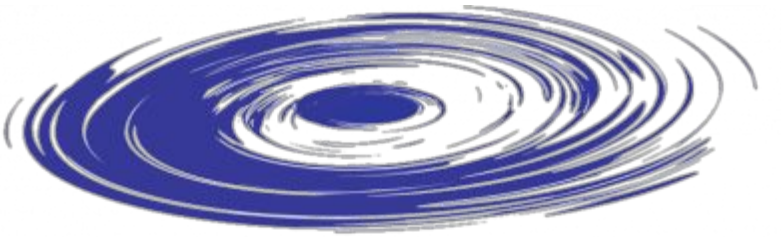
Université de Nice - Sophia Antipolis
IUT Nice Côte d'Azur
Département Informatique
41 bd. Napoléon III - 06200 Nice

Rapport de stage pour l'obtention du Diplôme Universitaire de Technologie
Session 2014 - 2015

Développement d'un outil de contrôle intelligent de la voie pupille du télescope MISOLFA

Présenté par Alexis Metge



LAGRANGE

Observatoire de la Côte d'Azur
Boulevard de l'Observatoire
CS 34229
06304 NICE Cedex 4

Sous la direction de :
M. Frédéric Morand - tuteur entreprise
Mme. Seye Oumy - tutrice IUT

Remerciements

Je tiens à remercier M. Frédéric Morand, mon tuteur au sein de l'Observatoire de la Côte d'Azur, qui m'a suivi et guidé durant ce stage.

Je remercie également Mme Catherine Renaud, informaticienne, et M. Mammar Fodil, électronicien, pour l'aide qu'ils m'ont apportée.

Mes remerciements s'adressent aussi à tout le personnel présent sur le site de Calern qui a su rendre mon séjour agréable.

Enfin, je remercie Mme Oumy Seye, ma tutrice IUT, pour l'intérêt qu'elle a porté au bon déroulement de mon stage.

Résumé

L'obtention du Diplôme Universitaire de Technologie passe nécessairement par la réalisation d'un stage en entreprise d'une durée minimale de 10 semaines. J'ai donc réalisé ce stage au sein du laboratoire Lagrange de l'Observatoire de la Côte d'Azur, du 20 avril au 26 juin 2015. L'équipe PICARDSOL a pour mission l'observation du Soleil afin de pouvoir, entre autres, mesurer son diamètre tout en étudiant les perturbations atmosphériques.

L'objectif du stage est l'analyse et le développement d'un outil de remplacement du programme de contrôle et d'acquisition d'une voie d'analyse d'un instrument qui observe le Soleil, MISOLFA, afin de permettre une mise en fonctionnement plus aisée, une synchronisation avec les autres outils et une adaptabilité selon les conditions d'observations.

Les difficultés rencontrées concernent surtout la gestion du temps et l'adaptabilité aux conditions d'observation, ce qui a nécessité de longues discussions avec mon tuteur et le personnel du projet, afin de déterminer quels signaux doivent être observés et comment réagir.

Abstract

To obtain a University Diploma in Computer Science, an internship of 10 weeks is required. It was carried out from 20 April to 26 June in the laboratory Lagrange of the Observatoire de la Côte d'Azur. The PICARDSOL project is dedicated to the observation and measurement of the solar diameter, and also study the atmospheric perturbation.

My mission was to develop a software capable of controlling the acquisition part of a telescope, in order to interact with the electronic components and allow users to modify easily those components, to automatically synchronize the acquisition of data with the others instruments and finally to adapt to environmental conditions, such as weather or heat.

The main difficulty was to time the data and to adapt to environmental conditions, which led to long discussions with my tutor in order to find which signals have to be monitored and how the software should react.

Table des matières

1. Introduction

Ce rapport présente le stage effectué au sein de l'Observatoire de la Côte d'Azur, et plus précisément au sein de l'équipe PICARDSOL du laboratoire Lagrange, du 20 avril au 26 juin 2015.
L'instrumentation PICARDSOL observe le Soleil à l'aide de deux instruments présents sur le plateau de Calern. Le premier, SODISM2, est une réplique au sol du télescope SODISM embarqué sur le satellite PICARD. Leur mission est d'observer le Soleil afin de pouvoir mesurer, entre autres, le diamètre du soleil. Le second instrument, MISOLFA, est installé à côté de SODISM2. Sa mission est de quantifier les perturbations atmosphériques grâce à ses deux voies d'acquisition : la voie image, qui capture des images de deux bords opposés du soleil, et la voie pupille, qui échantillonne l'intensité lumineuse prélevée en quatre points d'une image du miroir primaire.
Durant l'exploitation de SODISM, il a été possible de comparer les images obtenues par SODISM et SODISM2, et d'en comprendre les différences grâce à MISOLFA. Depuis l'arrêt de la mission PICARD en avril 2014, les observations réalisées par MISOLFA permettent de quantifier l'influence de l'atmosphère sur les images obtenues par SODISM2.

L'objectif du stage est de réaliser un logiciel permettant de remplacer le programme actuellement utilisé pour le contrôle, l'acquisition et l'enregistrement des données de la voie pupille de MISOLFA.

2. Présentation de l'entreprise

2.1. L'Observatoire de la Côte d'Azur

L'Observatoire de la Côte d'Azur (OCA) est un établissement public national d'enseignement supérieur et de recherche. L'OCA est présent sur quatre sites géographiques :

- le site historique du Mont Gros,
- le campus Valrose de l'Université de Nice Sophia Antipolis (UNS) à Nice,
- le site de Sophia Antipolis,
- le site d'observation de Calern.

L'OCA est l'un des 27 Observatoires des Sciences de l'Univers et contribue aux progrès de la connaissance par l'acquisition de données d'observations et le développement d'outils nécessaires à ces missions, mais également par la formation d'étudiants et de personnels de recherche grâce à l'UNS.

L'Unité Mixte de Services Galilée assure la maintenance des infrastructures et permet la mise en oeuvre de la politique scientifique de l'établissement.
Trois unités de recherche, Artémis, Géoazur et Lagrange, sont sous la tutelle de l'UNS, du CNRS, de l'IRD et de l'OCA.

L'activité principale de l'UMR Artémis est la recherche et le développement d'antennes gravitationnelles. La construction de Virgo, leur première antenne, a été réalisée en partenariat avec l'Institut National de Physique Nucléaire (INFN) italien et construite à côté de Pise. Virgo permet la détection d'ondes gravitationnelles, produites lors de l'explosion d'une supernova ou lors de la fusion d'étoiles binaires, apparaissant dans la voie lactée ou des galaxies voisines. Une seconde antenne, Advanced Virgo, est en projet et devrait être 10 fois plus sensible que Virgo.

Le laboratoire Géoazur effectue principalement des recherches sur l'activité sismique et le niveau moyen des mers, avec des projets tels que GLOBALSEIS qui à pour objectif de déterminer comment la Terre régule sa température interne ou bien le réseau national de GPS RENAG qui permet l'étude des déformations et les glissements du terrain. Enfin , Géoazur est impliqué dans des recherches sur la métrologie de l'espace proche, notamment grâce à MéO, décrit ci-après, ou bien la Station laser Ultra Mobile FTLRS, facilement transportable grâce à ses 300 kilos et son télescope de 13cm de diamètre, capable de tirer sur des satellites.

2.2. Le site de Calern

Le site de Calern, sur lequel se déroule mon stage, est un plateau qui se trouve à une altitude de 1270m. Divers instruments y sont présents tels que MISOLFA et SODISM2 ou bien MéO, C2PU et TAROT.

MéO, une station de télémétrie laser, est composé d'un télescope Cassegrain coudé d'un diamètre de 1,54m et d'une distance focale de 31m. Il est équipé d'un laser pouvant émettre

deux longueurs d'ondes différentes (532nm et 1064nm). Cette station est capable de tirer sur la Lune ou sur des satellites positionnés à une altitude inférieure à 36000km. L'utilisation d'un chronomètre couplé à un détecteur de retour permet de mesurer le temps mis par l'impulsion laser à effectuer l'aller-retour entre MéO et sa cible, et ainsi de calculer la distance séparant les deux. Ces mesures sont rendues possibles par la présence de rétroréflecteurs sur les cibles. La Lune en est équipée depuis la mission Apollo 11 en 1969. Les mesures de distance Terre-Lune ont une précision de quelques millimètres et ont permis de déterminer que la Lune s'éloigne de la Terre d'environ 3.5cm par an.

C2PU, le projet Centre Pédagogique Planète et Univers, est un projet en partenariat avec l'Université de Nice Sophia Antipolis utilisant deux télescopes professionnels réhabilités pour faire de la recherche, notamment sur les astéroïdes, et permettant à des étudiants en licence ou en master d'accéder à des outils astronomiques afin de travailler sur des données réelles.

TAROT, le Télescope à Action Rapide pour les Objets Transitoires, est un télescope autonome mis en service en 1999 sur le plateau. En mode *routine*, il est programmé pour s'ouvrir cinq minutes après le coucher du soleil et se fermer cinq minutes avant le lever, et il gère de manière automatique son planning d'observation de la nuit. Les utilisateurs pouvant utiliser cet outil interagissent au travers d'une interface web afin de demander l'observation d'un objet, et le système insérera l'observation selon divers critères, tels que la proximité de la Lune, ou les heures d'apparition de l'objet. TAROT peut également fonctionner en mode *opportunité* qui lui permet de donner la priorité à l'observation de sursauts gamma dès que l'un d'entre eux est signalé.

Enfin, le GI2T et le Schmidt sont deux instruments maintenant arrêtés qui ont contribué au développement de Calern. Le premier, avec ses deux télescopes de 1,5m de diamètre, est le précurseur des interféromètres. En effet, il a permis d'effectuer des recherches sur les méthodes d'interférométrie, qui sont maintenant utilisées sur tous les grands interféromètres du monde, comme le VLT avec ses 4 télescopes de 8,2m. Le Schmidt a été en activité pendant environ 30 ans et a produit plus de 5000 clichés de divers objets célestes, et a permis la découverte de plusieurs comètes, de nombreux astéroïdes et supernovae.

2.3. Le laboratoire Lagrange

Le laboratoire Lagrange est une unité mixte de recherche qui a été formée en 2012 par la fusion de deux laboratoires : Fizeau et Cassiopée. Ce laboratoire pluridisciplinaire regroupe des équipes d'astrophysique, de mécanique des fluides et de traitement de signal et d'images. Une partie du personnel du laboratoire enseigne directement au sein de l'UNS, en première et seconde année de licence et également au sein du master IMAG2E, qui forme des experts en traitement du signal et modélisation, et propose des offres de stage et de thèse.

Ce laboratoire est à la tête d'un consortium européen développant MATISSE, un instrument de seconde génération pour l'interféromètre VLTI. Le laboratoire est également impliqué

dans divers projets, comme par exemple dans la préparation de l'instrumentation du 'Extremely Large Telescope' européen grâce à ses recherches en imagerie ou bien dans traitement et l'analyse de données issues de la mission Gaia de l'ESA.

Enfin, une salle de contrôle à distance de l'instrument VEGA@CHARA, installé sur le mont Wilson en Californie, est en cours de migration de l'Observatoire de Nice vers le site de Calern.

2.4. L'équipe PicardSol

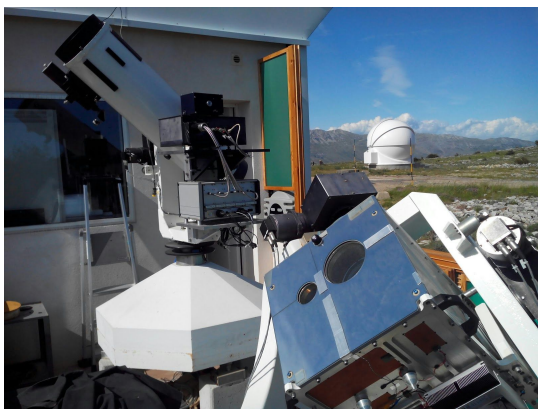
L'équipe du projet PicardSol

De gauche à droite :

- Catherine Renaud
- Rabah Ikhlef
- Thierry Corbard
- Frédéric Morand
- Moi-même
- Amel Zaatri
- Fabrice Ubaldi
- Mammar Fodil

2.5. L'instrumentation PicardSol

Depuis 2011, MISOLFA et SODISM2 sont en fonctionnement simultané et observent le Soleil. Une partie de leurs observations a été réalisée simultanément avec celles de SODISM, installé sur le satellite PICARD lancé en juin 2010. SODISM2 est la réplique au sol de SODISM, et tous deux ont pour mission l'observation et la mesure du diamètre du Soleil dans 5 longueurs d'ondes différentes (393nm, 535nm, 607nm, 782nm et 1025nm). MISOLFA, quant à lui, a pour objectif de caractériser la turbulence atmosphérique grâce à ses deux voies d'acquisition.



