Interface homme/machine universelle pour transceiver

Dominique Boell, F4FEI

1ère partie - Plate-forme de développement. Les fonctions de base : affichage, boutons poussoirs.

2ème partie - Les fonctions avancées : encodage optique, synthèse en fréquence (DDS), sauvegarde des données.

3ème partie - Gestion des menus - Les fonctions annexes : scanneur, horloge, thermomètre...

INTRODUCTION

De nombreux OM hésitent à se lancer dans la conception d'équipements à base de microcontrôleurs, car cela demande de concevoir également les logiciels correspondants et donc d'apprendre à utiliser un langage de programmation, ce qui n'est pas forcément évident la première fois.

Le but de cette série d'articles est d'aider pas à pas ces OM, à l'aide d'un exemple concret, à concevoir ou moderniser un transceiver « analogique », en lui ajoutant une interface gérée par microcontrôleur, munie d'un écran LCD pour afficher les différentes informations nécessaires à son pilotage (réglage en fréquence, S-mètre, ROS-mètre, type de modulation, etc.), et bien sûr des boutons de commande.

On trouve de fort belles réalisations, sous forme de kits ou décrites sur les pages web d'OM talentueux. Lorsque la partie interface est gérée par un microcontrôleur, le code source est souvent mis à disposition, mais sa réutilisation en vue de l'adapter à ses besoins n'est pas forcément possible ou nécessite des connaissances approfondies en langage de développement logiciel (assembleur, C, C++ et autres...).

L'originalité de l'interface présentée ci-dessous tient dans l'utilisation d'un microprocesseur très populaire, simple à programmer, ne demandant pas des compétences particulières et de surcroît peu onéreux. L'environnement de programmation est "Open Source" et les exemples de programmes sont facilement adaptables en fonction de la réalisation de chacun, le but étant de donner envie de réaliser sa propre interface et non pas de simplement copier une réalisation, ce qui reste bien entendu possible.

Enfin, cette description permet d'aboutir, de façon progressive, à une réalisation très flexible, ouverte et malgré tout performante, pour un moindre investissement (financier, temps passé, connaissances...).

La première partie est consacrée à l'utilisation de l'environnement de développement et à la conception des premiers programmes correspondant à des fonctions de base comme l'affichage sur un écran LCD, le S-mètre, le ROS-mètre et les commandes par boutons poussoirs.

La deuxième partie traitera des fonctions avancées telles que l'encodage optique, la synthèse directe en fréquence (DDS) et la gestion de périphériques I2C, comme par exemple la sauvegarde des données.

Enfin, la troisième et dernière partie sera consacrée à la gestion des menus ainsi qu'à la réalisation de fonctions annexes qui peuvent être bien utiles, comme par exemple une horloge UTC, la mesure de la température du PA et autres fonctions.

1. L'INTERFACE HOMME-MACHINE

L'interface homme-machine est réalisée grâce à l'emploi d'un microcontrôleur. Celui-ci va traduire vers la machine les gestes de l'opérateur et lui renvoyer des informations à travers un afficheur, d'où le nom d'interface homme-machine (*figure 1-1*).

Qui dit microcontrôleur dit développement logiciel à l'aide d'une « plate-forme logicielle » comme on dit dans les laboratoires de développement.

Pour faire très simple (les « *softeux* »

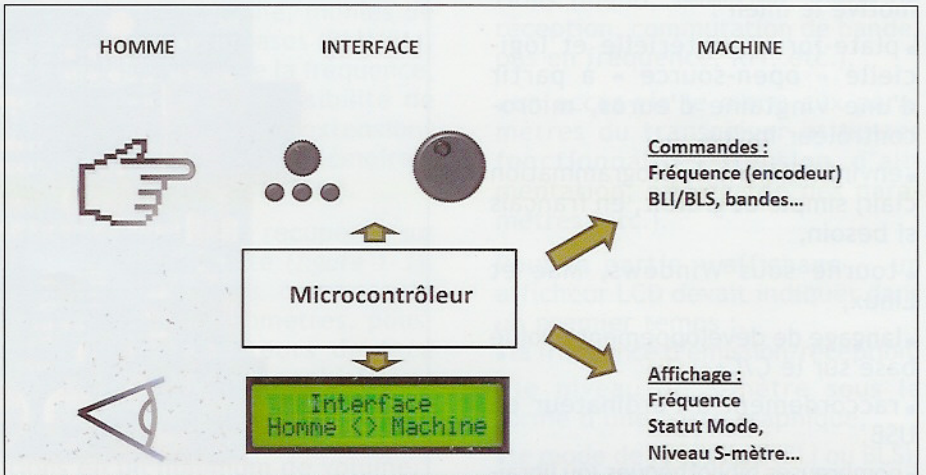

Figure 1-1

me pardonneront je l'espère...), une plate-forme logicielle est constituée de matériel (le microcontrôleur avec ce qu'il faut autour pour qu'il fonctionne et soit programmable), d'un environnement de développement intégré (EDI) avec éditeur de programme, compilateur et débogueur, d'un langage de programmation, et divers outils facilitant, par exemple, la programmation, le développement d'interface graphique, les tests, etc.

Motiver un choix de microcontrôleur plutôt qu'un autre est difficile et peut être source de polémiques car il existe bon nombre de possibilités, chacune ayant ses avantages et ses inconvénients.

Le choix du microcontrôleur s'est porté sur l'ATEMEGA 328P d'Atmel, avec une plate-forme de développement appelée ARDUINO UNO, ce qui permet de remplir totalement les exigences du cahier des charges de cette réalisation :

- accessible au débutant n'ayant pas ou peu de connaissances en développement logiciel,
- capable de traiter des signaux numériques et analogiques (par ex. la tension du S-mètre...),
- ouverte, pour permettre des améliorations et susciter la conception de nouveaux projets à partager entre OM,
- d'un coût abordable (quelques dizaines d'euros), y compris la plate-forme de développement,
- simple à mettre en œuvre (peu de composants...).

D'autres choix auraient bien sûr été possibles, mais voici tout de même quelques raisons qui ont motivé le mien :

- plate-forme matérielle et logicielle « open-source » à partir d'une vingtaine d'euros, microcontrôleur inclus,
- environnement de programmation clair, simple et gratuit, en français si besoin,
- tourne sous Windows, Mac et Linux,
- langage de développement évolué basé sur le C/C++,
- raccordement à l'ordinateur en USB,
- nombreuses bibliothèques (ou librairies) disponibles, simplifiant considérablement la programmation,
- de nombreux livres, documents, tutoriaux et pages web, en français si besoin.

Pour le microcontrôleur :

- 14 entrées/sorties digitales, dont certaines utilisables spécifiquement (Tx/Rx asynchrone, modulation de largeur d'impulsions, bus I2C, bus SPI, interruptions...),
- 6 entrées/sorties analogiques avec convertisseur de 10 bits,
- chargeur d'amorçage intégré dit « bootloader »,
- 32 kilo-octets de mémoire de programme (Flash),
- 2 kilo-octets de mémoire vive (RAM),
- 1 kilo-octet de mémoire programmable et effaçable électriquement (EEPROM),
- Microcontrôleur seul à partir de 7€.

J'encourage le lecteur intéressé, à consulter les sites web disponibles pour découvrir ce qu'est ARDUINO. Il en existe de nombreux, dont un officiel en anglais, mais aussi en français, le but ici, n'étant pas de faire une description exhaustive d'ARDUINO, mais de montrer comment l'utiliser pour l'application envisagée.

Remarque importante : il existe plusieurs types de module ARDUINO : ARDUINO UNO, ARDUINO UNO CMS, ARDUINO MINI, ARD UINO NANO, ARDUINO MEGA et bien d'autres, ainsi que des modules d'extension appelés « shield » qui permettent d'envisager toutes sortes d'applications (commande moteur, afficheur graphique, liaison Ethernet, WiFi, X-Bee, etc.).

