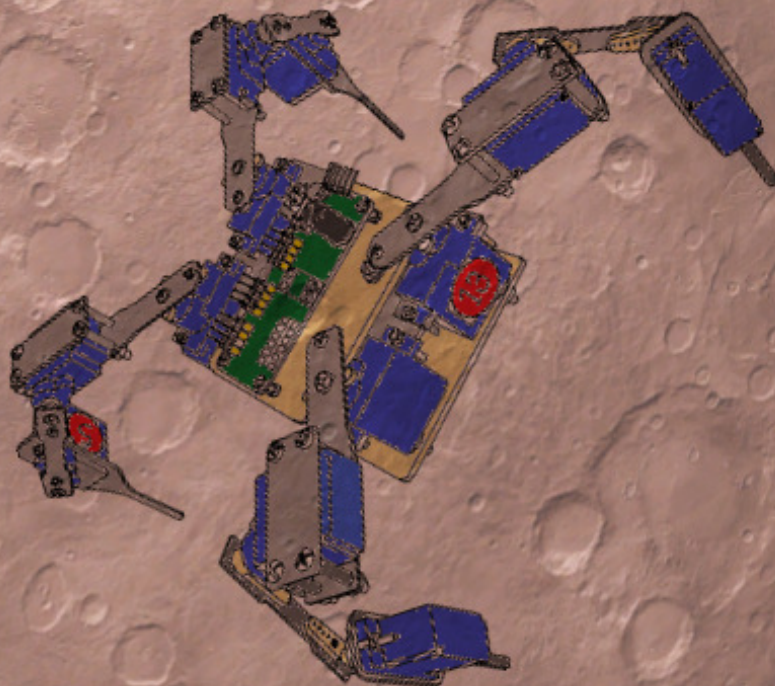


Sonde martienne JEKERT Logiciel de la raquette de pilotage.

TOME 6



Landing Site



Par Nulentout : Mardi 13 Avril 2018.

Développement du projet Sonde martienne JEKERT

Pour mémoire, l'ouvrage précédent s'était achevé tragiquement sur la foudre qui est tombée inopinément sur le complexe de la NDRMSE provoquant une perte de donnée sur ZARIA le gros ordinateur qui héberge l'ensemble des fichiers produits sur les ordinateurs de la salle informatique S4. Pour couronner le tout, les systèmes informatiques de traitement de texte ont sombré dans une saturation matérielle qui a définitivement paralysé la rédaction des documents de suivi du projet, réunis dans ce que l'on nomme à la NDRMSE le **TOME 5**. Des mesures de sauvegarde sont heureusement régulièrement engagées, et malgré ces aléas fréquents dans le monde de la technique, les personnels peuvent reprendre le travail comme si rien ne s'était passé. Les modules logiciels d'exploitation **P21J_Démonstrateur_Raquette.ino** associé à **P22J_Démonstrateur_Sonde.ino** ont été réimplantés sur les machines informatiques. La journée continue sereinement ...

53) 11/01/2018 : Reprise des activités à la NDRMSE (MJD 58129)

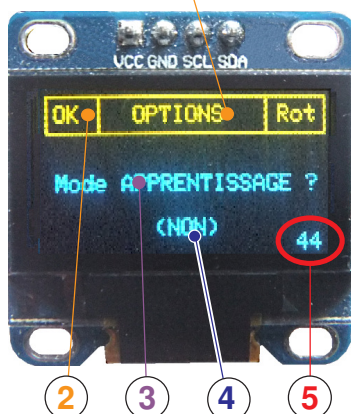
Revenant en salle informatique après avoir pris le repas de midi en compagnie des techniciens, je m'attendais à ce que les informaticiens planchent sur le pilotage manuel des moteurs, conformément au planning. Force est de constater que l'activité actuelle de Ferrando sur son terminal ne semble avoir aucun rapport.

- Jour Max, vous ne travaillez pas sur la motorisation cet après midi ?
- Ben non, c'est Tassin qui s'en charge. Comme on galère trop sur les échanges de données entre les pupitres et la sonde, on va appliquer une nouvelle stratégie. On modifie le contenu des écrans de maîtrise, ainsi l'opérateur aura sous les yeux aussi bien l'information relative à la consigne envoyée sur le canal montant que l'accusé de réception sur la voie descendante.
- Ha bon, la direction est au courant ?
- Naturellement, du reste c'est le Derlo qui nous l'a expressément demandé.

➤ Changement de protocole d'exploitation de l'écran OLED.

Concrètement, ce changement de stratégie a été inspiré par la lourdeur d'avoir à se servir du moniteur de l'IDE pour surveiller les consignes envoyées par le pupitre vers la sonde. Il en a résulté l'idée de faire afficher cette information sur l'écran OLED, artifice d'autant plus justifié qu'il consiste à n'écrire au maximum que deux chiffres sur la matrice de pixels du petit écran. Pas bien compliqué dans ces conditions de trouver un emplacement dans cette optique. Par ailleurs, sur le plan de la crédibilité, cette procédure est parfaitement justifiée. Dans les stations de poursuite ou de gestion de satellites, les opérateurs ont des écrans pour toutes les informations descendantes, mais également vers celles qui sont envoyées aux lointains vaisseaux. La nouvelle répartition des

Fig.273 informations sur la surface utile est représentée sur la Fig.273 avec en haut à droite



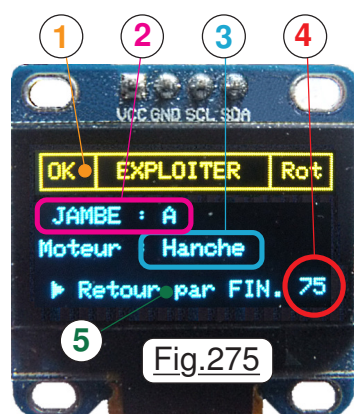
le petit cadre qui ne contiendra désormais que Rot ou CLV. En 1 les titres des menus ont été réduits en largeur pour occuper moins de place : Les caractères '<' et '>' ont été enlevés. Ainsi, sur la petite zone à gauche du cadre jaune en 2 seront affichés les accusés de réception. On retrouve en 3 l'information sur l'item des OPTIONS actuellement indexé dans la liste "déroulante". En 4, s'affiche ici de l'état de l'option. Le code 72 a été envoyé pour sortir du mode apprentissage. JEKERT a satisfait cette consigne et retourné OK. C'est précisément l'accusé de réception qui reste présent en 2. Puis le logiciel du pupitre a transmis la directive 44 affichée en 5. La sonde a alors retourné la donnée d'état du système. Le programme ayant analysé cet ACR en a déduit qu'actuellement la sonde est hors mode apprentissage, et 4 reflète une réalité physique effective.

NOTE sur la nouvelle stratégie d'utilisation de l'écran graphique OLED :

- Le code de consigne envoyé sur **TX** est affiché en bas et à droite. L'emplacement est prévu pour compatibilité avec la grille d'affichage des programmes enregistrés.
(Un chapitre complet sera consacré au mode apprentissage qui a considérablement évolué.)
- L'ACR est affiché en haut à gauche. Donc **E99** n'est plus utile et abandonnée.

➤ Ajout du pilotage manuel des servomoteurs.

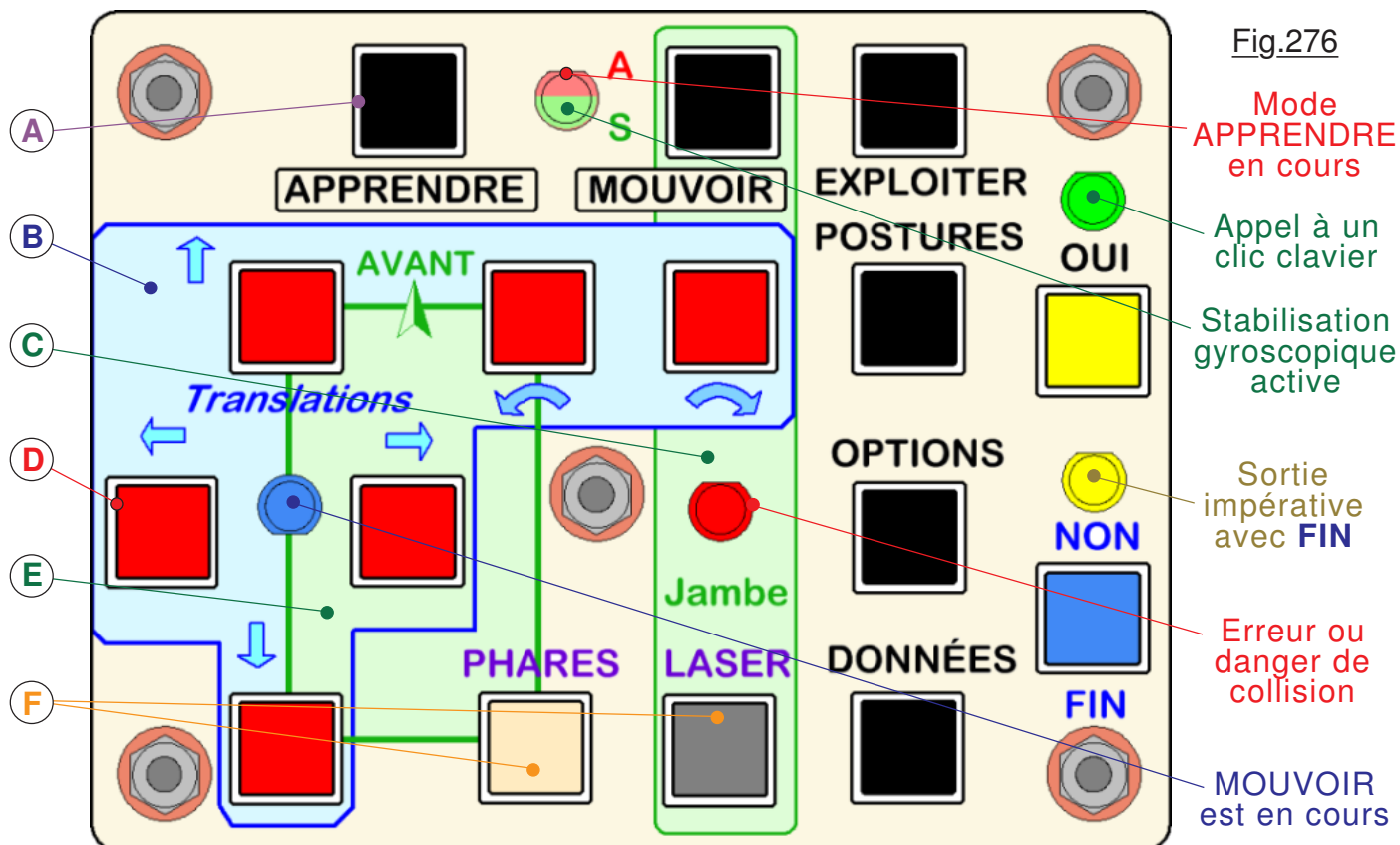
Ayant logé les textes définitifs en EEPROM, téléversez **P21J_Démonstrateur_Raquette.ino** sur la carte Arduino du pupitre et son associé **P22J_Démonstrateur_Sonde.ino** sur celle de la sonde. Nous allons, pour cette journée Julienne 58129 passer en revue les apports de la défunte version **H** dont les octets reposent en paix dans les oubliettes du software. Les fonctions de base étant en place, on commence par ajouter et tester le Pilotage MANUEL et individuel des divers servomoteurs de la petite machine. Pour mémoire les limites de consigne ne sont pas surveillées. Il faut par conséquent ne pas aller chercher les valeurs qui engendrent une divergence des asservissements. Évitez soigneusement d'amener les mouvements trop vers les extrémités des débattements angulaires ... et ne placez pas trop loin le bouton de panique. Pourvoir à tout moment se faire afficher les valeurs des consignes de la posture actuelle permet par comparaison avec celles de la **Fiche n°10** de visualiser l'approche des zones de fonctionnement critiques. Quand on valide la fonction avec **OUI**, la LED rouge de "motorisation en action" se met à clignoter. La LED jaune de "Sortie par **FIN** impérative" aussi. L'écran de la Fig.275 s'affiche, mais surtout, il ne se passe rien sur les moteurs. Pourtant nous



n'avons pris aucune précaution de centrage du potentiomètre comme c'était le cas pour la version initiale du système. **L'avantage considérable d'un potentiomètre numérique, c'est de pouvoir l'initialiser à une valeur quelconque.** Donc, chaque fois que l'on change de servomoteur, le programme va chercher sa valeur actuelle de consigne et la recopie dans la variable gérée par le codeur incrémental. L'ensemble des moteurs reste maintenant totalement inerte, seule l'action sur le codeur engendrera des mouvements. En **4** est indiqué le dernier code envoyé par la raquette avec en **1** l'accusé de réception. En **2** sera indiqué la **Jambe** sur laquelle on agit actuellement, et en **3** quel servomoteur est sollicité. Vous pouvez aussi remarquer qu'en **5** la LED clignotante jaune est compétée par une directive visuelle du genre "j'insiste lourdement !".

Pour des raisons d'agrément, dans ce mode le bouton central du capteur rotatif permet d'alterner entre des incréments angulaires importants ou faibles. Il est ainsi facile à tout moment d'adapter la finesse avec laquelle on engendre les mouvements individuels. Dans sa version ultime, le programme de la raquette de pilotage a considérablement évolué, et telles qu'elles était présentées sur la Fig.243, la disposition des touches et des LEDs du clavier n'est plus idéale. La Fig.276 propose un réaménagement envisagé actuellement pour améliorer la convivialité opérationnelle.

Bien que toutes les LEDs actuellement gérées par l'ATmega328 ne sont pas regroupées sur le clavier, on note que la LED d'incitation à cliquer sur une touche est toujours voisine de la touche **OUI** mais de couleur verte. La diode jaune qui clignote pour sortir d'un mode impérativement par la touche **FIN** est maintenant juste au dessus de la touche concernée. Changement de stratégie qui sera effectif dans la version du logiciel qui traite de l'APPRENTISSAGE, un menu spécifique sera réservé à ce mode particulier d'exploitation de la sonde sur le terrain. En version ultime, APPRENTISSAGE a considérablement évolué, le nombre de ses options encombrerait exagérément le menu **EXPLOITER**. Aussi, on a abandonné la touche dédiée à l'annulation des efforts, item qui fait alors partie des **POSTURES**. La touche **A** est donc devenue une fonction permanente au même titre que les cinq autres touches fonctionnelles. Du coup la zone **B** relative aux mouvements n'encadre plus **S10**. Naturellement cette zone colorisée en bleu regroupe tous les boutons rouges tel que **D**, générant les déplacements élémentaires. Quand on pilote un moteur en mode manuel, il



devient évident de savoir instinctivement sur quelle touche agir pour changer d'articulation. Dans ce but la sérigraphie comporte deux zones vertes **C** et **E**. Quand on est en mode pilotage manuel des servomoteurs, il faut oublier l'usage standard des diverses touche du clavier, seules celles coloriées

sur la Fig.277 auront un effet. En particulier les deux boutons **F** n'agissent plus sur les phares et sur le LASER. La zone verticale **C** doit faire penser à une **Jambe** dont les boutons en partant du haut vers le bas représentent respectivement la **Hanche**, le **Genou** et le pivot du **Pied**.

La zone rectangulaire **E** représente la sonde vue par dessus, les touches situées dans les angles symbolisant les **Jambes**. Dans cette symbolique, l'**Avant** de la petite machine est indiquée par la flèche verte. Quand on désire changer de servomoteur, on désignera dans un ordre arbitraire l'articulation concernée et avec **A**, **B**, **C** ou **D** la **Jambe** sur laquelle on désire agir.

Lorsque l'on a abouti à une configuration intéressante, on peut à tout moment sauvegarder les valeurs de la posture actuelle en EEPROM avec **Sauver la POSTURE ?** la commande dédiée qui suit l'écriture en EEPROM la commande d'affichage d'un balayage panoramique télémétrique. (Voir éventuellement la Fig.259 **B**) Il sera ainsi rapide et facile d'imposer cette configuration avec la commande de la Fig.278 en étant bien conscient qu'une telle manipulation peut présenter un danger pour la mécanique. Par exemple vous avez enregistré la posture de décollage.

Puis, quelques temps plus tard, la sonde étant posée au sol, vous croyez que la dernière sauvegarde concernait la posture **Stable transversal**. Passer d'une machine posée au sol à la rétractation vers le bas des **Jambes** ne sera absolument pas apprécié par la motorisation qui subira des efforts exagérés, sans compter les "chaussettes" qui vont copieusement frotter sur le sol. **Donc à utiliser cet item avec prudence.**



OUPSSsss : J'ai oublié de vous prévenir. Avec la version actuelle et toutes celles qui suivent, l'affichage d'un spectre colorimétrique n'est plus du tout correct. Les valeurs numériques sont bien en place, mais les textes sont incohérents. Ce détail n'a pas été corrigé, car dans la version la plus aboutie ce sera un beau graphique qui remplacera ces données numériques.

NOTE : Le petit livret **Manuel d'utilisation** de la sonde réserve une page spécifique pour décrire les limites et le **Protocole de pilotage manuel des moteurs**.

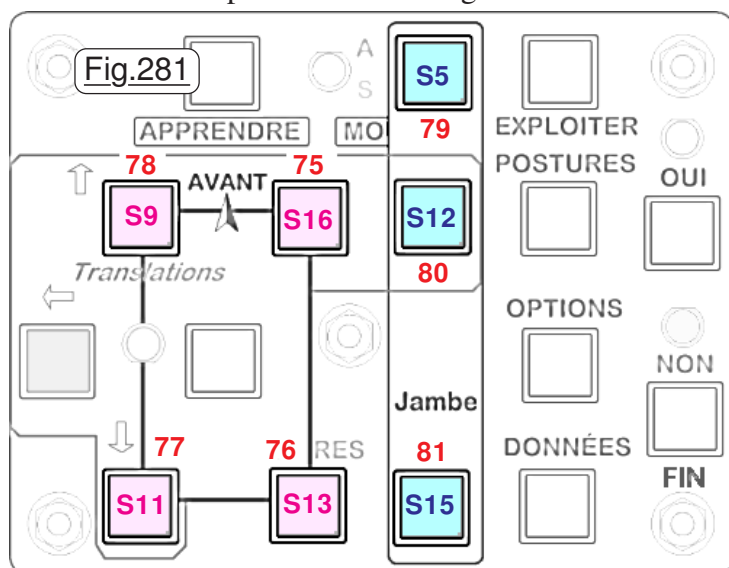
➤ Affichage des consignes d'une posture.

Particulièrement utile pour effectuer des analyses diverses, on peut par deux items du menu **EXPLOITER** se faire afficher à convenance les douze consignes de motorisation correspondant soit à la configuration actuelle, soit à la posture sauvegardée en EEPROM. La première s'obtient avec la commande de la Fig.279 **X** alors que la posture logée en EEPROM sera listée par l'item **Y**. Quand on valide avec la touche **OUI** on obtient un écran comme celui représenté sur la Fig.280 et la LED jaune se met à clignoter annonçant qu'il sera important de quitter par usage de la touche **FIN**. La copie d'écran montre que les valeurs des consignes sont organisées sous forme d'une matrice dont les lignes sont concernées par les articulations et les colonnes sont représentatives des **Jambes** de l'insecte mécanique. Compte tenu de la répartition des touches sur le clavier ergonomique, la Fig.281 situe la référence des divers boutons poussoir. En rouge le dessin est complété



Fig.280

	EXPLOITER				Rot
Hanche	239	344	222	346	
Genou	315	296	298	291	
Pied.	421	192	426	199	
	A	B	C	D	
	Jambe				



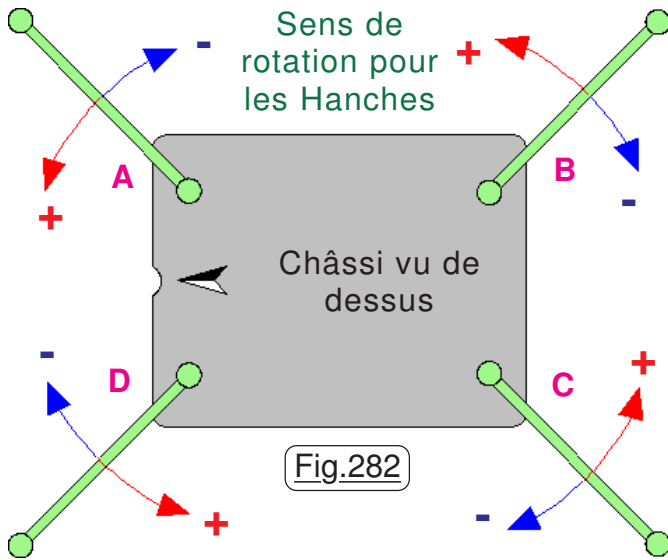
pour chaque touche par la valeur du code consigne qui permet au logiciel de la Raquette de dialoguer avec celui de la sonde. Analysons la chaîne logicielle qui anime les moteurs dans le mode manuel d'exploration sur le sol poussiéreux. La première instruction que reçoit la sonde est codée **73** et le **switch case** qui trie les ordres reçus se contente de forcer la variable **Debattement** à **12** pour générer par défaut des amplitudes de mouvements importantes. Cette

ligne d'instructions initialise l'identificateur **Cible_Codeur_incremental** à la valeur **5**. Ce sélecteur chiffre signifie que le **codeur incrémental** concerne la **motorisation en mode manuel**. Par défaut la raquette transmet aussi les consignes **75** et **79** pour forcer par défaut la **Jambe A** et le servomoteur de la **Hanche** pour cible. Quand sur la raquette on clique sur l'une des touches colorisées en rose ou en bleu sur la Fig.281 la voie montante transmet l'instruction associée telle que **80** ou **77** notée en rouge à proximité de chaque bouton poussoir sur le dessin. Le logiciel de la sonde change alors de servomoteur cible. Quand on clique sur le bouton central du codeur rotatif, la variable **Debattement** alterne entre **12** et **2** changeant les balayages angulaires entre "grands" et "petits" :

```
case 66 : {if (Debattement == 12) Debattement = 2; else Debattement = 12; break;}
```

Les mouvements sont déclenchés par la rotation du codeur incrémental qui, en fonction du sens de rotation, fait envoyer à la sonde le code **40** ou le code **41** :

```
case 40 : {Faire_un_pas_sens_Positif(); break;} // Codeur Incrémenté.
case 41 : {Faire_un_pas_sens_Negatif(); break;} // Codeur Décrémenté.
```



Les deux procédures qui sont consacrées à la gestion des rotations du codeur incrémental traitent aussi bien les pointages LASER, les gradateurs lumineux ainsi que le pilotage manuel. C'est la variable **Cible_Codeur_incremental** qui initialisée à la valeur **5** aiguille le comportement du programme pour les ordres **40** et **41**.

```

void Faire_un_pas_sens_Positif() {
    switch (Cible_Codeur_incremental) {
        .....
        case 5 : {Ajuste_la_position_du_servomoteur(true); break;}
    }
}

void Faire_un_pas_sens_Negatif() {
    switch (Cible_Codeur_incremental) {
        .....
        case 5 : {Ajuste_la_position_du_servomoteur(false); break;}
    }
}

```

L'observation des listage partiels de ces procédures montre qu'en fin de compte une seule sous-routine **Ajuste_la_position_du_servomoteur(sens)** réalise l'intégralité des "mouvements manuels". Pour en saisir le fonctionnement, la Fig.282 présente le comportement des servomoteurs des **Hanches** en fonction du signe de la consigne qu'ils doivent recevoir, alors que la Fig.283 résume l'action des moteurs qui animent les **Griffe**. Cette procédure optimisée se résume à peu de chose :

```

① void Ajuste_la_position_du_servomoteur(boolean Augmente) {
②   int Consigne;
③   Consigne = Configuration_actuelle[Num_Sortie];
④   if (Augmente) Consigne = Consigne + Debattement;
⑤   else Consigne = Consigne - Debattement;
⑥   Mouvoir(Num_Sortie, Consigne);}

```

La variable locale **Augmente** passée en paramètre sous la forme d'un booléen à la procédure en ① précise si la valeur de **Consigne** doit augmenter ou diminuer. En ② on déclare en local **Consigne** pour ne pas interférer sur les variables globales. L'instruction ③ récupère la valeur actuelle de configuration du servomoteur cible. En fonction du sens de rotation désiré, les lignes ④ et ⑤ ajustent la valeur de **Consigne** pour la nouvelle position angulaire à adopter. Enfin, en ⑥, si les moteurs ne sont pas sur OFF, la procédure **Mouvoir** transmet la **Consigne** au multiplexeur et met à jour le tableau **Configuration_actuelle[Num_Sortie]**.

Notez au passage que ce sont les ordres compris entre **75** et **81** qui initialisent la variable **Num_Sortie**. La valeur affectée à **Num_Sortie** est obtenue par une "astuce de programmation" en vue d'optimiser le code. Comme on peut le constater sur le listage de la Fig.284 chaque code reçu pour changer de cible modifie le contenu des deux variables dédiées **Jambe** et **Articulation**. Pour en déduire la valeur du moteur concerné et de **Num_Sortie** associé du multiplexeur,

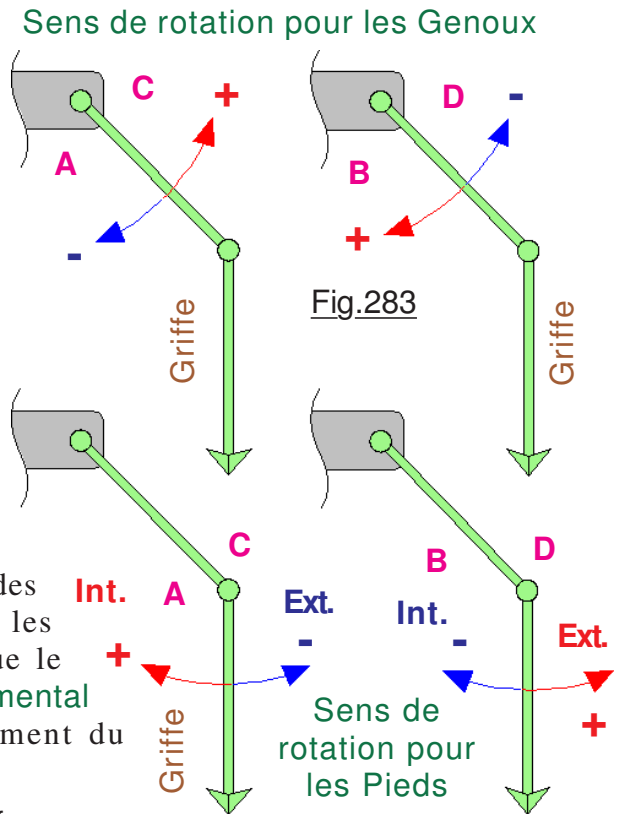


Fig.284

```

case 75 : {Jambe = 0; Calcule_num_sortie(); break;} // Jambe A.
case 76 : {Jambe = 3; Calcule_num_sortie(); break;} // Jambe B.
case 77 : {Jambe = 6; Calcule_num_sortie(); break;} // Jambe C.
case 78 : {Jambe = 9; Calcule_num_sortie(); break;} // Jambe D.
case 79 : {Articulation = 0; Calcule_num_sortie(); break;} // Hanche.
case 80 : {Articulation = 1; Calcule_num_sortie(); break;} // Genou.
case 81 : {Articulation = 2; Calcule_num_sortie(); break;} // Griffes.

```

```
void Calcule_num_sortie() {Num_Sortie = Jambe + Articulation;}
```

le calcul dans la sous-routine `Calcule_num_sortie` devient élémentaire. Cette technique est infiniment plus économe en taille de programme que si le traitement s'était orienté de manière "banale" vers un `switch case` ou une batterie de test de type `if`. Le petit tableau de la Fig.285 résume ces traitements.

Fig.285

Jambe	A			B			C			D		
Moteur de	Hanche	Genou	Pied.	Hanche	Genou	Pied.	Hanche	Genou	Pied.	Hanche	Genou	Pied.
Num_Sortie	0	1	2	3	4	5	6	7	8	9	10	11
Articulation	0	1	2	0	1	2	0	1	2	0	1	2
Jambe	0			3			6			9		

← Somme

➤ Simulation de tirs au LASER.

L'expérience scientifique qui a été mise au point par nos chercheurs et nos techniciens toulousains, cette technologie (*Cocorico !*) a été sélectionnée par les Américains pour se voir embarquer à bord du plus gros "rover" posé sur Mars : L'explorateur Curiosity, une machine absolument hors du commun. Le fondement de cette manipulation consistait à bombarder un rocher avec un laser très puissant pour faire fondre le matériau. L'analyse spectrale de la couleur des gaz qui en résulte permet aux savants d'en déduire une foule d'informations relatives à la planète rouge. La petite simulation présentée ici consiste à réaliser une rafale d'éclairs de quelques secondes à la puissance maximale quelle que soit l'état actuel du LASER. Seule limite à cette action : Le système électrique de bord ne peut en aucun cas passer outre le disjoncteur s'il est déclenché. Dans ce cas il n'y aura aucun effet mis à part l'accusé de réception **OK** qui termine l'action issue de l'instruction ayant pour code **9**.

Pas besoin de se torturer les méninges pour comprendre que c'est l'item de la Fig.286 qui permet de déclencher un tir LASER.

ATTENTION : Le "rayon de la mort" sera émis inconditionnellement et sans aucune vérification, y compris si le LASER est actuellement éteint ou à un niveau énergétique minimal de 1. *N'activez cette fonction* avec la touche **OUI** *que si la direction du canon à photons est dirigée loin du regard de toute personne présente dans le local.*



Fig.286

Sur la sonde, la procédure invoquée pour réaliser cette action est du genre presque dérisoire :

```

void Simule_un_Tir_au_LASER() {
  ① for (byte N = 0; N < 35; N++)
  ②   {analogWrite(LASER,254); delay(25); analogWrite(LASER,1); delay(50);}}

```

En ① on organise une boucle qui va faire 35 fois la séquence ② : Allumer le LASER à 254 le maximum, durant 25 millisecondes. Puis l'éteindre durant 50 millisecondes. Concrètement on ne l'éteint pas complètement, car le code généré est plus compact si on se contente de ramener la valeur à 1. Cette méthode est induite par le fait que la sortie binaire nommée LASER fonctionne en P.W.M. Tout compris, elle n'augmente la taille du programme de la sonde que de 40 octets.

Avec cette description nous en avons terminé la liste des apports qui étaient le fait de la version **H** des deux programmes. Nous allons pouvoir passer en revue les compléments qui émanent de la version **J** actuellement dans les méandres binaires des deux microcontrôleurs.

54) 13/01/2018 : JEKERT retourne à l'école en apprentissage (MJD 58131)

Bien que l'apport considérable qu'apporte la version **J** des démonstrateur réside dans une approche entièrement revue du mode apprentissage, avant de s'embarquer dans la complexité des procédures qui sont impliquées, nous allons passer en revue quelques détails plus élémentaires, qui toutefois modifient notablement la façon dont on s'occupera de la petite machine.

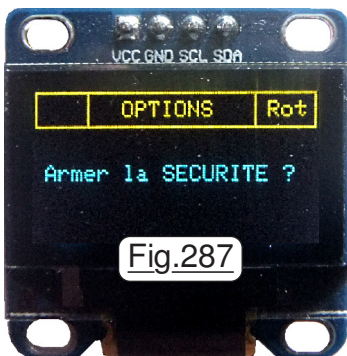
- *Jour les binaires, zavez l'air réjouis ce matin !*

- *Normal chef chef, on a consulté le planning. Ce n'est pas ce samesi qui va nous donner une méningite. Que des brouilles à peaufiner. On vous fait ça avant ce soir, pour une fois on va terminer la semaine relativement pénards.*

➤ Les petits changements qui émaillent la version J.

L'amélioration de détail la plus importante consiste à associer au pilotage de JEKERT la notion de SECURITÉ. Une manipulation malheureuse, et sans vraiment le vouloir on déclenche une action pas du tout souhaitée. Par exemple on passe en stabilisation automatique alors que la sonde est sur son berceau. Comme elle porte sur son bouclier, les membres ont beau s'animer dans tous les sens, sans qu'elle ne bouge d'un iota. Hors, même si l'inclinaison en roulis ou en tangage est faible, par exemple 1°, l'automatisme va faire tourner les servomoteurs jusqu'à avoir l'assiette strictement nulle pour les deux axes. Comme l'écart ne s'annule pas, les moteurs vont aller inexorablement en butée. Pas bon du tout ça ! Si l'on n'y prête pas attention, les moteurs bloqués vont chauffer jusqu'à leur dégradation comme ce fut le cas pour le moteur n°17 de JEKERT.

