INNOVATIONS... MONTAGES FIABLES... ÉTUDES DÉTAILLÉES... ASSISTANCE LECTEUR

ELECTRONIQUE

ELECTRONIQUE

ET LOISIRS

magazine

www.electroniquemagazine.com

n°138 PRINTEMPS 2017

ANALYSEUR DE SPECTRE

Émetteur FM

Alimentation à courant constant

- RaspberryPi 2 & 3
- Cours Arduino VIII
- Breakout Board - IV
- DEMOBOARD PIC18F – III
- Imprimante 3D bicolore - IV
- Programmez sous iPhone, iPad - IV
- Interface USB pour le bras robotisé

Thermomètre Chromatique

N° 138 Mars 2017

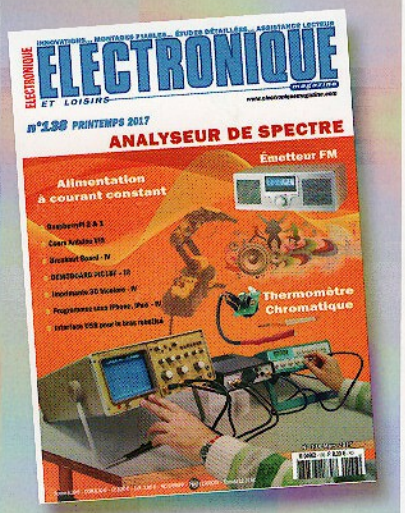
M 04662 - 138 - F: 8,30 € - RD

France 8,30 € - DOM 8,30 € - CE 8,80 € - LUX. 8,80 € - NC(S)990FP - Poly.fr 1300CFP - Canada 12,70 $C

Sommaire

ARTICLES

Numéro 138
Printemps 2017
100 pages



Les typons des circuits imprimés et les programmes **lorsqu'ils sont libres de droits** sont téléchargeables

L'EDITO PRINTEMPS 2017

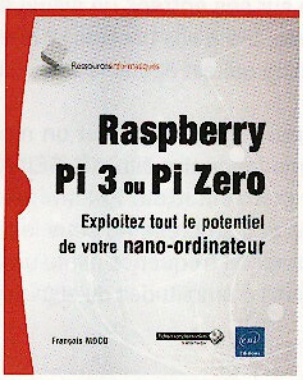
Chères lectrices, chers lecteurs, en ce début d'année 2017 nous vous proposons onze projets couvrant les principaux domaines de l'électronique et de l'informatique.

Commençons par l'émetteur FM qui vous permettra de réaliser votre propre radio domestique, le code source en BASIC du microcontrôleur est entièrement listé dans l'article afin d'adapter le montage à vos besoins.

D'autre part, nous faisons le point pour nos lecteurs sur le RaspberryPi versions 2 et 3, et notamment en ce qui concerne les problèmes de dépannage.

Pour le laboratoire, nous vous proposons de transformer votre oscilloscope en analyseur de spectre. Le programme source sera disponible en téléchargement dans le prochain numéro lorsque nous aborderons la réalisation pratique. Nous décrivons aussi une alimentation à courant constant afin de tester vos montages en toute sécurité. Le thermomètre chromatique vous permettra de déterminer la température d'un liquide en fonction de la couleur de la LED.

Bien évidemment, nous n'oublions pas les suites des articles du numéro précédent avec les cours Arduino et iPhone, la platine de développement PIC avec là encore de nombreux exemples de codes sources dans l'article, la fin de la partie mécanique de la 3DVERTEX et aussi celle des cartes de prototypages. Enfin nous vous faisons découvrir l'interface USB pour piloter le bras robotisé ainsi que des exemples de commandes en langage Python à partir d'un RaspberyPi.



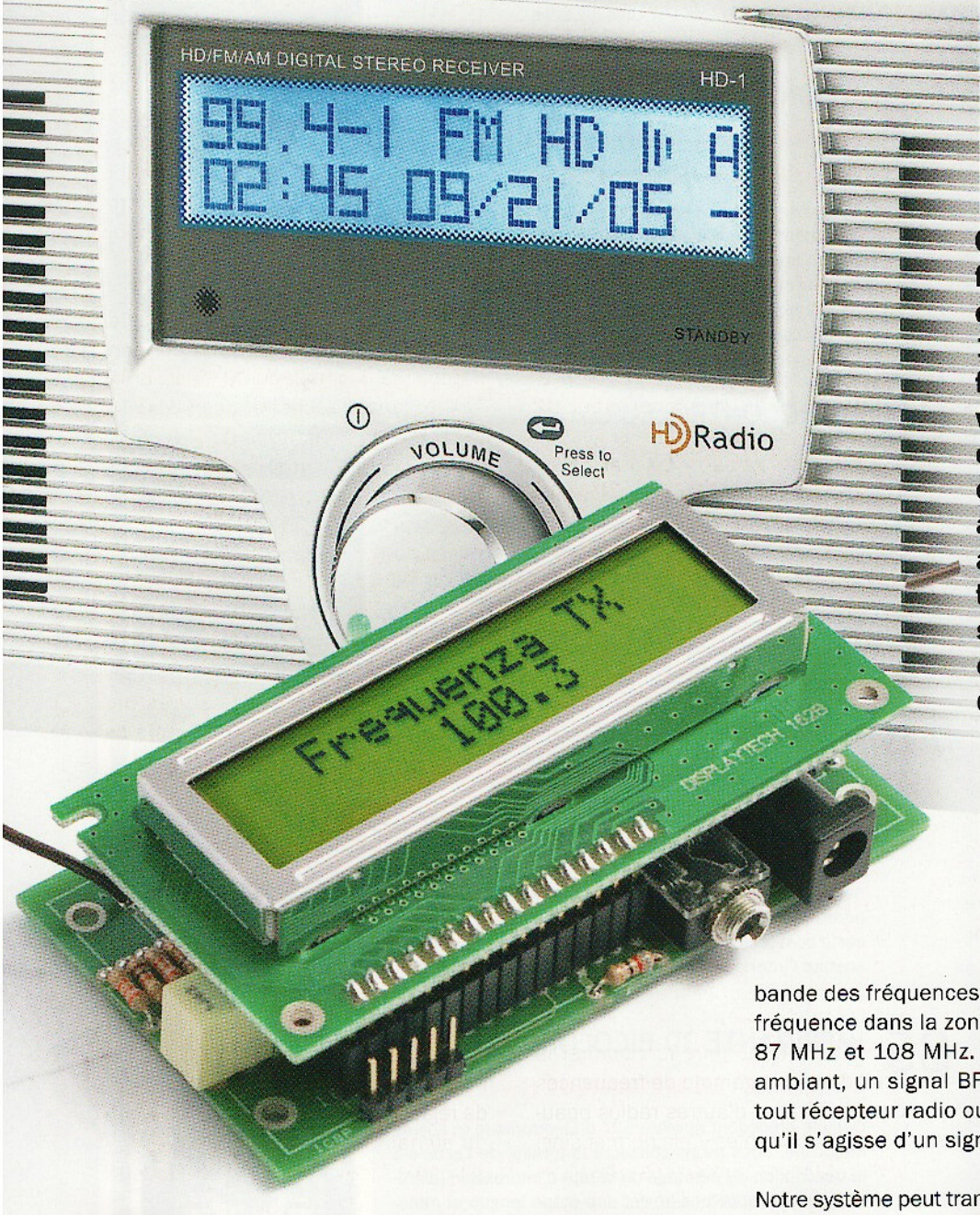
www.framboise314.fr

à l'adresse **www.electroniquemagazine.com** dans le sommaire de la revue 138 et à l'onglet « Télécharger »

ET 801

Émetteur FM programmable

de Boris LANDONI



Cet émetteur programmable fonctionne dans la gamme des fréquences de 87 MHz à 108 MHz en modulation de fréquence. Sa puissance peut varier de 0,5 mW à 2 mW. Il dispose d'une sortie audio BF qui permet de le raccorder à un amplificateur Hi-Fi, vous pouvez alors réaliser votre propre station radio domestique. Grâce à son afficheur LCD, vous pouvez effectuer tous les réglages des différents paramètres.

Parfois, vous aurez besoin d'envoyer un signal audio à un enregistreur, à une table de mixage, à un système de sonorisation ou à un amplificateur Hi-Fi qui ne peuvent pas être raccordés à l'aide de câbles traditionnels de par leur emplacement.

Dans ce cas, la solution la plus pratique est d'utiliser un tuner FM qui sert de récepteur radio et de lui envoyer le signal audio à l'aide d'un émetteur FM. Cela permet de ne pas se soucier de la longueur et de l'agencement des câbles de raccordement. Il suffit d'un émetteur de faible puissance fonctionnant dans la gamme des fréquences FM et le tour est joué.

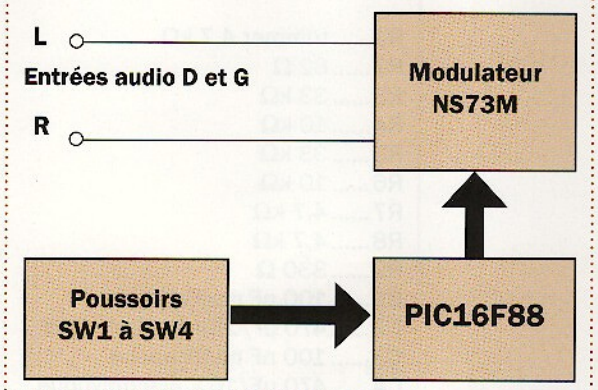
Le circuit décrit ici a été conçu pour résoudre un problème de connexion. Il s'agit d'un modulateur/émetteur dans la bande des fréquences FM qui fonctionne en modulation de fréquence dans la zone réservée à la radio libre, soit entre 87 MHz et 108 MHz. Il peut transmettre, dans le milieu ambiant, un signal BF stéréo appliqué à ses entrées que tout récepteur radio ou tuner FM peut capter et reproduire, qu'il s'agisse d'un signal musical ou d'un son (voix).

Notre système peut transmettre un signal audio stéréo de qualité Hi-Fi ! Le circuit accepte sur ses entrées un signal audio dont l'amplitude peut être adaptée à vos besoins et le transmettre sur un canal (fréquence) qui est également réglable.

Notre montage utilise le circuit **NS73M** qui est un module émetteur fabriqué par la firme japonaise NIIGATA SEIMITSU. Il comprend un **modulateur** et un **émetteur FM intégré**, ses dimensions sont très petites et il se présente sous la forme d'un boîtier CMS. Le modulateur de fréquence pilote un oscillateur sur la base des variations d'amplitudes du signal audio reçu sur les entrées.

Le circuit est conçu pour réaliser des transmissions stéréo, le modulateur génère une **sous-porteuse** ayant une fréquence de **38 kHz** à partir de laquelle est dérivé

Synoptique du transmetteur FM



le signal de modulation contenant les 2 canaux du signal stéréo qui module l'oscillateur.

L'étage de puissance du circuit peut développer une puissance de **0,5 mW** à **2 mW** efficaces sur une charge (antenne) de **50 Ω**. Cela permet de transmettre un signal audio à quelques dizaines de mètres, par exemple à l'intérieur d'un appartement, d'un bureau ou dans un grand local.

La modulation est effectuée selon les spécifications de la bande FM, c'est-à-dire avec un **écart de fréquence** de seulement **± 50 kHz**. La consommation du circuit est très faible, de l'ordre de **20 mA**, ce qui permet de réaliser des montages fonctionnant sur piles ou accumulateurs comme des microphones sans fil par exemple.

La particularité de ce modulateur/transmetteur FM est que les paramètres de puissance d'émission et de fréquence d'émission ne sont pas contrôlés à partir de tensions analogiques, mais par un **contrôleur numérique** compatible avec une interface de type **BUS I2C**.

Par conséquent, le circuit peut être géré de manière relativement simple en utilisant un microcontrôleur à partir duquel il est possible de régler la fréquence et la puissance d'émission.

Schéma électrique

Le schéma électrique de notre montage est visible en figure 1, vous pouvez voir que le transmetteur FM (U1) est piloté via plusieurs lignes de données appropriées reliées au microcontrôleur **U2** (**PIC16F88**). Celui-ci assure la gestion de toutes les lignes : « **CK** » et « **DA** » pour le bus de données, mais aussi « **LA** », « **IIC** » et « **TEB** ».

Le microcontrôleur pilote le transmetteur U1 selon les instructions provenant des boutons poussoirs en utilisant des procédures qui s'affichent sur l'écran LCD. En **fonctionnement normal**, l'écran **LCD affiche la fréquence d'émission**, et lors des **réglages**, les **valeurs choisies**. Par conséquent, le microcontrôleur assure l'interface entre l'utilisateur et l'émetteur. Nous devons faire en sorte que le circuit U1 fonctionne selon la fréquence et la puissance désirées.

La possibilité de choisir la fréquence d'émission dans la gamme FM tout entière permet de faire fonctionner le circuit dans une gamme de fréquences non utilisées par d'autres radios beaucoup plus puissantes. Cela permet ainsi d'éviter les interférences des émetteurs voisins, sinon la qualité sonore obtenue serait médiocre en raison de la superposition des signaux des émetteurs à celui de notre montage.

Le **réglage de la puissance d'émission** permet également de **limiter les perturbations** de la réception des fréquences proches.

Cela est très important car même si l'émission dans la bande est tolérée pour des tests, l'**émission en continu est strictement interdite car elle peut troubler les radios officielles qui détiennent une licence**.

En France il existe une organisation appelée l'**ARCEP** (Autorité de Régulation des Communications Electroniques et des Postes) qui régule toutes les bandes de fréquences.

Regardons le schéma électrique (figure 1) et voyons comment s'articule le montage. Le module transmetteur U1 est utilisé dans une configuration classique, il est interfacé avec le microcontrôleur U2 qui contrôle la section numérique de U1.

Le **signal audio** stéréo arrive directement, par l'intermédiaire des **condensateurs de découplage C9** (canal droit) et **C8** (canal gauche) et des **diviseurs de tensions** constitués par les résistances **R5/R6** (canal droit) et **R3/R4** (canal gauche), et respectivement sur les broches **13** « **RIN** » (canal droit) et **12** « **LIN** » (canal gauche) du circuit U1.

La section « entrée audio » du circuit U1 est dotée d'une sensibilité relativement élevée si bien que les étages d'entrées peuvent être facilement saturés par des signaux provenant d'une table de mixage ou d'un lecteur CD.

Comme vous pouvez le constater il n'y a pas de trimmer ou de potentiomètre de réglage du niveau, car le circuit **U1 permet un ajustement du niveau du signal qui module la porteuse FM**.

Le réglage s'effectue au moyen d'un potentiomètre électronique interne qui est contrôlé par les instructions provenant du microcontrôleur. Cela permet de régler la sensibilité et donc le niveau du signal d'entrée parmi 3 valeurs : 100 mVeff ; 140 mVeff et 200 mVeff.

La présence des diviseurs de tension sur chacune des entrées (R3/R4 et R5/R6) **atténue 4 fois l'amplitude** des signaux avant d'atteindre les entrées de U1, si bien que vous pouvez appliquer des signaux compris entre **400 mVeff** et **800 mVeff au maximum** (avant saturation).

Les broches **RB0**, **RB1**, **RB6** et **RB7** du microcontrôleur sont configurées en « **entrées** » avec les résistances internes de **pull-up activées**, ces broches scrutent l'état des boutons qui permettent les différents réglages.

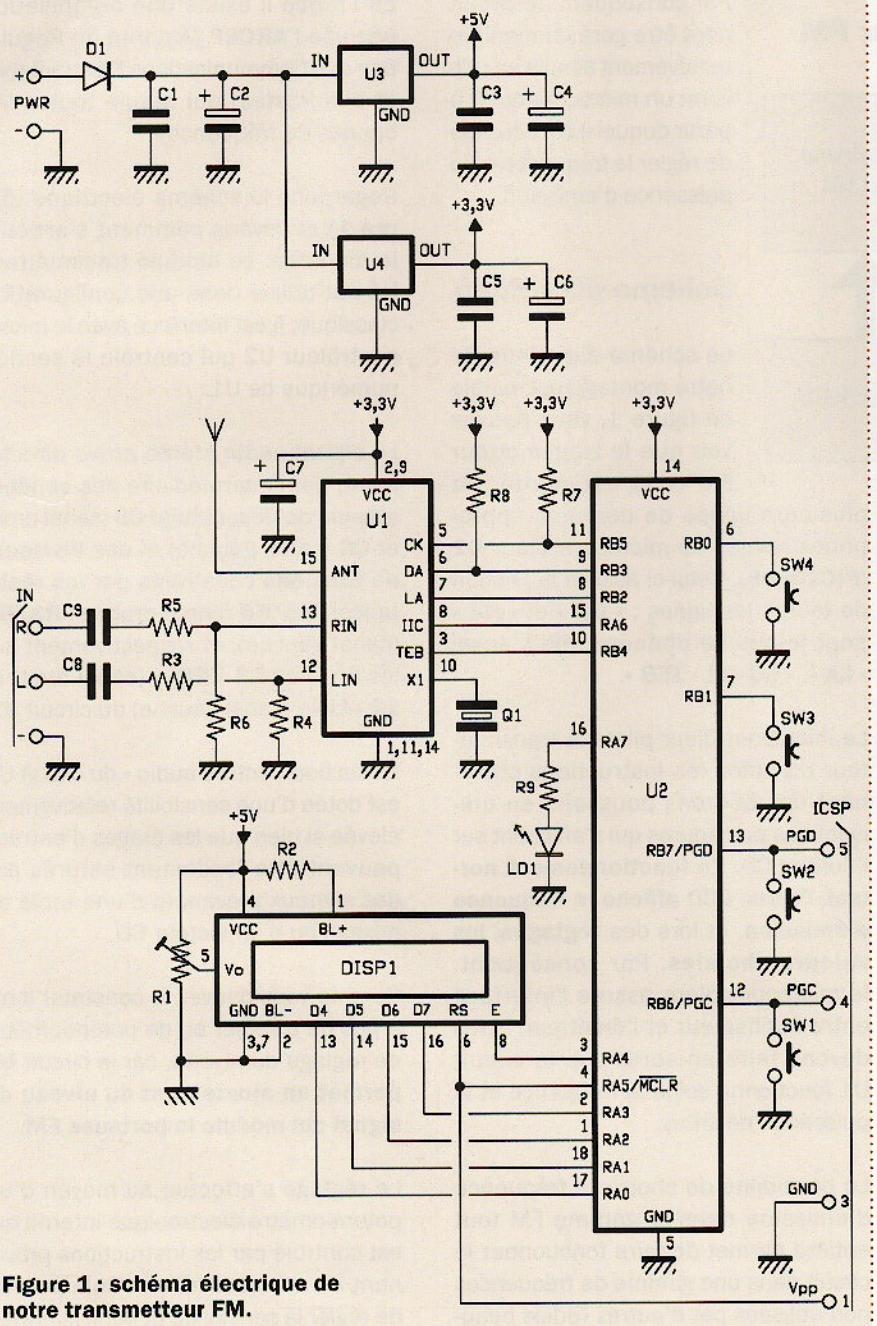


Figure 1 : schéma électrique de notre transmetteur FM.

Liste des composants du ET801

R1..... trimmer 4,7 kΩ
R2..... 82 Ω
R3..... 33 kΩ
R4..... 10 kΩ
R5..... 33 kΩ
R6..... 10 kΩ
R7..... 4,7 kΩ
R8..... 4,7 kΩ
R9..... 330 Ω
C1..... 100 nF multicouche
C2..... 470 µF/25V électrolytique
C3..... 100 nF multicouche
C4..... 470 µF/16 V électrolytique
C5..... 100 nF multicouche
C6..... 220 µF/16 V électrolytique
C7..... 100 µF/16 V électrolytique
C8..... 1 µF 63 V polyester
C9..... 1 µF 63 V polyester
D1..... 1N4007
U1..... NS73M (1340-CHIPTXFM)
U2..... PIC16F88-I/P
U3..... 7805
U4..... LD1086V33
DISP1 afficheur LCD 2 lignes 16 caractères
SW1 à SW4mini poussoirs coudés 90°
LD1.... LED 3 mm verte
Q1..... quartz 32,768 kHz

Divers
Support ci 2 x 9 broches
Dissipateur pour régulateur
Vis 10 mm 3 MA (x2)
Ecrou 3 MA (x2)
Barrette mâle 5 broches
Barrette mâle 16 broches
Barrette femelle 16 broches
Fiche alimentation pour circuit imprimé
Prise jack 3,5 mm stéréo pour circuit imprimé

Les lignes **RB6**, **RB7**, la **masse** et la broche **MCLR** sont utilisées pour la **programmation en circuit** du microcontrôleur afin de charger le « firmware » (programme) au format « **.hex** ».

Les broches **RA0**, **RA1**, **RA2**, **RA3** et **RA7** constituent l'**interface** avec l'afficheur **LCD**. Elles sont configurées en « **sorties** ». Le protocole de communication avec l'afficheur prévoit que celui-ci doit recevoir et non pas transmettre des données, donc la broche « **R/W** » doit être reliée à la **masse** (GND). Les lignes RA0 à RA3 sont connectées aux broches D4 à D7 de l'afficheur et servent à envoyer les données correspondantes aux caractères à afficher. Les lignes E et RS sont respectivement utilisées pour habiliter le buffer de l'afficheur et pour définir si les données arrivant sur les lignes RA0, RA1, RA2, RA3 se réfèrent à des caractères ou aux emplacements des caractères.

Terminons l'étude du schéma par la section d'alimentation. Elle est constituée par 2 régulateurs de tension qui fournissent **5 VDC** (U3) et **3,3 VDC** (U4) stabilisés, respectivement pour l'afficheur LCD et pour le microcontrôleur (U2) et le transmetteur(U1).

Chacun des 2 régulateurs est pourvu de condensateurs de filtrage nécessaires pour filtrer les parasites.

La diode **D1 protège** l'ensemble du montage contre les **inversions de polarités** qui pourraient se produire sur les broches « **PWR+** » et « **PWR-** ».

Le montage est alimenté par une tension continue comprise entre 9 VDC et 15 VDC.

La consommation de courant est de l'ordre de 100 mA, due en grande partie au rétroéclairage de l'afficheur LCD.

Plan de montage

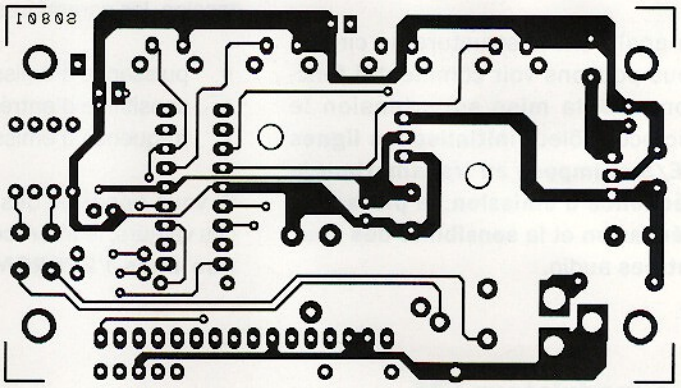


Figure 2 : circuit imprimé à l'échelle 1 : 1 côté soudures de notre transmetteur FM.

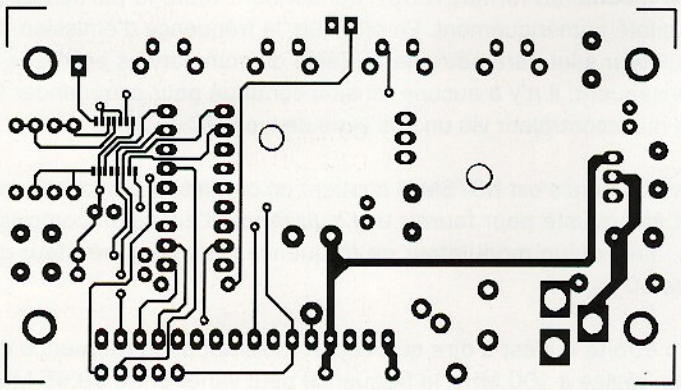


Figure 3 : circuit imprimé à l'échelle 1 : 1 côté composants de notre transmetteur FM.

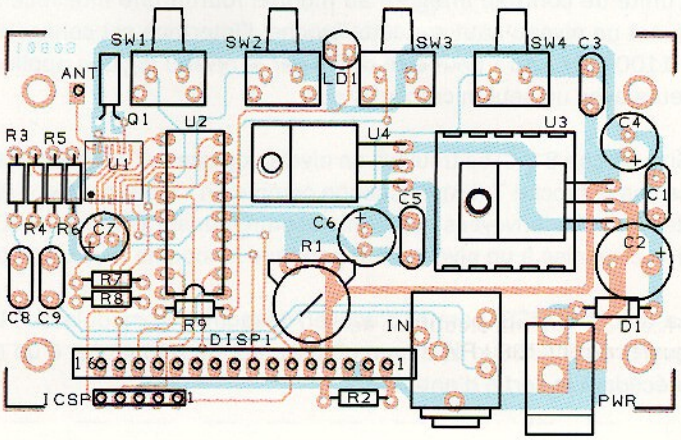


Figure 4 : schéma de câblage des composants du transmetteur FM.

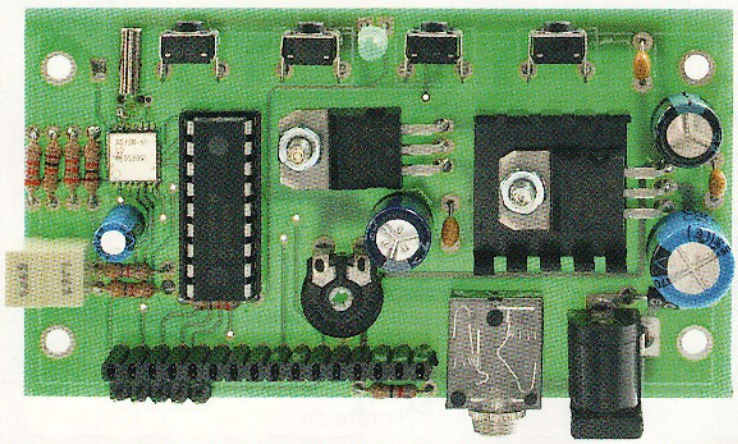


Figure 5 : photo de l'un de nos prototypes. Vous remarquerez en haut à gauche en blanc le modulateur U1.

Le fonctionnement

En analysant la structure du circuit, nous pouvons voir comment il fonctionne. À la mise sous tension le microcontrôleur **initialise les lignes d'E/S** et **impose** au **transmetteur** la **fréquence d'émission**, la **puissance d'émission** et la **sensibilité des deux entrées** audio.

Vous remarquerez qu'à la 1ère mise sous tension, les paramètres par défaut sont :

- puissance d'émission : **2 mW** ;
- sensibilité d'entrée : **200 mV** ;
- fréquence d'émission : **100 MHz**.

Si vous apportez des modifications à ces valeurs, le **microcontrôleur mémorise dans l'EEPROM les nouveaux paramètres et les applique à chaque mise en service suivante**. Parmi les configurations que le PIC exécute au démarrage, il y a celle du bus de communication avec le transmetteur U1.

Le PIC impose une communication respectant le protocole I2C en appliquant un niveau logique haut sur la broche 8 (IIC) du transmetteur U1.

Le modulateur FM

Pour réaliser l'émetteur FM stéréo, nous avons utilisé un module au format TSSOP qui contient toute la partie haute fréquence. Par rapport à un transmetteur classique, il est piloté numériquement. En pratique, la fréquence d'émission, la puissance de l'oscillateur et la sensibilité de l'étage modulateur sont paramétrables à l'aide de commandes envoyées à travers un bus série avec une syntaxe particulière. Par conséquent, il n'y a aucune tension continue pour commander la « varicap » ou l'étage VCO, mais une interface gérée par un microcontrôleur via un bus série de type I2C ou SPI.

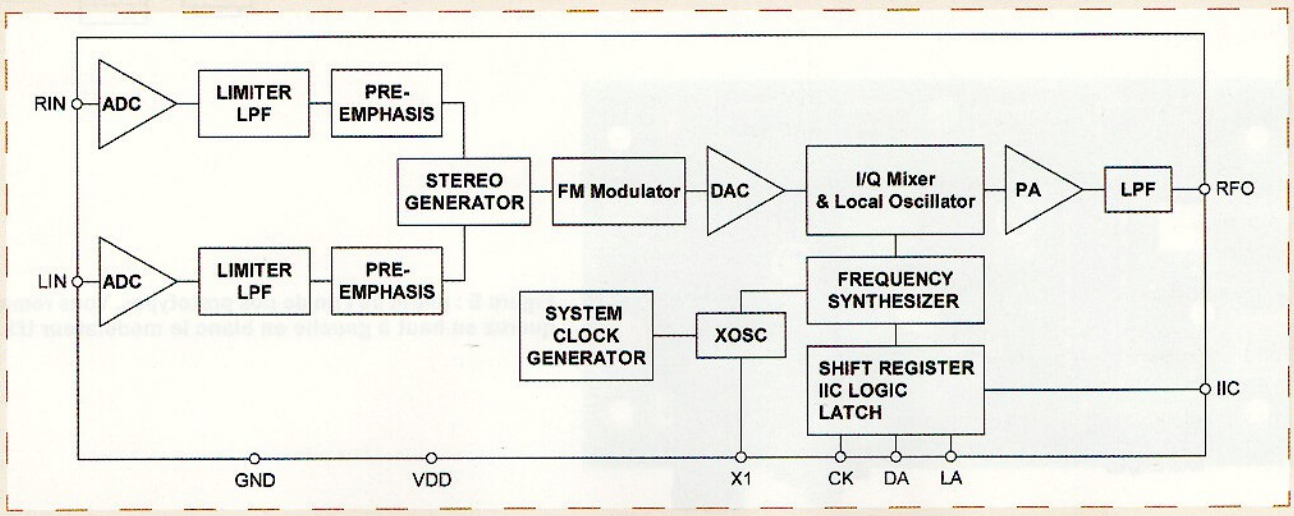
Ce modulateur est produit par la firme NIIGATA SEIMITSU, sa référence est NS73M. Il contient un oscillateur qui fonctionne de 87 MHz à 108 MHz modulé en fréquence et qui peut être ajusté pour fournir une puissance d'émission comprise entre 0,5 mW et 2 mW à une charge (antenne) de 50 Ω, ainsi qu'un modulateur de fréquence avec un générateur de sous-porteuse pour la stéréo avec une préaccentuation de 50 µs.

La modulation est conforme à la spécification FM « bande étroite », c'est-à-dire que l'écart maximal de la fréquence du canal est de ± 50 kHz. Donc pour une fréquence d'émission réglée à 100 MHz, la fréquence peut varier entre 99,95 MHz et 100,05 MHz, ce qui correspond respectivement au niveau minimum et maximum du signal audio.

L'unité de contrôle intégrée au module fournit une interface série configurable à l'aide de la broche 8 (IIC). En appliquant un niveau haut sur cette broche, l'interface est configurée en bus I2C et dans ce cas l'adresse du modulateur est « 1100110x » où x peut être défini par le niveau logique appliqué sur la broche 7 (LA). Cela permet de contrôler 2 émetteurs avec un seul microcontrôleur.

Si la broche 8 (IIC) se trouve à un niveau logique 0, l'interface est alors de type SPI (Serial Peripheral Interface) à 3 fils. Dans ce cas la broche 7 (LA) fonctionne comme un « latch-enable », chaque fois qu'elle est mise à 1 elle détermine l'acquisition des données envoyées par le microcontrôleur sur la broche 6 (DA). En d'autres termes, DA reçoit les données de CK et LA doit être mise à un niveau haut pour que le module reçoive un octet entier et le charge.

La broche 3 (TEB) permet d'activer/désactiver le modulateur. La partie émettrice dispose d'un oscillateur de base, d'un quartz de 32,768 kHz, d'un multiplicateur de fréquence, d'un modulateur FM, d'un amplificateur et d'un filtre passe-bas précédant la sortie d'antenne.



Listing 1

```
' TRANSMETTEUR FM AVEC CHIPTXFM

DEFINE OSC 4

@DEVICEINTRC_OSC
@DEVICEWDT_OFF
@DEVICEPWRT_OFF
@DEVICEMCLR_OFF
@DEVICEBOD_OFF
@DEVICELVP_OFF
@DEVICECPD_OFF

DEFINE LCD_DREG PORTA              'Afficheur sur PORTA
DEFINE LCD_DBIT 0                  'Bit 0 (D4) sur RA0
DEFINE LCD_RSREG PORTA             'RS sur RA5
DEFINE LCD_RSBIT 5
DEFINE LCD_EREG PORTA              'E sur RA4
DEFINE LCD_EBIT 4
DEFINE LCD_BITS 4                  'Affichage sur 4 bits
DEFINE LCD_LINES 2                 '2 lignes
DEFINE LCD_COMMANDUS 2000
DEFINE LCD_DATAUS 50

'DEBUG
DEFINE DEBUG_REG PORTA
DEFINE DEBUG_BIT 3
DEFINE DEBUG_BAUD 9600
DEFINE DEBUG_MODE 0

SYMBOL P1=PORTB.6                  'Poussoir SW1 sur RB6
SYMBOL P2=PORTB.7                  'Poussoir SW2 sur RB7
SYMBOL P3=PORTB.1                  'Poussoir SW3 sur RB1
SYMBOL P4=PORTB.0                  'Poussoir SW4 sur RB0

'Données pour le modulateur U1
symbol CK=PORTB.5
symbol DA=PORTB.3
symbol LA=PORTB.2
symbol IIC=PORTA.6
symbol TEB=PORTB.4

INPUT P1                           'RB6 en entrée soit SW1
INPUT P2                           'RB7 en entrée soit SW2
INPUT P3                           'RB1 en entrée soit SW3
INPUT P4                           'RB0 en entrée soit SW4
OUTPUT CK                          'RB5 (CK) en sortie
OUTPUTDA                           'RB3 (DA) en sortie
OUTPUT LA                          'RB2 (LA) en sortie
OUTPUT IIC                         'RA6 (IIC) en sortie
OUTPUT TEB                         'RB4 (TEB) en sortie

'Définition des varaibles pour le programme
TMP    VAR BYTE
TMP1   VAR WORD
TMP2   VAR WORD
SENS   VAR BYTE
```

```
PW              VAR BYTE
TMPW            VAR WORD
TMPW1           VAR WORD
PULS            VAR BYTE
MENU            VAR BYTE
FREQ            VAR WORD
FREQ2           VAR WORD
REG             VAR BYTE
VAL             VAR BYTE
ADDRESS CON %11001100

ADCON0 = %00000000          'Désactive l'ADC
ADCON1 = %00000000          'Désactive l'ADC
ANSEL  = %00000000          'Toutes les broches en digital
CMCON  = %00000111          'Toutes les broches en digital
OPTION_REG = %00000000
OSCCON = %01100000          'Impose l'horloge interne à 4MHz

HIGH IIC                    'Selection du bus I2C
LOW  LA                     'Utilisée pour l'adresse
HIGH TEB

PAUSE 1000
LCDout $FE,$01,» Transmetteur  «
LCDOUT $FE,$C0,»   FM  1.0   «

DEBUG «TRANSMETTEUR»,13,10
pause 1000
CLEAR

EEPROM 0,[3,232,0,0]        'Lecture de la fréquence FREQ.BYTE1 FREQ.BYTE0, puissance PW et sensibilité SENS
READ 0,FREQ.BYTE1
READ 1,FREQ.BYTE0
READ 2,PW
READ 3,SENS

SETUP:
  REG=$0E:VAL=$05:GOSUB ECRITURE    ' Réinitialisation
  REG=$01:VAL=$B4:GOSUB ECRITURE    ' sous-porteuse forcée
  REG=$02:VAL=$07:GOSUB ECRITURE    ' puissance 2 mW power, Unlock detect on
  REG=$00:VAL=$A1:GOSUB ECRITURE    ' Puissance audio 200 mV pour une modulation à 100 %
  REG=$06:VAL=$1E:GOSUB ECRITURE
 GOSUB FREQUENCE

MAIN:
   FOR TMP=0 TO 100
      GOSUB POUSSOIRS
      PAUSE 1
      IF PULS<>0  THEN
         TMP=250
      ENDIF
   NEXT TMP
   GOSUB VISUAMENU
goto main

ECRITURE:
I2CWRITE DA,CK,%11001100,REG,[VAL]
DEBUG «Ecrit à l'adresse «,BIN8 REG,» la valeur «,BIN8 VAL,13,10
```

```
RETURN
POUSSOIRS:
    IF P1=0 OR P2=0 OR P3=0 OR P4=0 THEN
      IF P1=0 THEN
        PULS=1
      ENDIF
      IF P2=0 THEN
        PULS=2
      ENDIF
      IF P3=0 THEN
        PULS=3
      ENDIF
      IF P4=0 THEN
        PULS=4
      ENDIF
  ENDIF
RETURN

VISUAMENU:
   SELECT CASE MENU
      CASE 0
        TMP1=FREQ/10
        TMP2=FREQ//10
        LCDout $FE,$01,»  Fréquence TX  «
        LCDOUT $FE,$C0,»      «,#TMP1,».»,#TMP2
        if puls=2 then
          MENU=1
        ENDIF
        IF PULS=1 THEN
          MENU=2
        ENDIF
        IF PULS=4 THEN
          IF freq<1080 THEN
            freq=freq+1
            GOSUB FREQUENCE
          ELSE
            FREQ=870
          ENDIF
        ENDIF
        IF PULS=3 THEN
          IF freq>870 THEN
            freq=freq-1
            GOSUB FREQUENCE
          ELSE
            FREQ=1080
          ENDIF
        ENDIF

      CASE 1
        LCDout $FE,$01,»  Puissance TX  «

 SELECT CASE PW
          CASE 0
            LCDOUT $FE,$C0,»    0,5 mW»
            REG=$02:VAL=%00000001:GOSUB ECRITURE
          CASE 1
            LCDOUT $FE,$C0,»    1,0 mW»
            REG=$02:VAL=%00000010:GOSUB ECRITURE
```

En outre, la broche **7** (**LA**) de U1 est **mise à un zéro logique** pour affecter l'adresse **I2C** « **11001100** » au transmetteur.

À ce stade, le circuit est prêt à transmettre les signaux appliqués aux entrées stéréo, le canal FM correspond à la fréquence d'émission. À tout moment vous pouvez modifier les paramètres de fonctionnement par le biais d'une procédure constituée de menus simples, à l'aide des 4 poussoirs. En accédant au menu, il est possible de modifier dans l'ordre la sensibilité d'entrée, la puissance d'émission et la fréquence d'émission.

En appuyant sur les boutons poussoirs **SW1** et **SW2**, vous pouvez **parcourir les différents menus**. Dans l'ordre vous pouvez régler la sensibilité, la puissance et la fréquence et vice versa. SW1 fait défiler le menu en avant alors que SW2 le fait retourner en arrière.

Les boutons **SW3** et **SW4** permettent de **modifier les différentes valeurs**, en augmentant ou diminuant les valeurs des paramètres. SW4 augmente ou incrémente d'une unité la valeur du paramètre alors que SW3 diminue ou décrémente la valeur.

Initialement l'afficheur indique « Fréquence TX » et sur la ligne en dessous la fréquence d'émission actuelle en MHz (par exemple 100.0). Dans ce cas, en appuyant sur SW3 vous diminuez la valeur 100 kHz et sur SW4 vous l'augmentez.

En appuyant sur SW1, vous passez au réglage de la sensibilité. L'écran affiche sur la première ligne « **Sensibilité IN** » et en dessous la valeur (par exemple 100 mV).

Si vous appuyez sur SW3 vous diminuez la sensibilité et sur SW4 vous l'augmentez. Les valeurs prévues sont **100 mV**, **140 mV** et **200 mV**. En appuyant 3 fois sur SW4 vous passez successivement de 100 mV à 140 mV puis 200 mV, si vous appuyez une quatrième fois sur SW4 rien ne se produit. Pour revenir en arrière vous devez appuyer sur SW3.

En appuyant une fois de plus sur SW1, vous passez au réglage de la puissance d'émission.

```
CASE 2
      LCDOUT $FE,$C0,»   2,0 mW»
      REG=$02:VAL=%00000011:GOSUB ECRITURE
      END SELECT

      if puls=2 then
        MENU=2
      ENDIF
      IF PULS=1 THEN
        MENU=0
      ENDIF
      IF PULS=4 THEN
        IF PW<2 THEN
          PW=PW+1
          WRITE 2,PW
        ENDIF
      ENDIF
      IF PULS=3 THEN
        IF PW>0 THEN
          PW=PW-1
          WRITE 2,PW
        ENDIF
      ENDIF
    CASE 2
      LCDout $FE,$01,» Sensibilité IN  «
      SELECT CASE SENS
        CASE 0
          LCDOUT $FE,$C0,»   100 mV»
          REG=$00:VAL=%00100001:GOSUB ECRITURE
        CASE 1
          LCDOUT $FE,$C0,»   140 mV»
          REG=$00:VAL=%01100001:GOSUB ECRITURE
        CASE 2
          LCDOUT $FE,$C0,»   200 mV»
          REG=$00:VAL=%10100001:GOSUB ECRITURE
      END SELECT
      if puls=2 then
        MENU=0
      ENDIF
      IF PULS=1 THEN
        MENU=1
      ENDIF
      IF PULS=4 THEN
        IF SENS<2 THEN
          SENS=SENS+1
          WRITE 3,SENS
        ENDIF
      ENDIF
      IF PULS=3 THEN
        IF SENS>0 THEN
          SENS=SENS-1
          WRITE 3,SENS
        ENDIF
      ENDIF

    END SELECT
    IF PULS<>0 THEN
      PAUSE 300
```

Si vous appuyez sur SW2 vous revenez au réglage de la sensibilité puis de la fréquence.

La valeur de la puissance d'émission est sélectionnable à l'aide des boutons SW3 et SW4. Les valeurs sont : **0,5 mW**, **1 mW** et **2 mW**. En appuyant sur SW4, vous passez de 0,5 mW à 1 mW, avec une 2ème pression à 2 mW. Pour revenir en arrière vous appuyez sur SW3.

Tous vos **paramètres sont stockés dans la mémoire EEPROM** (non volatile) du microcontrôleur, de sorte que vous pouvez
éteindre l'appareil sans crainte d'avoir à renouveler la configuration lorsque vous le rallumez.

Le code source complet du programme en BASIC est disponible dans le Listing 1. Il est commenté, vous pouvez cependant l'adapter pour un autre transmetteur, le principe de commande restant le même.

Réalisation pratique

Eh bien à ce stade, nous pouvons passer à la construction de l'émetteur FM. Le circuit imprimé est un circuit double face, vous pouvez télécharger les typons sur notre site www.electroniquemagazine.com dans le sommaire détaillé de la revue 135 à l'onglet « Télécharger ». Ensuite vous devez procéder à la gravure et au perçage des trous.

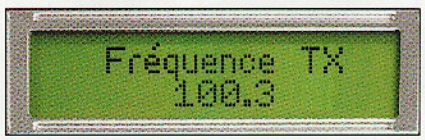
En premier lieu, vous devez souder le module émetteur qui est en CMS. Positionnez-le correctement, le repère indique la broche 1 et il doit se situer face à C7. Avec un point de colle fixez-le, utilisez un fer à souder de 20 W avec une pointe fine et de la soudure pour CMS de 0,5 mm de Ø. Commencez par souder la broche 1, puis laissez refroidir et soudez la broche diamétralement opposée c'est-à-dire la broche 9 proche du microcontrôleur. Laissez refroidir et continuez en soudant la broche 16 puis 8 et ainsi de suite toujours en diagonale.

Ensuite, comme pour un montage traditionnel soudez les résistances, la diode, les condensateurs non polarisés puis le support du microcontrôleur.

```
    ENDIF
     PULS=0
RETURN

FREQUENCE:
  WRITE 0,FREQ.BYTE1
  WRITE 1,FREQ.BYTE0
  FREQ2=10779 +((FREQ-880)*12+((FREQ-880)/5))
  REG=$03:VAL=FREQ2.BYTE0:GOSUB ECRITURE      ' octet bas (moins significatif) de la fréquence  (93.9 megahertz)
 REG=$04:VAL=FREQ2.BYTE1:GOSUB ECRITURE       ' octet haut (le plus significatif) de la fréquence  (93.9 megahertz)
  IF FREQ <= 885 THEN
       REG=$08:VAL=%00011011:GOSUB ECRITURE
       TMP=3
  ELSE
    IF FREQ > 885 AND FREQ <= 987 THEN
      REG=$08:VAL=%00011010:GOSUB ECRITURE
      TMP=2
    ELSE
      IF FREQ > 987 AND FREQ <= 1030 THEN
        REG=$08:VAL=%00011001:GOSUB ECRITURE
        TMP=1
      ELSE
        REG=$08:VAL=%00011000:GOSUB ECRITURE
        TMP=0
      ENDIF
    ENDIF
  ENDIF
RETURN
```

Continuez avec les condensateurs électrolytiques en respectant la polarité (le + correspond à la patte la plus longue). N'oubliez pas le quartz. Soudez les régulateurs, U3 nécessite un dissipateur dont la résistance thermique est de 20 °C/W. Coudez les pattes du régulateur sans les casser, positionnez l'ensemble « radiateur + régulateur » et vissez le sur le circuit.



Les 3 écrans ci-dessus correspondent aux réglages de l'émetteur. Les différents menus sont accessibles à l'aide des boutons SW1 et SW2. Les réglages des valeurs s'effectuent à l'aide des boutons SW3 et SW4.

Ensuite soudez les pattes du régulateur. Pour U3 il suffit de couder les broches et le souder. Insérez une vis et un écrou pour le plaquer contre le circuit imprimé.

Pour l'afficheur, soudez une barrette femelle au pas de 2,54 mm à 16 contacts sur le circuit imprimé. Soudez une barrette mâle au pas de 2,54 mm à 16 contacts sur l'afficheur. Insérez l'afficheur en utilisant des entretoises de 15 mm de hauteur.

Terminez le montage en plaçant les boutons adaptés pour un montage vertical (ils doivent avoir les broches pliées à 90 °), la prise d'alimentation et la prise jack 3,5 mm stéréo pour les entrées audio. Insérez le microcontrôleur dans son support, le repère en « U » doit être vers R9.

Chargez le « firmware » (c'est-à-dire le programme fonctionnant dans le microcontrôleur, c'est un fichier portant l'extension « .hex ») en utilisant le connecteur pour la programmation en circuit (ICSP).

Notez qu'il vous faudra un programmateur compatible avec la programmation en circuit des PIC. Pour alimenter le montage, utilisez une alimentation fournissant 9 VDC à 15 VDC avec un courant d'au moins 100 mA.

Pour tester le montage, allumez un récepteur FM ou un poste radio FM. Positionnez-le dans la même pièce que l'émetteur et choisissez une fréquence où il n'y a pas de diffusions d'autres radios (plutôt vers les extrémités de la bande FM).

Alimentez le montage, réglez la fréquence d'émission vers 87 MHz à l'aide de SW3 et SW4, lorsque vous vous approchez de la fréquence sur laquelle est réglé le récepteur vous devez percevoir un silence dans le HP. Connectez la sortie d'un lecteur MP3 ou d'un lecteur de CD vers la prise jack de l'émetteur et vérifiez que le récepteur reproduise la musique que vous émettez.

RASPBERRYPI 2 & 3

de Marco MAGAGNIN

Dès sa sortie en début d'année 2012, le RaspberryPi s'est vendu à des dizaines de milliers d'exemplaires, une performance respectable. Pour quelques dizaines d'euros, ce nano-ordinateur offre des performances équivalentes à un PC datant de quelques années. Il existe plusieurs types de RaspberryPi en vente, récemment une nouvelle version 3 est sortie en parallèle de la version 2 toujours en vente, de plus un marché de l'occasion se développe. Nos lecteurs nous ont posé d'innombrables questions, nous allons donc dans cet article faire le point sur le RaspberryPi afin d'effectuer votre choix en fonction de vos besoins.

Le RasperryPi étant un produit informatique, doté de microprocesseurs, suit la fameuse « **Loi de Moore** ». **Gordon Earle Moore**, cofondateur d'**Intel** en 1968, a constaté en 1965 que le nombre de transistors contenus dans un microcontrôleur **doublait tous les 18 mois**.

À l'époque le « microcontrôleur » le plus performant comportait 64 transistors. En figure 1, vous pouvez voir l'évolution de cette Loi au cours du temps avec quelques variations, cependant la prédiction de la Loi de Moore a été respectée jusqu'en 2010. De nos jours nous atteignons des limites physiques que nous ne pourrons franchir que si la recherche se tourne vers la physique quantique.

Logiquement, le RaspberryPi depuis sa sortie a suivi cette Loi avec l'apparition de « clones » ayant des processeurs plus

puissants mais des applications difficilement compatibles les unes avec les autres. Dernièrement, même la Fondation RaspberryPi a apporté des modifications dans les distributions au niveau de la partition « **root** » avec l'apparition du fichier « **.dtb** » pour la configuration matérielle.

Cependant dans le monde de crise que nous connaissons actuellement, des idées et des solutions révolutionnaires surgiront, qui marqueront le chemin de la prochaine période de croissance et de prospérité.

Tout d'abord, nous allons voir ce qui a changé avec l'apparition du RaspberryPi 2 ainsi qu'avec la nouvelle version RaspberryPi 3, notamment la structure interne, la taille, le montage, l'agencement des connecteurs et le connecteur de carte SD. Vous pourrez réutiliser des accessoires ou des boîtiers existants. Mais tout d'abord avant d'entrer dans les caractéristiques détaillées de chacune des versions, examinons les grandes différences entre le RaspberryPi 2 et le RaspberryPi 3.

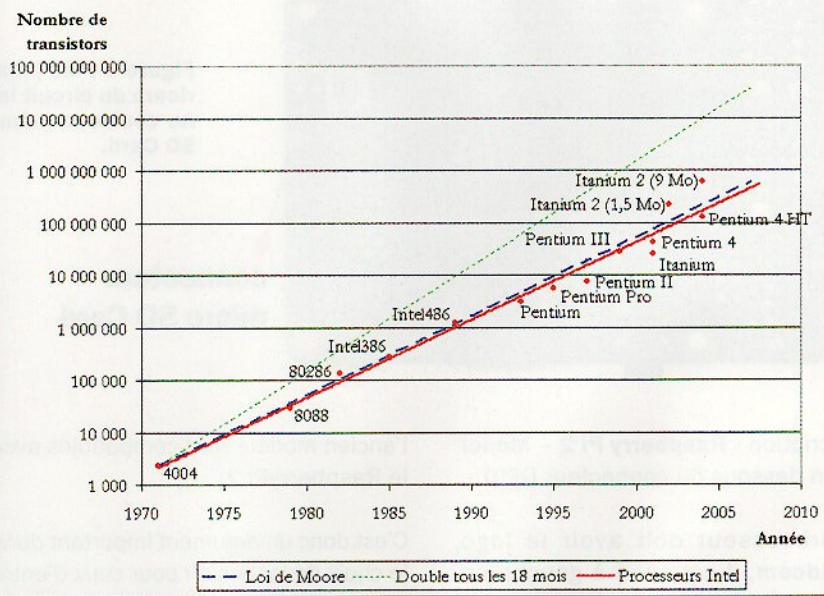
Le RaspberryPi 2 coute aux alentours de 40 €, il est encore disponible à la vente neuf mais un marché de l'occasion se développe. Ses principales caractéristiques sont :

- processeur 32 bits 4 cœurs ARM Cortex-A7 ;
- fréquence : 900 MHz ;
- RAM : 1 Go ;
- Wi-Fi : Non ;
- Bluetooth : Non ;
- alimentation : 5 V / 2 A ;
- stockage : carte MicroSD ;
- ports USB : 4.

Le RaspberryPi 3 coute aux alentours de 50 €, ses principales caractéristiques sont :

- processeur : 64 bits 4 cœurs ARM Cortex-A53 ;
- fréquence : 1,2 GHz ;
- RAM : 1 Go ;
- Wi-Fi : Oui ;
- Bluetooth : Oui, 4.1 ;
- alimentation : 5 V / 2,5 A ;
- stockage : carte MicroSD ;
- ports USB : 4.

Figure 1 : évolution du nombre de transistors dans un circuit au cours du temps.



Nous constatons immédiatement que la **version 3 est en 64 bits** donc forcément plus rapide et qu'elle est dotée de **connexions sans fil Wi-Fi** et **Bluetooth** pour la communication. Il est donc évident que si vous voulez découvrir et apprendre à programmer le RaspberryPi, optez pour la version 2 ; vous pourrez en trouver d'occasion à un prix attractif.

Par contre si vous êtes expérimenté dans le domaine et que votre application nécessite une communication sans fil, optez pour la version 3.

Le RaspberryPi 2

Si vous désirez vous procurer un RaspberryPi 2 d'occasion, il est **essentiel de bien reconnaître la carte** que vous avez entre les mains afin de ne pas confondre un RaspberryPi 2 avec un ancien modèle RasperryPi B+. Pour cela vous devez analyser les éléments suivants, visibles aux figures 2 et 3, ainsi que le positionnement de certains composants principaux.

Premièrement, vérifiez que sur la face supérieure du circuit imprimé figure

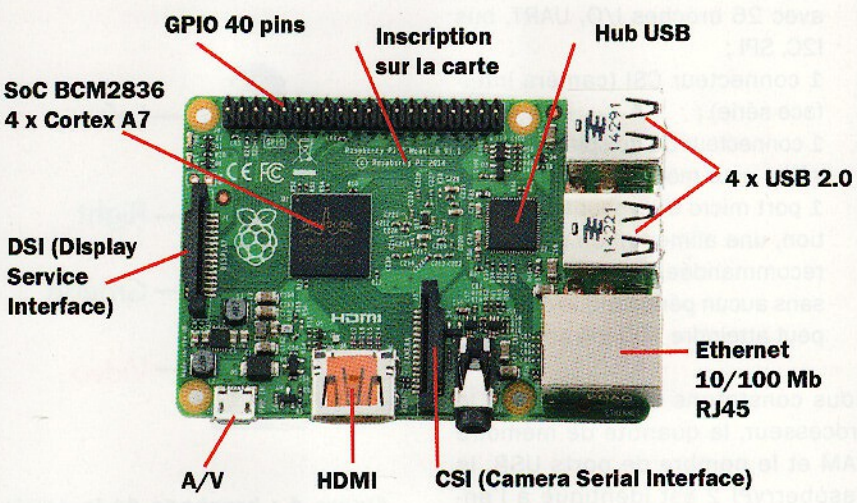


Figure 2 : les points importants à vérifier sur la partie supérieure du circuit imprimé pour reconnaitre un RaspberryPi 2.

