climat
• Résistance d'entrée (Ni 100)	10 MΩ
• Ni 100 selon GOST	Non
• Ni 1000	Oui; Standard / climat
• Résistance d'entrée (Ni 1000)	10 MΩ
• Ni 1000 selon GOST	Non
• LG-Ni 1000	Oui; Standard / climat
• Résistance d'entrée (LG-Ni 1000)	10 MΩ
• Ni 120	Non
• Ni 120 selon GOST	Non
• Ni 200 selon GOST	Non
• Ni 500	Non
• Ni 500 selon GOST	Non
• Pt 10	Non
• Pt 10 selon GOST	Non
• Pt 50	Non
• Pt 50 selon GOST	Non
• Pt 100	Oui; Standard / climat
• Résistance d'entrée (Pt 100)	10 MΩ
• Pt 100 selon GOST	Non
• Pt 1000	Oui; Standard / climat
• Résistance d'entrée (Pt 1000)	10 MΩ
• Pt 1000 selon GOST	Non
• Pt 200	Oui; Standard / climat
• Résistance d'entrée (Pt 200)	10 MΩ
• Pt 200 selon GOST	Non
• Pt 500	Oui; Standard / climat
• Résistance d'entrée (Pt 500)	10 MΩ
• Pt 500 selon GOST	Non
Etendues d'entrée (valeurs nominales), résistances	

• 0 à 150 ohms	Oui
• Résistance d'entrée (0 à 150 ohms)	10 MΩ
• 0 à 300 ohms	Oui
• Résistance d'entrée (0 à 300 ohms)	10 MΩ
• 0 à 600 ohms	Oui
• Résistance d'entrée (0 à 600 ohms)	10 MΩ
• 0 à 3000 ohms	Non
• 0 à 6000 ohms	Oui
• Résistance d'entrée (0 à 6000 ohms)	10 MΩ
• PTC	Oui
• Résistance d'entrée (PTC)	10 MΩ
Thermocouple (TC)	
Compensation en température	
— paramétrable	Oui
— Compensation interne de température	Oui
— Compensation externe de la température via le thermomètre à résistance	Oui
— Compensation pour température de jonction de référence 0 °C	Oui; valeur fixe réglable
— Canal de référence du module	Oui
Longueur de câble	
• blindé, maxi	800 m; pour U/I, 200 m pour R/RTD, 50 m pour TC

Formation des valeurs analogiques pour les entrées	
Temps d'intégration et de conversion/résolution par voie	
• Résolution avec domaine de dépassement (bits avec signe), maxi	16 bit
• Temps d'intégration paramétrable	Oui
• Temps d'intégration (ms)	2,5 / 16,67 / 20 / 100 ms
• Temps de conversion de base y compris temps d'intégration (ms)	9 / 23 / 27 / 107 ms
— Temps de conversion supplémentaire pour surveillance de rupture de fil	9 ms (à prendre en compte pour mesure R/RTD/TC)
— Temps de conversion supplémentaire pour mesure de résistance	150 Ohm, 300 Ohm, 600 Ohm, Pt100, Pt200, Ni100 : 2 ms, 6000 Ohm, Pt500, Pt1000, Ni1000, LG-Ni1000, CTP : 4 ms
• Réjection des tensions perturbatrices pour fréquence perturbatrice f1 en Hz	400 / 60 / 50 / 10 Hz
• Temps pour calibrage d'offset (par module)	Temps de conversion de base de la voie lente
Lissage des valeurs de mesure	
• paramétrable	Oui
• Niveau: néant	Oui
• Niveau: faible	Oui
• Niveau: moyen	Oui

• Niveau: fort	Oui

Capteurs	
Raccordement des capteurs de signaux	
• pour mesure de tension	Oui
• pour mesure de courant comme transmetteur de mesure 2 fils	Oui
— Charge du transmetteur 2 fils	820 Ω
• pour mesure de courant comme transmetteur de mesure 4 fils	Oui
• pour mesure de la résistance en montage 2 fils	Oui; seulement pour CTP
• pour mesure de la résistance en montage 3 fils	Oui; toutes les plages de mesure sauf CTP ; compensation interne des résistances de ligne
• pour mesure de la résistance en montage 4 fils	Oui; toutes les plages de mesure sauf CTP

Défauts/Précisions	
Erreur de linéarité (rapportée à l'étendue d'entrée), (+/-)	0,02 %
Erreur de température (rapportée à l'étendue d'entrée), (+/-)	0,005 %/K; pour type de TC T 0,02 +/- %/K
Diaphonie entre entrées, max.	-80 dB
Répétabilité en régime établi à 25 °C (rapportée à l'étendue d'entrée), (+/-)	0,02 %
Erreur de température de la compensation interne	+/-6 °C
Limite d'erreur pratique dans toute la plage de température	
• Tension, rapportée à l'étendue d'entrée, (+/-)	0,3 %
• Courant, rapporté à l'étendue d'entrée, (+/-)	0,3 %
• Résistance, rapportée à l'étendue d'entrée, (+/-)	0,3 %
• Thermomètre à résistance, rapporté à l'étendue d'entrée, (+/-)	Ptxxx standard : ± 1,5 K, Ptxxx climat : ± 0,5 K, Nixxx standard : ± 0,5 K, Nixxx climat : ± 0,3 K
• Thermocouple, rapporté à l'étendue d'entrée, (+/-)	Type B : > 600 °C ± 4,6 K, type E : > -200 °C ±1,5 K, type J : > -210 °C ±1,9 K, type K : > -200 °C ±2,4 K, type N : > -200 °C ±2,9 K, type R : > 0 °C ±4,7 K, type S : > 0 °C ± 4,6 K, type T : > -200 °C ±2,4 K
Limite d'erreur de base (limite d'erreur pratique à 25°C)	
• Tension, rapportée à l'étendue d'entrée, (+/-)	0,1 %
• Courant, rapporté à l'étendue d'entrée, (+/-)	0,1 %
• Résistance, rapportée à l'étendue d'entrée, (+/-)	0,1 %
• Thermomètre à résistance, rapporté à l'étendue d'entrée, (+/-)	Ptxxx standard : ± 0,7 K, Ptxxx climat : ± 0,2 K, Nixxx standard : ± 0,3 K, Nixxx climat : ± 0,15 K
• Thermocouple, rapporté à l'étendue d'entrée, (+/-)	Type B : > 600 °C ±1,7 K, type E : > -200 °C ±0,7 K, type J : > -210 °C ±0,8 K, type K : > -200 °C ±1,2 K, type N : > -200 °C ±1,2 K, type R : > 0 °C ±1,9 K, type S : > 0 °C ±1,9 K, type T : > -200 °C ±0,8 K

Réjection des tensions perturbatrices pour f = n x (f1 +/- 1 %), f1 = fréquence perturbatrice	
• Perturbation de mode série (valeur de pointe de la perturbation < valeur nominale de l'étendue d'entrée)	40 dB
• Tension de mode commun, maxi	10 V
• Perturbation de mode commun, mini	60 dB
Mode synchrone	
Mode synchrone (application synchronisée jusqu'à la borne)	Non
Alarmes/diagnostic/information d'état	
Fonctions de diagnostic	Oui
Alarmes	
• Alarme de diagnostic	Oui
• Alarme de dépassement de seuil	Oui; deux seuils inférieurs et deux seuils supérieurs
Messages de diagnostic	
• Surveillance de la tension d'alimentation	Oui
• Rupture de fil	Oui; Uniquement pour 1 ... 5 V et 4 ... 20 mA, TC, R et RTD
• Débordement haut / Débordement bas	Oui
Signalisation de diagnostic par LED	
• LED RUN	Oui; LED verte
• LED ERROR	Oui; LED rouge
• Surveillance de la tension d'alimentation (LED PWR)	Oui; LED verte
• Affichage de l'état de la voie	Oui; LED verte
• pour diagnostic de la voie	Oui; LED rouge
• pour diagnostic du module	Oui; LED rouge
Séparation galvanique	
Séparation galvanique des canaux	
• entre les voies	Non
• entre les voies, par groupes de	8
• entre voies et bus interne	Oui
• entre les voies et la tension d'alimentation de l'électronique	Oui
Différence de potentiel admissible	
entre les entrées (UCM)	20 V CC
entre les entrées et MANA (UCM)	10 V CC
Isolation	
Isolation vérifiée avec	707 V CC (type Test)
Conditions ambiantes	
Température ambiante en service	

• Montage horizontal, mini	0 °C
• Montage horizontal, maxi	60 °C
• Montage vertical, mini	0 °C
• Montage vertical, maxi	40 °C

Mode décentralisé	
Démarrage prioritaire	Non

Dimensions	
Largeur	35 mm
Hauteur	147 mm
Profondeur	129 mm

Poids	
Poids approx.	310 g

Autres	
Remarque:	Erreur de base supplémentaire et bruit de fond pour temps d'intégration = 2,5 ms : Tension : ±250 mV (±0,02 %), ±80 mV (±0,05 %), ±50 mV (±0,05 %); Résistance : 150 Ohm ±0,02 %; Sonde thermométrique à résistance : Pt100 climat : ±0,08 K, Ni 100 climat : ±0,08 K; Thermocouple : type B, R, S : ±3 K, type E, J, K, N, T : ±1 K

dernière modification : 23-05-2017

S7-300 Analogeingaben

Baugruppentyp	Analogeingaben SM 33x									
Besonderheiten dieser Baugruppe	Universell einzusetzende Baugruppe, die alle gängigen Messbereiche abdeckt und damit die Ersatzteilhaltung wesentlich vereinfacht		Hochauflösende und hochgenaue Baugruppe zum Erfassen von Strömen und Spannungen		Preiswerte, universelle Mischbaugruppe zum Erfassen bzw. Ausgeben von Strömen und Spannungen	Universelle Mischbaugruppe zum Messen von Spannungen, Widerständen und Temperaturen über Widerstandgeber (RTD) sowie zum Ausgeben von Spannungen	Mischbaugruppe für sehr schnelle Applikationen, wie z. B. Kunststoffmaschinen; integrierter Komperator	Universell einsetzbare Baugruppe, die alle gängigen Messbereiche abdeckt (keine TC-Messung) und damit die Ersatzteilhaltung vereinfacht	Sehr schnelle Baugruppe, die mit Momentanwertverschlüsselung arbeitet; geeignet für taktsynchrone Applikationen	Hochauflösende, einzelpotentialgetrennte Baugruppe für Spannungsmessung und Thermoelemente, zweikanalig redundant für die erhöhten Anforderungen in der Prozesstechnik
Spannungsmessbereich Geber	± 80 mV ± 250 mV ± 500 mV ± 1 V ± 2,5 V		± 5 V ± 10 V 1 ... 5 V		0 ... 10 V		± 1 V ± 2,5 V ± 10 V 0 ... 2 V 0 ... 10 V	± 10 V ± 50 mV ± 500 mV 1 ... 5 V ± 1 V ± 5 V 0 ... 10 V	± 1 V ± 5 mV ± 10 V 1 ... 5 V	± 25 mV, ± 50 mV, ± 80 mV, ± 250 mV, ± 500 mV, ± 1 V
Diagnosefähig	●			●			●		●	●
Alarmfähig	●			●			●		●	●
Gebrauchsfehler	± 1 %		± 0,1 %	± 0,1 %	± 0,9 %	± 0,7 %	± 0,15 %	± 0,6 %	± 0,4 %	± 0,12 %
Anzahl Kanäle	8	2	8	8	4	2	4	8	8	6
Potentialtrennung: Anzahl Gruppen	4	1	4	4	1	1	4	1	1	6
Auflösung	max. 14 Bit + VZ	max. 14 Bit + VZ	15 Bit + VZ	15 Bit + VZ	8 Bit	12 Bit + VZ	13 Bit + VZ	12 Bit + VZ	13 Bit + VZ	15 Bit + VZ
Wandlungszeit pro Kanal (bei 50 Hz)	22 ms	22 ms	65 ms	83 ms [2]	100 µs	85 ms	200 µs	60 ms	52 µs [1]	20 ms
Bestell-Nr.- Rumpf: 6ES7	331- 7KF0. [3]	331-7KB0. [3]	331-7NF0. [3]	331-7NF1. [3]	334-0CE0.	334-0KE0. [3]	335-7HG0.	331-1KF0. [3]	331-7HF0.	337-7PE1.

Baugruppentyp	Analogeingaben SM 33x								
Besonderheiten dieser Baugruppe	Universell einzusetzende Baugruppe, die alle gängigen Messbereiche abdeckt und damit die Ersatzteilhaltung wesentlich vereinfacht		Hochauflösende und hochgenaue Baugruppe zum Erfassen von Strömen und Spannungen		Preiswerte, universelle Mischbaugruppe zum Erfassen bzw. Ausgeben von Strömen und Spannungen	Mischbaugruppe für sehr schnelle Applikationen, wie z. B. Kunststoffmaschinen; integrierter Komperator	Universell einzusetzende Baugruppe, die alle gängigen Messbereiche abdeckt (keine TC-Messung) und damit die Ersatzteilhaltung vereinfacht	Sehr schnelle Baugruppe, die nach dem Prinzip der Momentanwertverschlüsselung arbeitet; geeignet für taktsynchrone Applikationen	Unterstützt kommunikation mit HART-fähigen Feldgeräten; hohe Kanaldichte und damit günstiger Preis
Strommessbereich Geber	± 3,2 mA, ± 10 mA, ± 20 mA, 0 ... 20 mA, 4 ... 40 mA		0 ... 20 mA 4 ... 20 mA ± 20 mA		0 ... 20 mA	± 10 mA 0 ... 20 mA 4 ... 40 mA	± 20 mA 0 ... 20 mA 4 ... 20 mA		± 20 mA 0 ... 20 mA 4 ... 20 mA HART
Anschlussart	2- und 4-Drahtmessumformer				4-Drahtmessumformer		2- und 4-Drahtmessumformer		
Diagnosefähig	●			●		●		●	●
Alarmfähig	●			●		●		●	●
Gebrauchsfehler	± 1 %		± 0,3 %	± 0,1 %	± 0,8 %	± 0,25 %	± 0,5 %	± 0,3 %	± 0,15 %
Anzahl Kanäle	8	2	8	8	4	4	8	8	8
Potentialtrennung: Anzahl Gruppen	4	1	4 (8)	4	1	1	1	1	1
Auflösung	max. 14 Bit + VZ	max. 14 Bit + VZ	15 Bit + VZ	15 Bit + VZ	8 Bit	13 Bit + VZ	12 Bit + VZ	13 Bit + VZ	15 Bit + VZ
Wandlungszeit pro Kanal (bei 50 Hz)	22 ms	22 ms	65 ms	83 ms[2]	100 µs	200 µs	60 ms	52 µs[1]	65 ms
Bestell-Nr.- Rumpf: 6ES7	331-7KF0. [3]	331-7KB0. [3]	331-7NF0. [3]	331-7NF1. [3]	334-0CE0.	335-7HG0.	331-1KF0. [3]	331-7HF0.	331-7TF0. [3]

[1] unabhängig von der eingestellten Störfrequenzunterdrückung

[2] im 4-Kanalmodus 10 ms

[3] als SIPLUS extreme-Komponente auch für erweiterten Temperaturbereich -25 ...+60/+70 °C und aggressive Atmosphäre/Betauung (Weitere Details siehe Seite 98 oder **www.siemens.de/siplus-extreme**)

S7-300 Analogeingaben

Baugruppentyp	Analogeingaben SM 33x				
Besonderheiten dieser Baugruppe	Universell einzusetzende Baugruppe, die alle gängigen Messbereiche abdeckt und damit die Ersatzteilhaltung wesentlich vereinfacht		Hochauflösende und hochgenaue Baugruppe für die Erfassung von Temperaturen über Widerstandsgeber (RTD) inkl. Kennlinienlinearisierung nach der russischen GOST-Norm	Universelle Mischbaugruppe zum Messen von Spannungen, Widerständen und Temperaturen über Widerstandgeber (RTD) sowie zum Ausgeben von Spannungen	Universell einzusetzende Baugruppe, die alle gängigen Messbereiche abdeckt (keine TC-Messung) und damit die Ersatzteilhaltung wesentlich vereinfacht
Widerstandsmessbereich Geber	150 Ω, 300 Ω , 600 Ω			10 kΩ	600 Ω, 6 kΩ
Anschlussart	2- /3- /4-Leiteranschluss				
Diagnosefähig	●				
Alarmfähig	●				
Gebrauchsfehler	± 1 %		± 0,1 %	± 3,5 %	± 0,5 %
Anzahl Kanäle	4	1	8	4	8
Potentialtrennung: Anzahl Gruppen	4	1	4	2	1
Auflösung	max. 14 Bit + VZ	max. 14 Bit + VZ	max. 15 Bit + VZ	12 Bit + VZ	12 Bit + VZ
Wandlungszeit pro Kanal (bei 50 Hz)	23 ms	23 ms	80 ms	170 ms	132 ms
Bestell-Nr.-Rumpf: 6ES7	331-7KF0. 3)	331-7KB0. 3)	331-7PF0. 3)	334-0KE0. 3)	331-1KF0. 3)

Baugruppentyp	Analogeingaben SM 33x								
Besonderheiten dieser Baugruppe	Universelle Mischbaugruppe zum Messen von Spannungen, Widerständen und Temperaturen über Widerstandgeber (RTD) sowie zum Ausgeben von Spannungen	Universell einzusetzende Baugruppe, die alle gängigen Messbereiche abdeckt und damit die Ersatzteilhaltung wesentlich vereinfacht		Hochauflösende und hochgenaue Baugruppe für die Erfassung von Temperaturen über Widerstandsgeber (RTD) inkl. Kennlinienlinearisierung nach der russischen GOST-Norm	Universell einzusetzende Baugruppe, die alle gängigen Messbereiche abdeckt und damit die Ersatzteilhaltung wesentlich vereinfacht		Hochauflösende und hochgenaue Baugruppe für die Erfassung von Temperaturen über Thermoelemente (TC) inkl. Kennlinienlinearisierung nach der russischen GOST-Norm	Universell einzusetzende Baugruppe, die alle gängigen Messbereiche abdeckt (keine TC-Messung) und damit die Ersatzteilhaltung wesentlich vereinfacht	Hochauflösende, einzelpotentialgetrennte Baugruppe für Spannungsmessung und Thermoelemente, zweikanalig redundant für die erhöhten Anforderungen in der Prozesstechnik
Temperaturmessbereich Geber	Pt 100 (-120 ... +130 °C)	Pt 100 Ni 100 (-200 ... +385 °C) jeweils Standard und Klima		Pt: 100; 200; 500; 1 000; Ni: 100; 120; 200; 500; 1 000; Cu 10 (-200 ... +850 °C und -120 ... +130 °C) 1)	Thermoelemente Typ E, N, J, K, L		Thermoelemente Typ B, C, E, N, J, K, L, R, S, T, U 2)	Pt 100 (-120... +130 °C); Ni 100; Ni 1 000; LG-Ni 1 000; (je Standard und Klima)	Thermoelemente Typ T, U, E, J, L, K, N, R, S, B, C, TxK, XK (L) 2)
Diagnosefähig		●							
Alarmfähig		●							
Gebrauchsfehler	± 1 %			± 0,1 %	± 1 %		± 0,1 %	± 1 %	± 0,15 %
Anzahl Kanäle	4	4	1	8	8	2	8	8	6
Potentialtrennung: Anzahl Gruppen	2	1	1	4	4	1	4	1	6
Auflösung	max. 14 Bit + VZ			15 Bit + VZ	max. 14 Bit + VZ		15 Bit + VZ	12 Bit + VZ	15 Bit + VZ
Wandlungszeit pro Kanal (bei 50 Hz)	170 ms	23 ms		80 ms	22 ms	44 ms	95 ms	110 ms	20 ms
Bestell-Nr.- Rumpf: 6ES7	334-0KE0. 3)	331-7KF0. 3)	331-7KB0. 3)	331-7PF0. 3)	331-7KF0. 3)	331-7KB0. 3)	331-7PF1. 4)	331-1KF0. 3)	331-7PE1.

