Thomas HUOT-MARCHAND
Timothée GAUTHIER

Réseaux et télécommunications
1° année

PROJET TRANSMISSION

année 2008/2009

I°) Description du travail effectué

La première partie de ce projet consistait à configurer le circuit d'interface 16550 ou UART 16550 (Universal Asynchronous Receiver Transmitter = transmission série asynchrone). La configuration de ce circuit consiste programmer différents registres de ce composants pour permettre à l'ordinateur de dialoguer avec le GPS ou l logiciel NMEA_C1.exe

Nous avons pris la norme 4800-8-n-1.
_ 4800 bauds
_ 8 bits par caractères
_ « n » = aucune parité
_ 1 => 2 bits de stop

Dans la seconde partie, il fallait décoder les trames émises et afficher des informations compréhensibles pour des humains.

II°) Les difficultés rencontrées:

L'HyperTerminal & DOS

Nous avons perdu du temps à vouloir transmettre des caractères par l'HyperTerminal. Lorsque l'on appuyait sur une touche dans l'HyperTerminal, ce caractère ne se transmettait pas sous DOS, donc nous avons pensé que notre UART était mal configuré.

Suite à de nombreuses modifications sans réussite, nous avons décidé de reprendre notre première configuration, puis lorsque nous avons essayé sous DOS, il s 'est avéré que l'UART était fonctionnel.

Les trames:

Le décodage des trames est la difficulté majeure car il a plusieurs possibilités pour le réaliser.

1°) méthode n°1

Nous avions commencé par les décoder en comptant le nombre de caractères sur les trames fournies dans la documentation. Lorsque nous avons branché le vrai GPS, nous nous sommes aperçus que les trames étaient différentes.

Nous avons aussi remarqué que parfois le nombre de caractères entre 2 virgules n'est pas le même entre le document que nous avons reçu en début de TP et les trames envoyés par le GPS. Par moment, on peut aussi voir qu'il n'y a aucun caractère entre 2 virgules, donc il fallait aussi gérer ce problème pour éviter de se retrouver avec des valeurs incohérentes.

Le dernier problème rencontré avec cette méthode est le fait que quelquefois, au début des trames, au lieu d'avoir par exemple $GPGGA nous avions $$GPGGA. Ce qui faussait toute la suite de la trame. Ce problème a été résolu en changeant de méthode de décodage et aussi en grande partie grâce à la fonction « virgule ».

Nous avons donc changé de méthode.

2°) méthode n°2

Pour finir, le décodage des trames va être fait à l'aide des virgules présentes dans les trames. Pour cela, on a créé une fonction « virgule » qui récupère les caractères entre chaque virgule.
Si il n'y a aucun caractère, donc 2 virgules à la suite, ou si la trame est mal reçue,la fonction retourne 0.

Autre problème dans les trames.

Dans les trames, il y a des « 0 » qui ne servent à rien dans les données par exemple 005.
Le problème est donc d'arriver à enlever ces caractères inutiles. Pour cela, on regardait la trame caractère par caractère et si les premiers étaient des « 0 », on les enlevaient et on affichait le caractère suivant. Puis nous avons trouvé la fonction atof(string) qui retourne la valeur en réel d'une chaine de caractères.

Autre:

Pour plus de simplicité dans la lecture des programmes, nous programmions sous un autre ordinateur tournant sur un autre OS. Donc il fallait à chaque fois enregistrer le programme sur une disquette pour pouvoir l'importer sous DOS et rajouter les librairies dos.h et conio.h.

Au cour du projet, nous avons décidé de fragmenter le programme en plusieurs parties. Donc chaque trame décodée se trouve dans une fonction à part qui est appelée à la manière d'une librairie en début du programme principal.

L'utilisation de ces fonctions nécessaires au décodage des trames doivent être appelé en début de programme. Nous pensions qu'il suffisait de les appeler comme une librairie normale avec : < >. Mais cela ne fonctionnait pas, il faut les appeler avec des guillemets.

Nous avons remarqué qu'il faut que le programme principal et les librairies que nous avons créé soient dans le même dossier pour que le programme les prennes en compte.
Il nous est aussi impossible d'exécuter notre programme avec toutes ses librairies lorsqu'il se trouve sur la disquette. Nous sommes obligé de copier la totalité du dossier sur le disque dur pour que cela fonctionne.

