SP4abc

Travaux Pratiques

de

Programmation

de

Microcontrôleur

Polytech GE3a - Module SP4abc

J. Laffont – M. James – J. Sérot – S. Lengagne – T. Feraud

v5.1 05/04/21

Séance 1 – SP4a1 - JL
Séance 2 – SP4a2 - JL
Séance 3 – SP4a3 - JL

Séance 4 – SP4b1 - JL
Séance 5 – SP4b2 - JL
Séance 6 – SP4b3 - JL

Séance 7 – SP4c1 - JL
Séance 8 – SP4c2 - JL
Séance 9 – SP4c3 - JL

Durée : 36 H

Pré requis :

- ✔ modules INFO4, SP1, SP2 et SP3.

Disciplines concernées :

- ✔ programmation (langage C),
- ✔ architecture et programmation des systèmes à microprocesseurs / microcontrôleurs.

Objectifs (ce que vous devez savoir faire à la fin du TP) :

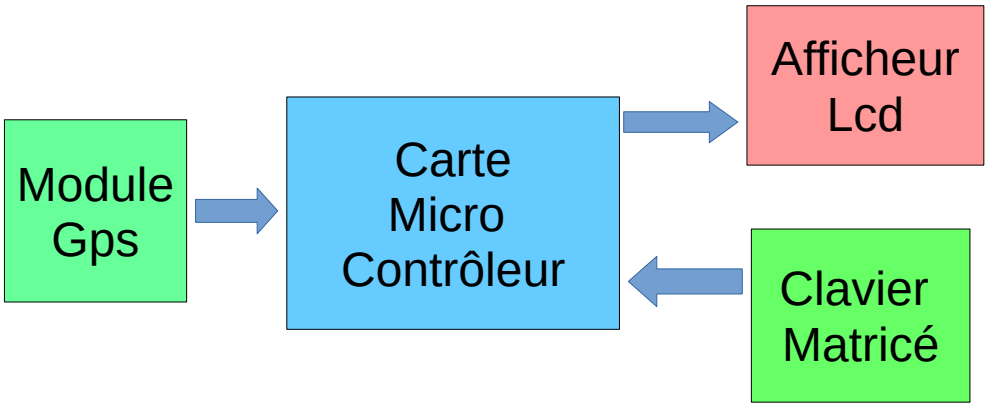
- ✔ savoir découper une application en plusieurs blocs fonctionnels,
- ✔ savoir implanter une application sur une cible de type microcontrôleur en utilisant une chaîne de développement intégrée et les périphériques du microcontrôleur,
- ✔ être capable de valider cette application, en procédant de manière incrémentale.

1 Description du sujet :

L'application étudiée est un système (simplifié) de rappel de proximité de zones dangereuses. Le principe d'un tel système - tout à fait légal, rappelons-le - consiste à calculer en temps réel la position et la vitesse du véhicule, à l'aide d'un G.P.S. typiquement, et à la comparer à une "carte" des zones dangereuses connue à l'avance.

En pratique, on calcule la distance à la plus proche zone et on vérifie, si cette distance est inférieure à un seuil donné, et si la vitesse est inférieure au seuil de vitesse admissible pour cette zone. Si ce n'est pas le cas, on déclenche une alarme. Notons que la détection de zones dangereuses n'est pas la seule application possible d'un tel système qui peut servir, de manière plus générale, à un grand nombre d'applications d'aide à la navigation à partir de positions G.P.S. et/ou de cartographie dynamique.

Description de la plate-forme



Dessin 1: Plate-forme matérielle

Le microcontrôleur est un M32 de Renesas (M32C87) cadencé à 20Mhz. Ce microcontrôleur 16 bits embarque 48 Ko de RAM et 1 Mo de Flash. L'interfaçage se fera via les ports d'entrée sortie et la liaison série asynchrone.

La programmation se fera à l'aide de l'environnement HEW (High Performance Embedded Workshop), qui intègre un compilateur, un simulateur et un simulateur/débugger. L'application sera écrite en langage C.

Le récepteur G.P.S. fournit toutes les secondes des messages (« trames ») contenant des informations telles que la latitude, la longitude ou l'altitude. Ces messages sont formatés d'une manière spécifique ce qui imposera un décodage de la trame avant de pouvoir traiter l'information récupérée.

Exemple de trame :

```
"$GPGLL,4545.6424,N,00306.6036,E,141914.00,A*0E"
```

L'affichage est réalisé à l'aide d'un afficheur LCD à 4 lignes de 16 caractères. L'afficheur donnera en permanence la vitesse du véhicule et la distance à la zone dangereuse la plus proche avec la vitesse limitée associée. En cas de « danger », c.-à-d. lorsque cette distance est inférieure à un seuil donné et que la vitesse est supérieure à la vitesse limite associée à la zone un message d'alarme spécifique sera émis sur l'afficheur de manière répétitive. L'alarme sera aussi signalée par la mise à 1 d'une broche prédéfinie d'un port du microcontrôleur, cette sortie pouvant commander une alarme visuelle ou sonore.

On utilisera un clavier matricé comportant 12 touches. Une touche de ce clavier ('0') servira à remettre la condition d'alarme à zéro. Quatre autres touches permettront à l'utilisateur d'enregistrer des emplacements de zones avec la vitesse adéquate associée (ou d'actualiser la vitesse d'une zone déjà répertoriée). Ces deux éléments constituent l'interface homme-machine (IHM) du système

Dessin 2: Synoptique du clavier et affectation des touches

Pour des raisons pratiques évidentes, la mise au point de l'application ne pourra s'effectuer

en se branchant effectivement sur le G.P.S. d'un véhicule. Cette phase se fera donc en simulant la position du véhicule, ceci à partir de trames d'informations pré-calculées.

En fonctionnement normal, ces trames sont émises à intervalle régulier par le G.P.S..

Dans un premier temps, pour simplifier, on supposera que la carte des zones est fixe (non modifiable). Elle contient, pour chaque zone, sa position (longitude, latitude) et la vitesse à partir de laquelle le signal sonore se déclenche. En cas de détection positive, on se contentera d'afficher un message d'alarme sur l'afficheur.

2 Découpage de l'application et plan de travail

L'application peut-être découpée en trois grands blocs (cf fig. 3) :

- un bloc chargé de la réception des trames G.P.S. sur la liaison série (RecTrame),
- un bloc chargé du traitement et de l'interprétation de ces trames, intégrant en particulier le calcul de la vitesse et de la distance à la plus proche zone (Traitement),
- un bloc chargé de la gestion du clavier et de l'affichage (IHM).

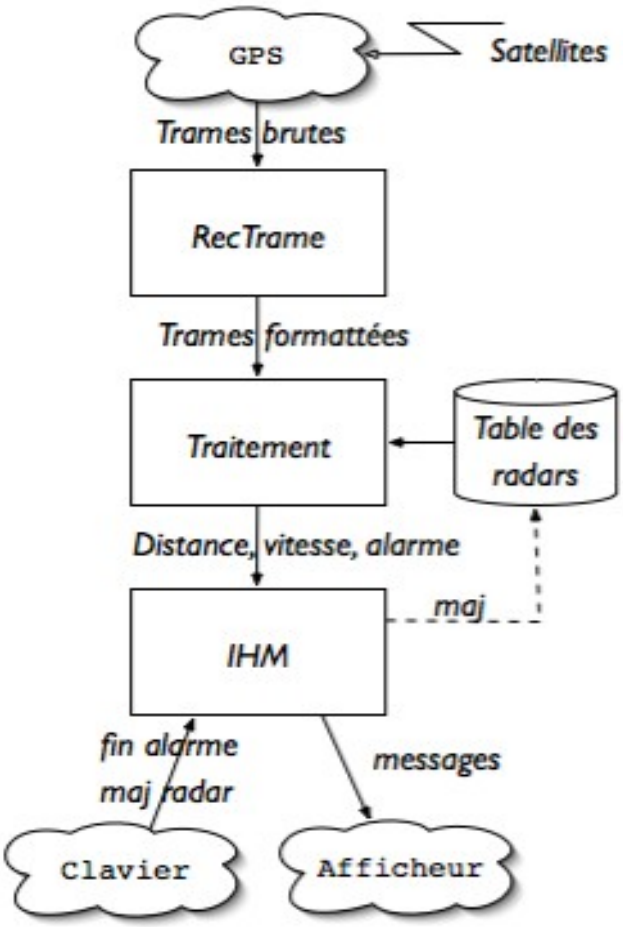


Illustration 1: Synoptique de l'application

Chacun de ces blocs sera étudié et validé séparément. L'ordre à suivre sera le suivant :

1. Étude, implantation et validation des fonctions du bloc Traitement (séances 1,2,3)
2. Étude, implantation et validation des fonctions du bloc IHM (séances 4,5,6)
3. Étude, implantation et validation du bloc RecTrame (séances 7,8,9)

Les séances 6 et 9 seront dédiées à l'intégration finale de l'application, aux tests en situation réelle et, éventuellement, à des améliorations / extensions non décrites dans ce document.

Pour chaque point, la liste et les spécifications des fonctions à écrire seront données ainsi qu'un échéancier (ce qui doit fonctionner à l'issue de chaque séance). Afin de respecter cet échéancier, il sera indispensable de bien préparer les séances de TP, de préparer sur papier les algorithmes et fonctions à implanter, ceci afin de réserver les séances au travail de codage et à la mise au point.

3 Environnement de développement

Pour les 3 premières séances, le travail de programmation n'étant pas spécifique au microcontrôleur cible, le code correspondant sera tout d'abord mis au point dans un environnement de développement C « générique » en utilisant l'IDE CodeBlocks. En fin de série, le portage sous HEW se fera par importation des fichiers sources.

Deux programmes de tests vous seront fournis, un pour les séances 1 et 2, un autre pour la séance 3. Vous travaillerez ensuite sur les projets que vous aurez créés sous HEW.

4 Gestion des programmes :

Les codes sur lesquels vous allez travailler seront versionnés.

Veuillez suivre seul et à votre rythme la procédure suivante pas à pas. La procédure est simple et détaillée, il vous est demandé de la suivre scrupuleusement et intelligemment.

1. Dans la suite, remplacer les XX par l'information pertinente.
2. En opération préalable, veuillez vous connecter à la forge de l'université :
   - https://forge.clermont-universite.fr/
3. S'assurer que vous avez été ajouté au dépôt du TP : Vous devriez apparaître dans la liste des membres :
   - Vérifiez que vous pouvez aller au projet Polytech Ge SP4 20XX
   - https://forge.clermont-universite.fr/projects/tp_sp4_20XX/settings/members
4. Sur le disque « travail » de l'ordinateur de Travaux Pratiques, créer sous la racine un répertoire **tp_sp4_20XX_votre_nom** (sans espace ni caractère accentué).
5. Ouvrez ce répertoire
6. Récupérez le projet type dans ce répertoire, en utilisant le menu contextuel (bouton droit de la souris),

- Svn Checkout → Entrer l'adresse du projet type :
  - L'adresse du dépôt se trouve en haut à gauche de l'onglet dépôt du projet de la forge :

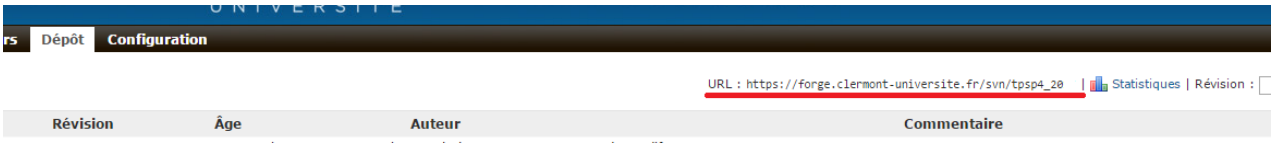


Illustration 2: Adresse du dépot

elle est du type https://forge.clermont-universite.fr/svn/polytech-ge-sp4-20XX

- Copier et coller l'adresse du dépôt dans le champ url of repository
- Utiliser le bouton de parcours (…) pour sélectionner le répertoire **trunk/sp4abc**
- Le répertoire de destination (checkout directory) doit être **tp_sp4_20XX_votre_nom**

- Vérifiez que votre copie de travail ne contient que le projet de la racine (pas de répertoire « trunk », « branch » ou « tags »).

Vous allez développer votre programme dans une branche spécifique et personnelle :

7. Menu contextuel -> sectorise svn → branch/tag
8. Remplacer le champ « to path » avec /branch/votre_nom.
9. Comme message, mettez « création branche votre nom ».
10. Sélectionner « HEAD revision in repository »
11. Sélectionnez « create intermediate folders »
12. Sélectionnez « switch to new branch »
13. « ok ».

Après chaque question, mettez à jour votre programme sur le serveur :

- menu contextuel et SVN Commit,
- Mettez OBLIGATOIREMENT un message correspondant à l'état du programme ou aux améliorations apportées,
- Ne jamais faire de commit sans message explicatif (« réponse question2 », « correction fonction xxx »,…),
- **Le contenu du commit doit absolument être vérifié avant de faire ok :**
  - Le commit a-t-il bien au bon endroit ?
    - bon repository, bonne branche
  - Les fichiers concernés sont-ils les bons ?
    - Vérifiez qu'ils m'appartiennent et que j'ai effectivement modifié
    - Ne jamais commiter des fichiers pouvant être générés par l'outil
      - **Ne jamais commiter de fichiers objets ou d'executables**
      - **Ne jamais commiter de fichiers de backup**
  - Des fichiers sont-ils manquants ?

