

Machine de TURING Élémentaire.

Par Nulentout : Vendredi 11 Novembre 2022.

Soyons honnête : Ce petit projet n'a vraiment été motivé que par le plaisir de programmer. Quand la machine de TURING décrite sur ce site (1) a été achevée et le didacticiel mis en ligne, je me suis retrouvé un peu tristounet, comme c'est souvent le cas quand on vient de passer de très agréables vacances et qu'il faut reprendre le joug. Une sorte de "vide", car ce n'est pas les activités ludiques qui me manquent. Mais ... la programmation est une drogue, une activité qui oblige à faire des efforts intellectuels sans lesquels, à mon âge le cerveau se rouille inexorablement. Bref, je cherchais un prétexte pour rebrancher un Arduino sur le P.C, et des raisons de me creuser les méninges.

(1) <https://www.robot-maker.com/ouvrages/machine-de-turing-tome1-presentation/>

Prototype achevé, la petite machine matérielle a été capable de dérouler correctement plus de cinquante algorithmes. Mais leur mise au point a été assez laborieuse et exigé de réaliser bien des feuilles perforées qui ont terminé dans la poubelle. Par ailleurs, compte tenu de la lenteur de fonctionnement de cette machine électromagnétique, il est assez peu raisonnable d'activer un programme qui a fait ses preuves sur des données comportant un nombre de BITS important. Du coup, bien que le carrousel simule 56 BITS, dans la pratique on se limite à une dizaine dans les données, rarement plus. Aussi, pouvoir soumettre à l'algorithme des données "étendues" sans avoir à attendre des lustres est un avantage incontestable.

Par ailleurs, la vérification "automatique" d'un programme avant d'envisager de faire une feuille perforée pour la machine matérielle est vraiment très agréable. Comme vous pouvez le constater dans le didacticiel cité dans le chapitre précédent, la simulation d'un déroulement de programme peut se faire avec la Machine de Turing "version papier". Mais cette méthode est très fastidieuse dès que le nombre de cycles d'horloge devient important, sans compter que dans ce cas le risque de se tromper augmente de façon significative. Enfin, l'analyseur syntaxique durant la saisie des instructions nous signale plusieurs types d'erreurs ou d'illogismes. On peut ainsi corriger le programme en temps réel. Comme c'est le cas sur le prototype matériel, on peut configurer un mode PAS à PAS, avec plusieurs autres options de comportement bien utiles. Pour clore cette introduction, on peut aussi mentionner la faculté de conserver dans la mémoire EEPROM du microcontrôleur ATmega328 jusqu'à 14 feuilles perforée virtuelles et une configuration de BARILLET.

Youppiiiiiii, le voilà le prétexte pour tortiller du code C++, et y consommer des heures par paquet de beaucoup ! Une idée toute simple qui consiste à frapper du texte sur un clavier d'ordinateur, puis de le valider. Et prouitchhh, un programme avale et gloutonnement ce verbiage et se transforme en une Machine de Turing virtuelle dialoguant de façon classique avec le **Moniteur de l'IDE**. Deux unités sont envisagées. Dans ce descriptif il s'agit de la version "du pauvre" nommée **Élémentaire** car elle n'impose pratiquement aucune soudure, sauf si on désire l'intégrer dans un petit coffret tel que celui de la Fig.1 ou l'équiper du Bruiteur sonore. La carte NANO branchée seule sur le P.C. par sa ligne USB est suffisante. *Seule contre-partie, elle impose l'utilisation d'un ordinateur pour fonctionner*, contrairement à l'unité autonome qui elle impose un travail de soudage et d'assemblage relativement conséquent et un certain coût.

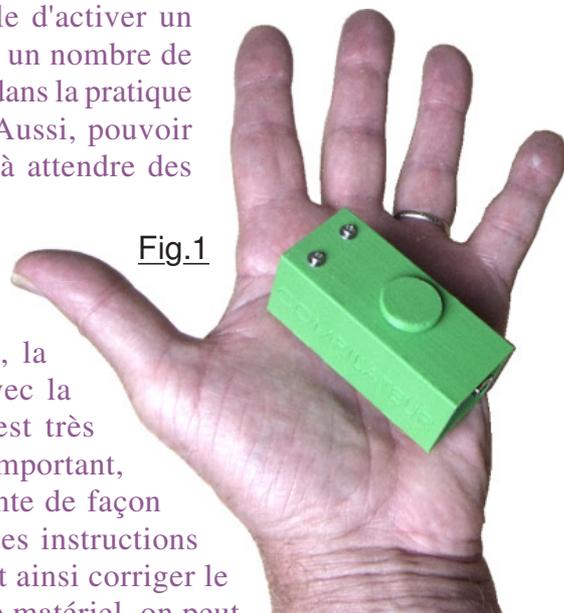


Fig.1



T'as vu ce container marin de la société Machine de Turing ? Ils doivent être très écolo, car il est tout vert leur conteneur.

1) Utilisation concrète de cette Machine de Turing.

Presque logiquement, j'aurais été tenté de commencer par la description détaillée de la petite carte électronique, puis du coffret, et enfin de la façon de s'y prendre pour téléverser les données dans la mémoire non volatile de l'ATmega328. La carte étant disponible, nous aurions alors passé en revue les menus d'exploitation et l'usage de cette petite merveille. Au final, il m'a semblé plus pertinent de vous donner envie de concrétiser cette réalisation "modeste" avant de plonger dans les détails techniques et toujours "trop encombrant" de la réalisation proprement dite. Considérons donc, que comme montré sur la Fig.2 nous avons le petit coffret vert réuni au P.C. par sa ligne série USB. On suppose également qu'une version de l'environnement **IDE** d'Arduino est présente sur l'ordinateur pour disposer de son **Moniteur** qui assurera le dialogue entre l'opérateur et la carte microcontrôleur. (*Le chapitre 12 page 39 explique en détails comment procéder pour les débutants.*)

➤ **Protocole d'utilisation du système.**

Dialogue Homme / machine oblige, on va utiliser dans notre cas un ordinateur de bureau ou un P.C. portable, ce qui en Fig.2 est le cas. On a branché sur la mini prise USB de la carte Arduino NANO en **1** la fiche **2**. La ligne USB **3** est à son tour branchée sur l'une des prises de l'ordinateur **4**. En **5** on a activé l'**IDE** qui a ouvert sa fenêtre contextuelle. En cliquant en **6** on invoque

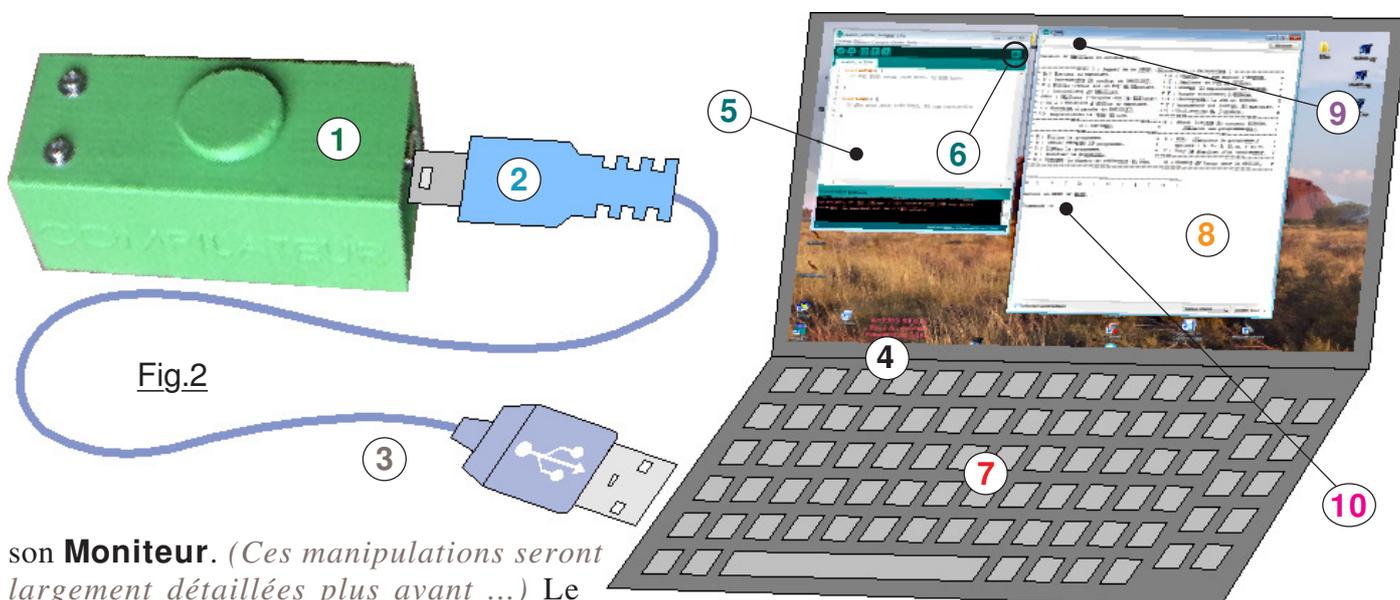


Fig.2

son **Moniteur**. (*Ces manipulations seront largement détaillées plus avant ...*) Le

moniteur ouvre à son tour sa fenêtre contextuelle **8**. Pour peu que les vitesses d'échanges d'informations sur la ligne série sont les mêmes sur le P.C. et sur Arduino, apparaît alors le **MENU de base**. Dans la petite fenêtre de saisie **9** on visualise les commandes frappées sur le clavier **7**. Le résultat est alors retourné en **10** établissant ainsi de façon très simple "le dialogue Homme / Machine".

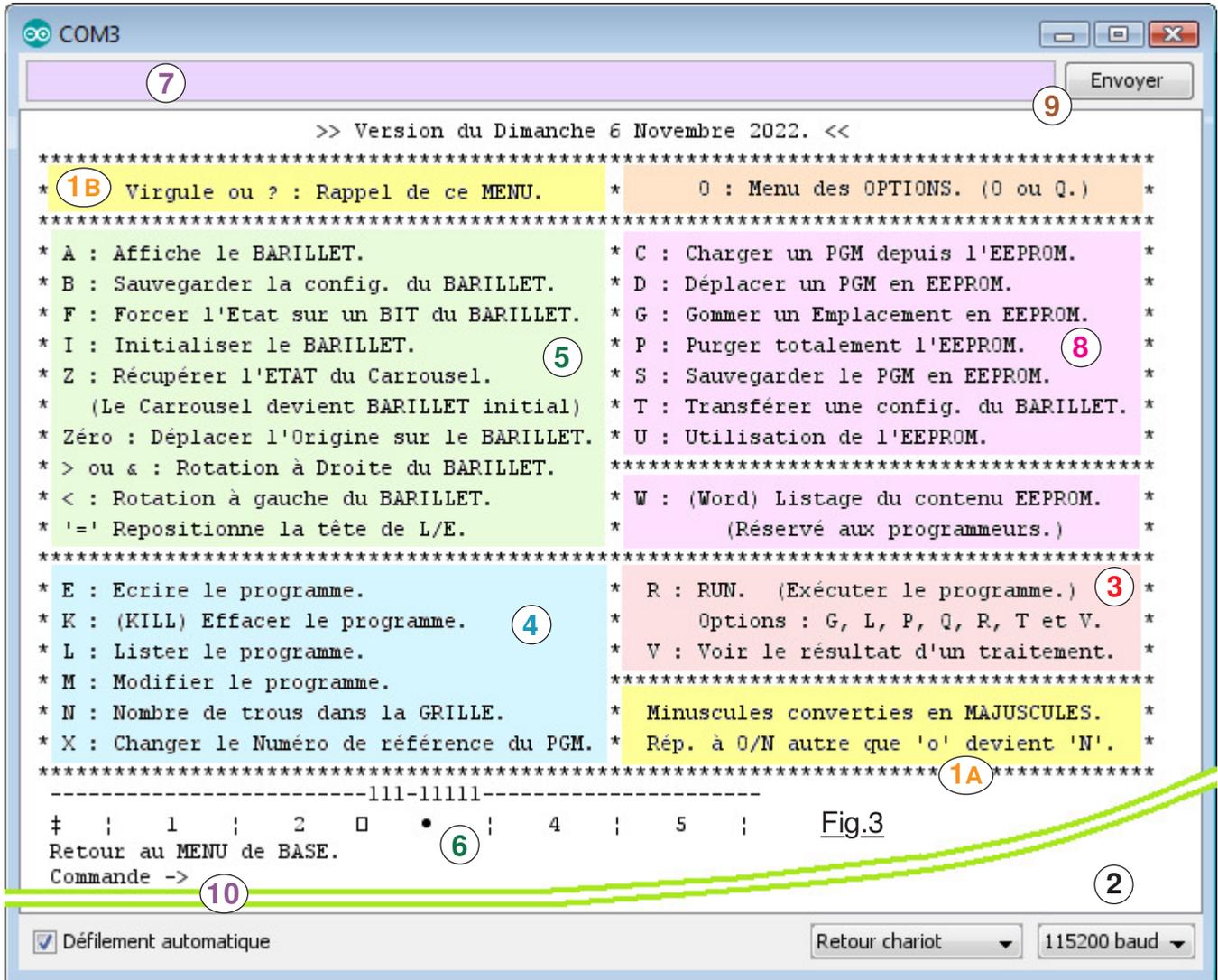
➤ **Avantages principaux de la version électronique.**

Avant d'aborder les très nombreuses commandes disponibles, passons en revue les particularités spécifiques à cette petite réalisation qui n'impose strictement aucune soudure. Si vous acceptez de ne pas avoir le petit bruiteur et de la LED bleue, il suffit de brancher la carte Arduino NANO sur le P.C. et vous possédez déjà une Machine de Turing très élaborée dont voici quelques possibilités :

- Fonctionne à une cadence vertigineuse, au ralenti, ou à la vitesse de la machine électromécanique,
- Visualise ligne à ligne en mode Pas à PAS ou en continu l'exécution de l'algorithme,
- La saisie d'un algorithme complet de 33 lignes ne prend que quelques minutes. (*C'est autrement plus facile que d'avoir à perforer entre 30 et 50 trous dans une feuille de programme !*)
- Le déroulement du programme pouvant se faire à raison de 4400 cycles d'HORLOGE par seconde, on peut proposer des données qui en réalité imposeraient des jours, des semaines, voir des années.
- Chaque ligne de programme saisie est analysée syntaxiquement et ne sera acceptée que si elle respecte les protocoles. Certains illogismes sont également détectés et signalés.
- On peut étendre si on le désire la taille de la "grille" à 20 TRANSITIONS.
- Le déroulement du programme peut se faire "en aveugle", en surveillance tous les N cycles, voir anticiper la sortie à une "Borne" définie préalablement en nombre de cycles ...

► **Le MENU de BASE.**

Équivalent au tableau de maîtrise de la machine électromagnétique, il intègre forcément un grand nombre de possibilités. Dans une version initiale, il se résumait à une liste de 22 lignes affichées sur l'écran. Cette dernière était peu commode à exploiter et prenait pratiquement toute la hauteur de la fenêtre de dialogue. Lorsque le logiciel sur Arduino fonctionnait parfaitement et regroupait "toutes mes envies", il restait encore de la place disponible pour le programme. J'ai alors opté pour la version de la Fig.3 qui se présente sous la forme d'un cadre avec des zones typées. Sur



un RESET, la fenêtre contextuelle ressemble à celle de la Fig.3 mais en plus "aéré" verticalement, car ce dessin à été "tassé" pour minimiser la taille du fichier mis en ligne. La première caractéristique à repérer en zone 1A c'est la **transformation systématique des caractères que vous frapperez en MAJUSCULES**. C'est extrêmement agréable, car l'intégralité du dialogue se fait en Majuscule. Du coup il n'y aura pas besoin de "Shifter". À tout moment ce MENU sera affiché quand vous frapperez "?". (Ou son équivalent MAJ : La virgule comme précisé en 1B.) Remarquons au passage en 2 que la vitesse de transfert sur la ligne USB est maximale à 115200 bauds, **cadence imposée par programme sur la carte Arduino**. C'est dans la zone rouge pastel 3 que sont résumées les commandes en mode RUN. Encore faut-il avoir un programme dans la matrice virtuelle de cette machine en silicium. C'est par les commandes de la zone bleu pastel 4 que l'on rédigera un programme, où qu'il sera corrigé ligne à ligne si on le désire. La zone pastel verte 5 est relative à la manipulation du BARILLET virtuel et de ses 56 pions simulés. En zone 6 sous le menu est affichée la représentation symbolique du BARILLET, nous y reviendrons plus avant. Dans la ligne 7 on saisit les commandes. Il sera bien plus agréable de la valider avec la touche "Entrée" au lieu de cliquer en 9. Très importante, la zone 8 concerne la mémoire non volatile EEPROM de l'ATmega328 dans laquelle on peut logger des données qui ne s'effacent pas quand on coupe l'alimentation de la carte

électronique. Est fourni en accompagnement de ce tutoriel un petit logiciel Arduino qui va vous permettre de "gaver" cette mémoire avec 13 algorithmes de démonstration et une configuration BARILLET. Enfin, en **10** le programme vous invite à proposer une commande.

2) Le MENU de base.

Carrefour des nombreuses commandes de la machine, il peut à tout moment être réaffiché par la commande "?". Notez que toutes les commandes du **MENU de base** se font avec une seule touche. Toute saisie de longueur supérieure sera sanctionnée par un BIP sonore et affichera le texte "LGR > 1". Dans la pratique un analyseur syntaxique surveille toutes vos saisies et détecte pratiquement toutes les erreurs que l'on peut commettre. J'imagine que vous avez fort envie de faire fonctionner un algorithme. Aussi on va commencer par passer en revue les commandes EEPROM.

➤ **Exploitation de la mémoire non volatile EEPROM.**

Ordonnées par ordre alphabétique, la première que nous allons utiliser est "u" qui affiche comme en Fig.4 le contenu actuel de cette mémoire EEPROM. Les 1024 OCTETS de cette dernière autorisent le stockage de 15 programmes. Chaque emplacement peut contenir au choix un programme ou une **Configuration de BARILLET**. Une seule de ces configurations sera possible, et

```

Commande -> [U]
Programmes en EEPROM :
Emplacement 01 -> Programme 001
Emplacement 02 -> Programme 002
Emplacement 03 -> Programme 011
Emplacement 04 -> Programme 054
Emplacement 05 -> Programme 055
Emplacement 06 -> Programme 046
Emplacement 07 -> Configuration du BARILLET.
Emplacement 08 -> Programme 028
Emplacement 09 -> Programme 034
Emplacement 10 -> Programme ###
Emplacement 11 -> Programme 029
Emplacement 12 -> Programme 052
Emplacement 13 -> Programme 018
Emplacement 14 -> Programme 012
Emplacement 15 -> Programme 053
Commande ->
 Défilement automatique
  
```

uniquement s'il reste un emplacement libre comme ici le numéro 10 par exemple lorsque l'on utilise la commande de sauvegarde "B". Les trois symboles "###" précisent la disponibilité de l'emplacement. Si la ligne contient un programme, le nombre à trois chiffres constitue une simple Référence pour l'opérateur, qui en toute logique doit indiquer le n° de l'algorithme porté sur les petites fiches de programme. Par exemple **l'emplacement n°4 contient le programme n°54** sur nos fiches, c'est à dire le petit programme personnel qui maximise le nombre de cycles d'HORLOGE, sorte de "Castor Affairé". Pour sauvegarder un programme que nous avons saisi avec les commandes de la zone 4 en Fig.3 il faudra utiliser la commande "s" de la zone 8 qui nous demandera dans quel emplacement on désire stocker cet algorithme. Si on tente de sauvegarder un programme et qu'il n'y en a pas eu de rédigé ou de transféré depuis l'EEPROM, un message d'erreur sera généré assorti d'un BIP sonore. Quand on validera la commande "s", la première action du programme consiste à lister le contenu de l'EEPROM comme si nous avions au préalable frappé "u". Ainsi on peut

voir dans l'espace mémoire l'occupation actuelle et éventuellement choisir "d'écraser" un emplacement déjà occupé. Toutefois, nous ne pourrons pas provoquer une perte du contenu de cet emplacement par erreur, car le logiciel nous informe que l'emplacement n'est pas vide et demande confirmation. Réciproquement, la commande "c" permet de recharger le programme contenu dans un emplacement. **ATTENTION, si on valide après avoir indiqué un emplacement entre 1 et 15 il y a transfert inconditionnel** et si un programme était présent en mémoire vive, il sera définitivement perdu. Comme précisé sur le texte d'invite, si on indique un emplacement > 15 c'est l'équivalent du classique **Esc** annulant l'ordre. Si vous tentez de charger comme programme un emplacement qui contient une configuration de BARILLET, un message d'erreur sera généré et la commande ignorée.

MANIPULATIONS :

- 1) Faire un RESET pour débiter sur une configuration "vierge de toute action préalable". Pour faire un RESET on peut cliquer sur le petit bouton de la carte Arduino, ou fermer la fenêtre de dialogue et la rouvrir en cliquant en **6** de la Fig.2 donnée en page 2.
- 2) Frapper la commande "L" pour lister le programme actuellement en mémoire. La commande se solde par un message d'erreur car il n'y a actuellement rien à lister en mémoire dédiée.
- 3) Commande "c" suivie de **7** : Message d'erreur car c'est une configuration de BARILLET.
- 4) Consigne "c" suivie de **10** : Message d'erreur car l'emplacement est vide.

(Décidément, vous faites erreur sur erreur ! faites un effort nom d'une pipe !)

- 5) Encore l'ordre "c" suivie de **12** par exemple.

Il y a retour au menu de base sans autre forme de procès. Si la frappe d'une commande ne fait que retourner le texte d'invite **Retour au MENU de BASE** suivi de **Commande ->** c'est que la commande a été correctement exécutée.

MANIPULATIONS (Suite) :

- 6) Pour vérifier que le programme souhaité à bien été transféré de l'EEPROM vers la mémoire vive du microcontrôleur, frappons "l" (*Lettre minuscule de L*) pour s'assurer que le programme actuel est bien celui que l'on désirait. On obtient le résultat de la Fig.24 commentée en page 19 qui prouve dans le titre encadré et colorié en rose que c'est bien celui que nous avons désigné comme étant le n°52.
- 7) Maintenant que l'on a un programme en mémoire dédiée, on peut le sauvegarder en **10** avec "s". Comme l'emplacement est libre, pas de compte rendu mis à part le listage de type "u".
- 8) Avec "c" transférer l'emplacement n°8. Calme absolu, donc l'opération a été correctement exécutée. Du reste avec "l" on vérifie que c'est bien le programme n°52 qui a remplacé le 28 et sans préavis. (*Si on avait frappé un algorithme au clavier il serait définitivement perdu !*)
- 9) Avec "s" sauvegarder en **10**. Comme l'emplacement n'est pas libre il faut confirmer ... et le listage montre qu'il n'y a plus d'emplacement libre et que le PGM n°52 est en doublon.
- 10) Donc avec la commande "g" on libère cet emplacement. (*Sans demande de confirmation.*)
- 11) Toujours avec "g" proposez l'emplacement **7**. Persistez. (*Rassurez-vous, c'est une configuration neutre que l'on pourra très facilement retrouver par un RESET.*)
- 12) Avec "b" sauvegarder la configuration actuelle. Il faut confirmer. Le listage de type "U" nous montre que c'est le premier emplacement libre qui a été utilisé.
- 13) Comblé le dernier emplacement vide avec "s" suivi de **10**.
- 14) Avec "b" sauvegarder à nouveau "le barillet". Pas de BIP sonore et *l'affichage précise les cellules utilisées en EEPROM*. Quand on sauvegarde "un BARILLET", le programme se contente de remplacer celui qui éventuellement serait déjà présent en mémoire non volatile EEPROM.
- 15) "g" avec **7**, persister, puis "s" associé à **7**, il n'y a plus de place disponible. Frapper "b". Après confirmation, comme il n'y a plus de place le programme nous le fait clairement savoir.
- 16) Retrouver l'occupation EEPROM du début avec en 7 la configuration du BARILLET et en 10 un emplacement libre. Cette fois c'est à vous de trouver comme des grands les bonnes commandes et

Programmes en EEPROM : retrouver l'occupation EEPROM montrée sur la Fig.4 de la page 4.
 Emplacement 01 -> Programme 001 Et pour finir "c" avec l'emplacement n°8.
 Emplacement 02 -> Programme 002

Emplacement 07 -> Configuration du BARILLET. Nous allons maintenant utiliser *la commande "d"*
 Emplacement 08 -> Programme 028 qui *présente un risque potentiel de perte de l'un*
 Emplacement 09 -> Programme 034 *des algorithmes préservé en EEPROM*. Pour éviter ce
 Emplacement 10 -> Programme 028 risque, on va commencer par créer en **B** un "doublon" du
 Emplacement 11 -> Programme 029 programme **A** que l'on pourra se permettre de "détruire" sans
 Emplacement 12 -> Programme 052 regret. Donc on commence par "s" suivi de **10**, et l'EEPROM
 Emplacement 13 -> Programme 018 est de nouveau saturée. Puis on va utiliser la commande "d"
 Emplacement 14 -> Programme 012 qui permet de déplacer un algorithme, *mais qui "écrase" celui*
 Emplacement 15 -> Programme 053 *qui pourrait se trouver dans l'emplacement cible.*

- 17) Commencer par "d" avec **13** comme algorithme à déplacer, et indiquer **10** comme emplacement d'arrivée en **B**. La commande est réalisée et le résultat listé montre que **028** a été perdu et que l'emplacement de l'original **C** est maintenant effacé et redevient libre. (*Si par une fausse manipulation vous écrasez un programme, ce n'est pas la peine de vous rouler par terre. Enregistrer un algorithme prend environ cinq minutes, ce n'est pas si tragique que ça ...*)
- 18) La commande "p" constitue une *armes de destruction massive*. Elle a pour effet d'*effacer entièrement le contenu de l'EEPROM*. C'est la raison pour laquelle il faut confirmer la confirmation. Allez, on va titiller le destin : Commande "p" suivie de "o" pour oui. OUFFFFfffffffff, le programme insiste et redemande confirmation. Je suggère à ce stade de frapper n'importe quelle touche du clavier SAUF LE "o" ! (*Si vraiment vous insistez, ce n'est pas mortel, mais il faut alors téléverser le programme d'initialisation de l'EEPROM suivi du logiciel d'émulation de la machine de Turing. Autant éviter toutes ces manipulations à cause d'un petit clic qui fait un grand choc !*)

NOTE : Dans le but de rendre le dialogue Homme / Machine le plus agréable possible, (*Dialogue Femme / Machine pour l'égalité des sexes*) **chaque fois que le programme attend une réponse de type O ou N, tout caractère différent de "o" sera transcodé en "N"**.