Nous avions aussi essayé de configurer l'affichage pour que les trames soient de couleurs différentes. Pour réaliser cela, nous avons expérimenté la commande « textcolor() ».
Ne connaissant pas les paramètres nous avons essayé avec plusieurs nombres.
Parfois la couleur ne changeait pas, l'affichage clignotait ou encore tout le texte était de la même couleur que l'arrière plan et à la fin du programme la couleur ne retournait pas à son état d'origine. Nous avons donc abandonné cette idée.

III°) Choix techniques:

Mettre le programme principal et le décodage des trames dans un seul programme n'est pas très judicieux pour la lisibilité et la compréhension car sous DOS, la coloration syntaxique n'est pas prise en charge.
Donc chaque fonction permettant de décoder une trame en particulier se trouve dans un endroit à part et est appelé en début de programme principal.

Pour éviter de devoir « recréer » la fonction virgule à chaque trame, nous avons décidé de créer une librairie qui est appelé au début du programme principal.

Exemple d'utilisation de la fonction « virgule ».

```
trame = "$GPGGA,,4807.038,N,01131.324,E,1,,0.9,545.4,M,46.9,M, , *42"
virgule(trame, 2, plop);
printf("%s",plop);
```

Son utilisation est très simple: il suffit de l'appeler comme une fonction puis on lui envoie les paramètres que l'on veut:
_ « trame » la trame sélectionnée
_ « 2 » ensuite le nombre de virgule où le programme doit commencer à stocker les caractères dans un tableau.
_ « plop » nom de la variable dans laquelle on stocke le tableau de caractères.

Dans cette exemple, il s'affichera donc à l'écran le nombre 4807.038 et retournera 1 car il n'y a pas d'erreur, 0 si il n'y a aucun caractères entre deux virgules.

Pour temporiser l'affichage des trames pour plus de lisibilité, nous avons décidé de réceptionner n trames sans les afficher plutôt que de se servir de la fonction delay() qui parfois n'affichait qu'un seul type de trame.

On peut aussi choisir de ne visualiser qu'un seul type de trames parmi les 5 décodées.