MISOLFA (à gauche) et SODISM2 (à droite)

SODISM2 est un télescope de 11cm de diamètre qui intègre une caméra d'une résolution de 2048x2048 pixels. Lors de sa mise en fonctionnement, SODISM2 génère un plan d'observation pour une durée de 180 minutes. Ce plan prévoit les heures auxquelles une image sera enregistrée, une par minute, et le filtre qui sera utilisé. Afin d'obtenir des conditions d'observations similaires à celles de l'espace, la cuve de SODISM2 est maintenue sous vide à une température constante de 20°C et sa caméra à -10°C.

MISOLFA, le Moniteur d'Image SOLaire Franco-Algérien, est un télescope né d'une coopération entre l'OCA et le CRAAG, Centre de Recherche en Astronomie, Astrophysique et Géophysique, situé en Algérie.

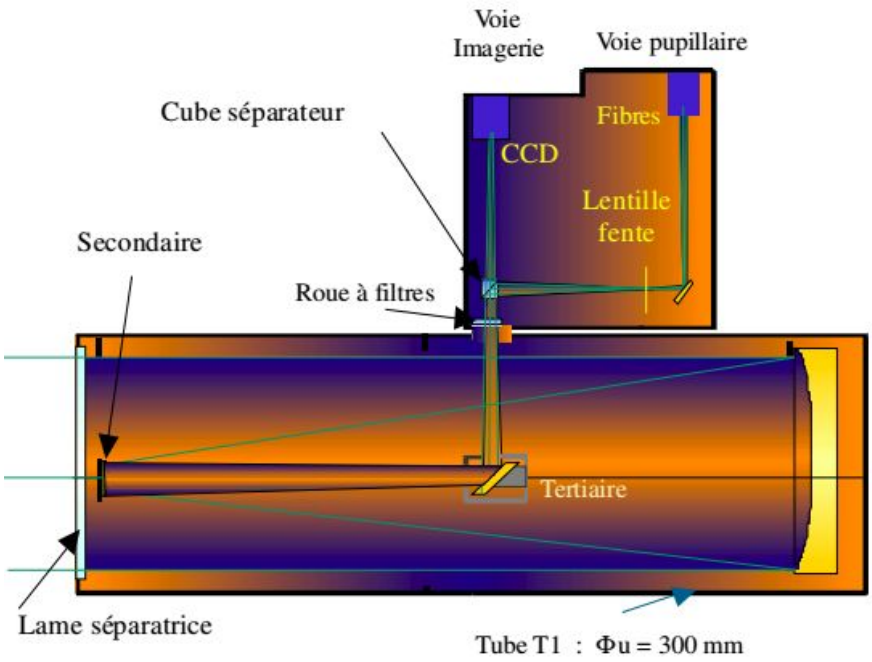


Schéma technique de MISOLFA

Ce télescope possède deux voies d'acquisition afin de mener à bien sa mission de caractérisation de la turbulence atmosphérique : la voie image et la voie pupille. Une roue à filtres permet d'effectuer les observations dans les mêmes longueurs d'onde que SODISM2, et une lentille fente permet de transformer les fluctuations de phase dues à la turbulence en fluctuations d'intensité sur une image de la pupille du télescope.

En tout, ce sont sept ordinateurs qui sont utilisés pour gérer ces deux télescopes. Les deux instruments nécessitent l'utilisation d'un système de pilotage afin de suivre le Soleil durant les observations, et une machine est utilisée pour chaque voie d'acquisition disponible. SODISM2 nécessite en plus un poste dédié à la régulation de la température de sa cuve et de sa caméra.

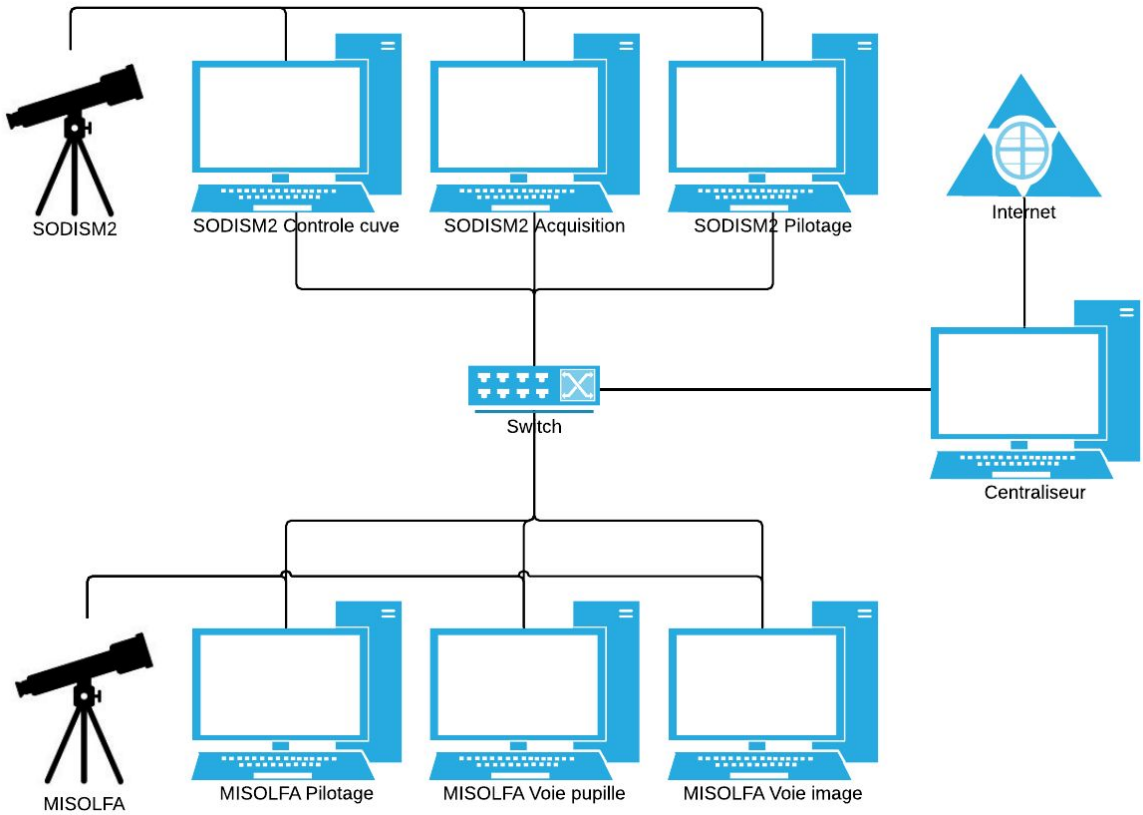


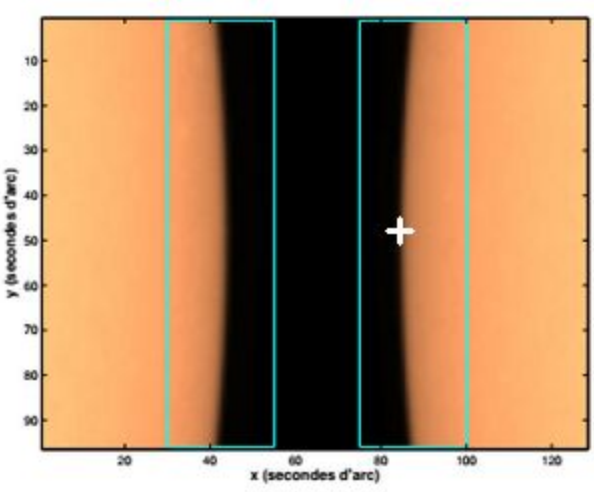
Schéma du matériel de l'instrumentation PicardSol

Enfin, le centraliseur, seul ordinateur connecté à internet, surveille et collecte les données acquises afin de les stocker sur un disque local et un serveur distant.

2.6. Voies d'acquisition et contrôle de MISOLFA

La monture de MISOLFA est contrôlée par le logiciel de pilotage, au travers d'une carte d'axes pour interagir avec les moteurs, la roue à filtres et la roue à fentes, et un module National Instruments permet de gérer les alimentations et certains signaux de sécurité. Les modules National Instruments sont des boîtiers électroniques qui peuvent être connectés par un port USB, et qui permettent d'interagir avec le monde physique au travers de voies numériques (lecture ou écriture d'états logiques) et de voies analogiques (lecture ou écriture de tensions).

La voie image de MISOLFA utilise une caméra d'une résolution de 640 par 480 pixels afin d'enregistrer des images de deux bords opposés du Soleil.

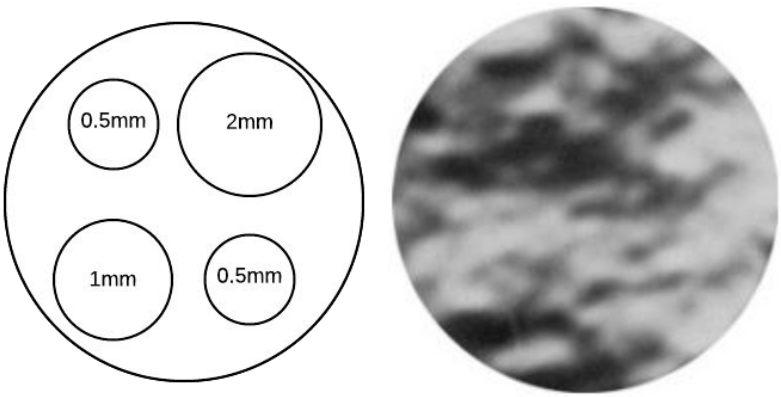


Images observées par MISOLFA,
avec le ventre du limbe signalé par une croix

Ces images permettent, par une observation des fluctuations du limbe solaire au cours du temps, de déterminer les paramètres spatiaux de la turbulence.

Comme chaque séquence d'acquisition de la voie image prend plus d'une minute, elle n'enregistre qu'une fois sur deux avec certains filtres. Enfin, le logiciel de la voie image recherche le ventre du limbe (croix blanche sur la figure), c'est-à-dire le point du bord du Soleil tangent à une droite verticale, et envoie des corrections de guidage au logiciel de pilotage.

La voie pupille, quant à elle, échantillonne à l'aide de fibres optiques l'intensité lumineuse en quatre points d'une image du miroir principal, plus une voie globale qui collecte le flux lumineux de toute l'image et qui permet de corriger sur les observations réalisées par les autres fibres les effets dus à une mauvaise qualité de l'environnement, comme par exemple le passage de nuages ou les erreurs de pointage du télescope.



Position des quatre fibres sur l'image du miroir primaire (à gauche)
et représentation de la fluctuation de l'intensité lumineuse (à droite)

Ces cinq fibres conduisent la lumière dans les boîtiers électroniques qui transforment le flux lumineux en une tension exploitable pour les recherches. Le schéma suivant décrit l'architecture du rack de la voie pupille.

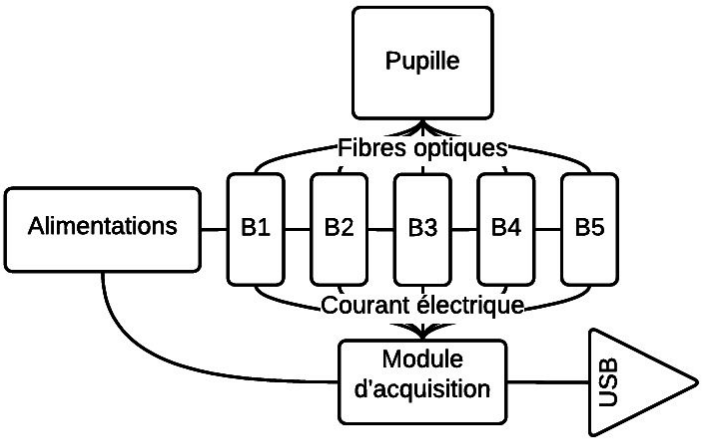


Schéma fonctionnel du rack de MISOLFA

2.7. Électronique de la voie pupille de MISOLFA

Dans chacun des boîtiers est placée une photodiode, qui capte la lumière et génère un courant proportionnel à l'intensité lumineuse. Ensuite, le signal délivré par la photodiode subit une première amplification au sein d'un préamplificateur qui peut offrir un gain faible ou un gain fort. Un filtre passe-haut et un filtre passe-bas sont ensuite utilisés, afin de filtrer des parasites. Ces deux filtres peuvent être désactivés. Avant d'être lu, le courant traverse un dernier amplificateur qui permet quatre gains différents d'une progression géométrique de raison deux.

Enfin, la photodiode est équipée d'un élément Peltier qui permet de la refroidir. Ce refroidissement est nécessaire afin de réduire et stabiliser le bruit qu'émet la photodiode, car la température est un facteur de bruit important. Ces éléments Peltier ne permettent pas d'atteindre une température de plus de 35 degrés inférieure à la température ambiante.

