LA LIAISON SERIE AVEC DOMOTICOM

1 - Qu'est-ce que DOMOTICOM ?

DOMOTICOM est une version de PANORAMIC spécialisée dans la technique, la robotique, la domotique, les communications, la télémétrie.

Elle possède :
- des objets spécialisés que PANORAMIC ne possède pas (des LEDs, des jauges, des interrupteurs, des indicateurs, des cadrans, ...),
- des commandes/fonctions que PANORAMIC ne possède pas, et c'est normal, c'est une version spécialisée, elle a des fonctionnalités supplémentaires.

Par contre, elle ne possède pas certains objets de PANORAMIC (pas de SCENE2D, pas de SCENE3D, pas de SOUND, …).

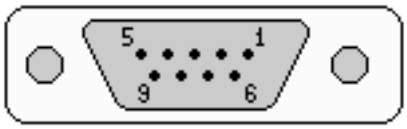
2 - Qu'est-ce qu'une liaison série ?

Une liaison série est un moyen de communication qui est très utilisé par les ordinateurs pour communiquer avec des périphériques ou avec d'autres ordinateurs, car elle est simple à utiliser et ne nécessite que peu de fils.

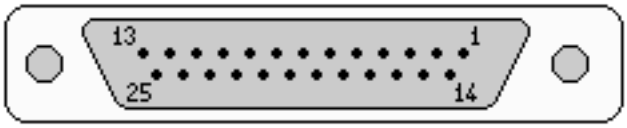
Disponibles sur presque tous les PC jusqu'au milieu des années 2000, les ports série sont désignés par les noms COM1, COM2, … sur les systèmes d'exploitation MS-DOS et Windows. Cela leur a valu le surnom de « ports COM », encore utilisé de nos jours.

2.1 – Les connecteurs utilisés

Le connecteur utilisé à l'origine pour le port série est le connecteur DB25 (à 25 points), mais on trouve actuellement beaucoup de connecteurs DB9 (à 9 points), car ils ont été introduits par IBM. Au niveau du PC, on utilise des connecteurs mâles, tandis qu'au niveau du périphérique, on trouve des connecteurs femelles.



Connecteur DB9 (9 broches numérotées de 1 à 9)



Connecteur DB25 (25 broches numérotées de 1 à 25)

2.2 – Niveaux électriques

Les différents signaux présents sur les broches de la prise sont définis par la norme RS232 :
- les tensions utilisées sont comprises entre -25 et +25V,
- une tension comprise entre -3V et -25V représente un « 1 » logique,
- une tension comprise entre +3V et +25V représente un « 0 » logique.

En pratique, des niveaux de +12V et -12V sont utilisés, et :
- un "1" est reconnu si la tension est inférieure à -3 V,
- un "0" est reconnu si la tension est supérieure à +3 V.

2.3 – La transmission de caractères

La transmission des informations s'effectue bit par bit, de manière séquentielle, d'où le nom de « série ». Une communication en série consiste à transmettre des informations après les avoir préalablement découpées en plusieurs morceaux de taille fixe, en général des caractères (des octets, soit 8 bits).

Pour établir une communication entre un ordinateur et un périphérique, il est nécessaire de définir le protocole utilisé : le débit de la transmission, le codage utilisé, le découpage en trame, etc.

On découpe les échanges en trames d'un caractère ainsi constituées :
1 bit de départ (start bit)
5 à 8 bit de données (caractère)
1 bit de parité (parity bit, qui est optionnel)
1 ou 2 bits d'arrêt (stop bits).

Voici comment se déroule une transmission :

1) Lorsqu'il n'y a pas d'échange, le niveau sur la ligne de transmission est au 1 logique, c'est à dire à une tension de -12 V.

2) Le bit de départ (**start bit** en anglais) a un niveau logique "0" pour indiquer au récepteur qu'une transmission va commencer.

3) Les bits de donnée sont ensuite transmis, **en commençant par le bit de poids faible**.

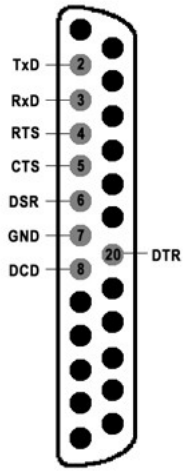
4) On ajoute parfois un bit de parité. Ce bit peut être un bit de parité paire ou de parité impaire.