ANNEXE

GPS_TT.C
VIRGULE.H
VTG.H
RMC.H
GSV.H
GGA.H
GSA.H

GPS_TT.C

```
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "VIRGULE.H"
#include "VTG.H"
#include "RMC.H"
#include "GSV.H"
#include "GGA.H"
#include "GSA.H"
#define ADBASE 0x3F8
void init(){
    outportb(ADBASE+3,0x82); // LCR <- pour mettre DLAB à 1  -> LCR 10000111
-> 87h
    outportb(ADBASE,0x18);   // DLL -> bit de poids faible de la vitesse
115200/4800 = 24 = 18h
    outportb(ADBASE+1,0x00); // DLM -> bit de poids fort de la vitesse
    outportb(ADBASE+3,0x07); // on remet le DLAB à 0
    outportb(ADBASE+1,0x00); // ier a 0 scrutation
}
void void_trame(char *trame2){
    int mask = 1 ;
    int bit0 = 0;
    int i=0;
    int j;
    int debut=0;        //sert a marquer le $
    char recu;
    while(recu !='\n'){      //recevoir 1 trame de $ a \n
        while(bit0 == 0){
            bit0 = inportb(ADBASE+5) & mask;
        }

        recu = inportb(ADBASE) ;
        if (recu=='$'){
            debut = 1;
        }
        if (debut == 1){
            bit0 = inportb(ADBASE+5) & mask; //on "vide" bit0
            trame2[i]=recu;
            i++;
        }
        if (debut == 0){
            recu='\0';
        }
    }
    i--;
    if(trame2[1]=='$'){          //ce if sert a supprimer le $ en trop
parfois recu
        for (j=0;j<i-1;j++){
            trame2[j]=trame2[j+1];
        }
    }
}
```

```
int quel_trame(char *trame2){
    char QuelTrame[6];
    int selection ;
    virgule(trame2,0,QuelTrame);
    selection=0; // On le remet a 0 pour si les caracteres ne sont pas bons
les erreurs
    if (strcmp(QuelTrame, "GPGGA") == 0) { selection=1 ;} // Si strcmp
renvoie 0 (chaines identiques)
//  if (strcmp(QuelTrame, "GPGLL") == 0) { selection=x ;} //GLL n'apparait
jamais.
    if (strcmp(QuelTrame, "GPGSA") == 0) { selection=2 ;}
    if (strcmp(QuelTrame, "GPGSV") == 0) { selection=3 ;}
    if (strcmp(QuelTrame, "GPVTG") == 0) { selection=4 ;}
    if (strcmp(QuelTrame, "GPRMC") == 0) { selection=5 ;}
    return selection;
}
void sel_case(int selection, char *trame){
    switch (selection)
    {
        case 1:
        GGA(trame);
        break;
        case 2:
        GSA(trame);
        break;
        case 3:
        GSV(trame);
        break;
        case 4:
        VTG(trame);
        break;
        case 5:
        RMC(trame);
        break;
        default:
        printf("\nERREUR dans la trame :\n%s\n", trame); //si erreur on
affiche la trame qui pose probleme
        break;
    }
}
int main(){
    char trame[90];
    int selection=0 ; //determinera la trame
    int i=1; //boucle pour nbr de trame

    int j; // selection trame à afficher
    int k=-1; //pour temporiser
    int l ; //pour boucle de temporisation

    int temp[5] = {0,0,0,0,0}; //savoir quelle trame est sortie
    int count_temp=0;           //nbr de trame sortie

    init();

    printf("\n\nProgramme de decodage des trames NMEA\n");
```

```
    printf("Par le port serie configure : 4800-8-n-1\n\n");
    printf("Quelles trames voulez vous afficher ?");
    printf("\n1 : GGA\t\t2 : GSA\n3 : GSV\t\t4 : VTG\n5 : RMC\t\t6 :
Toutes\n\n");
    while(j<1 || j > 6){
        printf("Faites votre choix : ");
        scanf("%d",&j);
    }
    printf("\nAfin de temporiser l'affichage, entrez le nombre de trame a
laisser passer, conseille : 30");
    while(k<0){
        printf("Faites votre choix : ");
        scanf("%d",&k);
    }
    trame[0]='\0'; //initialisation
    while(i){ //Analyse x trames, on peut mettre une boucle infinie, et
quitter avec ctrl+C
        void_trame(trame);  //"telecharge la trame"
        selection = quel_trame(trame); //analyse de quelle trame il s'agit
GGA VGT...
/*******************************************///si 1 seule trame choisie au
départ
        if (selection==j){
            sel_case(j, trame);           //En fonction de la selection de
l'utilisateur on affiche la ou les trames
            for(l=0;l<k;l++){
                void_trame(trame);          //on temporise
            }
        }
/*****************************************/
/*****************************************/ //Si toutes trames choisies:
        else if (j==6){                 //toutes les trames
            if (temp[selection-1]==0){         //vérifie si la trame est déjà
sortie
                sel_case(selection, trame);
                temp[selection-1]=1;             //1 = trame déjà lu
                count_temp=count_temp+1;

                for(l=0;l<k;l++){                 //on temporise
                    void_trame(trame);
                }
            }
            if (count_temp>=5){
                for(l=0;l<=4;l++){
                    temp[l]=0;
                }
                count_temp=0;
            }
        }
/*****************************************/
// en cas de probleme
        else if (selection==0){
            printf("Erreur : %s",trame);
        }
    }
    return 0;
}
```

VIRGULE.H

```
char virgule(char *trame2, int nombre, char *tableau){
    unsigned int i;
    int p=0;
    int j=0;
    char k=0; // <- si probleme de trame ou de virgule

    *(tableau+j)='\0';
    for(i=0;i<strlen(trame2);i++){
        if (*(trame2+i)==','){
            p++;
        }

        if (p==nombre){
            i++; //on avance de la virgule
            if (*(trame2+i)==','){ //si apres la virgule on a une virgule
                k=0;              //rien ne sera marqué
                *(tableau+j)='\0';
                break;
            }
            while(*(trame2+i)!=',' && i<strlen(trame2)){
                    *(tableau+j)=*(trame2+i);
                    i++;
                    j++;
            }
            k=1;
            *(tableau+j)='\0';
            break;
        }
    }
    return k; // <- si k= 1 then tableau en %s marqué qqchose, si k=0 -> rien
de marqué
}
```

