UNIVERSITE PIERRE & MARIE CURIE
LA SCIENCE A PARIS

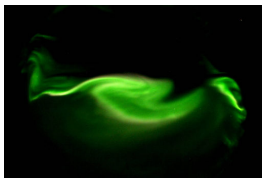
Licence de Physique Générale Appliquée
Année : 2006 – 2007

Rapport de stage en laboratoire

(29/01/2007 – 9/02/2007)

Projet ALFA

(Auroral Light Fine Analysis)



CETP

Clément MOUSSU - 2400099
Laboratoire d'accueil : CETP
Sous la tutelle d'Elena SERAN

Ce document résume le stage que j'ai effectué pendant deux semaines au Centre d'étude des Environnements Terrestres et planétaires du Lundi 29 janvier au vendredi 17 février 2007. Y seront présentés : des informations sur le laboratoire et l'équipe d'accueil, le sujet du stage et la description du projet dans lequel il s'inscrit, le travail réalisé et un bilan personnel de mon expérience.

1. Informations générales

1.1. Le laboratoire

Le **Centre d'étude des Environnements Terrestres et Planétaires** (CETP) est une unité mixte de recherche du département Sciences de l'Univers du CNRS et de l'Université de Versailles Saint-Quentin au sein de l'institut Pierre Simon Laplace. Il regroupe 130 permanents, chercheurs, ingénieurs et administratifs et accueille régulièrement une trentaine d'étudiants en thèse, visiteurs ou stagiaires. Il est implanté sur deux sites : le site du centre universitaire de Vélizy (une centaine de personnes) et le site de l'**Observatoire du Parc de Saint-Maur** où j'ai effectué mon stage (une trentaine de personnes).

Les recherches du CETP concernent principalement l'étude de l'atmosphère, les couplages entre surfaces et atmosphère et la physique des plasmas terrestres. Le CETP est organisé en **cinq départements scientifiques** : *Atmosphère Basse et Moyenne, Interactions Océan-Terre-Atmosphère, Electromagnétisme et Méthodes d'Analyse, Electrodynamique des Milieux Ionisés, Ondes dans les Plasmas Naturels* et **trois départements techniques** : *Instrumentation Terrestre et Spatiale, Informatique Distribuée et Applications, Moyens Généraux et Compatibilité*. Le département dans lequel j'ai effectué mon stage est le département **E**lectrodynamique des **M**ilieux **I**onisés.

1.2. L'équipe d'accueil

L'équipe d'accueil était principalement composée de trois personnes (les autres membres ayant été des intervenants extérieurs ou stagiaires) :

- **Elena SERAN** *(voir annexe 1)*, chercheur en physique (CETP / IPSL), est investigatrice principale du projet ALFA. C'est principalement avec elle que j'ai travaillé tout au long des deux semaines. Elle a régulièrement suivi l'avancée de mes activités et m'a guidé pour que le résultat de mon travail s'inscrive correctement dans le projet et puisse être utile à l'avancée des recherches.

- **Michel GODEFROY** *(voir annexe 2)*, ingénieur de recherche (CETP / IPSL), est responsable expérimental du projet. J'ai pu voir avec lui les questions techniques et concrètes traitant d'électronique ou de mécanique. Il m'a permis d'envisager les futures innovations du projet, et donc d'adapter mon travail pour qu'il puisse être amélioré et repris en main par la suite.

- **Jean-Claude CERISIER** *(voir annexe 3)*, enseignant-chercheur (Université Pierre et Marie Curie, CETP / IPSL), est co-investigateur du projet ALFA. Il m'a fourni des explications précieuses sur les phénomènes des aurores et des orages magnétiques.

2. Sujet du stage

2.1. Présentation du projet ALFA

Le projet ALFA (Auroral Light Fine Analysis) a comme objectif le développement d'instruments optiques pour effectuer des mesures fines des arcs auroraux. Il est articulé en deux modules dont la composition est détaillée dans les prochaines sous parties. Ces modules incluent chacun une caméra différente. Le module Plein Ciel (PLC) comprend un appareil photo qui permet d'observer des structures lumineuses avec une résolution spatiale de ~200 m (à l'altitude de 200 km) et une résolution temporelle de ~20s. Le module Haute Résolution Spatiale et Temporelle (HRST), quand à lui, comprend une caméra qui permet d'observer des structures lumineuses avec une résolution spatiale de ~50m (à l'altitude de 200 km) et une résolution temporelle de ~0.03 s. Pour faire simple, le module Plein Ciel prend des photographies *grand-angle* du ciel puis après analyse des résultats on pointe la caméra Haute Résolution du second module vers le phénomène précis que l'on veut étudier. Encore en développement, ce projet a néanmoins déjà donné lieu à plusieurs expérimentations. Des clichés d'orages magnétiques ont été pris au Niger *(voir annexe 7),* des clichés d'aurores boréales ont été pris à Kiruna en Suède *(voir annexe 6).* Les objectifs du projet sont pour l'instant l'étude des phénomènes suivants :

- Formation des irrégularités de la densité électronique dans l'ionosphère
- Information sur la source de mouvement du plasma dans le plan perpendiculaire au champ magnétique
- Analyse fine de la dynamique des émissions de lumière
- Information sur l'altitude des émissions (arcs en mouvement)
- Étude multi échelle des structures lumineuses
- Étude de l'interaction des électrons énergétiques avec l'ionosphère neutre et ionisée
- Estimation de l'énergie des électrons précipités
- Fermeture des courants alignés au champ magnétique
- Energie d'électrons associés

2.2. Dispositifs expérimentaux

Il est ici question de la composition et du rôle de chacun des deux modules.

2.2.1. Le module All-Sky ou Plein Ciel (PLC)

Le module Plein Ciel *(voir annexes 4, 5, 6, 7)* est une boîte de forme cylindrique (contrôlée par ordinateur) principalement composée d'un un appareil photo numérique *grand-angle* qui peut prendre des photographies sur un angle de 180° (le haut du module étant une demie sphère de plexiglas). La résolution des photos prises est de 3024x2016 pixels. Ce module possède son propre système de ventilation / chauffage permettant ainsi de protéger l'appareil photo des basses températures auxquelles il est soumis dans les zones où il est utilisé (pôle nord). La température interne est mesurée à l'aide d'un capteur et le système de

ventilation / chauffage régule la température automatiquement. Le module est équipé d'un GPS qui permet de donner la position et la date d'une mesure. Le tout est contrôlé en temps réel par ordinateur via plusieurs interfaces (USB, Firewire). Les logiciels de contrôle du module (traitement des images, contrôle du GPS, etc.) ont entièrement été réalisés au CETP par l'équipe. Pour certaines expériences il sera possible (en cours de conception) de faire des acquisitions sonores permettant ainsi de déterminer la distance à laquelle on se trouve du phénomène photographié. Des analyses fréquentielles du tonnerre (transformées de Fourier) pourront également être faites. Le poids du module est de 6kg, le rendant ainsi acceptable en bagage cabine dans un avion. Il a été conçu pour fonctionner sur une batterie au lithium et pour ne consommer que 42W. Cela lui permet d'avoir une autonomie de 4h. L'annexe n°5 présente un schéma détaillé de la composition du module Plein Ciel.

2.2.2. Le module Haute Résolution Spatiale et Temporelle (HRST)

Encore en phase de conception actuellement, le module Haute Résolution *(voir annexes 8, 9, 10)* sera composé d'une caméra dont l'angle de photographie sera étroit (18°) mais dont la résolution sera très élevée. La camera devra être mobile et dirigée à l'aide d'un système mécanique contrôlé par ordinateur. La caméra Haute Résolution est monochrome, c'est-à-dire qu'elle capte toutes les longueurs d'onde pour ne restituer que l'intensité lumineuse, d'où la nécessité d'utiliser des filtres interférentiels. Alors que l'appareil photo du module Plein Ciel comportait des filtres interférentiels RGB classiques (400 – 800 nm) à largeurs de bandes élevées, le module Haute Résolution comportera des filtres fréquentiels beaucoup plus fins à placer devant l'objectif ne laissant passer que des bandes d'une largeur de 2 nm. Le module comporte déjà un outil de mesure appelé **centrale inertielle** permettant de connaître la direction dans laquelle pointe la caméra Haute résolution. *C'est sur cet outil de mesure que s'est concentré l'essentiel de mon travail durant le stage.* Il devra également être équipé d'un système de régulation de la température afin d'éviter d'endommager la caméra. Il fonctionnera sur batteries au lithium et devra être léger et autonome afin d'être facilement transportable en avion.

2.3. Sujet du stage et objectifs

Le sujet de mon stage était de développer un programme pilotant la centrale inertielle et permettant ainsi de déterminer dans quelle direction (angle par rapport au nord et angle par rapport à la verticale) pointe la camera Haute Résolution au moment d'une mesure. Cela comprend 4 étapes principales :

- Comprendre le fonctionnement de la centrale
- Réussir à récupérer les informations via l'interface RS-232 et convertir ces informations en données dimensionnées
- Traiter les informations, trouver les algorithmes adaptés
- Présenter un résultat à l'utilisateur et lui permettre d'en garder une trace

Ces différentes étapes seront détaillées dans la partie traitant du travail réalisé pendant le stage.

3. Travail réalisé

3.1. La centrale inertielle

La centrale inertielle **Crossbow AHRS 400** *(voir annexes 11, 13)* est un outil de mesure comportant essentiellement 3 capteurs. Tout d'abord un capteur magnétique permettant de mesurer l'intensité du champ magnétique sur 3 axes liés à la centrale *(voir annexe 12)*. Elle comporte également un accéléromètre permettant de mesurer sur 3 axes les composantes de l'accélération que subit la centrale. Elle comporte des capteurs de vitesses angulaires autour des 3 axes qui ne nous serons pas nécessaires car le but du projet et de relever une position statique de la camera Haute Résolution. Enfin elle possède son propre capteur de température. La centrale calcule elle-même et fournit (au même titre que les accélérations ou intensités de champ magnétique) des angles stabilisés (angles d'Euler : roll, pitch et Yaw) permettant de connaître l'angle que fait cette dernière autour de chaque axe par rapport au Nord et à l'accélération de la pesanteur. Cette centrale transmet ses informations via une interface **RS-232** dont le port de connexion est plus connu sous le nom de port COM. Les informations sont récupérées sous forme de nombres binaires qu'il faut faire passer en base décimale. Il faut ensuite multiplier ces informations par un certain nombre de constante pour obtenir des données dimensionnées.

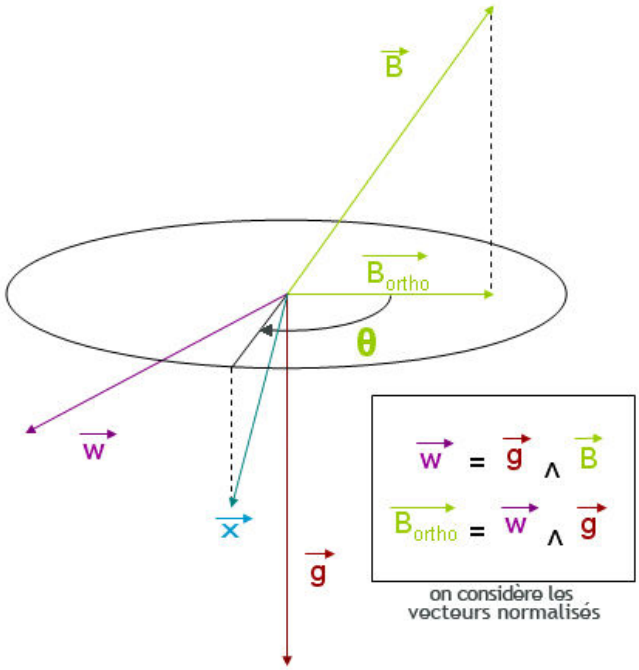
3.2. Physique du problème

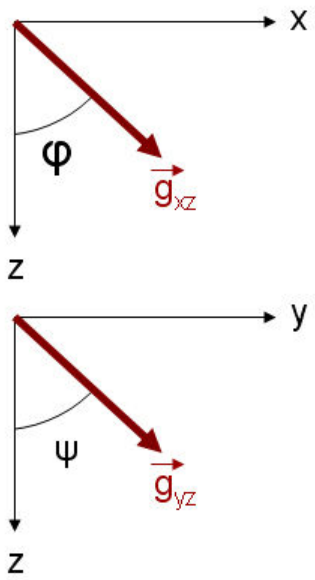
Le but de l'opération est de trouver deux angles : l'angle par rapport au nord et l'angle par rapport à la verticale.

Les informations concernant la latitude et la longitude sont données par le GPS. Les angles donnés par la centrale sont des angles d'Euler qui sont des angles de rotation autour de chaque axe (Roll : angle autour de x, Pitch : angle autour de y, et Yaw : angle autour de z). Un point de l'espace peut être défini par plusieurs triplets d'angles d'Euler. L'ordre des rotations angulaires que l'on doit appliquer à un point est important : Roll-Pitch-Yaw. Nous nous sommes d'abord servi de ces angles pour calculer nos angles finaux. Cela marchait très bien dans le plan horizontal (accélération de la pesanteur selon z), l'angle Yaw présentait l'angle au nord et l'angle pitch présentait l'inclinaison de la caméra. Mais sorti du plan horizontal cela demandait plus de calculs que de calculer nous même les angles finaux.