Un bit de parité paire est tel que le nombre des bits de donnée et le bit de parité est un nombre pair.
Exemples :
pour l'octet **01011001**, qui a 4 bits à 1, le bit de parité sera mis à 0 car 4 est pair. On a donc un nombre total de bits à 1 qui est pair.
pour l'octet **01010001**, qui a 3 bits à 1, le bit de parité sera mis à 1, comme cela le nombre total de bits à 1 (donnée + parité) est un nombre pair.

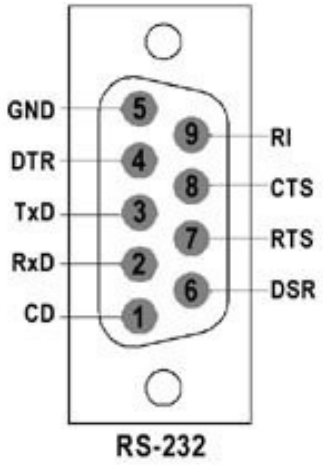
Un bit de parité impaire est tel que le nombre des bits de donnée et le bit de parité est un nombre impair.
Exemples :
pour l'octet **01011001**, qui a 4 bits à 1, le bit de parité sera mis à 1, le nombre total de bits à 1 sera donc impair,
pour l'octet **01010001**, qui a 3 bits à 1, le bit de parité sera mis à 0, comme cela le nombre total de bits à 1 sera un nombre impair.

5) Le transmetteur ajoute un bit d'arrêt (**stop bit** en anglais).
Le but du « stop bit » est de ramener la ligne à l'état 1 (tension -12 V), qui est l'état normal entre 2 caractères et quand il n'y a pas d'échange. Un autre but du stop bit est de donner du temps au récepteur pour traiter le caractère reçu. Il faut noter que pour certains périphériques lents, on utilise 2 stop bits au lieu d'un pour lui laisser plus de temps. Certains vieux périphériques demandaient le temps de 1,5 bit entre chaque caractère !

2.4 – Le brochage des prises



Brochage de la prise type DB25



Brochage de la prise de type DB9

Nous allons nous intéresser à la prise de type DB9 qui est la plus utilisée :

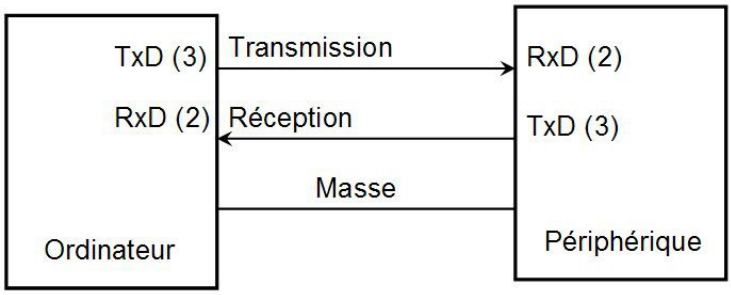
Numéro	Nom	Désignation
1	CD - Carrier Detect	Détection de porteuse
2	RxD - Receive Data	Réception de données
3	TxD - Transmit Data	Transmission de données
4	DTR - Data Terminal Ready	Terminal prêt
5	GND - Signal Ground	Masse logique
6	DSR - Data Set Ready	Données prêtes
7	RTS - Request To Send	Demande d'émission
8	CTS - Clear To Send	Prêt à émettre
9	RI - Ring Indicator	Indicateur de sonnerie
	Shield	Blindage

Signification des broches de la prise de type DB9

Rassurez-vous, toutes les broches ne sont pas utilisées.
Dans la plupart des liaisons série, il n'y a que **3 broches** qui sont indispensables, et on réalise une liaison série avec 3 fils : TxD (Transmission de données), RxD (Réception de données) et la masse, qui sert de référence pour les tensions.

