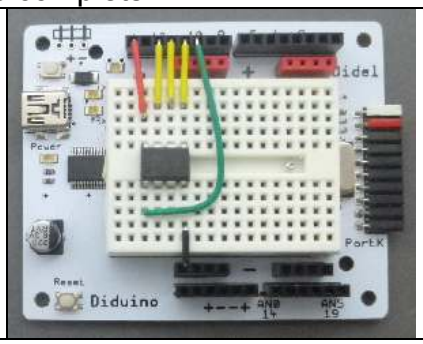
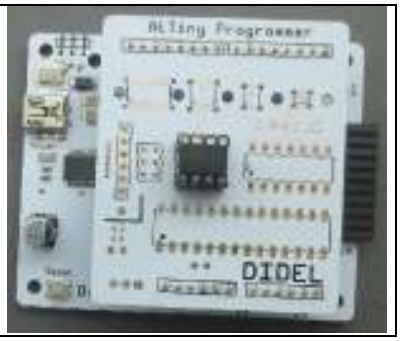
Programmer un AVR AtTiny avec un Arduino/Diduino

Le Tiny13A/25A/45/85 en boitier 8 broches est utilisé entre autre sur le Dé de PYR. Il peut être utilisé comme contrôleur spécialisé dans de nombreuses applications.
Son avantage est de se programmer en C et il peut utiliser comme programmateur une carte compatible Arduino.
Le Tiny24/44/84 a 14 pattes, avec 11 entrées sorties.
En fait, on peut programmer tous les AVR de la même façon. Nous utilisons le terme AtTiny en pensant que pour programmer des AVR plus performants, on utilise de préférence des programmeurs plus flexibles et plus complets.

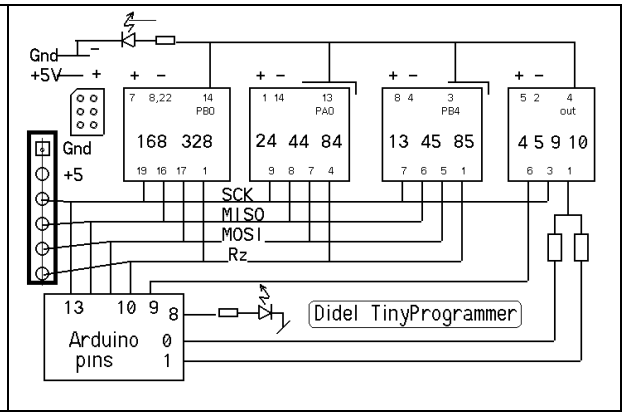
Le câblage sur breadboard est facile pour le Tiny13/25/45/85.
Le shield AtTinyProgrammer permet de programmer facilement le Tiny24/44/84 et leurs versions SMD.
Il devrait permettre aussi de programmer les Tiny 6 pattes à venir.
Descriptions détaillée sous www.didel.com/diduino/AtTinyProgrammer.pdf





A noter que cette documentation est valable pour la présérie début 2015. Les commentaires des utilisateurs permettront une version améliorée à mi-2015.

Le schéma simplifié du AtTiny programmer montre l'assignation des signaux de programmation.
Une Led est câblée sur une ligne inutilisée (RB4 sur 45, RA1 sur 24, TB0 sur 328. rien sur Tiny4 dont l'interface sera revu quand cette famille de boitiers 6 bits sera mieux connue.
Pour les boitiers SMD, des pins positionnent le circuit et on presse sur le circuit pendant la programmation. La solution a été abondamment utilisée avec des Pics 10, 12, 16F depuis 2003.



