Note d'application

Acquisition de données séries binaires via une interface graphique Matlab®

P09AB08 : Odométrie ferroviaire par fusion de données accélérométriques et GPS

Présentation

Cette note d'application a pour but de présenter une manière simple et efficace de réaliser l'acquisition sur PC de données séries binaires pour permettre un traitement ultérieur.

L'outil choisi est Matlab®, pour la polyvalence qu'il permet en aval de l'acquisition. Matlab® fonctionne à partir d'un langage interprété dont les bases ne sont pas l'objet de cette note. Un minimum de connaissances dans la syntaxe et la gestion des objets graphiques (figures, axes…) est recommandé pour une bonne compréhension.

L'intérêt de l'utilisation d'un GUI pour l'acquisition de données série est de pouvoir paramétrer de façon simple la connexion et de pouvoir sauvegarder facilement le résultat.

1 Rappels sur la liaison série

1.1 Format des données

Le format RS232 définit la transmission de données sous forme de trames, comprenant un paquet de données de 5 à 8 bits.

	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7	Bit8	Bit9	Bit10
Longueur	1	1	1	1	1	1	0 – 1	0 – 1	0 – 1	0 – 1	1 - 2
Signification	Start	D0	D1	D2	D3	D4	D5	D6	D7	Parité	Stop

Tableau 1 : Format d'une trame RS232

Nous traiterons dans le cadre de cette note des données 8 bits sans vérification de parité, et avec un bit de stop.

Ceci permet de transmettre la majorité des données binaires : par exemple, la réception d'une donnée 8, 16, 32 bits peut se faire respectivement par la réception d'une, de deux, ou de quatre trames 8 bits.

1.2 Interface série sur PC

La liaison série sur PC, et principalement son connecteur associé (DB9) tend à disparaître des nouvelles machines. De nombreux adaptateurs série → USB permettent d'émuler des ports séries et donc de contourner le problème de connectique.

Tous les systèmes d'exploitation courants sont capables de gérer ce type de périphériques, mais il faut noter que la plupart ne proposent que des drivers Windows®.

2 Présentation du GUI Matlab® utilisé

2.1 Fonctionnalités

Le GUI[1] utilisé ici a été créé dans un but de simplicité. Il doit permettre l'ouverture d'un port série directement sur Matlab®, et proposer la configuration de la vitesse et du port à utiliser.

Note : Matlab® n'est pas en mesure de détecter directement les ports connectés à un périphérique. Il est nécessaire d'avoir identifié au préalable, via le système d'exploitation par exemple, le port série concerné.

On a donc les fonctionnalités suivantes :

- Ouverture, fermeture d'un port série
- Configuration des paramètres de bases de la liaison (vitesse, port)
- Lancement, et arrêt de l'acquisition
- Mise en forme des données (totalement dépendant de l'application)

[1] GUI : Graphic User Interface

- Sauvegarde des données produites.