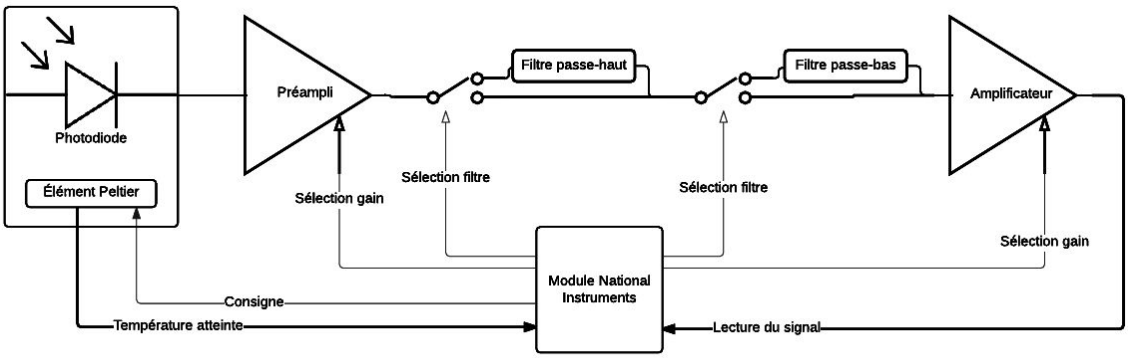


Schéma de l'électronique de la voie pupille,
en gras la voie signal et en clair les voies de contrôle

Le logiciel de la voie pupille communique avec un module National Instruments afin de lire les tensions grâce aux entrées analogiques et de modifier l'état des alimentations, des amplificateurs et des filtres au travers des sorties numériques.

3. Cahier des charges

3.1. Problème posé et solution existante

Le but du stage est de développer un logiciel permettant de faciliter et d'optimiser l'acquisition et l'enregistrement des données de la voie pupille de MISOLFA. MISOLFA possède déjà un logiciel minimal en exploitation, cependant il présente plusieurs limitations :

- Il ne monitore que les signaux des photodiodes ;
- Il est impossible d'agir sur la configuration des boîtiers durant une observation (gain en sortie, préampli…) ;
- Il est décomposé en trois programmes : allumage du rack et mise en place d'une configuration figée, enregistrement des données, arrêt du rack ;
- Les fichiers d'observations obtenus doivent être traités par le centraliseur avant d'être exploitables par les observateurs ;
- Il n'est pas synchronisé avec le reste de l'instrumentation (SODISM2 et voie image de MISOLFA) ;
- Il est incapable de s'adapter aux conditions d'observation, comme la météo.

Le nouveau logiciel doit donc remplacer le programme actuel en permettant de contrôler le rack d'acquisition, d'acquérir et enregistrer les données dans un format défini, suivre le plan d'observation, communiquer avec les autres outils et s'adapter aux conditions d'observation.

3.2. Exigences et contraintes de la maîtrise d'ouvrage

Les besoins exprimés par le maître d'ouvrage sont les suivants :

B1. Gérer le contrôle et la configuration initiale du rack
B2. Réaliser l'acquisition et l'enregistrement des données
B3. Synchroniser l'enregistrement des données avec SODISM 2
B4. Fournir un aperçu des données en cours d'acquisition
B5. S'adapter au contexte de l'observation
B6. Réaliser l'interface avec l'utilisateur

De plus, les contraintes suivantes sont appliquées :

C1. Utilisation du langage C++
C2. Développement sur une plateforme Windows XP
C3. Utilisation de l'environnement de développement C++ Builder
C4. Utilisation d'un fichier de configuration
C5. Réutilisabilité des composants

4.3. Fonctionnalités à implémenter

Au regard des besoins de ce projet, 7 fonctions de service ont été identifiées :

FS1. Configurer l'électronique du rack selon des paramètres prédéfinis au moyen d'un fichier de configuration, ou pendant l'exécution.

FS2. Lire les données depuis le rack. Elles comprennent les signaux science (tensions mesurées dans les quatre sous-pupilles), la tension de la voie globale de calibration et les données ancillaires (température du rack, état des filtres et amplis, etc).

FS3. Enregistrer en temps réel les données lues depuis le rack. Cela inclut les données science, les données ancillaires ainsi que les alarmes et l'historique des actions effectuées (demandées par l'utilisateur ou exécutées automatiquement).

FS4. Mettre en place une connexion entre le logiciel et les autres éléments du système (pilotage, voie image, centraliseur) en utilisant un protocole d'échange existant.

FS5. Fournir une visualisation "temps réel" des données ancillaires et science lues depuis le rack d'acquisition.

FS6. Évaluer les signaux acquis afin de modifier automatiquement l'état du système pour l'adapter aux conditions d'observation et optimiser la qualité des données.

FS7. Permettre à un utilisateur de modifier l'état du rack, d'observer les données acquises, de consulter l'historique des événements et de corriger d'éventuelles anomalies.

4.4. Planning prévisionnel

	Avril		Mai				Juin			
	17	18	19	20	21	22	23	24	25	26
Découverte du sujet										
Rédaction du cahier des charges										
Prise en main des outils et systèmes										
Analyse et conception										
Développement et documentation										
Tests unitaires										
Intégration et tests de validation										
Préparation du rapport et de la soutenance										

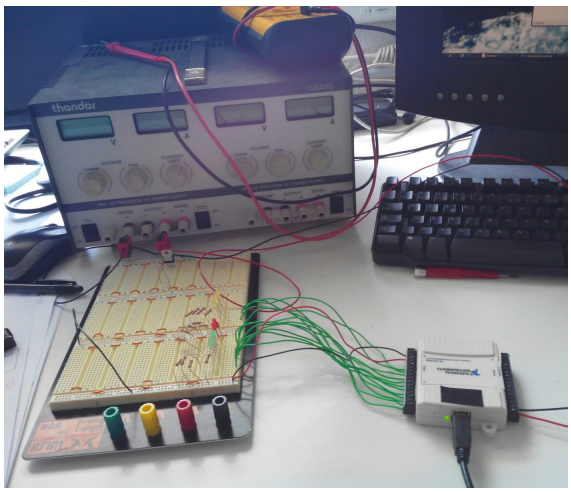
4. Analyse et conception

De longues discussions avec les acteurs concernés par l'utilisation de ce logiciel, c'est-à-dire l'équipe PICARDSOL, et notamment avec M. Morand et Mme Renaud, ont permis de cerner les besoins du problème et d'établir le cahier des charges du projet.

4.1. Matériel utilisé

Le matériel sur lequel doit fonctionner l'application est déjà en place depuis plusieurs années. C'est donc le système d'exploitation Windows XP qui est utilisé sur les machines de l'instrumentation.

Le module National Instruments utilisé pour réaliser les acquisitions et contrôler le rack est un NI USB-6212. Ce module possède 16 entrées analogiques, 2 sorties analogiques et 32 entrées/sorties numériques et il est capable de réaliser un échantillonnage de ses voies à 400 kilo-échantillons par seconde et par voie, ce qui est largement suffisant pour notre utilisation car un maximum de 1000 échantillons par seconde seront acquis pour chaque voie.



Le 6008, le générateur et la plaque de tests

Pour me familiariser avec les modules National Instruments, un second module a été mis à ma disposition, un NI USB-6008. Ce module possède huit entrées analogiques, deux sorties analogiques et 12 entrées/sorties numériques. De plus, des LEDs ont été montées sur une plaque de tests afin de pouvoir visualiser les sorties numériques. Le nombre d'entrées et de sorties que possède ce module ne permet pas de simuler l'utilisation des cinq voies simultanément, c'est pourquoi mon logiciel permet de configurer le nombre de signaux lus. D'autre part, un générateur et un voltmètre m'ont permis de tester les entrées et sorties analogiques.

4.2. Analyse

Afin de répondre à tous les besoins, mon programme doit interagir avec différents éléments du système. Le schéma suivant montre quels éléments vont être utilisés pour chaque fonction demandée.

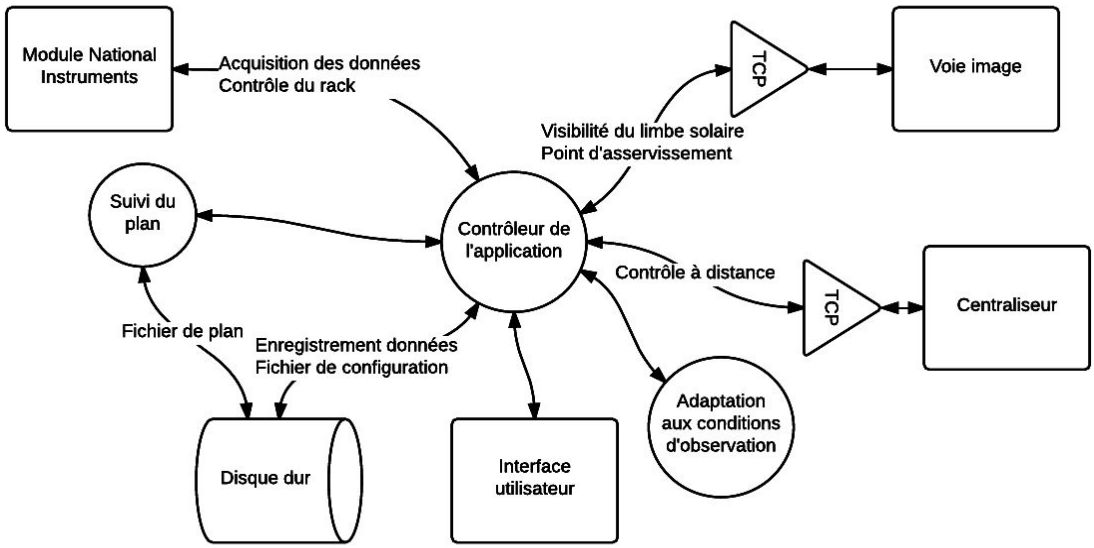
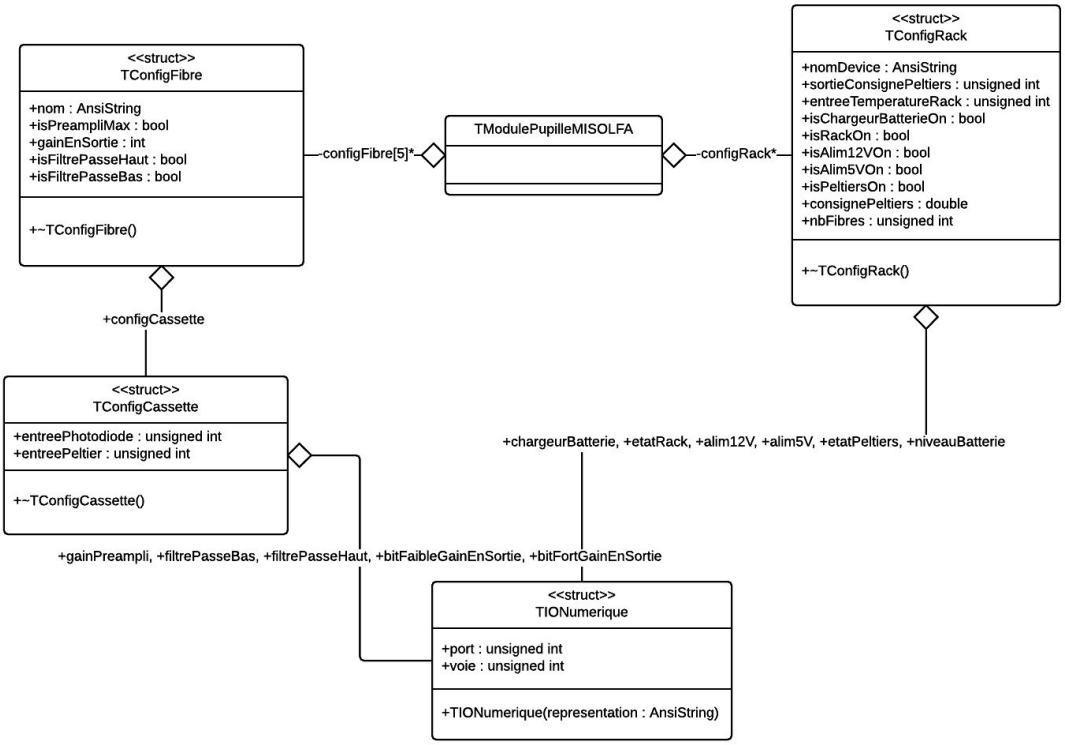


Schéma récapitulatif des fonctionnalités du logiciel

FS1 - Configuration

L'utilisation d'un fichier de configuration est nécessaire pour enregistrer les paramètres d'un programme afin de conserver la même configuration à chaque démarrage. Ces fichiers sont souvent écrits en ASCII, comme c'est le cas du format ini utilisé pour ce projet. L'avantage d'un fichier écrit dans ce format est qu'il est facilement transportable et éditable, que ce soit par un opérateur humain ou par un logiciel. Afin de stocker en mémoire les nombreux paramètres durant l'exécution de mon programme, des structures sont utilisées.



Digramme de classes des structures contenant la configuration

Ainsi, une structure est utilisée pour stocker la configuration générale du rack, et il y a autant de structures TConfigFibre que de voies déclarées dans le fichier de configuration. C'est dans ces structures que sont stockés les identifiants des voies numériques et analogiques à utiliser pour contrôler les composants électroniques et pour lire les températures ou les tensions des photodiodes.