        - **Les ajouter si nécessaire**
    - **La taille du commit est elle raisonnable ?**
        - Corresponds à une modification unitaire ?
            - Sinon sélectionner SI POSSIBLE un sous-ensemble et commiter en plusieurs étapes
- **Faire une deuxième vérification du contenu du commit**
    - Une fois effectué un commit restera « à vie » dans les journaux
    - Le contenu du commit sera sauvegardé à vie
        - Vérifier la taille des documents commités
    - En cas de doute demander à un enseignant de valider
- Si vous êtes sur de vous, vous pouvez effectuer le commit.

Vous pouvez travailler sur vos programmes depuis un ordinateur personnel :

- L'IDE CodeBlocks est en téléchargement libre sur internet
    - attention : installez la version incluant le compilateur MINGW
- Pour accéder à vos programmes versionnés :
    - Installer le logiciel libre TortoiseSvn
- Pour récupérer une copie de vos programmes sur une machine personnelle
    - Utiliser Svn Checkout
        - Utiliser l'adresse suivante :
            - https://forge.clermont-universite.fr/svn/tpsp4_201X/branch/votre_nom
- Vous pouvez mettre à jour vos programmes après correction sur votre machine :
    - SVN Commit.
- Le logiciel HEW utilisé en deuxième partie de TP. Peut être installé sur vos machines personnelles :
    - En faire la demande en début de TP.

5 Tests unitaires et validation de la fonction

Un Test unitaire est une fonction permettant de valider le fonctionnement élémentaire d'une partie de votre programme. Pour cela la fonction est appelée avec des données connues et la valeur retournée est comparée au résultat théorique.

Toute erreur entraîne une sortie prématurée de l'application avec un code d'erreur non nul (exit(-1);). Les valeurs testées peuvent être exhaustives (tous les cas possibles) ou représentatives :

- valeurs extrêmes,
- valeur nulle,
- valeurs spécifiques,
- valeurs entraînants un traitement particulier,
- valeur ayant historiquement généré un résultat erroné …)

Le dernier point est très important : lorsqu'une erreur est détectée dans une fonction, un test

unitaire doit être ajouté afin de garantir que cette erreur ne sera pas répétée.

ATTENTION : Les tests unitaires ne permettent pas de valider complètement une fonction.

6 Préparations et Manipulations

Un travail préparatoire est à effectuer avant chaque séance. Essayez dans la mesure du possible de faire cette préparation seul, en une seule séance de travail.

Il vous est conseillé de suivre la démarche suivante :

Assurez-vous d'avoir 1H30 de temps libre devant vous, un accès Internet :

- Essayez de répondre à chaque question,
  - justifiez votre réponse,
  - utilisez Internet pour rechercher des éléments de réponse si nécessaire.
    - Ne JAMAIS recopier un élément de réponse si vous le l'avez pas compris, ou bien signalez le très clairement.
- Si vous ne trouvez par la réponse à une question :
  - indiquer l'état d'avancement de votre réflexion,
  - identifiez et décrivez le ou les points bloquants.
- Passer à la question suivant en considérant cette question comme résolue.

Dès la préparation terminée, si vous ne pouvez plus répondre à aucune question et dans tout les cas après 1H30, marquer l'heure de fin et le temps écoulé. Vous avez fini la préparation.

Les préparations sont des documents informatiques non calligraphiés, que vous aurez pris soin d'imprimer avant de début de la séance. Un QCM pourra être proposé en début de séance afin de valider le travail préparatoire.

Si un programme vous est demandé dans la préparation, vous pouvez l'écrire en langage naturel, en utilisant les mots correspondants aux idiomes du langage C (faire tant que, si / alors / sinon, répéter jusqu'à, appeler fonction, renvoyer valeur, créer un tableau, accéder au ième élément du tableau / de la chaîne, et bien sûr les opérations additions, multiplications …)

Chaque étape de programme proposée doit être élémentaire.

Si vous souhaitez proposer le programme directement en langage C, saisissez-le directement sous CodeBlocks et copiez-le sur clef USB pour éviter une double saisie.

7 Convention de code

Les codes que vous allez écrire devront répondre aux conventions suivantes :

1. Les blocs seront correctement indentés par une tabulation :
   - Après chaque `if / else`
   - Après chaque boucle (`while, do, for`)
2. Les fonctions débutent sur la gauche sans indentation,

3. Les variables locales aux fonctions sont alignées sur la gauche sans indentation,
4. Les indentations seront consistantes, c'est-à-dire que les accolades ouvrantes et fermantes doivent se trouver sur la même colonne (ou à défaut au début de la ligne contenant l'accolade ouvrante)
5. L'IDE CodeBlocks vous aide à indenter correctement votre code, veuillez à ne pas trop dégrader le travail fait automatiquement,
6. Les commentaires doivent être utiles et décrire le rôle d'un ensemble de lignes,
    - par exemple pour la ligne suivante :
      `for(i=0;i<10;i++) tab[i]=0 ;`
        - `// Boucle de 0 à 10 avec tab[i] mis à zéro :`
            - **N'est pas un bon commentaire !**
        - `// Initialisation des valeurs des mesures de la vitesse à 0 :`
            - **Est correct.**
7. Un commentaire n'est pas une paraphrase du C, mais une explication de rôle des lignes.

8 Décodage de la trame G.P.S.

Les trames fournies par le bloc RecTrame seront de simples chaînes de caractères. Les latitudes et longitudes sont codées en sexagésimal. L'angle est codé en degré minute (de 0 à 60) avec une précision de 1/10000 de minute. Le format est décrit à l'annexe 2.

L'annexe 3 donne une séquence de trames. Les trames qui nous intéressent sont celles qui commencent par le mot-clé $GPGGA. Dans ces trames la longitude et la latitude apparaissent dans les 3e, 4e, 5e et 6e champs (les champs étant séparés par des virgules) :

- les 3e et 4e champs donnent la latitude; par ex. : « 3723.2475,N » signifie 37°23,2475' de latitude Nord (c.-à-d. 37° et 23,2475 minutes – on rappelle qu'il y a 60 minutes dans 1°), ou encore +37,387458°), en décimal
- les 5e et 6e champs donnent la longitude (ex : « 00306.6036,E» signifie 3°06,6036' de longitude Est, soit 3,11006°)
- Les autres champs peuvent être ignorés.

Les positions seront codées avec le type suivant

```
typedef struct {
   float latitude;
   float longitude;
} Position ;
```

La fonction `int decode_trame(char *trame, Position *p)` prend en argument la chaîne de caractères contenant la dernière trame reçue et renvoi :

- Comme valeur de retour : un entier indiquant si la trame est valide (c.-à-d. s'il s'agit bien d'une trame de type « GPGGA » et si elle a pu être interprétée correctement),
  - 1 → Trame Valide et position valide
  - 0 → Trame incomplète.
- Dans la structure p : la position indiquée par cette trame (ou une valeur indéterminée).

Indication : procéder en itérant explicitement sur la trame à la recherche des caractères « , » - qui jouent le rôle de séparateur de champs – en stockant chaque champ dans un tampon puis en analysant ce tampon. Attention à la conversion sexagésimale -> décimale pour la latitude et la longitude…

9 Préparation : Séance N°a1

- Noter l'heure de début de préparation.

1. Que vaut en décimal le code ASCII du caractère '0' ? Du caractère '5' ?
2. Déterminer une manière simple de passer du code ASCII d'un chiffre de '0' à '9' à une valeur de 0 à 9.
3. Proposer le code en C ou en langage naturel d'une fonction `int decode_int(char c)` qui renvoie la valeur décimale associée à un caractère donné en paramètre.
   - `decode_int('0')` → renvoie `0`,
   - `decode_int('5')` → renvoie `5`
   - `decode_int('A')` → renvoie `-1`

4. Comment est spécifiée la fin d'une chaîne de caractères ?
5. Combien la chaine `char test[]= «BONJOUR»` contient-elle de caractères ?, Quelle est l'occupation en octets de la chaine test en mémoire ?
6. Proposer le code en C ou en langage naturel d'une fonction `int trame_cmp(char * trame, char * type)` qui renvoie 1 si la trame commence par la chaîne de caractère `type` et zéro dans les autres cas. Pour des raisons pédagogiques l'utilisation de la bibliothèque string, des fonctions strcmp ou autres fonctions équivalentes, est interdite.
   - `trame_cmp(« $GPGGA suite chaine », « GPGGA ») → retourne 1`
   - `trame_cmp(« $GPRMC suite chaine », « GPGGA ») → retourne 0`
   - `trame_cmp(« $GPRMC... », « GPRMC » → retourne 1`
   - `trame_cmp(« $APRMC... », « GPGGA » → retourne 0`

7. Proposer une suite d'opérations permettant de traduire une latitude sexagésimale en valeur réelle.
   - Pour cela vous n'utiliserez que les fonctions :
     - addition, soustraction, multiplication, division entière, modulo.

8. Valider, en détaillant les calculs étape par étape, l'algorithme ci-dessus avec cette valeur : la chaîne d'entrée étant« 3723.2475 » -> la valeur obtenue doit être 37,387458 (en flottant).
9. Proposer le code en C ou en langage naturel d'une fonction `int decode_nombre(char * ch,int n)` qui renvoie la valeur décimale des n premiers caractères de la chaîne `ch` en utilisant `decode_int()`. Il est interdit d'utiliser les fonction `atoi`, `atof` et `sscanf` pour cette fonction.
   - decode_nombre(« 7541 »,2) -> 75
   - decode_nombre(« 7541 »,3) -> 754

10. Quel le résultat devrait être renvoyé par les appels suivants :
    - decode_nombre(« 123 »,3), decode_nombre(« 987654321 »,2)

- Noter l'heure de fin de préparation.

10 Manipulations: Séance N°a1

Les essais seront faits sur des trames de tests. Ces trames sont dans un fichier gps.log (cf annexe 3). Quelques trames complémentaires sont définies dans un tableau dans le corps programme `main.c`. Vous pouvez modifier et ajouter des trames dans ce tableau si nécessaire.

En aucun cas vous ne devez modifier la fonction `main()` du programme ou le contenu des fichiers `trames.c / trames.h`.

Dans l'intégralité des programmes, seul le code des fonctions `traitement()` et `tests_unitaires()` seront modifiés. D'autres fonctions peuvent être ajoutées, mais elles ne seront appelées qu'à partir de `traitement()` et `tests_unitaires()`

Votre programme de validation doit alors ressembler à cela :

```
void ma_fonction(...){
...
}
void decode_trame(char * tr ...){
//implanter une partie de votre code ICI.
   Appel de ma_fonction() ;
}
void traitement (char * trame){
   //implanter une partie de votre code ICI par exemple :
   printf («%s \n »,trame) ;
   decode_trame(trame,...) ;
   printf …
}
```

1. Réaliser les opérations détaillées dans le chapitre 4.
   - Ouvrez votre projet `sp4a12\sp4a.cbp` avec `CodeBlocks`,
   - Vérifiez que le projet fourni compile et s'exécute sans erreurs (`Menu Build → Build`)
   - Vérifier que le projet fourni affiche les trames de test (`Menu Build → Run`)
2. Déclarer une variable statique dans la fonction traitement, et l'incrémenter à chaque appel.
   - `static int cpt=0 ;`
   - `cpt++ ;`
3. Placer un point d'arrêt sur la ligne correspondant à l'incrémentation de la fonction traitement (Placer le curseur sur la ligne, `Menu Debug, Toggle BreakPoint`),
   - Executer le programme dans le débugger (`Menu Debug, Start / Continue`),
   - Afficher le contenu de la variable trame (Selectionner la variable cpt, Menu contextuel, `Watch 'cpt')`
   - Exécuter la ligne (touche f7)
   - Vérifier que la variable a bien été incrémentée.

Déposer la nouvelle version de votre code sur le dépôt.

- Dans le répertoire contenant le projet,
  - le fichier main apparaît avec une icône rouge (il a été modifié).
- Envoyer le nouveau fichier sur le serveur :
  - menu contextuel (bouton droit de la souris)

- SVN commit,
- ajouter un commentaire (impératif).

À partir de maintenant et jusqu'à la fin du TP, pensez à déposer votre code après chaque question.