➤ **Listage du contenu de l'EEPROM.** (*Si vous n'avez pas le temps : Sautez ce chapitre.*)

C'est bien parce qu'il y avait largement assez de place dans la mémoire de programme, que je me suis autorisé à laisser "trainer" dans le logiciel cet outil qui n'est vraiment utile que pour les "affolés de la programmation". Du reste dans le menu de base il est bien stipulé que c'est un outil qui ne concerne vraiment que les programmeurs. Autant, maintenant que le programme fonctionne correctement, et tout particulièrement avec les fonctions relatives à l'EEPROM il est devenu un luxe non indispensable, autant lors du développement il a dévoilé bien des erreurs de logique. Ceci étant précisé, comme il ne fait que lister le contenu de l'EEPROM en HEXADÉCIMAL, autant jeter un petit regard en passant et scruter l'intimité de la mémoire non volatile.

MANIPULATIONS :

- 1) Étant dans le **Menu de base**, proposez la commande "y" pour compléter de cet exercice de base. (*Bande de Dudules, ce caractère n'est pas dans la liste ! Un chtit effort siouplé ...*)
- 2) Commande "w". OUPS ... mais ça défile à donf le binaire ! Les 1024 octets préservés en EEPROM sont listés par ce que les programmeurs en "Assembleur" nommaient une **page mémoire**, c'est à dire un regroupement de 256 Octets. Sur une ligne sont listés 16 Octets en HEXADÉCIMAL. Comme la mémoire non volatile fait 1024 cellules, il y a donc quatre pages mémoire.

Commande -> [W]
 Listage en HEXADECIMAL du contenu de l'EEPROM.

ADRS	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	01	09	02	00	00	0C	02	09	03	00	00	0C	03	09	04	00
0016	00	0C	04	09	05	00	00	0C	05	09	06	00	00	0C	06	09
0032	07	00	00	0C	07	09	08	00	00	0C	08	09	09	00	00	0C
0048	09	09	0A	00	00	0C	0A	09	0B	00	00	0C	0B	09	01	00
0208	0A	03	00	0C	00	00	08	00	10	04	10	04	12	05	00	0C
0224	00	00	10	00	08	02	00	00	00	00	00	00	00	00	00	00
0240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0256	00	00	00	00	00	00	00	00	00	00	00	00	37	00	02	00
0272	00	08	00	08	00	00	00	08	03	12	00	10	00	10	04	0A
0288	05	10	00	00	00	00	08	00	08	06	12	07	08	00	10	00
0304	09	00	00	10	00	10	08	0A	05	10	00	0A	05	00	00	14
0432	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
0448	02	02	02	02	02	02	02	02	02	02	02	00	00	14	0B	14
0464	08	14	08	14	08	1C	11	06	11	02	10	00	11	03	11	03
0480	11	03	11	04	11	04	11	04	12	05	12	05	12	05	14	04
0496	14	04	14	04	11	07	11	07	11	07	11	08	11	08	11	08
0608	00	00	08	04	08	04	00	07	10	04	10	00	08	06	12	05
0624	14	03	0C	06	10	00	10	04	12	04	08	00	08	00	10	01
0640	00	00	00	00	00	0C	00	00	00	00	00	00	00	00	00	00
0656	00	00	00	00	00	00	00	00	00	00	00	00	00	00	1D	00
0976	01	00	00	00	0B	00	00	14	00	10	09	00	00	00	0A	10
0992	00	00	01	11	00	00	00	00	0C	00	00	10	00	FF	FF	FF
1008	FF	FF	FF	00	0F	0E	06	0A	0A	0E	00	FF	FF	FF	09	FF

Lorsque la mémoire EEPROM est vierge et n'a jamais été modifiée, elle ne contient que des "1" binaires, c'est à dire que les Octets en décimal font 255 qui en Hexadécimal se représente avec **FF**. Chaque enregistrement de programme consomme 67 octets et commence par son numéro représentatif. Les 15 emplacements consomment donc 1005 Octets et sont logés à partir de l'adresse 0000. C'est dans les 19 octets qui restent que sont inscrites les informations qui

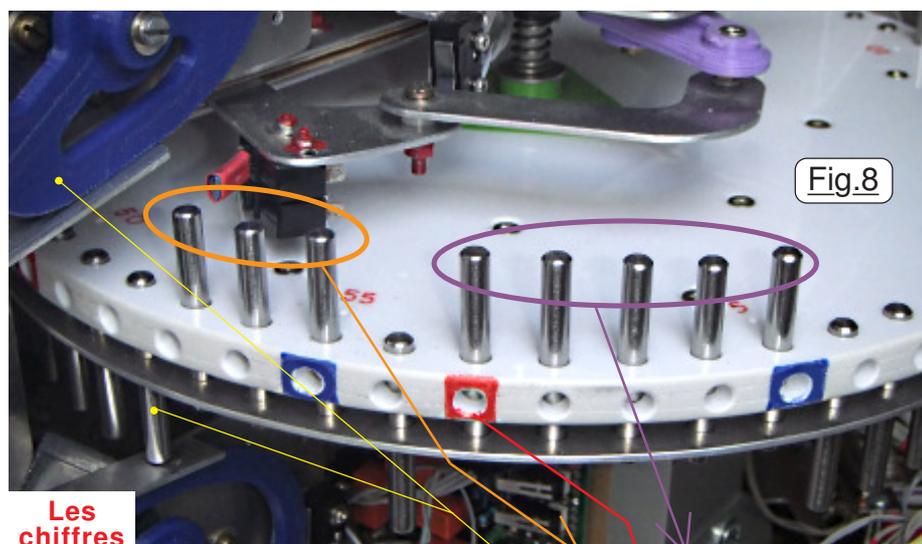
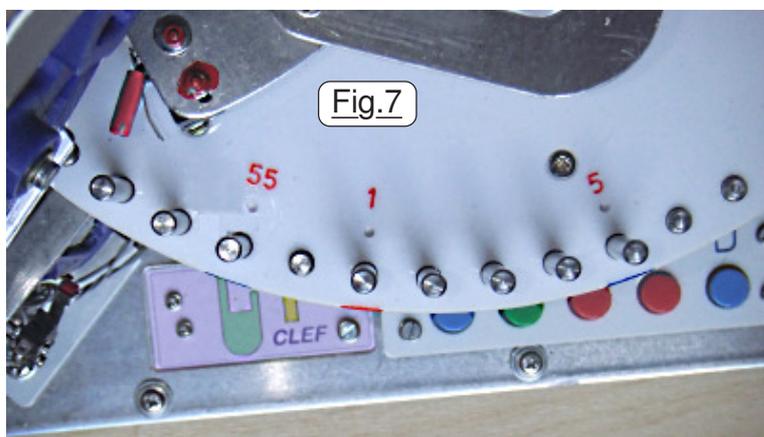
permettent de charger une configuration du Barillet ou (ET) un programme lors du RESET du microcontrôleur. Ces possibilités se programment dans le menu des OPTIONS dont il sera question plus avant. Enfin, dans le cadre violet je n'ai pas résisté à la tentation d'inscrire manuellement ma signature, c'est à dire l'indicatif qui m'était attribué quand j'étais radioamateur. (D'où l'algorithm n°15.)

3) Le MENU relatif aux fonctions du BARILLET.

Oui, je sais que vous être fébriles et nerveux et que vous voulez tout de suite faire tourner un programme. C'est promis, dès que ce sera pertinent on la déclenchera cette commande RUN. Toutefois, avant de faire dérouler un algorithme, il faut comprendre ce qu'affiche l'écran, et en particulier la représentation "graphique" du barillet de la machine électromécanique. Hors on ne dispose que d'un affichage alphanumérique, et il a fallu trouver une représentation la plus visuelle possible. Aussi nous allons commencer par les fonctions de la zone 5 verte de la Fig.3 donnée en page 3. De la sorte, nous saurons comment initialiser le BARILLET, c'est à dire positionner les pions fictifs et placer la tête de Lecture / Ecriture à convenance.

➤ Représentation symbolique du BARILLET.

Avant de manipuler pour tester quelques fonctions, il nous faut absolument établir le parallèle entre la réalité matérielle, et la symbolique adoptée pour sa représentation sur l'écran de l'ordinateur. Commençons par regarder Image 15.JPG et Image 16.JPG qui sont préservées dans le dossier <Galerie d'Images> qui accompagne ce didacticiel. Ces deux photographies présentent le BARILLET du prototype en cours de réalisation. Il importe sur ces deux photographies de voir comment sont repérées les positions des pions de cinq en cinq. Sur ces images vous constaterez que les BITS ont été repérés et numérotés en rouge tous les cinq pions sur le dessus du plateau. Ils sont également repérés en rouge et en bleu comme visible sur la Fig.8 également tous les cinq pions sur la tranche du carrousel. *Ce ne sont que des repères visuels utiles à l'opérateur pour analyser*



plus aisément le résultat du déroulement d'un algorithme.

Mais au préalable consultons la fiche du programme n°55 car nous allons travailler sur un exemple concret. C'est dans <FICHES de PROGRAMMES> que sont rangées les 33 mini fiches pour l'exploitation des *Algorithmes Utilisateur* ainsi que les 34 fiches au format A5 qui détaillent les consignes à perforer dans les pages de programme associées. La Fig.7

Les chiffres sont pour les dizaines



Repères tous les 5 pions

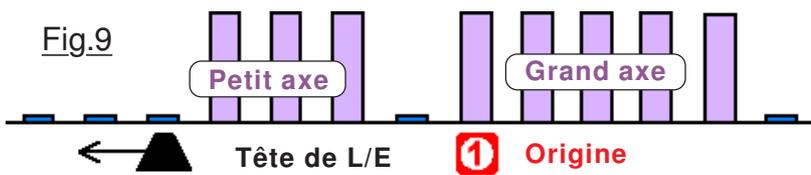
et la Fig.8 reprennent la configuration du BARILLET imposée dans la fiche de programme et qui a été initialisée sur la machine. Les tirets "-" représentent des pions à "B", les deux autres états sont symbolisés par "0" et par "1". Les deux traits verticaux symbolisent les repères bleus du plateau tracés sur la tranche tous les cinq pions. Les chiffres 1, 2, 3, 4 et 5 représentent les repères rouges tracés en regard des pions 10, 20, 30, 40 et 50. Le disque noir représente l'origine arbitraire que choisit l'utilisateur. Le petit rectangle représente la position de la tête de L/E. Ce rectangle

sera prioritaire sur tous les autres repères du plateau. S'il se trouve à l'une de leurs positions ils seront alors "en arrière" et non représentés. L'origine quand à elle sera masquée par la tête de L/E, mais restera prioritaire sur les autres "graduations" de repérage des pions. Comme c'est le cas sur la machine, la tête de L/E est immobile, c'est le plateau et ses repères qui se déplacent "latéralement".

> Initialisation du carrousel pour le programme Utilisateur n°55.

Quand Apollo circulait en orbite basse autour de la Lune en vue d'y déposer des explorateurs, le calculateur de bord n'était pas assez puissant pour effectuer les calculs de rendez-vous sur une orbite elliptique. Il fallait au préalable circulariser l'orbite d'attente pour pouvoir donner au LM l'autorisation de se séparer et de vivre sa vie. C'est le petit axe de l'orbite qui était "remonté" et égalisé au grand axe pour des raisons de stabilité orbitale pour le train spatial resté en orbite. Le petit Programme Utilisateur n°55 est un clin d'œil à la blague qui conclue le TOME 2 du didacticiel mis en ligne sur <https://www.robot-maker.com/ouvrages/machine-de-turing-tome1-presentation/>.

Au préalable consultons cette fiche pour travailler sur un exemple concret.



La Fig.9 reprend le dessin de la configuration que doit présenter le barillet lorsque l'on va déclencher un RUN. Dans cette structure tous les BITS sont à l'état "1". La tête de L/E peut se

trouver plusieurs pions à gauche de la donnée qui symbolise le Petit Axe. *Nous allons préparer le plateau de la machine virtuelle pour initialiser cette configuration de BARILLET :*

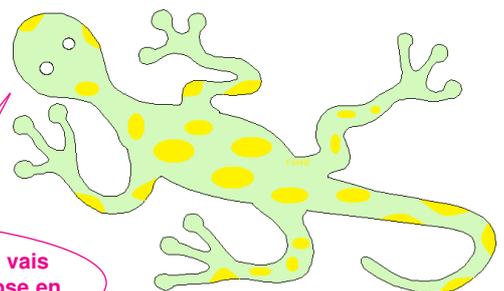
MANIPULATIONS :

- 1) Faire un RESET pour débuter sur une configuration "vierge de toute action préalable".
- 2) Frapper la commande "i" pour initialiser le BARILLET et accepter l'effacement avec "o".
- 3) Pour l'origine on va proposer 30. (Et ainsi "placer la visualisation vers le centre".)
- 4) Arbitrairement on désire placer la tête de L/E en 25 mais on va se tromper et indiquer 22,
- 5) Pour le pion de gauche on ne se trompe pas et on précise 26.
- 6) Puis frapper la séquence 111b11101 avec une erreur volontaire.

Interprétons le résultat : Comme la tête de L/E n'est plus en position 1 on peut voir sur l'écran la petite croix double qui la symbolise. En position 30 l'origine est prioritaire sur le chiffre 3.



- 7) On désire corriger la position de la tête de L/E : Frapper la commande "=" et préciser 25. Le barillet est immédiatement affiché et cette fois la tête de L/E est dans la position souhaitée.
- 8) Maintenant on veut corriger la donnée. Par la commande "f" on peut Forcer à l'état désiré n'importe quel pion du BARILLET. Préciser 33 pour la position du pion et "1" pour son état. Cette fois la configuration symbolisée est conforme à ce que l'on désirait, on va la sauvegarder :
- 9) Consigne "b" puis on confirme avec "o". *Maintenant cette configuration est pérenne et disponible en mémoire non volatile EEPROM.* Les manipulations qui vont suivre vont nous le confirmer.



Ben Mômôa j'ai une idée méga géniale, je vais commercialiser des machine de Trucingchouse en porte-clefs et en boucles d'oreilles.

MANIPULATIONS (Suite) :

10) Faire un RESET et ainsi reprendre les conditions de mises sous tension.

Sur RESET la tête de L/E se trouve en position n°1 et de ce fait masque la "croix double". L'origine est également en position n°1 et cachée par le symbole de la Tête de L/E.

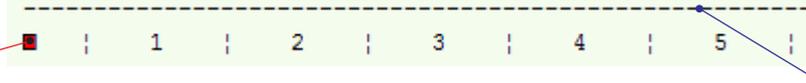
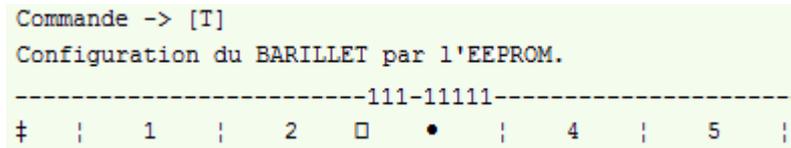


Fig.11

Tous les pions du BARILLET sont à l'état "B".

11) Frapper la commande "t" pour Transférer la configuration préservée en EEPROM.

Fig.12A



La Fig.12A confirme un résultat qui correspond bien à ce qui avait été sauvegardé en EEPROM et qui y reste jusqu'à ce que l'opérateur désire le remplacer par une structure différente.

12) Pour tester toutes les commandes de la zone 5 sur la Fig.3 on va utiliser "0" pour déplacer l'Origine en position 34. Frappez le zéro, validez puis préciser 34.

13) Remplacer maintenant pour exercice l'Origine en position 30.

➤ Chapitre de complément sur l'initialisation du BARILLET.

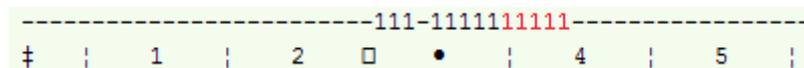
Reprenons la commande "i" du MENU de BASE qui permet de triturer les pions virtuels sur le plateau immatériel comme nous le ferions de façon moins commode sur le carrousel réel. Pour illustrer ce propos nous allons continuer à mettre à contribution le programme n°55 :

MANIPULATIONS :

- On désire changer le grand axe et par exemple le passer à dix pions. Trois possibilités potentielles :
 - Commande "i" et on reffappe entièrement la configuration,
 - Utiliser cinq fois la commande "f", (*Bien trop indigeste, autant utiliser la première approche.*)
 - *Commande "i" en se contentant de n'ajouter que cinq pions supplémentaires.*

14) Exactement comme sur le plateau réel on va "pousser vers le haut" les cinq pions à droite de la donnée du grand axe de l'état "B" à l'état "1". On commence par invoquer l'initialisation du BARILLET avec "i". On répond "n" pour l'effacement de tous les pions actuels. Pour l'origine on conserve 30 et pour la position de la tête de L/E la position 25. C'est pour le placement des pions que l'on change et l'on précise 35. On propose alors la configuration "11111" et le tour est joué. Les anciens pions n'ont pas été modifiés quel que soit leur état, seuls les nouveaux emplacements ont été changés. La configuration Fig.12B montre clairement le changement de structure dont la donnée a bénéficié avec en rouge ce qui a été modifié.

Fig.12B



Pour épuiser toutes les possibilités de la zone 5 il faudrait tester ">" et "<", mais c'est prématuré, car pour en comprendre l'effet exact et ce qui est affiché, il faut avoir fait "tourner" un algorithme.

- *Ben Nul tout il faudrait peut être que tu t'y colles, car les lecteurs deviennent nerveux !*
- *Pour une fois Dudule, je vais abonder dans ton sens, mais c'est exceptionnel !*

4) Le MENU et les commandes relatives au mode "RUN".

Enfin on va pouvoir la faire ronronner cette machine virtuelle ! Pour tester le mode RUN nous allons utiliser le petit Programme Utilisateur n°55 puisque le barillet de la machine est déjà conditionné. Et puis si on désire que le LM puisse descendre vers le sol sélène, il faut absolument circulariser l'orbite. Imaginez-vous dans la cabine exigüe du module de commande. Sur le petit clavier de l'ordinateur de bord vous invoquez la procédure de circularisation orbitale :

MANIPULATIONS :

- 1) Cliquer sur "c" pour rechercher la bonne procédure.
- 2) Dans la liste on voit que le Programme Utilisateur n°55 est en emplacement n°5 :
Invoquer l'emplacement n°5 par validation de "5".

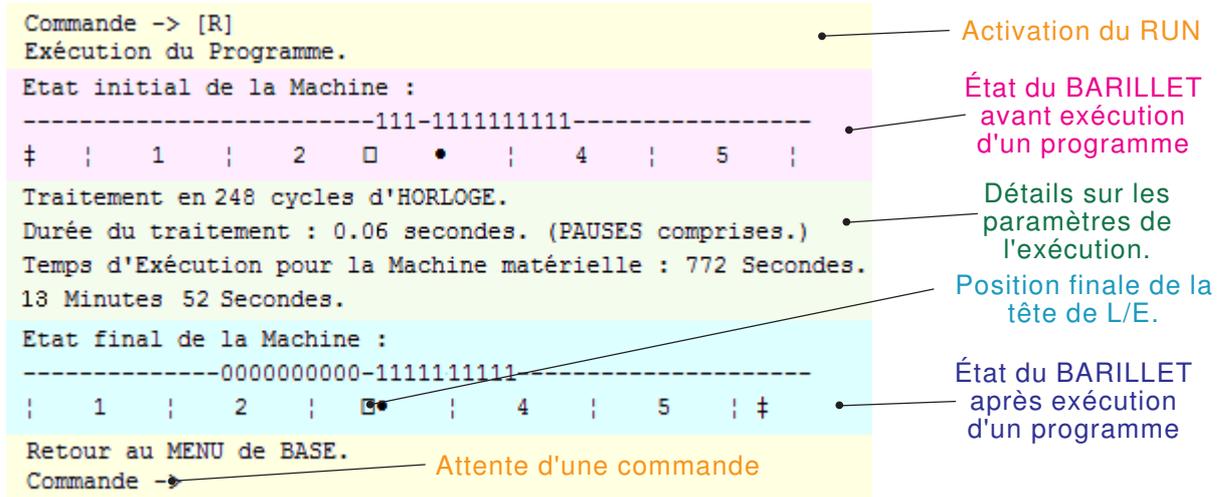
En toute logique avec la commande "L" on comparerait la grille de programme avec la fiche concernée, mais pour le moment on va faire l'impasse, car il faut circulariser de toute urgence. En effet, nous arrivons à l'aposélène, le point de déclenchement de la manœuvre :

3) Commande "r" de la zone 3 sur la Fig.3 en page 3.

- *Glups ... mais on a rien vu !*

- *Ben oui, l'Arduino à circularisé en 248 cycles d'HORLOGE ce qui n'a pris que 6 centièmes de seconde. Sur la machine réelle il aurait fallu 13 minutes et 52 secondes.*

Fig.13



> La mémorisation de la configuration du BARILLET.

Matériellement il est impossible sur la machine électromécanique de mémoriser la situation de départ. La seule façon d'en avoir une trace c'est de la noter sur un morceau de papier, ou d'utiliser la configuration indiquée dans la petite fiche de l'algorithme. En revanche, sur la version Silicium de la Machine de Turing nous pouvons à tout moment visualiser l'aspect du carrousel réel et celui de la configuration initiale :

MANIPULATIONS :

- 1) Cliquer sur "a" pour **A**fficher l'état initial qui n'existe plus sur la machine réelle.
- 2) Cliquer sur "v" pour **V**oir le résultat d'un traitement.

Nous pouvons à tout moment alterner entre ces deux présentations du BARILLET virtuel. Sur un RESET le BARILLET du carrousel est dans le même état que celui initialisé, car il n'y a pas eu d'exécution. Du reste, pour le vérifier effectuer un RESET puis cliquer sur "v" puis sur "a". Vous noterez que s'il s'agit de l'état mémorisé ce n'est pas précisé, alors que si c'est le BARILLET virtuel qui est montré il y a le texte **Résultat d'un traitement**.

Comme c'est le cas sur la machine matérielle, il est possible de faire tourner MANUELLEMENT le carrousel à notre convenance vers la gauche ou vers la droite avec les deux commandes "<" et ">". Bien que ces deux commandes soient mentionnées dans la zone 5 du MENU de base, elles concernent le **BARILLET virtuel après exécution d'un algorithme**.