L'idée de SECURITÉ est toute simple. On ajoute une LED rouge sur le pupitre (*Pilotée par la broche d'entrée analogique **A2** utilisée en sortie.*) qui sera nommée très originalement **SECURITÉ**. (*Sacrément bien trouvé le nom de ce témoin lumineux !*) Par un item dans l'un quelconque des menus, on peut à convenance l'allumer. On peut aussi l'éteindre, mais en général ce ne sera utile que si on l'a allumée par erreur de manipulation. Chaque fois que le programme est sollicité par la



touche **OUI** sur une fonction potentiellement à risque, il vérifie si la LED rouge est allumée. Si ce n'est pas le cas, il refuse l'ordre et retourne **E12**. Si la LED est allumée, alors le programme réalise correctement la directive. (*À condition que ce soit possible. Par exemple un code **3** sera refusé et générera un **E6** si un obstacle est trop proche.*) L'action étant terminée, alors la LED rouge est systématiquement éteinte. Ainsi on sera obligé d'activer la sécurité pour le prochain code à risque potentiel. Cette notion sera généralisée dans la version ultime de la raquette de commande. Dans l'état actuel de l'écriture des programmes, l'item se trouve dans la liste du menu des **OPTIONS**. Montré sur la Fig.287, cliquer sur **OUI** ou

sur **NON** n'engendre pas d'effet sur l'écran. Afficher l'état comme pour les autres options n'est pas utile, puisque la LED dédiée rouge est directement représentative de l'état du booléen **SECURITÉ**.

Affecter une fonction au bouton central du codeur incrémental dans chaque menu. Le logiciel étant encore hors de vue de son achèvement, il est probable que la fonction actuellement obtenue risque de changer dans les versions ultérieures. Peu importe, disposant de cinq menus, c'est cinq possibilités à exploiter. Dans l'état actuel des choix effectués, dans le menu des **OPTIONS** le BP central allume la LED rouge **SECURITÉ**. L'avantage de cette procédure, c'est de ne pas avoir à cheminer dans de nombreux items. On clique sur la touche du menu **OPTIONS**, puis sur le **BPCCR** et l'on peut alors faire afficher à l'écran l'appel à la fonction désirée qui teste la LED rouge.

Pour la suite du didacticiel, on utilisera **BPCCR** qui signifie :
Bouton **P**oussoir **c**entral du **c**odeur **r**otatif.

Notez que le **BPCCR** ne fait qu'allumer la **SÉCURITÉ**. Si on a armé sans vraiment le vouloir, pour éteindre la LED il faut faire appel à l'item de la Fig.287 dans le menu des **OPTIONS**.

Pour le moment, ce qui semble le plus pertinent dans le menu des **POSTURES**, c'est la fonction **Réveiller JEKERT**. Il est possible, voir probable, qu'à l'expérience nous soyons amenés à trouver bien plus pertinent : Demain est un autre jour ...

N'étant pas prioritaire à ce stade du développement, pour le moment le **BPccr** est ignoré dans les menus **MOUVEMENTS** et **DONNEES**. Travailler sur le mode apprentissage a fait ressortir un petit manque dans les fonctionnalités logicielles embarquées. Imaginons que lors d'une démonstration, on désire faire clignoter les phares par exemple. Manuellement les allumer et les éteindre en respectant une certaine cadence est possible. En revanche, aucun moyen actuellement ne permet d'effectuer une temporisation automatique. Hors une telle instruction serait bien utile quand on désire marquer une petite pause entre deux mouvements particuliers par exemple. Le code **90** est réservé pour imposer au programme de JEKERT de réaliser une temporisation d'une seconde.

Étant en train de faire apprendre un programme à la sonde, nous pouvons être amenés à jongler entre les instructions de déplacements et celles d'exploitation. Intercaler des pauses devient un enfer si cette fonction est disponible sous forme d'un item noyé dans un quelconque menu. C'est la raison pour laquelle, dans cette version du démonstrateur, le code **90** sera généré par le **BPccr** dans le menu **EXPLOITER**. Cette commande ne sera effective que si l'on n'est pas en mode Utilisation du LASER ou en pilotage MANUEL des servomoteurs, pour ces actions le **BPccr** n'étant pas disponible. Autre facilité apportée par la version **J**, enregistrer en EEPROM un spectre colorimétrique, un balayage ultrasons, ou bien la posture actuelle est suivi de l'affichage de l'écran des données générées pour voir immédiatement le résultat. Ce n'est pas fabuleux en soit, mais bien utile en exploitation.

55) 15/01/2018 : Apprendre c'est avant tout mémoriser (MJD 58133)

Belle journée qui commence ce lundi MJD58133. Passant devant la vitre de la salle S6, je constate que JEKERT bêta est au repos, les lumières sont éteintes. Bêta est un clone strictement identique à la machine qui a été sélectionnée pour le grand voyage. On a réalisé une deuxième sonde totalement identique à JEKERT, mis à part qu'elle n'a pas sa centrale de génération électrique nucléaire. Cette machine sera activée chaque fois que les ingénieurs chargés du suivi auront à tester de nouveaux programmes, ou mettre au point des manipulations qui sur site pourraient mettre en danger la mission. Bêta étant en sommeil, j'ai compris que ce jour les "ingés" vont plancher sur du visuel animant les consoles, et non sur des séquences complexes agissant sur la motorisation.

➤ Visualiser un programme sur l'écran OLED.

Incontestablement, le mode apprentissage constitue le thème principal de la version **J** avec des séquences assez lourdes à mettre au point. Dans ce chapitre on va s'occuper de la présentation sur le petit écran du listage d'un programme. On peut supposer qu'afficher des codes numériques n'est pas spécialement vital, et qu'il serait plus judicieux de chercher en premier à faire fonctionner correctement l'édition, l'exécution, l'effacement etc. La réalité informatique est plus compliquée que ça. En effet, visualiser un listage sur le Moniteur vidéo de l'**IDE** ne présentait pas de contraintes en nombre possible de codes dans un programme, **la liste défilant verticalement sur l'écran**.

Il serait possible d'envisager sur OLED un tel défilement, ce qui génèrerait deux gros problèmes :

- Le traitement d'un défilement vertical engloutit beaucoup d'octets, (*À été testé.*)
- Outre qu'il impose de cliquer sur des touches pour faire évoluer les lignes affichées, le résultat n'est pas agréable du tout car **nous n'avons jamais la totalité du programme sous les yeux**.

Aussi, la solution adoptée consiste à afficher intégralement le programme sur la petite surface du cadre bleu de la matrice graphique. Mettre au point une présentation est complexe et impose de très nombreux essais. Si chaque tentative doit téléverser un programme de plus de 18ko, le temps de transfert devient excessif. Il faut impérativement, dans un tel cas, se créer un démonstrateur très allégé ne comportant que les routines de servitude pour afficher sur l'écran. En l'occurrence, c'est le module **Test_AFF_PGM.ino** que vous téléversez sur Arduino, "Sketch" qui pourra éventuellement vous servir à modifier à votre guise le code source pour adopter une présentation personnelle.

Purement visuel, quand vous activez ce programme, il commence par construire une grille dans laquelle seront placés les codes du programme listé. L'ordre de lecture consiste à balayer du haut vers le bas la colonne la plus à gauche. Puis on passe à la colonne voisine à droite et ce jusqu'à ce que tous les ordres soient affichés. Ce démonstrateur a permis d'optimiser l'usage de l'écran. On a recherché à placer le plus de lignes et le plus de colonnes possibles, tous en ayant facilité de lecture pour les codes chiffres inscrits dans les cases. En conservant un espace de deux pixels tout le tour des codes, on aboutit à une grille de 8 x 4 soit 32 cellules.

Désirant un maximum d'informations simultanément présentes à l'écran, l'avant dernière cellule, mise en évidence par une inversion de luminosité, sert à préciser quel est le programme actuellement en cours de traitement. Nous savons que maintenant ce seront neuf programmes indépendants qui seront possibles. La grille étant définie, c'est à ce stade qu'ont été définitivement adoptées les coordonnées d'affichage de la commande envoyée à la sonde. Ainsi, on réserve la dernière case pour y afficher le symbole des ordres transmis sur la voie montante.

Deuxième difficulté à aplanir, et pas des moindres : Comment présenter à l'écran l'évolution d'un programme qui serait déclenché en exécution ? Plusieurs méthodes sont possibles, il importe de sélectionner la plus "visuelle". L'une de celles qui ont été testée consistait à effacer dans les cases les consignes au fur et à mesure qu'elles sont réalisées par la sonde. L'inconvénient, c'est que l'on voit bien ce qui reste à réaliser, mais on perd l'information de ce qui a été déroulé. Aussi, la solution estimée la plus judicieuse consiste à cocher chaque case lorsque l'action a été confirmée par la sonde. C'est ce que fait le démonstrateur actuel. Pour voir ce que donne ce cochage, assez similaire au protocole suivi par un pilote sur un aéronef qui consulte ses check-lists, vous devez enlever le "/" du démonstrateur en ligne 45 de la boucle de base. Vous pouvez ensuite valider la ligne 47 du code source et voir l'effet qui en résulte sur OLED.

➤ Nouvelle implantation EEPROM du programme Esclave.

Avant mis au point la présentation des listage qui sera émulée par le programme, il devient alors possible de définir le nombre de commandes maximal par programme, soit trente, et par voie de conséquence établir l'implantation en mémoire non volatile de ces derniers. Chaque programme va donc consommer 30 octets pour la place réservée aux trente instructions possibles. Il faut également prévoir un octet pour indiquer combien de codes sont contenus dans le programme.

Comme nous disposons en EEPROM de place à revendre, on peut se permettre de "gaspiller" une cellule par programme. De cette façon chaque bloc sera séparé du suivant de 32 octets ce qui sur un listage du type de celui de la Fig.288 facilite le travail informatique. Dans l'ancienne organisation, la zone violette était réservée à la sauvegarde d'un balayage télémétrique. C'est toujours le cas du reste. C'est à la suite que l'on va placer les neuf blocs réservés à l'apprentissage. Les encadrés rouges indiquent le nombre d'instructions contenues dans le bloc. Dans l'exemple de la copie d'écran tous les programmes sont vierges, dans ce cas la valeur dans la cellule est zéro. Puis les zones colorées en rose correspondent aux emplacements des codes d'instructions. Le plus grand

0464	11	02	1A	03	1C	03	1B	03	1A	03	1B	03	1A	03	FF	FF
0480	00	FF	(PGM 1)	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0496	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0512	00	FF	(PGM 2)	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0528	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0544	00	FF	(PGM 3)	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0560	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0576	00	FF	(PGM 4)	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0592	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0608	00	FF	(PGM 5)	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0624	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0640	00	FF	(PGM 6)	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0656	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0672	00	FF	(PGM 7)	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0688	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0704	00	FF	(PGM 8)	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0720	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0736	00	FF	(PGM 9)	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0752	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0768	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0784	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Fig.288

code envisagé est 99, donc un octet est largement suffisant. Enfin laissés en blanc, les neuf octets non représentatifs qui sont honteusement gaspillés. La Fig.288 résulte d'une mémoire EEPROM vierge pour laquelle les neuf programmes ont été effacés, ainsi il sera impossible de travailler (*Mis à part y loger des instructions.*) sur des blocs dont les cellules ne contiennent que des aberrations.

Tester les nombreux démonstrateurs qui sont proposés dans ce didacticiel peut engendrer des incidents si vous utilisez un microcontrôleur neuf, avec une EEPROM vierge. Vous avez impérativement inscrit les postures pour que les mouvements de la sonde ne soient pas des "étrangetés" qui amènent les moteurs au blocage mécanique. Reste que faire afficher un spectre coloribétrique, un panoramique ultrasons ou une posture personnelle conduira à des incohérences sur l'écran. Pour avoir dès le début une mémoire EEPROM entièrement remplie par des données cohérentes, un programme spécifique **P60_Clone_EEPROM_sonde.ino** sera mis à votre disposition. **Page 10**

56) 16/01/2018 : Nouvelles méthodes d'apprentissage (MJD 58134)

Pièce maitresse du mécano logiciel embarqué par la sonde : Sa faculté d'apprendre, de pouvoir installer dans sa mémoire non volatile de nouveaux programmes qui seront développés en toute sérénité à Terre. L'expérience a montré depuis des décennies que cette possibilité de télécharger du code à bord des vaisseaux spatiaux a sauvé plus d'une mission, les exemples astronautiques foisonnent. En salle S4 les ingénieurs logiciel se sont partagé la tâche, car c'est un gros morceau.

- *Bonjour les binaires, ça discute fort ce matin, zavez gagné au LOTO ?*
- *Ben non chef, sinon nous ne serions plus ici à gamberger des lignes de code à n'en plus finir.*
- *Ouais ... on a réparti les blocs à écrire, car ya du maille.*
- *Moi j'ai pas de chance, je dois me coltiner l'éditeur.*
- *Plains-toi, ma pomme c'est le listage à l'écran que je vais devoir pondre.*
- *Bricole l'affichage, perso c'est l'exécution des ordres qu'il me faut tortiller, c'est le plus costaud.*
- *Arrêtez les lamentations les gars, dès que l'un aura torché son module, il aidera les autres !*

➤ **Chamboulement du mode APPRENTISSAGE.**

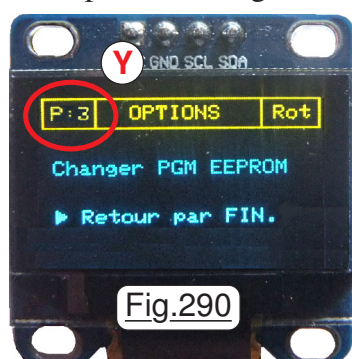
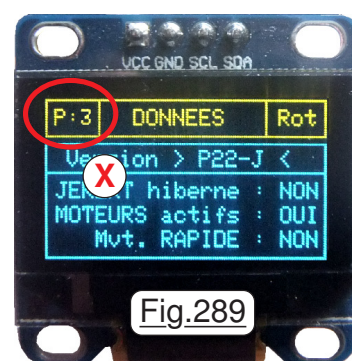
Effectivement, émuler un mode apprentissage impose plusieurs fonctions supplémentaires à ajouter au jeu d'instructions des programmes résidents sur la machine. On comprend pourquoi les personnels se sont répartis la tâche. Il aurait été possible de se contenter du minimum vital comme c'était le cas pour la version qui dialoguait sur la ligne USB du moniteur de l'**IDE**. La version de pilotage par pupitre autonome apporte des améliorations substantielles au regard du service minimal. Voici la liste des fonctionnalités qui seront désormais disponibles, les perfectionnements étant colorés en violet, alors que ce qui relève du strict indispensable est en bleu :

- Créer un programme préservé en EEPROM, (*Et parer d'éventuelles erreurs.*)
- Effacer le programme pour pouvoir en inscrire un autre,
- Lister le programme pour impérativement le vérifier avant de le soumettre à la machine,
- Faire exécuter les instructions du programme sauvegardé par JEKERT,
- Éditer le programme, c'est à dire pouvoir y ajouter des instructions,
- Corriger le programme, c'est à dire pouvoir effacer sa (Ou ses) dernière(s) instruction(s),
- Créer non pas un seul programme, mais jusqu'à neuf logiciels indépendants,
- Pouvoir déplacer l'un des programmes dans l'une des séquences à condition qu'elle soit vide.


Nous allons pas à pas passer en revue ces différentes fonctions, et décrire leurs spécificités ainsi que les écrans dédiés sur l'afficheur OLED. C'est l'objet des chapitres qui vont se succéder.

➤ **Indexer le programme cible.**

Indexer signifie : Montrer du doigt. Instinctivement, quand on désire désigner une direction à notre interlocuteur, on utilise le doigt le plus long de notre main, c'est à dire l'index. Le vocable "indexer" en dérive. Dans notre cas il signifie que dans les neuf candidats "nominés", on devra systématiquement préciser au logiciel lequel on veut effacer, lister, modifier etc. Que l'opérateur soit en permanence informé de la cible est vital. Aussi, quand on consulte les pages du menu des **DONNEES**, l'emplacement réservé aux ACR dans les autres menus indiquera quel est actuellement le programme indexé par la sonde. Par exemple sur la Fig.289 en **X** on note qu'actuellement la sonde pointe le programme n°3 dans la liste,

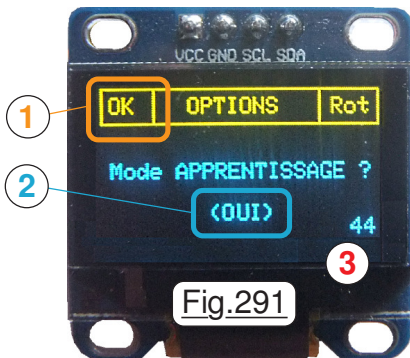


c'est lui qui sera l'objet des traitements relatifs à l'apprentissage. C'est dans le menu des **OPTIONS** que l'on trouve l'item de la Fig.290 qui si on le valide avec **OUI** affiche dans le cadre **Y** le n° du programme indexé. Lorsque l'on fait tourner le **BPCCR** dans un sens ou dans l'autre, chaque nombre est incrémenté ou décrémenté, envoyé à la sonde puis **P : n** s'affiche en **Y**. L'information présente dans le cadre jaune concerne celle qui est effectivement indexée par le programme esclave. Quand on arrive aux valeurs extrêmes il y a recirculation à 1 ou à 9. La LED jaune clignote, il faut donc sortir impérativement par **FIN** pour revenir à un écran "propre".

 **NOTE :** Dans la version actuelle, la LED rouge de SÉCURITÉ n'est pas systématiquement effacée quand une action est effectuée. Ce sera le cas dans la version ultime du logiciel.

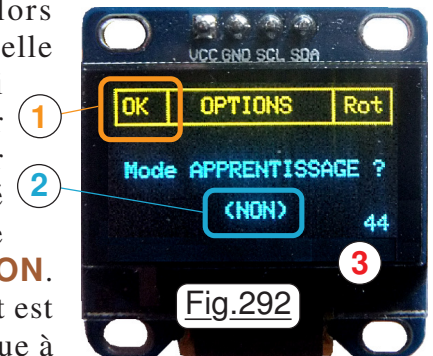
➤ **Activer ou suspendre le mode APPRENTISSAGE.**

Particulier par certains des effets qu'il implique, engager ce mode ne doit pas risquer de se faire inopinément. C'est la raison pour laquelle il faudra au préalable allumer la LED rouge de SÉCURITÉ. C'est encore dans le menu des **OPTIONS** que l'on trouve la commande. Tenter d'activer l'apprentissage alors que la LED rouge est éteinte retournera l'accusé de réception **E12** qui va souvent venir nous titiller. Sécurité armée, la touche **OUI** peut générer une erreur **E13** qui signale



que l'on veut tenter un apprentissage sur un programme non effacé. Ce n'est que provisoire, car vous pouvez constater que dans la version ultime l'erreur n°13 n'existe plus, nous verrons pourquoi plus avant. Pour l'heure il suffit d'indexer l'un des programmes, de l'effacer avec le code **88**. Suite à ces deux manipulations il sera alors possible d'activer l'apprentissage. (Si la sécurité est armée !) La sonde retourne l'accusé de réception **1**. Comme l'ordre n'a pas posé de problème, le programme maître transmet l'ordre **44** comme montré en **3**. Le programme esclave renvoie alors l'état des booléens de la sonde, information qui décortiquée est alors

visualisée en **2**. Si les LED de servitude sur la sonde sont actives, celle qui en rouge témoigne du mode apprentissage est alors allumée. Si on tente une deuxième ouverture, le code erreur **E11** signale qu'activer l'apprentissage une deuxième fois n'est pas valide. Comme pour l'erreur **E12** ce n'est qu'un comportement provisoire qui sera modifié sur les programmes "définitifs". Pour mettre fin au mode apprentissage, il faut revenir sur l'item de la Fig.291 et frapper sur **NON**. La sécurité n'est pas du tout exigée car l'action est neutre. Le résultat est montré sur la Fig.292 sur laquelle on retrouve une procédure analogue à celle de l'ouverture du mode, le résultat final en **2** étant cette fois négatif.



➤ **Enregistrement des instructions dans un programme.**

Naturellement, vous avez compris que chaque commande envoyée à la sonde verra son code s'inscrire en EEPROM à partir de la première cellule du bloc concerné, puis les unes à la suites des autres viendront encombrer la zone réservée au programme actuellement indexé. Ce naturel admet toutefois des limites, car il faut parer un certain nombre de problèmes. Le premier qui vient à l'esprit, c'est la saturation des trentes cellules. L'opérateur portant son attention sur les instructions qu'il impose, et sur les conséquences possibles n'a pas le loisir de compter. Aussi, quand toutes les cellules sont remplies, la sonde n'inscrira plus rien dans l'EEPROM et retournera le code d'erreur **E14**. L'attention étant attirée par le BIP sonore, l'opérateur sera alors averti et pourra prendre les mesures adéquates. La deuxième pierre d'achoppement réside dans certaines fonctions qui bloqueraient l'exécution du programme enregistré. La difficulté réside dans l'alternat qui sera engagé entre les deux machines. Ce problème sera abordé dans le chapitre d'utilisation du bloc enregistré. Consultez les pages **Codage des consignes** 1/3 à 3/3 du **MANUEL d'UTILISATION** qui donnent l'intégralité des codes possibles et précise ceux qui ne seront pas valides pour un enregistrement.

NOTE : Actuellement il n'est pas possible de modifier un programme quand on a quitté le mode apprentissage. Pour effacer le dernier code par exemple, il faut être en apprentissage. Hors on ne peut ouvrir le mode que sur un programme effacé. Naturellement ce problème sera corrigé dans les versions ultérieures du logiciel. Il sera possible de modifier strictement librement un programme pour pouvoir l'éditer à convenance. Actuellement tenter d'ouvrir le mode apprentissage engendre une alerte **E13** qui ne sera plus utilisée sur la version ultime comme vous pouvez le constater sur la page **Codes des erreurs retournés par la sonde** du manuel d'utilisation.

➤ Effacer un programme.

Comme vu dans le chapitre précédent, pour pouvoir ouvrir le mode apprentissage il faut dans la version actuelle que le programme indexé soit vide. Aussi, on se doute qu'une commande spécifique permet d'effacer à convenance le programme de notre choix, ce qui suppose au préalable de l'indexer. Présente actuellement dans le menu **EXPLOITER**, la commande présentée sur la Fig.293 impose d'avoir allumé la LED **SÉCURITÉ** pour pouvoir valider l'effacement du programme actuellement indexé. Dans le cas contraire une erreur **E12** déclenchera un bip sonore. Le dernier code transmit à la sonde est **44**, le plus souvent, c'est celui qui achève une commande pour en vérifier les effets réels sur la sonde.



➤ Corriger un programme.

Actuellement n'est possible que durant une ouverture du mode apprentissage. Quand on ferme la session la seule modification possible reste l'effacement total pour recommencer. Déjà signalé, cette limite sera levée dans les versions futures. "Corriger" est à prendre dans ces lignes avec un sens très restrictif. Comme le précise la commande montrée sur la Fig.294 la correction consiste à effacer le dernier code de la liste. Tenter la manipulation hors mode apprentissage vous gratifiera d'une erreur **E16** signalant l'obligation d'avoir la LED apprentissage allumée. Vous savez qu'en version opérationnelle il sera facile d'ouvrir le mode y compris sur un programme dont le listage n'est pas vide. Sécurité activée et mode apprentissage ouvert, **OUI** effacera le dernier code du programme actuellement indexé. On peut réitérer cette commande à notre guise, les effacements "remontant" vers le début. Les codes **89** se soldent par un **OK** tant que c'est possible. Quand il ne reste plus rien à enlever, **E17** nous précise que ce n'est plus la peine d'insister ... circulez ya rien à voir ! Notez que cette commande n'est logique que dans la mesure où quelques codes sont à enlever en vue de corriger un listage. Si c'est pour tout effacer autant faire appel à la commande de la Fig.293 qui élimine la totalité d'un coup.



➤ Éditer un programme.

Également restreinte, cette commande constitue la réciproque de la précédente. Quand on active ce mode, on se retrouve dans l'état qui correspond à l'ouverture d'un apprentissage, sauf que cette fois on ajoute des instructions à la suite de celles qui sont déjà enregistrées en EEPROM. Sécurité active et mode apprentissage ouvert, la raquette de commande se contente d'envoyer l'ordre **20** et de faire positionner sur la sonde le pointeur correctement en EEPROM à la suite du dernier code inscrit pour le programme indexé. En version ultime du logiciel cette commande sera disponible inconditionnellement et servira à ouvrir le mode apprentissage. (Voir la Fig.295) On pourra ainsi éditer directement à partir d'un programme vide, ou le compléter à convenance.



La Fig.296 résume les deux méthodes qui nous octroient la possibilité de modifier le contenu d'un programme. En **A** on procède avec **Suppr. dernier code**. Chaque validation avec **OUI** décale "vers la gauche" l'emplacement de la dernière "case disponible" dans le bloc des trente codes possibles. La valeur de **Nb** est décrémentée conjointement avec le pointeur d'écriture en EEPROM.

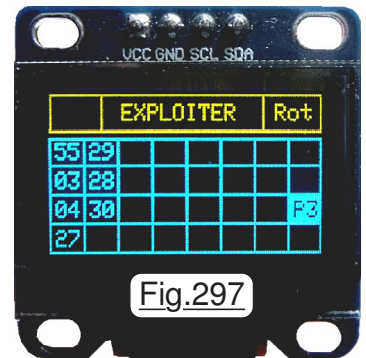
Corriger le programme indexé. Fig.296



En **B** la fonction **Editer le PGM n** est active. Chaque commande valide est inscrite en EEPROM à l'emplacement du pointeur qui est alors incrémenté conjointement avec la valeur **Nb**.

➤ **Lister le contenu du programme actuel.**

Fonction absolument indispensable, que ce soit en cours d'écriture du programme ou avant d'en déclencher l'exécution, il importe de pouvoir à tout moment en consulter le contenu. La présentation à l'écran est actuellement acquise puisqu'avec le module **Test_AFF_PGM.ino** nous avons effectué les tests préliminaires pour organiser au mieux les informations présentes à l'écran. Du coup, la Fig.297 présente un aspect de "déjà vu" et n'appelle que peu de commentaires. Il est évident que sur cet exemple c'est le programme n°3 qui est indexé, donc listé. Ce dernier comporte actuellement sept instructions, la dernière consistant à éteindre le LASER. Il est clair que pour passer en revue un listage, nous aurons besoin d'avoir sous la main la correspondance des divers codes, dont l'intégralité figure dans les pages **Codage des consignes** 1/3 à 3/3 du **MANUEL d'UTILISATION** ouvert à la demande. Dans la grille de la Fig.297 nous ne devrions jamais voir figurer les codes non valides repérés par les symboles '/'. Si c'est le cas il faudra impérativement analyser le problème potentiel.



➤ **Déplacer le contenu d'un programme.**

Plusieurs raisons peuvent nous amener à déplacer un programme. Neuf zones possibles constituent une manne qui sera probablement largement mise à contribution. Chaque utilisateur va se concocter des programmes "meumeu" que l'on sera ravi de déclencher pour faire une petite présentation de JEKERT à des amis. Aussi, au cours du temps certains vont ressortir comme méritant une résidence pérenne en EEPROM alors que d'autres moins ludiques seront effacés. On aura probablement envie de placer à la fin les incontournables, et mettre au début les expérimentations nouvelles. Enfin, en version ultime il sera possible de chaîner en automatique plusieurs programmes. On pourra de ce fait réaliser des modules qui une fois mis au point devront se succéder en EEPROM. Cinq étapes seront nécessaires pour déplacer un programme d'une zone à une autre :



- Écrire dans le bloc qui servira de source. (*Voui voui voui, on s'en doutait un tantinet !*)
- Préciser à la sonde quel est le bloc ciblé pour effectuer un transfert. La commande sur la raquette est visualisée sur la Fig.298 qui n'impose pas la **SÉCURITÉ**, car ne présente pas de risque particulier.
- **Indexer** le programme qui servira de réception du code **SOURCE** transféré.
- Effacer la zone de réception. En effet, pour ne pas courir le risque d'écraser un module, (*C'est la terminologie consacrée en informatique.*) le transfert ne peut se faire que sur une zone vide. Comme l'effacement impose d'armer la **SÉCURITÉ**, on ne perdra pas un listage par erreur ... tout au moins en principe mais alors c'est vraiment que vous n'aurez pas fait bien attention !
- Transférer le programme source dans la zone de réception supposée préalablement vidée, ou vous serez "punitonnés" par un **E24**. Transférer un programme sur lui même est illégal, car par définition le transfert se termine par l'effacement de la zone **SOURCE** pour qu'elle soit disponible à l'édition. De ce fait, un transfert gigogne effacerait le programme source. (*Domage !*) Pour éviter cet aléa une telle tentative se solde par une erreur **E23**. Pour procéder à un transfert, il faut que le mode apprentissage soit fermé. Comme montré sur la Fig.299 si ce n'est pas le cas une erreur **E22** sera générée. Enfin, vous vous en doutez probablement, la **SÉCURITÉ** sera armée pour que le **OUI** ne se solde pas par **E12** dont on finit par bien cerner la signification. Quand toutes les conditions sont respectées, enfin le code **13** se solde par le **OK** tant attendu. Certains vont trouver que la procédure complète est un peu compliquée. Concrètement, elle est bien plus facile à pratiquer qu'à décrire. Seule lourdeur, le fait d'avoir à quitter **EXPLOITER** pour armer **SÉCURITÉ**, puis y revenir et avoir à naviguer dans les nombreux items pour retrouver la commande souhaitée. Ce n'est que provisoire, car en version ultime toutes les commandes du mode apprentissage seront regroupées dans un menu **APPRENDRE** indépendant.

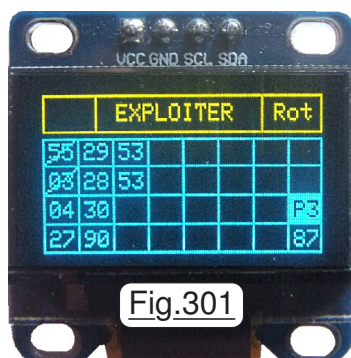


➤ Exécuter le programme indexé.

Raison d'être de toutes les commandes d'apprentissage qui ont occupé les chapitres précédents, déclencher l'exécution d'un programme enregistré n'a rien d'anodin. La logique de base semble simple et se résume à une boucle du genre : *"tant que l'on n'a pas terminé, on analyse le code pointé, on réalise la fonction qui le concerne, on passe au code suivant"*. Sous cette forme, l'opération globale semble assez simple. La difficulté consiste à continuer le dialogue entre la sonde et le pupitre, hors perdre la synchronisation est particulièrement risqué dans une telle séquence. C'est du reste pour cette raison que certains codes ne sont pas autorisés, le chapitre suivant va décrire la boucle et son fonctionnement. Si tout se passe bien, quand la raquette aura donné l'ordre **87**, le programme esclave va dérouler les instructions. Quand toutes auront été réalisées, il retournera le **OK** qui sur le pupitre rendra la main à l'opérateur. Plusieurs problèmes peuvent se présenter. Vous avez compris que pour JEKERT cette fonction est l'une des plus dangereuses. Par exemple vous avez aligné un nombre exagéré de mouvements en avant, puis sans trop réfléchir déclenché le processus :

- Et Totoche, ya plus la sonde, c'est normal ?
- Ben oui Dudule, regardes en bas du ravin !
- Et Totoche, la caméra montre le ciel de Mars, c'est prévu ça ?
- Ben non Dudule, mais sur la droite il y avait un obstacle et JEKERT a basculé sur son flanc ...