1) Kennlinien nach GOST 6651-94

2) Kennlinien nach GOST P8.585.2001

3) als SIPLUS extreme-Komponente auch für erweiterten Temperaturbereich -25 ... +60/+70 °C und aggressive Atmosphäre/Betauung (Weitere Details siehe Seite 98 oder **www.siemens.de/siplus-extreme**)

4) als SIPLUS extreme-Komponente auch für aggressive Atmosphäre/Betauung (Weitere Details siehe Seite 98 oder **www.siemens.de/siplus-extreme**)

S7-300 Analogausgaben

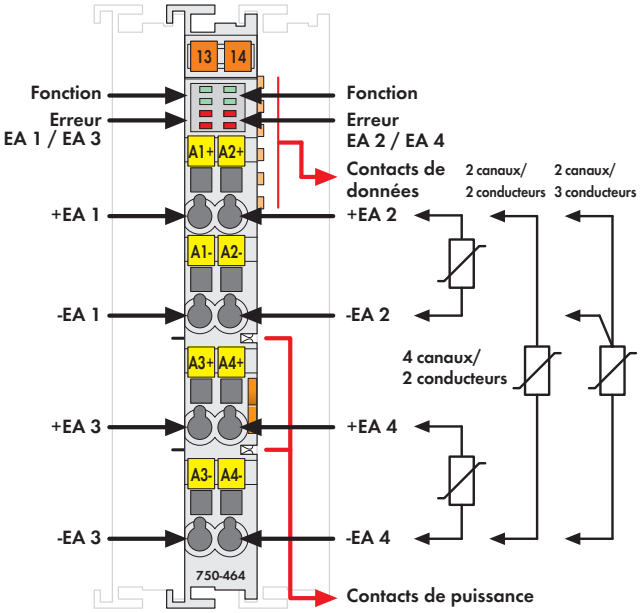
Baugruppentyp	Analogausgaben SM 33x						
Besonderheiten dieser Baugruppe	Universell einzusetzende Analogausgabe		Universell einzusetzende Analogausgabe; preiswert durch hohe kanaldichte	Sehr schnelle Baugruppe mit hoher Auflösung und Genauigkeit; geeignet für den taktsynchronen Betrieb	Preiswerte, universelle Mischbaugruppe zum Erfassen bzw. Ausgeben von Strömen und Spannungen	Universelle Mischbaugruppe zum Messen von Spannungen, Widerständen und Temperaturen über Widerstandgeber (RTD) sowie zum Ausgeben von Spannungen	Mischbaugruppe für sehr schnelle Applikationen, wie z. B. Kunststoffmaschinen; integrierter Komparator
Ausgabebereich	0 ... 10 V, 1 ... 5 V, ± 10 V				0 ... 10 V		0 ... 10 V ± 10 V
Diagnosefähig	●						●
Alarmfähig	●						●
Gebrauchsfehler	± 0,5 %			± 0,12 %	± 0,6 %	± 1 %	± 0,5 %
Anzahl Kanäle	2	4	8	4	2		4
Potentialtrennung: Anzahl Gruppen	1	1	1	4	1		1
Auflösung	11 Bit + VZ			max. 15 Bit + VZ	8 Bit	12 Bit	max. 13 Bit +VZ
Wandlungszeit pro Kanal	< 0,8 ms			> 1,6 ms	0,5 ms	0,5 ms	>0,8 ms
Bestell-Nr.-Rumpf: 6ES7	332-5HB0 [2]	332-5HD0. [1]	332-5HF0. [2]	332-7ND0. [1]	334-0CE0.	334-0KE0. [2]	335-7HG0.

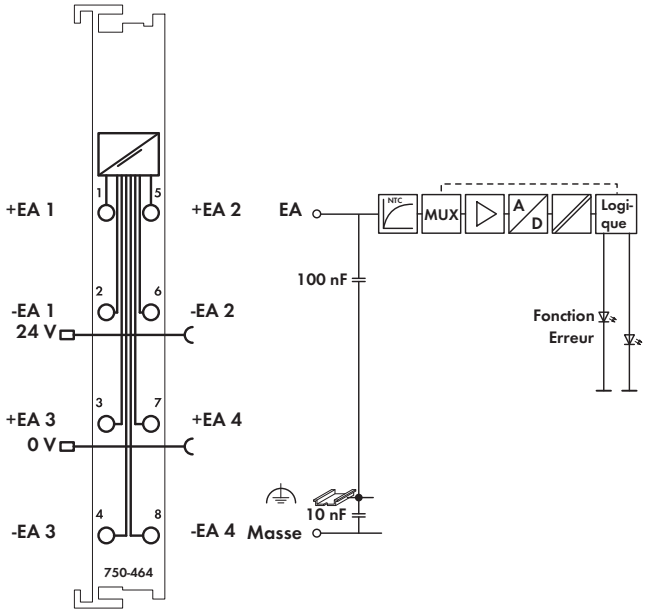
Baugruppentyp	Analogausgaben SM 33x					
Besonderheiten dieser Baugruppe	Universell einzusetzende Analogausgabe		Universell einzusetzende Analogausgabe; preiswert durch hohe kanaldichte	Sehr schnelle Baugruppe mit hoher Auflösung und Genauigkeit; geeignet für den taktsynchronen Betrieb	Preiswerte, universelle Mischbaugruppe zum Erfassen bzw. Ausgeben von Strömen und Spannungen	Unterstützt Kommunikation mit HART-fähigen Feldgeräten; hohe Kanaldichte und damit günstiger Kanalpreis
Ausgabebereich	± 20 mA, 0 ... 20 mA, 4 ... 20 mA				0 ... 20 mA	0 ... 20 mA HART 4 ... 20 mA HART
Diagnosefähig	●					●
Alarmfähig	●					●
Gebrauchsfehler	± 0,6 %			± 0,18 %	± 1 %	± 0,2 %
Anzahl Kanäle	2	4	8	4	2	8
Potentialtrennung: Anzahl Gruppen	1	1	1	4	1	1
Auflösung	11 Bit + VZ			max. 15 Bit + VZ	8 Bit	15 Bit + VZ
Wandlungszeit pro Kanal	< 0,8 ms			1,6 ms	0,5 ms	50 ms
Bestell-Nr.-Rumpf: 6ES7	332-5HB0. [2]	332-5HD0. [1]	332-5HF0. [2]	332-7ND0. [1]	334-0CE0.	332-8TF0. [2]

[1] als SIPLUS extreme-Komponente auch für aggressive Atmosphäre/Betauung (Weitere Details siehe Seite 98 oder **www.siemens.de/siplus-extreme**)

[2] als SIPLUS extreme-Komponente auch für erweiterten Temperaturbereich -25 ... +60/+70 °C und aggressive Atmosphäre/Betauung (Weitere Details siehe Seite 98 oder **www.siemens.de/siplus-extreme**)

Borne d'entrées analogiques à 2/4 canaux pour sondes de température (RTD)





Livraison sans Mini-WSB

Cette borne d'entrées analogiques permet une connexion directe de sondes de température Pt ou Ni, ainsi que des potentiomètres.
Elle peut être utilisée comme borne à 2 canaux (technique de raccordement à 2 et 3 fils) ou à 4 canaux (technique de raccordement à 2 fils)
La linéarisation sur toute la plage des températures est réalisée par un microprocesseur. La LED erreur rouge signale un court-circuit, l'interruption du circuit du capteur ainsi que tout dépassement de capacité.
La borne peut être configurée à l'aide de WAGO-I/O-CHECK et des fichiers GSD.
Elle se caractérise par de nombreuses possibilités de réglage par une précision élevée.

La version 750-464/020-000 permet la connexion de capteurs NTC.

Données techniques différentes pour la réf. 750-464/020-000 :
- Nombre d'entrées : 4
- Types de capteur : NTC 10 kOhm, NTC 20 kOhm, NTC 10 kOhm
- Connexion capteur : 2 conducteurs
- Plage de température : -30 °C ... +120 °C
- Erreur de mesure : ≤ 2 K sur toute la plage de température

Description	N° de produit	Unité d'emb.
2/4 AI RTD configuration libre	**750-464**	1
4 AI NTC configuration libre	**750-464/020-000**	1
Pour des données techniques différentes voir le texte descriptif		

Accessoires		N° de produit	Unité d'emb.
	**Connecteur, série 753**	**753-110**	25
	Eléments de codage	**753-150**	100
	**Système de repérage rapide Mini-WSB**		
	vierge	**248-501**	5
	avec impression	voir pages 352 ... 353	

Approbations	Voir aussi aperçu des approbations dans le chapitre 1
Marquage de conformité	CE
Applications Marine	
(variantes sur demande)	ABS, DNV, GL, KR
UL 508	En préparation
ANSI/ISA 12/12/01	en préparation
EN 60079-0, -15	en préparation
EN 61241-0, -1	

Données techniques	
Nombre d'entrées	2/4 (préréglage)
Alimentation	par système interne DC/DC
Consommation de courant typ. (interne)	50 mA
Types de capteur	Pt100 (préréglage), Pt200, Pt500, Pt1000, Ni 100, Ni 120, Ni 1000, potentiomètre, 10 Ohm ... 1,2 kOhm, 10 Ohm ... 5 kOhm
Type de raccordement	2 conducteurs (préréglage), 3 conducteurs (mode 2 canaux)
Plage de température	-200 °C ... +850 °C (Pt100), -60 °C ... +300 °C (Ni 100, Ni 1000), -60 °C ... +250 °C (Ni 1000 TK5000), -80 °C ... +260 °C (Ni 120)
Résolution (sur la pleine échelle)	0,1 °C
Taux de répétition de mesure (standard)	1,1s
Taux de répétition de mesure (2 canaux/2conducteurs)	0,63s
Retard d'activation max.	4 s
Erreur de mesure 25 °C	≤ 1 K dans toute la plage de température, ≤ 0,5 K dans une plage de température réduite* *-30 °C ... +120 °C
Coefficient de température	≤ 20 ppm/K
Séparation galvanique	500 V (système/alimentation)
Courant de mesure typ.	≤ 350 µA
Unité d'adressage	4 (2) x 16 Bits données, 4 (2) x 8 Bits contrôle/état (optionnel)
Type de connexion	CAGE CLAMP®
Sections	0,08 mm² ... 2,5 mm² / AWG 28 ... 14
Longueurs de dénudage	8 ... 9 mm / 0.33 in
Dimensions : largeur	12 mm
Poids	47,3 g
CEM : CE - susceptibilité en réception	selon EN 61000-6-1 (2007), EN 61000-6-2 (2005)
CEM : CE - en émission	selon EN 61000-6-3 (2007)

Borne d'entrées analogiques à 2 canaux pour sondes de température (RTD)

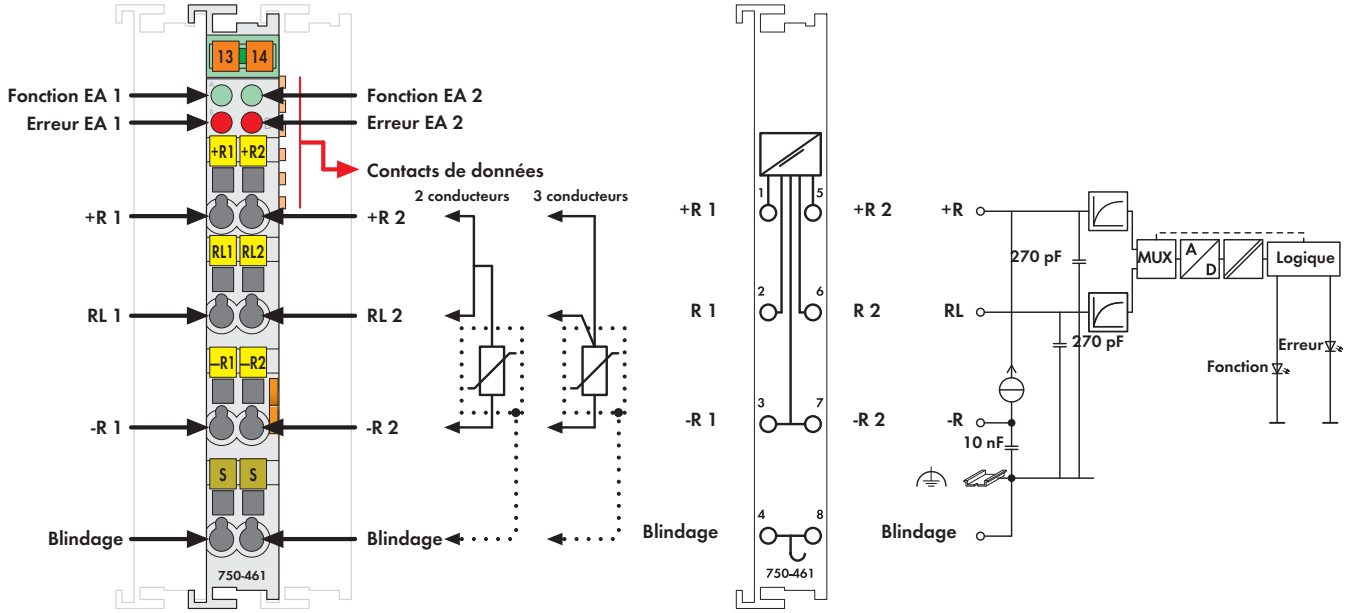


Illustration : série 750 / représentation voir page 24 /
Livraison sans Mini-WSB, repérage série 750 / 753, voir pages 10 ...11 / 12 ...13

Cette borne d'entrées analogiques permet une connexion directe de sondes de température Pt ou Ni.
La connexion est réalisée avec des sondes à 2 ou 3 fils.
La linéarisation sur toute la plage des températures est réalisée par un microprocesseur. La LED erreur rouge signale un court-circuit, l'interruption du circuit du capteur ainsi que tout dépassement de capacité.
La LED verte signale la disponibilité du service et la libre communication avec le coupleur du bus de terrain.
Les bornes de raccordement du blindage sont directement reliées au rail.

La variante configuration libre supporte tous les types de capteur mentionnés. Réglage par l'intermédiaire du logiciel WAGO-I/O-CHECK.

Pour les sondes suivantes une livraison d'autres versions de la borne de bus est possible rapidement :
Pt100, Pt200, Pt500, Pt1000, plage de température -200 °C ... + 850 °C
Ni 100, Ni 100, plage de température -60 °C ... +250 °C
Mesure de résistance
Données techniques différentes pour la réf. 750-461/020-000 :

- Consommation de courant max. (interne) : 65 mA
- Type de capteur : NTC 20 kOhm
- Plage de température : -30 °C ... +130 °C
- Erreur de mesure : 0,5 K ... 3,0 K (selon la température)
- Coefficient de température : < +/- 0,002 %/K de la pleine échelle
- Courant de mesure typ. : 0,05 mA à 25 °C

Description	N° de produit	Unité d'emb.
2AI Pt 100/RTD	**750-461**	10[1]
2AI mesure de résistance 10R-1k2	**750-461/000-002**	10[1]
2AI Pt 1000/RTD	**750-461/000-003**	10[1]
2AI Ni 100/RTD	**750-461/000-004**	10[1]
2AI Ni 1000 TK6180/ RTD	**750-461/000-005**	10[1]
2AI mesure de résistance 10R-5k0	**750-461/000-007**	10[1]
2AI Ni 1000 TK5000/ RT	**750-461/000-009**	1
2AI Pt 100/RTD S5 [2]	**750-461/000-200**	10[1]
2AI Pt 100/configuration libre	**750-461/003-000**	10[1]
2AI NTC 20k	**750-461/020-000**	1
Pour des données techniques différentes voir le texte descriptif		
2AI Pt100/RTD/T	**750-461/025-000**	1
(Température de fonctionnement -20 °C ... +60 °C)		
2AI Pt 100/RTD (sans connecteur)	**753-461**	1
2AI Pt 100/configuration libre (sans connecteur)	**753-461/003-000**	1

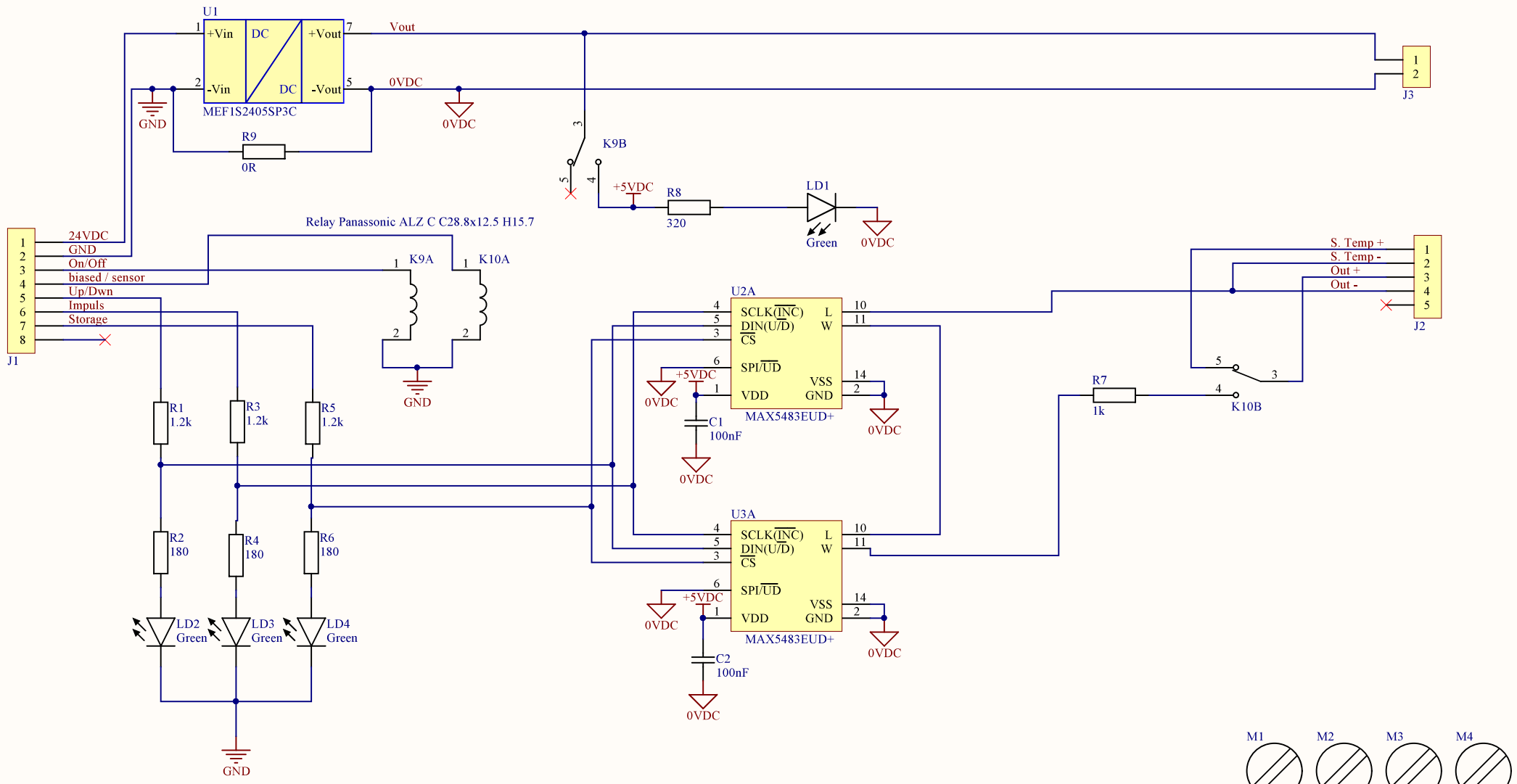
[1] Une livraison de pièces individuelles est également possible
[2] Ce format est approprié au système S5 avec le bloc fonctionnel FB 250.

