



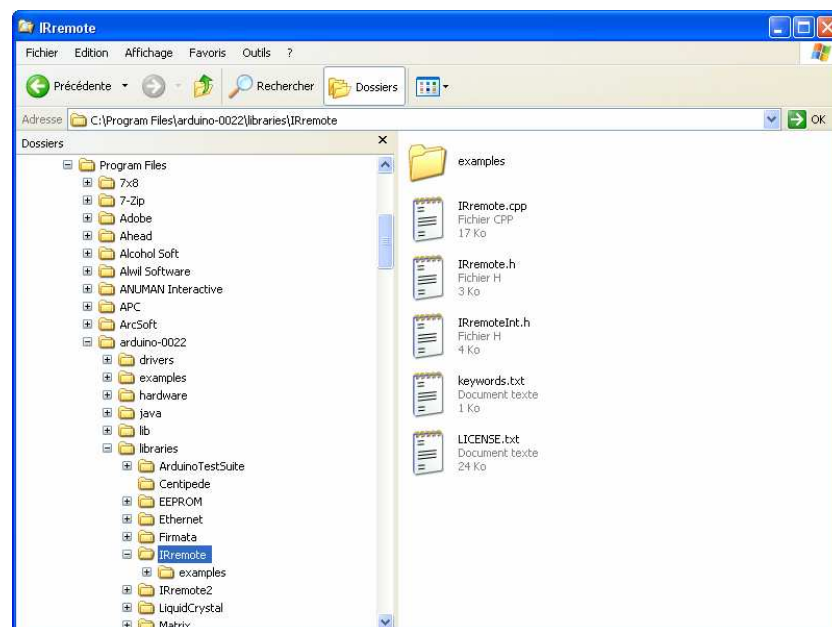
## Manuel de l'interpréteur de commande

VER 2.7 (Maj 29/12/13)

### Prérequis

Il vous faut :

- Une carte Arduino Uno ou une carte Arduino Mega. Pour se procurer ces cartes, rendez vous sur l'adresse <http://www.arduino.cc/en/Main/Buy>
- L'environnement de programmation Arduino disponible gratuitement sur <http://arduino.cc/en/Main/Software/> à télécharger et décompresser sur son disque dur ( de préférence dans le répertoire c:/Program Files
- Télécharger la librairie *IRremote* à l'adresse [www.technozone51.fr/logiciel/IRremote.zip](http://www.technozone51.fr/logiciel/IRremote.zip) et décompresser la sur votre disque dur. Puis copier le répertoire */IRremote* et son contenu dans le sous répertoire */libraries* du répertoire d'installation d'arduino comme le montre l'image ci-dessous :



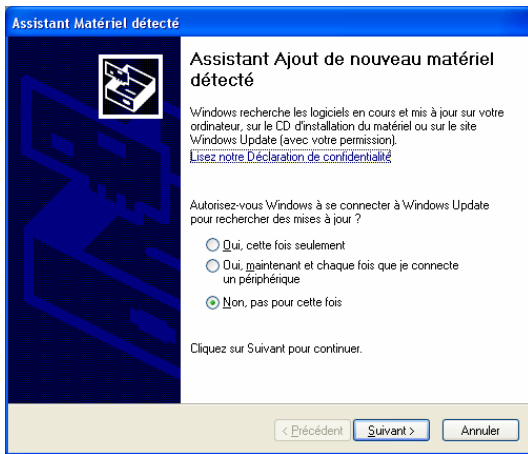
- Le programme *SHELL\_UNO.PDE* si vous avez une carte Arduino Uno ou *SHELL\_MEGA.PDE* si vous avez une carte Arduino Mega : Ces programmes sont disponibles gratuitement sur <http://techno-zone-51.fr/> dans la catégorie « logiciel ». Télécharger le programme nécessaire et sauvegarder le par exemple dans */Mes documents/arduino/*

### Pour démarrer

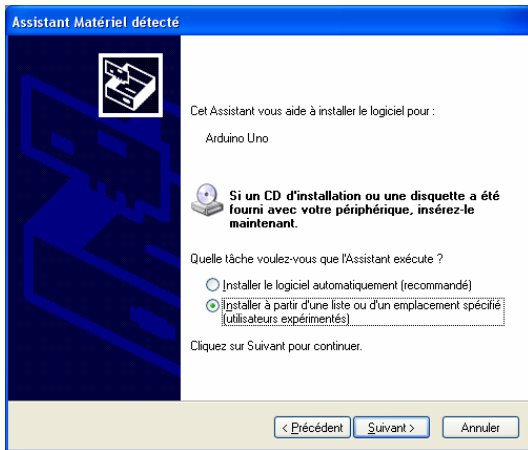
Connecter votre carte à l'aide d'un câble USB à votre ordinateur. Si c'est la première fois que vous connectez la carte, il faudra installer le driver sinon vous pouvez directement passez à la section « Programmer la carte ».

### Installer le driver

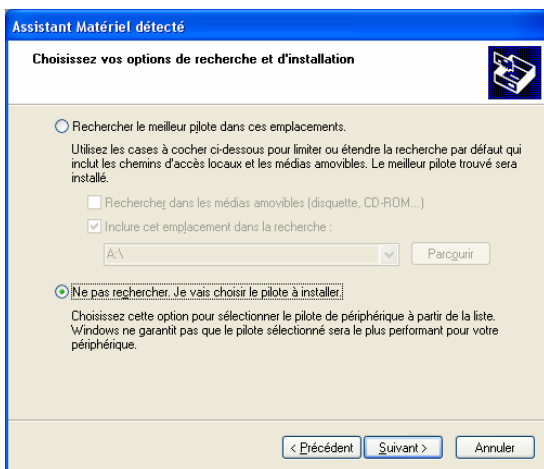
Lors de la première connexion d'une carte, l'ordinateur détecte un nouveau périphérique et tente d'installer son driver.



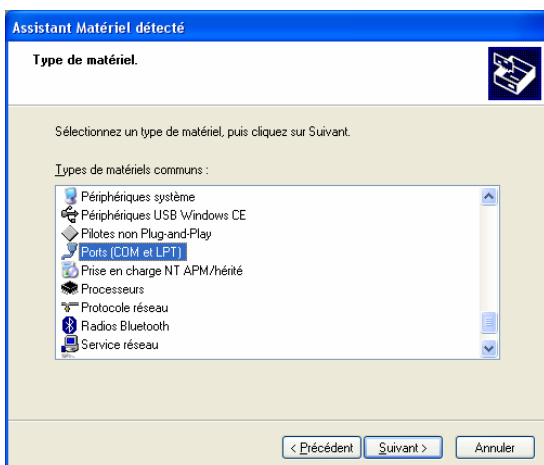
Windows demande l'autorisation pour vous connecter à Windows Update, Répondez « Non, pas pour cette fois » puis cliquez suivant.



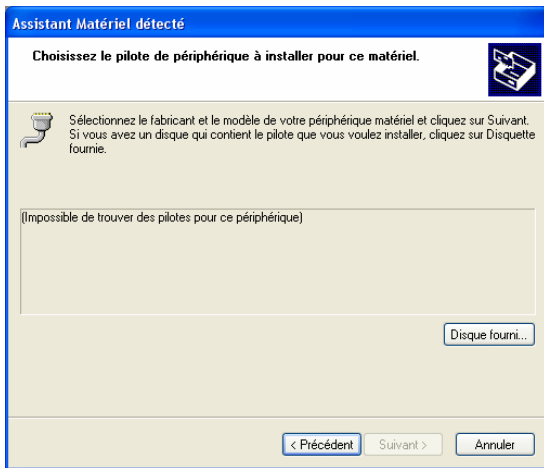
Windows demande si un CD d'installation ou une disquette a été fourni. Répondez « Installer à partir d'une liste ou d'un emplacement spécifié (utilisateurs expérimentés) » puis cliquez suivant.