4. Implanter la fonction `trame_cmp()` (cf préparation).

   En utilisant les informations données sur les tests unitaires (cf chapitres 5), et l'exemple de contenu des tests unitaires correspondant à la fonction trame_cmp() donnée ci-dessous :

```
void tests_unitaires(void){
    if (5!=5){
        printf ("Erreur Test unitaire basique.\n");
        exit(-1);
    }
    if (trame_cmp("$GPGGA suite chaine","GPGGA")!=1){
        printf ("Erreur Test unitaire trame_cmp.\n");
        exit(-1);
    }
    if (trame_cmp("$GPRMC suite chaine","GPGGA")!=0){
        printf ("Erreur Test unitaire trame_cmp.\n");
        exit(-1);
    }
    if (trame_cmp("$GPRMC... ", "GPRMC" )!=1){
        printf ("Erreur Test unitaire trame_cmp.\n");
        exit(-1);
    }
    if (trame_cmp("$APRMC...", "GPGGA")!=0){
        printf ("Erreur Test unitaire trame_cmp.\n");
        exit(-1);
    }
}
```

5. Ajouter les tests unitaires de `trame_cmp()` à la fonction `tests_unitaires() et` vérifier le fonctionnement de votre fonction `trame_cmp()`
6. modifier votre programme pour qu'il n'affiche que les trames `GPGGA`
7. Implanter la fonction `decode_int(),` écrire un test unitaire `test_decode_int(void)` de `decode_int()` fonction qui teste **exhaustivement** votre fonction `decode_int()`.
8. Implanter la fonction `decode_nombre()`. Proposer un test unitaire `test_decode_nombre()` de `decode_nombre()`.
9. Implanter une fonction convertissant la chaîne de caractère contenant la latitude en nombre flottant. Tester la fonction en créant un test_unitaire adéquat.
10. Implanter une fonction convertissant la chaîne de caractère contenant la longitude en nombre flottant et son test unitaire.
11. Unifier les deux fonctions précédentes dans une seule fonction générique. Qui décode indifféremment une latitude ou une longitude. Tester cette fonction avec quatre valeurs différentes (2 latitudes et 2 longitudes).

11 Étude du bloc Traitement

Le bloc Traitement se charge :

1. pour chaque nouvelle trame reçue,
   - de décoder cette trame, c.-à-d.. de s'assurer que :
     - que la trame possède bien le bon format,
       - si oui, d'extraire de cette trame (simple chaîne de caractères), l'information de position (longitude et latitude),
       - Sinon la trame est ignorée.
2. de tenir à jour la vitesse du véhicule (à partir de ses positions successives),
3. de calculer la distance à la zone la plus proche et, si cette distance est inférieure à un seuil et que la vitesse du véhicule est supérieure à celle définie pour cette zone, de déclencher une alarme.

Dans l'application finale, la trame sera fournie par le bloc `RecTrame` dans une variable globale (un booléen `TrameOK` indiquant la disponibilité d'une nouvelle trame – cf section 34). L'affichage de la vitesse et la signalisation de l'alarme seront faits en appelant des fonctions de bloc IHM (cf section 28). L'algorithme général de la fonction principale du bloc Traitement sera alors le suivant :

```
Faire en boucle
  Si TrameOK = 1    // Trame disponible
    t := Trame;
    TrameOk := 0;   // Acquittement
    trame_valide := decode_trame(t,pos);
    Si trame_valide
      v := calcul_vitesse(Pos,Pos_prec);
      d,vmax := distance_a_la_plus_proche_zone(Pos);
      si d < seuil && v > vmax
        alarme := on;
        afficher_alarme()
      fsi;
    fsi;
    afficher_vitesse_et_pos(v,d);
    lire_clavier();
    si touche_appuyée = FA et alarme=on
      alarme := off
    fsi
  fsi
finBoucle
```

Les blocs `RecTrame` et `IHM` n'étant pas disponibles pour l'instant, le travail va consister, dans cette première série, à écrire les fonctions ou à terminer les fonctions :

- `decode_trame()`,
- `calcul_distance()`, qui calcule la distance (en km) entre deux points repérés par leur latitude et longitude.
- `calcul_vitesse()`, qui calcule la vitesse moyenne (en km/h) entre deux relevés

G.P.S. successifs connaissant la période G.P.S.'échantillonnage du GPS (1 sec ici).

- `distance_a_la_plus_proche_zone()`.

Pour le calcul de la vitesse, vous aurez besoin de connaître le rayon terrestre et le théorème de Pythagore (compte tenu des faibles distances considérées, on assimilera la portion de surface terrestre considérée à un plan).

Calcul de la distance à la plus proche zone est décrit par l'énoncé suivant :

Étant donné un tableau d'entiers t, dont on connaît la taille N, et un entier e, il vous faudra trouver l'algorithme de la fonction qui retourne l'index i de l'élément du tableau t qui minimise | t[i]-e | (Par exemple, si t={1,8,4,12,26} et e=5, l'algorithme retourne l'index de l'élément 4, soit 2).

Il faudra donc écrire une fonction :

```
int distance_a_la_plus_proche_zone(
                                    Position p,
                                    Zone r[],
                                    int nb_zones,
                                    float *d)
```

qui retourne le numéro et la distance de la zone la plus proche d'une position donnée.

Cette fonction prend en argument :

- la position courante p du véhicule,
- une table r de zones prédéfinie (un pointeur sur le premier élément et le nombre d'éléments)

et retourne :

- l'index (dans la table) de la zone la plus proche de la position courante (-1 si la table des zones est vide),
- la distance d à cette zone.

Chaque zone sera décrite par une structure du type :

```
typedef struct {
  Position rpos;
  float vitmax;
  } Zone ;
```

renseignant sa position et la vitesse limite associée.

À ce niveau, la table des zones sera supposée constante et déclarée en variable globale sous la forme :

```
Zone zones[] = {
  {{44.7887762, -3.012}, 50}, /* Descripteur de la première zone */
  {{44.7891220, -3,013}, 70},
   ...
};
```

12 Préparation : Séance N°a2

- Noter l'heure de début de la préparation.
  1. Quel est le rayon de la terre au niveau de l'équateur ?
  2. Quelles sont la précision et la dispersion d'une position donnée par G.P.S. ?
  3. Connaissant la latitude et la longitude de deux points p_1 et p_2 sur une sphère donnez une formule mathématique permettant de calculer la distance entre ces deux points.
  4. En utilisant les positions GPS de 4 villes de France, trouvez les distances à vol d'oiseau les qui les séparent.
  5. Sachant que les trames G.P.S. sont espacées d'une seconde, donner la vitesse en km/h correspondant à deux positions successives.

À partir des deux trames suivantes :

```
$GPGGA,141914.00,4545.0000,N,00306.6036,E,1,05,3.4,499.3,M,,M,,*7D
$GPGGA,141915.00,4545.0242,N,00306.6039,E,1,05,3.4,499.5,M,,M,,*72
```

  6. Donner la distance séparant ces deux trames et la vitesse correspondante.
  7. La précision des données fournies par le G.P.S. étant de 1/1000 de minute d'angle. Quelle est la précision d'une géolocalisation sous nos latitudes ? à l'équateur ?
  8. Proposer en langage naturel ou en pseudo-code le code de la fonction `decode_trame()`.
  9. Proposer le code de la fonction `calcule_vitesse()`.
  10. Proposer le code de la fonction `int distance_a_la_proche_zone(position p, Zone r[],int nb_zones, float *d)`
      qui renvoie le numéro de la zone la plus proche du point p ainsi que sa distance.
  11. Existe-t-il une autre solution permettant d'avoir la vitesse du véhicule ?
- Noter l'heure de fin de la préparation.

13 Manipulations: Séance N°a2

1. À partir des fonctions précédentes, écrire et tester la fonction `decode_trame()`.

2. Valider par tests_unitaires sur quatre trames types le fonctionnement de `decode_trame()`.

3. Afficher pour chaque trame GPGGA la position renvoyée par `decode_trame()`.

4. Implanter et valider `float calcule_distance(Position p_1, Position p_2)` en utilisant les résultats de la préparation.

5. Implanter la fonction `float calcule_vitesse(Position p_1, position p_2)` qui renvoie la vitesse correspondant au parcours de la distance entre les points p_1 et p_2 pendant 1s.

Créer une table de zone dangereuse de calquée sur cette définition :

```
Zone zones[] = {
  {{44.7887762, -3.012}, 50}, /* Descripteur de la première zone */
  {{44.7891220, -3,013}, 70},
   ...
};
```

6. Écrire la fonction `int distance_a_la_plus_proche_zone()`.

7. Écrire un test unitaire pour cette fonction.

8. Valider sur les trames de tests votre programme, en affichant après chaque trame reçue l'état de l'alarme (alarme on / alarme off).

9. Indenter et commenter correctement votre programme.

10. Marquer la version définitive de votre programme en créant une étiquette tag/rc_1/votre_nom

14 Amélioration du calcul de la vitesse

Le système G.P.S. fournit des trames contenant des données de position d'un récepteur par la triangulation des signaux reçus depuis la constellation de satellites observable. On se propose ici de créer un estimateur de la vitesse linéaire d'un véhicule embarquant un récepteur G.P.S.. Les données d'entrée sont des trames G.P.S. pré-traitées pour ne fournir que la date et la position (x,y) du récepteur.

Ces données sont cependant bruitées. Il y a structurellement, et historiquement pour des raisons de militaire, une imprécision sur chaque position fournie.

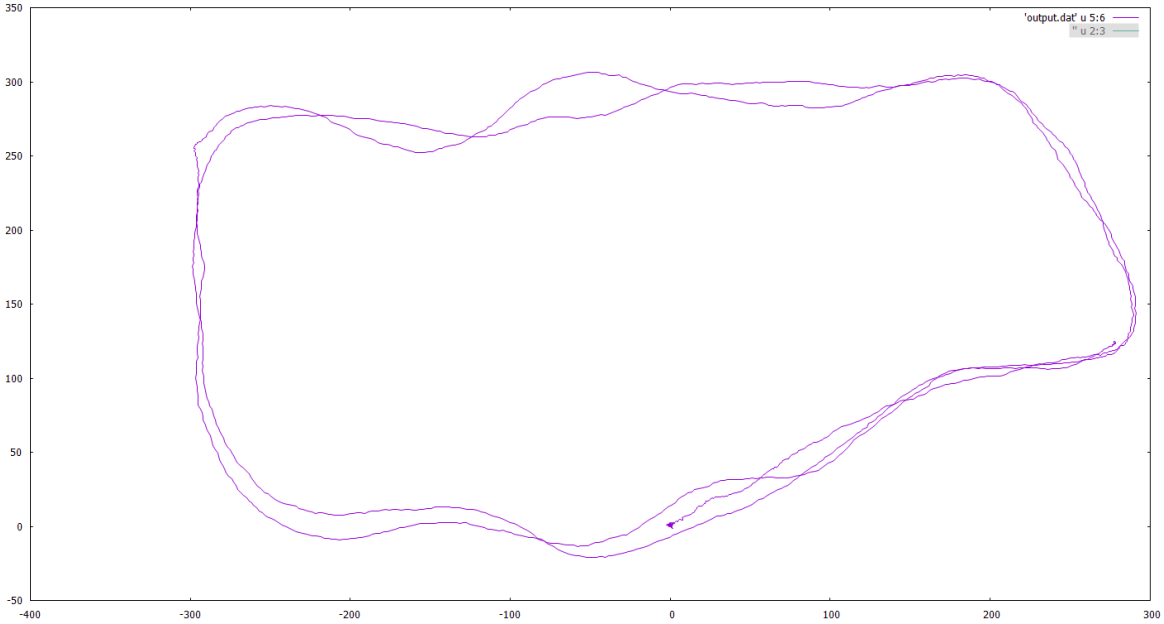


Illustration 3: Trajet réel du véhicule

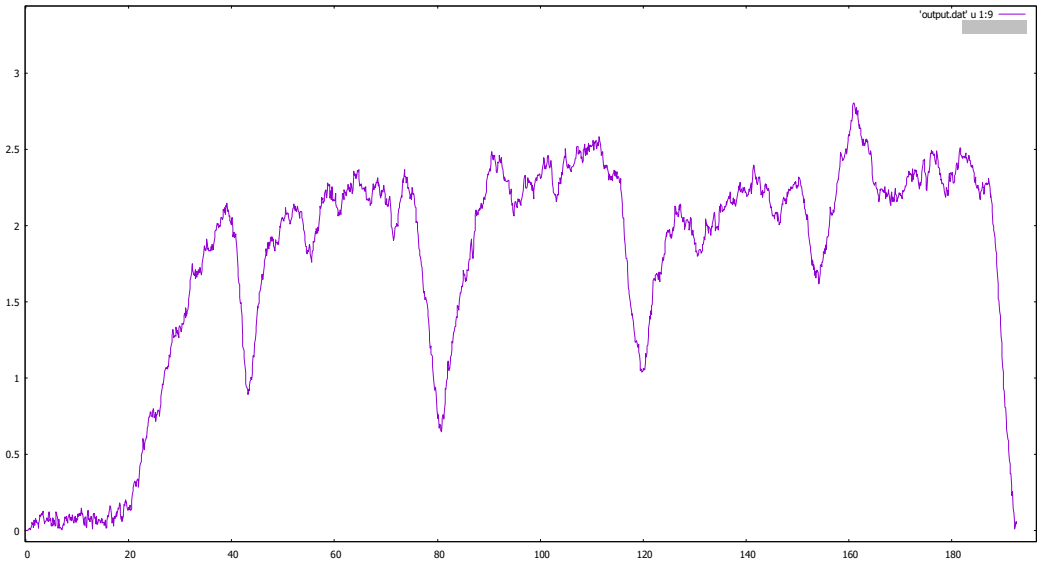


Illustration 4: Profil de vitesse du véhicule

Le trajet qu'il est possible de tracer à partir des points fournis par le G.P.S. est le suivant :