FS 2 - Acquisition

Le contrôle du rack se fait au travers d'un module National Instruments. Une classe permettant de piloter de tels modules a été mise à ma disposition : TModuleNI. Il est nécessaire d'améliorer cette classe afin de permettre la lecture simultanée de plusieurs voies, ce qui n'est pas le cas dans la version fournie. La classe dérivée TModulePupilleMISOLFA que j'ai écrite permet de contrôler les alimentations et les gains à appliquer plus aisément.

FS 3 - Enregistrement

L'acquisition des données est réalisée en continu grâce à une des méthodes que j'ai ajoutée dans TModuleNI, et leur enregistrement sur le disque est effectué par une méthode de callback. Les fichiers de données générés par mon programme doivent être au format FITS.

Un fichier FITS (Flexible Image Transport System) est un type de fichier utilisé mondialement dans la recherche astronomique. Ainsi, tous les logiciels qui utilisent des données astronomiques sont normalement capables de lire des fichiers enregistrés dans ce

format. Ce format qui permet de stocker une ou plusieurs images dans un unique fichier, et permet d'adjoindre à ces images un en-tête, composé de mots-clés associés à une valeur. Il permet également d'enregistrer des tables de données ASCII ou binaires. Le contenu de l'en-tête est décidé par le créateur du fichier, mais doit respecter certaines normes, et certains mots-clés standardisés existent. Un mot clé d'un en-tête FITS ne doit pas dépasser 8 caractères, et chaque ligne du fichier ne doit pas excéder 80 caractères.

```
SIMPLE  =                    T / STANDARD FITS FORMAT (REV OCT 1981)
BITPIX  =                    8 / CHARACTER INFORMATION
NAXIS   =                    0 / NO IMAGE DATA ARRAY PRESENT
EXTEND  =                    T / THERE IS AN EXTENSION
ORIGIN  = 'ESO     '           / EUROPEAN SOUTHERN OBSERVATORY
OBJECT  = 'SNG - CAT.      '   / THE IDENTIFIER
DATE    = '27/ 5/84'           / DATE THIS TAPE WRITTEN DD/MM/YY
END
```

Sur cet exemple d'en-tête FITS, nous pouvons voir que les mots-clés ne dépassent pas 8 caractères et que la valeur associée est bien après un '='. Les textes après les '/' sont des commentaires pour les lecteurs.

FS 4 - Communication

Une communication avec la voie image doit être mise en place afin d'obtenir des informations concernant le pointage et la météo ou pour demander une modification du point d'asservissement du Soleil sur la fente. La voie image estime la qualité de l'image obtenue sous la forme d'une note, et cette note est utilisée par mon logiciel afin de déterminer si le limbe est suffisamment visible ou non.
Les échanges avec le centraliseur permettront de contrôler mon logiciel à distance, ainsi que de mettre à disposition des aperçus des données en cours d'acquisition.

Les messages échangés doivent respecter le protocole existant entre les autres machines, c'est-à-dire qu'ils doivent avoir la forme suivante :
$commande:param1;param2!
Les messages échangés doivent toujours commencer par un '$' et finir par un '!'. La première partie du message est la commande. C'est la seule partie obligatoire, une commande pouvant ne pas prendre de paramètre. Ensuite, s'il y a des paramètres, le symbole ':' est utilisé pour signaler la fin du nom de la commande, et les paramètres suivent, séparés par des ';'. Afin de faciliter la compréhension des messages échangés, les commandes sont des chaînes ASCII facilement lisibles par un opérateur. Cela permet de savoir quelles commandes sont échangées sans avoir à se référer à la documentation.

SODISM 2, quant à lui, n'est pas prévu pour fonctionner avec d'autres instruments et il n'est pas possible de modifier son programme pour l'adapter, ainsi il reste le clone de SODISM. Pour pouvoir se synchroniser avec lui, le seul moyen disponible est donc la lecture des fichiers de plan qu'il génère dans la journée.

Le centraliseur récupère la date de début et le type de plan que génère SODISM2, et il crée un nouveau fichier qu'il distribue aux deux ordinateurs d'acquisition de MISOLFA. Le fichier généré est de la forme suivante :

```
09 33 34350 WL535DL
09 37 34590 WL607DL
09 40 34770 WL535HL
09 44 35010 WL607DL
```

Sur chaque ligne, nous pouvons lire de gauche à droite, l'heure TU de début d'acquisition, la minute de début d'acquisition, le temps en seconde de début d'acquisition et le filtre à utiliser.

FS 5 et 7 - Interface utilisateur

L'interface graphique de l'application est construite grâce à l'éditeur proposé par C++ Builder. La gestion des événements est ainsi simplifiée car l'IDE prépare le squelette d'une méthode à chaque événement que je souhaite traiter.

FS 6 - Adaptation aux conditions d'observation

Enfin, un module 'intelligent' est développé afin de permettre une adaptation automatique du programme et du rack selon les conditions d'observation, comme la météo ou un problème instrumental. C'est ce module qui va permettre de détecter des problèmes et de déclencher des alarmes afin de prévenir l'utilisateur.

Pour cette partie, il m'a d'abord fallu discuter avec M. Morand de ce qui peut perturber les observations sur la voie pupille, analyser comment cela peut être détecté lors de l'observation et enfin quelles actions peuvent être prises afin d'améliorer la qualité des observations. Une des difficultés de cette partie réside dans le fait que les cinq voies ne sont pas affectées de la même manière car elles correspondent à des diamètres de fibre différents et qui sont positionnées en différents points de l'image. Ainsi, j'ai réalisé un organigramme pour chaque situation.

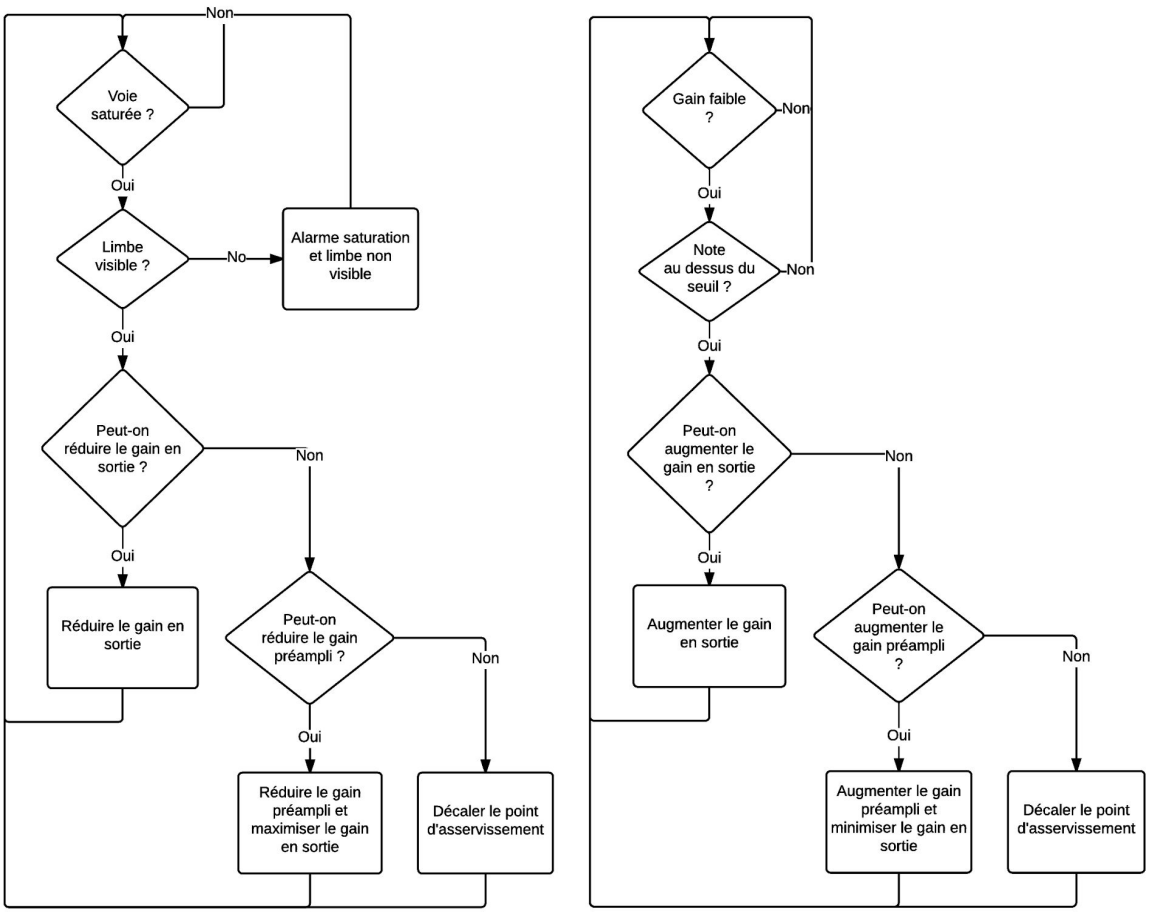
Quatre situations ont été identifiées, et j'ai réalisé un arbre de décision pour chacune d'entre elles :

- Signal saturé,
- Signal faible,
- Aucun signal,
- Température de la photodiode supérieure à la consigne.

Signal saturé ou faible

Si une voie sature, la première chose à faire est de jouer sur les gain des deux amplis, en commençant par le gain en sortie. Si malgré cela la voie sature toujours, il est nécessaire de demander la modification du point d'asservissement afin de s'éloigner du centre du Soleil.

Si, au contraire, une voie n'a pas assez de signal, la manipulation inverse est effectuée.

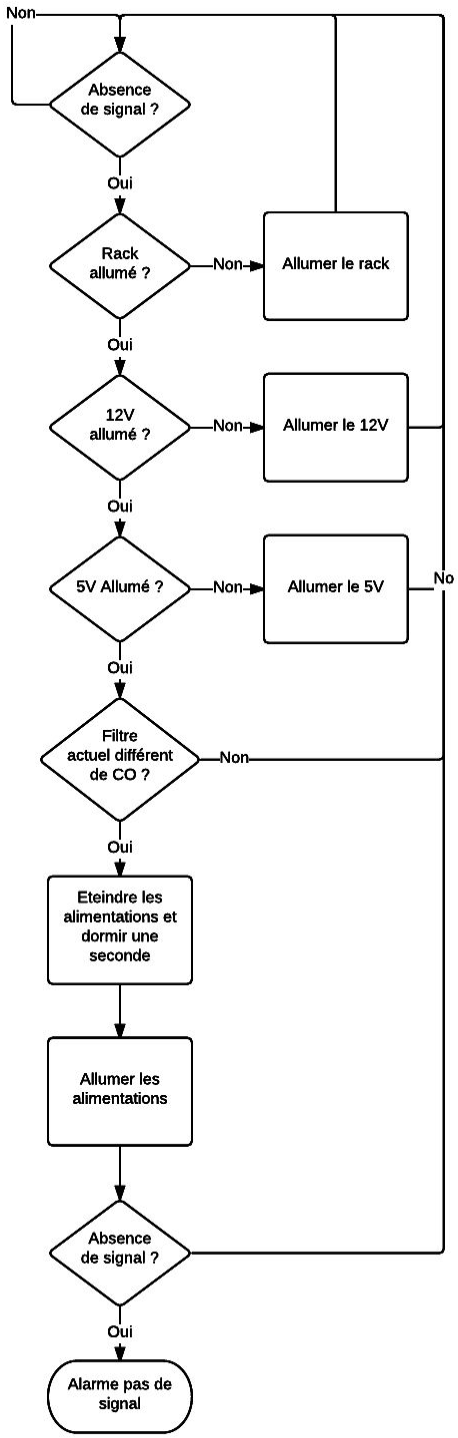


Organigrammes décrivant les actions à suivre dans le cas d'une voie saturée ou d'un signal faible

Dans les deux cas, la météo doit être prise en compte. C'est la note envoyée par la voie image qui sert de référence.

Absence de signal

L'absence de signal peut être due à deux choses : la roue à filtres est en position cache ou une alimentation n'est pas allumée.
Après avoir vérifié dans le plan qu'une acquisition en courant d'obscurité n'est pas programmée, mon logiciel vérifie l'état des alimentations et les allume si nécessaire. Si cela est insuffisant et que le signal reste nul, toutes les alimentations sont éteintes, puis rallumées. Si le problème persiste, une alarme est déclenchée et un message informe l'utilisateur du problème, l'invitant à vérifier que le rack est bien allumé, et que les branchements sont corrects.

