Article ADREF 13

Bonjour,

Je voulais juste vous partager une petite réalisation autour de l'IC-7300.

L'idée part de différents besoins et constats :

LE CHOIX DES MANIPULATEURS

J'ai 2 manips connectés en permanence sur le 7300 : une pioche et un double contact. Le problème est que lorsque l'on veut passer de l'un à l'autre il faut rentrer dans les menus et sous menus pour pouvoir changer la configuration. Ça n'est pas instantané.

Je me suis donc inspiré de diverses réalisations vues sur le net pour faire mon montage.

Il met en œuvre un Arduino Nano qui communique avec l'IC-7300 par le port CI-V. Le protocole CI-V est décrit en détail dans les documents ICOM. Il permet de lire et/ou de modifier quasiment l'ensemble des paramètres du transceiver grâce à une liaison série ou éventuellement USB.

Il suffit donc de modifier directement en mémoire de l'IC-7300 la configuration du manipulateur.

Et tant qu'à avoir établi la liaison entre le cœur de la bête et l'Arduino, autant en profiter et mettre à disposition d'autres paramètres.

LA PROPRETE DE L'ECRAN TACTILE

Là, c'est un autre besoin qui préside au choix : J'aime que l'écran LCD du 7300 reste propre et donc j'aimerais ne plus avoir à y mettre les doigts.

Donc je vais donc télépiloter les paramètres que j'utilise souvent auxquels on accède par l'écran tactile :

- Le changement de mode USB – LSB – CW – AM
- Le changement de filtres FILTRE 1 – 2 – 3

Il y aura donc 4 Boutons Poussoirs sur le boitier qui permettront de modifier les paramètres cités :

- BP1 Configuration Keyer = Pioche
- BP2 Configuration Keyer = Paddle (Double contacts)
- BP3 Choix LSB USB CW
- BP4 Choix Filtre 1 2 3

LE BARGRAPH S-METRE EST TROP PETIT

Lorsque l'on utilise le Waterfall de taille maxi, ce qui est mon cas habituellement, l'afficheur du S-mètre devient tout petit. De plus, étant un peu nostalgique des aiguilles, le protocole CI-V nous permet de récupérer en temps réel la valeur affichée sur le S-mètre. Cette valeur sera donc envoyée

via une sortie analogique sur un bon vieux S-mètre à aiguille. Une résistance ajustable, permettra de calibrer la valeur affichée.