MANIPULATIONS :

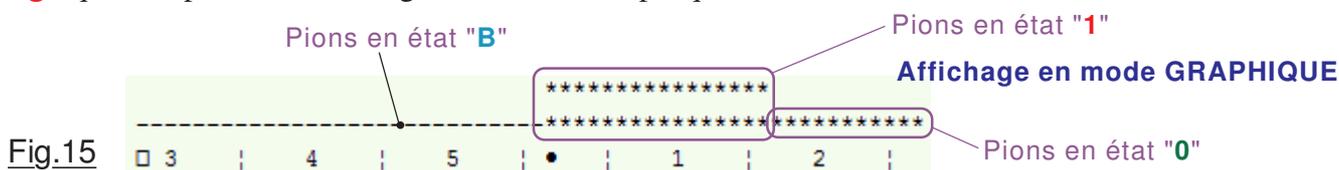
- 1) Cliquer sur "a" pour Afficher l'état initial qui n'existerait plus sur la machine réelle.
- 2) Cliquer sur "<" et valider plusieurs fois. La première action est un peu troublante, car on saute de l'état initial à l'état du BARILLET qui vient de tourner d'une position à gauche.

 **Pour éviter cette impression peu "visuelle" il est fortement conseillé de commencer par une commande "v" avant d'utiliser "<" ou ">".** 

- 3) Cliquer sur ">" et validez plusieurs fois pour faire tourner le plateau vers la droite. Comme avoir à "Shifter" n'est vraiment pas commode, utiliser de préférence "&" qui est transcodé en ">".

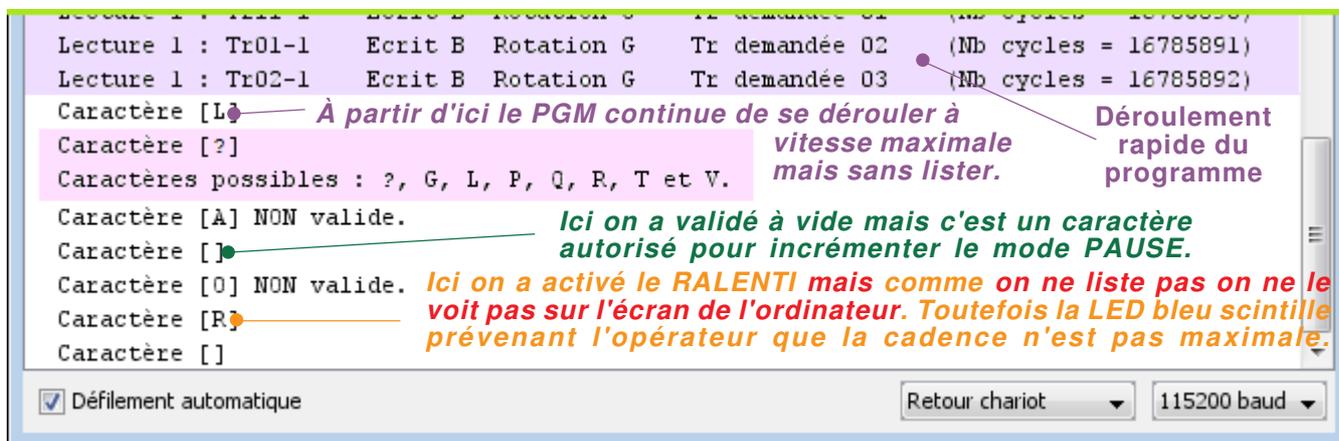
Par ces quelques commandes très simples, on pourra à tout moment initialiser le BARILLET, faire exécuter un programme, puis visualiser le résultat en décalant à notre guise Page 10

- 12) Frapper "p" et valider pour reprendre le mode aveugle. Le mode PAUSE est suspendu, mais pas le LISTAGE. Ce dernier ralentit le processus. Du coup, la LED bleue est à nouveau allumée signalant que "ça tourne", mais elle est nettement moins lumineuse étant éteinte durant l'affichage.
- 13) Proposer la commande "l" qui est une bascule de type OUI/NON pour le Listage. Tester plusieurs fois cette possibilité et observer la LED bleue et l'écran. Puis stopper le listage pour revenir au mode aveugle le plus rapide. *(La LED bleue éclaire au maximum.)*
- 14) Cliquer sur "v" qui a pour effet de Visualiser le BARILLET.
- 15) Comme ça défile un peu trop rapidement, Cliquer sur "r" pour **Ralentir le déroulement à la cadence d'une instruction par demi-seconde**. Frappez à nouveau sur "r".
- 16) Proposer maintenant la commande "t" pour passer en "Temps réel". Cette fois la LED bleue éclaire au maximum et **le déroulement du programme** devient lamentablement lent. Dans la pratique il **fonctionne exactement à la cadence de la Machine électromécanique**. Pour quitter ce mode il suffit de frapper une deuxième fois "t" sachant que **la prise en compte ne sera effective que lorsque le cycle en cours sera achevé ce qui peut prendre environ quatre secondes**.
- 17) Revenir au mode PAS à PAS avec "p" pour constater que "l" et "v" ont des effets "cumulatifs". Pour le vérifier alterner plusieurs fois ces possibilités tout en imposant de faire des pas.
- 18) Toujours en PAS à PAS et uniquement avec la visualisation du BARILLET proposer la commande "g" qui fait passer l'affichage "en mode Graphique".



MANIPULATIONS (Suite) :

- 19) Comme pour le MENU de BASE et le Menu des OPTIONS on peut à tout moment obtenir le rappel des options durant l'exécution du programme en frappant la touche "?" ou mieux, **la virgule**.



Remarquer que tout caractère frappé durant le RUN est affiché entre crochet à titre d'information pour l'opérateur. Si le caractère n'est pas valide une alerte sonore et textuelle est générée.

Mise à part la commande "q" nous avons expérimenté toutes les possibilités de commandes durant le mode "RUN". Toutes ces commandes peuvent se "cumuler" et l'on peut rendre actifs ou annuler chaque choix potentiel. Noter au passage que **les items qui seront resté actifs quand on sort de l'exécution du programme resteront actifs dans les OPTIONS** ce que l'on va constater dans le chapitre qui suit.

- 20) Suspendre le mode PAS à PAS et valider le Listage ainsi que la Visualisation du BARILLET en mode Graphique. Validez le fonctionnement en temps réel avec "t". Pendant que l'écran liste lentement toutes les informations, frapper "q" et valider pour sortir de l'exécution du programmes. *(Le "temps réel" peut engendrer un délai de réactions jusqu'à quatre secondes.)*

La sortie imposée du déroulement de l'algorithme fait afficher les informations du type de celles de la Fig.13 et le logiciel attend une nouvelle commande de l'opérateur. Nous allons à ce stade passer en revue les possibilités offertes par les OPTIONS.

5) Le MENU des OPTIONS.

Constitué d'une liste de possibilités qui presque toutes constituent des entités de valeur OUI ou NON qui sont alors actives ou suspendues. Sur un RESET par défaut la combinatoire des options correspond à celle du mode "Aveugle" pour générer l'exécution la plus rapide possible. Toutefois, comme dans cet exercice nous allons activer le **Menu des OPTIONS** alors qu'un programme a été exécuté, certains de ces choix sont restés mémorisés.

MANIPULATIONS :

1) Frapper "o" au clavier pour activer le mode OPTIONS qui commence par afficher son menu.

Plusieurs types d'options sont présents dans cette longue liste de possibilités. Comme c'est le cas pour le **Menu de BASE**, la directive "?" (*Ou la virgule.*) en 1 ont pour effet d'afficher dans la fenêtre du **Moniteur** le **Menu des OPTIONS**. La commande "q" fait **Q**uitter ce mode et revenir au **Menu de BASE**. Toutes les options mises en évidence en rouge pastel en 2 concernent la façon dont on peut gérer le déroulement d'un algorithme. En pastel bleu en 3 sont regroupées trois options très utiles quand on désire Écrire ou déverminer un nouveau programme.

Fig.16
Résume l'état actuel de toutes les options

Diverses options qui concernent le mode RUN et en conditionnent le fonctionnement

Options relatives à la saisie d'un programme

Possibilité sur RESET de charger un algorithme et une configuration de BARILLET

Passer à une Machine de Turing beaucoup plus puissante

Outils d'aides à la programmation qui ne concernent que le programmeur et qui n'ont pas été enlevées car il restait de la place dans la mémoire de programme

À titre de complément, noter que dans tous les menus, qu'ils soient de simples rappels sur une ligne ou dans des zones encadrées, sont listés par ordre alphabétique des commandes utilisées. Par ailleurs, en standard dans chaque menu *la virgule* ou le "?" servent à réafficher le menu du mode en cours. Donc si à un moment quelconque vous êtes perdu, y compris durant l'exécution d'un programme, ayez le réflexe "virgule". Comme c'est précisé sur la Fig.16 les deux commandes de la zone 5 ne concerne que les habitués du langage binaire. Vous pouvez naturellement les invoquer, elles ne font que lister à l'écran sans rien changer au contexte, par curiosité sans plus. **Page 13**

La commande "c" dans la zone verte 4 permet de charger automatiquement un contexte lors d'un RESET. C'est très commode quand on développe un algorithme et que l'on a modifié trop de lignes avec un doute sur leur pertinence et surtout très avantageux quand on pense travailler plusieurs jours sur un algorithme, ou dans la journée en cours, et avoir à le modifier un grand nombre de fois. Le recharger rapidement avec la configuration BARILLET qui lui convient sur un simple RESET s'avère très convivial.

- *Comme tu y vas Nul tout, mais elle s'use l'EEPROM ?*

- *Oui Dudule, chaque fois que l'on écrit dans une cellule, elle se dégrade un tout petit peu.*

- *Mais alors, va être rapidos H.S. le Microcontrôleur trucmachin ?*

- *Rapidement, pas vraiment. L'EEPROM est comme une mémoire S.D. Elle est certifiée pour une infinité de LECTURES mais limitée à environ 100000 écritures. Donc quand tu auras sauvegardé 100000 fois, changé l'option 100000 fois, il est possible que tu devras envisager de changer la petite carte NANO. Ceci dit, 100000 fois des écritures en EEPROM, toi aussi tu commencera à avoisiner sérieusement ta date de péremption !*

MANIPULATIONS (Suite) :

2) Dans le Menu des OPTIONS, frapper la commande "r" se contente de lister l'état actuel de toutes les options. Notons au passage que si l'on frappe "?" il y a listage du Menu des OPTIONS, suivi de l'état de ces dernières comme si l'on avait aussi ajouté l'appel à "r". Le plus simple consiste à effectuer quelques petits exercices :

3) Proposer "r" puis valider. La liste de la Fig.17 est affichée, dans laquelle on retrouve mis en évidence en rouge pastel les options qui ont été imposées durant le RUN et qui sont restées mémorisées.

En bleu pastel on observe les items qui n'ont pas été encore modifiés. (Des options par défaut.)

4) Commande "q" en 1 de la Fig.16 pour revenir au Menu de BASE.

5) Activer derechef le programme avec "r". Commande "t" pour revenir à une cadence rapide. Puis frappez "p" pour valider à nouveau le mode PAS à PAS et enfin "q" pour quitter.

6) Consigne "o" suivi de "r" pour faire lister l'état des options.

On peut alors vérifier que l'option PAS à PAS est bien restée à OUI et que le Temps Machine Réel respecté a été mémorisé à NON. **Conclusion : Les options imposées durant l'exécution d'un algorithme restent mémorisées.** Tout au moins tant que l'on ne coupe pas l'énergie, car au RESET toutes les options relatives au mode RUN sont par défaut initialisées en configuration aveugle :

```
Option -> [R]
Machine ETENDUE : NON.
Mode PAS à PAS : NON.
Lister à chaque cycle : NON.
Visualiser le BARILLET durant le RUN : NON.
Format d'affichage Graphique : OUI.
Ignorer les lignes NON utilisées : OUI.
F : Ignorer [Pas de FIN dans le PGM] : OUI.
Borne de Sortie anticipée du PGM : NON.
Surveiller le PGM : NON.
Temps Machine Réel respecté : NON.
```

Fig.18

7) Frappez "a" et valider, puis confirmer par "o" votre intention de passer en mode aveugle. On peut observer sur la Fig.18 que les items surlignés en rouge sont tous forcés à NON, *car ce sont ces options qui ont pour effet de ralentir le programme.* La commande "r" n'est pas dans la liste des OPTIONS pour simplifier le logiciel et éviter d'encombrer inutilement les menus. Elle agit uniquement durant le mode RUN et est systématiquement forcée à NON au début d'une exécution de programme. En revanche, on constate que l'option de l'affichage graphique n'est pas changée par la commande "a". C'est normal pour deux raisons : D'une part cette option n'interfère pas sur la rapidité d'exécution. Soit le barillet est en graphique, soit il est en "binaire". Dans les deux cas sa durée d'affichage sera identique, qu'il soit visualisé ou non. D'autres part on peut désirer rester dans ce mode quelles que soient les autres options, d'où cet invariant.

7) Faire un RESET, puis "o" suivi de "r". L'initialisation par défaut des OPTIONS est identique à celle de la commande "a" mis à part le fait que "Graphique" est forcé à NON.

8) Commande "p" pour valider le mode PAS à PAS. On constate que pour pouvoir vérifier l'état actuel des OPTIONS, quand on propose une consigne, sa réalisation est suivie

```
Option -> [R]
Machine ETENDUE : NON.
Mode PAS à PAS : NON.
Lister à chaque cycle : OUI.
Visualiser le BARILLET durant le RUN : OUI.
Format d'affichage Graphique : OUI.
Ignorer les lignes NON utilisées : OUI.
F : Ignorer [Pas de FIN dans le PGM] : OUI.
Borne de Sortie anticipée du PGM : NON.
Surveiller le PGM : NON.
Temps Machine Réel respecté : OUI.
```

Fig.17

de l'affichage de l'état actuel des possibilités. Surtout, on constate que valider le PAS à PAS impose implicitement la validation de l'affichage Ligne à ligne.

9) Commande "s" et valider pour activer l'option Surveillance. Confirmer ce choix avec "o".

Cette option particulière modifie le comportement du mode PAS à PAS. Au lieu de réaliser une instruction à chaque validation, le processeur en réalise **N** avant de repasser la main à l'opérateur. **N** est le nombre de cycles d'HORLOGE à indiquer lors de l'initialisation de cette option.

10) Au diable l'avarice. Par exemple proposez **12345**. Lister à chaque cycle ainsi que PAS à PAS sont forcés à NON. On va voir ce que ça donne : Commande "q" pour revenir au MENU de BASE.

11) Frappez "c" et activez l'algorithme de l'emplacement n°1. Soumettre "r" pour activer le déroulement du programme. En approximativement deux secondes les 12345 instructions ont été exécutées et une PAUSE visualise l'état de la machine virtuelle. Chaque validation "à vide" réalise 12345 instruction de plus et ainsi de suite.

Rien n'empêche durant le déroulement rapide qui allume la LED bleue, de frapper l'une des options du mode RUN comme l, p, g etc. L et G ne modifient que l'affichage, mais P invalide la Surveillance, car c'est une option avec interaction. Avant d'analyser plus finement cette notion d'interactions, expérimentons une autre possibilité pour sortir d'un programme. Sur un algorithme sans Fin comme celui en cours d'expérimentation, ou un programme qui impose un très grand nombre de cycles d'HORLOGE, on peut décider du nombre de cycles à réaliser, puis de revenir au MENU de BASE. Manipulations pour appréhender cette possibilité :

12) Commande "q" pour sortir de l'exécution, puis "o" pour revenir au Menu des OPTIONS.

- Hé, Nulentout, t'as oublié de préciser que dans le menu c'est par ordre alphabétique !

- Pas la peine Dudule, les lecteurs l'ont tous déjà remarqué. (Mauvaise foi de Nulentout...)

13) Consigne "b" pour imposer une Borne au déroulement de l'algorithme. Avec exagération on impose la valeur de **54321** instructions. (Comme le processeur va être débridé, il ne faudra qu'environ 12 secondes pour les réaliser !) "q" puis "r" pour relancer le programme. Éventuellement "l" pour annuler le Listage qui était resté actif. Et dans un temps nettement inférieur à celui que l'on aurait mis sur la vraie machine, le verdict tombe et l'on retrouve le logiciel en attente après avoir listé l'état du système suite à ces 54321 instructions réalisées. Commande "o".

14) Frapper "r" et observer que l'écran affiche **Sortie de PGM à 54321 cycles d'HORLOGE**. puis consignez à nouveau "b". **Pour invalider cette option il faut indiquer la valeur "1"** et non "0" qui semblerait plus logique mais qui n'est pas accepté pour des raisons informatique. Maintenant le programme affiche **Borne de Sortie anticipée du PGM : NON**. confirmant l'annulation.

Nous en savons assez pour pouvoir faire un bilan relatif aux OPTIONS du mode RUN et surtout analyser plus en détails les interactions qui génèrent des interférences entre ces diverses possibilités. Le tableau proposé en Fig.19 donnée ci-dessous résume les diverses configurations combinatoires envisageables en fonction des actions diverses effectuées par l'opérateur. (Noter que contrairement à l'option "A" qui valide l'initialisation globale en mode Aveugle, la combinatoire

OPTION	RESET	Aveugle	Diverses Interactions
PAS à PAS	NON	NON	Surveillance le force à NON.
Lister	NON	NON	PAS à PAS force à OUI / Surveillance force à NON.
Aff BARILLET	NON	NON	Pas d'interaction par les autres fonctions.
GRAPHIQUE	NON	X	Forcé à NON sur un RESET.
Temps Machine	NON	NON	Ralenti le force à NON.
Surveillance	NON	NON	PAS à PAS le force à NON.
Ralenti	X	X	RUN et PAS à PAS forcent à NON.
Borne	NON	X	Surveillance force à NON. Saisie = 1 force à NON.

Fig.19

15) Tableau sous les yeux, proposer à votre guise et dans un ordre quelconque ces diverses options pour vérifier les interactions qui se produisent et vous imprégner de leur logique.

➤ Les **OPTIONS** spécifiques.

Concrètement ce sont plus des fonctions réalisant un traitement particulier que de simples bascules de type OUI / NON que l'on prépositionne à notre guise. Par exemple nous avons la directive **4** de la Fig.16 qui avec la commande "i" permet de saturer en "un seul clic" la totalité du BARILLET avec des blancs, des "0" ou des "1". Le mieux est encore de l'expérimenter :

MANIPULATIONS :

1) Faire un RESET pour revenir sur une configuration "standard". Puis "o" suivi de "i".

Fig.20

```
Option -> [I]
Initialiser tout le barillet avec (B,0,1 ou N) -> [1]
Le BARILLET est initialisé avec des 1.
```

Si on frappe un caractère "binaire" correct il y a confirmation.

Tout caractère différent de "B", "0" ou "1" sera considéré comme "n" et engendrera la fuite.

Si on frappe un "n" pour NON il y a fuite et l'accusé de réception n'est pas affiché.

- 2) Frapper "1" et valider. Puis respectivement "q" suivi de "a" pour visualiser le BARILLET. La visualisation de l'état initial du carrousel confirme effectivement l'opération.
- 3) Pour se convaincre de la pertinence des informations de l'encadré bleu de la Fig.20, nous allons volontairement **commettre une erreur**. Consignes "o" suivi de "i" mais au lieu de proposer un zéro, incident qui m'arrive assez souvent, *on frappe la lettre "o"*. Déjà nous n'avons pas la confirmation textuelle. On ne le remarque pas, car avec l'habitude sauf exception on ne regarde plus vraiment l'écran avec attention. Les consignes "q" suivi de "a" montrent clairement que le barillet est resté inchangé. Donc à l'avenir il faudra ce méfier de "o" et de "0" qui dans la fenêtre contextuelle du **Moniteur** de l'**IDE** ont des apparences très similaires.
- 4) À titre d'exercice, remplir le BARILLET avec des "0", revenir au **MENU de BASE**, puis faire de même avec des "B". (Des espaces.)

Remplir le barillet avec des "B" peut ressembler à l'initialisation du BARILLET sur un RESET. Pas tout à fait, car le premier cas impose la position n°1 pour l'Origine et pour la tête de L/E alors que *la fonction "i" les laisse inchangées et ne modifie que la configuration des pions virtuels.*

5) Revenir au **MENU de BASE** et recharger la configuration du BARILLET avec "t". Puis en expert de la fonction "i" saturer le plateau avec des "0". Enfin "q" suivi de "a" pour voir le résultat.

Le barillet a bien été rempli intégralement par des "0", par contre, la tête de L/E et l'Origine ont conservé leurs positions respectives. Du coup, si on remplit avec des "B", (*Manipulation que vous devez faire immédiatement !*) on pense retrouver la configuration d'un RESET, sauf que les deux éléments cités restent dans leurs positions. Cette fonction étant bien comprise, on passe à une autre :

L'option "c" en **4** de la Fig.16 est bien du type bascule OUI / NON mais n'impose un traitement particulier que lorsque se produit un RESET, qu'il soit issu de l'opérateur ou à la mise sous tension. Pour illustrer son effet, on va supposer que vous désiriez faire des présentations de cette petite machine de Turing à des amis, et que dans ce but vous allez systématiquement commencer par le programme de circularisation des orbites préservé actuellement dans l'emplacement n°5 en mémoire non volatile EEPROM.

- 6) RESET par habitude, puis passage dans les **OPTIONS** avec "o". Appel de la fonction en frappant 'c' et valider. Vous noterez au passage que si vous avez activé cette option par erreur, toute valeur numérique supérieure à 15 engendrera la fuite. Frapper **77** par exemple pour tester. Comme prévu il ne se passe rien, donc la fuite est toujours possible.
- 7) Invoquer une deuxième fois 'c' et valider. Remarquez au passage que l'Utilisation de l'EEPROM est affichée pour pouvoir vérifier que le programme n°55 dans les petites fiches est bien en emplacement 5. Indiquer cette référence **5**. Pour cette manipulation répondez "n" pour le chargement d'une configuration de BARILLET. À partir d'ici, tout listage de l'état

des différentes options commencera par `Sur RESET charge le PGM d'Emplacement 5.` pour informer l'opérateur de cette action automatique.

- 8) RESET pour constater que sous le cadre du MENU de BASE est présente une information analogue. Commande "r" pour activer l'exécution.

- *GLUPS mais il ne se passe rien !*

- *Si Dudule, la LED bleue est illuminée, l'ATmega328 tourne à fond de cale !*

- *Mais normalement l'orbite est circularisée en un rien de temps.*

- *Ben encore faut-il que la machine soit correctement configurée cher Dudule.*

- 9) Caractère "l" suivi de "v" et enfin de "p" pour voir ce qui se passe.

Quand on laisse la validation activée en répétition sur le clavier, le film se déroule et l'on observe que :

- Le plateau tourne continuellement à gauche,
- La tête de lecture ne lit que des "B", (*C'était un tantinet prévisible !*)
- La seule instruction "perforée" indique sur **Tr01-B** un mouvement à gauche.

Il est donc assez normal que nous soyons dans le cas d'un mouvement perpétuel. En réalité, ce déplacement à gauche est prévu jusqu'à trouver la donnée du petit axe qui devrait commencer par un "1". Comme il n'existe pas, le programme va continuer à chercher en vain ...

Il faut au préalable imposer au barillet la configuration indiquée dans la mini-fiche.

Comme nous désirons avoir une machine directement utilisable sur RESET ou à la mise sous tension, il est bien plus convivial de charger une configuration de BARILLET associée au transfert d'un programme utilisateur à partir de la mémoire non volatile EEPROM :

- 10) RESET puis "o" suivi de "c". Enfin on indique 5 suivi de "o" pour confirmer. L'affichage de l'état des options commence maintenant par :

`Sur RESET charge le PGM d'Emplacement 5.`
`Charge également une Configuration Barillet.` Fig.21

- 11) Encore un RESET pour se rendre compte sur la Fig.21 que l'information initiale 1 est complétée par celle en 2, et surtout que l'on retrouve bien en 3 le carrousel tel qu'il avait été "cloné"

`L'algorithme d'emplacement EEPROM numéro 5 est chargé.`

`La config. BARILLET est également chargée.`

-----111-11111-----
‡ | 1 | 2 | □ | • | 4 | 5 |

dans l'EEPROM lors d'une manipulation précédente. Commande "r", cette fois le résultat est immédiat.

- 13) Pour les fanas du binaire, vous pouvez vous amuser avec "w" et comparer les trois dernières cellules de l'EEPROM avec la Fig.6 donnée en page 6 du tutoriel.

- 14) En vue de ne pas alourdir d'autres exercices qui vont suivre, revenons "à la normalité" en invalidant le chargement de programme sur RESET : "o", "c", 77 et éventuellement "q" suivi de "w".

Lorsque l'on fait dérouler un algorithme, il est totalement normal et pratiquement systématique d'avoir des lignes perforées qui ne seront pas utilisées. C'est particulièrement vrai quand un programme présente deux options. Les instructions du choix non validé seront ignorées. Toutefois, j'ai prévu une analyse durant le RUN pour prévenir si un tel cas se produit. Par défaut cet avertissement est invalidé sur un RESET. Quand on a mis au point un algorithme, il me semble pourtant utile de valider cette option. On peut alors vérifier si c'est normal, ce qui éviterait de perforer pour rien des trous dans la grille de programme.