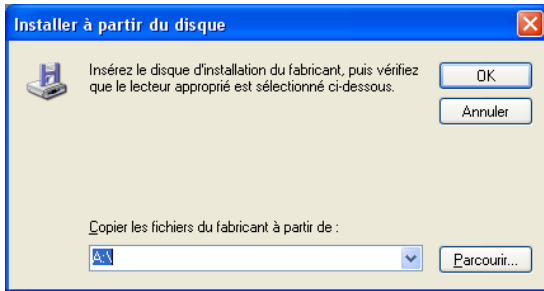
Sélectionnez ensuite « Ne pas rechercher. Je vais choisir le pilote à installer » puis cliquez suivant.



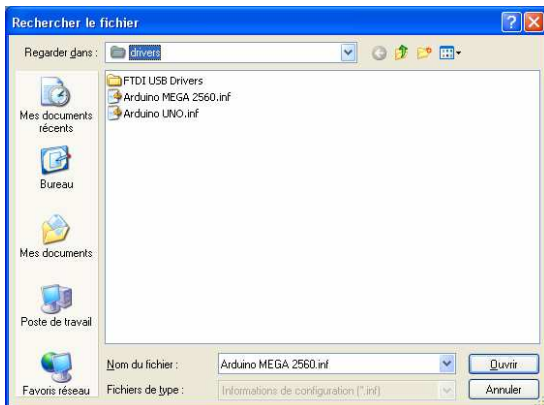
Une liste de matériel s'affiche. Parcourir la liste pour sélectionner « Ports (COM et LPT) » puis cliquez sur suivant.



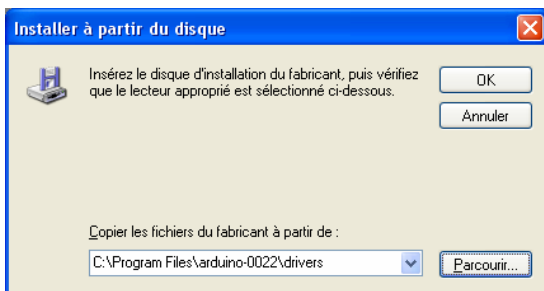
Cliquez sur « Disque fourni... »



Cliquez sur « Parcourir... »



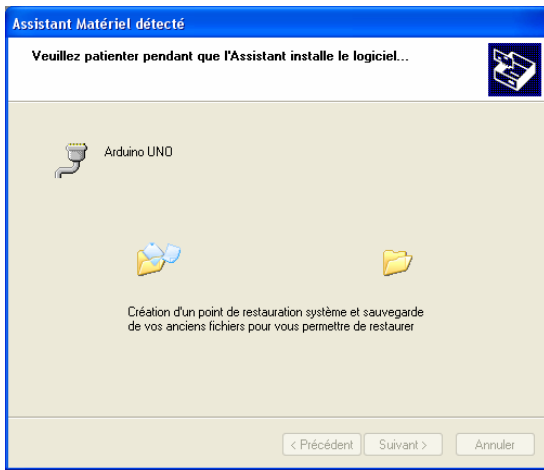
Sélectionner le répertoire dans lequel vous avez décompresser le logiciel arduino puis le sous répertoire drivers puis cliquez sur Ouvrir.



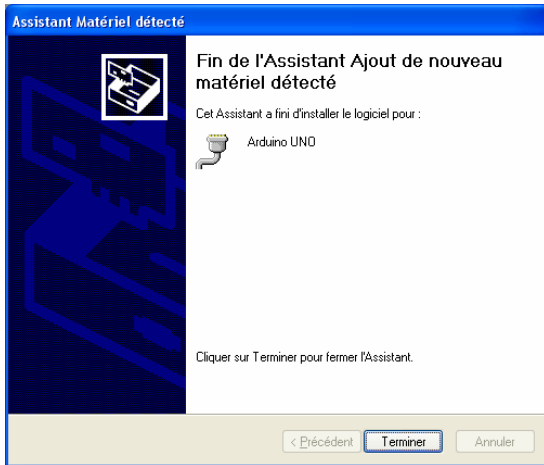
Puis cliquez sur Ok.



Cliquez sur suivant.



Windows installe le pilote. Selon les paramètres de sécurité de windows, il se peut qu'il faille appuyer une ou deux fois sur « continuer » pour passer des messages de sécurité.



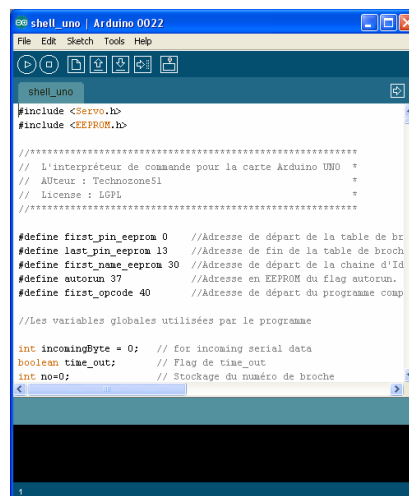
Cliquez Terminer. Votre carte Arduino est prête à être utilisée.

## Programmer la carte

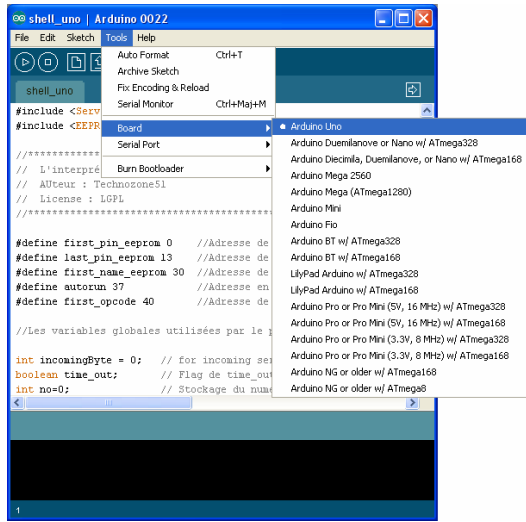
- Lancer l'environnement de programmation Arduino ( Programme Arduino.exe dans le répertoire Arduino )



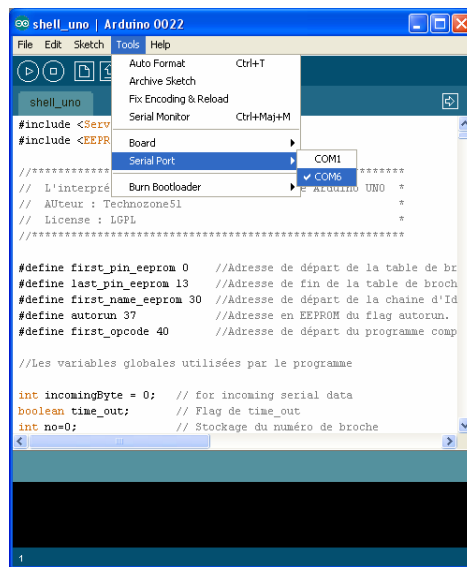
- Cliquez sur « File » puis « Open... » puis sélectionner SHELL\_UNO.PDE ou SHELL\_MEGA.PDE selon le type de carte que vous possédez.