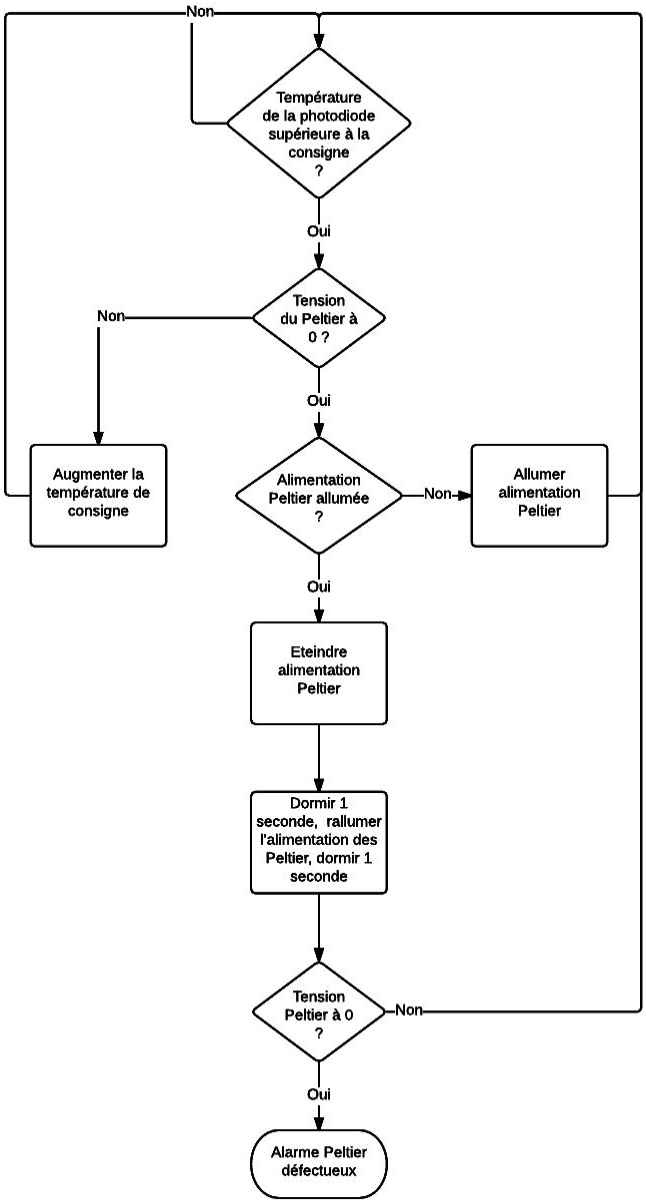


Organigramme décrivant les actions à suivre en cas d'absence de signal

Évidemment, il ne faut pas que mon logiciel tente de corriger une situation passagère, comme le passage d'un nuage, ou une perte complète du limbe solaire. Pour cela, la note envoyée par la voie image est utilisée : en dessous d'un certain seuil, le module d'adaptation au conditions d'observations est arrêté, et dès que le limbe solaire n'est plus visible, les observations et le module d'adaptation sont arrêtés.

Température d'une photodiode supérieure à la consigne

Les photodiodes sont refroidies afin de réduire le bruit du signal, mais les modules Peltier sont limités : ils ne peuvent pas atteindre une température inférieures de 35 degrés à la température ambiante. Il est préférable d'avoir une température de fonctionnement légèrement supérieure à -10°C mais stable plutôt que l'inverse. En effet, une température instable augmente le bruit plutôt que de le réduire. C'est pourquoi, dès qu'une photodiode peine à atteindre la température de consigne, cette consigne est augmentée.



Organigramme décrivant les actions à suivre lorsque la température d'une photodiode n'atteint pas la consigne

À chaque changement d'état de l'alimentation, il est nécessaire de réaliser une pause afin de permettre aux éléments de se mettre en température.

5. Implémentation

5.1. Configuration

Un maximum de paramètres doivent être lus depuis le fichier de configuration, afin d'augmenter la flexibilité du logiciel. Une classe permettant de lire et d'écrire des fichiers au format .ini, TGestionINI, m'a été fournie. Les fichiers ini peuvent être décomposés en paragraphes, dans lesquels sont rangés des paramètres. Pour cette application, les paragraphes ont été définis comme suit :

GENERAL : Contient toutes les informations générales du logiciel, telles que la fréquence de rafraîchissement de l'affichage, la fréquence d'enregistrement, les répertoires d'enregistrement des données acquises, l'emplacement du fichier de plan, les adresses ip des machines avec lesquelles des échanges se font et le rack utilisé.

RACK : Cette catégorie contient les informations relatives au rack d'acquisition : le nombre de fibres, les sorties numériques des alimentations, la sortie analogique de la consigne des éléments Peltier et la consigne à leur appliquer et la fréquence d'acquisition.

FIBRE : Un paragraphe doit être écrit pour chaque fibre, et chacun de ces paragraphes doit se nommer "Fx", où x est le numéro de la fibre, allant de un au nombre de fibres présentes. Chacun de ces paragraphes contient le nom de la fibre, le nom du paragraphe du boitier associé, la valeur du gain en sortie, le réglage du préampli, du filtre passe-bas et du filtre passe-haut.

BOITIERS : Ces paragraphes contiennent les identifiants des voies numériques ou analogiques sur lesquelles chaque composant électronique est branché, et indiquent donc comment communiquer avec les filtres et les amplificateurs. Ainsi, si le câblage du rack est modifié, il n'est pas nécéssaire de modifier le corps du programme.

SEUILS : Ce paragraphe est utilisé par le module intelligent du programme. Il permet de définir des seuils à partir desquels le logiciel doit réagir face à un problème.

Tous les paramètres sont modifiables depuis l'interface ou à distance, mais les modifications ne sont pas enregistrées systématiquement car le fichier de configuration doit correspondre à un fonctionnement dans des conditions normales d'observations. En revanche, il est possible d'enregistrer la configuration en cours afin de permettre une modification aisée du fichier de configuration, si ces changements sont définitifs.

Le logiciel lit le fichier de configuration au lancement et applique directement les paramètres, ainsi l'utilisateur n'a qu'à lancer le logiciel et démarrer les acquisitions. Le fichier de configuration obtenu à la fin du stage est disponible en annexe 4.

5.2. Journalisation

La journalisation des événements survenant lors de l'exécution du logiciel est nécessaire car cela permet de conserver une trace de l'exécution du logiciel afin de pouvoir comprendre des éventuelles observations anormales ou un plantage du programme.

Afin de réaliser l'écriture sur le disque de ces événements, la classe TJournal, utilisée par les autres logiciels de l'instrumentation, m'a été fournie. Cette classe permet de gérer l'écriture dans un fichier avec un format normalisé pour la date et l'heure.

En plus de cette écriture dans un fichier, l'événement est affiché à l'écran afin de pouvoir être vu directement par l'opérateur. La tâche est facilitée par l'utilisation d'un composant

graphique de C++ Builder qui gère automatique la mémoire et le nombre de lignes affichées. Au total, trois de ces éléments sont utilisés : un premier pour consigner les changements d'état du système (allumage d'une alimentation, changement du gain d'un ampli…) et deux autres pour les communications, afin de différencier aisément les messages échangés avec le centraliseur et la voie image.

5.3. Acquisition des données

Les données devant êtres lues à une fréquence de 1000Hz, nous ne pouvons pas faire confiance à l'horloge de l'ordinateur et il est donc nécessaire de laisser le module d'acquisition gérer le temps. Pour cela, le SDK de National Instruments propose une fonction permettant de créer une tâche d'acquisition continue qui appelle une fonction de callback dès qu'un nombre de points défini a été acquis. Malgré le fait que seuls les signaux issus des photodiodes ont besoin d'être enregistrés avec une telle fréquence, les températures des voies et du rack sont acquises à la même fréquence car le module National Instruments n'est pas capable d'effectuer simultanément plusieurs tâches d'acquisition à des fréquences différentes.
La méthode d'acquisition continue que j'ai ajoutée à la classe TModuleNI nécessite la création d'un objet dérivé de la classe abstraite TCallback, ce qui permet à l'utilisateur de définir la méthode de callback à utiliser.

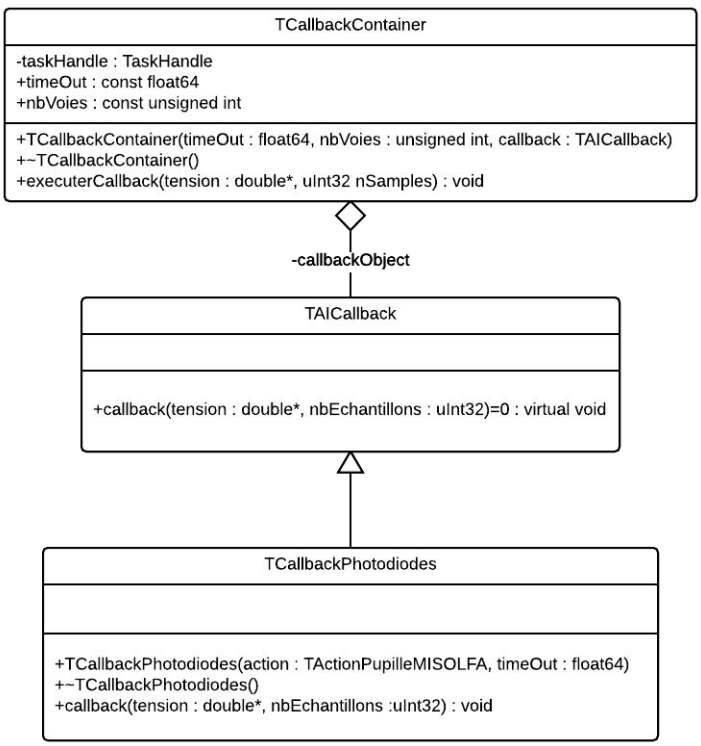


Diagramme de classe de l'implémentation de la callback

La classe TCallbackContainer est connue uniquement de TModuleNI et permet de stocker les informations nécessaires à la récupération des tensions. Enfin, la callback reçoit, à chaque appel, les tensions acquises durant la dernière période et la taille du tableau qui les contient.

Pour l'affichage et l'enregistrement des tensions, il m'a fallu définir le nombre de chiffres significatifs. Le module National Instruments utilisé pour les acquisitions, un NI USB-6212, échantillonne les voies analogiques sur 16 bits, et le SDK permet de définir un seuil de valeurs attendues afin d'améliorer la précision des lectures.
Le signal de photodiodes est attendu comme étant entre -10V et 0V, ce qui donne une plage de 10V. Le résultat étant encodé sur 16 bits, nous pouvons trouver le nombre de chiffres significatifs à conserver lors de enregistrements : $10/2^{16} = 0.0001525V$. Cinq chiffres significatifs sont donc à conserver.
Les tensions correspondant aux températures des photodiodes sont situées entre 0V et 1V, ce qui laisse une plage de seulement 1V. Six chiffres sont donc significatifs car la précision est de plus ou moins $1/2^{16} = 0.00001525V$.

Lors d'une acquisition de plusieurs points par voies sur un module National Instruments, deux modes de réception des données sont proposés : le mode *entrelacé* ou *non-entrelacé*. Le premier mode permet d'obtenir un tableau contenant les données rangées par date d'acquisition plutôt que par voie, tandis que le second privilégie un rangement par voie plutôt que par date.

Entrelacé	Non-entrelacé
Voie 1 - Échantillon 1	Voie 1 - Échantillon 1
Voie 2 - Échantillon 1	Voie 1 - Échantillon 2
Voie 3 - Échantillon 1	Voie 1 - Échantillon 3
Voie 1 - Échantillon 2	Voie 2 - Échantillon 1
Voie 2 - Échantillon 2	Voie 2 - Échantillon 2
...	...

Exemple de mode entrelacé et non-entrelacé

J'ai choisi d'utiliser le mode entrelacé car l'enregistrement des données collectées s'effectue dans cet ordre, c'est-à-dire par date d'acquisition.

5.4. Enregistrement des données

Les données n'étant pas traitées en temps réel, il est nécessaire de les enregistrer sur le disque. Deux modes d'enregistrement sont disponibles : le mode manuel, qui enregistre les tensions en continu, et le mode suivi de plan qui enregistre les données selon un plan. Ce mode est expliqué plus tard. Durant une utilisation en mode manuel, le logiciel génère un simple fichier ASCII et utilise le format suivant :
*YYYY-MM-DD*T*hh:mm:ss:zzz F1 F2 F3 F4 F5*
où *YYYY* est l'année en cours, *MM* le mois, *DD* le jour, *hh* l'heure, *mm* la minute, *ss* la seconde et *zzz* les millisecondes, suivis par les tensiosn relevées pour chaque photodiode.

A chaque rappel de la callback, les données sont écrites à la suite du fichier, ce qui permet de ne pas perdre de données si un problème instrumental apparaît.

Le mode suivi de plan, quant à lui, n'enregistre les données qu'à la fin d'une séquence d'observation. En effet, la perte d'une seule séquence d'observation en cas de problème n'est pas catastrophique. Les fichiers sont générés au format FITS, ce qui permet de ne pas avoir à traiter le fichier de sortie avant son exploitation. De plus, dans l'en-tête de ces fichiers, un mot-clé est utilisé afin de préciser si l'enregistrement est complet ou non, ce qui permet de savoir si les enregistrements ont été interrompus, que ce soit par l'utilisateur ou par le logiciel du fait d'un changement des conditions d'observation.

