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

Au cours des deux articles précédents nous avons vu comment concevoir de manière progressive l'interface homme-machine d'un transceiver, en réalisant une maquette fonctionnelle sur une platine d'essais et en écrivant les différents programmes nécessaires à son fonctionnement.

La réalisation finale dépendra du projet de chacun, la mienne devrait avoir l'aspect suivant (*figure 3-1*).

Ce troisième et dernier article de cette série est essentiellement consacré à la gestion par menus de certaines fonctions et à rendre cette interface homme-machine la plus « universelle » possible.

1. GESTION DES MENUS (UTILISATION DES BOUTONS SÉLECT. ET MENU).

Avant de m'y attaquer, je ne pensais pas que les menus allaient me donner autant de fils à retordre...

En effet, après avoir consulté la communauté ARDUINO, je n'ai pas vraiment trouvé de quoi m'aider pour réaliser les menus de cette interface. Il a donc fallu se jeter à l'eau.

Le principe retenu est le suivant :

Lorsque l'on appuie sur le bouton Menu, l'écran s'efface pour afficher sur la première ligne les titres des différents menus disponibles. Une rotation de l'encodeur optique, dans un sens ou dans l'autre, les fait apparaître en boucle.

Un appui sur le bouton Menu ferme le mode menu avec retour à l'affichage normal (fréquence, etc.).

L'appui sur le bouton Sélect., alors que le mode menu est activé, affiche les différents paramètres possibles sur la seconde ligne de l'afficheur pour le menu sélectionné. Une rotation de l'encodeur optique, dans un sens ou dans l'autre, fait apparaître en boucle ces paramètres, le dernier affiché étant mémorisé.

Lorsqu'un choix supplémentaire est proposé, une seconde pression sur le bouton Sélect. fait afficher le paramètre à régler. Le réglage se fait, soit à l'aide de l'encodeur rotatif, soit à l'aide des boutons poussoirs sous l'afficheur (cas du réglage de l'horloge). Il n'y a pas de raison particulière d'utiliser l'encodeur ou les boutons, c'est juste pour montrer une alternative dans la gestion des menus.

Un nouvel appui sur le bouton Menu fait remonter les menus d'un niveau jusqu'à la fermeture du mode Menu et déclenche la mémorisation des nouveaux paramètres.

Bien entendu, chaque choix dans un menu a pour conséquence d'affecter une valeur à une variable qui sera traitée pour actionner les fonctions correspondantes.

Le programme IHM_Partie_3 utilise les sous-programmes déjà vus dans les articles précédents auxquels ont été rajoutées les lignes de code spécifiques au traitement des menus ainsi que celles nécessaires aux fonctions appelées par les menus.

Voyons cela en détail.
Les fonctions accessibles par les menus sont :

- sélection de la largeur du Filtre FI (2,7 ou 2, 2 kHz),
- activation du Notch (Marche, Arrêt),
- sélection de la mesure à afficher en mode émission (ROS ou puissance),
- gestion de la tension d'alimentation (Mesure, Seuil d'alerte),
- gestion de la température du PA (Mesure, Seuil d'alerte),
- activation du scanneur (Marche, Arrêt, Excursion),
- sélection du mode horloge (UTC, Local, réglages),
- réinitialisation des paramètres par défaut (RAZ).

D'autres fonctions peuvent être envisagées, il suffit dans ce cas de changer les titres, les choix possibles des paramètres et leur nombre, ainsi que les noms des variables associées comme cela est décrit dans les tableaux ci-dessous.

• Déclaration des variables

Une grande partie du programme est dédiée à la déclaration des variables pour la gestion des menus. En voici un résumé :

Char* Menu [ ] est un tableau contenant les titres des menus à afficher sous forme de chaînes de caractères : Filtre FI, Notch, Mesure TX, ALIM, TEMP, Scanneur, Horloge, RAZ.

Exemple : Menu [0] correspond à la chaîne de caractères « Filtre FI ».