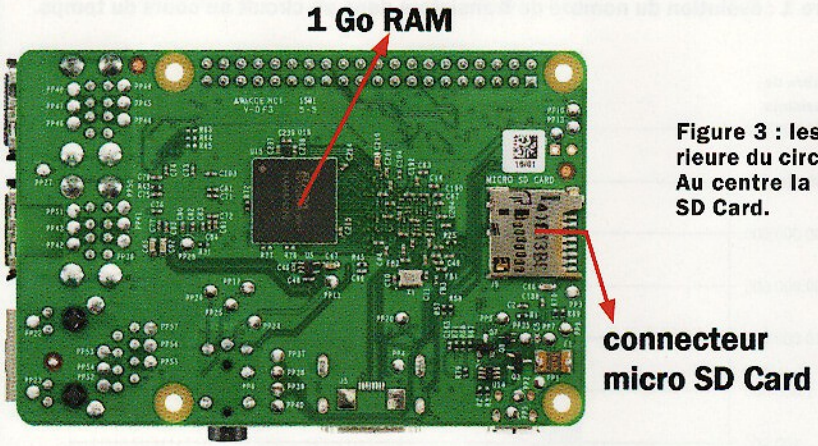


Figure 3 : les points importants à vérifier sur la partie inférieure du circuit imprimé pour reconnaitre un RaspberryPi 2. Au centre la mémoire RAM et à droite le connecteur micro SD Card.

l'inscription « **Raspberry Pi 2 – Model B** » en dessous du connecteur GPIO.

Le processeur doit avoir le logo **Broadcom**, il est situé à gauche en dessous du connecteur GPIO et identifié par « SoC BCM2836 4 X Cortex A7 » sur la figure 2. La mémoire RAM et le connecteur micro SD Card sont situés sur la partie inférieure du circuit imprimé (voir la figure 3).

Les caractéristiques du RaspberryPi 2 sont les suivantes :

- processeur : Broadcom BCM2836 avec CPU ARM Cortex-A7 cadencé à 1 GHz ;
- mémoire : 1Go LPDDR2 SDRAM ;
- processeur vidéo : VideoCore IV ;
- 4 ports USB ;
- 1 prise Ethernet 10/100 Mb ;
- 1 sortie vidéo HDMI et A/V (le brochage est visible en figure 4) ;
- 1 connecteur GPIO de 40 broches avec 26 broches I/O, UART, bus I2C, SPI ;
- 1 connecteur CSI (caméra interface série) ;
- 1 connecteur DSI (display interface série) pour moniteur LCD ;
- 1 port micro USB pour l'alimentation, une alimentation de 2 A est recommandée. La consommation sans aucun périphérique connecté peut atteindre 700 mA sous 5 V.

Nous constatons que mis à part le processeur, la quantité de mémoire RAM et le nombre de ports USB, le RaspberryPi 2 est identique à l'ancien modèle RasperryPi B+. Cela veut dire que les applications qui ont été développées par la communauté sur l'ancien modèle sont compatibles avec le RaspberryPi 2.

C'est donc un argument important dans le choix de la version pour ceux d'entre vous qui désirez s'initier au RaspberryPi.

La figure 5 représente le brochage du connecteur GPIO du RaspberryPi 2, qui est identique à celui du RaspberryPi B +.

En figure 6, nous pouvons vérifier la présence du processeur à 4 cœurs (quad core) grâce à la commande :

```
cat /proc/cpuinfo
```

Normalement, lors du démarrage du RaspberryPi 2, un message dans l'écran de « **boot** » indique la présence du processeur à 4 cœurs.

La figure 7 représente la quantité de mémoire disponible et utilisée, grâce à la commande « **free** » ».

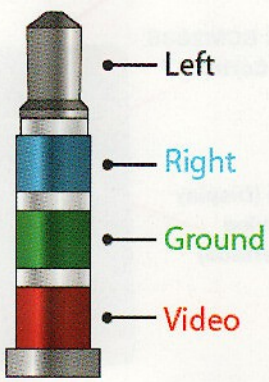


Figure 4 : brochage de la sortie vidéo. En gris la sortie audio gauche, en bleu la sortie audio droite, en vert la masse et en rouge la sortie vidéo.

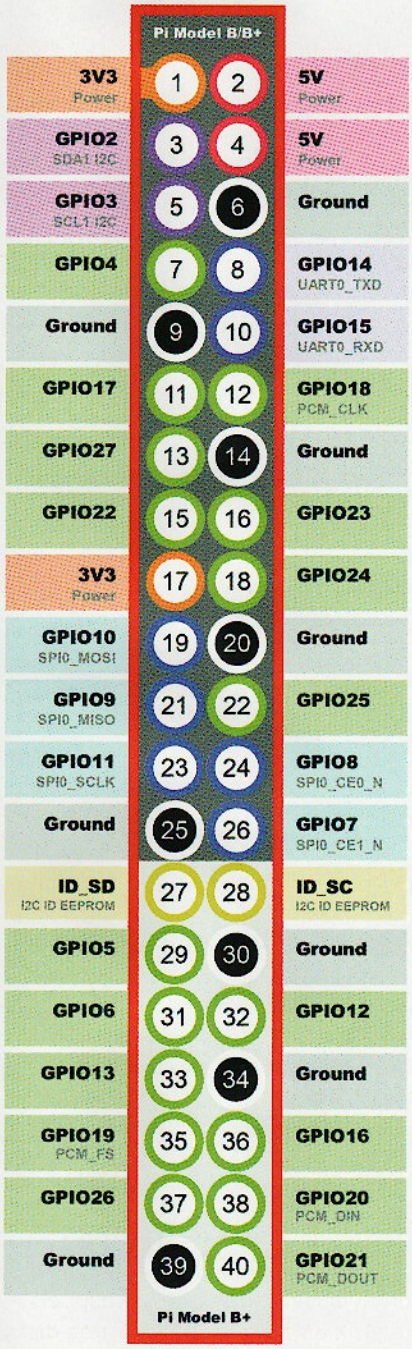


Figure 5 : brochage du connecteur GPIO du RaspberryPi 2.

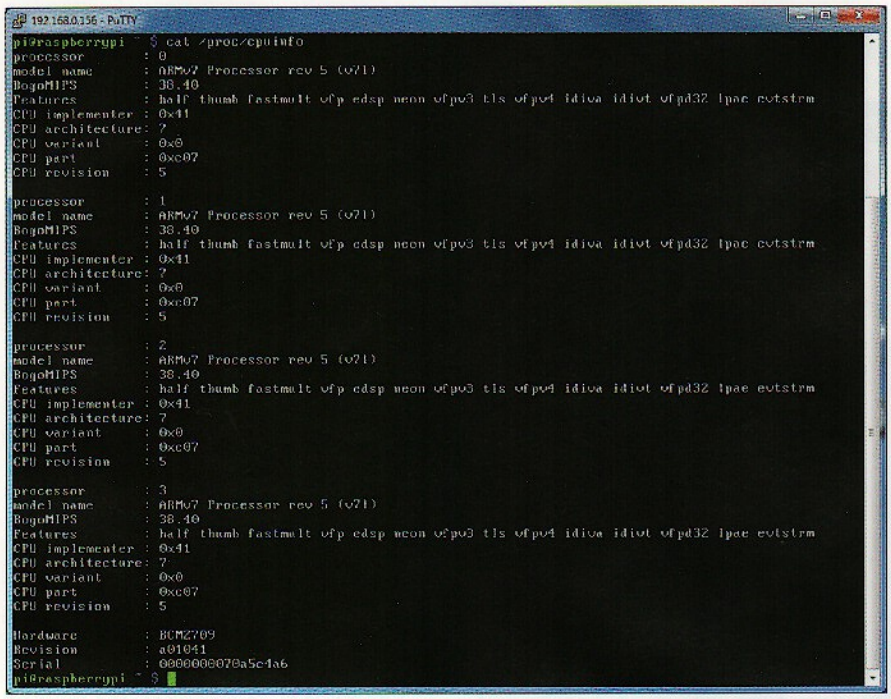


Figure 6 : détail de la configuration du processeur, vous pouvez voir la présence des 4 cœurs.

Compatibilité du RaspberryPi 2

La Fondation, en cherchant à rentabiliser les investissements, a développé des logiciels, des systèmes d'exploitation et des applications optimisés pour le processeur Broadcom BCM2835 de la série précédente.

L'investissement principal étant l'adaptation de la distribution GNU/Linux Debian pour fonctionner sur un matériel de type « ARMv6 », y compris un support matériel pour les opérations en virgule flottante et une stabilité du système lui-même.

De même pour optimiser le fonctionnement du RaspberryPi, une grande quantité de librairies Open Source et d'applications telles que WebKit, LibreOffice, Scratch, Pixman, XBMC/Kodi, libav et PyPy ont été développées.

Enfin, les efforts déployés pour développer et améliorer constamment les outils afin de rendre la vie plus facile aux utilisateurs novices (ou non) comme par exemple « **NOOBS** » qui est un utilitaire conçu pour rendre le premier démarrage d'un RaspberryPi facile en gérant les différentes distributions GNU/Linux pour passer d'une configuration à une autre ou le package de mise à jour de la partition de root « **rpi-update** » peuvent poser certains problèmes lorsqu'ils sont utilisés sur une nouvelle configuration matérielle.

Basculer vers un nouveau matériel signifie souvent perdre ce patrimoine de librairies et de logiciels, il faut donc recommencer à partir de zéro avec les nouvelles plates-formes dont les caractéristiques ne sont pas complètement compatibles avec les anciens modèles.

Lors de l'élaboration du RaspberryPi 2 la compatibilité a été pensée, et le SoC BCM2836 avec le CPU Cortex A7 reste en grande partie compatible avec les anciennes applications. D'autre part la mémoire 1 Go a été placée sur la partie inférieure du circuit imprimé, permettant un meilleur refroidissement du processeur.

Cependant il a fallu adapter le noyau de Raspbian afin qu'il respecte les spécifications matérielles du Soc BCM2836. Mais grâce à l'architecture GNU/Linux, qui présente une séparation entre l'espace « **noyau** » (kernel) et l'espace « **utilisateur** », cette opération a été relativement facile. L'objectif est de simplifier l'utilisation du système, en normant l'interaction entre les applications, le matériel et les périphériques.

Pour résumer, le noyau (kernel) du système d'exploitation gère l'interaction entre les différents matériels (CPU, mémoire, bus, périphériques physiques) et fournit une première couche de « **drivers** » (pilotes) standardisés.

La couche suivante permet de simplifier la communication avec les périphériques, et de rendre « visible » l'ensemble par l'utilisateur final à travers un système de **fichiers virtuels**

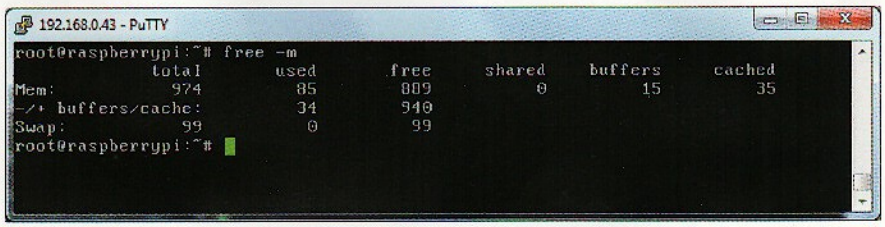


Figure 7 : avec la commande « free », vous pouvez visualiser la quantité de mémoire disponible et utilisée.

dans une « fenêtre » unique. Celle-ci permet à l'utilisateur d'interagir avec le système physique.

En fait, le système de **fichiers virtuels** constitue la frontière entre l'espace « **noyau** » (kernel) et l'espace « **utilisateur** » où nous écrivons et/ou exécutons nos applications.

L'architecture GNU/Linux a déjà été décrite à plusieurs reprises dans les anciens numéros de la revue, elle reste valable pour le RaspberryPi 2.

En ce qui concerne les composants externes au noyau, ils ont été optimisés pour le nouveau CPU afin de préserver la compatibilité avec les cartes RaspberryPi antérieures.

De même les librairies ont été mises à jour pour obtenir un meilleur équilibre entre la fonctionnalité, la stabilité et la performance du RaspberryPi 2.

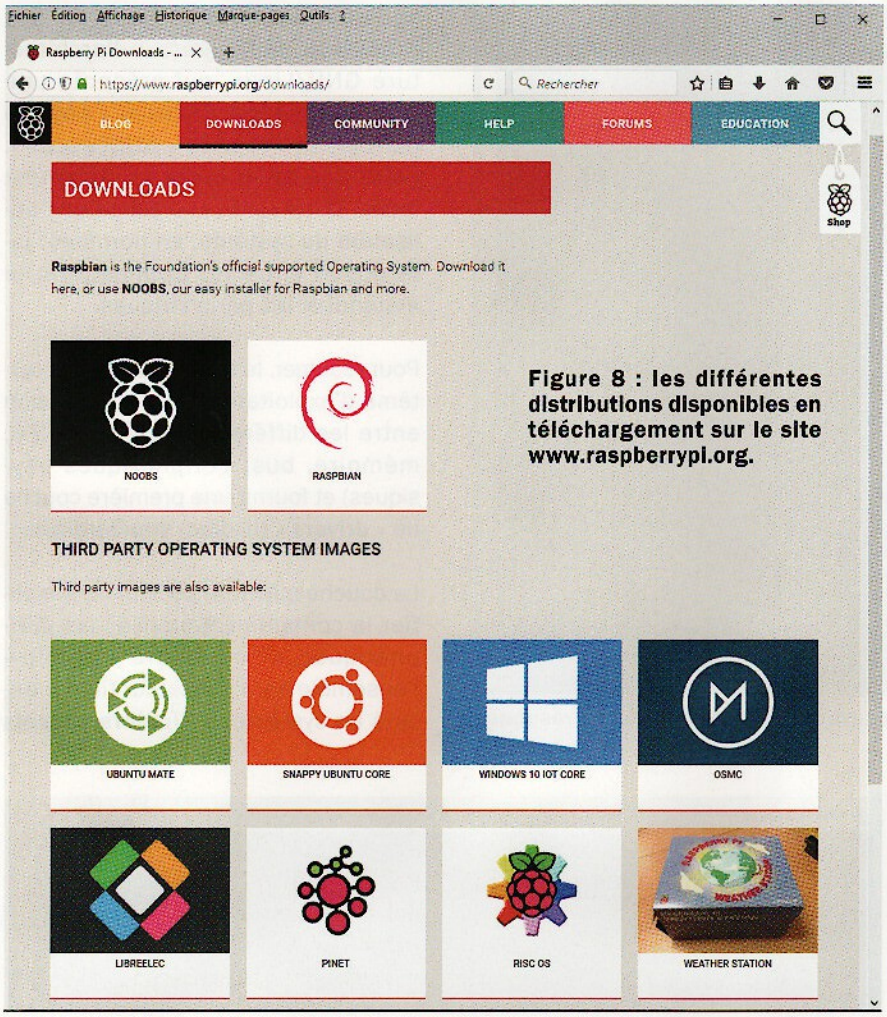


Figure 8 : les différentes distributions disponibles en téléchargement sur le site www.raspberrypi.org.

différents langages de programmation pris en charge par Visual Studio 2015 puis de les tester directement sur un PC équipé de Windows 10, avant de les exécuter sur le système final.

« Windows 10 IoT » dispose d'une véritable interface graphique, il peut exécuter des applications Windows dites universelles. Par contre l'édition « Windows 10 IoT » est mono-tâche et est dépourvue de bureau, d'écran démarrer ou d'un quelconque lanceur d'application. Il est cependant possible de faire fonctionner de véritables applications interactives, un lecteur multimédia par exemple.

Utilisons le RaspberryPi 2

Pour utiliser le RaspberryPi 2 nous avons plusieurs options. La première est de télécharger la dernière distribution **Raspbian** ou l'outil « **NOOBS** » sur le site **www.rasperrypi.org** (voir la figure 8) et transférer l'image sur une carte micro SD.

Attention vous devez utiliser une micro SD Card comme pour le RaspberryPi B +, insérez-la dans le slot correspondant.

D'autres distributions de systèmes d'exploitation sont disponibles pour le RaspberryPi 2 à l'adresse suivante :

https://www.raspberrypi.org/downloads.

Reportez-vous à la figure 8, vous y trouverez « **Windows 10 IoT Core** » pour Raspberry Pi 2, distribué gratuitement aux développeurs du moins pour l'instant.

Lorsque vous cliquez sur l'onglet « Windows 10 IoT Core », vous êtes redirigé vers le centre de développement de Microsoft (voir la figure 9).

« Windows 10 IoT » est une alternative à Linux en tant que système d'exploitation minimaliste pour des systèmes à usages spécifiques, la domotique, la robotique etc.

Par rapport à Linux, il permet de développer des applications avec

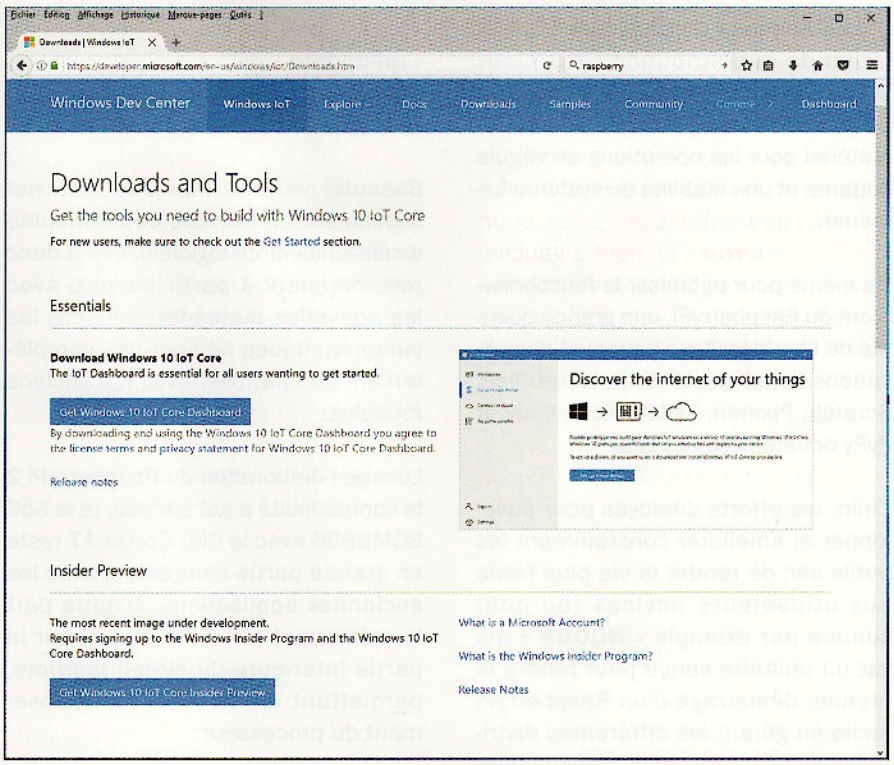


Figure 9 : ici la page web du centre de développement de Microsoft, suivez les étapes pour télécharger « Windows 10 IoT Core ».

Cette méthode est certainement la plus adaptée pour ceux qui découvrent pour la première fois le RaspberryPi. De cette manière, vous allez installer tous les paquets dont vous avez besoin et configurer le RaspberryPi 2 selon vos besoins (pour plus de détails reportez-vous aux anciennes revues d'Electronique et Loisirs Magazine à partir du numéro 123).

Si vous avez opté pour un RaspberryPi 2 d'occasion qui comporte une carte SD avec une distribution Raspbian ancienne, voici la méthode à suivre pour le mettre à jour. Dans un premier temps, nous vous recommandons de créer une sauvegarde de la carte SD, en utilisant le logiciel « **WinDiskImager** ».

Ensuite, mettez à jour la distribution de manière habituelle (éventuellement reportez-vous aux anciennes revues comme indiqué précédemment), l'utilisateur étant connecté en tant que « **root** » à l'aide des commandes suivantes :

```
apt-get update
apt-get upgrade
apt-get dist-upgrade
```

Si la distribution est relativement ancienne, le processus de mise à jour prendra un certain temps.

Notez que la distribution **Raspbian** adopte une **interface graphique** « **LXDE** », dont la barre des applications se situe en haut de l'écran, un bureau avec des applications préférées et un menu « **Démarrer** » en haut à gauche. Pour mettre à jour l'interface graphique de la distribution tapez la commande :

```
apt-get install raspberrypi-ui-mods
```

Après cette opération, vous devez « nettoyer » la distribution avec les commandes :

```
apt-get autoremove
```

qui supprime les paquets qui ne sont plus utilisés et avec :

```
apt-get purge
```

qui élimine les fichiers de configuration qui ne sont plus nécessaires.

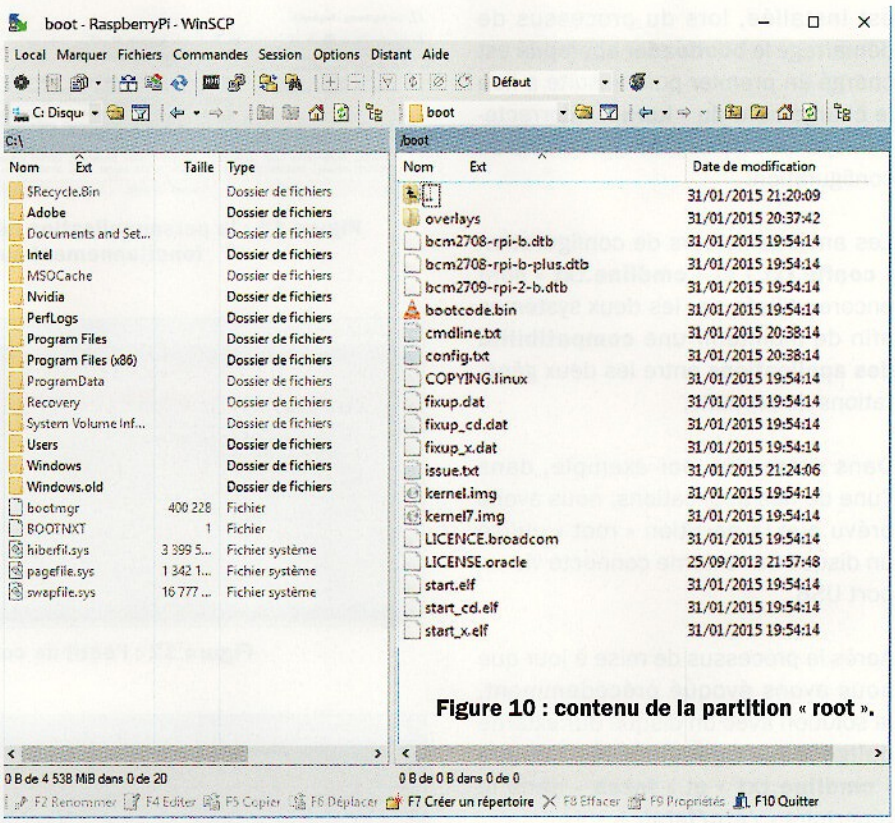


Figure 10 : contenu de la partition « root ».

Enfin, mettons à jour la partition « **root** » avec la commande :

```
rpi-update
```

Si vous obtenez une erreur de type « **command not found** », vous devez installer l'outil de mise à niveau avec la commande :

```
apt-get install rpi-update
```

Une fois la mise à jour correctement effectuée, redémarrez le RaspberryPi 2, et testez une application. Pour mesurer l'utilisation du CPU tapez la commande :

```
top
```

Pour sortir tapez : **q**

Ensuite pour afficher l'utilisation de la mémoire, tapez la commande : **free –m**

Voici ce que nous considérons comme étant le vrai concept de compatibilité ascendante.

Pour mieux comprendre comment la distribution Raspbian a été configurée pour atteindre cet objectif, analysons la partition « root » dont le contenu est visible en figure 10.

Par rapport aux anciennes distributions, nous trouvons dans la partition « root » un certain nombre de fichiers supplémentaires qui sont :

- le fichier **image du noyau** (kernel) Linux ;
- le fichier « **start.elf** » ;
- le fichier « **config.txt** » ;
- le fichier « **cmdline.txt** ».

Fondamentalement, nous trouvons l'image d'un second « kernel » nommé « **kernel7.img** » qui fait référence au CPU Cortex A7 et certains autres fichiers portant l'extension « **.dtb** » (device tree blob). Ces fichiers gèrent l'arborescence matérielle, c'est-à-dire les **ressources des périphériques** et le **chargement des modules** pendant le processus de « **boot** » (démarrage).

Pour résumer, nous pouvons dire que les fichiers « **.dtb** » sont une sorte de base de données compilée au format binaire qui contient la description de l'arborescence (structure) matérielle dans un standard compréhensible par le « bootloader » (équivalent du BIOS d'un PC) et le « kernel » (noyau).

Avec cette configuration, selon la version du RaspberryPi où la carte SD

est installée, lors du processus de démarrage le **bootloader** approprié est chargé en premier puis ensuite a lieu le chargement du « **kernel** » correctement paramétré à l'aide des fichiers de configuration.

Les anciens fichiers de configuration « **config.txt** » et « **cmdline.txt** » sont encore utilisés par les deux systèmes afin de maintenir une **compatibilité des applications** entre les deux générations de produits.

Dans notre cas, par exemple, dans l'une de nos applications, nous avons prévu que la partition « root » utilise un disque dur externe connecté via un port USB.

Après le processus de mise à jour que nous avons évoqué précédemment, la solution avec un disque dur externe oblige à personnaliser les fichiers « **cmdline.txt** » et « **fstab** » dans le répertoire « **/etc/fstab** ».

Le fichier « **fstab** » (file systems table) est la **table des différents systèmes de fichiers**, il contient une liste des disques utilisés au démarrage et des partitions de ces disques. Pour chaque partition, il indique comment elle sera utilisée et intégrée à l'arborescence du système de fichiers global (voir la figure 11).

D'autre part, dans la dernière version de Raspbian, l'outil de configuration « **raspi-config** » a été amélioré avec de nouvelles fonctionnalités. Outre les fonctions habituelles qui vous permettent d'étendre la partition « root » (étendre l'espace mémoire à toute la carte SD), de lancer l'interface graphique au démarrage, d'overclocker la CPU, d'activer la « Camera PI »,

En sélectionnant « **Advanced Options** » (options avancées) situé en bas de la figure 12, vous entrez dans un second menu qui vous permet d'exécuter différentes configurations accessibles ultérieurement qu'à partir de lignes de commande.

En figure 13, vous pouvez voir les options du menu « Advanced Options » telles que « **Overscan** » qui gère les barres noires autour de l'écran, « **SSH** »

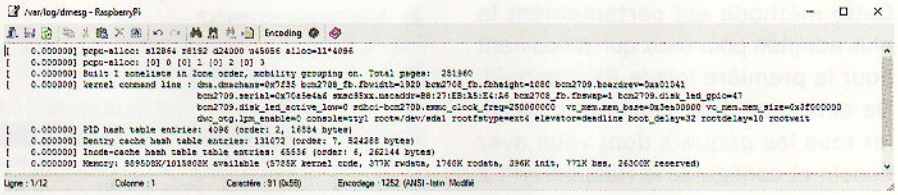


Figure 11 : la personnalisation de « cmdline.txt » et « fstab » permet le fonctionnement sur les 2 configurations.

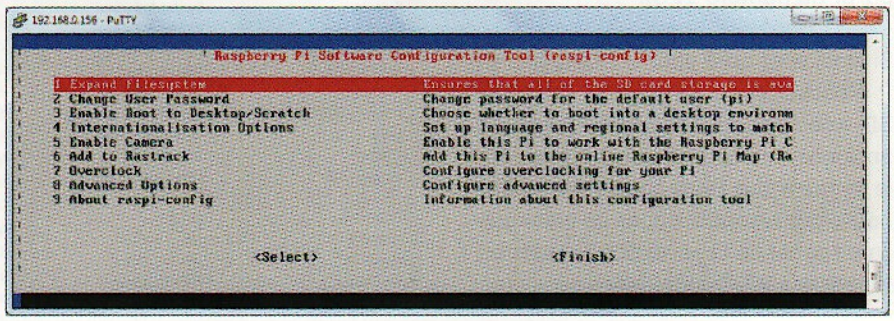


Figure 12 : l'outil de configuration « raspi-config ».

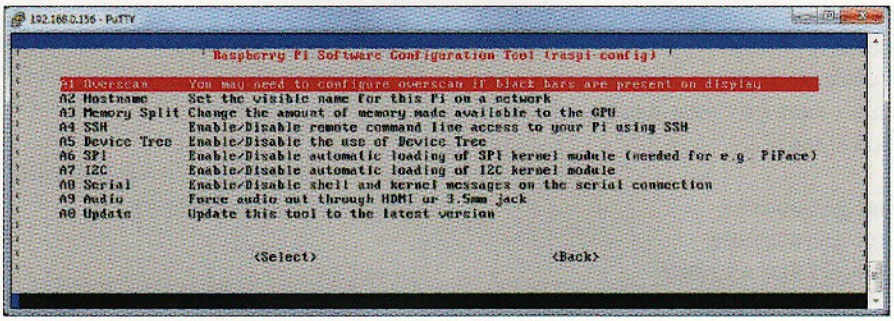


Figure 13 : le menu « Advanced Options ».

qui permet d'activer/désactiver le serveur pour l'accès distant, « **memory split** » qui permet d'assigner plus ou moins de mémoire au GPU vidéo, « **hostname** » qui permet de modifier le nom système, ou encore « **Audio** » qui configure la sortie audio vers le connecteur HDMI ou la prise jack 3,5 mm.

Une attention particulière doit être accordée aux options « **SPI** » et « **I2C** » qui permettent d'activer/désactiver les bus SPI et I2C.

L'ancienne méthode, de la « **blacklist** » et du chargement du driver avec « **modprobe** », doit être intégrée avec l'activation des bus I2C et/ou SPI en utilisant les fonctions mises à disposition.

En ce qui concerne nos futurs projets en développement, qui seront bientôt présentés dans les prochaines revues, vous pouvez voir en figure 14 l'écran principal d'une de nos applications.

Les figure 15 et 16 montrent l'utilisation du CPU respectivement sur le RaspberryPi B + et 2, les chiffres parlent d'eux-mêmes.

Figure 14 : une de nos applications en développement.

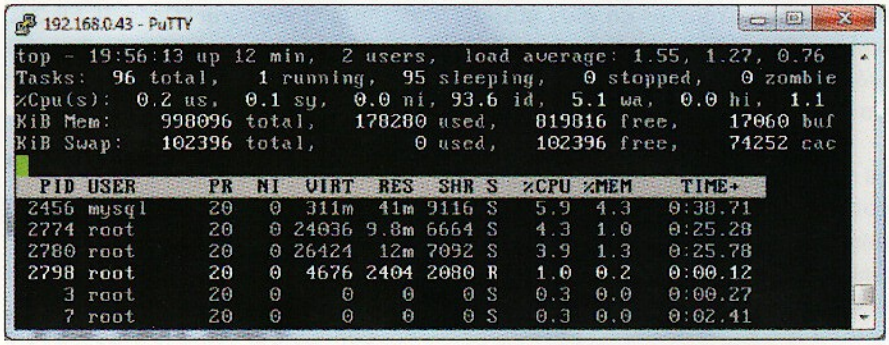


Figure 15 : utilisation du CPU sur le RaspberryPi B +.

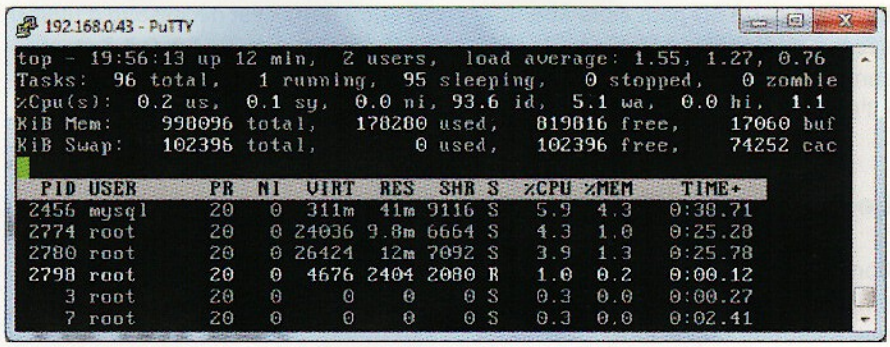


Figure 16 : utilisation du CPU sur le RaspberryPi 2.

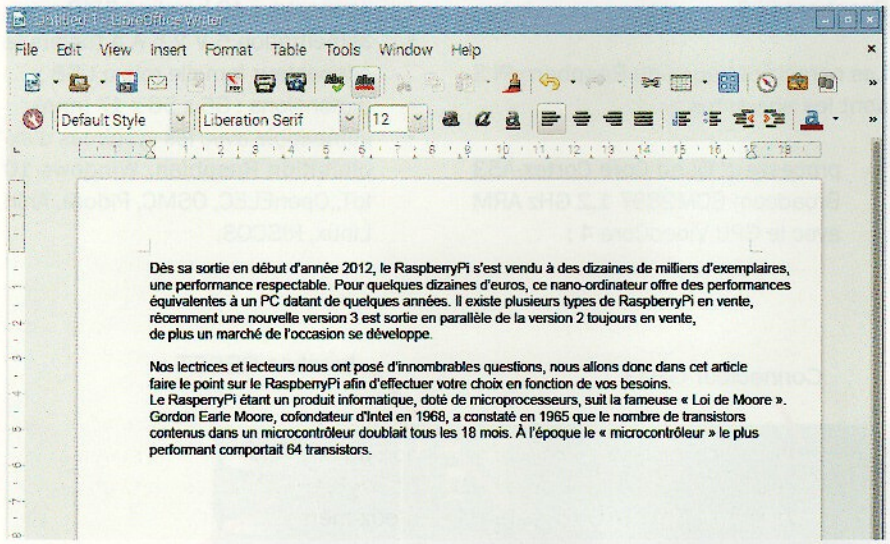


Figure 17 : fonctionnement de « LibreOffice » sur le RaspberryPi 2.

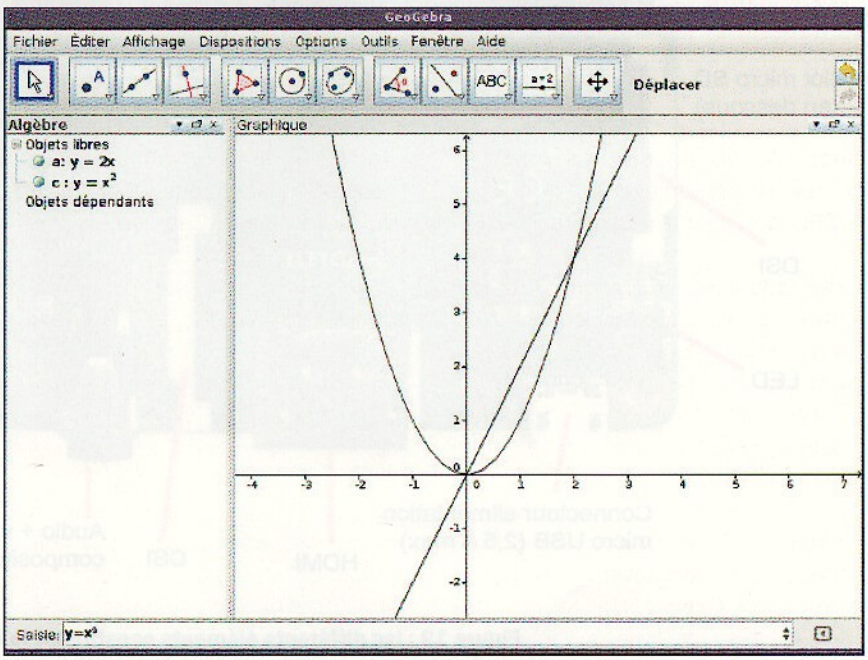


Figure 18 : fonctionnement de « Geogebra » sur le RaspberryPi 2, c'est un logiciel éducatif en mathématiques pour les élèves d'un niveau collège : algèbre, géométrie et calcul analytique.

En fait le RaspberryPi 2 peut être comparé à un ordinateur personnel bas de gamme, sur lequel vous pouvez exécuter des applications bureautiques de manière acceptable. Des applications comme « **LibreOffice** » ou « **Geogebra** » fonctionnent avec cependant une occupation importante des ressources (voir les figures 17 et 18).

Avec quelques RaspberryPi 2 qui représentent un investissement équivalent à l'achat de deux PC, il est possible de mettre en place un réseau informatique pour une classe d'élèves d'un collège, sans frais supplémentaires pour l'achat des licences des systèmes d'exploitation et des applications car le RaspberryPi est Open Source.

Le but de la Fondation RaspberryPi est de fournir des outils didactiques Open Source, très économiques et en phase avec les nouvelles technologies.

En outre, chaque élève entre dans la classe avec sa propre carte SD en bénéficiant de la mise à jour du dernier exercice sans interférer avec d'autres élèves.

La même carte SD peut alors être utilisée sur un RaspberryPi situé à la maison pour continuer le travail commencé à l'école et approfondir les cours.

Le RaspberryPi 3

Comme nous l'avons évoqué en début d'article, le RaspberryPi 3 représente une évolution importante en ce sens que l'architecture de son processeur est en 64 bits, contre 32 bits pour le RaspberryPi 2.

Le RaspberryPi 3 dispose d'un **processeur 4 cœurs** (quad core) cadencé à **1,2 Ghz** qui est **30 % plus rapide** que l'ancienne version.

Il peut donc exécuter un plus grand nombre d'opérations que les versions précédentes. Sa mémoire RAM est de 1 Go comme pour le RaspberryPi 2.

La nouveauté la plus importante réside dans le fait que le Raspberrypi3 dispose désormais d'une **connectivité sans fil Wi-Fi** à la norme **802.11.b/g/n** et **Bluetooth 4.1**.

D'autre part, il est nécessaire d'utiliser une alimentation de **5 V 2,5 A** au minimum. Le reste est identique aux versions précédentes, notamment le port GPIO, le processeur graphique, etc., ceci afin d'assurer une certaine compatibilité aux niveaux des connecteurs. Un nouveau modèle de connecteur micro SD Card est présent afin de libérer de l'espace. En figure 19 vous pouvez identifier les différents éléments du RaspberryPi 3.

Les caractéristiques du RaspberryPi 3 sont les suivantes :

- processeur Quad Core Cortex-A53 Broadcom BCM2837 1,2 GHz ARM avec le GPU VideoCore 4 ;
- le GPU supporte l'Open GL ES 2.0, l'accélération matérielle OpenVG et le décodage « H.264 » (standard Blu-ray) ;
- mémoire SDRAM LPDDR2 1 Go ;
- module Wi-Fi BCM43143 intégré ;
- module Bluetooth (BLE) intégré ;
- sortie vidéo HD 1080p ;
- connecteur femelle Ethernet RJ45 10/100 BaseT ;
- connecteur femelle vidéo/audio HDMI 1.3 et 1.4 ;
- jack femelle 3,5 mm pour sortie audio/vidéo composite ;
- 4 connecteurs USB 2.0 (1,2 A) ;
- connecteur MPI CSI-2 15 pôles pour Camera RaspberryPi haute définition ;
- connecteur interface série 15 pôles pour écran (DSI) ;
- connecteur femelle micro-SD Card ;
- boot à partir de la carte SD ;
- connecteur 40 broches GPIO ;
- alimentation 5 V 2,5 A à travers le connecteur femelle micro USB ;
- dimensions : 86 x 56 x 17 (mm) ;
- compatible avec les systèmes d'exploitation Raspbian, Windows 10 IoT, OpenELEC, OSMC, Pidora, Arch Linux, RISCOS.

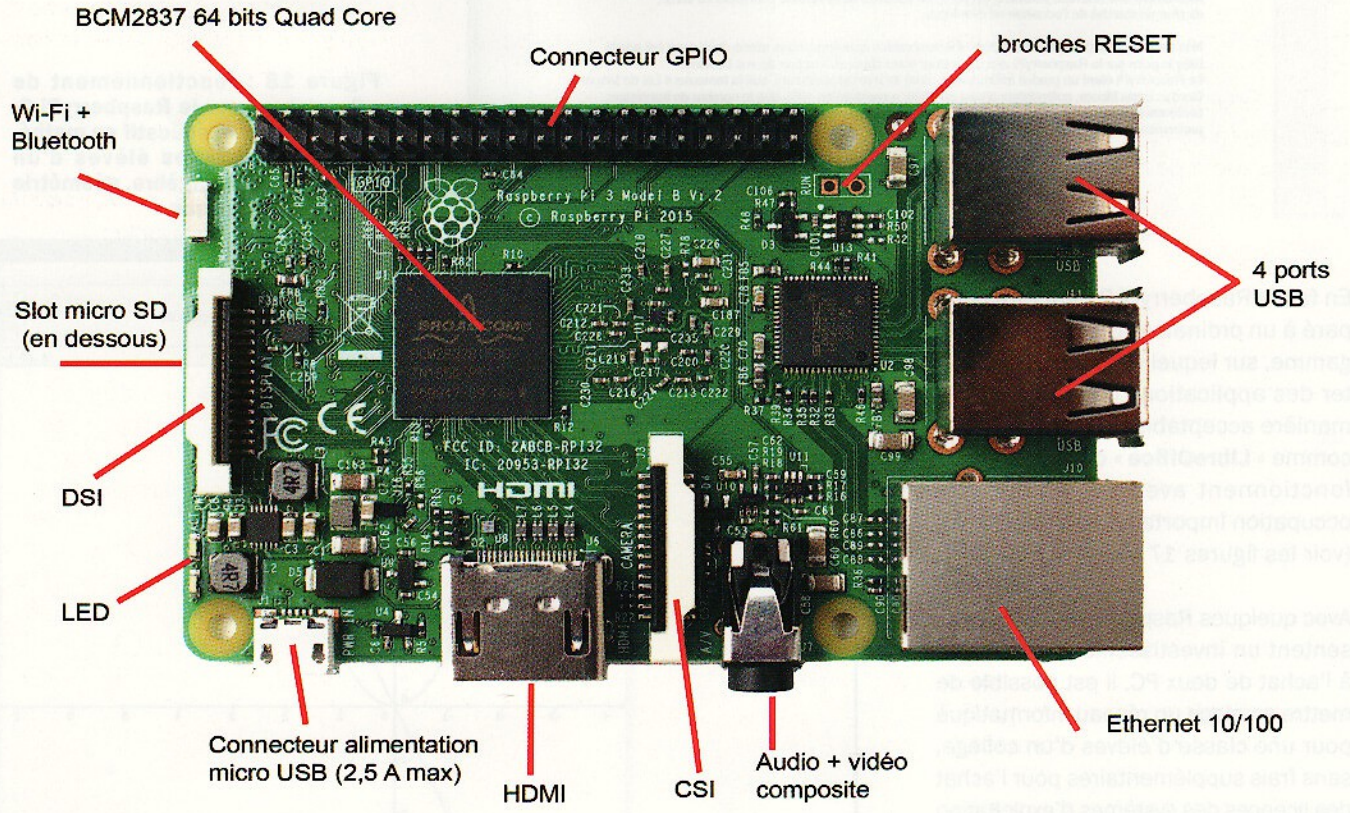


Figure 19 : les différents éléments constituant le RaspberryPi 3

Nous allons aborder les points essentiels qui différencient le RaspberryPi2 de la version 3, à savoir la connectivité sans fil. Pour l'installation de Raspbian sur le RaspberryPi 3, la procédure est identique aux anciennes versions (reportez-vous aux anciens numéros d'Electronique et Loisirs Magazine).

Nous attirons votre attention sur le fait que la version de Raspbian téléchargeable à l'adresse **https://www.raspberrypi.org/downloads** est en **32 bits** (du moins à la date où ces lignes sont écrites), ce qui ne pose pas de problème d'installation et de fonctionnement sur une **architecture processeur 64 bits**. De plus les anciennes applications fonctionneront normalement.

Cependant **il est possible que certains drivers ne fonctionnent pas correctement**. Par contre dès que la version **64 bits** de **Raspbian** sera disponible, vous pourrez l'installer sur le RaspberryPi 3, cependant **certaines anciennes applications développées sur les versions antérieures risquent de ne plus fonctionner**.

L'idée est donc de préparer une carte SD avec Raspbian en version 32 bits et une autre carte avec la version 64 bits. Vous pourrez ainsi utiliser la version de Raspbian selon vos besoins en prenant soin au préalable d'éteindre le RaspberryPi 3 avant de changer la carte.

Configurer le Wi-Fi sur le RaspberryPi 3

Sur les anciennes versions du RaspberryPi il était nécessaire d'utiliser une clé Wi-Fi connectée sur un port USB pour obtenir une connexion sans fil. Avec le **RaspberryPi 3 le Wi-Fi est intégré**, et donc la connexion à un réseau sans fil s'effectue **sans clé Wi-Fi.**

La première des choses à faire est de mettre à jour la distribution Raspbian, il est possible que les drivers nécessaires au bon fonctionnement du Wi-Fi et Bluetooth ne soient pas installés.

Tapez les commandes habituelles, notez que le RaspberryPi doit être connecté à Internet à travers la prise Ethernet (reportez-vous aux articles parus dans la revue).

```
apt-get update
apt-get upgrade
apt-get dist-upgrade
```

Une fois la mise à jour effectuée, vous avez deux manières de vous connecter.

1. connexion Wi-Fi via l'interface graphique

L'icône réseau se situe en haut à droite de l'écran, à côté de l'horloge. Débranchez le câble Ethernet avant de continuer.

Cliquez sur l'icône Wi-Fi, une liste de réseaux disponibles doit apparaître, il faut trouver le vôtre, par exemple dans notre cas nous trouvons le réseau suivant « Freebox-7D42B0 » (vous aurez un nom de réseau différent).

Une fois votre réseau identifié, sélectionnez-le et tapez votre mot de passe (normalement il est donné par votre fournisseur d'accès).

Nous vous rappelons que le **SSID correspond au nom du réseau.**

Une fois le champ « mot de passe » complété, cliquez sur le bouton OK.

Le RaspberryPI 3 établit la connexion, vous devez voir l'icône Wi-Fi devenir bleue, cela veut dire que la connexion est établie.

2. connexion Wi-Fi à l'aide de lignes de commandes

Tout d'abord vous devez installer les modules qui permettent de reconnaitre les composants installés, tapez la commande suivante :

```
apt-get install apt-utils firmware-brcm80211 pi-bluetooth wpa-supplicant
```

Vérifiez que l'interface soit bien reconnue à l'aide de la commande :

```
ifconfig –a
```

L'interface « **wlan0** » doit apparaître, puis vous devez chercher les réseaux Wi-Fi à portée, pour cela vous devez installer les outils Wi-Fi avec la commande suivante :

```
apt-get install iw wireless-tools
```

Ensuite activez l'interface à l'aide de la commande :

```
ifconfig wlan0 up
```

Puis cherchez les réseaux disponibles afin de vérifier que le Wi-Fi soit correctement configuré et que votre « box » apparaisse dans la liste, tapez la commande :

```
iwlist wlan0 scan | grep ESSID
```

Editez le fichier « **wpa_supplicant.conf** » avec la commande suivante :

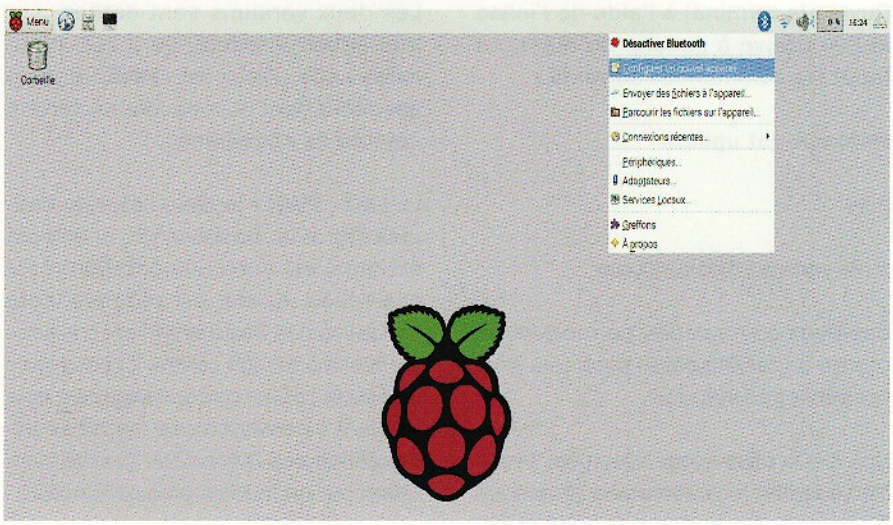


Figure 20 : l'icône avec le symbole du Bluetooth dans le coin supérieur droit de l'écran

sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

Allez à la fin du fichier, il ressemble à ceci :

```
network={
   ssid= « nom de la box »
   psk= « mot de passe »
   key_mgmt=WPA-PSK
}
```

Remplacez « **nom de la box** » par votre SSID et « **mot de passe** » par votre mot de passe. Si votre box utilise une clé de type « **WEP** » à la place d'une clé de type « **WPA/WPA2** », insérez la valeur **NONE** dans **key_mgmt** comme ceci : **key_mgmt=NONE.**

Une fois les modifications effectuées, enregistrez le fichier en appuyant sur « **CTRL+O** », puis quittez avec « **CTRL+X** ».

Arrêtez le Raspberry, enlevez le câble Ethernet et redémarrez.

Eventuellement activez le Wi-Fi avec la commande **fconfig wlan0 up.**

Configurer le Bluetooth sur le RaspberryPi 3

Par défaut le Bluetooth n'est pas activé dans Raspbian. Contrairement au Wi-Fi, il n'y a pas d'interface graphique pour configurer le Bluetooth, vous devez donc utiliser des lignes de commandes.

Ouvrez le terminal à l'aide de l'icône qui se trouve à gauche du menu, et tapez les commandes suivantes :

sudo apt-get update

puis

sudo apt-get dist-upgrade

Ces deux commandes permettent de mettre à jour Raspbian ainsi que tous les packages installés.

Une fois la mise à jour effectuée, il vous faut installer le gestionnaire graphique Bluetooth. Normalement celui-ci est déjà installé si vous avez suivi la procédure Wi-Fi décrite plus haut.

Sinon tapez la commande :

sudo apt-get install pi-bluetooth

Ensuite vous devez installer le gestionnaire graphique « **blueman** » à l'aide de la commande suivante :

sudo apt-get install bluetooth bluez blueman

Redémarrez le RaspberryPi 3, vous devez voir une icône avec le symbole du Bluetooth dans le coin supérieur droit de l'écran. Si ce n'est pas le cas, allez dans « **Menu → Préférences → Gestionnaire Bluetooth** » (voir la figure 20).

Maintenant, vous allez appairer le RaspberryPi 3 à un périphérique Bluetooth. Pour cela vérifiez que le RaspberryPi 3 a détecté les périphériques Bluetooth à portée, en ouvrant le gestionnaire Bluetooth (Menu → Préférences).

Cliquez sur l'onglet « **Adaptateur** » puis sur « **Préférences** », sélectionnez l'option « **Toujours visible** » puis fermez les fenêtres. À partir du périphérique (cela peut être un smartphone, une tablette, etc.) faites un appairage du Bluetooth, vous devez voir une notification dans le coin supérieur droit de l'écran du RaspberryPi 3.

Détection des problèmes sur les RaspberryPi 2 & 3

Les deux versions sont dotées, sur leur circuit imprimé, de 2 LED nommées « **ACT** » (verte) et « **PWR** » (rouge) qui permettent de résoudre un certain nombre de problèmes.

La LED « **PWR** » indique l'**état de l'alimentation** du RaspberryPi, si elle est éteinte c'est que l'alimentation n'est pas reliée. Si elle est allumée, cela indique que le RaspberryPi est correctement alimenté. Par contre, il peut arriver que cette LED se mette à clignoter, cela provient surement d'une mauvaise alimentation trop faible. Changez l'alimentation par un modèle plus performant.

La LED « **ACT** » est éteinte lorsqu'il n'y a pas de carte SD, elle clignote lors de l'accès à la carte SD en fonctionnement normal.

La LED « **ACT** » est **constamment allumée**, cela veut dire que le fichier « **start.elf** » est **endommagé**. Essayez de formater la carte SD et de réinstaller Raspbian (sur la carte).

La LED « **ACT** » **clignote** plusieurs fois :

- **3 fois** : le fichier « start.elf » est introuvable ;
- **4 fois** : le fichier « start.elf n'est pas démarré ;
- **7 fois** : le fichier « kernel.img » ou « kernel7.img » est introuvable ;
- 8 fois : la SDRAM n'est pas reconnue.

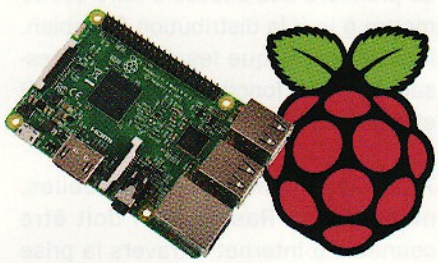
Enfin, le **port Ethernet** est doté de 2 LED indiquant l'**état de la connexion** et l'**activité du réseau**, elles peuvent vous donner quelques indications si vous avez un problème à ce niveau.

