http://www.didel.com/ **info@didel.com**

www.didel.com/wbot2/Wb2Debut.pdf

Wellbot2 Pinguino/C
Mise en route et premières expériences

-- en travail – ne s'adresse pas à tout débutants -- voir Kidules,
-- approche Arduino voir Wellbot2

Pour se familiariser avec le C, les Kidules, le Wellbot2, la carte Kicar et la carte Picstar sont compatibles Pinguino et offrent un environnement qui tourne sur PC et Mac, facile à installer et utiliser. Des documentations spécifiques couvrent ces différents systèmes.
Pour celui qui ne sait rien sur le C et les microcontrôleurs, les Kidules permettent de se familiariser très progressivement à la programmation. La carte Picstar permet des robots très performants,
La présente documentation suppose que les connaissance de bases sont acquises, ou que le lecteur saura aller chercher sur la documentation Kidule, sur internet et sur des forums la réponse à ses questions (voir www.didel.com/Forum.html)
Le Wellbot2 est configuré pour un groupe d'application et le but de ce document est de comprendre comment programmer les ressources à disposition, et comment utiliser les librairies qui supportent ces ressources pour permettre des applications intéressantes avant de passer sur un robot plus performant permettant des application complexes.

Installation de Pinguino et premier programme
Vous devez installer Pinguino, ce qui dépends de votre PC
Windows XP `www.didel.com/PinguinoInstallXP.pdf`
Windows 7 `www.didel.com/PinguinoInstallW7.pdf`
Vista - pas possible actuellement
MacOS-X `http://sites.google.com/site/pinguinotutorial/installation/mac-osx`
Linux `http://jpmandon.blogspot.com/search/label/Install%20Pinguino%20IDE`

Il faut ensuite se familiariser avec Pinguino, ce qui est détaillé avec copies d'écran au début de www.didel.com/kits/KiPing1.pdf.
La photo ci-contre montre l'emplacement du poussoir Reset et de la LED bleue qui est en principe éteinte pendant que le chargeur (bootloader) attend le transfert du programme.
Il faut utiliser comme programme le fichier `Wb2Bip.pde` qui se trouve avec les autres programmes du Wellbot2 sous www.didel.com/wbot2/Wbot2Soft.zip
Ce programme sera expliqué plus loin.

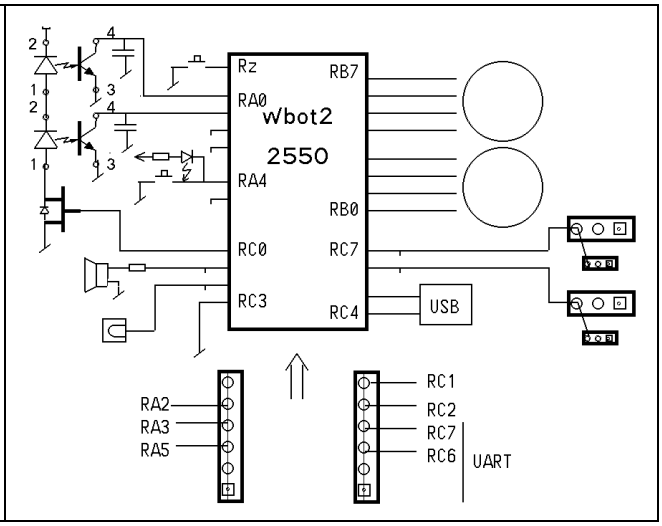
Structure des programmes
Un programme doit dire au processeur ce qui est entrée et sortie, c'est le ''set-up''. Il répète ensuite dans une ''loop'' toujours la même chose, ce qui peut être très simple comme ce premier programme qui bippe sur le haut-parleur.

Ce programme suit pour la définition des entrées-sorties l'approche Arduino, qui a été reprise pour Pinguino, mais que nous abandonnerons rapidement, car peu efficace.
Le bits du processeur sont numérotés (sauf la LED bleue) et dans le setup, on définit (en respectant les min/maj) si la pin est en entrée ou sortie.
Ce programme ne fait un son que si on appuie sur le poussoir.

```
//Wb2Son.pde son sur le HP
#include "Wb2Def.h" // déclarations et set-up
u16 periode = 500 ; // pour 1 kHz
void loop ()
{
    if (PousProc == PousOn)
    {
    HP = ! HP ; // membrane attirée ou relâchée
    delayMicroseconds (periode) ;
    }
    // else on ne fait rien, pas besoin de le dire
}
```

Set-up

Le ‘’set-up’’ dit au compilateur quelles sont les entrées et sorties utilisées par le programme.
Il faut connaître le ‘’plan’’ de sa carte et définir les entrées sorties
Les entrées n’ont pas besoin d’être déclarées, c’est leur état par défaut, mais la documentation est plus claire si on le fait. La seule entrée est le capteur infrarouge IRM.
L’entrée poussoir, associée à la LED bleu, n’a pas de numéro de pin.
Toutes les définitions et le set-up sont dans le fichier Wb2Def.h, qui est inséré au début de chaque programme.