Pour ce projet, deux bibliothèques C++ permettant d'écrire des fichiers FITS ont été identifiées grâce au *FITS support office*, un site internet maintenu par la NASA. La première, CCfits, ne propose pas de fichier pré-compilé. J'ai donc essayé de la construire avec Borland C++ Builder 6, mais cela s'est avéré infructueux, faute d'instructions et d'incapacité manifeste de Builder 6 à comprendre certaines parties du code. Des instructions étant données pour compiler CCfits grâce à Visual Studio, j'ai pu compiler avec succès la bibliothèque. Malheureusement, Builder 6 a été incapable de l'exploiter car le format de DLL qu'il utilise est différent du format généré par Visual Studio, et les utilitaires de conversion se sont avérés inefficaces.
La seconde, AIPS++ FITS library, n'est proposée avec aucune forme d'instructions, ni même de manuel utilisateur et je n'ai pas réussi à la compiler, que ce soit avec Visual Studio ou C++ Builder 6.
J'ai donc opté pour l'utilisation de la bibliothèque C CFITSIO, qui est proposée en tant que fichier pré-compilé, et dont le manuel utilisateur est très bien écrit. J'ai donc encapsulé cette bibliothèque dans une classe afin de pouvoir réaliser les opérations nécessaires à ce projet.

Dès qu'un enregistrement est en cours, toute modification des paramètres de configuration du rack est bloquée afin de ne pas avoir d'incohérence dans les enregistrements.
De plus, il est possible de choisir ce qui est enregistré dans le fichier d'enregistrement manuel. Un fichier peut donc contenir :

- Les seuls relevés des photodiodes et l'heure toutes les 500 acquisitions ;
- Les relevés des photodiodes et de leur température, ainsi que l'heure toutes les 500 acquisitions ;
- Les relevés des photodiodes et de leur température, ainsi que l'heure à chaque ligne (toutes les millisecondes) ;
- Les relevés des photodiodes et l'heure à chaque ligne.

Le dernier mode d'écriture cité est celui utilisé lors du fonctionnement nominal, et les fichier FITS générés en mode suivi de plan utilisent obligatoirement ce format.

Lors du fonctionnement du logiciel, un fichier de servitude est également généré. Ce fichier contient des informations relatives à l'environnement, ce sont des données ancillaires. Les données ancillaires qui sont écrites comprennent la température du rack, des photodiodes et de leur température de consigne Ces informations peuvent être utilisées afin de comprendre un éventuel comportement étrange des données collectées, ou simplement pour vérifier le bon fonctionnement de l'instrument.

5.5. Synchronisation avec SODISM 2

Un thread permet de suivre le plan. Il est démarré dès le passage en mode suivi de plan, et il cherche le fichier de plan à l'endroit désigné par le fichier de configuration. Si le fichier n'est pas encore présent, l'utilisateur en est informé et le thread attend qu'il soit disponible. Dès que le fichier est présent, il est lu afin de trouver la prochaine date de début d'enregistrement. Les acquisitions de MISOLFA doivent commencer 30 secondes avant la prise de vue de SODISM 2 et s'arrêter 30 secondes après et le plan fourni prend en compte ce décalage. Seul les deux derniers éléments me sont utiles, l'heure et la minute n'étant présents que pour faciliter la lecture du fichier par un éventuel opérateur. Le temps en seconde est donc utilisé pour décider du moment auquel démarrer l'enregistrement, et le filtre utilisé est enregistré dans l'en-tête des fichiers de données. La durée d'un enregistrement est bien entendu paramétrable via le fichier de configuration.

5.6. Communication

Afin de gérer ces deux connexions, un composant socket proposé par C++ Builder sera utilisé. Ce composant facilite l'implémentation et l'utilisation d'une connexion utilisant le protocole TCP/IP.

Pour pouvoir piloter la voie pupille à distance, un dictionnaire de commandes a été mis en place afin de pouvoir contrôler chaque paramètre. Ce dictionnaire est documenté en annexe 3.

Dès réception d'une nouvelle note de la part de la voie image, mon logiciel vérifie que cette note est en dessous du seuil, et arrête les enregistrements le cas échéant. Cette socket permet également à mon logiciel de demander une correction du point d'asservissement grâce a la commande OFFSET et en passant comme paramètre le déplacement horizontal du point de fonctionnement, en pixels.

La socket utilisée pour le centraliseur est passive, c'est à dire que aucun message n'est envoyé par mon logiciel sans sollicitation préalable. Le centraliseur peut demander à tout moment des informations concernant l'état du logiciel, comme par exemple si un enregistrement est en cours ou non.

5.7. Interface

J'ai réalisé l'interface en même temps que je développais les fonctionnalités du logiciel. Elle est donc en constante évolution depuis le début du projet. Afin de la construire, j'ai utilisé l'interface que propose C++ Builder ainsi que ses composants graphiques.

Capture d'écran de l'interface

Les boutons sont groupés par fonction ou par élément modifié. Sur cette image, ces groupes sont colorés. En bleu, les paramètres qui ont un effet sur l'intégralité du rack : la gestion des alimentations et de la consigne des modules Peltier. En vert, ce sont les informations des voies, comme la tension délivrée par la photodiode ou la température de la voie. C'est également dans cette partie que l'utilisateur peut modifier les paramètres des voies. Les zones blanches et violettes correspondent aux journaux. Le blanc contient l'historique des actions sur le rack, tandis que les deux dans la zone violette contiennent les messages échangés avec, de haut en bas, le centraliseur et la voie image. Lorsque une erreur survient, la zone d'état, en rouge sur l'image, devient rouge et un message décrivant l'erreur apparaît afin que l'utilisateur corrige l'état d'erreur.
Enfin, l'utilisateur peut changer le mode d'observation grâce aux boutons de la zone jaune. Si des enregistrements sont en cours lorsqu'il modifie le mode de fonctionnement, une boite de dialogue l'en avertit. Durant un enregistrement, comme il n'est pas permis de modifier les paramètres du rack, tous les contrôles sont désactivés, sauf l'arrêt de l'enregistrement.

5.8. Adaptation aux conditions d'observation

Cette partie a été implémentée grâce à un thread qui fonctionne dès que l'utilisateur active l'adaptation aux conditions d'observation. Ce thread surveille les signaux définis durant l'analyse et prend, le cas échéant, les mesures nécessaires à la correction des problèmes. Il est important que ce thread soit régulièrement endormi afin de permettre au système de prendre en compte les modifications.

6. Tests

Durant ce stage, les tests ont été effectués de manière simultanée avec le développement, afin de vérifier que la fonctionnalité développée correspondait à la fonctionnalité attendue. De plus, des tests supplémentaires ont été réalisés et sont décrits ici.

6.1. Tests unitaires

L'ajout de fonctionnalités à la classe TModuleNI, permettant le contrôle du module d'acquisition National Instruments, nécessite des tests afin de vérifier que le module réagit effectivement comme souhaité.

Test du temps d'écriture sur disque

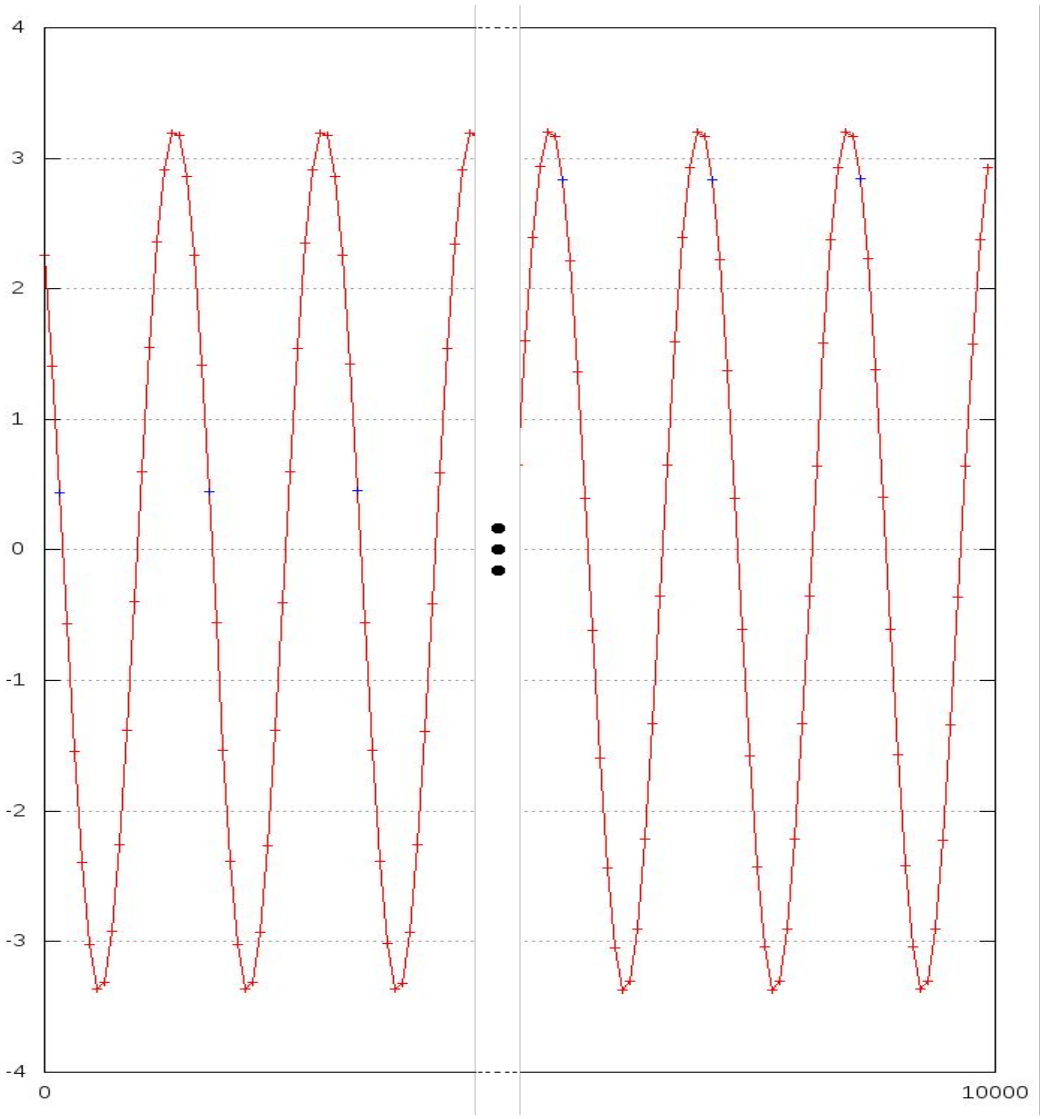
La première chose que j'ai testé est le temps d'écriture des données sur le disque. En effet, il ne faut pas que l'écriture dépasse 500 millisecondes, sous peine de gêner l'acquisition de ces données, quel que soit le mode d'acquisition.
Pour le mode manuel, j'ai effectué ce test avec le plus gros volume de données qu'il est possible d'écrire avec mon logiciel, c'est à dire lorsque sont enregistrés les relevés des températures ainsi que ceux des photodiodes et que l'heure est écrite à chaque ligne. Cela représente environ 119ko de données à écrire sur le disque chaque seconde. Le temps d'écriture constaté est inférieur à 20ms, ce qui est largement suffisant.
Pour le mode suivi de plan, le volume de données est nettement supérieur puisque ce n'est plus 500 millisecondes d'acquisitions qui doivent être écrites, mais une minute. En revanche, les températures ne sont plus enregistrées à chaque ligne, mais une seule fois par fichier. Cela représente tout de même plus de 4Mo de données à écrire en moins de 500ms. Le temps d'écriture constaté est inférieur à 80ms, et reste donc correct.

Test de restitution du signal

Afin de réaliser un échantillonnage continu à une fréquence donnée des voies analogiques, il est nécessaire de démarrer une tâche d'acquisition en lui définissant la fréquence d'acquisition, les voies analogiques à échantillonner et une callback qui sera appelée dès que le nombre de points souhaité aura été acquis. J'ai donc réalisé un test avec un oscilloscope, un générateur de fonctions et mon logiciel, l'objectif étant de vérifier que la sinusoïde générée est bien retrouvée lors de l'acquisition et l'enregistrement des données, et à la même fréquence.

Montage utilisé pour tester l'exactitude temporelle du boîtier

Sur ces deux courbes, l'acquisition est réalisée à une fréquence de 1000Hz, ce qui signifie qu'une milliseconde doit séparer deux points, et la fonction générée est un sinusoïde d'une fréquence de 50Hz, c'est à dire d'une période de 20 millisecondes. Tous les vingt points, un point est donc mis en bleu.



Courbe d'échantillonnage d'une sinusoïde de 50Hz à une fréquence de 1000Hz

Nous pouvons constater qu'après environ 10 secondes, le point bleu s'est déplacé sur la période de la sinusoïde. D'après les spécifications techniques du module fournis par le constructeur, ce décalage n'est pas anormal. Ces variations ont été prises en compte par l'équipe durant la conception de l'instrument et ne gênent en rien les acquisition.