2.5 – Câblage

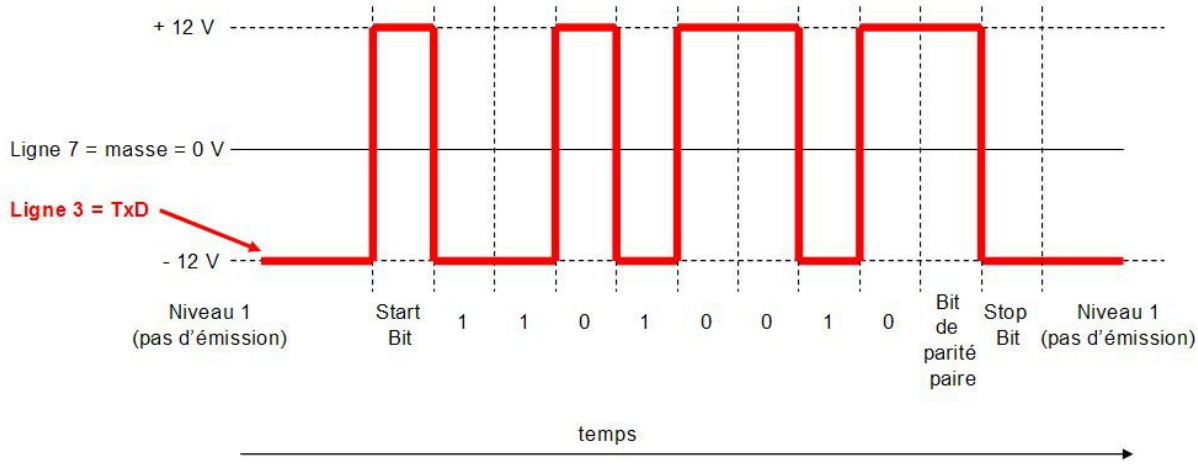
Pour réaliser un câble qui relie un ordinateur à un périphérique, on a donc besoin de 2 prises DB9 et de 3 fils. Noter qu'entre l'ordinateur et le périphérique, le fil qui va de la broche 2 de l'un va sur la broche 3 de l'autre (la transmission de l'un est la réception de l'autre).



Liaison série avec 3 fils

2.6 – Récapitulons

Quand l'ordinateur envoie le caractère « K » à un périphérique (soit la valeur 11010010), avec un port série configuré pour des caractères de 8 bits, avec une parité paire et un stop bit, la tension entre le fil de masse (broche 7) et le fil TxD (broche 3) varie comme ceci :



Cette longue, mais nécessaire partie théorique étant terminée, passons à la pratique, et passons du côté programmation.

Tout d'abord, ne vous affolez pas, le programmeur n'a pas du tout à gérer ces échanges.

Il a juste à **spécifier la configuration utilisée** : la vitesse, le nombre de bits de caractères, s'il y a une parité et si oui, si elle est paire ou impaire, et le nombre de stop bits.
Ensuite, **il envoie ou il reçoit des chaines de caractères** ! Le hard se débrouille pour tout le reste : start bit, timing de la transmission, calcul du bit de parité, stop bit, ...

3 - Comment utilise-t-on une liaison série avec DOMOTICOM ?

Pour utiliser une liaison série, il y a 10 commandes et 3 fonctions.

Les commandes :

SERIAL N
SERIAL_PORT N,P
SERIAL_DATARATE N,R
SERIAL_STOPBITS N,S
SERIAL_PARITY N,P
SERIAL_OPEN N
SERIAL_CLOSE N
SERIAL_WRITE N,V$
ON_RECEIVE N,LABEL
DELETE N

Les fonctions:

SERIAL_PORT_EXISTS(X)
COUNT(N)
SERIAL_READ$(N)

3.1 - Détail de ces commandes et fonctions :

3.1.1 - création d'un objet gérant les liaisons série

SERIAL N

Crée un objet système SERIAL et lui donne le numéro N comme référence.
Cet objet sert à gérer la liaison série qui portera le numéro N.
Il y a erreur:
- si un objet système porte déjà le numéro N

3.1.2 - destruction d'un objet gérant les liaisons série

Un objet SERIAL est un objet système non visible.
Par conséquent, on le détruit par la commande classique **DELETE N**.

3.1.3 - configuration d'une liaison série

SERIAL_PORT N,P

définit le numéro de port P de la liaison série numéro N.
P doit avoir une valeur de 1 à 10.
Il y a erreur:
- si l'objet SERIAL numéro N n'a pas été créé
- si la valeur P n'est pas l'une des valeurs suivantes : 1,2,3,4,5,6,7,8,9,10

Exemple : SERIAL_PORT 1,2 utilise le port COM2 de l'ordinateur pour y gérer une liaison série.