(NB : les vecteurs sont notés en gras)

Exprimés dans le repère défini par les axes de la centrale nous connaissons le vecteur « direction du nord magnétique » que nous noterons **B** et le vecteur « accélération de la pesanteur » que nous noterons **g**. Le vecteur **B** est tangent aux lignes de champ et pointe bien en direction du Nord magnétique mais il peut comporter également une composante verticale (surtout au pôle nord ou sa composante verticale est assez grande devant les autres). Par composante verticale nous entendons composante colinéaire au vecteur **g**. Pour calculer l'angle avec le Nord nous devons nous intéresser à la projection du vecteur **B** dans le plan orthogonal au vecteur **g**. Nous appellerons $\mathbf{B_{ortho}}$ cette projection. On peut définir un troisième vecteur **w** tel que $\mathbf{(g,B_{ortho},w)}$ forme un trièdre direct (après avoir normalisé ces 3 vecteurs). Connaissant les expressions de **g,** $\mathbf{B_{ortho}}$ et **w** dans le repère **(x,y,z)** il est donc facile d'exprimer **x** (vecteur choisi comme direction de pointage du dispositif) dans le repère $\mathbf{(g,B_{ortho},w)}$. Il suffit après d'utiliser les relations de passage entre coordonnées sphériques et cartésiennes pour connaître les angles finaux.





$$\theta = arctan\left(\frac{\vec{x}\cdot\vec{w}}{\vec{x}\cdot\vec{B}}\right) + \pi \times H\left(-\vec{x}\cdot\vec{B}\right) \times sgn\left(\vec{x}\cdot\vec{w}\right)$$

H est la fonction de Heaviside

$$\varphi = arccos\left(\frac{\vec{g_{xz}}}{\left\|\vec{g_{xz}}\right\|}\cdot\vec{z}\right)$$

$$\Psi = arccos\left(\frac{\vec{g_{yz}}}{\left\|\vec{g_{yz}}\right\|}\cdot\vec{z}\right)$$

3.3. Perturbations, calibration

La batterie au lithium utilisée dans le module Haute Résolution possède (comme celle utilisée dans le module Plein Ciel) son propre champ magnétique. Celui-ci vient s'ajouter au champ magnétique terrestre et fausse ainsi les mesures. Le but est donc de calibrer le système avant une série de mesure afin d'apporter les correction qui s'imposent pour que les mesures soient valables. La batterie est fixe dans le repère de la centrale inertielle. Le champ perturbateur est donc un champ constant dans le repère lié à cette dernière. Si on fait tourner le système autour de l'axe vertical la projection du champ magnétique terrestre sur un des deux axes horizontaux varie comme un cosinus en fonction de l'angle. Sans champ perturbateur la valeur moyenne du cosinus doit être nulle. Le champ perturbateur va donc induire un décalage vertical ou « offset » sur la valeur du champ magnétique de chacun des axes horizontaux. L'idée de départ est d'enregistrer en fonction du temps les valeurs du champ magnétique selon x et y pendant que l'utilisateur fait tourner assez lentement le système dans le plan horizontal. La courbe ainsi obtenue n'est plus un cosinus car fonction du temps donc de la vitesse de rotation qui n'est pas constante. Il suffit alors de chercher le maximum et le minium de ces mesures et d'en calculer la moyenne. On trouve une valeur corrective pour x et une autre pour y que l'on soustraira à chaque nouvelle mesure effectuée. C'était l'idée de départ puisque nous nous sommes aperçus que la centrale inertielle possédait son propre système de calibration fonctionnant sur le même principe mais gardant les résultat dans sa mémoire interne pour donner des résultats déjà corrigés. Nous avons gardé les deux systèmes en faisant la calibration de la centrale en premier puis la calibration logicielle…

3.4. Programmation

Pour développer une interface graphique sous Windows j'ai choisi d'utiliser (comme me l'a conseillé Elena SERAN) Lab Windows. Ce logiciel permet de créer rapidement et efficacement une interface graphique (orientée scientifique) en ne se préoccupant que des tâches que l'on veut faire faire au programme mais sans avoir à programmer toute la partie affichage. La programmation sous Lab Windows se fait en C. L'intégralité de mon programme se trouve en *annexe 15* ainsi qu'une copie d'écran *(annexe 14)* de l'interface graphique qui lui est liée.

Le programme a beaucoup changé au cours des deux semaines nous avons dû abandonner certaines fonctionnalités en cours de route, cela explique que certaines parties soient commentées dans le programme donné en *annexe 15*. Au final, j'ai choisi de créer plusieurs modes d'acquisition. Le mode instantané permet de prendre une mesure et d'en obtenir les deux angles souhaités. Le mode moyenné permet de prendre entre 2 et 40 mesures et de faire la moyenne classique de toutes les valeurs acquises. Enfin le mode continu prend autant de mesures que possible (fréquence d'échantillonnage environ 100 ms) ce qui permet de voir évoluer les valeurs et les graphes en temps réel. Une dernière fonction permet de sauvegarder une mesure à laquelle on aura donné un titre dans un fichier texte avec la date l'heure et toutes les informations nécessaires pour pouvoir resituer la direction de la caméra Haute Résolution. Un mode d'emploi du logiciel conçu pendant le stage est présenté en *annexe 16*.

Le programme actuellement et d'après les tests effectués fonctionne correctement. Cependant je dois repasser au CETP prochainement pour faire des tests plus approfondis.

4. Bilan personnel

Ces deux semaines de stage m'ont aidé à avoir un bon aperçu des multiples facettes du métier de chercheur ou de ce que permettent d'envisager des études de physique. J'ai rencontré lors de ce stage aussi bien des chercheurs, ingénieurs de recherche ou enseignants-chercheurs que des stagiaires / étudiants en thèse. Cela m'a aidé à réfléchir sur l'orientation que je vais suivre dans mes études et sur les choix que je vais devoir faire d'ici la fin de l'année. En effet, je souhaite intégrer un master professionnalisant dont la mention est « physique informatique ». Par rapport à cela, ce qui m'a été proposé de faire pendant le stage cadrait parfaitement avec mon projet d'orientation. Cela demandait aussi bien des compétences en informatique qu'en physique. J'ai apprécié que l'approche du problème se focalise sur l'utilisation de plusieurs compétences pour réaliser un projet avec un cahier des charges précis. J'ai justement choisi le parcours de Physique Générale Appliquée proposé par l'université Pierre et Marie Curie car je souhaite acquérir des compétences dans des domaines multiples plutôt que d'être spécialiste dans un seul domaine. J'ai également apprécié de travailler sur quelque chose de concret et plus pratique que théorique. Cela correspond à ce que j'imagine (pour l'instant) faire partie de mon métier plus tard. Au contraire, je peux dire que sachant déjà que je m'intéresse à des choses appliquées et concrètes j'aurai pu découvrir autre chose en faisant un stage plus théorique et dont la physique aurait été plus abstraite.

Le fait qu'on me fasse confiance en me donnant un projet précis à réaliser et de l'avoir mené à son terme m'a vraiment plu. J'ai trouvé également intéressant de voir toutes les étapes de conception d'un outil. Nous avons dans un premier temps du réfléchir à ce que nous attendions de cet outil et ce que nous avions à notre disposition pour le réaliser. La phase de conception a été un peu différente de ce que j'aurais imaginé. Nous avons du, à certains moments, faire marche arrière et revoir la méthode utilisée car nous nous trouvions face à des problèmes auxquels nous n'avions pas pensé. Le fait de devoir faire marche arrière demande un peu de courage car il n'est pas facile de se dire que l'on était sur une mauvaise piste et que le travail des derniers jours doit être revu. Je pense que la remise en question de son propre travail et le changement font partie du métier de chercheur et que c'est comme cela que l'on peut avancer lorsque l'on travaille sur un problème dont on ignore beaucoup de paramètres. Les paramètres ignorés étaient, dans mon cas, les conditions dans lesquelles serait utilisé le système Haute Résolution. Le projet ALFA a pour but d'être assez polyvalent, il a donc fallu penser aux différentes conditions dans lesquelles mon logiciel fonctionnerait. Enfin il y a eu une phase de communication qui consistait en la rédaction d'un mode d'emploi du logiciel et à l'explication de son fonctionnement. Là aussi j'ai découvert qu'il n'était pas si simple de créer des choses claires et utilisables pour quelqu'un d'autre que soi. Cela m'a permis de revenir sur le programme et de simplifier des choses pour que l'utilisateur ne dispose efficacement que des informations dont il a besoin.

Pour faire le bilan de ce stage je peux dire qu'il m'aura aidé à préciser mon projet d'orientation. Vous l'aurez compris, aimant plutôt les applications que la théorie, j'ai été très intéressé par le métier d'ingénieur de recherche. Le fait de travailler en collaboration avec une équipe de recherche et de concevoir des outils adaptés à un projet est quelque chose qui m'a intéressé. La manière positive d'envisager les problèmes de ma tutrice de stage m'a beaucoup aidé à avancer. Il n'est finalement pas utile de paniquer face à quelque chose que l'on ne connaît pas, il suffit de se pencher dessus. Je la remercie donc, elle, et toute l'équipe pour l'accueil qui m'a été fait et la confiance qui m'a été accordée.

Bibliographie

Au cours du stage j'ai du me documenter sur certains points et j'ai trouvé des réponses dans les ouvrages suivants :

- C en action (Yves Mettier - *O'Reilly*)
- Aide du logiciel Lab Windows
- Mode d'emploi centrale inertielle Crossbow AHRS 400
- Introduction to Space Physics (M. G. Kivelson & C. T. Russel - *Cambridge University Press)*
- L'environnement spatial de la Terre : la magnétosphère (Article par Fabrice Mottez, *chargé de recherches au CETP/CNRS*).

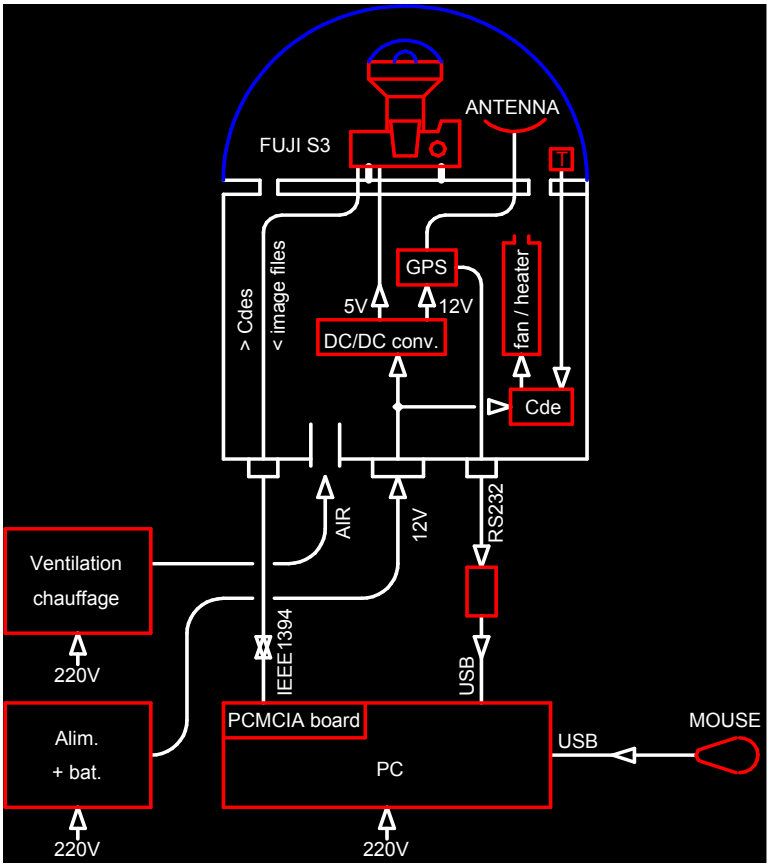
Annexes

Annexe 1 : Elena SERAN avec le module All-Sky du projet ALFA à Kiruna (Suède)

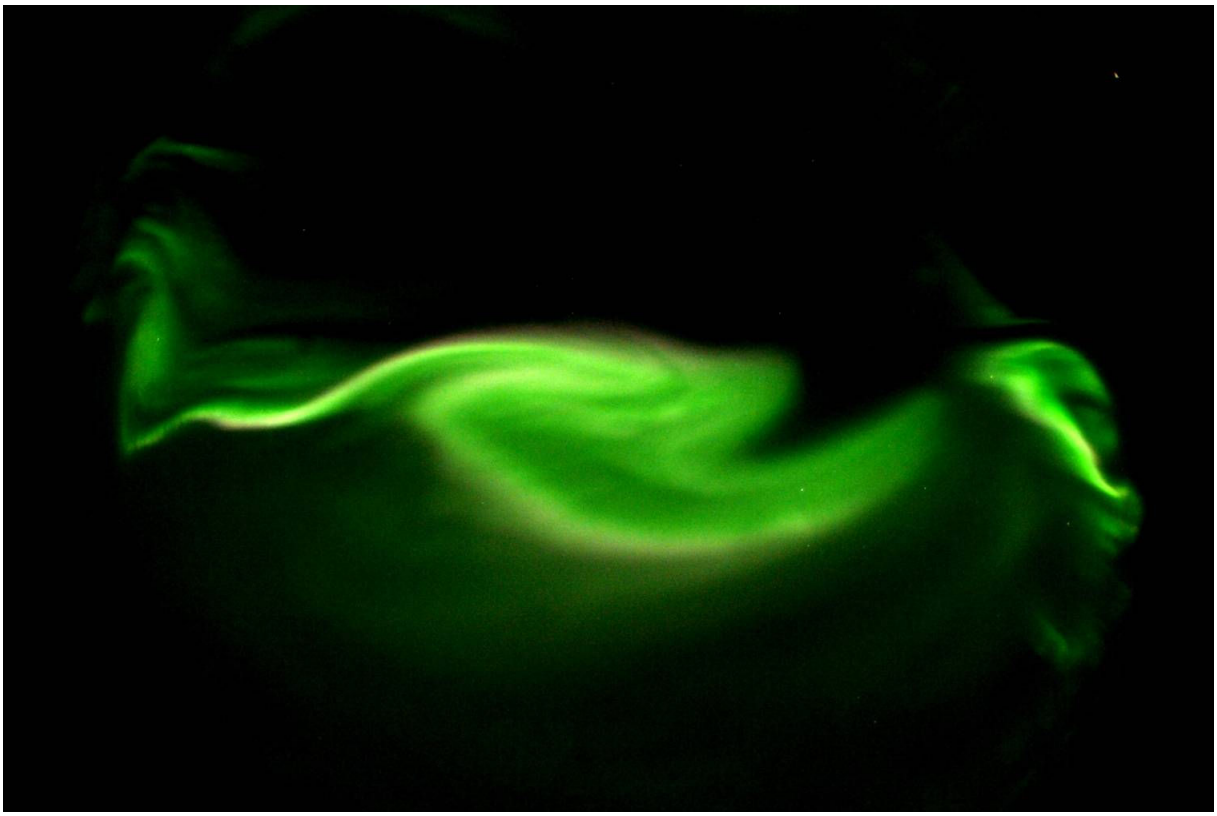
Annexe 2 : Michel GODEFROY avec le module All-Sky du projet ALFA à Kiruna (Suède)