D'accord, pour déclencher l'exécution du programme indexé il sera sage de bien analyser le listage pour valider la procédure. Ensuite, on devra impérativement armer **SÉCURITÉ** pour ne pas croiser l'inévitable **E12**. Imaginez également ce qui se passerait si la fonction était invoquée alors que le mode apprentissage est en cours. Il y aurait de l'auto-apprentissage en boucle avec à la clef une saturation à trente, suivie de plus rien ... car le dialogue et la synchronisation seraient bloqués. Aussi pour vous éviter cette fin tragique de la mission, la Fig.300 montre que tout au plus un **E19**



pénalisera notre amour propre, on s'en remettra plus rapidement que les scientifiques qui ont investi dans JEKERT tous leurs espoirs. Par ailleurs, si vous tentez de déclencher un programme vide, **E18** se rappellera à votre bon souvenir. Si tout va bien, **OUI** sera accepté par le logiciel esclave qui affichera la grille, listera les codes du programme pointé du doigt, ainsi que dans la case à droite en bas visualisera le **87**. À partir d'ici, le pupitre n'est plus opérationnel, inutile de titiller les touches ou le codeur rotatif. Sur l'écran OLED, un dialogue de type alternat étant établi entre pupitre et sonde, chaque fois qu'une instruction est achevée on voit, comme sur la Fig.301, la cellule la concernant être biffée. Ainsi

nous savons exactement ce qui se passe. (*Frimeur !*) Puis, si toutes les instructions sont réalisables, un **OK** final achèvera la séquence, le programme maître rendant alors la main à l'opérateur. Outre l'accusé de réception qui signale que le processus a été jusqu'à son terme, la LED **SÉCURITÉ** s'éteint sur le pupitre. L'écran reste affiché jusqu'à ce que l'on clique sur l'un des boutons permanent de menu ou que l'on tourne le codeur incrémental. Reste qu'il n'est jamais exclus que l'une des instructions soit irréalisable. Par exemple le code **3** est puisé dans le listage pour faire avancer la petite machine d'un pas. Si un obstacle se trouve à moins de 8cm, le programme esclave retournera le code d'erreur **E20**, c'est à l'opérateur de déterminer la cause du problème rencontré. Ce ne sera probablement pas trop délicat d'en diagnostiquer l'origine, car l'écran reste figé et la case de l'instruction qui a engendré une impossibilité n'est pas cochée. Avec dix messages d'erreur typés et spécifiquement réservés pour le mode apprentissage, on peut dire que le logiciel pare déjà pas mal de cas illogiques. Par exemple, et j'ai oublié de le mentionner sur le chapitre du listage d'un programme indexé, si vous tentez la fonction sur un module vide, **E15** vous informe pourquoi, comme visible sur la Fig.302, les cases de la grille réservées aux codes des instructions restent toutes sans contenu. Nous allons pouvoir passer au démonstrateur suivant, mais avant on va analyser un petit bout de logiciel.



➤ Exécuter un programme et simultanément continuer le dialogue.

Incontournable, les deux machines sont intimement liées par la nécessité de maintenir un alternat permanent au cours du déroulement d'un programme EEPROM. En effet, le choix d'afficher sur l'écran l'état du système en cochant "la grille du LOTO" implique forcément qu'entre la sonde et le pupitre on conserve l'échange des informations. Le dialogue ressemble à ceci :

- L'esclave est à l'écoute et attend sagement une consigne envoyée par le Maître,
 - Le Maître donne son ordre **87** puis immédiatement repasse à l'écoute,
 - L'esclave décode **87**, vérifie que le programme existe et répond **OK** puis immédiatement écoute,
 - Le Maître parle et dit : Continue, puis repasse à l'écoute,
- À ce stade l'esclave entame la boucle pour épuiser toutes les instructions :
- L'esclave capte l'instruction suivante, la réalise, dit "*tu peux cocher*" et poursuit **en aveugle**,
 - Le Maître coche la case suivante dans la grille et immédiatement repasse à l'écoute,
 - L'esclave vérifie s'il reste encore une instruction. Si OUI
 - Si NON l'esclave retourne **OK** qui signale que c'est terminé puis attend une nouvelle commande,
 - Le Maître éteint la LED **SÉCURITÉ** puis attend une touche au clavier ou un mouvement du **BP_{CCR}**.

"*tu peux cocher*" n'est pas un code comme **87**, car tous ont été consommés. Aussi, dans le but de simplifier au maximum le logiciel, l'esclave n'attend que **OK** ou **E20** pour détecter la fin du processus. Tous les autres ACR sont vides et ne comportent que la sentinelle '*', signalant de ce fait "*on continue, je viens de traiter une instruction*".

ATTENTION, dans ce processus il n'y a plus alternat ou chacun attend que l'autre ait parlé pour intervenir à son tour. **En aveugle** veut dire que l'Esclave ne se préoccupe plus de savoir si le Maître est à l'écoute. Il continue son travail inexorablement. **Cette technique simple ne peut fonctionner QUE SI LE MAÎTRE EST REPASSÉ À L'ÉCOUTE AVANT QUE L'ESCLAVE NE PARLE.** C'est ici le cas, car pour cocher une case dans la grille il faut bien moins de temps que pour récupérer le code en EEPROM, pointer le code suivant, exécuter l'instruction, vérifier qu'il n'y a pas un **E20** et envoyer l'ACR. Remarquez au passage que tout problème survenu en exécution retourne un code unique **E20**. C'est une facilité informatique, car sur le Maître il n'y a besoin de tester que les deux cas possibles pour détecter la fin du processus et informer l'opérateur. (*OK ou E20.*)

Sont interdits par '/' en apprentissage tous les codes qui retournent en ACR autre chose que **OK**, car il y aurait perte de la synchronisation et blocage des deux microcontrôleurs.

```
void Activer_le_PGM_cible() {  
  ① Saisie_NB_Codes_dans_le_PGM();  
  ② if (Compteur_de_Code_PGM == 0) num_ERR = 18;  
  ③ else {num_ERR = 0;  
  ④   while (Compteur_de_Code_PGM > 0) {  
  ⑤     PTR_Apprentissage++; // Passer au code suivant.  
  ⑥     Code_Consigne = Lire_un_OCTET_en_EEPROM(PTR_Apprentissage);  
  ⑦     delay(200); AFF_etoile(); // Pour barrer le code sur la grille de listage.  
  ⑧     Traite_la_consigne();  
  ⑨     if (num_ERR == 0) Compteur_de_Code_PGM--;  
  ⑩     else {num_ERR = 20; Compteur_de_Code_PGM = 0;}}}
```

La procédure **Activer_le_PGM_cible()** qui traite la directive **87**, est listée ci-dessus. En ligne ①, non seulement la variable **Compteur_de_Code_PGM** reçoit la valeur du nombre d'instructions dans le programme, mais également positionne le pointeur de codes **PTR_Apprentissage** sur "début du PGM - 1". Soit en ② on constate que le programme est vide, le processus se termine avec un ACR **E18**, soit en ③ on poursuit en précisant que pour le moment tout va bien. Tant qu'en ④ on n'a pas terminé, en ⑤ on pointe la prochaine instruction. Puis en ⑥ on capture son code en EEPROM. Avant d'envoyer la sentinelle '*' en ⑦ on commence par un **delay** d'attente de sécurité de 0,2 seconde. Puis, en ⑧ on réalise l'instruction comme si elle avait été instanciée par le pupitre. Si le traitement se déroule normalement, **num_ERR** est nul et en ⑨ on décompte le nombre d'instructions qui restent à traiter. Si une erreur quelconque a entravé l'exécution de l'instruction, le code **20** est généré pour se voir envoyé en ACR et simultanément **Compteur_de_Code_PGM** est forcé à zéro précisant à la boucle **while** qu'elle est achevée et qu'il faut en sortir. Fastoche tout ça non ?

57) 24/01/2018 : On améliore le logiciel d'exploitation (MJD 58142)

Quand on arrive en salle informatique, les programmeurs de S4 ne sont pas au travail. Une table a été dressée rapidement dans le coin ouvert ou il reste encore un peu de place. Des boissons et des grignoteries sont dispersées sur cette dernière. C'est visiblement la fête.

- Jour les binaires, que fêtez-vous ce matin, c'est pas RTT au planning ?

À peine avons-nous interpellé les personnels, que le Directeur en personne se détache du groupe et vient nous serrer la main et avec un franc sourire et répond :

- Presque cinq jours d'avance sur les prévisions, alors j'ai pensé qu'on pouvait se faire une petite récré tous ensemble. Je suis d'autant plus ravi que je n'ai presque jamais le plaisir de venir parmi vous tous. C'est pas mal, les consoles commencent à avoir de la gueule.

Effectivement, embarqué dans le tourbillon des journées qui se suivent et qui se ressemblent, on enlève carton coloré par carton coloré sans forcément vérifier si l'on est en avance ou en retard sur les prévisions, focalisant sur ce qui reste à faire. Joyeux, Ferrando m'apporte un verre.

- Et oui chef, vaut mieux être un peu en avance sur le tableau que beaucoup en retard !

➤ Les petits "plus" de la version K des deux programmes.

Quelques textes seront encore un peu brouillés, car ce démonstrateur de la raquette était associé à une version ancienne de la répartition des textes en EEPROM. Presque tous sont maintenant en mémoire non volatile et utilisent `P35_Ecrire_les_textes_en_EEPROM.ino` pour les y logger. On télécharge `P21K_Démonstrateur_Raquette.ino` et `P22K_Démonstrateur_Sonde.ino` qui apportent quelques brouilles, et surtout l'affichage en continu des données de navigation.

Sans que ce ne soit une révolution, à partir de cette version le bouton central en mode **OPTIONS** inverse l'état de la LED **SÉCURITÉ**. C'est bien plus efficient que de se contenter de l'allumer dans le démonstrateur précédent et d'avoir à naviguer dans un menu pour trouver l'item correspondant. (On peut l'éteindre à convenance si **SÉCURITÉ** est allumé par erreur ...)

L'affichage en continu des données de navigation est incompatible avec la stabilisation gyroscopique automatique. Aussi, vu que cette fonction génère des problèmes dans divers cas d'utilisation de la sonde, une LED verte dédiée à "Gyro stabilisation active / coupée" est ajoutée et pilotée par l'entrée analogique **A1** qui ici est initialisée en sortie binaire.

➤ Affichage en continu des données de navigation.

Bénéficiant de l'efficacité impressionnante du circuit MPU-6050, afficher en permanence les valeurs de Tangage, de Roulis et d'orientation en Lacet est particulièrement séduisant. Il importe également d'indiquer l'écart de route, ce qui implique d'afficher simultanément la référence, valeur que l'on a imposé lorsque l'on a utilisé l'item **Calage gyro.** du menu **EXPLOITER**. Toutes ces entités doivent se voir présentées simultanément à l'écran, il faut donc déterminer leur répartition sur la surface réduite de la matrice graphique du module OLED. Représentée sur la Fig.303 la pagination répartit les diverses données dans la zone bleue de la matrice de pixels. La fonction d'affichage en continu des données de navigation s'obtient par un item disponible dans le menu des **OPTIONS**. Quand on valide l'item avec **OUI** le texte jaune est changé et remplacé par celui repéré en 1. La LED jaune clignote. Il faudra impérativement quitter ce mode par utilisation de la touche **FIN** et surtout pas avec une autre touche de menu ou par usage du **BPccr**. En 3 et 4 sont indiquées les angles d'inclinaison, alors qu'en 2 s'affiche la valeur du CAP magnétique actuel. En 5 est indiquée la valeur du calage gyroscopique, c'est en réalité une référence interne qui n'est pas liée au cap magnétique. Du coup, en 6 la valeur de l'écart de route semble complètement fausse, car on a tendance à comparer la référence interne au cap magnétique. L'**Ecart de route** représente la rotation mesurée par le gyroscope de Lacet entre le moment où il a été recalé et l'instant présent. Sur la Fig.303 la sonde est restée immobile, posée sagement sur les sabots du bouclier. On en déduit que l'**Ecart de route** devrait être nul puisque la petite machine n'a pas bougé. D'où viennent ces -17° d'écart, valeur qui lentement augmente en grandeur. La centrale gyroscopique surchauffe ? Le MPU-6050 nous fait une déprime ? Démystifions cette étrangeté malicieuse.



La dérive gyroscopique.

Toutes celles et tous ceux qui ont eu le privilège de piloter un avion muni d'un conservateur de cap gyroscopique connaissent ce phénomène. De quoi s'agit-il ?

Si vous utilisez une boussole magnétique, cette dernière s'oriente par rapport au champ magnétique terrestre. Posez cette dernière sur une table. Elle indique une direction. Revenez plusieurs heures ou plusieurs jours plus tard. Si elle n'a pas été déplacée, elle indique toujours la même direction.

- Normal direz-vous, elle s'oriente par rapport aux lignes du champ magnétique local qui est fixe par rapport au géoïde. Si la boussole n'est pas réorientée, elle indiquera immuablement un CAP magnétique invariable.

Laissez JEKERT bien tranquille, pénarde en hibernation par exemple. À bord les systèmes de navigation, les dispositifs expérimentaux restent opérationnels. Par les télémesures on continue à recevoir les valeurs météorologiques, la valeur du CAP magnétique ainsi que les données gyroscopiques de la centrale inertielle. Tangage et Roulis ne varient pas au cours du temps. Mais la valeur **Ecart route** évolue en permanence dans le sens négatif, comme si on tournait en LACET. Cette variation augmente régulièrement d'environ 15° par heure. Curieux non ?

L'explication réside dans la nature même des capteurs gyroscopiques. Par conception ils conservent "une orientation par rapport à l'Univers", ou si vous préférez par rapport aux belles étoiles qui illuminent merveilleusement nos cieux nocturnes.

-Et alors ?

-Ben ... certains affirment avec conviction que la Terre tourne.

Elle entraîne dans ce tourbillon circadien tout ce qui est immobile par rapport au sol. Du coup, bien qu'immobile sur le bureau, par rapport aux étoiles JEKERT effectue un tour en 24H, soit 15° par heure. C'est cette rotation à laquelle nous ne sommes pas sensibles qui inexorablement est enregistrée par les dispositifs inertiels gyroscopiques.

CONCLUSION : Contrairement au GPS qui calcule votre déplacement par enregistrement des points successifs occupés sur votre trajectoire, un conservateur de CAP gyroscopique sera inexorablement influencé par la rotation de l'astre sur lequel on se trouve **et que l'on désire prendre en référence pour indiquer une direction**. C'est la raison pour laquelle sur les petits aéronefs le "GYRO" est toujours complété par un compas magnétique qui permet de recalculer régulièrement le conservateur de CAP pour tenir compte de ce phénomène. Le premier qui a mis ce phénomène en évidence est Foucaud, savant illustre et présent dans tous les ouvrages de physique par son expérience célèbre sur le pendule qui porte son nom.

La centrale inertielle MPU-6050 intégrée à bord de la petite sonde est réellement un dispositif gyroscopique. Il est donc influencé par la rotation de la Terre. Avec 360° en 24 heures, on doit s'attendre à une dérive gyroscopique de 15° par heure ce qui n'a vraiment rien de négligeable. N'oubliez-donc jamais ce phénomène. Il vous arrivera de faire des manipulations qui peuvent durer facilement trente minutes à une heure. Le CAP magnétique continuera à indiquer une direction crédible. En revanche, l'écart de route pourra vous interpellier. Par exemple vous remplacez la Sonde plein Nord, direction pour laquelle vous aviez procédé au calage gyroscopique. Vous vous attendez à un **Ecart route** nul puisque l'orientation de calage est retrouvée.

- Mazette, la télémesure indique -43°, vas pas bien l'truc !

- Ben si Dudule, fonctionne bien la centrale, mais pendant que t'as siroté ton Pastis la Terre a tourné. Le GYRO c'est comme l'heure, son "aiguille" tourne en permanence.

La course du Soleil dans notre ciel correspond exactement à la réciproque. Ce n'est pas lui qui se déplace, mais la Terre qui change d'orientation par rapport à l'Univers. Comme le Soleil fait partie intégrante de l'Univers, on le voit tourner, comme pour toutes les autres étoiles. Il devient possible de déterminer l'heure en observant sa position, (*Les cadrans solaires pour ne pas les nommer ...*) De façon analogue, il serait possible d'utiliser le module MPU-6050 comme chronomètre en utilisant l'évolution du Lacet. Cette possibilité impliquerait on s'en doute de maintenir JEKERT strictement immobile durant le chronométrage. C'est une hypothèse amusante pour illustrer le propos, mais scientifiquement ajouter un chronomètre informatique s'imposerait.

voir <https://www.lavionnaire.fr/InstVolDirect.php>

58) 29/01/2018 : Changement de la carte des menus (MJD 58147)

Planning remis en cause, les informaticiens sont sollicités par les personnes qui ont testé les consoles de maîtrise de la mission. Globalement la satisfaction domine. Cependant il ressort de la concertation, que manifestement le menu **EXPLOITER** est bien trop encombré. Plusieurs opérateurs ont émis le souhait de prévoir un chapitre particulier pour y regrouper tout ce qui concerne l'apprentissage. Le cahier des charges fonctionnel a donc été revu, il comporte un nouveau chapitre **APPRENDRE** dont la touche d'appel sera permanente comme pour les autres menus. À l'unanimité les ingénieurs en charge de JELERT ont estimé que la fonction "Libérer les efforts" pouvait fort bien se voir déclenchée par le **BPccr** quand le menu des **POSTURES** est actif. C'est très facile d'accès et rapide, cette idée rend tout à fait commode cette fonction motrice souvent sollicitée.

➤ Les interruptions : Le retour.

Infinitement plus agréable à l'usage, la prise en compte de la rotation du codeur incrémental par interruptions est réintroduite à partir de cette version et ne sera plus remise en cause. Plus logique qu'un traitement par sondage dans la boucle principale du programme, elle présente de nombreux avantages. Par exemple si on tourne le codeur pendant que le programme n'est pas disponible et attend un accusé de réception sur la ligne **RX**, l'action de l'opérateur est détectée et mémorisée. Dès que l'esclave rend la main à la raquette, l'action de l'opérateur est alors prise en compte. Autre avantage, cette technique fait prendre en compte deux pas sur le codeur. C'est à dire que si on tourne rapidement le bouton, on voit que deux items seront affichés successivement à l'écran. Nous avons abandonné cette technique, car les composants "bas de gamme" approvisionnés présentaient trop d'aléas de fonctionnement. Ayant craqué pour un élément à 4.35• port non compris, plus couteux que ceux vendus par cinq, cette fois la qualité au rendez-vous. Le pupitre de JELERT méritait cet effort financier, son utilisation étant incomparablement plus agréable.

➤ Une LED rouge de plus sur l'arbre de Noël.

Saturer toutes les broches de l'ATmega328 n'est pas un but en soi. Ceci dit, il restait encore une broche de disponible avec **A0**. L'expérience montre que le mode apprentissage étant engagé, mis à part le fait que les instructions valides sont inscrites en EEPROM, il ne se passe rien de visuel. Trop discret, surtout si sur JELERT le mode "économie d'énergie" est validé et que les LEDs sont coupées, l'opérateur oublie que la machine espionne et enregistre les dialogues avec le pupitre. Aussi, une LED rouge dédiée est ajoutée à l'ensemble et pilotée par l'entrée analogique **A0** qui pour la circonstance est initialisée en sortie binaire.

Quand on commence un gros projet tel que celui-ci, il est totalement impossible, sauf cas rarissimes, d'avoir présent à l'esprit l'intégralité des ressources qui seront branchées sur les broches du microcontrôleur. En général, les grandes lignes ont défini les périphériques principaux. On peut commencer à affecter des broches pour les piloter. **IL IMPORTE quand c'est possible d'UTILISER EN PRIORITÉ LES BROCHES LES MOINS SOUPLES** et laisser pour la suite les plus riches en fonctionnalités. Par exemple pour les sorties binaires éviter sauf nécessité d'employer au début celles qui peuvent fonctionner en PWM. Pour les entrées analogiques, préserver au maximum celles qui sont configurables également en sortie. On voit dans ce projet que c'est tout à la fin qu'un besoin de sorties binaires s'est fait sentir. Comme on avait préservé les broches analogiques pouvant fonctionner en sorties, l'affectation a été immédiate et n'a pas obligé à reprendre le logiciel pour réaffecter d'autres broches.

Refermons cette parenthèse, il est grand temps de concrétiser. Nous changeons de version en téléversant **P21L_Démonstrateur_Raquette.ino** et **P22L_Démonstrateur_Sonde.ino** sur les deux cartes NANO Arduino. Cette version est encore relative à une ancienne répartition des textes en EEPROM. Aussi des petites verrues du genre **JEKERTShiberne** viendront "crabouiller" certains écrans ... la routine. Nous savons que sur la version ultime qui n'est plus très éloignée, l'écriture sera soignée. Enfin, tournez le codeur incrémental. Si votre composant génère un "sens inversé", permutez les broches **CLK** et **DT**, nous sommes parés pour tester cette nouvelle mouture qui apporte des améliorations opérationnelles significatives.

➤ **Faire le ménage dans le logiciel version L de la raquette de commande.**

Conformément aux nouvelles clauses du cahier des charges fonctionnel, la touche **S10** du clavier devient celle d'appel du menu **APPRENDRE** et sera disponible quel que soit le menu actif sur le pupitre de commande. Réserver un menu allège considérablement le travail de l'opérateur car le menu **EXPLOITER** était trop encombré. Par ailleurs, regrouper toutes les commandes du même type va forcément dans le sens de la simplification. Lorsque le menu **APPRENDRE** est actif, le **BPccr** sert à activer ou à éteindre la LED **SÉCURITÉ** évitant les anciennes navigations entre les menus chaque fois qu'une erreur **E12** était affichée dans le petit cadre des accusés ACR, obligeant ensuite à balayer à nouveau les items pour retrouver la fonction souhaitée. Au point de vue qualité opérationnelle, le nouveau programme fait un bon en avant considérable. Faisons le ménage dans le logiciel :

- On peut activer le menu **APPRENDRE** alors que l'apprentissage est déjà ouvert sur JEKERT.
- L'activation du mode Apprentissage sur JEKERT n'impose pas la sécurité car on ne fait qu'ajouter du code à celui qui existe, (*Sauf si erreur n°14.*) donc pas de risque d'effacer par erreur.
- L'item EDITION n'est plus utile puisque on peut activer le mode enregistrement à tout moment et ce dernier fait exactement pareil que l'ancienne fonction **Editer le PGM**.
- On peut enregistrer un programme alors que JEKERT est en hibernation ce qui sur le plan de la théorie est séduisant. L'apprentissage se fait au minimum de consommation d'énergie.
- Dans cette version, la fonction "LIBÉRER LES EFFORTS" s'obtient avec le bouton central du codeur rotatif quand on est en menu des **POSTURES**.

Le menu **APPRENDRE** ouvre sur la page montrée en Fig.304 sur laquelle on voit dans le cadre jaune le titre qui lui est affecté. Compte tenu du manque de place on a un peu compacté sous forme d'un mnémonique.



Fig.304

➤ **L'apprentissage s'enrichit d'une nouvelle fonctionnalité.**

Faisant peau neuve, la possibilité de chaîner plusieurs programmes chamboule fortement les possibilités opérationnelles. Il sera maintenant possible de réaliser toutes les combinaisons possibles en taille de programme entre 30 et 270 instructions. Un programme étant indexé, quand on va valider son exécution, on aura au préalable initialisé une variable indiquant combien de blocs successifs devront être parcourus. La Fig.305 présente un exemple de ce qu'il est possible d'agencer dans l'éventail de la combinatoire offerte par cette nouvelle option. Dans l'exemple **A** nous avons un premier programme de **43 instructions**, un qui suit de **34 instructions** etc. Dans l'exemple **B** on trouve un premier programme de **56 instructions** suivi d'un gros bloc de 267 codes obtenu par le chaînage de sept modules. La plus grande séquence possible sera forcément de neuf fois 30 codes, soit les 270 annoncés ci-avant. Pour construire un gros programme, on ne peut que procéder par des apprentissages bloc par bloc. Sur les exemples **A** et **B** les blocs qui débutent les séquences sont saturés à 30, ce qui va dans la logique d'une optimisation. Cependant, rien n'interdit

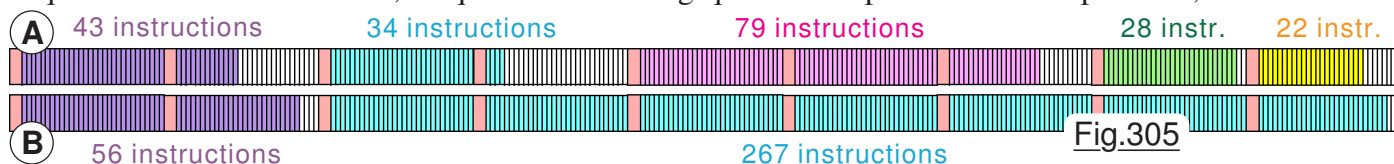


Fig.305

de chaîner des paquets incomplets, l'exécution n'utilisant que les codes enregistrés. C'est dans ce contexte de "panachage" qu'il sera séduisant de pouvoir transférer un paquet à convenance. Ainsi



Fig.306

on va mettre dans l'ordre des séquences déjà au point mais réparties un peu au hasard en EEPROM. Ouvrir le menu **APPRENDRE** puis faire tourner le **BPccr** de trois pas en sens antihoraire fait afficher l'item de la Fig.306 qui validé par **OUI** indique en **1** le nombre mémorisé. (*Par défaut il est initialisé à un.*) La **SÉCURITÉ** doit être activée. La sortie de la saisie qui se fait avec le **BPccr** dans un sens ou son contraire s'achève impérativement avec la touche **FIN**. La LED jaune qui clignote nous évite de l'oublier. Chaque fois qu'un chaînage est déclenché, les **N** blocs vont être déroulés. À la fin, qu'il se produise une erreur ou pas, la valeur du nombre de chaînages est forcée à **1** et **SÉCURITÉ** est éteinte.

Scandale ! Toutes les fonctionnalités souhaitées sont en place, les écrans bavards à convenance. À force d'optimiser, de coller tous les textes en EEPROM, on compacte, on tasse, on optimise pour finalement aboutir à un programme qui ne consomme que 66% de la place disponible dans les neurones de l'ATmega328. **Rentabilité minable, c'est inacceptable !** Il faut absolument trouver un moyen d'augmenter la taille du logiciel de la Raquette. Comme tout produit de luxe, c'est l'emballage qui compte, peu importe le contenu. Pour nous, le flacon c'est l'affichage, aussi on va s'en donner à cœur joie avec les possibilités graphiques. Le pupitre va se voir déplacé dans les étagères des produits de luxe ... non mais alors ! *(Disposer d'une marge considérable de place pour se faire plaisir avec du graphisme est la cerise sur le gâteau. C'est en fait la récompense à tous nos aux efforts permanents consentis pour optimiser octet par octet l'occupation de la zone mémoire. À la base c'était dans le but prioritaire d'avoir la possibilité d'émuler toutes les fonctions désirées. Bénéfice collatéral, en fin de course on constate que nous avons la possibilité de "gaspiller" et on ne va pas s'en priver ...)*

➤ Les modifications de détails.

Commençons par les brouilles, c'est à dire quelques aménagements pour rendre plus agréable l'utilisation du pupitre, ou des détails imposés par l'avènement du mode graphique sur certaines fonctions. Commençons par téléverser **P21M_Démonstrateur_Raquette.ino** dans NANO Arduino de la raquette. Pour cette version du démonstrateur on en reste à **P22L_Démonstrateur_Sonde.ino** pour le programme Esclave. Certains textes sont toujours "srabouillés", tout particulièrement ceux de l'affichage d'un spectre couleur en mode numérique. Peu importe, ce n'est pas le sujet dans ce

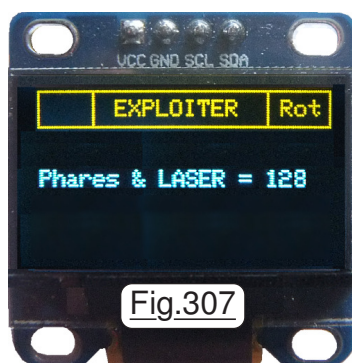


Fig.307

chapitre. Dans cette version en mode **OPTIONS** le BP central du codeur rotatif active ou désactive la **SÉCURITÉ**. Pour activer la stabilisation gyroscopique il faut maintenant armer la sécurité. Pour activer l'affichage gyroscopique permanent il faut également armer la **SÉCURITÉ**. Cette mesure de prévention se généralise en fait à toutes les fonctionnalités potentiellement risquées. Pour finir la liste des petits détails propres à cette version, PHARE et LASER sont forcés à 128 au lieu de 127 pour anticiper l'affichage des graduations en mode ajustement lumineux. *(Voir Fig.307)* Feuilletons maintenant le catalogue en papier glacé de la version de luxe du programme de la raquette pour nous mettre l'eau à la bouche :

➤ Graduations de distance sur un spectre de balayage ultrasons. (Inconditionnel)

Inconditionnel car ce complément graphique est permanent et n'impose aucune initialisation d'une option préalable. Dorénavant, le dessin graphique d'un balayage télémétrique sera complété comme montré sur la Fig.308 par des graduations précisant la distance des mesures visualisées. Le tracé graphique peut "écraser" l'affichage de 1m ou de 2m. Il est peu probable que cette superposition se fasse à la fois en haut et en bas. Aussi, pour ne pas exagérer le coût en octet de ce petit perfectionnement, les écrasements ne sont pas traités. Ce serait possible, par exemple en inversant la luminosité des "Pixels écrasés". Ceci dit, l'augmentation de programme qui en résulterait n'en vaut pas la chandelle, nous aurons bien mieux à proposer plus avant dans le didacticiel.

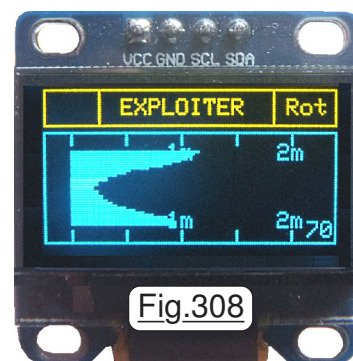


Fig.308

➤ Gradateurs avec ruban analogique. (Inconditionnel)



Fig.309

Commun au gradateur des phares et à celui du LASER, ce type de représentation est également inconditionnel. La Fig.309 est relative au dosage de l'énergie dans les phares. On vient d'ouvrir la page écran sans avoir encore tourné le **BPccr**, la valeur énergétique est donc égale au 128 initialisé avec l'item approprié. *(Commande de la Fig.307)* Pour ne pas que le dosage lumineux soit trop laborieux, la totalité de la plage est couverte en quatorze pas sur le codeur rotatif. Comme on couvre une plage maximale de 255 unité, chaque pas va varier de $255 / 14 \approx 18,2$ unités.

Sachant que le processeur binaire ne peut manipuler que des entiers, le pas de variation d'énergie sera arrondi à 18. Un rapide calcul tel que $255 / 18 \approx 14,17$ permettra 14 marches possibles. (*Normal vu que l'on est parti de ce postulat !*) L'échelle lumineuse comporte donc des graduations pour 14 intervalles. Par exemple, sur la Fig.310 le **BP_{CCR}** a effectué 4 pas dans le sens antihoraire. La valeur initiale était de 128, le petit trait surchargé en rouge situe ce point de départ. On vérifie que $128 - (4 \times 18)$ donne bien comme résultat 56, valeur affichée à l'écran. Si l'on fait encore trois indexations, toujours dans le même sens, $56 - (3 \times 18) = 2$. C'est exactement le cas de la photographie sur la Fig.311 sur laquelle on peut observer un petit détail curieux agrandi dans l'encadré rose. Le ruban lumineux pour la valeur 2 est parfaitement aligné avec la graduation, la verticale étant pointée pas la flèche jaune. Assez étrange, le rectangle qui délimite la largeur du ruban linéaire dépasse, ce que met en évidence la flèche rouge. Erreur



Fig.310



Fig.311

de programmation ou petit détail subtil ? Pour lever le doute, faites un pas de plus dans le sens antihoraire. Pafff, on se coltine un BIP d'alerte avec pour explication **E2**. Faisons un rapide calcul : $2 - 18 = -16$ ce qui n'est pas possible puisque la donnée est un **byte** dont la valeur est toujours positive. Donc le logiciel signale le débordement "par le bas" et bloque la variable à la valeur 1. Si dans cette situation on tourne le "robinet électrique" dans le sens horaire, chaque pas va ajouter 18 à cette valeur initiale de 1. En zone centrale on franchira avec 127 pour aboutir au maximum valide de 253. Allez, au point où nous en sommes, poussons encore d'un cran. Cette fois le BIP est associé à **E1** et la valeur talonne à 254. On a retrouvé les conditions pour lesquelles le franchissement de la zone centrale se fait avec la valeur 128. Dans l'état actuel de développement du programme, quand on quitte la fonction gradateur lumineux, le ruban graphique n'est pas effacé de l'écran laissant un résidu très beaucoup laid vraiment pas beau. Ce n'est pas fondamental, car à ce stade il importe de bien définir les effets visuels. Les petits bugs tel que celui-ci seront facilement filtrés et éliminés par la suite.