Char* Choix [ ] [ ] est un tableau à 2 dimensions de chaînes de caractères contenant la liste de tous les paramètres à afficher accessibles depuis chaque menu. Ce tableau est organisé comme suit :
Voir "tableau 1" page ci-contre

Numéro de menu (index_menu)	Menu correspondant	Choix possibles dans chaque menu (index_choix)						
		0	1	2	3	4	5	6
0	Filtre FI (index_FI)	2,7 kHz	2,2 kHz					
1	Notch (index_Notch)	Arrêt	Marche					
2	Mesure TX (index_mesure)	ROS	Puissance					
3	ALIMentation (index_Alim)	=	Alerte					
4	TEMPérature (index_Temp)	=	Alerte					
5	Scanneur (index_Scanneur)	Arrêt	Marche	Excursion				
6	Horloge (index_horloge)	UTC	Local	He				
7	RAZ (raz)	Non	Oui					

Les noms entre parenthèses correspondent aux

Tableau 1 :

Numéro de menu	Choix
0	0 1
1	0 1
2	0
3	0
4	0
5	
6	
7	

Les n

Minutes	Jour	Date	Mois	Année

riables associées.

Nombre de choix possibles par menu (nb_choix[ ])
2
2
2
2
2
3
8
2

ssibles dans chaque menu (index_choix)

1							
1							
0	1	2					
0	1	2	3	4	5	6	7
0	1						

ns entre parenthèses correspondent aux variables associées.

bleau 2 :

Chaque choix possible est donc repéré par le numéro de menu (variable associée : index_menu) et sa position sur chaque ligne de menu (0, 1, ..., 7).

Rien n'empêche bien sûr d'enlever ou d'ajouter des menus ou des choix possibles. Il faudra juste redimensionner le tableau en conséquence.

Pour sélectionner un paramètre à afficher dans un menu, il faut donc adresser ce paramètre avec le numéro du menu (index_menu) et la position du paramètre (index_choix).

Exemple : Choix[5][2] correspond à la chaîne de caractères « Excursion ».

Menu[ ] et Choix[ ][ ] sont utilisés dans la fonction MENU () en association avec la fonction lcd.print () pour afficher les titres des menus.

Chaque nom de menu est associé à une variable.

Ces variables sont : index_FI, etat_Notch, index_Mesure, index_Alim, index_Temp, etat_Scanneur, index_Horloge, raz dont les valeurs sont contenues dans le tableau byte valeur_var[ ].

Ces variables peuvent prendre différentes valeurs en fonction du menu et des paramètres correspondants (choix possibles). Ces valeurs sont regroupées dans le tableau à deux dimensions byte valeur_choix[ ][ ] organisé de la même façon que le tableau Choix [ ][ ] :

Voir "tableau 2" ci-dessus

Par exemple, valeur_choix [index_menu][index_choix] au index_menu = 4 et index_ch 1 correspond à la valeur 1 menu 4, c'est-à-dire Seui pour le menu Tempéra

Pour faire les teste des tableaux e les menus, de constantes necessaire

byte

x =
ur le
'alerte
re.
ur les limites
naviguer dans
variables et des
pplémentaires sont
:.
eau_max[][] est un tableau
dimensions, de même struc-

ture que le tableau ci-dessus, mais indiquant le niveau maximal que peut atteindre un menu (voir le détail dans le programme).

byte nb_choix [] = {2,2,2,2,2,3,8,2} est un tableau contenant le nombre maximal de choix possibles pour chaque menu. Par exemple le menu 1 (Notch) n'a que deux choix (Arrêt, Marche). Dans ce cas, lorsque l'encodeur optique tourne, le choix est limité à 2 positions. Dans le cas du menu 6, 8 positions sont autorisées.

int etat_menu et int select indiquent si les modes Menu et Sélect. sont activés.

Enfin, la variable volatile int pulse permet de compter les impulsions générées par l'encodeur optique et de ne changer de menu ou de paramètre que toutes les 10 impulsions par exemple. Cela donne plus de souplesse lors de la rotation de l'encodeur.