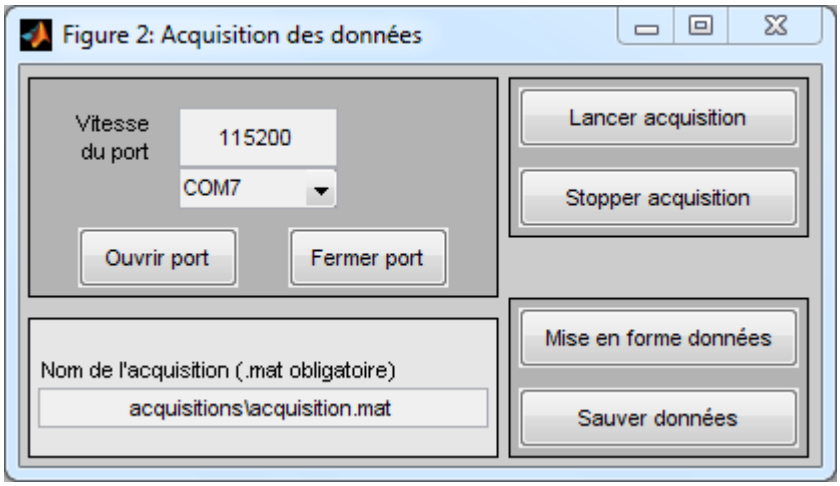

Figure 1 : Visualisation de l'interface

2.2 *Structure*

Voici un aperçu d'une version simple du GUI.

```
%---------------------------------------------------------------
%Creation du GUI
%---------------------------------------------------------------
function acquisition_serie_donnees()


%Ouverture de la fenetre
%---------------------------------------------------------------
handle(1) = figure('Name','Acquisition des données',...
                   'Position',[500, 300, 400, 200],...
                   'menubar','none');
%---------------------------------------------------------------

%Creation de la zone d'acquisition
%---------------------------------------------------------------
%Bouton : lancer acquisition
handle(2) = uicontrol('pos',[250,160,140,30],...
                      'style','pushbutton',...
                      'callback',@cb_start_acquisition,...
                      'String','Lancer acquisition');

%Bouton : stopper acquisition
handle(3) = uicontrol('pos',[250,120,140,30],...
                      'style','pushbutton',...
                      'callback',@cb_stop_acquisition,...
                      'String','Stopper acquisition');
%---------------------------------------------------------------

%Zone mise en forme et sauvegarde
```

```
%-------------------------------------------------------------
%Bouton : Mise en forme des données
handle(4) = uicontrol('pos',[250,50,140,30],...
                      'style','pushbutton',...
                      'callback',@cb_mise_forme_donnees,...
                      'String','Mise en forme données');

%Bouton : Sauver donnees
handle(5) = uicontrol('pos',[250,10,140,30],...
                      'style','pushbutton',...
                      'callback',@cb_sauv_donnees,...
                      'String','Sauver données');
%-------------------------------------------------------------

%Zone paramètres liaison série
%-------------------------------------------------------------
%Bouton : Ouvrir port
handle(6) = uicontrol('pos',[30,90,80,30],...
                      'style','pushbutton',...
                      'callback',@cb_ouvrir_port,...
                      'String','Ouvrir port');

%Bouton : Fermer port
handle(7) = uicontrol('pos',[135,90,80,30],...
                      'style','pushbutton',...
                      'callback',@cb_fermer_port,...
                      'String','Fermer port');

%Champ de texte : Vitesse du port
handle(8) = uicontrol('pos',[28,148,40,30],...
                      'style','text',...
                      'string','Vitesse du port',...
                      'backgroundcolor',[0.7,0.7,0.7]);

%Champ parametrable de la vitesse
handle(9) = uicontrol('pos',[80,150,80,30],...
                      'style','edit',...
                      'string','115200',...
                      'tag','champ_vitesse');

%Popupmenu choix port serie
handle(10) = uicontrol('pos',[80,120,80,30],...
                       'style','popupmenu',...
                       'String',{'COM1','COM2','COM3',...
                       'COM4','COM5','COM6','COM7',...
                       'COM8','COM9','COM10'},...
                       'tag','selection_com');
%-------------------------------------------------------------

%Zone nom de l'acquisition
%-------------------------------------------------------------
%Affichage choix du nom
handle(11) = uicontrol('pos',[10,40,180,15],...
                       'style','text',...
                       'string','Nom de l''acquisition (.mat)',...
                       'backgroundcolor',[0.9,0.9,0.9]);

%Champ contenant le nom
handle(13) = uicontrol('pos',[10,20,225,20],...
```

```
                        'style','edit',...
                        'string','acquisitions\acquisition.mat',...
                        'tag','champ_nom');
%--------------------------------------------------------------

data=guihandles(gcf);
guidata(gcf,data);
%--------------------------------------------------------------
%Fin : Creation du GUI
%--------------------------------------------------------------
```

Code I : GUI contenu dans `acquisition_serie_donnees.m`

3 Structure `serial`

L'objet serial permet l'accès aux différents ports série d'une machine. Plusieurs possibilités existent pour l'ouverture d'un port, suivant le système d'exploitation.