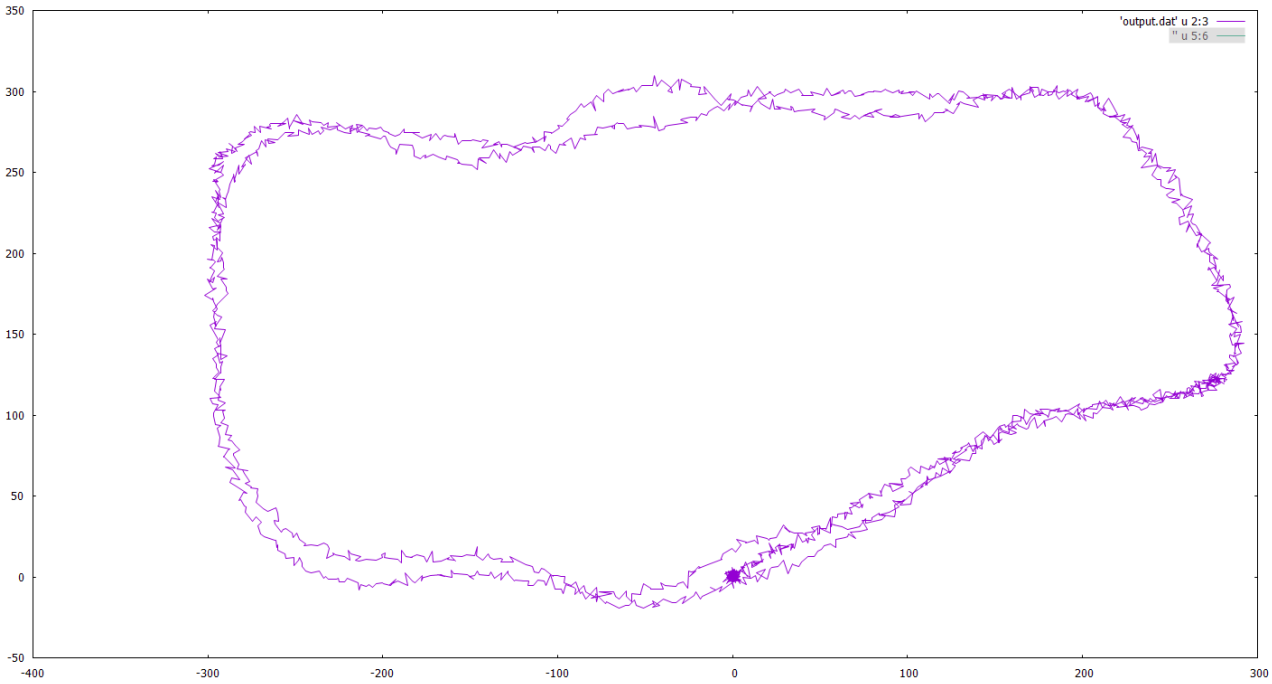


Illustration 5: Relevé G.P.S. du Trajet du véhicule

Le trajet est respecté, cependant l'imprécision due à chaque mesure est évidente. Ceci est dû à dispersion de la mesure fournie par le G.P.S..

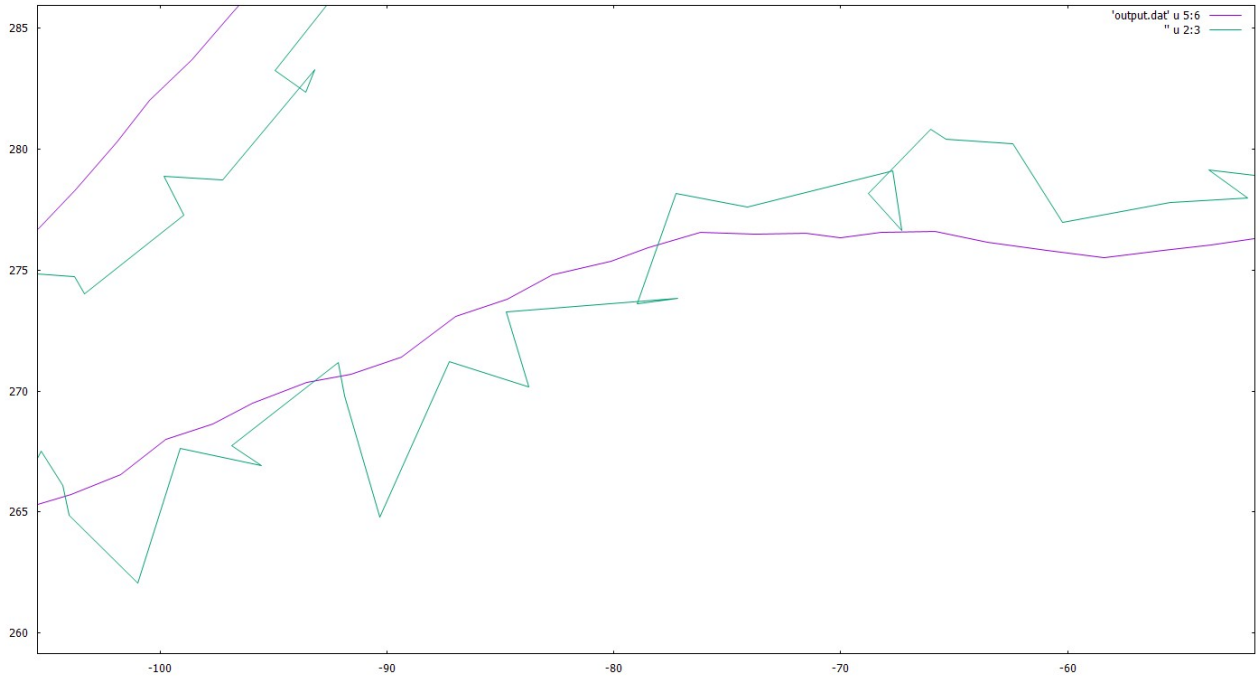


Illustration 6: Détail des positions G.P.S. par rapport aux positions réelles

De manière intuitive, la vitesse se calcule comme la distance parcourue divisée par le temps. Avec nos données, cela donne la distance entre deux poses G.P.S. (bruitées) divisées par le delta temps entre deux trames : $vitesse = \frac{\sqrt{((Xr_k - Xr_{k-1})^2 + (Yr_k - Yr_{k-1})^2)}}{dt}$

La courbe de vitesse alors obtenue est la suivante :

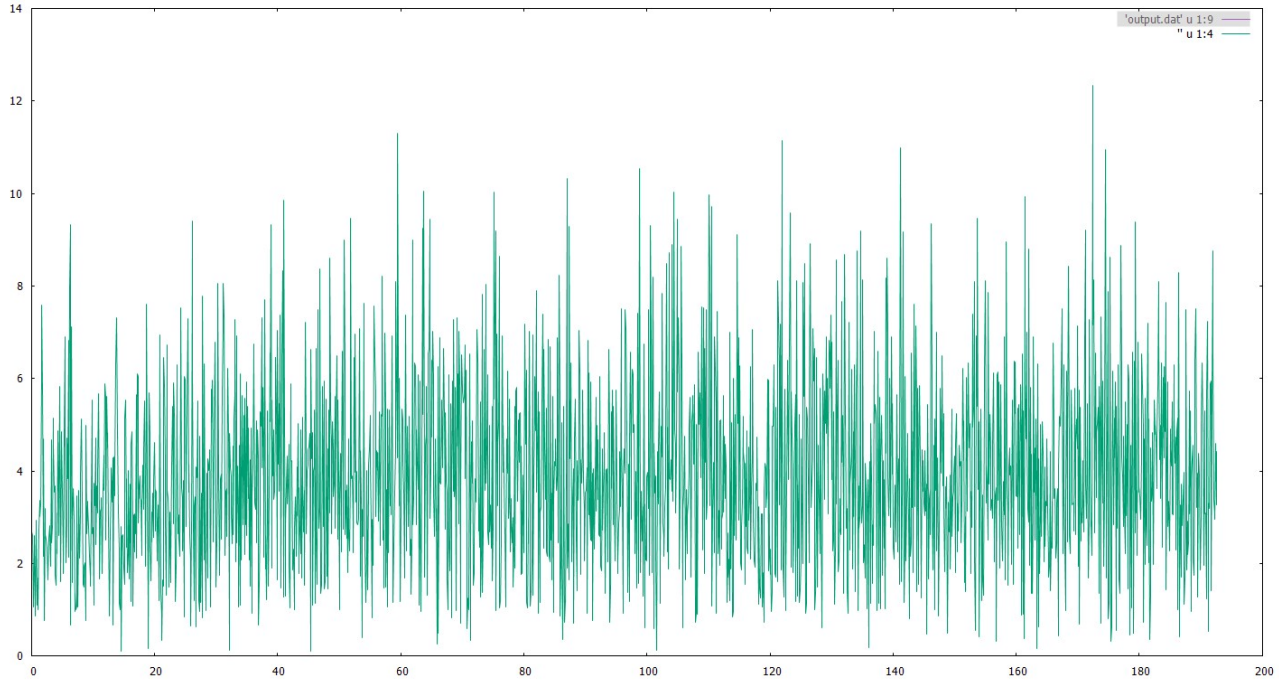


Illustration 7: Courbe de Vitesse sans filtrage

Il apparaît que les vitesses obtenues n'ont rien de réaliste et sont inexploitables en l'état.

Afin d'améliorer l'estimation de la vitesse, on se propose d'utiliser un filtre de Kalman.

Filtre de Kalman

Le filtre de Kalman permet d'estimer l'état d'un système à partir d'observations incomplètes ou bruitées. Il se décompose en 3 étapes :

- Prédiction
- Gain
- Mise à jour

Prédiction : intervient ici un modèle dit d'évolution, car il prédit la position de l'état à l'instant suivant.

Les équations sont simples :

- $X_{k+1|k} = F . X_{k|k}$
- $P_{k+1|k} = F . P_{k|k} . F^T + Q$

La matrice F code le modèle d'évolution tandis que la matrice Q code l'erreur entre le modèle et la réalité. On peut le traduire comme la liberté de mouvement que l'on donne à un

paramètre pour les phases à venir. Q est complexe à calculer et dans notre cas, nous la supposerons simplement diagonale.

Gain : le gain nécessite l'écriture de l'équation d'observation. Il s'agit d'une équation qui lit le vecteur d'état et les observations. Il s'agit d'une matrice, notée H, que nous définirons pour chaque filtre. Le gain se calcule alors ainsi :

- $K = P_{k+1|k} . H^{T} . \left(H . P_{k+1|k} H^{T} + R \right)^{-1}$

Où R est le bruit d'observation que nous définirons pour chaque filtre.

Mise à jour : on vient simplement appliquer les équations pour mettre à jour le vecteur d'état et son incertitude :

- $X_{k+1|k+1} = X_{k+1|k} + K . \Delta$
- $P_{k+1|k+1} = P_{k+1|k} - K . H . P_{k+1|k}$

où Δ est la différence entre l'observation et la projection du vecteur d'état :

- $\Delta = \begin{pmatrix} x_{obs} \\ y_{obs} \end{pmatrix} - H . X_{k+1|k}$

Filtrage de Kalman basé vitesse et position

Le vecteur d'état sera de quatre paramètres à savoir la position et la vitesse sur x et sur y :

$$X = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix}$$

La matrice d'incertitude associée sera une matrice P=[4,4] contenant les variances-covariances des paramètres du vecteur d'état.

Dans ce cas, nous pouvons prédire l'évolution de la position puisque nous avons les vitesses sur chaque coordonnée. On trouve le système d'équations suivant :

$$X = \begin{pmatrix} x_{k+1} = x_k + \dot{x}_k dt \\ y_{k+1} = y_k + \dot{y}_k dt \\ \dot{x}_{k+1} = \dot{x}_k \\ \dot{y}_{k+1} = \dot{y}_k \end{pmatrix}$$

Par conséquent : $F = \begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ sachant que les données G.P.S. sont fournies à 10Hz.

Le bruit d'évolution sera : $Q = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0,1 & 0 \\ 0 & 0 & 0 & 0,1 \end{pmatrix}$ soit une variance de 0.1m/s sur chaque

paramètre vitesse.

Si on considère des données d'état exactes, alors la relation que doit satisfaire H est :

- $Observation = H \,.\, X_{k+1|k}$

On prendra comme observation le vecteur $\begin{pmatrix} x_{gps} \\ y_{gps} \end{pmatrix}$ et on peut écrire le modèle d'observation : $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{gps} \\ y_{gps} \end{pmatrix} = H\,X$ donc $H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$.

Le bruit d'observation est le bruit par lequel nos données ont été bruitées. Par conséquent, les données étant indépendantes, nous allons mettre $R = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}$. La valeur σ étant la variance de l'imprécision de mesure estimée à 2m.

Toutes les variables ayant été définies, on applique simplement le filtre de Kalman pour trouver l'évolution de la vitesse.

Les données utilisées ont été centrées sur le point de départ, le vecteur X peut donc être initialisé à $X_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$, la matrice à $P_0 = \begin{pmatrix} \sigma_e^2 & 0 & 0 & 0 \\ 0 & \sigma_e^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ l'écart type σ_e représente l'incertitude sur l'état qui pourra être de plusieurs mètres (10 par exemple).

L'application itérative du filtre de Kalman permet de filtrer les positions et les vitesses.