- Cliquez sur « Tools » puis « Board » puis sélectionner la carte dont vous disposez




- Cliquez sur « Tools » puis « Serial Port » puis sélectionner le port COM sur lequel est connectée votre carte ( Si plusieurs port COM vous sont proposés, il faudra les tester un à un jusqu'à trouver lequel fonctionne ).

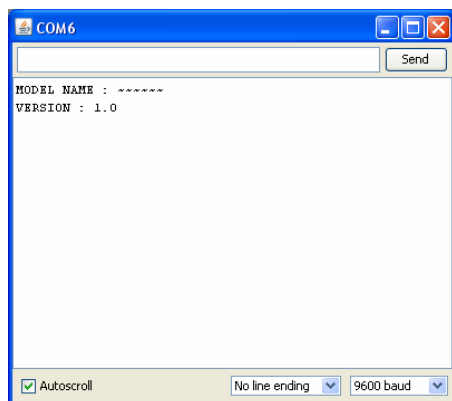


- Cliquer sur l'icône  pour Uploader le programme dans la carte. Le programme est tout d'abord compilé puis il est transféré dans la carte. Normalement le message « Done uploading » indique que tout c'est bien passé. En cas de message d'erreur, retenter l'upload en ayant pris soin de choisir un autre port COM...



## Test de l'interpréteur de commande

Dans l'environnement de programmation Arduino, cliquez sur l'icône  « Serial Monitor »



La carte répond en donnant son nom ( ici vierge ) et la version de l'interpréteur.

Voici un exemple d'une cession de communication avec la carte ( Voir la suite du guide pour une explication de chaque commande ) :

**NCOUCOU**

*La carte répond « DONE NCOUCOU » pour signaler qu'elle a configuré le nom de la carte à COUCOU. Le nom de la carte doit comporter exactement 6 caractères( compléter éventuellement celui-ci avec des espaces )*

**V**

*La carte répond en donnant son nom ( COUCOU ) et la version de l'interpréteur.*

**E131**

*La carte répond « DONE E131 » pour signaler qu'elle a bien configurée la broche 13 comme une sortie numérique.*

**Z**

*La carte se réinitialise ( pour tenir compte du changement de fonction d'une broche ) et renvoie son nom et la version de l'interpréteur.*

**W131**

*La carte répond « DONE W131 » pour signaler que la broche 13 est mis à un ( la led qui est connectée dessus doit s'allumer )*

**W130**

*La carte répond « DONE W130 » pour signaler que la broche 13 est mis à zéro ( la led qui est connectée dessus doit s'éteindre )*

## COMMANDES RECONNUS PAR L'INTERPRETEUR DE COMMANDE

L'utilisateur du logiciel ORGANIGRAM disponible sur le site [www.technozone51.fr](http://www.technozone51.fr) n'as pas besoin de connaître les commandes de l'interpréteur de commande qui suivent car le logiciel ORGANIGRAM interprète tous seul les organigrammes et les convertis de lui-même dans une série d'ordres reconnues par l'interpréteur de commande. La section suivante ne s'adresse qu'au développeurs qui souhaitent améliorer cet interpréteur de commande ou comprendre son fonctionnement.

### Gestion du Time-out

Lors de l'envoi d'une commande qui nécessite des paramètres, la carte attends de recevoir chacun des paramètres nécessaires durant un maximum de 100ms. Si ce temps est dépassé, la carte considère qu'un caractère s'est perdu et elle annule toute la commande. Dans ce cas, la carte répond **TIME OUT**

### Reset de la carte

L'envoi du caractère 'Z' provoque le reset de la carte, ceci peut être utile lorsque celle ci est bloquée dans la réception d'un ordre de lecture ou d'écriture incomplet ou pour prendre en compte le changement de fonction d'une broche.

La carte répond ensuite par sa chaîne d'identification. Par exemple :

**MODEL NAME : ASC2**

- Le plan de brochage correspond à la maquette dont l'id est ASC2

### Modification de l'Identifiant de la maquette

L'envoi de la commande 'N<cccccc>' provoque l'écriture en mémoire EEPROM du nom d'identification de la maquette. La carte répond par sa chaîne d'identification.

- <cccccc> est le nom d'identification de la maquette qui doit comporter exactement 6 caractères ( à compléter éventuellement avec des espaces ).

Par exemple : En envoyant 'NASC2 ', la carte mémorise l'ID de la maquette en mémoire EEPROM puis réponds :

**MODEL NAME : ASC2**

L'ID de la maquette est ASC2

## Identification de la carte

L'envoi du caractère 'V' provoque l'envoi de la chaîne d'identification de la carte. Contrairement à la commande 'Z', la carte n'est pas réinitialisée.

La carte répond par exemple :

**MODEL NAME : ASC2**

- Le plan de brochage correspond à la maquette dont l'id est ASC2

## Activation et désactivation des Pull-up

L'envoi de la commande **P<bb><v>** permet d'activer ou non les résistances de tirage Pull-Up de la broche concernée.

- **<bb>** est le *numéro de broche* écrit en décimal sur deux chiffres
- **<v>** est la *Valeur* qui vaut 0 pour désactiver le Pull-up et 1 pour activer le Pull-up

*Exemples :*

- L'envoi de la commande **P080** désactive le Pull-up sur la broche 8. La carte répond **DONE P080**
- L'envoi de la commande **P121** active le Pull-up sur la broche 12. La carte répond **DONE P121**

## Ecriture sur une sortie PWM

L'envoi de la commande **M<bb><vvv>** permet de commander le rapport cyclique du signal PWM sur la broche concernée.

- **<bb>** est le *numéro de broche* écrit en décimal sur deux chiffres.
- **<vvv>** est la *Valeur* du rapport cyclique compris entre 0 ( 0% toujours éteint ) et 255 ( 100% toujours allumé ), écrit en décimal sur 3 chiffres

*Exemples :*

- L'envoi de la commande **M05000** fixe le rapport cyclique à 0 sur la broche 5. La carte répond **DONE M05000**
- L'envoi de la commande **M11128** fixe le rapport cyclique à 128 ( environ 50% ) sur la broche 11. La carte répond **DONE M11128**

## Lecture d'une entrée Télécommande Infrarouge

L'envoi de la commande **I<bb>** provoque la lecture du tampon de la télécommande et renvoie la valeur stocké dans celui-ci.

- **<bb>** est le *numéro de broche* utilisé pour la télécommande Infrarouge écrit en décimal sur deux chiffres.

*Exemples :*

- L'envoi de la commande **I11** lit la valeur stockée dans le tampon de la télécommande connectée à la broche 11. La carte répond **VALUE = -1** si aucune touche n'a été appuyée. Dans le cas contraire, la carte renvoie la valeur de la touche écrite en Hexadécimal .Par Exemple **VALUE=FF5A52**
- La lecture de la valeur dans le tampon n'efface pas cette valeur, il faut donc qu'après avoir lu une valeur de touche, demander la vidange du tampon avec la commande **J**



## Vidange du tampon de la Télécommande Infrarouge

L'envoi de la commande **J** provoque la vidange du tampon de la télécommande. Cette action est nécessaire après chaque lecture d'une valeur de touche pour que le module infrarouge puisse procéder à la lecture suivante. La carte répond **DONE J**

## Ecriture sur une sortie numérique

L'envoi de la commande **W<bb><v>** permet de commander le niveau logique de la broche concernée.