Deux constantes sont définies pour fixer les limites des menus et des choix possibles :

```
# define max_menu 8 // Nombre maximal de menus de premier niveau

# define choix_max 8 // Choix maximal de paramètres par menu
```

• Traitement des interruptions dans les menus

Comme indiqué précédemment, c'est l'encodeur optique géré par interruptions qui permet de naviguer dans les menus. Les traitements A et B des interruptions démarrent par un test. Si le mode Menu est activé, l'encodeur optique sera totalement dédié à la navigation horizontale dans les menus, et pour :

- le réglage du seuil d'alerte de l'alimentation,
- le réglage du seuil d'alerte de la température du PA,
- le réglage de l'excursion du scanneur.

Les commentaires dans la partie « Traitement des interruptions » du programme en mode MENU doivent permettre d'en comprendre le fonctionnement.

• Sous-programme MENU ()

Le sous-programme MENU() est appelée par le programme principal pour gérer la partie affichage des menus et la navigation verticale dans les menus.

Il est structuré de la façon suivante (voir les commentaires dans le programme pour plus de précisions) :

- on teste tout d'abord l'appui sur le bouton Menu, puis sur le bouton Sélect,
- si le niveau des menus est 1 (cas des titres des menus) on teste la rotation droite ou gauche de l'encodeur optique pour afficher le menu correspondant,
- puis une série de tests sont effectués pour savoir quel menu est activé et, dans ce cas, effectuer les instructions d'affichage et de réglages correspondantes, parfois, grâce à l'appel de fonctions spécifiques (SCANNEUR (), HORLOGE (), DATE () ...).

Côté affichage :

Le symbole <o> permet d'identifier la ligne active dont la valeur changera par rotation de l'encodeur optique.

Si le symbole S est affiché, cela signifie que la touche Select. est active pour le menu en cours.

Si S et <o> sont affichés simultanément, cela signifie que deux choix de navigation sont possibles.

La dernière ligne de la fonction MENU() fait appel à la fonction Rafraich_var() qui a pour but de rafraîchir les variables à chaque passage pour mémoriser les éventuels changements dans la gestion des menus.

• Programme principal void loop ()

Le programme principal est quant à lui très simple. Au début, la fonction Rafraich_tab_var(), inverse de Rafraich_var(), initialise le tableau des variables des menus avec certaines valeurs stockées en mémoire.

Puis, les différents sous-programmes et fonctions étudiés dans cette série d'articles sont appelés en séquence.

Après la gestion de la commutation de bande (40/10 m), vient la sauvegarde des données avant la mise hors tension, en mode manuel par défaut.

Le programme principal se termine par la fonction SYNTHESE () qui a pour effet de lancer la synthèse directe en fréquence. Par défaut, le mode « démo » est activé et la fréquence affichée est celle qui est synthétisée.

2. TRAITEMENT DES FONCTIONS APPELÉES PAR LES MENUS

Certaines fonctions ci-dessus ne sont que des exemples et leur traitement n'est pas décrit dans ce paragraphe, seules quelques pistes sont proposées. Elles pourront faire l'objet d'un article supplémentaire si besoin, mais d'ores et déjà, si vous avez compris les programmes précédents, vous êtes en mesure de réaliser les lignes de code nécessaires pour les traiter.

C'est le cas des fonctions Filtre FI, Notch et mesure de la puissance qui nécessitent l'ajout de 3 ports d'extension, car ceux du microcontrôleur sont quasiment tous utilisés (reste le port analogique A3). Pour cela, on pourra utiliser un circuit I2C d'extension de port, en utilisant la bibliothèque Wire.h.

Les autres fonctions décrites sont réalisées par voie logicielle et/ou avec des composants dédiés.

• Mesure de la tension d'alimentation : ALIM ()

Une image de la tension d'alimentation est disponible sur le port analogique A2 qui sert à déclencher la sauvegarde des données (voir article précédent).