Approbations	Voir aussi aperçu des approbations dans le chapitre 1	
Marquage de conformité	CE	
Applications Marine	ABS, BV, DNV, GL, KR, LR*, NKK*, PRS*, RINA*	
(variantes sur demande)	*Série 753 en préparation	
cULus UL 508		
cULus ANSI/ISA 12/12/01	Class I Div2 ABCD T4	750-461,-/00x-xxx, -461/020-000
		753-461, -461/...
ATEX EN 60079-0, -15	I M2 / II 3 GD Ex nA IIC T4	750-461/0x0-xxx
EN 61241-0, -1		753-461, -461/...
ATEX EN 60079-0, -11, -15	I M2 Ex d I	750-461*
EN 61241-0, -1, -11	II 3 G Ex nA IIC T4	750-461*
	II 3 D Ex tD A22 IP6X T135 °C	750-461*
	* Température de fonctionnement autorisée 0 °C ... +60 °C	

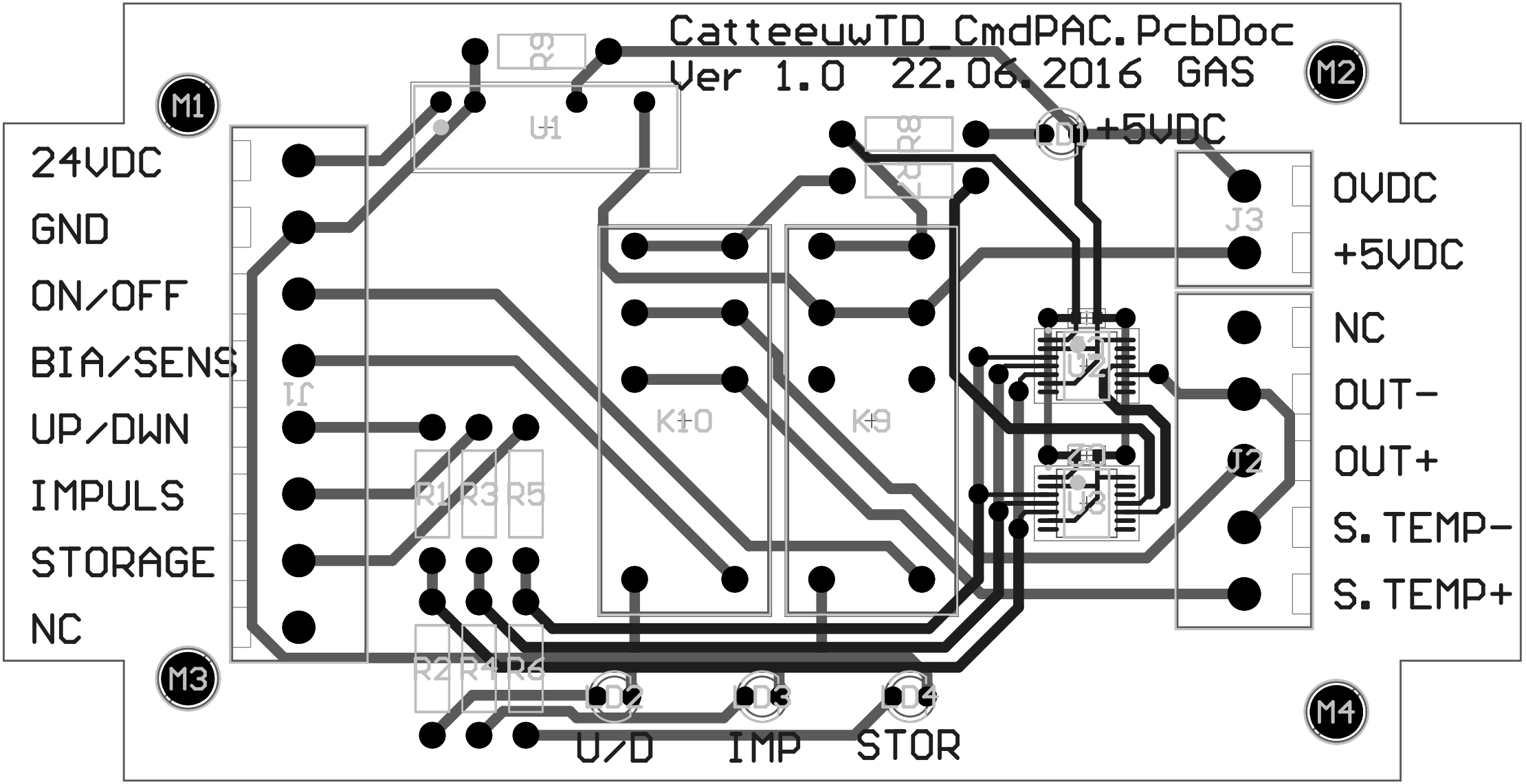
Données techniques	
Nombre d'entrées	2
Alimentation	par système interne DC/DC
Consommation de courant typ. (interne)	80 mA
Types de capteur	Pt100 (version de base), en option version disponible pour Pt200, Pt500, Pt1000, Ni 100, Ni 120, Ni 1000, mesure de résistance
Type de raccordement	2 ou 3 conducteurs (préréglage)
Plage de température	-200 °C ... +850 °C (Pt), -60 °C ... +250 °C (Ni)
Résolution (sur la pleine échelle)	0,1 °C
Temps de conversion	320 ms (par canal)
Retard d'activation max.	4 s
Erreur de mesure 25 °C	< ± 0,2 % de la pleine échelle
Coefficient de température	< ± 0,01 % /K de la pleine échelle
Séparation galvanique	500 V (système/alimentation)
Courant de mesure typ.	0,5 mA
Unité d'adressage	2 x 16 bits (données)
	2 x 8 bits (contrôle/état) (optionnel)
Type de connexion	CAGE CLAMP®
Sections	0,08 mm² ... 2,5 mm² / AWG 28 ... 14
Longueurs de dénudage de la série 750 / 753	8 ... 9 mm / 0.33 in 9 ... 10 mm / 0.37 in
Dimensions : largeur	12 mm
Poids	52,5 g
CEM : CE - susceptibilité en réception	selon EN 61000-6-2 (2005)
CEM : CE - en émission	selon EN 61000-6-4 (2007)
CEM : Marine - susceptibilité en réception	selon Germanischer Lloyd (2003)
CEM : Marine - en émission	selon Germanischer Lloyd (2003)
Accessoires	Connecteur, série 753, Eléments de codage, Système de repérage rapide Mini-WSB

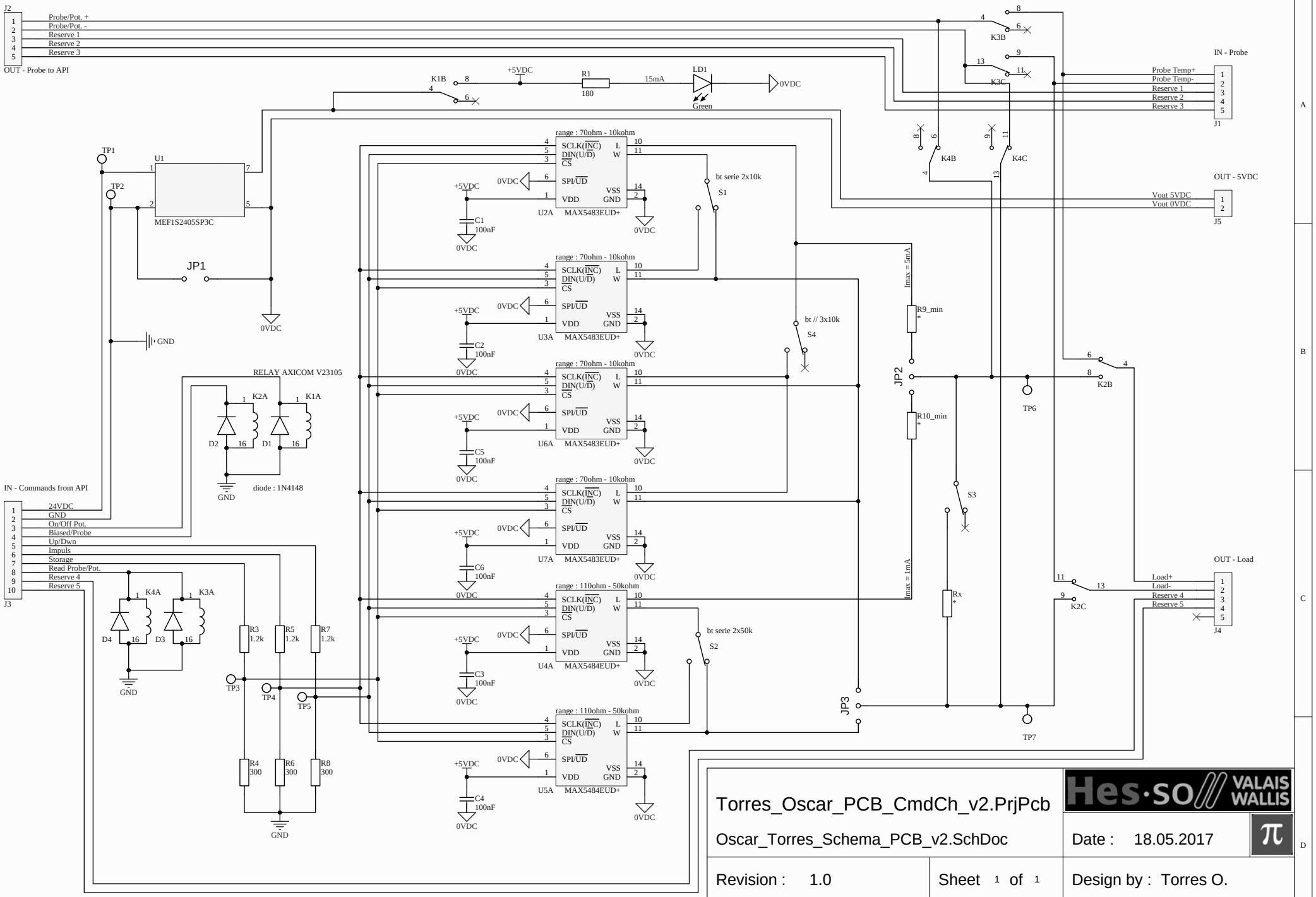
Annexe 2 : Schémas électroniques PCB

a. Schéma électronique, PCB Jérôme Catteeuw v1, Travail de diplôme 2016
b. Schéma électronique, PCB Oscar Torres v1
c. Schéma électronique, PCB low-cost Oscar Torres v1



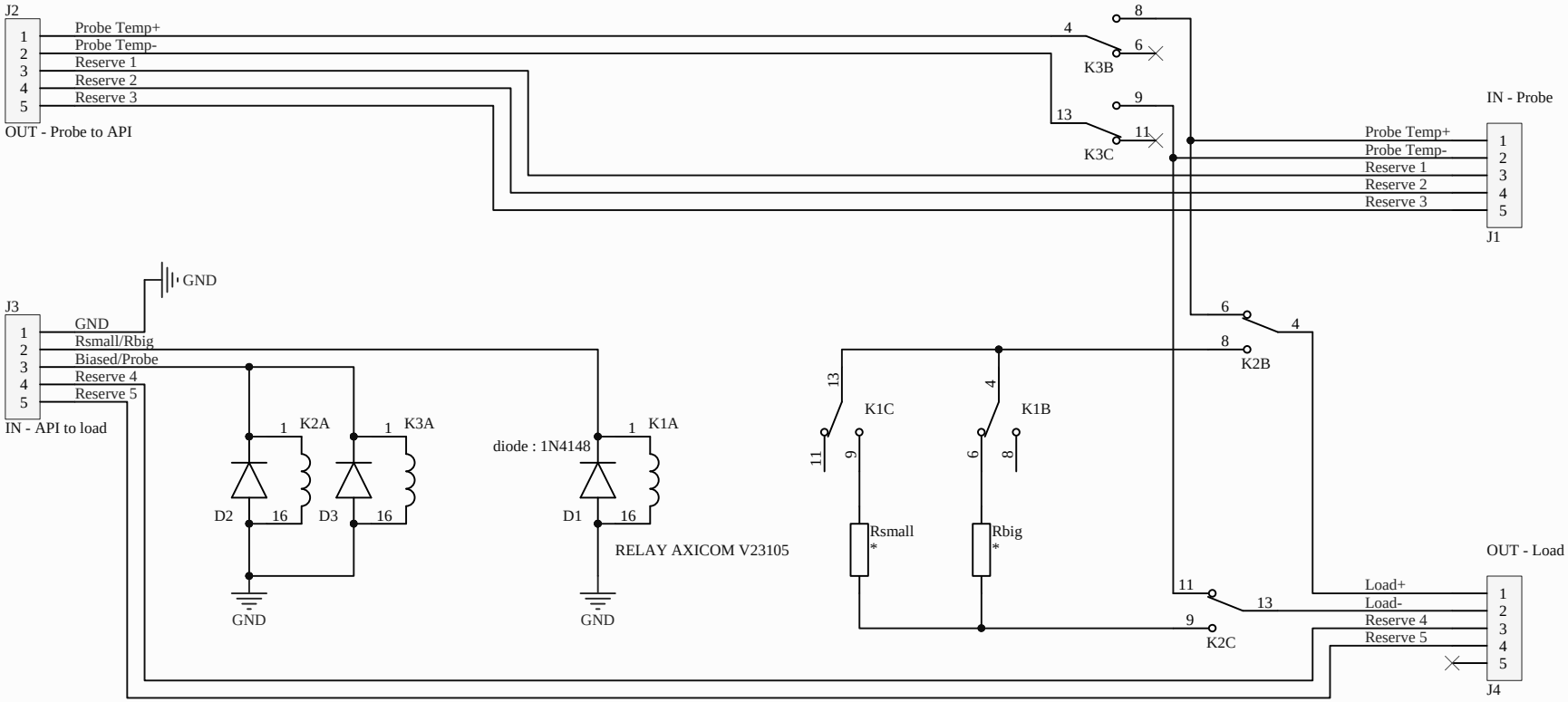
CatteeuwTD_CmdPAC.PrjPcb		Hes·so VALAIS WALLIS
CatteeuwTD_CmdPAC.SchDoc		Date : 16.06.2016
Revision : 1.0	Sheet 1 of 1	Design by : Catteeuw Jérôme
P:\PCB\Student\TSTD\Catteeuw Jerome\CatteeuwTD_CmdPAC.SchDoc		


CatteeuwTD_CmdPAC.PcbDoc
Ver 1.0 22.06.2016 GAS
M1
M2
M3
M4
24VDC
GND
ON/OFF
BIA/SENS
UP/DWN
IMPULS
STORAGE
NC
+5VDC
0VDC
+5VDC
NC
OUT-
OUT+
S.TEMP-
S.TEMP+
U/D
IMP
STOR



Note : Rx, R9_min and R10_min to choose

Torres_Oscar_PCB_CmdCh_v2.PrjPcb		Hes·SO VALAIS WALLIS
Oscar_Torres_Schema_PCB_v2.SchDoc		Date : 18.05.2017
Revision : 1.0	Sheet 1 of 1	Design by : Torres O.
R:\Diploma\TD2017\SYND\oscar.torres\modif_altium_AND_low-cost\PCB_Torres\Oscar_Torres_Schema_PCB_v2.SchDoc		



Note : Rsmall and Rbig to choose

Title	***Oscar_Torres_PCB_lowcost_v1***		* *HES-SO Valais* *Route du Rawyl 47* *1950 Sion* *Switzerland*
Size: A4	Number:1	Revision:*	
Date: 09.08.2017	Time: 16:25:45	Sheet 1 of 1	
File: R:\Diploma\TD2017\SYND\oscar.torres\modif_altium_AND_low-cost\PCB_Torres\Oscar_Torres_Schema_PCB_lowcost.Sch			

Annexe 3 : Carte électronique PCB Oscar Torres v1 - Hardware

a. Mode d’emploi de la carte électronique
b. Liste de matériel de la carte électronique
c. Protocole de test

Mode d’emploi

Carte électronique : O. Torres_TD2017_Cmd.Ch._v1

Avant branchement

IMPORTANT : Tous ces paramétrages doivent être fait par un spécialiste en électricité/électronique/automation. Si une erreur de paramétrage est faite, la valeur de la résistance de sortie ne sera pas celle attendue !