Il est alors possible de calculer la vitesse du véhicule comme étant : $vitesse = \sqrt{(\dot{x}_k^2 + \dot{y}_k^2)}$

La courbe ci-dessous représente la vitesse obtenue et la vitesse réelle du véhicule.

Illustration 8: Vitesse réelle et vitesse filtrée

Lors de cette séance, vous allez implanter ce filtre de Kalman.

Les valeurs initiales des matrices ayant été définies, les 5 étapes du calcul ont été présentées. Le travail demandé consiste uniquement à transposer ce calcul en langage C.

La partie de travaux pratiques traitant du filtre de Kalman a été mise en place grâce aux idées, compétences et à l'aide de M. Thomas Feraud ancien étudiant du Génie électrique.

15 Préparation Séance N°a3

Cette préparation va concerner principalement la partie calcul matriciel correspondant à l'implantation du filtre.

En considérant les matrices A et B sous le format suivant :

$$A\begin{pmatrix} A_{11}\,A_{12}\ldots A_{1j} \\ A_{21}\,A_{22}\ldots A_{2j} \\ \ldots\ \ \ldots\ \cdots\ \ldots \\ A_{i1}\,A_{i2}\cdots\ A_{ij} \end{pmatrix} \qquad B=\begin{pmatrix} B_{11}\,B_{12}\ldots B_{1j} \\ B_{21}\,B_{22}\ldots B_{2j} \\ \ldots\ \ \ldots\ \cdots\ \ldots \\ B_{i1}\,B_{i2}\cdots\ B_{ij} \end{pmatrix}$$

1. Donner l'expression des éléments de la matrice $C_{i,j}$ si :
   - C = A+B
   - C= A-B
   - C = A.B (Produit matriciel)
   - C = A' (transposée)
   - C = A^{-1} (inverse) (Cas d'une matrice 2x2 uniquement).
2. Définir à partir de : V_1=votre jour de naissance, V_2=mois de naissance, V_3=siècle de naissance, V_4=année dans siècle de naissance, V_5=heure de la préparation, V_6=minutes de la préparation, V_7=secondes actuelles, V_8=chiffre porte-bonheur; les valeurs pour chacun des éléments de deux matrices $A\begin{pmatrix} V_1 V_3 \\ V_2 V_4 \end{pmatrix}$ et $B\begin{pmatrix} V_5 V_7 \\ V_6 V_8 \end{pmatrix}$ de dimension 2x2. Éviter d'obtenir des matrices singulières (composées d'un grand nombre de zéro, de valeur identique …)
3. Donner les résultats théoriques attendus pour les opérations de 1 sur ces deux matrices.

Vous pouvez vous aider d'un logiciel quelconque (wxMaxima, octave, matlab, scilab, calculatrice ...)

- Par exemple sous wxMaxima vous pouvez :
  - créer les matrices :
    - `a:matrix([13,11],[19,23]);b:matrix([20,30],[17,-5]);`
  - Vous pourrez obtenir les résultats (voir ci-contre:)
    - `a+b;a-b;a.b;transpose(a);invert(a);`

```
(%i22)
(%o18) [33  41]
       [36  18]
(%o19) [-7  -19]
       [ 2   28]
(%o20) [447  335]
       [771  455]
(%o21) [13  19]
       [11  23]
(%o22) [ 23/90   -11/90 ]
       [-19/90    13/90 ]
```

Illustration 9: Résultat Maxima

Les tests unitaires, pour cette partie auront cette forme :

```
void test_unitaires(void){
   //Matrices de tests
   double T22a[2][2]={{13,11},{19,23}};
   double T22b[2][2]={{20,30},{17,-5}};
   //Matrice résultat du calcul
   double R22[2][2];
   //Matrice résultat attendu
   double RAT22[2][2]={{33,41},{36,18}};

   //Opérations
   Add_Mat_Mat(T22a,T22b,R22);
```

```
        //Validation
        if (!Equal_Mat_Mat(RAT22,R22))
              error("Erreur calcul Addition 2x2");
    }
```

4. Écrire sous CodeBlocks le test unitaire complet pour les matrices et les opérations définies.
5. Quelle est la spécificité de l'opération A=A.B, que faudrait-il modifier à voter programme pour que le résultat soit correct.

6. Compléter la table suivante en donnant la dimension de la matrice résultante en fonction de la dimension des matrices d'entrée et de l'opération effectuée.

Opération	Dimension Matrice A	Dimension Matrice B	Dimension Résultat
Transposition	4x4		4x4
Transposition	2x4		4x2
Addition	4x4	4x4	4x4
Addition	2x2	2x2	2x2
Addition	4x1	4x1	4x1
Inversion	2x2		2x2
Soustraction	2x1	2x1	2x1
Soustraction	4x4	4x4	4x4
Multiplication	4x4	4x4	4x4
Multiplication	4x4	4x1	4x1
Multiplication	4x4	4x2	4x2
Multiplication	2x4	4x4	2x4
Multiplication	2x4	4x2	2x2
Multiplication	4x2	2x2	4x2
Multiplication	2x4	4x1	2x1
Multiplication	4x2	2x1	4x1
Multiplication	4x2	2x4	4x4

Les opérations effectuées seront toutes de la forme :

- Pour les opérateurs binaires :
  - Op_Mat_Mat(Mat_A,Mat_B,Res), Op étant Mul, Add, Sub, ...
- Pour les opérateurs unitaires :
  - Op_Mat(Mat_A,Res), Op étant Transpose, Inverse

7. Réécrire le code d'une itération du filtre de Kalman en utilisant uniquement ces opérateurs et des variables intermédiaires si nécessaire.

Si vous souhaitez travailler sur votre machine personnelle, pensez à installer Gnuplot (à partir de la page sourceforge) et pensez à valider l'option ajoutant le programme dans les chemins par défaut (PATH).

16 Détail sur l'implantation du calcul matriciel

La solution retenue dans le projet fourni n'est pas optimale, elle est par contre pédagogique, car elle permet d'utiliser les matrices sans se référer aux pointeurs. L'implantation correcte et optimale d'une matrice se faisant sous forme d'un tableau de pointeur.

La taille de la matrice est passée en argument, les valeurs doivent donc être disponible lors de l'appel à la fonction les prototypes seront donc de la forme :

```
void Mul_Mat_Mat (int lignes_matrice_A, int colones_matrice_A,
   double Matrice_A[lignes_matrice_A][colones_Matrice_A],
   int lignes_matrice_B, int colones_matrice_B,
   double Matrice_B[lignes_matrice_B][colones_Matrice_B],
   double Matrice_Resultat[lignes_matrice_Resultat][colones_matrice_resultat])
```

17 Manipulation Séance N°a3

La séance est dédiée à l'implantation du filtre de Kalman.

Un squelette du programme vous est fourni dans le répertoire `sp4abc/spa3`.

Pour chaque fonction, implanter la fonction de calcul ainsi que son test unitaire, des tests unitaires complémentaires vous sont présents dans le squelette du programme.

Implanter dans l'ordre les fonctions suivantes :

1. Transposée,
2. Addition,
3. Soustraction,
4. Inverse,
5. Multiplication.
6. Lorsque l'ensemble des fonctions est implanté et passe les tests unitaires, implanter la boucle permettant d'effectuer les itérations du filtre de Kalman.
7. Dès que vous obtenez une courbe correcte en sortie de programme, vous avez terminé le TP.

Valeurs des deux premières itérations du filtre :

```
Kalman
-------------0000--------------
F = [4][4]
1.000000 0.000000 0.100000 0.000000
0.000000 1.000000 0.000000 0.100000
0.000000 0.000000 1.000000 0.000000
0.000000 0.000000 0.000000 1.000000
H = [2][4]
1.000000 0.000000 0.000000 0.000000
0.000000 1.000000 0.000000 0.000000
R = [2][2]
4.000000 0.000000
0.000000 4.000000
```

```
-------------0001--------------
 X(k|k) = [4][1]
0.000000
0.000000
0.000000
0.000000

 X(k+1|k) = [4][1]
0.000000
0.000000
0.000000
0.000000

P(k|k) = [4][4]
100.000000 0.000000 0.000000 0.000000
0.000000 100.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000

P(k+1|k) = F.P(k|k).FT + Q = [4][4]
100.000000 0.000000 0.000000 0.000000
0.000000 100.000000 0.000000 0.000000
0.000000 0.000000 0.100000 0.000000
0.000000 0.000000 0.000000 0.100000

P(k+1|k).HT = [4][2]
100.000000 0.000000
0.000000 100.000000
0.000000 0.000000
0.000000 0.000000

H.P(k+1|k).HT + R = [2][2]
104.000000 0.000000
0.000000 104.000000

INV(H.P(k+1|k).HT + R) = [2][2]
0.009615 -0.000000
-0.000000 0.009615

K = [4][2]
0.961538 0.000000
0.000000 0.961538
0.000000 0.000000
0.000000 0.000000

Obs = [2][1]
-0.289421
2.673427

DELTA = Obs - H.X(k+1|k)[2][1]
-0.289421
2.673427

 X(k+1|k+1) = X(k+1|k) + K.Delta = [4][1]
-0.278289
2.570603
0.000000
0.000000

 P(k+1|k+1) = P(k+1|k)-K.H.P(k+1|k) = [4][4]
3.846154 0.000000 0.000000 0.000000
0.000000 3.846154 0.000000 0.000000
0.000000 0.000000 0.100000 0.000000
0.000000 0.000000 0.000000 0.100000
```

```
-------------0002--------------
 X(k|k) = [4][1]
-0.278289
2.570603
0.000000
0.000000

 X(k+1|k) = [4][1]
-0.278289
2.570603
0.000000
0.000000

P(k|k) = [4][4]
3.846154 0.000000 0.000000 0.000000
0.000000 3.846154 0.000000 0.000000
0.000000 0.000000 0.100000 0.000000
0.000000 0.000000 0.000000 0.100000

P(k+1|k) = F.P(k|k).FT + Q = [4][4]
3.847154 0.000000 0.010000 0.000000
0.000000 3.847154 0.000000 0.010000
0.010000 0.000000 0.200000 0.000000
0.000000 0.010000 0.000000 0.200000

P(k+1|k).HT = [4][2]
3.847154 0.000000
0.000000 3.847154
0.010000 0.000000
0.000000 0.010000

H.P(k+1|k).HT + R = [2][2]
7.847154 0.000000
0.000000 7.847154

INV(H.P(k+1|k).HT + R) = [2][2]
0.127435 -0.000000
-0.000000 0.127435

K = [4][2]
0.490261 0.000000
0.000000 0.490261
0.001274 0.000000
0.000000 0.001274

Obs = [2][1]
2.036558
1.506443

DELTA = Obs - H.X(k+1|k)[2][1]
2.314847
-1.064160

 X(k+1|k+1) = X(k+1|k) + K.Delta = [4][1]
0.856590
2.048887
0.002950
-0.001356

 P(k+1|k+1) = P(k+1|k)-K.H.P(k+1|k) = [4][4]
1.961044 0.000000 0.005097 0.000000
0.000000 1.961044 0.000000 0.005097
0.005097 0.000000 0.199987 0.000000
0.000000 0.005097 0.000000 0.199987
```

18 Étude du bloc IHM

Ce bloc est chargé :

- d’afficher la vitesse du véhicule,
- les coordonnées de la zone la plus proche avec la limitation de vitesse associée
- le dépassement de vitesse
- de commander l'enregistrement des coordonnées d'une nouvelle zone
- la suppression d'un enregistrement erroné.

Il s'interface avec le reste du programme comme suit :

- Entrée : vitesse, coordonnées zone la plus proche avec limitation de vitesse associée,
- Sortie : affichage vitesse, coordonnées zone la plus proche et dépassement vitesse commande enregistrement coordonnées nouvelle zone et/ou suppression d'enregistrement erroné.

Le bloc IHM comporte lui même deux parties distinctes : la gestion des touches du clavier matricé et la gestion de l'afficheur alphanumérique à cristaux liquides. Nous consacrerons 4H à la gestion du clavier et 4H à la gestion de l'afficheur, 4H à la mise en commun des fonctions.

Pour gérer le clavier, vous devrez écrire des fonctions qui ont pour rôle :

- le balayage des colonnes,
- la lecture des touches.

Pour piloter l'afficheur, vous devrez écrire des fonctions dont le rôle est :

- L'écriture d'une donnée 4bits
- L'écriture d'une donnée 8bits
- l’écriture d'une commande,
- l’écriture d'un caractère à afficher,
- l’attente de la fin du traitement de la commande ou du caractère précédemment transmis,
- l’initialisation afficheur,
- l’effacement de l'écran,
- l’écriture d'une chaîne de caractère.

Les documents concernant le processeur le clavier et l'écran LCD sont disponibles à cette adresse : https://forge.clermont-universite.fr/documents/756

19 Préparation : Séance N°b1

La documentation du clavier matricé et du microcontrôleur sont téléchargeables en suivant ce lien : `https://forge.clermont-universite.fr/documents/756`

Un clavier matricé 12 touches comporte en général 4 lignes et 3 colonnes, dans notre cas nous connecterons les colonnes aux broches au port 10 bits 7 à 4 et les lignes aux broches au port 10 broches 3 à 0.