C'est l'instruction tension_sonde_alim = analogRead (2) qui permet de lire la tension d'alimentation. Cette fraction de tension (4,4 V pour 13,8 V d'alimentation) est convertie par l'instruction map () pour obtenir une lecture directe de la tension (à un coefficient 100 près) :

```
tension_sonde_alim = map (analogRead (2), 0, 900, 0, 1380) ;
```

En fonction de l'afficheur utilisé (2 ou 4 lignes), une alerte clignotante sur 3 caractères (ALM en alternance avec un point d'exclamation) est affichée selon le seuil réglé via le menu.

Sur un afficheur de 4 lignes, la tension en volts est affichable en permanence (*figure 3-7*).

Dans le programme IHM_Partie_3 de cet article, c'est l'option alerte qui a été retenue, puisque l'interface présentée ici

fré-
produit,
MHz. J'ai
FI à partir
de 2,2 kHz de
e, l'autre de 2,7
des valeurs dans le
e FI).

élange choisi étant
adyne, la fréquence de
scillateur local est égale à la
réquence à recevoir plus la fré-
quence intermédiaire.

Un décalage (« offset ») de 1,5 kHz
est appliqué à l'oscillateur local
(-1,5 kHz en BLI, + 1,5 kHz en
BLU pour assurer le centrage
du spectre FI dans les filtres à
quartz.

De ce fait, le calcul de l'OL est
le suivant :

freqOL = freq + freqFI + offset.

Exemple 1 : la fréquence à rece-
voir est de 7100 kHz en mode
BLI.

freqOL = 7100 + 11059 -1,5 =
18157,5 kHz

Exemple 2 : la fréquence à rece-
voir est de 28480 kHz en mode
BLS.

freqOL = 28480 + 11059 + 1,5 =
39540,5 kHz

Attention : les lignes de code
correspondant à ce calcul sont
masquées par // dans le pro-
gramme principal.

En cas de mélange infradyne, il
ne faudra pas oublier d'inverser
le sens de rotation de l'encodeur
optique en changeant le signe
d'incrémentation/décrémen-
tation de la fréquence dans le
traitement des interruptions.

• Horloge

Cette fonction est confiée à un
circuit dédié piloté par le bus
I2C, le DS1307. Ce circuit est ali-
menté par une pile lithium de
V assurant le fonctionnemen
l'horloge lorsque le tran
est éteint. Le schéma d
de l'horloge est très

Un circuit intég
deux résistanc
lithium de 3 V
« tourner »

Deux m
vus :
do

ne comporte qu'un afficheur de
deux lignes.

Le menu ALIM permet d'acce-
der au réglage du seuil d'alerte
(variable '. seuil_alim) réglable
par l'encodeur optique. Ce seuil
d'alerte doit être réglé au-dessus
du seuil de déclenchement de la
sauvegarde automatique qui est
de 11 V. Prendre par exemple
11,5 V.

• Mesure de la température : TEMP ()

Plusieurs options de capteurs
sont possibles, un circuit dédié
(I2C ou à sortie analogique), une
diode, une thermistance, etc.

Puisque le port analogique A3 est
encore disponible, la solution la
plus simple est d'utiliser un cap-
teur à sortie analogique, comme
le LM35 (figure 3-2).

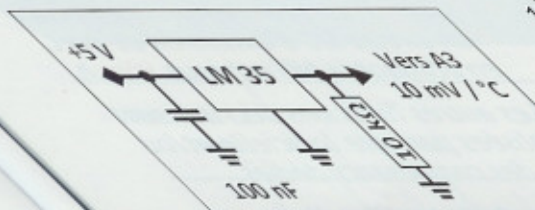

Figure 3-2 : LM35

Celui-ci fournit une t
nue de 10 mV par
Pour 20 °C la t
est de 200 m
1000 mV a
l'ordre d
mais l
l'ap

Le menu TEMP permet d'acce-
der au réglage du seuil d'alerte
(variable '. seuil_temp) réglable
par l'encodeur optique.

La fonction ALERTE() gère l'af-
fichage des alertes ALIM et TMP,
même si les 2 alertes sont acti-
vées simultanément.