NOTE : *Vous constaterez dans la pratique que passer son temps à faire un RESET n'est pas vraiment pertinent. Si on abuse dans ces exercices de cette facilité, c'est pour qu'à chaque manipulation le lecteur soit dans des conditions identiques à celle initiales lors de la rédaction du tutoriel.*

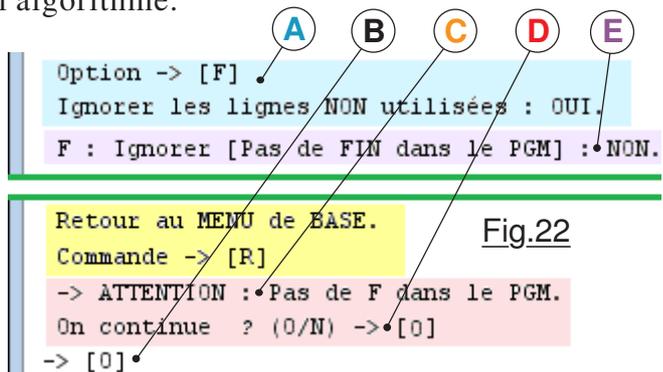
- 15) RESET suivi de "o" puis de "n". Suite à cette commande, l'état de l'option est devenu

`Ignorer les lignes NON utilisées : NON.`

et chaque RUN sera analysé pour voir si toutes ses lignes d'instructions sont bien indexées au moins une fois. Pour tester ce type de vérification frapper "c" pour l'emplacement 5 et "t" pour avoir une machine initialisée. Puis "r" pour circulariser l'orbite lunaire. L'opérateur est informé, mais il est bien précisé que ne n'est qu'une REMARQUE. C'est à lui de vérifier si ce constat est normal et qu'il était prévisible. Noter au passage que si plusieurs lignes ne sont pas explorées, seule la première sera signalée.

ATTENTION : L'information "*ATTENTION : L'instruction nn NON VIDE n'est pas utilisée.*" n'est significative d'un illogisme potentiel dans l'algorithme que si le programme est entièrement exécuté. Si on sort prématurément avec "q" cette information n'est plus pertinente et il ne faut surtout pas en tenir compte et enlever la ligne dans l'algorithme.

16) Expérimentons une fonction qui par défaut est active. Dans le menu des options, proposer "f" en **A** et observez que maintenant en **E** l'option est suspendue, donc les programmes sans le "F" seront signalés. Chargez l'emplacement n°1 qui est dans ce cas. Le **RUN** provoque l'avertissement en **C** et il faut confirmer par "o" en **D** si on désire continuer. Quand on met au point un algorithme elle peut parfois s'avérer utile, mais en standard elle reste pénalisante.



Toutes les options et fonctions de modification de l'algorithme présentes dans la zone dédiée 3 colorée en bleu pastel sur la Fig.16 de la page 12 sont relative à la saisie d'un algorithme. Au final il sera plus pertinent d'aborder ces dernières dans le prochain chapitre prévu pour expérimenter l'éditeur de PROGRAMME.

Aussi, nous allons terminer cette analyse des possibilités du menu des OPTIONS par la fonction "E" qui change radicalement la morphologie de notre machine de Turing. C'est bien une option de type OUI / NON, mais elle produit un effet très important qui la classe à part des autres possibilités. En effet, si on valide cette option particulière, la Machine de Turing virtuelle passe de 11 TRANSITIONS possibles à 20 transitions. Ajouter 9 transitions de plus semble assez dérisoire. La matrice de 33 lignes devient une grille à 60 lignes. Ce n'est même pas le double. Pourtant ce "petit changement" augmente de façon COLOSSALE la combinatoire explosive du nombre d'algorithmes possibles. Cette fois c'est le nombre de grains de sable de la Terre auquel il faut ajouter tous ceux des autres planètes du système solaire ! C'est surtout pour nous la possibilité d'expérimenter des algorithmes bien plus riches que ceux possibles sur la machine matérielle. Comme il ne s'agit plus du tout d'une fille électronique du prototype, forcément certaines fonctionnalités ne seront plus possibles. Expérimentons le comportement de cette "super machine option plus" :

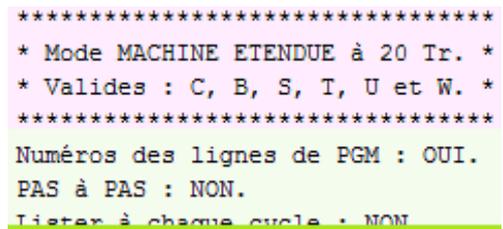
MANIPULATIONS :

16) Faire un RESET de plus.

(Le petit bouton va finir par s'user prématurément !)

17) Lettre "o" pour activer le **Menu des OPTIONS**, puis "e" et confirmer par "o". Tant que la configuration restera en mode "Machine Étendue", chaque fois que l'état des options sera affiché avec "r" par exemple, les items seront précédés de l'encadré coloré en rose pastel sur la Fig.23 qui résume les seules commandes "EEPROM" qui restent valides. En effet, il n'est plus question de déplacer des programmes par exemple car la modification du logiciel n'est plus possible à ce stade du développement par manque de place en mémoire dédiée dans l'ATmega328.

Fig.23



REMARQUE : La commande "u" reste utile car il est toujours possible de recharger un algorithme en vue de le compléter, ou d'ajouter un formatage sur un programme qui saturait les 11 transitions ...

17) Commande "q" pour revenir au **MENU de BASE**. Caractère "," pour un petit rappel des commandes. Aucun signe particulier des nouvelles caractéristiques de la machine.

18) Tentez successivement les commandes "d", "g" et "p". À chaque tentative un message d'alerte nous informe de l'impossibilité d'user de la commande et fournit un rappel des possibles.

19) Commande "c" pour charger un programme. Sa position en EEPROM n'est plus demandée et le logiciel prévient qu'il a chargé un programme ÉTENDU. Autre différence : Avec la commande "l" on constate que le **Listage** est beaucoup plus long puisqu'il comporte 60 lignes en mode étendu. *(Nous verrons plus tard que "n" n'affiche plus la feuille perforée virtuelle.)*

➤ Les particularités du mode "Machine Étendue."

M élangier en EEPROM des algorithmes "standards" à maximum 11 TRANSITIONS avec des programmes rédigés en mode Étendu n'est pas possible, car le développement était trop avancé au moment où cette option a été envisagée. Il ne restait absolument pas assez de place pour ajouter le code nécessaire. D'un autre côté, ne pas pouvoir sauvegarder ne serait-ce qu'un algorithme étendu générerait une frustration. Aussi, un compromis a été implémenté. Toutefois, pour y arriver il a été obligatoire de "faire de la place". C'est donc au détriment des affichages que l'espace vital a été dégagé. De ce fait, *il est possible que certains affichages dont les copies d'écran figurent en amont dans ce tutoriel ne soient plus tout à fait conformes à ceux de la version actuelle. Surtout ne vous en étonnez pas et n'en tenez aucun compte.*

(En particulier plusieurs textes "Configuration" ont été réduits à "Config. par exemple.")

Un seul algorithme Étendu sera possible en sauvegarde EEPROM. Il écrasera les deux emplacements 14 et 15 en fin de listage. Par convention sa référence sera systématiquement le n°254 et ne sera pas demandée par la commande "s".

MANIPULATIONS :

- 1) Faire un RESET de plus pour "repartir de zéro". Caractère "c" avec la référence 3 pour charger un programme "standard" initial à 11 transitions. Indiquer "l" pour vérifier que c'est le n°11.
- 2) "o" pour activer le Menu des OPTIONS, puis "e" que l'on confirme par "o". La transformation est clairement précisée avec un BIP sonore. Enfin, imposer la directive "q" pour revenir au Menu de BASE.
- 3) Proposer le caractère "l" pour constater que l'on a bien un algorithme de 20 transitions, *avec pour les onze premières celles de l'algorithme qui était déjà présent*. La référence est bien le 254 qui est annoncé dans l'encadré précisant cette particularité en mode "Machine Étendue".

ATTENTION : Avec la commande "c" en mode "Machine Étendue", le programme n'a plus à demander l'emplacement désiré. Il y a retour immédiat au Menu de BASE.

- 4) Frapper "c" pour transférer un programme depuis l'EEPROM *qui implicitement sera celui contenu dans les emplacements 14 et 15 quand on est en mode "Machine Étendue"*.
- 5) Caractère "l", on observe que les deux algorithmes qui étaient en 14 et 15 remplissent maintenant les 20 transitions, car Tr11 de l'emplacement 15 comportait des instructions sur ses 3 lignes.



Quand le mode "Machine Étendue" est actif, il n'y a aucune vérification du contenu actuel de l'EEPROM lors d'un chargement avec "c". Le contenu de 14 et de 15 est systématiquement chargé. C'est au programmeur à gérer la cohérence de ses manipulations et ne garder ou modifier que les lignes utiles à son algorithme. Si on désire conserver ces 20 transitions pour un algorithme Étendu, il faut impérativement utiliser la commande "e" dans les options avant de sauvegarder avec la commande "s".

- 6) On teste la commande "x" qui annonce clairement la référence 254 imposée par ce mode.
- 7) Enchaîner dans ce but "o", "e", "e", "o", "q", et "l" pour vérifier la référence.
- 8) Le marqueur 254 étant effectif, consigne "s" pour sauvegarder en EEPROM. Comme l'opération va écraser les deux emplacements 14 et 15 il y a demande de confirmation. Accepter avec 'o'.

Comme on peut le voir sur la Fig.24 le listage du contenu de l'EEPROM précise maintenant que l'emplacement n°14 contient la référence 254 et que le n°15 est PGM Étendu.

Programmes en EEPROM : Fig.24
Emplacement 01 -> Programme 001
Emplacement 02 -> Programme 002
~~Emplacement 03 -> Programme 001~~
Emplacement 14 -> Programme 254
Emplacement 15 -> Programme Etendu.

- 9) Revenir dans les options avec "o", puis frapper "c" pour prendre le gros programme en téléchargement automatique. Préciser l'emplacement 14 : Le logiciel accepte et prévient.

Sur RESET le chargement de l'emplacement 14 avec la référence 254 télécharge 14 et 15 et force le mode Machine ETENDUE.

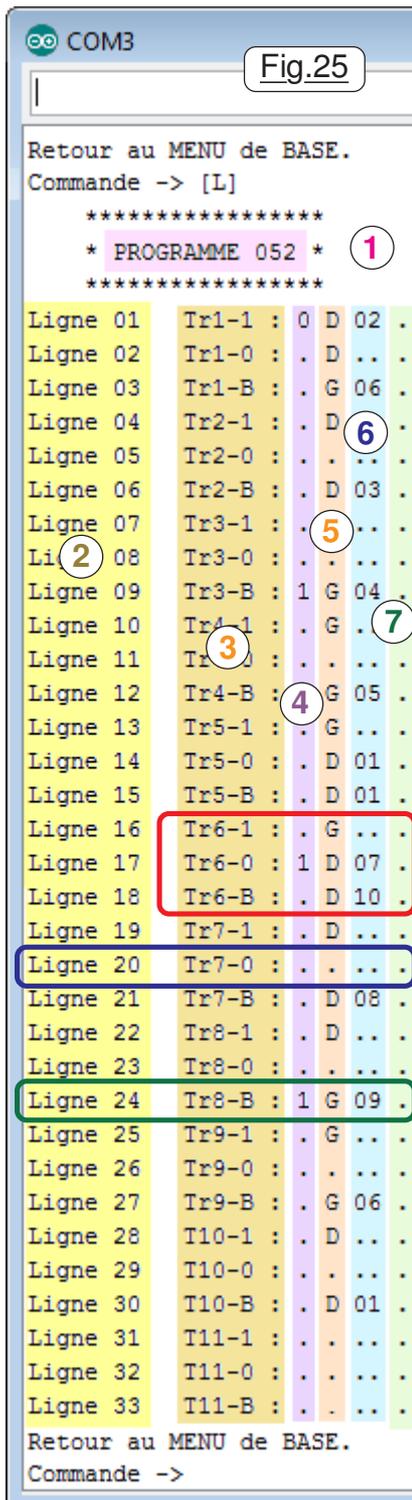
- 10) Faire un RESET pour le vérifier, puis annuler l'option "c" et annuler Machine ETENDUE. Dans le MENU de BASE on va restituer les deux références : Saisir "c, 14,x,12,s,14,o" pour charger 14, le référencer avec l'ancien 12, le sauvegarder à nouveau en 14. Comme ce n'est plus une référence 254, l'emplacement 15 retrouve immédiatement son ancienne référence.

6) Les fonctions de saisie et de modifications d'un programme.

Lorsque l'on met sous tension la machine virtuelle, il n'y a pas de programme présent dans la mémoire dédiée, sauf si l'on a validé l'option "c". On peut aussi transférer un algorithme présent en EEPROM avec "c" du MENU de BASE. Toutefois, dans la "vie normale" de ce type d'appareil, la routine habituelle consiste à lui soumettre de nouvelles grilles de programme pour en tester l'efficacité. Bref, le plaisir de la programmation dans toute sa splendeur. Pour ce faire il faut disposer d'un **ÉDITEUR de PROGRAMME**, c'est à dire des fonctions qui :

- Permettent de rédiger l'algorithme, (Avec un éditeur de texte spécifique.)
- Permettent de modifier n'importe quelle ligne individuellement,
- D'insérer à convenance du code dans une ligne vide,
- De lister l'ensemble pour pouvoir le relire facilement et l'analyser en vue de déverminer.

➤ Lister un programme ou visualiser la feuille perforée.



Préambule à la rédaction complète d'un nouveau programme, il me semble plus judicieux de commencer par l'étude des fonctions de listage et de visualisation de la grille perforée virtuelle. Dans ce but nous allons mettre l'algorithme Utilisateur n°52 à contribution. On commence donc par disposer sur le bureau la mini fiche n°52 ainsi que sa Fiche de PROGRAMME. Cet algorithme construit la suite des puissances de deux sous forme **UNAIRE**. L'ensemble du carrousel est au préalable forcé à "B" sauf le BIT situé sous la tête de L/E qui est initialisé à "1". (Peu importe la finalité de cet exemple, car c'est l'analyse de la structure du listage abordée ci-dessous qui ici présente de l'importance.)

MANIPULATIONS :

- 1) Étant en MENU de BASE, commande "c" suivie de la valeur 12 pour charger le programme.
- 2) Commande "l" pour Lister cet algorithme qui consomme dix TRANSITIONS sur les onze possibles.

En 1 on trouve la **référence** encadrée. Quand on rédige un nouveau programme on peut lui allouer **n'importe quelle valeur arbitraire inférieure à 254 et supérieure à zéro**. En 2 pour faciliter le repérage, et tout particulièrement quand on désirera modifier le code instruction par instruction, les lignes sont numérotées par ordre croissant de 1 à 33. (Et de 1 à 60 en mode **ÉTENDU**.)

- 3) Menu des **OPTIONS** avec "o" et observez dans la zone 3 coloriée en bleu qu'il y a trois autres fonctions qui seront étudiées plus tard et qui concerne la l'édition de programme. Touche "q".

Exactement comme sur les feuilles à perforer pour la MATRICE on retrouve des colonnes identiques disposées dans le même ordre. À gauche en 3 les "inscriptions" de repérage des TRANSITIONS avec pour chacune trois lignes relatives aux états ordonnés du haut vers le bas respectivement "1", "0" et "B". Puis on trouve les colonnes à perforer. En 4 on peut imposer l'écriture d'un "1", d'un "0" ou d'un "B". En 5 on peut ajouter une rotation à Gauche ou à Droite. En 6 une transition est possible, à condition toutefois qu'un F dans la colonne 7 ne soit pas présent, et réciproquement. Dans l'encadré rouge on met en évidence les trois lignes relatives à la TRANSITION n°6. Dans l'encadré bleu une ligne "vide" qui ne comporte aucune instruction. Enfin, par exemple dans l'encadré vert une ligne qui fait écrire un "1", tourner à Gauche et "commuter" la came virtuelle sur la transition N°9.

4) Le Listage par "l" est de type opérationnel. La fonction qui va suivre bascule directement dans le "je me fais un petit plaisir". Frappez "n" étant en MENU de BASE et vous comprendrez visuellement la nature de ce qui s'affiche sur l'écran. On observe une feuille perforée virtuelle assez analogue à celle prévue par le prototype électromécanique, sauf qu'ici les trous sont automatiquement perforés sans avoir à prendre l'emportepièce. Les trous sont en noir. (Ils sont légèrement plus haut que les cercles non perforés, c'est la police de caractère du Moniteur de l'IDE qui impose cette petite imperfection.) Autant la symbolisation de la grille perforée constitue un luxe, autant le nombre de trous est intéressant. Quand vous saisissez un algorithme, il est facile de se tromper. Si ce nombre ne correspond pas à celui de la Fiche il faut alors en chercher la raison.

TRn-L	B01	GD	T1	T2	T3	T4	T5	T6	T7	T8	T9	10	11	F
----- Programme Utilisateur n°052 -----														
TR1-1	o•o	o•	o	•	o	o	o	o	o	o	o	o	o	o
TR1-0	ooo	o•	o	o	o	o	o	o	o	o	o	o	o	o
TR1-B	ooo	•o	o	o	o	o	o	•	o	o	o	o	o	o
TR2-1	ooo	o•	o	o	o	o	o	o	o	o	o	o	o	o
TR2-0	ooo	oo	o	o	o	o	o	o	o	o	o	o	o	o
TR2-B	ooo	o•	o	o	•	o	o	o	o	o	o	o	o	o
TR3-1	ooo	o•	o	o	o	o	o	o	o	o	o	o	o	o
TR3-0	ooo	oo	o	o	o	o	o	o	o	o	o	o	o	o
TR3-B	ooo	o•	o	o	o	o	o	o	o	o	o	o	o	o
T10-1	ooo	o•	o	o	o	o	o	o	o	o	o	o	o	o
T10-0	ooo	oo	o	o	o	o	o	o	o	o	o	o	o	o
T10-B	ooo	o•	•	o	o	o	o	o	o	o	o	o	o	o
T11-1	ooo	oo	o	o	o	o	o	o	o	o	o	o	o	o
T11-0	ooo	oo	o	o	o	o	o	o	o	o	o	o	o	o
T11-B	ooo	oo	o	o	o	o	o	o	o	o	o	o	o	o

Fig.26

Nombre de trous dans la feuille perforée : 40.

5) Pour ce deuxième exemple nous allons transférer le programme Utilisateur de l'emplacement n°6 ... car sa finalité est élémentaire. C'est un défi qui consiste à perforer une feuille de programme et obtenir un "dessin" artistique. En revanche l'algorithme doit fonctionner sur la machine. Il peut librement comporter des instructions "redondantes" et des lignes non utilisées. Il doit en outre avoir une instruction de Fin. Caractère "c" complété par 6. "l" qui ne visualise pas grand chose de spécial, uniquement les instructions d'un algorithme. Par contre, la lettre "n" fait apparaître instantanément la particularité artistique de ce programme.

➤ La saisie d'un algorithme complet.

Rédiger entièrement un nouvel algorithme sur la machine virtuelle est d'autant plus rapide que nous n'avons jamais besoin de passer en lettres MAJuscules. Il suffit avec rigueur de parcourir ligne à ligne la fiche de programme. Pour aborder les protocole de rédaction et de modifications d'un programme, on va se donner comme objectif de créer un programme qui :

- Part d'un barillet vierge sur lequel tous les pions sont escamotés à "B".
- La tête de L/E sera dans une position quelconque ainsi que l'origine.
- Pour simplifier le travail de saisie, on va forcer à "1" trois pions avec décalage à Droite.
- Pour chaque pion on utilisera une transition différente réparée de deux autres inutilisées.

État	Lectr	Ecrt	Mvt	ÉTAT
1	1:			
	0:			
	B:	1	G	T4
2	1:			
	0:			
	B:			
4	1:			
	0:			
	B:	1	G	T7
6	0:			
	B:			
7	1:			
	0:			
	B:	1	G	F

Fig.27

Première étape : On prend une table de codage vierge, une gomme, un crayon et on cogite avec intensité. Glups que c'est difficile ! Puis on teste sur la Machine de Turing en papier décrite dans le tutoriel. On teste. Berk, c'est pas bon ! On recommence jusqu'à aboutir au tableau "creux" de la Fig.27 dont la grille ne devra comporter que 9 trous. Pour appréhender les nombreuses facettes de la saisie d'un programme avec analyse syntaxique immédiate, nous allons commettre volontairement quelques petites erreur, nous obligeant à effectuer des corrections. C'est parti, nous décidons de créer notre tout premier programme WOUACHEMENT original. Comme à chaque manipulation qui a pour but d'expérimenter un aspect particulier de l'utilisation de la machine : RESET ! Page 21

Deuxième étape :

On code avec soin et sans erreur les instructions de la table de codage. Avant de passer aux nombreuses manipulations qui vont suivre, je vous invite fortement à lire l'intégralité des informations sur l'**ANALYSEUR SYNTAXIQUE** proposé en page 23. Comme je vais vous engager volontairement sur des erreurs déclenchant des alertes, suivez avec rigueur les exercices que je vous propose. Suite à un RESET le **MENU de BASE** est affiché et la table de la Fig.27 domine notre environnement.

MANIPULATIONS :

- 1) Touche "e" pour faire appel à l'éditeur de programme et confirmer l'effacement par "o".
- 2) Comme référence on donne arbitrairement **123** par exemple.
|| L'éditeur attend les instructions que l'on désire inscrire suite à la LECTURE d'un État "1" alors que la came des TRANSITIONS est sur la position n°1. *Chaque fois que l'on validera à vide on passera à la ligne suivante dans la grille de PROGRAMME.*
- 3) Valider deux fois à vide pour "aller" sur Transition 1 Etat B. On va coder correctement cette ligne de programme en proposant la chaîne de caractères 1gt4 sauf qu'au lieu de frapper le 4 sur la touche du pavé numérique, vous utilisez celui du clavier principal et oubliez que MAJ n'est pas activée. La chaîne **1gt'** n'est pas acceptée. On corrige est cette fois on valide la saisie **1gt4**.
- 4) Valider huit fois à vide pour "aller" sur Transition 4 Etat B. On devrait soumettre 1gt7, mais par étourderie on inverse les deux premiers caractères et on propose **g1t7**. L'analyseur syntaxique y voit une écriture contradictoire. Mais peu importe, la ligne était incorrecte et n'a pas été acceptée. On corrige et on valide la suite **1gt7**.

|| Ici je vous entraîne dans des erreurs qui généralement ne se produiront que rarement. *En effet, la méthode rigoureuse pour rédiger un long programme consiste à garder un doigt ou déplacer une règle sur la TABLE du programme* et à lire la ligne "presque à haute voix". Dans ces conditions, on arrive à saisir un programme avec trente à quarante lignes sans erreur et en trois ou quatre minutes. Ce n'est que lorsque notre esprit s'égaré un peu car un beau papillon vient de passer devant le fenêtre que l'on commet des erreurs d'inversion. Les diverses commandes de l'**EDITEUR de PROGRAMME** peuvent à tout moment être affichées en frappant la virgule durant la saisie. Le protocole pour revenir sur la ligne à éditer est indiqué dans le chapitre suivant en Fig.30 de la page 24.

Retour au MENU de BASE.

Commande -> [E]

Effacer le programme actuel ? (O/N) -> [0]

Numéro de référence du programme (Num < 254) -> [123]

Ligne 01 Tr1-1 : ->

Fig.28

- 5) Valider huit fois à vide pour "aller" sur Transition 7 Etat B. La ligne correcte à écrire serait 1gf. Pour compléter cette expérience on se trompe et on propose **1gt9** en supposant que la grille serait bien plus remplie et que l'on aurait lu la mauvaise ligne. Si l'on s'en tient au protocole de la zone bleue sur la Fig.30 pour quitter la saisie, le caractère "q" devrait être employé. Une autre façon est possible. Il suffit de valider et laisser en répétition. Tester cette façon de sortir.

|| Cette dernière expérience montre que dès que l'on valide la TRANSITION n°11 pour son dernier l'état "B", le logiciel sait que l'on a terminé et nous ramène d'autorité dans le **MENU de BASE**. Comme l'option "f" est active, la sortie de saisie ne nous avertit pas de la non présence de "F", lors du développement d'un algorithme il serait pertinent d'inverser l'option.