Datation des enregistrements

Lors des tests de la fonction d'enregistrement, un décalage périodique de l'horloge de l'ordinateur d'une dizaine de millisecondes est apparu. Cela pose problème si ce saut apparaît au milieu d'une minute d'acquisition puisque deux lignes peuvent ainsi avoir la même date, ce qui ne devrait pas être possible. Afin de comprendre l'origine de ce décalage, j'ai écrit un logiciel qui utilise l'horloge interne d'un module National Instruments et enregistre l'heure de l'ordinateur à intervalles réguliers. Grâce à ce logiciel, les sauts ont pu être quantifiés : toutes les cinq à six minutes, un retour de 15 millisecondes a lieu. Au vu de la régularité du saut, il en a été déduit que ce décalage est dû à la synchronisation avec le serveur de temps, qui est programmée pour arriver toutes les cinq minutes. Puisque

l'apparition d'un tel décalage au milieu d'une séquence d'acquisition perturberait l'exploitation des données, il a été décidé de ne prendre l'heure qu'une seule fois, au début d'une séquence d'acquisition et de s'en servir pour dater chaque ligne du fichier.

6.2. Test de déploiement

Afin de vérifier le bon fonctionnement de mon logiciel, il a été mis en fonctionnement pendant plus d'une semaine sur l'instrumentation. Cela a permis de détecter certains problèmes, comme des fuites de mémoire, et a permis aux utilisateurs de tester le logiciel durant des conditions normales d'utilisation. Des problèmes de fonctionnement de l'interface ont ainsi pu être mis en évidence, comme le besoin d'avoir une barre d'état indiquant ce qu'est en train de faire le logiciel (rien, enregistre, attend le début d'une séquence d'enregistrement…).
C'est également durant un de ces tests que j'ai constaté que le module National Instruments utilisé n'était pas capable d'acquérir plusieurs voies simultanément à une fréquence différente.

7. Résultats

7.1. État final du produit par rapport au cahier des charges

Le logiciel développé est en exploitation et remplace l'ancien. La configuration, la gestion du rack, l'acquisition et l'enregistrement des données, la synchronisation avec SODISM2 et la communication avec les autres machines sont complètement réalisés et testés.
En revanche, la partie intelligente du programme, c'est-à-dire l'adaptabilité aux conditions d'observation, n'a pas pu être entièrement testée car les conditions météorologiques ne le permettait pas. Les seuils de réaction sont encore à définir grâce au fichier de configuration. C'est un travail de longue haleine qui nécessite beaucoup d'expérimentations et qui dépend de la météo, ce qui limite les périodes de test.

7.2. Planning constaté en fin de stage

Après ces dix semaines de stage, je constate que le planning effectif ne diffère pas de mon planning prévisionnel, mais plusieurs semaines supplémentaires seront nécessaires pour mettre au point l'adaptation aux conditions d'observation.

7.3. Quantification

	Total
Nombre de classes	15
Nombre de méthodes	227
Lignes de code	4088
Lignes de commentaire	805

Conclusion

Durant ce stage de fin de DUT au sein de l'Observatoire de la Côte d'Azur, j'ai eu pour mission de développer un logiciel permettant le remplacement de l'existant, avec une contrainte importante qui est l'utilisation du langage de programmation C++.

Plus qu'un simple remplacement de programme, ce stage comprenait également l'ajout de nouvelles fonctionnalités afin de faciliter la tâche d'enregistrement des données aux observateurs, tout en réduisant la quantité de données inutiles ou inexploitables.

L'outil développé propose donc toutes les fonctionnalités demandées et remplace déjà l'ancien logiciel. Le code produit est commenté et devrait être facilement maintenable par l'équipe PicardSol, et un grand nombre de modifications sont réalisables au travers du fichier de configuration.

Ce stage m'a permis de découvrir le milieu professionnel, et plus précisément celui de la recherche. Cette expérience m'a permis de mettre en application les connaissances acquises durant ma formation tout en me permettant d'en découvrir de nouvelles et d'en perfectionner d'autres.

De plus, l'interaction avec l'électronique m'a permis de découvrir un moyen d'agir sur le monde physique et la nécessité de comprendre certains concepts d'astrophysique afin de mieux cerner les besoins des utilisateurs a grandement contribué à l'intérêt de ce stage.

Glossaire

SODISM : SOlar Diameter Imager and Surface Mapper, télescope ayant pour but l'acquisition d'images du soleil dans différentes longueurs d'onde. Deux copies existent, une dans l'espace, à bord du satellite PICARD, et une au sol, sur le site de Calern.

MISOLFA : Moniteur d'Image SOLaire Franco-Algérien, télescope servant à quantifier la turbulence atmosphérique. Il est installé sur le site de Calern, à côté de SODISM 2.

OCA : Observatoire de la Côte d'Azur

UNS : Université de Nice Sophia Antipolis

IRD : Institut de Recherche pour le Développement, organisme français de recherche

CNRS : Centre National de la Recherche Scientifique, organisme français de recherche

ASCII : American Standard Code for Information Interchange, norme d'encodage de caractères contenant les caractères nécessaires pour écrire en anglais.

FITS : Flexible Image Transport System, format de fichier principalement utilisé dans l'astronomie permettant de stocker des images et des tables de données ASCII ou binaires.

Bibliographie

Documentation en ligne des fonction du SDK de National Instruments
http://zone.ni.com/reference/en-XX/help/370471W-01/

Site de l'Observatoire de la Côte d'Azur
https://www.oca.eu/

Site du Laboratoire Lagrange
https://lagrange.oca.eu/

Documentation en ligne de la librairie cfitsio
http://heasarc.gsfc.nasa.gov/docs/software/fitsio/c/c_user/cfitsio.html

Table des annexes

Annexe 1

Cahier des charges
Logiciel voie pupille MISOLFA

20 Avril - 28 juin 2015

Metge Alexis

Table des matières

9. Objet du document

9.1. Présentation du sujet

Ce projet fait l'objet d'un stage de fin d'études de 10 semaines au sein du laboratoire Lagrange de l'Observatoire de la Côte d'Azur, sur l'instrumentation PICARDSOL, depuis le site d'observation de Calern.

PICARDSOL a pour objet de mesurer le diamètre solaire et de quantifier les effets de la turbulence atmosphérique sur les observations réalisées au sol en les comparant aux observations réalisées dans l'espace. Il est constitué de deux instruments principaux :

- SODISM2, qui mesure le diamètre solaire,
- MISOLFA, qui quantifie la turbulence atmosphérique.

De plus, MISOLFA réalise des acquisitions grâce à ses deux voies :

- La voie image enregistre les images de deux bords opposés du Soleil.
- La voie pupille mesure les fluctuations d'intensité prélevées en 4 points d'une image de la pupille.

Le stage porte sur la réalisation d'un logiciel de contrôle et d'acquisition de la voie pupille de MISOLFA.

9.2. But du document et de son usage

Le but du projet est de réaliser un logiciel de contrôle de la voie pupille de MISOLFA permettant d'acquérir, enregistrer, visualiser et synchroniser l'acquisition de données et de réaliser une adaptation automatique en fonction des conditions d'observation.

Ce document est un cahier des charges destiné à servir de référence pour le développement et la mise en place du logiciel.

10. Terminologie

SODISM : SOlar Diameter Imager and Surface Mapper, télescope ayant pour but l'acquisition d'images du soleil. Deux copies existent, une dans l'espace et une au sol, sur le site de Calern.

MISOLFA : Moniteur d'Image SOLaire Franco-Algérien, télescope installé sur le site de Calern, à côté de SODISM 2.

11. Contexte et motivation de l'action

Le satellite d'observation PICARD comporte différents instruments dont SODISM qui acquiert des images du Soleil dans 5 longueurs d'ondes différentes sans être perturbé par l'atmosphère terrestre. Il a été décidé d'installer au sol une réplique de SODISM, SODISM 2, afin de pouvoir comparer les images acquises simultanément hors et dans l'atmosphère. MISOLFA est installé à côté de SODISM 2 afin de permettre la quantification des perturbations atmosphériques.

Actuellement, la procédure d'acquisition de données de la voie pupille du télescope MISOLFA est figée et ne tient pas compte de son environnement instrumental (enregistrements non synchronisés avec SODISM 2 et la voie image) ou météorologique. Or les données collectées par MISOLFA ne sont utiles que lorsqu'elles sont comparées avec les observations réalisées par SODISM2 et de bonne qualité, ce qui signifie qu'une grande quantité de données enregistrées est inutile.

Afin de pouvoir supprimer les données inutiles, il est demandé de créer un logiciel permettant de synchroniser ces enregistrements avec les autres éléments du système, de tenir compte de la qualité des données (météo, …), de permettre une modification aisée des paramètres instrumentaux et d'afficher un aperçu des données acquises.

12. Rôle et utilisation

12.1. Besoins essentiels et principes associés

B1. Gérer le contrôle et la configuration initiale du rack
B2. Réaliser l'acquisition et l'enregistrement des données
B3. Synchroniser l'enregistrement des données avec SODISM 2
B4. Fournir un aperçu des données en cours d'acquisition
B5. S'adapter au contexte de l'observation
B6. Réaliser l'interface avec l'utilisateur

13. Description fonctionnelle

13.1. Fonctions de service

FS1. Régler les paramètres du rack d'acquisition
FS2. Acquérir les données
FS3. Enregistrer les données acquises
FS4. Interagir avec les autres éléments du système
FS5. Afficher les données en cours d'acquisition
FS6. Évaluer la qualité des signaux acquis
FS7. Interagir avec l'utilisateur

13.2. Caractérisation de chaque fonction

FS1 : Régler les paramètres du rack d'acquisition

- But : Configurer l'électronique du rack selon des paramètres prédéfinis au moyen d'un fichier de configuration, ou pendant l'exécution.
- Priorité : Haute.
- Contrainte : Utiliser un fichier de configuration.
- Répond aux besoins B1 et B6.

FS2 : Acquérir les données

- But : Lire les données depuis le rack. Elles comprennent les signaux science (tensions mesurées dans les quatre sous-pupilles), la tension de la voie globale de calibration et les données ancillaires (température du rack, état des filtres et amplis, etc).
- Priorité : Haute.
- Contrainte : Utiliser une classe C++ existante.
- Répond au besoin B2.

FS3 : Enregistrer les données acquises

- But : Enregistrer en temps réel les données lues depuis le rack. Cela inclut les données science, les données ancillaires ainsi que les alarmes et l'historique des actions effectuées (demandées par l'utilisateur ou exécutées automatiquement).
- Priorité : Haute.
- Contrainte : Aucune.
- Répond au besoin B2.

FS4 : Interagir avec les autres éléments du système

- But : Mettre en place une connexion entre le logiciel et les autres éléments du système (pilotage, voie image, centraliseur) en utilisant un protocole d'échange existant.
- Priorité : Haute.
- Contrainte : Utiliser le protocole d'échange existant.
- Répond aux besoins B3 et B4.

FS5 : Afficher les données en cours d'acquisition

- But : Fournir une visualisation "temps réel" des données ancillaires et science lues depuis le rack d'acquisition.
- Priorité : Moyenne.
- Contrainte : Pouvoir visualiser ces données à distance.
- Répond aux besoins B4 et B6.

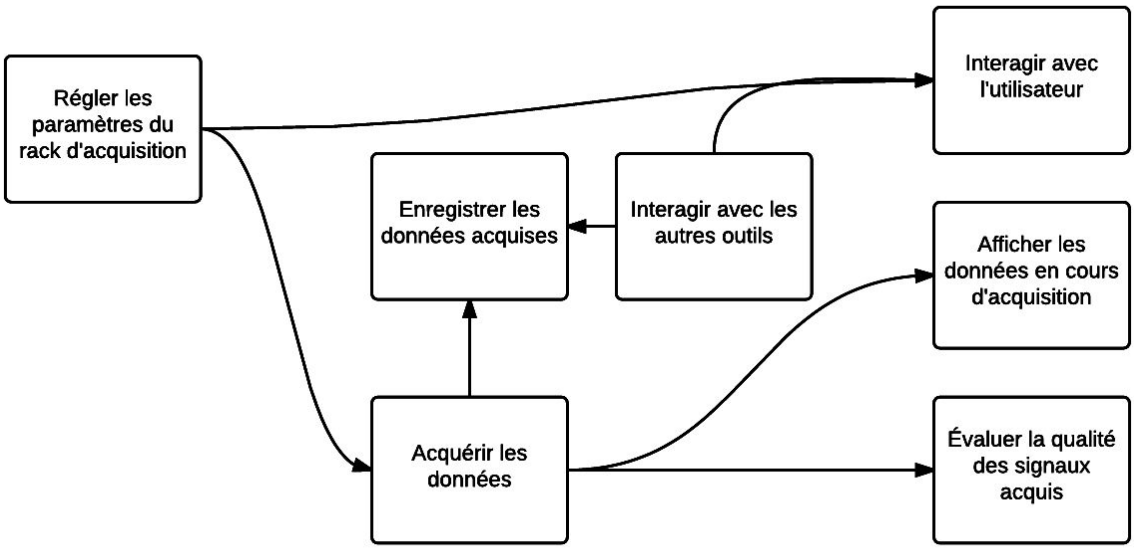
FS6 : Évaluer la qualité des signaux acquis

- But : Évaluer les signaux acquis afin de modifier automatiquement l'état du système pour l'adapter aux conditions d'observation et optimiser la qualité des données.
- Priorité : Moyenne.
- Contrainte : Aucune.
- Répond au besoin B5.