• Mise en œuvre du scanneur : SCANNEUR ()

Une fonctionnalité intéressante
pilotée par logiciel est le scan-
neur qui permet de scruter une
portion de bande choisie sans
avoir à intervenir sur l'équipe-
ment.

Lorsque le scanneur est activé
via le menu Scanneur, la syn-
thèse en fréquence se fait dans
les limites de l'excursion deman-
dée, autour de la dernière fré-
quence affichée, et avec un pas
de 100 Hz.

L'excursion crête à crête peut
être choisie entre 10 et 200 kHz
par menu.

Le pas est fixe, mais pourra
être rendu variable en s'inspi
des lignes de code de l'
sion, et, bien sûr, en aj
menu correspondant
constituer un bon

La vitesse de s
matiquement
de l'excurs

- pour
100

En fait, la vitesse est réglée par
l'introduction d'un retard dû à
l'instruction delay (200 - excur-
sion*0.75), qui donne un retard
de 50 ms par pas pour 200 kHz
et près de 200 ms pour 10 kHz.
D'autres valeurs peuvent être
choisies.

La rotation dans n'importe quel
sens de l'encodeur optique
arrête le scanneur sur la fré-
quence en cours et désactive le
mode Menu.

• Fréquence affichée, fréquence intermédiaire et oscillateur local

Depuis le début, la fréquence
affichée et celle synthétisée
étaient identiques. Il est temps
maintenant de déterminer la
fréquence de l'oscillateur local
(OL) en fonction de la fréquence
intermédiaire choisie (FI).

Là encore, le choix de ces fré-
quences dépendra de chacun
et surtout du principe d'émis-
sion/réception retenu (simple
ou double changement de fré-
quence, principe de mélange et
de conversion de fréquence, type
de détecteur de produit, etc.).

A titre d'exemple, ayant un
stock de quartz de 11,059 MHz,
j'ai opté pour une réception

à simple changement
quence et détecteur d
avec une FI de 11,05
réalise deux filtr
des quartz, l'u
bande passa
kHz (raisso
menu Fil

Le

sur

sion conti-
egrée Celsius.
sion de sortie
, pour 100 °C de
et une précision de
degré selon le modèle,
gement suffisante pour
cation qui en est faite.

mme pour la tension d'alimen-
tation, la gestion de la tempéra-
ture consiste à lire la tension sur
le port analogique A3 et à affi-
cher une alarme clignotante ou
la température en degré Celsius
selon l'afficheur utilisé.

de
éiver
ctrique
mple.

, un quartz,
s et une pile au
uffisent pour faire
horloge (figure 3-5).

des d'affichage sont pré-
UTC (Temps universel coor-
nié) et Local, plus la date.

En fonction du type d'afficheur utilisé on peut prévoir un affichage de l'heure à la demande en entrant dans le menu correspondant (cas avec un écran de 2 lignes) ou un affichage permanent dans le cas d'un écran à 4 lignes comme montré *figure 3-7*.

C'est le sous-programme HORLOGE() qui est chargé d'afficher l'heure UTC ou locale sur une ligne (pratique pour noter l'heure de début et de fin des QSO), et le programme DATE () pour afficher la date en-dessous.

Le programme consiste à lire dans l'EEPROM du circuit DS1307 la valeur des variables à partir de l'adresse 0x68 (adresse du circuit DS1307).

Ces variables sont sauvegardées en format BCD (Binaire Codé Décimal). La fonction BCD_DEC (byte dec) permet de convertir chaque valeur en format décimal. C'est plus pratique de travailler en décimal pour mettre à l'heure l'horloge et afficher les informations.

Comme indiqué plus haut, le menu permettant le réglage de l'heure et de la date utilise deux boutons poussoirs.

Par exemple, pour changer d'heure, une fois rentré dans le menu de réglage de l'heure, les symboles H- et H+ s'affichent au-dessus des boutons 3 et 4 pour augmenter ou diminuer l'heure d'une unité (*figure 3-3*).