- 6) Touche "n" : Le nombre de trous est correct. Commande "l" pour Lister ce qui a été retenu par l'éditeur de texte. On réalise que la ligne n°21 pour **Tr7-B** ne correspond pas à ce qui est dans la TABLE. On va donc la corriger.
- 7) Frapper "m" pour **M**odifier le programme. Préciser que c'est la ligne n°21 qui est à reprendre. Proposer **1gt3f** comme nouvelle ligne. L'erreur de syntaxe est détectée et la ligne est inchangée comme on peut le vérifier avec "l". Proposer alors **m>21** puis **1fg**. OUPS, le bruiteur se déchaîne. Décidément il en passe plein plein plein des papillons devant la fenêtre ! Bon, on se concentre et avec **m>21** suivi de **1gf** on code correctement la dernière ligne. On Liste cette grille qui est presque vide. Une commande "a" pour vérifier que le plateau est entièrement effacé puis "r" pour tester.

>>> **CHAMPAGNE, on a écrit et corrigé notre premier algorithme ... et il tourne !**

ANALYSEUR SYNTAXIQUE.

Toute machine informatique digne de ce nom doit effectuer une analyse de ce que lui propose le programmeur de façon à découvrir un maximum d'incohérences par rapport au langage de programmation utilisé sur cette dernière. Sur le prototype électromécanique cet analyseur est réduit à sa plus simple expression, puisqu'il ne détecte que **ER1** et **ER2**. Une troisième erreur de logique est signalée "visuellement" sur les cercles des grilles de programme pour avvertir l'opérateur qu'il va perforer un trou pour rien dans les colonnes réservées à l'ÉCRITURE.

Plus le langage de programmation est évolué, plus le nombre d'erreurs de syntaxe potentielles augmente de façon significative. Le langage de programmation imaginé par Alan Turing est tellement élémentaire, que ce nombre reste très réduit. La stratégie adoptée pour la petite machine électronique consiste à effectuer l'analyse "en temps réel" au fur et à mesure de la rédaction ligne à ligne du code. Ce n'est qu'en sortie de saisie que ce fait l'analyse pour vérifier si le programme proposé contient bien un ordre de **Fin**. L'analyseur se contentera de faire une remarque, car ce n'est pas forcément une erreur. Pour les erreurs effectives, la ligne n'est pas acceptée, quand on la valide il n'y a pas passage à la ligne suivant et une alerte sonore assortie d'un texte précisant la nature de l'erreur de logique est générée. Si l'on code une ÉCRITURE surabondante, ce n'est pas forcément une erreur, (Voir le programme Utilisateur n°46 par exemple.) l'analyseur passe à la ligne suivante, toutefois un BIP sonore et un texte nous en informe.

Bien que la "grammaire" imaginée par Alan Turing soit d'une simplicité remarquable, il reste possible sur la petite émulation en silicium d'effectuer huit vérifications dont sept se font en temps réel durant la saisie des lignes du programme. En voici la liste :

- L'instruction "**F**" ne termine pas la ligne de programme. (La ligne est refusée.)
- Transition incorrecte. (1) (La ligne est refusée.)
- ÉCRITURES contradictoires. (2) (La ligne est refusée.)
- ROTATIONS contradictoires. (3) (La ligne est refusée.)
- "**F**" avec une Transition. (4) (La ligne est refusée.)
- Écriture surabondante. (5) (Engendre une remarque "sonore".)
- Saut sur la Transition actuelle. (6) (Engendre une remarque "sonore".)
- ATTENTION Pas de "**F**" dans le PGM. (Simple remarque sonore en sortie d'édition.)
- La ligne contient un caractère invalide. (7) (La ligne est refusée.)
- Ordre ÉCRITURE / ROTATION non respecté. (8) (La ligne est refusée.)

ATTENTION : Bien que le caractère 9 par exemple soit valide, si le **T** qui doit le précéder n'est pas précisé, **il n'y a pas de message d'alerte. Par contre, la ligne ne le prendra pas en compte.** Exemple pour la saisie **1G9** le **9** sera ignoré.

- (1) : Transition supérieure à 11 ou égale à zéro. (Ou supérieure à 21 en mode Étendu.)
- (2) : On demande d'écrire deux ou trois valeurs "binaires". [**1-0**], [**0-1**], [**0-0**], [**0-B-1**] etc.
- (3) : On demande d'effectuer deux rotations. [**D-G**] ou [**G-D**]. Si on double la consigne l'instruction est acceptée le deuxième mouvement est ignoré. Exemple [**1GGT8**] etc.
- (4) : Transition et **Fin** s'excluent mutuellement.
- (5) : On consigne d'écrire un "**1**" sur la ligne "1" de la transition par exemple.
- (6) : On veut faire effectuer un saut sur la transition actuelle. (Accepté bien qu'illogique.)
- (7) : Les seuls caractères valides sont : **B, 0, 1 à 9, G, D, et T.**
- (8) : Par exemple au lieu d'indiquer **1DT5** on se trompe et on code **D1T5**.

REMARQUE : Une écriture surabondante ou un saut sur la transition actuelle sont fondamentalement illogiques et engendrent des perforations inutiles sur la grille de programme. Ces "incongruités" sont toutefois acceptées, car dans certains cas très particuliers elles se justifient. Par exemple pour le programme utilisateur "artistique" n°46. Un programme qui ne comporte pas de "**F**" peut également rester cohérent, par exemple le PGM n°28.

➤ **Corriger un programme pendant sa saisie.**

Autant corriger ligne à ligne comme précisé dans la troisième étape reste un peu indigeste, autant *corriger individuellement une ligne dans l'EDITEUR de programme est facile et rapide*. Pour tester les procédures "de déplacement" dans le listage, on va prendre comme objectif d'Ecrire le programme de la Fig.29 qui ne fait qu'incrémenter la came des TRANSITIONS, le BARILLET initial étant supposé "vierge", c'est à dire avec uniquement des pions à l'état "B".

- 8) Cliquer sur le petit bouton d'initialisation de la carte Arduino NANO pour "repartir à zéro", ou fermer la fenêtre du Moniteur de l'IDE et y revenir. Le BARILLET est dans l'état attendu entièrement à "B".
- 9) Touche "e" pour ouvrir l'éditeur. Pour l'effacement on peut au choix répondre "o" ou "n" car de toute façon actuellement la zone réservée à l'algorithme est vierge. Enfin, pour la référence on donne **123** par exemple histoire de ne pas se prendre la tête.

On va saisir les instructions en cascade, mais pour la ligne n°3 par exemple, on va se tromper et oublier de frapper le "t" qui signale que le chiffre qui suit est celui d'une transition. Dans ce qui suit **V** signifie que vous *validez à vide* sans indiquer de caractère *pour sauter la ligne qui suit sans la modifier*.

- 10) Avant de soumettre des lignes de codage, saisir "?" ou mieux "," pour avoir le rappel des commandes possibles. Cette directive est disponible à tout moment.

Commande -> [E]
 Effacer le programme actuel ? (O/N) -> [0]
 Numéro de référence du programme (Num < 254) -> [123]

```
Ligne 01 Tr1-1 : -> [?]
*****
* Commandes de l'EDITEUR de PGM. *
*****
* Ecriture PUIS Rotation en premier. *
* Transition indiquée par 'T' *
* T et F s'excluent mutuellement. *
* Exemples : BDT3 ou GF ou OT11 ... *
*****
* ? ou Virgule : Rappel de ce menu. *
* Q : Sortie de l'EDITEUR de PGM. *
* Valide seul : Ligne suivante. *
* < : Ligne précédente. *
*****
Ligne 02 Tr1-0 : -> [<]
Ligne 01 Tr1-1 : ->
```

Fig.30

infiniment plus *facile* que de modifier ligne à ligne étant revenu dans le Menu de BASE. Donc il faut revenir en arrière. Mais vous débutez et avez oublié la commande qui permet de le faire.

- 13) Touche standard "," pour avoir l'aide qui s'affiche comme sur la Fig.30 sauf que vous observez que cette fois l'index n'a pas changé de ligne, et vous pourriez l'éditer directement sans avoir comme dans l'encadré rouge à corriger avec la consigne "<" qui fait revenir en arrière d'une ligne.
- 14) Proposer des "<" jusqu'à revenir sur la ligne n°3 à corriger. Puis "t2" comme prévu.
- 15) Sur le clavier de l'ordinateur maintenir "V" en répétition jusqu'à revenir sur la ligne n°15, moyen facile et rapide de "descendre" dans le listage. Mais vous dépassez jusqu'en ligne n°17.
- 16) Pas de problème, deux fois la consigne "<" suivi de "f" et enfin de "q" pour sortir.
- 17) Commande "l" pour Lister le programme et vérifier sa cohérence.
- 18) Enfin, touche "r" pour constater le fonctionnement correct de cet algorithme fabuleux.

```
*****
* PROGRAMME 123 *
*****
Ligne 01 Tr1-1 : Fig.29
Ligne 02 Tr1-0 : . . . . .
Ligne 03 Tr1-B : . . 02 .
Ligne 04 Tr2-1 : . . . . .
Ligne 05 Tr2-0 : . . . . .
Ligne 06 Tr2-B : . . 03 .
Ligne 07 Tr3-1 : . . . . .
Ligne 08 Tr3-0 : . . . . .
Ligne 09 Tr3-B : . . 04 .
Ligne 10 Tr4-1 : . . . . .
Ligne 11 Tr4-0 : . . . . .
Ligne 12 Tr4-B : . . 05 .
Ligne 13 Tr5-1 : . . . . .
Ligne 14 Tr5-0 : . . . . .
Ligne 15 Tr5-B : . . . . F
Ligne 16 Tr6-1 : . . . . .
Ligne 17 Tr6-0 : . . . . .
Ligne 18 Tr6-B : . . . . .
Ligne 19 Tr7-1 : . . . . .
Ligne 20 Tr7-0 : . . . . .
Ligne 21 Tr7-B : . . . . .
Ligne 22 Tr8-1 : . . . . .
Ligne 23 Tr8-0 : . . . . .
Ligne 24 Tr8-B : . . . . .
Ligne 25 Tr9-1 : . . . . .
Ligne 26 Tr9-0 : . . . . .
Ligne 27 Tr9-B : . . . . .
Ligne 28 Tr10-1 : . . . . .
Ligne 29 Tr10-0 : . . . . .
Ligne 30 Tr10-B : . . . . .
Ligne 31 Tr11-0 : . . . . .
Ligne 32 Tr11-1 : . . . . .
Ligne 33 Tr11-B : . . . . .
```

Protocole d'utilisation de "?" dans l'EDITEUR de programme :

Si on fait appel à "?" et que l'on se trouve sur la première ligne comme c'est les cas dans l'encadré bleu de la Fig.30, on remarque dans l'encadré rouge que *la saisie passe à la ligne n°2*. Ce cas particulier ne se produit exclusivement que sur cette ligne n°1. Il suffit de frapper sur la touche "<" pour indexer à nouveau la première ligne.

- 11) Touche "<" pour revenir sur la ligne n°1.
- 12) Dans l'ordre saisir "V V 2 V V t3 V V t4 V V t5".

||| (Attention, dans le formalisme de la saisie, 2, t3, t4 et t5 sont des lignes saisies, donc elles se terminent par une validation non précisée dans la chaîne de caractères.)

Vous venez de vous apercevoir que sur la ligne n°3 le "T" a été oublié et vous décidez de *corriger immédiatement, car c'est*

Troisième étape :

Compléter, modifier un programme qui commence à fonctionner. Les manipulations qui vont suivre ne sont rentables que si le nombre de lignes à modifier reste faible ou en option Machine ETENDUE. S'il est important, il peut s'avérer bien plus rapide de tout réécrire depuis le début avec l'éditeur au lieu de corriger laborieusement ligne à ligne. Pour tester ce genre de manipulations on va rajouter deux sauts, séparés par une seule transition. Notre objectif : De la transition n°5 on sautera à la n°7, et de la n°7 on sautera à la n°9 avec à chaque instruction l'écriture d'un "1" et rotation à gauche.

MANIPULATIONS :

- 1) Frapper "m" pour Modifier le programme. Préciser que c'est la ligne 15 qui est à reprendre. Proposer alors la séquence 1gt7 comme nouvelle instruction.
- 2) Commande "m" et indiquer la ligne n°21. Saisir la chaîne 1gt9.
- 3) Enfin "m" et désigner la ligne n°27. Saisir le groupe 1gf. Sans que ce soit spécifiquement désagréable, s'il faut corriger 20 lignes ou plus, autant tout réécrire à partir de zéro.
- 4) Comme on a changé d'idée, on désire de plus modifier la référence de cet algorithme. Il suffit de faire appel à "x" et d'indiquer 100 par exemple. "l" pour vérifier notre travail.
- 5) La frappe de "r" confirme le fonctionnement correct. (CHAMPAGNE !)
- 6) Il se fait tard et vous désirez continuer à modifier ce programme demain ou en fin de la semaine prochaine. Caractère "s" pour sauvegarder. Proposer 10 pour l'emplacement qui actuellement est libre. Le listage de type "U" nous confirme que maintenant cet emplacement contient la grille référencée avec le n°100.

Quatrième étape :

Chaque fois que l'on doit adopter une "tradition" logique, se pose le problème d'avoir à sélectionner la norme la plus évidente. Hors, la logique des uns peut correspondre à une vision opposée des autres. Cette antinomie dans les façons de raisonner engendre un rationnel qui ne dépend au final que de la façon très personnelle dont on construit notre pensée.

- Hé, Nulentout, t'as été faire un séjour dans une secte ? Je pige rien du tout à ton baratin.
- Tu as raison cher Dudule, je vais écouter ma philosophie péremptoire de comptoir et utiliser un verbiage plus simple.

Dans l'idée purement théorique de Machine imaginé par Turing, la bande de papier est de longueur infinie, concept qui ne pose strictement aucun problème en mathématiques, le symbole ∞ lui est du reste réservé. Ce n'est envisageable que dans l'immatériel, car dans la pratique elle aurait une masse infinie, serait pratiquement impossible à faire bouger et courberait l'espace temps comme le prédit sans contestation possible la théorie de la relativité annoncée par un certain Einstein.

- Encore, mais il va pas bien du tout le Nulentout aujourd'hui !
- OK, je me calme.

Vraisemblablement, on peut imaginer qu'Alan aurait privilégié de déplacer la Tête de L/E dont la masse serait considérablement plus faible. C'est cette vision des choses que l'on retrouve sur un grand nombre d'exemples qui sont proposés sur Internet et dont je me suis inspiré. *Hors sur de nombreuses machines mécaniques, les concepteurs préfèrent de loin avoir une tête de L/E immobile et déplacer "la bande de papier"*. C'est exactement ce que fait le prototype électromécanique, sauf que la bande de papier est refermée sur elle même et tourne au lieu de se translater.

Du coup, pour "perforer notre grille virtuelle", on doit mentalement changer les G en D et les D en G avec une probabilité de risque d'erreur qui tend vers "beaucoup" par valeur définitive. Aussi, je vous suggère fortement de procéder en deux étapes. La première consiste à saisir l'algorithme comme vous le trouvez sur la toile. Puis "ligne à ligne" on effectue l'inversion. Et comme cette modification en manuel est totalement irrationnelle, le programme de la carte Arduino NANO intègre une fonction qui va effectuer automatiquement ce travail à notre place.



Zavez vu ? Le Nulentout profite de toutes les occasions pour se biberonner du CHAMPAGNE. Ben Mômôa je dis que ce n'est pas sérieux de piloter une Machine de Turing en état d'ébriété !

MANIPULATIONS :

1) Frapper une fois de plus la touche "o" pour revenir dans le menu des OPTIONS.

On remarque dans la zone verte de la Fig.16 page 13 **Options pour la Rédaction du PROGRAMME.** l'option **O : Inverser les ORIENTATIONS de D et G dans le programme.** prévue dans ce but.

2) Commande "o" pour invoquer cet outil, et confirmer avec "o".

3) Touche "q" pour revenir au MENU de BASE suivie de "l" pour Lister le programme.

On observe qu'effectivement les trois G initiaux sont bien remplacés par des D.

4) Déclencher une exécution avec "r", cette fois les "1" sont écrits "de l'autre coté".

➤ Un standard personnel.

D' une façon générale, chaque fois qu'un programme fonctionne et qu'il reste au moins une transition disponible, j'ajoute en tête de grille une transition pour pouvoir placer les données à droite de la Tête de L/E et ainsi trouver les pions initiaux bien en vue quand on déclenche l'exécution. Comme exemple typique on va reprendre l'algorithme de la Fig.27 donné en page 21, mais en partant d'un BARILLET avec des "0" pour avoir une donnée visuelle à dégager sur la machine.

MANIPULATIONS :

1) Pour recommencer sur une configuration globale propre, RESET sur la carte Arduino.

2) Touche "e" pour invoquer l'EDITEUR de PROGRAMME puis "o" suivi de "123".

3) Dans l'ordre saisir "V 1gt4" puis avec des "V" aller sur la ligne n°11.

4) Saisir la suite "1gt7" puis avec des "V" aller sur la ligne n°20.

5) Terminer par "1gf" puis sortie avec "q" pour revenir au MENU de BASE.

6) On va Initialiser le BARILLET avec "i" suivi de "o" et trois fois 30.

7) Pour les données on propose "000", la Fig.31 traduisant le résultat obtenu. (*Attention : Trois fois zéro.*)

8) Un petit RUN avec "r" pour vérifier que le programme fonctionne.

On décide maintenant de positionner la tête de L/E à gauche des données, le programme doit alors faire tourner le plateau de la machine jusqu'à trouver un "0". Dans ce but il faut décaler tout le programme d'une transition vers "le bas" pour pouvoir sur la transition n°1 installer les instructions

```

*****
* PROGRAMME 123 *
*****
Ligne 01 Tr1-1 : . . 02 .
Ligne 02 Tr1-0 : . . . .
Ligne 03 Tr1-B : . G . .
Ligne 04 Tr2-1 : . . . .
Ligne 05 Tr2-0 : 1 G 04 .
Ligne 06 Tr2-B : . . . .
Ligne 07 Tr3-1 : . . . .
Ligne 08 Tr3-0 : . . . .
Ligne 09 Tr3-B : . . . .
Ligne 10 Tr4-1 : . . . .
Ligne 11 Tr4-0 : . . . .
Ligne 12 Tr4-B : . . . .
Ligne 13 Tr5-1 : . . . .
Ligne 14 Tr5-0 : 1 G 07 .
Ligne 15 Tr5-B : . . . .
Ligne 16 Tr6-1 : . . . .
Ligne 17 Tr6-0 : . . . .
Ligne 18 Tr6-B : . . . .
Ligne 19 Tr7-1 : . . . .
Ligne 20 Tr7-0 : . . . .
Ligne 21 Tr7-B : . . . .
Ligne 22 Tr8-1 : . . . .
Ligne 23 Tr8-0 : 1 G .. F
Ligne 24 Tr8-B : . . . .
    
```

de déplacement de la tête. Le problème, c'est que toutes les instructions de type **Trn** doivent être incrémentées. C'est tellement rébarbatif à faire que le logiciel de la carte Arduino NANO prévoit une fonction pour réaliser cette transposition de façon automatique.

Considérons la Fig.32 qui résume le travail que doit accomplir cette option à notre place. En première action, elle doit déplacer l'ensemble de la "grille de trous" d'une transition vers le bas, à condition toutefois qu'il reste au moins trois lignes de disponibles dans **Tr11** ou **Tr20** si le mode ETENDU est actif. Cette première étape est coloriée en vert pastel sur la Fig.32 montrant le "haut" du listage. Puis, la transition **Tr1** est remplacée automatiquement par le code du cadre colorié en rouge. Comme le logiciel ne peut pas savoir si le

```

Commande -> [I]
-----
| 1 | 2 | 3 | 4 | 5 |
> Effacer tous les pions actuels ? (O/N) -> [O]
Position de l'ORIGINE -> [30]
Position de la Tête de L/E -> [30]
Position du pion de GAUCHE -> [30]
Tout caractère différent de [1,0,B] sera forcé à "B".
Configuration des données -> [000]
-----
| 1 | 2 | 3 | 4 | 5 |
Retour au MENU de BASE.
Commande ->
    
```

Fig.32

Fig.31

tion de gauche de la donnée sera un "1" ou un "0", arbitrairement le logiciel de la carte Arduino NANO privilégie le "1", et *il sera à la charge de l'opérateur de corriger l'algorithme si nécessaire*. La Fig.32 est en réalité un "montage" pour visualiser ce que donnerait la fonction "m" si la troisième étape n'avait pas été prévue. Les instructions mises en évidence en bleu programmées dans l'original pointeraient une transition pas assez loin. La dernière étape que doit réaliser le logiciel consiste alors à incrémenter toutes les instructions **Trn** sauf la première naturellement.

MANIPULATIONS (Suite) :

- 9) Directive "o" pour aller dans le **Menu des OPTIONS**.
- 10) Option "m", confirmée par "o". Touche "q" pour revenir au **MENU de BASE**.
- 11) Caractère "l" pour **L**ister le programme et constater qu'effectivement les transitions concernées ont correctement été incrémentées conduisant à un programme cohérent dont l'algorithme est identique à celui de l'original, avec la première transition ajoutée au tout début.

C'est maintenant à nous de corriger Tr1-1 et Tr1-0 pour tenir compte du fait que les données commencent par un bit "0" à gauche au lieu d'un BIT initialisé à "1".

- 11) Touche "m", indiquer **1** pour la ligne et valider à vide. Le logiciel nous prévient que maintenant cette ligne est vide, donc effacée. Ce n'est qu'un avertissement pour inciter le programmeur à vérifier ce qu'il a fait. Du reste cliquer sur "l" pour vérifier que la ligne n°1 est bien effacée.
- 12) Commande "m", indiquer **2** pour la ligne à compléter et proposer "t2". Lister pour vérifier la cohérence du programme. Il nous reste à déplacer la tête de L/E à gauche des données pour s'assurer que le programme continue de fonctionner correctement indépendamment de cette position :
- 13) Frapper "=" puis **25** par exemple.

*Pour observer visuellement ce qui va se passer quand on va déclencher l'automatisme, on va aller dans les **OPTIONS** pour préparer l'exécution.*

- 14) Imposer "o", puis "a" et confirmer avec "o" pour annuler toutes les options du **RUN**.
- 15) Caractère "p" pour activer le mode PAS à PAS et "l" pour annuler le **L**istage.
- 16) Ne pas oublier "v" pour visualiser le plateau virtuel de la machine.
- 17) Indiquer "q" pour revenir au **MENU de BASE**.
- 18) Enfin "r" pour déclencher les hostilités.
- 19) Valider à vide plusieurs fois pour voir le plateau tourner dans un premier temps, puis les rotations assorties des écritures des "1" qui remplacent les "0" conformément à ce que prévoit l'algorithme.

*Pour terminer ces ultimes exercices, nous allons vérifier qu'en mode **ETENDU** "m" est toujours possible, et que la place disponible est bien vérifiée avant d'effectuer la transposition.*

- 20) **RESET**, "c" suivi de **1** pour charger le programme n°1 en mémoire dédiée.
- 21) Imposer "o", puis "e" et accepter le mode Machine **ETENDUE**. Comme le programme qui était en mémoire n'a pas la référence 254, le logiciel nous prévient qu'il a été transformé.
- 22) Touche "q", puis dans le **MENU de BASE** frapper "l" pour voir ce que ça donne. Le listage est devenu plus long, les instructions ont été conservées, et la référence a été changée en 254.
- 23) "o" pour revenir dans les options. Puis "m" et "o" **deux fois** suivi de "q" et de "l". On constate que l'on peut "translater" plusieurs fois le programme, car la fonction "m" des options n'analyse pas l'algorithme qui peut être absolument quelconque. Elle ne fait qu'ajouter une transition "spéciale typique" au tout début et incrémenter tous les **Trn**.
- 24) Touche "m" et préciser **60** puis "f" pour "consommer" la dernière transition. "l" pour vérifier.
- 25) Répéter "o" puis "m" pour tenter une nouvelle **M**odification "conventionnelle".
- 26) Caractère "o" pour accepter. Comme il ne reste plus une transition de disponible en fin de listage, le logiciel prévient l'opérateur par un BIP sonore et le texte **Place insuffisante.**

Nous avons passé en revue de détails tous les menus disponibles sur cette petite unité informatique si discrète dans le tiroir de rangement. Du moins j'espère n'avoir rien oublié. De toute façon, pratiquement l'intégralité du comportement logiciel est résumé dans les très nombreux affichages de l'écran vidéo. Pour clore cette formation on va "faire tourner l'HORLOGE système". **Page 27**

7) Derniers petits exercices pour achever votre formation.

Dans ce chapitre ultime de votre formation, vous allez passer votre permis de conduire une Machine de Turing virtuelle et obtenir votre habilitation à rédiger et modifier des algorithmes. Ce long préambule théâtral ne fait qu'introduire l'exploration des programmes qui sont disponibles dans l'EEPROM et ainsi se faire un petit plaisir avant de se quitter, ou plus exactement en arriver au chapitre de réalisation matérielle de la "petite chose verte" qui a absolument tout d'une grande !