➤ Une LED bleue de plus sur le pupitre.

Bien que mentionnée tardivement dans ce tutoriel, il y a longtemps que ce témoin lumineux à été ajouté à l'arsenal optique, et bien avant d'ajouter la LED rouge sur l'entrée analogique **A0**. Les données de navigation sont affichées en valeurs numériques. Dès cette version on va aussi bénéficier d'une option graphique qui transformera l'écran OLED en un minuscule tableau de bord symbolisant les paramètres de navigation. C'est le **BP_{CCR}** lors de l'utilisation du menu des **DONNEES** qui inversera l'état du booléen de l'option graphique. Une LED bleue pilotée par la sortie binaire **D5** représentera visuellement l'état actuel du booléen **Mode_Graphique**. On pourra librement choisir l'une des deux façons de présenter les données à l'écran, bien que le mode graphique soit incomparablement plus agréable. Ceci dit, initialement il était vraisemblable d'imaginer que modifier en temps réel tous ces pixels sur l'écran prendrait un temps rendant l'application incompatible avec une sensation d'immédiateté. Le mode textuel aurait alors pallié cette difficulté. Le programme d'affichage graphique a démontré que la rapidité de rafraîchissement serait suffisante. Tant mieux. Inutile toutefois d'éliminer l'affichage textuel, car il fait partie intégrante du menu des **DONNEES**.

➤ Définir la présentation graphique d'un mini tableau de bord.

Représenter un maximum d'informations sous forme symbolique n'est jamais évident. La surface graphique d'un écran ouvre un éventail infini sur la façon dont on représentera les entités visualisées. Rubans linéaires, cercles, portion de disques, tout est possible, globalement limité par notre imagination. Reste que le dessin informatique est avant tout un brassage intense d'équations. La trigonométrie alliée à la géométrie seront nos muses les plus prolifiques. Avant de chercher à lier le dessin aux réalités de la centrale gyroscopique, une bonne technique consiste à oublier les paramètres de navigation réels et travailler sur une simulation pour étudier ce qui sera le tableau de bord de JEKERT. Pour aboutir au résultat qui sera intégré dans le logiciel du pupitre, les études préliminaires sont réalisées sur un démonstrateur indépendant. L'avantage de

- A)** Déterminer avec rigueur l'ensemble des valeurs numériques à indiquer et à symboliser,
- B)** Choisir diverses options graphiques pour symboliser les paramètres importants,
- C)** Répartir géographiquement sur l'écran toutes ces informations pour en tirer le meilleur parti.

Préambule incontournable à la conception d'un tableau de bord, cette phase est primordiale car elle engage fortement toute la suite de l'étude. Non seulement il faut penser à lister l'intégralité des informations vitales. Si on en oublie une, et que l'on s'en rend compte quand presque tout est en place, la remise en cause impliquera de nombreuses heures perdues. Par ailleurs, s'il ne faut rien oublier, encombrer l'écran par des informations peu pertinentes, on perdra considérablement en qualité opérationnelle du "produit fini". Heureusement pour nous, sur cet aspect du projet notre petite machine est d'une exigence dérisoire. Le tableau donné ci-dessous résume les données :

➤ **Phase B de l'étude : Choisir des options graphiques.**

Page 23

➤ Phase B de l'étude : Affiner le visuel pour les inclinaisons.

T éléversez le petit *simulateur* graphique `Test_mode_graphique.ino` qui a servi à mettre au point les représentations graphiques, à déterminer les formes, à effectuer des choix. Plusieurs tailles ont été expérimentées, ainsi que divers positionnements sur la surface disponible. Il s'agit d'un *simulateur en ce sens que le Cap magnétique, le Roulis, le Tangage sont calculés dans une boucle d'affichage pour en faire varier en permanence la valeur*. Ainsi on peut observer sur l'écran leurs évolutions et les effets visuels sur l'affichage. Pour optimiser ce minuscule tableau de bord, il a été décidé de tracer la rose des caps la plus grande possible, et sans discontinuité. Son plus grand diamètre possible sera donc égal à la hauteur de la zone bleue sur l'afficheur. Pour laisser un maximum de place aux autres informations, le cercle gradué est placé en "butée" à gauche. L'information de navigation la plus importante est le Cap Magnétique, car c'est lui qui va orienter la trajectoire quand on fait évoluer JEKERT sur Mars. Aussi, pour attirer notre attention sa valeur est située dans la zone Jaune et encadrée. La logique veut que cette information soit située au dessus de la rose des caps. **Choix des symboles pour le ROULIS et le TANGAGE.**

Considérons la Fig.313 sur laquelle en **A** se trouve montrée la première représentation envisagée : Celle d'un secteur circulaire avec une aiguille tournant autour d'un pivot fictif situé bien en dessous des graduations. *(En réalité ce n'est pas un arc de cercle qui est tracé, mais une portion d'ellipse.)*

L'expérience a montré que l'aiguille α longue n'était pas idéale, en particulier quand elle se superposait aux graduations. Elle a donc été remplacée par une plus courte en enlevant l'extrémité violette. Comme les manipulations ont montré que l'effet visuel manquait de lisibilité, finalement l'aiguille a été abandonnée au profit d'un ruban linéaire courbe représenté en β .

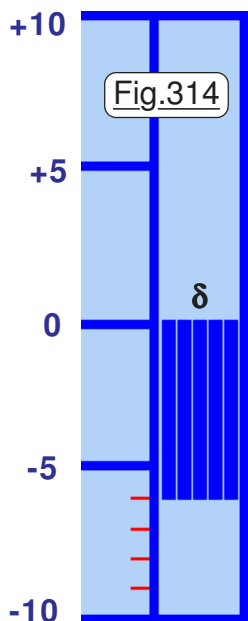
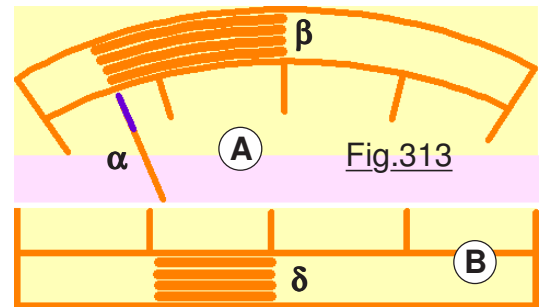
Pour que l'impact visuel soit suffisant, le ruban est composé de quatre lignes. La surface occupée par ce mode de représentation diminuait de la quantité représentée en rose clair.

Visuellement, cette représentation graphique s'avérait parfaitement utilisable. Elle a été abandonnée au profit de la solution **B** où l'on procède avec un ruban analogique rectiligne. Manifestement la surface occupée en hauteur est considérablement plus faible. Par ailleurs, programmer le cadran de base rectiligne est infiniment moins gourmand en octets que construire les secteurs elliptiques. Enfin, faire afficher les quatre lignes δ prend beaucoup moins de temps que construire les arcs d'ellipse β . Tangage et Roulis seront par conséquent visualisés par deux rampes analogiques rectilignes.

Choix des échelles pour le ROULIS et le TANGAGE.

Pour montrer à des visiteurs le fonctionnement magique de la centrale gyroscopique, il sera amusant d'incliner la sonde jusqu'à des angles de 90° dans un sens où dans l'autre, et ce tant en Roulis qu'en Tangage. Pour ce genre de manipulations elle sera tenue en l'air à la main. Puis, lorsque l'on reviendra à des simulations de gestion martienne, les plus grandes pentes autorisées ne dépasseront guère les quinze degrés. Aussi, pour amplifier l'efficacité visuelle, on utilise un effet de loupe. Les plages d'inclinaison maximales en Tangage et en Roulis seront comprises entre $\pm 10^\circ$. Chaque graduation représente dans ces conditions un intervalle de 5° . Ce choix rend parfaitement repérable un angle d'inclinaison aussi faible que 1° . Si les 10° sont dépassés, le ruban analogique "talonnera" sur des butées logicielles et restera limité dans le rectangle d'évolution du ruban. La graduation centrale sur l'échelle des valeurs correspond au neutre, c'est à dire à une inclinaison nulle. Par exemple sur la Fig.314 le Tangage est représenté verticalement. Les intervalles représentant 5° , les graduations pour 1° telles que

celles ajoutées en rouge ne seront pas visualisées car la surface disponible n'autorise pas une telle finesse de représentation. C'est mentalement que l'on évaluera la grandeur de l'angle. Sur la Fig.314 le tangage représenté vaut -6° . Du reste c'est l'impression générale qui importe, savoir instantanément si la sonde cabre ou pique. La valeur précise est indiquée en numérique. Le symbole d'inclinaison visuel est plus approprié à une prise en compte globale mais rapide.



► **Phase C de l'étude : Répartir les valeurs numériques et les dessins.**

Capitale pour aboutir à un résultat optimal, cette étape impose forcément de nombreuses tentatives pour tirer le meilleur parti d'un si petit afficheur. Au départ il n'était pas du tout évident qu'avec une définition de 128 x 64 pixels on puisse montrer toutes ces données avec un écran facile à interpréter. Un démonstrateur spécifique pour y parvenir s'avérait donc obligatoire. Considérons la Fig.315 sur laquelle en **1** nous avons la zone des pixels jaunes et en **8** celle des luminophores bleus. Entre ces deux surfaces, se trouve la ligne de séparation **3** qui engendre une discontinuité dans les tracés. C'est la raison pour laquelle le cercle de la rose des vents **4** a vu ses dimensions limitées par la définition en hauteur de la matrice **8**. Pour des raisons de lisibilité, les graduations **5** ont une longueur comprise entre six et sept pixels. La longueur de l'aiguille **6** se voit alors limitée entre le centre du cercle **4** et la zone interne des graduations car une superposition compliquerait singulièrement la programmation. (Quand l'aiguille "tourne", on efface l'ancien rayon et l'on trace le nouveau. Il faudrait tester si dessous il y avait une graduation pour la reconstituer ce qui augmente le code et ralentit le rafraîchissement que l'on désire "en temps réel".) En **7** est visualisé le "bug" qui indique la direction dans laquelle on désire faire évoluer JEKERT. (Par convention la rose des caps doit être considérée comme une carte géographique vue de dessus avec le Nord "en haut".) En **2**, bien encadrée, est précisée la valeur de l'orientation magnétique actuelle. L'aiguille **6** est tracée en fonction de cet angle, le Nord correspondant par convention à l'origine zéro. Pour obtenir le plus grand étalement possible du ruban analogique de Roulis, l'échelle graduée

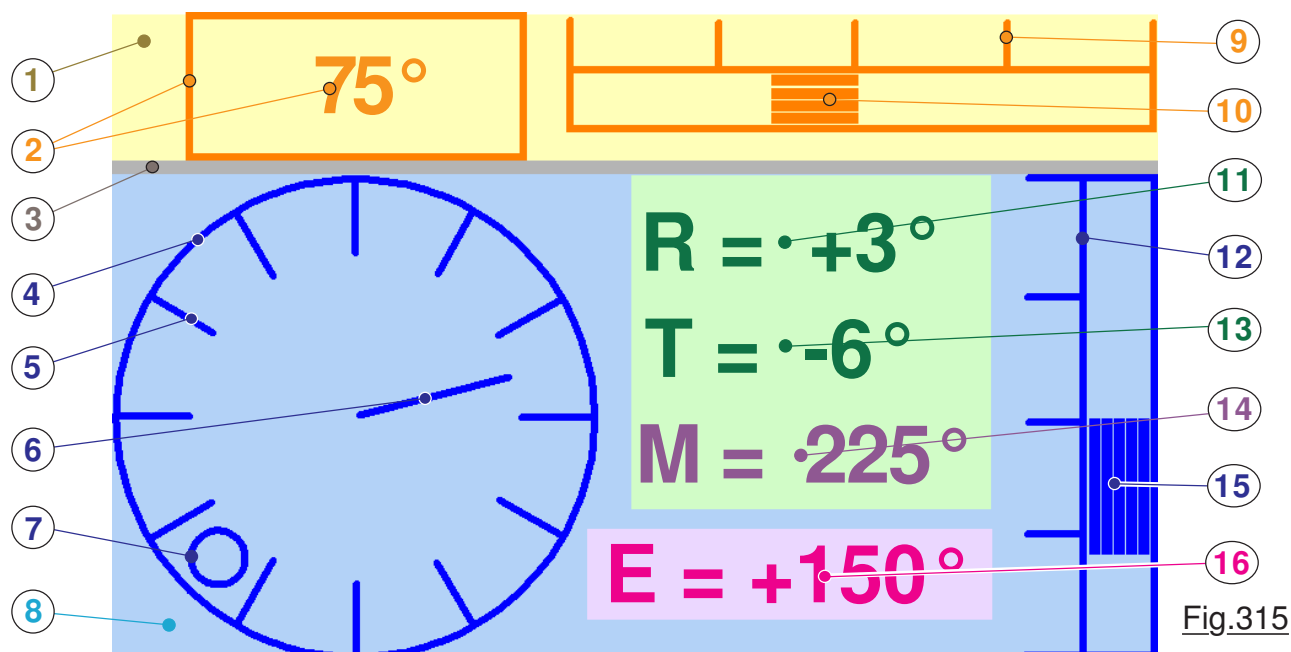


Fig.315

9 occupe un maximum de pixels en largeur dans la place disponible à droite de **2**. Cette largeur est toutefois limitée par le fait qu'il faut conserver un multiple de vingt pour assurer une proportion linéaire dans le tracé du ruban analogique **10**. L'étalement pour les graduations **12** affectées au **T**angage utilise toute la hauteur de la matrice **8**. Les luminophores de la matrice graphique sont plus espacés en hauteur qu'en largeur. Il est assez facile de s'en rendre compte en observant un caractère à la loupe. Il en résulte que pour obtenir visuellement deux rubans analogiques de largeurs équivalentes, **10** est tracé avec quatre lignes, alors que verticalement en **15** il en faut cinq. Il restait globalement les zones centrales verte et rose pour afficher les valeurs numériques des paramètres de navigation. L'entité qui utilise potentiellement le plus de caractères est l'**E**cart de route. Elle est placée en zone rose **16**, car sur la surface encore disponible c'est la partie inférieure du cercle **4** qui libère latéralement le plus d'espace. Dans la zone verte, trois lignes d'écriture sont possibles. Le **R**oulis en **11** est placé en haut, le plus proche possible du ruban analogique associé. En **14** est précisée la valeur de la direction souhaitée qui dans cet exemple vaut 225°. Elle est située assez logiquement juste au dessus de l'**E**cart de route. Il restait alors la ligne d'écriture **13** pour préciser la valeur du **T**angage actuel. La direction de la route souhaitée est indiquée en violet, car sur la version ultime du programme, en option il sera possible d'indiquer à sa place la valeur de la référence Gyroscopique interne. Dans ce cas le **M** sera remplacé par un **G**. Un chapitre précisera ce que représente cette référence gyroscopique, qui constitue une valeur "interne" au MPU-6050.

➤ Interprétation de la symbolisation graphique.

Agencée dans le but de conduire à une interprétation immédiate et naturelle, c'est forcément une évidence pour le concepteur. C'est du reste l'intérêt de créer, on est forcément satisfait du résultat puisqu'il correspond exactement à nos désirs. Chacun a sa logique, et souvent notre façon d'appréhender le monde est fonction de notre vécu. Est-il plus logique de circuler en automobile à droite de la chaussée ou à gauche ? Démontrez votre affirmation ...

Aussi, la logique des uns peut se montrer antagoniste avec d'autres. C'est la raison pour laquelle :

- Vous avez avec **Test_mode_graphique.ino** la possibilité d'adapter à votre mental le visuel actuel,
- Dans ce qui suit sera explicitée la logique qui s'applique à la construction des symboles graphiques.

➤ Sens positif d'orientation ou de rotation.

Dans notre vie de tous les jours, la notion de sens horaire ou antihoraire est souvent la plus utilisée, car les générations qui nous précèdent ont été habituées à voir tourner des aiguilles sur des horloges qui font "Tic tac tic tac". (Avec l'affichage numérique cette notion est en voie de disparition !)

C'est assez naturellement que le sens positif adopté est celui de la Fig.316 en **A**. (Pour bien vous montrer

que la logique peut parfois prendre des apparences étranges : Vous avez remarqué que sur ces pendules d'un autre temps, l'aiguille la plus longue correspond aux intervalles de temps les plus faibles !) En revanche, quand on aborde des notions d'espaces tridimensionnels, les sens d'orientations positifs puisent leur

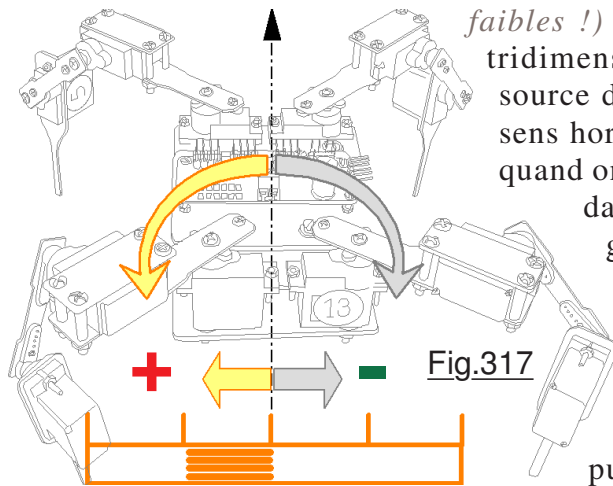


Fig.317

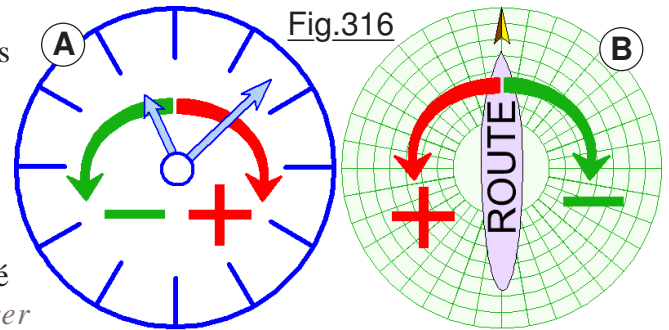


Fig.316

source dans des développements techniques bien différents. Le sens horaire qui "traditionnellement" est positif devient négatif quand on doit raisonner 3D. La Fig.316 en **B** précise les sens qui

dans cette étude seront utilisés. Positif quand on tourne à gauche et négatif quand on pivote vers tribord. Ce choix est conditionné par plusieurs facteurs que je crois pertinents : C'est le sens trigonométrique positif conventionnel. Hors pour tracer les graphes, la trigonométrie est fondamentale. Et puis certains d'entre vous, très aventureux, vont aller sur internet pour y puiser des explications sur le circuit MPU-6050. Ils seront

confrontés à des entités du genre "Angles d'Euler" et des sens de rotation connexes.

➤ **Interpréter l'inclinaison en ROULIS.** Lorsque l'on pilote un mobile, sauf cas très particulier, on regarde de l'arrière vers l'avant. (C'est tout de même mieux pour observer la route !) La Fig.317 nous place dans ces conditions. Notre regard va de l'arrière de la sonde vers l'avant. Si elle s'incline à gauche, la rotation sera considérée comme positive puisqu'elle se fait dans le sens trigonométrique direct. De l'autre côté, l'inclinaison sera affectée du signe moins.

INTERPRÉTATION IMMÉDIATE : De loin le plus simple consiste à ignorer toutes ces théories. Il suffit de se dire que le ruban analogique se décale vers le côté où la machine s'incline ... et ça c'est au final assez naturel.

➤ Interpréter l'inclinaison en TANGAGE.

Toujours en regardant de l'arrière vers l'avant, on veut savoir comment se déplace le "nez" de la sonde. De façon simple, le ruban analogique se décalera soit vers le bas, soit vers le haut. Sur la Fig.318 JEKERT est vu du côté bâbord et dans cet exemple "pique du nez" d'environ 7°.

INTERPRÉTATION IMMÉDIATE : Si la bande lumineuse est tracée vers le bas, la sonde est en train de piquer. Si le ruban analogique est tracé du centre vers le haut, c'est que JEKERT est cabrée.

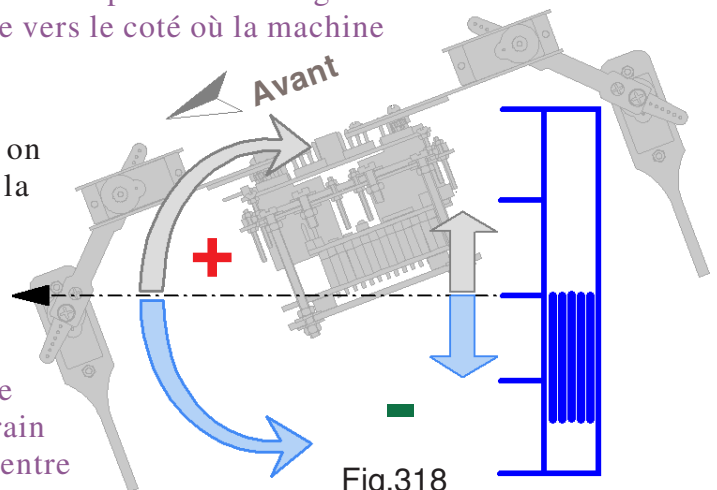


Fig.318

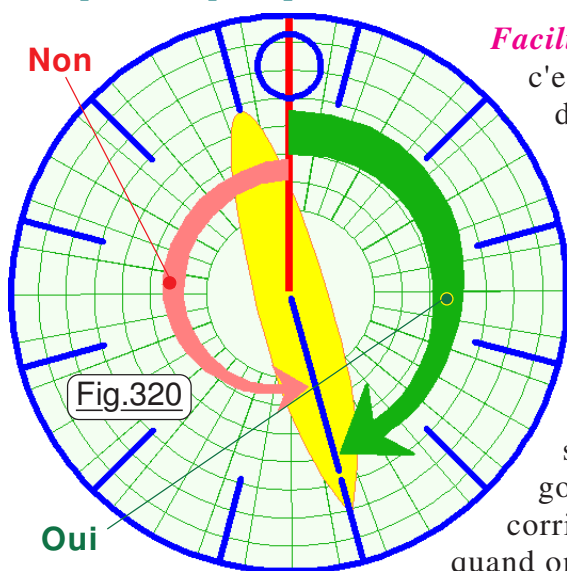
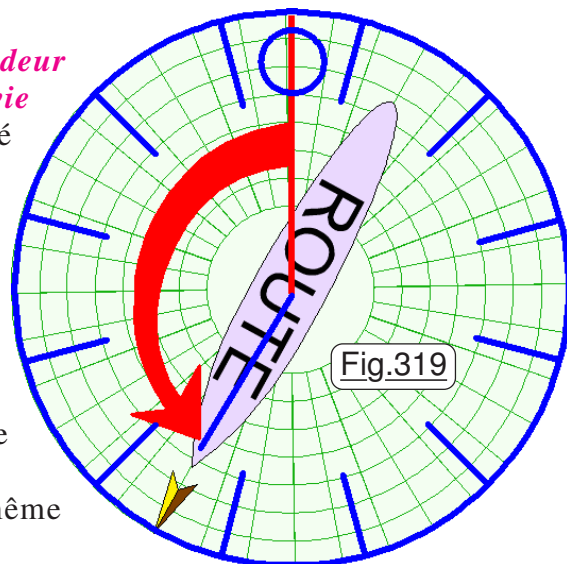
➤ Interpréter l'Ecart de route.

Par définition, *l'écart de route est la différence de grandeur angulaire entre la route désirée et celle actuellement suivie* par la petite machine. Cette assertion confine à une banalité évidente. Pourtant, la grandeur numérique qui sera affichée pour informer l'opérateur de la divergence entre le cap actuel de la sonde et celui dans laquelle on désire l'orienter n'est pas particulièrement immédiate, surtout lorsque la rose des vents est orientée en absolu, c'est à dire avec le zéro "vers le haut". Prenons le cas de la Fig.315 pour lequel on désire une route au 225 alors que l'orientation actuelle pointe le 75. Un simple calcul sur la définition explicitée ci-avant donne $225 - 75 = 150$. OUF ... c'est tout bon ! Supposons que la sonde navigue au 30°. Utilisant la même définition, l'écart affiché serait de $225 - 30 = +195$.

- Et alors ?

- Plouf spaltchhhh, la valeur indiquée sera concrètement de -165 !

- Glups, mais pourquoi donc ?



Faciliter le travail du navigateur consiste à raisonner en relatif,

c'est à dire de placer "en haut" la direction dans laquelle on désire naviguer. C'est la direction de la route que l'on observera quand le vaisseau respectera le cap. Et surtout on va diviser l'environnement en deux moitiés séparées par l'axe longitudinal du vaisseau. Peu importe que l'on dise gauche ou bâbord, droite ou tribord. La direction désirée étant "en haut", soit notre cap actuel est trop à gauche, soit trop à droite. Au point de vue angulaire on ne dépassera jamais 180°. Cette façon d'indiquer l'Ecart de route est bien plus facile à interpréter : Valeur positive supérieure à zéro : "Sens trigo" donc trop à bâbord il faut gouverner à droite. Valeur négative : Trop à droite il faut corriger vers bâbord. C'est à la fois simple et naturel ... enfin, quand on a barré un minimum de temps pour s'y habituer.

➤ Le minuscule tableau de bord.

Vraiment petit petit, tant par les dimensions que par la définition de la matrice de points, il n'en reste pas moins parfaitement exploitable et très agréable à utiliser. Avant de réinstaller le démonstrateur suivant dans la sagades logiciels de développement, observez attentivement le simulateur `Test_mode_graphique.ino` et habituez-vous aux diverses représentations symboliques et numériques. Sur la Fig.321 obtenue à partir du démonstrateur `P21M_Démonstrateur_Raquette.ino` est représentée une copie d'écran avec des valeurs réelles issues directement de la sonde. Pour ce cas le cap souhaité est plein Nord Magnétique soit la valeur nulle affichée sous la forme $M = 0^\circ$. Comme la sonde est tournée plein Sud, son Cap actuel est au 180° . L'écart de route est indiqué à droite pour 180° , c'est un choix "arbitraire" informatique. Soit on adopte les deux limites $[+179^\circ \text{ et } -180^\circ]$, soit on privilégie l'équivalent $[+180^\circ \text{ et } -179^\circ]$. Comme il n'est pas logique d'avoir deux valeurs différentes quand l'orientation actuelle est à l'opposée de celle désirée, on vote arbitrairement -181° ou $+180^\circ$ pour la représenter l'autre secteur étant réduit d'un degré par rapport au demi-cercle. Du reste, si pour corriger la route il faut faire un demi-tour, peu importe que l'on tourne d'un côté ou de l'autre, la manœuvre étant strictement équivalente. (Dans l'hypothèse où il n'y a pas des effets moteurs parasites, un côté plus favorable en fonction de la direction d'Éole sur un voilier etc.) Sur cet exemple réel on constate que les deux inclinaisons de seulement 1° d'amplitude sont parfaitement visibles.



Dans la salle informatique S4, le ronron des ordinateurs atteste sereinement de la concentration des divers ingénieurs logiciels les yeux rivés sur leurs écrans. L'informatique est une esclavagiste impitoyable. Par moment, en tant que chef nous sommes obligés de leur imposer de prendre leurs temps de pause. Ils sont tellement passionnés de ce qu'ils font qu'ils en oublient parfois d'aller manger quand la petite sirène sonne le rappel au réfectoire. Dans le coin à revues deux de nos esclaves sont en pleine discussion.

- Jour Tassin, jour Ferrando. Bigre, c'est quoi ça, vous avez dévalisé un musée ?
- Non chef, on retourne à l'école. Ce sont les manuels de trigo de Féfer.
- À bon, vous n'avez pas eu le BAC ni l'un ni l'autre ?
- Et vous chef, on peut le voir votre diplôme ? En fait on révise un peu les équidifs, car on doit faire tracer des ronds et des gradus, alors révisons un peu quelques Maths, on va en avoir besoin.

➤ **Un petit fifrelin de Théorie géométrico mathématique.**

Pléonasme "un Petit fifrelin", qui traduit assez bien le fait que l'on ne va pas trop se prendre la tête. Nous allons à peine survoler un ou deux détails pour comprendre les séquences qui sur l'écran tracent des traits inclinés. Les bibliothèques ... c'est génial ! Mais elles ne peuvent jamais se substituer totalement à un travail d'analyse. Tracer un cercle avec la bibliothèque de l'afficheur OLED est enfantin, tracer un segment également. Alors pourquoi diable se prendre la tête avec de la trigonométrie ? Pour frimer ? *(Comme faire valoir à une soirée mondaine ou lors d'un discours politique ... évitez, le sujet n'est pas porteur !)*

La réponse à cette question tient en deux questions :

- Comment tracer les graduations de la rose des caps ?
- Comment tracer l'aiguille analogique qui tourne autour d'un axe de rotation virtuel ?

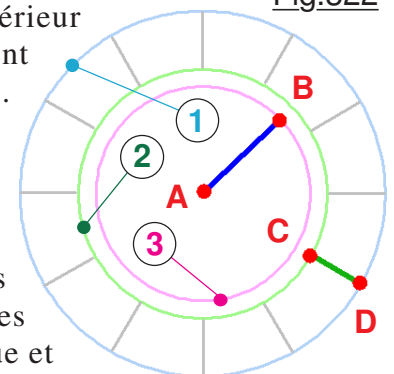
Pour les graduations, il serait possible de s'en tirer en imprimant une grille de 128 x 46 carrés. Puis on tracerait le cercle qui sur la Fig.315 est repéré 4. On ajouterait un cercle plus petit correspondant à l'extrémité centrale des graduations. On cryonnerait pour tracer les douze petits traits. Feutre de couleur en main chaque petit carré sous ces traits serait colorié, ainsi seraient représentés tous les pixels représentant les graduations. Fastoche, en plus ça ne prend que trois jours. Puis, on repèrerait les coordonnées de l'origine et de l'extrémité de chaque graduation et la procédure `display.drawLine()` se chargerait du reste. C'est une solution. Pas très élégante, vorace en temps pour tracer le modèle et inutilement couteuse en octets de programme. De toute façon le problème de l'aiguille analogique qui tourne n'est pas résolu. Pour sortir de cette impasse ... vive la trigo même si pas trop n'en faut !

Que ce soit pour tracer des graduations qui convergent vers un centre virtuel ou représenter une aiguille qui tourne, et dont l'extrémité se déplace sur un cercle, dans les deux cas on "tourne en rond". Traduisez : Il faut calculer la position d'un point situé sur un cercle.

Fig.322

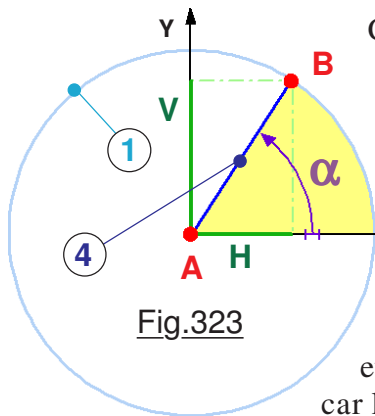
Pour vous en convaincre, observez la Fig.322 sur laquelle le cercle extérieur de la rose des vents est située en 1. Toutes les graduations convergent vers le centre de ce cercle. Leurs extrémités sont situées sur le cercle 2.

(Je vous l'avais bien dit que l'on allait faire des ronds !) Observons aussi l'aiguille qui tourne autour du pivot virtuel centré en A. Comme sa longueur est constante, son extrémité B se trouve forcément sur un cercle ayant pour rayon la distance AB. Pour tracer un segment tel que celui en vert, il suffit aussi de déterminer la position de ces extrémités C et D. Mis à part pour A qui se trouve au centre de la figure, tous les autres points sont situés dans des "vecteurs rayons" d'inclinaison connue et dont l'extrémité est située sur un cercle.