Le choix de l'ARDUINO UNO (non CMS) est important ici, car le microprocesseur est monté sur support, contrairement aux autres modules ARDUINO. Cela signifie qu'une fois le programme développé par l'utilisateur et testé « sous toutes les coutures », on peut retirer le microprocesseur.

En effet, celui-ci contient alors le programme de l'application développée et est donc prêt à fonctionner dans le montage envisagé (par exemple l'interface que vous allez développer) avec un minimum de composants autour (1 quartz, 2 condensateurs, 1 résistance).

L'environnement de développement ne sert plus, mais sera bien entendu réutilisé pour concevoir d'autres applications ou modifier celle-ci, à condition de lui remettre un microcontrôleur ATmega 328P.

2. COMMENT DÉMARRER ?

2.1. Le matériel

Voici le minimum requis pour se lancer dans la conception de cette interface et réaliser les fonctionnalités de base (affichage, commande par boutons poussoirs, S-mètre et ROS-mètre) :

- un module ARDUINO UNO (*figure 1-2*),
- le logiciel de développement,
- un afficheur de 2 ou 4 lignes de 16 ou 20 caractères alphanumériques,
- 7 boutons poussoirs,
- une platine d'essais sans soudure avec quelques fils de câblage,
- un mini potentiomètre de 10 kΩ,
- un potentiomètre (entre 4,7 kΩ et 100 kΩ),

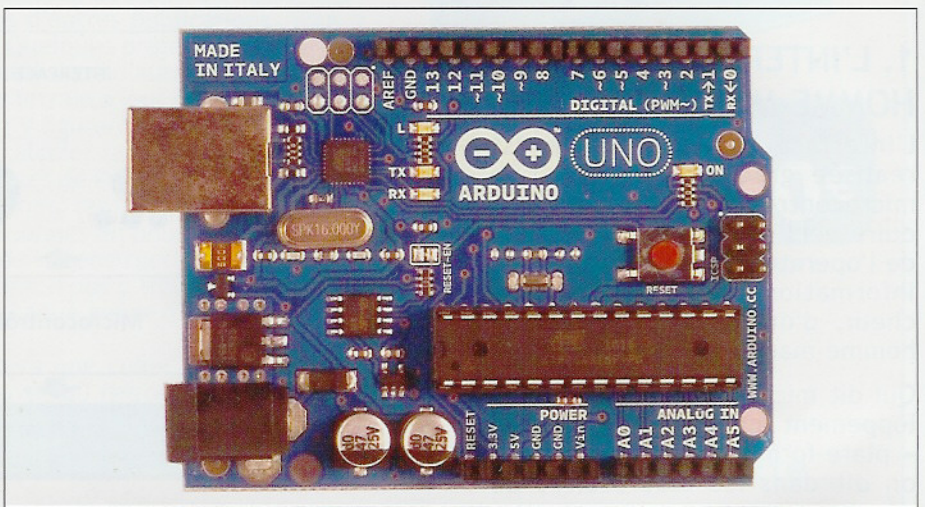

Figure 1-2

- quelques résistances (100 Ω, 8 x 1 kΩ, 4,7 kΩ, 1 MΩ),
- un condensateur de 100 pF.

Pour réaliser les fonctions dites « avancées » de cette interface homme-machine, comme la synthèse directe en fréquence, la sauvegarde des données, il sera nécessaire de dispose r d'un encodeur optique, d'un module DDS à base du circuit AD9851, d'une EEPROM I2C et de quelques autres composants. Nous verrons cela dans les articles suivants.

2.2.Le logiciel

Pour utiliser le module ARDUINO, il faut télécharger et installer le logiciel de développement mis gratuitement à la disposition des utilisateurs sur l'un des sites de référence, par exemple le site en français http://arduino.cc/fr/Main/HomePage.

Si vous découvrez ARDUINO pour la première fois à travers cet article, il est indispensable de consulter ce site. Vous y trouverez les informations essentielles pour vous familiariser avec le matériel, installer et utiliser le logiciel de développement (en images), découvrir le langage de programmation ainsi que les principales bibliothèques disponibles, et même, de quoi vous faire la main en écrivant vos premiers programmes.

Il est également important de bien comprendre la syntaxe du langage de programmation, la moindre erreur provoquant une erreur lors de la compilation, ou, pire, des comportements bizarres dans les programmes (« bug »). Les exercices sur le site d'ARDUINO permettent de s'entraîner progressivement.

Une fois cette étape préliminaire franchie, vous allez pouvoir concevoir votre interface homme-machine, et bien plus encore. Attention toutefois, on devient vite « addict » et vos réalisations à base de ce microcontrôleur risquent de se multiplier ;-).

<u>A propos des programmes de la première partie</u> :

Ils sont conçus, fonction par fonction (afficheur, S-mètre, puis boutons poussoirs), pour permettre à l'OM débutant en programmation, de bien comprendre leur fonctionnement et de se familiariser avec le langage de programmation. Ils ne sont pas non plus optimisés, d'une part parce que je suis, moi aussi, débutant en programmation, et d'autre part parce que cela permet de gagner en clarté, au détriment de quelques lignes de code. Avec un peu d'expérience, il sera possible d'en réduire la taille, de trouver de meilleures façons de programmer et aussi de libérer de la place en mémoire.

Les programmes ne sont pas non plus listés de façon exhaustive, cela pour éviter de longues pages de codes dans la revue. Ceux-ci étant largement commentés dans les fichiers mis à disposition, seules les parties demandant plus d'explications sont imprimées.

Enfin, à la fin de chaque article, tous les sous-programmes sont intégrés en un seul programme, de façon à faire fonctionner l'interface selon les spécifications prévues dans chaque partie.

A la fin de l'article, vous trouverez la liste des programmes disponibles sous forme de fichiers au format ARDUINO (.ino), qui seront mis sur le site de REF-Union.

3. CONCEPTION DE L'INTERFACE HOMME-MACHINE

La réalisation finale dépendra de votre projet, tant sur le plan mécanique, qu'électrique ou logiciel.

Pour illustrer les propos ci-dessus, voici à titre d'exemple comment j'ai démarré mon projet.

Il s'agissait de concevoir, puis réaliser, un transceiver BLU QRP bi-bandes 40 m et 10 m, utilisable en portable et mobile, munies de fonctionnalités de bases (BLU/BLI, S-mètre, affichage de la fréquence, etc.), mais avec la possibilité de lui ajouter de futures extensions (autres bandes, Notch, mémoires, divers filtres FI et BF, etc.).

La mécanique a été récupérée sur un autoradio obsolète (*figure 1-3*), ainsi que certains composants comme les potentiomètres, poussoirs et commutateurs de face avant, souvent très sophistiqués sur les autoradios, puisque devant commander un maximum de fonctions en un minimum de volume.

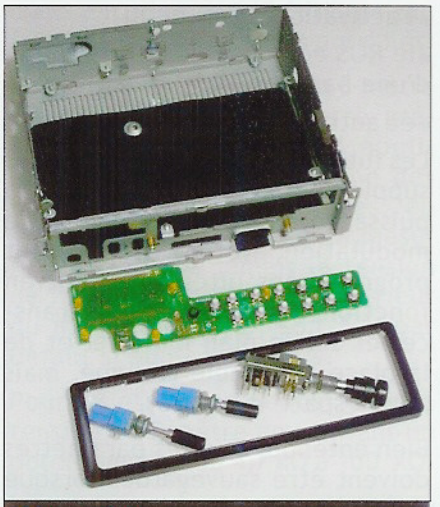

Figure 1-3 :

Mais seule la conception de l'interface de commande du transceiver fait l'objet d'une description dans cette série d'articles (donc pas la partie électronique du transceiver).

La maquette fonctionnelle de l'interface est réalisée sur une platine d'essais sans soudure. Par la suite, chacun pourra réaliser, en fonction de ses besoins, un circuit imprimé correspondant à son projet.