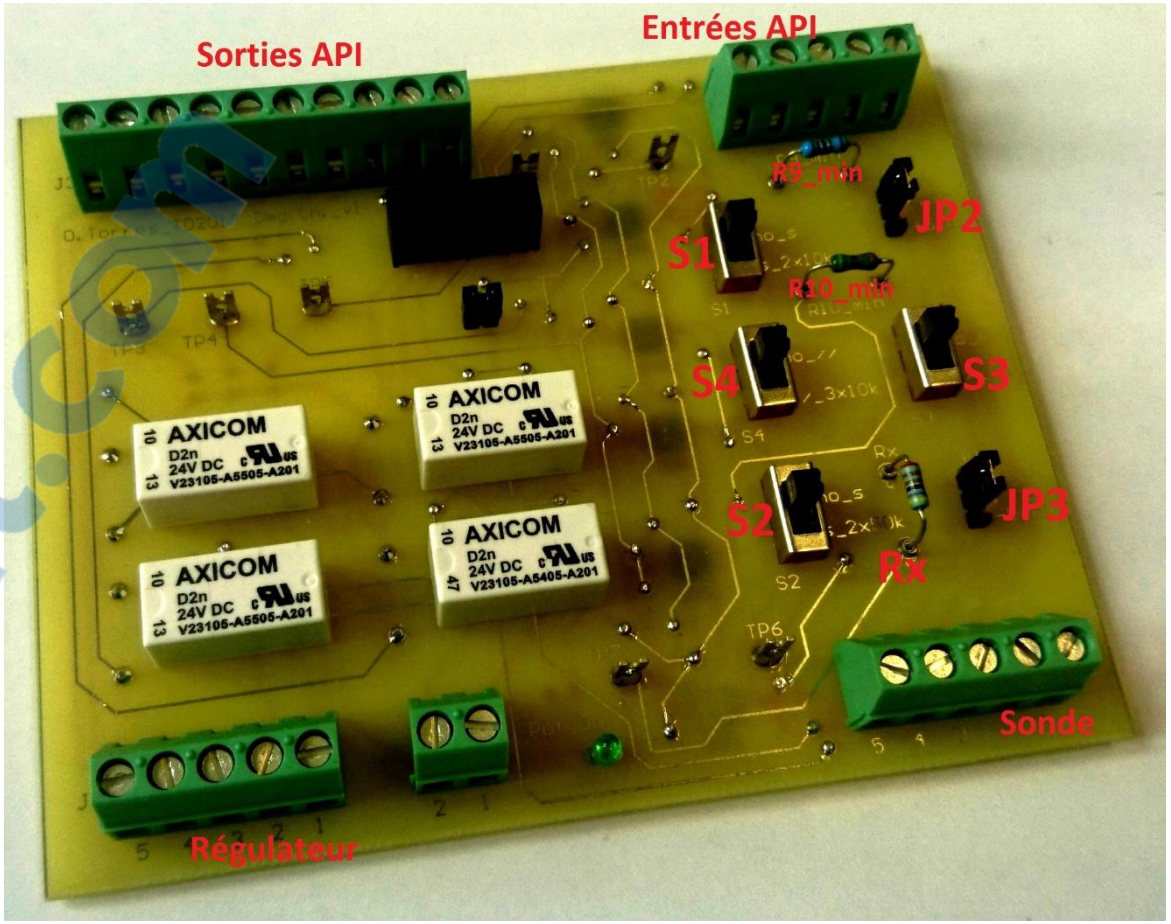
Remarque : pour tout le présent mode d’emploi, API veut dire Automate programmable industriel.

Tout d’abord, il faut paramétrer manuellement la carte électronique PCB en fonction de la sonde extérieure que vous allez utiliser.

Les sondes les plus communément utilisées : (pour température extérieure -20°C … 50°C)

Sonde	Type de fonctionnement à utiliser
PT100	F
PT500	C
PT1000	C
Ni1000 (TK5000 ou TK6180)	C
Lg-Ni1000	C
NTC 1k	A
NTC 2k	B
NTC 10k	E
NTC 20k	E

Fonctionnements :



- **Fonctionnement A : de (R9_min + 70Ω) à (R9_min + 10kΩ)**
  - S1 en position haute
  - S2 en position haute
  - S3 en position haute
  - S4 en position haute
  - JP2 en position haute
  - JP3 en position haute
  - R9 _min selon la tension du régulateur (expliqué dans le chapitre : Type de régulateur)
- **Fonctionnement B : de (R9_min + 140Ω) à (R9_min + 20kΩ)**
  - S1 en position basse
  - S2 en position haute
  - S3 en position haute
  - S4 en position haute
  - JP2 en position haute
  - JP3 en position haute
  - R9 _min selon la tension du régulateur (expliqué dans le chapitre : Type de régulateur)
- **Fonctionnement C : de (R9_min + 23Ω) à (R9_min + 3333Ω)**
  - S1 en position haute
  - S2 en position haute
  - S3 en position haute
  - S4 en position basse
  - JP2 en position haute
  - JP3 en position haute
  - R9 _min selon la tension du régulateur (expliqué dans le chapitre : Type de régulateur)

- **Fonctionnement D : de (R10_min + 110Ω) à (R10_min + 50kΩ)**
    - S1 en position haute
    - S2 en position haute
    - S3 en position haute
    - S4 en position haute
    - JP2 en position basse
    - JP3 en position basse
    - R10 _min selon la tension du régulateur (expliqué dans le chapitre : Type de régulateur)

- **Fonctionnement E : de (R10_min + 220Ω) à (R10_min + 100kΩ)**
    - S1 en position haute
    - S2 en position basse
    - S3 en position haute
    - S4 en position haute
    - JP2 en position basse
    - JP3 en position basse
    - R10 _min selon la tension du régulateur (expliqué dans le chapitre : Type de régulateur)
- **Fonctionnement F : Biaiser la température avec une résistance fixe à choix**
    - S1 en position haute
    - S2 en position haute
    - S3 en position haute
    - S4 en position haute
    - JP2 non branché
    - JP3 non branché
    - Rx à choix

Type de régulateur

Les valeurs de résistance R9_min et R10_min dépendent du type de régulateur et de la tension qu'il applique pour lire le courant qui y revient et en déduire la résistance lue !

Cette résistance se calcule comme suit :

- $R9\_\min\,[\Omega] = \frac{U\_reg[V]}{I\_\max[A]}$

Un exemple pour illustrer le calcul ci-dessus :

Pour un régulateur fournissant une tension de 5V, les résistances minimales R9_min et R10_min sont les suivantes :

Choix possibles :

						exemple : 5V	R9 ou R10	normalisé E24	**plage de fonctionn.**		
			[Ω]	[Ω]	[mA]	[V]	[Ω]	[Ω]	[Ω]	[Ω]	
	Nbr pot.	**branchem.**	**val. min**	**val.max**	**I_max**	**U_reg**	**R_min**	**min_norm**	**R_min**	**R_max**	**FCT**
potentiomètres 0-10kΩ	1		70	10070	5	5	1000	1000	1070	11070	**A**
	2	série	140	20140	5	5	1000	1000	1140	21140	**B**
	3	//	23,3	3356,7	15	5	333,3	360	383,3	3716,7	**C**
potentiomètres 0-50kΩ	1		110	50110	1	5	5000	5100	5210	55210	**D**
	2	série	220	100220	1	5	5000	5100	5320	105320	**E**

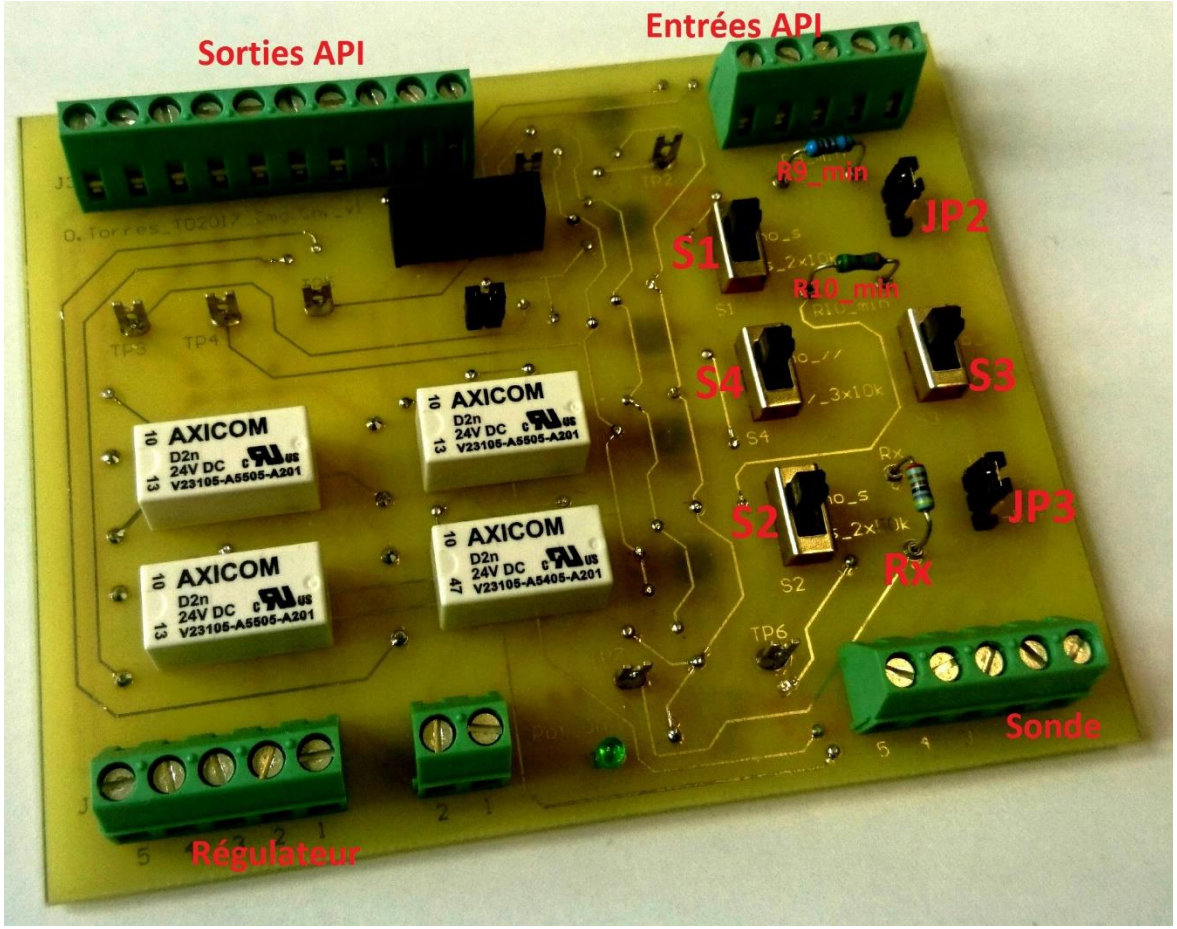
Remarque : Les valeurs minimales des potentiomètres sont dues aux résistances internes

R_min et R_max (en rouge) sont les valeurs à insérer dans le bloc de fonction « *FB_ControlPCB_Heating* » qui est dans l'API.

FCT (en bleu) sont les modes de fonctionnement expliqués dans le chapitre : Modes de fonctionnement

Branchement

Comme le démontre la figure à droite :



- La sonde se branche sur le bornier J1, bornes 1 et 2.
- Le régulateur sur le bornier J4, bornes 1 et 2.
- L'entrée analogique RTD (Résistance variant en fonction de la température) de l'API sur le bornier J2.
    - AI1 bornes 1 et 2.
- Les sorties digitales de l'API sur le bornier J3, bornes 3 à 8.
    - DO1 borne 3 Potentiomètres On/Off
    - DO2 borne 4 Régulation : faussée/normale
    - DO3 borne 5 Direction impulsions : Up/Down
    - DO4 borne 6 Impulsions
    - DO5 borne 7 Enregistrement valeur potent.
    - DO6 borne 8 Lecture : sonde/potentiomètre
- L'alimentation de la carte électronique 24VDC sur le bornier J3.
    - 24 VDC sur la borne 1.
    - 0 VDC sur la borne 2.

Tests avec alimentation

Des points de test sont prévus pour tester le bon fonctionnement de la carte électronique :

- TP1 et TP2 24 VDC
- TP3 Enregistrement valeur potent.
- TP4 Impulsions
- TP5 Direction impulsions : Up/Down
- TP6 et TP7 Valeur potentiomètres [Ω]

Remarques générales :

- Si besoin, 5VDC sont disponibles sur le bornier J5 !
- La LED verte est allumée lorsque les potentiomètres sont alimentés.
- Les autres bornes servent de réserve.
- Veuillez regarder le schéma électrique de la carte électronique pour plus d'informations !

Liste de matériel : PCB TD Oscar Torres v1

Version 1.1, 18.08.2017

Carte PCB :

www.mouser.ch

composant	référence	fabricant	nbr/ carte	prix/pce CHF	Prix/carte
plaque électronique de type PCB			**1**	**5**	5,00
relais 2 contacts	V23105-A5505-A201	Axicom	4	2,58	10,32
transfo 24VDC-5VDC	MEF1S2405SP3C	Murata	1	5,68	5,68
potentiomètre électronique 10k	MAX5483EUD+	Maxim integrated	4	2,74	10,96
potentiomètre électronique 50k	MAX5484EUD+	Maxim integrated	2	2,74	5,48
connecteur bornier 2 entrées	1935161	phoenix contact	1	0,39	0,39
connecteur bornier 5 entrées	1753064	phoenix contact	3	1,93	5,79
connecteur bornier 10 entrées	1989528	phoenix contact	1	1,74	1,74
commutateur on-on	MFP-221 N	knitter	3	0,77	2,31
résistance			10	0,10	1,00
condensateur			4	0,10	0,40
jumper			3	0,05	0,15
point de mesure			7	0,04	0,29
				Total/carte (CHF)	49,51

Boitier :

order.ch@wago.com

composant	référence	fabricant	nbr/carte	prix/pce	
support pcb (taille : 1m)	288-620	wago	0,15	14,00	2,10
pied DIN	288-622	wago	2	0,35	0,70
Extrémité latérale	288-626	wago	2	0,60	1,20
Couvercle transparent (taille : 1m)	288-627	wago	0,15	18,00	2,70
				Total/carte (CHF)	6,70

Frais de ports divers			Total/carte (CHF)	5

TOTAL général (CHF)	**61,21**

Consommable :

visserie

étain (soudure)

Protocole de tests PCB

Travail de diplôme 2017 - Oscar Torres

Contrôle	Validité	Commentaires
Avant soudures		Vias et pins de connexion de résistances non-SMD soudés
Contrôle visuel	ok	
Test ohmmètre connection 5 VDC	ok	
Test ohmmètre connection GND	ok	
Test ohmmètre connection 0 VDC	ok	
Test ohmmètre GND - 0 VDC	ok	
Test ohmmètre GND - 24 VDC	ok	
Test ohmmètre 0 VDC - 5 VDC	ok	
Test ohmmètre entrées entre elles : pas de CC	ok	
Test ohmmètre sorties entre elles : pas de CC	ok	
Après soudures		Tous les composants sont soudés
Test ohmmètre 0.6 kΩ entre pin 3 et 4 du potentiomètre	ok	
Test ohmmètre 0.6 kΩ entre pin 4 et 5 du potentiomètre	ok	
Test ohmmètre 0.6 kΩ entre pin 3 et 5 du potentiomètre	ok	
Test ohmmètre pin 2-6-14 potentiomètre	ok	
Test ohmmètre Rx en sortie	ok	4.69k avec résistance de 4.7k posée
Test ohmmètre 10 kΩ seul	ok	6.39k
Test ohmmètre 2x10 kΩ en série	ok	11.56k
Test ohmmètre 3x10 kΩ en //	ok	2.94k
Test ohmmètre 50 kΩ seul	ok	31.83k

Test ohmmètre 2x50 kΩ en série	ok	56.7k
Test ohmmètre pistes de réserves	ok	

Tests avec alimentations

J2 - 5 VDC	ok	5.01V
enlever JP1 et mesurer entre GND et 0 VDC	ok	0.4V => On remarque que ce jumper est indispensable si l'on veut mettre le GND et le 0VDC au même potentiel !
relais K1 K2 K3 K4	NON !	Les relais ont mal été configurés dans altium => Inversion des numéros de pates 6 <-> 8 ainsi que les 9 <-> 11 => M. Steve Gallay a changé de relais sur ma carte électronique pour qu'elle puisse fonctionner. => A changer pour la version 2 du PCB

Après modifications M. Steve Gallay

commutation K1 et contrôle des contacts	ok	
commutation K2 et contrôle des contacts	ok	
commutation K3 et contrôle des contacts	ok	
commutation K4 et contrôle des contacts	ok	
Led verte allumée si K1 est enclenché	ok	
sortie direction régulateur : bornier J4, bornes 1 et 2	ok	si K2 désactivé, sortie sonde de température ; si K2 activé, sortie potentiomètres
entrée API : bornier J2, bornes 1 et 2	ok	si K3 et K4 désactivé, entrée sonde de température ; si K3 et K4 activé, entrée potentiomètres

Remarque : K2, K3 et K4		Si K2, K3 et K4 désactivés en même temps, la sonde sera lue par l'API en même temps qu'elle sera appliquée au régulateur, ce qui pourrait induire une erreur lors de la mesure. Donc les relais K3 et K4 devront être forcés à 1 lors du démarrage du programme automate => changer pour la version 2 le branchement des contacts des relais K3 et K4. De cette manière,lorsqu'ils sont désactivés, l'API lit les potentiomètres et la sonde est appliquée au régulateur.

La programmation de la carte électronique peut dès lors commencer !

Annexe 4 : Programmation API – Pilotage de chauffages

a. Tutoriel de programmation pour une sonde de température différente de la PT 1000
b. Programme principal (Main program) : *Control_Heating*
c. Bloc de fonction principal : *Fb_ControlPCB_Heating*
d. Blocs de fonctions de conversion de valeurs : Conversion ohm <-> degrés
e. Programme de test du code : *Test_the_code*

Tutoriel programmation

Projet : ControlPCB.pro CODESYS-Project

Table des matières

Introduction

Ce tutoriel démontre comment reprendre le code selon le type de chauffage et type de sonde désiré.

Vue globale

Codesys contient des Programmes (PRG), des blocs de fonction (FB) et des fonctions (FU).

Pour ce projet, *PLC_PRG* n'est pas utilisé, tous les autres modules sont utilisés.