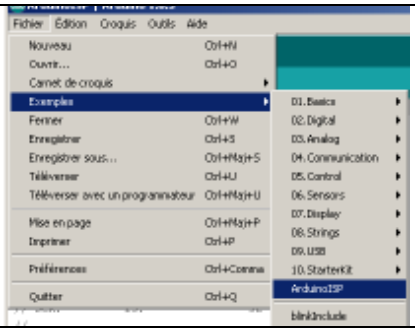
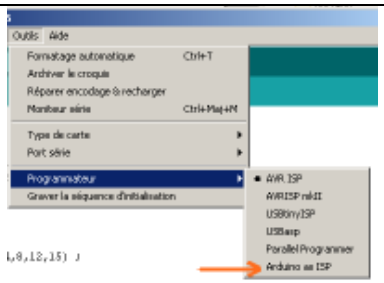
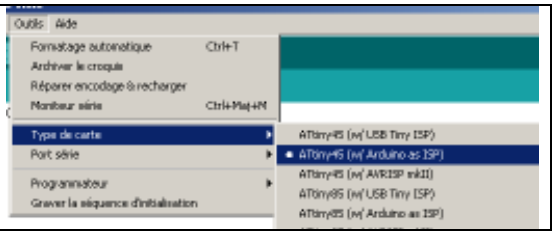
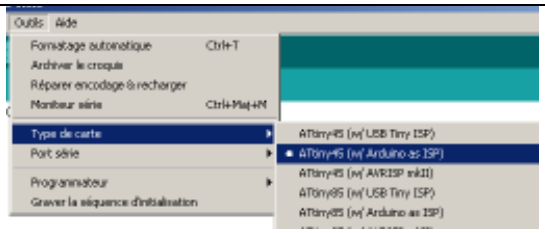
1- **Les 2 options de programmation d'un AtTiny**

Il faut comprendre que les efforts de la communauté Arduino sont pour programmer un AtTiny avec des programmes Arduino dans lesquels on retrouve le digitalWrite(), delay(), etc et on peut ainsi garder ses (mauvaises) habitudes de programmation. Cette approche implique d'installer un "ATTiny master zip file" qui ajoutera à votre programme AtTiny les fonctions Arduino nécessaires qui prennent de la place en mémoire, et excluent parfois les Tiny 13 et 24 qui ont peu de mémoire. Voir https://code.google.com/p/arduino-tiny/ ou chercher Arduino AtTiny programming pour suivre cette approche. Pour le Tiny13, une description qui semble simple se trouve sous http://www.hobbytronics.co.uk/arduino-attiny

Notre approche ici est différente. Elle implique de programmer en C les fonctions nécessaires et si on a besoin de timers, d'entrées analogiques, comprendre la documentation du fabriquant et/ou notre documentations. Avec 1000 instructions on peut faire des programme incroyablement riches, lisibles et modifiables si bien programmés.
Nous voulons disposer d'un environnement efficace pour programmer, qui utilise une carte Arduino et compatible. ScratchMonkey http://microtherion.github.io/ScratchMonkey/ est une solution proche de celle documentée ici, avec un meilleur support pour commander les "fuse settings". Ce qui est documenté ici convient tant que l'on utilise l'oscillateur interne à 1MHz et plus, que la pin Reset ne doit pas être utilisée comme signal, et que le mode ISP programming est autorisé. Il y a dans des menus des options qui ne nous concernent pas.

2- Principe de la programmation

Vous savez comment charger des programmes dans votre carte Diduino/Arduino.Pour programmer un autre processeur, il faut charger dans la carte un programme qui va envoyer le programme dans le processeur choisi, qu'il faut avoir trouvé dans les menus d'Arduino (cela peut limiter le choix). Après avoir programmé, il faut se remettre dans son environnement de programmation.

1) Le programme ArduinoISP est chargé depuis la rubrique Exemples. Il faut le téléverser comme un programme usuel.	
2) Il faut ensuite modifier le comportement du téléverseur pour que le binaire envoyé transite vers le Tiny. Sélectionner dans le menu Outil/Programmateur le mode Arduino as ISP	
3) Dans le menu Outil/Types de cartes Sélectionner le processeur. Les processeurs de la même famille ont plusieurs noms selon la taille mémoire. Les options offertes sont troublantes et la liste dépends de la date d'installation.	
Vérification: on est maintenant dans les modes Type de carte AtTiny xxx PortSérie Com port attribué à la carte Programmateur Avr ISP 4) Si c'est un ATtiny qui n'a jamais été programmé, cliquez sur Outils>Graver la séquence d'initialisation (onglet Burn bootloader, mais ce n'est pas un "bootloader", il n'agit que sur les fusibles)	
5) Il faut maintenant charger le programme pour le AtTiny, vérifier la compilation et téléverser comme d'habitude. Si vous avez le message : *avrdude: please define PAGEL and BS2 signals in the configuration file for part ATtiny XX* Continuez, ce message est inutile.	