➤ **Et rond, et rond petit patarond.**

Tracer un cercle point par point dans une matrice cartésienne consiste à déterminer la position de chaque élément dans un repère orthogonal centré dans la zone concernée. Pour simplifier les études, nous avons tout bénéfice à centrer le repère sur le centre des cercles 1, 2 et 3. On aboutit à la Fig.323 qui inclut les ingrédients indispensables à la cuisine mathématique qui va suivre. La saveur de notre recette réside dans des entités bien connues nommés SINUS et



COSINUS qu'il faut faire mijoter à feux doux. Pour les coordonnées du point **A** c'est facile, deux fois zéro puisque le centre du repère **YOX** est placé précisément au centre du cercle **1**. Nous savons que le point **B** se trouve sur un cercle de centre **A** et de rayon **R** égal à **AB**. Par définition des entités trigonométriques, nous avons la position horizontale **H = R Cosinus (α) = R Cosinus (AB)**. De façon analogue, la position verticale **V = R Sinus (α) = R Sinus (AB)**. Donc, pour tracer l'aiguille en fonction de l'orientation de la sonde il suffira de déterminer la valeur de **α** en fonction du Cap Magnétique et de calculer **V** et **H** avec les formules trigonométriques. Nous avons bien de la chance, car l'**IDE** d'Arduino a bien écouté ses Professeurs quand il étudiait pour passer

son BAC. Il connaît parfaitement sa trigonométrie et met à notre disposition les fonctions **sin(Alpha)** et **cos(Alpha)**. C'est génial de simplicité, il suffit d'utiliser ces fonctions pour nos formules.

- Génial, génial, pas tant génial que ça Totoche.
- Ben pourquoi Dudule, t'es encore en train de râler ?
- Parce que l'angle Alpha il est toxique !
- Ha bon pourquoi ça ?
- Ben parce qu'il doit être exprimé en Radians, pas en Degrés !

G lups, faut immédiatement en informer le Ministre des angles. Bon, pas de quoi se rouler dans la poussière de colère. Il suffit de savoir que 360° correspondent à 2 x π Radians. Et π c'est ce nombre transcendantal qui fait environ 3,141592654 et des broutilles. En langage C il n'est donc pas bien compliqué de convertir un angle exprimé en degrés en son équivalent exprimé en radians. Une simple proportionnalité entre 360° et 6.2831852 encore que le compilateur du langage C d'Arduino manipule la constante **PI** sous forme d'un **float**. Et pour ceux à qui multiplier **PI** par deux et comparer avec du 360 donne des migraines, il y a en plus une fonction **radians(Angle)** qui retourne en Radians la valeur du paramètre **Angle** exprimé en degrés. L'est pas belle la vie arduinotique ?

➤ Tracer l'aiguille analogique.

G râce aux formules de la trigonométrie, vous aller voir que finalement animer un rayon vecteur virtuel devient un jeu d'enfant. Pour le prouver, nous allons analyser la séquence qui dans le logiciel du pupitre est chargée de faire bouger cette aiguille virtuelle et voir que la procédure se résume à pas grand chose. Dans cette séquence **X** et **Y** sont les coordonnées de l'extrémité du segment de droite à tracer et qui représente l'aiguille. **Angle** est la variable de type **int** qui contiendra la valeur du Cap actuel exprimée en degrés. **Alpha** de type **float** est l'identificateur du cap magnétique exprimé en radians. **SAVX** et **SAVY** préservent les anciennes valeurs de **X** et **Y** pour effacer. Cette séquence se trouve intégrée dans la boucle de base et explorée à chaque passage dans **void loop()**.

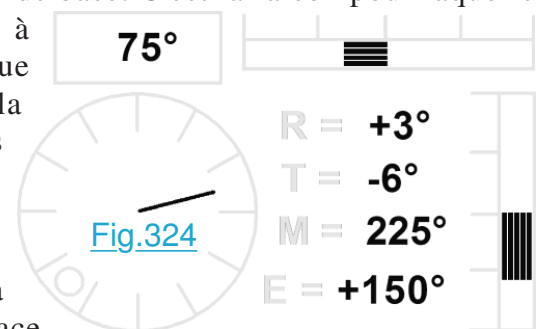
```
//----- Traite l'affichage continu des données de NAVIGATION -----
① if (Affichage_NAV_Continu) {
②   if (Mode_Graphique) {
③     Attendre_une_chaine_sur_le_Maitre(); Angle = Chaine_Memorisee.toInt();
        //----- Calcule les coordonnées -----
④     Alpha = radians(Angle);
⑤     SAVX = X; SAVY = Y;
⑥     X = 32 + (15 * sin(Alpha)); Y = 40 - (15 * cos(Alpha));
⑦     //----- Trace le rayon vecteur du CAP -----
⑧     display.drawLine(32, 40, SAVX, SAVY, BLACK);
⑨     display.drawLine(32, 40, X, Y, WHITE);
```

La séquence qui fait tourner l'aiguille virtuelle ne sera prise en compte que si en ① le booléen signale que l'option **affichage des données de navigation en continu** est validé. Puis, si le mode graphique n'est pas actif, les données seront affichées en mode numérique. Si en ② le booléen vaut **true**, alors la séquence sera déroulée. En ③ le programme attend de recevoir un message sur **RX**. À ce stade du programme, l'Esclave envoie la valeur du Cap Magnétique. La chaîne de caractères est convertie en un entier et logée dans **Angle**. En ④ l'angle est converti en radians dans **Alpha**. Les variables **X** et **Y** contiennent les coordonnées du rayon vecteur tracé dans la boucle

précédente. Avant de les recalculer pour l'orientation actuelle, on les sauvegarde temporairement dans **SAVX** et **SAVY** en ligne ⑤. En ligne ⑥ on calcule nos formules qui sont directement issues des études qui précèdent. La constante **15** représente la longueur de l'aiguille exprimée en pixels. Sur **X** on ajoute **32** et sur **y** on ajoute **40**. Ce sont les décalages du centre sur la matrice en latéral et en vertical. Le signe moins pour **Y** vient du fait que les coordonnées de la matrice OLED sont orientées positivement du haut vers le bas. Pour effacer l'aiguille on commence par la retracer en luminosité **BLACK**. L'origine du trait est **32,40** puisque ce sont les coordonnées du centre de la rose des vents dans la mosaïque graphique. L'extrémité est l'ancienne position préservée dans **SAVX** et **SAVY**. La ligne ⑧ efface l'ancien tracé. Comme il ne se superpose jamais sur les graduations, ces dernières ne sont affichées qu'une seule fois quand on valide le mode graphique. Moins il y a de pixels à changer, plus rapide sera le rafraichissement de l'écran. Enfin en ⑨ on trace l'aiguille dans sa nouvelle position. Si la sonde ne change pas de Cap, on aura l'impression d'une immobilité.

➤ Tracer les douze graduations de la rose des caps.

Désirant un affichage en temps réel, c'est à dire quasi instantané, il importe d'avoir le minimum de tracés à effectuer dans chaque passage de la boucle de base. C'est la raison pour laquelle tout ce qui reste constant ne sera affiché qu'une seule fois à l'ouverture du mode. Ensuite on se contente de ne rafraichir que ce qui est réputé pouvoir changer au cours du temps. Sur la [Fig.324](#) ce qui sera effacé et réécrit à chaque boucle est mis en évidence en noir alors que ce qui n'est affiché qu'à l'ouverture du mode graphique est représenté en gris clair. En minimisant ainsi le nombre de pixels à modifier sur la matrice on aboutit à une cadence de rafraichissement tout à fait acceptable. Dans cette ligne de conduite la séquence qui trace les douze graduations n'est pas incluse à la boucle de base, et placée en procédure d'initialisation. À regarder le listage de la séquence du tracé des graduations angulaires, on peut se demander si nous n'avons pas oublié quelque chose tellement elle est concise. Autant dire qu'elle n'est pas couteuse en octets. Par ailleurs, plus une séquence est rudimentaire, plus elle sera déroulée rapidement.



```
//----- Affiche la rose des CAPs -----
① display.drawCircle(32, 40, 23, WHITE); // Affiche le cercle.
//----- Tracer les graduations tous les 30 degrés -----
② for (int Angle = 0; Angle < 360; Angle = Angle + 30) {
③   Alpha = radians(Angle);
④   X = 32 + (16 * sin(Alpha)); Y = 40 - (16 * cos(Alpha));
⑤   SAVX = 32 + (23 * sin(Alpha)); SAVY = 40 - (23 * cos(Alpha));
⑥   display.drawLine(X, Y, SAVX, SAVY, WHITE);}
```

En ligne ① on commence par tracer le cercle extérieur. Dans ce but on utilise la procédure de la bibliothèque pour laquelle on donne les coordonnées du centre **32, 40**. Puis le paramètre **23** indique le rayon en pixels. Enfin avec **WHITE** on précise que les pixels seront allumés. En ② on génère la variable locale **Angle** et on va faire $360 / 30 = 12$ fois ce qui suit le délimiteur '{'. En ③ la variable globale **Alpha** reçoit exprimée en radians la valeur du compteur de boucle **Angle**. Il se trouve que **Angle** va prendre les valeurs de 0, 30, 60, 90 ... 330 qui sont les inclinaisons souhaitées pour les graduations. En ④ on calcule les coordonnées **X** et **Y** des extrémités centrales des graduations. Elles sont situées sur le cercle rose de la Fig.322 repéré **3**. Ce cercle est centré en **32,40** ce qui n'est pas nouveau. Le rayon de ce cercle fait **16** pixels. En ⑤ ce sont les coordonnées extérieures des tracés qui sont calculées. On réutilise ici les variables globales **SAVX** et **SAVY**. Le rayon du cercle extérieur est choisi à **23** pixels, soit identique au rayon du cercle extérieur. On n'a pas pris un pixel de moins, car avec les arrondis il se produisait quelques discontinuités nuisibles à l'esthétique. Enfin c'est la ligne ⑥ qui se charge de tracer en **WHITE** le segment de droite qui représentera la graduation. Avec cette analyse, nous allons clore le chapitre relatif au tout petit tableau de bord. Il est manifeste que parfois quelques rudiments de trigonométrie peuvent nous rendre de signalés services en programmation. Que vivent les sinus, les cosinus, les trucs mathématiques en nus ...

61) 12/02/2018 : L'aspect logiciel du graphisme géométrique (MJD 58161)

C hamboulement du planning. Les informaticiens avancent bien plus rapidement que prévu. JEKERT est encore loin du freinage de mise en orbite. Aussi, la direction a réuni les utilisateurs et les programmeurs. Puisque les ingénieurs logiciel disposent de bien plus de temps que prévu, fait assez rare dans le domaine de l'astronautique pour être souligné, les divers participants à la réunion ont mis en commun leurs spécificités pour ajouter au cahier des charges fonctionnels des fonctions qui n'avaient pas été envisagées initialement. Les campagnes d'essais ont montré qu'il était possible d'améliorer substantiellement les outils de pilotage du petit robot. Les dernières télémessures montrent qu'il hiberne parfaitement. Tous va bien pour JEKERT. Nous retournons en salle S4.

➤ **Améliorer la lisibilité du programme.**

G raphismes désirés en place, petit tableau de bord affiché en temps réel, **SÉCURITÉ** à tout va et tout y quanti ... rien à faire, nous n'en sommes qu'à 80% d'utilisation de l'espace réservé au programme dans le circuit ATmega328. Au sens de la rentabilité c'est encore tristounet. Plus exactement, ayant assouvi nos désirs "graphiques", il nous reste encore beaucoup de place pour perfectionner le pupitre. Les chapitres qui suivent seront consacrés aux nouvelles fonctions. Toujours affectés des défauts de présentation des logiciels précédents, ce sont **P21N_Démonstrateur_Raquette.ino** associé à **P22N_Démonstrateur_Sonde.ino** qui sont à télécharger sur Arduino pour expérimenter les nouveautés. Un programme lisible ne doit pas contenir une boucle de base **void loop()** comportant des dizaines de ligne. Une bonne pratique consiste à agencer une séquence qui ne fait appel qu'à quelques procédures dont les identificateurs se lisent "comme un livre ordinaire". Dans les versions précédentes, la boucle de base était devenue inutilisable car infiniment trop encombrée. Le remède qui s'impose a consisté à remplacer chaque "bloc fonctionnel" par l'appel d'une routine indépendante. Je vous invite à consulter le nouveau listage, car **P21N** fait peau neuve. Et encore, si nous poussions la logique jusqu'à créer une procédure pour les clignotements, la boucle principale du programme ne contiendrait que cinq instructions. Pour la lisibilité il serait impossible de faire mieux.

Contrairement à certaines écoles, je suis persuadé qu'un programme doit être avant toute chose lisible, **peu importe qu'il fonctionne mal**. Si le programmeur comprend immédiatement ce que devrait faire son logiciel, alors corriger les erreurs sera "facile". Si noyé dans un flot d'instructions dont six mois après les identificateurs ne sont plus si évident que ça, trouver la source d'une erreur deviendra impossible. Faisons un test : Ci-dessous je vous donne le listage d'un programme qui comporte une erreur. Sans autre explication, cherchez et corrigez cette dernière :

```
void loop() {  
  Faire_embarquer_les_passagers();  
  Mettre_en_route_les_moteurs();  
  Decoller();  
  Circuler_pour_aller_au_debut_de_piste();  
  Voler_jusqua_destination();  
  Se_poser_et_faire_debarquer_les_passagers();  
  Stopper_les_moteurs_et_couper_les_energies();  
}
```

La lisibilité c'est ça.

➤ **Imposer numériquement la valeur du Cap désiré.**

I nitialement, cette fonction n'avait pas été envisagée, car imposer la valeur d'un cap oblige à saisir un nombre, et notre clavier ergonomique n'a pas de touches numériques. Ce serait envisageable, il suffirait en mode saisie que certaines touches soient affectées aux dix chiffres. Cette idée simple n'est pas exploitable, car visuellement la "sérigraphie" du clavier deviendrait confuse, sans compter le fait que géométriquement les touches seraient mal réparties. Aussi, quand cette possibilité d'imposer un Cap a été éludée, caler le gyroscope avec la commande **59** se contentait d'affecter la valeur du Cap magnétique actuel. Si l'opérateur désirait suivre une route particulière, la première étape consistait à faire tourner la sonde jusqu'à ce qu'elle soit correctement orientée. Ensuite, on calait le gyroscope. Le pilotage consistant alors à corriger vers la gauche ou vers la droite quand en se déplaçant JEKERT dérive. Outre que cette procédure est **contraire aux concepts fondamentaux** du pilotage, elle s'avère pratiquement **irréalisable** sur le petit insecte mécanique.

Contraire au concepts de base de la conduite d'un mobile car "à l'envers". En toute logique on ajuste les systèmes de navigation pour la route désirée. Ensuite, on corrige la trajectoire en fonction des écarts indiqués par les systèmes de bord. Irréalizable, car JEKERT n'est pas un véhicule sur roues que l'on peut orienter finement en direction. La petite machine se dandine. Quand elle avance, les **Griffes** frottent, glissent, engendrant des dérives angulaires significatives. Quand on lui fait effectuer une rotation, l'amplitude du changement angulaire est de plusieurs degrés. Il sera impossible de l'orienter finement avec les mouvements élémentaires. On peut utiliser la **Torsion** pour caler la navigation, c'est faisable et prévu dans ce but. N'est pas très conviviale est cette méthode ...



Fig.325

Imposer un **Cap Magnétique désiré** peut se faire agréablement par utilisation du codeur incrémental. Dans les items du menu **EXPLOITER**, avec cinq pas dans le sens négatif à son appel, on fait afficher la fenêtre de la Fig.325 qui validée avec le **BPccr** ouvre la page de saisie. Par



Fig.326

défaut sur un RESET la route désirée est plein Nord, donc un Cap Magnétique nul. C'est ce que montre la Fig.326, le code **96** servant à récupérer la valeur initialisée sur la sonde. Comme plusieurs fois abordé, nous savons que pour des raisons de fiabilités des informations affichées sur la console, *les valeurs sont préservées sur la sonde et c'est par interrogation de cette dernière que l'on rafraichit l'écran OLED*. Les deux touches de "déplacement latéral **S6** et **S14** servent à déplacer en permutation circulaire le curseur d'écriture sur les trois chiffres du Cap. (Trois chiffres significatifs sont affichés systématiquement avec éventuellement des zéros en tête.) Ce curseur symbolisé par une flèche

verticale indique le chiffre qui sera modifié si on tourne le codeur rotatif quand on est en mode de saisie de la route souhaitée. Quand le codeur incrémental est manipulé dans un sens ou dans l'autre, on a l'impression visuelle que seul le chiffre pointé est modifié. En réalité, on agit sur un nombre, c'est à dire que l'entité complète est augmentée ou diminuée. Par exemple faire un pas négatif d'une unité quand le cap vaut **100°** fait passer l'affichage à **099°**. En négatif une butée virtuelle bloque la valeur à zéro. En positif, si arrivé à **359°** on tourne le capteur dans le sens horaire, quel que soit le chiffre indexé la valeur repasse à **000°**. La sortie du mode "Saisie du cap pour la route souhaitée" se fait impérativement avec **FIN**. Pour des raisons de convivialité la saisie du cap s'ouvre avec l'index sur un **Poids_Decimal** = 10. Tout B.P. autre que **FIN**, **S6** et **S14** génèrent un BIP sonore d'erreur.

➤ Référence Magnétique, référence Gyroscopique.

Information secondaire, la référence Gyroscopique interne est une donnée technique qui concerne plus la maintenance que la gestion du robot sur le sol de Mars. Quand le mode affichage continu des données est actif et que l'on visualise le petit tableau de bord graphique, chaque action sur le **BPccr** change la référence. Lorsque c'est la route magnétique initialisée par la procédure décrite dans le chapitre précédent qui est prise en compte, la lettre **M** indique clairement que c'est le Cap Magnétique actuel comparé à la Route Magnétique désirée qui génèrent la valeur de l'**Ecart de route**. Si on clique sur le **BPccr** la référence devient alors la donnée interne du MPU-6050 associée au gyroscope de Lacet. La lettre **M** est changée par **G** pour informer l'opérateur de l'option en cours. Cette référence est initialisée chaque fois que l'on valique dans le menu **EXPLOITER** l'item **Calage GYRO**, qui prend une valeur qui n'a rien à voir avec l'orientation en Lacet de la sonde. On peut considérer que cette valeur dépend de l'orientation du moment par rapport à l'univers. L'**Ecart de route** correspond alors à la différence entre l'orientation en Lacet actuelle et cette valeur préalablement enregistrée. En référence Gyroscopique on affiche bien un **écart d'orientation** entre la direction initiale et celle actuelle de la sonde, **MAIS PAR RAPPORT AUX ÉTOILES**. La rotation de l'astre sur lequel se trouve la sonde intervient comme déjà explicité en détails dans le chapitre sur la dérive gyroscopique. On peut considérer que la référence Gyroscopique conduit à une observation absolue par rapport à l'Univers, alors que la référence Magnétique est relative par rapport à l'astre sur lequel se trouve posé JEKERT. L'opérateur préfère généralement se repérer par rapport à l'environnement local, il privilégie en standard le Cap Magnétique pour s'orienter.

► Utilité pratique de la référence Gyroscopique.

Puisque par nature la référence magnétique est plus facile à prendre en compte par le pilote de la sonde, pourquoi se préoccuper de cette référence interne qui s'oriente par rapport à l'Univers et ignore le lieu sur lequel on se trouve ? Plusieurs raisons techniques viennent plaider sa cause. Initialement, le Gyroscope vient remplacer le compas magnétique quand ce dernier n'est plus crédible. Par exemple, quand sur un petit avion on engage un virage standard, on incline l'appareil en roulis. Si à cette inclinaison acquise relativement vite on ajoute de plus des turbulences, la boussole s'affole. Un navire, même de taille imposante, peut se faire chahuter fortement en roulis et en tangage par une mer forte. (*Rien n'est grand par rapport à la mer.*) Le compas de route dans ces conditions se dandine avec frénésie. Dans ces exemples, le gyroscope reste de "marbre". C'est l'outil idéal pour assurer la navigation ... sauf qu'il dérive car la Terre tourne par rapport à sa référence universelle. C'est alors le compas de route, quand il n'est pas perturbé, qui sert pour le pilote à recalibrer le gyroscope de bord. Il est clair qu'avec l'avènement du GPS, surtout comparé au coût important d'un gyroscope mécanique, les conservateurs de cap "à l'ancienne" sont en voie de disparition ...

Reste que la centrale gyroscopique est bien plus précise que la boussole. Nous savons que celle utilisée à bord de JEKERT est corrigée, le calcul assurant cette fonction conduit à une imprécision angulaire de plusieurs degrés. Enfin, notre petit robot est aussi un vaisseau spatial. Ces machines que l'on éjecte à des distances phénoménales se déplacent loin loin loin de tout astre. Pour les orienter en vue d'effectuer les corrections de trajectoire ou pour le freinage de capture gravitationnelle pour se mettre en orbite, elles n'ont que les étoiles pour faire le point. La centrale gyroscopique est alors la technologie incontournable pour assurer la navigation et ce d'autant plus qu'il n'y a plus présence de champs magnétiques pour orienter une boussole.

* Par défaut, sur un RESET de l'ATmega328, c'est le Cap Magnétique qui est initialisé, correspondant à une donnée plus opérationnelle que celle de la référence gyroscopique.

* La page de navigation dans le menu des **DONNEES** continue à afficher la valeur gyroscopique même si l'on est en option référence Magnétique car ce menu concerne la maintenance. C'est donc la dérive de l'Ecart gyroscopique qui observée à divers moments permet de s'assurer du bon fonctionnement de la centrale de navigation MPU-6050.

► Calculer l'Ecart de route avec la convention $\pm 180^\circ$.

Illustré par les Fig.319 et Fig.320 le chapitre **Faciliter le travail du navigateur consiste à raisonner en relatif** à exprimé en détails l'avantage à placer "en haut" la direction dans laquelle on désire aller. Puis, par rapport à cette route souhaitée, effectuer une dichotomie Gauche / Droite pour indiquer l'écart de route avec les valeurs limitées à $+179^\circ$ / -180° .

Pour mémoire, la Fig.327 résume le but à atteindre. Le petit cercle noir nommé parfois "bug" quand on parle d'un index spécifique à l'aviation, précise sur la rose des vents la direction que l'on désire emprunter. Par rapport à cette direction, on affecte par convention à gauche le signe positif, et à droite le signe négatif. L'écart de route sera ainsi limité en amplitude à 180° . Par exemple, sur la Fig.327 **E** sera égal à $+150^\circ$ pour le navire cas **A** et -120° pour la direction **B**. Pour calculer la valeur de l'écart, il serait possible de partir de la définition **E** = Cap actuel - Cap désiré. Si on dépasse 180° on prend la différence avec 360° . Puis on affecte le signe conventionnel. Cette approche qui se résume à une soustraction et deux comparaisons semble idéale de simplicité. Testez et vous allez vous rendre compte que l'on ne peut pas faire aussi élémentaire. Soit les amplitudes calculées ne sont pas bonnes, soit le signe ne convient pas. J'ai pourtant effectué de nombreux tests, aligné du code à profusion : Rien à faire, pas moyen de traiter par des calculs !

C'est la deuxième fois que confronté à ce type de problème, je n'ai pas réussi à trouver un algorithme de calcul relativement simple. Dans une telle situation, il faut bien trouver une solution. Pour ma part je fais alors appel à des boucles pour simuler le phénomène. Pour qu'une telle solution soit raisonnable il importe que la boucle de détermination ne comporte pas trop d'itérations, car alors le temps de calcul deviendrait prohibitif et ce d'autant plus que dans cette application ludique on désire un affichage en temps réel.

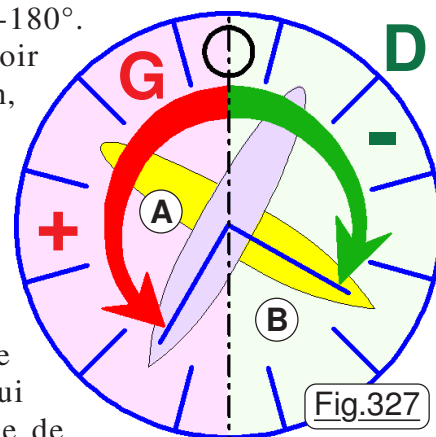


Fig.327

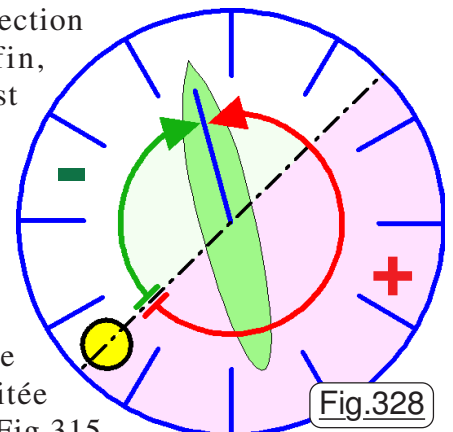
Considérons le listage de la procédure qui sert à calculer l'écart de route. Quand on fait appel à cette subroutine, préalablement la variable **Angle** est initialisée à la valeur du Cap Magnétique actuel d'orientation de la sonde. La variable **Ecart_gyroscopique** va servir à calculer la grandeur de l'écart de route, limitée à une amplitude maximale de 180°, et affectée du signe conventionnel, quel que soient au départ les valeurs de l'entier **Angle** codé sous la forme d'un **int** et de l'entier **Cap_magnetique_de_consigne** également préservé dans un **int**. Dans le premier cas le choix d'un **int** se justifie car la valeur manipulée peut aller jusqu'à 359 et dépasse donc les 124 d'un **byte**. **Ecart_gyroscopique** quand à lui ne dépassera pas 180. Toutefois un **byte** ne peut contenir cette variable, car elle est signée. (*Signée au sens de signe algébrique.*)

```

void Recalculer_Ecart_de_route() {
    // Calculer l'amplitude de l'écart de route.
    ① Ecart_gyroscopique = 0;
    ② while (Angle != Cap_magnetique_de_consigne) {
    ③     Ecart_gyroscopique++; Angle++;
    ④     if (Angle > 359) Angle = 0;}
    ⑤     if (Ecart_gyroscopique > 180)
    ⑥         Ecart_gyroscopique = (360 - Ecart_gyroscopique) * (-1);}

```

L'idée pour déterminer la valeur d'**Ecart_gyroscopique** consiste à imaginer que l'on est actuellement dans la bonne direction, donc en ① on indique une déviation nulle. Puis on simule une rotation du vaisseau, arbitrairement dans le sens positif. En ligne ③ **Ecart_gyroscopique** augmente de 1° ainsi que la valeur **Angle**. En ④ on "franchit" la valeur 360, donc on recycle à zéro. Puis, en ② on recommence tant que l'orientation fictive n'est pas égale à la direction souhaitée. Le programme sort alors de la boucle **while**. Enfin, l'instruction ⑤ détecte si l'on a tourné de plus d'un demi-tour. Si c'est le cas, en ⑥ on recalcule le complément à 360 et l'on inverse le signe. Dans le pire des cas, la boucle sera parcourue 359 fois. Elle ne compote que deux petites additions binaires et une comparaison. Compte tenu de la rapidité de fonctionnement de l'ATmega328, le temps pour effectuer cette simulation reste dérisoire : Solution adoptée. Pour faciliter l'analyse à ceux qui le désirent, la Fig.328 résume la méthode utilisée, sauf que cette fois le dessin décrit la scène le Nord étant conventionnellement vers le haut. La route souhaitée



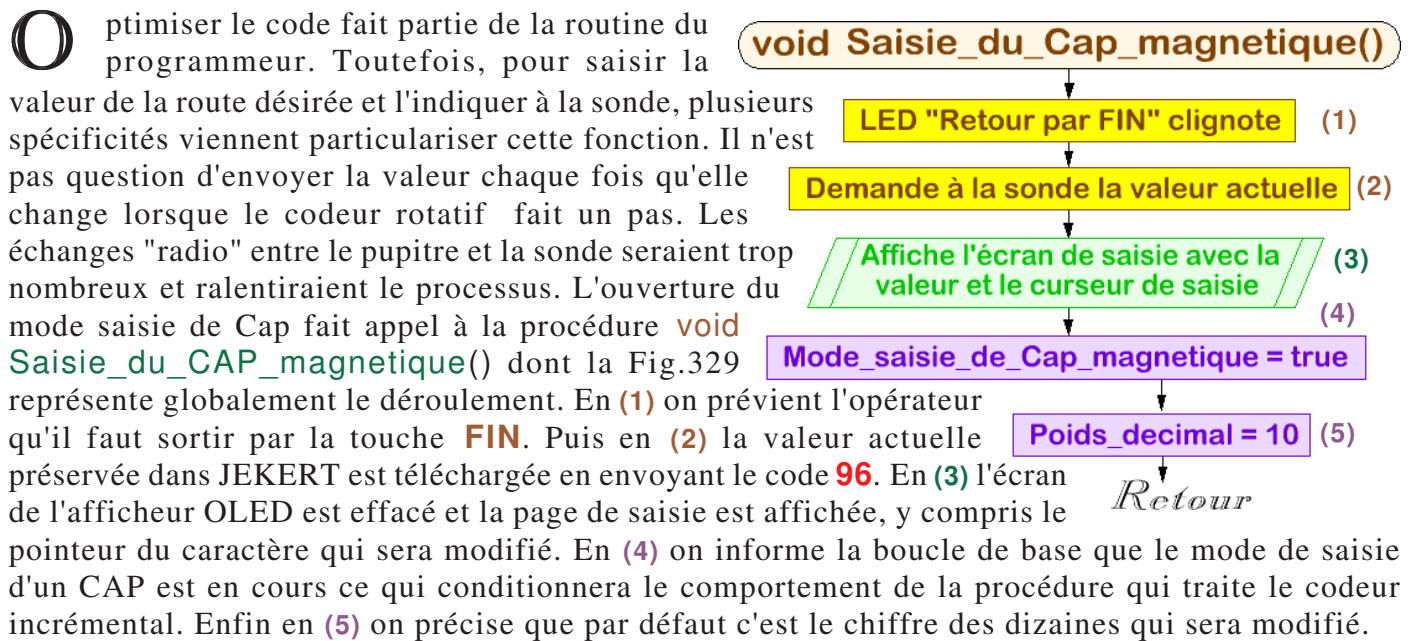
Désiré	Effectuer les tests pour :			
0	350 0 10	170 180 190	Réal ± 90° Réal ± 270°	
90	80 90 100	280 270 260	10 0 350	170 180 190
180	170 180 190	10 0 350	80 90 100	280 270 260
270	280 270 260	80 90 100	10 0 350	170 180 190
45	35 45 55	215 225 235	125 135 145	325 315 305
359	358 359 0	169 179 189	79 89 109	279 269 259

correspond aux 225° de la Fig.315 et nous évoluons actuellement au 355°.

- 1) Calculer par incréments de 1° la grandeur de l'écart ici montré par la flèche courbe rouge,
- 2) Si l'angle dépasse 359 degrés le forcer à zéro, (*Ici ce n'est pas le cas.*)
- 3) Quand la flèche rouge = le Cap actuel du vaisseau stopper l'évolution, (*Ici Ecart = 240°*)
- 4) Si **Ecart** > 180 recalculer : **Ecart** = 360 - **Ecart** = 120, et changer de signe.

(*Dans notre exemple on trouve -120° qui correspond bien à la flèche courbe verte sur le dessin.*)

ATTENTION : Avec une ou deux petites formules, vous aller coder un calcul et voir que ça fonctionne. Du coup vous en déduirez que Nulentout s'est fourvoyé. Les cas particuliers sont nombreux, les pièges variés. **Pour savoir si votre programme est correct, il faut impérativement établir un jeu d'essais exhaustif.** Le tableau donné ci-contre propose ma campagne de vérification. Toutes ces variantes réfléchies ont été soigneusement vérifiées. Surtout ne pas imaginer que moins de vérifications suffise. **Page 34**



D uring la boucle de base, le programme vérifie si le codeur rotatif a été utilisé. Si c'est la cas, alors `void loop()` fait appel à `TRAITE_LA_ROTATION_DU_CODEUR_INCREMENTAL()` qui teste quel mode est actuellement en cours. Si `Mode_saisie_de_CAP_magnetique` est à `true` alors cette procédure modifiera la valeur de l'identificateur `Cap_magnetique_de_consigne` en fonction du sens de rotation détecté et de la valeur actuelle de la variable `Poids_decimal`.