REMARQUES :

- Je suis presque certain que durant les exercices de familiarisation avec la machine virtuelle, vous avez remarqué que le texte d'invite à frapper une directive **Option ->** ou **Commande** nous précise "visuellement" et en permanence dans quel menu principal on se trouve.
- Lors des innombrables manipulations pour mettre au point le logiciel, utiliser "q" pour revenir des OPTIONS vers le **MENU de BASE** est devenu un réflexe quasi incontrôlé. Du coup, étant en mode **Commande**, je frappais parfois "q" au lieu de "o" pour aller dans les options. Aussi, dans une optique "d'uniformisation", cette lettre "q" est également valide dans le **MENU de BASE** comme précisé dans le tableau de la Fig.3 donné en page 3 de ce didacticiel.

MANIPULATIONS :

- 1) RESET, puis charger l'emplacement n°2. Passer en OPTIONS, et "n" pour annuler l'option qui consiste à ignorer des lignes non vides et inutilisées au cours d'un RUN. Puis lettre "q" et dans le MENU de BASE "r" pour déclencher l'automatisme. Avec "l" vous analysez le programme et vous comprendrez immédiatement que la ligne 25 ne sera jamais invoquée. À titre d'expérience vous validez l'option de PAS à PAS, vous testez à nouveau ce programme qui ne fait que faire tourner la came des transitions par incrémentations successives. Enfin vous terminez par l'option "a".
- 2) Charger l'emplacement n°3. Touche "a" pour vérifier que le BARILLET est vierge. Vous prenez en main la mini-fiche du **Programme Utilisateur 11**. Imposer "r" pour observer le résultat. Puis trois ou quatre fois "&" pour dégager les données de la tête de L/E. Enfin, dans les options vous validez le mode graphique et recommencez l'exercice. Bravo, vous êtes un opérateur Morse confirmé !
- 3) Charger maintenant le **Programme Utilisateur 54** en emplacement n°4. Il consiste à créer un programme avec un minimum d'instructions qui conduise à un nombre de cycles d'HORLOGE maximal. La tête de L/E "en standard" peut se trouver à plusieurs "B" à gauche de la donnée constituée de BITS à "1". Le programme va remplacer tous ces "1" par des "0" mais en partant alternativement des extrémités vers le centre obligeant ainsi à balayer "factoriellement" la zone des données. Tester en mode graphique et en PAS à PAS en commençant par une donnée de taille réduite. Par exemple [11111]. Puis, avec l'option "i" remplir le barillet avec des "1". (*La tête de L/E peut se trouver n'importe où.*) Recommencer l'exécution en PAS à PAS avec répétition, puis passer en mode aveugle. On verra visuellement le balayage systématique de la zone des données puis la fin indiquera 1654 cycles horloge pour mettre à zéro les 55 BITS ce qui sur la machine imposerait une heure et vingt minutes. Ici le mode "NON graphique" me semble plus pertinent.
- 4) Dans l'exercice précédent, la remarque sonore pour la ligne non utilisée est assez "agacive" et ne se justifie qu'en développement d'algorithme. Annuler avec "n" cette petite lourdeur.
- 5) Transférer l'emplacement n°8 qui correspond au **Programme Utilisateur 28** de réalisation d'une frise. C'est un algorithme sans fin bien adapté à un fonctionnement en mode PAS à PAS avec répétition. Dans ce but, "p", puis "l" pour ne pas lister les lignes, et surtout "v" pour visualiser le BARILLET. Adopter l'option graphique et utiliser "f" pour valider l'option qui prévient l'opérateur que la machine va tourner pour l'éternité. Dans le MENU de BASE conditionnez le plateau comme montré sur la Fig.33 et activer l'exécution. (*Utiliser la touche "V" en répétition.*) On observe qu'après avoir fait "un tour de plateau" la frise se superpose et la configuration ne change plus.

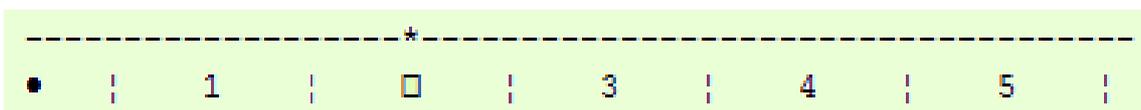
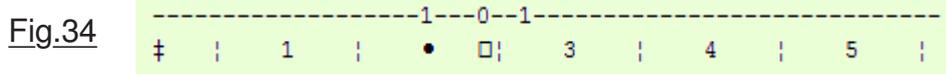


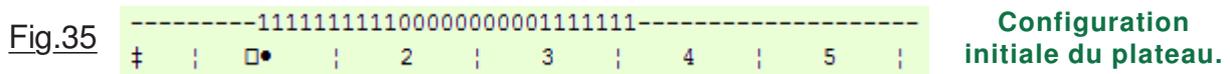
Fig.33

MANIPULATIONS (Suite) :

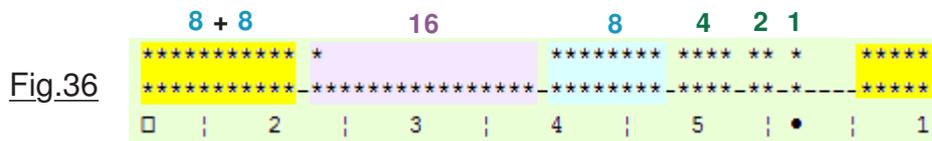
6) Avec l'emplacement n°9, on dispose du **Programme Utilisateur 34** de construction d'un mur entre deux limites. Ce programme est bien adapté à un fonctionnement en mode PAS à PAS avec des "V", en mode graphique et sans listage. Charger l'algorithme, et conditionner les options et le plateau de la machine conformément aux informations de la mini-fiche. La Fig.34 présente l'allure du plateau de la machine avant de déclencher le **RUN**. Analyser le listage pour en déduire la morphologie de cet algorithme et établir la relation avec le comportement de la machine.



7) Transférer l'emplacement n°11, pour avoir le **Programme Utilisateur 29** qui duplique une chaîne BINAIRE. Il est typique d'un algorithme dont le nombre de cycles d'HORLOGE augmente de façon significative avec la taille des données. Tester avec la taille maximale de 27 BITS. (Car deux de plus par rapport à la fiche signalétique sont possibles.) L'exécution, conseillée vivement en mode aveugle avec "a", impose 1543 cycles d'HORLOGE et plus d'une heure sur la Machine.

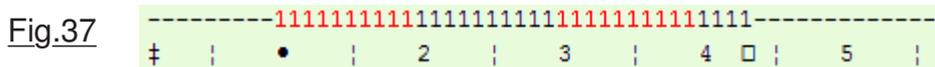


8) Le **Programme Utilisateur 52** de l'emplacement **12** qui construit la suite des puissances de deux est assez pertinent pour observer en mode PAS à PAS avec répétition le déroulement d'un processus. Passer en mode graphique et listage pour surveiller l'évolution de l'HORLOGE. Dans les options, "f" pour suspendre l'avertissement actuellement non souhaité. Initialiser le plateau avec "i", "o", trois fois 20, puis 1. En répétition laisser fonctionner et observer les changements des pions sur le plateau virtuel. Quand vous en aurez "assez observé", sortir avec "q" et dans les options imposer "a" confirmé avec "o" et avec "b" placer une butée à 1200 cycles d'HORLOGE. La machine est alors dans la configuration de BARILLET montrée sur la Fig.36 :



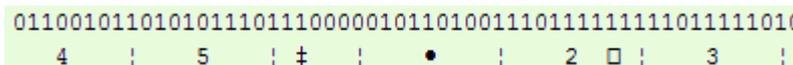
Le programme a déjà écrit les quatre premiers éléments de la suite. Pour passer à l'élément $n+1$ consulter le bas de la fiche. Sur la Fig.36 la zone jaune gauche est celle qui va être balayée pour construire la zone violette de construction de $n+1$. Revenir dans les options et annuler la borne avec "b" suivi de 1. Relancer **RUN** : conformément à la fiche du programme le résultat "diverge" et après 1698 cycles il y a sortie du déroulement du programme et retour au **MENU de BASE**.

9) Avec l'emplacement 13, donc le **Programme Utilisateur 18** on va se faire plaisir et réaliser la Conjecture de SYRACUSE. On peut se permettre sur la machine virtuelle de soumettre à l'algorithme la configuration la plus "musclée" possible de la Fig.37 que l'on obtient avec "i", "o", 10, 43, 10, et une suite de 34 BITS tous à "1".



On déclenche un **RUN** en mode aveugle, et en 13396 cycles d'HORLOGE qui prennent environ 2,5 secondes, le processus se termine correctement à la valeur prévue de un. Sur le prototype électromécanique il faudrait pas moins de 13 Heures et 20 minutes ! On ne peut pas choisir une valeur initiale plus grande, car le plateau de la machine avec ses 56 pions n'a pas assez de BITS pour effectuer le traitement. Pour le vérifier, on ajoute un "1" en poids fort : "f", 9 puis "1". On relance l'exécution, et il ne se passe plus rien, alors que la LED bleue est allumée prouvant que le microcontrôleur travaille à sa cadence la plus rapide. Frapper un "q" pour stopper l'évolution. Le BARILLET présente alors un aspect tel que celui de la Fig.38 avec une durée calculée de plusieurs jours sur la machine matérielle. En fait, l'algorithme se trouvant en présence d'une donnée altérée diverge et tourne sans fin. Il est facile quand on estime avoir saturé le plateau d'imposer un listage avec 'l' pour observer le comportement de la machine. Elle ne

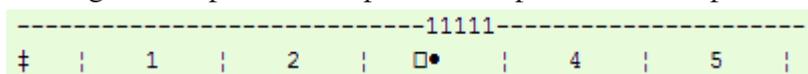
Fig.38



fait plus que provoquer une rotation à gauche du plateau et reste en permanence sur la transition n°6. La raison est évidente, et pour s'en convaincre lister l'algorithme. En ligne 16 et en ligne 17 les lectures de "0" et de "1" ne font que rechercher un "B" pour alors sauter en transition n°1. Hors le plateau est saturé et ne comporte strictement aucun BIT à l'état "B". La machine va donc faire tourner le plateau à l'infini, tout au moins tant qu'il y aura de l'énergie ou qu'elle ne tombera pas en panne. On est en présence d'un mouvement perpétuel ... virtuel !

- 10) Avec l'emplacement 14 qui préserve le Programme Utilisateur 12 on est en présence d'un programme typiquement "ordinateur" avec du transcodage. Dans cet algorithme on transpose en BINAIRE PUR un Entier codé en UNAIRE. C'est l'occasion pour celles et ceux qui veulent un peu explorer la base 2 de mettre à contribution la petite machine de Turing. Charger cet emplacement, et pour commencer configurer le plateau en prenant le premier exemple de la mini-fiche :

Fig.39



- 11) Lister le programme. Visiblement Nulentout s'est trompé quand il a saisi les instructions. S'il l'avait testé il s'en serait aperçu. Du reste, frappez un "r". Le résultat BINAIRE est correct, mais ça râle durdur. Vous pouvez observer que dès que l'erreur a été détectée, il y a eu sortie inconditionnelle du RUN assorti d'un texte d'erreur. On corrige avec "m", 30, "f" et "r" pour voir. Cette fois tout est conforme. La mini-fiche annonce 65 cycles, mais la tête était sous la donnée.
- 12) Expérimentons pour finir le dernier emplacement n°15 relatif au Programme Utilisateur 53. Il recherche le PGCD entre deux nombres UNAIRES. Proposer 9 pour X et 36 pour Y :

GLUPS, mais c'est un truc de matheux ça !

```
Effacer tous les pions actuels ? (O/N) -> [O]
Position de l'ORIGINE -> [9]
Position de la Tête de L/E -> [9]
Position du pion de GAUCHE -> [1]
Tout caractère différent de [1,0,B] sera forcé à "B".
Configuration des données -> [111111111B111111111111111111111111111111111111]
111111111B1111111111111111111111111111111111111-----
‡ | 01 | 2 | 3 | 4 | 5 |
```

Fig.40

Résultat en 1838 cycles soit environ 2 HEURES 31 Minutes sur la machine réelle.

➤ Un petit complément sur le mode Machine ÉTENDUE.

Dans cette ultime série de manipulations, nous allons aborder un petit détail qui ne pouvait être expérimenté dans les chapitres relatifs à l'option de Machine ÉTENDUE, car on aurait perdu les deux derniers programmes qui n'avaient pas encore été explorés. Pour se retrouver dans le contexte approprié, on va rapidement recréer un programme ÉTENDU :

- 1) RESET, puis "q" suivi de "e" et confirmation avec "o".
- 2) Lettre "q" pour revenir au MENU de BASE.
- 3) Commande "c" pour charger le long programme et "x" pour le référencer correctement.
- 4) Caractère "s" pour sauvegarder, ordre confirmé par "o".
- 5) On suppose ici que l'algorithme est achevé, et qu'au final vous désirez maintenant récupérer les deux emplacements : Commande "g" dans le Menu de BASE. Elle est refusée puisqu'en mode ÉTENDU cette lettre n'est pas dans la liste des directives autorisées.
- 6) Annuler l'option Machine Étendue, puis, revenir dans le MENU de BASE. Touche "g" suivie de 14. L'emplacement n°15 n'est pas libéré. Donc recommencer avec "g" suivi de 15.

Avec cette dernière petite expérience, on peut considérer que globalement vous avez une bonne connaissance du logiciel qui anime la petite unité électronique. Dans le dossier nommé <Fiches de PROGRAMMES> pour disposerez d'un nombre important d'algorithmes que vous pourrez proposer au petit simulateur, et avec les données de votre choix. Avec l'option PAS à PAS vous pourrez en observer le comportement et en déduire la "pertinence" des instructions. *Il ne reste plus qu'à réaliser la petite machine, la programmer en interne et la mettre en service.* C'est précisément l'objet des chapitres qui vont suivre.

8) Matériel investi dans la petite Machine de TURING.

Mise à part la carte **Arduino Pro Mini** qui est légèrement plus petite, et encore en longueur, mais légèrement plus large, le petit circuit imprimé **Arduino NANO** est le plus modeste en dimensions de toutes les références Arduino. Comme la **NANO** est disponible à des tarifs similaires ou inférieurs, il n'y a pas à hésiter, et ce d'autant plus que la **Pro Mini** ne dispose pas de prise USB. L'agencement global contribue à minimiser le volume qui sera occupé par l'ensemble de l'électronique. Au final le coffret du petit simulateur montré sur la Fig.41 ne mesure que 62mm de long, 25mm de large et 32mm de haut, feutres situés sur le dessous et bouton de **RESET** compris. C'est presque la ligne USB qui se branche sur l'ordinateur de dialogue qui prend le plus de place. Les diverses photographies proposées sont trompeuses, en apparence le boîtier semble gros. Dans la pratique ce sont des images saisies en **MACRO** et le coffret (Voir la Fig.46) est vraiment peu encombrant. Il suffit de savoir que les vis visibles sur la Fig.41 sont au diamètre nominal de 2mm, donc très petites. Le schéma électronique retenu et présenté en Fig.42 est très complexe puisqu'il n'ajoute à la carte **NANO** que deux résistances une LED bleue et un **BUZZER** actif. Vu le peu de composants mis en œuvre il serait possible de se passer de circuit imprimé support. Ceci dit, comme une toute petite carte de prototypage sera suffisante, cette dernière assurera l'immobilisation de l'ensemble électronique dans le coffret. Les sorties **D4** et **D5** ont été choisies pour faciliter l'implantation des composants. (*Critère faible.*) Le **BUZZER** est à votre convenance de type actif ou passif. S'il était directement branché sur **D4** les BIPs d'alerte seraient bien trop agressifs, aussi, la résistance **R1** diminue le courant envoyé au transducteur lorsque la sortie est à l'état "1". Le logiciel est totalement compatible avec les deux types de composants. Sur la Fig.43 la carte **Arduino NANO** n'est pas installée sur les deux lignes de barrette **HE14 2** qui servent de support. On peut penser qu'il s'agit d'un luxe coûteux. Toutefois, vu que j'approvisionne ces barrettes par lots, leur prix de vente est pratiquement dérisoire. Par ailleurs, la carte **Arduino** qui a été installée sur ce circuit me sert à développer des logiciels depuis des années. Aussi, vu le nombre de téléchargement famélique qu'elle a enduré, il n'est pas exclu que l'on finisse par dépasser

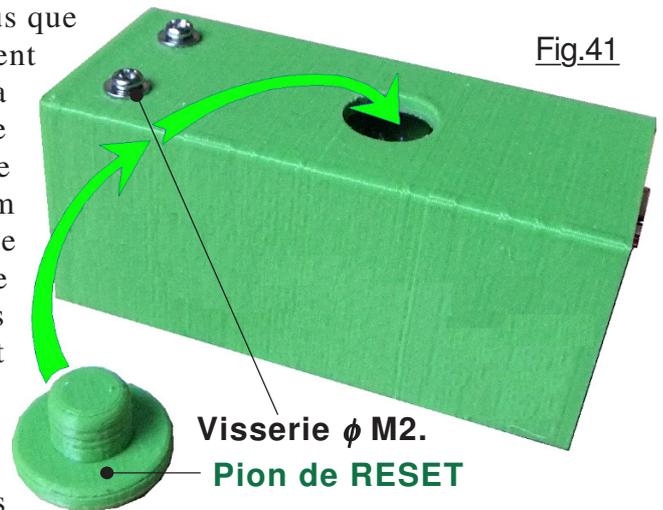


Fig.41

Visserie ϕ M2.

Pion de RESET

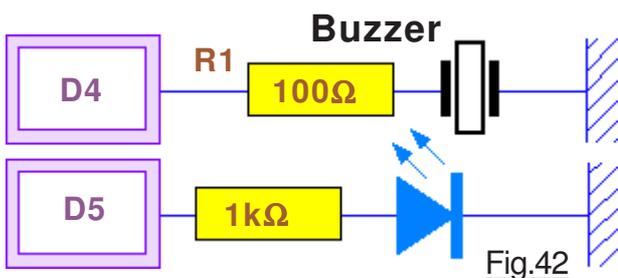


Fig.42

le seuil de fiabilité. De plus, cette carte a subi un incident électronique, la diode d'alimentation de protection sur l'entrée 5Vcc a ... grillé. Aussi, si cette pauvre petite carte fidèle finit par rendre l'âme, la remplacer facilement sera rapide, alors que dessouder les 30 broches serait pratiquement infaisable sans y laisser la santé nerveuse. Par ailleurs, comme la carte **Arduino** est surélevée, le **BUZZER** "passe" dessous ce qui autorise un circuit le plus court possible. La LED et sa résistance de limitation de courant de **1kΩ** ne sont pas encore soudées. En 5 on retrouve le

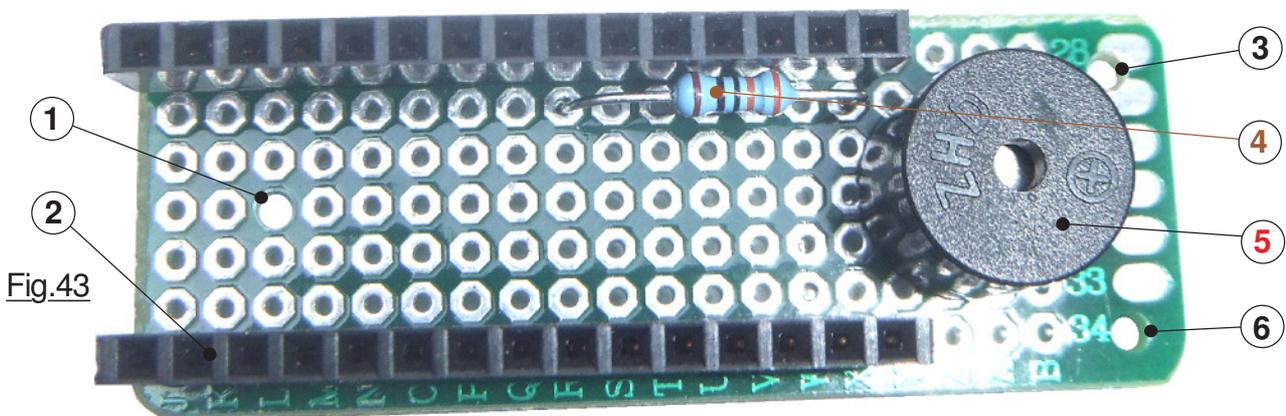


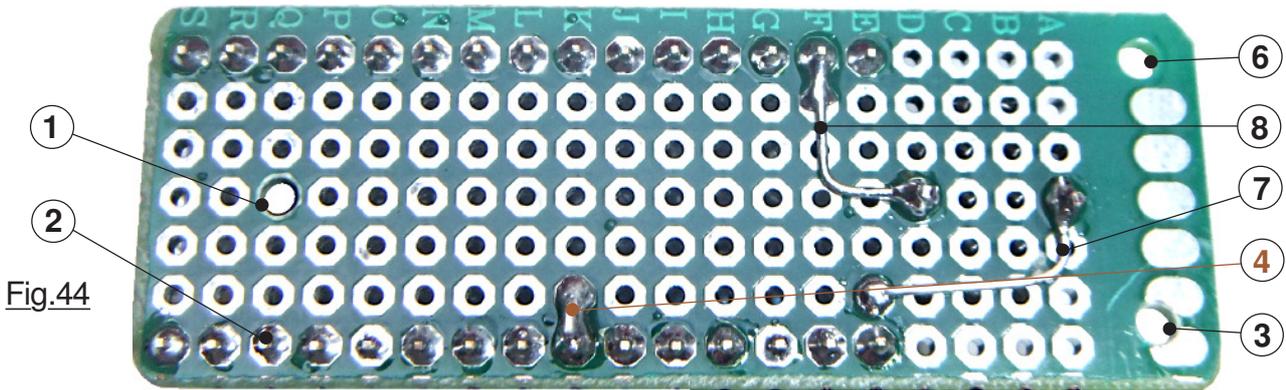
Fig.43

le seuil de fiabilité. De plus, cette carte a subi un incident électronique, la diode d'alimentation de protection sur l'entrée 5Vcc a ... grillé. Aussi, si cette pauvre petite carte fidèle finit par rendre l'âme, la remplacer facilement sera rapide, alors que dessouder les 30 broches serait pratiquement infaisable sans y laisser la santé nerveuse. Par ailleurs, comme la carte **Arduino** est surélevée, le **BUZZER** "passe" dessous ce qui autorise un circuit le plus court possible. La LED et sa résistance de limitation de courant de **1kΩ** ne sont pas encore soudées. En 5 on retrouve le

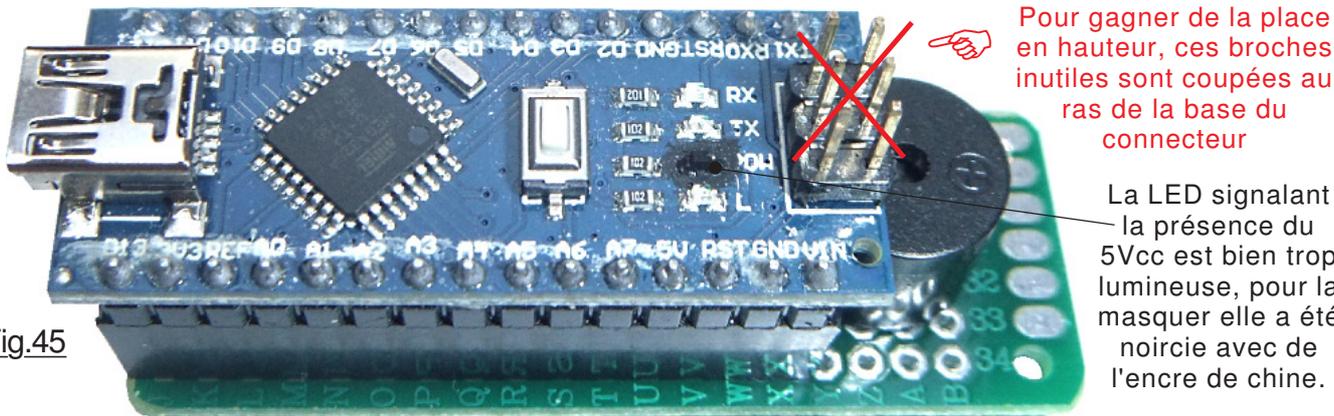
BUZZER *polarisé* avec sa résistance en ligne **4**. Le circuit imprimé est supporté en trois points dont les trous de passage des vis ϕ M2 sont bien visibles en **1**, en **3** et en **6**.

> La vue coté soudures.