Pour la partie « commandes » les spécifications minimales étaient les suivantes :

- un bouton marche/arrêt, éventuellement couplé au volume,
- un réglage du volume,
- un réglage du gain HF,
- un bouton rotatif de commande de fréquence pour piloter un synthétiseur de fréquence de type DDS,
- un réglage de la bande passante audio,
- quelques boutons poussoirs pour activer rapidement certaines fonctions (menu, sélection, mode de réception, commutation de bande, pas en fréquence, RIT, etc.),
- un accès via le menu aux paramètres du transceiver et autres fonctionnalités (tension d'alimentation, sauvegarde des paramètres, etc.).

Pour la partie « affichage », un afficheur LCD devait indiquer dans un premier temps :

- la fréquence d'émission/réception,
- le niveau du S-mètre sous la forme d'une barre graphique,
- le mode de réception (BLI ou BLS),
- le pas de synthèse en fréquence (10 Hz, 100 Hz, 1 kHz ou 10 kHz),

- l'activation du mode RIT,
- le ROS en émission sous la forme d'une barre graphique,
- et autres paramètres retenus.

Les futures fonctionnalités (bandes supplémentaires, affichage de la puissance émission, du niveau de modulation, Notch, etc.) seront programmées ultérieurement en fonction de l'avancée du transceiver sur le plan matériel et de la place disponible (volume, mais aussi espace mémoire).

Bien entendu, tous les paramètres doivent être sauvegardés lorsque l'appareil est mis hors tension.

La face avant devrait avoir l'aspect suivant (*figure 1-4*) :

Figure 1-4 :

Mais, pas de précipitation ! Avant d'en arriver à une face avant qui fonctionne correctement, revenons à la conception du programme de l'interface homme-machine et à la réalisation d'une maquette fonctionnelle.
Note : appellation FdT 40/10 pour Fonds de Tiroirs, et 40/10 pour les bandes choisies, car autant que possible, utilisation des composants récupérés.

3.1. PREMIERS PAS

Une fois que vous serez familiarisés avec les petits programmes d'exemples décrits sur le site d'ARDUINO, vous serez à même de réaliser le programme de votre propre interface. Le mieux est de démarrer avec l'afficheur.

En effet, une fois compris la façon de l'utiliser, c'est la meilleure façon de tester vos premiers programmes.

- Commençons par le matériel :

Tout d'abord il faut raccorder l'afficheur au module ARDUINO.

Pour ma part, j'ai choisi un afficheur de 2 lignes de 16 caractères très courant, avec un contrôleur graphique compatible HD44780, mais rien n'empêche d'utiliser un afficheur de 4 lignes de 20 caractères par exemple.

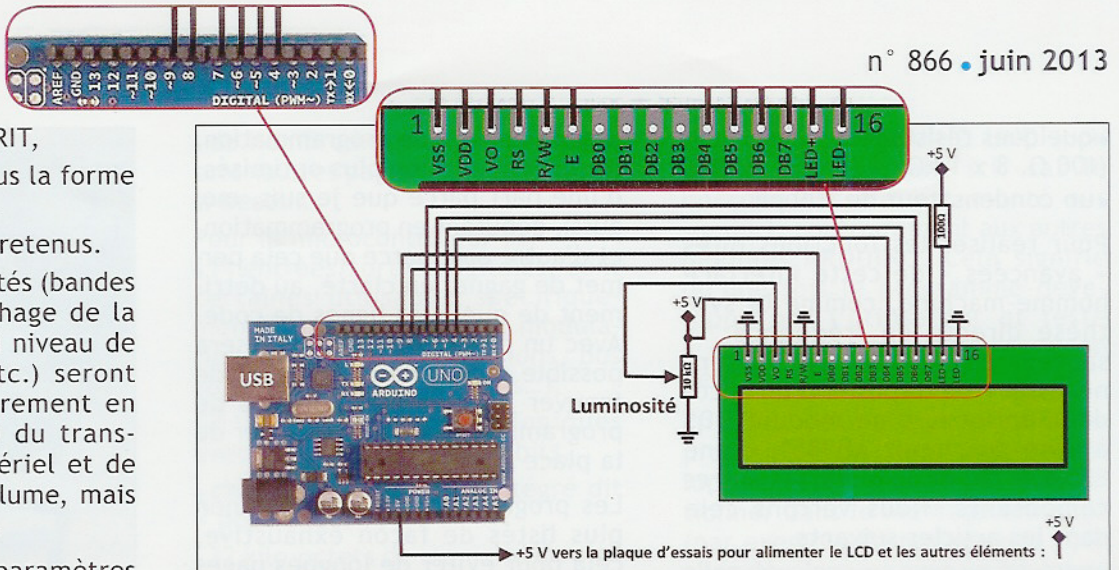

Figure 1.5 :

Deux modes de connexions sont possibles : mode 4 bits ou mode 8 bits. Pour économiser le nombre d'entrées/sorties utilisées sur le microcontrôleur, le mode 4 bits est choisi car il n'utilise que 6 sorties digitales sur les 14 disponibles au lieu de 11 dans le mode 8 bits. Nous verrons plus loin à quoi serviront les autres.

Il faut bien entendu se référer à la notice de l'afficheur pour le connecter au microcontrôleur. Les afficheurs courants comportent 16 bornes dont 6 vont donc être raccordées au microcontrôleur (*figure 1-5*).

Les bornes 15 et 16 de l'afficheur peuvent être utilisées pour rétro éclairer l'afficheur (voir sa notice).

Pour tester l'afficheur, le mieux est de le câbler sur la platine d'essais sans soudure et de le raccorder au module ARDUINO (*figure 1-6*).

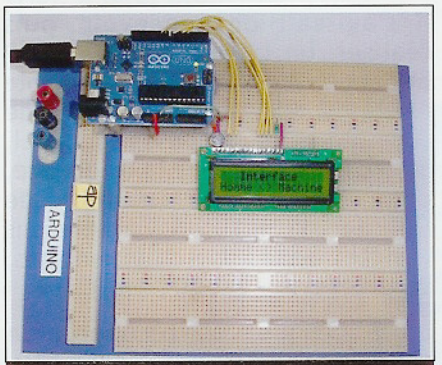

Figure 1-6 :

L'afficheur est alimenté par le +5 V sur le connecteur « Power » du module ARDUINO, lui-même alimenté par le PC via sa prise USB.

Une fois l'afficheur relié au module ARDUINO par des fils de câblage, il faut régler le potentiomètre de luminosité à la limite de l'apparition de la matrice de point des caractères de l'afficheur.

- Premier programme : l'affichage

Comme indiqué plus haut, le choix de l'ARDUINO a été largement influencé par la disponibilité de bibliothèques logicielles facilitant grandement la programmation.

Pour un débutant, la gestion d'un afficheur graphique en assembleur ou même en C/C++ n'est pas évidente. Grâce à une bibliothèque dédiée aux afficheurs LCD, utiliser un afficheur devient très simple.

En effet, afficher « Interface Homme <> Machine » sur l'afficheur LCD revient à :
- déclarer une fois pour toutes en début de programme l'utilisation de la bibliothèque d'affichage LCD,
- définir le mode de connexion de l'afficheur (4 bits ou 8 bits),
- définir le nombre de lignes et de caractères, par exemple 2 lignes de 16 caractères,
- positionner le curseur là où les caractères doivent s'afficher,
- « imprimer » le mot « Interface » au centre de la première ligne de l'afficheur,
- et « Homme <> Machine » sur la seconde ligne.

En langage ARDUINO, cela donne (*figure 1-7*) :
Si vous avez câblé correctement l'afficheur à l'ARDUINO, une fois le code saisi et transféré ou plutôt « téléversé » dans le microcontrôleur comme on dit en langage ARDUINO, « Interface Homme <> Machine » doit s'afficher sur deux lignes.