1. Faites le schéma correspondant,
2. Expliquez le phénomène de rebond pour les claviers « mécanique ».

La gestion des rebonds des touches de clavier peut être gérée de façon logicielle ou de façon matérielle.

3. Expliquez comment peut-on concrètement gérer un dispositif anti rebond par logiciel et par matériel.

Remarque : sur les sites des fabricants de microcontrôleurs, vous trouverez tous les renseignements nécessaires pour mener à terme ces travaux de préparation en ce qui concerne la gestion de l'interface homme-machine.

4. On vous fournit le programme suivant :

```
#include "sfr32c87.h"

void init_keyboard(void);
void tpo_50ms(void) ;

unsigned char touche = 0;

void main(void){
   init_keyboard()

    while(1){
      if ( (p10 & 0xf0)!= 0xf0 ){
            touche = p10 ;
            tpo_50ms() ;
      }
    }
}
void init_keyboard(void){
    pd10 = 0x0F;p10=0 ;pu31 = 1;
}

void tpo_50ms(void){
      tcspr = 0x8A;
      ta0mr = 0x82;
      ta0 = 50000;
      ta0s = 1;ta0os = 1;
ta0ic = 0x00;
while(ir_ta0ic != 1);
ta0s = 0;
      }
```

5. Quel est le rôle du registre pu3 ?
6. Quelles broches de P10 sont définies en sorties ?
7. Que se passe-t-il lorsqu'une touche est appuyée ?
8. Quel périphérique est configuré par le registre `ta0mr` ?
9. Détaillez la configuration du Timer 0.
10. Avec ce programme, est-il possible de distinguer deux touches sur la même ligne ? Sur la même colonne ?

20 Manipulation séance N°b1 :

La séance va débuter avec la création d'un nouveau projet. Le projet sera créé dans le répertoire sp4b1. Afin de placer ce nouveau projet sur le serveur, il va falloir ajouter les fichiers utiles (et uniquement ceux-là).

1. Procédure création projet :
   - Ouvrir le Logiciel High Performance Workshop(HEW)
   - créer un nouveau projet de type Application ; Workspace name : SP4b1
   - Choisir le répertoire sp4b1 dans votre répertoire de travail.
   - CPU Family : M16C/80 M32C, Toochain : M32C standard
   - CPU series M32C80 87B
   - Desactiver Use Heap Memory
   - dans targets cocher : M32C E8A System et M32C Simulator, puis Finnish
   - Vérifier que le projet compile.
   - Cliquer sur Default sesssion, choisir M32C E8A System
   - MCU Group : M32C/87 Group, Device M30879FL
   - Firmware location : FFF0-*00* et C3-*00*
   - copier le fichier sfr32c87.h qui se trouve dans votre répertoire sp4abc à côté de votre sp4b1.c (dans le sous répertoire sp4b1 du projet.
   - Afficher l'état et agir sur certaines broches : View -> CPU -> IO
2. Après la création du projet, ajoutez (Menu contextuel → Svn → Add) les fichiers suivants :
   - Dans le répertoire principal les fichiers ayant l'extension :
     - .hws
   - Dans le répertoire du projet les fichiers ayant l'extension :
     - .hsf, .ini, .c, .h, .a30, .inc, .hwp.
   - Ne pas ajouter les répertoires debugXXX / releaseXXX
3. Vérifier la liste des fichiers et les placer sur le dépôt `Svn Commit`.
4. Implanter le programme donné en préparation,
5. Placer un point d'arrêt dans le corps du if
6. Vérifier que le programme s'interrompt lorsqu'une touche est appuyée
7. Ajouter la variable touche à une fenêtre de Watch et activer la mise à jour automatiquement
8. Corriger le programme pour que chaque touche génère une valeur spécifique et unique dans la variable touche.
9. Écrire une fonction decode_touche() qui en fonction de la valeur de la variable touche retourne le code ASCII du caractère correspondant à la touche du clavier.

21 Préparations séance N°b2 :

À partir de la documentation de l'écran LCD.

https://forge.clermont-universite.fr/documents/756

Écrivez les algorithmes des différentes fonctions de gestion de l'afficheur. Le Contrôleur qui gère l'écran est un HD44780 du constructeur HITACHI. Vous devrez vous baser sur la documentation de ce circuit en sachant que l'on fonctionnera en mode 4bits.

L'écran est connecté sur le port 3 du microcontrôleur. Les connexions sont les suivantes :

- P3_0 commande le signal D/C,
- P3_1 commande le signal R/W,
- P3_2 commande le signal E,
- P3_4 à P3_7 reliées aux data4_7 de l'écran.

Les macros (#define) LCD_E, LCD_RW, LCD_DC, LCD_E_DIR, LCD_RW_DIR, LCD_DC_DIR, LCD_PORT, LCD_PORT_DIR sont à définir en conséquence.

Les prototypes des fonctions sont les suivants :

```
void tpo_us(unsigned short duree);// temporisation en µs
void tpo_ms(unsigned short duree);// temporisation en ms
void lcd_init_port(void);       // initialisation des broches du µC
void lcd_4b(unsigned char car); // envoi des 4 bits de poids fort
void lcd_8b(unisgned char car);   // envoi des 8 bits de donnée
void lcd_car(unsigned char car);// envoi d'une donnée à afficher
void lcd_com(unsigned char com);// envoi d'une commande
void lcd_init(void);            // initialisation de l'afficheur
void lcd_str(unsigned char *str);     // envoi d'une chaîne de
caractère
```

```
// initialize the LCD display
void lcd_init(void)
{
   LCD_DC=0 ;
   tpo_ms(15);
   lcd_4b(0x30);
   tpo_ms(5);
   lcd_4b(0x30);
   tpo_us(100);
   lcd_4b(0x30);
   tpo_us(100);
   lcd_4b(0x20);
   lcd_com(0x28);
   lcd_com(0x06);
   lcd_com(0x0E);
   lcd_com(0x01);
   tpo_ms(5);
}
```

À l'aide de la documentation du contrôleur de gestion d'écran, commentez ce programme et Expliquez comment est configuré l'afficheur.

22 Manipulations séance N°b2 :

Mettez en œuvre l'afficheur, et affichez les touches appuyées au clavier.

23 Mesure des performances

Les fonctions développées jusqu'à maintenant n'avaient pas de contraintes en terme de temps d'exécution.

Dans cette partie on se propose de mesurer les performances (en terme de temps de calcul) des fonctions étudiées dans les deux premières séances et dans un deuxième temps d'optimiser les temps de calcul.

Le simulateur intégré à l'outil de développement permet de donner le temps d'exécution entre deux points d'arrêts.

Passage en virgule fixe

Les performances modestes de l’application telle qu’écrite jusque là résultent en grande partie de l’usage de nombres codés en virgule flottante (float) pour représenter positions et vitesses.

Afin de pallier ce problème, on se propose d’utiliser un codage en virgule fixe des grandeurs considérées, c.-à-d.., de ne réaliser que des calculs sur des entiers.

Il faudra déterminer quelle taille d'entiers (16 ou 32 bits) faut-il choisir si l'on veut conserver une précision de l'ordre du mètre sur les positions.

24 Préparations séance b3 :

1. Quelle est la dynamique d'un nombre codé sur 16 bits, 24 bits et 32 bits.
2. Quelle est la dynamique d'une latitude fournie par le G.P.S., même question pour la longitude.
3. Quelle est la précision d'une mesure G.P.S. ?
4. En déduire le nombre de bits nécessaire et suffisant pour stocker l'information.
5. Proposer un programme permettant de stocker en virgule fixe dans un entier la latitude et la longitude obtenues à partir de la trame G.P.S.
6. Modifier la fonction obtenue en 5 pour qu'il n'y ait aucun appel à des fonctions flottantes ni l'utilisation de variables flottantes.
7. Écrire une version `_int` des différentes fonctions faisant les calculs uniquement sur des nombres entiers.
8. Trouvez – une simple recherche sur le Web devrait suffire – un algorithme itératif de calcul de la racine carrée ne faisant appel qu'à des additions, multiplications et divisions.
9. Testez cet algorithme sur les données types manipulées par votre programme. Combien faut-il d'itérations au maximum pour obtenir la précision requise ?
10. Définissez un codage – en pratique un facteur d'échelle – pour représenter les positions, les distances et les vitesses sous la forme d'entiers dans votre programme et effectuer les opérations nécessaires sans générer de « débordement ».

25 Manipulations séances N°b3 :

Pour débuter, le code écrit sous CodeBlocks lors de ces deux premières séances sera utilisé tel quel.

1. Implanter l'ensemble des fonctions enveloppées en séances 1 et 2, sur la cible M32.
   Attention :
   - l'accès aux fichiers ne sera alors plus possible.
     - Seule la partie du main utilisant la table pourra être implantée.
   - On ne dispose plus, sous cet environnement, des fonctions printf et scanf.
     - Il faut donc les retirer de votre code. L'inspection des valeurs se fera en plaçant judicieusement des points d'arrêt (break point) et en observant les variables concernées à l'aide de watch point.
2. Corriger votre programme pour qu'il s'exécute sur le microcontrôleur.
3. En utilisant l'onglet de timing (Menu View>CPU>Status, Onglet Platform), calculez le temps (en cycle et en ms) passé par votre programme pour :
   - décoder une trame
   - calculer une nouvelle position et une nouvelle distance
   - trouver la plus proche zone, lorsque la table contient 1, 2, puis 10 zones.
4. Déduisez-en, par extrapolation, le nombre maximum de zones que votre application pourra gérer si les trames arrivent à la cadence d'une par seconde.

Le passage en virgule fixe, peut être fait dans un premier temps en appliquant un facteur d'échelle aux valeurs avant l'appel à la fonction, puis en appliquant un facteur d'échelle inverse sur les valeurs de retour afin de les comparer avec les résultats flottants.

5. Réécrivez ces fonctions avec ce codage en virgule fixe (c.-à-d. en remplaçant le type float par le type long).
6. Vérifiez que les résultats donnés par vos deux programmes (virgule flottante et virgule fixe) sont compatibles (aux erreurs de quantification près).
7. Donner les temps de calcul nécessaires aux différentes fonctions dans leur version virgule fixe.

Fonction :	Test unitaire oui/non	Implantation virgule Fixe oui / non	Durée d'exécution flottante µs	Durée d'exécution virgule fixe µs
trame_cmp()				
decode_int()				
decode_nombre()				
decode_trame()				
calcule_distance()				
calcule_vitesse()				
distance_a_la_proche_zone()				

26 Étude du bloc RecTrame

Le rôle du bloc RecTrame consiste à recevoir les trames en provenance du module G.P.S.

sur la liaison série et à formater ces trames pour le bloc Traitement.

Le G.P.S. fournit les trames sous la forme de séquences de caractères ASCII, au format NMEA 0183 (cf annexe 2). Ces caractères transitent via une liaison série de type RS232.

Du coté G.P.S., la configuration du port série est définie dans la documentation du module (cf annexe 1), soit : `4800 bauds, 8 bits de données, 1 bit de stop, pas de parité, pas de contrôle de flux.`

La liaison est asynchrone.

Votre travail consiste donc à assurer, du côté microcontrôleur, la réception correcte des caractères et la mise en forme des trames.

Le plan de travail sera le suivant :

1. Configuration du port série en fonction des paramètres du G.P.S..
   - Validation de cette configuration en émission.
   - Fonction assurant la réception de caractères.
   - Validation de la chaîne émission/réception.

2. Synchronisation de la réception avec le début d'une trame (caractère ‘$’)
3. Mémorisation des caractères de la trame,
4. Validation des données de la trame (CheckSum),
   - Validation en utilisant des chaînes tests.
5. La libération de ressources du processeur en utilisant les interruptions,
   - Validation avec un jeu de trames définies.

27 Préparation : Séance N°c1

Le travail de préparation est basé sur la lecture et l'analyse de la documentation du processeur M32C87 à rechercher le cas échéant sur le site de Renesas :

- `https://forge.clermont-universite.fr/documents/756`

On s'intéressera plus particulièrement au chapitre sur le port série en mode asynchrone. On travaillera avec la liaison série numéro 0. Afin de configurer correctement le port série il va falloir définir les différentes valeurs à placer dans les registres de configuration de la liaison série.

Veuillez détailler, justifier et donner les références utilisées pour chacune des réponses. (référence du document, page, copie d'écran si possible avec les bits marqués en couleur par exemple).

Noter l'heure de début de préparation.

1. Donner le détail d'une trame d'UART avec un bit de parité paire et deux bits de stops.
2. À quoi correspondent le bit de start et le bit de stop ?
3. Qu'elle est niveau d'un bit de start ?, Quel est le niveau d'un bit de stop ?
4. Rappeler la configuration à utiliser pour le port série (bauds, parité, polarité …),
5. Sur quel port sera branché le G.P.S. ? Quels sont les registres de configuration de ce port ?
6. Dessiner le câblage entre le port du microcontrôleur et les broches 1,2,3,4,5,6 du G.P.S..
7. Quelle sera la configuration du mode d'entrée sortie des différentes broches ? En déduire les valeurs des registres correspondants.
8. Expliquez le rôle des bits du registre U0MR, donnez la valeur à utiliser.
9. Quels sont les autres registres utilisés pour configurer le port série 0 ?
   - Pour chaque bit de ces registres donner la valeur à configurer 0, 1 ou X si indifférent.
10. Quel registre faut-il utiliser pour envoyer un caractère ?
11. Comment peut-on connaître le dernier caractère reçu ?