Fait assez rare dans mes réalisations pour qu'il soit souligné, ce petit circuit imprimé n'est soudé que d'un seul coté ce qui en facilite les opérations d'assemblage. Il me semble inutile de "rabâcher" ici les divers conseils habituels pour réaliser les soudures. La Fig.44 qui reprend certains repères employés sur la Fig.43 est largement suffisante. On soude la résistance **4**, on ajoute le BUZZER puis les deux lignes HE14 telles que celle située en **2**. Pour réaliser la liaison entre la résistance **4** et le picot de la ligne HE14 **2** je me suis contenté de faire un "paquet de soudure". On continue par la liaison **7** qui dans la pratique est constituée de la queue de **R1** pliées au ras des pastilles de la sérigraphie et coudée à angle droit pour aboutir au picot du BUZZER. Enfin on complète par le fil de liaison **8** constitués par la queue de la résistance **R1** qui a été raccourcie en **4**.



On ajoute la LED et sa résistance de $1k\Omega$ pour aboutir au résultat visible sur [Image 01.JPG](#) que l'on trouve dans le dossier [<Galerie d'images>](#) qui accompagne ce didacticiel. Noter que la valeur de $1k\Omega$ adoptée sur le prototype est déterminée expérimentalement pour obtenir une luminosité correcte tout en consommant le courant le plus faible possible. On complète nos observations en visualisant [Image 02.JPG](#) et [Image 03.JPG](#) alors que sur [Image 04.JPG](#) le mini-coffret a été personnalisé pour cette application. Puis de façon classique on vérifie qu'aucune liaison accidentelle de réunit deux picots voisins du HE14, et que les liaisons prévues soient effectives. On applique **+5Vcc** entre la lyre pour la broche de **D4** et **GND**, le BUZZER doit couiner. On injecte ensuite **+5Vcc** entre la lyre pour la broche de **D5** et **GND**, la LED bleue doit s'illuminer. On peut enfin mettre en place la carte Arduino NANO. Comme montré sur la Fig.45 ou sur [Image 08.JPG](#) on coupe les broches du petit connecteur inutile dans ce projet, et gagner ainsi du volume en hauteur.



Parfaitement visible sur [Image 05.JPG](#) j'ai masqué totalement la petite LED rouge qui sur la carte NANO atteste de la présence du **+5V** régulé. Très lumineuse, ce témoin lumineux est franchement peu utile. En revanche, sa lumière parasite considérablement le signal de la LED 13 qui invite l'opérateur à envoyer une commande sur la ligne série du **Moniteur**. Enfin, il importe de noter que sur [Image 06.JPG](#) et [Image 07.JPG](#) l'entretoise est collée au circuit imprimé avec du vernis à ongles. Ce détail facilite de façon considérable l'intégration de l'ensemble dans le coffret, objet du prochain chapitre. Sur les [Image 09.JPG](#) à [Image 14.JPG](#) on peut voir divers aspect du coffret prototype sur lequel le texte "**Machine de TURING**" n'était pas encore moulé.

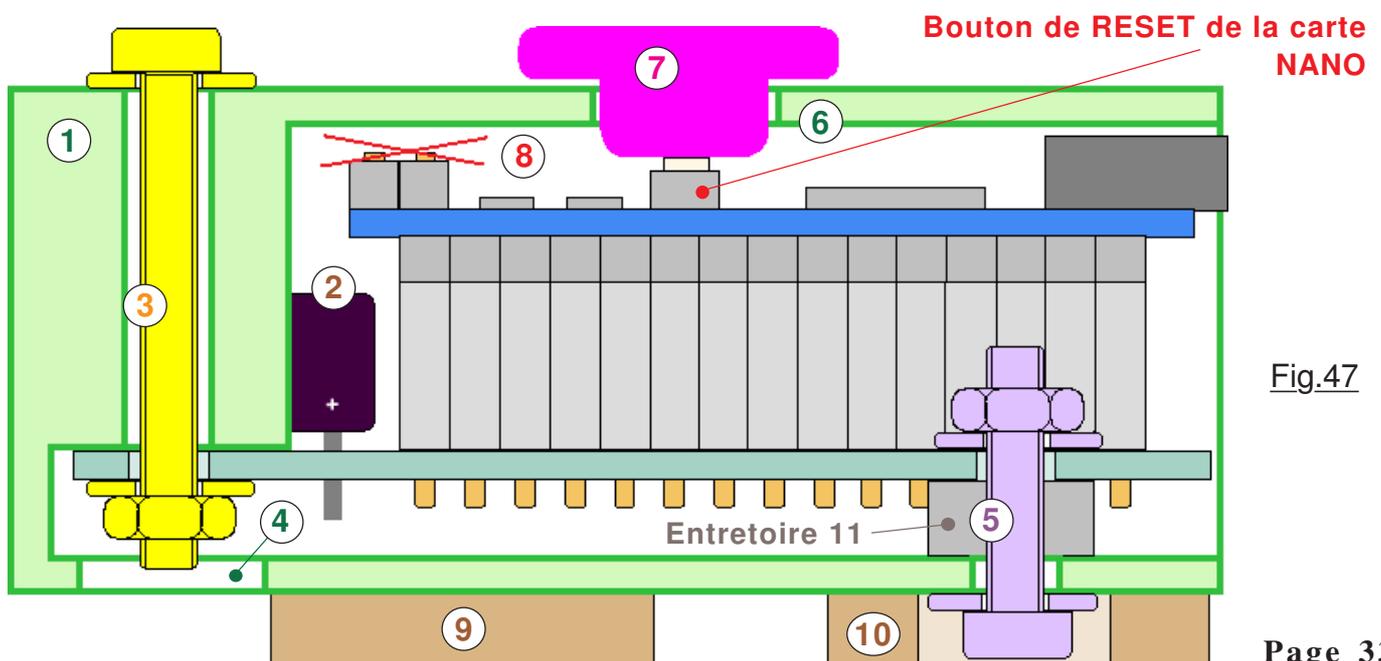
9) Réalisation d'un petit coffret.

Phase ultime de ce projet, je n'ai pas résisté au plaisir de mouler sur mon imprimante 3D un petit boîtier qui englobe entièrement le module électronique, ne laissant visible que la prise USB qui sera reliée par le cordon adaptateur à celle de l'ordinateur. Franchement, on pourrait facilement s'en passer, mais en ce qui me concerne c'est la cerise sur le gâteau. Un simple petit regard sur la Fig.46 indique que l'auteur de ces lignes est marié, *information dont vous vous fichez royalement*, et surtout montre que la concrétisation matérielle est bien modeste et n'encombrera pas du tout le plan de travail. Accompagnant ce didacticiel vous trouverez les fichiers relatifs au moulage de ce coffret dans le dossier <Imprimante 3D>.

ATTENTION : Pour gagner un peu en hauteur, les broches situées sur le dessus de la carte NANO ont été coupées au ras du connecteur HE14, car elles ne servent pas. Cette précision était déjà présente sur la Fig.45 donnée en page précédente.

Sans respecter avec une rigueur absolue toutes les dimensions du coffret, la Fig.47 le présente en coupe pour préciser un peu mieux la solution moulée en 3D. Le fond du boîtier est muni de deux bossages latéraux **1** situés de part et d'autre du BUZZER situé en **2**. Ces deux bossages servent de support au circuit imprimé et sont traversés par les deux boulons **3** de diamètre ϕ M2. Sur le dessous on trouve deux trous **4** facilitant l'introduction des rondelles et des écrous, et surtout de la tête d'une clef de serrage en cloche. En **5** le petit boulon ϕ M2 traverse le dessous et immobilise le circuit imprimé qui s'appuie sur l'entretoise grise **11**. La petite électronique est ainsi solidement bridée à l'intérieur du coffret dont la partie supérieure présente en **6** un orifice de diamètre 10mm pour laisser passer un quelconque stylet prévu pour agir sur le bouton de RESET.

Personnellement je trouve peu commode d'introduire un tel dispositif et d'avoir à doser l'effort pour ne pas exagérer les contraintes sur le micro-switch. Aussi, également moulé en 3D le petit bouton de RESET **7** a été calculé de telle façon qu'en appuyant dessus jusqu'à ce qu'il porte sur le dessus du coffret, il déclenche le RESET avec certitude, sans pour autant amener le mini-bouton poussoir en butée mécanique. On retrouve en **8** les six broches du dessus qui ont été coupées le plus court possible sur le connecteur HE14. La tête de vis **5** dépasse sur le dessous rendant le coffret instable sur sa base. Aussi, deux morceaux de feutre autocollants sont ajoutés en **9** et en **10**. Celui situé en **9** est placé le plus à l'arrière possible sans pour autant masquer les trous de passage **4**. Celui placé en **10** est percé d'un trou central pour dégager la zone de la vis et de la rondelle **5**. Dans le dossier <Galerie d'images> plusieurs photographies complètent cette description.



Présentation de la carte Arduino NANO.

Oups ... mais j'ai complètement oublié de vous présenter la vedette principale du film ! je fais allusion ici à la minuscule carte Arduino NANO. Présentée sur la Fig.48 cette petite chose peut remplacer totalement une carte UNO tout en étant vraiment très petite. On la trouve à des tarifs très variables dans le commerce en ligne, mais des clones dont le fonctionnement est sans reproche sont disponibles pour quelques Euros port compris. Par exemple j'ai commandé un groupe de cinq sur : https://www.amazon.fr/gp/product/B078S8BJ8T/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1

Naturellement il n'est pas du tout obligatoire d'en approvisionner cinq d'un coup, mais comme j'en utilise souvent sur de multiples petites applications, j'ai pris de l'avance !

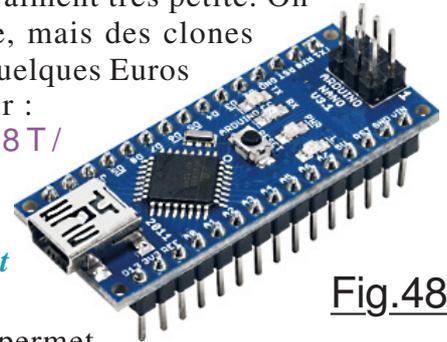


Fig.48

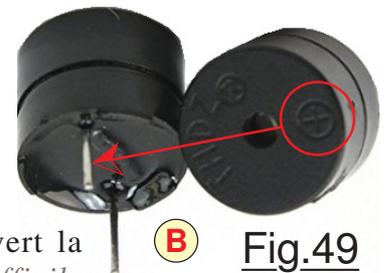
À ce coût elle est moins onéreuse qu'un ATmega328 seul, et permet vraiment d'effectuer tout ce que l'on peut traiter avec une platine Arduino UNO. Des recherches sur la toile préviennent que des clones de cette carte issus de Chine ne sont pas compatibles pour dialoguer avec l'**IDE**. Ce point s'est avéré exacts mais sur d'autres sources. Les cartes dont l'adresse est précisées ci-dessus ont fonctionné sans problème dès leur premier téléchargement. *(Quand je réceptionne de tels composants, immédiatement pour les validés je transfère un petit programme qui fait clignoter à 10Hz la LED en sortie I3.)*

Alors, vu le prix d'achat de cette toute petit chose, qui est plus que compatible avec une Arduino UNO, franchement, si ce n'est pas pour enficher en gigogne un quelconque SHIELD, à mon sens il ne faut pas hésiter. Les cinq exemplaires commandés ont fonctionné parfaitement et ont été livrés très rapidement. Que désirer de plus ?

➤ Composants spécifiques.

Déjà spécifié dans le didacticiel, le petit bruiteur n'est absolument pas indispensable et vous pouvez vous contenter de la carte Arduino totalement seule au bout de la ligne USB de dialogue série. Ceci dit, bien qu'il oblige à réaliser un petit circuit imprimé, ajouter ce BUZZER est un plus incontestable, et du coup on bénéficie de la présence de la LED bleue qui signale l'activité du programme en mode "RUN".

Le chapitre n°10 en page 35 explicite le changement de stratégie qui a consisté à tester un bruiteur passif à la place du buzzer actif qui est vraiment "trop présent.". En fonction des circonstances, vous allez opter pour l'une ou l'autre des deux options. À toute fin utile, je vous livre deux adresse où j'ai approvisionné ce type de composant avec en violet le lien pour le composant actif et en vert la référence pour le composant passif. *(Vous remarquerez qu'il est difficile de trouver des adresses où on ne livre qu'un seul composant ... et c'est souvent plus coûteux.)* :



B Fig.49

https://www.amazon.fr/gp/product/B00GX6YCB1/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1

https://www.amazon.fr/gp/product/B01KO3MD4S/ref=ppx_yo_dt_b_asin_title_o04_s00?ie=UTF8&psc=1

ATTENTION : Les deux références correspondent à des composants polarisés dont la Fig.49 en **B** précise la broche positive, l'autre devant aller sur **GND**. Notons au passage que *le composant passif ne respecte pas l'écartement standard au dixième de pouce*, il faudra "finasser" un peu au moment de réaliser le circuit imprimé.

Pour les résistances et la LED bleue il y a tellement de sources sur les étagères du commerce en ligne, qu'il me semble inutile de préciser ici mes sources d'approvisionnement. Il est probable que le rendement de votre diode électroluminescente ne sera pas équivalent à celui de mon prototype. Aussi, avant de souder la résistance de limitation de courant je vous suggère fortement de faire un essai préalable pour sélectionner la valeur idoine.

Enfin, comme nous en sommes à approvisionner les composants indispensables, personnellement je réalise mes circuits imprimés avec des plaquettes préperçées telle que celles dont voici une adresse possible pour le commerce en ligne :

https://www.amazon.fr/planches-bricolage-prototypes-diff%C3%A9rentes-compatibles/dp/B0734XYJPM/ref=d_qpr_scc1_1_3/259-2726334-0830566?pd_rd_w=xsCV8&content-id=amzn1.sym.d0d30fa7-dde4-45c7-b2ca-23e4d858e154&pf_rd_p=d0d30fa7-dde4-45c7-b2ca-23e4d858e154&pf_rd_r=1Y66RZ9DDB5SD6RJYAQ5&pd_rd_wg=rCtDo&pd_rd_r=550750f3-4cb0-44cd-a0fb-20793e7ca9a3&pd_rd_i=B0734XYJPM&psc=1

10) Changement de stratégie. *(Lecture pour celles et ceux qui ont du temps de libre.)*

Initialement, c'est un BUZZER actif de fréquence de l'ordre 4000Hz qui a été implanté sur le première version du prototype. Même avec une résistance de 100Ω, il fait sursauter quand il génère son BIP sonore. Du coup, dans le menu des Options était prévu avec la commande "D" pour Discret, de pouvoir le couper. La séquence de génération du BIP allumait alors la LED bleue pour que l'opérateur puisse avoir un avertissement visuel. Le code était le suivant :

```
boolean BIP_Sonore,
void setup() {... BIP_Sonore = true; ...
void BIP() {
  if (BIP_Sonore) digitalWrite(BUZZER, HIGH);
  for (byte l=0; l < 4; l++) {
    digitalWrite(LED_bleue, HIGH); delay(6);
    digitalWrite(LED_bleue, LOW); delay(50);}
  digitalWrite(BUZZER, LOW);}
void Resumer_les_OPTIONS() { ...
  Serial.print(F("BIP Sonore : ")); AFF_OUI_NON (BIP_Sonore);
void Menu_des_OPTIONS() { ...
  Serial.println(F("D : Discret. (Plus de BIPs sonores.)"));
void OPTIONS() { ...
  case 'D' : {BIP_Sonore = !BIP_Sonore; break;}
```

L'adoption d'un bruiteur piezzoélectrique passif a rendu la "nuisance sonore" du BIP d'avertissement bien plus supportable. Du coup, il devenait inutile d'ajouter l'option "D" ce qui a engendré un important gain en taille de programme. Tout le code étalé ci-avant avec son booléen et la gestion de l'Option "D" a été remplacé par :

```
void BIP() {
  for (int l=0; l < 700; l++) {
    digitalWrite(BUZZER, HIGH); delayMicroseconds(130);
    digitalWrite(BUZZER, LOW); delayMicroseconds(130);}}
```

Du coup, on dispose de plus de place pour gérer le reste du programme. Au lieu de me contenter de modifier  `Machine_elementaire.ino` j'avais envisagé un deuxième programme nommé  `Buzzer_Passif` laissant ainsi la possibilité au lecteur d'utiliser à sa convenance l'un ou l'autre des deux bruiteurs envisageables.

Enfin, on remarque que dans la version actuelle on génère un signal découpé d'environ 3000Hz. La tonalité sur le BUZZER passif est agréable et le niveau sonore parfait. Il se trouve que si l'on soumet le bruiteur actif au même régime, le son émis devient du même ordre de discrétion. Alors les deux variantes de logiciel ne se justifient plus. On n'utilisera plus que

 `Machine_elementaire.ino` ce qui simplifie d'autant la liste des fichiers fournis et les protocoles de téléversement du programme dans l'ATmega328. Vous conservez dans tous les cas le choix du bruiteur ... l'actif étant au pas standard du dixième de pouce.

11) Un ou deux petits détails informatiques. (Réservé aux programmeurs.)

Généralement, sur un gros projet, je joins un petit livret contenant les explications détaillées sur le logiciel, avec organigramme des procédures principales. Pour cette fois, je ne suis pas certain que ce soit vraiment utile, car le but n'est pas d'apprendre à programmer Arduino. Toutefois, je ne dois pas exclure que certaines ou certains puissent avoir envie de modifier le programme pour ajouter une ou plusieurs petites fonctions. La première aide réside dans la façon dont je développe un programme. Très inspiré du langage le plus "propre" que j'ai pratiqué il y a longtemps, le PASCAL, j'en ai gardé des réflexes lors du codage en C++. Par exemple toutes les constantes et les variables sont déclarées en tête de listage et sont inscrites par ordre alphabétique. Sauf exception j'évite comme une plaie les procédures ou les fonctions qui comportent un nombre trop grand de lignes d'instructions. ***J'estime qu'il ne faut pas dépasser en "hauteur" la taille disponible dans la fenêtre de l'IDE pour avoir intégralement le listage sous les yeux.*** Quand une procédure invoque une autre procédure, l'appelée est située avant, du coup vous savez où la chercher. Je m'impose l'utilisation d'identificateurs évocateurs de façon à ce qu'en lisant le nom de la procédure vous compreniez immédiatement à quoi elle sert. Enfin, dans les procédures je repère certaines séquences spécifiques par des remarques de séparation. Je saupoudre un maximum de remarques pour préciser certaines instructions ou justifier des choix effectués qui peuvent sembler non optimisés. Si pour une quelconque raison je dois souvent modifier une constante, pour la retrouver facilement dans le listage je la jalonne avec une remarque visuelle du type `//@@@@@@@@@@@@@@@@@`.

➤ **L'optimisation du programme.**

Généralement, quand j'engage mes heures de loisir sur un gros projet contenant du logiciel, je cherche à en optimiser le code à l'OCTET près. C'est un réflexe conditionné de programmeur qui dans la majorité de sa vie a utilisé l'ASSEMBLEUR. Dans ce programme comme dans tous les autres, j'optimise à outrance le choix du type des données. Habituellement j'optimise intensément le codage des procédures. Par exemple, pour traiter un BIP() sonore je préfère réaliser une petite boucle avec deux `delay()` au lieu d'utiliser la facilité d'un `tone()`. Toutefois, je me suis rendu compte assez rapidement que j'aurais plus de place qu'il n'en faut pour satisfaire toutes mes envies. ***Aussi, j'ai dilapidé de façon scandaleuse la place disponible pour le programme*** dans l'ATmega328. Le programme `Machine_elementaire.ino` est ***l'exemple typique de ce qu'il ne faut pas faire***. En effet, ***ce qui gaspille le plus de place dans un programme C++ ce sont les textes que l'on fait afficher***. Non seulement ce programme est le plus "bavard" de tous ceux que j'ai concocté jusqu'à présent, mais il est truffé d'accentués. Hors chaque accentué oblige à couper le `Serial.print(F("Teste à afficher"))` en deux appels à procédure, mais il faut en intercaler une de plus pour l'accentué sous la forme `Serial.print(char(224))`. Par ailleurs, au lieu de minimiser les textes affichés, j'ai profité de l'abondance pour étaler des phrases absolument pas "compactées".

CONCLUSION : Si vous devez dégager de la place pour ajouter des commandes personnelles le moyen le plus radical sera d'***éliminer les options "\$" et "w"*** et de ***commencer à chasser tous les accentués***. Faire des textes de dialogue plus simples. De plus, les menus peuvent être sans encadrés d'étoiles. De simples listages en ligne engendrent une économie de place considérable.

➤ **Méga radin le Nulentout.**

L'ATmega328 laisse disponible 30720 octets de disponibles au programmeur pour loger son logiciel. Si on en n'utilise que la moitié, le reste ne nous sera jamais remboursé ! Aussi, je les ai payés ... je les utilise ! Bon, ce n'est pas vraiment ma motivation. Personnellement, j'y vois un aspect d'élégance, c'est à dire faire traiter un maximum de choses au programme. Quand vous compilez le sketch, vous obtenez un résultat qui ressemble à celui de la Fig.50 sur laquelle on peut vérifier que j'ai consommé la totalité de l'espace disponible. C'est à mon sens le summum de l'élégance informatique au point de vue de la "rentabilité" du programme. Notez au passage qu'***il n'y a aucun risque et aucune pénalité à utiliser la totalité de la place disponible pour le programme***. Aussi, tant que le compilateur ne râle pas, vous pouvez en tartiner des couches et des couches. Ce n'est que si le compilateur se fâche tout rouge comme sur la Fig.51 qu'il faut faire profil bas ! Le premier moyen et de loin le plus facile consiste à modifier les textes. Ceci dit, comme je savais

```

162 Retour_cadre(); Serial.print(F("< : Rotation ");A_accent()); Serial.print(F(" gauche du BÂRILLET.")); N
163 Serial.print(F("** W : (Word) Listage du contenu EEPROM.  ")); Retour_cadre();

```

Fig.50

Compilation terminée.

Le croquis utilise 30720 octets (100%) de l'espace de stockage de programmes. Le maximum est de 30720 octets. Les variables globales utilisent 813 octets (39%) de mémoire dynamique, ce qui laisse 1235 octets pour les variables locales. Le maximum est de 2048 octets.

130 Arduino Nano, ATmega328 sur COM3

Erreur de compilation pour la carte Arduino Nano

Fig.51

Le croquis utilise 30754 octets (100%) de l'espace de stockage de programmes. Le maximum est de 30720 octets. Les variables globales utilisent 813 octets (39%) de mémoire dynamique. **Croquis trop gros** : voirs <http://www.arduino.cc/en/Guide/Troubleshooting>. Erreur de compilation pour la carte Arduino Nano

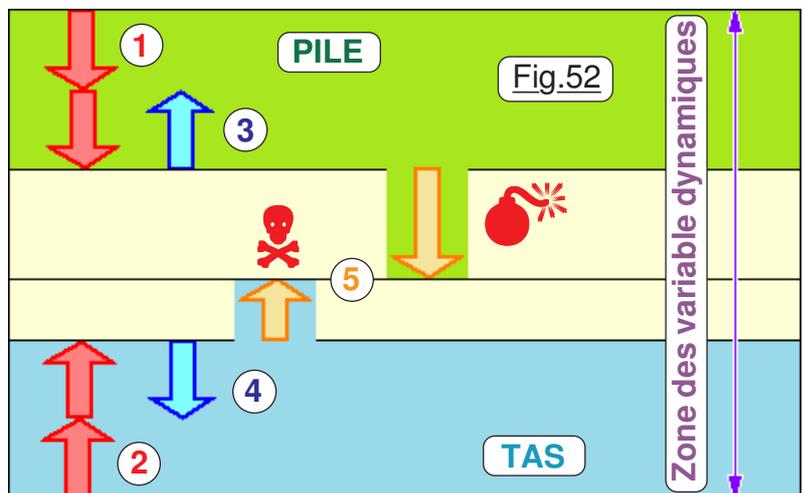
OUPS, il faut faire maigrir le code de 34 octets pour que ça passe. Il suffit d'enlever trois accents pour y arriver !

que le manque de place ne serait pas un obstacle, j'ai gaspillé sans compter. En particulier il y a une foule de séquence qui se répètent. Chaque fois je me suis contenté de les coder directement, au lieu de faire une procédure utilisé plusieurs fois. J'insiste : Pour les données, c'est du bon travail, pour le reste j'ai donné libre cours à une paresse intellectuelle condamnable ...

> La PILE : Tout programmeur doit s'en préoccuper.