La **LED verte** sur le port Ethernet s'allume, cela indique qu'une **liaison physique est établie avec le réseau**. Lorsqu'elle clignote, une activité a lieu sur le port Ethernet (fonctionnement normal).

La **LED jaune s'allume** lorsqu'une connexion à **100 Mb/s** est détectée, sinon elle reste éteinte lorsque la vitesse de la connexion est de 10 Mb/s.

Nous arrivons au terme de cet article qui nous l'espérons éclaircira nos lecteurs qui nous ont posé d'innombrables questions sur le sujet.

Dans le prochain numéro nous apprendrons comment envoyer et recevoir des notifications « push » avec un RaspberryPi grâce au service « Pushetta », qui permet l'envoi de notifications sur un smartphone iOS ou Android. ■



TRANSFORMEZ VOTRE OSCILLOSCOPE EN ANALYSEUR DE SPECTRE

PREMIÈRE PARTIE

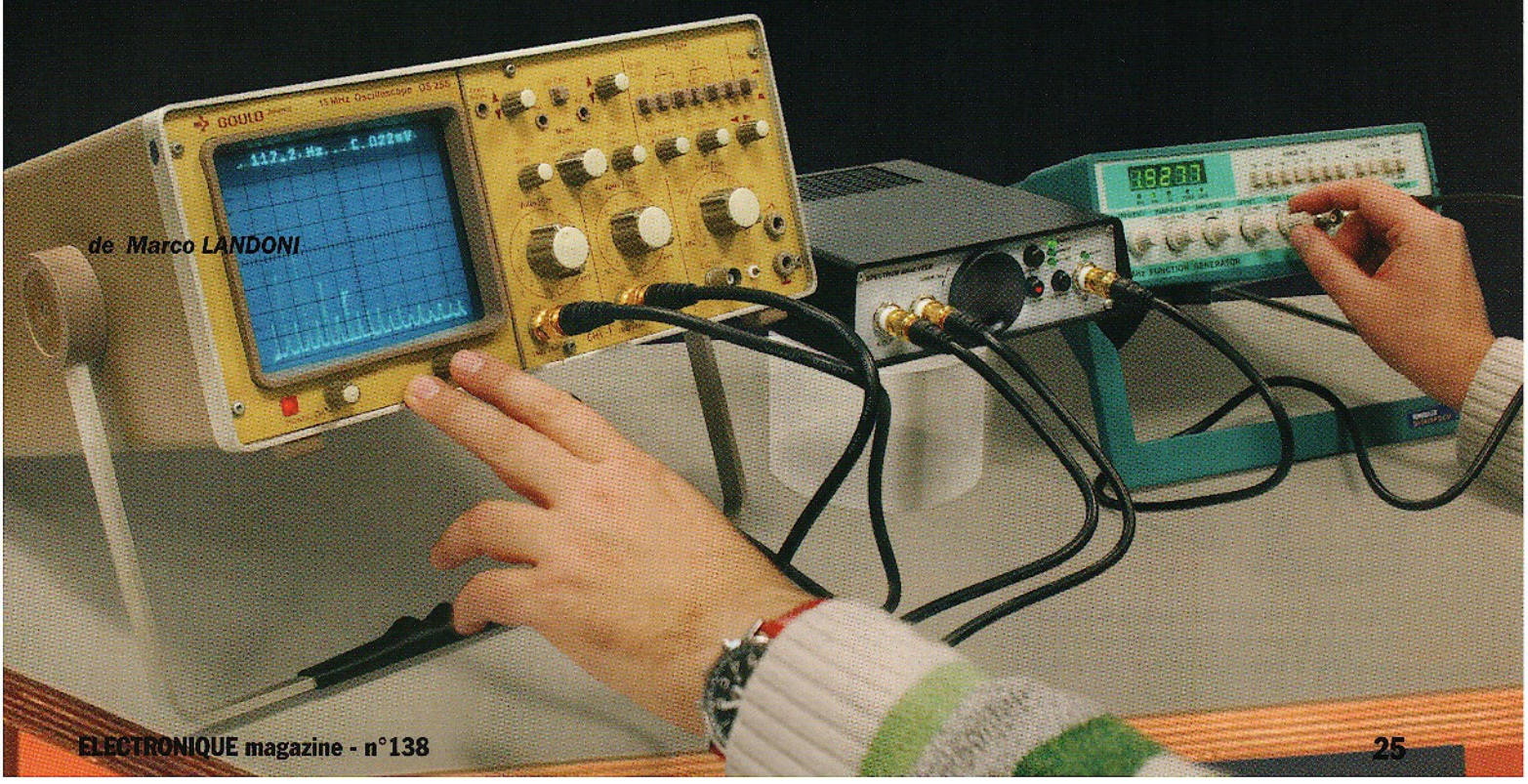
Nous vous proposons dans cet article en deux parties l'étude et la réalisation d'un analyseur de spectre capable d'échantillonner le signal dans toute sa bande passante et de l'afficher sur l'écran d'un oscilloscope doté de la fonction « X/Y », appelée aussi mode « X/Y ». Première partie.

de Marco LANDONI

Si vous disposez d'un oscilloscope analogique dont vous ne savez quoi faire, car peut-être vous vous êtes équipé d'un modèle numérique plus performant, alors ce projet est certainement pour vous.

En fait, nous vous présentons un montage capable de « transformer » un oscilloscope analogique en un analyseur de spectre. Peu importe que votre oscilloscope ait une bande passante limitée ou certaines caractéristiques particulières, l'important est qu'il doit disposer de deux canaux avec le mode « X/Y ».

Notre circuit se connecte aux entrées X et Y (généralement le canal CH1 correspond à l'entrée X et le canal CH2 correspond à l'entrée Y) de l'oscilloscope. En pilotant convenablement ces deux canaux (entrées), il est possible de visualiser sur l'écran de l'oscilloscope le spectre du signal que vous désirez analyser.



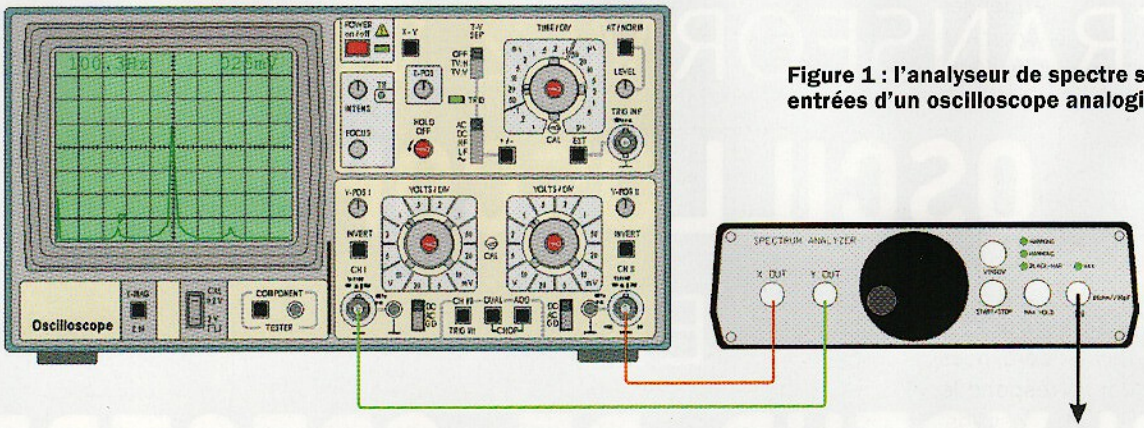


Figure 1 : l'analyseur de spectre se connecte aux entrées d'un oscilloscope analogique classique.

En outre, grâce à divers réglages, le montage indique l'amplitude et la fréquence des différentes composantes spectrales. Le circuit effectue essentiellement une transformée de Fourier d'une séquence d'échantillons acquis numériquement, qui gérée de façon appropriée, affiche sur l'écran de l'oscilloscope le résultat de la transformation.

Tout d'abord rappelons que **Joseph Fourier** (1768 -1830) était un mathématicien Français qui a démontré que **tout signal continu** pouvait se **décomposer** sous la forme de la **somme de sinusoïdes.**

La **transformée de Fourier** est une opération qui permet de **représenter en fréquence des signaux non périodiques**. La fréquence d'échantillonnage maximale est de 10 Ms/s (« mega samples per second » ou méga échantillons par seconde), soit 10 MHz pour une bande passante utile de 5 MHz.

Selon le théorème de « **Nyquist-Shannon** », pour qu'il n'y ait pas de perte flagrante, la **fréquence d'échantillonnage doit être au moins deux fois plus élevée que la fréquence maximale à numériser**, soit dans notre cas pour une bande passante utile de l'ordre de 5 MHz il est nécessaire d'avoir au moins une fréquence d'échantillonnage de 10 MHz. Cela vous permettra d'analyser pratiquement tous les signaux basse fréquence (BF) ainsi que les signaux vidéos.

La fréquence est codée sur 512 lignes et l'amplitude est codée avec une résolution de 8 bits, soit 256 niveaux possibles appelés niveaux de quantification. Plus le nombre de bits est grand, plus la valeur numérique de l'amplitude sera proche de la valeur originale.

Le montage permet une fonction « **zoom** » jusqu'à « **x4** » et une fonction de défilement du spectre visualisé.

Vous pouvez choisir parmi 4 échelles pour visualiser l'amplitude :

- 2 échelles **linéaires** en **V** ;
- 2 échelles **logarithmiques** en **dB**.

Les fonctions de « **Windowing** » (fenêtrage) sont au nombre de trois pour optimiser la résolution en amplitude et/ou en fréquence, pour fixer un affichage sur l'écran à un moment donné, ou pour visualiser le niveau maximum atteint par chaque composante.

Un signal réel ne peut qu'avoir une durée limitée dans le temps, pour observer un signal sur une durée finie il faut le multiplier par une fonction de fenêtrage ou fenêtre d'observation (également appelée fenêtre de pondération).

La plus simple est la fenêtre rectangulaire. L'utilisation d'une fonction de fenêtrage va donc changer la transformée de Fourier du signal.

Par exemple, le fenêtrage d'une forme d'onde simple comme « cosωt » amène sa transformée de Fourier à développer des valeurs non nulles communément appelées fuites spectrales (spectral leakage) à des fréquences différentes de la valeur « ω ». La fuite a tendance à être plus élevée pour des valeurs proches de

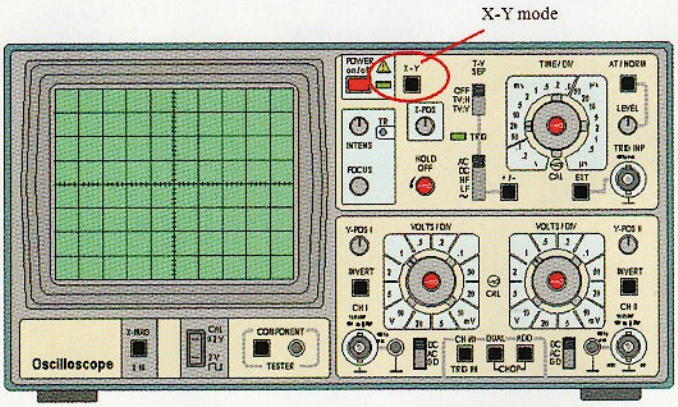
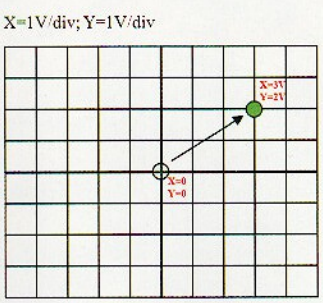


Figure 2 : panneau avant d'un oscilloscope à deux voies (double traces). Généralement, le bouton du mode « X/Y » se situe à côté de celui d'allumage (ON/OFF).

Figure 3 : dans le mode « X/Y », les 2 voies de l'oscilloscope commandent la déviation horizontale et verticale du faisceau lumineux et permettent de déplacer sur l'écran un faisceau proportionnel à l'amplitude du signal appliqué sur chaque voie.



« ω » et moindre pour des fréquences plus éloignées de « ω ». Ce phénomène provoque une perturbation dans l'analyse fréquentielle.

Gestion des graphiques

Avant d'étudier en détail le schéma électrique, nous allons analyser le fonctionnement des différents blocs de l'ensemble du système. Tout d'abord, nous allons expliquer à quoi correspond le **mode « X/Y »** qui doit équiper l'oscilloscope que vous souhaitez convertir en un analyseur de spectre, et comment cette fonction doit être utilisée.

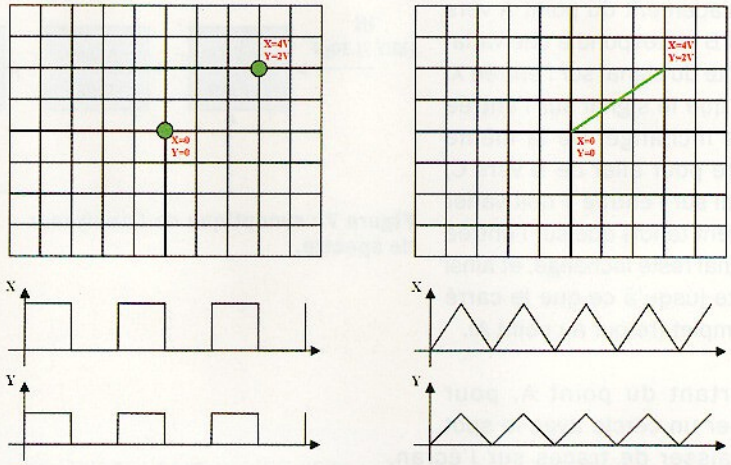


Figure 4 : un signal avec des fronts très raides et des niveaux constants va générer des points, alors qu'un signal avec des fronts moins raides mais non constants génèrera des lignes.

Pour rappel disons un mot sur le fonctionnement interne d'un oscilloscope. Le signal à mesurer est visualisé sur un tube cathodique généralement de couleur verte. La trace de l'oscilloscope est déterminée par deux composantes, une horizontale et une verticale.

La **composante horizontale** se situe sur l'**axe des abscisses** (généralement l'axe des X), et correspond au **temps** (ou à une tension dans le mode « X/Y »). La **composante verticale** se situe sur l'**axe des ordonnées** (généralement l'axe des Y), et correspond à la **tension du signal appliqué sur l'entrée** de l'oscilloscope.

Lorsque le **mode « X/Y »** est en fonction, la base de temps est désactivée et le contrôle du faisceau d'électrons du tube cathodique de l'oscilloscope est complètement lié à la tension présente sur les entrées des deux voies X et Y (CH1 et CH2).

En particulier, l'entrée X commande le déplacement horizontal du faisceau d'électrons (déviation horizontale) et l'entrée Y le déplacement vertical (déviation verticale).

Ainsi, par exemple, en appliquant une tension continue de 3 V sur l'entrée X et de 2 V sur l'entrée Y de l'oscilloscope, le « point lumineux » (spot) sur l'écran généré par le faisceau d'électrons se déplace de 3 carreaux vers la droite et de 2 carreaux vers le haut (voir la figure 3).

La vitesse avec laquelle la trace se déplace sur l'écran est déterminée par la vitesse des signaux appliqués aux entrées. L'application des tensions précédentes sous la forme d'un signal carré ou en dents de scies (rampe) aura des effets différents sur l'écran.

Dans le premier cas (signal carré), nous visualiserons seulement deux points isolés à 2 positions différentes. Dans le second cas, nous visualiserons une ligne qui relie les deux points (voir la figure 4). Comme vous l'aurez deviné, une combinaison appropriée des deux effets peut créer sur l'écran des figures identiques à celles de la figure 5. Ces figures sont appelées **figures de Lissajous**.

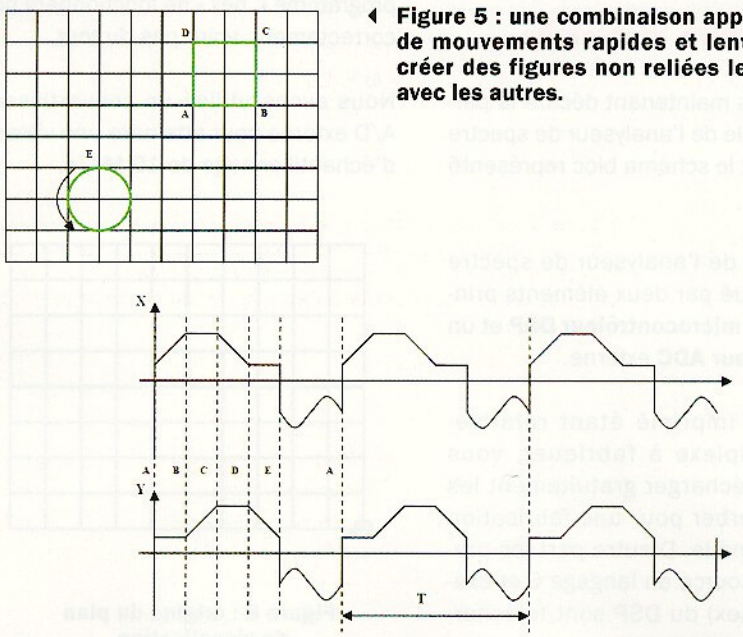


Figure 5 : une combinaison appropriée de mouvements rapides et lents peut créer des figures non reliées les unes avec les autres.

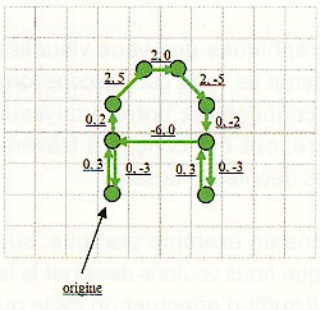


Figure 6 : exemple de réalisation de la lettre « A » à l'aide de 10 segments, les décalages spécifiés sont relatifs.

Le déplacement du point A vers le point B correspond à une variation lente du signal sur l'entrée X, tandis que le signal sur l'entrée Y reste inchangé. De la même manière pour aller de B vers C, le signal sur l'entrée Y doit varier lentement tandis que sur l'entrée X le signal reste inchangé, et ainsi de suite jusqu'à ce que le carré soit complet (retour au point A).

En partant du point A, pour dessiner un cercle avec le spot sans laisser de traces sur l'écran, nous devons déplacer rapidement les signaux sinusoïdaux du point A au point E considéré comme le début du cercle.

À travers un cycle de deux ondes sinusoïdales déphasées de 90° l'une avec l'autre, nous dessinons sur l'écran un cercle qui retourne vers le point E.

Les termes « lentement » et « rapidement » utilisés ci-dessus font référence à la perception visuelle lorsque l'on regarde l'écran.

Si nous voulons obtenir une image visible, il est nécessaire de déplacer continuellement le faisceau d'électrons sur l'écran. En particulier, la période (T) avec laquelle une image complète est répétée, doit être inférieure à la durée de la persistance d'une image sur la rétine de l'œil humain, soit environ 25 ms. Dans le cas contraire, nous apercevrons un scintillement de l'écran.

Les graphiques que vous visualiserez sur l'écran de votre oscilloscope lorsque vous connecterez notre analyseur de spectre, ont été conçus à travers les étapes décrites ci-dessus.

Prenons un exemple pratique, supposons que nous voulons dessiner la lettre « A ». Il suffit d'effectuer un cycle qui se répète, en commençant à partir d'un point pris comme origine. La séquence des mouvements relatifs est représentée en figure 6.

Comme nous devons faire en sorte que les lignes reliant les différents points soient visibles, la rapidité avec laquelle le faisceau d'électrons doit se déplacer sur l'écran ne doit pas être trop élevée.

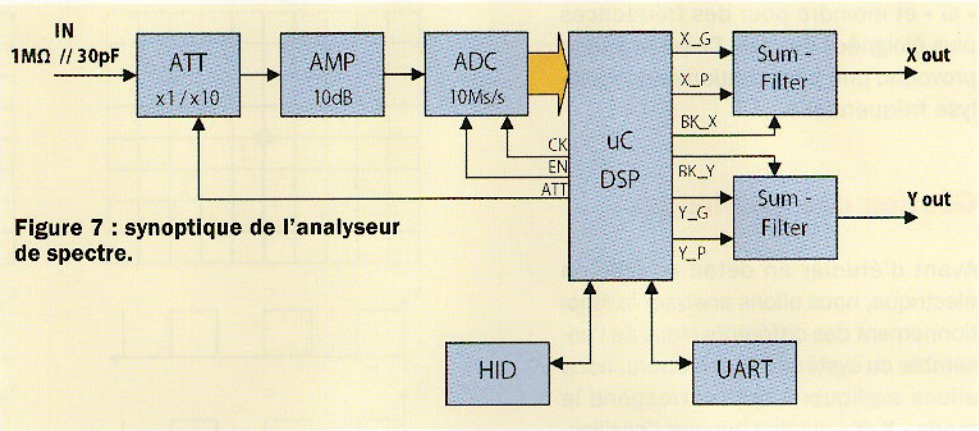


Figure 7 : synoptique de l'analyseur de spectre.

En fait, nous devons nous assurer que le faisceau se déplace lentement en accord avec le nombre de segments à afficher pendant une durée de 25 ms.

Si nous dessinons seulement la lettre « A » avec 10 segments (figure 6), comme nous l'avons évoqué précédemment, nous devons nous assurer que le temps pour dessiner un segment ne dépasse pas : 25 ms/10 segments = 2,5 ms. Si nous devons visualiser 2 lettres composées chacune de 10 segments, nous aurons pour chaque segment un temps maximal de 1,25 ms.

Notez que le **mode « X/Y »** permet par exemple de **visualiser les caractéristiques d'un dipôle**, à condition qu'une des tensions soit l'image du courant qui le traverse. Ce mode permet aussi de **visualiser le déphasage** entre deux tensions sinusoïdales.

Architecture matérielle

Nous allons maintenant décrire la partie matérielle de l'analyseur de spectre en étudiant le schéma bloc représenté en figure 7.

L'essentiel de l'analyseur de spectre est constitué par deux éléments principaux : un **microcontrôleur DSP** et un **convertisseur ADC** externe.

Le circuit imprimé étant relativement complexe à fabriquer, vous pouvez télécharger gratuitement les fichiers Gerber pour une fabrication professionnelle. D'autre part les programmes source en langage C et exécutable (.hex) du DSP sont téléchargeables gratuitement.

Consultez notre site www.electronique-magazine.com dans le sommaire détaillé de la revue 138 à l'onglet « Télécharger ».

En ce qui concerne le microcontrôleur, nous avons choisi un « **dsPIC-33FJ64GP804** » de chez Microchip, car en utilisant le processeur de signal numérique (DSP pour Digital Signal Processor), nous sommes en mesure d'effectuer les opérations mathématiques dans des délais raisonnables nécessaires pour gérer en même temps l'oscilloscope et effectuer l'analyse en fréquence du signal échantillonné.

Nous attirons l'attention sur le fait qu'il est nécessaire de disposer d'un matériel de programmation adéquate, c'est à dire un programmateur en circuit de type MPLAB ICD 3. Les programmateurs en kits ou fait maison ne peuvent pas gérer correctement la phase de programmation en circuit, le programme « .hex » ne fonctionnera pas correctement, voire pas du tout.

Nous avons utilisé un convertisseur A/D externe pour atteindre une vitesse d'échantillonnage de **10 Ms/s**.

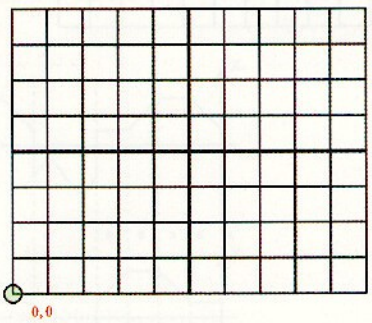
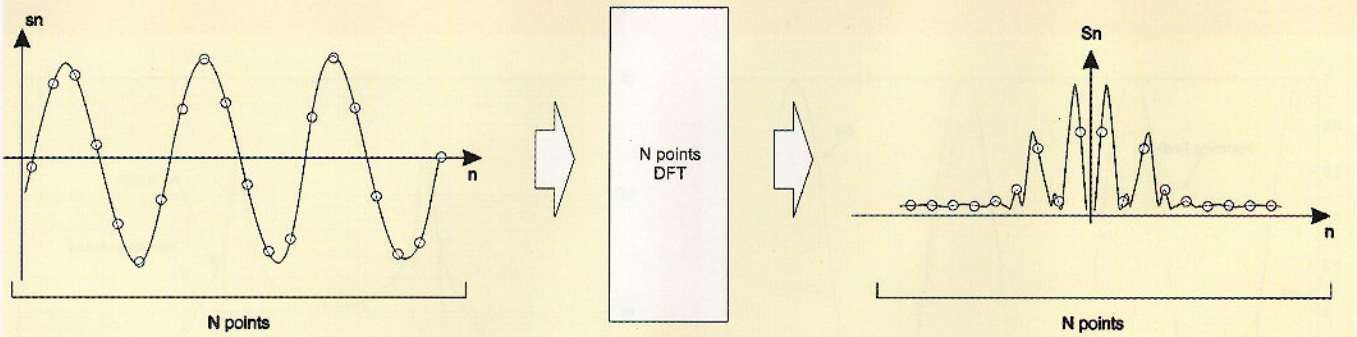


Figure 8 : origine du plan de visualisation.

Figure 9 : structure simplifiée du processus logiciel de la « Transformée de Fourier Discrète ».



Le dsPIC est doté de 4 canaux PWM nommés « X_G », « Y_G », « X_P », « Y_P ». Si nous contrôlons indépendamment la partie graphique (« X_G », « Y_G ») et les origines des différents symboles à représenter (« X_P », « Y_P »), nous pourrons former l'image qui sera affichée sur l'écran de l'oscilloscope.

Ces signaux seront combinés au moyen d'un double étage de filtrage, le résultat génèrera les deux signaux X et Y qui seront ensuite envoyés à l'oscilloscope. Comme nous utilisons des sorties séparées pour le graphique et le positionnement, ces 2 sections nécessitent une vitesse de déplacement différente.

En fait, comme nous l'avons évoqué plus haut, nous devons avoir une vitesse de déplacement du faisceau d'électrons élevée pour « passer » d'un symbole à un autre. Mais pour visualiser la forme qui apparaît sur l'écran de l'oscilloscope, la vitesse doit être plus lente. En effet, la dynamique de mouvement du faisceau lumineux dépend du nombre de segments devant être représentés dans un temps égal à la période T.

Dans notre cas, nous avons choisi de représenter 10 segments pour chaque symbole (caractère) et pour une chaîne de 20 caractères (ce qui donne un total de 200 segments), tandis que les graphiques (formes) seront représentés par 128 lignes.

Cela signifie que, pour représenter un total de 328 segments en moins de 25 ms, un déplacement doit être effectué en environ 80 µs (microsecondes). La section de filtrage de la partie graphique fera en sorte qu'un temps de 80 µs sera attribué à chaque variation de position, temps nécessaire pour rendre la trace (une partie de la forme) visible sur l'écran de l'oscilloscope.

Inversement la section de filtrage, dont le but est de déplacer le curseur rapidement pour les symboles, effectuera le déplacement beaucoup plus rapidement afin de rendre invisible la trace sur l'écran.

Pour notre application, nous avons choisi de prendre comme origine de l'image créée sur l'oscilloscope le point qui se trouve sur l'écran en bas à gauche (voir la figure 8). Les deux signaux « BK_X » et « BK_Y », lorsqu'ils sont activés, permettent de faire revenir rapidement le faisceau d'électrons à cette position.

Le signal d'entrée à analyser, s'il est dirigé vers l'atténuateur « **ATT** » est atténué par un facteur « **10** », sinon il entre dans l'étage « AMP » à gain fixe et large bande.

Le dsPIC contrôle l'atténuateur, le convertisseur « ADC » ainsi que les périphériques d'interface utilisateur (HID) et UART. L'HID est constitué par 3 boutons, un codeur rotatif et 6 LED.

Architecture logicielle (firmware)

Structurellement, la partie matérielle de l'analyseur de spectre n'est pas très compliquée, car elle est composée essentiellement d'un microcontrôleur, d'un convertisseur ADC, et de quelques composants actifs. Il en est autrement pour la partie firmware.

En fait, l'opération d'analyse en fréquence n'est pas une procédure simple, étant donné qu'il faut effectuer des opérations dans le domaine des nombres complexes, plus précisément le calcul du module d'un nombre complexe, ainsi que le déplacement de blocs de mémoire, etc.

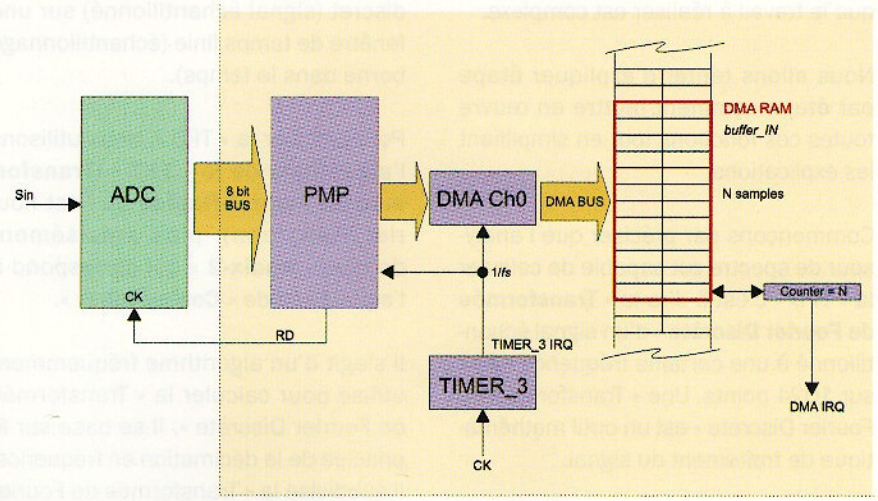


Figure 10 : pour atteindre les 10 Ms/s, nous devons exploiter tout le potentiel de la partie matérielle du dsPIC. Cette figure montre les interconnexions réciproques entre les périphériques impliqués dans l'acquisition de données.

Le processus de « Windowing » (fenêtrage)

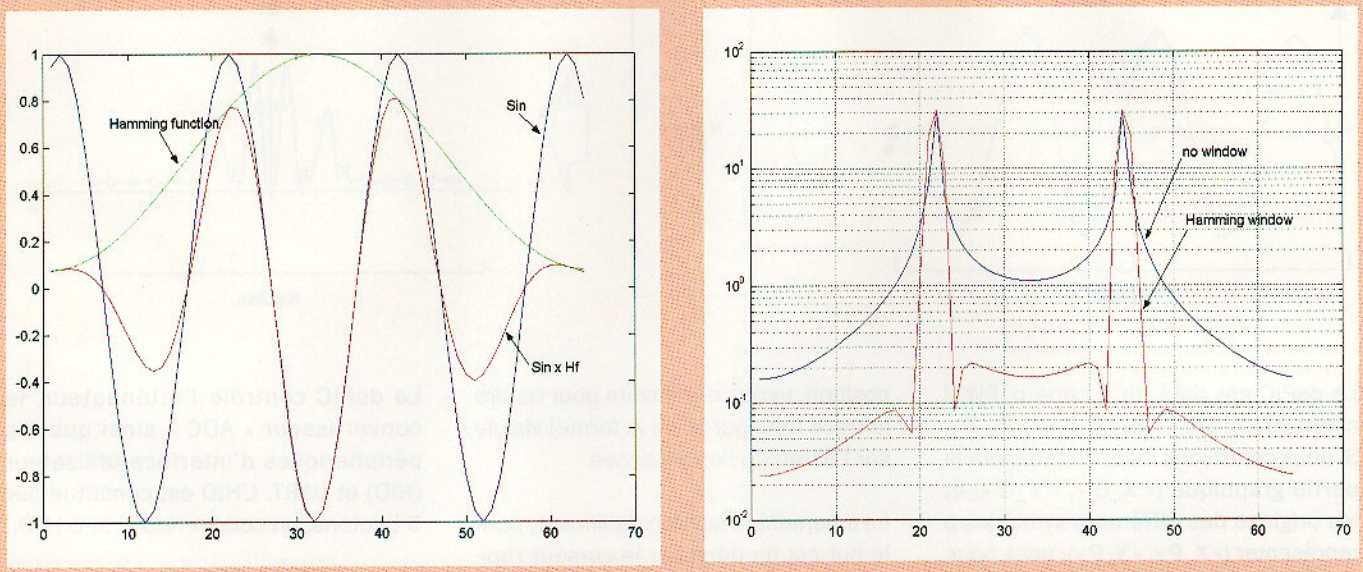


Figure 11 : phénomène de fuite spectrale : à gauche le signal de départ (Sin) et fenêtré (Sin x Hf) ; à droite, le résultat de la Transformée de Fourier du signal fenêtré et non fenêtré. Dans le premier cas, nous constatons un « bruit » (fuite) en raison de l'absence de la fenêtre.

La **fuite spectrale** est un phénomène dans lequel la plage dynamique de la « TFD » calculée sur une certaine séquence d'échantillons est réduite en raison de la troncature de ladite séquence. En figure 11, nous apercevons le signal original « **Sin** » échantillonné dans le temps. Il présente au début et à la fin de la séquence une discontinuité, en ce sens que le premier et le dernier échantillon ne sont pas nuls. En général avec des signaux discontinus nous obtenons des spectres de fréquences très étendus, avec une dynamique spectrale élevée. Le même signal est multiplié par la fonction window « Hf ». Comme vous pouvez le voir, le signal présent dans son enveloppe est beaucoup plus régulier et totalement dépourvu de discontinuités. Ainsi, le spectre de ce nouveau signal « **Sin x Hf** » a une dynamique spectrale moins importante en fréquence.

Si nous tentons d'implémenter ces opérations sur un microcontrôleur de moyenne gamme, il faut nous attendre à des erreurs de fonctionnement ou des situations de dépassement (overflow). C'est là que nous nous rendons compte que le travail à réaliser est complexe.

Nous allons tenter d'expliquer étape par étape comment mettre en œuvre toutes ces fonctions tout en simplifiant les explications.

Commençons par préciser que l'analyseur de spectre est capable de calculer la « TFD » c'est-à-dire la « **Transformée de Fourier Discrète** » d'un signal échantillonné à une certaine fréquence « **Fs** » sur 1024 points. Une « Transformée de Fourier Discrète » est un outil mathématique de traitement du signal.

Ainsi nous obtenons une **représentation spectrale discrète du signal échantillonné.**

Il est important de comprendre que la « TFD » ne calcule pas le spectre continu d'un signal continu, elle permet seulement d'évaluer une représentation spectrale discrète (spectre échantillonné) d'un signal discret (signal échantillonné) sur une fenêtre de temps finie (échantillonnage borné dans le temps).

Pour calculer la « TFD », nous utilisons l'algorithme de la « **FFT** » (**Transformée de Fourier Rapide** ou Fast Fourier Transform), plus précisément de type « **Radix-2** » qui correspond à l'algorithme de « **Cooley-Tukey** ».

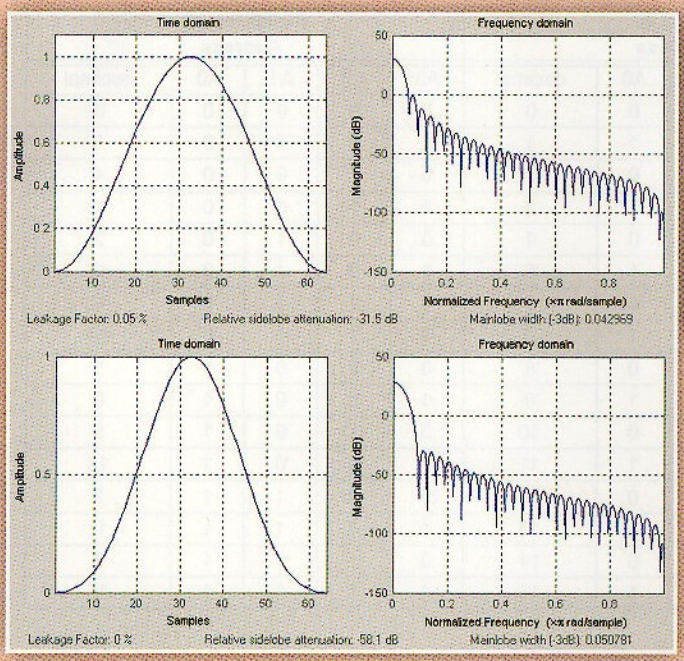
Il s'agit d'un algorithme fréquemment utilisé pour calculer la « Transformée de Fourier Discrète ». Il se base sur le principe de la décimation en fréquence. Il subdivise la « Transformée de Fourier Discrète » d'une taille « n » en plusieurs transformées de Fourier discrètes de tailles inférieures « n1 » et « n2 ».

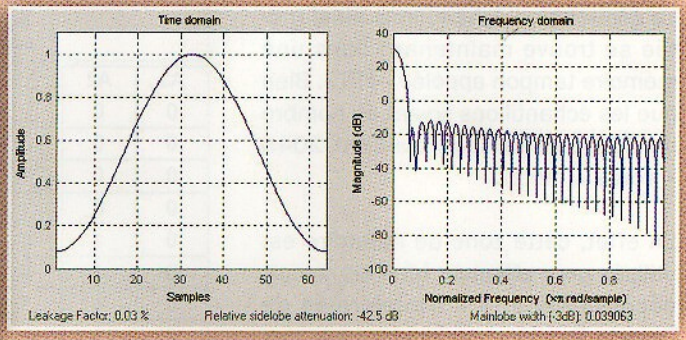
Cela revient à construire une suite d'échantillons que nous aurions obtenus si nous avions échantillonné moins vite.

En d'autres termes, l'analyse de la fréquence de nos signaux, ou du moins la séquence de 1024 échantillons acquise, est calculée en utilisant une « Transformée de Fourier Discrète » optimisée pour la vitesse de calcul.

Etant donné la séquence s = {x_0, x_1, x_2, ... x_n} de « n » échantillons, la « TFD » appliquée à la même séquence produira une nouvelle séquence S = {X_0, X_1, X_2, ... Xn} où X_n sont des nombres complexes qui représentent les « n » échantillons du spectre de « s ».

Le spectre d'un signal est symétrique par rapport à la fréquence et donc la séquence S sera symétrique. Par conséquent, seuls les « n/2 » échantillons sont significatifs, l'autre moitié étant redondante.





En observant les deux « TFD », le signal non fenêtré (bleu) a une dynamique spectrale plus faible par rapport au signal fenêtré (rouge). Cela est dû au fait que l'énergie du signal « Sin » dans le premier cas est appliquée sur toutes les fréquences, tandis que pour le signal « Sin x Hf » l'énergie se concentre autour de la fréquence fondamentale. En revanche, le processus de fenêtrage étend la plage dynamique mais réduit la résolution en fréquence.

L'utilisation de la fonction de fenêtrage permet d'accentuer certaines caractéristiques du signal, de manière à réaliser des mesures plus précises.

Par exemple, une fonction de fenêtrage très étendue permettra une meilleure résolution de la fréquence, à l'inverse une fonction de fenêtrage très serrée permettra une plus grande précision dans la dynamique des harmoniques.

Notre analyseur de spectre comporte 3 fonctions de fenêtrage :

- **Hanning window** : fenêtre de Hanning adaptée aux signaux génériques ;
- **Blakmann-Harris window** : fenêtre de Blakmann-Harris adaptée pour les mesures en amplitude ;
- **Hamming window** : fenêtre de Hamming window adaptée pour une meilleure résolution en fréquence.

* ne pas confondre « fenêtre de Hanning » avec « fenêtre de Hamming », ces noms sont en fait issus des noms de leurs inventeurs (respectivement Julius Von Hann et Richard Hamming).

C'est pour cette raison que même si l'acquisition comporte 1024 échantillons, la résolution efficace ne sera que de 512 lignes, comme mentionné précédemment.

Donc la **première étape pour obtenir la transformée d'un signal est d'acquérir « n » échantillons**, soit 1024 dans notre cas. Cela occupera 3 périphériques intégrés du dsPIC, à savoir un « timer », le périphérique « Parallel Master Port » (PMP) et un canal « DMA ». Plus précisément, c'est le « TIMER_3 » qui est utilisé comme base de temps pour analyser la période d'acquisition « Ts » (Ts est l'inverse de la fréquence d'échantillonnage « Fs »). Le périphérique « PMP » et le canal « DMA #0 » sont configurés pour générer un signal d'horloge « CK » à l'ADC à chaque interruption du « timer », c'est-à-dire à chaque « Ts ».

Par ailleurs, à chaque impulsion « CK », les données de l'ADC présentes sur ses 8 lignes de données sont lues indépendamment par le « PMP » et transférées via le canal « DMA » sélectionné dans une zone de la mémoire appelée « buffer_IN ».

Pour faire toutes ces opérations avec un microcontrôleur commun (comme par exemple un PIC16 ou un PIC18), il faudrait des dizaines d'instructions. Avec un dsPIC, il suffit simplement de gérer l'interruption du « TIMER_3 » et de réinitialiser son flag (drapeau).

En fait dans un dsPIC doté d'un contrôleur « DMA », 2 bus distincts sont présents et peuvent indépendamment accéder à la mémoire RAM. Un bus dédié au « DMA » et l'autre à la « CPU ».

Une fois les n = 1024 échantillons acquis par l'ADC et transférés dans le « buffer_IN », le contrôleur « DMA » déclenche une interruption « DMA_IRQ ». La fonction qui est associée à « DMA_IRQ » désactive le « timer », le périphérique « PMP » et le contrôleur « DMA ». Ensuite, le processus d'analyse de Fourier commence.

La première opération effectuée sur les 1024 échantillons est un fenêtrage grâce à la fonction $W = \{w_1, w_2, w_3, \ldots w_n\}$. Ce processus a pour but de réduire la discontinuité d'amplitude aux extrémités de la séquence acquise. Cela réduit le phénomène de fuite spectrale qui consiste en l'apparition d'harmoniques dans le spectre du signal acquis.

Les signaux exploités numériquement sont toujours une troncation des signaux réels. Le fait de tronquer un signal peut affecter son spectre. La conséquence principale est que les pics s'affaissent et s'élargissent. Suivant la méthode de troncation (fenêtre de pondération) choisie, il est possible de privilégier soit la résolution (largeur des pics), soit la mesure (hauteur des pics).

La séquence d'échantillons ainsi traitée se trouve maintenant dans une mémoire tampon appelée « FFT ». Bien que les échantillons soient au nombre de 1024, ce tampon contient 2048 « **words** » (mots).

En effet, cette zone de mémoire est utilisée pour effectuer tous les calculs nécessaires à la « Transformée de Fourier Rapide » qui portent sur des nombres complexes. Cela signifie que chaque nombre sera composé d'une partie réelle et d'une partie imaginaire, ce qui explique pourquoi le tampon (buffer) de la « FFT » est deux fois plus grand que le nombre d'échantillons acquis.

Une fois l'opération de « FFT » terminée (nous évoquerons un bref aperçu de la FFT dans la prochaine partie de l'article), nous obtenons 2048 « words » représentant les échantillons du spectre de la séquence d'entrée « sW ».

Maintenant, il faut extraire la valeur absolue (ou module) de ces échantillons, afin de représenter leurs amplitudes respectives sur un graphique.

Cela est réalisé par une partie du programme qui calcule, pour chaque échantillon complexe, la quantité $Mn = \sqrt{Re(Xn)^2 + Im(Xn)^2}$ représentant le spectre sur une échelle linéaire, où la quantité $Ln = 10\ log[\sqrt{Re(Xn)^2 + Im(Xn)^2}]$ pour représenter le spectre sur une échelle logarithmique.

Les opérations sur les racines carrées et les logarithmes sont effectuées à l'aide de recherches dans une « **Look Up Table** », c'est-à-dire une table. Les résultats sont stockés une fois de plus dans le buffer de travail FFT. Ensuite, il est nécessaire de réorganiser les échantillons ainsi obtenus.

En fait, l'algorithme « FFT » utilisé est beaucoup plus efficace en terme de vitesse d'exécution que de calcul de la « TFD », cependant l'inconvénient est que les échantillons de sortie sont dans l'ordre inverse (bit reverse) et non dans l'ordre normal.

Si l'index de chaque échantillon d'une séquence normale est représenté en binaire, la séquence dans l'ordre binaire inversé associée est obtenue en inversant l'ordre des bits des index.

Normal Address					Bit-Reversed Address				
A3	A2	A1	A0	decimal	A3	A2	A1	A0	decimal
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	8
0	0	1	0	2	0	1	0	0	4
0	0	1	1	3	1	1	0	0	12
0	1	0	0	4	0	0	1	0	2
0	1	0	1	5	1	0	1	0	10
0	1	1	0	6	0	1	1	0	6
0	1	1	1	7	1	1	1	0	14
1	0	0	0	8	0	0	0	1	1
1	0	0	1	9	1	0	0	1	9
1	0	1	0	10	0	1	0	1	5
1	0	1	1	11	1	1	0	1	13
1	1	0	0	12	0	0	1	1	3
1	1	0	1	13	1	0	1	1	11
1	1	1	0	14	0	1	1	1	7
1	1	1	1	15	1	1	1	1	15

Tableau 1 : association entre l'adressage normal (à gauche) et l'adressage « bit-reverse » (à droite) pour 16 valeurs. Le mode d'adressage « bit-reverse » sert à accélérer les calculs des « Transformées de Fourier rapides ». Ces algorithmes vont prendre des données, stockées dans le tableau, et fournir des résultats, eux aussi écrits dans un tableau. L'ordre de calcul des résultats dans le tableau d'arrivée suit une logique particulière. Les bits de l'adresse du résultat sont partiellement inversés comparé aux bits de l'adresse normale.

Le problème réside dans le fait qu'écrire un programme qui réordonne les échantillons est une opération très complexe, mais heureusement le **dsPIC est capable d'adresser de manière autonome chaque unité de la mémoire** en activant seulement le mode « bit inversé » ou « bit reverse ».

Par conséquent, le tampon FFT se limite à un déplacement avec un pointeur. Le tampon est lu dans l'ordre normal et avec un autre pointeur vers le tampon de destination, l'écriture est faite dans l'ordre « bit inversé ». Il est à noter que dans notre cas, seulement 512 des 1024 échantillons de « Mn » ou « Ln » seront déplacés et classés. Comme nous l'avons mentionné précédemment, le résultat de la « TFD » est redondant.

À ce stade, l'analyse en fréquence de la séquence des échantillons acquis est terminée. Nous pouvons représenter le résultat sur l'écran, et il est alors possible de redémarrer une nouvelle acquisition/conversion des données.

En figure 12, vous pouvez voir le mappage de la mémoire d'un dsPIC, c'est-à-dire la façon dont la mémoire est allouée, ainsi qu'une indication des principaux buffers de données utilisés.

Schéma électrique

L'ensemble de l'analyseur de spectre est géré par un microcontrôleur dsPIC-33FJ64GP804, dont les principales caractéristiques sont :

- Fréquence d'horloge : 40 MIPS
- Convertisseur de données intégré : 13 A/D de 10 bits/12 bits, et 2 D/A de 16 bits ;
- Interface : CAN, I²C, IrDA, LIN, SPI, UART/USART ;
- Nombre I/O : 35 ;
- Périphériques : AC'97, Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT ;
- Mémoire RAM : 16 Ko ;
- Mémoire FLASH programme : 64 Ko ;
- Taille du cœur : 16 bits ;
- Tension d'alim. max. : 3.6 V ;
- Tension d'alim. min. : 3 V ;
- Type d'oscillateur : interne ;
- Accès direct à la mémoire : 8 canaux DMA, permet le transfert de données entre la RAM et un périphérique alors que l'UC exécute le code ;
- Boîtier : TQFP44.

La figure 14 illustre la structure du « core » (cœur) du dsPIC.

Nous remarquons que la mémoire RAM est divisée en deux bancs « X RAM » et « Y RAM ». Cette subdivision de la mémoire RAM n'est pas physique, mais seulement logique.

Cette séparation est nécessaire car certaines instructions nécessitent que les deux opérandes soient sur des bancs distincts, l'un sur le banc X et l'autre sur le banc Y. Cette division n'est pas due à une limitation du dsPIC, mais c'est une fonctionnalité qui permet à la CPU de charger simultanément deux opérandes dans une seule instruction du cycle machine, grâce aux deux bus (X et Y) séparés.

La mémoire est adressée par une logique dédiée appelée « **Address Generator Unit** » (AGU). Grâce à cette unité logique, il est possible d'effectuer un adressage circulaire (modulo) dans lequel le contrôle de la capacité limite du buffer est prise en charge directement par le matériel ou encore

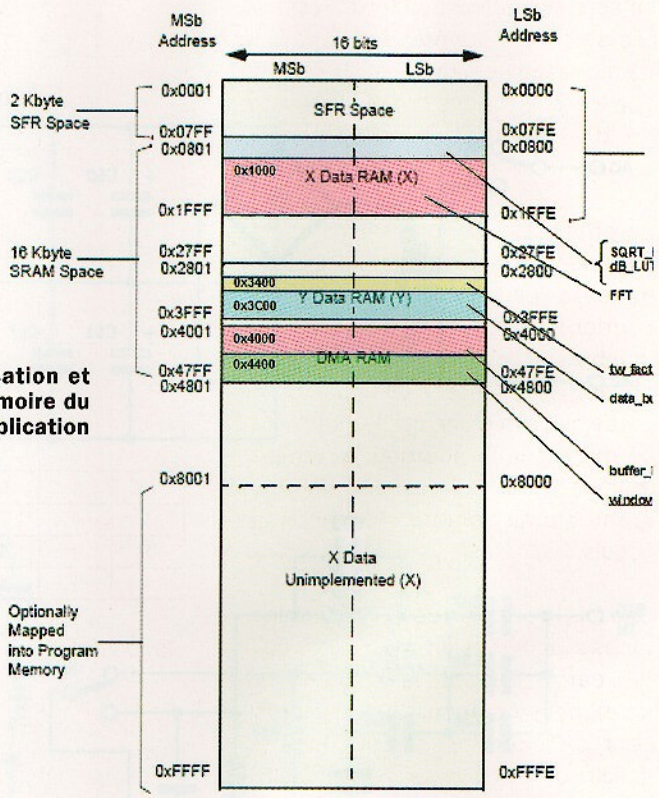
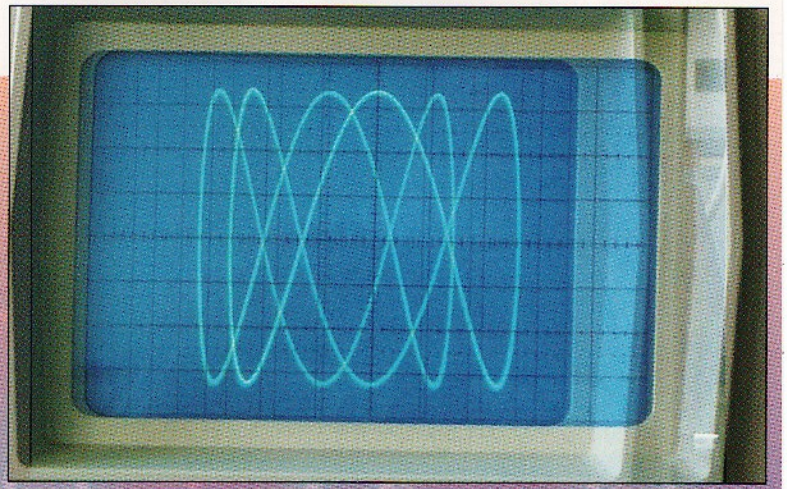


Figure 12 : organisation et allocation de la mémoire du dsPIC pour cette application spécifique.



Les figures de Lissajous

L'analyseur de spectre que nous avons réalisé fonctionne grâce à la capacité de l'oscilloscope à tracer des figures de Lissajous. Ce sont des courbes tracées en combinant deux fonctions périodiques orthogonales.

Elles ont été étudiées par le physicien Français **Jules Antoine Lissajous** (1822 – 1880) et par le mathématicien Américain **Nathaniel Bowditch** (1773 – 1838). Une **figure de Lissajous** est aussi appelée **courbe de Bowditch**, elle représente la trajectoire d'un point dont les composantes rectangulaires ont un mouvement sinusoïdal.

Lorsque les fonctions sont des formes d'ondes, les figures de Lissajous peuvent être visualisées très facilement sur l'écran d'un oscilloscope. Pour cela il suffit d'envoyer les signaux correspondants sur les deux voies et piloter convenablement la déviation horizontale et la déviation verticale.

L'aspect des figures obtenues dépend du rapport entre l'amplitude et la différence de phase. En particulier, lorsque ce rapport est égal à 1, la figure ressemble à une ellipse. Elle devient un cercle dans le cas où les amplitudes sont égales et le déphasage vaut soit 90° ($\varphi = \pi/2$), soit 270° ($\varphi = 3\pi/2$). La figure est un segment de droite dans le cas où les amplitudes sont égales et qu'il n'y a pas de différence de phase ($\varphi = 0$). Elle devient une parabole si une forme d'onde a une fréquence double par rapport à l'autre.

Les figures de Lissajous permettent la mesure des phases et des fréquences relatives entre deux signaux, cependant cette méthode n'est pas très précise.