T ransmettre la valeur du cap souhaité pose un problème particulier, qui du reste avait fortement influencé l'abandon initial de cette fonction. Au début du projet, un bilan pessimiste prévoyait entre trente et quarante consignes, et déjà autant de fonctions semblaient beaucoup. Pour optimiser le programme, les messages envoyés sur **TX** ont donc été envisagés avec trois caractères. Deux pour le code compris entre **1** et **40**, suivi de la sentinelle **'*'**. Et puis une idée en attire une autre. Une nouvelle fonction induit parfois plusieurs consignes. La liste s'est allongée. L'optimisation du code a repoussé en permanence les limites, et arrivé vers **90** codes il restait encore de la place pour de nouvelles séquences. Les dernières commandes ont imposé une étude sévère, car la limite à deux chiffres pour le code imposait au maximum **99** consignes. Aussi n'imaginez-pas que le code **96** indispensable pour interroger la sonde était prévu d'avance. Tous les codes étaient déjà pris. Pour dégager une consigne, diverses modifications ont été apportées, raison pour laquelle du reste, la répartition des ordres dans le tableau des consignes n'est pas forcément constituée de regroupements logiques et ne respecte pas obligatoirement l'historique de leurs affectations.

P roblème : Non seulement il faut transmettre une valeur sur trois chiffres, et surtout il n'y a plus de codes disponibles dans les étagères informatiques. La solution consiste à envoyer une consigne particulière qui par sa structure sera différenciée de toutes les autres. Avant d'analyser le code reçu, l'esclave commence par tester s'il s'agit de cette dernière. La technique adoptée consiste à transmettre la consigne sous la forme d'une chaîne de caractères de la forme **"Cnnn*"**. Contrairement à tous les autres messages, celui-ci commence par **C** indiquant qu'il s'agit d'un Cap magnétique à enregistrer. L'esclave détectant ce caractère en tête de chaîne sait alors qu'il doit convertir le reste de la chaîne en nombre et affecter cette valeur à `Cap_magnetique_de_consigne`. Invoquée quand on clique sur **FIN**, la séquence qui traite la transmission de cette directive se résume à très peu de chose :

```

void Fermer_la_saisie_du_CAP_magnetique() {
  ① Serial.print('C'); Serial.print(Cap_magnetique_de_consigne); Envoyer_etoile();
  ② Attendre_une_chaine_sur_le_Maitre(); Efface_le_curseur(); display.update();
  ③ Mode_saisie_de_CAP_magnetique = false; LED_Retour_par_FIN_clignote = false;}
  
```

La ligne ① en trois instructions transmet la chaîne de cinq caractères commençant par l'identificateur spécifique **'C'**. Puis en ② la maître attend l'accusé de réception retourné par l'esclave. L'écran est alors modifié puis affiché pour informer l'opérateur de la prise en compte par la sonde. Pour finir, en ligne ③ ferme le mode de saisie et éteint la LED clignotante jaune.

61) 27/02/2018 : Liberté artistique, le pinceau informatique (MJD 58178)

Réunion de tous les personnels en salle S7 qui ressemble à une galerie d'exposition. Plusieurs tableaux présentent des logos étudiés par des artistes spécialistes en infographie. Les conversations vont bon train. Le Directeur est présent, et participe à l'ambiance joyeuse. C'est même lui qui a servi un verre de champagne à la serveuse présente pour cette petite fête. L'entreprise va choisir un LOGO pour symboliser sa dynamique, un petit dessin étant toujours préférable pour le grand public qu'un sigle NDRMSE difficile à retenir. Ce sont les personnels qui vont choisir le dessin qui emporte leur préférence. Autant dire que la bonne humeur embellit cette journée 58178 ...

➤ **Dessiner n'est pas jouer !**

Quand le destin s'acharne, arrive le moment où il faut savoir lâcher prise. La version **21N** du démonstrateur s'achemine vers l'aboutissement ultime du logiciel. Tout est écrit et mis à part quelques petites misères à corriger, le programme commence à ressembler au "produit fini". Rien à faire, on titille à peine les 83% d'occupation de la zone réservée dans l'ATmega328. La seule possibilité qui nous reste, c'est de dilapider des octets à tout va ! Traduisez ... on va se faire un petit plaisir qui informatiquement est boulimique en code et qui fonctionnellement n'apporte strictement rien : La page d'écran d'accueil du pupitre affichera un LOGO.

NOTE : L'expérience montre que faire afficher un dessin, y compris si ce dernier est de faible définition, consomme beaucoup de code, surtout si la définition de la matrice graphique qui le compose ne peut être logée en EEPROM. Hors c'est ici le cas puisque la mémoire non volatile de l'ATmega328 est saturée de textes. Ce luxe n'est donc envisageable que si nous avons vraiment "de la place à gaspiller". Dans ce projet, environ 470 octets seront engloutis dans cette fantaisie. C'est acceptable dans l'état actuel du développement. Si par la suite nous avons réellement besoin de dégager de la place pour une fonction importante ou pour corriger des erreurs, il serait facile de supprimer l'affichage du petit dessin. Ouvrir le chapitre "graphisme artistique" aborde un domaine passionnant informatiquement, et va aussi nous précipiter dans la pire des situations pour un programmeur : La collision de PILE. Informatiquement la suite est prometteuse.

➤ **La version P : Le dernier des démonstrateurs.**

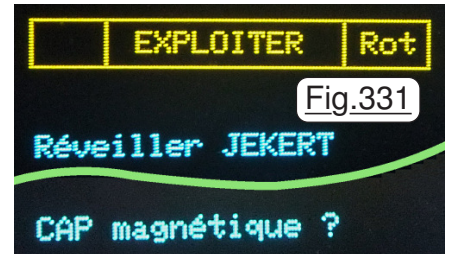
Téléversez **P21P_Démonstrateur_Raquette.ino** qui continue de fonctionner avec la version **P22N** du logiciel esclave. Il n'y aura plus de descendance mis à part la version ultime du programme, on commence à distinguer nettement le bout du tunnel. Suite à la version N on s'attendrait à la version O. Cette lettre a été écartée de la liste, car elle peut se confondre avec zéro. Donc ne cherchez pas un P21O hypothétique. En fin d'écriture du code dans le microcontrôleur, l'écran d'accueil de la Fig.330 s'affiche et la LED verte du clavier clignote rapidement nous invitant à cliquer sur une touche pour passer à la suite. Avouons que les personnels de la NDRMSE ont bien choisi en sélectionnant cette image qui présente la petite machine avec ses deux gros yeux ultrasons. Dans le cadre jaune est précisée la version du logiciel qui fonctionne actuellement sur le pupitre. Avant d'aborder les techniques pour créer un tel dessin et le faire afficher, nous allons passer en revue un certain nombre de petits perfectionnements qui accompagnent cette version du logiciel.



➤ **Les petits détails font les bons programmes.**

Bien que **P21P_Démonstrateur_Raquette.ino** nous réserve encore pas mal de bonnes surprises, quelques évolutions visant à améliorer la convivialité émaillent ce démonstrateur. Il suffit souvent de peu de chose pour "changer la vie". Quelques octets de plus pour beaucoup d'agacements en moins, l'investissement est bien plus que rentable. Les innombrables manipulations inhérentes au développement logiciel ont souvent conduit à un effet nul. Erreur de programme ou mauvaise touche cliquée ? Difficile de répondre. On utilise à nouveau le clavier et cette fois ça fonctionne. Le doute subsiste. Aussi, à partir de ce démonstrateur, toute touche qui n'a pas d'effet dans un menu engendrera un BIP sonore d'avertissement. L'opérateur sera ainsi averti qu'il a

sollicité un bouton poussoir qui dans le contexte actuel n'est pas valide. Rien de révolutionnaire certes, mais pour un coût de 46 octets je vous assure qu'à l'usage c'est très utile. Autre petit détail presque insignifiant, **SÉCURITÉ** est dorénavant imposée pour pouvoir enregistrer un balayage télémétrique ou un spectre colorimétrique. Pour 42 octets investis, nous ne risquons plus d'écraser un enregistrement précédent auquel nous tenions car caractéristique pour une démonstration publique. La LED **SÉCURITÉ** doit également être allumée pour **SAV POSTURE** et **Test UHF** dont il sera question plus avant. Pour un coût totalement exagéré de 360 octets, des détails de présentation dont certains sont montrés sur la Fig.331 ajoutent un fifrelin de rigueur aux affichages. Regardez bien. Vous ne voyez pas ce qui a changé ?



Ben ... chez Môamôa c'est maladif. Je ne supporte pas la vision de textes sur lesquels il manque les accentués. Alors pour envoyer un texto sur mon téléphone portable bas de gamme, je dois cheminer dans six écrans pour arriver à placer l'accent sur mon 'é'. C'est ainsi. Ajouter un accent sur l'écran OLED revient à tracer un petit segment de deux pixels exactement au bon endroit. C'est calamiteux en temps de programmation et dramatique en augmentation de taille de programme. Tant pis, dilapidons, bouliminons et l'écran contiendra de merveilleux petits détails qui passeront totalement inaperçus. Donc si un jour vous voulez gagner de la place vous saurez où trouver 360 octets !

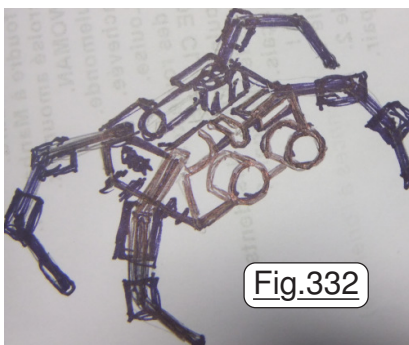
➤ La genèse d'un dessin informatique.

Griffonner un dessin sur une feuille de papier s'apprend quand nous sommes tout petits. Puis certains génies s'affirment et sont capables de vous croquer un visage en quelques coups de crayon. Sans prétendre à posséder ce talent, on peut se débrouiller, et ce d'autant plus que la définition de notre portrait sera dérisoire, n'imposant pas à l'artiste une grande finesse. Il importe toutefois de s'y prendre avec méthode pour ne pas engloutir un nombre d'heures déraisonnable. Pour illustrer ce propos, commencez par téléverser **Test_affiche_le_LOGO.ino** ce qui revient à pratiquer :

Mettre au point une présentation est complexe et impose de très nombreux essais. Si chaque tentative doit téléverser un programme de plus de 25ko, le temps de transfert devient excessif. Il faut impérativement, dans un tel cas, se créer un démonstrateur très allégé ne comportant que les routines de servitude pour afficher sur l'écran. En l'occurrence, c'est le module **Test_affiche_le_LOGO.ino** que vous téléversez sur Arduino, "Sketch" qui pourra éventuellement vous servir à modifier le code source pour construire un LOGO personnelle.

Plagiat éhonté vont hurler ceux qui ont remarqué que le texte de l'encadré qui précède est un Copié / Collé d'un chapitre déjà abordé quand on a voulu mettre au point la visualisation à l'écran du listage d'un programme enregistré. Seul le nom du démonstrateur spécifique a été corrigé. L'histoire se répète, la méthodologie ne change pas. Chaque fois que se présente à vous un cas épineux, il sera toujours avantageux de consommer un peu de temps pour créer un petit module qui ensuite rendra le développement bien plus facile. *N'oubliez jamais cette affirmation péremptoire.* La fin du téléversement du code dans l'ATmega328 s'achève par la visualisation d'un écran analogue à celui de la Fig.330 mis à part la date du logiciel. Ce qui impressionne, c'est la table des codes placée en tête de listage quand on consulte le programme source. Construite cette matrice de nombres binaires n'a rien d'élémentaire. Les chapitres qui suivent vont nous expliquer comment s'y prendre.

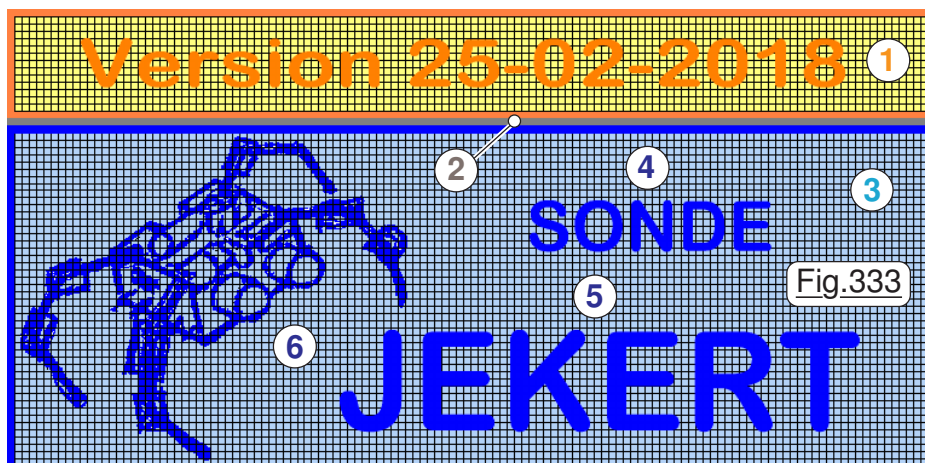
➤ Première étape : "Scibouiller" une caricature.



Préalable à la cuisine informatique qui va suivre, commençons par dessiner ce que l'on désire. Si nous n'avons pas les facilités d'un illustre peintre, rien n'interdit de faire une photographie numérique de ce que l'on désire représenter. Puis, à l'aide d'un quelconque logiciel de dessin, la triturer pour en faire un croquis initial. Ensuite on imprime ce dernier. Gomme et crayons en renfort, on réalise une épure avec des traits volontairement épais. Quand vu de loin l'ensemble semble satisfaisant, avec un feutre noir on surligne pour, comme sur la Fig.332 améliorer le contraste.

➤ Deuxième étape : Évaluer la définition du petit dessin.

Approche assez semblable à celle qui a conduit à la conception du petit tableau de bord virtuel, on doit disperser sur l'écran d'accueil les informations qui y figureront, définir leurs tailles respectives, et leurs positions précises. La Fig.333 représente grossièrement la page écran souhaitée. Dans le rectangle jaune **1** sera affichée la version du logiciel du pupitre à la plus petite police de caractères disponible dans la bibliothèque OLED. En **2** la ligne grise met en évidence la discontinuité de la matrice totale non pourvue de diodes électroluminescentes. En bleu clair la zone **3**, correspondant aux luminophores bleus, est encadrée par souci esthétique. En **4** le mot **SONDE** est en taille minimale, contrairement en **5** au nom **JEKERT** affiché en double taille. Surtout en **6** nous avons "glissé" le dessin du LOGO avec sa plus grande taille possible en hauteur. Notez que sous la **Griffe** avant gauche il n'y a pas de tracé, le mot **JEKERT** est donc centré latéralement entre le dessin et le bord droit du cadre. Il importe maintenant de "numériser" la mosaïque du minuscule dessin.



➤ Troisième étape : Transformer le dessin en une matrice de points.

Depuis l'abandon des tubes cathodiques qui généraient leurs images avec des lignes horizontales juxtaposées verticalement, les systèmes d'affichages actuels procèdent pratiquement tous par activation de points lumineux répartis sur des "grilles" généralement rectangulaires. Le nombre de points horizontaux et verticaux caractérisent la définition de ces écrans électroniques. Pour "numériser" l'esquisse représentée en **6** la technique consiste à imprimer une grille correspondant à la zone réservée au LOGO. Sur cette grille on trace au crayon le dessin désiré. Puis, soigneusement on noirci au feutre chaque petit carré qui se trouve sous une zone grisée initialement. Compte tenu de la surface affectée pour **6** on aboutit au résultat de la Fig.334 sur laquelle les carrés noirs correspondent en réalité aux pixels qui sur l'écran devront s'allumer. Dans la pratique, le résultat sera plus proche de la représentation Fig.335 sur laquelle les points allumés sont représentés en bleu clair. Le dessin a été épuré et ne représente plus les servomoteurs, les membres de l'insecte mécanique sont ainsi plus fins. Conserver leur présence conduisait à un ensemble confus les jambes ne se détachant pas assez du corps de la sonde. Maintenant que le chef-d'œuvre est abouti, il reste à le "binariser".

Fig.334



Fig.335



➤ Quatrième étape : Numériser la matrice de points.

Lorsque la grille artistique souhaitée est déterminée, il faut la faire correspondre aux luminophores physiques de la mosaïque électronique, trouver un moyen de l'y inscrire et si possible optimiser le code binaire qui sera chargé de ce traitement. Technique banale et incontournable, on utilise des OCTETs dont chaque BIT sera représentatif d'un point lumineux. État "1" le point sera allumé, état "0" il restera noir. Puis les octets seront balayés point par point pour être transportés dans la mémoire de l'afficheur OLED avec l'instruction `display.drawPixel()`. Il nous faut déterminer la structure du dessin "binaire" et la façon dont il sera balayé. Comme toujours se présente diverses possibilités, à nous de choisir le plus adaptée. (Optimiser !)

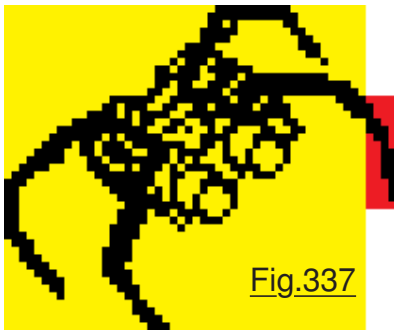
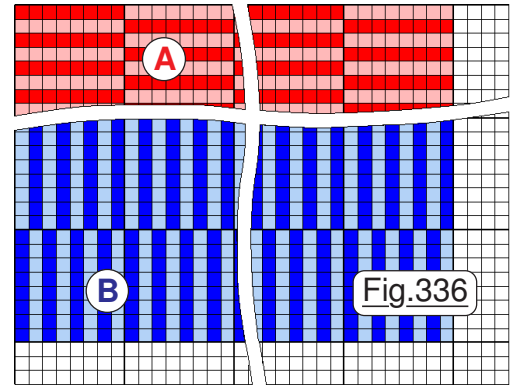


Hé ... t'as vu ? Ya un pou dans cette page !



Ben non gros malin, c'est le LOGO en matrice réelle de 52 x 43.

L' image binaire sera donc architecturée sous la forme d'un tableau d'OCTETs. Ces OCTETs seront rangés les uns à la suite des autres dans la mémoire dynamique de l'ATmega328, car pour le langage C++ les tableaux ne sont pas des constantes et peuvent évoluer au cours du temps. L'image à numériser doit donc être compartimentée en OCTETs, c'est à dire considérée comme des regroupements de huit points les uns contre les autres. En observant la Fig.336 on a mis en évidence deux possibilités strictement équivalentes en théorie. En **A** on regroupe les carrelages de la mosaïque horizontalement, contrairement au cas **B** où les OCTETS sont construits verticalement. Dans les deux cas, le nombre d'OCTETs horizontalement ou verticalement n'est pas



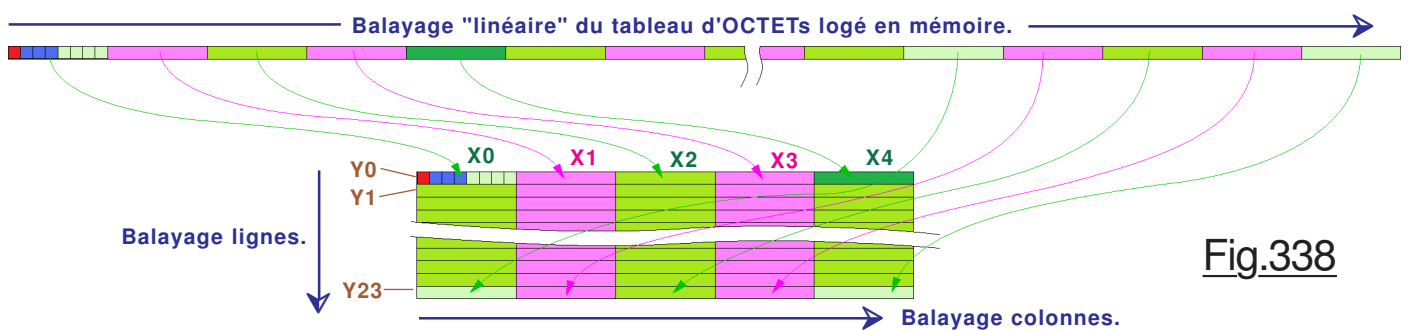
entier. L'image retenue sur la Fig.333 fait 43 lignes x 52 colonnes. Latéralement nous avons la relation $52 = 6 \times 8 + 4$. Verticalement, la répartition fait $43 = 5 \times 8 + 3$. L'optimisation consiste à remplir l'image avec des OCTETs "entiers", puis à tracer ce qui se trouve en dehors par construction élémentaire de segments. Comme j'avais adopté la solution **A** sur d'autres programmes, par paresse mentale j'ai opté pour cette approche. Ici **B** serait plus logique car il n'y a que trois lignes à tracer au lieu de quatre. Comme le bénéfice escompté par **B** est faible, alors profiter d'une séquence déjà au point reste un choix raisonnable. Nous allons donc construire le dessin de la zone jaune sur la Fig.337 par exploitation d'une table de pixels rangés en OCTETs dans un tableau de bytes. La Griffe contenue dans la zone rouge sera construite à l'aide de quatre segments de droite verticaux.

➤ Principe du balayage de construction d'une image.

Balayer, c'est déposer carrelage par carrelage et au bon endroit les divers éléments sur une surface qui représentera notre "fresque". Dans notre cas, chaque carrelage est informatiquement un petit rectangle allongé de huit petits carrés "jaunes" ou noirs nommés PIXELs. Chaque élément allongé est nommé OCTETET. Tous les carrelages qui vont composer notre œuvre d'art sont emballés dans un long paquet nommé `byte Image_48x43[258]`. (Tableau "linéaire" de $6 \times 43 = 258$ Octets.) Informatiquement, le dessin du LOGO est préservé en mémoire sous forme d'un tableau d'OCTETs dans lesquels chaque PIXEL est matérialisé par un BIT. Si le BIT est à "1" il faudra allumer le point image correspondant, s'il est à "0" il faudra l'éteindre. Avec la bibliothèque `Adafruit_ssd1306syp.h` l'instruction `display.drawPixel(X, Y, État);` permet d'initialiser indépendamment chaque BIT de la matrice électronique de l'afficheur OLED. On indique ses coordonnées et l'état désiré.

Dans le programme codé en C++, l'image complète est une suite d'OCTETs dans lesquels nous avons imposé manuellement un "0" où un "1" pour que le total soit représentatif de la fresque que l'on veut transposer dans la mémoire de l'écran électronique.

Comme on peut le visualiser sur la Fig.338, traiter un OCTET du tableau informatique (Constitué de 258 OCTETs successifs dans la RAM.) consiste à "plaquer" les huit bits de ce dernier au bon endroit dans la grande matrice de RAM de 128 x 64 luminophores de l'afficheur OLED.



Le BALAYAGE consiste à prendre un à un et dans l'ordre les OCTETs dans le tableau `byte Image_48x43[258]` et à en déposer les huit BITS dans la mémoire RAM de l'afficheur.

Revenons au principe du BALAYAGE. Encore un mot issu de notre quotidien. Balayer une surface consiste à en "traiter" la totalité en procédant par éléments dont les dimensions sont fonction de notre allonge et de la largeur du balais. C'est exactement ce que l'on fait informatiquement pour une image. Dans ce but, il nous faut plusieurs INDEX. Un premier INDEX va être initialisé sur

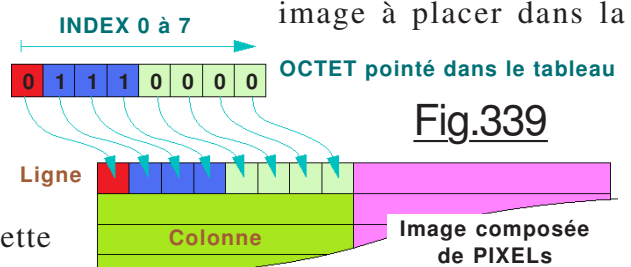
le tout premier élément du tableau `byte Image_48x43[258]`. Comme pour notre mental une image est à deux dimensions, autant raisonner comme des humains. La mémoire image devient alors une "macro matrice" de 6 COLONNES et de 24 lignes. La Fig.336 met en évidence le fait qu'une COLONNE est constituée de macroéléments pour le balayage, et constituée d'OCTETS. Donc, chaque COLONNE sera en réalité pour l'image un groupe de huit "colonnes pixels".

Pour effectuer un balayage simultané de la source (*C'est à dire du tableau d'octets.*) et de la destination, (*La mémoire de l'afficheur OLED.*) il nous faut concrètement trois INDEX. Le premier va explorer le tableau en ligne de l'élément 0 à l'élément 257. Le deuxième va balayer horizontalement les 6 COLONNES de la gauche vers la droite. Puis, arrivé tout à droite de la "macro matrice", il sera ramené à gauche en reprenant la valeur zéro. L'INDEX pour les lignes sera incrémenté, et l'on traitera alors la ligne suivante. Notez que l'on effectue l'exploration de la gauche vers la droite, et du haut vers le bas. C'est purement arbitraire, il serait totalement équivalent de faire du bas vers le haut, de la droite vers la gauche. Toutes les variantes sont autorisées. L'auteur de ces lignes est très influencé par la télévision de Papa, celle où les écrans étaient des tubes cathodiques. Le balayage image était effectué comme choisi arbitrairement dans ce programme. Quand on prend de l'âge ...

A ce stade de l'exposé, vous avez compris que chaque OCTET du tableau `Image_48x43` représente huit PIXELs qui dans l'image seront successifs. L'OCTET devient une sorte de fenêtre qui va se déplacer d'élément en élément dans `Image_48x43`, et balayer COLONNES/ligne dans la mémoire de l'afficheur. (*Pour l'écriture, ligne est en lettres minuscules car elles désignent réellement des lignes de PIXELs dans l'image. Alors que COLONNES est en majuscule car ce sont des groupements de huit "colonnes pixels".*) Ainsi vous pourrez faire plus facilement la distinction. Quand on va imbriquer dans deux boucles de type `for(...)` on imagine assez bien la "fenêtre jumelle" constituée d'un OCTET qui puise dans l'ordre des paquets de huit BITS dans le tableau, et qui par balayage ligne/COLONNE les dépose dans la RAM de la grille électronique de luminophores. C'est ici que l'on rencontre une petite difficulté informatique. Les trois INDEX étant conditionnés, le plus simple consisterait à lire l'OCTET dans le tableau, puis à l'écrire directement dans la RAM de l'afficheur. Sauf ... que la bibliothèque `Adafruit_ssd1306syp.h` ne nous fournit pas d'instruction pour lire et écrire des OCTETS. On peut soit utiliser des caractères, mais leurs empreintes nous sont imposées, soit déposer un à un des PIXELs en précisant leurs coordonnées.

➤ Déposer des OCTETS dans l'afficheur OLED.

A busons une fois de plus d'un petit dessin, toujours plus facile à cerner qu'un long verbiage. La Fig.339 suppose que l'identificateur OCTET contient les huit "PIXELs" puisés dans le tableau par lecture de l'un de ses éléments. Ce schéma suppose également que les index COLONNE et ligne pointent les coordonnées du premier point RAM de l'afficheur. Pour "déposer l'OCTET" dans l'image, il faut extraire BIT par BIT l'état du PIXEL, puis avec l'instruction `display.drawPixel(...)`; le placer dans la RAM image. On passe ensuite au BIT suivant et l'on recommence huit fois. Inutile de préciser que l'on va encore employer un index qui cette fois va varier entre 0 et 7. Il reste à "isoler" l'état de chaque BIT. Le plus rationnel dans ce type de traitement consiste à utiliser l'opération logiques de décalage. (*Revoir les informations sur le décalage logique SHIFT abordé en Fig.141 du TOME 3.*)



➤ Les opérateurs LOGIQUES travaillant sur des BITS.

D ésolé, mais dans la séquence qui construit le dessin sur la mosaïque de l'afficheur graphique, nous allons avoir besoin de faire appel à la notion de masque logique et d'opérateurs binaires travaillant BIT à BIT. Nous n'y consacrerons pas une thèse, toutefois un minimum d'informations sont indispensables pour pouvoir analyser le fonctionnement du programme. Le langage C++, comme tout système d'exploitation orienté microprocesseur du reste, met à notre disposition des opérateurs logiques qui permettent d'agir BIT à BIT dans un OCTET. Savoir les utiliser me semble absolument incontournable pour qui veut s'engager un tant soit peu en robotique.

Fondamentalement :

- A** • L'opérateur **OU** permet de **forcer des BITS à "1"** quand dans le **masque logique** les BITS associés sont à "1",
- B** • L'opérateur **OU EXCLUSIF** permet d'**inverser des BITS** quand dans le **masque logique** les BITS correspondants sont à "1",
- C** • L'opérateur **ET** permet de **forcer des BITS à "0"** quand dans le **masque logique** les BITS de mêmes rangs sont à "0".

Par définition, le **masque logique** est un OCTET que l'on "superpose" avec un opérateur logique à un autre OCTET pour n'en "voir" que certains BITS.

La Fig.340 présente quelques exemples. Comme un exposé complet sur le sujet sortirait largement du cadre de ce document, nous allons restreindre notre réflexion au **ET** logique.

Fig.340

	A OU								B OU EXCLUSIF								C ET							
OCTET	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
Masque logique	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
Résultat	1	1	1	1	1	0	1	0	0	1	0	1	1	0	1	0	1	0	1	0	0	0	0	0

Pour isoler la valeur d'un BIT dans un OCTET, il suffit de mettre à "0" tous les autres dans le masque logique. L'exemple de la Fig.341 focalisant sur le cinquième octet de **Image_48x43** montre que le résultat de l'opération donne bien un "1" si le bit examiné est à l'état haut, et un "0" s'il est à l'état bas.

OCTET	1	1	1	1	1	1	0	0	1	1	1	1	1	0	0
Masque logique	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Résultat du ET	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig.341

L'état du PIXEL étant connu, il suffit alors de le confier à l'instruction de décalage SHIFT qui le "déposera au bon endroit" dans la RAM de l'afficheur graphique. Nous avons tous les éléments en main pour analyser la séquence qui écrit le petit dessin dans la mémoire de l'afficheur.

```
//----- Affiche le LOGO -----
// Il faut reconstruire l'image sous la forme d'une matrice de 43 lignes de 48 colonnes.
① PTR = 0;
② for (byte Ligne = 0; Ligne < 43; Ligne++) { // Trace une ligne.
③   Colonne = 0;
④   for (byte Tranche = 0; Tranche < 6; Tranche++) { // Trace un octet horizontal.
⑤     OCTET = Image_48x43[PTR];
⑥     PTR++;
⑦     for (byte BIT = 0; BIT < 8; BIT++) { // Trace un BIT de l'OCTET.
// Extraire le BIT et le "poser" dans l'afficheur.
⑧     display.drawPixel(Colonne + 5, Ligne+19, OCTET & B10000000);
⑨     OCTET = OCTET << 1; Colonne++;}}}
```

PTR sera chargé de pointer l'octet à saisir dans le tableau **Image_48x43[258]**. En ligne ① il est placé au tout premier élément du tableau. En ligne ② on crée la variable locale **Ligne** qui va assurer le balayage vertical sur OLED. L'identificateur **Colonne** en variable globale sera chargé d'effectuer le balayage horizontal. En ligne ③ il pointe complètement à gauche sur l'afficheur. En ligne ④ on organise le balayage des six COLONNES avec la variable locale **Tranche**. L'instruction ⑤ recopie dans **OCTET** l'élément pointé par **PTR** dans **Image_48x43[258]**. En ⑥ on incrémente **PTR** pour pointer le prochain élément qui sera puisé dans le tableau. C'est l'instruction ⑦ qui va extraire **BIT** à **BIT** les huit PIXELs contenus dans **OCTET** et les inscrire dans la mémoire de l'afficheur OLED. La ligne ⑧ déroule toute une séquence d'instructions. En premier la fonction **OCTET & B10000000** isole le BIT le plus à gauche contenu dans **OCTET**. Cette fonction ne fait que retourner une valeur et n'affecte pas le contenu d'**OCTET**. Dans cette écriture, le masque logique est **B10000000**, **B** précisant que la valeur est exprimée en binaire. La valeur "0" ou "1" retournée par l'opérateur **&** est alors déposée aux coordonnées convenables dans l'afficheur. Les constantes **5** et **19** décalent à notre convenance le dessin en largeur et en hauteur. En ⑨ on décale à gauche **OCTET** d'une position pour faire passer en poids fort le BIT suivant à l'aide de l'instruction de décalage **<<**. **Page 41**

62) 02/03/2018 : Rechargement de la version **P** sur les consoles (MJD 58179)

Déjà le mois des giboulées qui pointe son museau. Nous sommes tous ravis, car nous avons reçu les nouveaux badges avec le LOGO de la NDRMSE. C'est un peu idiot, mais une certaine fierté à le porter démontre à quel point les personnels sont attachés à ce projet. Pour le moment JEKERT est endormie, mais personne ne doute qu'elle remplira vaillamment sa mission. Sur l'orbite de transfert qui la conduit à destination, elle a franchi plus de la moitié de la distance. Maintenant le Soleil la ralentit, car durant la première moitié de la trajectoire il l'attirait dans son emprise, mais elle a franchi le périhélie avec une vitesse considérable qui va la faire "remonter" jusqu'à Mars. Elle ne commencera à réaccélérer que lorsqu'elle arrivera dans sa sphère d'influence. Ce n'est pas demain la veille, les ingénieurs ont largement le temps de lui préparer encore quelques petits programmes.

