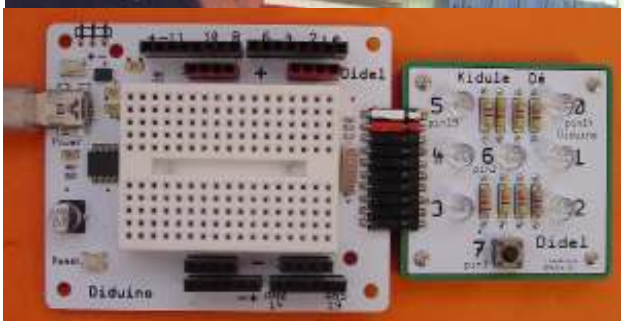
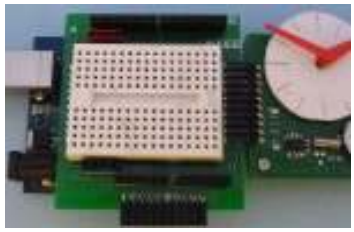
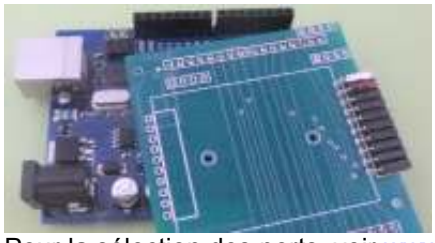
ArduiKi – un shield pour utiliser les Kidules sur carte Arduino





Les Kidules, héritiers des Microdules et du connecteur Mubus des Smaky, base de l'enseignement des périphériques d'ordinateurs à l'EPFL depuis 1978 ont une valeur pédagogique prouvée.
Les Kidules se reconnaissent à leur forme carrée de 46x46mm, parfois une taille double, mais ce qui est caractéristique c'est le connecteur 10 pins au pas de 2.54mm. Ce connecteur est robuste et facile à insérer. Les signaux sur ce connecteur sont Gnd, +3 à 5V et 8 bits de données correspondant à un port de microcontrôleur. Le Diduino a un connecteur Kidule en standard, qui utilise les bits de deux ports de l'AVR, aucun port complet n'étant disponible. Arduiki offre le même service avec un shield sur un carte Arduino compatible.
Le PortKi est vu par l'étudiant comme un port virtuel, accédé par les fonctions WriteKi() et ReadKi(). Un choix de Kidules permet d'introduire la programmation en C/Arduino, puis comprendre les interfaces, capteurs, moteurs. C'est le matériel idéal pour les travaux pratiques qui doivent être nécessairement associés à un cours sur les microcontrôleurs.

Avec Arduino, le concept de shield permet de faire ce que font les Kidules, mais les shields ne se sont pas développés avec un objectif pédagogique. De plus, insérer et enlever un shield nécessite un soin que des élèves n'ont pas nécessairement.
On peut faire câbler sur un breadboard les composants que l'on voit sur un Kidule, mais le câblage est lent et source d'erreurs, et n'apporte rien pédagogiquement.



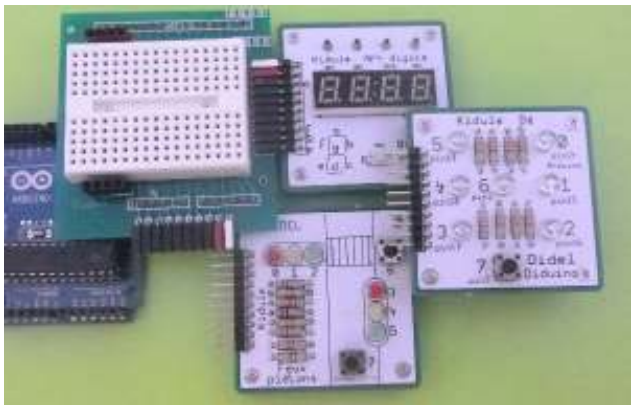
La solution proposée par Didel est le shield ArduiKi, que l'on installe sur une carte Arduino compatible, et qui permet ensuite de travailler avec tous les kidules existants.
Un deuxième port 8bits est possible, prévu pour un microdule d'affichage ou l'entrée d'un mot de 8 bits.