3- Programme de test

Pour lire/écrire directement sur les ports et écrire des fonctions efficaces, voir en particulier www.didel.com/coursera/LC4.pdf et autres chapitres. Il ne faut pas utiliser les setup et loop, mais travailler en C avec un int main () suivi par les instructions de setup et le while(1) qui remplace le loop.

Le programme blink comme premier test utilise une led et résistance sur la pin 3 du Tiny 13 (PB4 PORTB) vers le Gnd (déjà câblé sur le shield Didel). La fonction delay() remplace le

delay() d'arduino. On peut utiliser le même nom, puisque l'environnement arduino n'est pas appelé. Par contre, il faut un `#include  <avr/io.h>` pour pouvoir utiliser les noms des ports, des timers et de leurs bits.

```
// Blink.ino   Tiny13/45/85 clignote PB4 pin 3
#include <avr/io.h>  // un fichier spécifique serait mieux
#define LedToggle PORTB ^= (1<<4)
void DelMs (int dm) {  // calibré 8 Mega avec 450 (56 à 1 MHz)
   for (volatile int i=0; i<dm; i++) {
      for (volatile int j=0; j<450; j++) {}
   }
}
int main () {
  DDRB |= (1<<4);  //PB4 out
  while (1){
     LedToggle ;
     DelMs (500);
  }
}
```

Pour le Tiny24 en boitier 14 broches, la Led du shield TinyProgrammer est sur PA0

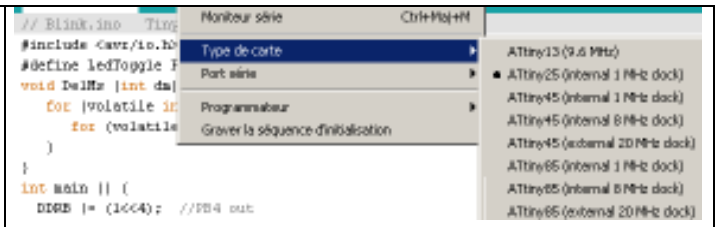
Pour retourner travailler avec votre Diduino ou Arduino, n'oubliez pas de remettre votre type de carte et le mode AVR ISP

4- Processeurs supportés

La liste des processeurs se trouve sous `..\Mes Documents\Arduino\hardware\attiny\boards.txt`. Après avoir modifié cette liste, il faut recharger Arduino.
A noter que cette liste complète celle existant dans `..\Program files\arduino\hardware`, que vous pouvez aussi éditer, mais qui sera mise à jour à chaque révision en ignorant vos modifications.
On a par exemple

```
attiny13.name=ATtiny13 (9.6 MHz)
attiny13.bootloader.low_fuses=0x69
attiny13.bootloader.high_fuses=0xff
attiny13.bootloader.extended_fuses=0xff
attiny13.upload.maximum_size=1024
attiny13.build.mcu=attiny13
attiny13.build.f_cpu=9600000L
attiny13.build.core=arduino:arduino
attiny13.build.variant=tiny8

attiny25.name=ATtiny25 (int 1 MHz clock)
...
```



On peut supprimer des processeurs jamais utilisés pour alléger la liste, en rajouter d'autre en respectant la syntaxe, avec des paramètres s'inspirant d'autres processeurs à défaut de savoir ou trouver l'information.

Un autre fichier est utile pour savoir la correspondance entre les pattes du processeur et les pins Arduino. On le trouve sous `...\Mes documents\Arduino\hardware\attiny\variants\tiny8` qui regroupe les processeur 8 pattes, ou `...tiny14` pour les 14 pattes.

6- Remplacement des fonctions et librairies Arduino

-- voir le document en Anglais TinyProgramming

7- Passer de Arduino à AVR studio

Les programmes faits selon les exemples précédents sont compatibles AVR studio. Il faut créer un projet et y mettre le programme Arduino

8- Exemple d'application pédagogique

Le dé électronique DePyr après soudure par les élèves est reprogrammé avec un programme facile à comprendre et modifiable par l'élève, avant de remettre le programme original qui endort le processeur
www.didel.com/diduino/DePyr.pdf