FS7 : Interagir avec l'utilisateur

- But : Permettre à un utilisateur de modifier l'état du rack, d'observer les données acquises, de consulter l'historique des événements et de corriger d'éventuelles anomalies.
- Priorité : Moyenne.
- Contrainte : Permettre certains contrôles à distance.
- Répond aux besoins B4 et B6.

13.3. Dépendances de réalisation des fonctions de service



14. Impositions de conception

- Langage C++
- Plateforme Windows XP
- Utilisation de C++ Builder

15. Planning et affectation des tâches

15.1. Ressources humaines

Alexis Metge
Statut : Étudiant stagiaire
Disponibilité : 350 heures sur 10 semaines

15.2. Charge totale du projet

Tâche	Durée prévisionnelle (heures)
Découverte du sujet	15
Rédaction du cahier des charges	20
Prise en main des outils et systèmes	20
Analyse et conception	50
Développement et documentation	95
Tests unitaires	40
Intégration et tests de validation	40
Préparation du rapport et de la soutenance	70
Total	350

15.3. Planning prévisionnel

	Avril		Mai				Juin			
	17	18	19	20	21	22	23	24	25	26
Découverte du sujet										
Rédaction du cahier des charges										
Prise en main des outils et systèmes										
Analyse et conception										
Développement et documentation										
Tests unitaires										
Intégration et tests de validation										
Préparation du rapport et de la soutenance										

Annexe 2

Documentation des signaux de MISOLFA

Voie pupille

16. Description

La voie pupille de MISOLFA est contrôlée par un module NI USB-6212 qui possède :

- 32 entrée/sorties numériques
- 16 entrées analogiques
- 2 sorties analogiques

17. Commandes générales du rack

Nomenclature	Commande	Voie
PWRBAT	Chargeur de batterie	$\overline{P1.0}$
PWRRack	Allumage rack	$\overline{P1.1}$
PWR+-12V	Alimentation +/- 12V (Amplis et filtres)	$\overline{P1.2}$
PWR+5V	Alimentation 5V (électronique de commande)	$\overline{P1.3}$
PWRPeltier	Allumage Peltier	P1.4
BATchk	Niveau de la batterie	P1.5
SETP	Consigne Peltiers	AO.0
ACTR	Température du rack	AI.15

18. Commandes des voies

Nomenclature	Commande		F1 (globale)	F2 (0.5mm)	F3 (1mm)	F4 (0.5mm)	F5 (2mm)
$PrGa_n$	Gain préampli		P2.0	P2.5	P0.2	P0.7	P0.12
$LPFon_n$	Filtre passe-bas		P2.1	P2.6	P0.3	P0.8	P0.13
$HPFon_n$	Filtre passe-haut		P2.2	P2.7	P0.4	P0.9	P0.14
$AmGalb_n$	Gain en sortie	Bit faible	P2.3	P0.0	P0.5	P0.10	P0.15
$AmGahb_n$		Bit fort	P2.4	P0.1	P0.6	P0.11	P1.7

Gain en sortie : chaque fois que la valeur du gain en sortie augmente de 1, le gain effectif double.

19. Gain nominaux

	Préampli	Gain en sortie
F1	0	00
F2	1	11
F3	0	10
F4	1	11
F5	0	00

20. Acquisition de données

	Photodiode		Peltier
	PHD_n	GND_n	ACT_n
F1	AI0	AI8	AI5
F2	AI1	AI9	AI6
F3	AI2	AI10	AI7
F4	AI3	AI11	AI13
F5	AI4	AI12	AI14

Annexe 3

Communication de la voie pupille

Pour toutes les communications, lors de la réception d'une commande, plusieurs messages peuvent être renvoyés :

- $BADPARAM! si le nombre de paramètre est invalide ou si le format d'un paramètre est invalide (réception d'une chaîne de caractère plutôt qu'un entier),
- $ENREGENCOURS! si des enregistrements sont en cours, car il n'est pas permis de modifier les paramètres du rack durant un enregistrement,
- $NOTCOMMANDE! si la commande n'est pas connue,
- $COMMANDEOK! si la commande à bien été prise en compte.

21. Communication avec le centraliseur

21.1. Réception

$FILTREPASSEBAS:numFibre;etat!

Permet d'activer ou désactiver le filtre passe bas d'une fibre. Le numéro de fibre doit être compris entre 0 et le nombre de fibres spécifié dans la configuration. État indique si le filtre doit être activé (true) ou désactivé (false).

$FILTREPASSEHAUT:numFibre;etat!

Permet d'activer ou désactiver le filtre passe haut d'une fibre. Le numéro de fibre doit être compris entre 0 et le nombre de fibres spécifié dans la configuration. État indique si le filtre doit être activé (true) ou désactivé (false).

$PREAMPLI:fibre;etat!

Permet de minimiser ou maximiser le gain du préampli d'une fibre. Le numéro de fibre doit être compris entre 0 et le nombre de fibres spécifié dans la configuration. État indique si le gain du préampli doit être maximisé (true) ou minimisé (false).

$GAINENSORTIE:fibre;gain!

Permet de définir le gain en sortie qui doit être appliqué à une fibre. Le numéro de fibre doit être compris entre 0 et le nombre de fibres spécifié dans la configuration. Le paramètre gain doit être compris entre 0 et 3, 0 correspondant au gain minimum et 3 au gain maximum (x8).

$ALIMRACK:etat!

Commande l'allumage et l'arrêt du rack. Si le paramètre etat vaut 'true', le rack est allumé, sinon il est éteint.

$ALIM12V:etat!

Commande l'allumage et l'arrêt de l'alimentation 12 volts. Si le paramètre etat vaut 'true', l'alimentation est allumée, sinon elle est éteinte.

$ALIM5V:etat!

Commande l'allumage et l'arrêt de l'alimentation 5 volts. Si le paramètre etat vaut 'true', l'alimentation est allumée, sinon elle est éteinte.

$ALIMPELTIER:etat!

Commande l'allumage et l'arrêt de l'alimentation des modules de Peltier. Si le paramètre etat vaut 'true', l'alimentation est allumée, sinon elle est éteinte.

$CONSIGNEPELTIER:consigne!

Permet de définir la consigne à appliquer aux modules Peltier. La consigne est reçue en degrés celsius.

$GETFILTREPASSEBAS:fibre!

Permet d'obtenir la configuration actuelle du filtre passe-bas de la fibre passée en paramètre. La réponse est $FILTREPASSEBAS:fibre;etat! avec etat à 'true' si le filtre est activé, 'false' sinon.

$GETFILTREPASSEHAUT:fibre!

Permet d'obtenir la configuration actuelle du filtre passe-haut de la fibre passée en paramètre. La réponse est $FILTREPASSEHAUT:fibre;etat! avec etat à 'true' si le filtre est activé, 'false' sinon.

$GETPREAMPLI:fibre!

Permet d'obtenir la configuration actuelle du préamplificateur de la fibre passée en paramètre. La réponse est $PREAMPLI:fibre;etat! avec etat à 'true' si l'amplificateur est activé, 'false' sinon.

$GETGAINENSORTIE:fibre!

Permet d'obtenir la configuration actuelle du gain en sortie de la fibre passée en paramètre. La réponse est $GAINENSORTIE:fibre;etat! avec etat un entier de 0 à 3.

$GETALIMRACK!

Permet d'obtenir l'état de l'alimentation du rack. La réponse est $ALIMRACK:etat! avec etat à 'true' si l'alimentation est allumée, 'false' sinon.

$GETALIM12V!

Permet d'obtenir l'état de l'alimentation 12V du rack. La réponse est $ALIM12V:etat! avec etat à 'true' si l'alimentation est allumée, 'false' sinon.

$GETALIM5V!

Permet d'obtenir l'état de l'alimentation 5V du rack. La réponse est $ALIM5V:etat! avec etat à 'true' si l'alimentation est allumée, 'false' sinon.

$GETALIMPELTIER!

Permet d'obtenir l'état de l'alimentation du rack. La réponse est $ALIMPELTIER:etat! avec etat à 'true' si l'alimentation est allumée, 'false' sinon.

$GETCONSIGNEPELTIER!

Permet d'obtenir la consigne actuelle des modules Peltier, en degrés celsius. La réponse est $CONSIGNEPELTIER:consigne! avec consigne un nombre décimal.

$GETETAT!

Permet de savoir ce qu'est en train de faire le logiciel. La réponse est sous la forme $ETAT:valeur!
Les valeurs possibles sont :

- STANDBY : Le logiciel attend des instructions.
- ENREGPLAN : Le logiciel effectue un enregistrement demandé par le plan.
- ENREG : Le logiciel enregistre des données en mode manuel.
- ATTENTEPLAN : Le logiciel attend que le fichier de plan soit déposé.
- PLAN : Le logiciel attend le début de la prochaine séquence programmée par le plan.

$SETSUIVIPLAN:mode!

Défini le mode d'enregistrement à utiliser. Si mode vaut 'true', le logiciel passe en mode suivi de plan. Sinon, le logiciel passe en mode manuel et attend.

$STARTENREG:temps!

Demande le démarrage d'une séquence d'enregistrement pour le temps donné. Si temps est omis ou égal à zéro, l'enregistrement continu jusqu'à une demande d'arrêt.

$STOPENREG!

Demande l'arrêt des enregistrements manuels en cours.

22. Communication avec la voie image

22.1. Réception

$NOTE:valeur!

Cette commande permet de faire parvenir à la voie pupille la note calculée par la voie image. Le paramètre 'valeur' correspond à la valeur de la note, qui est un entier lorsque le limbe solaire est visible, et qui devient -9999 lorsque le limbe est perdu.
La réception d'une note de -9999 entraîne l'arrêt des enregistrements en cours et empêche le démarrage d'une nouvelle séquence jusqu'à réception d'une nouvelle note.

22.2. Envoi

$OFFSET:offsetx!

Cette commande permet de demander le déplacement horizontal du points d'asservissement. L'offset envoyé est un offset relatif, la voie image calcule la modification à effectuer et répond $OFFSETOK! lorsque le déplacement est effectif.

Annexe 4

Fichier de configuration

```
[GENERAL]
CheminJournaux=journaux
CheminHK=HK
CheminObservations=observations
CheminPlan=
PrefixeFichierPlan=plan_misolfa_
DaterObservations=true
NoterTemperatures=false
FrequenceAffichage=2
DureeEnSecondesDesEnregistrements=60
Rack=6212
IPCentraliseur=10.152.1.169
PortCentraliseur=123456
IPVoieImage=10.152.1.169
PortVoieImage=12345
[SEUILS]
SautOffset=5
SeuilSaturation=-9.5
SeuilSignalFaible=0.5
[6212]
NomDevice=Dev1
PWRBAT=P1.0
PWRack=P1.1
PWR12V=P1.2
PWR5V=P1.3
PWRPeltier=P1.4
BATchk=P1.5
AOSETP=0
AIACTR=15
ConsignePeltiers=0.4
FrequenceAcquisition=1000
[F1]
Nom=Globale
Cassette=WTC1
PreampliMax=false
GainEnSortie=0
FiltrePasseHaut=false
FiltrePasseBas=false
[F2]
Nom=0.5mm
Cassette=WTC2
PreampliMax=false
GainEnSortie=0
FiltrePasseHaut=false
FiltrePasseBas=false
[F3]
Nom=1mm
Cassette=WTC3
PreampliMax=true
GainEnSortie=0
FiltrePasseHaut=false
FiltrePasseBas=false
[F4]
Nom=0.5mm
Cassette=WTC4
PreampliMax=true
GainEnSortie=3
FiltrePasseHaut=false
FiltrePasseBas=false
[F5]
Nom=2mm
Cassette=WTC5
PreampliMax=false
GainEnSortie=0
FiltrePasseHaut=false
FiltrePasseBas=false
[WTC1]
Preampli=P2.0
FiltrePasseBas=P2.1
FiltrePasseHaut=P2.2
BitFaibleGainEnSortie=P2.3
BitFortGainEnSortie=P2.4
AIPhotodiode=0
AIPeltier=5
[WTC2]
Preampli=P2.5
FiltrePasseBas=P2.6
FiltrePasseHaut=P2.7
BitFaibleGainEnSortie=P0.0
BitFortGainEnSortie=P0.1
AIPhotodiode=1
AIPeltier=6
[WTC3]
Preampli=P0.2
FiltrePasseBas=P0.3
FiltrePasseHaut=P0.4
BitFaibleGainEnSortie=P0.5
BitFortGainEnSortie=P0.6
AIPhotodiode=2
AIPeltier=7
```

[WTC4]
Preampli=P0.7
FiltrePasseBas=P0.8
FiltrePasseHaut=P0.9
BitFaibleGainEnSortie=P0.10
BitFortGainEnSortie=P0.11
AIPhotodiode=3
AIPeltier=13
[WTC5]
Preampli=P0.12
FiltrePasseBas=P0.13
FiltrePasseHaut=P0.14
BitFaibleGainEnSortie=P0.15
BitFortGainEnSortie=P0.17
AIPhotodiode=4
AIPeltier=14