Système d'exploitation	Constructeur type
Linux / Linux 64	`serial('/dev/ttyS0');`
Mac OS	`serial('/dev/tty.KeySerial1');`
Solaris 64	`serial('/dev/term/a');`
Windows 32/64	`serial('com1');`

Tableau 2 : Exemples de constructeurs des objets sériels

L'appel de la primitive serial renvoie un identifieur contenant tous les paramètres de la liaison série.

Dans le cas où nous l'utilisons :

- 8 bits
- pas de parité
- un bit de stop

et vis à vis des valeurs par défaut des propriétés de l'objet serial, il faut configurer au minimum les champs Port, BaudRate, InputBuffer (choix de la tailler du buffer d'entrée) et Timeout (si la valeur par défaut de 0.01s ne convient pas à l'application) .

```
s = serial('COM1');
set(s)
    ByteOrder: [ {littleEndian} | bigEndian ]
    BytesAvailableFcn
    BytesAvailableFcnCount
    BytesAvailableFcnMode: [ {terminator} | byte ]
    ErrorFcn
    InputBufferSize
    Name
    OutputBufferSize
    OutputEmptyFcn
    RecordDetail: [ {compact} | verbose ]
    RecordMode: [ {overwrite} | append | index ]
    RecordName
    Tag
    Timeout
    TimerFcn
    TimerPeriod
    UserData
    SERIAL specific properties:
    BaudRate
    BreakInterruptFcn
    DataBits
    DataTerminalReady: [ {on} | off ]
    FlowControl: [ {none} | hardware | software ]
    Parity: [ {none} | odd | even | mark | space ]
    PinStatusFcn
    Port
    ReadAsyncMode: [ {continuous} | manual ]
    RequestToSend: [ {on} | off ]
    StopBits
    Terminator
```

Figure 2 : Propriétés de l'objet `serial` (sous Windows®)

4 Précautions d'usage

4.1 Allocation dynamique/Pré allocation

Ce point est critique dans la réception des données. L'acquisition série sous Matlab®, en flux continu et avec un débit élevé (au delà de quelques ko.s^{-1}), ne peut pas se faire directement sur un vecteur de taille inconnue au départ. L'augmentation constante de la taille du vecteur force une série de réallocations qui ont pour effet de provoquer une perte de données à chaque fois.

La technique la plus simple consiste à préallouer un vecteur de taille suffisante au regard du débit et de la durée d'acquisition.

4.2 Ouverture/fermeture de ports

La gestion des ports série est rigoureuse dans Matlab®. Il est important de toujours respecter la séquence suivante :

1. ouverture du port
2. lancement de l'acquisition
3. arrêt de l'acquisition
4. mise en forme ou près traitement si nécessaire
5. fermeture du port

et en particulier, il faut impérativement éviter (sous peine de devoir relancer Matlab®) :

- d'ouvrir deux fois le même port
- d'ouvrir un port qui n'est pas connecté
- de quitter le GUI sans avoir fermé le port

4.3 Vidage du buffer d'entrée

Lors du démarrage de l'acquisition, si le flux est permanent, il faut impérativement vider le buffer d'entrée, quelle que soit sa taille.