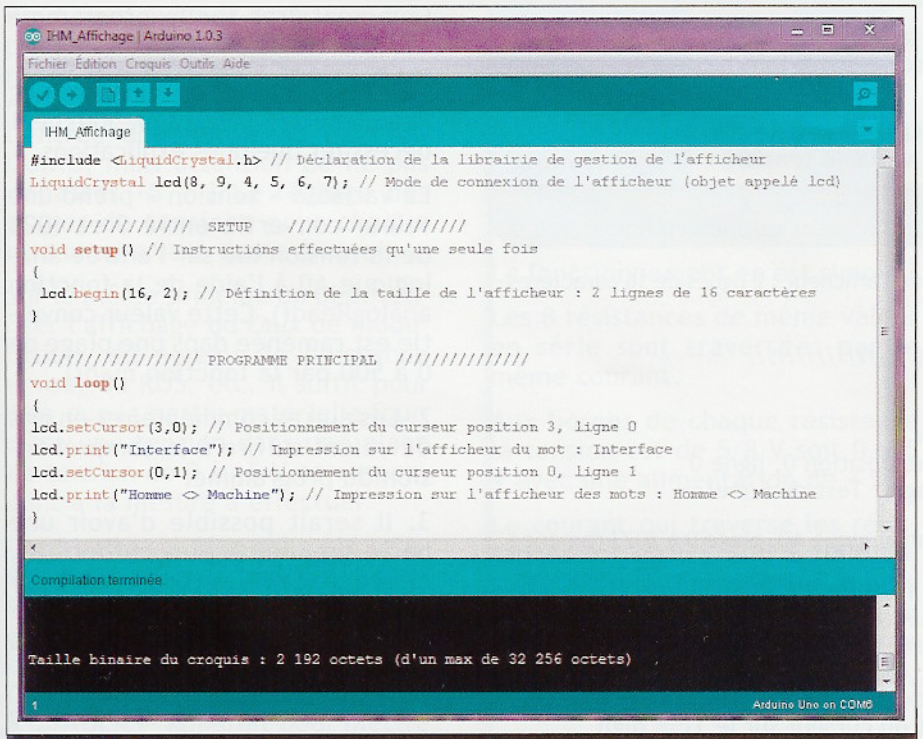

Figure 1-7 :

Point S	S1	S2	S3	S4	S5	S6	S7	S8	S9	S9+10	S9+20	MAX
Tension en V	0,07	0,47	0,76	1,10	1,46	1,81	2,16	2,49	2,90	3,50	4,00	5,00

Tableau 1

Commentaires sur le programme IHM_Partie_1_Affichage :

1. La bibliothèque LiquidCrystal.h est incluse dans le logiciel de développement
(Menu : « Croquis »,
« Importer bibliothèques... »,
« LiquidCrystal »).
Vous trouverez sur les sites ARDUINO une explication complète des fonctions disponibles, comme clear() qui efface l'afficheur ou setCursor(x, y) qui positionne le curseur colonne x, ligne y. Il y a bien sûr d'autres fonctions à découvrir pour gérer l'afficheur.

2. Lors de la description du mode de connexion de l'afficheur avec la déclaration LiquidCrystal lcd(), les chiffres entre les parenthèses correspondent au câblage et au mode choisis pour l'afficheur, à savoir dans l'exemple ci-dessus :

- broche 8 de l'ARDUINO reliée à RS de l'afficheur,
- broche 9 de l'ARDUINO reliée à E de l'afficheur,
- broches 4, 5, 6, et 7 de l'ARDUINO reliées respectivement à D4, D5, D6 et D7 de l'afficheur.

Remarque : rien n'empêche de relier différemment l'afficheur à l'ARDUINO, il faudra juste le déclarer correctement dans la fonction LiquidCrystal ().

Maintenant que nous pouvons afficher un texte, nous allons étudier les différentes fonctions de l'interface universelle, à savoir :

- le S-mètre,
- le ROS-mètre,
- la détection des boutons poussoirs,
- les fonctions activées par les boutons poussoirs.

3.2. S-MÈTRE

Il s'agit de convertir une tension issue du récepteur, en général de la commande automatique de gain de l'amplificateur FI, et de l'afficher sous forme d'une barre graphique qui évoluera de S1 à S9, et indiquera les dépassements de niveaux en-dessous de S1 et au-dessus de S9.

Comme indiqué précédemment, le module ARDUINO possède 6 entrées/sorties analogiques, chacune encodée sur 10 bits, ce qui fait 2^{10}-1 ou 1023 valeurs possibles.

Pour afficher la tension du S-mètre sous la forme d'une barre graphique, il faut d'abord appliquer cette tension sur l'une des entrées/sorties analogiques, en s'assurant que la tension à mesurer ne dépasse jamais la tension de référence définie à l'intérieur du microcontrôleur, soit 5 V par défaut si alimenté en 5 V.

Pour les essais, on peut simuler la tension du S-mètre en utilisant un potentiomètre de 4,7 kΩ à 100 kΩ.

Les extrémités du potentiomètre sont reliées entre le + 5 V et la masse, le curseur sur le port analogique A0 du microcontrôleur. De cette façon, la tension sur A0 peut varier de 0 à + 5 V.

Dans la réalité, le niveau du signal reçu suit une courbe non linéaire et ne varie pas forcément de 0 à 5 V. Il faudra donc ajuster l'électronique du circuit S-mètre pour obtenir cette plage de tension et déterminer à l'aide du programme le seuil d'affichage de chaque point S, ce qui, dans mon cas, est représenté dans le tableau ci-contre (*tableau 1*) :

Le programme d'affichage précédent va être réutilisé en y ajoutant des lignes permettant :

- de lire la tension du potentiomètre sur le port analogique A0,
- de définir 12 plages de tensions correspondant aux niveaux à afficher de <S1 à >>S9+20,
- d'afficher chaque niveau de S1 à S9 avec le caractère ▮,
- et de rappeler à la fin de la ligne le niveau en clair <S1, S1, S2, ..., jusqu'à >S9+10, >>S9+20.

Rien n'empêche d'ajouter un ou deux niveaux au-dessus si besoin, il faudra juste modifier le programme en conséquence.

En langage ARDUINO, cela donne (extrait du programme IHM_Partie_1_Smetre) :

```
#include <LiquidCrystal.h> // Déclaration de la librairie de gestion de l'afficheur
LiquidCrystal lcd (8, 9, 4, 5, 6, 7) ; // Mode de connexion de l'afficheur (objet appelé lcd)

// Déclaration des variables
#define Smetre 0 // Définition de la constante Smetre (= 0)
int tension = 0 ; // Initialisation à 0 de la variable tension

////////////////// SETUP ////////////////////
void setup() // Instructions qui ne seront effectuées qu'une seule fois
{
 lcd.begin (16, 2) ; // Définition de la taille de l'afficheur : 2 lignes de 16 caractères
}

////////////////// PROGRAMME PRINCIPAL ///////////////
void loop ()
{
 lcd.clear() ; // Effacement de l'afficheur
 lcd.setCursor(0,0) ; // Curseur positionné position 0, ligne 0
 lcd.print ("Test du S-metre") ; // Affichage : Test du S-metre

 // La variable tension est lue sur l'entrée analogique A0, plage 0 à 1023 ramenée
 de 0 à 500
 tension = map (analogRead (Smetre), 0, 1023, 0, 500);
 lcd.setCursor(0,1) ; // Curseur positionné position 0, ligne 1
 if (tension < 7) // En dessous du seuil de 7 ...
  {
   lcd.print("<") ; // ... affichage de < en début de ligne
   lcd.setCursor(10,1) ; // Positionnement du curseur sur le 11ème digit
   lcd.print("<S1") ; // Affichage de <S1 après la barre graphique
  }
 if (tension > 6 && tension < 47) // Affichage du niveau S1 pour des valeurs
 d'entrée entre 6 et 47
  {
   lcd.print(char (255)) ; // Le caractère ASCII 255 correspond à ▮
   lcd.setCursor(11,1) ; // Le niveau en clair est affiché à partir de la position 12
   lcd.print("S1") ;
  }
 if (tension > 46 && tension < 76)
  {
   lcd.print(char (255)) ;
   lcd.print(char (255)) ; // Deux ▮ sont donc affichés l'un après l'autre
   lcd.setCursor(11,1) ;
   lcd.print("S2") ;
  }
 if (tension >75 && tension < 110)
  {
   lcd.print(char (255)) ;
   lcd.print(char (255)) ;
   lcd.print(char (255)) ; // Trois ▮ et ainsi de suite...
   lcd.setCursor(11,1) ;
   lcd.print("S3") ;
  }
 if (tension > 109 && tension < 146)
  {
   ... // même principe de  S4 à S9 + 20
  }
 if (tension > 399)
  {
   lcd.print(char (255));
   lcd.print(char (255));
   lcd.print(char (255));
   lcd.print(char (255));
   lcd.print(char (255));
   lcd.print(char (255));
   lcd.print(char (255));
   lcd.print(char (255));
   lcd.print(char (255)); // Neuf ▮
   lcd.print(">>");
   lcd.setCursor(11,1);
   lcd.print("S9+20");
  }
   delay(200) ; // Retard provisoire de 200 ms pour un meilleur affichage
}
```