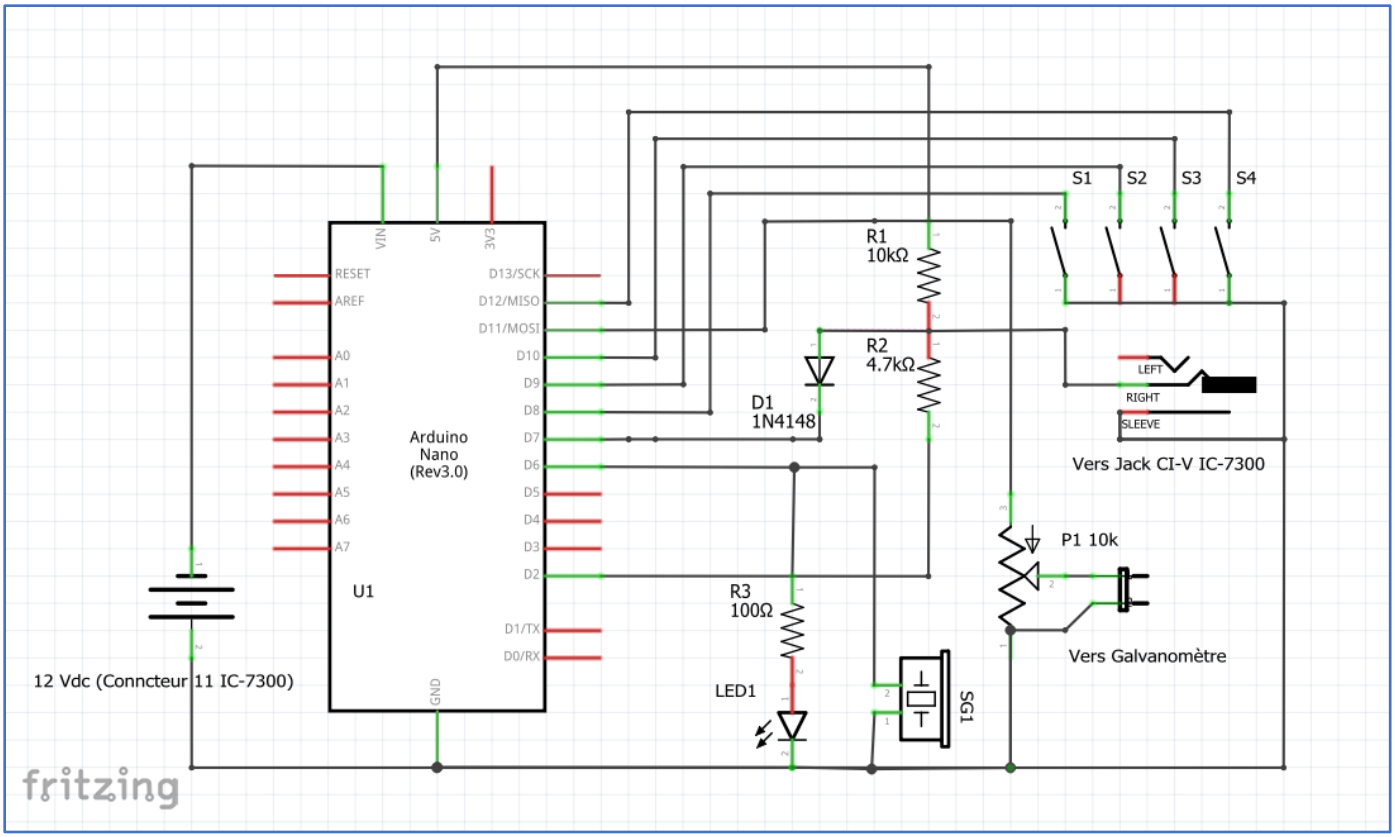
UNE FONCTION DE SECURITE

Il m'arrive parfois de passer en émission sans que le Tuner automatique soit enclenché. Il peut s'en suivre un SWR élevé.

De même que la valeur du S-mètre, le SWR peut être lu en temps réel.

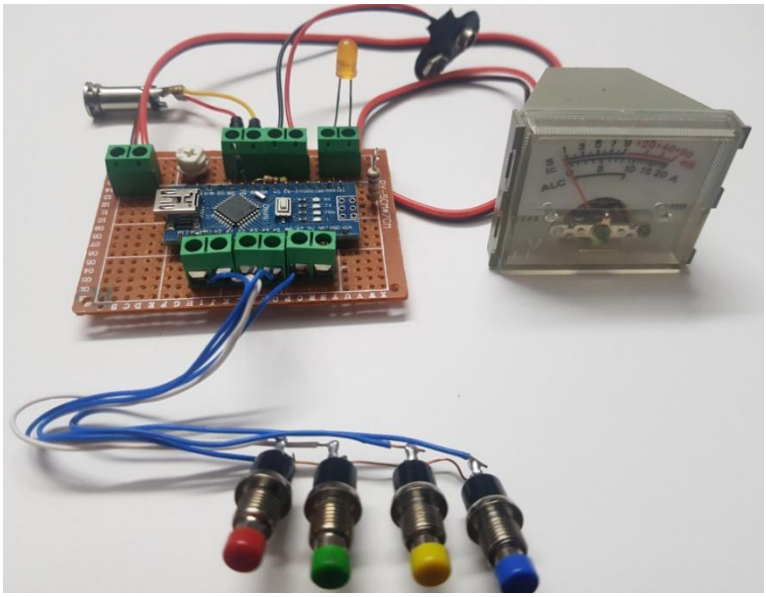
La valeur récupérée par l'Arduino servira donc à déclencher une alarme (LED Rouge clignotante et Buzzer) si sa valeur atteint ou dépasse 3:1.

LE SCHEMA



LA REALISATION

La réalisation se fit sur une plaquette d'essai vu le nombre réduit de composants.



Un boitier TEKO 362 reçu de chez Reichelt reçoit très simplement le montage.

CONCLUSION

Il s'agit d'un montage ne présentant pas de difficulté particulière.

Il permet de s'initier à l'Arduino et de pénétrer dans l'univers sans fond des paramètres internes de nos chers transceivers « modernes ».

D'autres fonctions peuvent être imaginées :

- Changement de puissance 0, 10W, 50W, 100W
- Affichage sur écran couleur OLED (j'ai laissé la place sur le boitier)
- Utilisation d'un codeur rotatif pour gérer les menus
- Sélection rapide des mémoires avec le codeur rotatif
- Commutation CENT/FIX
- Communication Bluetooth pour un module sans fil
- ……..