VTG.H

```
void VTG(char *trame2){
    char plop[20];
    float i ;
    printf("\n*** TRAME VTG ***\n");
    if (virgule(trame2, 1, plop)==0){
        printf("\nAucun mouvement detecte\n");
    }
    else {
        virgule(trame2, 1, plop);
        i = atof(plop);
        printf("\nLe cap reel est de %.1f deges",i);
        if(virgule(trame2, 3, plop)!=0){
            i=(plop[0]-48)*100+(plop[1]-48)*10+(plop[2]-48)+(plop[4]-48)*0.1;
            printf("\nLe cap vrai magnetique est de %.1f degres",i);
        }

        virgule(trame2, 5, plop);
        i = atof(plop);
        printf("\nLa vitesse du deplacement par rapport au sol est de %.1f
Noeuds",i);
        virgule(trame2, 7, plop);
        i = atof(plop);
        printf("\nLa vitesse de deplacement par rapport au sol est de %.1f
kilometres heure",i);
    }
}
```

RMC.H

```
void RMC(char *trame2){
    char plop[20];
    float i;
    printf("\n*** TRAME RMC ***\n");

    virgule(trame2, 9, plop);
    printf("\nLa date du fix est le
%c%c/%c%c/%c%c",plop[0],plop[1],plop[2],plop[3],plop[4],plop[5]);

    virgule(trame2, 1, plop);
    printf("\nIl est: %c%ch %c%cmin
%c%csec\n",plop[0],plop[1],plop[2],plop[3],plop[4],plop[5]);
    virgule(trame2, 3, plop);
    i=(plop[0]-48)*10+(plop[1]-48);
    printf("\nLa latitude est de %0.f degres ",i);

    i=(plop[2]-48)*10+(plop[3]-48)+(plop[5]-48)*0.1+(plop[6]-48)*0.01;
    printf("%.2f min ",i);

    virgule(trame2, 4, plop);
    if (plop[0]=='N'){ printf("Nord\n"); }
    if (plop[0]=='S'){ printf("Sud\n"); }
    if (plop[0]=='E'){ printf("Est\n"); }
    if (plop[0]=='W'){ printf("Ouest\n"); }
    virgule(trame2, 5, plop);
    i=(plop[0]-48)*100+(plop[1]-48)*10+(plop[2]-48);
    printf("La longitude est de %.0f degres ",i);
    i=(plop[3]-48)*10+(plop[4]-48)+(plop[6]-48)*0.1+(plop[7]-48)*0.01;
    printf("%.2f min ",i);
    virgule(trame2, 6, plop);
    if (plop[0]=='N'){ printf("Nord\n"); }
    if (plop[0]=='S'){ printf("Sud\n"); }
    if (plop[0]=='E'){ printf("Est\n"); }
    if (plop[0]=='W'){ printf("Ouest\n"); }
    virgule(trame2, 7, plop);
    i = atof(plop);
    printf("La vitesse sol est de %.2f Knots soit %.2f Km/h",i, i*1.852);

    if(virgule(trame2, 8, plop)==0){
        printf("\nAucun cap vrai detecte");
    }
    else{
        printf("\nLe cap vrai est de %s",plop);
    }

    if (virgule(trame2, 10, plop)==0){
        printf("\nPas de declinaison magnetique\n");
    }
    else{
        i=(plop[0]-48)*100+(plop[1]-48)*10+(plop[2]-48)+(plop[4]-48)*0.1;
        printf("\nLa declinaison magnetique et de %.1f\n",i);
    }
}
```