Commentaires sur le programme IHM_Partie_1_Smetre :

1.La fonction tension = map(analog Read(Smetre), 0, 1023, 0, 500) peut nécessiter quelques explications.

La variable « tension » prend une valeur convertie entre 0 et 1023 de la tension lue sur l'entrée analogique A0 à l'aide de la fonction analogRead(). Cette valeur convertie est ramenée dans une plage de 0 à 500 par la fonction map().

2. Les autres commentaires en gris devraient suffire à la compréhension du programme.

3. Il serait possible d'avoir une barre graphique plus esthétique, en remplaçant les caractères ▮ par des caractères dont la taille grandit progressivement : _ _ _ ▪ ▪ ▪ ▮ ▮

Pour cela, il faut trouver les numéros des caractères graphiques de l'afficheur utilisé qui pourraient remplacer le caractère ▮ = char (255). Se référer à la notice de l'afficheur ou faire des essais successifs en changeant le numéro du caractère pour savoir si de tels caractères sont disponibles.

4. Une fois le programme transféré dans le microcontrôleur, le fait de tourner le potentiomètre fait varier l'affichage de la barre graphique de <S1 à >>S9+20 comme le montre les *figures 1-8* et *1-9*.

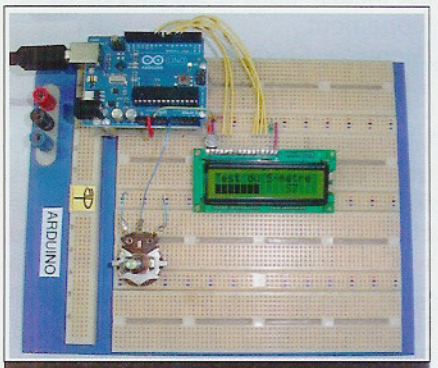

Figure 1-8 :

Figure 1-9 :

Pour voir évoluer la barre graphique en temps réel, on peut relier la sortie du S-mètre d'un récepteur ou d'un transceiver à l'entrée analogique A0 de l'ARDUINO, mais attention de ne pas dépasser +5 V ! Le réglage des plages de tensions pour chaque niveau sera sans doute à revoir.

Sur le même principe on peut réaliser l'affichage du taux de modulation, du niveau de puissance émise, du ROS, etc. Il suffit pour cela que la partie matérielle fournisse une tension dans une plage de tension de 0 à +5 V correspondant à la mesure à effectuer.

C'est cette solution qui a été retenue pour afficher le ROS. Le programme est quasiment identique à celui du S-mètre, au nombre de seuils et à leur valeur près. Lorsque le ROS dépasse 3, le mot « DANGER » apparaît pour mettre en garde l'opérateur (risque pour le PA). Voir les détails dans le programme complet de la partie 1 : IHM_Partie_1.ino.

3.3. BOUTONS POUSSOIRS

Pour savoir sur quel bouton l'opérateur appuie, il y a plusieurs possibilités :

- chaque bouton est relié à une entrée digitale du microcontrôleur, 6 boutons nécessitent donc 6 ports disponibles,
- les boutons sont câblés de façon matricielle. Dans ce cas, pour 6 boutons, il faut encore 5 ports, ou au moins 3 si un encodage préalable est fait,
- les boutons sont câblés en diviseur résistif sur une entrée analogique. Ici, un seul port suffit comme le montre le schéma de la *figure 1-10*.

Bouton poussoir	0	1	2	3	4	5	6
Tension (V)	4,37	3,75	3,12	2,50	1,87	1,25	0,62
Conversion	895	767	639	512	384	256	128

Tableau 2

Le fonctionnement en est simple :

Les 8 résistances de même valeur en série sont traversées par le même courant.

Aux bornes de chaque résistance la tension est de 5/8 V soit 0,625 V avec une alimentation de + 5 V.

Le courant qui traverse les résistances est égal à 5/(8 x 1000 Ω), soit 0,625 mA. Lorsque l'on appuie sur le bouton poussoir de rang n, la tension appliquée sur l'entrée analogique A1 est de (7-n) x 0,625, exprimée en V.

Ceci n'est vrai qu'en supposant que le courant drainé par la résistance de rappel de 1 MΩ est négligeable devant le courant traversant les résistances, ce qui est le cas (moins de 5 µA).

Exemple :
L'appui sur le bouton poussoir 5 provoque donc une tension de (7-5) x 0,625 = 1,25 V.

Il ne reste plus qu'à mesurer la tension sur l'entrée analogique pour déterminer quel bouton poussoir a été activé et le mémoriser.

Les 7 valeurs de tension doivent être converties en tenant compte des 1023 valeurs possibles sur l'entrée analogique.

L'alimentation +5 V est convertie à la valeur 1023, donc en reprenant l'exemple du bouton poussoir 5, la tension de 1,25 V devient :
1023/5 x (7-5) x 0,625 = 256.

Pour les 7 boutons poussoirs, cela donne : (*voir tableau 2 ci-dessus*)

Commentaires sur le programme : (*voir tableau page ci-contre*)

1. Pour les besoins de l'exercice, l'afficheur indique sur la première ligne la valeur convertie entre 0 et 1023 de la tension lue et sur la seconde ligne, le numéro du bouton poussoir.

2. Lors de l'intégration des différents sous-programmes de l'interface universelle, cet affichage disparaîtra, mais le numéro du bouton poussoir sera mémorisé pour activer la fonction correspondante.

3. L'instruction if (tensionClav > 246 && tensionClav < 266) permet de prendre une marge de + ou - 10 autour de la valeur théorique (ici 256 pour le bouton poussoir 5). Cette marge permet d'éviter le risque d'erreur en cas d'appui simultané sur 2 boutons adjacents.

4. La bouton poussoir 0 n'est, pour le moment, pas câblé sur la maquette et n'est donc pas détecté dans le programme.

5. Il n'est pas prévu de traitement anti-rebond dans le programme, le condensateur de 100 pF faisant office d'anti-rebond.

Une fois le câblage terminé et vérifié, le programme est testé en essayant les boutons poussoirs (*figure 1-11*).