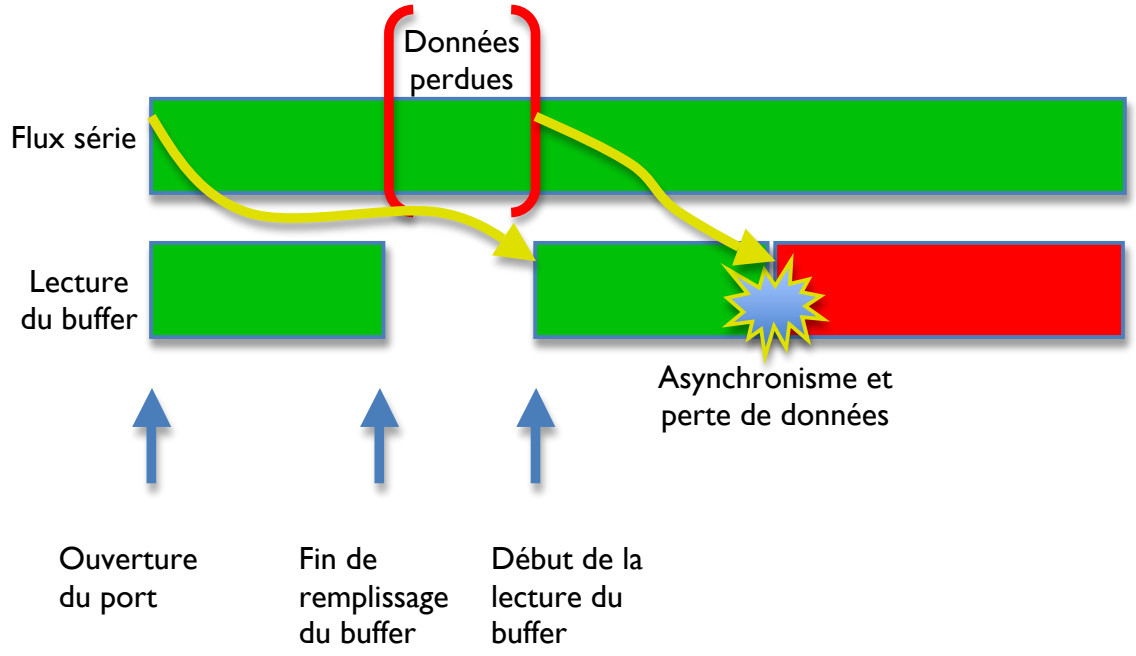

Figure 3 : Perte de données par non vidage du buffer

La perte de données peut être simple si les données n'ont qu'une seule signification, elle peut être plus complexe à gérer si la réception se fait sous forme de trames : la synchronisation est perdue.

5 Exemple des callbacks liés aux éléments du GUI

5.1 Bouton – Ouvrir port

Cette fonction a pour but de créer l'objet serial avec les paramètres choisis dans le GUI.

```
%-------------------------------------------------------------
%Callback : ouvrir port
function cb_ouvrir_port(obj,event)
data=guidata(gcbf);

global s;

%selection du port en fonction du popup menu
switch get(data.selection_com,'value')
    case 1,
        port = 'COM1';
    case 2,
        port = 'COM2';
    case 3,
        port = 'COM3';
    case 4,
        port = 'COM4';
    case 5,
        port = 'COM5';
    case 6,
        port = 'COM6';
    case 7,
        port = 'COM7';
    case 8,
        port = 'COM8';
    case 9,
        port = 'COM9';
    case 10,
        port = 'COM10';
    otherwise,
        port = 'COM1';
end;

s = serial(port,'BaudRate', str2num(get(data.champ_vitesse,'string')));
set(s,'InputBuffer', 8192);
fopen(s);

guidata(gcbf,data);
```

Code 2 : Callback - Ouvrir port

L'objet serial `s`, assimilable à un fichier, est déclaré en tant que variable globale : en effet, il doit être accessible par la fonction de fermeture du port, et d'acquisition des données.

5.2 Bouton – Fermer port

```
%-------------------------------------------------------------
%Callback : fermer port
function cb_fermer_port(obj,event)
data=guidata(gcbf);

global s;

fclose(s);  %séquence obligatoire de fermeture de l'objet
delete(s);
clear s;

guidata(gcbf,data);
```

Code 3 : Callback – Fermer port

5.3 Bouton – Lancer acquisition

Le démarrage de l'acquisition doit se faire en respectant les règles décrites précédemment : le buffer d'entrée doit être vidé avant de démarrer, et un vecteur d'acquisition doit être préalloué.

```
%-------------------------------------------------------------
%Callback : start acquisition
function cb_start_acquisition(obj,event)
data=guidata(gcbf);
global out;
global acqu_on;
global s;

limite_taille = 10000000;

out = zeros(1,limite_taille); %préallocation
index = 1;
acqu_on = 1;

%vidage du buffer
flushinput(s);
pause(0.01);

%acquisition, tant que acqu_on =1 et limite de taille non atteinte
while ((acqu_on)&&(index < (limite_taille-get(s,'InputBuffer'))))
    nb_octets = get(s,'BytesAvailable');
    if nb_octets
    out(index:(index + nb_octets - 1)) = fread(s,nb_octets);
    index = index + nb_octets;
    end;
    %rend la main a Matlab pour permettre le passage de acqu_on à 0
    pause(0.2);
end;

%suppression des elements non ecrits si arret manuel
out(index:limite_taille) = '';
guidata(gcbf,data);
```