Cette réalisation faite autour de l'IC-7300 pourrait être déclinée autour de nombreux autres transceiver pour autant que l'on ait accès aux informations concernant le protocole de communication.

A votre service pour tous renseignements complémentaires et bien sûr à l'écoute de vos suggestions.

73 de Pierrick

F1CEP

LE CODE

```
// IC7300 Modification des Paramètres par le protocole CI-V
// D'après entre autres ON7DQ/KF0CR
// Validé 30 octobre 2020

/*
 * Bilan des E/S
 *
 *
 * D1
 * D2      Rx IC-7300 (interruption)
 * D3*
 * D4
 * D5*
* D6*     LED Alarme SWR et Buzzer* = PWM sur Nano
 * D7   Tx IC-7300
 * D8      Key Pin
 * D9*     Paddle Pin
 * D10*    Mode Pin
 * D11*    Sortie Smètre
 * D12     Filtre Pin
 * D13  LedBuilt in
 *
 *
 *
 */


/*
```

Les fonctions réalisées sont :

- Télécommande des fonctions suivantes :

    BP1 Configuration Keyer Pioche (Straight)

    BP2 Configuration KeyerPaddle (Double contact)

    BP3 Changement de Mode LSB -> USB -> CW

    BP4 Changement de FILTRE 1 -> 2 -> 3

- Affichage de la valeur du S mètre

- Alarme sur SWR > 3

*/

```
#include <SoftwareSerial.h>  // Pour communication Série vers IC-7300

// IC7300 Paramètres configurés dans l'IC-7300

#define BAUD_RATE 19200     // CI-V speed
#define TRX_address (0x94)  // HEX 94

// Communication Série (CI-V)
//
// RX = IC7300 to Arduino  : Doit être une entrée Interruption Pin 2
// TX = Arduino to IC7300  : direct sur Pin 7


#define RxPin 2     // Pin Rx liaison série IC-7300
#define TxPin 7     // Pin Txliaison série IC-7300
```

```
#define keyPin      8     // Bouton manip Pioche to GND
#define paddlePin   9  // Bouton Paddle to GND
#define modePin     10   // Bouton Changement de mode to GND
#define filtrePin   12   // Bouton Changement de Filtre to GND
#define sMeterOut   11// Sortie analogique PWM S-meter
#define ledPin      13    // LED Interne Arduino Nano
#define ledSWR      6     // LED Alarme SWR
intreadCounter; // Compte le nombre d'octets reçus du transceiver
int sMeterVal1;  // Stocke le MSB octet BCD
int sMeterVal2;  // Stocke le LSB octet BCD

int Swr1Val ; // Pour lecture du SWR
int Swr2Val ;
intSwrVal  ;
intLimiteSWR = 104 ; // Pour un SWR de 3

byte byteMode;   // Pour lecture mode Actuel et écriture
byte byteFiltre; // Pour Lecture Filtre Actuel et écriture



// Pour Anti rebond des Bouton poussoirs
//////////////////////////////////////////

intreading ; // Lecture de l'entrée physique

intInitBPMode   = 0 ; // Pour créer un délai avant première action.
intInitBPFiltre = 0 ; // Pour créer un délai avant première action.

intBPmodeState ;      // Valeur courante de l'entrée
```

```
intlastBPmodeState;   // Valeur précédente de l'entrée

intBPfiltreState ;      // Valeur courante de l'entrée
intlastBPfiltreState;   // Valeur précédente de l'entrée

unsigned long lastDebounceTimeBPmode   = 0;  // La derniere fois que l'entrée a changé
unsigned long lastDebounceTimeBPfiltre = 0;  // La derniere fois que l'entrée a changé

unsigned long debounceDelay = 50;    // Tempo pour l'anti-rebond des entrées

// Création d'une instance mySerial de SoftwareSerial
SoftwareSerialmySerial(RxPin, TxPin);

////////////////////////////////
//   INITIALISATIONS            //
////////////////////////////////

void setup()
{

  // Serial.begin(9600); // Pour mise au point sur console


mySerial.begin(BAUD_RATE); // Pour se connecter à l'IC-7300

pinMode(keyPin, INPUT_PULLUP); // Entrée avec Résistance interne 10k en Pull-up


pinMode(paddlePin, INPUT_PULLUP); // Entrée avec Résistance interne 10k en Pull-up
```

```
pinMode(modePin, INPUT_PULLUP); // Entrée avec Résistance interne 10k en Pull-up