Annexe 3 : Jean-Claude Cerisier

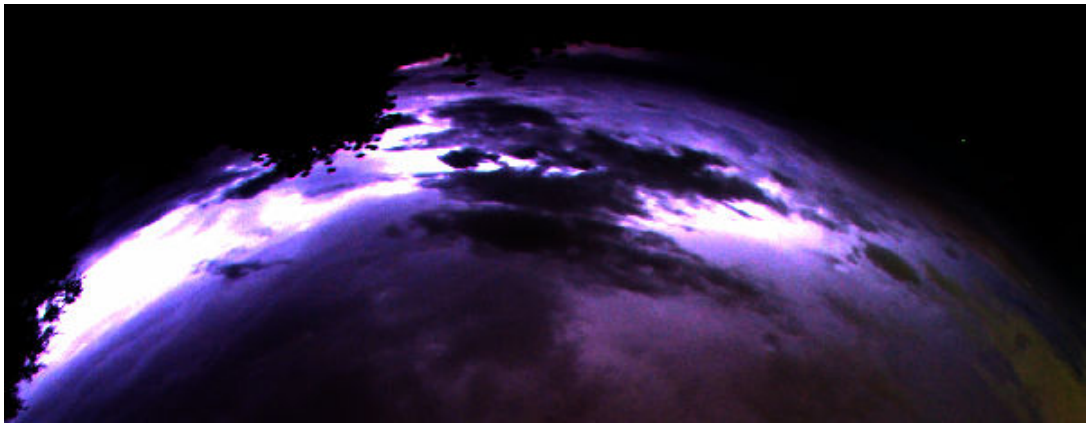
Annexe 4 : Le module Plein Ciel et son ordinateur de contrôle…



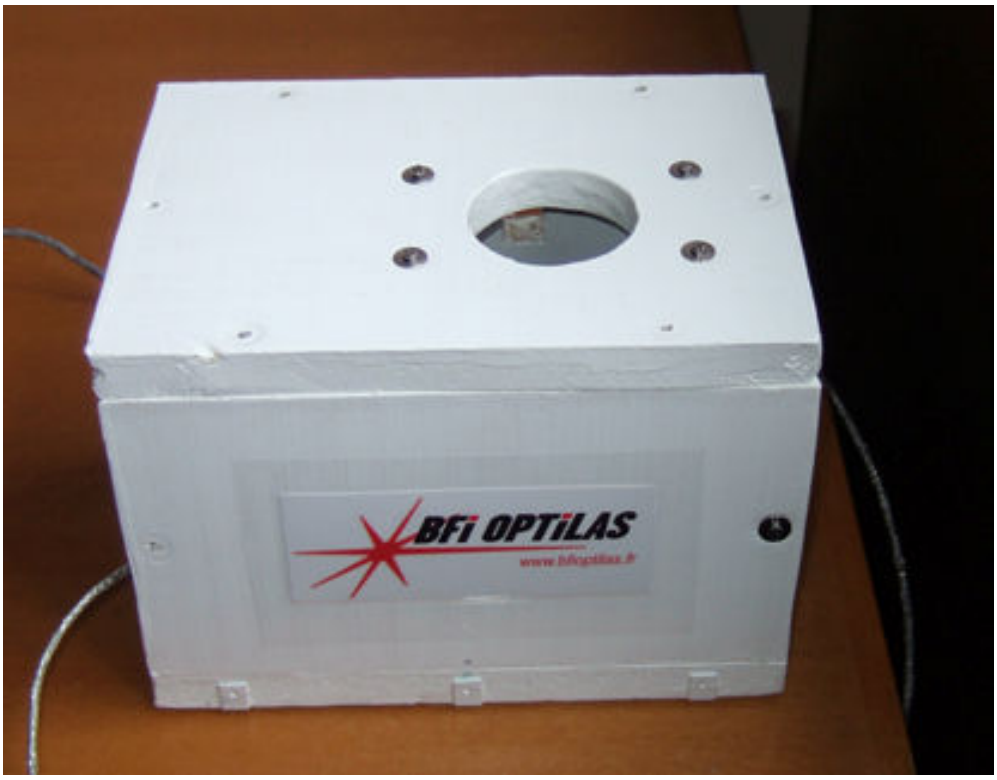
Annexe 5 : Schéma détaille de composition du module Plein Ciel



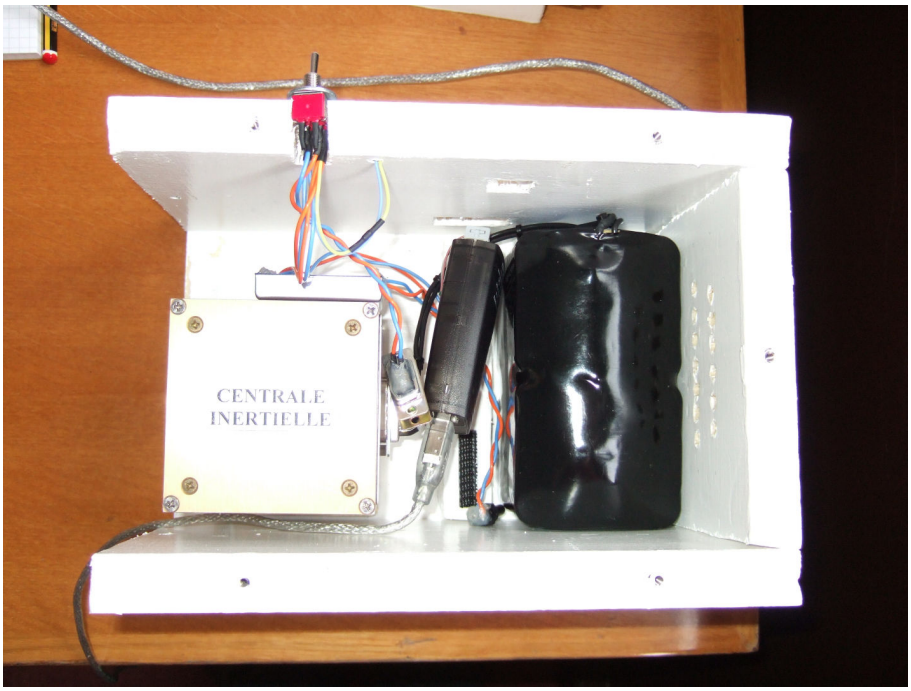
Annexe 6 : Exemple de prise de vue avec le module Plein Ciel…



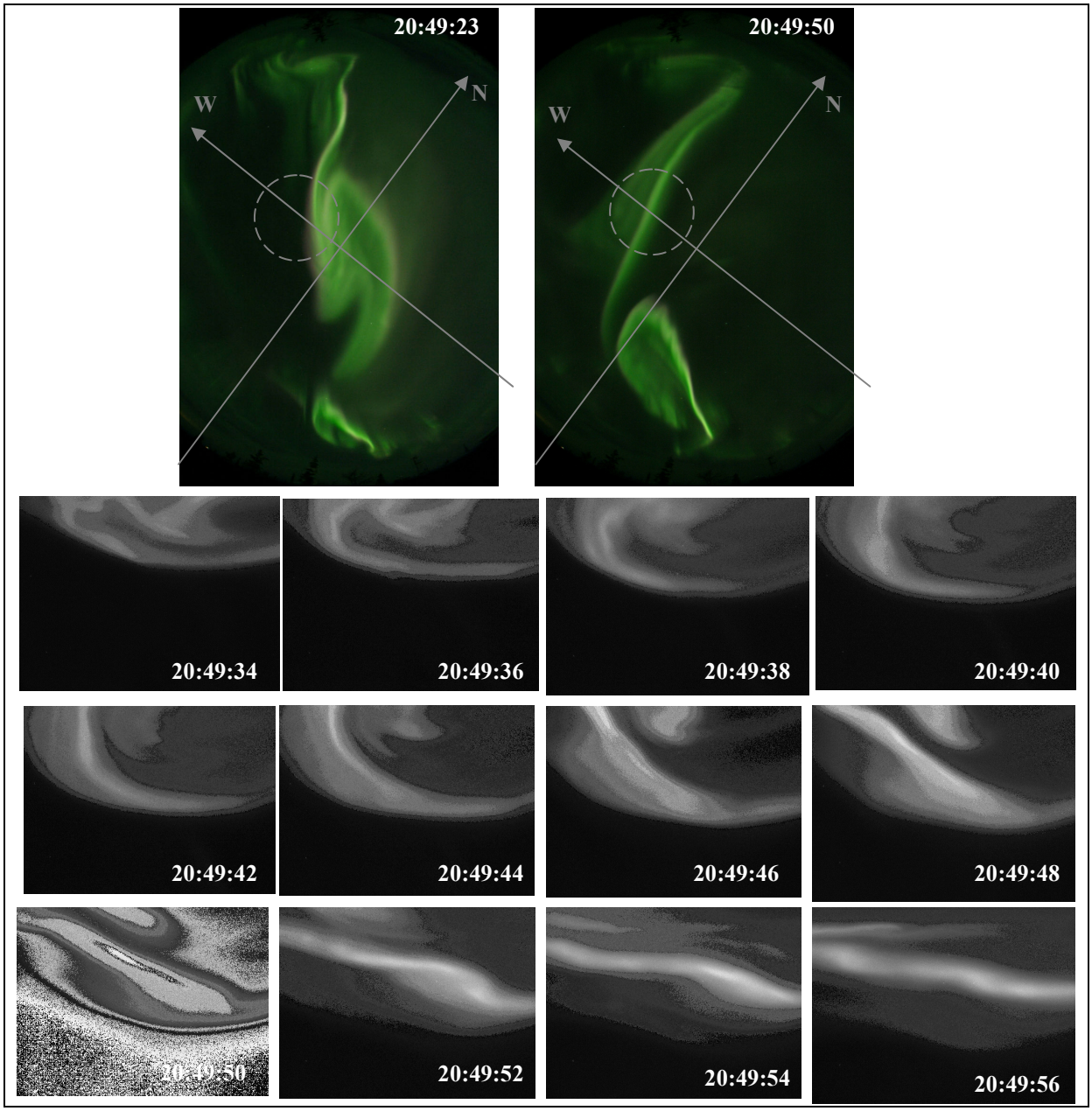
Annexe 7 : Photographie d'éclairs faite par le module Plein Ciel…



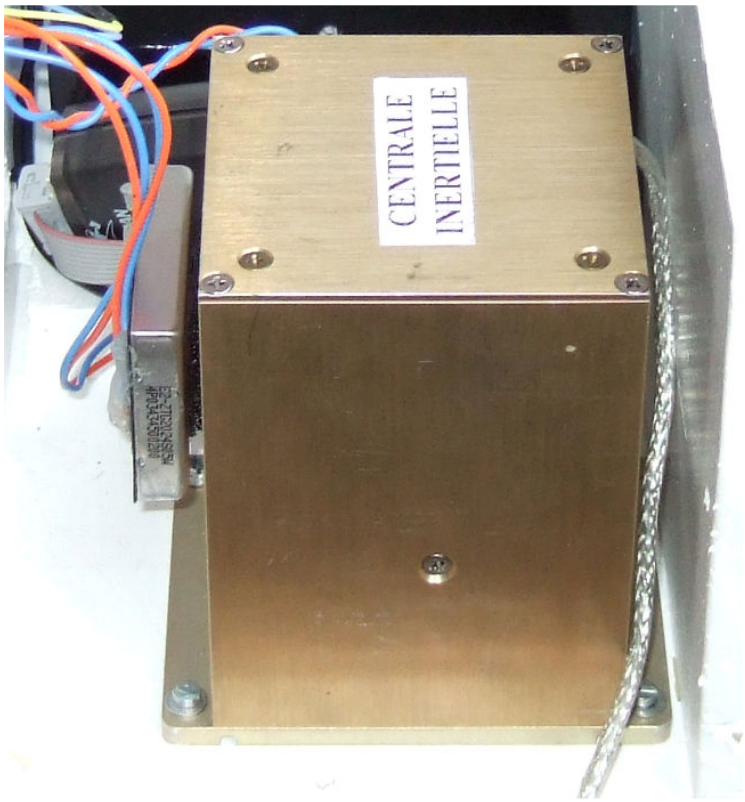
Annexe 8 : Une partie du futur module Haute Résolution actuellement en conception…