GSV.H

```
void GSV(char *trame2){
    // POSSIBILITER DE METTRE DANS UNE BOUCLE POUR NE PAS SE REPETER !!
    char plop[20];
    float i;
    virgule(trame2, 1, plop);
    printf("\n*** TRAME GSV ***\n");
    printf("\nIl y a %s trames GSV avec des donnes completes\n",plop);
    if (virgule(trame2, 3, plop)==0){
        printf("Il n'y a pas de satellites visibles\n"); // si erreur -> pas
de sattelite
    }
    else{
        virgule(trame2, 3, plop);
        i = atof(plop);
        printf("Il y a %.0f satellite visible\n", i);

        virgule(trame2, 4, plop);
        i = atof(plop);
        printf("Le numero d'identification du premier satellite est
%.0f\n",i);
        virgule(trame2, 5, plop);
        i = atof(plop);
        printf("L'elevation est de %0.f degres\n",i);
        virgule(trame2, 6, plop);
        i = atof(plop);
        printf("Son azimuth est %.0f\n",i);
        virgule(trame2, 7, plop);
        if (plop[0]=='*'){
         printf("Il n'y pas d'autre satellites visibles\n");
         //si 1 seul sattelite on quitte car pas d'autre a afficher
        }
        else{
            virgule(trame2, 8, plop);
        i = atof(plop);
            printf("Le numero d'identification du deuxieme satellite est
%.0f\n",i);
            virgule(trame2, 9, plop);
        i = atof(plop);
            printf("L'elevation est de %.0f degres\n",i);
            virgule(trame2, 10, plop);
        i = atof(plop);
            printf("Son azimut est %.0f\n",i);
        }

        if (virgule(trame2, 11, plop)==0){
            printf("Il n'y pas d'autres satellites visibles\n");
        }
        else{
            virgule(trame2, 12, plop);
        i = atof(plop);
            printf("Le numero d'identification du troisieme satellite est
%.0f\n",i);
            virgule(trame2, 13, plop);
        i = atof(plop);
            printf("L'elevation est de %.0f degres\n",i);
```

```
            virgule(trame2, 14, plop);
        i = atof(plop);
            printf("Son azimut est %.0f\n",i);
        }

        if (virgule(trame2, 15, plop)==0){
            printf("Il n'y pas d'autres satellites visibles\n");
        }
        else{
            virgule(trame2, 16, plop);
        i = atof(plop);
            printf("Le numero d'identification du deuxieme satellite est
%.0f\n",i);
            virgule(trame2, 17, plop);
        i = atof(plop);
            printf("L'elevation est de %.0f degres\n",i);
            virgule(trame2, 18, plop);
                    i = atof(plop);
            printf("Son azimut est %.0f\n",i);
        }
    }
}
```

GGA.H

```
void GGA(char *trame2){
    char plop[20];
    float i;

    printf("\n*** TRAME GGA ***\n");
    virgule(trame2, 1, plop);
    printf("\nIl est: %c%ch %c%cmin
%c%c.%csec\n",plop[0],plop[1],plop[2],plop[3],plop[4],plop[5],plop[7]);

    virgule(trame2, 2, plop);
    i=(plop[0]-48)*10+(plop[1]-48);
    printf("\nLa latitude est de %0.f degres ",i);
    i=(plop[2]-48)*10+(plop[3]-48)+(plop[5]-48)*0.1+(plop[6]-48)*0.01+(plop[
]-48)*0.001;
    printf("%.3f' ",i);
    virgule(trame2, 3, plop);
    if (plop[0]=='N'){ printf("Nord\n"); }
    if (plop[0]=='S'){ printf("Sud\n"); }
    if (plop[0]=='E'){ printf("Est\n"); }
    if (plop[0]=='W'){ printf("Ouest\n"); }

    virgule(trame2, 4, plop);
    i=(plop[0]-48)*100+(plop[1]-48)*10+(plop[2]-48);
    printf("La longitude est de %0.f degres ",i);
    i=(plop[3]-48)*10+(plop[4]-48)+(plop[6]-48)*0.1+(plop[7]-48)*0.01+(plop[
]-48)*0.001;
    printf("%.3f' ",i);

    virgule(trame2, 5, plop);
    if (plop[0]=='N'){ printf("Nord\n"); }
    if (plop[0]=='S'){ printf("Sud\n"); }
    if (plop[0]=='E'){ printf("Est\n"); }
    if (plop[0]=='W'){ printf("Ouest\n"); }

    virgule(trame2, 7, plop);
    i = atof(plop);
    printf("Satellites en poursuite : %.0f\n" , i);
    virgule(trame2, 9, plop);
    if (plop[0] != '0') {
        i = atof(plop);
        printf("%s Altitude : %.2f\n",plop,i);
    }
    else{
        printf("Aucune altitude détectée\n\n");
    }
}
```

GSA.H

```
void GSA(char *trame2){
    int i;
    int j=0;
    char plop[20];
    printf("\n*** TRAME GSA ***\n\n");

    virgule(trame2, 2, plop);
    if (plop[0]=='1'){ printf("Information peu fiable car peu de satellites
reçu.\n"); }
    else if (plop[0]=='2'){ printf("Informations fiable.\n"); }
    else if (plop[0]=='3'){ printf("Informations très fiable, suffisament de
satellites.\n"); }

    printf("Id des satellites capte :\n");
    for (i = 1 ;i < 12;i++){
        if(virgule(trame2,i+3, plop) != 0){
            printf("%s, ",plop);
            j++;
            }
    }
    if (j==0){printf("aucun Id reçu !\n\n");}

}
```