➤ Recharger P21 version **P** sur l'ATmega328.

Poursuivre l'analyse des améliorations apportées au logiciel implique de le téléverser à nouveau, avec en début de son listage source la grande table `Image_LOGO[228]`. Elle a changé de nom pour avoir un identificateur plus parlant, mais surtout elle a subi un amaigrissement, passant de 258 octets à 228 soit une "perte de poids" de 30 octets. Nous verrons plus avant la raison de ce changement, car le dessin n'est pas modifié, impliquant qu'il a fallu compenser par tracé de segments de droite pour reconstituer l'image intégrale. Observez maintenant une autre petite modification :

- RESET > Un BP pour démarrer le programme > Touche **DONNEES** >
- Activer le **BP_{CCR}** pour allumer la LED bleue du mode graphique >
- Touche **OPTIONS** > **BP_{CCR}** pour allumer la LED **SÉCURITÉ** >
- **BP_{CCR}** jusqu'à afficher l'item **AFF. Nav. continu ?** > **OUI** >

À ce stade l'écran ne montre rien de bien nouveau.

- **BP_{CCR}** pour sélectionner la référence interne Gyroscopique :

En affichage graphique la valeur du **Calage GYRO**. n'est plus affichée et remplacé par "////" car correspond à une donnée interne au MPU6050. Ce n'est pas un paramètre vraiment opérationnel et en afficher la valeur peut prêter à confusion quand on observe l'**Ecart de route**. C'est le "bug" du CAP Magnétique qui est affiché en permanence, car il correspond à la direction dans laquelle on désire aller. L'interprétation de l'écran est plus simple, augmentant la convivialité d'utilisation.

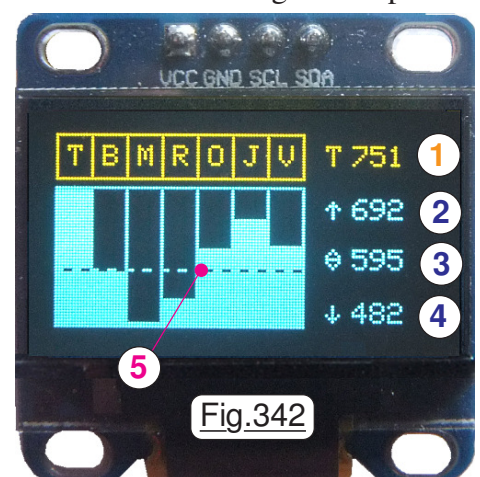
➤ Représentation graphique du spectre colorimétrique.

Rentabilité toujours aussi médiocre puisque le programme à ce stade ne consomme que 84% de l'espace disponible. Il reste encore 4384 octets à dilapider dans notre escarcelle. Hors de question de laisser toutes ces cellules à \$FF. Afficher les valeurs mesurées d'énergie du spectre colorimétrique sous forme d'un Barregraphe est bien plus parlant que sept valeurs numériques. La Fig.342 en donne un exemple de représentation. Chaque barre verticale présente une hauteur proportionnelle à son pourcentage énergétique. Les lettres dans la zone jaune précisent la couleur concernée :

T : Luminosité Totale **B** : Bleu **M** : Mauve
R : Rouge **O** : Orange **J** : Jaune **V** : Vert

La barre pour **T** occupera systématiquement l'intégralité de la hauteur du cadre bleu pour "dilater" au maximum l'amplitude verticale du graphe. En **1** est indiquée la valeur totale mesurée sans la filtration. Elle correspond à la hauteur du cadre bleu. En **2** comme l'indique le symbole \uparrow est précisée la valeur pour la couleur la plus énergétique. Dans cet exemple c'est le jaune. En **4** nous trouvons la valeur numérique de la couleur la moins énergétique symbolisée par \downarrow . La ligne pointillée **5** représente la valeur moyenne de toutes les couleurs. ATTENTION, ce n'est pas la valeur moyenne entre le maximum et le minimum (*Qui ici vaudrait $(692 + 482) / 2 = 587$.*) mais bien la moyenne des six valeurs énergétiques du spectre. Dans l'exemple présenté sur la Fig.342 elle vaut :

$\text{Moyenne} = (594 + 482 + 531 + 632 + 692 + 642) / 6 = 3573 / 6 = 595,5$. Il est donc normal que la ligne pointillée **5** ne soit pas exactement à mi-hauteur entre la barre représentative du **Mauve** et celle du **Jaune**. Le barregraphe engloutit 1148 octets soit environ 5% de l'espace mémoire



réservé au programme. C'est vraiment un produit de luxe ! Il faut toutefois relativiser un fifrelin. En effet, l'affichage textuel des versions précédentes ne présente plus d'intérêt. Il a donc été purement enlevé des options. Cette simplification engendre un gain en taille de programme d'autant plus important que les textes pour préciser les couleurs ont été enlevés de l'EEPROM. La place ainsi libérée a été remplacée par d'autres textes diminuant d'autant l'embonpoint du code objet. Au passage, faites afficher la **Posture actuelle** dans le menu **EXPLOITER**. Sortez correctement de cette fonction avec **FIN**. La petite erreur de programme est corrigée. Maintenant l'écran s'efface correctement.

➤ Des postures à gogo !

Exagérée ce titre, en réalité on ne pourra en sauvegarder que huit au maximum. Durant les diverses campagnes de tests, notamment pour la mise au point du pilotage manuel des moteurs, des configurations originales ou séduisantes ont été expérimentées. Quand on a passé un bon moment

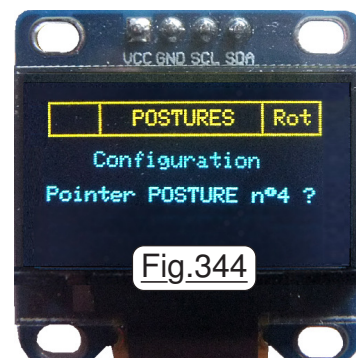
à peaufiner une configuration particulière, l'envie de la pérenniser s'impose. Aussi, reprendre la gestion de l'EEPROM de la sonde et ajouter les instructions nécessaires pour pouvoir immortaliser plusieurs postures originales devenait "prioritaire" dans la mesure où il reste encore 3176 emplacements non utilisés dans l'espace réservé au programme. Pour la modique somme de 306 octets on peut se faire raisonnablement ce petit plaisir. Avec 91% d'occupation, on passe la barre psychologique des 9/10, coté rentabilité on commence à se sentir mieux.

Pour implanter ces données en mémoire

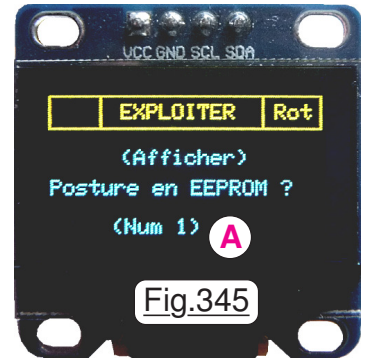
0768	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0784	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0800	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0816	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0832	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0848	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0864	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0880	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0896	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0912	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0928	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0944	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0960	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0976	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0992	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1008	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

non volatile EEPROM, il nous faut revoir "le plan d'occupation des sols". Nous en étions à la répartition des informations représentée en Fig.288 qui laissait tout le haut de la mémoire disponible à partir de l'adresse 768. Chaque posture contient douze consignes codées sur des **int** soit 24 octets. Il serait possible d'en sauvegarder $255 / 24 = 10,625$. (*255 est la taille actuelle de la zone libre en haut de la mémoire EEPROM.*) Pour simplifier le logiciel on limitera leur n° à un seul chiffre significatif. Le nombre maximum possible sera donc de neuf configurations. Initialement, pour simplifier le logiciel il a été décidé de loger chaque posture sur des zones de 32 octets. (Voir la Fig.343) Le nombre maximum possible est alors de huit, et comme on utilise la procédure de sélection d'un programme du menu **APPRENDRE**, si 9 a été sélectionné nous aurons génération d'une erreur **E11**. Au final, ce n'est pas une très bonne idée. Il serait préférable de placer les postures les unes à la suite des autres et d'en limiter naturellement le nombre à neuf lors de la saisie. Si le cœur vous en dit ... La méthode retenue pour pouvoir choisir entre les huit empreintes possibles n'est pas du tout compliquée. Elle consiste à ajouter un item vu sur la Fig.344 qui recopie la valeur de l'index de programme dans le n° de posture traitée pour l'affichage et pour l'utilisation. Le protocole pour sélectionner une posture parmi les huit possible est précisé dans le manuel d'utilisation du pupitre au chapitre ➤ **Protocole pour indexer le n° de posture EEPROM** et reproduit ci-dessous pour que vous puissiez l'expérimenter avec **P21P**.

- 1) Activer le menu **APPRENDRE**,
- 2) Un pas en **Rotation** ⤴ pour avoir l'item **Changer PGM EEPROM**,
- 3) **OUI** pour valider la saisie, (*Le cadre jaune indique le n° actuel*)
- 4) **Rotation** ⤴ ou ⤵, pour imposer le n° désiré, (*Valeurs de 1 à 8*)
- 5) Sortir de la saisie par **FIN**,
- 6) Activer le menu des **POSTURES**,
- 7) Deux pas en **Rotation** ⤴ pour **Pointer POSTURE n°N ?**,
- 8) Touche **OUI** pour la valider. La consigne **99** est envoyée à la sonde. Si la valeur est valide l'ACR est **OK**, si **N** vaut **9** un **E11** est généré assorti d'un BIP sonore.



S cratchhhh prouitchhhh bom bring protchhhh ! Normalement, avec toutes ces onomatopées vous devriez comprendre qu'un incident grave vient de se produire. Si ce n'est pas le cas, je peux vous en ajouter quelques lignes ! C'est un peu comme le SIDA. La toute première fois qu'il en a été question au journal télévisé de 20h, le présentateur n'a évoqué ce sigle qu'en quelques phrases. Et puis il est passé à des thèmes bien plus importants comme les résultats sportifs du moment. Discretion, oublie ... et quelques années plus tard, c'est mondialement que le sigle SIDA occupe la tête d'affiche. COLLISION DE PILE, ces trois mots constituent le "SIDA de l'informaticien". Ce virus s'insère dans le programme avec une sournoiserie absolue et cause une épidémie dramatique. Particulièrement délicat à diagnostiquer ce problème vicieux peut arriver alors que tout semble normal. Par exemple, ayant ajouté la séquence qui permet d'enregistrer jusqu'à huit posture, le programme semblait fonctionner correctement. Vérification effectuée, dans le menu EXPLOITER l'item de la Fig.345 montre que maintenant en **A** est précisée l'**empreinte actuellement pointée en mémoire**. La sauvegarde ainsi que le rechargement de la posture visée s'avèrent corrects et ce pour les huit emplacements possibles. Bref, nous pouvions voir la vie en rose. Et **PAHTATRAKKKKkkkkkk !** (Une exclamaturgique de plus !) Allant dans le menu des **DONNEES** certains affichages étaient devenus totalement incohérents.



- **Brusquement un programme présente un comportement anormal,**
- **La séquence qui diverge n'a rien à voir avec les dernières modifications effectuées.**
- **Quand un tel comportement étrange se produit avec la certitude qu'il n'y a aucune relation entre la séquence anormale et les derniers correctifs apportés au logiciel, il faut diagnostiquer une collision de PILE, c'est la cause la plus probable du problème.**

➤ Collision entre la PILE et le TAS !

P hénomène particulièrement sournois, la collision de pile survient brusquement sans qu'aucun signe avant-coureur ne nous prévienne. Pour comprendre de quoi il s'agit, il faut entrer dans la vie intime du fonctionnement des microcontrôleurs. Il serait hors propos dans ces lignes d'étudier à la loupe l'agencement matériel de l'ATmega328 est d'en détailler finement le fonctionnement interne. Nous allons dans ce chapitre nous en tenir au strict minimum vital.

Fonctionnement de la mémoire vive SRAM.

La mémoire vive (256 + 2Ko) est généralement divisée en quatre zones :

- Les 256 premiers octets pour les registres généraux du microcontrôleur (Représentée en jaune sur la Fig.346) occupent "le bas" de la SRAM.
- La zone nommée **BSS** qui contient toutes les variables globales, allouées statiquement au moment de l'édition de lien lors de la compilation. La **BSS** est utilisée par de nombreux compilateurs pour désigner une zone de données contenant les variables statiques déclarées dans les initialisations, et les déclarations.
- Le **TAS** sur lequel on entasse du bas vers le haut est destiné aux **allocations dynamiques** dans lequel on peut attribuer et libérer des blocs de mémoire. (Nommé **HEAP**) Le **TAS** se fragmente généralement au cours de l'évolution du programme, (Car une variable locale libérant de la place laisse "un trou" non utilisé.) avec un risque notable de le rendre inutilisable.

*Défragmenter **HEAP** par une séquence de code de type "Ramasse miettes" est faisable mais relativement dangereux, car si l'on déplace une variable en cours d'utilisation, les conséquences peuvent s'avérer ingérables.*

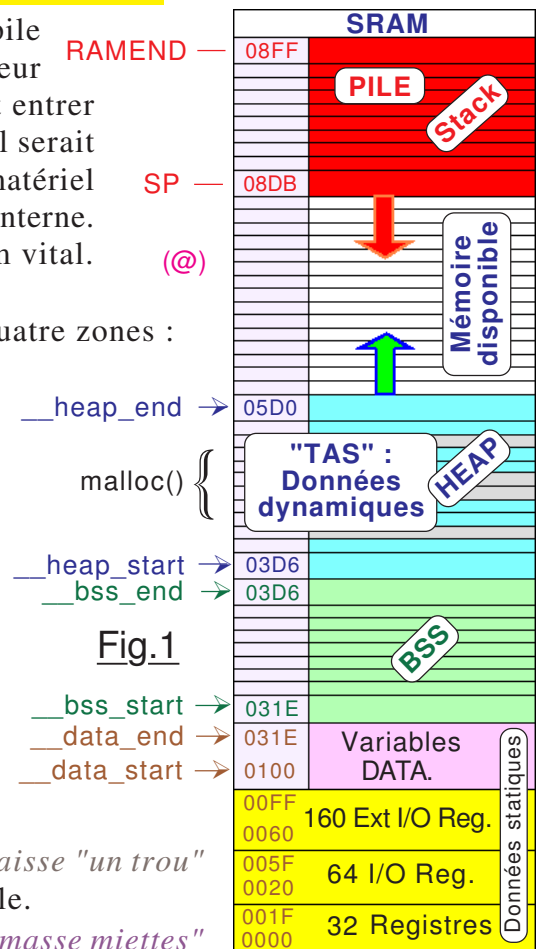


Fig.346

- La **PILE** nommée **STACK** mémorise temporairement :
 - * Les paramètres associés à l'appel des fonctions et procédures,
 - * **Les adresses de retour des fonctions et procédures**,
 - * Les variables locales aux fonctions et procédures.

La **PILE** est une zone de mémoire commençant en haut de la SRAM qui se charge vers le bas de façon linéaire et continue lors des appels des fonctions ou des procédures. Elle se réduit vers le haut lors des retours. Chaque appel à une procédure suppose que lorsque cette dernière s'achève, le déroulement du programme doit passer à la suite. Pour aborder d'une façon volontairement très simplifiée le fonctionnement du compilateur, nous allons raisonner sur la fig.347 qui dans la boucle de base **void loop()** enchaîne deux procédures. Arrivé à la ligne d'instruction ① l'analyseur syntaxique



Fig.347

sait que le microcontrôleur va devoir "se brancher" sur les instructions de ce qui pour le programmeur est un identificateur nommé **TRAITER_LE_CLAVIER()**. L'analyseur syntaxique "saute" l'accolade **{** et recherche la première instruction cohérente qui suit. Il note alors l'adresse dans le programme du début de cette subroutine dans une cellule nommée **Tester_le_clavier**. Le déroutement ne peut se faire immédiatement. En effet, que devra faire l'ATmega328 quand toutes les instructions de la subroutine auront été déroulées, événement jalonné par la "fermeture du bloc" avec **}** ?

Il faudra passer à la suite du programme, c'est à dire en ligne ②. Hors le microcontrôleur ne peut pas inventer cette adresse de retour. Donc, avant de brancher sur **Tester_le_clavier** par la route colorée en bleu ciel, le compilateur demande à l'ATmega328 d'EMPLILER l'adresse de retour de procédure. Quand le programme arrive à l'instruction "Retour de procédure" en **}**, l'adresse est DÉPILÉE et le processeur se branche sur cette dernière. Vous avez noté que chaque appel à une fonction, à une procédure, à une interruption, c'est à dire toute instruction qui déroute le programme, le microcontrôleur commence par empiler deux octets d'adresse. La distance qui sépare le **TAS** et la **PILE** diminue d'autant. N'oublions pas que les chaînes de caractères et les tableaux sont placées sur le **TAS**. Sans que nous en ayons conscience, la zone libre peut rapidement devenir trop faible pour tout contenir.

➤ Configuration qui produit la collision de PILE avec le TAS.

Concrètement, bien que l'on utilise l'expression "COLLISION DE PILE", le pointeur **PILE** est injustement accusé. C'est l'inverse qui se produit. Dans notre programme, nous avons un nombre d'octets significatifs qui sont entassés sous forme des chaînes de caractères. Et puis on a dilapidé 258 emplacements pour loger le tableau qui représente notre merveilleux LOGO. Suite aux méthodes préconisées dans ce didacticiel, on privilégie des sous-routines "simples" qui font appel à des procédures et des fonctions. Elles même récursivement vont invoquer d'autres séquences, avec passage de paramètres qui s'empilent à leur tour. Et puis on oublie qu'un nombre important d'interruptions se produisent en tâche de fond. Le codeur rotatif génère des interruptions ... mais il est loin d'être le seul coupable. Les fonctions **delay()**, **millis()**, la PWM ... une foule de ressources internes déclenche des interruptions. C'est transparent car c'est le compilateur C++ qui sur ces instructions fait sa cuisine interne. Arrive un moment, ou trop de données sont empilées sur le **TAS** et viennent écraser les adresses empilées. À ce stade, tout va encore pour le mieux. Puis le délimiteur **}** demande au processeur de dépiler une adresse de retour. Comme cette dernière contient les résidus de la variable qui a "écrasé" les octets, le programme se "branche" strictement n'importe où. Étant alors sur du code objet incohérent, le comportement du logiciel devient totalement aléatoire. C'est ce qui s'est passé sur le démonstrateur et qui nous a imposé d'apporter un remède à cette situation incongrue.

PRÉVENTIF : Aucun programmeur n'est à l'abri d'une telle "catastrophe". Aussi, pour minimiser les risques il faut placer le minimum de chaînes de caractères dans le programme car en réalité elles sont placées sur le **TAS**. Il faut également minimiser les tableaux. Enfin, quand le programme est terminé, il importe de vérifier que le risque est faible en faisant afficher la place encore disponible entre la **PILE** et le **TAS**. Le compilateur C++ d'Arduino nous fournit des outils pour ça. **Page 45**

63) 12/03/2018 : Suivi des transmissions UHF vers la sonde (MJD 58189)

Lentement, le tableau du planning se déplume. Un à un on enlève les petits cartons quand de nouvelles séquences sur les pupitres sont validées. Arrivant ce matin en salle informatique S4 nous trouvons Tassin et Ferrando en grande discussion. Visiblement ils ne semblent pas d'accords.

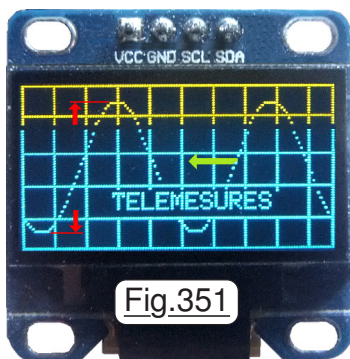
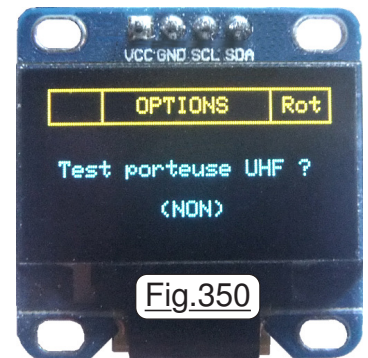
- *À chef vous tombez bien, faut nous départager !*
- *Ha j'ai compris, vous voulez faire une grille de LOTO en commun et n'êtes pas unanimes pour cocher les cases qui vont de toute façon vous faire perdre votre mise.*
- *Pire que ça chef, ya Tassin qui veut absolument me prouver que l'on doit moduler en phase.*
- *Exact chef, c'est de la modulation de phase, pas de fréquence,*
- *Attendez les gars, vous me parlez chinois là j'entrave rien à votre baratin.*
- *Ben on doit maintenant gérer les transmissions radio, alors faut pas se gourer sur la modulation.*
- *Et relax les copains, c'est fastoche pour trancher la question !*

Les deux ingénieurs nous regardent stupéfaits que nous ayons la réponse sur ce point épineux.

- *OK chef chef, alors, c'est de la phase ou de la fréquence ?*
- *Bande de Dudules, allez donc voir les électroniciens en S12, c'est leur tasse de thé les UHF !*

➤ **Simuler un oscilloscope de réception radio.**

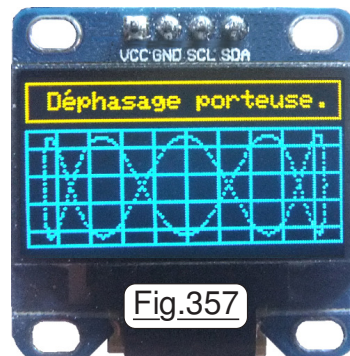
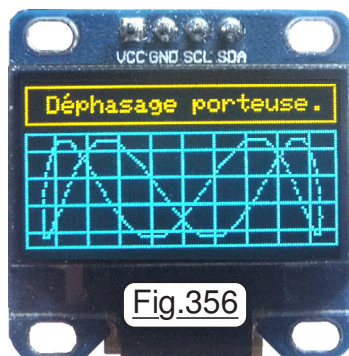
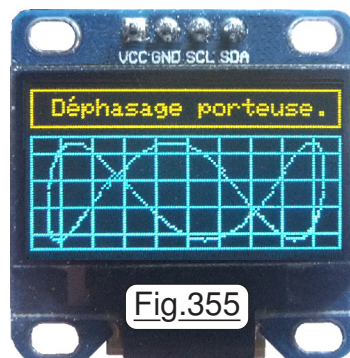
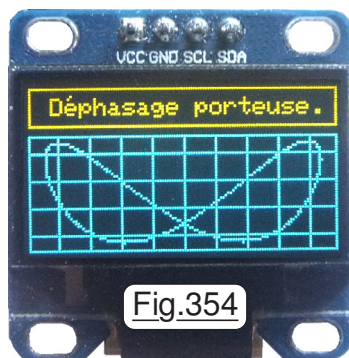
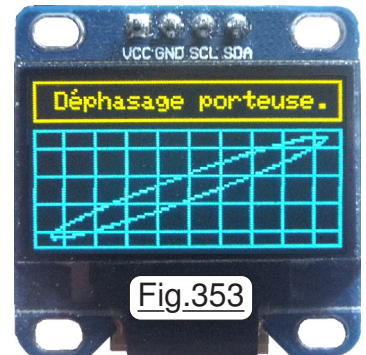
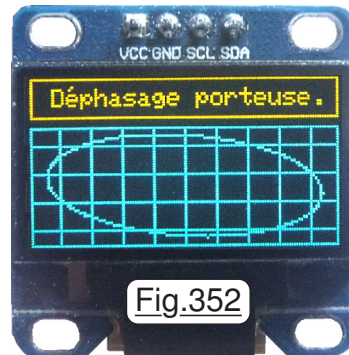
Agassif ces 2620 octets encore disponibles pour ajouter des fonctions au programme du pupitre. Aussi, dans ce chapitre nous allons nous amuser avec les fonctions graphiques. La surveillance des TELEMESURES est vitale dans la réalité astronautique, surtout quand on dialogue avec un vaisseau lointain et que les temps de propagation entre consignes montantes et accusés de réception descendants deviennent très important. Pour Mars on frise une heure entre les deux. Aussi, les stations de poursuite sont pourvues de tout un arsenal technique pour ne jamais risquer de perdre définitivement la liaison radio. Par exemple l'antenne de la sonde qui serait par erreur orientée en direction opposée à celle de la Terre. Les deux options émuloées dans ce chapitre sont purement fictives. On va juste s'amuser à créer du graphisme. La Fig.350 présente l'item qui validé dans le menu des **OPTIONS** permet de pénétrer dans la salle virtuelle des transmissions radio UHF. Pour pouvoir valider cette simulation il faut armer la **SÉCURITÉ** sous peine de se voir pénalisé d'un **E12**. Donc avant de cliquer sur **OUI** pensez à appuyer sur le **BPccr** pour allumer la LED rouge dédiée. La fonction ouvre sur la page des **TELEMESURES** montrée sur la Fig.351 qui représente l'écran d'un oscilloscope dont la grille permet à l'électronicien de mesurer en hauteur l'amplitude du signal. Il s'agit d'une sinusoïde si l'onde porteuse est propre et sans harmoniques. Le décalage horizontal symbolisé par la flèche verte correspond au déphasage progressif entre l'onde radio arrivant sur le canal descendant et une référence locale très précise en fréquence. Surtout, on constate que l'amplitude verticale symbolisée par les deux petites flèches rouges varie en permanence. C'est un phénomène qui était permanent en ondes courtes à l'époque des récepteurs de radio à "lampes". C'est ce que les techniciens nomment le "fading". L'évanouissement de la réception était à



cette époque principalement due au fait que l'onde radio arrivait sur l'antenne du récepteur après réflexion ionosphérique sur les couches mouvantes de la haute atmosphère. Dans le cas d'un satellite, s'il utilise des ondes polarisées, la mauvaise orientation de son antenne engendre une diminution du signal capté. Sur la simulation de la Fig.351 manifestement JEKERT tourne autour de l'axe central de son antenne de transmission des télémesures. Quand la perte d'amplitude est maximale, le signal reçu est encore largement suffisant. Il n'est donc pas nécessaire de réveiller la sonde et de transmettre des ordres pour stopper cette rotation. Il faut savoir qu'effectuer une manœuvre n'est engagé que si c'est indispensable. C'est long, car il faut préchauffer les canalisations des ergols et les tuyères des moteurs d'attitude. C'est couteux en énergie électrique, et on diminue la quantité des ergols disponibles dans les réservoirs. Aussi tant que l'évanouissement de l'onde porteuse UHF reste acceptable on laisse en l'état, quitte sur Terre à augmenter la sensibilité des récepteurs radio ou la taille des antennes de réception.

► Simuler un oscilloscope en utilisation de phasemètre.

Confortablement installé dans la salle des récepteurs radio de poursuite, c'est à dire que l'item de la Fig.350 a été validé, le **BPCCR** est ignoré ainsi que le codeur rotatif. En revanche le clavier est disponible pour plusieurs options. Vous avez déjà compris que la touche **FIN** sera impérativement à utiliser pour quitter la simulation de la station de poursuite. Mis à part la touche **OUI** réservée aux "commutations harmoniques", toutes les autres touches provoquent l'alternance entre **TELEMESURES** et le mode **Déphasage porteuse**. Ce mode assez particulier simule l'écran d'un oscilloscope dont la déviation verticale reçoit le signal de réception de la fréquence porteuse. La déviation horizontale est obtenue à partir d'un signal étalon ultra précis généré en local dans la station de poursuite. On obtient un 'O' qui se tortille jusqu'à devenir un segment de droite. Cette variation représente électroniquement le déphasage qui existe entre l'onde reçue sur les antennes et le signal étalon local. Il serait également possible de travailler sur des signaux harmoniques, c'est à dire que la fréquence de l'onde porteuse correspond à un multiple du signal assurant le balayage horizontal. Le dessin continue à se "tortiller" en fonction du déphasage entre les deux sinusoides. Toutefois, le nombre de lobes verticaux correspond au rapport des fréquences. C'est la touche **OUI** du clavier qui dans notre station d'écoute et de poursuite fictive est chargée de changer le rapport harmonique. Chaque fois que l'on clique sur ce bouton poussoir, on augmente



d'une unité le nombre de lobes "verticaux". Sur la Fig.354 le rapport des fréquences est de deux alors que sur la Fig.355 le signal vertical présente une fréquence triple de celle de l'étalon local. Sur la Fig.356 le rapport atteint quatre, alors que sur la Fig.357 il est de cinq. Un clic de plus sur **OUI** et l'on retrouve deux fréquences identiques. Cette façon pratique d'utiliser un oscilloscope a été imaginée par l'illustre marseillais LISSAJOUS. C'est un grand classique pour les électroniciens. Il faut avouer que ces figures qui se tortillent sont particulièrement artistiques. Le cinématographe en fait une consommation fréquente. Difficile de regarder un film de science fiction ou se passant dans un quelconque laboratoire sans que l'on n'y trouve un ou plusieurs écrans visualisant de telles courbes qui se dandinent. C'est toujours très spectaculaire ...

Un regard sur le listage du démonstrateur

P21P_Démonstrateur_Raquette.ino nous apprend que cette fantaisie purement graphique engloutit environ 1232 octets de programme soit 4% de l'espace disponible. On arrive enfin à un taux d'occupation de 94%, la rentabilité est au rendez-vous. Vous avez compris que ce gaspillage de zone mémoire n'est possible que dans la mesure où l'on peut dilapider impunément. Ceci-dit, il va falloir maintenant redevenir raisonnable car il ne reste plus que 1328 octets. Ce n'est pas énorme, car il est probable que la campagne de validation du logiciel ultime va probablement faire émerger des aléas de programme qu'il faudra corriger. En général, supprimer un "bug" consomme souvent moins de 150 octet, la marge est encore suffisante. Si pour une quelconque raison il nous fallait dégager une place importante, il serait toujours possible d'enlever cette fonction purement visuelle. J'avoue que je n'envisagerai cette mesure qu'en dernière limite. Ne soyons pas pessimistes. En fin d'historique on observe que quelques correctifs n'ont consommé que 28 octets. Donc corriger de petites erreurs n'est pas forcément glouton en octets de programme.

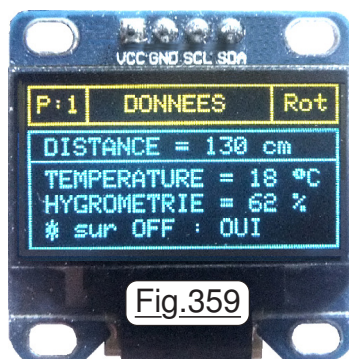
64) 19/03/2018 : Réveil de la centrale gyroscopique (MJD 58196)

Incident mineur sur JEKERT, hier les ingénieurs ont fait sortir de son hibernation le petit insecte mécanique. Quand les échanges de données ont été établis, les techniciens chargés de l'exploitation du vaisseau ont programmé une légère correction de trajectoire. Cette dernière implique forcément l'usage de la centrale gyroscopique pour orienter correctement la sonde avant l'allumage du moteur orbital. Ce dernier a fonctionné exactement durant les 2,35S prévues. Les radars de poursuite sont unanimes : La trajectoire ne correspond pas à ce qui était prévu. Rien de dramatique une autre correction inférieure à une seconde d'allumage sera suffisante pour corriger le tir. Toutefois, avant d'engager cette manœuvre il est capital de trouver l'origine de l'erreur de trajectoire.