Pour la sélection des ports, voir www.didel.com/kidules/PortK.pdf

Ensemble de base C-facile

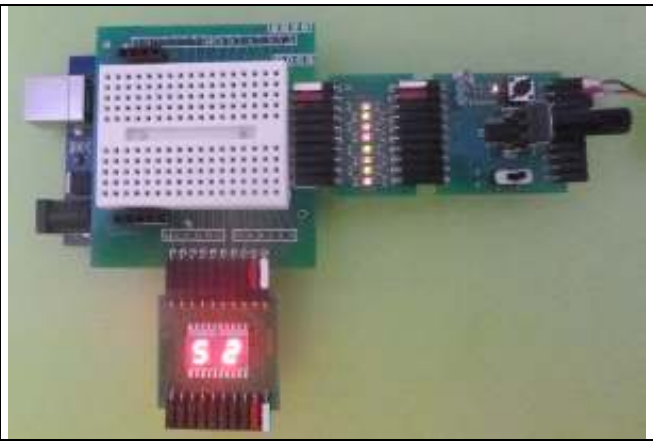
les trois kidules qui ont fait leur preuve avec la carte Diduino, plus l'adaptateur Arduiki compatible avec votre carte Arduino.
Cette carte vous permet d'ajouter en option un breadboard de 60x70mm et de vous trouver avec tous les avantages du l'ensemble Diduino C-facile.

Cfacile Arduiki 80.-
Breadboard démontable avec vis 10.-
Shield carte universelle à souder 5.- (pas sur la photo)



Exemple

Le kidule PoToD8 converti la position d'un potentiomètre en mot de 8 bit envoyé sur le port K, programmé en entrée. Ce mot est transmis à un affichage Kidule branché sur le port K2 programmé en sortie.



Le programme qui copie le PortK (la position du pot est convertie en binaire par un processeur local) sur le PortK2 (affichage 8 bits) peut être écrit avec un minimul d'instruction de copie des champs de bits.

```
//TestArduiki.ino – 8 bits in, 8 bits out
//PortK = PORTC bits 0-5 in, PORTD bits 2,3 in
//PortK2 = PORTD bits 4..7 out PORTB bits 0..3 out
void setup() {
 DDRC = 0;
 DDRB = 0b00101111 ;
 DDRD |=  0b11110000 ;
 DDRD &= ~0b00001100 ;
}
byte data;
void loop() {
  data = (PINC & 0x3F) | ((PIND & 0x0C)<<4);
  PORTD = (data & 0x0F)<<4;
  PORTB =  data >> 4;   // & 0xF0 inutile
}
```

Dans des applications de mise au point, on utilise parfois seulement le PortK2 (s1 on lit des entrées analogiques) ou le PortK (si on commande des moteurs ou teste le SPI). Une librairie est alost utile. Il y a 4 cas à prévoir, avec dans chaque cas un setup spécifique.

```
// PortK.h  PortK et PortK2
#include <Arduino.h>

 void SetupKiOut ()
{
  DDRB |= 0b00100000 ;  // led 13 out
  DDRC   = 0b00111111;  // bits 0-5
  DDRD  |= 0b00001100 ; // bits 2 3
}

void SetUpKiInOut () {  //0-3 out 4-7 in
   DDRC  = 0b00001111 ; // bits  0-3 out 4 in
   DDRD &= 0b11110011 ; // bits 5 6 in = 3 2
}

void SetupKiIn () {  //0-3 out  4-7 in lect
   DDRC  = 0b00000000 ; // bits  0-3  4 in
   DDRD &= 0b11110011 ; // bits 5 6 in =3 2
}

// ecrire 8 bits
void KiWrite (int kk) {
  PORTC = kk;
  PORTD |= (kk>>4)& 0x0C; // force les 1
  PORTD &= (kk>>4)|~0x0C; // force les 0
}

byte KiRead () {  // lit 8 bits
  byte dd ;  // data to be read
  dd = PINC & 0x3F ; // keep 6 bits
  dd |= (PIND & 0x0C) << 4 ; // add after a 4-bit shift
  return dd ;
}
```

```
//--------------

void SetupK2Out () {
  DDRB |= 0b00101111 ; // led 13 out
  DDRD |= 0b11110000 ;  // bits 4-7 pour K2
}
void K2Write (int kk) {
  PORTD &= (kk&0x0F)<<4 ; // force les 0
  PORTD |= (kk&0x0F)<<4 ; // force les 1
  PORTB &= (kk>>4) ;      // force les 0
  PORTB |= (kk>>4);       //force les 1
}
byte K2Read () {  //pas utilisé ici
  byte dd;  // data to be read  bits 0..7
  dd = (PIND & 0xF0) >> 4 ; // bits 4 5 6 7 --> low
  dd |= (PINB & 0x0F) << 4 ; // bits 0 1 2 3 --> high
  return dd ;
}

void SetupK2In () {
 ... exercice à faire
}
```

jdn 14.02.04/15.08.05