SERIAL_DATARATE N,R

définit la vitesse en Bauds R de la liaison série numéro N.
R doit avoir une des valeurs suivantes : 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200.
Il y a erreur:
- si l'objet SERIAL numéro N n'a pas été créé
- si la valeur R n'est pas l'une des valeurs suivantes : 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200

SERIAL_DATABITS N,D

définit le nombre de bits de données D de la liaison série numéro N.
D doit avoir une des valeurs suivantes : 5, 6, 7, 8.
Il y a erreur:
- si l'objet SERIAL numéro N n'a pas été créé
- si la valeur D n'est pas l'une des valeurs suivantes : 5, 6, 7, 8

SERIAL_STOPBITS N,S

définit le nombre de bits d'arrêt S de la liaison série numéro N.
S doit avoir une des valeurs suivantes : 1, 1.5, 2.
Il y a erreur:
- si l'objet SERIAL numéro N n'a pas été créé
- si la valeur S n'est pas l'une des valeurs suivantes : 1, 1.5, 2

SERIAL_PARITY N,P

définit le contrôle de parité P lors des échanges de données sur la liaison série de numéro N.
P doit avoir une valeur égale à 0 (pas de parité), 1 (parité impaire), 2 (parité paire).
Si P vaut 0, il n'y a pas de bit de parité.
Il y a erreur:
- si l'objet SERIAL numéro N n'a pas été créé
- si la valeur P n'est pas l'une des valeurs suivantes : 0, 1, 2

SERIAL_OPEN N

démarre les échanges sur la liaison série numéro N.
Il y a erreur:
- si l'objet SERIAL numéro N n'a pas été créé

SERIAL_CLOSE N

arrête les échanges sur la liaison série numéro N.
Il n'est plus possible de recevoir ou d'émettre des données.
Il y a erreur:
- si l'objet SERIAL numéro N n'a pas été créé

3.1.4 - utilisation d'une liaison série (échange de données)

en réception :

COUNT(N)

retourne le nombre de caractères reçus sur la liaison série numéro N depuis la dernière lecture.
Remarque : le fait de lire les caractères reçus par la fonction **SERIAL_READ$(N)** remet ce compteur à 0.
Il y a erreur:
- si l'objet SERIAL numéro N n'a pas été créé

ON_RECEIVE N,L

exécute le programme à partir du label L quand un caractère est reçu sur la liaison série numéro N.
Il y a erreur:
- si l'objet SERIAL numéro N n'a pas été créé

SERIAL_READ$(N)

retourne les caractères reçus sur la liaison série numéro N et vide le buffer de réception de cette liaison série (le nombre de caractères reçus **COUNT(N)** est remis à 0).
Il y a erreur:
- si l'objet SERIAL numéro N n'a pas été créé

en émission :

SERIAL_WRITE N,V$

envoie la chaine de caractères V$ sur la liaison série numéro N.
Il y a erreur:
- si l'objet SERIAL numéro N n'a pas été créé.

3.1.5 - test des ports physiques

SERIAL_PORT_EXISTS(X)

Teste si le port série numéro X existe physiquement et retourne 1 s'il existe et 0 sinon. Attention, cette commande teste si le port numéro X existe physiquement sur l'ordinateur. Elle ne teste pas si l'objet SERIAL de numéro X a été créé. Pour tester si un objet SERIAL de numéro X a été créé, il faut utiliser la commande OBJECT_EXISTS(X)

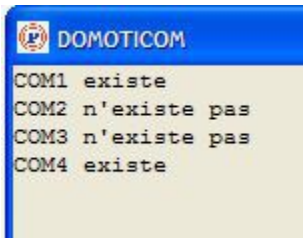
Appliquons maintenant ces commandes et ces fonctions au cours de quelques exemples.

4 - Premier exemple : test des ports série disponibles sur un ordinateur

On tape et on exécute le programme suivant :

```
if serial_port_exists(1)=1
print "COM1 existe"
else
print "COM1 n'existe pas"
end_if

if serial_port_exists(2)=1
print "COM2 existe"
else
print "COM2 n'existe pas"
end_if

if serial_port_exists(3)=1
print "COM3 existe"
else
print "COM3 n'existe pas"
end_if

if serial_port_exists(4)=1
print "COM4 existe"
else
print "COM4 n'existe pas"
end_if
```

On obtient par exemple :



Ce qui signifie que sur cet ordinateur, seuls les ports COM1 et COM4 sont utilisables.

5 - Deuxième exemple : envoi d'une commande à une carte Arduino

On va s'amuser avec la LED qui est sur la carte elle-même. Il n'y a besoin de rien d'autre qu'une carte Arduino et de son câble USB.

Je suppose que l'IDE Arduino est installé sur votre PC et qu'une carte Arduino y est connectée par sa liaison USB.

Lancer arduino.exe.