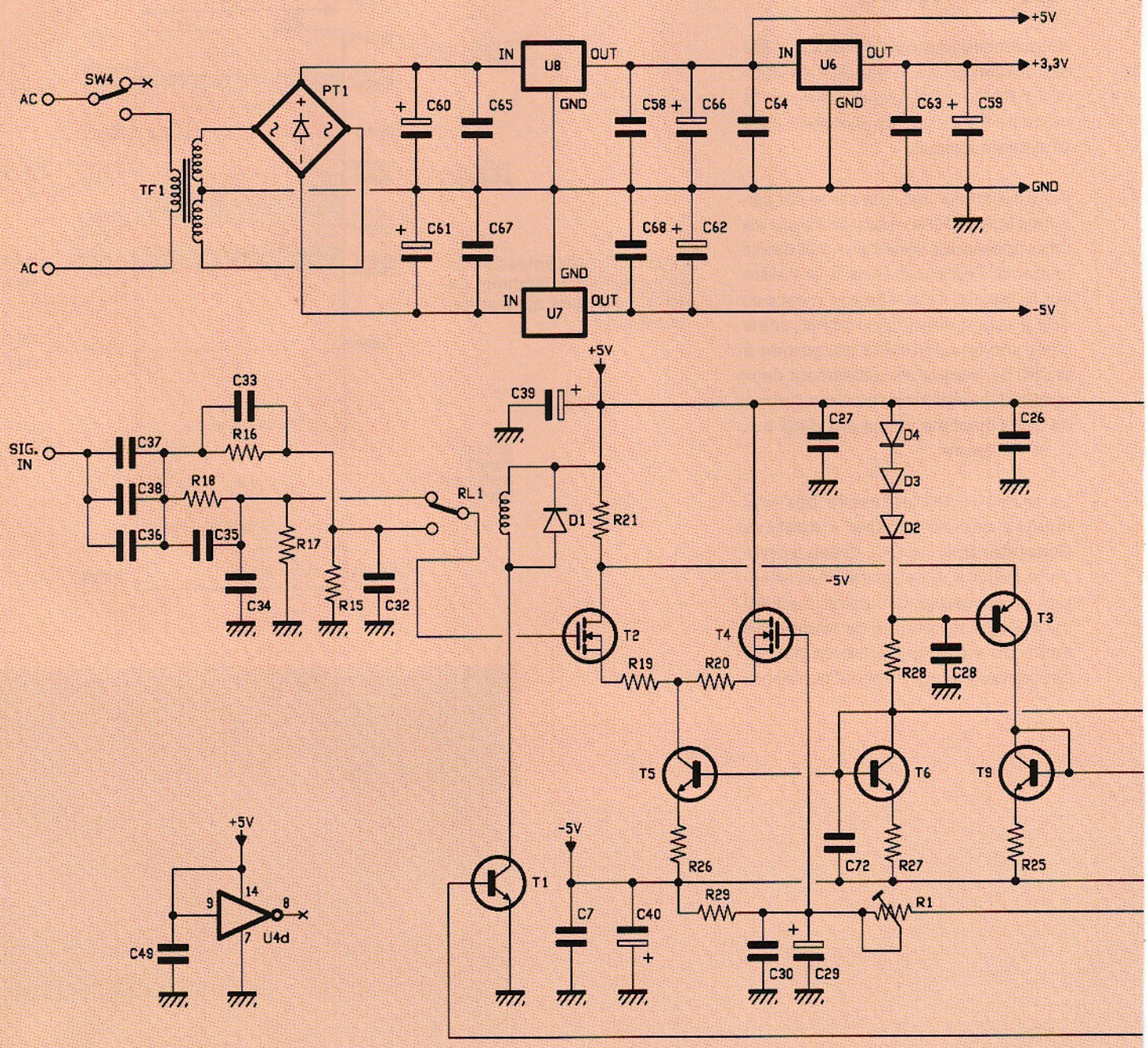


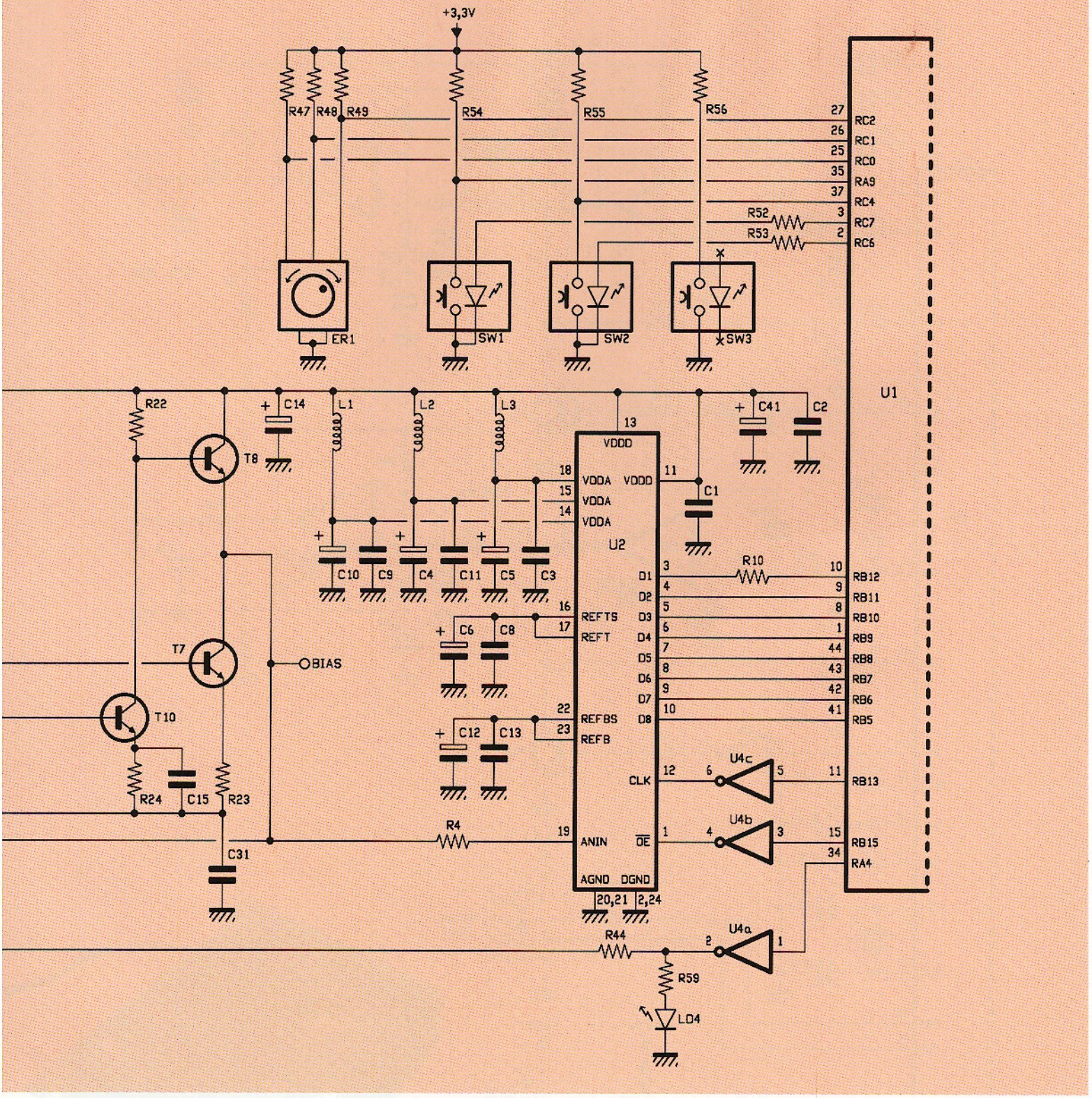
Figure A : schéma électrique de notre analyseur de spectre. À gauche du dsPIC, l'étage d'entrée et l'alimentation. Sur la droite, les 2 étages de sorties « X_OUT » et « Y_OUT ».

d'effectuer un adressage « bit reverse » dans une certaine zone de la mémoire. L'AGU fonctionne de manière autonome par rapport au reste du dsPIC, elle se configure à l'aide de certains registres.

Sur la figure 14, vous pouvez voir également le contrôleur DMA qui gère les 8 canaux DMA qui peuvent être activés simultanément. Le contrôleur DMA est indépendant du reste de la CPU et il peut accéder à une zone spécifique de la mémoire, appelée « DMA RAM ».

Enfin, l'**unité arithmétique et logique** (ALU) est contenue dans les blocs « DSP Engine », « 16 x 16 W Register Array » et « 16-bit ALU » (toujours en figure 14). Une représentation détaillée de l'ALU est visible en figure 15.

Comme nous l'avons mentionné précédemment, l'ALU peut accéder simultanément aux deux bus de données X et Y. Les deux opérandes sont transférés vers le **multiplexeur** (Multiplier) qui est capable de travailler sur 16 bits. Le multiplexeur gère le signe et les divers formats : entiers ou fractions. Le résultat est automatiquement étendu à 40 bits en tenant compte du signe.



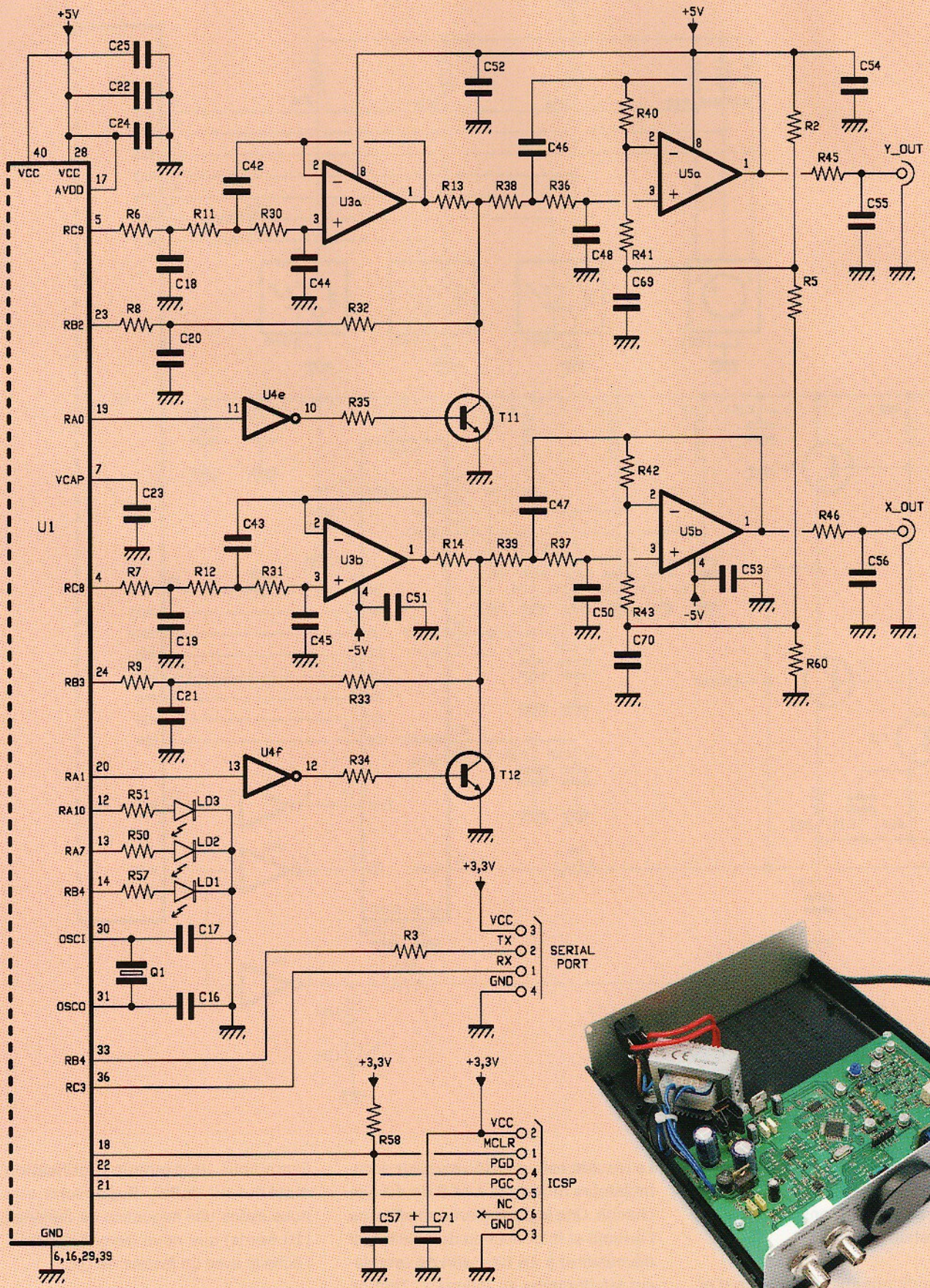
Après l'étage multiplexeur, nous trouvons un « Barrel shifter ». Cet étage effectue des opérations de décalage et de rotation binaires.

Enfin, l'additionneur est capable d'effectuer une addition ou une soustraction (en fonction du signe logique) sur les 40 bits provenant du multiplexer ou à partir du bus de données.

Le résultat est adressé vers deux accumulateurs différents A et B de 40 bits chacun. Une logique d'arrondi effectue l'arrondi et la troncation du résultat (si nécessaire) à 16 bits, afin de remplacer les données en mémoire. Tout ce matériel contenu dans l'ALU du dsPIC est utilisé de manière autonome par le processeur lorsque certaines instructions sont exécutées.

Examinons un exemple d'utilisation matériel du dsPIC. Supposons que nous ayons une séquence de données (10 words) que nous devons mettre à l'échelle avec un facteur de 0,5.

Les données se trouvent dans la RAM, plus exactement dans le tableau (array) « **buffer_IN** ». Le tableau des données, une fois mises à l'échelle, se trouve


+5V
C25
C22
C24
40
28
VCC
VCC
AVDD
17
C52
C54
R40
C46
R2
U5a
Y_OUT
R45
C55
C42
U3a
R13
R38
R36
C48
R41
C69
R5
RC9
5
R6
R11
R30
C18
C44
RB2
23
R8
R32
C20
U4e
R35
T11
RA0
19
11
10
VCAP
7
C23
U1
C43
U3b
R14
R39
R37
C47
R42
U5b
R46
X_OUT
C56
C53
C50
R43
-5V
C70
R60
RC8
4
R7
R12
R31
C19
C45
C51
RB3
24
R9
R33
C21
U4f
R34
T12
RA1
20
13
12
RA10
12
R51
LD3
RA7
13
R50
LD2
RB4
14
R57
LD1
OSCI
30
C17
Q1
OSCO
31
C16
+3,3V
VCC
TX
RX
GND
SERIAL PORT
R3
RB4
33
RC3
36
R58
MCLR
PGD
PGC
NC
GND
ICSP
18
22
21
C57
C71
GND
6,16,29,39

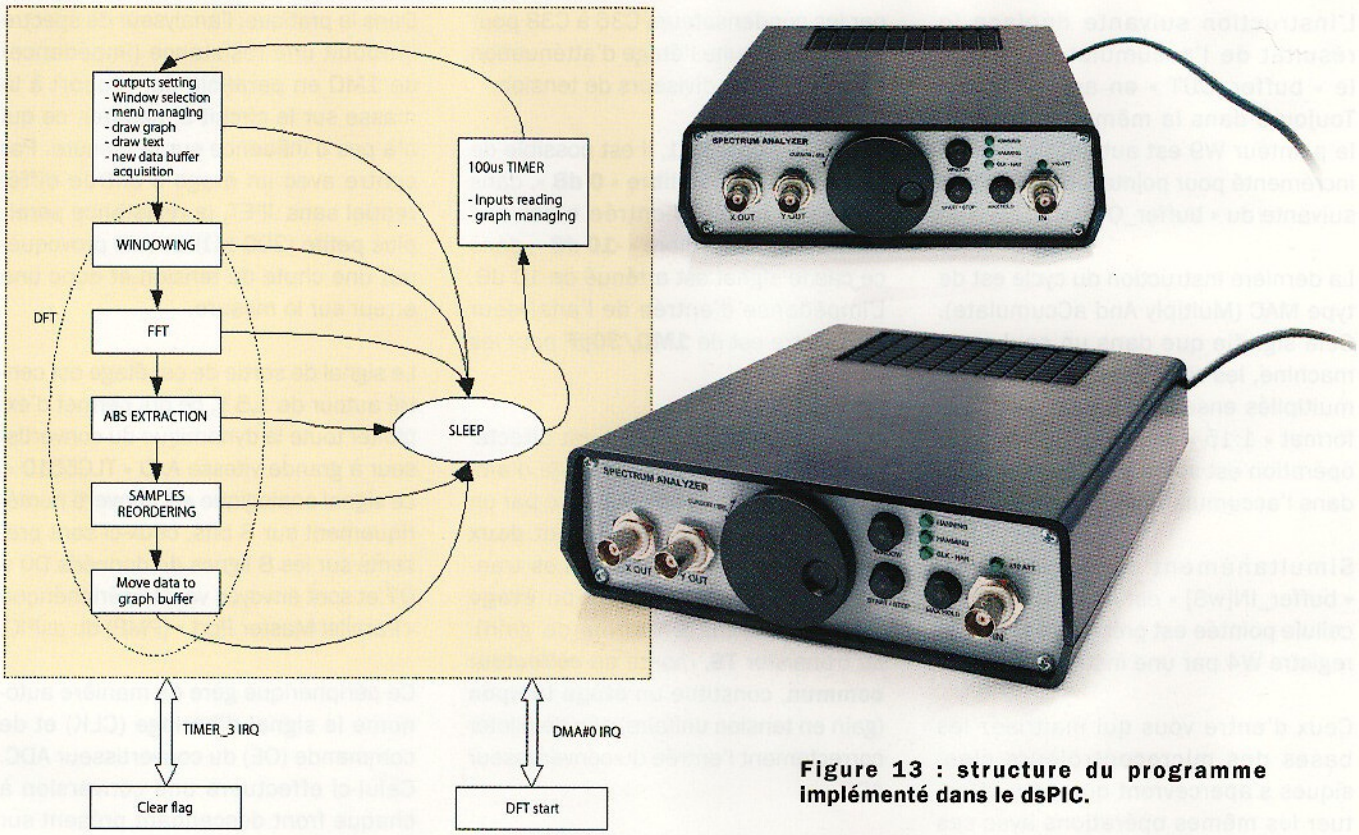


Figure 13 : structure du programme implémenté dans le dsPIC.

aussi dans la RAM et s'appelle « **buffer_ OUT** ». Le résultat de la moyenne est placé dans la variable « **result** ».

Le code pour exécuter cette tâche est composé de quelques lignes décrites dans le Listing 1.

La première instruction charge les valeurs des pointeurs vers les buffers des registres W8, W9 et W10 qui sont précisément utilisés comme pointeurs vers la RAM. Les registres W5 et W6 contiennent les constantes à l'aide desquelles sera multiplié l'array d'entrée « buffer_IN ».

La boucle « **do – loop** » est répétée 10 fois sur chaque élément du tableau (array), elle est initialisée à partir de l'instruction « **do #9, _mean_loop** ». Le dsPIC prend en charge ces cycles et répète « n + 1 » fois le code compris entre l'instruction « do » et le label « _mean_loop ».

Dans la boucle, nous trouvons l'opération de multiplication entre les nombres au format « 1:15 ». Cette opération a lieu dans les registres W4 et W6, le résultat est placé dans l'accumulateur B.

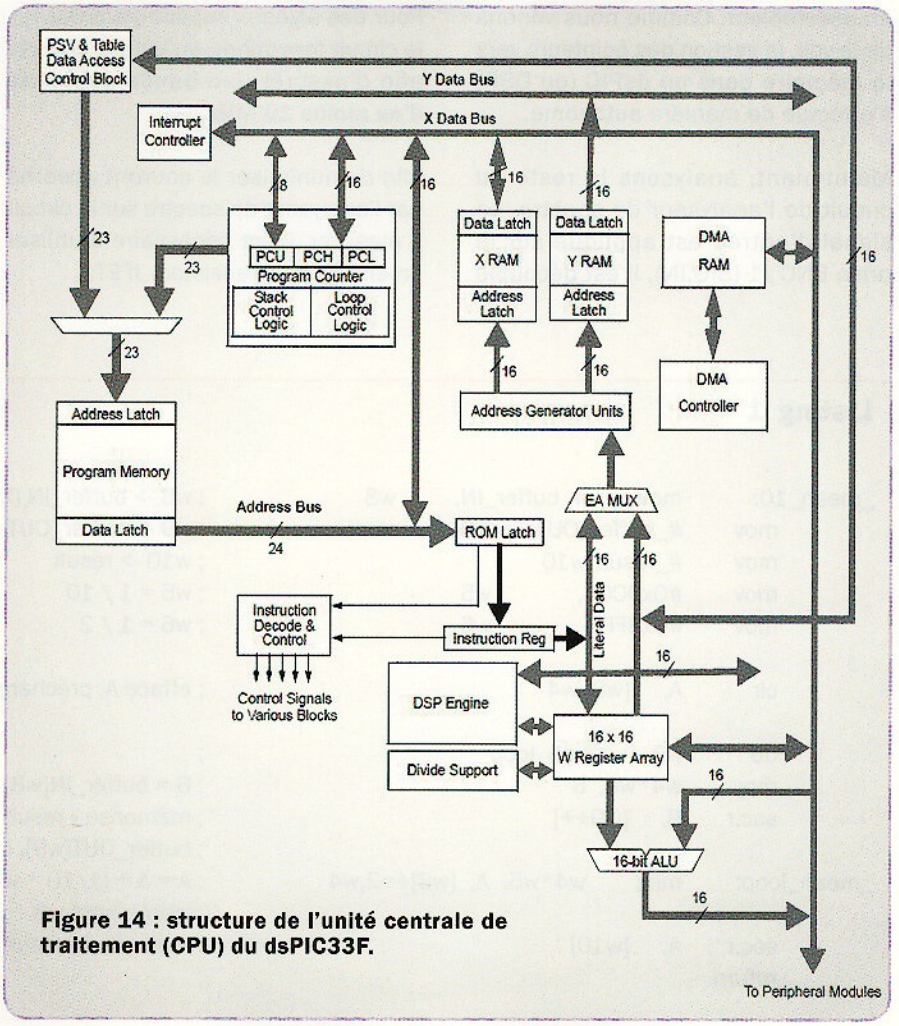


Figure 14 : structure de l'unité centrale de traitement (CPU) du dsPIC33F.

L'instruction suivante déplace le résultat de l'accumulateur B vers le « buffer_OUT » en arrondissant. Toujours dans la même instruction, le pointeur W9 est automatiquement incrémenté pour pointer vers la cellule suivante du « buffer_OUT ».

La dernière instruction du cycle est de type MAC (Multiply And aCcumulate). Cela signifie que dans un seul cycle machine, les registres W4 et W5 sont multipliés ensemble toujours dans le format « 1:15 » et le résultat de cette opération est sommé algébriquement dans l'accumulateur A.

Simultanément le pointeur du « buffer_IN[w8] » est incrémenté et la cellule pointée est préchargée dans le registre W4 par une instruction MAC.

Ceux d'entre vous qui maitrisez les bases des microcontrôleurs classiques s'apercevront que pour effectuer les mêmes opérations avec ces microcontrôleurs, il est nécessaire d'écrire un programme plus imposant en assembleur. Comme nous venons de le voir, la gestion des pointeurs vers la mémoire dans un dsPIC (ou DSP) s'effectue de manière autonome.

Maintenant, analysons le reste du circuit de l'analyseur de spectre. Le signal d'entrée est appliqué sur la prise BNC J1 (SIG.IN), il est découplé par les condensateurs C36 à C38 pour atteindre ensuite l'étage d'atténuation formé par deux diviseurs de tension.

Grâce au **relais RL1**, il est possible de **sélectionner un calibre** « **0 dB** », dans ce cas le signal d'entrée n'est pas atténué ; ou un calibre « **-10 dB** », dans ce cas le signal est atténué de 10 dB. L'impédance d'entrée de l'analyseur de spectre est de **1MΩ/30pF** pour les deux calibres.

Le point commun de RL1 est directement relié à l'entrée de l'étage d'amplification. Celui-ci est composé par un **étage différentiel** comportant **deux transistors JFET T2** et **T4**. Les transistors **T9** et **T10** forment un **étage d'amplification** (en terme de gain). Le transistor **T8**, monté en **collecteur commun**, constitue un **étage tampon** (gain en tension unitaire) afin de piloter correctement l'entrée du convertisseur ADC.

La contre réaction permet de stabiliser le point de fonctionnement en continu. Pour des signaux supérieurs à 0,2 Hz, le circuit fonctionne en boucle ouverte afin d'assurer une **bande passante d'au moins 10 MHz**.

Afin de minimiser le courant absorbé par l'analyseur de spectre sur le circuit à mesurer, il est nécessaire d'utiliser en entrée des transistors JFET.

Dans la pratique, l'analyseur de spectre introduit une résistance (impédance) de 1MΩ en parallèle par rapport à la masse sur le circuit à mesurer, ce qui n'a pas d'influence sur la mesure. Par contre avec un étage d'entrée différentiel sans JFET, la résistance serait plus petite (200 kΩ) et cela provoquerait une chute de tension et donc une erreur sur la mesure.

Le signal de sortie de cet étage est centré autour de 1,5 V, ce qui permet d'exploiter toute la dynamique du convertisseur à grande vitesse A/D « TLC5510 ». Le signal analogique est converti numériquement sur 8 bits, ceux-ci sont présents sur les 8 lignes de données D0 à D7 et sont envoyés vers le périphérique « Parallel Master Port » (PMP) du dsPIC.

Ce périphérique gère de manière autonome le signal d'horloge (CLK) et de commande (OE) du convertisseur ADC. Celui-ci effectuera une conversion à chaque front descendant présent sur la broche d'entrée « CLK ».

Comme le dsPIC fonctionne avec une tension d'alimentation de 3,3 VDC et que le convertisseur A/D fonctionne avec une tension de 5 VDC, il est nécessaire d'insérer entre les deux un translateur de niveaux logiques. Pour cela nous utilisons le circuit intégré U4 qui est un « 7404 HCT », il comporte 6 inverseurs en technologie TTL.

Listing 1

```
_mean_10:   mov    #_buffer_IN,    w8      ; w8 -> buffer_IN (X memory)
            mov    #_buffer_OUT,   w9      ; w9 -> buffer_OUT (Y memory)
            mov    #_result,w10            ; w10 -> result
            mov    #0x0CCC,        w5      ; w5 = 1 / 10
            mov    #0x3FFF,        w6      ; w6 = 1 / 2

            clr    A,   [w8],w4            ; efface A, précharge w4 = [w8]

            do     #9,  _mean_loop         ;
            mpy    w4*w6, B                ; B = buffer_IN[w8] * (1 / 2)
            sac.r  B,   [w9++]             ; mémorise « result » arrondi dans le
                                           ; buffer_OUT[w9], incrémente w9
_mean_loop: mac    w4*w5, A, [w8]+=2,w4    ; A = A + (1/10 * w4), précharge w4 = [w8]
                                           ; incrémente w8
            sac.r  A,   [w10]              ; mémorise « result » arrondi dans [w10]
            return                         ;
```

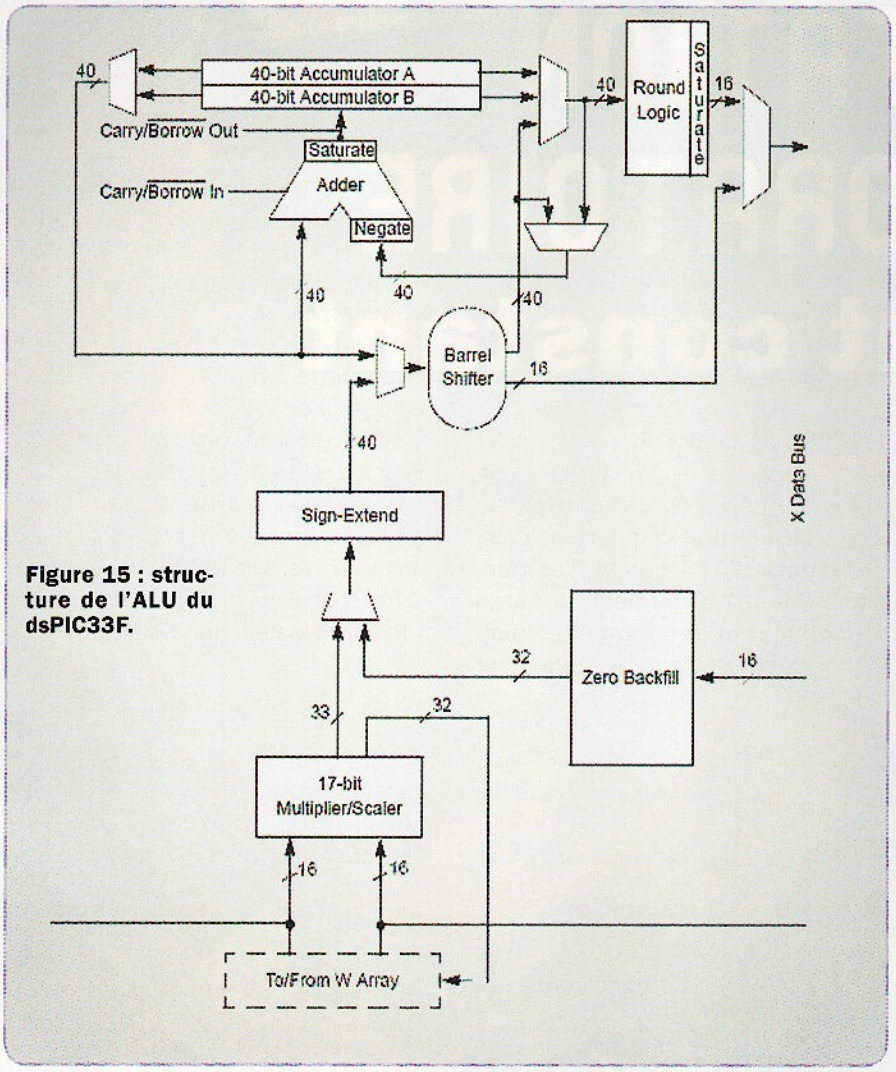


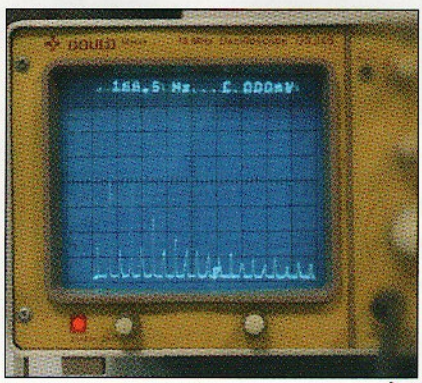
Figure 15 : structure de l'ALU du dsPIC33F.

En alimentant ce circuit avec une tension de 5 V, il accepte sur ces entrées des niveaux logiques TTL et fournit en sortie des niveaux logiques compatibles CMOS. En résumé, l'entrée s'effectue en TTL et la sortie à un niveau CMOS (3,3 V).

Le dsPIC gère l'interface utilisateur composée des 3 boutons P1, P2 et P3, d'un encodeur rotatif E1 et de 5 LED (LD1 à LD5) ainsi que la partie graphique qui pilote l'écran de l'oscilloscope.

En particulier, deux canaux PWM (OC2 et OC3) sont utilisés pour générer un signal qui sera envoyé au canal Y, tandis que les deux canaux OC1 et OC4 gèrent le canal X. Les canaux OC3 et OC4 sont utilisés pour déplacer le faisceau électronique de l'écran de l'oscilloscope.

Les signaux PWM servant pour les canaux X et Y sont filtrés par le circuit



intégré U5, qui constitue dans cette configuration un filtre passe-bas du 2ème ordre avec une fréquence de coupure d'environ 30 kHz.

Cela signifie que le temps de réponse pour la mesure sera de l'ordre de :

ts =1/30 kHz = 30µs.

Les signaux PWM générés par OC1 et OC2 sont utilisés pour visualiser sur l'écran les formes voulues et, en accord avec ce qui a déjà été dit, le temps de réponse du filtre passe-bas est largement suffisant.

En coupant à environ 10 kHz à l'aide du circuit intégré U3, le temps de montée de ces deux canaux sera d'environ :

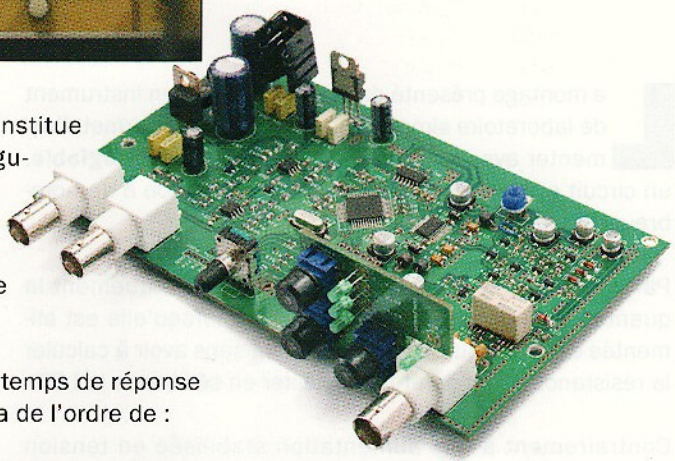
ts' = 1/10 kHz = 100 µs.

Les signaux sont additionnés au niveau des nœuds R13, R32, T11 et R14, R33, T12. Les transistors **T11** et **T12**, lorsqu'ils sont conducteurs (le nœud correspondant se trouve alors à la masse), **permettent de ramener rapidement le faisceau d'électrons à l'origine** de l'écran (voir la figure 8).

L'étage d'alimentation est constitué essentiellement par 3 régulateurs linéaires de tension. Les régulateurs U7 et U8 (respectivement 7905 et 7805) génèrent une tension symétrique de ± 5 VDC nécessaire pour alimenter la section analogique du montage.

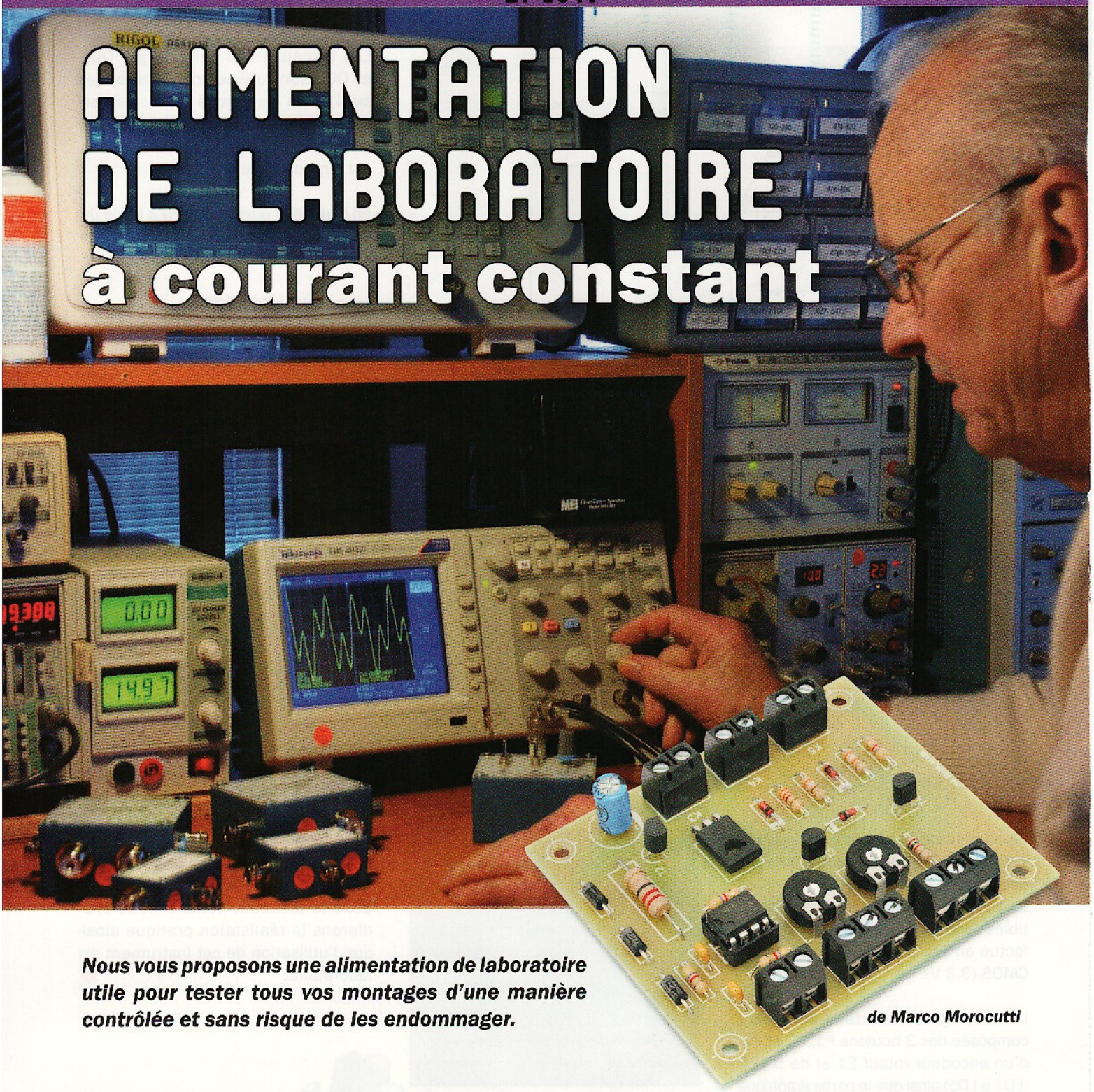
Le microcontrôleur est alimenté par une tension de 3,3 VDC fournie par le régulateur U6 (LD1117AV33) à faible chute de tension (low dropout).

Nous arrivons à la fin de la description théorique de l'analyseur de spectre, dans le prochain numéro nous étudierons la réalisation pratique ainsi que l'utilisation de cet instrument de laboratoire.



ET 1047

ALIMENTATION DE LABORATOIRE à courant constant



Nous vous proposons une alimentation de laboratoire utile pour tester tous vos montages d'une manière contrôlée et sans risque de les endommager.

de Marco Morocutti

Le montage présenté dans cet article est un instrument de laboratoire simple mais utile, qui vous permet d'alimenter avec un **courant constant** et donc **réglable**, un circuit qui doit être testé. Cette alimentation a de nombreuses applications.

Par exemple, elle peut servir à vérifier empiriquement la quantité de lumière émise par une LED lorsqu'elle est alimentée avec un courant prédéterminé, sans avoir à calculer la résistance de chute et la connecter en série avec la LED.

Contrairement à une alimentation stabilisée en tension classique, **celle-ci ne fournit pas une tension stabilisée**, mais un **courant constant.** Il est possible de connecter en série plusieurs LED de différents types ou de couleurs différentes, afin d'évaluer simultanément les différentes émissions de lumière.

Qu'il y ait une ou plusieurs LED, nous serons certains que le courant circulant dans le circuit à tester aura toujours la même valeur. L'**avantage est qu'il ne sera jamais possible d'endommager** la (les) LED lors du test.

Bien évidemment, cela reste valable tant que la chute de tension provoquée par la (les) LED en série ne dépasse pas la valeur maximale de l'alimentation, nous y reviendrons plus tard.

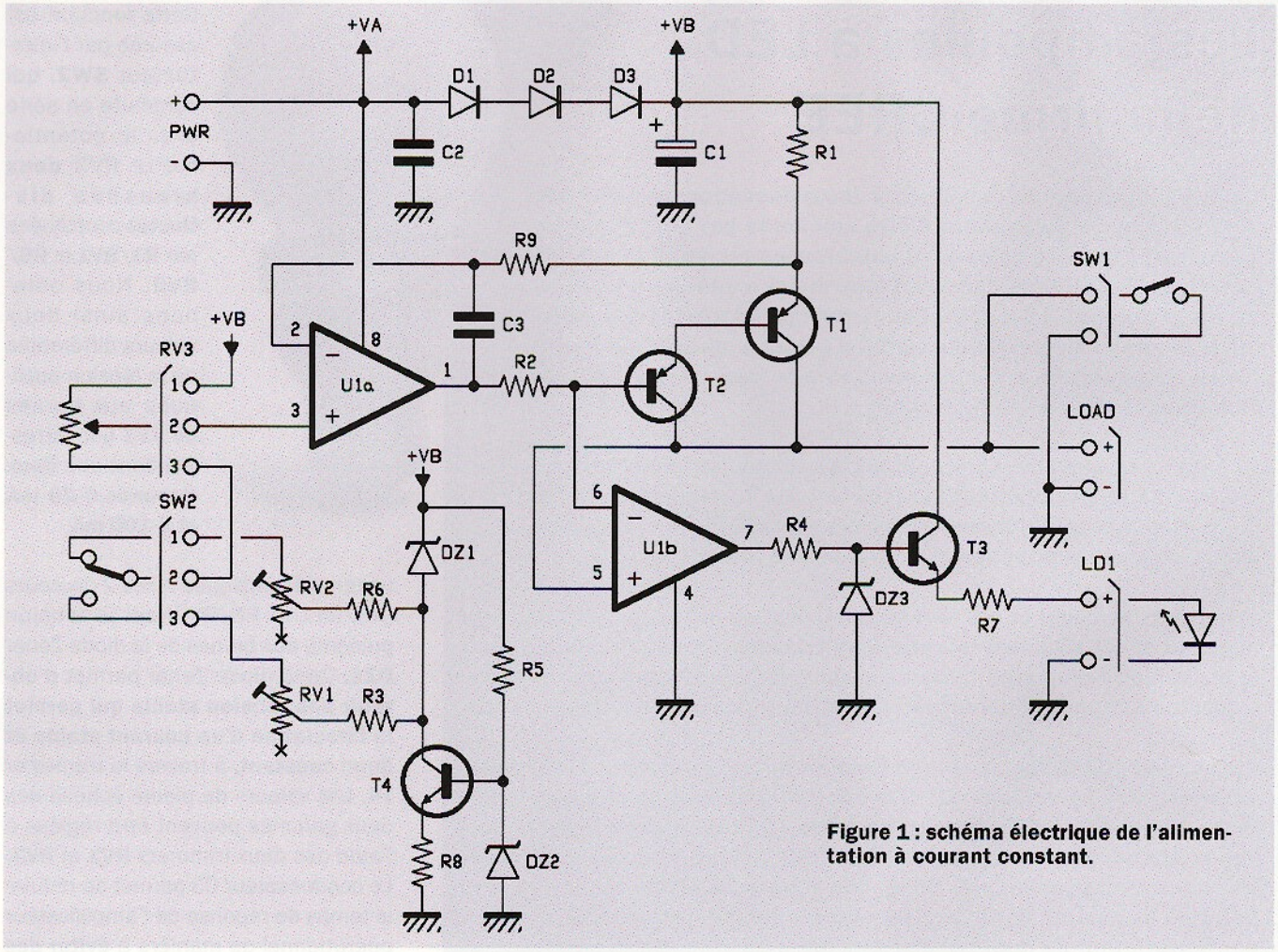


Figure 1 : schéma électrique de l'alimentation à courant constant.

Une autre application intéressante de cette alimentation est la **capacité à vérifier rapidement la tension caractéristique d'une diode Zener, sans risque de l'endommager.**

Par exemple en réglant le courant à une valeur appropriée, qui dans ce cas pourrait être de l'ordre de 10 mA, il suffit de relier la Zener à l'alimentation et mesurer la tension à ses bornes.

Caractéristiques de notre alimentation

Quelles devraient être les caractéristiques d'une alimentation de laboratoire comme celle décrite ici ? Tout d'abord, le **courant de sortie** doit être **réglable** entre « **0** » et une **valeur utile** pour effectuer différents types de tests.

Notre alimentation possède **deux gammes différentes** avec des valeurs « pleine échelle » différentes.

Les gammes sont sélectionnables à l'aide d'un interrupteur, et permettent de fournir un courant sur deux échelles : de **0 à 20 mA** et de **0 à 100 mA**.

Probablement, l'échelle la plus utilisée sera la première, mais dans un laboratoire il peut s'avérer utile de pouvoir générer 100 mA.

Une autre caractéristique de l'alimentation est de pouvoir **signaler une surcharge** lorsqu'elle ne parvient pas à fournir le courant nécessaire. Cela se produit si la chute de tension au niveau de la charge est trop élevée (par exemple lorsque trop de LED sont connectées en série sur la sortie), ou lorsque la tension d'alimentation fournie à notre montage est trop faible pour le type de charge que nous avons connecté en sortie. Si cette condition se produit, le montage l'indiquera en allumant une LED.

Enfin, une alimentation à courant constant doit toujours fournir un courant réglable, quelle que soit la charge connectée en sortie, bien sûr dans les limites de fonctionnement.

Le montage présenté dans cet article dispose de toutes ces caractéristiques, de plus il a été conçu avec des composants électroniques couramment utilisés et donc disponibles dans tout magasin d'électronique.

Le schéma électrique

Le schéma électrique de notre alimentation est visible en figure 1. L'étage de sortie utilise une paire de transistors T1 et T2 en configuration « **Darlington** » montés en « **émetteur commun** », cela permet d'augmenter le gain en courant.

En vertu du gain élevé dû à la configuration « Darlington », le courant traversant la charge « **LOAD** » connectée en sortie est **pratiquement identique au courant traversant** la résistance d'émetteur **R1**.

Les ampoules à LED de la firme CREE

Depuis l'invention de l'ampoule à incandescence en 1879 par l'Anglais Joseph Swan avant d'être améliorée par les travaux de l'Américain Thomas Edison, de nombreuses recherches ont été effectuées pour trouver une alternative moins exigeante en termes de consommation d'énergie et de recyclage. En effet la toute première ampoule électrique consistait à porter à incandescence un filament de tungstène placé dans le vide ou dans un gaz rare (en l'occurrence, le krypton ou l'argon).

Ce n'est qu'en 1983 que l'ampoule basse consommation fut inventée. En réalité, il s'agissait du fameux tube au néon fluorescent. Sa durée de vie était près de dix fois plus importante qu'une lampe à incandescence, pour une consommation d'environ cinq fois inférieure.

Ce n'est qu'au cours de ces dernières années que les ampoules à LED sont apparues et son devenues une alternative à toutes les autres solutions pour la réalisation d'ampoules faible consommation. Cependant il reste encore un obstacle important par rapport à une ampoule classique, c'est leur prix.

Il y a très peu de temps, la société CREE (http://lighting.cree.com) a lancé sur le marché ce qui pourrait être considéré comme la première étape réelle vers la distribution de masse d'ampoules à LED pour la maison. Avec une consommation électrique de seulement 9 W, ce modèle remplace les ampoules traditionnelles de 60 W. Elle génère 800 lumen avec une température de couleur de 2700 K ou 5000 K. De plus elle peut être pilotée par un variateur de lumière. Son prix est d'environ une dizaine d'euros selon la température de couleur.

Si le **courant circulant dans R1 est maintenu constant**, alors le **courant circulant dans la charge sera aussi constant**.

Le circuit U1 est un double amplificateur opérationnel commun de type **LM358**, dont la moitié nommée « U1a » est connectée en configuration « buffer » (tampon) de manière à maintenir la tension aux bornes de R1 constante.

Pour cela, le montage utilise comme référence le côté positif de R1 c'est-à-dire celui relié à la cathode de D3 et nommé « **+VB** » sur le schéma. Cette tension « +VB » est aussi reliée à un des côtés du potentiomètre RV3.

Grâce à la **rétroaction** réalisée par les composants **U1a**, **T2**, **T1**, la **tension présente sur le point milieu** du potentiomètre **RV3** (par rapport au point « + VB »), qui est appliquée à l'entrée **non-inverseuse** (broche 5) de U1a, entraîne la présence d'une **tension identique** aux bornes de **R1**.

En **déplaçant le curseur** du potentiomètre **RV3** d'une extrémité à l'autre, la **tension aux bornes** de **R1 varie** de **0** à la **pleine échelle**, et donc il en est de même pour le courant circulant dans la charge. C'est exactement le but que nous voulions atteindre.

Pour obtenir deux gammes différentes pour le courant maximal, c'est-à-dire la gamme 0-20 mA et 0-100 mA, il est nécessaire que l'extrémité du potentiomètre RV3 puisse être soumise à deux valeurs différentes d'une tension constante.

Cette fonction est assurée par l'interrupteur **SW2**, qui commute en série avec le potentiomètre RV3 **deux branches distinctes** constituées par **R3/RV1** et **R6/RV2**. Nous obtenons ainsi deux valeurs différentes de la tension appliquée aux bornes de RV3 qui correspondent aux **deux gammes 0-20 mA** et **0-100 mA**.

La tension appliquée à ces 2 diviseurs (R3/RV1 et R6/RV2) est la tension présente aux bornes de la diode Zener **DZ1**. Cette diode Zener permet **d'obtenir une tension stable qui permet la circulation d'un courant stable et donc constant**, à travers le transistor **T4**. Les valeurs de pleine échelle des deux gammes peuvent être réglées à l'aide des deux trimmers **RV1** et **RV2**. Le condensateur C3 permet de réduire le temps de réponse de l'amplificateur opérationnel de manière à **éviter des auto-oscillations** dues au gain élevé de la configuration Darlington.

Notez que la tension d'alimentation de l'amplificateur opérationnel **LM358 est maintenue à un niveau supérieur de 2 V** par rapport à la tension « **+VB** », grâce à la présence des diodes D1, D2 et D3. Cela est nécessaire car l'amplificateur opérationnel ne peut pas fournir une tension de sortie qui atteint pratiquement sa tension d'alimentation.

Pour que la tension aux bornes de R1 (et donc le courant circulant dans la charge) soit proche de 0 V, il est nécessaire que la tension de sortie puisse atteindre la valeur de la tension « +VB ».

La seconde partie de U1 (**U1b**) permet de **signaler** que l'alimentation n'est

Tableau 1

Tension d'entrée	Courant de sortie	Tension maximale sur la charge	
		Avec alimentation de 12 V	Avec alimentation de 24 V
de 12 VDC à 24 VDC	2 gammes : 0-20 mA et 0-100 mA	8 V à 20 mA 6 V à 100 mA	19 V à 20 mA 18 V à 100 mA

Plan de montage de l'alimentation à courant constant

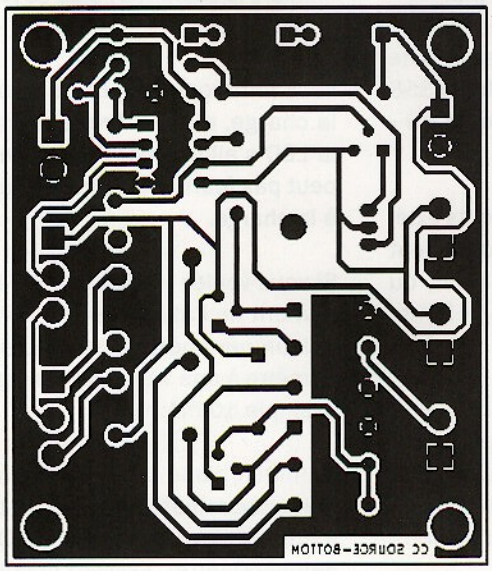


Figure 2 : circuit imprimé à l'échelle 1 : 1 côté soudures. Le circuit ne comporte qu'une seule face.

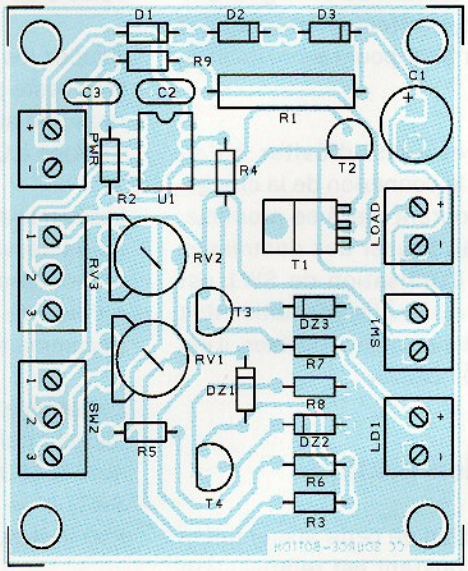


Figure 3 : schéma de câblage des composants de l'alimentation à courant constant.

Liste des composants du ET1047

R1..... 22 Ω 1W
R2..... 4,7 k Ω
R3..... 4,7 k Ω
R4..... 4,7 k Ω
R5..... 1,5 k Ω
R6..... 39 k Ω
R7..... 390 Ω
R8..... 330 Ω
R9..... 4,7 k Ω
RV1.... trimmer 1 k Ω
RV2.... trimmer 10 k Ω
RV3 ....potentiomètre linéaire 4,7 k Ω
C1..... 100 µF/35 V électrolytique
C2..... 100 nF céramique
C3..... 100 pF céramique

U1..... LM358
D1..... 1N4004
D2..... 1N4004
D3..... 1N4004
LD1.... LED 5 mm rouge
DZ1.... Zener 4,7 V 1/2W
DZ2.... Zener 4,7 V 1/2W
DZ3.... Zener 4,7 V 1/2W
T1 ..... BD140
T2 ..... BC327
T3 ..... BC337
T4 ..... BC337

SW1... interrupteur simple contact
SW2... interrupteur à point milieu 2 positions

Divers
Bornier 2 pôles (x 4)
Bornier 3 pôles (x 2)
Support circuit intégré 2 x 4 broches

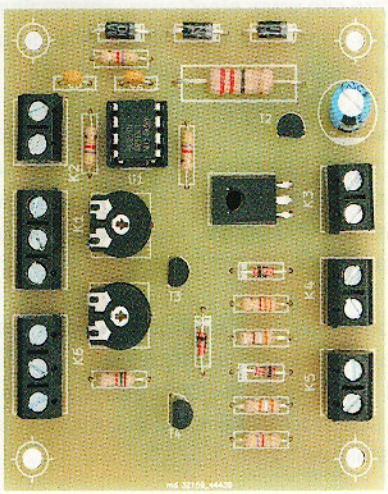


Figure 4 : photo de l'un de nos prototypes de l'alimentation à courant constant.

pas en mesure de fournir le **courant de consigne**. Elle surveille la tension présente sur le collecteur de T1 et la compare avec la tension présente sur la base de T2. Lorsque la tension de collecteur de T1 atteint la valeur de la tension de base de T2, cela signifie que le couple T1/T2 est proche de la saturation et donc que le circuit ne peut pas fournir le courant de consigne.

Dans ce cas, la **LED reliée à l'émetteur** de **T3 s'allume**. En effet, le circuit constitué par le transistor T3 et la diode Zener DZ3 fournit un courant suffisant, qui est indépendant de la tension d'alimentation générale.

L'interrupteur **SW1**, placé **en parallèle avec la charge**, permet **de faire circuler ou non le courant** dans la charge.