Le même principe est utilisé pour

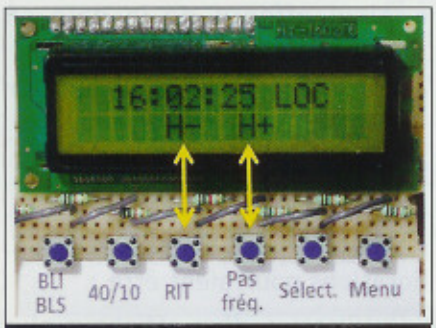

Figure 3-3 : Heures

les minutes, le jour du mois, le jour de la semaine, le mois et l'année.

Le réglage des secondes se fait automatiquement avec celui des minutes + 10 secondes. Cela signifie que pour synchroniser l'horloge sur celle du PC ou de l'horloge parlante, il faut régler les minutes en dernier en appuyant une fois sur le bouton Mn+ au moment où l'horloge de référence a dépassé la minute souhaitée + 10 secondes.

Les fonctions de réglage sont REG_HORLOGE () et REG_DATE() respectivement pour l'heure et la date.

Ces programmes font appel à la fonction ENVOI (byte pos_mem, byte parametre), qui écrit la valeur parametre à la position mémoire pos_mem.

Lors de l'écriture en mémoire du circuit DS1307, la fonction DEC_BCD () opère la conversion de chaque paramètre écrit en décimal vers le format BCD.

A noter qu'en mode réglage, la date est affichée sur la première ligne de l'afficheur, alors qu'en mode normal elle est affichée sur la seconde ligne, en dessous de l'heure. C'est la fonction DATE (byte reg) qui est chargée d'afficher la date, sur la première ligne quand reg = 0, sur la seconde quand reg = 1.

Attention : l'horloge calcule son changement de date à partir de l'heure locale. L'heure UTC est simplement calculée en ajoutant ou en enlevant des heures à l'heure locale. Cela signifie que s'il est 23:30 UTC, la date du lendemain est déjà affichée si l'heure locale est UTC + 1.

Encore un petit exercice en perspective pour pallier cet inconvénient.

Le décalage UTC/Local n'est pas réglable par menu, mais directement dans le sous-programme HORLOGE ().

- **Réinitialisation par défaut : RAZ ()**

La fonction RAZ a pour effet de réinitialiser toutes les variables mémorisées à leur valeur par défaut. Ces paramètres initiaux ne peuvent être modifiés que dans le programme RAZ ().

3. REMPLACEMENT DU MODULE ARDUINO PAR LE MICRO-CONTRÔLEUR PROGRAMMÉ.

Avant de remplacer le module ARDUINO par le microcontrôleur seul, il est temps de refaire tous les tests sur la platine d'essais qui est maintenant complètement câblée (figure 3-4).

Une fois que le programme a été testé et qu'il donne toute satisfaction, il reste à remplacer le module ARDUINO par le microcontrôleur seul. En effet, il n'est pas indispensable d'inclure la plate-forme de développement ARDUINO dans