Taper le programme suivant (vous n'êtes bien sûr pas obligé de taper les commentaires qui commencent par //) :

```
char reception;   // pour recevoir un octet
byte led = 13;    // led = broche 13

// s'exécute au démarrage ou au reset
void setup()
  {
  // ouvre la liaison série
  // et fixe le débit de communication à 1200 bauds
  Serial.begin(1200);
  // initialise la broche "led" en sortie
  pinMode(led, OUTPUT);
  }

// le programme principal est une boucle infinie
void loop()
  {
  // teste si quelque chose a été reçu
  if (Serial.available())
      {
      // lecture de la donnée reçue:
      reception = Serial.read();
      if (reception=='A')
         {
         // A => on allume la LED
         digitalWrite(led, HIGH);
         }
      if (reception=='Z')
         {
         // Z => on éteint la LED
         digitalWrite(led, LOW);
         }
      }
  }
```

Programmer la carte Arduino avec ce programme :
Fichier / Téléverser
La carte Arduino exécute alors le programme téléchargé.

Dans l'éditeur DOMOTICOM, taper le programme suivant :

```
label allume, eteint

width 0,200
height 0,90

rem création d'un bouton
button 1
caption 1,"Allume"
on_click 1, allume

rem création d'un autre bouton
button 2
left 2,100
caption 2,"Eteint"
on_click 2, eteint

rem création d'un objet qui gère les liaisons série
serial 3

rem paramétrage de la liaison série
serial_port 3,4 : rem si liaison avec COM4 !
serial_baudrate 3,1200
serial_databits 3,8
serial_stopbits 3,1
serial_parity 3,0
serial_open 3
end

allume:
rem transmission du caractère "A"
serial_write 3,"A"
return

eteint:
rem transmission du caractère "Z"
serial_write 3,"Z"
return
```

On a réalisé un programme avec DOMOTICOM qui possède 2 boutons, un clic sur l'un allume la LED en envoyant le caractère « A » sur la liaison série, et un clic sur l'autre éteint cette LED en envoyant le caractère « Z » sur la liaison série.

L'envoi d'un caractère A ou Z revient à envoyer à une carte Arduino une commande d'allumage / extinction de sa LED.

Attention, sur votre ordinateur, la ligne **serial_port 3,4** est peut-être à modifier car elle dépend de votre configuration : la carte Arduino utilise peut-être un autre port !

6 - Troisième exemple : réception d'une chaîne de caractères d'une carte Arduino

La réception de caractères sur une ligne série est un événement extérieur imprévisible (le programme ne peut pas deviner quand des données seront reçues), et il y a 2 façons de traiter cela avec PANORAMIC (ou DOMOTICOM), selon votre manière de programmer :

- la manière événementielle,
- la manière séquentielle.

Rappelons la différence essentielle entre les 2 méthodes :

- en programmation événementielle, une partie du programme s'exécute automatiquement quand des caractères sont reçus.
- en programmation séquentielle, c'est le programme lui-même qui teste si des caractères ont été reçus ou non, puis qui les traite.

6.1 - réception de données par la manière événementielle

La réception de données par la manière événementielle se fait en gérant un événement « RECEIVE » qui se déclenche à chaque réception d'un caractère : **ON_RECEIVE N,Label**

6.2 - réception de données par la manière séquentielle

La réception de données par la manière séquentielle se fait en testant de temps en temps s'il y a des caractères reçus par la fonction **COUNT(N)** qui retourne le nombre de caractères reçus sur la liaison série numéro N.

6.3 – lecture de ce qui a été reçu

Dans les 2 méthodes, la lecture des caractères reçus se fait ensuite par la fonction **SERIAL_READ$(N)** qui retourne les caractères reçus sur la liaison série numéro N.

Exemples:

Lancer arduino.exe.

Taper le programme suivant :

```
// la routine setup qui s'exécute au démarrage ou au reset
void setup()
  {
  // ouvre la liaison série
  // et fixe le débit de communication à 1200 bauds
  Serial.begin(1200);
  }
// le programme principal qui est une boucle infinie
void loop()
  {
  // envoie un texte sur la liaison série
  Serial.println("Les Panoramiciens font de la robotique !");
  // attend 5 secondes (5000 millisecondes)
  delay(5000);
  }
```