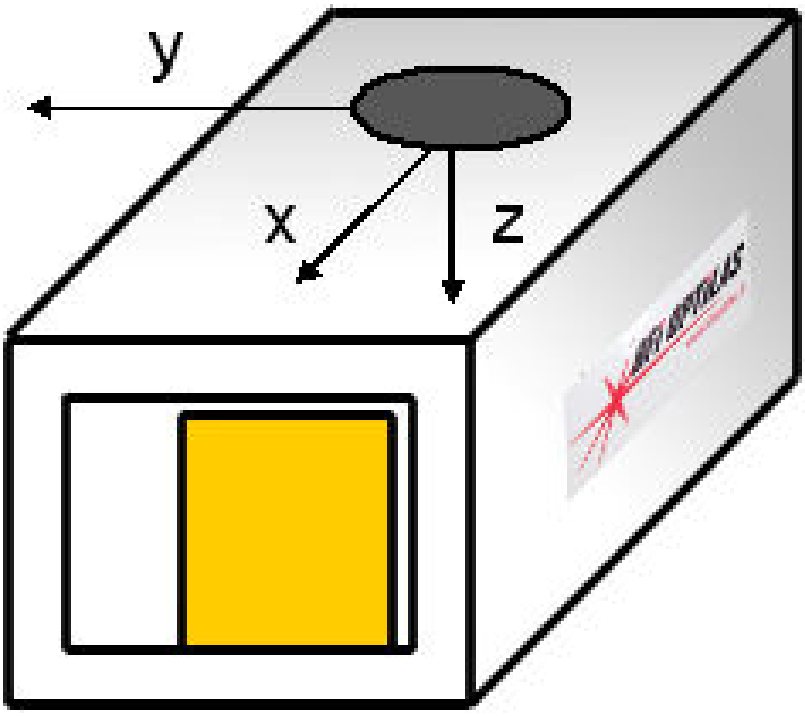
Annexe 9 : Vue d'intérieur du système Haute Résolution / Pointage…



Annexe 10 : Exemples de prises de vues avec la camera Haute Résolution lors d'essais (En haut les photos prises par le module Plein Ciel et en bas les photos Haute Résolution).



Annexe 11 : Vue approchée de la centrale inertielle Crossbow AHRS 400…



Annexe 12 : Définition des axes liés à la centrale inertielle Crossbow AHRS 400…

Specifications	AHRS400CD-200	Remarks
Performance		
Update Rate (Hz)	> 50	Continuous update mode
Start-up Time Valid Data (sec)	< 1	
Fully Stabilized Data (sec)	< 60	Under static conditions
Attitude		
Range: Roll, Pitch (°)	± 180, ± 90	
Static Accuracy (°)	<± 0.75	
Dynamic Accuracy (° rms)	± 2.5	
Resolution (°)	< 0.1	
Heading		
Range (°)	± 180	
Static Accuracy (°)	<± 2	
Dynamic Accuracy (° rms)	± 4	
Resolution (° rms)	< 0.1	
Angular Rate		
Range: Roll, Pitch, Yaw (°/sec)	± 200	
Bias: Roll, Pitch, Yaw (°/sec)	<± 1.0	Scaled sensor mode
Bias: Roll, Pitch, Yaw (°/sec)	<± 0.05	Angle mode
Scale Factor Accuracy (%)	< 1	
Non-Linearity (% FS)	< 0.3	
Resolution (°/sec)	< 0.05	
Bandwidth (Hz)	>25	-3 dB point
Random Walk (°/hr$^{1/2}$)	< 4.5	Typical
Acceleration		
Input Range: X/Y/Z (g)	± 4	
Bias: X/Y/Z (mg)	<± 12	
Scale Factor Accuracy (%)	< 1	
Non-Linearity (% FS)	< 1	
Resolution (mg rms)	< 0.6	
Bandwidth (Hz)	>10	-3 dB point
Random Walk (m/s/hr$^{1/2}$)	< 1.0	
Environment		
Operating Temperature (°C)	-40 to +71	
Non-Operating Temperature (°C)	-55 to +85	
Non-Operating Vibration (g rms)	6	20 Hz - 2 KHz random
Non-Operating Shock (g)	1000	1 ms half sine wave
Electrical		
Input Voltage (VDC)	9 to 30	
Input Current (mA)	< 300	
Power Consumption (W)	< 4	at 12 VDC
Digital Output Format	RS-232	
Analog[1] Range (VDC)	± 4.096	Pins 8, 9, 10, 12, 13, 14
	0 to 5.0	Pins 5, 6, 7
Physical		
Size (in)	3.0 x 3.75 x 4.1	Incl. mounting flanges
(cm)	7.62 x 9.53 x 10.42	Incl. mounting flanges
Weight (lbs)	< 1.7	
(kg)	< 0.77	
Connector	"D" male	

Spécifications techniques Crossbow AHRS 400…

Extraits du mode d'emploi de la centrale inertielle…

To convert the acceleration data into G's, use the following conversion:

$$\mathbf{accel = data*(GR} * 1.5)/2^{15}$$

where **accel** is the actual measured acceleration in G's, **data** is the digital data sent by the DMU, and **GR** is the G Range for your DMU. (The data is scaled so that 1 G = 9.80 m s^{-2}.) The G range of your DMU is the range of accelerations your DMU will measure. For example, if your DMU uses a ± 2 G accelerometer, then the G range is 2.

To convert the angular rate data into degrees per second, use the following conversion:

$$\mathbf{rate = data*(AR*}1.5)/2^{15}$$

where **rate** is the actual measured angular rate in °/sec, **data** is the digital data sent by the DMU, and **AR** is the Angular rate Range of the DMU. The angular rate range of your DMU is the range of angular rates your DMU will measure. For example, if your DMU uses ± 150 °/s rate sensors, then the **AR** range is 150.

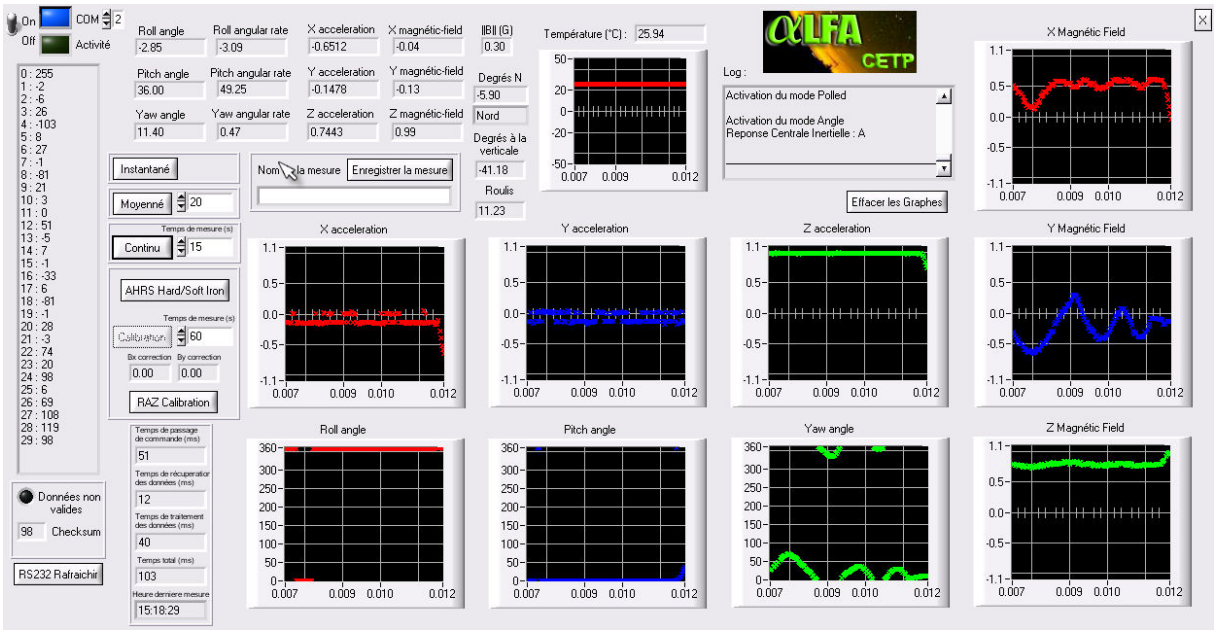
To convert the acceleration data into Gauss, use the following conversion:

$$\mathbf{mag = data*(MR*}1.5)/2^{15}$$

where **mag** is the actual measured magnetic field in Gauss, **data** is the digital data sent by the DMU, and **MR** is the Magnetic field Range of the DMU. **MR** is 1.25 for the AHRS.

The AHRS Kalman filter is not enabled in scaled sensor mode. Therefore, the rate sensor bias will change slightly due to large changes in temperature and time. If the unit is changed from angle to scaled mode, the last estimated rate sensor bias values are used upon entering scaled mode.

Annexe 13 : Informations à propos de la centrale inertielle…



Annexe 14 : Copie d'écran du programme développé pendant le stage…

Annexe 15 : Programme

pointage.c

```
#include <utility.h>
#include <formatio.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>
#include <ansi_c.h>
#include <cvirte.h>
#include <userint.h>
#include <interface.h>
#include <rs232.h>


#define TWO_EXP_EIGHT 256.
#define TWO_EXP_FIFTEEN 32768.
#define TWO_EXP_SIXTEEN 65536.
#define TWO_EXP_TWENTYFOUR 16777216.
#define TWO_EXP_THIRTYTWO 4294967296.
#define PI 3.14159265358979323846264338327950

#define TRUE 1
#define FALSE 0
#define CALIBRATE_MODE 1
#define GET_MODE 2
#define STABILISED_MODE 3

// ---------------------------------- DECLARATIONS ----------------------------------
-----------

static int  panelHandle;
FILE *calibrate_file;

// CHAR ---------------
char deviceName[]="COM2",timeStr[9],t_mesure[9],dateStr[11],int_ctrl[50],mes_file_name[50],
hms[50];
char str[256],version[300],read_data[5000],write_data[5000],cache[5000],log_message[5000];


// INT ----------------
long int m,i,j,k,l,n,x,y,z,cpt;

int RS232Error,RS232Open,  read_cnt, write_cnt, bytes_write, bytes_read, bit, fichier,heures
;
int old_timecode,stab_count,annule,log_file,mes_file,time_continuum,switch_state,calibrate_
file_w;

int old_plots=0,comport=2;

int serial[7],old_results[14],results[5000];


// DOUBLE -------------

double stabilised[18],moyenne[1000],x_mag[2000],y_mag[2000],calibrate[6000];
double x_mag_correction=0,y_mag_correction=0,Bnorme,angle,norme2d;
long double t_fin,t_fin_passage_commande,t_passage_commande,t_recuperation,t_traitement,tim
ecode,tps_lancement;




// ---------------------------------- FONCTIONS ----------------------------------------
--------

void logg(char *str){
   SetCtrlVal(panelHandle, PANEL_log_box, "\n");
   SetCtrlVal(panelHandle, PANEL_log_box, str);
   SetCtrlVal(panelHandle, PANEL_log_box, "\n");

    WriteFile(log_file,"\n",1);
    WriteFile(log_file,str,strlen(str));
    WriteFile(log_file,"\n",1);
}

void logg_mes(char *str){
   sprintf(mes_file_name,"mesures %s.txt",DateStr());
  mes_file=OpenFile(mes_file_name,VAL_WRITE_ONLY,VAL_APPEND,VAL_ASCII);
    WriteFile(mes_file,str,strlen(str));
    CloseFile(mes_file);
}