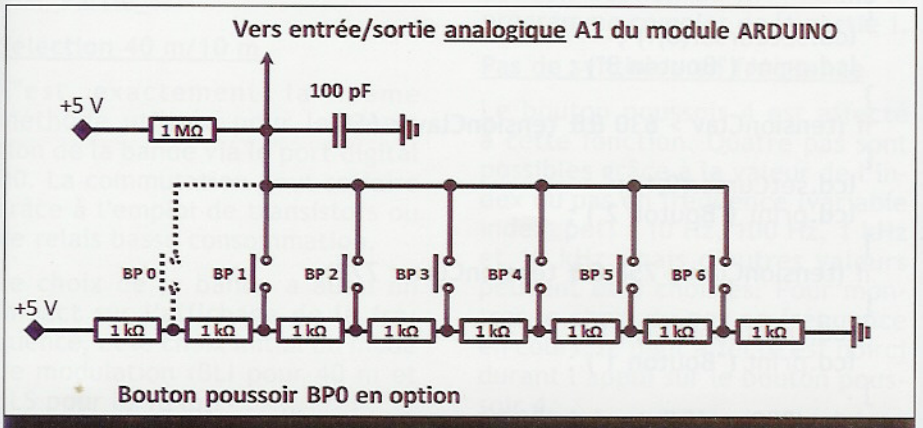

Figure 1-10 :

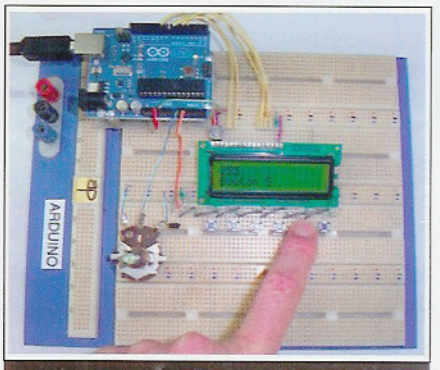

Figure 1-11 :

3.4. UTILISATION DES BOUTONS POUSSOIRS

Maintenant que l'on sait détecter l'appui sur un bouton poussoir, il faut que la fonction qu'il commande soit activée ou désactivée et l'affichage mis à jour.

Parmi ces fonctions, il y a :
- la sélection BLI, BLS,
- la sélection de la bande (40 m ou 10 m),
- l'entrée ou la sortie du mode RIT,
- la sélection du pas de synthèse en fréquence (Pas en fréq.),
- la confirmation de la sélection de la fonction (Sélect.),
- l'entrée dans le mode MENU.

L'utilisation des fonctions « Sélect. » et « Menu » sera vue dans la troisième partie de cette série d' articles.

Les boutons poussoirs 5 et 6 sont affectés provisoirement à la décrémentation (↓) et à l'incrémentation (↑) de la fréquence.

Les lignes de code nécessaires à la gestion des fonctions commandées par les boutons poussoirs et les explications correspondantes sont dans le programme IHM_Partie_1.

Y sont intégrés également les programmes précédents (affichage, S-mètre, ROS-mètre et boutons poussoirs).

Les boutons poussoirs BLI/BLS, Bande, RIT et Pas en fréquence ont une gestion de type « bascule » qui prend en compte leur état précédent (variable avant). Un appui, même long, envoie une seule information de changement d'état. Cela est réalisé grâce au code suivant :

```
if ((bouton == true) && (avant == false))
 {
   avant = bouton;
   instructions_à_exécuter ;
 }
if (bouton == false)
  {
    avant = false;
  }
```

Explications :
bouton == true : le bouton poussoir est enfoncé,

avant == false : l'état précédent du bouton était relâché,

Si la condition du premier if est vraie, les instructions qui suivent sont exécutées,

Programme IHM_Partie_1_Clavier_analogique en langage ARDUINO :

```
#include <LiquidCrystal.h>
LiquidCrystal lcd (8, 9, 4, 5, 6, 7) ;

// Déclaration des variables du clavier analogique
#define ClavierAnalog 1 // Définition de la constante ClavierAnalog égale à 1
int tensionClav = 0 ; // tensionClav est la variable qui sera lue sur l'entrée analogique 1
boolean bouton_1 = false ; // Bouton relâché
boolean bouton_2 = false ;
boolean bouton_3 = false ;
boolean bouton_4 = false ;
boolean bouton_5 = false ;
boolean bouton_6 = false ;

void setup()
{
  lcd.begin(16, 2);
}

// Programme principal
void loop()

{
  lcd.clear() ;
  lcd.setCursor(0,0) ;
 // La variable tensionClav prend la valeur lue sur l'entrée analogique A1
  tensionClav = analogRead(ClavierAnalog) ;
  lcd.print(tensionClav) ; // Affichage de la valeur de la tension du clavier convertie de 0 à 1023
  lcd.setCursor(0,1) ; // Curseur positionné position 0, ligne 1

   if (tensionClav > 117 && tensionClav < 137)  // Si tensionClav comprise entre 117 et 137...
   {
    lcd.setCursor(0,1) ;
    lcd.print ("Bouton 6") ; // ... affichage du bouton sur lequel on appuie
   }
   if (tensionClav > 246 && tensionClav < 266)
   {
     lcd.setCursor(0,1) ;
     lcd.print ("Bouton 5") ;
   }
   if (tensionClav > 374 && tensionClav < 394)
   {
     lcd.setCursor(0,1) ;
     lcd.print ("Bouton 4") ;
   }
   if (tensionClav > 502 && tensionClav < 522)
   {
     lcd.setCursor(0,1) ;
     lcd.print ("Bouton 3") ;
   }
   if (tensionClav > 630 && tensionClav < 650)
   {
     lcd.setCursor(0,1) ;
     lcd.print ("Bouton 2") ;
   }
   if (tensionClav > 758 && tensionClav < 778)
   {
     lcd.setCursor(0,1) ;
     lcd.print ("Bouton 1") ;
   }
   delay(200) ; // Retard de 200 ms pour un meilleur affichage
}
```

avant = bouton : la variable avant prend la valeur de l'état du bouton, c'est-à-dire enfoncé (true); instructions_à_exécuter : par exemple, commuter la bande.

Si l'on continue à appuyer sur le bouton (bouton == true), sans le relâcher, la condition du second if ne sera pas vraie et l'instruction avant = false sera ignorée. Dans ce cas, au prochain passage sur le premier if, la condition de celui-ci sera fausse et les instructions suivantes non exécutées, ce qui est le but recherché.

En revanche, dès que l'on relâche le bouton, la variable avant prend la valeur false (bouton relâché) et le bouton redevient donc actif.

BLI/BLS

Le premier exemple d'utilisation des boutons poussoirs consiste à affecter au numéro 1 la fonction « sélection du mode d'émission/réception », par exemple BLI ou BLS et à l'afficher.

Le port digital D10 est déclaré en sortie par la fonction pinMode (10, Output) et, lorsque le mode BLS est affiché, le port D10 est mis à l'état haut par digitalWrite (10, HIGH) pour actionner la commutation (étage tampon, transistor ou relais basse consommation) de l'oscillateur de battement ou de tout autre dispositif selon l'architecture de retenue.

A noter : chaque port ne doit pas dépasser un courant de 40 mA, sachant que l'ensemble des ports ne doit pas dépasser 200 mA.

Programme : voir les sections «// Déclaration des variables pour sélectionner le mode BLI/BLS» et «// Sélection BLI ou BLS» dans le programme complet de la partie 1 : IHM_Partie_1.

Sélection 40 m/10 m

C'est exactement la même méthode utilisée pour la sélection de la bande via le port digital D0. La commutation peut se faire grâce à l'emploi de transistors ou de relais basse consommation.

Le choix de la bande a aussi un impact sur l'affichage de la fréquence, et le choix initial du mode de modulation (BLI pour 40 m et BLS pour le 10 m).

Programme : voir les sections «// Déclaration des variables pour sélectionner la bande 40 m ou 10 m » et « //Sélection de la bande à afficher » et « // Commutation 40 m/10 m » dans le programme complet de la partie 1 : IHM_Partie_1.