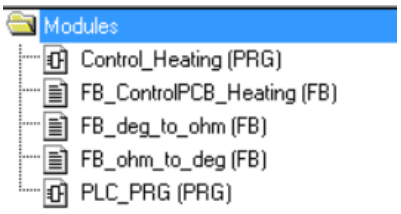


Figure 1 : arborescence projet

Control_Heating est le programme principal qui contient :

- Le bloc de fonction *FB_ControlPCB_Heating*.
- Les conversions entre les [°C] et les [Ω] avec les blocs de fonctions *FB_deg_to_ohm* et *FB_ohm_to_deg*.
- Les liaisons entre les variables utilisées dans le code et les entrées/sorties

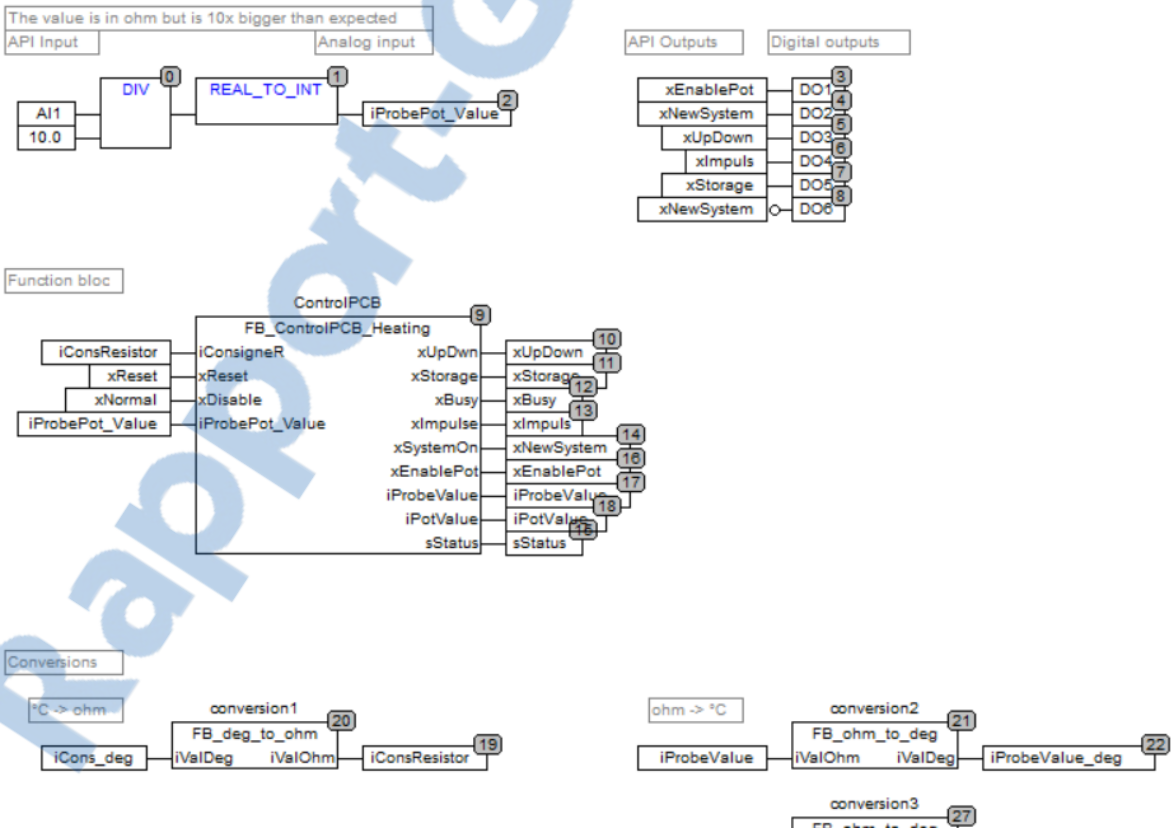


Figure 2 : Control_Heating (PRG)

Changement de sonde

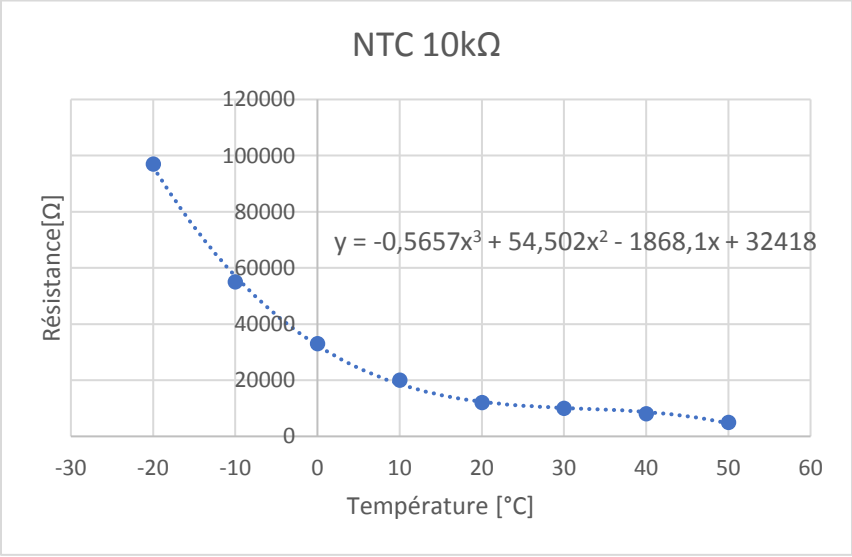
Ci-dessous il y a un tableau avec quelques sondes existant sur le marché :

Temp °C	PT100 Ω	PT1000 Ω	Ni1000 Ω	Ni1000 TK5000 Ω	NTC 1kOhm Ω	NTC 2kOhm Ω	NTC 5kOhm Ω	NTC 10kOhm kΩ	NTC 20kOhm kΩ	KTY 81-210 Ω
-50	80	803	743	791	32886	77977	333914	668	1668	1030
-40	84	843	791	831	18641	43040	167835	336	813	1135
-30	88	882	842	872	10961	24651	88342	177	415	1247
-20	92	922	893	913	6662	14615	48487	97	221	1367
-10	96	961	946	956	4175	8947	27649	55	122	1495
0	100	1000	1000	1000	2961	5642	16325	33	70	1630
10	104	1039	1056	1045	1781	3657	9952	20	42	1772
20	108	1078	1112	1091	1205	2431	6247	12	25	1922
25	110	1097	1141	1114	1000	2000	5000	10	20	2000
30	112	1117	1171	1138	834	1655	4028	8	16	2080
40	116	1155	1230	1186	589	1151	2662	5	10	2245
50	119	1194	1291	1235	424	816	1800	4	7	2417
60	123	1232	1353	1285	310	590	1244	2	5	2597
70	127	1270	1417	1337	231	434	876	2	3	2785
80	131	1309	1483	1390	175	324	628	1	2	2980
90	135	1347	1549	1444	134	246	458	1	2	3162
100	139	1385	1618	1500	104	189	339	1	1	3392
110	142	1422	1688	1557	81	147	255	1	1	3607
120	146	1461	1760	1615	65	116	194	0	1	3817
130	150	1498	1883	1675	52		150	0	0	4008
140	154	1536	1909	1736	42		117	0	0	4166
150	157	1573	1987	1799	34		92	0	0	4280

Figure 3: Liste de sondes de températures

Source : http://lampopumput.info/foorumi/index.php?topic=18236.15

Exemple pour un changement de sonde, admettons qu'une sonde NTC 10k soit utilisée :



Temp [°C]	NTC 10kΩ [Ω]
-20	97000
-10	55000
0	33000
10	20000
20	12000
30	10000
40	8000
50	5000

Figure 4 : Graphique sonde NTC 10k-ohm

Pour la conversion, il faudra faire la fonction de la courbe dans le bloc *FB_deg_to_ohm* et l'inverse de cette fonction dans le bloc *FB_ohm_to_deg.*

L'inverse d'une fonction :

- Les + sont changés par des – et inversément
- Les * sont changés par des /
- Les opérations sont inversées.

Exemple : Si une fonction fait y=x*2+6, l'inverse fait x=(y-6)/2 ;

FB_ControlPCB_Heating

Variables constantes

La première chose à faire lors de changement de sonde est de changer les variables constantes du bloc de fonction :

```
VAR CONSTANT
    xResetAtStart              : BOOL := TRUE;          (* If TRUE, automatic reset in case of power outage *)
    R_resolution               : INT := 1024;           (* 10 bits potentiometers -> 2^10=1024 *)
    udiTimeBetweenReadProbe : UDINT := 7200000;         (* Updates Potentiometer value when no consigne changes every 2h *)
                                                        (* 2h = 1000ms*60*60*2 = 7'200'000ms*)

    R_pas                      : REAL := 3.232422;      (* ohm *)       (*here : 3.232422 = (3333ohm - 23 ohm) / 1024*)
                                                        (*Change value before running program for the first time with the step value*)

    iCons_beginn               : INT := 1060;           (* ohm *)   (*PT1000 : 15°C = 1060 ohm*)
                                                        (*Change value before running program for the first time with the real 15°C value of the probe*)

    R_min                      : INT := 844;            (* ohm *)   (*Minimum PCB range selected : (R9_min or R10_min)+ RminPot *)
                                                        (* Change value before running program for the first time with your calculated minimum resistor*)
                                                        (*pot10k : RminPot = 70 ohm, pot50k : RminPot = 110ohm*)    (*here : 820 ohm + 23 ohm = 843 ohm*)

    R_max                      : INT := 1196;           (* ohm *)   (*here : 1196 ohm = 50 °C*)
                                                        (*Change value before running program for the first time with the real max value of the AI card*)
                                                        (* real max : 1200 but you have to let the last degree[°C] in case of rupture of probe cable !
                                                        Here 1°C = 3.9 ohm ! If rupture : the probe become the max value (Here = 1200) *)
END_VAR
```

Figure 5 :Variables constantes

Les variables à changer sont : *R_pas*, *iCons_beginn*, *R_min* et *R_max*

Exemple : Si l'on utilise 2 potentiomètres 50 kΩ en série :

R_min :

- Calculée avec la tension que le régulateur fournit pour lire le courant qui revient et en déduire la résistance.
- Admettons que cela soit une tension de 5V
  - I_max_pot50k-ohm = 1mA
  - R_min = U/I = 5/0.001 = 5000 Ω
  - R_min_interne_pot = 2*110 Ω = 220 Ω
  - R10_min = 5000-220 = 4780 Ω => E24 séries => 5600 Ω
  - R_min à mettre dans le bloc de fonction : 5600+220 = **5820 Ω**

A calculer les autres variables *R_pas*, *iCons_beginn* et *R_max*

Machine d'état

Du code existe avant et après la machine d'état, ce code est exécuté quel que soit l'état auquel la machine d'état se trouve. Voici son fonctionnement :

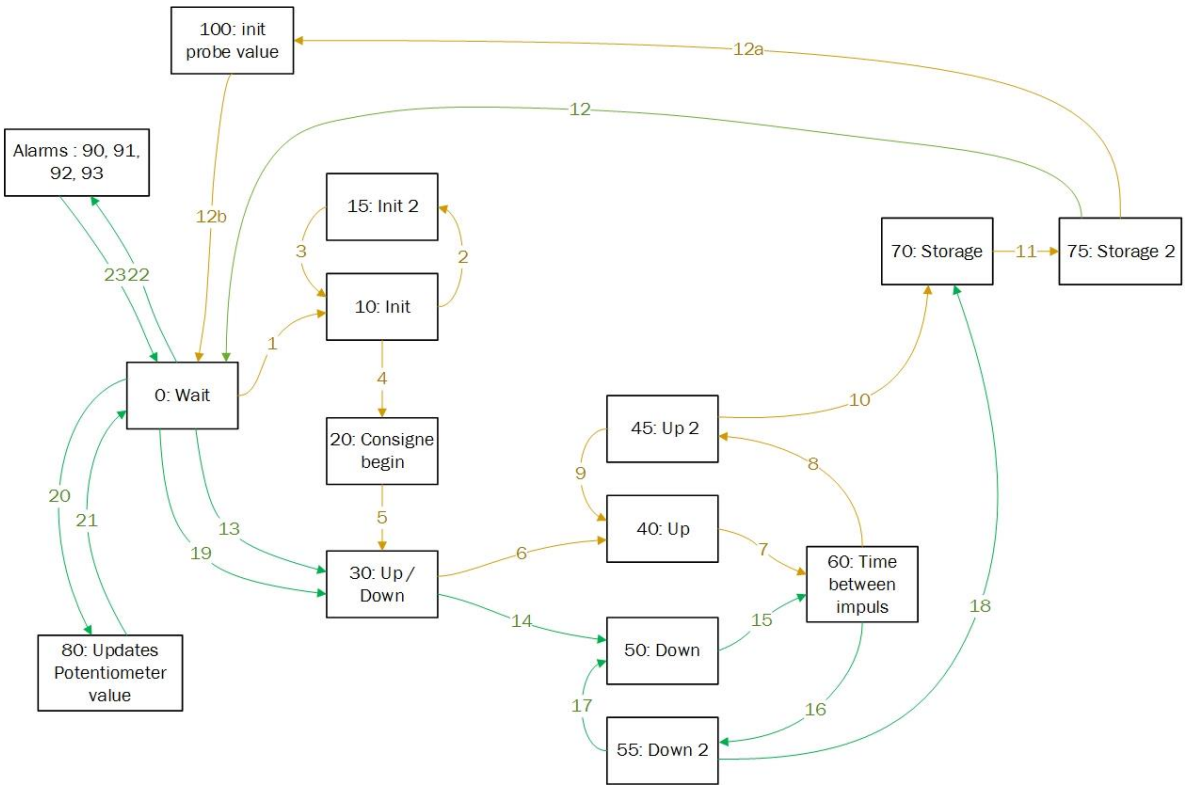


Figure 6 : Machine d'état

Initialisation : en jaune Fonctionnement normal : en vert (+ les transitions 6 à 12)

Transitions :

1) Condition d'initialisation
2) Maximum d'impulsions vers le bas (1024) pour avoir R_min (Les impulsions se font avec des flancs descendants (voir Figure 7), c'est pour cela qu'il faut les états 10 et 15 pour mettre à 1 et à 0 la sortie SCLK(INC))

Table 3. Truth Table

$\overline{CS}$	DIN(U/$\overline{D}$)	SCLK($\overline{INC}$)	W
L	L	↓	Decrement
L	H	↓	Increment
L	X	↑	No Change
H	X	X	No Change
↓	X	X	No Change
↑	X	L	Position Not Stored
↑	X	H	Position Stored

↑ = *Low-to-high transition.*
↓ = *High-to-low transition.*
X = *Don't care.*

Figure 7 : Fonctionnement des potentiomètres : table de vérité

3) Maximum d'impulsions vers le bas pour avoir R_min
4) Appliquer une consigne initiale à 15°C
5) Aller vers le calcul du nombre d'impulsions à faire

6) Aller dans la direction Up
7) Aller dans l'état d'attente (de base : pas d'attente, changement de résistance le plus rapide possible)
8) Impulsions vers le haut jusqu'à avoir la valeur de consigne
9) Impulsions vers le haut jusqu'à avoir la valeur de consigne
10) Enregistrement dans la mémoire non-volatile (front montant de CS)
11) Enregistrement dans la mémoire non-volatile (front montant de CS)
12) Aller dans la phase d'attente et lire la valeur de la sonde de température pour la première fois en appliquant les potentiomètres au régulateur
13) Si la consigne de la valeur faussée que l'on veut appliquer est différente à 15°C, une nouvelle boucle est faite
    a. Consigne inférieure : boucle : 13->14->**15->16->17**->18->11->12
       15,16 et 17 le nombre de fois qu'il faut faire des impulsions
    b. Consigne supérieure : boucle : 13->6->**7->8->9**->10->11->12
       15,16 et 17 le nombre de fois qu'il faut faire des impulsions
19) Précision : si la valeur_consigne n'est pas égale à la valeur_réelle_potentiometre ± 1°C, une nouvelle boucle est faite :
    a. Consigne inférieure : boucle : 19->14->**15->16->17**->18->11->12
       15,16 et 17 le nombre de fois qu'il faut faire des impulsions
    b. Consigne supérieure : boucle : 19->6->**7->8->9**->10->11->12
       15,16 et 17 le nombre de fois qu'il faut faire des impulsions
20) Actualise la valeur du potentiomètre si elle a changé de quelques ohms (chaque 2h)
21) Actualise la valeur du potentiomètre si elle a changé de quelques ohms (chaque 2h)
22) Blocage de valeurs supérieures et inférieure aux valeurs max et min !
23) Blocage de valeurs supérieures et inférieure aux valeurs max et min !