Le processeur est cadencé par un quartz à 20Mhz.

12. Qu'est-ce qu'un baud ?
13. Quelle formule donne la vitesse de la transmission en Baud de l'uart 0 en mode asynchrone ?
14. En déduire la valeur du registre U0BRG permettant d'obtenir un débit de 4800 Bauds.
15. Le registre U0BRG est initialisé à 32. Quels sont les autres registres à définir en conséquence ?
16. Quelle est l'erreur commise sur le débit ? Proposer une meilleure valeur pour U0BRG.
17. Comment peut-on détecter la fin de l'envoi d'un caractère ?
18. Détailler un protocole permettant de tester l'envoi d'un caractère.
19. Détailler un protocole permettant de tester le fonctionnement de la réception de caractères.
20. Quel protocole (question 18 ou question 19) faut-il mettre en œuvre en premier.

Noter l'heure de fin de préparation.

28 Manipulations: Séance N°c1

Dans le debugger, en activant le monitoring de la RAM (menu contextuel enable Ram Monitor), vous pourrez voir l'évolution de vos variables lors de l'exécution du programme et obtenir des informations très utiles quant au fonctionnement de votre programme.

L'ensemble des fonctions de gestion du port série seront regroupées dans le module C uart0 (fichiers uart0.c et uart0.h) :

- Le fichier uart0.c contiendra les fonctions :
  - `void uart0_init(void) ;`
    - Fonction assurant l'initialisation du port série.
  - `void uart0_tx(char c) ;`
    - Fonction permettant d'envoyer un caractère sur la liaison série.
  - `char uart0_rx(void) ;`
    - Fonction permettant la réception d'un caractère sur le port série
- Le fichier uart0.h contiendra les prototypes de ces fonctions.

1. Créer un nouveau projet dans le répertoire sp4abc_20XX/sp4c12, ajouter les fichiers uart0.c et uart0.h.
2. En vous référant au chapitre 20 point 1, ajouter ce nouveau projet au dépôt du code.
3. En se basant sur votre préparation écrire la fonction qui assure l'initialisation du port série, et dont le prototype est : `void uart0_init (void);`

L'ensemble des registres du processeur est décrit dans un fichier d'entête : sfr32c87.h. Ce fichier est disponible dans le dépôt que vous avez mis en place.

Si vous incluez ce fichier dans votre programme, vous pourrez dès lors utiliser le nom des registres et bits associés en écrivant par exemple :

```
smd0_u0mr = …;
smd1_u0mr = …;
smd2_u0mr = …;
u0brg = …;
…
```

Il n'est pas possible de tester la fonction d'initialisation seule. On va donc essayer d'envoyer un caractère, ceci afin de valider la configuration et la transmission sur le port série 0.

4. Écrire, la fonction qui permet d'envoyer un caractère, et dont le prototype est :
   - `void uart0_tx(char c).`
5. Proposer un programme qui configure le port série et émet en boucle un même caractère.
6. Vérifier le fonctionnement du programme en utilisant un câble série direct et le logiciel Terminal sur le PC, configuré correctement pour vérifier l'arrivée des caractères.
7. Modifier le programme précédent pour qu'il envoie sans interruption une suite de caractère de 'A' à 'Z'. Vous devriez avoir 'ABCDEFGHIJKLMNOPQRSTUVWXYZABCD... » sur l'écran du terminal.
8. Écrire la fonction d'attente et de lecture d'un caractère sur la liaison série, fonction dont le

prototype est : `char uart0_rx()`

9. écrire un programme qui attend l'arrivée d'un caractère sur le port série et renvoie le caractère suivant dans l'alphabet. ('A'→'B' ...)

Le G.P.S. envoie des trames de données en permanence. Le bloc Traitement, lui, ne peut travailler efficacement que sur des trames complètes. On pourra détecter la fin de trame grâce aux caractères <CR><LF> de valeur respective 0x0D et 0x0A. La réception de l'un de ces deux caractères sera interprétée comme la fin de la trame.

Les fonctions spécifiques à la réception de trames seront codées dans des fichiers `rectrame.h/rectrame.c` spécifiques.

10. Écrire la fonction : `int rectrame (char * Buffer)`. Cette fonction copie la trame reçue dans le tampon `buffer`, et signale la bonne réception d'une trame par une valeur de retour non nulle correspondant au nombre de caractères présents dans la trame.

Pour tester ce programme, on utilisera les macros M1 et M2 de terminal. Le programme Terminal peut envoyer des suites de caractères bien définies sur le port série. Ces « macros » de M1 à M12 ont été configurées comme suit :

- `M1 : Trame Gpgga correcte`
- `M2 : Trame Gpgll correcte`
- `M3 : Trame Gpgsv correcte`
- `M4 : Trame Gpgga non synchronisée`
- `M5 : trame Gpgga mal formée`
- `M6 : Trame Gpgga erreur de checksum`
- `M7 à 12 libre pour vos tests.`

11. Écrire le programme principal qui se placera en attente d'une trame, mémorisera les 10 dernières trames correctes dans une table prévue pour contenir 10 trames de 80 caractères.
12. Lors de la réception d'une trame, le programme écrira sur le port série le nombre de caractères de la trame reçue.
13. Vérifier le fonctionnement du programme en utilisant les macros de terminal.

À partir de la macro M4 les trames ne débutent pas sur le début d'une trame. Cette situation peut se produire dans le cas réel. Afin d'assurer un fonctionnement plus stable du système, on ne commencera pas la copie d'une trame dans le `buffer`, que lors de la détection du caractère identifiant le début d'une trame ('$').

14. Modifier la fonction `rectrame` en conséquence.

Chaque trame NMEA est sécurisée par un code de validation (checksum) obtenu en faisant un OU EXCLUSIF cumulé sur l'ensemble des caractères constituant la trame. Le calcul de la somme de validation (checksum) doit se faire au fur et à mesure de la réception de la trame. Le checksum est calculé avec tous les caractères reçus exceptés le caractère '$' initial et le caractère '*'.

15. Modifier la fonction rectrame pour qu'elle filtre toutes les trames mal formées (checksum non valide). Pour cela vous aurez besoin d'écrire une fonction qui transforme un code hexadécimal sur deux caractères en valeur décimale, fonction de prototype :

- `int hex2int (char * c)`

16. Valider votre programme avec l’ensemble des macros de terminal.
17. En fin de séance votre travail sera validé par l’enseignant.

29 Préparation : Séance N°c2

Dans cette séance de manipulation, on va écrire de façon plus lisible la fonction `rectrame()`, et améliorer les performances du programme en utilisant les interruptions.

La fonction `rectrame()` va être implantée sous la forme d'une machine à états. On propose un découpage en 4 états :

- Synchro : corresponds à l'attente du caractère de début de trame.
  - Dès que ce caractère est reçu, on passera dans l'état réception.
- Réception : mémorisation dans le buffer des caractères reçus.
  - Le checksum cumulatif est calculé au fur et à mesure de l'arrivée des caractères.
  - Dès la réception du caractère '*' on passe dans l'état checksum
- Checksum : Dans cet état on lit les 2 caractères correspondant au checksum.
  - Passe ensuite dans l'état validation
- Validation : le checksum de la chaîne reçu et celui récupéré dans l'état checksum sont comparés.
  - S'ils sont identiques, la fonction renvoie le nombre de caractères reçus.
  - Sinon on repasse en état de synchro.

1. Comment fait-on pour coder efficacement une machine à états en langage C ?
2. Dessiner le diagramme de la machine à états et renseigner les conditions de transitions.
3. En déduire les conditions de passage entre les différents états.
4. Quel est le meilleur moment pour initialiser les variables ?*

Afin d'améliorer les performances de la partie réception de trames, il va falloir utiliser les interruptions. Une fonction de réception de caractère sera activée à chaque arrivée de caractère.

5. Quelle est la durée d'un Bit ? Quelle est la durée correspondant à la transmission d'un octet ?
6. Dans votre programme initial, quel est le temps passé dans la routine de réception de trame, dans le cas d'une trame Gpgga ?
7. Dans quelle partie de cette routine le programme reste-t-il le plus longtemps ?
8. Pourquoi la fréquence en Baud est-elle exprimée en 1/16 de la fréquence du périphérique ?
9. Expliquer la notion de vote majoritaire pour la détermination de la valeur d'un Bit.
10. Quelle information permet à coup sûr de réinitialiser la machine à état ?

Pour travailler avec les interruptions, il faut configurer correctement le registre d'interruption, écrire la fonction d'interruption et autoriser les interruptions.

11. Quel est le numéro d'interruption correspondant à la réception d'un caractère sur l'Uart0 ?
12. Définir les valeurs à placer dans les registres adéquats.
13. Quel est le niveau de tension de sortie du G.P.S. ?
14. Ce niveau est-il compatible avec les ports du microcontrôleur ? Les niveaux de tension de sortie du G.P.S. et du microcontrôleur sont-ils strictement compatibles ?
15. Expliquer le rôle du registre PUR2. Pourquoi le registre PUR2 peut-être utile ?

30 Manipulation: Séance N°c2

1. Implanter et valider la nouvelle fonction `rectrame()`.

La routine d'interruption devra assurer la réception du caractère et le stockage dans la trame. Elle remplira donc le rôle des fonctions `uart0_rx()` et `rectrame()`. Le travail se basera sur une copie de la fonction `rectrame()` en `irectrame()`. Cependant :

- la fonction `irectrame()` ne pourra renvoyer de valeur de retour et placera la variable globale `trame_ok` à 1 lors de la réception d'une trame correcte.
- La variable `trame_ok` sera remise à zéro par le programme principal après prise en compte de la trame.
- Tout caractère arrivant tant que `trame_ok` est à 1 sera ignoré.
- La nouvelle fonction `irectrame()` sera automatiquement activée lors de la réception d'un nouveau caractère. Elle ne pourra donc plus contenir de boucle de type while. Il faut donc supprimer cette structure de boucle.

Pour que cette fonction soit reconnue comme une fonction d'interruption il ;faudra rajouter la ligne suivant juste avant la fonction elle-même :

```
#pragma INTERRUPT 18 irectrame
```

Cette ligne indique au compilateur qu'il doit traiter cette fonction comme une interruption (sauvegarde des registres, et instruction de retour spécifique). Elle indique aussi que cette interruption sera accrochée au vecteur 18. Il faut aussi ajouter aux options de l'assembleur (`Build option → Renesas M32 ToolChain → Assembly → Source -> Define`) ajouter la variable `__MVT__` avec la valeur 1. (attention il y a deux undersocre de chaque coté de MVT).

2. Afin de configurer cette interruption, écrire la fonction `uart0_inter_init()`, qui configure les registres afin que l'uart0 génère une interruption lors de l'arrivée d'un caractère.
3. Autoriser les interruptions du périphérique ainsi que les interruptions générales
   - asm(« FSET I »);

La fonction `irectrame()` sera dès lors activée lors de l'arrivée d'un caractère (il n'est plus nécessaire d'avoir une boucle d'attente de l'arrivée de ce caractère).

4. Vérifiez que l'interruption est bien générée en plaçant judicieusement un point d'arrêt.
5. Modifier la fonction `irectrame()`, pour qu'elle assure la synchronisation et la réception d'une trame.

Lors de la réception complète d'une trame, le tampon de réception sera recopié dans le tampon `trame` et la variable globale `trame_ok` sera mise a 1. Afin d'éviter des collisions de trames, aucune autre copie vers le tampon trame ne sera possible tant que `trame_ok` sera à 1.

À l'issue de la deuxième séance, la réception de trames par interruption doit fonctionner et être validée.

La validation se fera en envoyant des trames correctes et erronées à votre programme. Celui-ci devra signaler toute trame correcte et ne sera pas perturbé par l'arrivée de trames erronées. Dans le même temps le programme sera capable d'envoyer un compteur sur la partie émission de la liaison série indiquant la taille en caractères de la trame.

31 Préparation et Manipulation: Séance N°c3

La dernière séance est consacrée à la mise en commun de l'ensemble des programmes écrits durant cette série de Travaux Pratiques.

Vous pouvez aussi profiter de cette séance pour approfondir des points spécifiques de certains TP.

32 Évaluation

L'évaluation sera faite en partie en contrôle continu durant la durée du TP, un examen individuel clôturera la série de TP.

Les préparations :

- Pourront faire l'objet d'une correction par l'enseignant en début de séance, une note sera alors donnée à la préparation,
- Pourront être validée par un QCM à remplir en début de séance.

Votre comportement :

- Respect des consignes données en début de séance,
- Respect du répertoire de travail, de son nom, de sa localisation,
- Respect scrupuleux de la procédure de suivi des programmes, avec en particulier
  - Le dépôt des nouvelles versions après chaque question ou étape importante,
  - L'ajout de commentaire,
  - Le respect des fichiers ajoutés :
    - Pas d'exécutables, pas de fichiers objet ...
- La maîtrise des programmes écrits :
  - Vous serez pénalisé pour toute ligne de code dont vous ne maîtrisez pas le fonctionnement, si vous souhaitez vous inspirer du code d'un collègue :
    - Copiez son code avant la séance
    - Analysez-le, assurez-vous d'en comprendre l'intégralité du fonctionnement,
    - assurez-vous d'être en mesure de répondre à toute question concernant cette partie de programme,
- La qualité des programmes fournis :
  - Indentation, noms des variables …
  - Commentaires.