RIT (Receive Incremental Tuning) appelé aussi « clarifieur »

La sélection du mode RIT par le bouton poussoir 3 ne nécessite pas de port d'entrée/sortie. En effet, seule la fréquence de l'oscillateur local est modifiée en réception pour rattraper un éventuel décalage de la fréquence du correspondant. Le pas de commande est fixé à 10 Hz, mais tout autre choix est possible.

La valeur du décalage en fréquence se fait (provisoirement) à l'aide des boutons poussoirs 5 et 6.

Lorsque le RIT est activé, le signe = s'affiche après la fréquence, indiquant qu'il n'y a pas encore de décalage. Le réglage s'effectue « au son ». Dès que le décalage est différent de 0, le signe - ou + s'affiche après la fréquence, indiquant le sens du décalage.

Pour remettre ce décalage à 0, il faut être dans le mode RIT et l'ajuster à 0.

Un changement de bande désactive le RIT et réinitialise le décalage à 0.

Le passage en émission fait afficher la fréquence d'émission et indique par le signe =, + ou - après la fréquence que le RIT est activé.

La fréquence de réception suit toujours la fréquence d'émission, au décalage du RIT près.

Programme : voir les sections « // Gestion des boutons poussoirs Fréquence ↓ et Fréquence ↑ et RIT » et « // Gestion du RIT » dans le programme complet de la partie 1.

Pas de synthèse en fréquence

Le bouton poussoir 4 est affecté à cette fonction. Quatre pas sont possibles grâce à la valeur de l'index du pas en fréquence (variable index_pef) : 10 Hz, 100 Hz, 1 kHz et 10 kHz, mais d'autres valeurs peuvent être choisies. Pour montrer le choix du pas en fréquence en cours, le digit affecté est noirci durant l'appui sur le bouton poussoir 4.

Comme pour modifier la valeur du RIT, les boutons poussoirs 5 et 6 vont provisoirement augmenter ou diminuer la fréquence affichée.

Programme : voir les sections « // Déclaration des variables pour sélectionner fréquence et pas de fréquence » et « //Détermination du pas en fréquence (10, 100, 1000 ou 10000 Hz) » dans le programme complet de la partie 1 : IHM_Pa rtie_1.

Utilisation des boutons poussoirs 5 et 6

En attendant l'utilisation de l'encodeur optique, l'appui sur les boutons 5 ou 6 provoque une diminution ou une augmentation, soit de la fréquence, soit du décalage en fréquence (RIT).

Dans le programme, deux cas se présentent :

- si le RIT est désactivé, c'est la variable freq (fréquence courante) qui est augmentée ou diminuée du pas de fréquence sélectionné,
- si le RIT est activé, c'est la variable Rit (décalage) qui est augmentée ou diminuée du pas prédéfini (10 Hz).

Programme : voir les sections « // Déclaration des variables pour sélectionner fréquence et pas de fréquence » et « // Gestion des boutons poussoirs Fréquence ↓ et Fréquence ↑ » dans le programme complet de la partie 1 : IHM_Partie_1.

Emission / Réception ou «PTT»

Ici, pas de bouton poussoir, c'est l'appui sur la pédale du microphone qui va provoquer le passage en émission ainsi que :

- l'allumage de la diode électroluminescente en face avant,
- et l'affichage du ROS à la place du S-mètre.

Le port digital D1 est configuré en entrée pour que le microcontrôleur détecte le passage en émission et lance les sous-programmes d'affichage correspondants grâce à la fonction RXTX ().

Programme : voir les sections « // RXTX » et « // Gestion du ROS-mètre » dans le programme complet de la partie 1 : IHM_Partie_1.

3.5. MAQUETTE FINALE DE LA PARTIE 1

En ayant câblé et vérifié un par un le fonctionnement de l'afficheur, du potentiomètre et des boutons poussoirs sur la plaque d'essais, la première partie de la maquette doit être maintenant totalement opérationnelle après avoir chargé le programme complet de la partie 1.

Le mieux est de tester toutes les fonctions, les unes après les autres et de vérifier les affichages correspondants (*figures 1-12 et 1-13*).

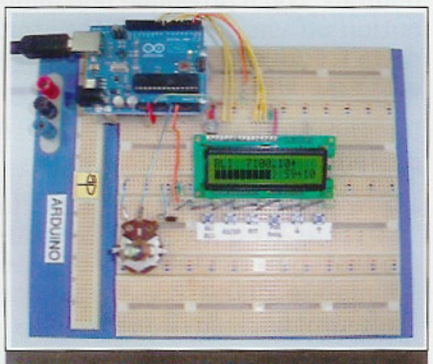

Figure 1-12 : réception BLI 40 m. RIT activé

Figure 1-13 : Emission BLS sur 10 m. RIT désactivé

La fonction émission/réception peut être testée en mettant l'entrée digitale D1 à la masse (mode réception) à travers une résistance de 4,7 kΩ. Le S-mètre doit s'afficher. Faire varier le potentiomètre pour visualiser la plage des valeurs affichables.

En laissant cette entrée en l'air, ou mieux, en la reliant au + 5 V à travers la même résistance de 4,7 kΩ, on passe en mode émission. Le ROS-mètre doit s'afficher. Faire varier le potentiomètre pour visualiser la plage des valeurs affichables.

Le schéma électrique complet de la partie 1 est représenté *figure 1-14*.

ATTENTION : les ports digitaux D0 et D1 servent au microprocesseur à recevoir et émettre des données pendant sa phase de programmation. Il ne faut donc pas relier D0 et D1 directement à la masse ou au + 5 V pendant cette phase.

La sortie D0 (commutation de bande par transistor ou relais basse consommation) peut être testée en mettant une diode électroluminescente (DEL) en série avec une résistance de 1 kΩ à la masse. La DEL doit s'allumer en position 10 m, mais durant la phase de programmation elle clignotera.