Réalisé par après :

12a) Initialise la valeur du potentiomètre pour la visualisation
12b) Retourne dans l'état wait

Annexe 6 : Programmation du serveur d'alarme et visualisation web – API MASTER

a. Programme du serveur d'alarme
b. Fonctions de requêtes MySQL pour l'ajout, la suppression, la modification ou la lecture d'informations de clients pour la table `*gsmtable*`
c. Visualisation web du serveur d'alarme

1 Global Variable List: GVL

```
1   {attribute 'qualified_only'}
2   VAR_GLOBAL CONSTANT
3       iMaxRowMYSQL            : INT := 200 ;
4       iMaxColMYSQL            : INT := 30 ;
5       iMYSQL_UPPER_BOUNDS     : INT := 50 ;   // Length of the conversion
    concat table (Wago STRING -> MySQL Text)
6       iMYSQL_SQL_LENGTH       : INT := 50 ;       // max STRING Length on a
    table compartment
7   END_VAR
8
```

2 POU: Alarm_DB

```
1   PROGRAM Alarm_DB
2   VAR
3
4   /// function blocks
5       fbMySqlLogin            : FbMySql_Login ;
6       fbMySqlExecute          : FbMySql_Execute ;
7       fbMySqlQuery            : FbMySql_Query ;
8       fbMySqlLogout           : FbMySql_Logout ;
9
10      // login variables
11      sMySqlHost              : STRING := '192.168.1.93' ; (* IP adress of
    computer containing mySql database server *)
12      uiMySqlPort             : UINT := 3306 ;
13      sMySqlUsername          : STRING := 'wago' ;
14      sMySqlPassword          : STRING := 'wago' ;
15      sMySqlDB                : STRING := 'r2d2database' ;
16      xMySqlLogin             : BOOL ;    (* Login at the beginn of the
    programm or in case of power outage *)
17      xMySqlLoginBusy         : BOOL ;
18      xMySqlLoginError        : BOOL ;
19      oMySqlLoginStatus       : WagoSysErrorBase . FbResult ;
20      sMySqlLoginStatus       : STRING ( 200 ) ;
21      xMySqlIsConnected       : BOOL ;
22
23      // table name
24      sMySqlTable             : STRING := 'gsmtable' ;
25      sMySqlTable_alarmtable  : STRING := 'alarmtable' ;
26
27      // execute INSERT variables
28      aMySqlInsertStatement   : ARRAY [ 0 .. MYSQL_SQL_UPPER_BOUND ] OF STRING
    ( MYSQL_SQL_LENGTH ) ;
29      aMySqlInsertStatement2  : ARRAY [ 0 .. MYSQL_SQL_UPPER_BOUND ] OF STRING
    ( MYSQL_SQL_LENGTH ) ;
30      xMySqlExecute           : BOOL ;
31      xMySqlExecuteBusy       : BOOL ;
32      xMySqlExecuteError      : BOOL ;
33      oMySqlExecuteStatus     : WagoSysErrorBase . FbResult ;
34      sMySqlExecuteStatus     : STRING ( 200 ) ;
```

```
35
36          // execute DELETE variables
37          aMySqlDeleteStatement    : ARRAY [ 0 .. MYSQL_SQL_UPPER_BOUND ] OF STRING
        ( MYSQL_SQL_LENGTH ) ;
38          xMySqlDelete : BOOL ;
39          xMySqlDeleteBusy : BOOL ;
40          xMySqlDeleteError : BOOL ;
41          oMySqlDeleteStatus : WagoSysErrorBase . FbResult ;
42          sMySqlDeleteStatus : STRING ( 200 ) ;
43
44          // execute EDIT variables
45          aMySqlEditStatement : ARRAY [ 0 .. MYSQL_SQL_UPPER_BOUND ] OF STRING (
        MYSQL_SQL_LENGTH ) ;
46          xMySqlEdit : BOOL ;
47          xMySqlEditBusy : BOOL ;
48          xMySqlEditError : BOOL ;
49          oMySqlEditStatus : WagoSysErrorBase . FbResult ;
50          sMySqlEditStatus : STRING ( 200 ) ;
51
52          // execute DISPLAS variables
53          aMySqlDisplayStatement   : ARRAY [ 0 .. MYSQL_SQL_UPPER_BOUND ] OF STRING
        ( MYSQL_SQL_LENGTH ) ;
54          xMySqlDisplay            : BOOL ;
55          typMySqlDisplayRslt      : typMySql_ResultSet ;
56          xMySqlDisplayBusy          : BOOL ;
57          xMySqlDisplayError       : BOOL ;
58          oMySqlDisplayStatus        : WagoSysErrorBase . FbResult ;
59          sMySqlDisplayStatus        : STRING ( 200 ) ;
60
61          // execute QUERY variables        (read database)
62          aMySqlQueryStatement     : ARRAY [ 0 .. MYSQL_SQL_UPPER_BOUND ] OF STRING
        ( MYSQL_SQL_LENGTH ) ;
63          xMySqlQuery              : BOOL ;
64          typMySqlQueryRslt        : typMySql_ResultSet ;
65          xMySqlQueryBusy          : BOOL ;
66          xMySqlQueryError         : BOOL ;
67          oMySqlQueryStatus        : WagoSysErrorBase . FbResult ;
68          sMySqlQueryStatus        : STRING ( 200 ) ;
69
70          // logout variables
71          xMySqlLogout             : BOOL ;
72          xMySqlLogoutBusy         : BOOL ;
73          xMySqlLogoutError        : BOOL ;
74          oMySqlLogoutStatus       : WagoSysErrorBase . FbResult ;
75          sMySqlLogoutStatus       : STRING ( 200 ) ;
76
77
78          // Visualization variables
79          sProjectNum              : STRING ;
80          sProjectName             : STRING ;
81          sGsm1                    : STRING ;
82          sGsm2                    : STRING ;
83          sGsm3                    : STRING ;
84          sGsm4                    : STRING ;
85          sGsm5                    : STRING ;
86          sResString1              : STRING    := '' ;
```

```
87          sResString2             : STRING    := '' ;
88          sResString3             : STRING    := '' ;
89
90          sProNumToDelOrEdit      : STRING ;
91          xDisplayMode            : BOOL ;
92          xAddMode                : BOOL ;
93          xDeleteMode             : BOOL ;
94          xEditMode               : BOOL ;
95
96          sDisplayName            : STRING ( 500 ) ;
97          sDisplayGsm1            : STRING ( 500 ) ;
98          sDisplayGsm2            : STRING ( 500 ) ;
99          sDisplayGsm3            : STRING ( 500 ) ;
100         sDisplayGsm4            : STRING ( 500 ) ;
101         sDisplayGsm5            : STRING ( 500 ) ;
102
103
104         R_TRIG_insert :  R_TRIG ;
105         R_TRIG_delete :  R_TRIG ;
106         iDisplayLoop :  INT  :=  0 ;
107         iDisplayLine : INT  :=  0 ;
108         R_TRIG_display :  R_TRIG ;
109         R_TRIG_edit :  R_TRIG ;
110
111         sProjectNumOLD              : STRING  :=  'something' ;
112         TON_Loop_is_done :  TON ;
113         SR_display :  SR ;
114
115
116
117         // read alarm database
118         FbMySql_Query_alarmtable :  FbMySql_Query ;
119         aMySqlQueryStatement2   : ARRAY  [ 0 .. MYSQL_SQL_UPPER_BOUND ]  OF  STRING
        ( MYSQL_SQL_LENGTH ) ;
120         xMySqlQuery2            : BOOL ;
121         typMySqlQueryRslt2      : typMySql_ResultSet ;
122         xMySqlQueryBusy2          : BOOL ;
123         xMySqlQueryError2       : BOOL ;
124         oMySqlQueryStatus2      : WagoSysErrorBase . FbResult ;
125         sMySqlQueryStatus2      : STRING ( 200 ) ;
126         sAlarmValue :  STRING ( 500 ) ;
127         sAlarmMessage :  STRING ( 500 ) ;
128         sAlarmMessageSMS  :  STRING ( 500 ) ;
129         sProjectName_Alarm :  STRING ( 500 )  :=  '' ;
130         iActualAlarmLoop :  INT  := 1 ;
131         iAlarmRow :  INT ;
132
133
134         sSendSMS_GsmNbr1 :  STRING ( 500 ) ;
135         sSendSMS_GsmNbr2 :  STRING ( 500 ) ;
136         sSendSMS_GsmNbr3 :  STRING ( 500 ) ;
137         sSendSMS_GsmNbr4 :  STRING ( 500 ) ;
138         sSendSMS_GsmNbr5 :  STRING ( 500 ) ;
139         xSendSMS1 :  BOOL ;
140         xSendSMS2 :  BOOL ;
141         xSendSMS3 :  BOOL ;
```

```
142         xSendSMS4 : BOOL ;
143         xSendSMS5 : BOOL ;
144
145
146         Fb_SMS_destination1 : FbSimpleSms_8207 ;
147         xInitDB : BOOL := TRUE ;
148         TON_read_alarmtable : TON ;
149         iInitPointers : INT := 1 ;
150         iInitPointers2 : INT := 1 ;
151         xInitPointers : BOOL ;
152         iEditPointers : INT := 1 ;
153         xEditPointers : BOOL ;
154         SR_edit : SR ;
155         iInsertPointers : INT := 1 ;
156         xInsertPointers : BOOL ;
157         SR_Insert : SR ;
158         iDeletePointers : INT := 1 ;
159         xDeletePointers : BOOL ;
160         SR_Delete : SR ;
161         F_TRIG_addMode : F_TRIG ;
162         F_TRIG_editMode : F_TRIG ;
163         F_TRIG_deleteMode : F_TRIG ;
164         SR_arrayPointer : SR ;
165         iDeletePointersSuperior : INT := 1 ;
166         TON_Wait_for_pointers : TON ;
167
168         F_TRIG_xSendSMS1 : F_TRIG ;
169         F_TRIG_xSendSMS2 : F_TRIG ;
170         F_TRIG_xSendSMS3 : F_TRIG ;
171         F_TRIG_xSendSMS4 : F_TRIG ;
172         F_TRIG_xSendSMS5 : F_TRIG ;
173
174         iTimeBeforeSecondSMS : INT := 30 ; // time in seconds
175         iHowManyTimes_Send1 : INT := 0 ;
176         iHowManyTimes_Send2 : INT := 0 ;
177         iHowManyTimes_Send3 : INT := 0 ;
178         iHowManyTimes_Send4 : INT := 0 ;
179         iHowManyTimes_Send5 : INT := 0 ;
180
181         SR_xSendSMS1 : SR ;
182         SR_xSendSMS2 : SR ;
183         SR_xSendSMS3 : SR ;
184         SR_xSendSMS4 : SR ;
185         SR_xSendSMS5 : SR ;
186
187         TON_waitBeforeSecondSMS1 : TON ;
188
189         asPointerLine : ARRAY [0..500] OF STRING(500) ;  // sert pour
    rechercher la ligne quand on veut faire un read
190         xHomePageVisu : BOOL ;
191
192     END_VAR
193     VAR CONSTANT
194         iDatabaseSize : INT := 500 ; // I don't know, 500 is maybe not
    enough..
195         iTimeForQuery : INT := 50 ; // take ~500ms to do a query (50x10ms)
```

```
196     END_VAR
197
198     VAR RETAIN PERSISTENT
199         iTotalNbLine : INT ;
200     END_VAR
201
```

```
1     (* Login at begin of program with the total nbr line of the database -----*)
2     IF iTotalNbLine <> 0 AND xInitDB THEN
3         xMySqlLogin := TRUE ;
4         xInitDB := FALSE ;
5     END_IF
6
7
8     (* Login ------------------------------------------------------------*)
9     fbMySqlLogin (
10    sHost := sMySqlHost ,
11    uiPort := uiMySqlPort ,
12    sUsername := sMySqlUsername ,
13    sPassword := sMySqlPassword ,
14    sDatabase := sMySqlDB ,
15    xTrigger := xMySqlLogin ,
16    xBusy => xMySqlLoginBusy ,
17    xError => xMySqlLoginError ,
18    oStatus => oMySqlLoginStatus ,
19    sStatus => sMySqlLoginStatus ,
20    xConnected => xMySqlIsConnected ) ;
21
22
23    (* Initialise the project pointers for reading database ----------------*)
24    IF iInitPointers2 <= iDatabaseSize AND xMySqlIsConnected THEN
25        // select all values from the table
26        aMySqlQueryStatement [ 0 ]    := 'SELECT * FROM `gsmtable` ORDER BY
      `ProjectNum` ASC' ;
27
28        IF iInitPointers < iDatabaseSize THEN
29            xInitPointers := TRUE ;
30        END_IF
31
32        // read project name
33        fbMySqlQuery (
34        aSqlCommand := aMySqlQueryStatement ,
35        xTrigger := xInitPointers ,
36        typResultSet := typMySqlQueryRslt ,
37        xBusy => xMySqlQueryBusy ,
38        xError => xMySqlQueryError ,
39        oStatus => oMySqlQueryStatus ,
40        sStatus => sMySqlQueryStatus ) ;
41
42
43        IF sMySqlQueryStatus = 'Successful executed' THEN
44            iInitPointers2 := iInitPointers2 + 1 ;
45        END_IF
46
47
```

```
48          IF iInitPointers >= iDatabaseSize THEN
49          iInitPointers :=1 ;
50          xInitPointers := FALSE ;
51          END_IF
52
53          IF iInitPointers <= iTotalNbLine THEN
54              // asPointerLine is an array of strings.
55              // This array contains the project names (e.g. Oscar Torres)
56              FuMySql_GetStringValue ( iRow := iInitPointers , iCol := 1 ,
        typQueryResult := typMySqlQueryRslt , sValue := asPointerLine [
        iInitPointers ] ) ;
57          END_IF
58          iInitPointers := iInitPointers +1 ;
59
60      END_IF
61
62
63      (*Insert row ------------------------------------------------------------*)
64      R_TRIG_insert ( CLK := xMySqlExecute ) ;
65      // update total # of row (+1)
66      IF R_TRIG_insert . Q AND xAddMode AND xMySqlIsConnected THEN
67          iTotalNbLine := iTotalNbLine +1 ;
68      END_IF
69      IF xAddMode THEN
70          //create INSERT Statement with a function       //gsmtable
71          aMySqlInsertStatement := funMySQLinsert (
72          sTable := sMySqlTable ,
73          sProNumber := sProjectNum ,
74          sProName := sProjectName ,
75          sGsmNum1 := sGsm1 ,
76          sGsmNum2 := sGsm2 ,
77          sGsmNum3 := sGsm3 ,
78          sGsmNum4 := sGsm4 ,
79          sGsmNum5 := sGsm5 ,
80          sReserveString1 := sResString1 ,
81          sReserveString2 := sResString2 ,
82          sReserveString3 := sResString3 ) ;
83          // add function block : commands execution for adding row
        //gsmtable
84          fbMySqlExecute (
85          aSqlCommand := aMySqlInsertStatement ,
86          xTrigger := xMySqlExecute ,
87          xBusy => xMySqlExecuteBusy ,
88          xError => xMySqlExecuteError ,
89          oStatus => oMySqlExecuteStatus ,
90          sStatus => sMySqlExecuteStatus ) ;
91
92          IF sMySqlExecuteStatus = 'Successful executed' THEN
93              sMySqlExecuteStatus := 'Client ajouté' ;
94          ELSE
95              sMySqlExecuteStatus := 'Client non ajouté, veuillez vérifier les
        données insérées' ;
96          END_IF
97
98          // update OLD
99          sProjectNumOLD := 'something' ;
```

```
100     END_IF
101
102     IF xAddMode = FALSE THEN
103         sMySqlExecuteStatus := 'Pour ajouter un client, veuillez activer le
        mode d$27ajout' ;
104     END_IF
105
106
107     (*Delete row ------------------------------------------------------------*)
108     R_TRIG_delete ( CLK := xMySqlDelete ) ;
109     // update total # of row (-1)
110     IF R_TRIG_delete . Q AND xDeleteMode AND xMySqlIsConnected THEN
111         iTotalNbLine := iTotalNbLine - 1 ;
112     END_IF
113     IF xDeleteMode THEN
114         // create DELETE Statement with a function
115         aMySqlDeleteStatement := funMySQLdelete (
116         sTable := sMySqlTable ,
117         sProNumber := sProjectNum ,
118         xDeleteMode := xDeleteMode ) ;
119         // add function block : commands execution for deleting row
120         fbMySqlExecute (
121         aSqlCommand := aMySqlDeleteStatement ,
122         xTrigger := xMySqlDelete ,
123         xBusy => xMySqlDeleteBusy ,
124         xError => xMySqlDeleteError ,
125         oStatus => oMySqlDeleteStatus ,
126         sStatus => sMySqlDeleteStatus ) ;
127
128         // update OLD
129         sProjectNumOLD := 'something' ;
130
131         IF sMySqlDeleteStatus = 'Successful executed' THEN
132             sMySqlDeleteStatus := 'Client supprimé' ;
133         ELSE
134             sMySqlDeleteStatus := 'Client non supprimé, veuillez vérifier les
        données insérées' ;
135         END_IF
136
137     END_IF
138
139     IF xDeleteMode = FALSE THEN
140         sMySqlDeleteStatus := 'Pour supprimer un client, veuillez activer le
        mode de suppression' ;
141     END_IF
142
143     (* Edit row ---------------------------------------------------------------*)
144     R_TRIG_edit ( CLK := xMySqlEdit ) ;
145     IF xEditMode THEN
146         // create EDIT Statement with a function
147         aMySqlEditStatement := funMySQLedit (
148         sTable := sMySqlTable ,
149         sProNumber := sProjectNum ,
150         sGsmNum1 := sGsm1 ,
151         sGsmNum2 := sGsm2 ,
152         sGsmNum3 := sGsm3 ,
```

```
153             sGsmNum4  :=  sGsm4 ,
154             sGsmNum5  :=  sGsm5 ) ;
155             // add function block : commands execution for editing row
156             fbMySqlExecute (
157             aSqlCommand :=  aMySqlEditStatement ,
158             xTrigger :=  xMySqlEdit ,
159             xBusy =>  xMySqlEditBusy ,
160             xError =>  xMySqlEditError ,
161             oStatus =>  oMySqlDeleteStatus ,
162             sStatus =>  sMySqlEditStatus ) ;
163
164             IF  sMySqlEditStatus  =  'Successful executed'  THEN
165                 sMySqlEditStatus  :=  'Informations du client modifiées' ;
166             ELSE
167                 sMySqlEditStatus  :=  'Informations du client non modifiées, veuillez
        vérifier les données insérées' ;
168             END_IF
169
170         END_IF
171
172         IF  xEditMode  =  FALSE  THEN
173             sMySqlEditStatus  :=  'Pour modifier les informations du client, veuillez
        activer le mode de modification' ;
174         END_IF
175
176         (* edit array asPointerLine ------------------------------------------------*)
177         SR_arrayPointer ( ) ;
178         F_TRIG_addMode ( CLK  :=  xAddMode ) ;
179         F_TRIG_deleteMode ( CLK  :=  xDeleteMode ) ;
180         F_TRIG_editMode ( CLK  :=  xEditMode ) ;
181         IF  F_TRIG_addMode . Q  OR  F_TRIG_editMode . Q  OR  F_TRIG_deleteMode . Q  THEN
182             SR_arrayPointer . SET1  :=  TRUE ;
183             SR_arrayPointer . RESET  :=  FALSE ;
184         END_IF
185
186         IF  SR_arrayPointer . Q1  THEN
187
188             // EDIT ARRAY POINTERS asPointerLine
189
190             IF  iEditPointers  <=  iDatabaseSize  AND  xMySqlIsConnected  THEN
191
192                 // select all values from the table
193                 aMySqlQueryStatement [ 0 ]    :=  'SELECT * FROM `gsmtable` ORDER BY
        `ProjectNum` ASC' ;
194
195                 IF  iEditPointers  < iTimeForQuery  THEN
196                     xEditPointers  :=  TRUE ;
197                 END_IF
198
199                 // read project name
200                 fbMySqlQuery (
201                 aSqlCommand :=  aMySqlQueryStatement ,
202                 xTrigger :=  xEditPointers ,
203                 typResultSet :=  typMySqlQueryRslt ,
204                 xBusy =>  xMySqlQueryBusy ,
205                 xError =>  xMySqlQueryError ,
```

```
206             oStatus => oMySqlQueryStatus ,
207             sStatus => sMySqlQueryStatus ) ;
208
209
210             IF iEditPointers <= iTotalNbLine THEN
211                 // asPointerLine is an array of strings.
212                 // This array contains the project names (e.g. Oscar Torres)
213                 FuMySql_GetStringValue ( iRow := iEditPointers , iCol := 1 ,
    typQueryResult := typMySqlQueryRslt , sValue := asPointerLine [
    iEditPointers ] ) ;
214             END_IF
215
216             IF sMySqlQueryStatus = 'Successful executed' THEN
217                 iEditPointers := iEditPointers + 1 ;
218             END_IF
219         END_IF
220
221         IF iEditPointers >= iDatabaseSize THEN
222             iEditPointers := 1 ;
223             xEditPointers := FALSE ;
224             SR_arrayPointer . SET1   := FALSE ;
225             SR_arrayPointer . RESET  := TRUE ;
226         END_IF
227
228         TON_Wait_for_pointers ( IN := TRUE , PT := T#2S ) ;
229
230         IF TON_Wait_for_pointers . Q AND iDeletePointersSuperior = 1 THEN
231             WHILE iDeletePointersSuperior < ( iDatabaseSize - iTotalNbLine ) DO
232             asPointerLine [ iTotalNbLine + iDeletePointersSuperior ] := '' ;
233             iDeletePointersSuperior := iDeletePointersSuperior + 1 ;
234             END_WHILE
235             TON_Wait_for_pointers ( IN := FALSE ) ;
236         END_IF
237
238     END_IF
239
240     IF SR_arrayPointer . Q1 = FALSE THEN
241         iDeletePointersSuperior := 1 ;
242     END_IF
243
244     (* read row database ----------------------------------------------------*)
245
246     // tempo to be sure that the loop is completely done !
247     // Make a tempo before resetting values
248     SR_display ( ) ;
249     IF R_TRIG_display . Q THEN
250         SR_display . SET1       := TRUE ;
251         SR_display . RESET      := FALSE ;
252     END_IF
253     TON_Loop_is_done ( IN := SR_display . Q1 , PT := T#500MS ) ;
254     // reset some values
255     IF  TON_Loop_is_done . Q AND sProjectNumOLD <> sProjectNum THEN
256         iDisplayLoop    := 0 ;
257         SR_display . RESET   := TRUE ;
258         SR_display . SET1    := FALSE ;
259         TON_Loop_is_done ( IN := FALSE ) ;
```

```
260     END_IF
261
262     R_TRIG_display  ( CLK :=  xMySqlDisplay ) ;  // Rising edge of xMySqlQuery
263     IF  xDisplayMode  THEN
264
265         // select all values from the table
266         aMySqlDisplayStatement [ 0 ]  :=  'SELECT * FROM `gsmtable` ORDER BY
        `ProjectNum` ASC' ;
267
268         // read project name
269         fbMySqlQuery (
270         aSqlCommand :=  aMySqlDisplayStatement ,
271         xTrigger :=  xMySqlDisplay ,
272         typResultSet :=  typMySqlDisplayRslt ,
273         xBusy =>  xMySqlDisplayBusy ,
274         xError =>  xMySqlDisplayError ,
275         oStatus =>  oMySqlDisplayStatus ,
276         sStatus =>  sMySqlDisplayStatus ) ;
277
278
279         IF  ( sProjectNum  <>  sProjectNumOLD )  AND  xMySqlDisplay  THEN
280             WHILE  iDisplayLoop <=  iTotalNbLine  DO
281                 // find the line to read
282                 IF  asPointerLine [ iDisplayLoop ]  =  sProjectNum  THEN
283                     iDisplayLine  :=  iDisplayLoop ;
284                 END_IF
285                 iDisplayLoop  :=  iDisplayLoop  + 1 ;
286
287                 // update old project num value
288                 IF  iDisplayLoop  =  iTotalNbLine  THEN
289                     sProjectNumOLD  :=  sProjectNum ;
290                 END_IF
291             END_WHILE
292         END_IF
293
294         FuMySql_GetStringValue ( iRow  :=  iDisplayLine ,  iCol  := 2 ,
        typQueryResult  :=  typMySqlDisplayRslt ,  sValue  :=  sDisplayName ) ;
295         FuMySql_GetStringValue ( iRow  :=  iDisplayLine ,  iCol  := 3 ,
        typQueryResult  :=  typMySqlDisplayRslt ,  sValue  :=  sDisplayGsm1 ) ;
296         FuMySql_GetStringValue ( iRow  :=  iDisplayLine ,  iCol  := 4 ,
        typQueryResult  :=  typMySqlDisplayRslt ,  sValue  :=  sDisplayGsm2 ) ;
297         FuMySql_GetStringValue ( iRow  :=  iDisplayLine ,  iCol  := 5 ,
        typQueryResult  :=  typMySqlDisplayRslt ,  sValue  :=  sDisplayGsm3 ) ;
298         FuMySql_GetStringValue ( iRow  :=  iDisplayLine ,  iCol  := 6 ,
        typQueryResult  :=  typMySqlDisplayRslt ,  sValue  :=  sDisplayGsm4 ) ;
299         FuMySql_GetStringValue ( iRow  :=  iDisplayLine ,  iCol  := 7 ,
        typQueryResult  :=  typMySqlDisplayRslt ,  sValue  :=  sDisplayGsm5 ) ;
300
301         IF  sMySqlDisplayStatus  =  'Successful executed'  THEN
302             sMySqlDisplayStatus  :=  'Table de données GSM correctement lue' ;
303         ELSE
304             sMySqlDisplayStatus  :=  'Table de données GSM non lue, veuillez
        vérifier les données insérées' ;
305         END_IF
306     END_IF
307
```

```
308     IF xDisplayMode = FALSE THEN
309         sMySqlDisplayStatus := 'Pour lire les informations du client, veuillez
        activer le mode d$27affichage' ;
310     END_IF
311
312     (*       An alarm comes from a customer
        ---------------------------------------*)
313
314     // select all values from the alarmtable
315     aMySqlQueryStatement2 [ 0 ]  := 'SELECT * FROM `alarmtable` ORDER BY
        `ProjectNum` ASC' ;
316
317     TON_read_alarmtable ( IN := TRUE , PT := T#1S ) ;
318     IF TON_read_alarmtable . Q THEN
319         xMySqlQuery2 := TRUE ;
320         TON_read_alarmtable ( IN := FALSE ) ;
321     END_IF
322
323
324     FbMySql_Query_alarmtable (
325     aSqlCommand := aMySqlQueryStatement2 ,
326     xTrigger := xMySqlQuery2 ,
327     typResultSet := typMySqlQueryRslt2 ,
328     xBusy => xMySqlQueryBusy2 ,
329     xError => xMySqlQueryError2 ,
330     oStatus => oMySqlQueryStatus2 ,
331     sStatus => sMySqlQueryStatus2 ) ;
332
333     WHILE iActualAlarmLoop <= iTotalNbLine DO
334         FuMySql_GetStringValue ( iRow := iActualAlarmLoop , iCol := 2 ,
        typQueryResult := typMySqlQueryRslt2 , sValue := sAlarmValue ) ;
335         FuMySql_GetStringValue ( iRow := iActualAlarmLoop , iCol := 3 ,
        typQueryResult := typMySqlQueryRslt2 , sValue := sAlarmMessage ) ;
336
337         IF sAlarmValue = '1' THEN
338             FuMySql_GetStringValue ( iRow := iActualAlarmLoop , iCol := 1 ,
        typQueryResult := typMySqlQueryRslt2 , sValue := sProjectName_Alarm ) ;
339             iAlarmRow := iActualAlarmLoop ;
340             iActualAlarmLoop := iTotalNbLine ;
341             // go out of the while loop when an alarm appears
342         ELSE
343             sProjectName_Alarm := '' ;
344         END_IF
345
346         iActualAlarmLoop := iActualAlarmLoop + 1 ;
347     END_WHILE
348
349     iActualAlarmLoop := 1 ;
350
351     (* read telephone numbers and AlarmMessage when an alarm appears -------*)
352     IF sProjectName_Alarm <> '' THEN
353         // select all values from the table
354         aMySqlQueryStatement [ 0 ]  := 'SELECT * FROM `gsmtable` ORDER BY
        `ProjectNum` ASC' ;
355
356         // Store the message in the SMS
```

```
357         sAlarmMessageSMS  :=  sAlarmMessage ;
358
359
360         // read project name
361         fbMySqlQuery (
362         aSqlCommand :=  aMySqlQueryStatement ,
363         xTrigger :=  xMySqlQuery ,
364         typResultSet :=  typMySqlQueryRslt ,
365         xBusy =>  xMySqlQueryBusy ,
366         xError =>  xMySqlQueryError ,
367         oStatus =>  oMySqlQueryStatus ,
368         sStatus =>  sMySqlQueryStatus ) ;
369
370
371         FuMySql_GetStringValue ( iRow  :=  iAlarmRow ,  iCol  := 3 ,  typQueryResult
    :=  typMySqlQueryRslt ,  sValue  :=  sSendSMS_GsmNbr1 ) ;
372         FuMySql_GetStringValue ( iRow  :=  iAlarmRow ,  iCol  := 4 ,  typQueryResult
    :=  typMySqlQueryRslt ,  sValue  :=  sSendSMS_GsmNbr2 ) ;
373         FuMySql_GetStringValue ( iRow  :=  iAlarmRow ,  iCol  := 5 ,  typQueryResult
    :=  typMySqlQueryRslt ,  sValue  :=  sSendSMS_GsmNbr3 ) ;
374         FuMySql_GetStringValue ( iRow  :=  iAlarmRow ,  iCol  := 6 ,  typQueryResult
    :=  typMySqlQueryRslt ,  sValue  :=  sSendSMS_GsmNbr4 ) ;
375         FuMySql_GetStringValue ( iRow  :=  iAlarmRow ,  iCol  := 7 ,  typQueryResult
    :=  typMySqlQueryRslt ,  sValue  :=  sSendSMS_GsmNbr5 ) ;
376
377
378     END_IF
379
380
381     (* Send SMS to the numbers ------------------------------------------------*)
382
383     // Send to the first GSM number
384     IF  sSendSMS_GsmNbr1  <>  ''  THEN
385         F_TRIG_xSendSMS1  ( CLK  :=  xSendSMS1 ) ;
386     //
387     // TODO : Bloc d'envoi de sms, Ce bloc met l'entrée xSend à zéro dès que
388     // le sms est envoyé
389     //
390         SR_xSendSMS1 ( ) ;
391
392         IF  F_TRIG_xSendSMS1 . Q  OR  SR_xSendSMS1 . Q1  THEN    // send second SMS
393
394             IF  F_TRIG_xSendSMS1 . Q  THEN
395                 iHowManyTimes_Send1  :=  iHowManyTimes_Send1 + 1 ;
396             END_IF
397
398             TON_waitBeforeSecondSMS1 (
399                 IN  :=  SR_xSendSMS1 . Q1 ,
400                 PT  :=  INT_TO_TIME ( iTimeBeforeSecondSMS * 1000 ) ) ;
401
402             IF  iHowManyTimes_Send1  = 1  THEN
403                 SR_xSendSMS1 . SET1 :=  TRUE ;
404                 SR_xSendSMS1 . RESET :=  FALSE ;
405             END_IF
406
407             IF  TON_waitBeforeSecondSMS1 . Q   THEN
```

```
                    xSendSMS1  := TRUE ;
                //        iHowManyTimes_Send1:= iHowManyTimes_Send1+1;
                    SR_xSendSMS1 . SET1 := FALSE ;
                    SR_xSendSMS1 . RESET := TRUE ;
            END_IF
        END_IF

        IF iHowManyTimes_Send1 = 0 THEN // Send for the first time a SMS
            xSendSMS1 := TRUE ;
        END_IF
    END_IF
    IF sAlarmValue = '0' AND sProjectName_Alarm = ''
    AND sSendSMS_GsmNbr1 <> '' THEN
        iHowManyTimes_Send1 := 0 ;
        TON_waitBeforeSecondSMS1 ( IN := FALSE ) ;
        sSendSMS_GsmNbr1 := '' ;
    END_IF

    // Send to the second GSM number
    IF sSendSMS_GsmNbr2 <> '' AND TON_waitBeforeSecondSMS1 . Q THEN
        F_TRIG_xSendSMS2 ( CLK := xSendSMS2 ) ;
    //
    // TODO : Bloc d'envoi de sms, Ce bloc met l'entrée xSend à zéro dès que
    // le sms est envoyé
    //
        SR_xSendSMS2 ( ) ;

        IF F_TRIG_xSendSMS2 . Q THEN
            iHowManyTimes_Send2 := iHowManyTimes_Send2 + 1 ;
        END_IF

        IF iHowManyTimes_Send2 = 0 THEN // Send for the first time a SMS
            xSendSMS2 := TRUE ;
        END_IF
    END_IF
    IF sAlarmValue = '0' AND sProjectName_Alarm = ''
    AND sSendSMS_GsmNbr2 <> '' THEN
        iHowManyTimes_Send2 := 0 ;
        sSendSMS_GsmNbr2 := '' ;
    END_IF

    // Send to the third GSM number
    IF sSendSMS_GsmNbr3 <> '' AND TON_waitBeforeSecondSMS1 . Q THEN
        F_TRIG_xSendSMS3 ( CLK := xSendSMS3 ) ;
    //
    // TODO : Bloc d'envoi de sms, Ce bloc met l'entrée xSend à zéro dès que
    // le sms est envoyé
    //
        SR_xSendSMS3 ( ) ;

        IF F_TRIG_xSendSMS3 . Q THEN
            iHowManyTimes_Send3 := iHowManyTimes_Send3 + 1 ;
        END_IF

        IF iHowManyTimes_Send3 = 0 THEN // Send for the first time a SMS
            xSendSMS3 := TRUE ;
```

```
464         END_IF
465     END_IF
466     IF sAlarmValue = '0' AND sProjectName_Alarm = ''
467     AND sSendSMS_GsmNbr3 <> '' THEN
468         iHowManyTimes_Send3 := 0 ;
469         sSendSMS_GsmNbr3 := '' ;
470     END_IF
471
472     // Send to the fourth GSM number
473     IF sSendSMS_GsmNbr4 <> '' AND TON_waitBeforeSecondSMS1 . Q THEN
474         F_TRIG_xSendSMS4 ( CLK := xSendSMS4 ) ;
475     //
476     // TODO : Bloc d'envoi de sms, Ce bloc met l'entrée xSend à zéro dès que
477     // le sms est envoyé
478     //
479         SR_xSendSMS4 ( ) ;
480
481         IF F_TRIG_xSendSMS4 . Q THEN
482             iHowManyTimes_Send4 := iHowManyTimes_Send4 + 1 ;
483         END_IF
484
485         IF iHowManyTimes_Send4 = 0 THEN // Send for the first time a SMS
486             xSendSMS4 := TRUE ;
487         END_IF
488     END_IF
489     IF sAlarmValue = '0' AND sProjectName_Alarm = '' AND sSendSMS_GsmNbr4 <>
        '' THEN
490         // reset values when the alarm disappears
491         iHowManyTimes_Send4 := 0 ;
492         sSendSMS_GsmNbr4 := '' ;
493     END_IF
494
495     // Send to the fifth GSM number
496     IF sSendSMS_GsmNbr5 <> '' AND TON_waitBeforeSecondSMS1 . Q THEN
497         F_TRIG_xSendSMS5 ( CLK := xSendSMS5 ) ;
498     //
499     // TODO : Bloc d'envoi de sms, Ce bloc met l'entrée xSend à zéro dès que
500     // le sms est envoyé
501     //
502         SR_xSendSMS5 ( ) ;
503
504         IF F_TRIG_xSendSMS5 . Q THEN
505             iHowManyTimes_Send5 := iHowManyTimes_Send5 + 1 ;
506         END_IF
507
508         IF iHowManyTimes_Send5 = 0 THEN // Send for the first time a SMS
509             xSendSMS5 := TRUE ;
510         END_IF
511     END_IF
512     IF sAlarmValue = '0' AND sProjectName_Alarm = ''
513     AND sSendSMS_GsmNbr5 <> '' THEN
514         // reset values when the alarm disappears
515         iHowManyTimes_Send5 := 0 ;
516         sSendSMS_GsmNbr5 := '' ;
517     END_IF
518
```

```
519     (*xHomePageVisu is TRUE when the user look the homepage------------------*)
520     IF xAddMode =FALSE AND xDeleteMode =FALSE AND xEditMode =FALSE
521     AND xDisplayMode = FALSE THEN
522         xHomePageVisu := TRUE ;
523     ELSE
524         xHomePageVisu := FALSE ;
525     END_IF
526
527     (* logout ------------------------------------------------------------*)
528     fbMySqlLogout (
529     xTrigger := xMySqlLogout ,
530     xBusy => xMySqlLogoutBusy ,
531     xError => xMySqlLogoutError ,
532     oStatus => oMySqlLogoutStatus ,
533     sStatus => sMySqlLogoutStatus ) ;
534
```