```

```
 77  void logg_s(char *str){
 78     SetCtrlVal(panelHandle, PANEL_log_box, str);
 79      WriteFile(log_file,str,strlen(str));
 80  }
 81
 82  void logg_line(){
 83    logg("_____");
 84  }
 85
 86  void t_field_set(int obj,char *str){
 87     SetCtrlVal(panelHandle, obj, str);
 88  }
 89
 90  void d_field_set(int obj,double nb){
 91     SetCtrlVal(panelHandle, obj, nb);
 92  }
 93
 94  void i_field_set(int obj,int nb){
 95     SetCtrlVal(panelHandle, obj, nb);
 96  }
 97
 98  void field_get(int obj){
 99     double temp=0;
100     GetCtrlVal(panelHandle, obj, &temp);
101     sprintf(str,"%g",temp);
102  }
103
104  double normalize(double *tab){
105     double sum=0;
106      int i;
107
108     for(i=0;i<3;i++){
109        sum+=(tab[i])*(tab[i]);
110      }
111
112     sum=sqrt(sum);
113
114     for(i=0;i<3;i++){
115        tab[i]/=sum;
116      }
117     return sum;
118  }
119
120
121  int verify_packet(int *results,int debut, int size){
122
123         int sum=0,i;
124        for(i=debut+1;i<size-1;i++){
125            sum+=results[i];
126          }
127       if(sum<0) sum+=256;
128       sum=sum%256;
129       i_field_set(PANEL__checksum,sum);
130
131        if(results[debut]==255 && sum==results[debut+29]){
132           SetCtrlVal(panelHandle,PANEL_unvalid_packet,0);
133            //logg("Correct data packet");
134            return TRUE;
135          }
136
137       SetCtrlVal(panelHandle,PANEL_unvalid_packet,1);
138      //sprintf(log_message,"Corrupted data packet : \nByte 0=%d\nByte 29=%d\nChecksum=%d
     ",results[debut],results[debut+29],sum);
139        //logg(log_message);
140        return FALSE;
141
142  }
143
144
145  //GET_PLOTS() : PREND COMME ARGUMENT UN TABLEAU DE MESURE ET TRACE LES POINT
     S SUR LES GRAPHES
146  void get_plots(double *return_results){
147
148        timecode=(Timer()-tps_lancement)/3600.;
149
150    /* //TRACE DES LIGNES ENTRE LES POINTS DE CAHQUE MESURE. NE MARCHE PAS CAR D
     ESACTIVE L'AUTOSCALE
151       //POUR ACTIVER ENLEVER TOUS LES COMMENTAIRES DE CETTE FONCTION...
152      if(old_plots){
153        PlotLine (panelHandle, PANEL_roll_plot, old_timecode, old_results[0],timecode, retur
     n_results[0], VAL_RED);
154        PlotLine (panelHandle, PANEL_pitch_plot, old_timecode, old_results[1], timecode, ret
```

```
urn_results[1], VAL_BLUE);
155         PlotLine (panelHandle, PANEL_yaw_plot, old_timecode, old_results[2], timecode, retu
rn_results[2], VAL_GREEN);
156         PlotLine (panelHandle, PANEL_ax_plot, old_timecode, old_results[6], timecode, retur
n_results[6],  VAL_RED);
157         PlotLine (panelHandle, PANEL_ay_plot, old_timecode, old_results[7], timecode, retur
n_results[7], VAL_BLUE);
158         PlotLine (panelHandle, PANEL_az_plot, old_timecode, old_results[8], timecode, retur
n_results[8], VAL_GREEN);
159         PlotLine (panelHandle, PANEL_Bx_plot, old_timecode, old_results[9], timecode, retur
n_results[9], VAL_RED);
160         PlotLine (panelHandle, PANEL_By_plot, old_timecode, old_results[10], timecode, ret
urn_results[10], VAL_BLUE);
161         PlotLine (panelHandle, PANEL_Bz_plot, old_timecode, old_results[11], timecode, ret
urn_results[11], VAL_GREEN);
162         PlotLine (panelHandle, PANEL_temp_plot, old_timecode, old_results[12], timecode, ret
urn_results[12], VAL_RED);
163       }
164
165      else{*/
166         PlotPoint (panelHandle, PANEL_roll_plot, timecode, return_results[0], VAL_BOLD_X, V
AL_RED);
167         PlotPoint (panelHandle, PANEL_pitch_plot, timecode, return_results[1], VAL_BOLD_X, V
AL_BLUE);
168         PlotPoint (panelHandle, PANEL_yaw_plot, timecode, return_results[2], VAL_BOLD_X, V
AL_GREEN);
169         PlotPoint (panelHandle, PANEL_ax_plot, timecode, return_results[6], VAL_BOLD_X, VAL
_RED);
170         PlotPoint (panelHandle, PANEL_ay_plot, timecode, return_results[7], VAL_BOLD_X, VAL
_BLUE);
171         PlotPoint (panelHandle, PANEL_az_plot, timecode, return_results[8], VAL_BOLD_X, VAL
_GREEN);
172         PlotPoint (panelHandle, PANEL_Bx_plot, timecode, return_results[9], VAL_BOLD_X, VAL
_RED);
173         PlotPoint (panelHandle, PANEL_By_plot, timecode, return_results[10], VAL_BOLD_X, VA
L_BLUE);
174         PlotPoint (panelHandle, PANEL_Bz_plot, timecode, return_results[11], VAL_BOLD_X, VA
L_GREEN);
175         PlotPoint (panelHandle, PANEL_temp_plot, timecode, return_results[12], VAL_BOLD_X,
VAL_RED);
176       //}
177
178      /*old_plots=1;
179     old_timecode=timecode;
180      for(i=0;i<14;i++){
181          old_results[i]=return_results[i];
182       }*/
183  }
184
185  int sg(double nb){
186      if(nb>0) return 1;
187      else if(nb<0) return -1;
188      else return 0;
189
190  }
191
192  int H(double nb){
193      if(nb>=0) return 1;
194      else return 0;
195
196  }
197
198  void vectoriel(double *u,double *v, double *w){
199      w[0]=u[1]*v[2]-u[2]*v[1];
200      w[1]=u[2]*v[0]-u[0]*v[2];
201      w[2]=u[0]*v[1]-u[1]*v[0];
202  }
203
204
205  //GET_FIELDS() : PREND EN ARGUMENT UN TABLEAU DE MESURES L'INDICE DE DEPART ET
     LA TAILLE, ON DONNE EGALEMENT LE MODE DANS LA VARIABLE OPTION
206  //LES 3 MODES SONT GET_MODE,CALIBRATE_MODE et STABILISED_MODE.
207  //CONVERTIT LES DONNEES EN VALEURS DIMENSIONNEES ET REMPLIT LES CHAMPS NUME
     RIQUES (CETTE FONCTION APPELLE GET_FIELDS() SI LE MODE EST GET_MODE
208  int get_fields(int *results,int debut, int size,int option){
209
210     double phy,theta,angles[3],rates[3],a[3],B[3],time,temp,return_results[14]={0,0,0,0,0,0,
     0,0,0,0,0,0,0,0},Bortho[3],w[3],aOrtho[2],aNorme2d,myroll;
211
212
213      if(verify_packet(results,debut,size)){
214
```

```
215         if(option==GET_MODE || option==STABILISED_MODE){
216
217              angles[0]=results[debut+1]*256+results[debut+2];
218              angles[1]=results[debut+3]*256+results[debut+4];
219              angles[2]=results[debut+5]*256+results[debut+6];
220              rates[0]=results[debut+7]*256+results[debut+8];
221              rates[1]=results[debut+9]*256+results[debut+10];
222              rates[2]=results[debut+11]*256+results[debut+12];
223              a[0]=results[debut+13]*256+results[debut+14];
224              a[1]=results[debut+15]*256+results[debut+16];
225              a[2]=results[debut+17]*256+results[debut+18];
226              B[0]=results[debut+19]*256+results[debut+20];
227              B[1]=results[debut+21]*256+results[debut+22];
228              B[2]=results[debut+23]*256+results[debut+24];
229
230             temp=results[debut+25]*256+results[debut+26];
231              temp*=5/4096.;
232              temp-=1.375;
233              temp*=44.4;
234              return_results[12]=temp;
235
236             time=results[debut+27]*256+results[debut+28];
237              return_results[13]=time;
238
239
240               for(i=0;i<3;i++){
241                angles[i]*=180./TWO_EXP_FIFTEEN;
242                rates[i]*=200.*1.5/TWO_EXP_FIFTEEN;
243                a[i]*=4.*1.5/TWO_EXP_FIFTEEN;
244                B[i]*=1.25*1.5/TWO_EXP_FIFTEEN;
245                  return_results[i+3]=rates[i];
246                 }
247
248              normalize(a);
249             //Bnorme=normalize(B);
250
251             B[0]-=x_mag_correction;
252             B[1]-=y_mag_correction;
253
254             Bnorme=normalize(B);
255
256              vectoriel(a,B,w);
257
258              normalize(w);
259
260              vectoriel(w,a,Bortho);
261
262              normalize(Bortho);
263
264             aNorme2d=sqrt(a[0]*a[0]+a[2]*a[2]);
265             aOrtho[0]=a[0]/aNorme2d;
266             aOrtho[1]=a[2]/aNorme2d;
267
268             phy=acos(aOrtho[1])*sg(a[0])*180./PI;
269
270             aNorme2d=sqrt(a[1]*a[1]+a[2]*a[2]);
271             aOrtho[0]=a[1]/aNorme2d;
272             aOrtho[1]=a[2]/aNorme2d;
273
274             myroll=acos(aOrtho[1])*sg(a[1])*180./PI;
275
276             if(w[0]!=0) theta=(atan(w[0]/Bortho[0])+PI*H(-Bortho[0])*sg(w[0]))*180/PI;
277              else theta=90*sg(B[1]);
278
279
280               for(i=0;i<3;i++){
281                  if(angles[i]<0) return_results[i]=angles[i]+360;
282                   else return_results[i]=angles[i];
283                   return_results[i+6]=a[i];
284                   return_results[i+9]=B[i];
285                 }
286
287
288             if(option==GET_MODE) {
289                d_field_set(PANEL_Bnorme,Bnorme);
290                 d_field_set(PANEL_roll,angles[0]);
291                d_field_set(PANEL_pitch,angles[1]);
292                d_field_set(PANEL_yaw,angles[2]);
293                 d_field_set(PANEL_roll_rate,rates[0]);
294                 d_field_set(PANEL_pitch_rate,rates[1]);
295                d_field_set(PANEL_yaw_rate,rates[2]);
296                 d_field_set(PANEL_ax,a[0]);
```

```
297                 d_field_set(PANEL_ay,a[1]);
298                 d_field_set(PANEL_az,a[2]);
299                d_field_set(PANEL_Bx,B[0]);
300                d_field_set(PANEL_By,B[1]);
301                d_field_set(PANEL_Bz,B[2]);
302               d_field_set(PANEL_temp,temp);
303
304                  get_plots(return_results);
305
306                SetCtrlVal(panelHandle,PANEL_myroll,myroll);
307               SetCtrlVal(panelHandle,PANEL_deg,theta);
308               if(theta<105 && theta>75) SetCtrlVal(panelHandle,PANEL_dir,"Est");
309               if(theta>-105 && theta<-75) SetCtrlVal(panelHandle,PANEL_dir,"Ouest");
310               if(theta>-15 && theta<15 ) SetCtrlVal(panelHandle,PANEL_dir,"Nord");
311               if(fabs(theta)>165 && B[0]<0 ) SetCtrlVal(panelHandle,PANEL_dir,"Sud");
312               if(theta>15 && theta<75) SetCtrlVal(panelHandle,PANEL_dir,"Nord Est");
313               if(theta>-75 && theta<-15) SetCtrlVal(panelHandle,PANEL_dir,"Nord Ouest");
314              if(theta>-165 && theta<-105) SetCtrlVal(panelHandle,PANEL_dir,"Sud Ouest");
315               if(theta>105 && theta<165) SetCtrlVal(panelHandle,PANEL_dir,"Sud Est");
316                SetCtrlVal(panelHandle,PANEL_vertical,-phy);
317                }
318           if(option==STABILISED_MODE){
319                moyenne[m*17]=angles[0];
320               moyenne[m*18+1]=angles[1];
321               moyenne[m*18+2]=angles[2];
322                for(i=m*18+3;i<m*18+14;i++){
323                     moyenne[i]=return_results[i-m*18];
324                     }
325               moyenne[m*18+14]=Bnorme;
326                moyenne[m*18+15]=theta;
327               moyenne[m*18+16]=phy;
328                moyenne[m*18+17]=myroll;
329                }
330
331           }
332
333          if(option==CALIBRATE_MODE) {
334               B[0]=results[debut+19]*256+results[debut+20];
335               B[1]=results[debut+21]*256+results[debut+22];
336               B[2]=results[debut+23]*256+results[debut+24];
337              for(i=0;i<3;i++) B[i]*=1.25*1.5/TWO_EXP_FIFTEEN;
338                //normalize(B);
339                calibrate[m]=B[0];
340               calibrate[m+1]=B[1];
341               calibrate[m+2]=B[2];
342                m+=3;
343           }
344
345            return TRUE;
346         }
347        logg("Traitment operation aborded");
348        logg_line();
349       return FALSE;
350
351
352     }
353
354     int min(double *tab,int size){
355        double min=tab[0];
356        int pos=0;
357
358        for(i=1;i<size;i++){
359            if(tab[i]<min){ min=tab[i]; pos=i;}
360         }
361        return pos;
362     }
363
364     int max(double *tab,int size){
365        double max=tab[0];
366        int pos=0;
367
368        for(i=1;i<size;i++){
369           if(tab[i]>max){ max=tab[i]; pos=i;}
370         }
371        return pos;
372     }
373
374     //GET_MAG_CORRECTIONS() CALCULE LES VARIABLES DE CALIBRATION.
375     void get_mag_corrections(void){
376
377         j=0;
378
```

```
379     for(i=0;i<m-3;i+=3){
380         x_mag[j]=calibrate[i];
381        y_mag[j]=calibrate[i+1];
382         j++;
383      }
384
385    x_mag_correction=(x_mag[min(x_mag,j)]+x_mag[max(x_mag,j)])/2.;
386
387    y_mag_correction=(y_mag[min(y_mag,j)]+y_mag[max(y_mag,j)])/2.;
388
389    d_field_set(PANEL_x_correct,x_mag_correction);
390    d_field_set(PANEL_y_correct,y_mag_correction);
391
392
393
394
395  }
396
397  //GET_PACKETS() RECUPERE LES PACKETS D'INFORMATION SUR LE PORT COM SELECTION
     NE. CETTE FONCTION APPELLE GET_FIELDS AVEC LES VALEURS TROUVEES.
398  //ON DONNE LE MODE DANS LA VARIABLE OPTION. LES 3 MODES SONT GET_MODE,CALIBRA
     TE_MODE et STABILISED_MODE.
399  void get_packets(int option){
400
401             //logg(strcat(TimeStr()," : "));
402
403           t_passage_commande=Timer();
404
405            ComWrt(comport,"G",1);
406             Delay(0.050);
407
408            t_fin_passage_commande=Timer();
409
410           //sprintf(log_message,"Demande d'un packet en %d ms",(int)(1000*(t_fin_passage_c
     ommande-t_passage_commande)));
411            //logg(log_message);
412
413
414           ResetTextBox(panelHandle,PANEL_rs232,"");
415
416            t_recuperation=Timer();
417           t_field_set(PANEL_current_time,TimeStr());
418           read_cnt=GetInQLen(comport);
419           ComRd(comport,read_data,read_cnt);
420            sprintf(t_mesure,"%s",TimeStr());
421
422            for(i=0;i<read_cnt;i++){
423
424                if(i==0 || (i<=29 && i>=25)){
425                   sprintf(str,"%d : %u\n",i,(unsigned char)read_data[i]);
426                   sprintf(cache,"%u",(unsigned char)read_data[i]);
427                   }
428                  else{
429                    sprintf(str,"%d : %d \n",i,read_data[i]);
430                    sprintf(cache,"%d",read_data[i]);
431                   }
432
433                results[i]=atoi(cache);
434              if(option==GET_MODE) SetCtrlVal(panelHandle, PANEL_rs232, str);
435               }
436
437
438            t_traitement=Timer();
439             if(get_fields(results,0,i,option)){
440
441               //sprintf(log_message,"Récuperation des données effectuée en %d ms",(int)(1
     000*(t_traitement-t_recuperation)));
442                //logg(log_message);
443
444                 t_fin=Timer();
445
446               //sprintf(log_message,"Traitement des données effectué en %d ms",(int)(1000
     *(t_fin-t_traitement)));
447                //logg(log_message);
448