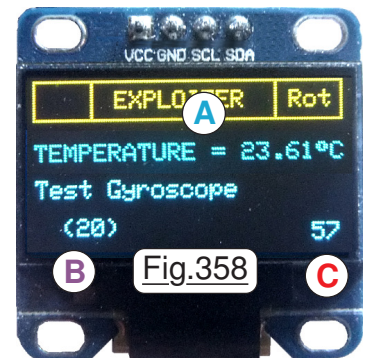
- Jour Tassin, jour Ferrando. zavez amené votre coffre fort en S4 ce matin ?
- Non grand chef, c'est une réplique du coffret dans lequel est enfermé l'IMU-6050 sur JEKERT.
- Ha bon, ya un prob avec la centrale ?
- Pas bien grave, mais on doit revoir son programme d'initialisation.
- Que se passe t'il ?
- Ben une correction de route a montré qu'elle a un petit PB thermique lors de l'initialisation.

➤ Une procédure d'initialisation digne de ce nom.

Avec l'ancienne version du programme, quand on passait rapidement de Calage GYRO. à l'affichage en continu des données de navigation en mode G, on constatait souvent que l'écart de route était notable ce qui ne peut s'expliquer par la rotation terrestre. C'était en réalité une dérive des références internes qui se produit quand on teste le bon fonctionnement de la centrale gyroscopique. En consultant les notices d'application du circuit intégré, on arrive à la conclusion que le MPU-6050 ne se stabilise pas vraiment avant que le circuit électronique n'atteigne les 22 degrés Celsius. La bibliothèque nous fournit un moyen de lire la valeur de la température mesurée directement "dans la puce de silicium". Aussi, la fonction

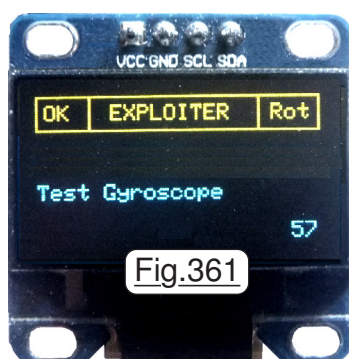


d'initialisation de la centrale gyroscopique par l'item Test Gyroscope du menu EXPLOITER est maintenant plus élaborée. Preuve en est dans la Fig.358 qui en A précise la température du circuit électronique interne au MPU-6050.

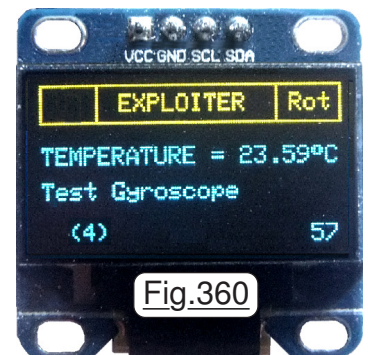


On peut remarquer la précision avec laquelle la sonde retourne cette information. On constate sur la Fig.359 que la température extérieure mesurée par le petit capteur météorologique est nettement inférieure. Pour s'assurer de l'initialisation correcte de la centrale gyroscopique, la procédure consiste à déclencher un compteur B. Toutes les secondes on

interroge la sonde sur la valeur de sa référence gyroscopique par la commande C. Quand le compteur B est initialisé à 20, on mémorise la valeur retournée par C. Chaque fois que la valeur retournée toutes les secondes est identiques à celle mémorisée, on décrémente le compteur B. Si la valeur change, on la mémorise à nouveau et l'on recommence le décomptage à 20. Quand la centrale gyroscopique est en régime stabilisé, le compteur va diminuer lentement sans être recyclé à son maximum. Si vingt mesures prouvent la stabilité, la procédure s'achève et rend la main au pupitre en retournant l'accusé de réception attendu OK. Sur



la Fig.360 si la valeur de l'ACR ne change pas, dans quatre secondes la centrale gyroscopique sera initialisée. C'est ce que montre la Fig.361 la température est effacée ainsi que le chronomètre de décomptage. Pour ceux qui vont se poser la question, c'est l'instruction de code 90 qui permet de générer une temporisation d'une seconde qui a été modifiée pour pouvoir interroger la sonde, car il n'y a plus de codes disponibles. Quand un programme enregistré en EEPROM est en cours d'exécution, l'instruction 90 ne retourne pas la valeur de la température ce qui bloquerait irrémédiablement le processus, l'ACR étant dans ce mode limité à '*'.



65) La dérive gyroscopique : Complément indispensable.

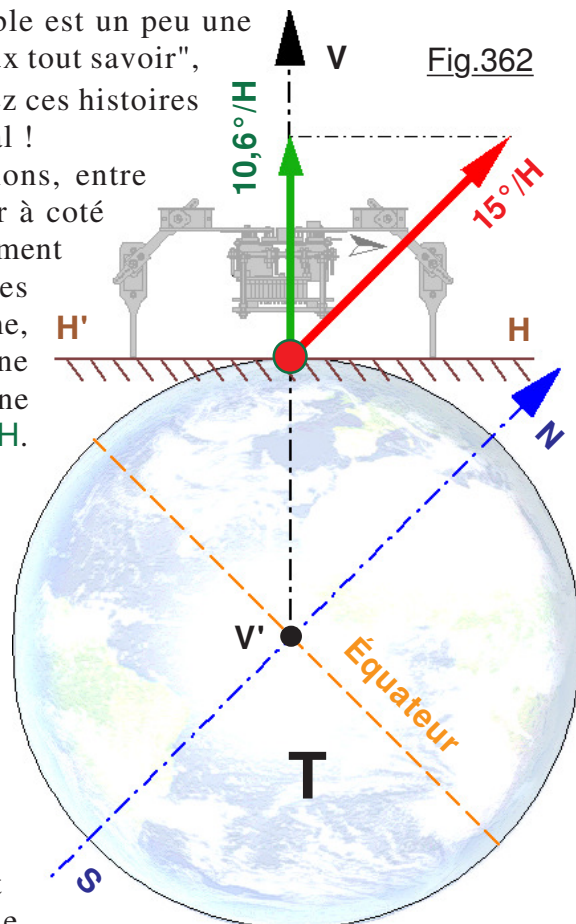
F ranchement, qualifier ce chapitre d'indispensable est un peu une escroquerie. Si vous n'êtes pas adepte de "je veux tout savoir",

prenez votre chemin, ignorez ces bavardages et oubliez ces histoires de dérive gyroscopique. La Terre tourne ... point final !

Néanmoins, quand on aborde le chapitre des rotations, entre l'évidence et le "ça se voit", on risque fort de passer à côté d'une réalité qui n'a rien de naturel, surtout si le mouvement est abordé "vectoriellement". Quand l'affichage des données de navigation a été introduit dans le programme, en référence gyroscopique je m'attendais non pas à une rotation de **360° par jour**, soit **15° par heures**, mais une variation angulaire de $15^\circ / (\sqrt{2})$ soit environ **10,6°/H**.

Explications relatives à cette hypothèse :

Considérons la Fig.362 sur laquelle est symbolisée la Terre **T** avec son axe de rotation Sud/Nord repéré par le vecteur **SN**. Par rapport à l'univers elle tourne à la vitesse de **15°/H**, mouvement représenté par le vecteur rouge. Pour simplifier, sur ce dessin la sonde est supposée posée sur le sol au 45° de parallèle Nord. Le sol est donc pour l'observateur perpendiculaire à la verticale locale **V'V** soit de direction **H'H**. Hors le LACET pour JEKERT est orienté dans la direction **V'V**. On devait donc s'attendre à ce que seule la composante verticale de la rotation terrestre soit enregistrée par le MPU-6050. Cette projection vectorielle de **15°/H** sur **V'V** donne le **10,6°/H** représentés par le vecteur vert.



P our mettre en évidence ce phénomène, le démonstrateur **P70_Derive_Gyroscopique.ino** a été rédigé et téléversé sur l'ATmega328 du pupitre, pour enregistrer sur une période de 24 heures cette rotation. La mise en œuvre de ce démonstrateur de dernière minute est décrite en détail dans le chapitre **Enregistrer la dérive gyroscopique** du livret **DOSSIER TECHNIQUE**.

La Fig.363 donne l'une des pages de données enregistrées. Entre chaque valeur il s'est écoulé une période de dix minutes. En théorie on devrait avoir entre chaque échantillon un écart de 0.166° correspondant aux **10,6°/H**. Comme les décimales ne sont pas affichées, on s'attendrait à une alternance de 1°, 2°, 1° etc. Hors l'écart fait 2°, 4°, 3°, 2°, 3° etc. Voir fluctuer à $\pm 2^\circ$ est normal puisque tout affichage numérique est donné à \pm une unité. Pour éliminer ce problème

Bloc n°2 / 9				E
044	046	050	053	
055	058	061	064	
067	070	073	076	
077	079	081	083	

Fig.363

Bloc n°5 / 9				E
163	166	168	170	
180	186	188	184	00
180	186	188	184	00
160	163	160	161	00

Fig.364

d'affichage, on utilise le bloc de données n°5 montré sur la Fig.364 dont les valeurs négatives sont barrées en rouge. Avant ce bloc il y a $4 \times 16 = 64$ échantillons. L'échantillon encadré en rose est donc le n°68. Il s'est écoulé 68 fois dix minutes. Donc pour **170°** de rotation enregistrés, une période de 680 minutes s'est écoulée. La vitesse de rotation enregistrée est donc de $170^\circ / 680 = 0,25^\circ/\text{min}$ soit exactement **15°/H**.

CONCLUSION : Le fonctionnement interne du PMU-6050 est bien

plus subtil qu'il n'y paraît au premier abord. Pour calculer la valeur de la variation angulaire, nous n'utilisons pas les projections sur les angles d'Euler, mais la projection sur le plan perpendiculaire au LACET du vecteur pesanteur. C'est le changement d'orientation de cette projection qui semble prise en compte et non une intégration temps réel du gradient de rotation du gyroscope. Peu importe, **retenons que l'indication d'orientation en référence Gyroscopique correspond à la rotation en LACET, affectée de la rotation terrestre de 15°/H. Comme la dérive est constante, il est donc possible de compenser par logiciel ...**

66) 26/03/2018 : Les versions ultimes des logiciels (MJD 58203)

C'est la fête en salle S12 qui pour la circonstance a été décorée. Les établis des électroniciens ont été recouverts par de grandes nappes en papier blanches. Boissons et comestibles encombrant joyeusement ces tables improvisées. Les conversations vont bon train. Globalement le projet est à son terme. Les consoles ont quitté les infrastructures pour être réparties tout le tour de la Terre dans les stations de poursuite. JEKERT a fait son petit bonhomme de chemin. Attirée dans la sphère d'influence de Mars elle a accéléré inexorablement. Quand elle est passée au périégée, tout était conforme. Le moteur orbital a été allumé pendant plus de huit minutes pour provoquer le freinage de capture. Elle orbite sagement autour de la planète rouge. Les radars terrestres confirment que son orbite actuelle est tout à fait conforme aux prévisions. Quand les caractéristiques orbitales auront été affinées, on provoquera le dernier freinage qui engagera sa descente vers le sol ocre et poussiéreux. Tout le monde est confiant, il n'y a aucune raison pour que cette ultime manœuvre ne se passe pas correctement. Tout va bien à bord, les signaux envoyés de si loin sont reçus 5/5 ...

➤ P40 et P50 les programmes d'exploitation.

Contrairement au domaine commercial où un programme est mis en vente dès que l'on estime qu'il fonctionne "presque bien", dans le cadre des loisirs nous savons tous qu'un logiciel n'est jamais terminé. Les idées foisonnent, la passion pousse au "toujours plus". Il faut toutefois bien arriver à considérer à un moment ou à un autre qu'il est temps de passer à autre chose. La porte n'est pas fermée, on y reviendra quand nous serons titillé par le démon de la programmation. Avec 96% d'occupation de la zone programme pour P40 et 97% pour le logiciel de la sonde P50, nous pouvons considérer que les deux ATmega328 sont largement exploités avec leurs EEPROMs respectives bien saturées. Il reste juste de quoi corriger d'éventuelles "vermines" qui vont forcément venir assombrir ce tableau idyllique. Les deux logiciels P40_PGM_RAQUETTE_MAITRE.ino et P50_PGM_ESCLAVE_SONDE.ino sont les versions d'exploitation, deux complices intimement liés par un cordon ombilical très bavard. Dans leurs versions actuelles ils sont presque directement issus des démonstrateurs P21P et P22N. Différence de base pour leurs versions du 08/04/2018 : Le début du listage source comporte en remarque l'historique de leurs élaborations. À partir de maintenant, ce ne sont plus des démonstrateurs. Les seules modifications qui viendront apporter des changements auront pour but de corriger des problèmes qui n'ont pas été détectés durant le développement. Avant de considérer ces deux compères comme étant fiables, une campagne de validation sévère est indispensable ... *mais elle ne sera possible que lorsque le pupitre* ne sera plus un gros paquet de fils électriques reliant des plaques à essais, mais *sera un véritable petit boîtier bien esthétique et fonctionnel*. Il reste encore pas mal de pain sur la planche car nous devons réaliser un clavier, définir les alimentations autonomes et créer le coffret. La saga n'est donc pas terminée, car un TOME 7 viendra compléter les six premiers ouvrages. Il sera consacré à la réalisation pratique du pupitre. *(Comme toujours, dans la mesure où l'on parle au futur, nous restons tributaires de la vie. Ceci étant précisé, il n'y a pour le moment strictement aucune raison de ne pas y croire.)*

➤ P60 le clone EEPROM.

Comme précisé en tête du source de P50_PGM_ESCLAVE_SONDE.ino avant de téléverser le programme il faut au préalable inscrire les postures en EEPROM. En version ultime c'est l'utilitaire de servitude P60_Clone_EEPROM_sonde.ino qu'il faut téléverser. Outre les tables de consignes indispensables au bon fonctionnement de la sonde, ce logiciel inscrit en EEPROM un spectre colorimétrique cohérent, un balayage ultrasons, efface les trois premiers programmes et en fournit six décrits dans le manuel d'utilisation de JEKERT. Enfin, pour éviter tout risque de divergence des servomoteurs, les huit postures enregistrées sont cohérentes. *(Également décrites dans le manuel d'exploitation de la petite machine.)* Pour résumer, avec ce logiciel de servitude, la petite sonde présentera un comportement cohérent quelles que soient les manipulations expérimentales engagées pour apprendre à s'en servir. *(Affirmer aussi péremptoirement que tout va bien fonctionner est assez pédant ... car ça suppose que les logiciels fournis sont totalement exempts de vermines !)* Compte tenu du nombre considérable de fonctionnalités, un manuel qui fournit les protocoles optimisés accompagne ce TOME 6 et résume globalement les divers menus d'exploitation à partir du pupitre ... supposé achevé. Avant de le décrire, ouvrons une parenthèse :

➤ Changements significatifs apportés à P40.

Réputé dériver directement de P21P il comporte plusieurs modifications pas forcément dérisoires qui ont été apportées durant la rédaction du manuel. Cette dernière imposait de manipuler en détails, pour faire émerger des protocoles optimisés. Ces expérimentations ont fait remarquer des petites erreurs, des complications inutiles, et surtout incité à ajouter quelques options au programme d'exploitation du pupitre de pilotage. L'historique situé en tête du listage source précise les modifications effectuées et leurs impacts sur la taille du programme de la raquette ainsi que celle du logiciel de la sonde qui a aussi bénéficié de changements notables. Dans ce chapitre il ne me semble pas très utile d'énumérer les correctifs de "bugs", seules les nouvelles options sont étudiées.

- **Historique des touches utilisées sur le clavier.**

Consultez le manuel d'utilisation de JEKERT en page 19 pour avoir les détails de cette petite fonction technique qui peut s'avérer utile en modifications logicielles. Si le comportement du programme n'est pas celui attendu, et que l'on a un doute sur la dernière touche cliquée sur le clavier, pour lever l'incertitude dans une telle situation, le programme comporte en permanence la chronologie des dix dernières touches utilisée, y compris le **BPCCR**. Pour ouvrir la page de visualisation de "l'historique clavier", cliquez sur **EXPLOITER** puis sur la touche de mouvement élémentaire vers l'avant **↑**. Bien que prévu pour des actions de programmation, dans l'hypothèse de pilotage d'une sonde réelle, on se doute que les consoles de maîtrise permettent à tout moment d'afficher la liste des actions effectuées par les techniciens. Ainsi, si un incident se produit, l'historique facilite la recherche de la cause et l'analyse des actions correctrices à entreprendre. Cette option va donc dans le sens du réalisme et accompagne notre petit délire astronautique.

- **Mode Directionnel en affichage des données de navigation.**

Quand on utilise l'affichage permanent des données de navigation en mode graphique, maintenant le **BPCCR** permet d'alterner entre quatre modes de références pour afficher les données de navigation : Cap Magnétique, écart de route en gisement par l'option **Directionnelle**, référence interne **Gyroscopique** et dérive gyroscopique **Compensée**. La référence **Directionnelle** est en réalité la plus naturelle à utiliser. Dans ce mode l'avant de la sonde est située en haut de la rose des CAPs. La donnée **D** précise la valeur de la direction désirée. (*CAP magnétique souhaité.*) Il n'y a plus besoin d'afficher le "bug". L'aiguille qui tourne indique la *direction dans laquelle se trouve le CAP désiré par rapport à l'axe longitudinal du vaisseau*. Pour corriger la route il suffit de faire tourner JEKERT du côté où se trouve l'aiguille. Aiguille à gauche on tourne vers bâbord, aiguille à droite on tourne vers tribord ... c'est immédiat.

- **Mode gyroscopique avec Compensation de la rotation terrestre.**

Pour conserver la direction d'un déplacement, c'est presque le mode d'affichage le plus fiable car nous ne sommes plus tributaires des déviations magnétiques et de la perturbation apportée par les moteurs. On commence par orienter la sonde en mode **Directionnel**. Puis, quand JEKERT est parfaitement orientée, on active le mode gyroscopique **Compensé**. Il suffit alors de tourner de façon à toujours annuler l'**E**carter de route affiché en bas de la page écran de navigation. La compensation par calcul de la rotation terrestre exagère de +3° en dix heures trente minutes de temps, (*Non compensé on serait à -157°*) c'est à dire à environ 1/3° par heure, autant dire rien du tout, et ce d'autant plus qu'il est peu probable que vous laissiez immobile JEKERT sur une période aussi importante. Le plus fort, c'est que sur le pupitre cette nouvelle option ne consomme au total que 48 octets : Une aubaine.


- **Variation d'amplitude sur les figures de LISSAJOUS.**

Pour une petite dépense de 38 octets tout compris, on peut s'offrir en promotion une simulation du "fading" sur l'onde porteuse qui arrive de JEKERT. En effet, les figures de LISSAJOUS sont supposées avoir en balayage horizontal un signal étalon local très précis, et en vertical l'onde radio qui arrive de Mars. Cette onde subit des phénomènes d'évanouissement comme le montre l'option **TELEMESURES**. En toute logique, l'amplitude verticale des figures de LISSAJOUS étant supposée issue de l'onde porteuse captée par les antennes de poursuite, elle doit forcément subir les diminutions temporaires de niveau de réception. C'est maintenant programmé, conduisant à une simulation plus réaliste. À ce stade il ne restait plus que 360 octets de disponibles pour pouvoir modifier **P40**. Heureusement que le logiciel a été bien simplifié ...

- **Simplifications des menus, allègement des protocoles, optimisations logicielles.**

Force est de constater à l'usage qu'un certain nombre d'items dans les menus étaient inutiles. Par exemple quand on désirait afficher les données de navigation en continu sous forme graphique et en référence gyroscopique, la procédure était bien trop lourde car mal pensée. Par exemple il fallait passer par le menu des **DONNEES** pour allumer une LED bleue qui signalait le mode graphique. Hors, à l'usage le mode graphique est tellement plus convivial que l'option purement numérique, que cette dernière n'était plus du tout justifiée. Et ce d'autant plus qu'avec le petit tableau de bord graphique on continue d'afficher les valeurs en numérique. L'affichage numérique des données a donc été purement et simplement éliminé faisant gagner une place substantielle pour les programmes. Du coup, étant forcément en affichage graphique, la LED bleue n'est plus utile, et le **BPccr** dans le menu des **DONNEES** devient disponible. Il permet maintenant d'afficher à notre convenance la température interne de la centrale inertielle. (*Voir le manuel en bas de la page 28.*) La broche **A3** devient disponible. Afficher les données de navigation au format numérique dans le menu des **DONNEES** a été enlevé car ne présente aucun intérêt. Du coup on a récupéré pas mal de place en EEPROM, maintenant tous les textes du pupitre y sont logés. Enfin, actuellement le recalage du gyroscope quand on active les références Gyroscopique et Rotation terrestre Compensée est automatique au moment où l'on valide l'option. L'item **Calage Gyro.** n'est plus du tout justifié. Enlevé du menu **EXPLOITER** les protocoles de pilotage sont bien plus simples, on gagne du code programme et on enlève des textes en EEPROM. Le code consigne **60** a été récupéré pour pouvoir afficher à volonté la température interne de l'IMU-6050. Toutes ces modifications peuvent sembler secondaires, à l'usage elles améliorent notablement la qualité opérationnelle du pupitre. Le manuel d'utilisation de JEKERT décrit dans le chapitre suivant tient compte de tous ces changements.

- **Chronométrage pour le rechargement des accumulateurs de puissance.**

Dans le **TOME 7** nous allons voir que finalement j'ai "craqué" pour une alimentation totalement autonome. Si vous effectuez un RESET sur le pupitre, la LED jaune clignote rapidement pour nous inciter à cliquer sur une touche du clavier. Toutes les touches font passer au menu **EXPLOITER** sauf une : La touche de translation vers l'avant. Si vous cliquez sur cette touche  alors s'affiche à l'écran une sorte de chronomètre. Le **TOME 7** sera accompagné d'un petit livret nommé **DOSSIER TECHNIQUE** qui contiendra une foule d'informations qui ne sont pas directement liées à l'utilisation de la sonde en tant que machine supposée posée sur le sol martien, mais qui sont indispensables à assurer la maintenance électrique, informatique et électronique de la sonde et du pupitre. (*Schémas électroniques, dessin des circuits imprimés, organigrammes et informations sur les logiciels ...*) Un chapitre particulier nommé **Protocole pour recharger les accumulateurs** est réservé au rechargement de la pile rechargeable 6Vcc et décrit l'usage de ce chronomètre. Quand on valide le chronomètre de recharge accumulateur, l'écran OLED présente deux informations : En **A** est affiché un décompte qui précise la durée restante exprimée en minutes avant la fin du rechargement complet des accumulateurs de puissance 6Vcc. En **B** se trouve un chronomètre partiel qui décompte par minutes.



- **Utilisation de la LED bleue sur la sonde pour témoigner de la stabilisation gyroscopique.**
Lors des évolutions logicielles, la LED bleue qui sur la sonde indiquait un calage du gyroscope de la centrale MPU-6050 n'est plus utilisée. Elle est donc devenue disponible. Le programme d'exploitation **P50_PGM_ESCLAVE_SONDE.ino** a donc été modifié pour s'en servir et signaler que le mode Stabilisation Gyroscopique est en cours. Maintenant la LED verte qui témoigne du fonctionnement de la boucle de base clignote à une fréquence régulière de 5Hz.
- **Utilisation de la LED rouge sur la sonde pour indiquer le mode pilotage manuel.**
Lors des modifications logicielles, la LED rouge sur la sonde pilotée par **S15** n'était plus utilisée. Le programme d'exploitation **P50_PGM_ESCLAVE_SONDE.ino** a donc été modifié pour s'en et signaler comme envisagé à l'origine que le mode pilotage manuel est en cours.

67) Un petit manuel d'utilisation de la sonde JEKERT.

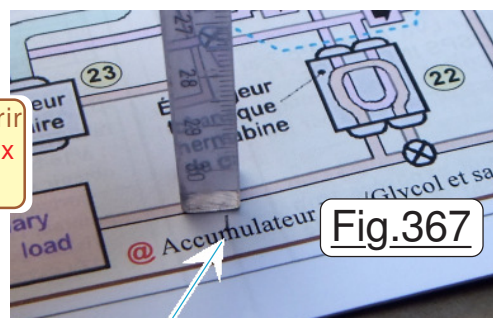
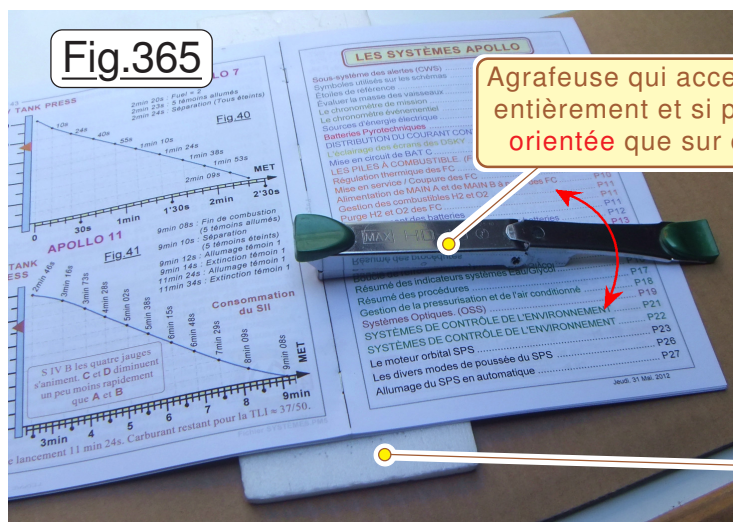
Compte tenu du nombre considérable de fonctions et options disponibles pour piloter JEKERT, il me semble impossible de se remémorer toutes les commandes sur le long terme, ainsi que le comportement attendu des programmes. Par ailleurs, certaines consignes peuvent engager des risques matériels si des conditions initiales ne sont pas respectées. Enfin, certaines manipulations imposent une suite notable d'instructions qui doivent s'enchaîner dans un ordre strict. Avoir des protocoles optimisés sera forcément un atout important. Encore faut-il que le manuel d'exploitation de la petite machine soit commode et convivial. Ce chapitre est prévu pour vous donner quelques détails sur la conception du livret et la façon rationnelle de s'en servir.

➤ Réalisation matérielle du livret d'exploitation de JEKERT.

Agencé à un format A5, les faibles dimensions de ce manuel en font un document parfaitement adapté à son usage. Pas trop petit, les dessins et schémas sont de dimensions suffisantes, pas trop gros, il trouve facilement sa place dans le coffret de rangement du petit robot et n'encombre pas exagérément la table lors des démonstrations. Le fichier **Manuel d'UTILISATION.pdf** est prévu pour imprimer RECTO/VERSO. Il importe donc de choisir du papier d'épaisseur "normale" pour ne pas que l'encre ne traverse. **Papier recyclé méga écolo** s'abstenir ! Personnellement je commence par imprimer toutes les pages impaires. Puis, paquet de feuilles replacé sur le bac à papier de la machine **CORRECTEMENT ORIENTÉ ET DANS LE BON ORDRE** je fais imprimer toutes les pages paires. Pour cette phase il me semble moins risqué d'opérer page par page, et vérifier à chaque tentative que deux A4 n'ont pas été "aspirés" par le mécanisme qui tracte les feuilles sous les têtes d'impression. Vous ne perdrez ainsi qu'une seule épreuve, alors que si vous engagez l'opération pour les huit feuilles ... c'est tout le paquet qu'il faudra entièrement réimprimer. Bien réfléchir quand on replace le paquet dans le bac de la machine, car les pièges sont nombreux. (*Inverser le haut et le bas, face sur le dessus qui n'est pas la bonne, pages entassées dans l'ordre incorrect ...*)

Puis, quand tout est imprimé, réaliser l'assemblage est relativement élémentaire :

- 1) Commencez par plier toutes les pages en deux. (*Et si possible du bon coté !*)
- 2) Trouvez une plaque de polystyrène ou du carton bien classique. (*Voir Fig.365*)
- 3) Positionnez les pages bien à plat et surtout bien les unes cadrées sur les autres.
- 4) Avec une petite agrafeuse qui accepte de s'ouvrir complètement mettre en place quatre "crochets".
ATTENTION : Quand on appuie sur l'agrafeuse il faut bien la tenir latéralement car elle veut se décaler sur les cotés. Du coup comme montré en Fig.366 l'agrafe est mal enfoncée et se plie. Quand une agrafe s'est tordue, la retirer avec un cutter et en placer une deuxième exactement au même endroit. Le deuxième essai sera le bon ...



- 5) Retourner le livret **sur un support rigide** et fermer les agrafes à la main avec un outil quelconque. Dans mon cas j'utilise une règle de section carrée comme montré sur la Fig.367 sur laquelle à peine visible on voit un coté de l'agrafe non encore replié.

Notez que pour faciliter la tâche les pages sont numérotées verticalement au centre pour repérer plus facilement l'ordre d'assemblage. Gutembérisez bien les amis ...

► Utilisation rationnelle du livret d'exploitation de JEKERT.

Après quelques tâtonnements et pas mal de "gros mots", vous avez enfin réalisé avec amour le petit manuel. Votre satisfaction est grande et c'est parfaitement mérité. Bien que se servir d'un document technique reste globalement évident, quelques petits détails peuvent vous faciliter les manipulations. Pour en tirer le meilleur parti, il me semble utile d'avoir à l'esprit certaines informations clef sur son agencement.

- Systématiquement le livret a tendance à s'ouvrir en son milieu. C'est la raison pour laquelle on trouve sur les pages 16 et 17 la liste des commandes pour les menus principaux. Tournez la page 17 et l'on trouve le menu des **POSTURES**.
- Chaque menu est présenté sous la forme d'un tableau séparé en deux zones. La partie colorée en jaune correspond aux commandes "de sens positif" (*Flèches dirigées vers le bas*) alors que la zone du bas concerne les commandes de sens rétrograde. (*Flèches orientées vers le haut*). Quand on ouvre un menu avec la touche clavier qui lui est affectée, on arrive à la première commande, celle qui est tout en haut dans la liste. Si la commande que vous désirez est en zone jaune, le plus rapide pour y arriver consiste à tourner le codeur rotatif dans le sens horaire. Si la commande est dans la zone des flèches orientées vers le haut il sera plus astucieux de tourner le bouton en sens antihoraire. Par exemple vous désirez l'item **Tir LASER ?** du menu **EXPLOITER**. Le plus rapide consiste à effectuer six indexages en sens antihoraire pour aboutir à cette fonction.
- S'ils sont concernés, les items sont complétés par des repères du genre **(S)**, **(F)** etc. Ces indications sont précieuses car elles précisent si la commande exige la **SÉCURITÉ**, s'achève par **FIN** ...
- Les divers protocoles sont optimisés. Ils tiennent compte des informations qui précèdent, assurent la sécurité pour la machine et peuvent vous éviter bien des erreurs de manipulation. Je ne peux que vous conseiller fortement de les prendre en considération.
- La couverture arrière du livret précise la signification des codes d'erreur. Pas besoin de chercher dans le livret pour savoir quel est l'origine d'un BIP d'alerte, il suffit de tourner le manuel. Quand on transmet des commandes, on peut avoir envie de savoir à quoi correspond un code de consigne. C'est également le cas lorsque l'on désire relire un programme enregistré. Faciles à trouver, tous les codes de consignes sont placés à la fin du livret, pas besoin de consulter la table des matières.
- La page 2 regroupe un certain nombre d'informations à ne pas oublier. Surtout, la page 3 est relative à la mise en service de la sonde et aux procédures de fin d'activité. Pour l'exploitation globale de la petite machine, la table des matières est placée en couverture pour pouvoir la consulter directement sans avoir à feuilleter.

Enfin, pour tout ce qui concerne le matériel et la maintenance, prenez patience, si tout va bien un livret technique de conception analogue au manuel d'utilisation sera fourni avec le **TOME 7**.

Chères lectrices, chers lecteurs, avec ce didacticiel, nous avons abordé des domaines absolument passionnants dans la programmation d'Arduino. Nous avons réalisé un dialogue entre deux machines, poussé assez loin l'apprentissage, titillé la muse pour réaliser des dessins, conçu un tableau de bord que l'on peut qualifier de rationnel et étés confrontés à une collision de PILE. Informatiquement si vous avez globalement cerné les programmes démonstrateurs, vous êtes désormais bien armés pour envisager la conception de machines plus importantes. Rigueur et méthodes seront vos plus précieuses alliées.

Ben Mâmôa je ne trouve que des pages paires et en triple dans mon manuel. C'est normal ça ?

*Je vous souhaite à toutes et à tous agréable lecture et ... à bientôt dans le TOME 7.
Amicalement : Nulentout.*