Si vous devez modifier la valeur du courant, il suffit de manœuvre SW1.

La charge ne sera plus reliée à l'alimentation, mais celle-ci continuera à fournir le courant de consigne. Une fois le réglage effectué, manœuvrer de nouveau SW1 pour reconnecter la charge.

SW1 permet aussi d'éviter qu'au moment de la connexion de la charge, un pic de courant élevé se produise. Cela pourrait causer des dommages au circuit. L'utilisation de SW1 est facultative, mais si l'alimentation doit tester des composants sensibles, il est conseillé de l'utiliser.

Réalisation pratique

La construction de cette alimentation à courant constant est relativement facile, la fabrication du circuit imprimé ne pose pas de problème particulier, étant donné que c'est un circuit simple face.

Vous pouvez télécharger le typon sur notre site **www.electroniquemagazine.com** dans le sommaire détaillé de la revue **136** à l'onglet « **Télécharger** ».

Une fois le circuit gravé et percé (voir la figure 2), commencez comme d'habitude par souder les composants ayant un bas profil, les résistances et les diodes (attention à l'orientation).

Continuez avec le support de U1, les deux trimmers, les transistors T2, T3 et T4 et les condensateurs. Consultez le schéma de câblage des composants visible en figure 3

Le transistor T1 mérite une attention particulière. La chaleur produite dans des conditions de fonctionnement défavorables (tension d'alimentation élevée, courant important et charge de faible valeur) nécessite l'ajout d'un dissipateur de chaleur.

Vous pouvez fabriquer un morceau d'aluminium en forme de « U » en le perçant en son centre avec un foret Ø 3 mm. Vous pouvez aussi vous procurer un dissipateur pour boîtier TO220 de type « ML26 ».

Positionnez le dissipateur par rapport au transistor en faisant coïncider les trous. Ensuite pliez les pattes à 90 ° délicatement sans les casser. Insérez l'ensemble sur le circuit imprimé à l'aide d'un écrou et d'une vis M3.

Soudez les pattes du transistor. Il ne vous reste plus qu'à insérer le LM358 dans son support, son repère en « U » fait face à C2.

Les connexions avec les éléments extérieurs peuvent être faites directement en soudant les fils sur le circuit, ou à l'aide de borniers au pas de 5 mm (recommandé).

L'ensemble du montage peut alors être installé dans un boîtier en plastique avec la LED, les interrupteurs (SW1 et SW2) et le potentiomètre de réglage RV3 fixés sur la face avant du boîtier.

À côté de l'interrupteur SW2, il sera judicieux de coller une étiquette indiquant les gammes 0-20 mA et 0-100 mA.

Le potentiomètre doit être raccordé de telle manière que le courant augmente lorsque vous le tournez vers la droite. Si cela n'est pas le cas, il suffit d'intervertir les 2 fils reliés aux extrémités.

Calibrage et tests

Alimentez le montage avec une tension stabilisée de 12 V, en respectant la polarité et sans rien connecter en sortie. Vérifiez que le montage consomme un courant d'environ 25 mA.

Pour cela reliez les **deux broches d'un ampèremètre** (multimètre) **en parallèle** sur le bornier « **PWR** », c'est-à-dire directement sur la sortie. Ensuite, sélectionnez la gamme 20 mA pleine échelle sur le multimètre.

De cette façon, **vous provoquez un court-circuit en sortie, mais étant donné qu'il s'agit d'une alimentation à courant constant**, le courant qui circulera dans le multimètre sera **limité** et **constant**.

Réglez le potentiomètre RV3 à la valeur maximale du courant, puis à l'aide du trimmer **RV2 ajustez** le courant pour obtenir exactement **20 mA**.

Ensuite, commutez l'alimentation sur la gamme 0-100 mA et réglez de la même manière le courant à l'aide de **RV1** pour obtenir exactement **100 mA**.

Vous remarquerez qu'en débranchant la charge, c'est-à-dire le multimètre, la LED s'allume car l'alimentation ne peut pas fournir le courant nécessaire à la charge.

Si vous voulez vérifier le bon fonctionnement de l'alimentation, utilisez un milliampèremètre et réglez le potentiomètre à une valeur de consigne, par exemple 10 mA.

Connectez une LED en série avec le milliampèremètre et notez le courant circulant dans le circuit (LED + milliampèremètre). Essayez ensuite avec plusieurs LED en série, sans atteindre la limite de saturation.

Vérifiez que le courant reste constant malgré la chute de tension présente sur la charge. Normalement la chute de tension doit passer de 0 (multimètre seul) à quelques volts (plusieurs LED).

L'alimentation est maintenant prête à être utilisée pour vos tests. Rappelez-vous que vous devez respecter les limites de fonctionnement indiquées dans le **Tableau 1**.

Cela évitera, lors de l'utilisation, l'allumage de la LED pour signaler que l'étage de sortie est saturé ou que le courant de consigne ne peut pas être fourni.

Les limites de fonctionnement

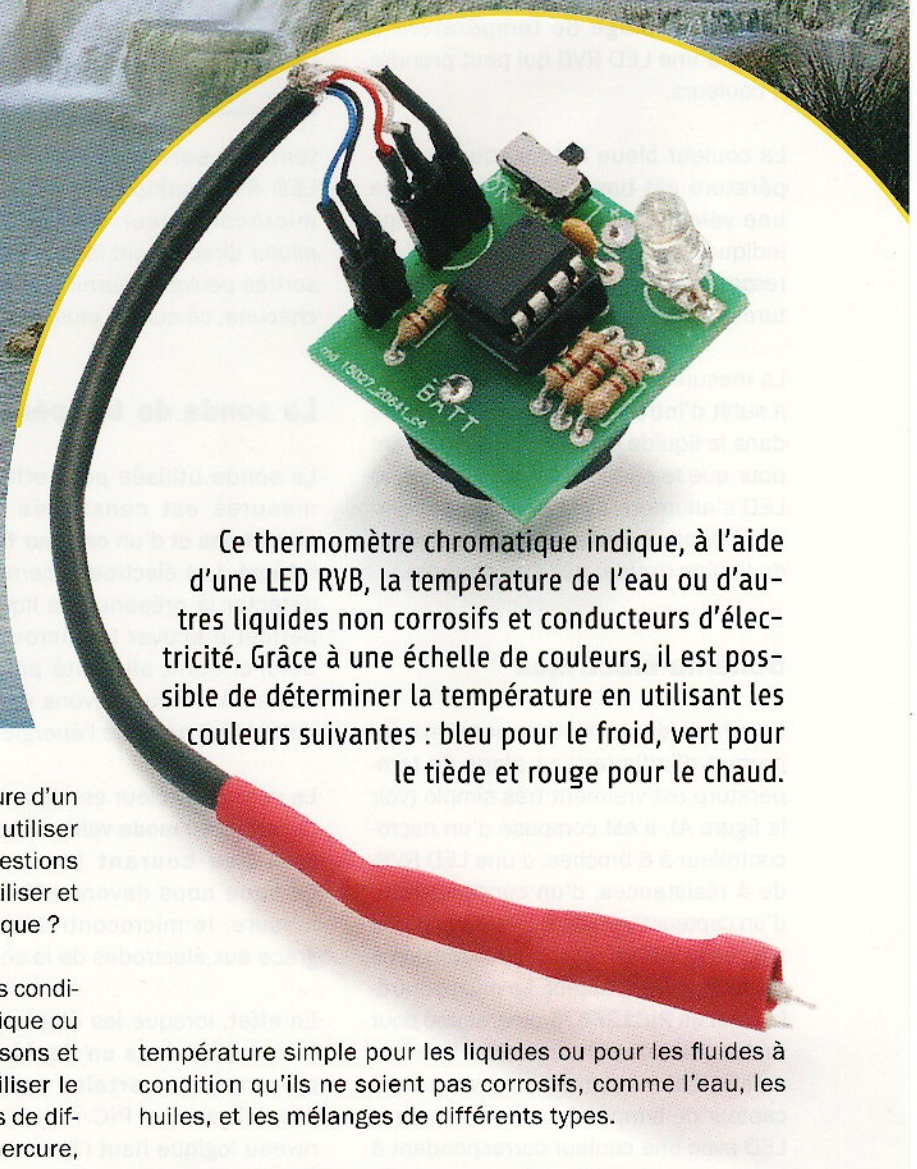
Dans le **Tableau 1**, sont reportées les limites de fonctionnement de l'alimentation; en particulier la tension maximale au niveau de la charge (tension de sortie) dans différentes conditions d'utilisation.

Notez que tous les composants utilisés dans ce projet sont facilement disponibles dans le commerce.

ET832

THERMOMETRE CHROMATIQUE pour liquides

De Boris Landoni



Ce thermomètre chromatique indique, à l'aide d'une LED RVB, la température de l'eau ou d'autres liquides non corrosifs et conducteurs d'électricité. Grâce à une échelle de couleurs, il est possible de déterminer la température en utilisant les couleurs suivantes : bleu pour le froid, vert pour le tiède et rouge pour le chaud.

Tout le monde sait que pour mesurer la température d'un environnement ou d'un liquide, nous devons utiliser un thermomètre. Cependant se posent les questions suivantes : quel type de thermomètre devons-nous utiliser et avec quel affichage pour une application bien spécifique ?

En fait cela dépend de nombreux facteurs, comme des conditions de fonctionnement, du type de lecture (analogique ou digitale, précise ou pas), du budget dont nous disposons et surtout des préférences de la personne qui doit utiliser le thermomètre. Par exemple il existe des thermomètres de différents types, ceux gradués avec une colonne de mercure, ceux avec des afficheurs LCD ou à LED, ou encore ceux avec un indicateur de couleur.

En nous inspirant de ce dernier type, qui est utilisé dans les instruments du tableau de bord de certains véhicules pour indiquer la température du liquide de refroidissement du moteur, nous avons pensé à réaliser un indicateur de température simple pour les liquides ou pour les fluides à condition qu'ils ne soient pas corrosifs, comme l'eau, les huiles, et les mélanges de différents types.

La seule condition requise pour un fonctionnement correct du thermomètre chromatique est que le liquide soit électriquement conducteur. Cela deviendra évident lorsque nous étudierons le fonctionnement du circuit.

Par contre ce thermomètre ne pourra pas mesurer, par exemple, la température des huiles diélectriques (celles

utilisées pour la lubrification de pièces sous tension ou pour le refroidissement de transformateurs de forte puissance), car par définition un diélectrique est un isolant !

Notre thermomètre n'est pas un instrument de mesure précis, mais plutôt un **indicateur capable de détecter 3 conditions**, chacune correspondant à une **plage de température** du liquide. Ces plages ne distinguent pas les degrés, mais forment 3 groupes bien définis qui correspondent chacun à une couleur.

Pour plus de précision, ce montage indique la plage de température à l'aide d'une LED RVB qui peut prendre 3 couleurs.

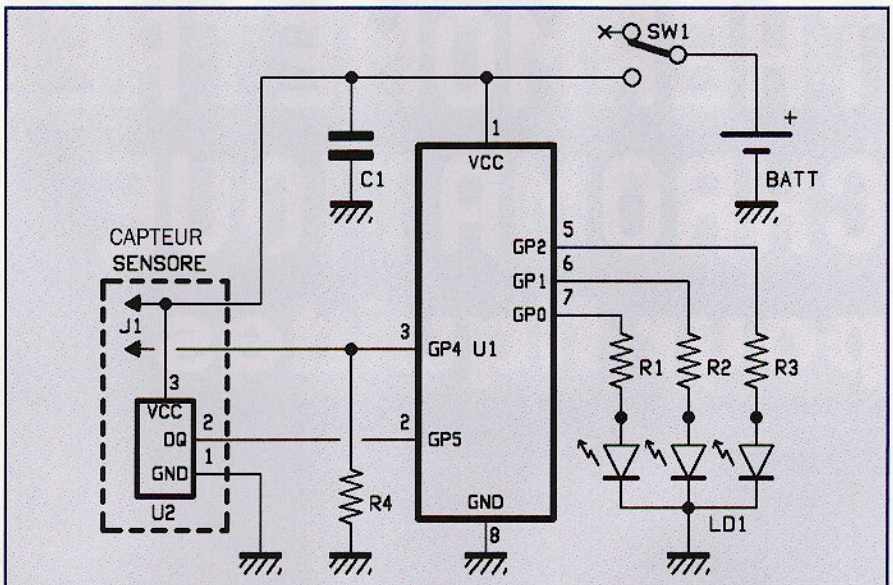


Figure A : schéma électrique du thermomètre chromatique.

La couleur bleue indique que la température est basse, la verte indique une valeur intermédiaire et la rouge indique que le liquide est chaud. Cela ressemble aux indicateurs de température utilisés dans le milieu automobile.

La mesure est très simple à effectuer, il suffit d'introduire la sonde en partie dans le liquide et attendre un moment pour que le circuit s'active. Ensuite, la LED s'allumera avec une couleur correspondant à la plage de température du liquide (froide, tiède ou chaude).

Schéma électrique

Le circuit de notre thermomètre qui permet d'indiquer une plage de température est vraiment très simple (voir la figure A). Il est composé d'un microcontrôleur à 8 broches, d'une LED RVB, de 4 résistances, d'un condensateur, d'un capteur thermique intégré et d'une pile. Le montage est mis sous tension à l'aide d'un interrupteur. Le microcontrôleur est un **PIC12F675** programmé pour rester normalement en veille. Une fois activé, il lit les données provenant du capteur de température, puis allume la LED avec une couleur correspondant à la température détectée.

À la mise sous tension, le PIC initialise ses lignes d'entrées/sorties en définissant la broche GP4 en entrée, la broche GP5 comme bidirectionnelle et les broches GP0, GP1, GP2 en tant que sorties afin d'alimenter la LED RVB (composée de 3 LED). Le microcontrôleur PIC12F675 peut piloter directement la LED RVB car ses sorties peuvent fournir jusqu'à 25 mA chacune, ce qui est plus que suffisant.

La sonde de température

La sonde utilisée pour effectuer les mesures est constituée de **deux électrodes** et d'un **capteur thermique intégré**. Les électrodes permettent de détecter la présence de liquide, cela permet d'activer le microcontrôleur. Celui-ci étant alimenté par une pile « bouton », nous devons économiser autant que possible l'énergie.

Le microcontrôleur est donc la plupart du temps en **mode veille** et **consomme très peu courant** (quelques µA). Lorsque nous devons effectuer une mesure, le microcontrôleur s'active grâce aux électrodes de la sonde.

En effet, lorsque les **électrodes sont introduites dans un liquide**, celui-ci présente une **certaine conductivité électrique**. Le PIC reçoit alors un niveau logique haut (1) sur sa broche GP4, ce qui a pour effet de le faire sortir de l'état de veille.

Il exécute alors le programme en lisant l'état de la sonde et en indiquant la température en allumant la LED correspondante.

Le programme effectue lors d'une première étape la lecture du capteur thermique, puis traite les données reçues afin de contrôler correctement les sorties (GP0, GP1 et GP2) pour allumer la LED correspondant à la température mesurée.

La lecture du capteur se produit de manière cyclique lorsque la broche GP4 reçoit un niveau logique haut (1), c'est-à-dire lorsque les électrodes sont en contact avec le liquide. Lorsque le courant circulant entre les électrodes J1 s'arrête, c'est-à-dire lorsque les électrodes ne sont plus en contact avec le liquide, la broche GP4 se trouve à un niveau logique 0.

Le PIC interprète cette condition comme un retour en mode veille. Dans ce cas, la LED RVB s'éteint et le microcontrôleur consomme un minimum de courant jusqu'au prochain « réveil ».

Le capteur de température intégré que nous utilisons dans ce thermomètre est un **DS18B20** de la firme **MAXIM**. Il est utilisé aussi dans l'article consacré aux « breakout board » publié dans la revue précédente (n°135). Ce capteur est connecté au PIC à travers la broche GP5, qui est initialisée comme une entrée/sortie bidirectionnelle.

Sur le schéma électrique de la figure A, vous pouvez remarquer que les connexions entre le capteur et le microcontrôleur sont extrêmement simples.

Il y a seulement 3 broches : deux broches correspondent à l'alimentation du capteur (VCC et GND), et la 3ème broche (DQ) correspond à la ligne de données qui est bidirectionnelle.

En fait, le capteur communique selon le **protocole** « **One-Wire** » qui prévoit la communication des données par l'intermédiaire d'un seul fil (référencé par rapport à la masse). Pour que le fonctionnement du capteur DS18B20 soit correct, la ligne de données doit disposer d'une résistance de « pull-up » (tirage). Dans notre cas, nous utilisons la **résistance interne** de « **pull-up** » du microcontrôleur.

Comme la communication avec le capteur est de type numérique, nous pouvons placer la sonde assez loin du circuit sans se soucier des bruits et des interférences provenant de l'environnement.

Le **DS18B20** mesure une température comprise entre **–55 °C et +125 °C** avec une résolution de **11 bits**. De plus sa précision est seulement de 0.5 °C dans une plage de température allant de -10 °C à +85 °C.

Pour interroger le capteur, nous utilisons la commande « **Convert T** » (code 44h) correspondant au début de la conversion de la mesure. Le capteur répond avec un « 0 » tout en effectuant la conversion. Lorsque celle-ci est terminée, le capteur envoie un « 1 » logique. À ce stade, dans les registres du capteur sont disponibles deux octets correspondant à la valeur de la température mesurée en degrés Celsius. Nous devons simplement lire la valeur à l'aide de la commande « **Read ScratchPad** » (code BEh).

Le registre à 2 octets, pour un échantillonnage sur 11 bits, est structuré selon la figure 1. Dans notre cas, des 16 bits renvoyés par le DS18B20 chaque fois que le microcontrôleur l'interroge, les 11 premiers bits (l'octet le moins significatif plus les 3 premiers bits de l'octet le plus significatif) expriment la valeur de la température.

Les **5 derniers bits indiquent le signe**, s'ils sont tous les 5 à « 0 » alors la température mesurée est positive. S'ils sont à « 1 » alors la température mesurée est négative (en dessous de 0 °C).

De Galilée à l'ère numérique

Le premier thermomètre a été développé par Galilée en 1581. Il était constitué d'un cylindre de verre contenant un liquide transparent (eau). Lorsque la température diminuait, l'air contenu dans le tube se contractait et donc le liquide montait dans le tube. A l'inverse lorsque la température augmentait, l'air se dilatait et le liquide était poussé vers le bas. Donc en fonction de la température, le niveau du liquide variait soit vers le haut ou vers le bas.

En 1699, le physicien français Guillaume Amontons inventa un thermomètre qui mesurait la température en faisant varier la pression d'un gaz. Il a démontré que l'eau bouillait toujours à la même température dans les conditions d'une pression atmosphérique standard.

Notez que lorsque la pression diminue, la température d'ébullition de l'eau diminue aussi. En haut du mont Blanc, la pression est inférieure à 0,5 atmosphère, l'eau bout à 85 °C. Plus haut, au sommet de l'Everest, la pression est encore plus faible et l'eau bout à 72 °C.

La première personne qui a inventé un thermomètre fermé, et donc insensible à la pression atmosphérique, était Ferdinand II de Médicis, en 1654. Initialement les thermomètres scellés utilisaient de l'eau ou de l'alcool, mais ces liquides produisaient des vapeurs qui modifiaient leurs comportements. En outre, l'eau ne se dilatait pas et ne se contractait pas de manière uniforme en fonction des variations de la température, comme cela était nécessaire pour obtenir une bonne précision. L'alcool avait un défaut, elle bouillait à une température trop basse, environ 78 °C.

Enfin le thermomètre à mercure fit son apparition grâce au physicien Allemand Gabriel Fahrenheit. En 2009 le mercure a été interdit d'utilisation dans les thermomètres médicaux, ceux qui sont disponibles à la vente sont électroniques.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
MS Byte	S	S	S	S	S	2^6	2^5	2^4

Figure 1 : structure des 2 bits émis en séquence par le DS18B20 chaque fois qu'il est interrogé par le PIC. « LSB » représente l'octet le moins significatif (de poids faible), « MSB » représente l'octet le plus significatif (de poids fort). Cette structure se réfère à la représentation à 11 bits plus le signe (5 bits).

Plan de montage du thermomètre chromatique

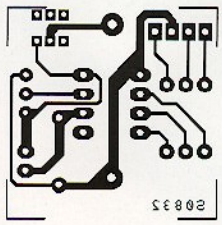


Figure B : circuit imprimé à l'échelle 1 : 1 côté soudures du thermomètre chromatique.

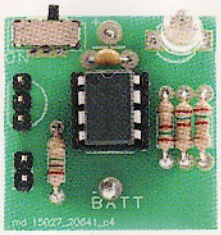


Figure D : photo de l'un de nos prototypes du thermomètre chromatique.

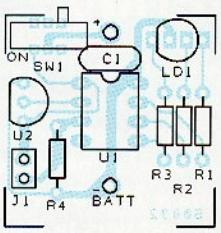


Figure C : schéma de câblage des composants du thermomètre chromatique.

Liste des composants du ET832

R1..... 150 Ω
R2..... 150 Ω
R3..... 150 Ω
R4..... 1 M Ω
C1..... 100 nF multicouche
LD1.... LED RVB 5 mm (540R2GBC-CC)
U1..... PIC12F675 (MF832)
U2..... DS18B20
SW1... interrupteur miniature à glissière coudé à 90°

Divers

Support circuit intégré 2 x 4 broches
Support de pile pour CR2032 pour circuit imprimé
Câble de 10 cm à 5 conducteurs

La particularité de ce capteur de température MAXIM est qu'il peut exprimer la température avec des décimales, reportez-vous à l'illustration de la figure 1. Vous vous apercevez que les 4 premiers bits de l'octet le moins significatif (LSB) se réfèrent à des valeurs inférieures à 1. Le premier bit vaut $\mathbf{2^{-4}}$ soit 1/16. Le deuxième bit vaut **0,125**, le troisième **0,25** et le quatrième **0,5**. Ainsi, le capteur peut exprimer la quasi-totalité des valeurs de la température qui peuvent être mesurées.

Reprenons l'exemple qui illustre la valeur binaire de la température à la page 47 de la revue 135 d'Electronique et Loisirs Magazine. Supposons que nous mesurons une température positive de 64,25 °C. En sortie du DS18B20, nous obtenons **2 octets**, dont le premier sera (en partant du bit le plus significatif) 00000100 et le second (toujours à gauche du bit le plus significatif) 00000100. L'analyse des données nous montre que l'octet « **LSB** » a une valeur de 0,25 et le second octet « **MSB** » une valeur de + 64 degrés.

La valeur totale formée par les deux octets en sortie du DS18B20 peut donc prendre les valeurs comprises entre « 1111101101110000 » (-55 °C) et « 0000011111010000 » (+125 °C).

En convertissant ces valeurs au format hexadécimal, nous obtenons les valeurs FB70h (-55 °C) et 07D0h (+125 °C). Toujours en hexadécimal, nous obtenons 00A2h pour une température de 10,125 °C et 14h pour une température de -20 °C.

Le programme

Maintenant nous allons étudier le fonctionnement du montage, en analysant le programme (firmware) du microcontrôleur. Le code source complet du programme est disponible dans le Listing 1. Il est écrit en BASIC à l'aide du compilateur **PIC BASIC PRO** disponible sur le site www.melabs.com. Vous pouvez télécharger gratuitement une version d'évaluation complètement fonctionnelle mais limitée à 2 Ko de code, ce qui est plus que suffisant pour notre programme.

En effet le listing peut vous sembler long, mais en réalité, une grande partie des lignes de code n'interviennent pas dans la compilation, car elles sont dotées d'une apostrophe au début (symbole « ' »).

Elles servent tout simplement pour la mise au point (débogage) du programme. Reportez-vous aussi aux commentaires dans le listing afin de bien comprendre le déroulement du code.

À la mise sous tension du montage, les entrées/sorties du microcontrôleur sont initialisées, puis les paramètres de la ligne de communication avec le DS18B20 sont définis.

Ensuite **3 routines de commandes** sont configurées pour **envoyer les impulsions adéquates** afin d'allumer la LED RVB (constituée de 3 LED). Les sorties correspondantes des LED sont mises à un niveau logique « 0 » au démarrage.

Ici vous pouvez voir le prototype monté. Le support de pile est soudé du côté des pistes du circuit imprimé (face inférieure).

Listing 1

```
'*************************************************
'* Name    : ET 832 - Thermomètre chromatique     *
'* Author  : Electronique et Loisirs Magazine     *
'* Notice  : Copyright (c) 2016 : All Rights Reserved *
'*                                                *
'* Date    : 02/06/2016                          *
'* Version : 1.0                                  *
'* Notes   :                                      *
'*                                                *
'*************************************************
DEFINE OSC 4

@DEVICEBOD_OFF
@DEVICEMCLR_OFF
@DEVICEINTRC_OSC
'@ DEVICE BOD_OFF

'DEFINE DEBUG_REG GPIO
'DEFINE DEBUG_BIT 4
'DEFINE DEBUG_BAUD 9600
'DEFINE DEBUG_MODE 0

OPTION_REG=%00000000
CMCON =%00000111                    'comparateur désactivé CM0 à CM2 à 1
ADCON0=%00000000
ANSEL=%00000000                     'utilisation des broches GPIO comme entrées/sorties numériques
WPU=%00100000                       'activation des résistances de pull-up
IOCB=%00000000
INTCON=%00000000

SYMBOL CAPTEUR =GPIO.5              'CAPTEUR DS18B20
SYMBOL EAU =GPIO.4
SYMBOL G =GPIO.0
SYMBOL B =GPIO.1
SYMBOL R =GPIO.2


TMP             VAR WORD
'TMP1           VAR WORD
TMP2            VAR BYTE
TMP3            VAR BYTE
VALEUR          VAR WORD
DEGRES          VAR BYTE
PAUSA           VAR WORD
TMPW            VAR WORD
TMPW1           VAR WORD

CONV            VAR BYTE
TEMP            VAR WORD
TEMPW           VAR WORD

TMPW = 500
TMP2 =15
TMPW1 = 7500
PAUSA=1000

 'La chaleur de la couleur (TMP) passe de 0 à 500
 'La température de l'eau va de 20 à 50

LOW G
LOW R
LOW B
```

```
HIGH R
PAUSE 1000
LOW R
PAUSE 1000

MAIN:

   IF EAU =1 THEN
      GOSUB TEMPERATURE

      VALEUR = (DEGRES * 35)     '700=20 DEGRES
      IF VALEUR>=700 THEN
         VALEUR=VALEUR-700
         IF VALEUR>1000 THEN
            VALEUR=1000
         ENDIF
      ELSE
         VALEUR=0
      ENDIF
      'DEBUG #VALEUR,13,10
      FOR TMP3=0 TO 20
         GOSUB COULEUR
         PAUSE 1
      NEXT TMP3
   ELSE
      LOW G
      LOW R
      LOW B
      DEGRES=9999
      SLEEP 1
   ENDIF

GOTO MAIN

COULEUR:

   SELECT CASE VALEUR
      CASE IS <500
         TMP=VALEUR
         LOW R
         HIGH B
         LOW G
         PAUSEUS TMPW1-(TMP*TMP2)
         LOW B
         HIGH G
         PAUSEUS TMP*TMP2

      CASE IS =>500
         TMP=VALEUR-500
         LOW B
         HIGH G
         LOW R
         PAUSEUS TMPW1-(TMP*TMP2)
         LOW G
         HIGH R
         PAUSEUS TMP*TMP2
   END SELECT

RETURN

TEMPERATURE:

         OWOUT CAPTEUR, 1, [$CC, $44]

CONVERSION:
```

```
        OWIN CAPTEUR, 4, [CONV]
           if DEGRES<>9999 THEN
              GOSUB COULEUR
           ENDIF
        IF CONV = 0 THEN CONVERSION                    'Si la conversion n'est pas finie
        OWOUT CAPTEUR, 1, [$CC, $BE]
        'OWIN CAPTEUR, 0, [TMP2, TMP1,skip 7]
        OWIN CAPTEUR, 0, [Temp.LOWBYTE,Temp.HIGHBYTE,skip 7]
        'SEROUT2 TXPC,84,[ «TEMPERATURE BIN «,BIN16 TEMP,13,10 ]


'       IF Temp.bit15 = 0 THEN                         ' Vérification de la température en dessous de 0 ° C
          'Sign = «+»
          DEGRES = 0
          DEGRES = (Temp >> 4)
          tmpw=1000/(160/(TEMP & %0000000000001111))
          IF (TEMP & %0000000000001111)=%1111 THEN
             TMPW=0
             DEGRES=DEGRES+1
          ENDIF
          'SEROUT2 TXPC,84,[«1Temp = +», DEC VALEUR,».», DEC2 TmpW, «C «,10,13]
'         TEMP=0
'         TEMP=VALEUR
'       ELSE

'         tmpw=1000/(160/(TEMP & %0000000000001111))
'         IF (TEMP & %0000000000001111)=%1111 THEN
'            TMPW=0
'            TMP3=TMP3+1
'         ENDIF
'         VALEUR = 0
'         VALEUR = 65535 - TEMP                        'obtient une valeur comme si elle était positive
'         VALEUR = 0
'         VALEUR = (VALEUR >> 4)
'         tmpw=1000/(160/(VALEUR & %0000000000001111))
'         IF (VALEUR & %0000000000001111)=%1111 THEN
'            TMPW=0
'            VALEUR=VALEUR-1
'         ENDIF
'         VALEUR = (TEMP & %0000011111111111)
'         VALEUR = VALEUR ^ %0000011111111111
'         VALEUR = VALEUR >> 4
'       SEROUT2 TXPC,84,[«1Temp = -», DEC VALEUR,».», DEC2 TmpW, «C «,10,13]
'         TEMP=0
'         TEMP=VALEUR
'         TEMP.7=1
'       ENDIF
'         TEMPW=(VALEUR*10)+(TMPW/10)
          ' SEROUT2 TXPC,84,[«TempW =», #TEMPW,»C «,10,13]

        ' SEROUT2 TXPC,84,[ «TEMPERATURE DEC «]
'       TEMP=0
'       IF TEMP.7=0 THEN
'         SEROUT2 TXPC,84,[ «+»]
'       ELSE
'         SEROUT2 TXPC,84,[ «-»]
'       ENDIF
'   DEBUG #VALEUR,» DEGRES»,13,10
 'valeur décimale de la température
RETURN
```

Les seuils des températures que nous avons définis sont d'environ **14 °C** pour le froid, **20 °C** pour le tiède et **30 °C** pour le chaud. À ce stade, le PIC interroge le capteur MAXIM pour mesurer la température du liquide. Il se présente alors 3 cas :

1. la température est inférieure à 14 °C ;
2. la température est supérieure à 14 °C et inférieure à 20 °C ;
3. la température est supérieure à 30 °C.

Dans le 1er cas, le microcontrôleur laisse à un niveau logique « 0 » la sortie qui commande la jonction rouge de la LED RVB (GP2) et pulse avec des intervalles différents les sorties relatives aux LED verte (GP0) et bleue (GP1).

Plus exactement, l'intervalle de temps pendant lequel la LED bleue reste éclairée est plus grand que celui pendant lequel la sortie contrôlant la LED verte reste à un niveau haut.

Plus la température est basse, plus la lumière émise par la LED RVB tend vers la couleur bleue. Si la température augmente, la durée d'allumage de la LED verte devient proche de celle de la LED bleue.

Lorsque la température dépasse 14 °C, mais reste inférieure à 20 °C, le firmware fixe à un niveau logique « 0 » la sortie (GP1) correspondant à la LED bleue, et pulse les sorties des LED rouge et verte en alternance. La période d'allumage de la LED rouge est initialement à « 0 », de sorte que pour des températures légèrement supérieures à 14 °C, la lumière émise par la LED RVB est verte.

Ensuite la période d'allumage de la LED rouge augmente au détriment de la période d'allumage de la LED verte. Il en résulte que lorsque la température augmente, la lumière devient jaune.

Lorsque la température dépasse 20 °C, la sortie de la LED bleue reste à « 0 » et la sortie de la LED rouge s'allume de plus en plus au détriment de la sortie de la LED verte. Lorsque la température atteint 30 °C, la LED RVB passe de l'orange au rouge.

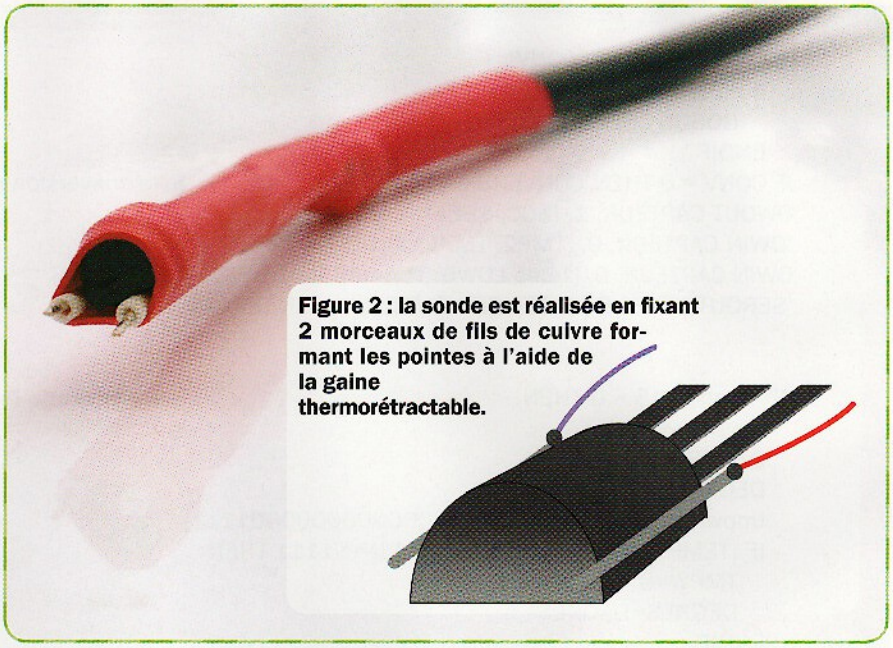


Figure 2 : la sonde est réalisée en fixant 2 morceaux de fils de cuivre formant les pointes à l'aide de la gaine thermorétractable.

Réalisation pratique

Nous allons voir maintenant comment construire notre thermomètre chromatique. Vous devez d'abord fabriquer le circuit imprimé, pour cela vous devez télécharger le typon sur notre site **www.electroniquemagazine.com** dans le sommaire détaillé de la revue **136**.

Le circuit est relativement simple à fabriquer puisqu'il ne dispose que d'une seule face, vous ne devriez pas rencontrer de problèmes particuliers.

Commencez par souder les 4 résistances, ensuite l'unique condensateur C1. Continuez avec le support du PIC dont le repère en « U » doit être orienté vers C1, l'interrupteur à glissière coudé à 90 °, le support de pile soudé du côté des pistes de cuivre (côté soudures). Faites attention à l'orientation.

Enfin terminez en soudant la LED tricolore, le méplat doit être orienté vers C1.

Maintenant nous allons préparer la sonde, pour cela nous allons utiliser du câble blindé à 2 fils plus la tresse de masse et un morceau de gaine thermorétractable de diamètre à froid de 7 à 8 mm. Il faudra prévoir aussi 2 morceaux de fil de cuivre dénudés de 1 mm de Ø ou sinon vous pouvez utiliser à la place des pattes de résistances.

Commencez par dénuder les 2 conducteurs du câble blindé que vous soudez à la broche « DQ » et « VCC » du capteur DS18B20. La tresse de mase du câble blindé doit être soudée à la broche GND du capteur et l'autre extrémité à la masse du montage.

Pour ne pas vous tromper, rappelez-vous que lorsque vous regardez le méplat du capteur DS18B20 face à vous, la broche « VCC » se trouve à votre droite, « DQ » au milieu et la broche GND (masse) à votre gauche.

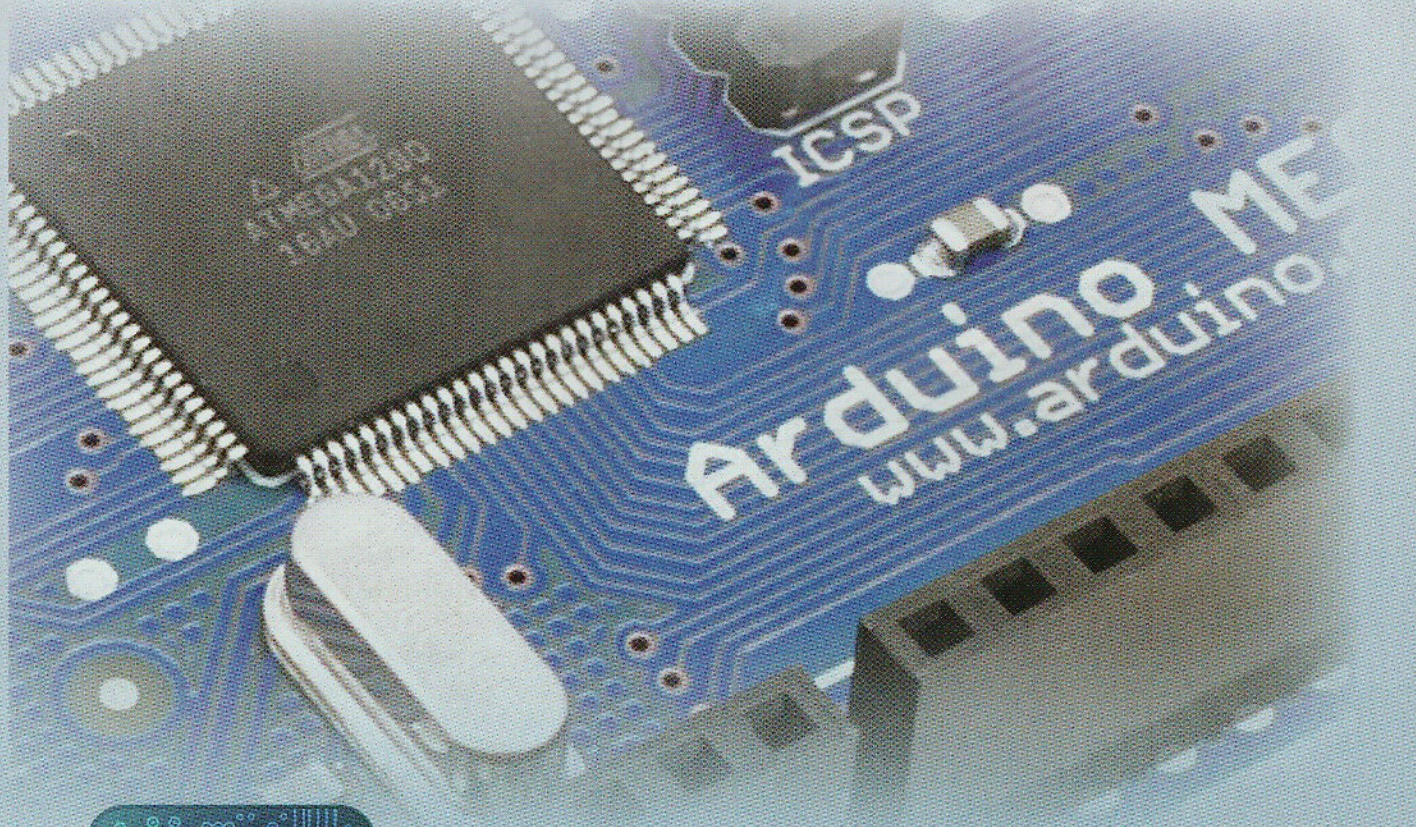
Maintenant, alignez les 2 morceaux de fils de cuivre dénudés (de longueur égale), avec du ruban adhésif fixez les 2 morceaux de cuivre de chaque côté du boîtier du capteur en laissant dépasser légèrement les extrémités sur la partie supérieure du DS18B20.

Ensuite chauffez la gaine thermorétractable en vous assurant de bien couvrir les soudures et en laissant dépasser les pointes (voir la figure 2). Ceci permet d'éviter que le liquide entre en contact avec les fils de la sonde.

Vous devez **plonger le DS18B20 à quelques millimètres seulement dans le liquide**,de sorte que, seuls les 2 électrodes et le haut du boîtier en plastique du capteur, baignent dans le liquide.

Evitez d'immerger la partie où se situe la gaine thermorétractable, du liquide pourrait s'introduire au niveau des soudures et perturber la mesure. ■

Cours ARDUINO - 8



Cours ARDUINO

Huitième partie

de Mirco SEGATELLO

Dans cette huitième partie du Cours Arduino, nous allons étudier dans le détail la carte Arduino UNO, qui est la carte de développement et de prototypage qui a remplacé la populaire « Duemilanove ».

D'origine **Italienne**, la carte **Arduino Uno** doit son nom à un bar nommé « **Re Arduino** » dans lequel se réunissaient les concepteurs de la carte. Sur l'emballage de la carte, la mention « **Made in Italy** » est clairement visible.

Avec l'arrivée de cette carte Arduino Uno, est apparu un nouveau logo pour le projet Arduino. Il s'agit du symbole mathématique de l'infini (∞), sans doute parce qu'il symbolise les possibilités « infinies » d'utilisation de la carte ou plus globalement du projet Open Source Arduino.

Vous trouverez dans l'emballage de la carte Arduino Uno un petit manuel en Anglais détaillant les instructions d'utilisation de la licence, comme vous pouvez vous y attendre dans le cas d'un produit Open Source international.

La carte présente une excellente qualité, avec une sérigraphie très précise et complète. La carte Arduino Uno est basée sur un microcontrôleur de type **ATmega328**. Elle dispose de **14 entrées/sorties digitales** (6 peuvent être utilisées comme sorties PWM), de **6 entrées analogiques**, d'un **oscillateur** de **16 Mhz**, d'un **connecteur USB**, d'un **connecteur ICSP** (programmation en circuit), d'un bouton de « **reset** » et d'une prise d'alimentation.

Schéma électrique

Le schéma électrique de la carte Arduino Uno est visible en figure A. Contrairement aux modèles précédents, la carte **Arduino Uno n'utilise pas le circuit FTDI pour l'interface USB**, mais un **microcontrôleur ATmega16U2 programmé**

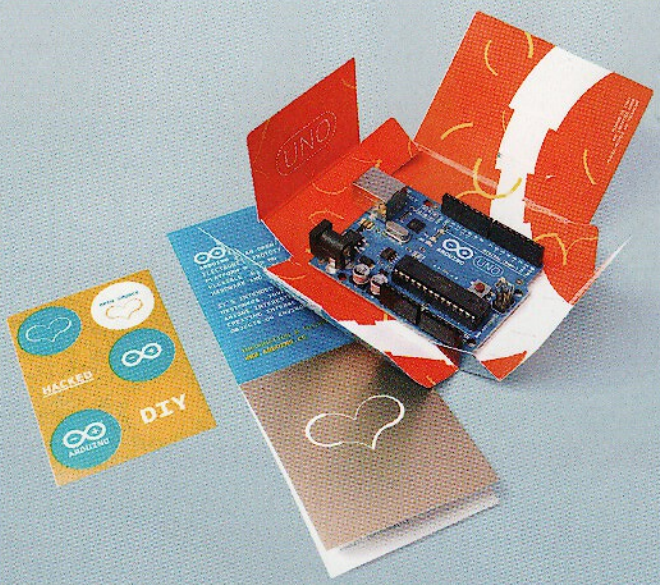


Figure 1 : ici le contenu de l'emballage de la carte Arduino Uno.

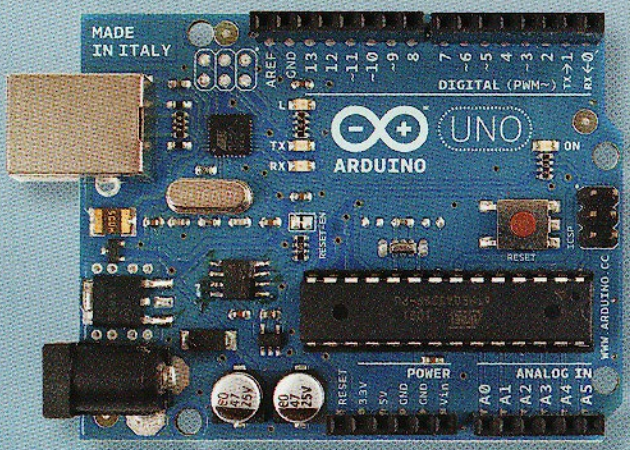


Figure 2 : aspect de la carte Arduino Uno vue de dessus.

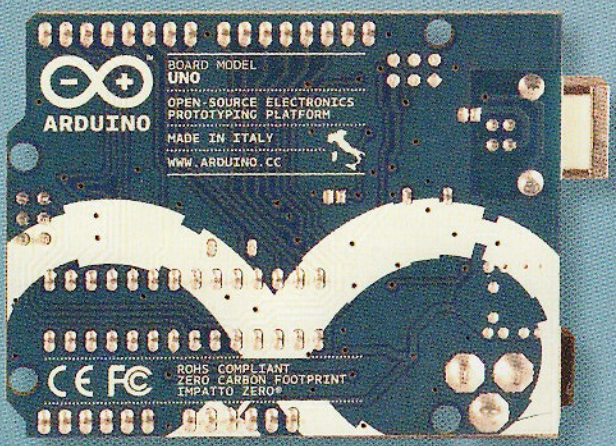


Figure 3 : aspect de la carte Arduino Uno vue de dessous. Notez que la carte Arduino Uno Rev 3 dispose d'une sérigraphie inversée de couleur blanche sur sa face inférieure.

comme **convertisseur USB/série**. L'ATmega16U2 est en fait un microcontrôleur doté d'un module USB entièrement programmable, qui est l'équivalent de celui de la série PIC18F de chez Microchip. Le connecteur USB, lorsqu'il est connecté à un PC, permet l'alimentation et la programmation de la carte.

La carte Arduino Uno est dotée d'une certification FCC concernant les émissions électromagnétiques (« ROHS Compliant »), et n'utilise que très peu de carbone pour sa fabrication (« Zero carbon footprint ») afin de protéger l'environnement.

La carte Arduino Uno adopte le nouveau standard de compatibilité des broches (pins) nommé « **1.0 PINOUT** » avec l'adjonction de quelques nouvelles broches :

- les broches « **SDA** » et « **SCL** » sont placées près de la broche « **AREF** » (liaison I2C/TWI), c'est une duplication des pins AD4 et AD5 qui réalise cette fonction ;
- la broche « **IOREF** » est disposée près de la broche « **RESET** », cela permet aux « shields » (cartes d'extension) de s'adapter à la tension de référence fournie par la carte (5 V et 3,3 V) ;
- une broche de réserve à proximité de « IOREF » est prévue pour une utilisation future.

Le **Tableau 1** résume les principales caractéristiques de la carte Arduino Uno. L'alimentation de la carte peut s'effectuer via le port USB, mais un connecteur classique est disponible sur la carte et accepte une tension non régulée comprise entre 7 V et 12 V.

L'idéal est d'alimenter la carte avec une tension de 9 V non régulée (attention aux blocs secteurs qui délivrent près de 20 V à vide bien que donnés pour 12 V). Vous pouvez aussi utiliser une batterie de 12 VDC. La source d'alimentation entre la prise classique et le connecteur USB est détectée automatiquement, il n'y a donc pas d'interrupteur.

Le port USB est protégé sur la carte contre les courts-circuits accidentels, dans tous les cas il ne consommera pas plus de 500 mA (maximum autorisé). Les broches d'alimentation de la carte Arduino Uno sont les suivantes :

- « **VIN** » : réplique la tension présente sur le connecteur d'alimentation lorsque la carte est utilisée avec une source de tension externe (à ne pas confondre avec le 5 V de l'USB). Cela permet d'alimenter d'autres cartes dotées d'un régulateur de tension ;
- « **5 V** » : c'est la tension régulée disponible en sortie du régulateur interne et qui permet d'alimenter le microcontrôleur et les autres composants de la carte. Cette tension peut donc provenir soit de la tension d'alimentation « VIN », soit du connecteur USB ;
- « **3V3** » : cette broche fournit une tension stabilisée de 3,3 VDC provenant d'un régulateur interne. Cela permet d'alimenter des circuits fonctionnant en 3,3 V, le courant maximal de sortie disponible sur cette broche est de 50 mA ;
- « **GND** » : c'est la broche de masse de la carte (0 V).

Tableau 1 - principales caractéristiques de la carte Arduino UNO.

Microcontrôleur	ATmega328
Tension de fonctionnement	5 VDC
Alimentation externe recommandée	7 V à 12 V
Alimentation externe limite	6 V à 20 V
Entrées/sorties (I/O) digitales	14 (dont 6 utilisables comme sorties PWM)
Entrées analogiques	6
Courant de sortie pour chaque broche I/O	40 mA (200 mA cumulé pour l'ensemble des broches I/O)
Courant de sortie sur la broche 3,3 V	50 mA
Mémoir FLASH	32 ko (ATmega328) dont 0,5 ko utilisés par le bootloader
SRAM	2 ko (ATmega328)
EEPROM	1 ko (ATmega328)
Fréquence d'horloge	16 MHz

En ce qui concerne les 14 broches d'entrées/sorties (I/O), chacune d'elle peut être configurée en entrée ou en sortie et peut délivrer (ou absorber) un **courant maximal de 40 mA** (dans la limite de **200 mA pour l'ensemble des broches** d'entrées/sorties). Il est possible d'activer, par programmation, pour chacune de ces broches une résistance de tirage (pull-up) d'une valeur allant de 20 kΩ à 50 kΩ.

Caractéristiques de la carte Arduino Uno

Cette carte est dotée d'un microcontrôleur **ATmega328** disposant de **32 ko** de mémoire programme (FLASH), dont 0,5 ko sont utilisés par le « **Bootloader** ». Celui-ci est un programme préprogrammé une fois pour toute dans le microcontrôleur ATmega, et qui permet à celui-ci de communiquer avec l'environnement de développement Arduino via le port USB (par exemple lors de chaque phase de programmation).

L'ATmega328 dispose également de **2 ko** de **SRAM** et de **1 ko** de mémoire **EEPROM** utilisée pour la sauvegarde de données permanentes (conserve les données sans alimentation). L'EEPROM peut être lue à l'aide de la librairie « EEPROM ».

D'autre part la carte Arduino Uno dispose de **6 entrées analogiques** (A0, A1, A2, A3, A4 et A5), qui par défaut ont une **résolution de 10 bits** et acceptent une tension comprise entre 0 et 5 VDC. Toutefois il est possible de paramétrer la broche « **Aref** » (tension de référence pour les entrées analogiques) de façon à modifier la plage de mesure.

Les broches **2** et **3** permettent **deux interruptions externes**, elles peuvent être configurées pour déclencher une interruption lors de la détection d'un front montant ou descendant, ou lors d'un changement de valeur.

Le **module PWM** du microcontrôleur ATmega est assigné aux broches **3**, **5**, **6**, **9**, **10** et **11**. Ce module est configurable par logiciel afin de **générer des signaux PWM** ayant une **résolution de 8 bits**.

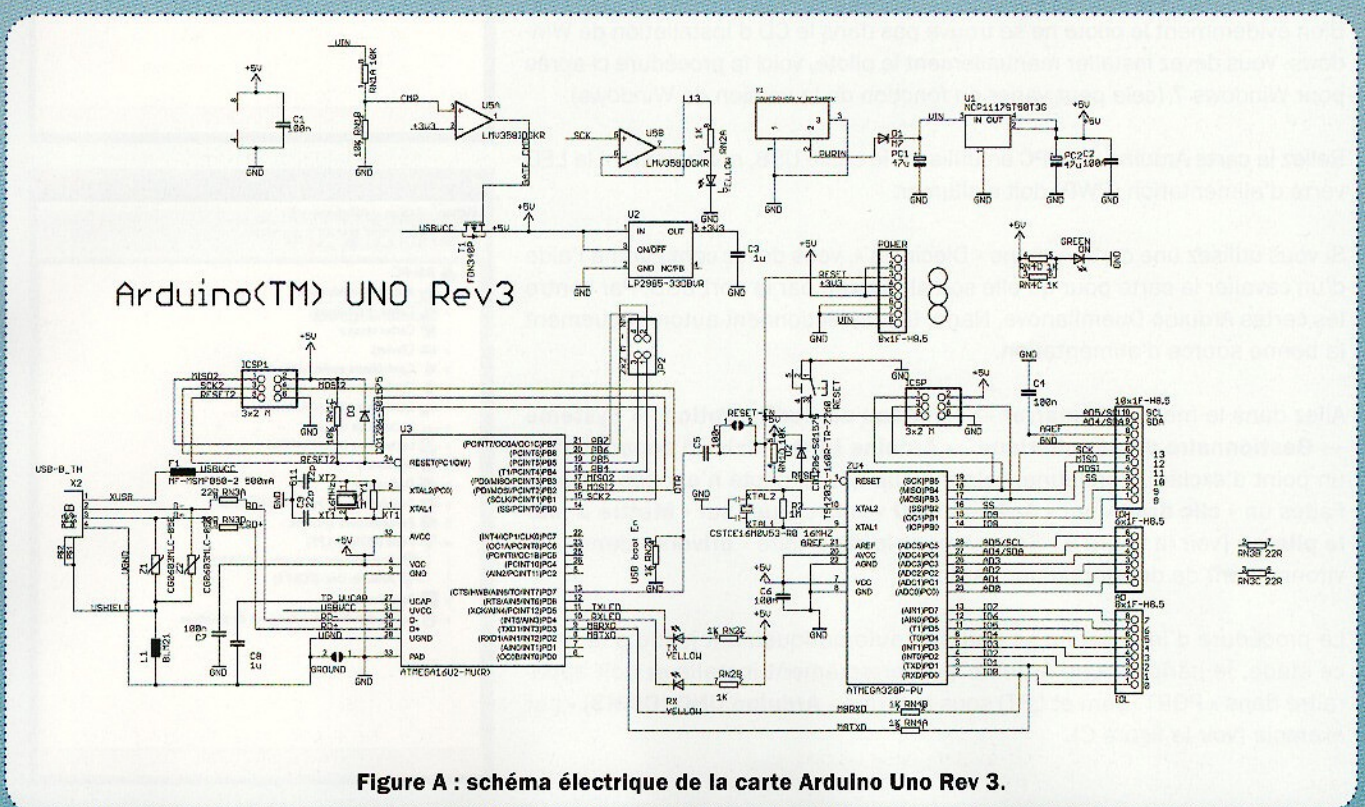


Figure A : schéma électrique de la carte Arduino Uno Rev 3.