- **<bb>** est le *numéro de broche* écrit en décimal sur deux chiffres.
- **<v>** est la *Valeur* qui vaut 0 pour un niveau logique bas et 1 pour un niveau logique haut

*Exemples :*

- L'envoi de la commande **W080** fixe la broche 8 au niveau bas. La carte répond **DONE W080**
- L'envoi de la commande **W101** fixe la broche 10 au niveau haut. La carte répond **DONE W101**

## Lecture d'une entrée numérique

L'envoi de la commande **R<bb>** provoque la lecture du niveau logique présent sur la broche concernée

- **<bb>** est le *Numéro de broche* écrit en décimal sur deux chiffres.

*Exemples :*

- L'envoi de la commande **R04** provoque la lecture de la broche numéro 4. La carte répond **VALUE=0** ou **VALUE=1**

## Lecture d'une entrée Analogique ( mode 8 bits )

L'envoi de la commande **A<bb>** provoque la lecture analogique sur la broche concernée

- **<bb>** est le *Numéro de broche* écrit en décimal sur deux chiffres.
- La carte renvoie une valeur comprise entre 0 et 255.

*Exemples :*

- L'envoi de la commande **A00** provoque la lecture de la broche Entrée analogique 0. La carte répond par exemple **VALUE=112**

## Lecture d'une entrée Analogique ( mode 10 bits )

L'envoi de la commande **a<bb>** provoque la lecture analogique sur la broche concernée

- **<bb>** est le *Numéro de broche* écrit en décimal sur deux chiffres.
- La carte renvoie une valeur comprise entre 0 et 1023.

*Exemples :*

- L'envoi de la commande **a00** provoque la lecture de la broche Entrée analogique 0. La carte répond par exemple **VALUE=1002**

## Commande d'un servo-moteur

L'envoi de la commande **S<bb><ppp>** provoque le positionnement du servomoteur connecté à la broche **<bb>** sur la position **<ppp>**

- **<bb>** est le *Numéro de broche* en décimal sur deux chiffres.
- **<ppp>** est la *position* en décimal sur trois chiffres compris entre 0° et 180°

*Exemples :*

- L'envoi de la commande **S04090** provoque le positionnement du servomoteur connecté à la broche 4 sur la position 90°. La carte renvoie **SERVO4=90**
- L'envoi de la commande **S11180** provoque le positionnement du servomoteur connecté à la broche 11 sur la position 180°. La carte renvoie **SERVO11=180**

## Configuration du brochage de l'interface

Cette commande permet de modifier le Plan de Brochage de la carte en fonction des besoins de la maquette. Ce Plan de Brochage est stocké en EEPROM : il est donc conservé même après coupure de l'alimentation. Au moment du redémarrage de la carte, Ce Plan de Brochage est lu pour initialiser correctement chacune des broches en fonction de l'usage que l'on souhaite en faire.

L'envoi de la commande **E<bb><s>** provoque l'écriture en EEPROM du statut de la broche

- **<bb>** est le *Numéro de broche* écrit en décimal sur deux chiffres.
- **<s>** est le *Statut* : un nombre entier qui désigne le statut de la broche selon le tableau ci-dessous :

Statut	Signification
1	Sortie Numérique
2	Entrée Numérique
3	Entrée Analogique
4	Sortie PWM
5	Commande de servo moteur
6	Entrée télécommande ( une seule télécommande par carte )
Autre	Indéfini

*Exemples :*

- L'envoi de la commande **E102** fixe la broche 10 comme étant une Entrée numérique et l'écrit en EEPROM. La carte répond **DONE E102**
- L'envoi de la commande **E045** fixe la broche 4 comme étant une commande de servomoteur l'écrit en EEPROM. La carte répond **DONE E045**

*Remarque :*

- Après une modification du brochage de la carte, il est conseillé d'envoyer une commande **<Z>** provoquant le reset de la carte pour tenir compte des changements de brochages.

## Mesure d'une distance avec le capteur Ultrason HC-SR04

Cette commande permet de mesurer une distance avec le capteur ultrason HC-SR04 et renvoie la valeur de cette mesure en cm. Une distance de 255 cm doit être considéré comme un obstacle à l'infini ( hors de porté du capteur ).

L'envoi de la commande **Q<tt><dd>** provoque une mesure ultrasonore

- **<tt>** est le *Numéro de la broche Trigger* écrit en décimal sur deux chiffres.
- **<dd>** est le *Numéro de la broche Dist ( ou Echo )* écrit en décimal sur deux chiffres.

La carte répond **VALUE=dist** ou **dist** est la distance mesurée en cm ( 255 pour un obstacle à l'infini )

*Exemples :*

- L'envoi de la commande **Q0429** provoque une mesure ultrason ( la broche de trigger étant la n°4 et la broche d'Echo étant la n°29 ). La carte renvoie par exemple **VALUE=12** si un obstacle se trouve à 12cm

## Production d'un son sur un buzzer piezoelectrique

Cette commande permet de produire un son sur un buzzer piezoelectrique.



La production du son est préemptive : c'est-à-dire qu'elle mobilise toutes les ressources du processeur pour cette unique tâche. Les programmes qui s'exécutaient en parallèle sont momentanément figés jusqu'à la fin de la production du son : Par conséquent, si plusieurs programmes doivent s'exécuter en parallèle, il faut veillez à ne produire que des sons d'une durée très brève pour ne pas figer trop longtemps les autres programmes.

L'envoi de la commande **O<bb><fff><ddd>** provoque la production d'un son de fréquence **<fff>** et d'une durée de **<ddd>** millisecondes sur la broche **<bb>**.

- **<bb>** est le *Numéro de la broche* écrit en décimal sur deux chiffres.
- **<fff>** est la fréquence en Hertz de la note à produire écrit en décimal sur quatre chiffres.
- **<ddd>** est la durée en millisecondes écrit en décimal sur quatre chiffres.

La carte répond **DONE O<bb><fff><ddd>**

*Exemples :*

- L'envoi de la commande **O2204400500** provoque la production d'une note de 440Hz et d'une durée de 500 ms sur la broche 22. La carte renvoie **DONE O2204400500** à la fin de la production du son

## Commande d'un écran LCD alphanumérique sur le port I2C

Cette commande permet de gérer un écran LCD connecté au port I2C de la carte Arduino. Le nombre de ligne et de colonne, ainsi que l'adresse I2C de l'écran peuvent être configurées afin de s'adapter à n'importe quel type d'écran LCD. La commande <L> suivi d'une sous commande permet de contrôler tous les aspects du LCD.