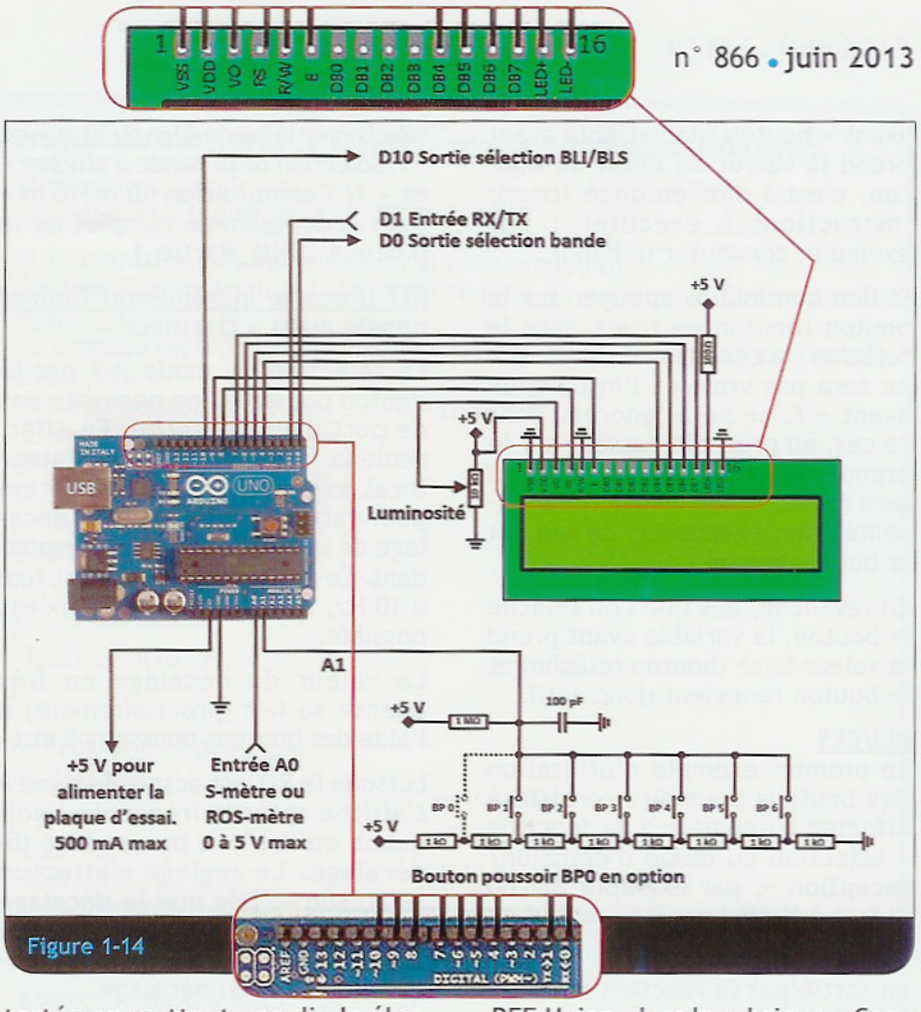

Figure 1-14

Si un transistor ou un relais de commutation sont reliés au port DO, prévoir un cavalier pour déconnecter D0 du transistor ou du relais afin d'éviter qu'ils ne changent d'état au rythme du transfert des données pendant la phase de programmation.

De même, le port D1 reste compatible avec la phase de programmation grâce à la résistance série de 4,7 kΩ mentionnée plus haut.

D'une manière générale, **ne jamais relier un port du microcontrôleur directement au + 5 V ou à la masse, ni court-circuiter deux ou plusieurs ports entre eux**, sous peine de risquer de griller le microcontrôleur.

Vous pouvez consulter le site ci-après pour connaitre les meilleures façons de protéger le microcontrôleur : http://ruggedcircuits.com/html/ancp01.html

3.6. CODES SOURCES

Les programmes présentés dans cet article sont disponibles sur le site de REF-Union, dans la rubrique « Compléments des articles techniques ».

- Affichage LCD : IHM_Partie_1_Affichage.ino
- S-mètre : IHM_Partie_1_Smetre.ino
- Boutons poussoirs : IHM_Partie_1_Clavier_analogique.ino
- Programme complet de la partie 1 : IHM_Partie_1.ino

Conclusion :

La fin de cette partie s'achève avec la réalisation d'une première maquette fonctionnelle de l'interface homme-machine. Elle doit vous permettre de vous familiariser avec l'environnement de développement ARDUINO et d'acquérir les connaissances de base pour comprendre la syntaxe du logiciel.

Avant d'aborder les fonctions avancées qui seront décrites dans les articles suivants, n'hésitez pas à vous entraîner avec les tutoriaux sur les sites ARDUINO, à revoir et comprendre les programmes de cet article, à les optimiser, car ils ne sont pas parfaits, loin de là, et, pourquoi pas, à commencer à créer vos premiers propres programmes en fonction de votre projet.

Si la place est disponible, il est possible d'utiliser un afficheur plus grand. L'avantage est d'avoir directement sous les yeux plus d'informations pour contrôler le transceiver.

Dans l'exemple de la *figure 1-15*, sont affichés :
- la température du PA,
- l'heure UTC,
- la valeur de la tension d'alimentation,
- le mode de transmission,
- la VFO utilisé,
- la fréquence avec des séparateurs,
- le mode RIT (ici activé),
- le S-mètre,
- le menu en cours : ici le menu 1 avec Mode et Bande.

Dans cette version, trois boutons seulement sont utilisés :
- le premier pour sélectionner le numéro du menu,
- les deux autres boutons permettant de rentrer dans les sous menus et d'y effectuer un choix.

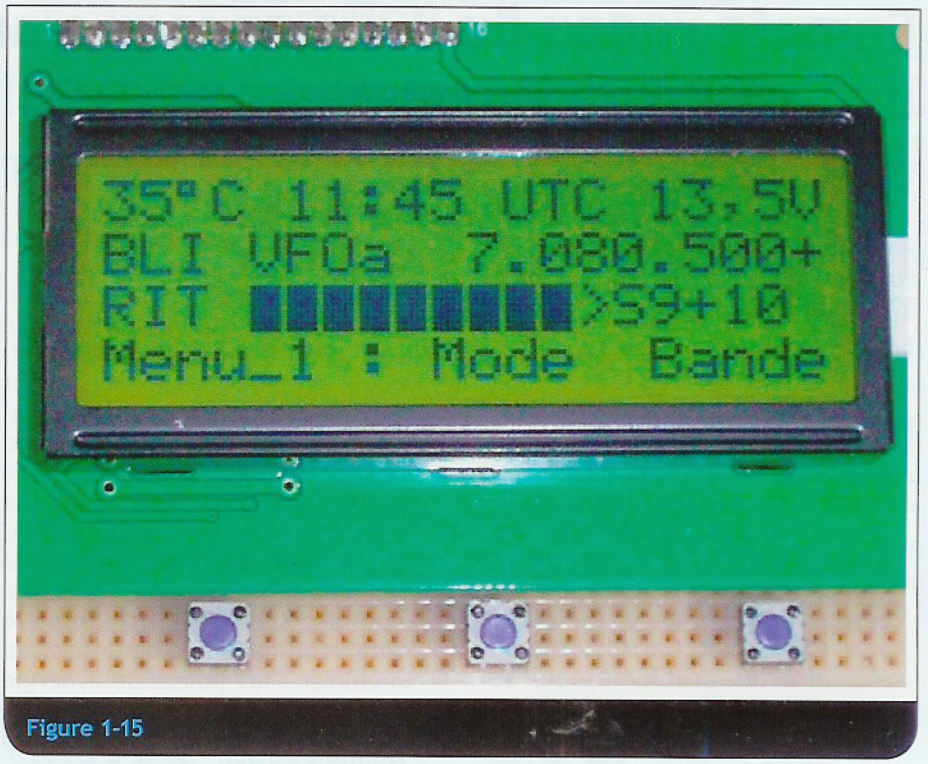

Figure 1-15

La dernière ligne peut être utilisée pour afficher d'autres informations lorsque le mode menu n'est pas sélectionné (autre filtre FI, Notch, etc.).

info

Le cobalt en remplacement du platine.

La production d'hydrogène par électrolyse de l'eau nécessite aujourd'hui l'utilisation de catalyseurs à base de platine, métal fort rare et donc fort cher.

S'inspirant de certains micro-organismes sachant produire de l'hydrogène à partir de l'eau grâce à des enzymes (hydrogénases) utilisant des métaux plus abondants, les chercheurs du CEA ont réussi à développer un nouveau catalyseur à base de sels de cobalt beaucoup moins coûteux. Pour en savoir plus :

http://www.cea.fr/recherche-fondamentale/du-cobalt-a-la-place-du-platine-95731

Information recueillie par Philippe Jeulin F1IFA.

Fin du télescope Herschel.

Quatre ans après son lancement, le télescope spatial de l'Agence Spatiale Européenne, Herschel, a cessé de fonctionner. Doté d'un miroir primaire de 3,5 mètres, il observait, depuis le point de Lagrange L2, l'univers dans l'infrarouge et les longueurs d'onde submillimétriques.

L'ensemble du dispositif devait donc être refroidi afin qu'il ne s'auto-sature pas avec son propre bruit. Cette réfrigération utilisait de l'hélium liquide superfluide dont les réserves sont aujourd'hui épuisées. Le satellite, baptisé ainsi en mémoire de William Herschel (1738-1822) l'inventeur des Infrarouges, laisse derrière lui une quantité considérable d'informations (600 programmes d'observation, 35 000 observations scientifiques et plus de 25 000 heures de données) qui occuperont les astronomes pendant de longues années encore. Pour en savoir plus :

http://www.esa.int/fre/For_Media/Press_Releases/Herschel_n_observera_plus_l_Univers.

Information recueillie par Philippe Jeulin F1IFA.