3 POU: funMySQLdelete

```
1     FUNCTION funMySQLdelete : ARRAY [ 0 .. MYSQL_SQL_UPPER_BOUND ] OF STRING (
      MYSQL_SQL_LENGTH ) ;
2     VAR_INPUT
3         sTable          : STRING ;
4         sProNumber      : STRING ;
5         xDeleteMode     : BOOL ;
6     END_VAR
7     VAR_OUTPUT
8         aMySqlQueryStatement    : ARRAY [ 0 .. MYSQL_SQL_UPPER_BOUND ] OF STRING
      ( MYSQL_SQL_LENGTH ) ;
9     END_VAR
10    VAR
11    END_VAR
12
13
```

```
1
2         aMySqlQueryStatement [ 0 ] := CONCAT ( 'DELETE FROM `' , sTable ) ;
3         aMySqlQueryStatement [ 1 ] := CONCAT3 ( '` WHERE `ProjectNum` = $27' ,
      sProNumber , '$27; ' ) ;
4         aMySqlQueryStatement [ 2 ] := CONCAT3 ( 'DELETE FROM `alarmtable` WHERE
      `ProjectNum` = $27' , sProNumber , '$27;' ) ;
5         funMySQLdelete := aMySqlQueryStatement ;
6
7
8     (* example :
9     DELETE FROM `gsmtable` WHERE ProjectNum = '1994_07';
10    DELETE FROM `alarmtable` WHERE ProjectNum '1994_07';
11    *)
12
```