pinMode(filtrePin, INPUT_PULLUP ); // Entrée avec Résistance interne 10k en Pull-up

pinMode(ledPin, OUTPUT);
pinMode(ledSWR, OUTPUT) ; //PWM
}

////////////////////////////////////////
//   Déclaration des Sous-Programmes //
////////////////////////////////////////


voidLectureModeetFiltre()
{

// lecture Mode et Filtre Actuels
mySerial.flush();

  // Sébut de la séquence d'interrogation de la valeur du Mode et Filtres
mySerial.write(0xFE);
mySerial.write(0xFE);
mySerial.write(TRX_address); // Adresse CI-V dans le 7300
mySerial.write(0xE0);
mySerial.write(0x04);  // Lecture mode, Commande 4    La réponse fera 8 caractères FE,FE,E0,94,04,Mode,Filtre,FD
mySerial.write(0xFD); // Fin de la séquence
delay(20);

  // Lecture des infos du transceiver
intnbChar = mySerial.available();
```

```
 if (nbChar> 0) {

    byte byteRien1 = mySerial.read() ; // FE
    byte byteRien2 = mySerial.read() ; // FE
    byte byteRien3 = mySerial.read() ; // E0
    byte byteRien4 = mySerial.read() ; // 94
    byte byteRien5 = mySerial.read() ; // 04
byteMode = mySerial.read()  ; // Mode
byteFiltre = mySerial.read(); // Filtre
    byte byteRien6 = mySerial.read() ; // FD



 /* Serial.print("mode ") ;
Serial.println(byteMode) ;

Serial.print("filtre ") ;
Serial.println(byteFiltre) ;
Serial.println("           ");
 */

delay(20);
              }
}

////////////////////////////////////
//   BOUCLE PRINCIPALE           //
////////////////////////////////////

voidloop()
```

```
{

// Gestion des Boutons Poussoir
////////////////////////////////

//Bouton Rouge Key Pioche
  if ( digitalRead (keyPin) == LOW)  {

mySerial.flush();
mySerial.write(0xFE);
mySerial.write(0xFE);
mySerial.write(TRX_address); // Adresse CI-V dans le 7300
mySerial.write(0xE0);

mySerial.write(0x1A); // Commande 1A Sous Cde 05 164 00 Pioche
mySerial.write(0x05);
mySerial.write(0x01);
mySerial.write(0x64);
mySerial.write((byte)00);
mySerial.write(0xFD);
delay(20);
  }

//Bouton Vert Key Paddle

    if ( digitalRead (paddlePin) == LOW)  {

mySerial.flush();
mySerial.write(0xFE);
mySerial.write(0xFE);
mySerial.write(TRX_address); // Adresse CI-V dans le 7300
```

```
mySerial.write(0xE0);

mySerial.write(0x1A); // Commande 1A Sous Cde 05 164 02 Paddle
mySerial.write(0x05);
mySerial.write(0x01);
mySerial.write(0x64);
mySerial.write(0x02);
mySerial.write(0xFD);
delay(20);
 }

//Bouton Jaune Changement de Mode

//
// Changemet de Mode : Si LSB(00) -> USB(01) Si USB(01) -> CW(03) Si CW(03) -> LSB(00)
//

reading = digitalRead(modePin);

 if (reading != lastBPmodeState) {
lastDebounceTimeBPmode = millis();
 }

 if ((millis() - lastDebounceTimeBPmode) >debounceDelay) {
   if (reading != BPmodeState) {
BPmodeState = reading;

     if (BPmodeState == HIGH) {


// Action Bouton1 Si Premier tour passé
```

```
if (InitBPMode == 1)
{
LectureModeetFiltre() ;
delay(20);

switch (byteMode) {
  case (0x00):
byteMode = (0x01) ;
    break;
  case (0x01):
byteMode = (0x03) ;
    break;
  case (0x03):
byteMode = (0x00) ;
    break;



  default:
    // Si rien à faire ...
    //
    break;
                }

mySerial.flush();
mySerial.write(0xFE);
mySerial.write(0xFE);
mySerial.write(TRX_address); // Adresse CI-V dans le 7300
mySerial.write(0xE0);

mySerial.write(0x01);       // Fonction 1 Ecriture Mode et Filtre
```

```
mySerial.write(byteMode);    // Nouveau Mode
mySerial.write(byteFiltre);  // Ancien Filtre
mySerial.write(0xFD);
}  else { InitBPMode = 1 ; }

     }
   }
 }
lastBPmodeState = reading;




//Bouton Changement Filtre Bleu

//