449               //sprintf(log_message,"Opération totale effectuée en %d ms",(int)(1000*(t_f
     in-t_passage_commande)));
450                //logg(log_message);
451                  //logg_line();
452               if(option==GET_MODE){
453                  d_field_set(PANEL_t_total,1000*(t_fin-t_passage_commande));
454                d_field_set(PANEL_t_passage,1000*(t_fin_passage_commande-t_passage_comm
```

```
    ande));
455                 d_field_set(PANEL_t_recup,1000*(t_traitement-t_recuperation));
456                  d_field_set(PANEL_t_traitement,1000*(t_fin-t_traitement));
457
458                   }
459
460               }
461  }
462
463  int main(void) {
464
465     tps_lancement=Timer();
466    log_file=OpenFile("log.txt",VAL_WRITE_ONLY,VAL_TRUNCATE,VAL_ASCII);
467
468     InitCVIRTE(0,0,0);
469
470     panelHandle = LoadPanel (0, "interface.uir", PANEL);
471     DisplayPanel (panelHandle);
472
473     /*if(!(NULL == (calibrate_file=fopen("calibrate.dat","r")))){
474
475         fscanf(calibrate_file,"%lf",&x_mag_correction);
476         fscanf(calibrate_file,"%lf",&y_mag_correction);
477        d_field_set(PANEL_x_correct,x_mag_correction);
478        d_field_set(PANEL_y_correct,y_mag_correction);
479          fclose(calibrate_file);
480      }*/
481
482
483     RunUserInterface ();
484
485     DiscardPanel (panelHandle);
486
487    /*calibrate_file_w=OpenFile("calibrate.dat",VAL_WRITE_ONLY,VAL_TRUNCATE,VAL_ASCII);
488     sprintf(cache,"%f\n",x_mag_correction);
489     WriteFile(calibrate_file_w,cache,strlen(cache));
490     sprintf(cache,"%f\n",y_mag_correction);
491     WriteFile(calibrate_file_w,cache,strlen(cache));
492     CloseFile(calibrate_file_w);*/
493
494     CloseFile(log_file);
495
496
497  return 0;
498
499  }
500
501
502
503  //ACTIVATION DU PORT COM SELECTIONNE DANS LA FENETRE. CE CVICALLBACK EST RATA
     CHE AU BOUTON ON/OFF ET PERMET D'ACTIVER LA CONNEXION A LA CENTRALE.
504  int CVICALLBACK RS232_ON(int panel, int control, int event,
505          void *callbackData, int eventData1, int eventData2) {
506
507     switch (event) {
508       case EVENT_COMMIT:
509
510           GetCtrlVal(panelHandle,PANEL_RS232_SWITCH,&switch_state);
511
512          if(switch_state){
513           SetCtrlVal(panelHandle,PANEL_RS232_SWITCH,1);
514            SetCtrlVal(panelHandle,PANEL_active_process,1);
515
516           GetCtrlVal(panelHandle,PANEL_COMPORT,&comport);
517            sprintf(deviceName,"COM%d",comport);
518
519           RS232Open=OpenComConfig(comport,deviceName,38400,0,8,1,4096,4096);
520            RS232Error=ReturnRS232Err();
521
522              logg_s(strcat(DateStr()," - "));
523             logg_s(strcat(TimeStr()," : \n"));
524
525            ComWrt(comport,"P",1);
526             Delay(0.100);
527
528            ComWrt(comport,"R",1);
529              Delay(0.100);
530            read_cnt=GetInQLen(comport);
531            ComRd(comport,read_data,read_cnt);
532             logg("Tentative de ping");
533
```

```
534         if(strlen(read_data)){
535            logg_s("Reponse Centrale Inertielle : ");
536            logg_s(strcat(read_data,"\n"));
537
538            if(RS232Error<0){
539              sprintf(log_message,"Erreur RS232 : %d",RS232Error);
540               logg(log_message);
541               if(RS232Error==-7){
542                 sprintf(log_message,"Impossible d'ouvrir le port %s",deviceName);
543                   logg(log_message);
544                  }
545                logg_line();
546              SetCtrlVal(panelHandle,PANEL_active_process,0);
547             SetCtrlVal(panelHandle,PANEL_RS232_SWITCH,0);
548                 break;
549              }
550             else{
551               logg_s("\nRS232 ");
552              logg_s(deviceName);
553              logg_s(" succesfuly opened\n");
554             SetCtrlVal(panelHandle,PANEL_COM_active,1);
555              }
556
557            logg("Clear Hard Iron Calibration");
558           ComWrt(comport,"h",1);
559            Delay(0.100);
560          read_cnt=GetInQLen(comport);
561          ComRd(comport,read_data,read_cnt);
562           logg_s("Reponse Centrale Inertielle : ");
563           logg_s(strcat(read_data,"\n"));
564
565           logg("Clear Soft Iron Calibration");
566           ComWrt(comport,"t",1);
567            Delay(0.100);
568          read_cnt=GetInQLen(comport);
569          ComRd(comport,read_data,read_cnt);
570           logg_s("Reponse Centrale Inertielle : ");
571           logg_s(read_data);
572
573           ComWrt(comport,"S",1);
574            Delay(0.100);
575          read_cnt=GetInQLen(comport);
576          ComRd(comport,read_data,read_cnt);
577         sprintf(cache,"%u,%u,%u,%u,%u,%u",(unsigned char)read_data[0],(unsigned char)rea
    d_data[1],(unsigned char)read_data[2],(unsigned char)read_data[3],(unsigned char)read_data[
    4],(unsigned char)read_data[5]);
578            serial[0]=atoi(strtok(cache,","));
579
580            for(i=1;i<6;i++) serial[i]=atoi(strtok(NULL,","));
581
582        serial[6]=TWO_EXP_TWENTYFOUR*serial[1]+TWO_EXP_SIXTEEN*serial[2]+TWO_EXP_EI
    GHT*serial[3]+serial[4];
583           if(serial[0]==255 && serial[5]==(serial[1]+serial[2]+serial[3]+serial[4])%256){
584              sprintf(log_message,"Serial number : %d",serial[6]);
585               logg(log_message);
586              }
587           else logg("Erreur dans la recuperation du numéro de série");
588
589
590           ComWrt(comport,"v",1);
591            Delay(0.100);
592          read_cnt=GetInQLen(comport);
593          ComRd(comport,version,read_cnt);
594           sprintf(log_message,"Version : ");
595
596          for(i=1;i<read_cnt-1;i++) sprintf(log_message,"%s%c",log_message,version[i]);
597
598           logg(log_message);
599
600           ComWrt(comport,"P",1);
601            Delay(0.100);
602          read_cnt=GetInQLen(comport);
603          ComRd(comport,read_data,read_cnt);
604           logg("Activation du mode Polled");
605
606           ComWrt(comport,"a",1);
607            Delay(0.100);
608          read_cnt=GetInQLen(comport);
609          ComRd(comport,read_data,read_cnt);
610           logg("Activation du mode Angle");
611           logg_s("Reponse Centrale Inertielle : ");
612           logg_s(strcat(read_data,"\n"));
```

```
613
614          SetCtrlAttribute(panelHandle,PANEL_get,ATTR_DIMMED,0);
615          SetCtrlAttribute(panelHandle,PANEL_continu,ATTR_DIMMED,0);
616          SetCtrlAttribute(panelHandle,PANEL_save,ATTR_DIMMED,0);
617          SetCtrlAttribute(panelHandle,PANEL_IRON,ATTR_DIMMED,0);
618          SetCtrlAttribute(panelHandle,PANEL_reset,ATTR_DIMMED,0);
619          SetCtrlAttribute(panelHandle,PANEL_refresh,ATTR_DIMMED,0);
620          SetCtrlAttribute(panelHandle,PANEL_stabilise,ATTR_DIMMED,0);
621          }
622
623         else{
624               logg("Erreur : La centrale inertielle ne répond pas");
625             SetCtrlVal(panelHandle,PANEL_RS232_SWITCH,0);
626              SetCtrlVal(panelHandle,PANEL_COM_active,0);
627          }
628
629          }
630
631         else{
632           SetCtrlVal(panelHandle,PANEL_unvalid_packet,0);
633          SetCtrlAttribute(panelHandle,PANEL_get,ATTR_DIMMED,1);
634          SetCtrlAttribute(panelHandle,PANEL_continu,ATTR_DIMMED,1);
635          SetCtrlAttribute(panelHandle,PANEL_save,ATTR_DIMMED,1);
636          SetCtrlAttribute(panelHandle,PANEL_IRON,ATTR_DIMMED,1);
637          SetCtrlAttribute(panelHandle,PANEL_calibrate,ATTR_DIMMED,1);
638          SetCtrlAttribute(panelHandle,PANEL_reset,ATTR_DIMMED,1);
639          SetCtrlAttribute(panelHandle,PANEL_refresh,ATTR_DIMMED,1);
640          SetCtrlAttribute(panelHandle,PANEL_stabilise,ATTR_DIMMED,1);
641
642          SetCtrlVal(panelHandle,PANEL_RS232_SWITCH,0);
643          SetCtrlVal(panelHandle,PANEL_COM_active,0);
644          RS232Error=CloseCom(comport);
645          sprintf(log_message,"Connexion avec le port %s désactivée",deviceName);
646           logg(log_message);
647
648
649
650          }
651
652        logg_line();
653       SetCtrlVal(panelHandle,PANEL_active_process,0);
654         break;
655
656      }
657
658     return 0;
659  }
660
661  int CVICALLBACK IRON(int panel, int control, int event,
662        void *callbackData, int eventData1, int eventData2) {
663
664     switch (event) {
665       case EVENT_COMMIT:
666
667           SetCtrlVal(panelHandle,PANEL_active_process,1);
668
669            x_mag_correction=0;
670            y_mag_correction=0;
671           d_field_set(PANEL_x_correct,0);
672           d_field_set(PANEL_y_correct,0);
673
674            logg("Start Hard Iron Calibration");
675           ComWrt(comport,"s",1);
676            Delay(0.100);
677          read_cnt=GetInQLen(comport);
678          ComRd(comport,read_data,read_cnt);
679           logg_s("Reponse Centrale Inertielle : ");
680           logg_s(strcat(read_data,"\n"));
681
682          GetCtrlVal(panelHandle,PANEL_t_calib,&time_continuum);
683           Delay(time_continuum);
684
685           logg("Stop Hard Iron Calibration");
686           ComWrt(comport,"u",1);
687            Delay(0.100);
688          read_cnt=GetInQLen(comport);
689          ComRd(comport,read_data,read_cnt);
690           logg_s("Reponse Centrale Inertielle : ");
691           logg_s(strcat(read_data,"\n"));
692
693          SetCtrlAttribute(panelHandle,PANEL_calibrate,ATTR_DIMMED,0);
694           SetCtrlVal(panelHandle,PANEL_active_process,0);
```

```
695
696             break;
697      }
698
699     return 0;
700  }
701
702
703
704  //SAUVE UNE MESURE EN PRENANT LES VALEURS CONTENUES DANS LES CHAMPS NUMERI
     QUES ET LE TITRE CHOISI, SAUVE DANS UN FICHIER mesures MM/DD/YYYY.txt
705  int CVICALLBACK save_mes(int panel, int control, int event,
706        void *callbackData, int eventData1, int eventData2) {
707
708     switch (event) {
709       case EVENT_COMMIT:
710
711           GetCtrlVal(panelHandle,PANEL_nom_mes,str);
712              if(!strlen(str)){
713              SetCtrlVal(panelHandle,PANEL_nom_mes,"Veuillez donner un nom !");
714                 Delay(2);
715               SetCtrlVal(panelHandle,PANEL_nom_mes,"");
716                  break;
717               }
718           SetCtrlVal(panelHandle,PANEL_nom_mes,"");
719
720            logg_mes("Mesure : ");
721            logg_mes(str);
722            logg_mes("\n");
723           logg_mes(t_mesure);
724
725            logg_mes("\nAngle par rapport au Nord : ");
726           field_get(PANEL_deg);
727            logg_mes(str);
728           logg_mes(" degrés ");
729
730            logg_mes("\nDirection (approximative) : ");
731           GetCtrlVal(panelHandle,PANEL_dir,str);
732            logg_mes(str);
733
734            logg_mes("\nAngle à la verticale : ");
735           field_get(PANEL_vertical);
736            logg_mes(str);
737           logg_mes(" degrés ");
738
739            logg_mes("\nAngle de roulis : ");
740           field_get(PANEL_myroll);
741            logg_mes(str);
742           logg_mes(" degrés ");
743
744            logg_mes("\nRoll : ");
745           field_get(PANEL_roll);
746            logg_mes(str);
747
748            logg_mes("\nPitch : ");
749           field_get(PANEL_pitch);
750            logg_mes(str);
751
752            logg_mes("\nYaw : ");
753           field_get(PANEL_yaw);
754            logg_mes(str);
755
756            logg_mes("\nRoll rate : ");
757           field_get(PANEL_roll_rate);
758            logg_mes(str);
759
760            logg_mes("\nPitch rate: ");
761           field_get(PANEL_pitch_rate);
762            logg_mes(str);
763
764            logg_mes("\nYaw rate: ");
765           field_get(PANEL_yaw_rate);
766            logg_mes(str);
767
768            logg_mes("\nax : ");
769           field_get(PANEL_ax);
770            logg_mes(str);
771
772            logg_mes("\nay : ");
773           field_get(PANEL_ay);
774            logg_mes(str);
775
```

```
776             logg_mes("\naz : ");
777            field_get(PANEL_az);
778             logg_mes(str);
779
780
781             logg_mes("\nBx (corrected) : ");
782            field_get(PANEL_Bx);
783             logg_mes(str);
784
785             logg_mes("\nBy (corrected) : ");
786            field_get(PANEL_By);
787             logg_mes(str);
788
789             logg_mes("\nBz (corrected) : ");
790            field_get(PANEL_Bz);
791             logg_mes(str);
792
793             logg_mes("\nTemp : ");
794            field_get(PANEL_temp);
795             logg_mes(str);
796
797             sprintf(str,"\nX Mg Correction : %g",x_mag_correction);
798              logg_mes(str);
799
800             sprintf(str,"\nY Mg Correction : %g",y_mag_correction);
801              logg_mes(str);
802
803
804           logg_mes("\n_____________________________________________\n\n");
805
806           break;
807       }
808
809      return 0;
810  }
811
812  //RECUPERE LES INFORMATIONS EN MODE GET_MODE (BOUTON ONE SHOT)
813  int CVICALLBACK GET(int panel, int control, int event,
814          void *callbackData, int eventData1, int eventData2) {
815
816      switch (event) {
817        case EVENT_COMMIT:
818
819             SetCtrlVal(panelHandle,PANEL_active_process,1);
820            get_packets(GET_MODE);
821             SetCtrlVal(panelHandle,PANEL_active_process,0);
822
823           break;
824       }
825
826      return 0;
827  }
828
829  //LANCE LA CALIBRATION SUR LE TEMPS CHOISI
830  int CVICALLBACK CALIBRATE(int panel, int control, int event,
831          void *callbackData, int eventData1, int eventData2) {
832
833     long double depart;
834
835      switch (event) {
836        case EVENT_COMMIT:
837
838          x_mag_correction=0;
839          y_mag_correction=0;
840
841          SetCtrlVal(panelHandle,PANEL_active_process,1);
842
843           m=0;
844          GetCtrlVal(panelHandle,PANEL_t_calib,&time_continuum);
845           depart=Timer();
846
847           annule=0;
848          while(Timer()<depart+time_continuum){
849               if(annule) break;
850            get_packets(CALIBRATE_MODE);
851            }
852           if(annule) break;
853
854          get_mag_corrections();
855
856          logg("Calibration successfuly done");
857          sprintf(log_message,"Bx correction : %g",x_mag_correction);
```