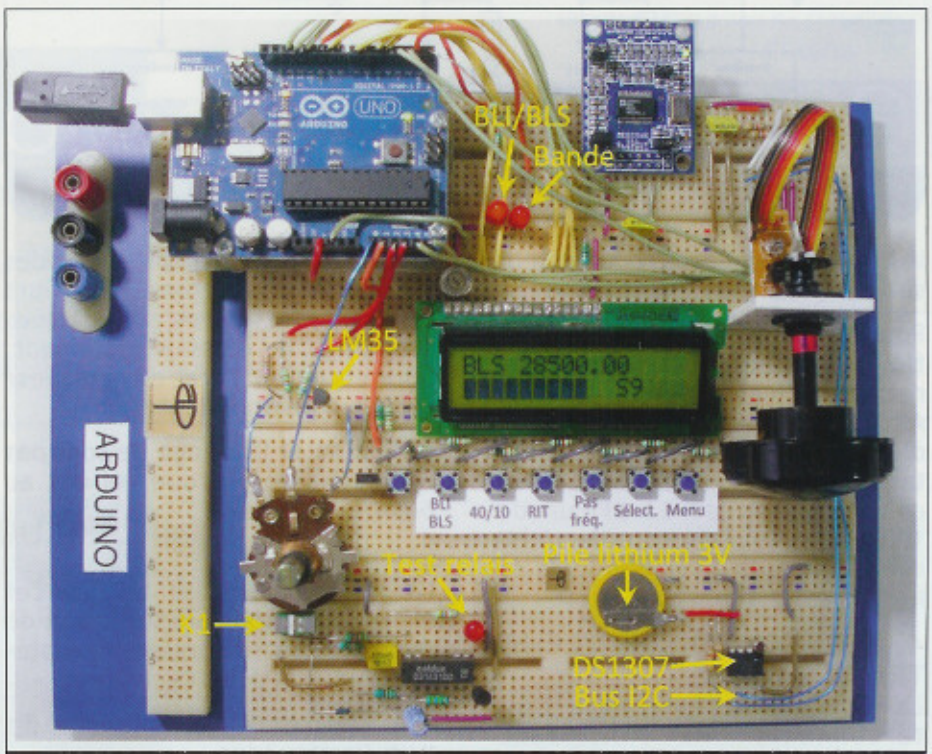

Figure 3.4 : Platine essais

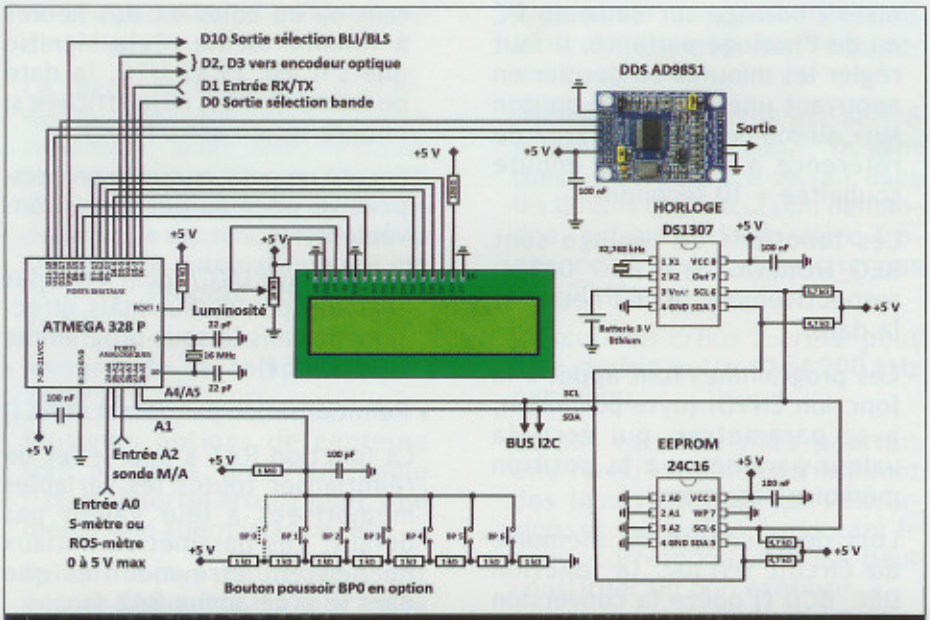

Figure 3.5 :