// Changemet de Filtre : Si Filtre(01) -> Filtre(02) Si Filtre(02) -> Filtre(03) Si Filtre(03) -> Filtre(01)
//
reading = digitalRead(filtrePin);

  if (reading != lastBPfiltreState) {
lastDebounceTimeBPfiltre = millis();
  }

  if ((millis() - lastDebounceTimeBPfiltre) >debounceDelay) {
    if (reading != BPfiltreState) {
BPfiltreState = reading;

      if (BPfiltreState == HIGH) {
```

```
// Action Bouton2 si pas premier tour de boucle

if (InitBPFiltre == 1)
{
LectureModeetFiltre() ;

switch (byteFiltre) {
  case (0x01):
byteFiltre = (0x02) ;
    break;
  case (0x02):
byteFiltre = (0x03) ;
    break;
  case (0x03):
byteFiltre = (0x01) ;
    break;



  default:
    // Si rien à faire ...
    //
    break;
                }

mySerial.flush();
mySerial.write(0xFE);
mySerial.write(0xFE);
mySerial.write(TRX_address); // Adresse CI-V dans le 7300
mySerial.write(0xE0);

mySerial.write(0x01);       // Fonction 1 Ecriture Mode et Filtre
```

```
mySerial.write(byteMode);    // Ancien Mode
mySerial.write(byteFiltre);  // Nouveau Filtre
mySerial.write(0xFD);

} else { InitBPFiltre = 1 ; }

     }
   }
 }
lastBPfiltreState = reading;



// Gestion du S-mètre
//////////////////////

// Lecture de la valeur du S-mètre et envoie sur la sortie analogique

mySerial.flush();

  // Sébut de la séquence d'interrogation de la valeur du S meter
mySerial.write(0xFE);
mySerial.write(0xFE);
mySerial.write(TRX_address); // Adresse CI-V dans le 7300
mySerial.write(0xE0);
mySerial.write(0x15); // Fonction 15 02 Lecture du S-Mètre.
mySerial.write(0x02);
mySerial.write(0xFD);
delay(20);

  // nowread info from radio
```

```
int nbChar2 = mySerial.available();


 if (nbChar2 > 0) {
  for (intreadCounter = 0; readCounter< nbChar2 ; readCounter++) {
   byte byteRead = mySerial.read();


   if (readCounter == 6){
    sMeterVal1 = ( (byteRead/16*10) + (byteRead%16) ); // First byte: convertfrom BCD to decimal.
   }


   if (readCounter == 7){
    sMeterVal2 = ( (byteRead/16*10) + (byteRead%16) ); // Second byte: convertfrom BCD to decimal.


analogWrite(sMeterOut, ((sMeterVal1 * 100) + sMeterVal2)); // Calculate and write the S-meter value on the S-meter output pin.


delay(20);
   }
  }
 }


// Gestion du SWR
////////////////////


// Lecture de la valeur du SWR et allumage d'une LED si >= 3
// Valeurs relevées
//  1.5    32
//  3.2    112  ==> Alarme sur LED
//  3.7    132
//  7      203
//
```

```
mySerial.flush();

 // Début de la séquence d'interrogation de la valeur du SWR
mySerial.write(0xFE);
mySerial.write(0xFE);
mySerial.write(TRX_address); // Adresse CI-V dans le 7300
mySerial.write(0xE0);
mySerial.write(0x15); // Fonction 15 12 Lecture du SWR 0000 à 0255.
mySerial.write(0x12);
mySerial.write(0xFD);
delay(20);

 // nowread info from radio
int nbChar3 = mySerial.available();

 if (nbChar3 > 0) {
   for (intreadCounter = 0; readCounter< nbChar3 ; readCounter++) {
     byte byteRead = mySerial.read();

     if (readCounter == 6){
       Swr1Val = ( (byteRead/16*10) + (byteRead%16) ); // First byte: convertfrom BCD to decimal.
     }

     if (readCounter == 7){
       Swr2Val = ( (byteRead/16*10) + (byteRead%16) ); // Second byte: convertfrom BCD to decimal.
```

```
SwrVal = Swr1Val * 100 + Swr2Val ; // Calculate and write the S-meter value on the S-meter output pin.


      // Serial.print("SWR = ");
      // Serial.println(SwrVal) ;


// Alarme si SWR >LimiteSWR
// ***************************


    if (SwrVal>= LimiteSWR)


     {
digitalWrite(ledSWR, HIGH);   //
delay(100);                   //
digitalWrite(ledSWR, LOW);    //
delay(100);
      }
else { digitalWrite(ledSWR, LOW); }

delay(20);
    }
  }
 }

}
```