Code 4 : Callback – Lancer acquisition

5.4 Bouton – Stopper acquisition

L'arrêt de l'acquisition, s'il n'est pas donné par la saturation du vecteur préalloué (`out`) est commandé par la valeur de `acqu_on`. Le passage à 0 fait sortir le callback `cb_start_acquisition` de sa boucle principale.

```
%-------------------------------------------------------------
%Callback : stop acquisition
function cb_stop_acquisition(obj,event)
data=guidata(gcbf);

global acqu_on;

acqu_on = 0;

guidata(gcbf,data);
```

Code 5 : Callback – Stopper acquisition

5.5 Bouton – Mise en forme des données

Cette fonction n'a pas d'intérêt à être présentée ici, étant donné qu'elle dépend entièrement du type de données reçues. On peut envisager simplement de convertir plusieurs octets en un nombre de dynamique plus grande, ou reconstituer des chaines de caractères…

De façon générale, et en accord avec l'interface, cette fonction doit générer un vecteur ou un tableau, ou une structure `donnees_formatees` à partir du vecteur `out`.

5.6 Bouton – Sauver Données

La sauvegarde des données reçues peut se faire sous différentes formes, ici on crée directement un fichier .mat ouvrable par Matlab®. On peut envisager de tester la fin de la chaine de caractères du nom (ici get(data.champ_nom,'string')) pour rajouter si nécessaire l'extension .mat, mais on peut aussi pour simplifier obliger à saisir nom de fichier contenant la bonne extension.

```
%-------------------------------------------------------------
%Callback : sauvegarde des données
function cb_sauv_donnees(obj,event)
data=guidata(gcbf);

global out;
global donnees_formatees;

%enregistrer sous le nom de champ_nom le vecteur out, et données formatées
save(get(data.champ_nom,'string'),'out','donnees_formatees');

guidata(gcbf,data);
```

Code 6 : Callback – Sauvegarde des données

6 Synchronisation des données reçues

6.1 Différences avec la réception de données ASCII

Les caractères ASCII permettent, au prix d'une surcharge d'information (pour le transfert d'une valeur 8 bits, il faut transférer 2 octets hexadécimaux), de réserver un bon nombre de caractères comme terminateurs d'une trame série (champ `Terminator` de l'objet serial).

La réception de données binaires ne peut pas fonctionner sur le même schéma. Les trames ne peuvent, par définition, pas être bornées par des valeurs connues, sauf dans certains cas particuliers.

6.2 Réception de données complexes

Les données reçues via la liaison série mise en œuvre ne sont pas forcement une succession de valeurs 8 bits représentant une même grandeur. Dans un cas aussi simple, aucun problème de synchronisation n'est à relever.

Si les données reçues sont plus complexes, i.e. représentant des grandeurs différentes et/ou de types différents, il faut choisir un motif que l'on peut prévoir a priori.

Il faut garder en tête que des valeurs binaires, sauf si elles sont bornées, peuvent prendre toutes les valeurs de leur dynamique. Une simple valeur ne peut donc pas servir artificiellement de terminateur. En revanche, on peut supposer qu'une succession de plusieurs valeurs connues et dans une séquence connue peut représenter un motif assimilable à un terminateur. Il convient de s'assurer que la probabilité d'occurrence de ce motif est très faible parmi les autres données.

Bibliographie

Briot, Jérôme. «Développement efficace des interfaces graphiques.» *Devellopez.com*. 17 03 2009.

—. «Présentation des objets graphiques.» *Developpez.com*. 17 03 2009.

RS232 Specifications and standard. http://www.lammertbies.nl/comm/info/RS-232_specs.html (accès le 01 24, 2010).

Contacts

Cette note d'application a été réalisée par Jean Doucement. Pour toute question ou précision :

jean.doucement@polytech.univ-bpclermont.fr

ou

jean.doucement@gmail.com