Format de la commande	Signification	Exemples
<b>LC</b>	Efface l'écran LCD	<b>LC</b>
<b>LSxcccraaa</b>	<p>La commande &lt;LS&gt; permet de configurer la carte Arduino en la renseignant sur la présence ou non d'un écran LCD et sur les caractéristiques de cet éventuel écran.</p> <p><b>x=0</b> si pas d'écran LCD connecté ( dans ce cas <b>cc</b>, <b>r</b> et <b>aa</b> n'ont pas d'importance )</p> <p><b>x=1</b> : un écran LCD de <b>cc</b> colonnes et de <b>r</b> lignes est connecté à la carte sur le port I2C</p> <p><b>cc</b> : le nombre de colonne de l'afficheur</p> <p><b>r</b> : le nombre de ligne de l'afficheur</p> <p><b>aaa</b> : l'adresse I2C de l'afficheur</p>	<p><b>LS1162032</b> : Un écran LCD de 2 lignes de 16 caractères est connecté. Son adresse I2C étant fixé à la valeur décimale 32 ( 20 en Hexadécimal )</p> <p><b>LS1204039</b> : Un écran LCD de 4 lignes de 20 caractères est connecté. Son adresse I2C étant fixé à la valeur Décimale 39 ( 27 en Hexadécimal )</p> <p><b>LS0000000</b> : Pas de LCD connecté</p> <p>Il faut effectuer un reset de la carte en envoyant la commande « <b>Z</b> » afin de tenir compte du changement de configuration</p>
<b>LGccr</b>	La commande <LG> Positionne le curseur à la position ( <b>cc,r</b> ) ou <b>cc</b> est le numéro de colonne sur deux chiffre et <b>r</b> le numéro de ligne	<p><b>LG000</b> : Positionne le curseur à la position (0,0)</p> <p><b>LG091</b> : Positionne le curseur à la position (9,1)</p> <p><b>LG153</b> : Positionne le curseur à la position (15,3)</p>
<b>LWssss...sss </b>	Ecrire la chaîne de caractères <b>ssss...sss</b> à partir de la position courante du curseur. La fin de la chaîne de caractères est marqué par le caractère   ( pipe)	<b>LWHello World </b> : écrit Hello World sur l'écran à partir de la position courante du curseur.
<b>LVvvvccr</b>	Ecrire la valeur <b>vvv</b> à la position ( <b>cc,r</b> ) . Avant d'écrire la valeur <b>vvv</b> , les 3 caractères à partir de la position d'affichage ( <b>cc,r</b> ) sont effacés afin que des affichages successifs d'une valeur ne se chevauchent pas !	<p><b>LV254101</b> : écrit la valeur 254 à la position (10,1)</p> <p><b>LV005020</b> : écrire la valeur 5 à la position (10,1) ( Deux espaces blanc sont également écrits ! )</p>
<b>LDh</b>	Ecrire la date contenue dans la mémoire horaire <b>h</b> à partir de la position courante du curseur. <b>h</b> est compris entre 0 et 7	<b>LD0</b> : Ecrit la date contenue dans la variable horaire n°0 à la position courante du curseur. Ecrit par exemple <b>23/03/12</b>
<b>LHh</b>	Ecrire l'heure contenue dans la mémoire horaire <b>h</b> à partir de la position courante du curseur. <b>h</b> est compris entre 0 et 7	<b>LD0</b> : Ecrit l'heure contenue dans la variable horaire n°0 à la position courante du curseur. Ecrit par exemple <b>10:32:05</b>
<b>LTvvvccr</b>	Ecrire la température correspondante à la valeur <b>vvv</b> à la position ( <b>cc,r</b> ). Cette routine est calibrée pour fonctionner avec les capteurs de température de la gamme Technozone51.	<b>LT125101</b> : écrit la température correspondante à la valeur 125 à la position (10,1)
<b>LLvvvccr</b>	Ecrire la luminosité correspondante à la valeur <b>vvv</b> à la position ( <b>cc,r</b> ). Cette routine est calibrée pour fonctionner avec les capteurs de luminosité de la gamme Technozone51.	<b>LL125101</b> : écrit la luminosité correspondante à la valeur 125 à la position (10,1)

## Commande de la gestion horaire sur le port I2C

Cette commande permet de gérer une horloge temps réel DS1307 connectée au port I2C de la carte Arduino. La commande <Y> suivi d'une sous commande permet de contrôler tous les aspects de l'horloge temps réel. L'EEPROM du DS1307 permet de stocker 8 variables horaires numérotées de 0 à 7.

Format de la commande	Signification	Exemples														
<b>YSjjmmaahhnss</b>	<p>La commande &lt;YS&gt; permet de mettre à l'heure l'horloge temps réel et démarre celle-ci. Les paramètres à transmettre sont :</p> <p><b>jj</b> : Le jour  <b>mm</b> : Le mois  <b>aa</b> : L'année  <b>hh</b> : L'heure  <b>nn</b> : Les minutes  <b>ss</b> : Les secondes</p>	<b>YS200312103120</b> règle l'horloge temps réel sur la date du 20/03/12 et l'heure sur 10:31:20. L'horloge est démarrée.														
<b>YRh</b>	<p>La commande &lt;YRh&gt; permet de lire la date et l'heure courante et stocke ces informations dans l'une des 8 variables horaires disponibles. Cette date et cette heure courante est également envoyé sur le port série. La carte réponds :</p> <p><b>VALUE=jj/mm/aa hh/mm/ss o</b></p> <p>Où <b>jj/mm/aa</b> correspond à la date courante</p> <p><b>hh/mm/ss</b> correspond à l'heure courante</p> <p><b>o</b> correspond au jour de la semaine ( 0=Dimanche, 1=Lundi, 2=mardi ... )</p> <p><b>h</b> : Le numéro de la variable horaire compris entre 0 et 7 utilisée pour stocker la date et l'heure courante.</p>	<p><b>YR0</b> : Provoque la lecture de la date et de l'heure courante et le stockage de ces données dans la mémoire horaire n°0. La carte réponds par exemple :</p> <p><b>VALUE=23/03/12 10:31:12 6</b></p>														
<b>YVhx</b>	<p>Renvoie la partie <b>x</b> de la date ou de l'heure de la variable horaire <b>h</b></p> <p><b>h</b> : Le numéro de la variable horaire ( compris entre 0 et 7 )</p> <p><b>x</b> : Précise l'information qu'il faut copier dans la variable selon le tableau ci-dessous</p> <table border="1" data-bbox="352 1666 730 1933"> <tbody> <tr> <td>x=J</td> <td>Le jour</td> </tr> <tr> <td>x=M</td> <td>Le mois</td> </tr> <tr> <td>x=A</td> <td>L'année</td> </tr> <tr> <td>x=h</td> <td>L'heure</td> </tr> <tr> <td>x=m</td> <td>Les minutes</td> </tr> <tr> <td>x=s</td> <td>Les secondes</td> </tr> <tr> <td>x=o</td> <td>Le jour de la semaine</td> </tr> </tbody> </table> <p>La carte renvoie <b>VALUE=xxx</b></p>	x=J	Le jour	x=M	Le mois	x=A	L'année	x=h	L'heure	x=m	Les minutes	x=s	Les secondes	x=o	Le jour de la semaine	<p><b>YV0o</b> : Renvoie Le jour de la semaine de la variable horaire n°0. La carte réponds par exemple :</p> <p><b>VALUE=1</b> si le jour est lundi</p> <p><b>YV6m</b> : Renvoie les minutes de la variable horaire n°6. La carte réponds par exemple :</p> <p><b>VALUE=42</b></p> <p><b>YV1M</b> : Renvoie le mois de la variable horaire n°1. La carte réponds par exemple :</p> <p><b>VALUE=2</b> ( si le mois est Février )</p>
x=J	Le jour															
x=M	Le mois															
x=A	L'année															
x=h	L'heure															
x=m	Les minutes															
x=s	Les secondes															
x=o	Le jour de la semaine															
<b>YCh<sub>1</sub>h<sub>2</sub>h<sub>3</sub></b>	<p>Calcule la variable horaire n° <b>h<sub>3</sub></b> comme le temps écoulé entre la variable horaire n° <b>h<sub>1</sub></b> et n° <b>h<sub>2</sub></b> :</p> <p><b>h<sub>3</sub>=h<sub>2</sub>-h<sub>1</sub></b></p>	<b>YC012</b> : Calcule le temps écoulé entre la variable horaire n°0 et n°1 et stocke le résultat dans la variable horaire n°2														

## Commande de composants divers sur le bus I2C

Cette commande permet de gérer tout type de composants à interface I2C connecté au port I2C de la carte Arduino. La commande <i> suivi d'une sous commande permet de contrôler tous les aspects de la communication avec un composant qui respecte le protocole I2C.