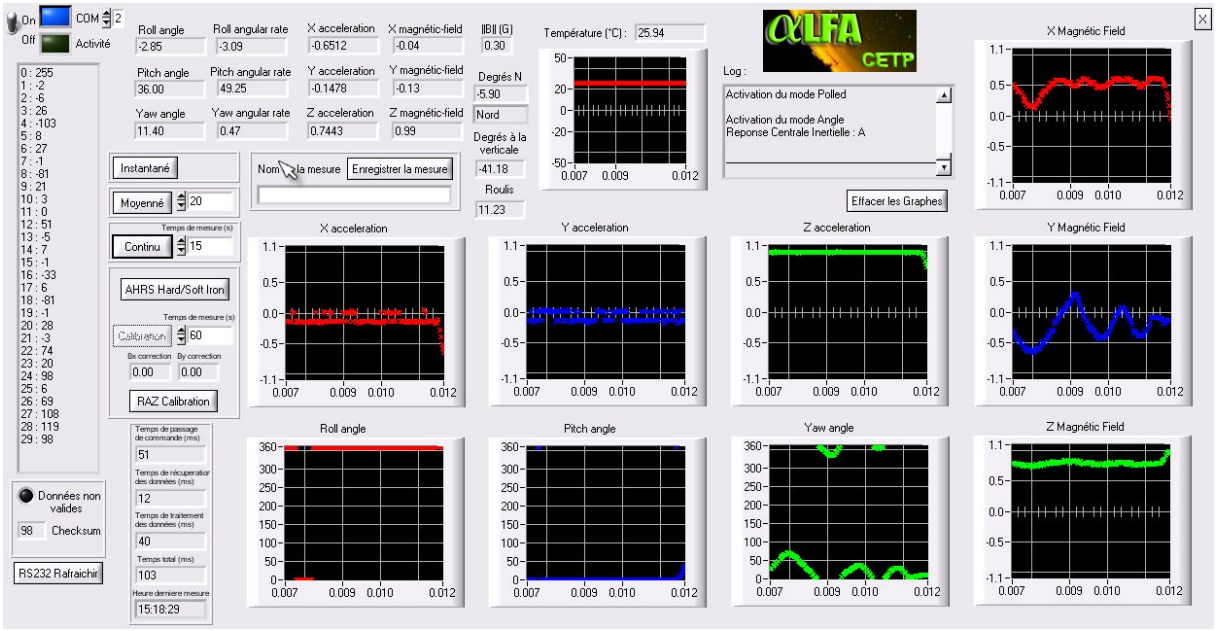
```
858         logg(log_message);
859         sprintf(log_message,"By correction : %g",y_mag_correction);
860         logg(log_message);
861          logg_line();
862
863         SetCtrlVal(panelHandle,PANEL_active_process,0);
864
865          break;
866       }
867
868      return 0;
869   }
870
871   //CE CVICALLBACK RECUPERE LES INFOS EN MODE STABILISED_MODE ET LES MOYENNE P
      OUR ENFIN LES AFFICHER DANS LES CHAMPS NUMERIQUES ET LES GRAPHES
872   int CVICALLBACK stabilise(int panel, int control, int event,
873         void *callbackData, int eventData1, int eventData2) {
874
875      switch (event) {
876        case EVENT_COMMIT:
877
878         SetCtrlVal(panelHandle,PANEL_active_process,1);
879         GetCtrlVal(panelHandle,PANEL_stab_count,&stab_count);
880
881          m=0;
882         while(m<stab_count){
883           get_packets(STABILISED_MODE);
884              m++;
885           }
886
887          for(i=0;i<18;i++){
888              stabilised[i]=0;
889           }
890
891          for(i=0;i<m*18;i+=18){
892              for(j=0;j<18;j++){
893                 stabilised[j]+=moyenne[i+j];
894               }
895           }
896
897          for(i=0;i<18;i++){
898             stabilised[i]/=(double)m;
899           }
900
901          d_field_set(PANEL_roll,stabilised[0]);
902          d_field_set(PANEL_pitch,stabilised[1]);
903          d_field_set(PANEL_yaw,stabilised[2]);
904          d_field_set(PANEL_roll_rate,stabilised[3]);
905          d_field_set(PANEL_pitch_rate,stabilised[4]);
906          d_field_set(PANEL_yaw_rate,stabilised[5]);
907          d_field_set(PANEL_ax,stabilised[6]);
908          d_field_set(PANEL_ay,stabilised[7]);
909          d_field_set(PANEL_az,stabilised[8]);
910          d_field_set(PANEL_Bx,stabilised[9]);
911          d_field_set(PANEL_By,stabilised[10]);
912          d_field_set(PANEL_Bz,stabilised[11]);
913          d_field_set(PANEL_temp,stabilised[12]);
914          d_field_set(PANEL_Bnorme,stabilised[14]);
915
916          get_plots(stabilised);
917
918         SetCtrlVal(panelHandle,PANEL_deg,stabilised[15]);
919         if(stabilised[15]<105 && stabilised[15]>75) SetCtrlVal(panelHandle,PANEL_dir,"Est");
920         if(stabilised[15]>-105 && stabilised[15]<-75) SetCtrlVal(panelHandle,PANEL_dir,"Oue
      st");
921         if(stabilised[15]>-15 && stabilised[15]<15 ) SetCtrlVal(panelHandle,PANEL_dir,"Nord"
      );
922         if(fabs(stabilised[15])>165 && stabilised[9]<0 ) SetCtrlVal(panelHandle,PANEL_dir,"S
      ud");
923         if(stabilised[15]>15 && stabilised[15]<75) SetCtrlVal(panelHandle,PANEL_dir,"Nord E
      st");
924         if(stabilised[15]>-75 && stabilised[15]<-15) SetCtrlVal(panelHandle,PANEL_dir,"Nord
      Ouest");
925         if(stabilised[15]>-165 && stabilised[15]<-105) SetCtrlVal(panelHandle,PANEL_dir,"Su
      d Ouest");
926         if(stabilised[15]>105 && stabilised[15]<165) SetCtrlVal(panelHandle,PANEL_dir,"Sud E
      st");
927
928          SetCtrlVal(panelHandle,PANEL_vertical,stabilised[16]);
929         SetCtrlVal(panelHandle,PANEL_myroll,stabilised[17]);
930
931         SetCtrlVal(panelHandle,PANEL_active_process,0);
```

```
932
933         break;
934      }
935
936     return 0;
937  }
938
939
940  //RECUPERE LES INFORMATIONS EN MODE CONTINU PENDANT LE TEMPS CHOISI
941  int CVICALLBACK CONTINUUM(int panel, int control, int event,
942         void *callbackData, int eventData1, int eventData2) {
943
944     double depart;
945
946     switch (event) {
947       case EVENT_COMMIT:
948
949             old_plots=0;
950           DeleteGraphPlot (panelHandle, PANEL_roll_plot, -1, VAL_IMMEDIATE_DRAW );
951           DeleteGraphPlot (panelHandle, PANEL_pitch_plot, -1, VAL_IMMEDIATE_DRAW );
952           DeleteGraphPlot (panelHandle, PANEL_yaw_plot, -1, VAL_IMMEDIATE_DRAW );
953           DeleteGraphPlot (panelHandle, PANEL_ax_plot, -1, VAL_IMMEDIATE_DRAW );
954           DeleteGraphPlot (panelHandle, PANEL_ay_plot, -1, VAL_IMMEDIATE_DRAW );
955           DeleteGraphPlot (panelHandle, PANEL_az_plot, -1, VAL_IMMEDIATE_DRAW );
956           DeleteGraphPlot (panelHandle, PANEL_Bx_plot, -1, VAL_IMMEDIATE_DRAW );
957           DeleteGraphPlot (panelHandle, PANEL_By_plot, -1, VAL_IMMEDIATE_DRAW );
958           DeleteGraphPlot (panelHandle, PANEL_Bz_plot, -1, VAL_IMMEDIATE_DRAW );
959           DeleteGraphPlot (panelHandle, PANEL_temp_plot, -1, VAL_IMMEDIATE_DRAW );
960
961           SetCtrlVal(panelHandle,PANEL_active_process,1);
962           GetCtrlVal(panelHandle,PANEL_t_cont,&time_continuum);
963
964            depart=Timer();
965
966            annule=0;
967           while(Timer()<depart+time_continuum){
968                if(annule) break;
969             get_packets(GET_MODE);
970              }
971            if(annule) break;
972
973           SetCtrlVal(panelHandle,PANEL_active_process,0);
974             break;
975      }
976
977     return 0;
978  }
979
980  //EFFACE LES GRAPHES
981  int CVICALLBACK clear_plots(int panel, int control, int event,
982         void *callbackData, int eventData1, int eventData2) {
983
984     switch (event) {
985       case EVENT_COMMIT:
986             old_plots=0;
987           DeleteGraphPlot (panelHandle, PANEL_roll_plot, -1, VAL_IMMEDIATE_DRAW );
988           DeleteGraphPlot (panelHandle, PANEL_pitch_plot, -1, VAL_IMMEDIATE_DRAW );
989           DeleteGraphPlot (panelHandle, PANEL_yaw_plot, -1, VAL_IMMEDIATE_DRAW );
990           DeleteGraphPlot (panelHandle, PANEL_ax_plot, -1, VAL_IMMEDIATE_DRAW );
991           DeleteGraphPlot (panelHandle, PANEL_ay_plot, -1, VAL_IMMEDIATE_DRAW );
992           DeleteGraphPlot (panelHandle, PANEL_az_plot, -1, VAL_IMMEDIATE_DRAW );
993           DeleteGraphPlot (panelHandle, PANEL_Bx_plot, -1, VAL_IMMEDIATE_DRAW );
994           DeleteGraphPlot (panelHandle, PANEL_By_plot, -1, VAL_IMMEDIATE_DRAW );
995           DeleteGraphPlot (panelHandle, PANEL_Bz_plot, -1, VAL_IMMEDIATE_DRAW );
996           DeleteGraphPlot (panelHandle, PANEL_temp_plot, -1, VAL_IMMEDIATE_DRAW );
997
998          break;
999      }
1000
1001     return 0;
1002 }
1003
1004 //AFFICHE CE QUI EST CONTENU DANS LE TAMPON RS232
1005 int CVICALLBACK refresh (int panel, int control, int event,
1006        void *callbackData, int eventData1, int eventData2) {
1007
1008     switch (event) {
1009       case EVENT_COMMIT:
1010
1011           SetCtrlVal(panelHandle,PANEL_active_process,1);
1012           ResetTextBox (panelHandle, PANEL_rs232, "");
1013
```

```
1014            read_cnt=GetInQLen(comport);
1015            ComRd(comport,read_data,read_cnt);
1016
1017             for(i=0;i<read_cnt;i++){
1018                 sprintf(str,"%d : %d \n",i,read_data[i]);
1019               SetCtrlVal(panelHandle, PANEL_rs232, str);
1020               }
1021
1022            SetCtrlVal(panelHandle,PANEL_active_process,0);
1023
1024              break;
1025        }
1026
1027
1028       return 0;
1029
1030   }
1031
1032   //REMET LES VARIABLES DE CALIBRATION A 0
1033   int CVICALLBACK clear_cal (int panel, int control, int event,
1034          void *callbackData, int eventData1, int eventData2) {
1035
1036       switch (event) {
1037         case EVENT_COMMIT:
1038
1039              x_mag_correction=0;
1040              y_mag_correction=0;
1041              d_field_set(PANEL_x_correct,0);
1042              d_field_set(PANEL_y_correct,0);
1043               logg(strcat(TimeStr()," : "));
1044               logg("Calibration cleared");
1045                logg_line();
1046
1047               logg("Clear Hard Iron Calibration");
1048              ComWrt(comport,"h",1);
1049               Delay(0.100);
1050              read_cnt=GetInQLen(comport);
1051              ComRd(comport,read_data,read_cnt);
1052               logg_s("Reponse Centrale Inertielle : ");
1053               logg_s(strcat(read_data,"\n"));
1054
1055               logg("Clear Soft Iron Calibration");
1056              ComWrt(comport,"t",1);
1057               Delay(0.100);
1058              read_cnt=GetInQLen(comport);
1059              ComRd(comport,read_data,read_cnt);
1060               logg_s("Reponse Centrale Inertielle : ");
1061               logg_s(read_data);
1062
1063             SetCtrlAttribute(panelHandle,PANEL_calibrate,ATTR_DIMMED,1);
1064
1065            break;
1066        }
1067       return 0;
1068   }
1069
1070
1071   int CVICALLBACK Quit (int panel, int control, int event,
1072          void *callbackData, int eventData1, int eventData2) {
1073
1074       switch (event) {
1075         case EVENT_COMMIT:
1076
1077              RS232Error=CloseCom(comport);
1078               QuitUserInterface (0);
1079
1080            break;
1081        }
1082       return 0;
1083   }
1084
```