4 POU: funMySQLedit

```
1   FUNCTION funMySQLedit : ARRAY [0..MYSQL_SQL_UPPER_BOUND] OF STRING(
    MYSQL_SQL_LENGTH);
2   VAR_INPUT
3       sTable          : STRING;
4       sProNumber      : STRING;
5       sGsmNum1        : STRING;
6       sGsmNum2        : STRING;
7       sGsmNum3        : STRING;
8       sGsmNum4        : STRING;
9       sGsmNum5        : STRING;
10  END_VAR
11  VAR_OUTPUT
12      aMySqlQueryStatement    : ARRAY [0..MYSQL_SQL_UPPER_BOUND] OF STRING
    (MYSQL_SQL_LENGTH);
13  END_VAR
14  VAR
15      i: INT;
16  END_VAR
17
18
```

```
1       i := 0;
2       aMySqlQueryStatement[0] := CONCAT3('UPDATE `', sTable, '` SET ');
3
4       IF sGsmNum1 <> '' THEN
5           i := i+1;
6       aMySqlQueryStatement[i] := CONCAT3(' `Gsm1`=$27', sGsmNum1, '$27');

7       END_IF
8
9       IF sGsmNum2 <> '' THEN
10          i := i+1;
11          IF i>1 THEN
12              aMySqlQueryStatement[i] := CONCAT3(', `Gsm2`=$27', sGsmNum2,
    '$27');
13          ELSE
14              aMySqlQueryStatement[i] := CONCAT3(' `Gsm2`=$27', sGsmNum2,
    '$27');
15          END_IF
16      END_IF
17
18      IF sGsmNum3 <> '' THEN
19          i := i+1;
20          IF i>1 THEN
21              aMySqlQueryStatement[i] := CONCAT3(', `Gsm3`=$27', sGsmNum3,
    '$27');
22          ELSE
23              aMySqlQueryStatement[i] := CONCAT3(' `Gsm3`=$27', sGsmNum3,
    '$27');
24          END_IF
```

```
25          END_IF
26
27          IF  sGsmNum4  <>  ''  THEN
28              i  :=  i + 1 ;
29              IF  i > 1  THEN
30                  aMySqlQueryStatement [ i ]  :=  CONCAT3 ( ', `Gsm4`=$27' , sGsmNum4 ,
    '$27' ) ;
31              ELSE
32                  aMySqlQueryStatement [ i ]  :=  CONCAT3 ( ' `Gsm4`=$27' , sGsmNum4 ,
    '$27' ) ;
33              END_IF
34          END_IF
35
36          IF  sGsmNum5  <>  ''  THEN
37              i  :=  i + 1 ;
38              IF  i > 1  THEN
39                  aMySqlQueryStatement [ i ]  :=  CONCAT3 ( ', `Gsm5`=$27' , sGsmNum5 ,
    '$27' ) ;
40              ELSE
41                  aMySqlQueryStatement [ i ]  :=  CONCAT3 ( ' `Gsm5`=$27' , sGsmNum5 ,
    '$27' ) ;
42              END_IF
43          END_IF
44
45          i  :=  i + 1 ;
46          aMySqlQueryStatement [ i ]  :=  CONCAT3 ( ' WHERE `ProjectNum`=$27' ,
    sProNumber , '$27;' ) ;
47
48          funMySQLedit  :=  aMySqlQueryStatement ;
49
50
51      (* example :
52      UPDATE gsmtable
53      SET Gsm1='0792225544', Gsm2='0789996655' , Gsm3='0789996655' ,
        Gsm4='0789996655' , Gsm5='0789996655'
54      WHERE ProjectNum='1993_10';
55      *)
56
```

5 POU: funMySQLinsert

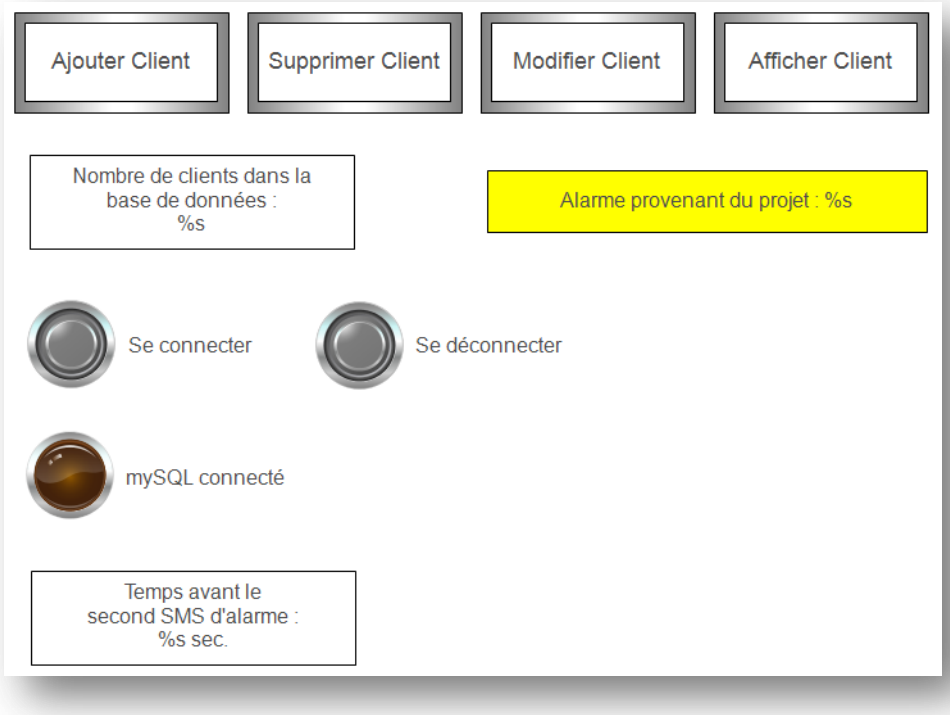
```
1       FUNCTION  funMySQLinsert  :  ARRAY  [ 0 .. MYSQL_SQL_UPPER_BOUND ]  OF  STRING (
        MYSQL_SQL_LENGTH ) ;
2       VAR_INPUT
3           sTable           :  STRING ;   // 20 char max
4           sProNumber       :  STRING ;   // 20 char
5           sProName         :  STRING ;   // 20 char
6           sGsmNum1         :  STRING ;   // 15 char
7           sGsmNum2         :  STRING ;   // 15 char
8           sGsmNum3         :  STRING ;   // 15 char
9           sGsmNum4         :  STRING ;   // 15 char
10          sGsmNum5         :  STRING ;   // 15 char
11          sReserveString1      :  STRING ;   // 90 char
12          sReserveString2      :  STRING ;   // 90 char
13          sReserveString3      :  STRING ;   // 90 char
```

```
END_VAR
VAR_OUTPUT
    // requête SQL
    aMySqlStatement : ARRAY [0..MYSQL_SQL_UPPER_BOUND] OF STRING(MYSQL_SQL_LENGTH);
END_VAR
VAR
    sConcat         : STRING;
    sConcat2        : STRING;
END_VAR

```

```
(* Note : $27 insert apostroph in mySQL langage *)

(* Note : Max 100 character by array box : aMySqlStatement[i] =
                                  'For example, this is a text with 100 characters. That's only to see how long are 100 characters !!!'       ;*)

aMySqlStatement[0] := CONCAT3('INSERT INTO `', sTable, '` (`ProjectNum`,`ProjectName`,`Gsm1`,`Gsm2`,`Gsm3`,' );
aMySqlStatement[1] := CONCAT3('`Gsm4`,`Gsm5`,`ResString1`,`ResString2`,`ResString3`) VALUES ($27', sProNumber, '$27, $27');
aMySqlStatement[2] := CONCAT6(sProName, '$27, $27', sGsmNum1, '$27, $27', sGsmNum2, '$27, $27');
aMySqlStatement[3] := CONCAT6(sGsmNum3, '$27, $27', sGsmNum4, '$27, $27', sGsmNum5, '$27, $27');
aMySqlStatement[4] := CONCAT(sReserveString1, '$27, $27');
aMySqlStatement[5] := CONCAT(sReserveString2, '$27, $27');
aMySqlStatement[6] := CONCAT(sReserveString3, '$27); ');

    // alarmtable
aMySqlStatement[7] := 'INSERT INTO `alarmtable` (`ProjectNum`, `AlarmIsActive`, `AlarmMessage`, `ResString1`, `ResString2`';
aMySqlStatement[8] := CONCAT3(') VALUES ($27', sProNumber, '$27, $270$27, $27$27, $27$27, $27$27);');

funMySQLinsert := aMySqlStatement;

(* example :
INSERT INTO `gsmtable` (`ProjectNum`, `ProjectName`, `Gsm1`, `Gsm2`, `Gsm3`, `Gsm4`, `Gsm5`,
`ResString1`, `ResString2`, `ResString3`)
VALUES ('année_numéro', 'Nom Prénom', '0790001234', '0780005678', '', '', '', '', '', '');
*)


(*
INSERT INTO `alarmtable`
(`ProjectNum`, `AlarmIsActive`, `AlarmMessage`, `ResString1`, `ResString2`)
VALUES ('2020_1', '0', '', '', '');

INSERT INTO `gsmtable` (`ProjectNum`, `ProjectName`, `Gsm1`, `Gsm2`, `Gsm3`,
```

Visualisation du serveur d'alarme

Page d'accueil

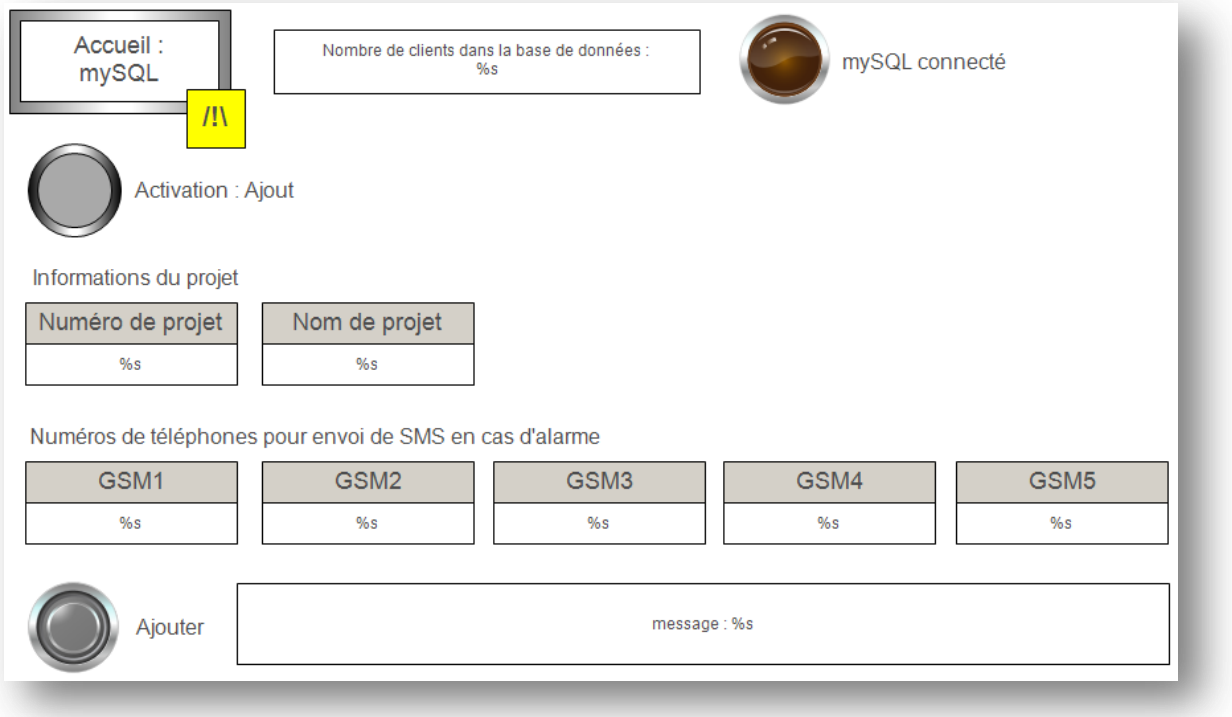


Lors du lancement du programme e!cockpit il faut insérer le nombre de clients déjà présent dans la base de données.

La page d'accueil permet de :

- Se connecter et déconnecter du serveur MySQL
- D'avoir un affichage du numéro de projet où une alarme est activée
- De choisir le temps avant l'envoi du second SMS
- D'atteindre les autres pages de cette visualisation pour la modification de la base de données

Page d'ajout

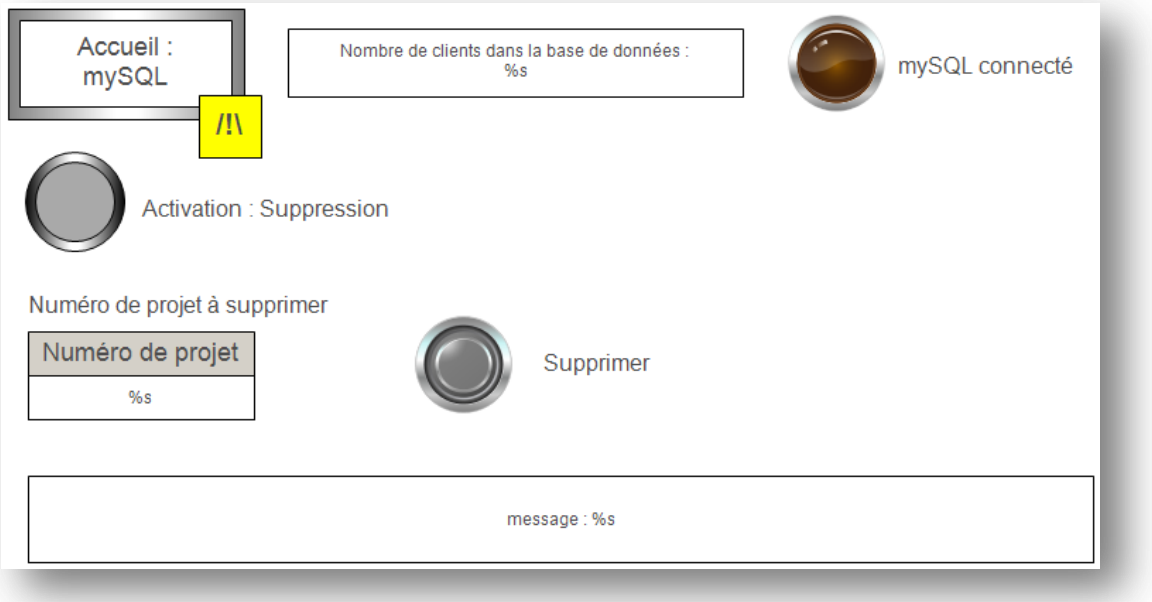


Pour ajouter un nouveau client il faut que le bouton *Activation : Ajout* soit appuyé (couleur vert), insérer les informations du projet client selon le standard de l'entreprise puis cliquer sur le bouton *Ajouter*.

A noter que l'ajout d'un numéro de projet identique à un numéro de projet existant est impossible ! La base de données peut contenir que des numéros de projet différents.

Le carré jaune s'affiche lorsqu'une alarme d'un client apparait dans la table d'alarmes `*alarmtable*`.

Page de suppression



Pour supprimer un client il faut que le bouton *Activation : Suppression* soit appuyé (couleur rouge), insérer numéro du projet client à supprimer puis cliquer sur le bouton *Supprimer*.

Le carré jaune s'affiche lorsqu'une alarme d'un client apparait dans la table d'alarmes `*alarmtable*`.

Page de modification

Pour modifier un client il faut que le bouton *Modification : Ajout* soit appuyé (couleur vert), insérer numéro de projet à modifier et insérer seulement les numéros GSM à modifier puis cliquer sur le bouton *Modifier*.

Le carré jaune s'affiche lorsqu'une alarme d'un client apparait dans la table d'alarmes `*alarmtable*`.

Page de recherche des informations d'un client

Pour afficher les informations d'un client il faut que le bouton *Activation : Affichage* soit appuyé (couleur vert), insérer numéro du projet client à afficher puis cliquer sur le bouton *Afficher*.

Le carré jaune s'affiche lorsqu'une alarme d'un client apparait dans la table d'alarmes `*alarmtable*`.

Annexe 7 : Planning du travail de diplôme

a. Planning Gantt – Travail de Bachelor 2017, Oscar Torres

Oscar Torres - Diplôme 2017

jeudi et vendredi : ascension

lundi : pentecôte ; présentation intermédiaire : 8-9 juin

jeudi : fête dieu

1 août : fête nationale

15 août : Assomption ; remise rapport 18 août 12h00

expositions : 30 août -1 sept.

défense orale TD : 5 sept

		N° de semaine du travail de diplôme	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**			
		N° de semaine du calendrier 2017	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*			
		Nombre de jours de travail dans la semaine	*5*	*3*	*5*	*4*	*4*	*5*	*5*	*5*	*5*	*5*	*5*	*4*	*5*	*4*	*Total : 64 jours*		
		Planning GANTT	*15-19 mai*	*22-26 mai*	*29 mai - 2 juin*	*5-9 juin*	*12-16 juin*	*19-23 juin*	*26-30 juin*	*3-7 juil.*	*10-14 juil.*	*17-21 juil.*	*24-28 juil.*	*31juil. - 4 août*	*7-11août*	*14-18 août*	*21-25 août*	*28 août - 1 sept.*	*4-8 sept.*
		prévu : x / réel : x																	
N°	Etat [%]	**Partie 0 : Rapport, Présentation et Flipchart**																	
000	80	Rapport	xx	xx	xx	xx	x	x	xx	x	xx	xx	x	xx	xx	xx			
010	0	Remise du rapport (en vert : rapport intermédiaire)				x										xx			
020	0	Définition Stratégie Présentation				xx											x		
030	0	création Flipchart															x	x	
040	0	création Powerpoint (en vert : présentation intermédiaire)				x											x	x	
050	0	Défense orale TD devant experts																	x
	13,3333	**<= TOTAL [%]**																	
N°	Etat [%]	**Partie 1 : Interface de mesure/commande de chauffage**																	
100	100	Reprise des informations de l'ancien TD	xx																
110	100	Faire schémas état actuel + état prévisionnel (futur)	xx																
120	100	Recherches internet pour être modulable selon systèmes existants	xx																
130	100	Premiers schémas à la main	xx																
140	100	Commande de matériel		xx															
150	100	Conception de la nouvelle carte (software)		xx	x														
160	100	Conception de la nouvelle carte (hardware)		x	xx														
170	90	Création/modification de blocs fonctionnels pour Wago/Siemens			x	xx	x			x									
180	90	Visualisation			x	x	x			x									
190	90	Tests			xx	x	x			x									
	97	**<= TOTAL [%]**																	
N°	Etat [%]	**Partie 2 : Serveur d'alarmes Wago**																	
200	100	Prise en main API Wago PLC200 + fonctionnement général voulu				x	x	x											
210	100	Définition Stratégie (visu + code) + Compréhension Biblio Wago envoi SMS					x	xx											
220	100	Création base de donnée						x	xx	x	x	x							
230	100	Décodage réception e-mail / Reception Alarme							x			x	x						
240	100	Création d'un sms type et test d'envoi							x					x					
250	100	Ajout/Modification/Suppression Client							x	xx	x								
260	100	Codage envoi SMS à plusieurs destinataires							x	x				x	x				
280	100	Création visualisation						x	x	x	xx	xx	x	x					

290	80	Tests							x	x	xx	xx	x	x	x				
	98	**<= TOTAL [%]**																	

N°	Etat [%]	**Partie 3 : Serveur météo (Dans la mesure du possible, optionnel)**																
300	0	Compréhension système actuel + programmation actuelle										x						
310	0	Définition Stratégie (visu + code)										x	x					
320	0	Stockage des valeurs dans des variables											x					
330	0	Retranscrire les variables dans un fichier XML créé											x	x				
340	0	Requête d'automates SLAVES et réception de valeurs voulues du MASTER												x	x			
350	0	Transmission d'autres variables (ex : heure) MASTER -> SLAVES													x	x		
360	0	Création visualisation												x	x			
370	0	Tests											x	x	x	x		
	0	**<= TOTAL [%]**																