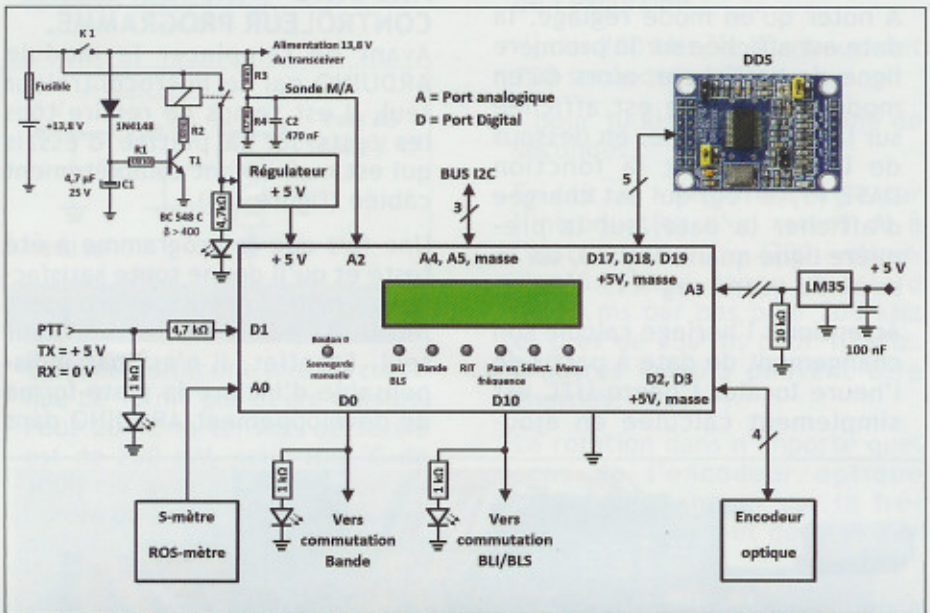

Figure 3.6 : Interconnexion

le transceiver, le microcontrôleur suffit.

Pour que le microcontrôleur fonctionne, il faut bien entendu le retirer délicatement du module ARDUINO et le câbler sur la plaque d'essai sans soudure en lui ajoutant un quartz de 16 MHz, deux condensateurs de 22 pF, un de 100 nF et une résistance de 10 kΩ comme indiqué dans le schéma de la *figure 3-5*.

4. INTERCONNEXION AVEC LE TRANSCEIVER

La *figure 3-6* représente l'interconnexion de l'interface homme-machine avec les différents modules et organes de commande du transceiver. Les diodes électroluminescentes, et leurs résistances associées, sont câblées sur la platine d'essais seulement, pour visualiser l'état des entrées/sorties correspondantes.

L'EEPROM et l'horloge ne sont pas représentées puisque reliées au bus I2C.

Codes sources

Le programme présenté dans cet article est disponible sur le site de REF, dans la rubrique « Compléments des articles techniques » :

- Programme complet de la partie 3 : IHM_Partie_3.ino

CONCLUSION

La conception de l'interface homme-machine est maintenant terminée, sa faisabilité confirmée.

« Il ne reste plus qu'à » la relier à l'électronique du transceiver, en réalisant un circuit imprimé qui prendra place, par exemple, derrière la face avant, et qui comportera, le microcontrôleur, l'afficheur, les boutons poussoirs, l'EEPROM, l'encodeur optique, l'horloge, le circuit du relais de sauvegarde et les quelques autres composants et connecteurs nécessaires.

Bien entendu, de nombreuses améliorations peuvent être envisagées, tant sur le plan logiciel que matériel.

Concernant le programme, les solutions retenues pour réaliser le programme de cette interface, sont celles d'un débutant qui ne connaissait rien au langage C/C++ avant de démarrer cette réalisation. C'est dire qu'il doit y avoir un bon nombre d'optimisations à faire pour améliorer sa robustesse, son efficacité, sa lisibilité, son intégrité, sa modularité, etc., comme disent les « pros » de la programmation.

Les fonctionnalités peuvent également être étendues ou améliorées, par exemple :

- sélection d'autres bandes,
- ajout d'autre mode (CW, FM...),
- VFO A, VFO B,
- mémorisation de fréquences favorites,
- commande du décalage FI,
- calcul effectif de la date en mode UTC,
- pas et vitesse du scanneur réglables,
- VOX,
- etc., etc.

Côté matériel, les premières améliorations pourraient être :

- l'utilisation d'un afficheur plus grand, plus moderne (type OLED) ou même graphique en couleur (*figure 3-7*) permettant d'afficher plus d'informations à la fois,
- l'ajout d'un port d'extension I2C (par ex. PCF8574, PCA9554, MCP23008, etc.) nécessaire aux commutations supplémentaires (bandes, filtres FI ou audio), etc.