À l'aide d'un simple **filtre RC** connecté sur les sorties PWM, il est possible d'obtenir des **tensions continues de valeurs variables**.

La **communication série** est assurée par les **broches TX(1)** et **RX(0)**. La broche « **RX** » permet de **recevoir** et la broche « **TX** » de **transmettre** des données séries avec un niveau TTL. Ces broches sont reliées aux broches correspondantes du microcontrôleur **ATmega16U2 programmé comme convertisseur USB/série**. Celui-ci assure l'interfaçage entre les signaux TTL et ceux du port USB de l'ordinateur. L'**ATmega16U2** assure la **communication** série vers le **port USB du PC**, cependant il apparaît comme un port « **COM virtuel** » pour l'ordinateur. L'IDE Arduino est doté d'un « **Moniteur série** » qui permet d'envoyer des commandes depuis le PC vers la carte Arduino Uno.

Les LED « RX » et « TX » sur la carte clignotent lorsque les données sont transmises via le convertisseur USB/série (connexion USB vers PC), mais pas lors de la communication série via les broches TX(1) et RX(0).

Contrairement au circuit intégré FDTI pour lequel il était nécessaire d'installer des pilotes appropriés, l'utilisation de l'ATmega16U2 permet d'utiliser des pilotes génériques déjà disponibles dans le système d'exploitation.

Cependant, avec les systèmes Windows, pour la création correcte d'un port « **COM virtuel** », il est nécessaire d'installer un **pilote supplémentaire.** Arduino Uno est compatible avec les systèmes d'exploitation Windows, Mac OS et Linux, pour lesquels les pilotes sont disponibles.

Installer Arduino Uno sous Windows 7

Windows 7, ainsi que les versions suivantes, sont structurées de manière à installer automatiquement les pilotes (drivers) pour chaque périphérique en recherchant éventuellement sur le web les plus appropriés. Cependant Arduino est un périphérique particulier, c'est pourquoi la procédure automatique d'installation des pilotes ne peut pas s'effectuer.

Lorsque la carte Arduino est connectée au PC, le système d'exploitation identifie la présence d'un nouveau périphérique sans être en mesure d'installer le pilote correspondant.

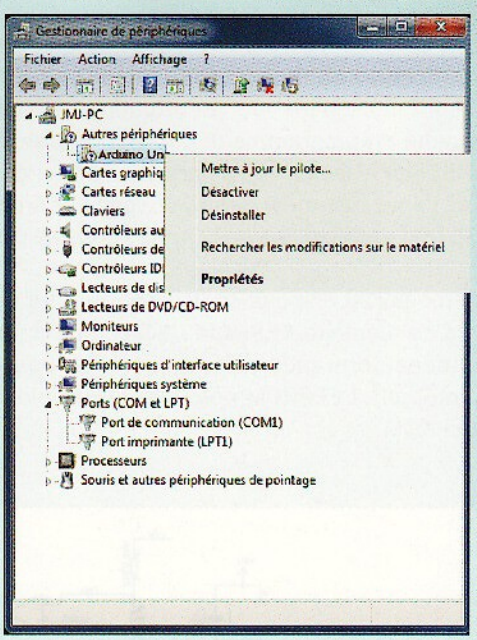
Bien évidemment le pilote ne se trouve pas dans le CD d'installation de Windows. Vous devez installer manuellement le pilote, voici la procédure ci-après pour Windows 7 (cela peut varier en fonction de la version de Windows).

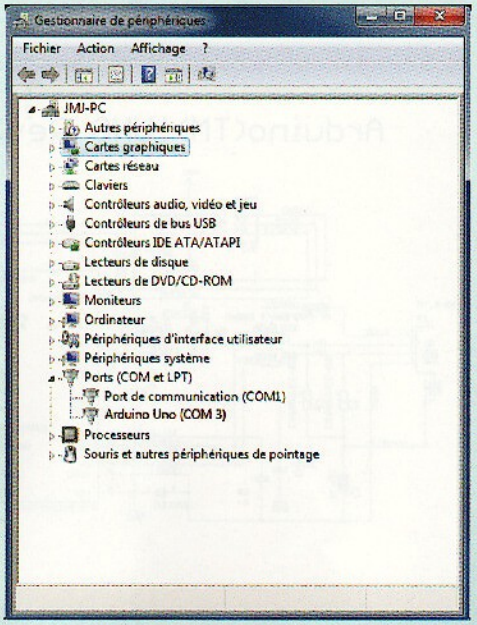
Reliez la carte Arduino à un PC en utilisant le câble USB, normalement la LED verte d'alimentation (PWR) doit s'allumer.

Si vous utilisez une carte Arduino « Diecimila », vous devez configurer à l'aide d'un cavalier la carte pour qu'elle soit alimentée par le port USB. Par contre les cartes Arduino Duemilanove, Nano, Uno sélectionnent automatiquement la bonne source d'alimentation.

Allez dans le menu « **Démarrer → Panneau de Configuration → Système → Gestionnaire de périphérique → Arduino UNO** ». Celui-ci apparait avec un point d'exclamation jaune, cela indique que le pilote n'est pas installé. Faites un « **clic droit** » sur « **Arduino UNO** » puis cliquez sur « **Mettre à jour le pilote** » (voir la figure B). Sélectionnez le répertoire « **drivers** » dans l'environnement de développement Arduino.

La procédure d'installation se poursuit automatiquement jusqu'à la fin. À ce stade, le périphérique Arduino est correctement installé et doit apparaître dans « PORT (Com et LPT) sous la forme « **Arduino UNO (COM3)** » par exemple (voir la figure C).





Le **bus SPI** occupe les broches **10 (SS)**, **11 (MOSI)**, **12 (MISO)**, **13 (SCK)**. Ces broches supportent la communication SPI (Interface Série Périphérique) à l'aide de la librairie SPI.

La carte Arduino Uno est dotée également d'un **bus I2C** disponible sur les broches **4 (SDA)** et **5 (SCL)**. Elles permettent la communication I2C standard sur 2 fils en utilisant la **librairie Wire/I2C**.

Nous terminons le descriptif des fonctions de la carte Arduino avec la broche 13 qui est reliée à une LED interne. Celle-ci permet de visualiser des messages de diagnostics.

La broche de « **RESET** », lorsqu'elle est mise à un **niveau bas**, réinitialise le microcontrôleur. Un bouton présent sur la carte permet aussi une réinitialisation.

Installation et utilisation

Pour une bonne utilisation de la carte Arduino Uno, vous devez disposer de la dernière version de l'environnement de développement Arduino, c'est-à-dire la version « **ARDUINO 1.8.0** » (au moment où ces lignes sont écrites). Vous pouvez télécharger cette version à l'adresse suivante : **https://www.arduino.cc/en/Main/Software**. Vous pouvez sélectionner la version en fonction de votre système d'exploitation.

Avec un système d'exploitation de type Windows, il suffit de connecter la carte Arduino au PC pour qu'elle soit reconnue en tant que nouveau périphérique. Windows vous demandera alors d'installer le pilote, il suffit de spécifier correctement le chemin du dossier « **driver** » dans le programme Arduino (Arduino UNO.inf).

À la fin de l'installation, un port **COM virtuel** est créé (voir l'encadré intitulé « Installer Arduino Uno sous Windows 7 » dans ces pages).

Lancez l'environnement de développement Arduino et sélectionnez la carte dans le menu « **Outils → Type de carte** ». Ensuite vous devez sélectionner le port série utilisé pour la communication avec la carte Arduino depuis le menu « **Outils → Port** ». Ce sera probablement le port COM3 ou supérieur (les ports COM1 et COM2 sont réservés).

Pour tester le fonctionnement de la carte, ouvrez le fichier « **Blink** » (**Fichier → Exemples → Digital → Blink**). Transférez le programme vers la carte Arduino en cliquant sur « **Croquis → Téléverser** ». Les LEDs RX et TX doivent clignoter rapidement, cela indique que le programme est en cours de transfert.

Une fois la carte Arduino réinitialisée à la fin du transfert, le programme s'exécute en faisant clignoter la LED de la broche 13.

Grâce au « **bootloader** » préinstallé dans le microcontrôleur Atmega, vous n'avez pas besoin d'utiliser un programmateur externe ou de retirer le microcontrôleur.

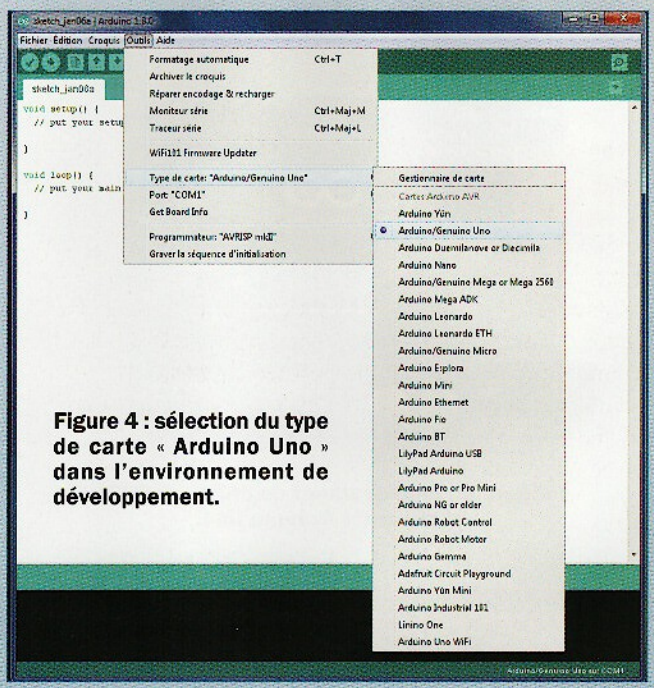


Figure 4 : sélection du type de carte « Arduino Uno » dans l'environnement de développement.

La connexion USB entre le PC et la carte Arduino permet la programmation directe et la gestion de la communication. La fonction de réinitialisation automatique interne à la carte permet la programmation d'un simple clic de souris.

Le « bootloader » de la carte Arduino Uno est basé sur le protocole « **STK-500** », il occupe 512 octets de mémoire et peut gérer une vitesse de communication jusqu'à 115 kbps. Arduino UNO est conçue pour communiquer en mode série avec un PC à l'aide de l'outils « Moniteur Série ». De cette manière, les données acquises par la carte peuvent facilement être affichées à l'écran.

Le microcontrôleur « ATmega16U2 » utilisé comme convertisseur USB/série peut facilement être programmé car il dispose aussi d'un « bootloader » préinstallé. Dans ce cas, vous devez utiliser un programme dénommé « **DFU Programmer** » qui met à jour le microcontrôleur « ATmega16U2 » (ou ATmega8U2 selon la version de la carte Uno que vous possédez). Attention, vous devez maîtriser l'environnement Arduino.

Le logiciel requis pour la mise à jour de l'ATmega16U2 (ou 8U2) s'appelle « **Atmel's FLIP software** » pour Windows. Vous pouvez le télécharger à cette page :

http://www.atmel.com/tools/flip.aspx.

Choisissez la dernière version « FLIP 3.4.7 for Windows ».

Sous linux, tapez la commande :

```
sudo apt-get install dfu-programmer
```

Ensuite vous **devez télécharger à cette adresse le firmware que vous allez transférer** dans l'ATmega16U2 (ou 8U2) pour le mettre à jour :

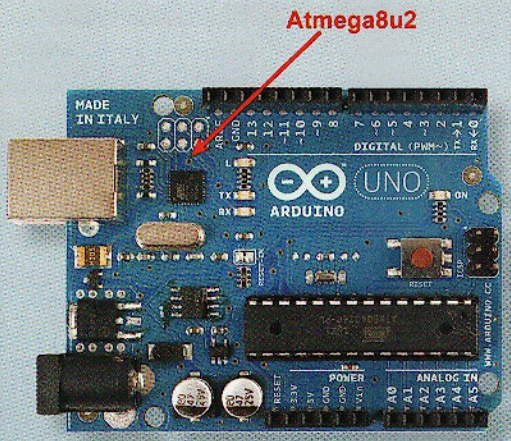


Figure 5 : emplacement de l'ATmega8U2 sur la carte Arduino Uno.

soudez une résistance de 10 kΩ sur le cavalier pour activer le mode de programmation du bootloader de l'ATmega8U2.

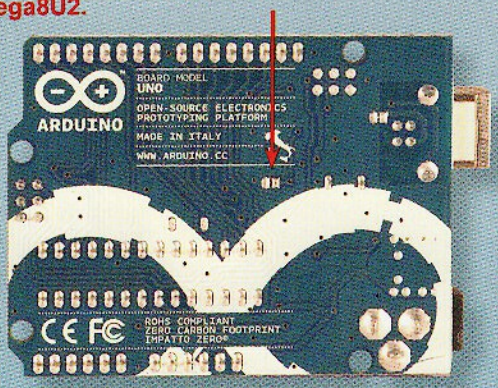


Figure 6 : soudez verticalement une résistance de 10 kΩ sur les 2 pastilles comme indiqué afin d'activer le mode de programmation pour la mise à jour du bootloader.

https://github.com/arduino/Arduino/tree/master/hardware/arduino/avr/firmwares/atmegaxxu2.

Choisissez la version qui convient à votre carte, cliquez sur « **arduino-usbserial** » puis sur « **Arduino-usbserial-uno.hex** ».

Cliquez sur le bouton « **Raw** » et copiez toutes les lignes depuis votre navigateur en cliquant sur « **Ctrl + A** » puis « **Ctrl +C** » dans le Bloc-note ou tout autre éditeur de texte. Enregistrez le fichier sous « **Arduino-usbserial-uno.hex** » **et non avec l'extension « .txt »**.

Ensuite plusieurs cas se présentent suivant la version de la carte Arduino Uno que vous possédez. Votre carte est une « **Uno révision 1 ou 2** », elle dispose d'un **ATmega8U2** qui est situé du côté du connecteur USB (voir la figure 5).

Vous **devez souder verticalement une résistance de 10 kΩ** sur les **2 pastilles** comme indiqué en figure 6.

Si votre carte est une « **révision 3** », elle dispose d'un **ATmega16U2**, il n'y a pas de modification à effectuer.

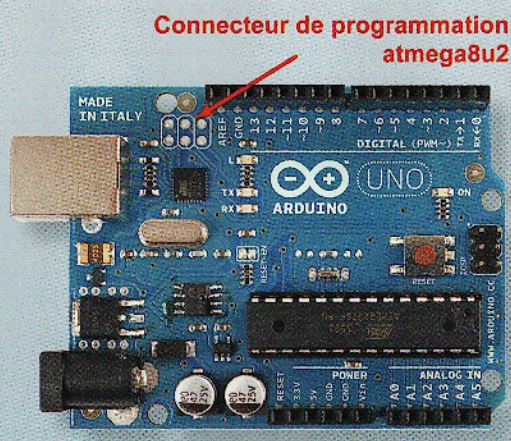


Figure 7 : emplacement de la broche de reset de l'ATmega16U2 ou 8U2 sur la carte Arduino Uno (la 3ème située sur la gauche près de la prise USB).

Reliez la carte à l'ordinateur à l'aide du câble USB. Vous devez maintenant effectuer un reset du 16U2 ou 8U2, à l'aide d'un morceau de fil reliez brièvement le cavalier de reset situé le plus proche de la prise USB (voir la figure 7).

Lancez l'IDE Arduino, pour vérifier que le microcontrôleur 8u2 ou 16u2 est bien réinitialisé, allez dans le menu et vérifiez la liste des ports série. **Le port série de votre carte ne doit plus apparaître.**

Maintenant pour mettre à jour le bootloader sous Windows utilisez « **FLIP** » pour télécharger le fichier « **hex** » vers votre carte.

Sous Linux, tapez la commande :

```
sudo dfu-programmer atmega16u2 erase
```

Lorsque cette commande est terminée, vous obtenez une nouvelle invite de commande, tapez :

```
sudo dfu-programmer atmega16u2 flash Arduino-usbserial-uno.hex
```

Enfin, tapez :

```
sudo dfu-programmer atmega16u2 reset
```

Si vous disposez d'un ATmega8U2, tapez atmega8u2 à la place de atmega16u2 dans les lignes ci-dessus.

Normalement votre carte doit être mise à jour, vérifiez le menu « port série » de l'IDE Arduino. **Le port de votre carte doit s'afficher de nouveau.** Si vous débutez avec Arduino, reportez-vous au livre l'**ABC d'Arduino**.

Nous arrivons à la fin de cette huitième leçon, dans la prochaine leçon nous étudierons comment interfacer un récepteur GPS avec une carte Arduino et nous ferons nos premières expériences.

Cours de programmation sur IPhone

de Walter Dal Mut et Francesco Ficili

Nous avons étudié dans les cours précédents l'interface utilisateur et les contrôles graphiques, nous allons maintenant approfondir les notions de « autorotating » (autorotation de l'affichage) et de « autosizing » (dimensionnement automatique de l'affichage). Enfin nous conclurons avec un projet concret réalisé à l'aide d'un « Table View ».

Dans cette 4ème partie de notre cours de programmation sur iPhone, nous évoquerons quelques notions sur l'interface utilisateur des appareils Apple. Nous allons faire un aperçu rapide de tous les contrôles graphiques à disposition et décrire un premier exemple d'une interaction « utilisateur/appareil ».

Ensuite, nous aborderons les notions d'autorotation et de dimensionnement automatique, deux propriétés intéressantes qui vous permettront de personnaliser l'affichage lorsque vous allumerez l'appareil, en analysant les problèmes liés à la rotation et le dimensionnement de l'écran.

Enfin nous terminerons par un nouveau projet pratique, réalisé à l'aide d'un « Table View », c'est-à-dire le type de vue le plus utilisé dans les applications iPhone.

L'interface utilisateur

L'iPhone, comme l'iPad Touch, dispose d'un écran tactile. Celui-ci est la seule interface sur laquelle l'utilisateur peut interagir. D'un point de vue programmation, il faut générer des actions qui fonctionnent avec le « **controller** », comme nous l'avons déjà étudié avec le modèle « **MVC** » (voir les cours précédents).

Tous les designs graphiques que nous pouvons réaliser sur l'iPhone, proviennent de la classe de base de type « **UIView** ».

Ce concept est fondamental, les labels, les boutons, les sélecteurs et tous les autres composants visuels sont des extensions de cette classe de base.

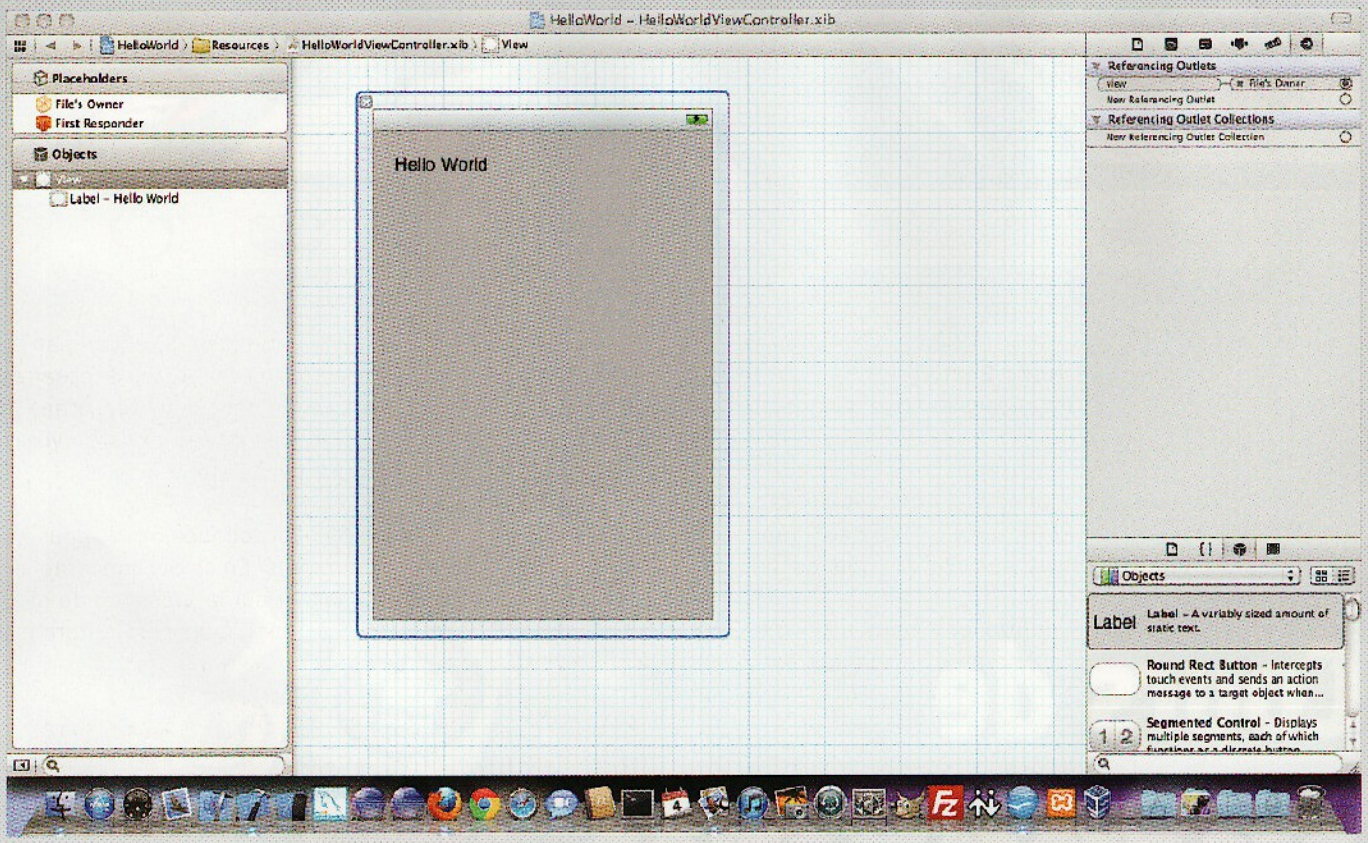


Figure 1 : ajout d'un bouton à notre Interface.

Les composants graphiques fournis par Apple sont nombreux, ils peuvent être divisés en 4 grandes catégories :

- **Controllers** ;
- **Data Views** ;
- **Input & Views** ;
- **Windows, Views & Bars.**

Les « **Controllers** » sont une méthode pour intégrer à notre graphisme des contrôles existants et prêts à l'emploi. Parmi eux se distinguent les « **Tab Bar Controller** », les « **Navigation Controller** », les « **TableView Controller** » et les « **View Controller** » que nous avons déjà étudiés et utilisés dans le dernier cours.

Nous allons analyser à partir de maintenant d'autres « Controllers » en commençant par le « TableViewController », nous en verrons d'autres dans les prochaines leçons.

La seconde catégorie « **Data Views** », concerne les vues dédiées à résoudre des problèmes particuliers, comme les « **TableView** » pour les tableaux, les « **ImageView** » pour visualiser des images, les « **WebView** » (que nous utiliserons prochainement pour créer notre propre navigateur web), les « **MapView** » pour afficher des cartes et les « **pickers** » pour sélectionner des dates ou des valeurs d'une liste.

La troisième catégorie « **Input & Views** » est la plus utilisée dans la conception de l'interface graphique. Nous trouvons dans cette catégorie les « **Label** » pour afficher une notification, les « **Button** » pour interagir avec l'utilisateur, les « **Text Field** » (zones de texte) pour récupérer le texte inséré par l'utilisateur, les « **Switch** », etc.

La dernière catégorie correspond aux « **Windows, Views & Bars** », elle est dédiée à la navigation dans l'interface graphique.

Nous trouvons la « **Search Bar** » pour effectuer des recherches personnalisées, la « **ToolBar** » (barre d'outils) pour créer une barre d'outils personnalisée pour l'application, ajouter du contenu et plus encore.

Vous pouvez naviguer dans ces catégories directement à partir de « **Interface Builder** » et approfondir vos connaissances grâce à la documentation en ligne.

Listing 1 : ajout du prototype dans l'en-tête

```
#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController {
        UILabel *label;
}

- (IBAction) cliquezIci:(id)sender;

@property (nonatomic, retain) IBOutlet UILabel * label;

@end
```

Reprenons le projet « Hello World »

Revenons sur le code écrit dans la leçon précédente (voir la revue 137), nous voulons ajouter l'interaction avec l'utilisateur. Pour cela, nous devons d'abord ajouter un « **Button** » dans notre vue, c'est-à-dire la partie graphique de notre application en cours de construction dans « Interface Builder », comme nous l'avons déjà fait dans la leçon précédente (voir la figure 1).

Le composant graphique que nous voulons ajouter se nomme « **Round Rect Button** », pour l'utiliser sélectionnons-le en cliquant dessus et en le faisant glisser sur la vue.

Une fois que nous l'avons placé à l'endroit voulu, faisons un double clic pour insérer du texte à l'intérieur. Tapez le texte « **Cliquez ici** » et appuyez sur « **Enter** » pour confirmer.

Pour envoyer l'action au « controller », nous devons créer notre première méthode, qui sera ensuite reliée à la vue grâce à « Interface Builder ».

Nous devons donc ajouter une action (méthode) dans le fichier « **HelloWorldViewController.h** », comme vous pouvez le voir dans le listing 1.

Nous avons déjà vu que le symbole « **-** » déclare une **méthode d'instance et non une classe**. Entre parenthèses, nous trouvons l'équivalent de « **IBOutlet** » en ce qui concerne les actions, à savoir « **IBAction** ».

Listing 2 : implémentation de la méthode « cliquezIci »

```
-(void) cliquezIci: (id)sender{
        self.label.text = @"L'utilisateur a cliqué";
}
```

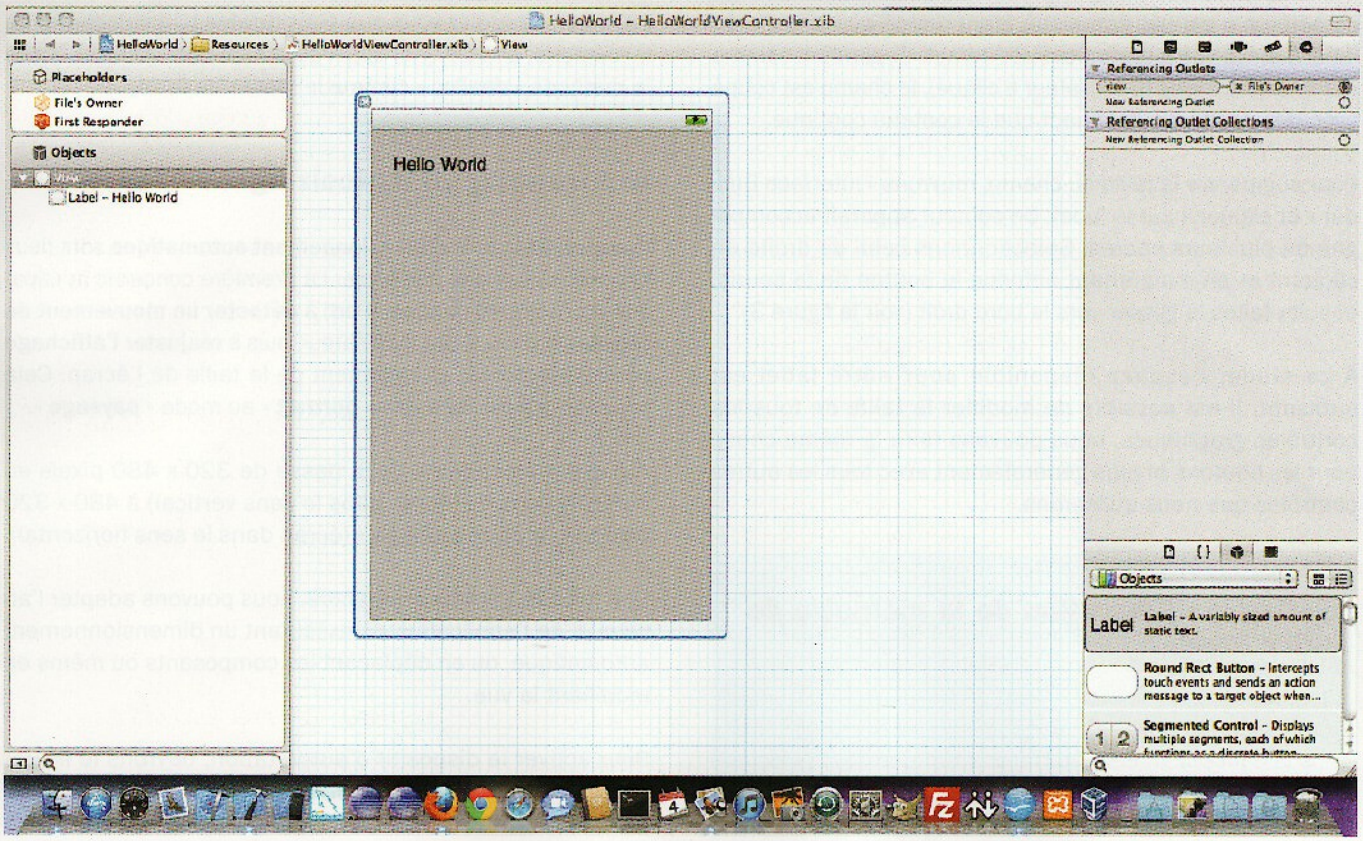
Bien que d'un point de vue programmation cela n'ait pas de sens, cela permet de générer les informations nécessaires dans « Interface Builder ». Comme nous le verrons dans la mise en œuvre de la **classe, le type de retour** sera « **void** » et non pas « IBAction » comme nous l'attendions.

Après cela, nommons la méthode « **cliquezIci** » et pour les paramètres laissez les deux points. En ce qui concerne les paramètres, nous devons insister sur la présence du mot-clé « **id** » car le type n'est pas défini, mais sera déterminé lors de l'exécution.

Nous avons vu qu'en ObjectiveC, nous pouvons ne pas avoir de type défini dans certaines circonstances. Notez que « **Id** » est suivi du nom de la variable, qui dans ce cas est la variable « **sender** ».

Jusqu'à présent, nous avons ajouté notre méthode ou « prototype », nous devons maintenant la mettre en œuvre. Pour cela ouvrons le fichier « **.m** » correspondant et ajoutons notre implémentation, comme vous pouvez le voir au Listing 2.

Figure 2 : sélection du type d'action.



Nous avons déclaré le type de retour en tant que « void », la marque substitutive « IBAction » introduite dans le fichier « **.h** » n'existe pas et sert uniquement à « Interface Builder ».

NB : une marque substitutive (ou placeholder) est un terme qui tient la place d'un contenu inconnu ou non identifié. En informatique, les variables libres et les variables liées sont des marques substitutives des variables métasyntaxiques. Cette dernière est une variable générique, l'utilisation des variables métasyntaxiques (comme toto, tata, etc.) permet de libérer le programmeur de la recherche d'un nom de variable logique adéquat au sujet étudié.

Maintenant, nous devons **lier le composant graphique** introduit dans « Interface Builder » avec le code. Ouvrons la vue « **HelloWorldViewController.xib** ». Cliquons sur « **File's Owner** » et regardons la boîte « **Connections** », nous remarquons qu'une boîte « **Received Actions** » a été créée avec le nom de la méthode que nous avons développée (voir la figure 2).

Comme nous l'avons fait pour lier le label au code, de la même manière cliquons et faisons glisser un « Button ». Une boîte grise apparaît avec différentes possibilités (voir la figure 2).

Il existe plusieurs types d'interactions possibles, mais celle qui nous intéresse pour le moment est la « **Touch Up Inside** » qui signifie que nous effectuerons un pression (touch) sur le bouton. Une fois cette information sélectionnée, il se forme un lien (link) dans la boîte « Connections ».

À ce stade, nous pouvons lancer notre application, cliquons sur le bouton. Nous nous apercevons que notre chaîne de caractères n'est pas complète, étant donné que la taille du label dans la vue a une valeur par défaut. Cependant comme notre texte est long (L'utilisateur a cliqué), le champ est coupé avec des points qui indiquent que le contenu continue.

Pour augmenter la taille du champ, rouvrons « Interface Builder » et cliquons sur le label. Un contour apparaît accompagné de plusieurs cadres. Sélectionnons celui de droite en cliquant et en maintenant enfoncé le bouton de la souris, ensuite faites-le glisser vers le bord droit (voir la figure 3).

À ce stade, l'espace disponible pour notre label est suffisant. Il est possible de modifier la taille de tous les contrôles graphiques, nous pouvons faire la même chose pour les boutons et plus généralement avec tous les autres contrôles que nous utiliserons.

Listing 3 : Libération de la mémoire de notre label.

```
- (void)dealloc {
        [label release];
        [super dealloc];
}
```

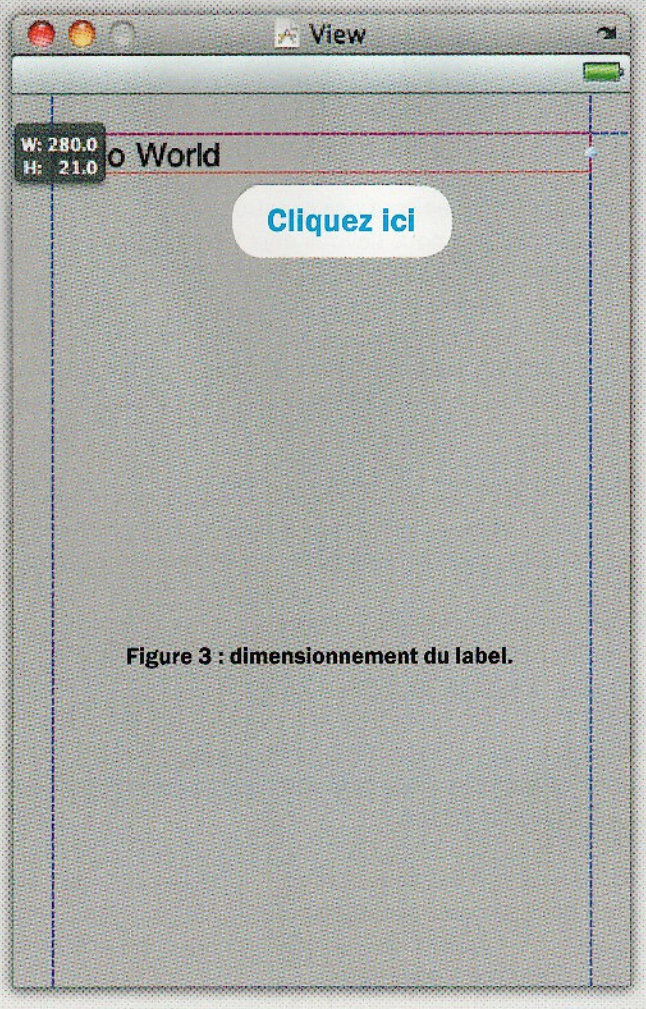


Figure 3 : dimensionnement du label.

Si nous relançons notre application et que nous cliquons, nous verrons que maintenant le texte n'est plus tronqué, il s'affiche en entier. Rappelez-vous que nous devons toujours libérer de la mémoire, en insérant l'appel de la méthode « release » dans la méthode « dealloc » comme le montre le listing 3.

Autorotating et Autosizing

L'**autorotation** et le **dimensionnement automatique** sont deux propriétés fournies par Apple. La première concerne la capacité de l'appareil (iPhone, iPad) à **détecter un mouvement de rotation** provoqué par l'utilisateur, puis à **réajuster l'affichage** pour l'adapter au changement de la taille de l'écran. Cela revient à passer du mode « **portrait** » au mode « **paysage** ».

Par exemple l'iPhone 3GS, passe de 320 x 480 pixels en mode « portrait » (écran dans le sens vertical) à 480 x 320 pixels en mode « paysage » (écran dans le sens horizontal).

Une fois l'autorotation détectée, nous pouvons adapter l'affichage de l'application en exécutant un dimensionnement automatique, ou en déplaçant les composants ou même en modifiant la vue.

Pour activer le dispositif d'autorotation, ouvrons le fichier « **HelloWorldViewController.m** » et cherchons la méthode « **shouldAutorotateToInterfaceOrientation** ». voir dans le

listing 4, nous constatons que cette méthode renvoie une valeur booléenne, cependant la configuration par défaut retourne le mode « portrait » (dernière ligne).

Si nous voulons gérer toutes les orientations possibles, nous devons tout simplement exécuter le « **return** » avec un « **YES** », comme visible dans le listing 5.

Si nous lançons maintenant notre application via le menu du simulateur « Hardware » et en cliquant sur « Rotation à gauche », nous voyons que l'affichage est automatiquement reconfiguré pour la nouvelle orientation (voir la figure 4).

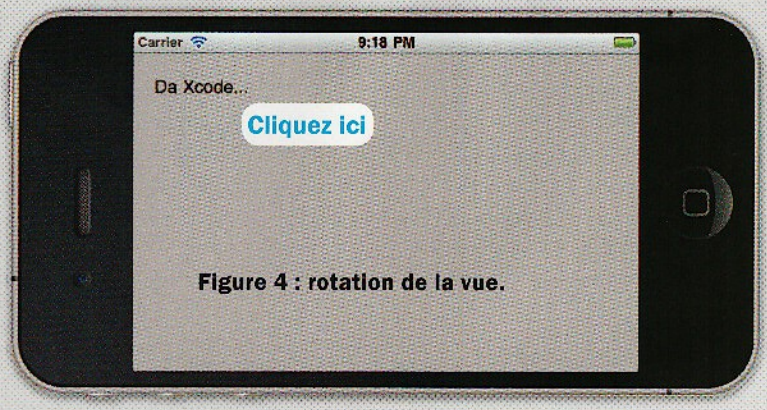


Figure 4 : rotation de la vue.

Si nous voulons seulement gérer des **vues horizontales**, nous devons nous servir de la méthode « **shouldAutorotateToInterfaceOrientation** » et utiliser la constante « **UIInterfaceOrientationLandscapeLeft** » à la place de « Portrait » (voir le listing 6).

Lorsque nous orientons l'appareil, nous devons prêter attention à l'affichage, car la taille de l'écran change et la vue affichée pour le mode « portrait » n'est pas adaptée pour le mode « paysage ». La gestion du dimensionnement automatique est par contre possible via le menu « **View Size** » directement à partir de « **Interface Builder** » (voir la figure 5) en utilisant la boîte à droite nommée « **Autosizing** ».

L'autosizing (dimensionnement automatique) est très pratique, car il permet l'étirement automatique des composants. Cependant toutes les ressources ne peuvent pas être redimensionnées facilement.

Pour nos besoins, nous nous concentrerons principalement sur la gestion d'une seule orientation.

Listing 4 : Méthode pour le contrôle de l'orientation de l'appareil.

```
// À modifier pour permettre des orientations autres que l'orientation portrait par défaut.

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {

// Retourne « YES » pour les orientations prises en charge

return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
```

Listing 5 : Activation de l'autorotation.

```
// À modifier pour permettre des orientations autres que l'orientation portrait par défaut.

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {

// Retourne « YES » pour les orientations prises en charge

return YES;
}
```

Listing 6 : Orientation de l'appareil horizontalement.

```
// À modifier pour permettre des orientations autres que l'orientation portrait par défaut.

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {

   // Retourne « YES » pour les orientations prises en charge

   return (interfaceOrientation == UIInterfaceOrientationLandscapeLeft); // horizontal gauche
}
```

MultiViews & Controllers

Comme nous l'avons déjà mentionné, tous les composants qui peuvent être placés sur l'écran sont de type « **View** » et étendent la classe de base « **UIView** ». Puisque tous les composants étendent la classe de base « UIView », nous pouvons les gérer de la même manière que la classe « **UILabel** » (voir la leçon précédente), ils ont tous des méthodes prédéfinies et des propriétés de base.

Avec « MultiViews » il est possible d'insérer dans une vue autant d'autres vues, comme nous l'avons déjà fait avec le label dans notre « View ».

Créer de nouveaux « Controllers » et les lier aux vues

Comme pour les vues, il est possible de générer et gérer plusieurs « controller », aussi bien individuellement que plusieurs à la fois. Pour cela, nous ne devons pas créer un nouveau « controller » avec une vue associée.

Appuyons sur les touches « APPLE + N » dans le projet ouvert, afin d'ajouter un fichier de type « **UIViewController subclass** » (voir la figure 6). Attention choisissez l'option « **With XIB for user interface** ». Cliquons ensuite en bas à droite pour nommer la classe « **SecondViewController** » et continuons.

Dans notre projet, en plus de la vue « **HelloWorldViewController.xib** », une seconde est maintenant disponible avec le code d'un autre « controller ».

Pour ajouter la vue du « controller » nouvellement crée à la vue du composant de base, nous n'avons pas d'autre choix que de créer le « controller » avec la vue associée par programmation, et ajouter sa vue à la première. Reportez-vous au Listing 7.

Dans la première ligne, nous créons un nouveau « controller » de type « SecondViewController » reliant la vue à travers son fichier « XIB ». Ensuite, il faut exécuter la méthode « **alloc** » et « **InitWithNibName** » en tant que **constructeur d'objet** (« Controller » dans ce cas).

Dans la deuxième ligne, nous appelons la méthode « **addSubView** » sur la propriété « view » du « controller » sur lequel nous travaillons.

Dans la pratique, nous ajoutons une « sous-vue » dans celle existante. Comme vous pouvez l'imaginer, les dimensions sont importantes et vous devez travailler dans « Interface Builder » pour construire la deuxième interface et la lier à la vue.

Avant de passer au « TableView », nous devons souligner qu'en attribuant le « controller » de cette façon, nous ne pouvons pas le libérer dans la méthode « **dealloc** », car **il n'est pas intégré dans notre objet.** Il y a alors deux possibilités : soit le libérer immédiatement, soit l'introduire comme propriété de l'instance pour qu'il soit visible et donc le libérer d'une manière conforme.

Nous avons fait cette brève analyse des « Multiviews » car nous allons les utiliser tout de suite pour le « TableView » et les projets que nous réaliserons dans les prochains cours.

Projet pratique avec un « TableView »

Le « **TableView** » est vraiment utile et très simple à mettre en œuvre. C'est l'un des composants graphiques les plus utilisés

Listing 7 : Ajout d'une vue à une autre.

```
SecondViewController*controller = [[SecondViewController alloc] initWithNibName:@"SecondViewController" bundle:nil];
        [self.view addSubView:controller.view];
```

Figure 5 : gestion de l'autosizing

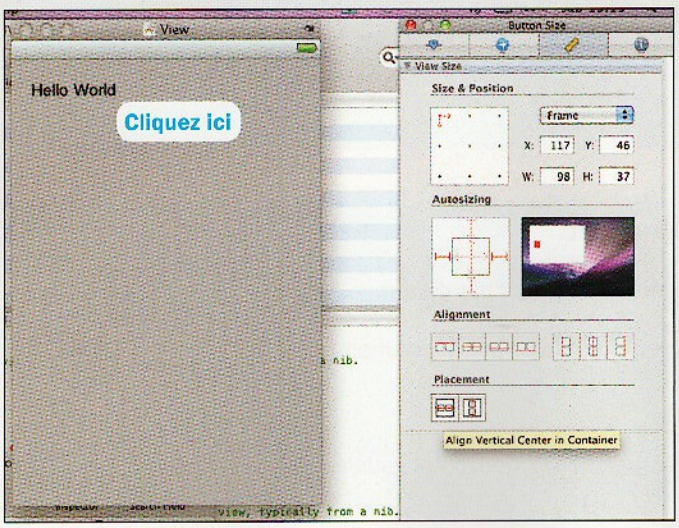
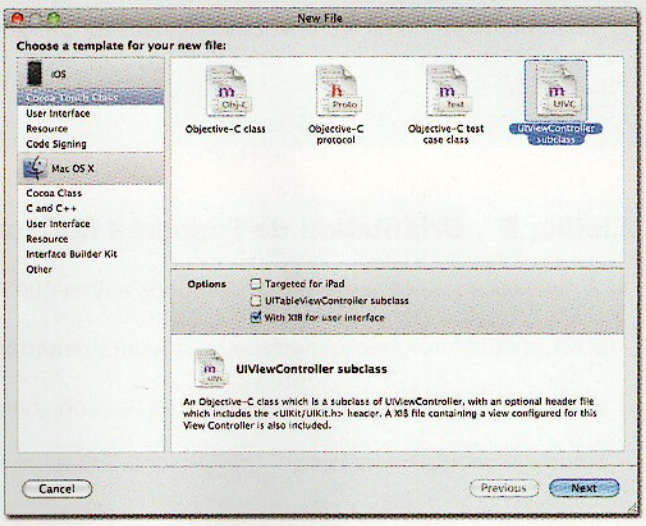


Figure 6 : création d'un nouveau « controller ».



Listing 8 : Méthodes de base de « TableView ».

```
// Configure le nombre de sections dans le « TableView ».

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return 1;
}

// Configure le nombre de lignes dans le « TableView ».

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    return 0;
}
```

Listing 9 : Déclaration de l'array dans l'en-tête et dans l'implémentation.

```
//RootViewController.h

#import <UIKit/UIKit.h>

@interface RootViewController : UITableViewController {
        NSMutableArray*donnees;
}

@property (nonatomic, retain) NSMutableArray*donnees;

@end
-----------------------------------------------------------------------
//RootViewController.m

@implementation RootViewController

@synthesize donnees;

//Reste de l'implémentation

- (void)dealloc {
        [donnees release];

    [super dealloc];
}
```

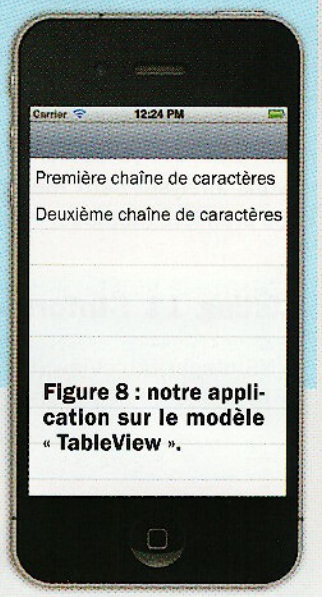


Figure 8 : notre application sur le modèle « TableView ».

Figure 7 : Exemple d'un « TableView ».

et personnalisés. La figure 7 montre la structure typique d'un tableau pour un iPhone.

Passons sur la partie supérieure de couleur bleue, qui représente la barre de navigation. Un « TableView » dans « iOS » est simplement représenté par des lignes, dont chacune correspond à un « UIView ». Nous pouvons utiliser la valeur par défaut ou la modifier à volonté.

Pour créer notre nouveau projet, nous devons fermer le précédent et en démarrer un nouveau. Indiquons que nous voulons créer un projet de type « **Navigation Based Application** » et nommons-le « **TableViewExample** ».

Une fois le projet créé, lançons le simulateur pour obtenir le résultat de la figure 8.

Commençons à colorer le code pour le rendre compréhensible, le fichier qui nous intéresse se nomme « **RootViewController** ».

Comme son nom l'indique, il s'agit du « controller » racine, c'est-à-dire le premier. Nous voyons que cet objet particulier étend la classe de base « **UITableViewController** » qui est

Listing 10 : Ajout d'éléments dans l'array.

```
- (void)viewDidLoad {
    [super viewDidLoad];

    // Décommentez (enlevez les //) la ligne suivante pour afficher un bouton « Edit » (modifier) dans la barre de navigation pour ce contrôleur de vue (view controller).

 // self.navigationItem.rightBarButtonItem = self.editButtonItem;

        //Allocation de l'array

        donnees = [[NSMutableArray alloc] init];

        //Création de la chaîne

        NSString *chaine = @"Première chaîne";

        //Ajout de la chaîne à l'array

        [donnees addObject: chaine];

        // Création de la chaîne
        NSString * chaine 2 = @"Deuxième chaîne";

        // Ajout de la chaîne à l'array
        [donnees addObject: chaine 2];
}
```

une **extension** de celle utilisée dans les projets précédents de type « **UIViewController** ». En effet, grâce à cette classe de base, nous introduisons les fonctionnalités qui permettent d'étendre un « controller » normal.

Avant d'introduire les données à l'intérieur de la vue, nous devons analyser deux méthodes présentes dans le Listing 8. La première méthode, « **numberOfSectionsInTableView** », **retourne un entier** qui représente le **nombre de sections** contenues dans notre tableau.

En effet, nous pouvons diviser le tableau en sections et, pour chacune d'elles, avoir des données spécifiques. Dans cet exemple, il y aura toujours une seule section, mais sachez que vous pouvez diviser le tableau en plusieurs parties (sections).

Listing 11 : Informe le « controller » que des données sont disponibles.

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    return [donnees count];
}
```

Listing 12 : Méthode de visualisation des cellules.

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {

    static NSString *CellIdentifier = @"Cell";

    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier] autorelease];
    }

        // Configure la cellule.

    return cell;
}
```

Listing 13 : Configuration de la cellule pour l'affichage.

```
// Configure la cellule.

NSString *chaine = [donnees objectAtIndex:indexPath.row];
cell.textLabel.text = chaine;
return cell;
```

Listing 14 : Exemple explicite d'une instance NSString.

```
NSString * chaine = (NSString *)[donnees objectAtIndex:indexPath.row];
cell.textLabel.text = chaine;
return cell;
```

La deuxième méthode, « **numberOfRowsInSection** », indique le **nombre de lignes** dans une section spécifique. Le paramètre section, qui est de type entier, indique que « iOS » traite la section spécifiée. Nous devons donc retourner le nombre de lignes que nous souhaitons afficher pour cette section.

Pour entrer les données, nous nous basons sur un modèle simplifié qui est de type « **NSString** ». Si nous voulons que les types soient plus complexes, nous pouvons créer une extension de « NSObject » et l'utiliser comme pour le type de base « string » (chaîne).

Nous devons d'abord créer un « **array** » (tableau) qui contiendra nos modèles de données, de sorte que dans notre fichier « **.h** », nous déclarerons un « **NSMutableArray** » et le synthétiserons dans l'implémentation de la classe. Le listing 9 contient les deux fichiers à éditer.

Rappelez-vous que nous devons libérer de la mémoire, cependant nous n'avons pas besoin de libérer chaque objet dans l'« array » (tableau), car « NSMutableArray » le fera pour nous.

Maintenant que nous avons déclaré les données, nous pouvons allouer la mémoire nécessaire à l'« array » et introduire des chaînes afin de les visualiser dans l'application. Dans la méthode « **viewDidLoad** » nous ajoutons le code nécessaire, comme nous pouvons le voir dans le Listing 10.

À ce stade, nous avons ajouté des données, nous devons maintenant indiquer au « controller » que nous voulons les visualiser dans le tableau.

Dans la méthode « **numberOfRowsInSection** », nous demandons au composant « **NSMutableArray** » le **nombre d'éléments** et nous **retournons ce comptage** dans la méthode « **numberOfRowsInSection** », comme nous pouvons le voir dans le Listing 11.

Enfin il ne reste plus qu'à informer la vue qu'il y a des données à visualiser dans les cellules. Pour cela, il existe une méthode fondamentale appelée « **cellForRowAtIndexPath** ». Avant d'écrire le code, nous devons savoir que le « **TableView** » est **divisé en cellules** (Cells) qui **contiennent les données** à afficher.

Pour des raisons de réutilisation et de gestion de la mémoire, elles ne sont pas toujours disponibles. Le « controller » « **UITableViewController** » les met dans une file d'attente, en évitant de toutes les charger dans la mémoire. Seules les cellules à visualiser par l'utilisateur sont chargées, évitant ainsi des problèmes de mémoire.

Dans le listing 12, examinons la méthode qui met à disposition les données. Comme nous pouvons le voir, cette méthode a deux paramètres. Le premier correspond à la vue associée au « controller » « **UITableView** » et le second est un objet appelé « **NSIndexPath** ». Cette structure englobe toute information relative à la cellule en cours de configuration, la section à laquelle elle appartient, la ligne que nous devons configurer, ainsi que d'autres informations.

Dans les premières lignes, la base de la cellule est configurée ainsi que son extraction à partir d'une file d'attente. Si elle existe, elle est directement utilisée, sinon elle est créée.

Concernant le commentaire « // Configure la cellule » du listing 13, nous avons besoin de seulement 3 lignes pour visualiser les données de la cellule indiquée dans la propriété « row » de l'objet « IndexPath ».

Pour construire la cellule, nous recevons l'objet chaîne du conteneur de type « **NSMutableArray** » via la méthode « **objectAtIndex** » et l'affectons à la variable locale appelée « **chaine** ».

Ensuite nous configurons la cellule en assignant cette référence à la propriété « **text** » de ladite cellule. À ce stade, lançons notre application, les cellules affichent le texte que nous demandons, comme vous pouvez le voir en figure 8.

Avant de terminer, soulignons que lorsque nous extrayons l'objet de l'« array », nous devons exécuter la classe « NSString », comme visible dans le listing 14.

Nous arrivons au terme de cette leçon, dans le prochain cours nous réutiliserons cette application en intégrant une « Navigation Bar » (barre de navigation) qui nous permettra de naviguer dans le contenu de notre application, à chaque fois que nous cliquerons sur une cellule. ■

Platine de développement pour PIC18F8XXX

Troisième partie

Dans cette dernière partie consacrée à la platine de développement pour PIC18F8XXX, nous allons analyser un programme de test qui lit l'état des boutons et affiche sur l'écran LCD l'identifiant numérique correspondant. Nous étudierons aussi un émulateur de terminal grâce auquel nous pourrons déboguer les programmes en cours de développement.

de Vincenzo Ligorio

Vous disposez maintenant de la platine de développement, dont la partie matérielle a été décrite dans le numéro 136, et de toutes les routines qui vous permettent de développer vos applications (les plus importantes ont été étudiées dans le précédent numéro 137).