Annexe 16 : Mode d’emploi

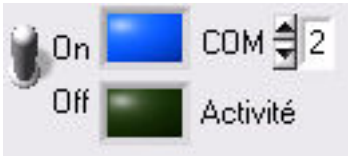
CETP
Projet ALFA
Mode d'emploi du système de pointage



NB : ce dispositif de pointage se repère par rapport au nord magnétique et non par rapport au nord géographique…

Mise en route :

Allumez d'abord le programme puis le dispositif. Sélectionnez votre port COM puis appuyez sur On/Off.



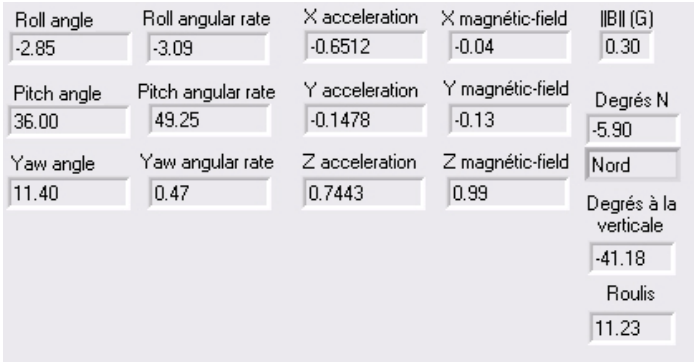
→ Le bouton « On/Off » permet d'activer la connexion au port COM dont le numéro peut être modifié à l'aide d'un sélecteur numérique.

→ Le témoin « COM » est allumé lorsque le port COM dont le numéro est indiqué dans le sélecteur numérique est ouvert.

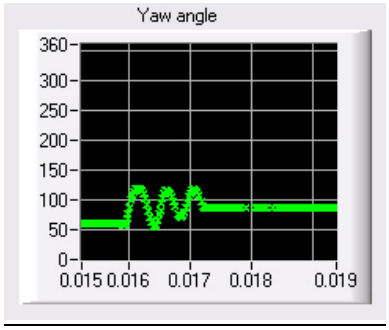
→ Le témoin « Activité » est allumé lorsque l'on récupère ou envoie des informations à la centrale inertielle.

Les modes :

Les mesures, quelque soit le mode dans lequel elles ont été prises, sont présentées numériquement comme ceci :



et sous forme de graphes comme ceci :

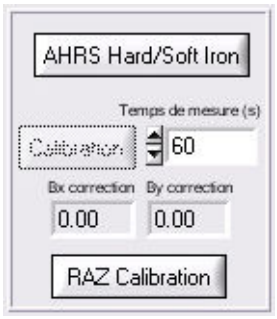


Les différents modes sont explicités ici :

→ Le mode « **instantané** » permet de recevoir un paquet d'information de la centrale inertielle et de calculer les données correspondantes. Les valeurs sont souvent fluctuantes pour une position constante…

→ Le mode « **moyenné** » permet d'éviter que les données soient fluctuantes. Comme son nom l'indique ce mode prendra plusieurs mesures et en fera la moyenne classique. On peut choisir dans un sélecteur numérique le nombre de mesures dont on veut faire la moyenne (de 2 à 40).

→ Le mode « **continu** » permet de prendre autant de mesures que possible dans l'intervalle de temps indiqué dans le sélecteur numérique. On peut ainsi voir les graphes s'actualiser (fréquence d'échantillonnage : environ 100ms). Pendant la mesure il sera impossible à l'utilisateur d'agir sur l'interface. Ce dernier devra attendre la fin du temps de mesure et l'extinction du témoin « Activité » pour pouvoir reprendre la main.

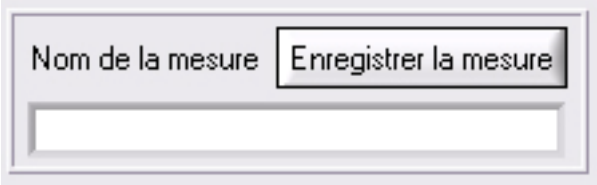


→ Les fonctions de calibration permettent de corriger les erreurs dues aux champs magnétiques perturbateurs internes au système. Ces champs sont considérés comme constants par rapport à la centrale inertielle. Vous devez d'abord effectuer une calibration matérielle en appuyant sur le bouton « **AHRS Hard/Soft Iron** ». La centrale s'auto-calibrera pendant le temps indiqué dans le sélecteur numérique et stockera les valeurs correctives dans sa propre mémoire. Une fois cette étape terminée vous aurez accès à la fonction de calibration logicielle, bouton : « **Calibration** ». Elle fonctionne de la même manière sauf que les valeurs correctives sont stockées et appliquées dans l'ordinateur (elle n'applique des corrections que sur le plan horizontal xy).

Vous devrez, après avoir lancé le mode **AHRS Hard/Soft Iron**, faire faire au moins 1 tour au système dans le plan horizontal et aussi lentement que possible.

Vous devrez, après avoir lancé le mode **Calibration**, faire faire au moins 2 tours au système dans le plan horizontal et aussi lentement que possible. Le programme détectera alors les offsets sur les valeurs du champ magnétique dans le plan horizontal. La correction consiste à soustraire les valeurs des offsets (pour B non normalisé) sur chaque axe à toute nouvelle mesure. Ces valeurs seront enregistrées et automatiquement appliquées à toute mesure par la suite. A chaque fois que vous effectuez une calibration le programme efface les anciennes valeurs des variables de correction.

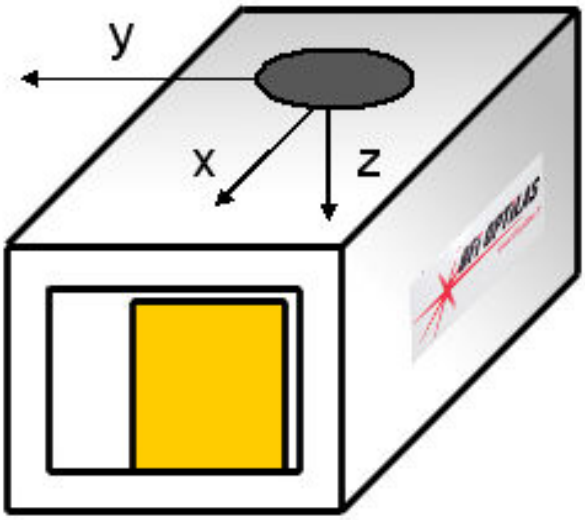
Pour remettre à zéro les corrections (matérielles et logicielles) appliquées aux mesures, cliquez sur le bouton « RAZ calibration ».



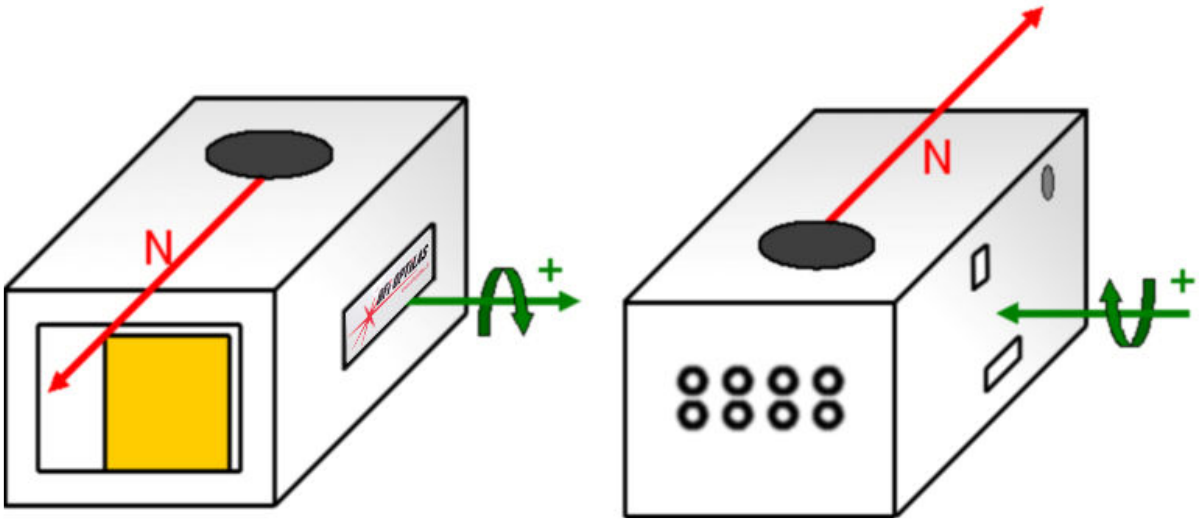
→ Il est possible **d'enregistrer une mesure**. Après avoir effectué une mesure vous pouvez donner un nom à cette dernière et l'enregistrer. Attention : les valeurs enregistrées seront celles contenues dans les champs numériques. Cela signifie que si vous avez utilisé le mode « moyenné » sur 20 mesures, les données enregistrées seront la moyenne sur ces 20 mesures. Ces données s'ajoutent au fur et à mesure dans un fichier de la forme « *mesures MM-JJ-AAAA.txt* ». Une mesure aura la forme suivante :

Mesure : NOM DE LA MESURE
16:36:29
Angle par rapport au Nord : -6.55459 degrés
Direction (approximative) : Nord
Angle à la verticale : -7.85431 degrés
Angle de roulis : 7.94079 degrés
Roll : -1.70288
Pitch : 0.32959
Yaw : 1.11511
Roll rate : -2.57263
Pitch rate: 0.137329
Yaw rate: 0.695801
ax : -0.135368
ay : -0.136878
az : 0.981295
Bx (corrected) : 0.567879
By (corrected) : -0.036556
Bz (corrected) : 0.8223
Temp : 30.0589
X Mg Correction : 0
Y Mg Correction : 0

Repères, axes et sens de rotation :



Ce schéma indique les 3 axes x, y et z du systèmes. C'est par leur projection sur ces 3 axes que l'on exprimera les différents vecteurs mesurés.



Ce schéma indique l'axe nord du système et le sens positif de rotation qui sera pris en compte pour le calcul de l'angle à la verticale.

Unités :

Les unités utilisées dans ce programme sont les suivantes :

- Les angles sont donnés en degrés.

- Les composantes du vecteur champ magnétique sont normalisées elle ne sont révélatrice que de la direction du nord magnétique mais ne correspondent à aucune unité courante.

- La valeur de la norme du champ magnétique est calculée avant la normalisation. D'après le constructeur de la centrale elle est donnée en Gauss.

- Le vecteur accélération est lui aussi normalisé. Donc ses composantes sont exprimées en g.

- Les vitesses angulaires sont données en degrés par seconde.