Format de la commande	Signification	Exemples
<b>iW</b> aaa <sup>n</sup> <b>d</b> <sub>1</sub> <b>d</b> <sub>1</sub> <b>d</b> <sub>2</sub> <b>d</b> <sub>2</sub> ... <b>d</b> <sub>n</sub> <b>d</b> <sub>n</sub>	La commande <iW> permet d'envoyer des données vers un composant I2C. Les paramètres à transmettre sont : <b>aaa</b> : L'adresse 7bits du composant I2C <b>nn</b> : Le nombre d'octets que l'on souhaite envoyer au composant. <b>d<sub>1</sub>d<sub>1</sub>d<sub>1</sub></b> : La donnée n°1 <b>d<sub>2</sub>d<sub>2</sub>d<sub>2</sub></b> : La donnée n°2 <b>d<sub>n</sub>d<sub>n</sub>d<sub>n</sub></b> : La dernière donnée	<b>iW10401000</b> envoie <b>1</b> octet de valeur <b>0</b> au composant I2C d'adresse 7bits <b>104</b> .  <b>iW10403010200255</b> envoie <b>3</b> octets de valeur <b>10,200</b> et <b>255</b> au composant I2C d'adresse 7bits <b>104</b> .
<b>iR</b> aaa <sup>n</sup>	La commande <iR> permet de lire des données en provenance d'un composant I2C. Les paramètres à transmettre sont : <b>aaa</b> : L'adresse 7bits du composant I2C <b>nn</b> : Le nombre d'octets que l'on souhaite lire depuis le composant. La carte réponds en renvoyant le nombre d'octets demandé :  <b>VALUE=</b> valeur <sub>1</sub> ,valeur <sub>2</sub> ,...valeur <sub>n</sub>	<b>iR10410</b> : Provoque la lecture de <b>10</b> octets depuis le composant I2C d'adresse 7bits <b>104</b> .  La carte réponds par exemple :  <b>VALUE=88,87,17,2,4,17,19,3,15,3</b>

## Modification de la vitesse de communication avec la carte Arduino

La commande **b<xx>** permet de fixer la vitesse de communication avec la carte Arduino selon le paramètre <xx> comme le montre le tableau ci dessous.

Paramètre <xx>	Vitesse de communication en bauds
00	300 bds
01	9600 bds <i>car la vitesse de 600 bds n'est pas supportés par le logiciel Arduino*</i>
02	1200 bds
03	2400 bds
04	4800 bds
05	9600 bds
06	14400 bds
07	19200 bds
08	28800 bds
09	38400 bds
10	57600 bds
11	115200 bds

Ce paramètre prends effet immédiatement : La réponse de la carte ne peut donc plus être reçu correctement si le logiciel qui communique avec la carte Arduino ne change pas sa vitesse de communication de manière adéquate. Le paramètre de vitesse est également stocké dans la mémoire EEPROM de la carte Arduino pour être automatiquement restauré à chaque redémarrage de la carte Arduino.

*\*Afin d'éviter une perte de communication irréversible depuis le moniteur série du logiciel Arduino, il n'a pas été implémenté la vitesse de 600 bauds car non supporté par ce dernier !*

## Envoie d'un programme compilé dans un slot mémoire EEPROM

La mémoire est divisée en 33 slots numérotés de 0 à 32.

Le slot n°0 qui pointe vers la mémoire EEPROM interne de la carte Arduino est toujours disponible.

L'activation ou non des slots n°1 à 32 dépend des mémoires externes installées sur les cartes EASYCON1 et EASYCON2. Le tableau ci-dessous récapitule les slots disponibles en fonction des composants installés sur les différentes BANK mémoire des cartes EASYCON1 et EASYCON2.

Par exemple, en installant un composant 24C128 sur l'emplacement BANK0 de la carte EASYCON1, les slots 1 à 4 deviennent disponibles.

Les commandes **F<SS>** et **B<b>** permettent de charger un programme compilé dans un slot de mémoire EEPROM. Celui-ci pourra s'exécuter de manière autonome sans ordinateur connecté à la carte. Un programme compilé est une suite d'octets appelés OPCODE. Le déchiffrement des OPCODE peut se faire à l'aide du tableau des OPCODE dans la section « Liste des opcodes reconnues » à la fin de ce document.

L'envoi de la commande **F<SS>** provoque l'entrée dans le mode chargement d'un programme compilé. Le compteur de programme s'initialise au début du slot mémoire spécifié dans l'EEPROM. L'interface attend alors de recevoir un octet du programme. **<SS>** spécifie le slot mémoire « cible » du programme selon le tableau ci-dessous.

Numéro de slot mémoire <SS>	Cible dans laquelle le programme est chargé	24C32	24C64	24C128	24C256
0	Dans l'EEPROM interne de la carte arduino	•	•	•	•
1	Dans l'EEPROM Externe ( Bank 0 carte Easycon1)	•	•	•	•
2	Dans l'EEPROM Externe ( Bank 0 carte Easycon1)		•	•	•
3	Dans l'EEPROM Externe ( Bank 0 carte Easycon1)			•	•
4	Dans l'EEPROM Externe ( Bank 0 carte Easycon1)			•	•
5	Dans l'EEPROM Externe ( Bank 0 carte Easycon1)				•
6	Dans l'EEPROM Externe ( Bank 0 carte Easycon1)				•
7	Dans l'EEPROM Externe ( Bank 0 carte Easycon1)				•
8	Dans l'EEPROM Externe ( Bank 0 carte Easycon1)				•
9	Dans l'EEPROM Externe ( Bank 1 carte Easycon1)	•	•	•	•
10	Dans l'EEPROM Externe ( Bank 1 carte Easycon1)		•	•	•
11	Dans l'EEPROM Externe ( Bank 1 carte Easycon1)			•	•
12	Dans l'EEPROM Externe ( Bank 1 carte Easycon1)			•	•
13	Dans l'EEPROM Externe ( Bank 1 carte Easycon1)				•
14	Dans l'EEPROM Externe ( Bank 1 carte Easycon1)				•
15	Dans l'EEPROM Externe ( Bank 1 carte Easycon1)				•
16	Dans l'EEPROM Externe ( Bank 1 carte Easycon1)				•
17	Dans l'EEPROM Externe ( Bank 2 carte Easycon2)	•	•	•	•
18	Dans l'EEPROM Externe ( Bank 2 carte Easycon2)		•	•	•
19	Dans l'EEPROM Externe ( Bank 2 carte Easycon2)			•	•
20	Dans l'EEPROM Externe ( Bank 2 carte Easycon2)			•	•
21	Dans l'EEPROM Externe ( Bank 2 carte Easycon2)				•
22	Dans l'EEPROM Externe ( Bank 2 carte Easycon2)				•
23	Dans l'EEPROM Externe ( Bank 2 carte Easycon2)				•
24	Dans l'EEPROM Externe ( Bank 2 carte Easycon2)				•
25	Dans l'EEPROM Externe ( Bank 3 carte Easycon2)	•	•	•	•

26	Dans l'EEPROM Externe ( Bank 3 carte Easycon2)		•	•	•
27	Dans l'EEPROM Externe ( Bank 3 carte Easycon2)			•	•
28	Dans l'EEPROM Externe ( Bank 3 carte Easycon2)			•	•
29	Dans l'EEPROM Externe ( Bank 3 carte Easycon2)				•
30	Dans l'EEPROM Externe ( Bank 3 carte Easycon2)				•
31	Dans l'EEPROM Externe ( Bank 3 carte Easycon2)				•
32	Dans l'EEPROM Externe ( Bank 3 carte Easycon2)				•

Ce programme restera en mémoire même après la coupure de courant. Le programme peut, si on le souhaite être exécuté de manière automatique à chaque démarrage de la carte. La réception se fait Octet par Octet grâce à la commande **B<b>** qui stocke l'octet <b> dans la cible sélectionnée précédemment puis incrémente le compteur de programme d'un octet . ( L'interface acquitte chaque octet reçu en renvoyant **DONE B<b>** ).