Visitions un court instant l'intimité du fonctionnement d'un programme. Chaque fois que le logiciel fait appel à une procédure, le microcontrôleur sauvegarde sur la **PILE** l'adresse de retour pour pouvoir continuer quand cette dernière est terminée. Si une procédure en appelle une deuxième, qui en invoque une troisième, toutes ces adresses de retour s'empilent les unes sur les autres. Puis, en retour de "subroutine" elles sont dépilées et l'emplacement sur la **PILE** libéré. Chaque fois qu'une procédure utilise une variable locale, on réserve de la place sur le **TAS**. Plus il y a de variables locales dans les procédures qui s'invoquent les une et les autres, plus on va entasser des octets de données. Naturellement, en retour de subroutine on libère la place sur le **TAS**. La Fig.52

schématise le fonctionnement interne du microcontrôleur pour la gestion des variables dynamiques. Le haut de la zone en vert est réservé à la **PILE** qui progresse "vers le bas". Par exemple en 1 on a empilé deux adresses de retour. Quand la deuxième subroutine se termine, son adresse de retour est récupérée et l'emplacement libéré en 3. En 2 ont été entassées deux variables locales. En retour d'une subroutine, en 4 la place de la variable qui n'est plus utilisée est libérée. Au cours du programme, la zone non utilisée jaune se rétrécit à chaque



appel de procédure ou de fonction, et se réduit pour chaque variable locale. Si l'on a trop de procédures qui appellent des procédures et qui entassent des données locales trop volumineuses, arrive un moment où il y a, comme en 5, **COLLISION de PILE**. Le **TAS** et la **PILE** se rencontrent, l'un écrasant les données de l'autre. **Le programme devient totalement incohérent et se met à faire n'importe quoi.** C'est un problème particulièrement sournois, car strictement rien ne prévient le programmeur. Par exemple dans une procédure vous remplacez une variable locale qui était un **byte** par un **int**. Imaginons que le contexte à empilé et entassé tellement de données qu'il ne restait plus de place. Maintenant la variable locale fait un octet de plus. Sur un **RUN** se produit alors la collision tant destructrice. Votre programme se met à faire n'importe quoi. Vous avez beau analyser, analyser et analyser encore, le croquis est propre et en aucun cas ne peut expliquer le comportement du logiciel.

CONCLUSION : Il faut impérativement s'assurer que ça n'arrivera pas.

➤ **Prendre une assurance vie pour la PILE.**

Parer le risque de **COLLISION de PILE** est tellement important, que le C++ d'Arduino met à notre disposition un outil spécial pour évaluer les risque encourus. Aussi, chaque fois que je termine un gros logiciel qui invoque des sous-routines à la pelle, avec des variables locales nombreuses, avant de considérer qu'il est fiable j'en teste la pile, ou plus exactement l'amplitude de la zone jaune qui reste quand le programme a fait son RESET. Le petit programme qui effectue ce travail consomme environ 140 octets. Aussi, sur un logiciel avec aucun gaspillage il n'y a pas la place. Je me contente de passer en remarque certains affichages, et valide la procédure. Puis elle repasse en remarques et je restitue les affichages. Dans notre application, vu que je gaspille à outrance, l'outil d'évaluation de la **PILE** est resté actif et de plus encombre le **MENU des OPTIONS**.

MANIPULATIONS :

- 1) Faire un RESET puis "o" pour passer dans le **Menu des OPTIONS**.
- 2) Commande "w" et obtenir le résultat de la Fig.53 qui annonce la place qui reste dans la zone jaune à ce stade précis du déroulement du programme. (*Adresse de retour de la procédure de "calcul" comprise et entassée dans la PILE.*)

```
Option -> [W]
>> Il reste actuellement 1150 octets dans la PILE de la mémoire dynamique. <<
```

Fig.53

Quand je programmait en ASSEMBLEUR, l'expérience m'avait largement convaincu que 50 Octets étaient suffisants pour ne pas risquer une collision de plus. Depuis que je programme en C++, j'estime expérimentalement que 150 Octets sont amplement suffisants, à condition toutefois d'avoir optimisé les variables locales. (*Ne pas passer en local un tableau de 50 éléments par exemple.*) Alors avec 1150 octets de disponibles on peut vraiment dormir sur nos deux oreilles.

➤ **Une petite astuce logicielle.**

Comme déjà précisé, la rédaction d'un livret décrivant les routines principales n'est pas estimée pertinente pour cette application. Toutefois, et si mes libertés ludiques le permettent, j'envisage de développer une deuxième version qui sera totalement autonome et qui n'imposera plus l'utilisation d'un ordinateur pour dialoguer avec la machine. Ce sera une application plus importante car il faudra ajouter un afficheur, un clavier et créer un coffret bien plus complexe. Une telle réalisation si elle voit le jour méritera un document informatique de type livret. Il est clair que toutes les routines de l'interface HOMME / MACHINE seront à créer. Mais celles propres au traitement des données de la Machine de Turing seront reprises sans modification. On peut donc croire que dans un avenir pas trop lointain le manque de description logicielle sera comblé.

Transformer une lettre minuscule en majuscule est très fréquent quand on programme des applications qui impliquent du dialogue HOMME/MACHINE. Hors, je n'ai pas trouvé d'instruction qui en C++ d'arduino effectuerai ce transcodage. Une telle modification est prévue pour des chaînes de caractères, (*Supposées codées en ASCII.*) mais pas pour des données de type **char**. Il existe toutefois un moyen très simple d'y parvenir. En effet, quand on observe une table de codage en ASCII on constate que la seule différence entre lettre MAJuscule et lettre MINuscule se situe sur le sixième BIT, celui de poids décimal 32. La technique consiste à enlever 32 au code ASCII du caractère par l'entremise d'un ET logique. Exemple :

```
if ((Caractere > 96) && (Caractere < 123)) Caractere = Caractere & 223;
Caractere >= a      Caractere >= z      Caractere = Caractere - 32;
```



"Dormir sur vos deux oreilles." Elle est vraiment stupide cette phrase de Nulentout. Vous avez tenté la chose ? Ou alors c'est que vos deux oreilles elles sont très longues et toutes moles !

12) Démarrer quand on ne connaît strictement rien d'Arduino.

Envisageons le cas d'une personne totalement naïve au sens noble de ce terme, mais qui par l'entremise d'un ami lointain s'est fait programmer convenablement une carte NANO. Cette connaissance habite à distance et n'a pas forcément le loisir de venir installer le contexte de l'IDE sur l'ordinateur. Ce chapitre traite de cette éventualité tout à fait vraisemblable. On part du principe que vous ne saviez même pas qu'existait la ferveur mondiale pour les cartes Arduino qui permettent de programmer de façon très facile le microcontrôleur ATmega328 pour lui faire gérer des petites applications de loisir. (Encore que dans le domaine professionnel il tient aussi le haut du pavé.) Le but de ce chapitre consiste à vous faire installer l'IDE sur votre ordinateur pour pouvoir y brancher le petit système avec une carte Arduino NANO dans laquelle votre ami a logé le programme et les données en mémoire EEPROM. L'IDE, est un environnement qui par l'entremise d'une prise USB de votre ordinateur permet de programmer en langage C++ les cartes électroniques de la famille ARDUINO. Vous ignorerez l'Editeur de texte et le compilateur. La seule fonctionnalité qui vous concerne est le **MONITEUR**. Mais pour en disposer il faut l'IDE, raison de ce chapitre.

➤ Installer l'IDE sur l'ordinateur.

L'environnement IDE est un ensemble de modules informatiques très propre qui n'interfère absolument pas avec Windows. Il est totalement autonome et peut parfaitement être installé sur une simple clef USB. Ceci dit, avec la capacité de stockage des mémoires de masse actuelles, nous n'en sommes plus à 1Go près. Pour ma part, j'utilise depuis des années la version 1.7.9 qui tourne bien sur ma machine gérée par l'ancien Windows VISTA et téléchargée sur :

<http://www.arduino.org/software#ide>

On commence par accepter les cookies. Cliquer sur l'onglet [Software]. La version actuelle proposée est la version 2.0.1 qui fonctionne en 64 BITS sur Windows 10. Je ne connais pas cette version n'ayant pas ce système d'exploitation. Quand on explore la page, vers le bas on trouve des versions plus anciennes. Curieusement, dans les références poussiéreuses on trouve les 1.6.n et les 1.8.n mais pas celle que j'utilise. À vous de

trouver la version qui ressemblera le plus à ce que je décris et surtout qui "tourne" sur votre ordinateur. Pour ce tutoriel j'ai téléchargé la version 1.8.0 pour voir ce que ça donne. Pour l'expérience, comme vous pouvez le vérifier sur la Fig.54 le fichier compressé de 153Mo en 1 a été logé sur une petite clef USB de 8Go. Extrait directement en 2 sur cette petite mémoire de masse extérieure le dossier pèse 410Mo. (Pour vérifier que l'IDE pouvait fonctionner sur un support extérieur à l'ordinateur, j'avais en 3 recopié le dossier d'utilisation "normale" qui réside sur mon P.C. de bureau.)

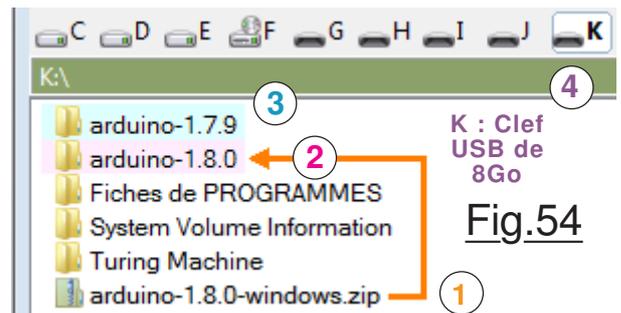


Fig.54

➤ Activer l'IDE sur l'ordinateur. (Ou sur un disque externe.)

C'est de loin la phase la plus compliquée de l'opération. Explorateur de Windows activé, il faut cliquer sans se tromper sur le dossier 2 qui ouvre une liste effrayante de fichiers. Dans cette liste, on cherche avec fébrilité un exécutable intitulé  **arduino.exe**. Courage, vous allez y arriver, on y est presque ! Cliquer nerveusement deux fois rapidement sur le nom de ce fichier étrange. Le truc étrange de la Fig.55 s'active avec plein plein plein de textes inquiétants en bas à gauche en A. Comme ces textes ressemblent à un compte à rebours, je me suis

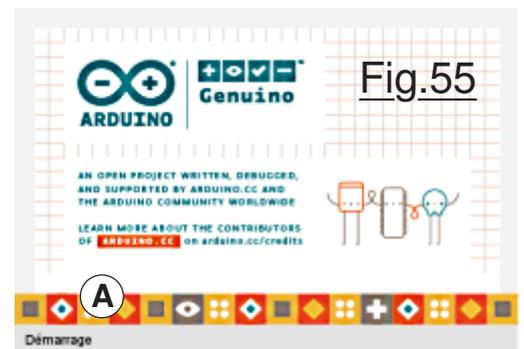


Fig.55

planqué sous le bureau et j'ai attendu une heure. Rien ne s'est passé, à part des crampes dans les jambes. Et,

Ben Mômôa j'ai une superbe IDE en tête. Je vais vendre mes portes-clefs et mes boucles d'oreilles en ligne sur Internet.

miraculeusement, quand je suis sorti de ma planque, sur l'écran de l'ordinateur il y avait la fenêtre contextuelle de la Fig.56 qui est exactement identique à celle qui s'ouvre quand j'active ma vieille version antédiluvienne 1.7.9 du compilateur C++.

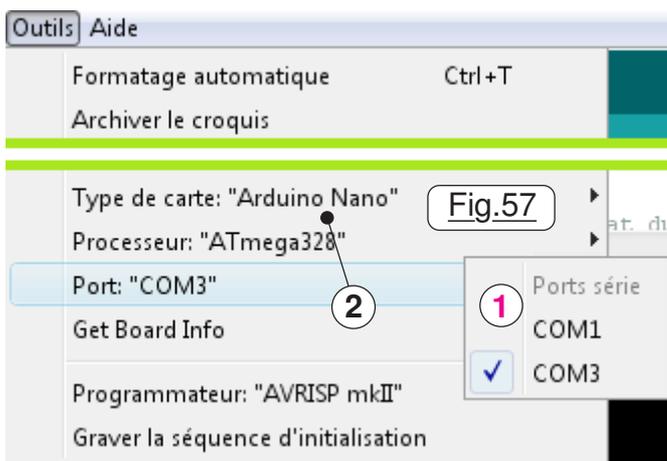
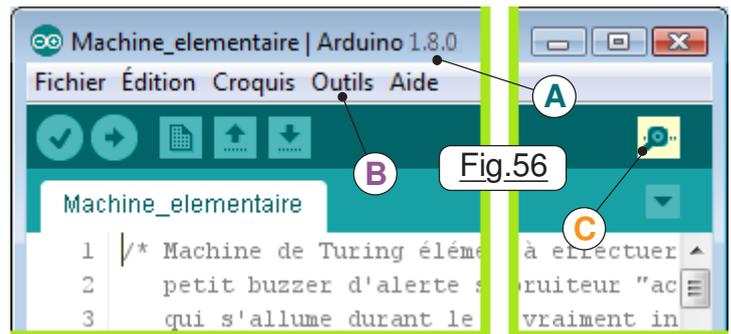
Par curiosité, j'invoque dans ce nouvel environnement IDE, car en fait c'est cette fenêtre verte qui constitue le domaine de développement Arduino, le programme **Machine_elementaire.ino** et je constate que le listage est un tout petit peu différent. Avant, tous "les mots" du langage étaient en couleur orange. Maintenant, les données sont en orange, mais les identificateurs des procédures et des fonctions sont en bleu clair. Pas de quoi sauter au plafond. Puis, m'attendant à une compilation strictement identique, j'engage une compilation. Le temps de traitement me semble un peu plus long, mais pas de beaucoup. Puis le résultat s'affiche : 27226 octets de programme au lieu de 30720 avec la version 1.719 soit 3494 octets de moins. Je n'en crois pas mes mirettes ! Pourtant dans la fonction "type de carte" j'ai bien validé la NANO. Je téléverse la compilation et je teste.

- GLUPS ... ça tourne !

CONCLUSION : Les développeurs ont sacrément optimisé le code généré par le compilateur. C'est une vraie catastrophe, car il reste 3494 octets non utilisés, et je ne livre pas un produit avec un tel gaspillage. Il me faut tout reprendre pour ajouter des perfectionnements et tout consommer. Dommage que je n'ai pas fait cette découverte il y a longtemps. Du coup, l'essayer c'est l'adopter.

> Activer le Moniteur de l'IDE.

L'historique de ce développement vous n'en avez cure, et vous avez bien raison. La seule chose qui vous importe, c'est d'avoir sur l'écran la fenêtre contextuelle du **Moniteur** de l'IDE et que le dialogue s'installe entre la petite machine dans son coffret vert et l'écran de l'ordinateur. Après avoir effectué les manipulations du chapitre précédent, la fenêtre de l'IDE montrée en Fig.56 est active sur l'écran de l'ordinateur. Branchez la petite carte Arduino NANO avec sa mini prise USB sur l'un des ports libres de l'ordinateur. Puis validez l'onglet **[Outils]** en **B**. Si le port est reconnu, on obtient un résultat qui ressemble à celui de la Fig.57 prouvant que la liaison série est correctement établie. Éventuellement si la bonne prise n'a pas été sélectionnée, il suffit de la valider en **1**. Bien qu'en principe ce ne soit pas du tout important si on ne compile pas, on va **par mesure de précaution** indiquer le type de carte raccordé. Dans ce but on active à nouveau l'onglet **[Outils]** en **B** et avec l'item



l'item on valide l'option Arduino Nano . Il ne reste plus qu'à établir le dialogue machine entre les deux entités informatiques. Il suffit tout simplement de cliquer sur le symbole **C** mis en évidence en jaune sur la Fig.56 en haut à droite. Immédiatement la fenêtre du **Moniteur** de l'IDE s'ouvre et affiche le **MENU de BASE**. (Le symbole représente une loupe qui analyse du texte, car le **Moniteur** est l'outil de base pour un programmeur qui sert à déverminer le logiciel qu'il est en train de développer.)

-HOOOooo NONNnnnon, l'écran affiche n'importe quoi !

Pas de panique, c'est normal. Le logiciel qui "tourne" sur **Machine_elementaire.ino** est initialisé à la plus grande vitesse possible sur la voie série. Par défaut dans l'IDE il peut être totalement différent. Donc, vous consultez le protocole décrit dans la Fig.59 du chapitre suivant. Vous effectuez un nouveau RESET ou vous cliquez une deuxième fois en **C**, ce qui revient au même, et enfin sous vos mirettes émerveillées s'affiche ce sacré **MENU de BASE ... CHAMPAGNE !**

13) Loger les données dans les cellules "grises" de l'ATmega328.

Probablement qu'une grande majorité des lectrices et des lecteurs Internautes savent parfaitement téléverser des programmes et des données sur une carte Arduino. Toutefois, je vais détailler la procédure pour les Naïfs, (*Naïf étant ici à prendre au sens noble du terme.*) les connaisseurs pouvant passer "en diagonale". Comme cette application impose de gaver des données dans la mémoire EEPROM du microcontrôleur, on doit procéder en deux étapes :

- Remplir l'EEPROM avec les 15 algorithmes prévus pour les démonstrations.

Comme cette opération impose de charger dans l'ATmega328 un petit programme spécifique, on commence par cette étape, car elle "écrasera" tout logiciel déjà présent.

- Téléverser le programme d'exploitation.

➤ **Inscrire 15 algorithmes dans l'EEPROM.**

Quel que soit le support dans lequel vous avez décompressé les fichiers de cette application, aller dans le dossier <Les programmes Arduino\Initialiser_EEPROM> puis :

- 1) Cliquer sur **Initialiser_EEPROM.ino** qui active l'IDE sur le **sketch** à téléverser.
- 2) Brancher la ligne USB puis cliquer sur l'onglet [**Outils > Port**] **COMn** correspondant.
- 3) Cliquer sur l'onglet [**Outils > Type de carte**] et validez **Arduino Nano**.

- 4) Cliquer en **A** sur la flèche colorisée en rouge sur la Fig.58 pour inscrire le programme dans la mémoire du processeur.
- 5) Cliquer en **B** de l'autre coté de la fenêtre contextuelle pour exécuter ce programme et voir le résultat s'afficher dans la fenêtre du **Moniteur** qui vient de s'activer.

Le logiciel précise la version du contenu qui sera inscrit dans

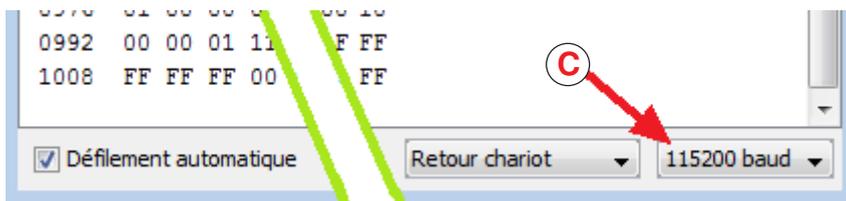
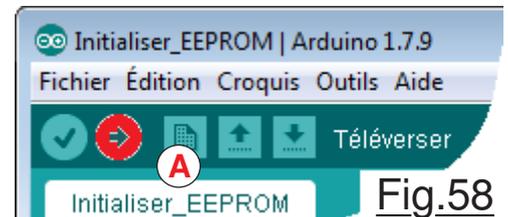


Fig.59



l'EEPROM. Puis, il faut patienter quelques secondes car l'inscription des 1024 Octets exige un petit intervalle de temps. Quand c'est terminé, une foule d'informations se bousculent. *Il est possible, voir probable, que l'ensemble de cet affichage puisse s'avérer incohérent.* C'est tout simplement que la vitesse de dialogue entre le **Moniteur** et Arduino n'est pas synchronisée. Dans ce cas, il suffit d'imposer en **C** de la Fig.59 une vitesse de 115200 bauds, car par logiciel je privilégie la cadence de transferts la plus rapide possible dans les options de l'IDE. Quand cette vitesse est correcte, cliquez une deuxième fois en **B** et les 1024 données sont réinscrites, ce qui n'est pas grave du tout, puis les quatre blocs de mémoire EEPROM sont affichés. Inutile de vous torturer l'esprit pour le moment sur l'analyse de ces données, des manipulations à cet effet ont précédé en Fig.6 de la page 6 du didacticiel.

➤ **Inscrire le programme d'exploitation dans le microcontrôleur.**

Pour cette deuxième étape, on commence par fermer la fenêtre contextuelle de l'IDE pour la rouvrir sur le logiciel d'exploitation du microcontrôleur. Donc, toujours avec l'explorateur de WINDOWS, on visualise le dossier <Les programmes Arduino\Machine_elementaire>.

- 1) Cliquer sur le programme d'exploitation **Machine_elementaire.ino** qui active l'IDE sur le **sketch** à téléverser dans l'étape qui suit.
- 2) Cliquer en **A** sur la flèche colorisée en rouge sur la Fig.58 pour inscrire le programme d'exploitation dans la mémoire du microprocesseur de la carte Arduino NANO.
- 3) Cliquer en **B** de l'autre coté de la fenêtre contextuelle pour exécuter ce programme et voir le résultat dans la fenêtre du **Moniteur** qui affiche la version du logiciel et déroule l'imposant "cadre" du **MENU de Base** de la Machine de Turing virtuelle.

Normalement l'affichage doit être cohérent puisque la vitesse du Moniteur de l'IDE a été initialisée dans la phase d'initialisation des données dans l'EEPROM. C'est fait, la petite unité est parée pour dérouler des centaines d'algorithmes.

14) Les "loupés".

Aucun projet, qu'il soit industriel ou de loisir ne saurait aboutir à la perfection absolue. Entre les désirs initiaux, ce qui était envisagé et ce qui résulte de compromis inévitables, s'insinuera forcément des divergences. Cette petite Machine de Turing échappe presque à ce principe non démontré mais qui frise l'absolu. Il a fallu composer avec les réalités matérielles, et l'on peut noter à peine un tout petit "regret" sur trois "manques" qui dans ce projet restent vraiment dérisoires : Ayant dilapidé beaucoup de place pour les affichages de textes à l'écran, il n'en restait plus assez pour implémenter les fonctions supplémentaires :

- * *Pouvoir sauvegarder en EEPROM plusieurs programmes en mode Machine étendue.*
- * *Dans l'éditeur pouvoir placer un Jalon et y revenir librement.*
- * *Dans l'éditeur pouvoir à convenance sauter en ligne N.*

Il est du reste rare, que sur un projet de cette "envergure" je ne trouve que si peu de points sombres, et aussi dérisoires. Au final, dans cette petite boîte verte j'ai intégré bien plus de fonctions que je ne l'avais imaginé initialement. À mes yeux, le logiciel correspond largement à ce que je désirais. Franchement je ne vois pas ce que l'on pourrait ajouter. Du reste, pour arriver à saturer la mémoire de programme, j'ai dilapidé un nombre d'octets faramineux dans les textes affichés et laissé "trainer" dans le code les deux fonctions qui ne présentent un peu d'intérêt que pour les programmeurs purs et durs. Bref, mission accomplie. La seule faiblesse de ce petit bloc vert, c'est qu'il n'est pas autonome, il lui faut un ordinateur pour dialoguer avec l'utilisateur ... Vlà une bonne idée pour reprendre le collier : Créer un équivalent Arduino, mais totalement indépendant. Ce sera mon prochain projet.

Peu importe ces petits détails, d'autant plus que pour le petit regret souligné, et encore, *c'est par tradition que je termine mes tutoriels par cette rubrique et qu'il fallait trouver quelque chose*, il a été finalement possible de sauvegarder l'un de ces programmes en mode "Machine Étendue" qui je pense ne sera pas du tout le plus utilisé, car il ne peut être testé sur la machine matérielle. Pouvoir en conserver un pour travailler plusieurs jours dessus est déjà pas si mal que ça. Ne boudons pas notre plaisir. La petite machine virtuelle est bien vivante et va nous octroyer le plaisir incontestable de programmer "en Turing" durant les longues soirées d'hiver. Sa performance nous autorisera des fonctionnements virtuels de plusieurs "années théoriques" sans coupure de courant. On pourra donner libre cours à notre imagination et proposer des algorithmes et des configurations de BARILLET bien cossus. Bref, de nombreuses heures de loisir en perspective pour un investissement vraiment dérisoire de quelque Euros.

Chères lectrices, chers lecteurs, cette longue présentation arrive à son terme. Tout à une fin, mis à part l'Univers, et arrive forcément un moment où il faut raisonnablement considérer que "le voyage d'agrément" est terminé.

Je souhaite intensément que certaines et certains oseront s'engager dans la réalisation d'un clone, je ne doute pas de leur réussite. Surtout, je vous souhaite à toutes et à tous de trouver dans ces lignes le plaisir de la découverte. Si d'aventure vous engagez vos heures de liberté dans une telle réalisation et que vous rencontriez une difficulté, les amis du forum pourront probablement vous aider. Dans le pire des cas, vous pouvez me contacter sur : michel.droui@laposte.net et dans les limites de mon temps de libre, c'est avec grand plaisir que je tenterai de vous dépanner. Je vous souhaite à toutes et à tous agréable lecture et ... que le génie d'Alan soit avec vous ...

Chaleureusement : Nulentout.