De l'examen final :

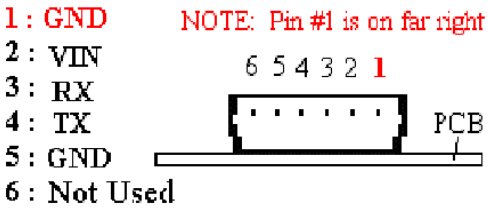
- Cet examen reprendra une partie du TP que vous devrez être en mesure de mettre en œuvre en complète autonomie.

ANNEXE 1

Description du module G.P.S. EM-406

Features:

- SiRF star? high performance G.P.S. Chip Set
- Very high sensitivity (Tracking Sensitivity: -159 dBm)
- Extremely fast TTFF (Time To First Fix) at low signal level
- Support NMEA 0183 data protocol
- Built-in SuperCap to reserve system data for rapid satellite acquisition
- Built-in patch antenna
- LED indicator for GPS fix or not fix
  - LED OFF: Receiver switch off
  - LED ON: No fixed, Signal searching
  - LED Flashing: Position Fixed



Specification:

- **General**
  - Chipset SiRF Star?Frequency L1, 1575.42 MHz
  - C/A code 1.023 MHz chip rate
  - Channels 20 channel all-in-view tracking
  - Sensitivity -159 dBm
- **Accuracy**
  - Position 10 meters, 2D RMS
  - 5 meters, 2D RMS, WAAS enabled
- Velocity 0.1 m/s
- Time 1us synchronized to G.P.S. time
- **Datum**
  - Default WGS-84
- **Acquisition Time**
  - Reacquisition 0.1 sec., average
  - Hot start 1 sec., average
  - Warm start 38 sec., average
  - Cold start 42 sec., average
- **Dynamic Conditions**
  - Altitude 18,000 meters (60,000 feet) max
  - Velocity 515 meters /second (1000 knots) max
  - Acceleration Less than 4g
  - Jerk 20m/sec **3

Power

- Main power input 4.5V ~ 6.5V DC input
- Power consumption 70mA

Protocol

- Electrical level TTL level, Output voltage level: 0V ~ 2.85V
- Baud rate 4,800 bps
- Output message NMEA 0183 GGA, GSA, GSV, RMC, VTG, GLL

Dimension 30mm*30mm*10.5mm
Operating temperature -40? to +85

ANNEXE 2

Format des trames G.P.S.

NMEA Output Command

GGA-Global Positioning System Fixed Data

Table B-2 contains the values for the following example:

$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M,,,,0000*18

Table B-2 GGA Data Format

Name	Example	Units	Description
Message ID	$GPGGA		GGA protocol header
UTC Time	161229.487		hhmmss.sss
Latitude	3723.2475		ddmm.mmmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmmm
E/W Indicator	W		E=east or W=west
Position Fix Indicator	1		See Table B-3
Satellites Used	07		Range 0 to 12
HDOP	1.0		Horizontal Dilution of Precision
MSL Altitude[1]	9.0	meters	
Units	M	meters	
Geoid Separation[1]		meters	
Units	M	meters	
Age of Diff. Corr.		second	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*18		
<CR><LF>			End of message termination

SiRF Technology Inc. does not support geoid corrections. Values are WGS84 ellipsoid heights.

Table B-3 Position Fix Indicator

Value	Description
0	Fix not available or invalid
1	GPS SPS Mode, fix valid
2	Differential GPS, SPS Mode, fix valid
3	GPS PPS Mode, fix valid

ANNEXE 3

Fichier de référence pour le test de `decode_trame()`

Ce fichier a été obtenu par simple recopie de la sortie du G.P.S. embarqué dans un véhicule roulant à une vingtaine de km/h sur une courte période.

```
$GPGSV,3,2,10,15,03,077,,18,04,041,42,19,85,271,,20,08,214,*7C
$GPGSV,3,3,10,22,39,053,50,28,15,320,*7E
$GPRMC,141914.00,A,4545.6424,N,00306.6036,E,0.4,99.4,010206,,*0C
$GPGLL,4545.6424,N,00306.6036,E,141914.00,A*0E
$GPGGA,141914.00,4545.0000,N,00306.6036,E,1,05,3.4,499.3,M,,M,,*7D
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,99.4,T,,M,0.4,N,0.7,K*57
$GPZDA,141914.00,01,02,2006,00,00*69
$GPGSV,3,1,10,01,13,142,46,03,48,144,47,11,41,277,,14,27,104,42*75
$GPGSV,3,2,10,15,03,077,,18,04,041,41,19,85,271,,20,08,214,*7F
$GPGSV,3,3,10,22,39,053,48,28,15,320,*77
$GPRMC,141915.00,A,4545.6423,N,00306.6039,E,0.6,110.2,010206,,*31
$GPGLL,4545.6423,N,00306.6039,E,141915.00,A*07
$GPGGA,141915.00,4545.0242,N,00306.6039,E,1,05,3.4,499.5,M,,M,,*72
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,110.2,T,,M,0.6,N,1.2,K*67
$GPZDA,141915.00,01,02,2006,00,00*68
$GPGSV,3,1,10,01,13,142,45,03,48,144,47,11,41,277,,14,27,104,41*75
$GPGSV,3,2,10,15,03,077,,18,04,041,42,19,85,271,,20,08,214,*7C
$GPGSV,3,3,10,22,39,053,49,28,15,320,*76
$GPRMC,141916.00,A,4545.6422,N,00306.6037,E,0.1,211.1,010206,,*3B
$GPGLL,4545.6422,N,00306.6037,E,141916.00,A*0B
$GPGGA,141916.00,4545.0484,N,00306.6037,E,1,05,3.4,500.0,M,,M,,*7A
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,211.1,T,,M,0.1,N,0.2,K*60
$GPZDA,141916.00,01,02,2006,00,00*6B
$GPGSV,3,1,10,01,13,142,45,03,48,144,48,11,41,277,,14,27,104,43*78
$GPGSV,3,2,10,15,03,077,,18,04,041,44,19,85,271,,20,08,214,*7A
$GPGSV,3,3,10,22,39,053,48,28,15,320,*77
$GPRMC,141917.00,A,4545.6422,N,00306.6039,E,0.3,122.3,010206,,*37
$GPGLL,4545.6422,N,00306.6039,E,141917.00,A*04
$GPGGA,141917.00,4545.0726,N,00306.6039,E,1,05,3.4,499.6,M,,M,,*72
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,122.3,T,,M,0.3,N,0.5,K*64
$GPZDA,141917.00,01,02,2006,00,00*6A
$GPGSV,3,1,10,01,13,142,45,03,48,144,46,11,41,277,,14,27,104,41*74
$GPGSV,3,2,10,15,03,077,,18,04,041,42,19,85,271,,20,08,214,*7C
$GPGSV,3,3,10,22,39,053,48,28,15,320,*77
$GPRMC,141918.00,A,4545.6421,N,00306.6034,E,0.5,252.2,010206,,*35
$GPGLL,4545.6421,N,00306.6034,E,141918.00,A*05
$GPGGA,141918.00,4545.0968,N,00306.6034,E,1,05,3.4,498.8,M,,M,,*7C
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,252.2,T,,M,0.5,N,1.0,K*63
$GPZDA,141918.00,01,02,2006,00,00*65
$GPGSV,3,1,10,01,13,142,45,03,48,144,48,11,41,277,,14,27,104,42*79
$GPGSV,3,2,10,15,03,077,,18,04,041,42,19,85,271,,20,08,214,*7C
$GPGSV,3,3,10,22,38,053,49,28,15,320,*77
$GPRMC,141919.00,A,4545.6420,N,00306.6040,E,0.6,95.2,010206,,*0C
$GPGLL,4545.6420,N,00306.6040,E,141919.00,A*06
$GPGGA,141919.00,4545.1210,N,00306.6040,E,1,05,3.4,500.2,M,,M,,*75
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,95.2,T,,M,0.6,N,1.2,K*5B
$GPZDA,141919.00,01,02,2006,00,00*64
$GPGSV,3,1,10,01,13,142,46,03,48,144,49,11,41,277,,14,27,104,42*7B
```

```
$GPGSV,3,2,10,15,03,077,,18,04,041,43,19,85,271,,20,08,214,*7D
$GPGSV,3,3,10,22,38,053,49,28,15,320,*77
$GPRMC,141920.00,A,4545.6419,N,00306.6039,E,0.2,133.1,010206,,*38
$GPGLL,4545.6419,N,00306.6039,E,141920.00,A*08
$GPGGA,141920.00,4545.1410,N,00306.6039,E,1,05,3.4,500.0,M,,M,,*79
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,133.1,T,,M,0.2,N,0.4,K*66
$GPZDA,141920.00,01,02,2006,00,00*6E
$GPGSV,3,1,10,01,13,142,45,03,48,144,46,11,41,277,,14,27,104,43*76
$GPGSV,3,2,10,15,03,077,,18,04,041,41,19,85,271,,20,08,214,*7F
$GPGSV,3,3,10,22,38,053,50,28,15,320,*7F
$GPRMC,141921.00,A,4545.6419,N,00306.6043,E,0.6,103.1,010206,,*33
$GPGLL,4545.6419,N,00306.6043,E,141921.00,A*04
$GPGGA,141921.00,4545.1610,N,00306.6043,E,1,05,3.4,499.8,M,,M,,*7C
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,103.1,T,,M,0.6,N,1.1,K*65
$GPZDA,141921.00,01,02,2006,00,00*6F
$GPGSV,3,1,10,01,13,142,46,03,48,144,48,11,41,277,,14,27,104,42*7A
$GPGSV,3,2,10,15,03,077,,18,04,041,42,19,85,271,,20,08,214,*7C
$GPGSV,3,3,10,22,38,053,48,28,15,320,*76
$GPRMC,141922.00,A,4545.6418,N,00306.6046,E,0.7,96.9,010206,,*00
$GPGLL,4545.6418,N,00306.6046,E,141922.00,A*03
$GPGGA,141922.00,4545.1810,N,00306.6046,E,1,05,3.4,500.6,M,,M,,*74
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,96.9,T,,M,0.7,N,1.3,K*53
$GPZDA,141922.00,01,02,2006,00,00*6C
$GPGSV,3,1,10,01,13,142,47,03,48,144,46,11,41,277,,14,27,104,43*74
$GPGSV,3,2,10,15,03,077,,18,04,041,43,19,85,271,,20,08,214,*7D
$GPGSV,3,3,10,22,38,053,48,28,15,320,*76
$GPRMC,141923.00,A,4545.6417,N,00306.6046,E,0.3,113.6,010206,,*39
$GPGLL,4545.6417,N,00306.6046,E,141923.00,A*0D
$GPGGA,141923.00,4545.2010,N,00306.6046,E,1,05,3.4,500.6,M,,M,,*7A
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,113.6,T,,M,0.3,N,0.7,K*61
$GPZDA,141923.00,01,02,2006,00,00*6D
$GPGSV,3,1,10,01,13,142,45,03,48,144,47,11,41,277,,14,27,104,41*75
$GPGSV,3,2,10,15,03,077,,18,04,041,43,19,85,271,,20,08,214,*7D
$GPGSV,3,3,10,22,38,053,49,28,15,320,*77
$GPRMC,141924.00,A,4545.6416,N,00306.6044,E,0.2,197.7,010206,,*31
$GPGLL,4545.6416,N,00306.6044,E,141924.00,A*09
$GPGGA,141924.00,4545.2210,N,00306.6044,E,1,05,3.4,500.5,M,,M,,*7D
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,197.7,T,,M,0.2,N,0.4,K*6E
$GPZDA,141924.00,01,02,2006,00,00*6A
$GPGSV,3,1,10,01,13,142,45,03,48,144,47,11,41,277,,14,27,104,40*74
$GPGSV,3,2,10,15,03,077,,18,04,041,43,19,85,271,,20,08,214,*7D
$GPGSV,3,3,10,22,38,053,49,28,15,320,*77
$GPRMC,141925.00,A,4545.6415,N,00306.6046,E,0.2,130.8,010206,,*33
$GPGLL,4545.6415,N,00306.6046,E,141925.00,A*09
$GPGGA,141925.00,4545.2410,N,00306.6046,E,1,05,3.4,501.4,M,,M,,*7D
$GPGSA,A,3,,03,,22,14,,01,,18,,,,3.9,3.4,1.9*39
$GPVTG,130.8,T,,M,0.2,N,0.5,K*6D
$GPZDA,141925.00,01,02,2006,00,00*6B
$GPGSV,3,1,10,01,13,142,45,03,48,144,47,11,41,277,,14,27,104,40*74
$GPGSV,3,2,10,15,03,077,,18,04,041,41,19,85,271,,20,08,214,*7F
$GPGSV,3,3,10,22,38,053,47,28,15,320,*79
$GPRMC,141926.°E,0.2,260.8,010206,,*34
```