Programmer la carte Arduino avec ce programme :
Fichier / Téléverser

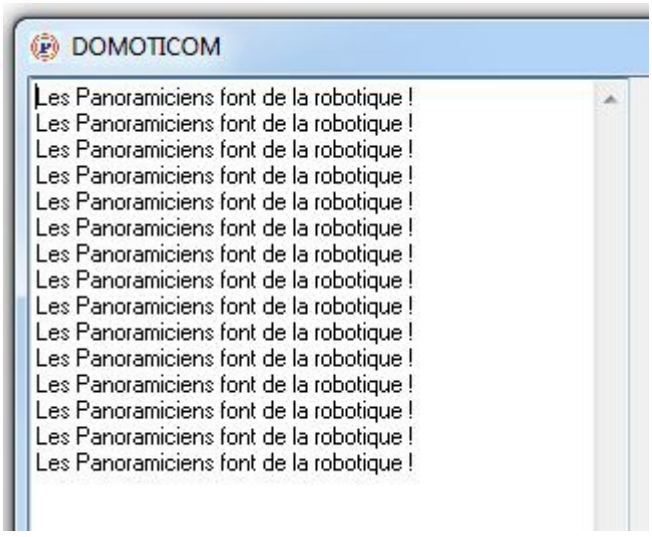
La carte Arduino exécute alors le programme téléchargé.

Dans l'éditeur DOMOTICOM, taper le programme suivant :

```
label reception

rem création d'un MEMO
memo 1
width 1,300
height 1,400
bar_vertical 1

rem création de l'objet qui gère les
liaisons série
serial 2

rem paramétrage de la liaison série
serial_port 2,4 : rem si liaison avec COM4 !
serial_baudrate 2,1200
serial_databits 2,8
serial_stopbits 2,1
serial_parity 2,0
serial_open 2

rem si réception
on_receive 2,reception

end

reception:
rem gestion de la réception d'un caractère
text 1,text$(1)+serial_read$(2)
return
```

Attention, sur votre ordinateur, la ligne **serial_port 2,4** est peut-être à modifier car elle dépend de votre configuration : la carte Arduino utilise peut-être un autre port !

On l'exécute et on obtient :



Pour la méthode séquentielle, c'est le programme qui teste si un caractère est reçu. Le programme est le suivant :

```
label reception

memo 1
width 1,300
height 1,400
bar_vertical 1

rem création de l'objet qui gère les liaisons série
serial 2

rem paramétrage de la liaison série
serial_port 2,4
serial_baudrate 2,1200
serial_databits 2,8
serial_stopbits 2,1
serial_parity 2,0
serial_open 2

reception:
rem test si réception
if count(2)<>0 then text 1,text$(1)+serial_read$(2)
pause 50
goto reception
```

Et il donne bien sûr le même résultat.

Il faut noter qu'avec la fonction **SERIAL_READ$**, TOUS les caractères reçus DEPUIS la dernière lecture sont récupérés. Ceci est très important à savoir, et c'est peut-être la seule chose important à retenir de ce tutoriel.
Car que ce soit en programmation événementielle comme en programmation séquentielle, même si le programme n'arrive pas à lire chacun des caractères au fur et à mesure de son arrivée, **aucun caractère reçu n'est perdu** et un caractère qui n'a pas été lu sera lu avec le caractère suivant.

7 - Conclusion

7.1 – Les liaisons USB et les liaisons séries

Ce tutoriel concerne la liaison série et les exemples utilisent une liaison USB !
Eh oui, et c'est ça que je voulais montrer : il existe des convertisseurs Série/USB et une liaison USB peut être alors utilisée comme une liaison série virtuelle. Et en particulier, comme ARDUINO offre des convertisseurs Série/USB, la gestion d'une telle carte revient à gérer une liaison USB exactement comme si c'était une liaison série.

7.2 – Une carte Arduino peut devenir un périphérique USB

La deuxième chose que je voulais montrer dans ce tutoriel est qu'avec une carte Arduino judicieusement programmée, on peut faire un périphérique n'utilisant qu' un seul port USB.
Pour obtenir un périphérique USB évolué de DOMOTICOM, il « suffit » de programmer un « handler » dans la carte Arduino qui traite les commandes envoyées et qui renvoie des états.
Et à partir de cela, tout est possible : l'intelligence est dans l'ordinateur qui communique par USB avec un périphérique évolué qui pilote des moteurs, des électro-aimants, des capteurs divers, des caméras, etc, etc.
Des exemples viendront…

7.3 – Où peut-on trouver DOMOTICOM ?

Un zip contenant une version alpha et limitée de DOMOTICOM, ainsi que les exemples du tutoriel (.bas et .ino) est téléchargeable à partir du répertoire DOMOTICOM de mon Webdav.