*Exemples :*

- L'envoi de la commande **F00** provoque l'entré en mode programmation et initialise le compteur de programme sur le slot mémoire n°0 qui se trouve dans l'eprom interne de la carte arduino. La carte est prête a recevoir les octets du programme. La carte renvoie "**DONE F00**"
- L'envoi de la commande **B#60** provoque l'envoi et le stockage de la valeur 60 dans l'eprom sélectionnée précédemment. Le compteur de programme est ensuite incrémenté de 1. La carte renvoie "**DONE B60**". Elle est prête à recevoir l'octet suivant.

*Remarque : #60 désigne le caractère de code ascii 60*

## Enregistrement du nom d'un programme stocké dans un slot mémoire EEPROM

L'envoi de la commande **H<SS><NNNNNNNN>** provoque le stockage du nom <NNNNNNNN> ( 8 caractères alphanumériques complétés éventuellement d'espaces ) pour le slot <SS>

*Exemples :*

- L'envoi de la commande « **H01PORTAIL** » indique que le programme s'appelle « PORTAIL » dans le slot n° 1. La carte renvoie le message "**DONE H01PORTAIL** ”.

## Activation ou désactivation d'un programme stocké dans un slot mémoire EEPROM

L'envoi de la commande **G<SS><V>** permet d'activer ou non le programme qui se trouve dans le slot mémoire <SS>. La valeur <V> spécifie si il s'agit d'activer ( 1 ) ou de désactiver ( 0 ) le programme.

Un programme désactivé n'est pas effacé de la mémoire, mais son exécution est arrêtée.

*Exemples :*

- L'envoi de la commande « **G001** » active le programme du slot n°0
- L'envoi de la commande « **G000** » désactive le programme du slot n°0 ( son exécution est arrêté )
- L'envoi de la commande « **G011** » active le programme du slot n°1
- L'envoi de la commande « **G320** » désactive le programme du slot n°32 ( son exécution est arrêté )



## Lecture des informations relatives à un slot mémoire EEPROM

L'envoi de la commande **K<SS>** permet de lire les informations relatives au slot n° <SS>. L'interpréteur réponds **VALUE=NNNNNNNN :V** où **NNNNNNNN** est le nom du programme stocké dans le slot et **V** est la valeur 1 ou 0 pour préciser si le slot est actif ou non.

*Exemples :*

- L'envoi de la commande « **K00** » permet de lire le nom du programme stocké dans le slot n°0 ainsi que son état actif ou non. La carte réponds par exemple **VALUE=LED CLIG:1** si le slot contient un programme ayant pour nom **LED CLIG** et si celui-ci est actif.

## Exécution des programmes stockés dans les slots mémoires EEPROM actifs

L'envoi de la commande **X** provoque l'interprétation des programmes stockés dans les slots mémoires EEPROM. Seul les programmes stockés dans les slots marqués comme actif sont exécutés. L'interpréteur est arrêté si une commande se présente sur le port série de la carte provoquant l'arrêt de tous les programmes des slots actifs.

On peut exécuter jusqu'à 33 programmes en parallèles ( slot n°0 à slot n°32 ). L'interpréteur est réellement multitâche. Si le programme qui s'exécute dans le slot n°0 effectue une temporisation de 5 secondes, cela n'aura aucune incidence sur les programmes des autres slots actifs qui s'exécuteront tout à fait normalement en même temps.

*Exemples :*

- L'envoi de la commande **X** provoque l'exécution des programmes stockés dans les slots actifs. La carte renvoie le message "**RUNNING PROGRAM**" avant de lancer l'interpréteur. Si l'utilisateur envoie à la carte une commande ( par exemple la commande "**V**" ), l'exécution de l'interpréteur sera interrompue , tous les programmes s'arrêtent et la commande « **V** » sera exécuté.

## Activation / Désactivation de l'AUTORUN

L'envoi de la commande **T1** provoque l'activation de l'AUTORUN. L'interprétation des programmes compilés et stockés dans les slots marqués actifs démarrent automatiquement à chaque reset ou mise sous tension de la carte. L'envoi de la commande **T0** désactive l'AUTORUN.

*Exemples :*

- L'envoi de la commande **T1** active l'autorun. La carte réponds « **DONE T1** »
- L'envoi de la commande **T0** désactive l'autorun. La carte réponds « **DONE T0** »

## Activation / Désactivation du debugging

L'envoi de la commande **D1** provoque l'activation du « mode debug ». **D0** provoque sa désactivation. Lorsque qu'un programme compilé en EEPROM est exécuté, on peut suivre son déroulement pas à pas dans la console série à condition que le « mode debug » soit actif. Cette fonction n'est utilisée que durant la mise au point de l'interpréteur de commande.

## Liste des OPCODES reconnues

Le programme stockée dans l'EEPROM de sauvegarde l'est sous une forme compressée nommée OPCODE. Celui-ci peut être exécuté par l'interpréteur de manière totalement autonome. Les OPCODE reconnus par l'interpréteur de la carte sont regroupés dans le tableau ci-dessous :

n° d'opcode	Format mémoire	Signification
0	0	Ne rien faire
1	1mvv	Affecte la valeur v ( 0 à 65025 ) à la variable m ( 1 à 26 )
2	2m	Incrémente la variable m ( 1 à 26 )
3	3m	Décrémente la variable m ( 1 à 26 )
4	4bm	Lire l'entrée analogique b ( 0 à 255 ) et stocké le résultat dans la variable m ( 1 à 26 )
5	5bm	Fixe le rapport cyclique d'une broche PWM b ( 0 à 255 ) avec la valeur contenue dans la variable m ( 1 à 26 )
6	6bv	Fixe le rapport cyclique d'une broche PWM b ( 0 à 255 ) avec la valeur v ( 0 à 255 )
7	7bv	Sort la valeur numérique v ( 0 ou 1 ) sur la broche b ( 0 à 255 )
8	8bv	Positionne le servo moteur connecté à la broche b ( 0 à 255 ) à la position v ( 0 à 255 )
9	9bm	Positionne le servo moteur connecté à la broche b ( 0 à 255 ) à la valeur contenu dans la variable m ( 1 à 26 )
10	10bvoo	Lit la valeur numérique sur la broche b ( 0 à 255 ) et teste l'égalité avec v ( 0 ou 1 ). Dans le cas ou le test est positif, le programme saute à l'instruction qui se situe à oo octets de l'opcode actuel. oo est donc une adresse relative appelé offset. Si la condition n'est pas remplie, on passe à l'opcode qui suit
11	11mvvoo	compare la valeur de la variable m ( 1 à 26 ) avec la valeur de référence v ( 0 à 65535 ). Dans le cas ou il y a égalité, le programme saute à l'instruction qui se situe à oo octets de l'opcode actuel. oo est donc une adresse relative appelé offset. Si la condition n'est pas remplie, on passe à l'opcode qui suit
12	12mvvoo	compare la valeur de la variable m ( 1 à 26 ) avec la valeur de référence v ( 0 à 65535 ). Dans le cas ou $m < vv$ , le programme saute à l'instruction qui se situe à oo octets de l'opcode actuel. oo est donc une adresse relative appelé offset. Si la condition n'est pas remplie, on passe à l'opcode qui suit
13	13mvvoo	compare la valeur de la variable m ( 1 à 26 ) avec la valeur de référence v ( 0 à 65535 ). Dans le cas ou $m > vv$ , le programme saute à l'instruction qui se situe à oo octets de l'opcode actuel. oo est donc une adresse relative appelé offset. Si la condition n'est pas remplie, on passe à l'opcode qui suit
14	14mvvoo	compare la valeur de la variable m ( 1 à 26 ) avec la valeur de référence v ( 0 à 65535 ). Dans le cas ou $m \square vv$ , le programme saute à l'instruction qui se situe à oo octets de l'opcode actuel. oo est donc une adresse relative appelé offset. Si la condition n'est pas remplie, on passe à l'opcode qui suit
15	15mvvoo	compare la valeur de la variable m ( 1 à 26 ) avec la valeur de référence v ( 0 à 65535 ). Dans le cas ou $m \geq vv$ , le programme saute à l'instruction qui se situe à oo octets de l'opcode actuel. oo est donc une adresse relative appelé offset. Si la condition n'est pas remplie, on passe à l'opcode qui suit
16	16mvvoo	compare la valeur de la variable m ( 1 à 26 ) avec la valeur de référence v ( 0 à 65535 ). Dans le cas ou $m \diamond vv$ , le programme saute à l'instruction qui se situe à oo octets de l'opcode actuel. oo est donc une adresse relative appelé offset. Si la condition n'est pas remplie, on passe à l'opcode qui suit