Le Wellbot2 convient bien pour jouer avec des sons, et c’est un bon moyen de vérifier que l’on maîtrise les structures de base du C. Les capteurs de distance et les moteurs pas-à-pas ne se gèrent pas correctement en agissant sur un bit à la fois. Voir les docs spécifiques :
www.didel.com/wbot2/Wb2PasAPas.pdf et www.didel.com/wbot2/Wb2IrDist.pdf .
Les connecteurs servos, Af2x8c, Bico64 feront l’objet de documentation ultérieures, si intérêt.

Remarque concernant les ports

Le compilateur C SDCC est sous-jascent au Pinguino, et les ports peuvent être vus comme ils sont : des registres 8bits associés à des registres de direction, pour lesquels un bit 0 indique une sortie (Output).
Le câblage du microcontrôleur se traduit donc par les déclarations ci-contre.
Il y a beaucoup à expliquer pour celui qui n’est pas familiarisé avec les microcontrôleurs, des explications plus précises seront données dans les documents spécifiques.

```
TRISA  = 2b000011 ; on peut
allumer la LED bleue
; ou TRISA = 2b100011 ; on peut
lire le poussoir
 bIrDD = 0  ; IrDistDroite
 bIrDG = 1  ; IrDistGauche
 ; pins 2 3 et 5 extensions
 bLedBleue = 4
  bPoussoir = 4
TRISB  = 2b00000000 ; moteurs
TRISC  = 2b00000100 ; selon
connecteurs
 bTrEcl = 0
 bHP   = 1
 bIRM  = 2
; pins 6 7 extensions
```

Structures du C

Le haut-parleur permet de vérifier facilement que l’on maîtrise les structures de base du C.
Elles sont expliquées dans de nombreux documents faciles à trouver sur internet, et pour un tout-débutant, les Kidules ont été développés pour permettre une approche progressive et ludique (voir www.didel.com/kidules/Liens.pdf).
Pour vérifier vos connaissances, modifiez ces programmes et créez des variantes.
Le programme `Wd2Sons0.pde` montre une utilisation du `while`
Le programme `Wd2Sons1.pde` montre une utilisation du `while`
Le programme `Wd2Sons2.pde` montre une utilisation du `if - else`
Le programme `Wd2Sons3.pde` montre une utilisation du `for`
Le programme `Wd2Sons4.pde` montre une utilisation du `for`
Le programme `Wd2Sons5.pde` montre une utilisation d’un `array (tableau)`
Le programme `Wd2Sons6.pde` montre une utilisation du `case switch`
Modifiez ces programmes et créez des variantes.
Les structures sont bien expliquées dans les documents Kidules C, et pour plus de détails, on peut chercher en français sous http://arduino.cc/fr/Main/Reference

Lire le poussoir

Le poussoir du Wellbot2 est lié à la LED bleue qui fait pull-up. Quand on pèse sur le poussoir qui est juste à côté, on allume nécessairement cette Led bleue. Le bouton reset agit aussi sur cette LED

bleue qui est éteinte pendant que le Wellbot2 attend le téléchargement éventuel. Au bout de 4 secondes, si un nouveau programme n'a pas été chargé, le chargeur allume cette LED (que le programme qui démarre peut éteindre immédiatement).
Cette LED et poussoir n'est pas une ligne avec un numéro Pinguino. Il faut la déclarer comme le bit RA4 et dire via le registre ''tristate'' du processeur si c'est une entrée ou une sortie.
Pour lire le poussoir il faut déclarer

```
TRISA = 0b00010011 ;  // RA4 entrée (actif 0)  RA0 RA1 sont les entrées du capteur de distance
```

Pour lire le poussoir, on définit la variable booleenne `Pous` qui vaut 0 si le poussoir est pressé.

```
#define Pous PORTAbits.RA4
```

Le problème maintenant est de faire un programme simple qui répond à une action sur le poussoir.
On sait faire des sons, donc comme premier exercice, obtenons le son quand on presse :
Au lieu de `if (Pous==0)`
on peut écrire `if (!Pous)`
	`void loop()`
	`{`
	`  if (Pous==0)`
	`  {`
	`  //les 4 instr.pour jouer une période`
	`  }`
	`}`

Pour qu'une action unique se passe quand on presse et relâche le poussoir, il faut attendre que l'on presse, puis que l'on relâche, et filtrer les rebonds de contact (5 à 20 millisecondes).	`void loop()`
	`{`
	`  while (Pous) delay(5);`
	`  while (!Pous) delay(5);`
	`//insérer ici les instructions pour faire un bip (Wd2Sons0.pde)`
	`}`

Voir les programmes `Wd2Pous1.pde` `Wd2Pous2.pde` si vous avez des problèmes.
Un programme très utile compte les actions sur le poussoir (Wd2Pous1.pde), ce qui permettra de choisir différentes démonstrations à l'enclenchement du Wellbot2.

Connecteurs d'extensions

On voit sur le schéma que 7 lignes sont à disposition sur les connecteurs d'extension.
On peut câbles sur ces lignes des LEDs, des interrupteur (avec pull-up) et les accéder soit avec les numéros de pin Pinguino, soit via les ports A et C. Le port B est réservé pour les moteurs, voir www.didel.com/wbot2/Wb2PasAPas.pdf
Les capteurs de distance sont documentés sous www.didel.com/wbot2/Wb2IrDist.pdf
et les extension prévues, Bico64 et Af2x8c seront documentées ultérieurement, voir www.didel.com/wbot2/Wb2Liens.pdf

jdn110306/110820