Vous pouvez commencer à écrire des programmes simples tel que celui que nous vous proposons dans ces pages, il représente un exemple didactique précieux.

Il s'agit d'un programme relativement simple qui exécute cycliquement 4 opérations à savoir : surveillance de l'UART pour identifier toute commande provenant de l'émulateur de terminal ; vérification et gestion de la commande reçue ; contrôle des boutons et enfin mise à jour de l'affichage.

Ce programme constitue réellement une application pratique, les procédures présentes dans celui-ci sont couramment utilisées dans des applications pratiques.

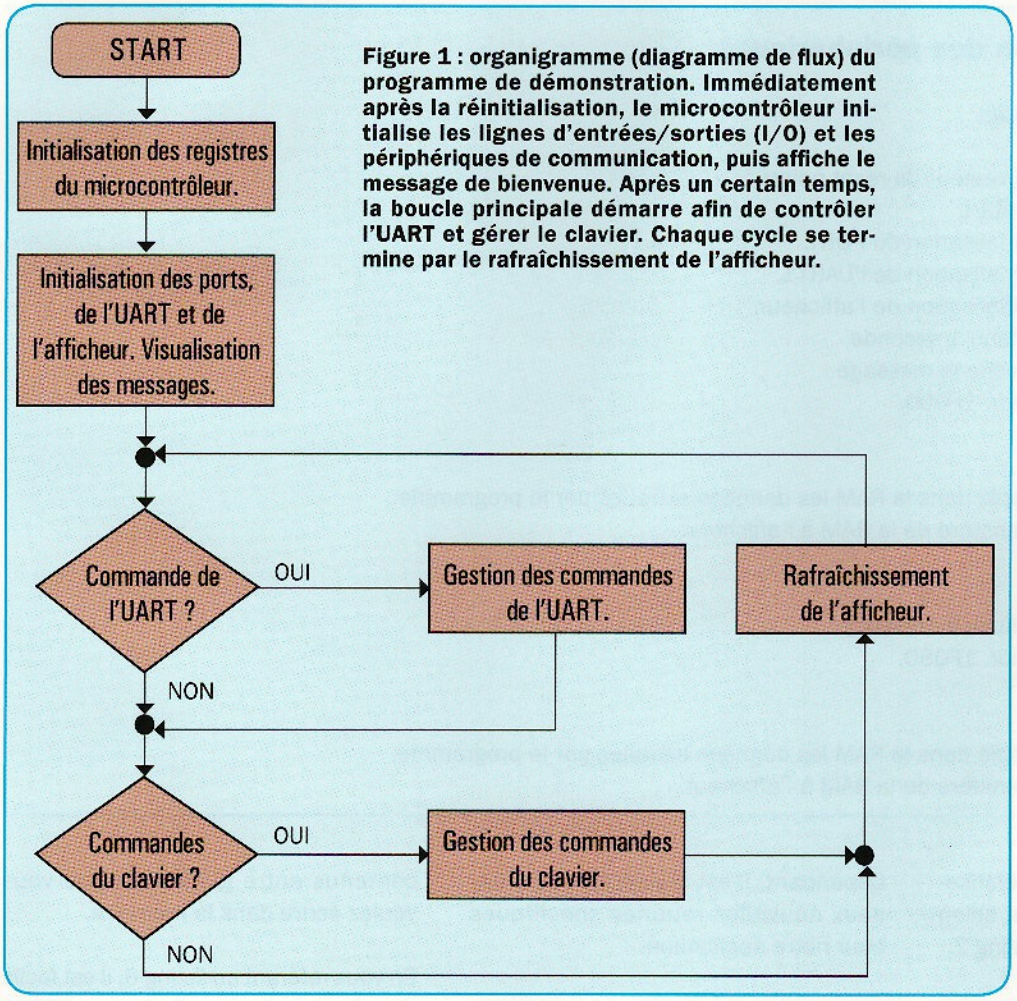


Figure 1 : organigramme (diagramme de flux) du programme de démonstration. Immédiatement après la réinitialisation, le microcontrôleur initialise les lignes d'entrées/sorties (I/O) et les périphériques de communication, puis affiche le message de bienvenue. Après un certain temps, la boucle principale démarre afin de contrôler l'UART et gérer le clavier. Chaque cycle se termine par le rafraîchissement de l'afficheur.

Le programme, appelé « **Demo-02.asm** », est téléchargeable sur notre site. Il offre de nombreuses possibilités à partir desquelles vous pouvez commencer à réaliser vos propres applications.

Le programme DEMO-02.ASM

En figure 1, vous pouvez voir l'organigramme de la boucle principale du programme, dans lequel certaines routines sont préétablies et d'autres spécialement développées (tout comme vous écrirez les vôtres pour vos applications).

Premièrement, le programme initialise certains registres spéciaux du microcontrôleur afin de configurer le type d'oscillateur, la programmation de la mémoire FLASH en basse tension (low voltage), le circuit de « watch-dog » ; etc. Tout cela est implémenté grâce aux instructions du listing 1.

Listing 1 : Initialisation des registres spéciaux

```
; Inclusion des fichiers Header.

#include  DEMO-02.inc
#include  p18f8722.inc

;Configuration des registres spéciaux.

__CONFIG  _CONFIG1H,  _OSC_HS_1H
__CONFIG  _CONFIG2L,  _BOREN_SBORDIS_2L & _BORV_43_2L & _PWRT_ON_2L
__CONFIG  _CONFIG2H,  _WDT_OFF_2H
__CONFIG  _CONFIG3L,  _MODE_MC_3L
;__CONFIG  _CONFIG4L,  _LVP_OFF_4L & _DEBUG_OFF_4L & _XINST_OFF_4L
__CONFIG  _CONFIG4L,  0x8B
__CONFIG  _CONFIG5H,  0xC0
__CONFIG  _CONFIG6H,  0xE0

org 0x0000      ; vecteur de reset:
goto start      ; pointe le label start.
org 0x0008      ; vecteur d'interruption 1:
nop             ; non utilisé.
retfie          ;
org 0x0018      ; vecteur d'interruption 2:
nop             ; non utilisé.
retfie          ;
```

Listing 2 : Initialisation des périphériques

```
; Configuration des périphériques.

start           ; Le vecteur de reset pointe
nop             ; le start.
call    portset         ; Initialisation du PORT.
call    inizUART1       ; Initialisation de l'UART1.
call    inizdisplay     ; Initialisation de l'afficheur.
call    sec1            ; Attend 1 seconde
movlw   0xF0            ; Affiche le message
movwf TBLPTRH           ; addr. 1F000.
movlw   0x00            ;
movwf TBLPTRL ;
call    copiadisplay    ; Copie dans la RAM les données extraites par le programme.
call    display         ; Transfère de la RAM à l'afficheur.
call    sec1            ;
call    sec1            ;
movlw   0xF0            ; Affiche le message
movwf TBLPTRH           ; addr. 1F050.
Movlw 0x50              ;
Movwf TBLPTRL ;
call    copiadisplay    ; Copie dans la RAM les données extraites par le programme.
call    display         ; Transfère de la RAM à l'afficheur.
```

Ensuite, nous procédons à l'initialisation des différents périphériques, selon la séquence reportée dans le listing 2.

Notez bien que, sauf pour les données à afficher qui ont été définies manuellement, le reste du listing se réduit à de simples appels (call portset, call display, etc.) des routines préétablies que nous avons analysées dans l'article précédent (voir le numéro 137 de la revue).

Cela permet de rester concentré sur la séquence logique du programme, sans devoir se soucier de l'écriture de routines spécialisées.

Dès que la phase d'initialisation des périphériques est effectuée, la **boucle principale** (**main**) démarre (voir dans le listing 3) selon la logique de l'organigramme de la figure 1.

La boucle vérifie et éventuellement traite les commandes reçues sur l'UART en provenance de l'émulateur du terminal, de même elle teste le clavier et met à jour l'affichage.

Notez que cette partie du programme établit la séquence logique des opérations à effectuer de manière cyclique en utilisant les routines préétablies.

Cependant, il est nécessaire de créer deux nouvelles routines spécifiques pour notre application.

La première est visible dans le listing 4, elle gère le clavier (keyboard) et la seconde (voir le listing 5) exécute les commandes associées (gestkeys). La première routine vérifie si une touche est pressée et identifie la touche pressée. La seconde exécute l'instruction correspondant à la touche pressée.

Dans la pratique, le véritable programme est représenté par ces deux dernières routines qui gèrent le clavier en fonction des différents besoins. Le programme (firmware) n'est toutefois pas encore complet, en effet nous devons afficher les informations provenant du clavier à l'écran ainsi que les opérations à effectuer en fonction des commandes reçues par le port série.

Le premier cas est assez simple, comme vous pouvez le voir dans le listing 6. Il suffit de définir l'adresse de départ des caractères à afficher (org 0x1F0A0), adresse qui dépend de la taille de la mémoire du PIC, et d'incrémenter à partir de cette adresse une série de déclarations « double-word » (dw) suivie chacune de deux caractères contenus entre guillemets que vous voulez écrire dans la mémoire.

En vous référant au listing 6, il est facile de comprendre que l'adresse 1F0A0 contient un espace, que l'adresse 1F0A1 contient un autre espace, que l'adresse 1F0A2 contient un « **P** » majuscule, que l'adresse 1F0A3 contient un « **r** » minuscule et ainsi de suite jusqu'à ce que toute la mémoire soit occupée.

Cependant, dans le second cas où les opérations effectuées sont fonction des commandes reçues, il est nécessaire d'adapter la routine « **incmdUART** ». En effet celle-ci doit reconnaître chaque commande reçue des routines préétablies et des deux routines que nous avons ajoutées. Or les actions ne peuvent être effectuées que si le code existe.

Reportez-vous au listing 7, la routine « **incmdUART** », après avoir vérifié que les caractères reçus du port série ne sont pas affectés par des erreurs ou des délais d'attente (timeout), appelle la routine « **gestcmdUART** » dont la tâche est d'identifier la commande reçue et d'exécuter les instructions correspondantes.

Listing 3 : Boucle principale

```
;*** Programme principal ***

main                    ; Début de la boucle principale (main).
 nop                    ;
 call   indUART         ; Vérifie si l'UART reçoit
 movlw  0x00            ; une commande.
 cpfseq RA              ; Si oui, alors
 call   incmdUART       ; gestion de la commande
 nop                    ;
 call   keyboard        ; Vérifie si une touche est pressée
 movlw  0x00            ;
 cpfseq KEY             ; Si oui (KEY <> 00), alors
 call   gestkeys        ; gestion de la touche.
 nop                    ;
 call   display         ; Rafraîchissement de l'afficheur.
 goto   main            ; Répète la boucle « main ».

;*** Fin du programme principal ***
```

Listing 4 : Contrôle des touches pressées

```
;*** Routine de contrôle des touches ***
; La routine renvoie la variable KEY
; KEY = 00 si aucune touche est pressée ;
; KEY = 01 à 08 en fonction de la touche pressée.

keyboard                ; Nom de la routine.

clrf    KEY             ; Par défaut KEY = 00 ; aucune touche pressée.
testkeys1               ; Teste la 1ère touche
 btfsc  PORTB,0         ; Touche 1 pressée ?
 goto   testkeys2       ; Non, alors vérifie la touche 2.
 movlw  0x01            ; Oui, écrit 01
 movwf  KEY             ; dans la variable KEY
 goto   testkeysend     ; et sort de la sous-routine.
testkeys2               ; Teste la 2ème touche.
 btfsc  PORTB,1         ; Touche 2 pressée ?
 goto   testkeys3       ; Non, alors vérifie la touche 3.
 movlw  0x02            ; Oui, écrit 02
 movwf  KEY             ; dans la variable KEY
 goto   testkeysend     ; et sort de la sous-routine.
testkeys3               ; Teste la 3ème touche.
...                     ; Le code est identique
...                     ; pour les 5 autres touches
...                     ;
testkeysend             ; Test terminé.
 return                 ; Sort de la routine.

;*** Fin de la routine de contrôle des touches ***
```

Par exemple, la réception des commandes « **A** » et « **a** » invoquent l'appel des routines prédéfinies « **util1a** » et « **util1b** ».

La première lit **16 octets** dans la RAM et les envoie sur le port série. La seconde calcule l'**adresse d'un emplacement** mémoire.

Il existe plusieurs commandes prédéfinies, mais la plupart d'entre elles ne sont pas associées à des routines, comme par exemple les commandes

Listing 5 : Gestion des touches pressées

```
;*** Routine de gestion des commandes des touches ***

gestkeys                    ; Début de la routine.

gestkeys1                   ; Instructions relatives à la touche 1.
  movlw 0x01                ;
  cpfseq KEY                ; Si KEY = 01 gestion de la touche 1
  goto   gestkeys2          ; sinon vérifie s'il doit gérer la touche 2.
  movlw 0xF0                ; sélectionne les caractères correspondants
  movwf TBLPTRH             ; contenues dans la mémoire FLASH à l'adresse 1F0A0.
  movlw 0xA0                ;
  movwf TBLPTRL                     ;
  call   copiadisplay       ; Affiche les caractères sélectionnés
  movff  KEY,RB             ;
  bsf    RB,4               ;
  bsf    RB,5               ;
  movff  RB,0x0CC           ;
  goto gestkeysend          ; Termine la procédure.

gestkeys2                           ; Instructions relative à la touche 2.
  movlw 0x02                ;
  cpfseq KEY                ; Si KEY = 02 gestion de la touche 2
  goto   gestkeys3          ; sinon vérifie s'il doit gérer la touche 3.
  movlw 0xF1                ; sélectionne les caractères correspondants
  movwf TBLPTRH             ; contenues dans la mémoire FLASH à l'adresse 1F100.
  movlw 0x00                ;
  movwf TBLPTRL             ;
  call   copiadisplay       ; Affiche les caractères sélectionnés
  movff  KEY,RB             ;
  bsf    RB,4               ;
  bsf    RB,5               ;
  movff  RB,0x0CC           ;
  goto gestkeysend          ; Termine la procédure.
gestkeys3                   ; Instructions relative à la touche 3
  ...                       ; code identique pour
  ...                       ; toutes les autres touches  ...          ;
gestkeysend                 ; fin de la routine
  return

;*** Fin de la routine de gestion des commandes des touches ***
```

« **E** » et « **e** » qui appellent les routines « **macommande_E** » et « **macommande_e** ». Elles doivent être écrites par le développeur du programme (c'est-à-dire par vous).

Dans tous les cas, insérer du code pour la reconnaissance d'une commande est assez simple. Il suffit de **l'ajouter après la ligne de comparaison ayant un résultat négatif** (qui provoque un saut vers la vérification de la commande suivante, goto gestcmd_a, goto gestcmd_B, goto gestcmd_b, goto gestcmd_c, etc.) pour **appeler votre routine** (call macommande_e, call macommande_E, etc.).

À ce stade, l'analyse du programme « **Demo-02.asm** » est terminée. Vous pouvez l'ouvrir dans l'environnement de développement **MPLAB**, le compiler pour créer le fichier exécutable (.hex) et programmer le microcontrôleur via le connecteur ICSP, puis redémarrer la carte pour la faire fonctionner.

Le programme démarre immédiatement, l'écran affiche deux messages espacés de quelques secondes l'un de l'autre. Appuyez sur l'un des 8 boutons et vérifiez chaque fois que l'identifiant numérique du bouton correspondant apparaisse sur l'afficheur.

En appuyant sur les boutons 1 et 2 (en bas à gauche), vous verrez apparaître sur l'afficheur les messages suivants : « Bonjour E.L.M » et « Touche pressée ».

Connectez maintenant l'interface série du PC sur lequel vous avez installé le programme d'émulation de terminal

Listing 6 : Gestion des touches pressées

```
; 00000000011111111112
; 12345678901234567890
; xxxxxxxxxxxxxxxxxxxx
org 0x1F0A0             ;1 Programme Demo 2
                        ;2 Touche ? (ce message indique qu'il faut
                          appuyer sur une touche)
                        ;3 Bonjour E.L.M
                        ;4 Touche pressée: -
 dw   " "       ; 1 ère ligne
 dw   "Pr"      ;
 dw   "og"      ;
 dw   "ra"      ;
 dw   "mm"      ;
 dw   "e "      ;
 dw   "De"      ;
 dw   "mo"      ;
 dw   " 2"      ;
 dw   " "       ;
 dw   " "       ; 2ème ligne
 dw   " To "    ;
 dw   "uc"      ;
 dw   "he"      ;
 dw   "? "      ;
 dw   " "       ;
 dw   " "       ;
 dw   " "       ;
 dw   " "       ;
 dw   " "       ;
 dw   "Bo"      ; 3ème ligne
 dw   "nj"      ;
 dw   "ou"      ;
 dw   "'r "     ;
 dw   "E. "     ;
 dw   "L. "     ;
 dw   "M "      ;
 dw   " "       ;
 dw   " "       ;
 dw   " "       ;
 dw   "To"      ; 4ème ligne
 dw   "uc"      ;
 dw   "he "     ;
 dw   " "       ;
 dw   "pr "     ;
 dw   "es"      ;
 dw   "sé"      ;
 dw   "e : "    ;
 dw   "- "      ;
 dw   " "       ;
```

au port COM1 de la platine de développement (connecteurs DB9 sur la gauche, le COM1 est celui du bas). Vous pouvez commencer à interagir avec la carte.

Le programme Terminal

Il s'agit d'un programme essentiel, simple et pratique. Il se compose d'une seule fenêtre qui s'affiche en cours d'exécution (voir la figure 2). Elle est divisée en 3 zones :

- la 1ère zone correspond à la partie supérieure qui comporte un menu déroulant permettant de sélectionner le port COM relié à la platine de développement ainsi que la vitesse de communication (la valeur par défaut est de 9600 bauds) ;
- la 2ème zone correspond à la partie gauche de la fenêtre, elle est divisée en 2 zones de texte qui permettent de visualiser les commandes envoyées à la platine de développement (TX) et celles reçues (RX) de la platine de développement. La zone de texte « TX » est entièrement éditable, et après avoir pris connaissance des commandes, vous pourrez les taper au clavier dans cette zone de texte et les envoyer directement à la carte. **Chaque commande envoyée génère une réponse de la platine** de développement, la réponse apparaît en lecture seule dans la zone de texte « RX » (cette zone est grisée). Si la commande est valide, la platine de développement **répond par le prompt « ! »** (point d'exclamation), confirmant ainsi la validité de la commande et l'exécution des instructions. L'absence de réponse est le signe d'un dysfonctionnement (un « bug » dans le logiciel) ou une commande erronée. Les zones « TX » et « RX » comportent chacune un bouton « **Effacer** » (**Clear**) qui permet d'effacer le contenu de chaque zone ;
- la 3ème zone correspond à la partie droite de la fenêtre, elle comporte les contrôles nécessaires pour interagir et debugger la platine de développement. La section « **CMDs** » dispose de 6 boutons de présélection à l'aide desquels il est possible d'interagir avec le périphérique « horloge » de la carte (si elle est utilisée par l'application, mais cela n'est pas notre cas), de 11 zones de texte chacune associée à un bouton d'envoi identifié par le symbole « **==>** » dans lesquelles vous pouvez entrer les commandes de débogage que vous utilisez le plus fréquemment. La partie la plus à droite de la fenêtre comporte 3 sections : « **CMD** », qui permet

Listing 7 : Gestion des commandes reçues

```
;*** Routine de réception des commandes de l'UART.

incmdUART               ; Début de la routine.
  call   carechoUART    ; Transmet le « ! » comme confirmation.
  call   rxUART         ; Prélève la chaîne de commandes.
  movlw  0X00           ; Vérifie si CARRX vaut 00
  cpfseq CARRX          ; c'est-à-dire erreur ou timeout de réception.
  call   gestcmdUART    ; CARRX <> 00: gestion de la commande.
  return                ; CARRX = 00: fin de la routine.

;*** Fin de la routine de réception des commandes de l'UART.

;*** Gestion des commandes de l'UART ***

; Routine de gestion des différentes commandes de l'UART.
; Cette routine est activée lors de la réception de l'adresse de la carte
; En fonction des diverses commandes, active la sous routine appropriée.

gestcmdUART             ; Routine qui appelle « incmdUART ».
  movff  0x100,RA       ; Copie le code de la commande dans le registre RA.
gestcmd_A               ; Vérifie la commande « A ».
  movlw  0x41           ;
  cpfseq RA             ; Commande = 41 (A) ?
  goto   gestcmd_a      ; Non, vérifie la commande suivante.
  call   util1a         ; Oui, appel de la routine qui place 16 octets
                        ; dans la mémoire RAM.
  goto   gestcmd_fine   ; Termine la gestion de la commande.
gestcmd_a               ; Vérifie la commande « a ».
  movlw  0x61           ;
  cpfseq RA             ; Commande = 61 (a) ?
  goto   gestcmd_B      ; Non, vérifie la commande suivante.
  call   util1b         ; Oui, appel de la routine qui calcule l'adresse des 3
                        ; caractères ASCII, transforme les données ASCII
                        ; en Hex et les écrit dans la RAM.
  goto   gestcmd_fine   ; Termine la gestion de la commande.
...                     ; Code pour la reconnaissance d'autres commandes.
...                     ;
gestcmd_E               ; Vérifie la commande « E ».
  movlw  0x45           ;
  cpfseq RA             ; Commande = 45 (E) ?
  goto   gestcmd_e      ; Non, vérifie la commande suivante.
  call   macommande_E   ; Appel de la routine propriétaire
                        ; de la gestion de la commande.
  goto   gestcmd_fine   ; Termine la gestion de la commande.
gestcmd_e               ; Vérifie la commande « e ».
  movlw  0x65           ;
  cpfseq RA             ; Commande = 65 (e) ?
  goto   gestcmd_F      ; Non, vérifie la commande suivante.
  call   macommande _e  ; Appel de la routine propriétaire
                        ; de la gestion de la commande.
  goto   gestcmd_fine   ; Termine la gestion de la commande.
...                     ; Code pour la reconnaissance d'autres commandes.
...                     ;
gestcmd_fine            ; Fin de la gestion des commandes.
  return                ; Sortie de la sous routine.
```

l'insertion et l'envoi d'une commande spécifique ; « **Last Char** » qui affiche le dernier caractère envoyé par la platine de développement ; « **Address** » à travers laquelle vous pouvez **sélectionner l'adresse de la carte** à déboguer.

L'envoi d'une commande s'effectue de manière très simple. Vous cliquez sur le bouton d'adresse « **Address** » (dans notre cas « **Chr$(128)** »), la carte répond par « **!** ».

Cliquez ensuite sur le bouton de la commande à envoyer.

De ce qui précède, il en résulte que pour déboguer le programme qui est en cours d'exécution sur la platine de développement, il suffit de prévoir les structures de code qui peuvent interagir avec les différentes commandes.

Les commandes prises en charge sont visibles dans le **Tableau 1.** Il est possible d'effectuer un débogage en temps réel de l'application, étant donné que les routines « incmdUART » liées au programme « Demo-02.asm » permettent de lire et modifier toute la mémoire RAM et la mémoire EEPROM ainsi qu'effectuer des « **DUMP** » (sauvegardes) de la RAM contenant toutes les **variables système**.

Comme nous l'avons évoqué précédemment, vous pouvez implémenter d'autres commandes spécifiques en modifiant par exemple, les messages de l'afficheur, ou les identifiants des touches, etc., l'envoi des commandes est très simple :

- **envoi de l'adresse** de la platine de développement (dans notre cas 128) ;
- la **platine de développement répond** avec le prompt (invite) « **!** » ;
- **envoi de la commande désirée** ;
- **la réponse** à la commande **est affichée**, le cas échéant.

Exemples pratiques simples

Supposons que vous développez un programme qui, en faisant usage d'une routine mathématique, lit et écrit des données dans la RAM et que, pour une

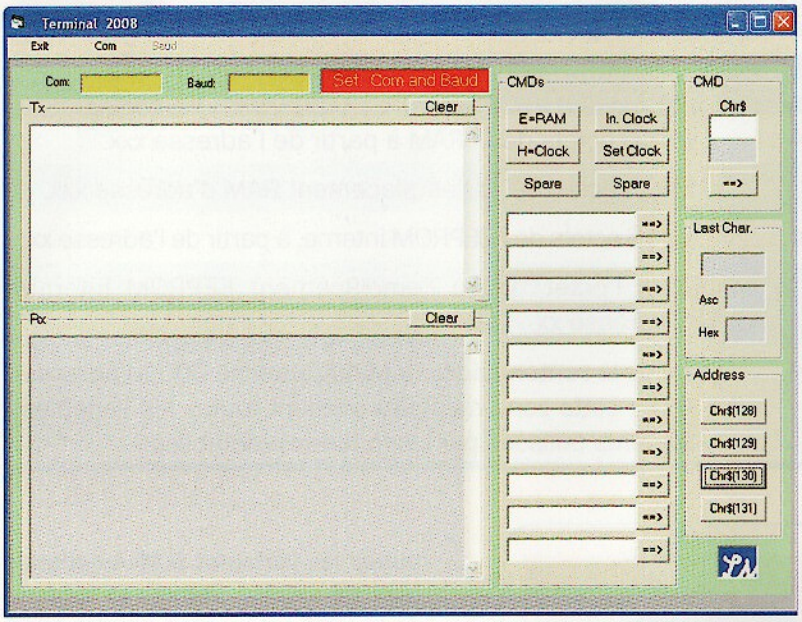


Figure 2 : l'écran principal du programme émulateur de terminal est très intuitif : en haut, vous trouvez le menu de configuration du port « COM », à gauche les sections relatives aux données reçues et envoyées ; à droite les boutons et les champs de commande grâce auxquels vous pouvez déboguer la carte.

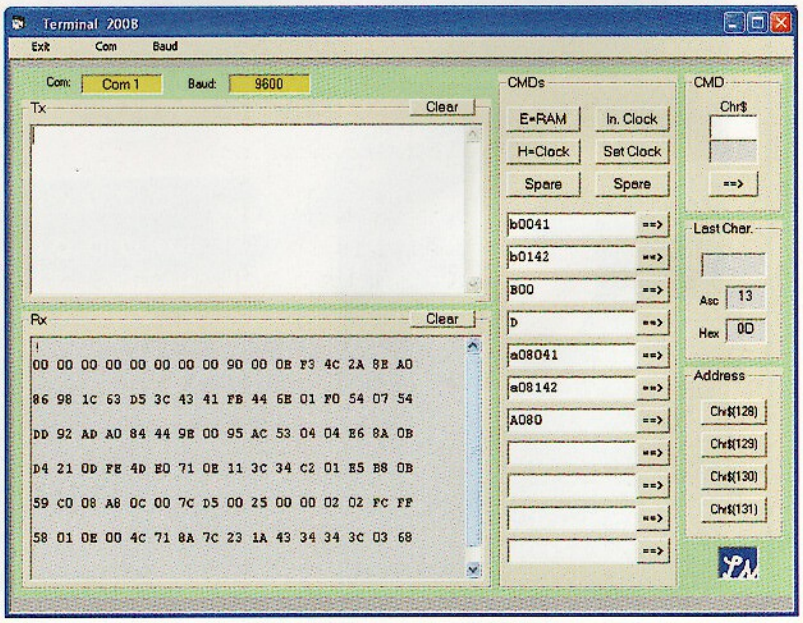
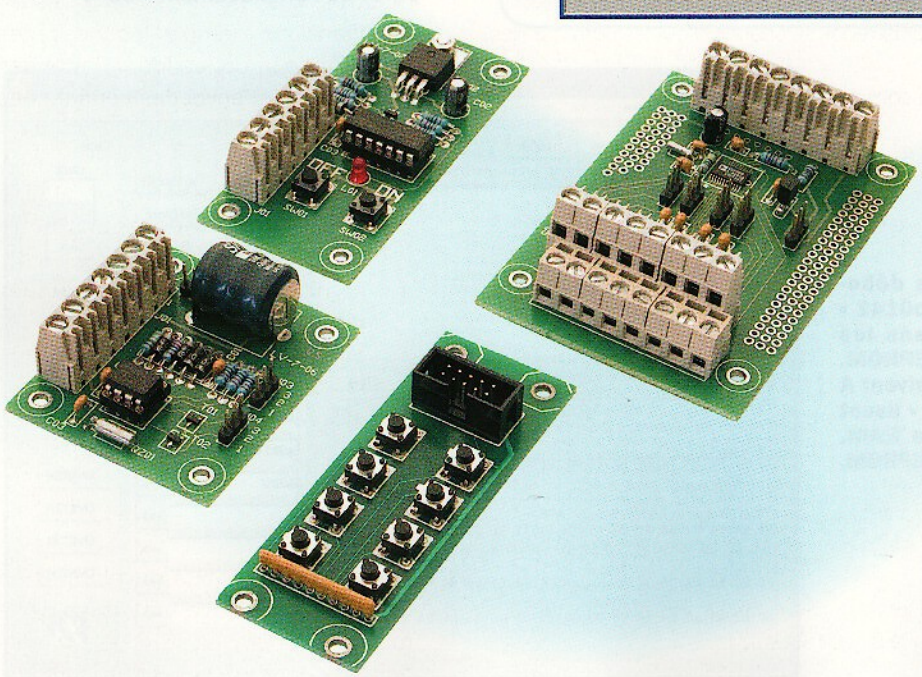


Figure 3 : Ici dans la zone de débogage, des contrôles par défaut ont été inclus dans les zones de texte. Par exemple « b0041 » écrit un « A » à l'emplacement « 00 » de l'EEPROM. Cliquez sur le bouton « Chr$(128) » dans la zone « Address » pour sélectionner l'adresse de la carte, ensuite sur le bouton associé à la commande D (sauvegarde de la mémoire EEPROM).



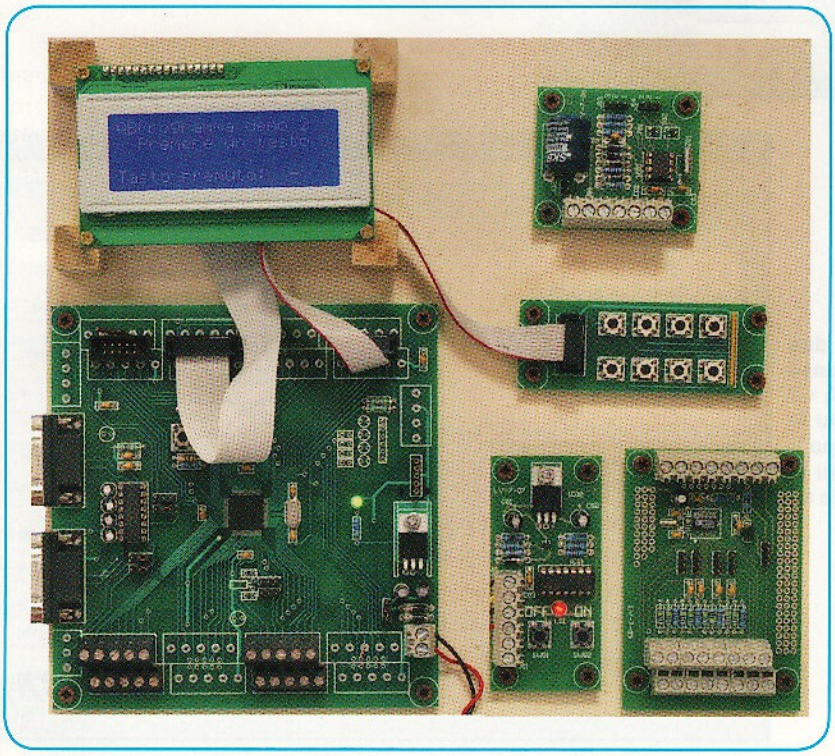
raison étrange, il y a des erreurs lors de l'écriture dans la RAM. Il est alors nécessaire d'effectuer un « **DUMP** » (lecture complète) des emplacements mémoire utilisés par les variables mathématiques. Cette opération est représentée en figure 3.

Cliquez sur le bouton d'adresse (Chr$(128)) afin de dialoguer avec la carte, celle-ci confirme la réception de la commande par l'envoi de l'invite « ! ».

Lorsque le bouton associé à la zone de texte où vous avez inséré la commande D (DUMP) est cliqué, la carte répond en envoyant sur le port série

Tableau 1 - Liste des commandes prises en charge

Commande	Syntaxe	Réception	Fonction
A	Axxx	8A BC ...24	Lit 16 octets de la RAM à partir de l'adresse xxx.
a	axxxyy	-	Ecrit l'octet "yy" à l'emplacement RAM d'adresse xxx.
B	Bxx	8A BC ...24	Lit 16 octets de l'EEPROM interne, à partir de l'adresse xx.
b	bxxyy	-	Ecrit l'octet "yy" à l'emplacement EEPROM interne d'adresse xx.
D	D	8A BC ...24	Copie le contenu de la RAM de l'adresse 000 à l'adresse 05F : cette zone mémoire contient toutes les variables système utilisées par les routines préétablies.



toutes les variables mathématiques (soit 96 octets). Comme le contenu de la RAM est lu, vous pouvez aussi le modifier (même celui de l'EEPROM).

Reportez-vous à la figure 4, dans laquelle la zone « RX » montre les différentes réponses relatives à la figure 3. Tout d'abord, les deux commandes « b0041 » et « b0142 » ont été envoyées (précédées toutes les deux de la commande d'adresse) avec lesquelles les caractères « A » et « B » ont été écrits dans les emplacements « 00 « et « 01 » de l'EEPROM.

La demoboard répond aux 2 commandes avec le prompt « ! ». Successivement, est envoyé la commande « B00 » avec laquelle les 16 caractères de l'EEPROM sont lus à partir de l'emplacement « 00 ».

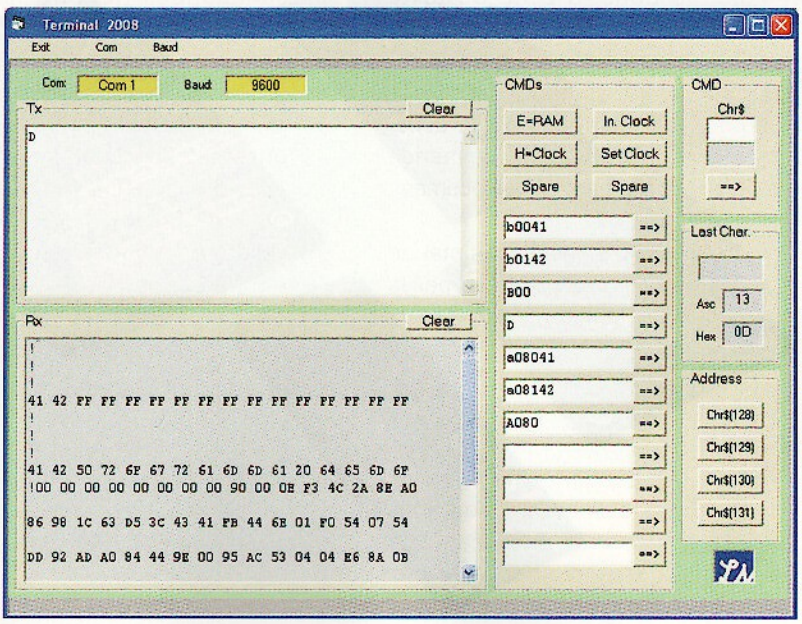


Figure 4 : ici un autre exemple de débogage. Les commandes « b0041 » et « b0142 » écrivent respectivement A et B dans les emplacements « 00 » et « 01 » de l'EEPROM. De même, « a08041 » et « a08142 » écrivent A et B sur l'afficheur. « B00 » et « A080 » lisent 16 caractères de l'EEPROM et de la RAM. La commande « D » lit l'ensemble de l'EEPROM.

Notez que les 2 premiers caractères correspondent à 41 et 42, c'est à dire « A » et « B » selon les commandes précédentes.

Ensuite les commandes « a08041 » et « a08142 » sont envoyées, avec lesquelles nous avons écrit les mêmes caractères « A » et « B » dans les emplacements « 080 » et « 081 » de la RAM (ceux qui contiennent les images des messages affichés sur l'écran).

Cette action a écrit les 2 caractères « AB » sur les deux premières cellules de la première ligne de l'afficheur.

À ce stade, faisons une autre hypothèse.

Supposons que l'application en cours de développement utilise l'écran pour afficher des messages d'information, à un moment donné des caractères étranges s'affichent alors qu'ils ne sont pas définis dans le programme, ils ne devraient pas y être.

Avec la commande « A080 », nous pouvons copier et vérifier 16 octets à la fois dans la partie de la RAM contenant les messages affichés sur l'écran.

De même, avec les commandes A0090, A0A0, etc., nous pouvons lire par groupe de 16 octets tout le contenu de la RAM dédié à l'affichage.

Comme dernière opération, effectuons de nouveau un « DUMP » des variables mathématiques, en introduisant la commande dans la zone de texte « TX ».

Rappelons que, quelle que soit la commande à envoyer, il est nécessaire d'abord d'envoyer la commande d'adresse à la platine de développement.

Si vous ne le faites pas, la commande suivante ne sera pas exécutée par la routine principale (main).

Notes concernant le programme Terminal

Selon la version de Windows dont vous disposez, il se peut que le programme Terminal ne fonctionne pas. Vous obtenez un message d'erreur indiquant que le fichier « **Mscomm32.ocx** » est manquant.

Vous devez d'abord télécharger le fichier « Mscomm32.ocx », il se trouve dans le sommaire détaillé du numéro 138 à l'onglet « Télécharger ». Vous y trouverez aussi le programme « Demo-02.asm ».

Sous **Windows XP** :

- copiez le fichier « Mscomm32.ocx » dans le répertoire C:\WINDOWS\SYSTEM32 ;
- allez dans le « **Menu démarrer** » ;
- sélectionnez « **Executer** » ;
- puis tapez « **cmd** » et validez en appuyant sur la touche « **Entrée** » ;
- une fenêtre DOS s'ouvre, tapez la commande suivante pour enregistrer l'OCX manquant : **regsvr32 c:\windows\system32\Mscomm32.ocx**
- validez en appuyant sur la touche « Entrée » ;
- Lancez le programme Terminal, vous devez obtenir la fenêtre principale indiquant que le programme fonctionne correctement.

Sous **Windows 7 32 bits** :

- copiez le fichier « Mscomm32.ocx » dans le répertoire C:\WINDOWS\SYSTEM32 ;
- allez dans le « **Menu démarrer** » ;
- sélectionnez « **Tous les programmes** » ;
- puis « **Accessoires** » ;
- faites un clic droit sur l'invite de commandes (fenêtre noire) et sélectionnez « **Exécuter en tant qu'administrateur** ».
- une fenêtre DOS s'ouvre et tapez la commande suivante pour enregistrer l'OCX manquant : **regsvr32 c:\windows\system32\Mscomm32.ocx**
- validez en appuyant sur la touche « Entrée » ;

Sous **Windows 7 64 bits** :

- copiez le fichier « Mscomm32.ocx » dans le répertoire C:\WINDOWS\sysWOW64 ;
- allez dans le « **Menu démarrer** » ;
- sélectionnez « **Tous les programmes** » ;
- puis « **Accessoires** » ;
- faites un clic droit sur l'invite de commandes (fenêtre noire) et sélectionnez « **Exécuter en tant qu'administrateur** » ;
- une fenêtre DOS s'ouvre et tapez la commande suivante pour enregistrer l'OCX manquant : **regsvr32 c:\windows\sysWOW64\Mscomm32.ocx**

Pour Windows 8 et 10 64 bits, la procédure est identique à celle de Windows 7 64 bits.

Assurez-vous aussi que votre PC dispose d'un port COM physique, les portables récents n'en disposent plus.

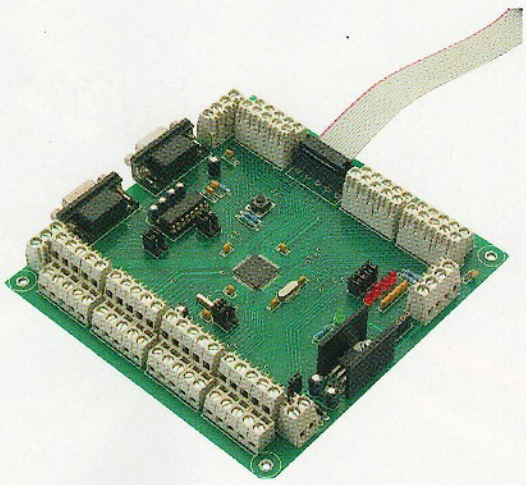
Vous obtiendrez alors un message d'erreur de type « Runtime error » indiquant que le programme Terminal s'est arrêté car il n'a pas trouvé de port COM sur l'ordinateur.

Conclusion

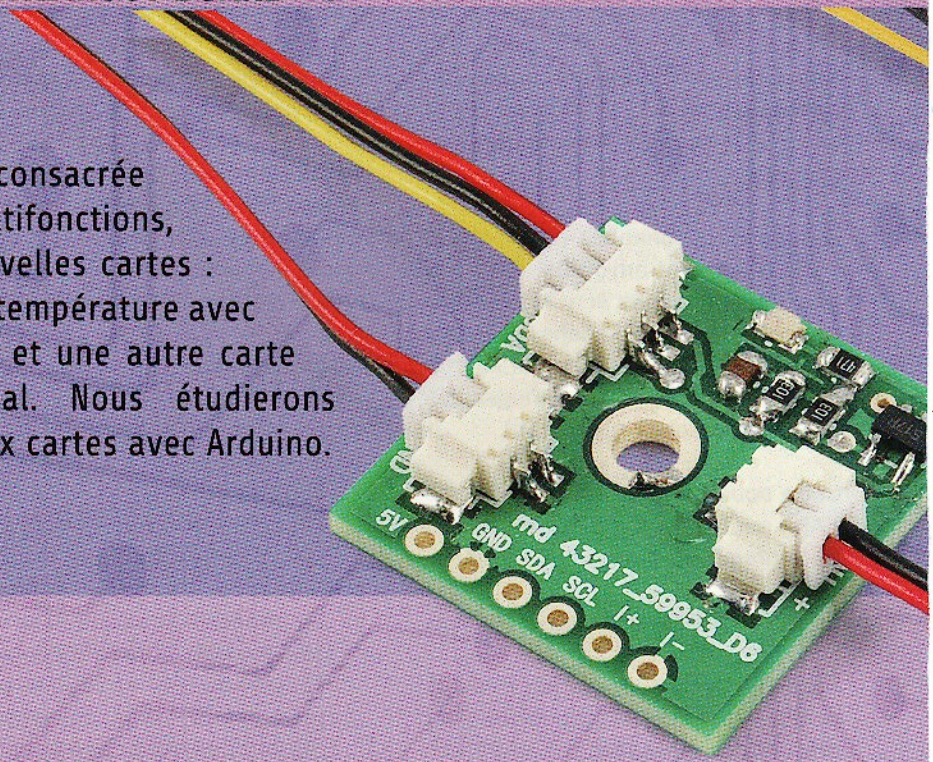
Si vous décidez d'entreprendre la réalisation de cette platine de développement pour PIC18FXXX, vous vous rendrez compte de sa polyvalence.

Si vous ne la réalisez pas, vous pourrez toujours utiliser les routines contenues dans les deux fichiers « Demo-01.asm » et « Demo-02.asm ».

Ils contiennent des routines préétablies et fonctionnelles offrant ainsi des solutions prêtes à l'emploi, vous pourrez les intégrer directement dans vos applications. ■



BREAKOUT BOARD - 4



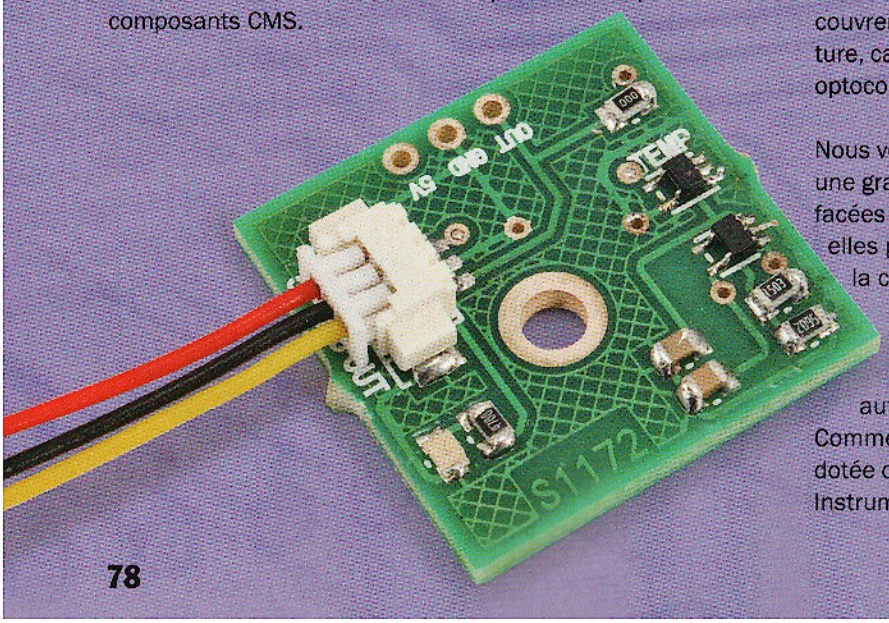
Dans cette quatrième partie consacrée aux cartes de prototypage multifonctions, nous vous proposons deux nouvelles cartes : une permettant la mesure de la température avec un signal de sortie analogique et une autre carte convertisseur analogique/digital. Nous étudierons ensuite l'interfaçage de ces deux cartes avec Arduino.

CARTES DE PROTOTYPAGE MULTIFONCTIONS

..... *de Alessandro Sottocornola*

Quatrième partie

Nous vous proposons les deux dernières cartes de cette série que nous avons commencée à décrire dans le numéro 135 d'Electronique et Loisirs Magazine. Ces cartes de prototypage ont été réalisées dans le but que tout électronicien amateur puisse « manipuler » des composants CMS.

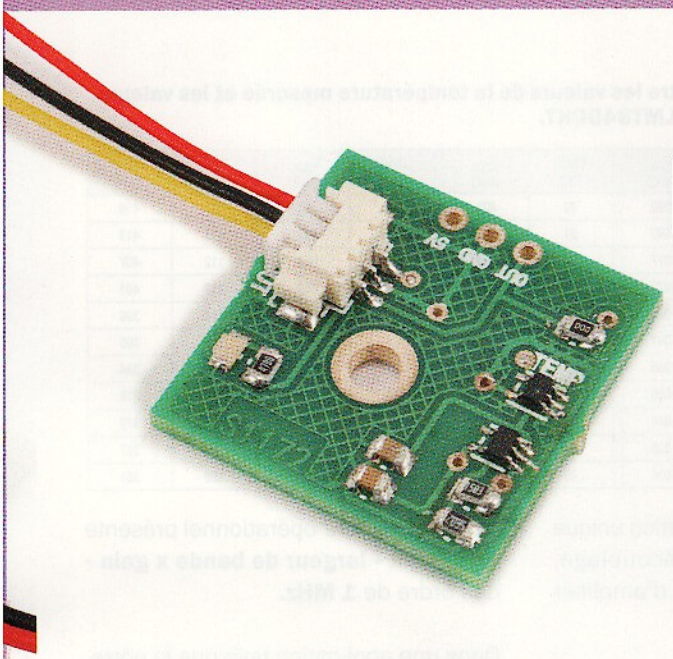


Cependant ces cartes peuvent être d'une grande utilité pour les professionnels désirant intégrer des fonctions supplémentaires dans des systèmes existants, à moindre coût.

Pour rappel les cartes présentées dans cette série d'articles couvrent les domaines les plus variés : capteurs de température, capteur d'humidité, capteur de pression, alimentations, optocoupleurs, relais et conversion digitale/analogique.

Nous vous avons donc proposé une série de cartes couvrant une grande quantité d'applications et lorsqu'elles sont interfacées avec Arduino ou une carte à base de microcontrôleur, elles permettent de tester différentes configurations avant la construction d'un circuit final.

Nous allons étudier maintenant une carte **capteur de température** avec **sortie analogique** et une autre contenant un **convertisseur analogique/digital**. Commençons par la carte capteur de température qui est dotée d'un circuit intégré « **LMT84DCKT** » fabriqué par Texas Instruments.



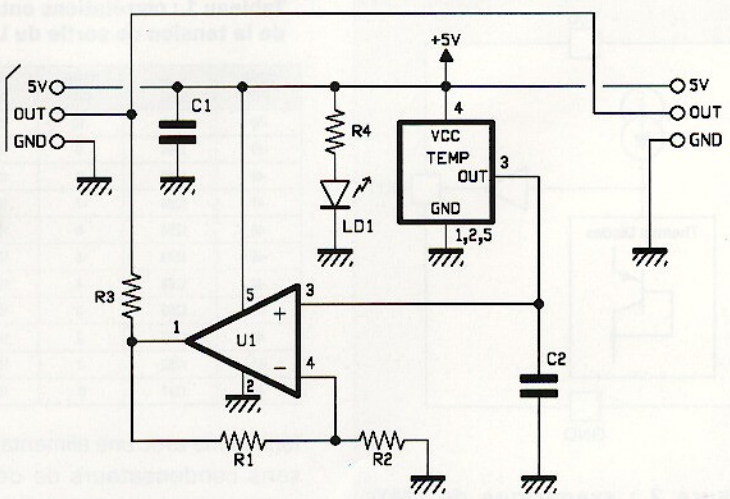


Schéma électrique de la carte capteur de température.

Carte capteur de température

En plus du capteur analogique, cette carte comporte un amplificateur opérationnel « **MCP6L01T-E/LT** » de chez Microchip, qui est utilisé pour amplifier la tension présente en sortie du capteur.

La tension de sortie de l'amplificateur est une tension continue de valeur inversement proportionnelle à la température mesurée. La tension de sortie maximale est de **1,299 V**.

Le capteur est en mesure de garantir une **corrélation linéaire entre la température et la tension de sortie** avec une précision sur la mesure de seulement 0,4 °C. Le graphique de la figure 1 montre la variation de la tension de sortie en fonction de la température.

Comme vous pouvez le constater, la variation de la tension en fonction de la température est typiquement de -5,5 mV/°C à partir d'un maximum de 1,299 mV à - 50 °C. À zéro degré centigrade, la tension délivrée en sortie est légèrement supérieure à 1 V.

Le tableau 1 montre les principales valeurs de la tension de sortie pour diverses températures.

Le capteur analogique **LMT84** utilise comme élément sensible la jonction base-émetteur d'un transistor bipolaire de type PNP polarisé directement (voir la figure 2).

Nous savons que **la tension de polarisation directe d'une jonction « PN » est fonction de sa température**, elle diminue de 2,5 mV pour chaque degré de plus. Il devient alors facile de déterminer l'évolution de la tension sur l'émetteur en fonction de la température.

La variation de la chute de tension directe est par ailleurs constante et très précise, et suit presque immédiatement la variation de la température de la jonction. Le capteur a une réponse relativement rapide lorsque la température varie.

Dans un **LMT84DCKT**, l'émetteur du transistor servant de capteur est relié à un étage tampon de sortie constitué par des **MOSFET « push-pull »** à symétrie complémentaire (canal N et à canal P), montés en source commune.

Le circuit **LMT84DCKT** peut fonctionner avec une tension d'alimentation très faible, de l'ordre de **1,5 V** et ce jusqu'à **5,5 V**, et dans une large plage de température (-50 °C à +150 °C). Il peut donc être utilisé dans le secteur industriel et dans l'automobile. Sa très faible consommation de courant sans charge reliée en sortie (seulement 5,4 µA) le rend idéal pour des applications fonctionnant sur piles ou batteries.

La sortie du capteur étant constituée par des MOSFET configurés en symétrie complémentaire, elle peut fournir des courants de l'ordre de ±50 µA sans que la tension soit altérée.

Cette configuration assure une parfaite corrélation entre la tension de sortie et la variation de température, sans que la charge ne perturbe la mesure.

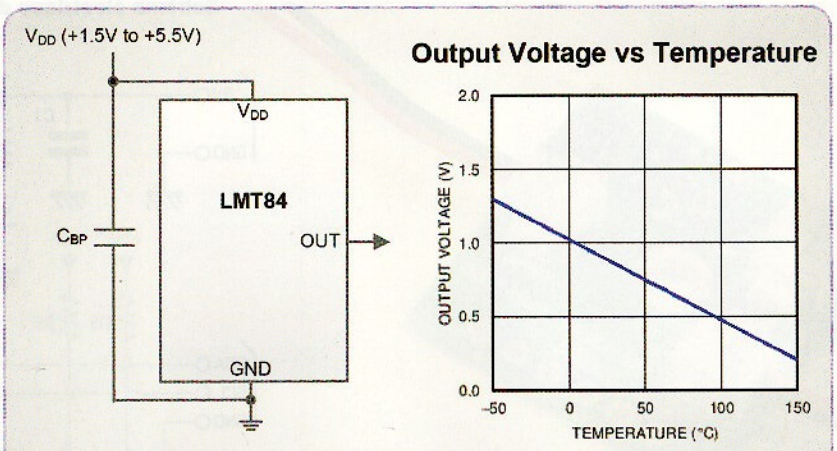


Figure 1 : schéma d'application typique et variation de la tension de sortie en fonction de la température mesurée du capteur LMT84DCKT.

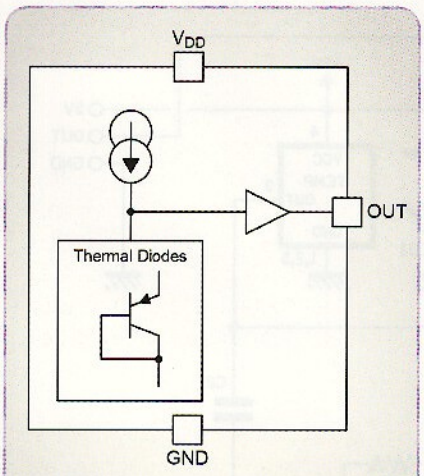


Figure 2 : synoptique du LMT-84DCKT: la partie sensible à la température est constituée par la jonction d'un transistor dont la tension directe entre la base et l'émetteur est mesurée

Le circuit intégré LMT84 est fabriqué par Texas Instruments et est disponible en plusieurs boîtiers : SC70 (SOT5) ou SOT-23-5 pour le montage en surface (CMS), TO-92 (LMT84/LMT84-Q1) et TO-126 pour le montage traversant (traditionnel). Pour notre carte, nous avons utilisé la version **SOT-23-5** à 5 broches pour un montage en surface, la référence du composant est : LMT84DCKT.