17	17vv	Le programme est suspendu pour une temporisation de vv millisecondes ( 0 à 65535 )														
18	18aa	Le programme saute à l'opcode situé à l'adresse absolue aa ( 1 à 65535 )														
19	19vvvvvvvvvoo	Lit l'entrée de la télécommande et teste l'égalité avec v ( chaîne de 8 caractères complété avec des espaces ). Dans le cas ou le test est positif, le programme saute à l'instruction qui se situe à oo octets de l'opcode actuel. oo est donc une adresse relative appelé offset. Si la condition n'est pas remplie, on passe à l'opcode qui suit														
20	20	Vidange du tampon de la télécommande														
21	21	Efface l'écran LCD														
22	22cr	Positionne le curseur du LCD à la position (c,r)														
23	23sss...ss	Ecrire à partir de la position courante du curseur sur l'écran LCD la chaîne de caractères ssss...ss. Le caractère   ( pipe ) indiquant la fin de la chaîne														
24	24mcr	Ecrire sur le LCD la valeur de la mémoire m à la position (c,r). Les 3 caractères à partir de la position (c,r) sont effacés avant d'écrire la valeur de la variable														
25	25vcr	Ecrire sur le LCD la valeur v à la position (c,r). Les 3 caractères à partir de la position (c,r) sont effacés avant d'écrire la valeur v														
26	26h	Ecrire la date sur le LCD au format JJ/MM/AA à partir de la variable horaire n°h														
27	27h	Ecrire l'heure sur le LCD au format hh:mm:ss à partir de la variable horaire n°h														
28	28h	Lire la date et l'heure courante de l'horloge temps réel et la copier dans la mémoire horaire n°h ( compris entre 0 et 7 )														
29	29hxv	Copie la partie x de la variable horaire n°h dans la variable v														
30	30h <sub>1</sub> h <sub>2</sub> h <sub>3</sub>	Calcul la durée séparant la variable horaire h <sub>1</sub> et h <sub>2</sub> et stocke le résultat dans la variable horaire h <sub>3</sub>														
31	31mcr	Positionne le curseur de l'écran LCD à la position (c,r) et écris la valeur de la température dans la mémoire m en convertissant en °C														
32	32vcr	Positionne le curseur de l'écran LCD à la position (c,r) et écris la valeur de la température contenu dans v en convertissant en °C														
33	33mcr	Positionne le curseur de l'écran LCD à la position (c,r) et écris la valeur de la luminosité dans la mémoire m en convertissant en Lux														
34	34vcr	Positionne le curseur de l'écran LCD à la position (c,r) et écris la valeur de la luminosité contenu dans v en convertissant en Lux														
35	35bffdd	Produit un son sur la broche b de fréquence ff et de durée dd. Cette commande est préemptive et fige l'exécution en parallèle des programmes actifs jusqu'à la fin de la production du son.														
36	36tdm	Lit la distance à l'aide d'un capteur ultrasonore HC-SR04 connecté à la broche trigger (t) et à la broche distance ou echo (d) et stocke la valeur de la distance en cm dans la variable m														
37	37mntoo	Effectuer un test entre les deux variable m et n . Dans le cas ou le test est vrai , le programme saute à l'instruction qui se situe à oo octets de l'opcode actuel. oo est donc une adresse relative appelé offset. Si la condition n'est pas remplie, on passe à l'opcode qui suit. On choisit le test à effectuer à l'aide du paramètre t comme indiqué dans le tableau ci-dessous : <table border="1" data-bbox="758 1859 1189 2128"> <thead> <tr> <th>Parametre t</th> <th>Le test effectué</th> </tr> </thead> <tbody> <tr> <td>=</td> <td>m = n</td> </tr> <tr> <td>&gt;</td> <td>m &gt; n</td> </tr> <tr> <td>&lt;</td> <td>m &lt; n</td> </tr> <tr> <td>s</td> <td>m &gt;= n</td> </tr> <tr> <td>i</td> <td>m &lt;= n</td> </tr> <tr> <td>!</td> <td>m &lt;&gt; n</td> </tr> </tbody> </table>	Parametre t	Le test effectué	=	m = n	>	m > n	<	m < n	s	m >= n	i	m <= n	!	m <> n
Parametre t	Le test effectué															
=	m = n															
>	m > n															
<	m < n															
s	m >= n															
i	m <= n															
!	m <> n															

38	38lmpn	<p>Effectue une opération arithmétique sur les variable m et n et stocke le résultat dans la variable l. p est le paramètre qui indique quelle opération l'on souhaite effectuer selon le tableau ci-dessous :</p> <table border="1" data-bbox="715 232 1238 421"> <thead> <tr> <th data-bbox="715 232 912 271">Paramètre p</th> <th data-bbox="912 232 1238 271">L'opération effectuée</th> </tr> </thead> <tbody> <tr> <td data-bbox="715 271 912 309">+</td> <td data-bbox="912 271 1238 309"><math>l=m+n</math></td> </tr> <tr> <td data-bbox="715 309 912 347">-</td> <td data-bbox="912 309 1238 347"><math>l=m-n</math></td> </tr> <tr> <td data-bbox="715 347 912 385">*</td> <td data-bbox="912 347 1238 385"><math>l=m*n</math></td> </tr> <tr> <td data-bbox="715 385 912 421">/</td> <td data-bbox="912 385 1238 421"><math>l=m/n</math></td> </tr> </tbody> </table>	Paramètre p	L'opération effectuée	+	$l=m+n$	-	$l=m-n$	*	$l=m*n$	/	$l=m/n$
Paramètre p	L'opération effectuée											
+	$l=m+n$											
-	$l=m-n$											
*	$l=m*n$											
/	$l=m/n$											
255	255	Arrête l'exécution du programme, La carte retourne en attente d'une nouvelle commande										