En regardant le schéma électrique de la carte capteur de température, nous constatons que la **sortie** (broche 3) du capteur est reliée à **l'entrée non-inverseuse** (+ ou broche 3) de l'amplificateur de tension **U1**, un **MCP6L01T-E/LT**. Celui-ci, en version CMS aussi, est monté dans une configuration non-inverseuse (il

Tableau 1 : corrélations entre les valeurs de la température mesurée et les valeurs de la tension de sortie du LMT84DCKT.

TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)
-50	1299	-10	1088	30	871	70	647	110	419
-49	1294	-9	1082	31	865	71	642	111	413
-48	1289	-8	1077	32	860	72	636	112	407
-47	1284	-7	1072	33	854	73	630	113	401
-46	1278	-6	1066	34	849	74	625	114	396
-45	1273	-5	1061	35	843	75	619	115	390
-44	1268	-4	1055	36	838	76	613	116	384
-43	1263	-3	1050	37	832	77	608	117	378
-42	1257	-2	1044	38	827	78	602	118	372
-41	1252	-1	1039	39	821	79	596	119	367
-40	1247	0	1034	40	816	80	591	120	361

fonctionne avec une alimentation unique sans condensateurs de découplage, puisque nous avons besoin d'amplifier une tension continue).

Avec les valeurs du diviseur de tension constitué par les résistances R1 et R2 de contre réaction, nous obtenons un **gain en tension égal à 3,857**.

Par conséquent pour une température de -50 °C nous obtenons une tension de 5,01 V en sortie de la carte, et pour une température de + 150 °C la tension de sortie sera de 0,705 V.

Afin de ne pas perturber les excellentes performances du capteur en termes de tension de fonctionnement, nous avons choisi cet amplificateur opérationnel car il est capable de fonctionner entre 1,8 V et 6 V (au repos il consomme seulement 85 µA). De plus il dispose d'une sortie de type « **rail-to-rail** », c'est-à-dire que la tension de sortie peut atteindre la tension d'alimentation (très faibles pertes).

Cet amplificateur opérationnel présente un produit « **largeur de bande x gain** » de l'ordre de **1 MHz**.

Dans une application telle que la nôtre, la valeur de ce produit n'est pas critique (nous avons des tensions continues ...), cependant pour des mesures de variations rapides de tensions, ce produit ne doit pas être négligé lors du choix des composants.

Cet amplificateur opérationnel peut fonctionner dans une plage de températures de -40 e +125°C, il peut donc être utilisé dans le milieu automobile.

Comme pour toutes les autres « breakout board », la carte capteur de température est dotée de connecteurs au pas de 1,25 mm où sont reportés le signal de sortie et l'alimentation.

D'autre part, cette carte possède entre autres des trous au pas de 2,54 mm sur le circuit imprimé où des barrettes (mâles ou femelles selon l'utilisation)

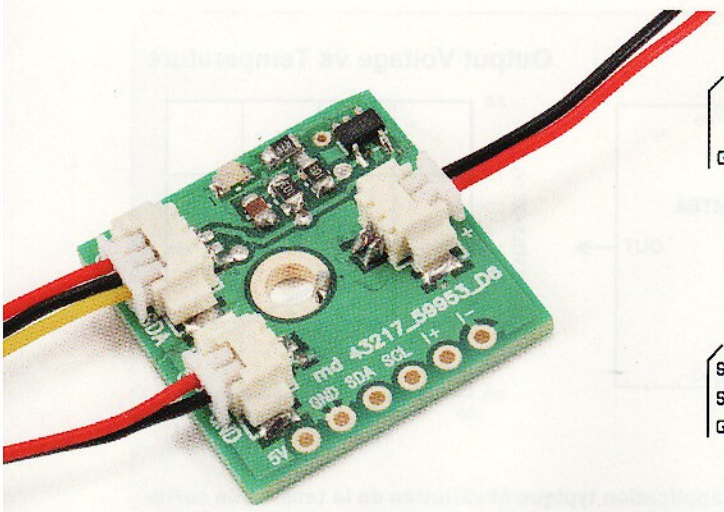
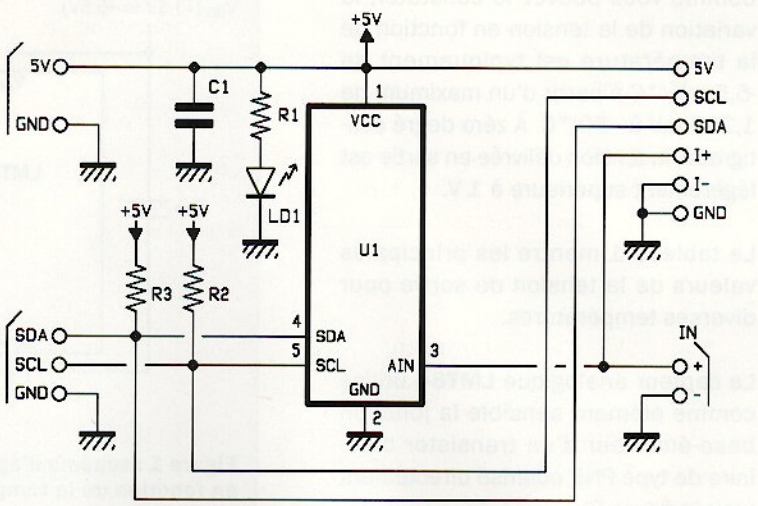


Schéma électrique de la carte convertisseur analogique/digital.



peuvent être insérées afin d'intégrer la carte dans un système existant.

L'alimentation est filtrée à l'aide du condensateur C1. Le condensateur C2 filtre la ligne de sortie du capteur en introduisant un temps de retard minimum (de l'ordre d'une milliseconde) afin d'adapter la tension de sortie aux variations de la température mesurée. La LED LD1, polarisée par la résistance R4, indique que le circuit est sous tension.

La carte convertisseur ADC

Passons maintenant à la deuxième et dernière carte « breakout board » de cette série, il s'agit de la carte convertisseur analogique/numérique basée sur un circuit intégré **MCP3221** de Microchip. Celui-ci permet la conversion numérique d'une tension analogique appliquée sur son entrée avec une **résolution de 12 bits** (4096 valeurs).

Le résultat de la conversion dépend de la tension d'alimentation, en ce sens que la définition et par conséquent la dynamique du signal échantillonné dépend de la tension d'alimentation.

En effet, la tension de référence des comparateurs est fixée par un diviseur de tension interne alimenté à partir de la même tension d'alimentation.

Si la carte est alimentée avec une tension de 5 V, l'amplitude maximale de la tension d'entrée est égale à 5 000 mV. Dans ce cas, la résolution est égale à : 5000 / 4096 = 1,22 mV. Cela veut dire que le plus petit échantillon vaut 1,22 mV.

La tension d'entrée maximale qui peut être convertie sans dépassement est fonction de la tension d'alimentation, en ce sens que la tension sur la broche $\mathbf{A_{IN}}$ du MCP3221 doit être au maximum égal à celle d'alimentation. Si la carte est alimentée en 3 V, la tension sur la broche $\mathbf{A_{IN}}$ à convertir doit être au maximum de 3 V.

Le convertisseur A/D est doté d'une sortie de type **BUS I²C** fonctionnant avec une vitesse d'horloge de 100 kHz dans le mode standard (Standard Mode) et jusqu'à 400 kHz en mode rapide (Fast Mode). L'échantillonnage s'effectue à une fréquence maximale de 22,3 ksps en mode rapide I²C.

Listing 1

```
#include <Wire.h>

#define ADDRESS 0x4D          // l'adressage du MCP3221-A5 est dans un format de 7 bits (il existe d'autres formats
d'adressage disponibles)

float vRef = 5000;            //vRef en mV
float stepSize = vRef/4096;   //chaque pas = vRef/ nombre de pas sur 12 bits

void setup(){
  Wire.begin();               // initialise la communication I2C
  Serial.begin(9600);         // initialise la communication série
}

 void loop(){
  byte adc_MSB;
  byte adc_LSB;
  int adcRaw;
  float milliVolts;

  Wire.requestFrom(ADDRESS, 2);   // demande 2 octets
  while(Wire.available() < 2);    //il faut 2 octets pour recevoir

  adc_MSB = Wire.read();
  adc_LSB = Wire.read();
  adcRaw = (adc_MSB * 256) + adc_LSB;
  milliVolts = adcRaw * stepSize;
  Serial.print("ADC: ");
  Serial.println(adcRaw);
  Serial.print("Voltage: ");
  Serial.println(milliVolts/1000);
  Serial.print("mV/Step    : ");
  Serial.println(stepSize,4);
  delay(1000);
}
```

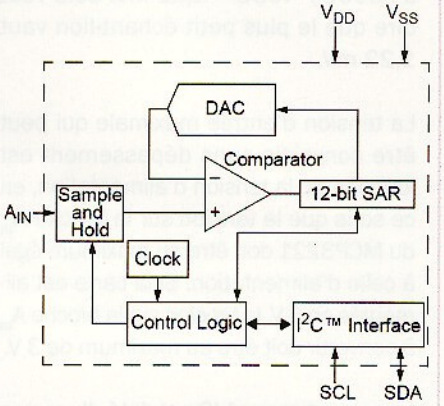


Figure 3 : schéma synoptique du convertisseur A/D MCP3221 de Microchip.

NB : lors de la conversion de données, un signal analogique est converti en un flux de nombres, chacun représentant l'amplitude du signal analogique à un moment donné. Chaque nombre est appelé « **échantillon** » (sample).

Le nombre d'échantillons par seconde (sample per second) est appelé taux d'échantillonnage (sampling rate), mesuré en échantillons par seconde. Donc l'unité « **ksps** » signifie « **kilo-sample per second** » soit « **kiloéchantillons par seconde** ».

Le circuit intégré MCP3221 peut être alimenté avec une tension continue de valeur comprise entre 2,7 V et 5,5 V. Dans notre cas, le circuit est alimenté par une tension de 5 V qui est filtrée de façon appropriée par le condensateur C1. Sur la ligne d'alimentation est également présente la LED LD1 polarisée par la résistance R1. Lorsqu'elle est allumée, elle indique que le circuit est sous tension et opérationnel.

Le circuit intégré MCP3221 est disponible sous différents types de boitiers plastiques traditionnels et CMS (SOT-23-5). Pour le modèle industriel, sa température de fonctionnement est comprise entre – 40 °C à +85 °C. Pour la version automobile sa température de fonctionnement est comprise entre -40 °C à +125 °C.

Les lignes **SCL** et **SDA** du bus **I²C** sont munies chacune d'une **résistance de tirage** (pull-up) reliée à la ligne d'alimentation (R2 pour SCL et R3 pour SDA).

Une partie de l'adresse du bus I²C du circuit intégré MCP3221 est fixée par le constructeur à « **1001** » (device code). Ces 4 premiers bits correspondent au code du périphérique.

L'utilisateur peut uniquement « modifier » l'adressage du MCP3221 sur les 3 derniers bits A2, A1 et A0 (adress bits), ce qui donne 8 possibilités. Nous pouvons donc relier sur un même bus I²C jusqu'à 8 convertisseurs MCP3221.

Les bits A2, A1 et A0 par défaut sont à « 101 ». **Pour des valeurs différentes, il est nécessaire de contacter le fabricant Microchip** (cas de plusieurs MCP3221 sur le même bus).

Les connexions d'alimentation, du signal d'entré analogique à numériser et des lignes du bus I²C sont reportées sur des connecteurs disposés sur le côté du circuit imprimé de la carte. De même, ces connexions sont reportées vers des pastilles dans lesquelles il est possible de souder des barrettes au pas de 2,54 mm afin d'intégrer la carte dans des systèmes existants.

Utilisation de la carte ADC avec Arduino

Cette carte « breakout board » proposée ici peut être interfacée avec **Arduino** sans avoir besoin d'une librairie spécifique. Il suffit d'utiliser simplement la librairie « **Wire** » déjà disponible dans l'environnement de développement IDE d'Arduino.

Le convertisseur A/D MCP3221 peut en théorie avoir sa propre adresse parmi 8 disponibles. Cela vous permettra d'utiliser jusqu'à 8 « breakout board » ADC connectées sur les mêmes lignes de communication SDA et SCL.

Comme nous l'avons indiqué, la configuration des adresses n'est pas accessible par l'utilisateur, en fait le registre qui contient l'adresse n'est pas configurable via le bus I²C mais paramétré en usine.

Dans la pratique, pour avoir des adresses différentes de celle par défaut, il faut choisir une référence différente.

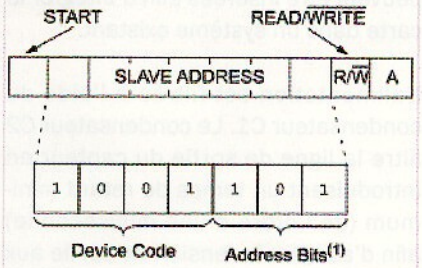


Figure 4 : structure des données envoyées par le MASTER BUS I²C au MCP3221.

Dans notre cas, nous avons choisi un **MCP3221A5T** où le chiffre « 5 » indique sans surprise l'adresse associée par défaut. En effet **5** en **décimal** équivaut à « **101** » en **binaire**. Pour d'autres références, il faut contacter le fabricant.

L'adresse est la même que « **Address Bit** » qui est appelée dans l'octet de commande envoyé par l'unité **MASTER** du bus **I²C** lors du démarrage d'une session de communication. Cela est représenté au format binaire « **101** » sur la figure 4.

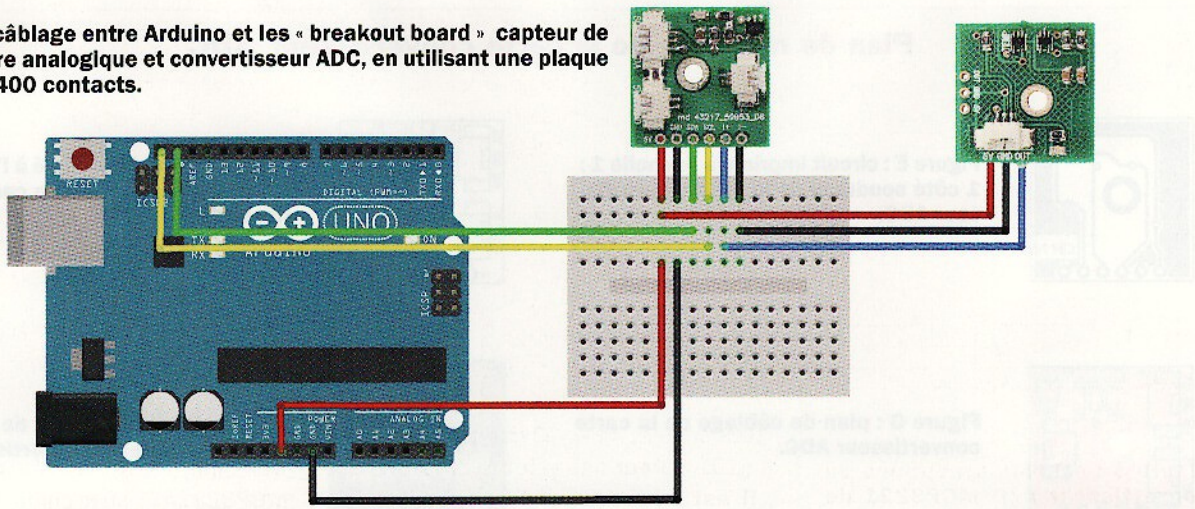
Afin de pouvoir gérer la carte contenant le MCP3221A5T avec Arduino, vous devez d'abord écrire dans l'IDE le code source proposé dans le **Listing 1**.

Avant de commencer, nous devons expliquer certains aspects du sketch (programme) afin de s'assurer que la conversion analogique/digitale génère une véritable valeur.

Le convertisseur A/D proposé ici dispose d'une résolution de 12 bits (4096 échantillons) dont la valeur résultante dépend de la tension d'alimentation du circuit intégré MCP3221. Dans notre cas, le MCP3221A5T est alimenté avec une tension de 5 V, donc la résolution est de 5000 mV / 4096 = 1,22 mV. Donc chaque échantillons (pas) vaut 1,22 mV.

Comme vous pouvez le voir dans notre code, l'instruction « **float stepSize = vRef/4096;"** » sert justement à cela. Connaissant le terme « stepSize », nous pouvons maintenant lire la valeur résultante de la conversion et obtenir la valeur en « mV » grâce à l'instruction « **milliVolts = adcRaw * stepSize;** » du Listing 1.

Figure 5 : câblage entre Arduino et les « breakout board » capteur de température analogique et convertisseur ADC, en utilisant une plaque d'essai à 400 contacts.



En outre, dans notre exemple, nous avons utilisé pour communiquer avec le circuit intégré l'adresse de commande I²C « **0x4D** », qui permet de communiquer avec le module A/D en utilisant son adresse par défaut.

En figure 5 est illustré le plan de câblage d'une carte Arduino Uno avec les cartes « breakout board » convertisseur ADC et capteur de température à sortie analogique.

Avec ce montage, il est ainsi possible de mesurer la température avec le capteur LMT84-Q1 en exécutant le sketch Arduino décrit, et ce avec une précision supérieure à celle offerte par le convertisseur intégré dans le microcontrôleur d'Arduino.

En fait, l'ensemble proposé permet de lire la sortie de la carte capteur de température contenant le LMT84-Q1 à l'aide de la carte convertisseur ADC basée sur le MCP3221A5T et d'envoyer à Arduino via le bus I²C les données correspondantes, réparties sur **2 octets**, de la conversion analogique/numérique.

L'avantage réside dans le fait que le convertisseur A/D de l'ATmega328P d'Arduino n'a qu'une résolution de seulement 10 bits, alors que le MCP3221A5T de la « breakout board » ADC échantillonne sur 12 bits.

Cela permet une résolution quadruple (4096 échantillons contre 1024 pour Arduino), c'est la solution visible en figure 5.

Dans le montage proposé, il faut **relier la sortie analogique de la carte capteur de température à l'entrée analogique de la**

Plan de montage de la carte capteur de température

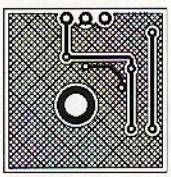


Figure A : circuit imprimé à l'échelle 1 : 1 côté soudures de la carte capteur de température.

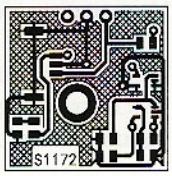


Figure B : circuit imprimé à l'échelle 1 : 1 côté composants de la carte capteur de température.

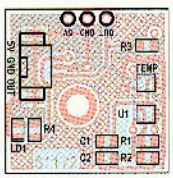


Figure C : plan de câblage de la carte capteur de température.

Figure D : photo de l'un de nos prototypes de la carte capteur de température.

Liste des composants de la carte capteur de température

R1..... 160 kΩ 1 % boîtier 0805
R2..... 56 kΩ 1 % boîtier 0805
R3..... 0 Ω boîtier 0805
R4..... 470 Ω boîtier 0805
C1..... 10 nF céramique boîtier 0805
C2..... 330 pF céramique boîtier 0805
LD1.... LED verte boîtier 0805)
U1..... MCP6L01T-E/LT
TEMP. LMT84DCKT

Divers

Barrette mâle 3 broches coudée à 90°
Connecteur JST 1,25mm 3 voies pour ci
Connecteur JST 1,25mm 3 voies fils volants

Plan de montage de la carte convertisseur ADC

Figure E : circuit imprimé à l'échelle 1 : 1 côté soudures de la carte convertisseur ADC.

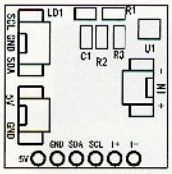


Figure F : circuit imprimé à l'échelle 1 : 1 côté composants de la carte convertisseur ADC.

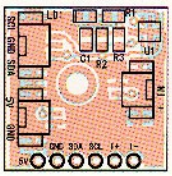


Figure G : plan de câblage de la carte convertisseur ADC.

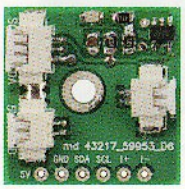


Figure H : photo de l'un de nos prototypes de la carte convertisseur ADC.

Plan de montage de la carte convertisseur ADC

R1..... 470 Ω boîtier 0805
R2..... 10 kΩ boîtier 0805
R3..... 10 kΩ boîtier 0805

C1..... 100 nF céramique boîtier 0805
LD1.... LED verte boîtier 0805
U1..... MCP3221A5T-I/OT

Divers

Barrette mâle 6 broches coudée à 90°
Connecteur JST 1,25mm 2 voies pour ci (x3)
Connecteur JST 1,25mm 2 voies fils volants (x3)

carte convertisseur ADC, de sorte que la tension fournie par le LMT84-Q1 soit échantillonnée (numérisée).

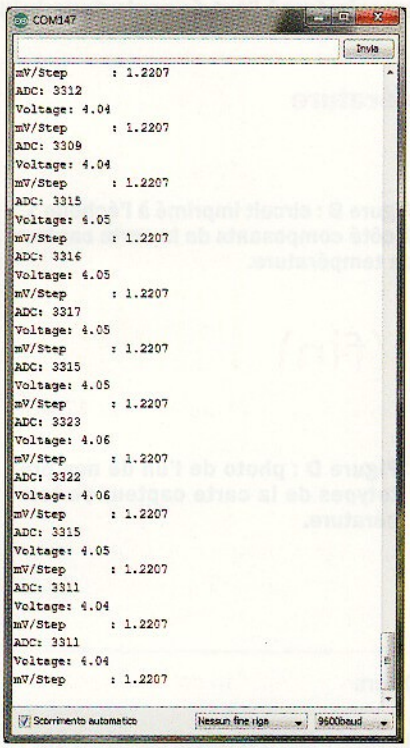


Figure 7 : en exécutant le firmware, vous pouvez visualiser dans la fenêtre du moniteur série de l'IDE Arduino, grâce à l'instruction « Serial.print », les valeurs des tensions provenant de l'ADC.

Réalisation pratique

Passons maintenant à la construction de ces deux « breakout board ». Elles sont également disponibles en version montée et testée pour ceux d'entre vous ne disposant pas de matériel spécifique pour le montage de composants CMS.

Vous pouvez télécharger dans le sommaire détaillé de la revue 138 les typons ainsi que les fichiers GERBER des circuits imprimés pour une fabrication professionnelle.

Une fois les circuits imprimés réalisés, vous devez vous procurer un fer à souder de 25 W pour composants CMS ainsi que de la soudure de diamètre 0,5 mm. Vous devez utiliser une loupe, de préférence éclairée.

À l'aide d'un peu de colle pour CMS placez les composants aux emplacements respectifs et, commencez par souder une broche à la fois en laissant refroidir entre chaque soudure. Evitez un excès de soudure qui provoquerait des courts-circuits sous les composants.

Les plans de câblage des cartes sont visibles aux figures C et G, aidez-vous des photos des prototypes.

Concernant la carte capteur de température, **faites attention à l'orientation du capteur et de l'amplificateur opérationnel.**

Les 2 composants ont leurs 3 broches orientées vers l'extérieur du circuit imprimé. Disposez correctement la LED et soudez délicatement les connecteurs JST.

En ce qui concerne la carte ADC, le circuit **U1 en boîtier SOT-23 a ses 3 broches dirigées vers l'extérieur du circuit imprimé.** Evitez un excès de soudure.

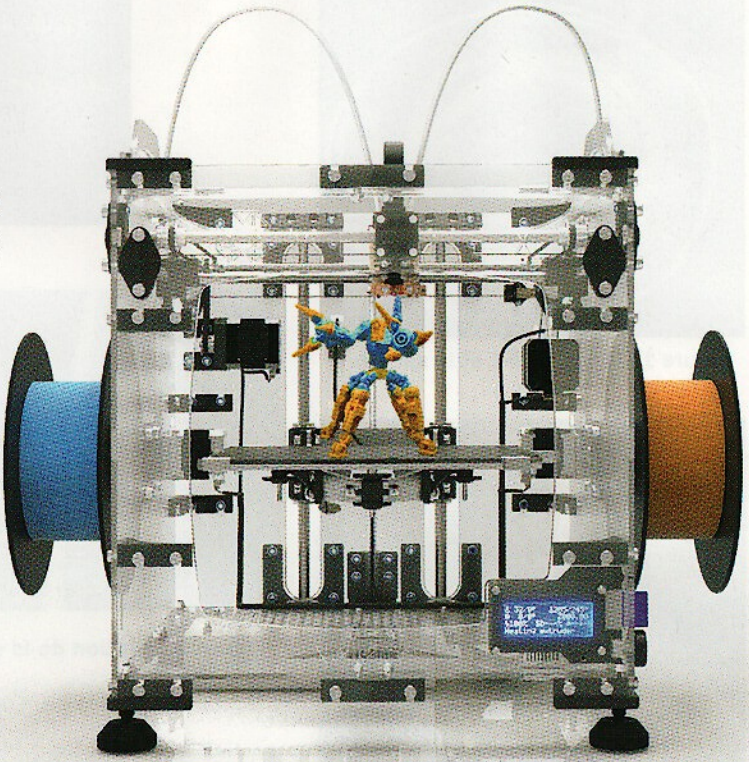
Soudez ensuite la LED, les composants passifs et les connecteurs toujours de manière délicate.

Si vous ne pouvez pas vous procurer les connecteurs JST, vous pouvez utiliser des barrettes mâles/femelles au pas de 2,54 mm pour connecter les cartes.

Nous arrivons au terme de la description de cette série de « breakout board ». Nous vous proposerons dans l'avenir d'autres cartes de ce type basées sur des amplificateurs opérationnels inverseurs/non-inverseurs et des comparateurs.

3DVERTEX - 4

Dans le précédent numéro 137 d'Electronique et Loisirs Magazine, nous avons consacré la totalité de l'article à la description du montage de la tête d'impression (extrudeuse) qui, rappelons-le, est une étape longue et minutieuse. Nous allons maintenant terminer le montage mécanique de la 3DEVERTEX avec la mise en place de l'axe des Z et de la structure de l'imprimante.



3DVERTEX

L'IMPRIMANTE 3D BICOLORE

Quatrième partie : La mécanique (fin)

de Boris LANDONI

Installation du tube à filament

Tout d'abord nous devons revenir un instant sur la tête d'impression, car nous n'avons pas installé le tube à filament qui alimente (en filament comme son nom l'indique) l'extrudeuse.

Munissez-vous du tube PTFE et de la spirale (voir la figure 1), le tube mesure 1 m. Coupez le tube à une longueur de 70 cm et insérez-le dans le raccord situé du côté du moteur d'alimentation en filament, vérifiez que le tube s'enfonce complètement (voir la figure 2).

Avec la spirale, commencez par enrouler le tube et le faisceau de câbles du côté du moteur d'alimentation en filament de l'extrudeuse, comme visible en figure 3.

Continuez d'enrouler la spirale jusqu'au connecteur situé à l'autre extrémité du câble qui se connecte sur l'extrudeuse, éventuellement coupez l'excédent de la spirale. Vous devez obtenir un résultat similaire à celui de la figure 4.

Desserrez le serre-câble et faites passer la spirale à l'intérieur (voir la figure 5), puis refixez l'ensemble. Enfichez le connecteur dans l'extrudeuse et insérez le tube PTFE

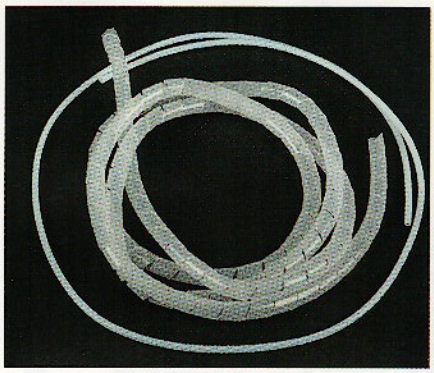


Figure 1 : le tube PTFE et la spirale.

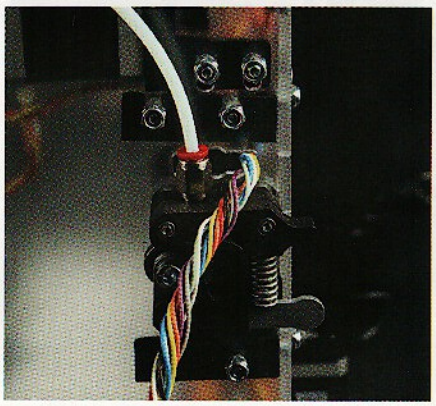


Figure 2 : insertion du tube du côté du moteur d'alimentation en filament.

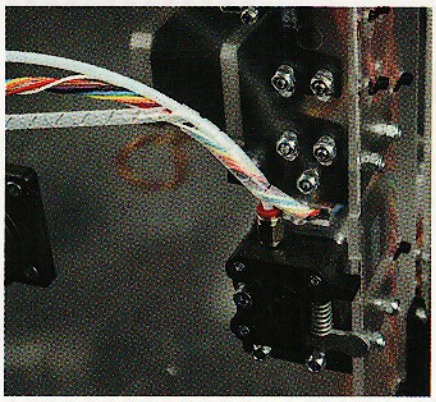


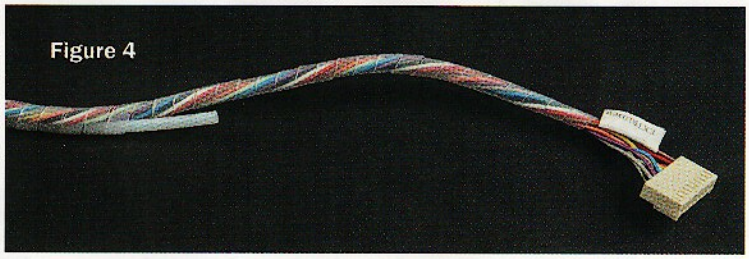
Figure 3 : mise en place de la spirale à partir du côté du moteur d'alimentation en filament.

dans le raccord situé sur la tête d'impression, vérifiez que le tube s'enfonce complètement (voir la figure 6). Vous devez obtenir un résultat semblable à celui de la figure 7.

Assemblage de l'axe des Z

Tout d'abord vous devez préparer les pièces suivantes visibles en figure 8 :

- 2 barres de 350 mm, de diamètre 10 mm ;



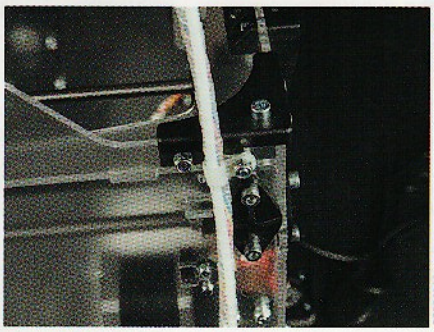


Figure 5 : fixation de la spirale dans le serre-câble.

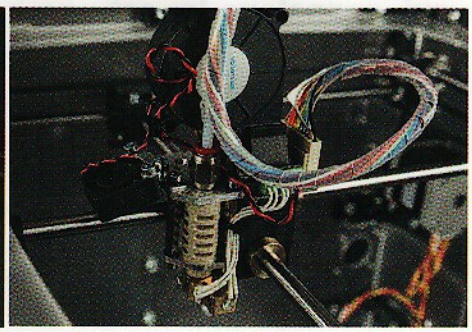
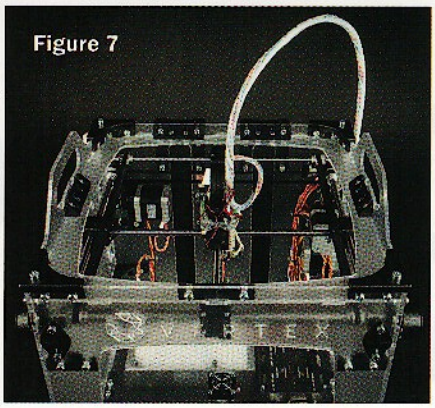


Figure 6 : insertion du connecteur et du tube PTFE dans le raccord de l'extrudeuse.



- 4 roulements de diamètre 10 mm ;
- 4 supports pour les barres ;
- 8 boulons M3 x 16 mm ;
- 8 circlips de 18 mm.

D'autre part vous devez vous munir d'une pince à circlips, commencez par installer un circlip sur l'un des côtés d'un roulement, en prenant soin de ne pas rayer ou endommager ce dernier avec la pince (voir la figure 9).

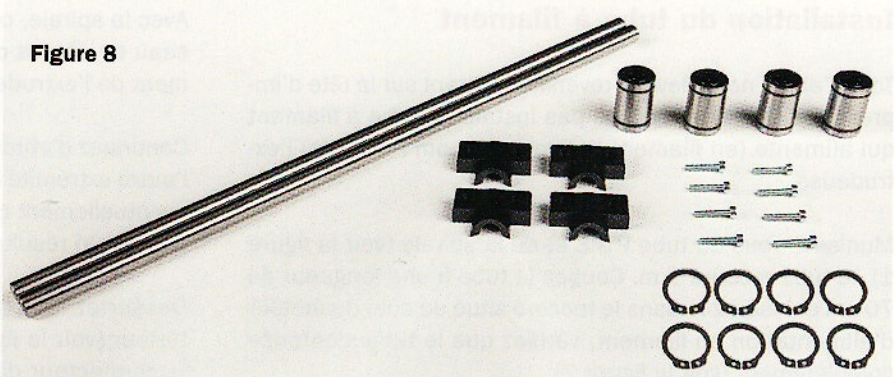
Ensuite insérez l'autre circlip de l'autre côté du roulement. Continuez avec les autres roulements, vous devez avoir les 8 circlips installés sur les 4 roulements comme visible en figure 10.

Ensuite, insérez 2 roulements dans chaque barre de 350 mm et placez ces dernières dans les supports de l'axe des Z.

À l'aide des boulons M3 x 16 mm fixez les 4 supports des barres de l'axe des Z.

Utilisez 2 écrous M3 pour chaque boulon, éventuellement incliner l'imprimante vers l'arrière pour plus de facilité (voir la figure 11).

Figure 8



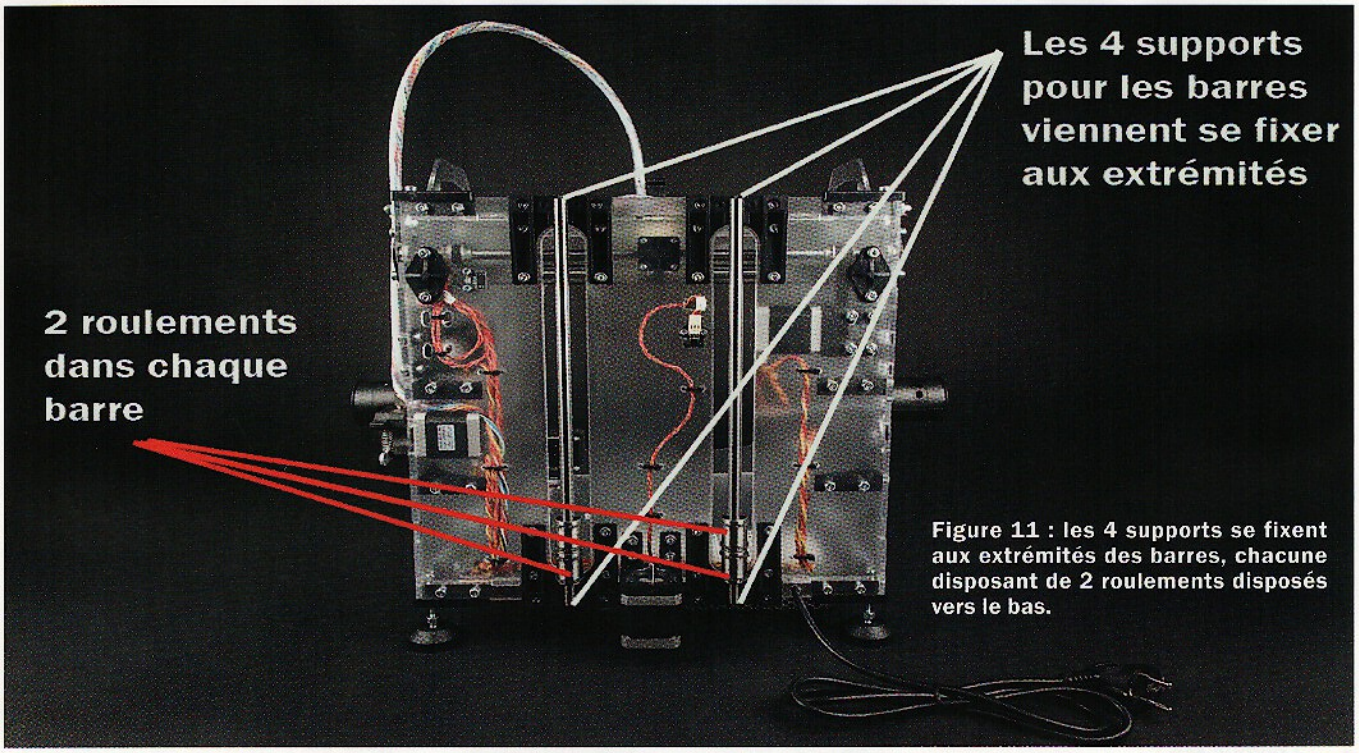


Figure 11 : les 4 supports se fixent aux extrémités des barres, chacune disposant de 2 roulements disposés vers le bas.

Figure 9 : installez les circlips avec la pince adéquate, vous devez avoir 2 circlips sur chaque roulement.

Figure 10

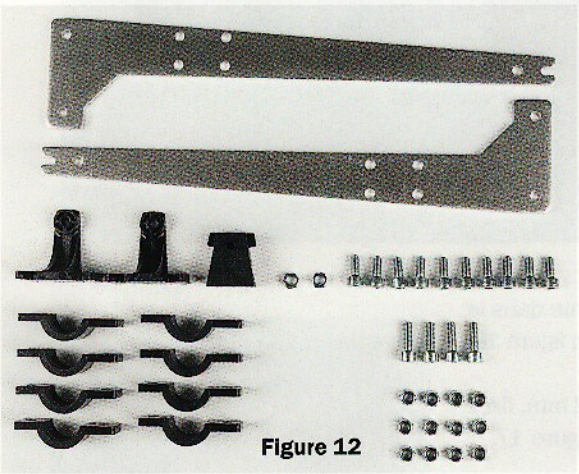


Figure 12

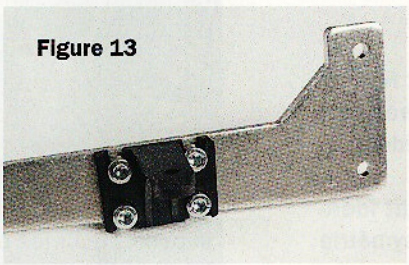


Figure 13

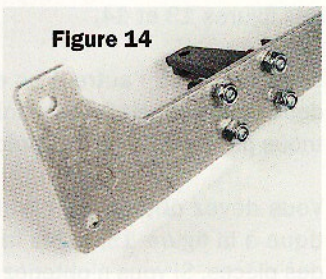


Figure 14

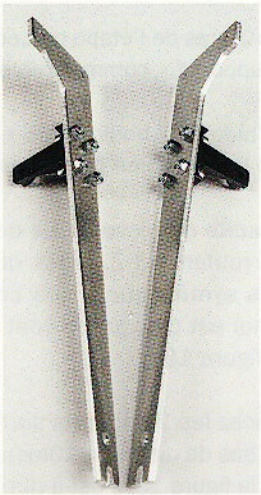


Figure 15 : les 2 bras du plateau forment un « V ».

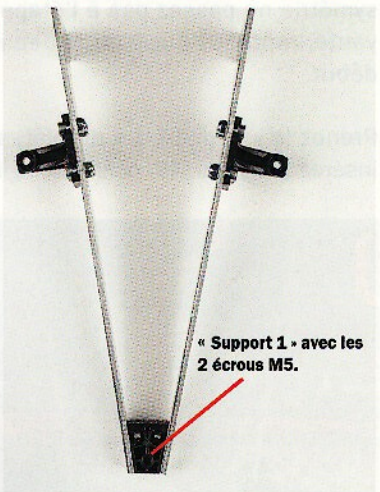


Figure 16 : fixation du « support 1 » à l'extrémité des bras.

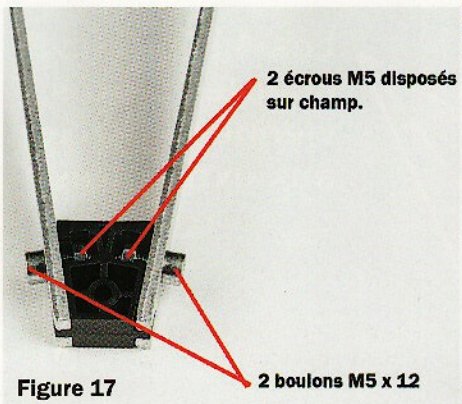


Figure 17

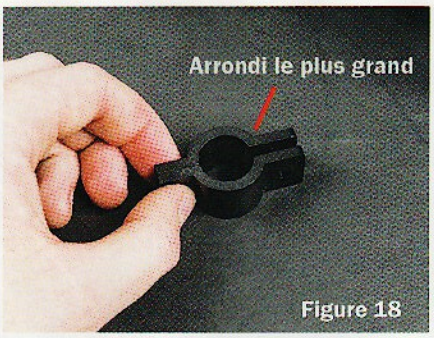


Figure 18

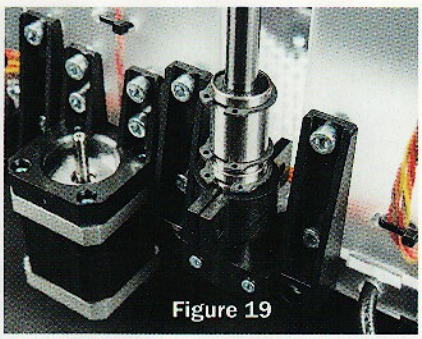


Figure 19

Maintenant, vous allez fixer les bras du plateau d'impression. Pour cela cherchez les pièces suivantes et mettez-les de côté (aidez-vous avec la figure 12) :

- 2 bras de plateau ;
- 2 « support 2 » du plateau ;
- 1 « support 1 » du plateau ;
- 8 supports de roulement Z ;
- 2 écrous M5 ;
- 10 boulons M5 x 12 mm ;
- 4 boulons M5 x 16 mm ;
- 12 écrous M5 autobloquants.

Fixez le « support 2 » du plateau à l'un des bras en vous servant de 4 écrous M5 autobloquants et de 4 boulons M5 x 12 mm. Vérifiez l'orientation des pièces, elle doit être identique à celle des figures 13 et 14.

Ensuite prenez l'autre bras et fixez le deuxième « support 2 » de manière symétrique par rapport au bras précédent.

Vous devez obtenir un résultat identique à la figure 15, notez la symétrie des pièces. Si vous n'obtenez pas cette symétrie ne passez pas à l'étape suivante, reprenez l'opération depuis le début.

Prenez le « support 1 » du plateau et insérez 2 écrous M5, puis faites glisser

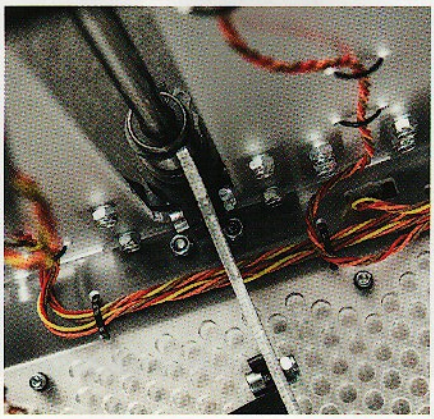


Figure 20 : serrez le boulon et l'écrou à la main sans forcer.

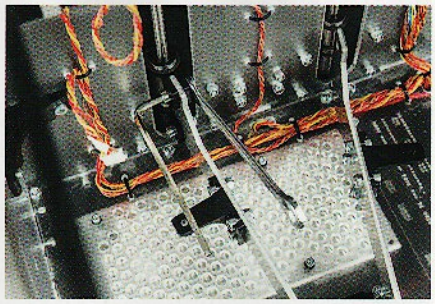


Figure 21 : les bras du tableau se trouvent à l'intérieur de l'imprimante.

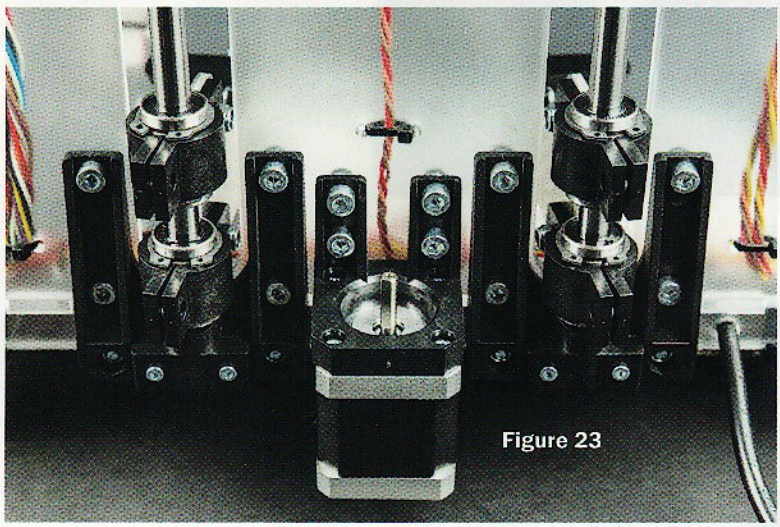


Figure 23

les 2 bras de l'étape précédente dans le « support 1 », comme visible en figure 16.

À l'aide de 2 boulons M5 x 12 mm, fixez les bras comme illustré en figure 17.

Ensuite munissez-vous de 2 supports de roulement Z, notez qu'ils ne sont pas symétriques. Les côtés où l'arrondi est plus grand vont de pair (voir la figure 18).

Placez les 2 supports sur le roulement du bas de diamètre 10mm, dans le sens de la figure 18, c'est-à-dire la partie des

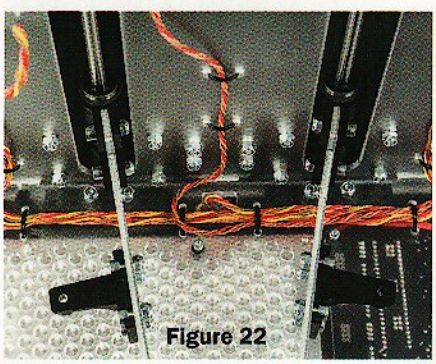


Figure 22

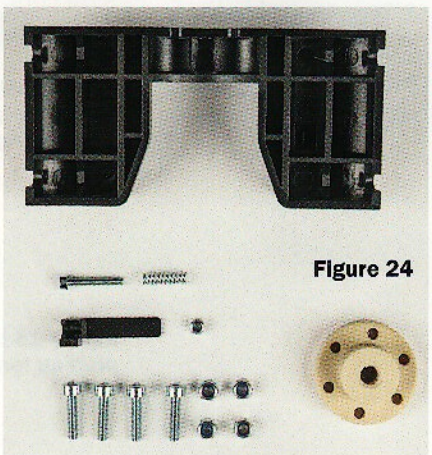


Figure 24

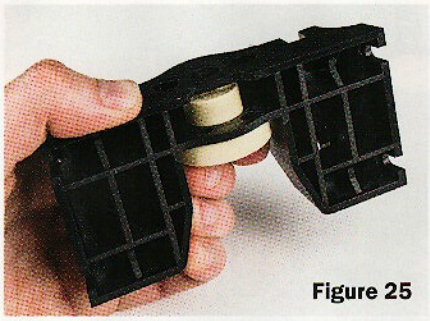


Figure 25


Figure 26

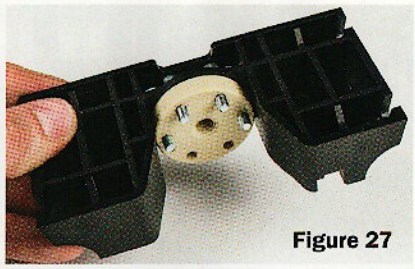


Figure 27

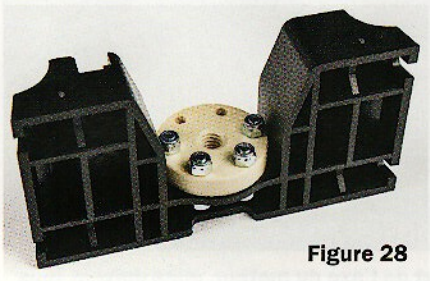


Figure 28

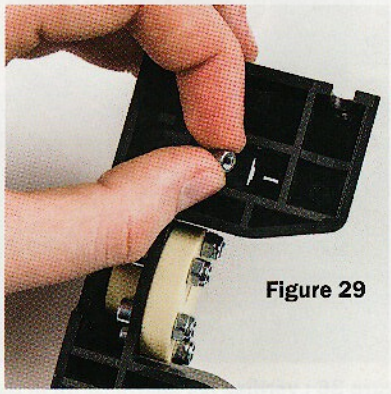
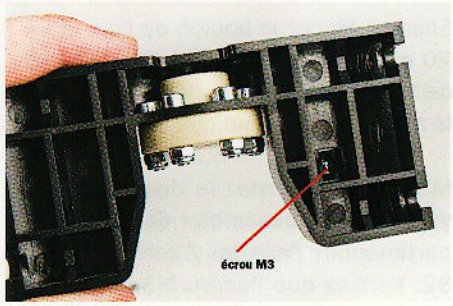


Figure 29

Figure 30 : insertion de l'écrou M3 dans le compartiment, faites attention à l'orientation.



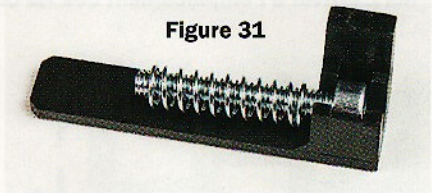


Figure 31

Figure 33 : la tête du boulon doit être à l'intérieur du compartiment.

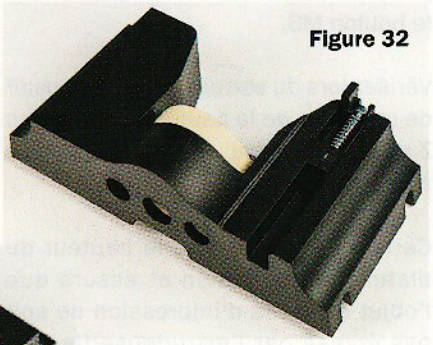


Figure 32

2 supports qui est maintenue par les doigts se trouve à l'extérieur de l'imprimante (voir la figure 19).

Maintenant faites glisser un des bras délicatement dans la partie interne des 2 supports (celle ayant les côtés où l'arrondi est plus grand, c'est-à-dire le côté opposé à l'ensemble maintenu par les doigts de la figure 18).

Notez que le bras se trouve à l'intérieur de l'imprimante. Commencez à fixer l'ensemble à l'aide d'un boulon M5 x 16 mm et un écrou M5 autobloquant sans serrer, laissez du jeu (voir la figure 20).

Faites de même avec le roulement situé au-dessus du précédent, remarquez l'orientation des bras du tableau à l'intérieur de l'imprimante en figure 21.

Ensuite continuez de la même manière pour les 2 autres roulements du côté opposé, vous devez obtenir un résultat semblable aux figures 22 et 23. Si cela n'est pas le cas, ne passez pas à l'étape suivante, reprenez l'opération depuis le début.

Vous allez maintenant procéder au montage du mécanisme de l'axe des Z à proprement parler. Pour cela vous devez préparer les pièces suivantes, visibles en figure 24 :

- 1 compartiment de l'axe des Z (pièce noire) ;
- 1 boulon M3 x 20 mm ;
- 1 ressort ;
- 1 dispositif de calibrage de la hauteur de l'axe des Z ;
- 1 écrou M3 autobloquant ;
- 4 boulons M4 x 16 mm ;
- 4 écrous M4 autobloquants ;
- 1 raccord en iglidur (pièce jaune ronde) ;

Insérez le raccord en iglidur dans le compartiment de l'axe des Z comme indiqué en figure 25, puis vissez les 4 boulons M4 x 16 mm dans le compartiment de l'axe des Z (voir les figures 26 et 27).

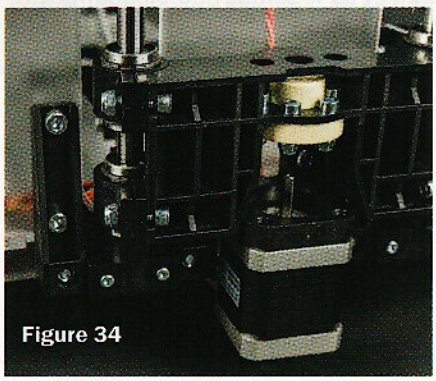


Figure 34

À l'aide des 4 écrous M4, fixez le raccord au compartiment (voir la figure 28).

Prenez l'écrou M3 et placez-le dans la cavité du compartiment de l'axe des Z comme indiqué en figure 29. L'orientation de l'écrou est indiquée en figure 30.

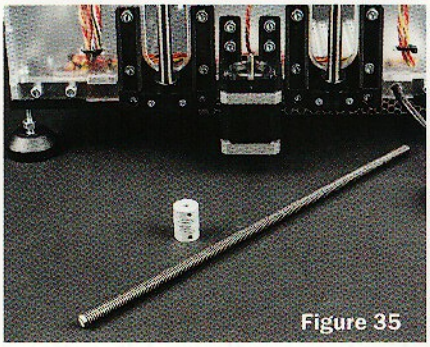


Figure 35

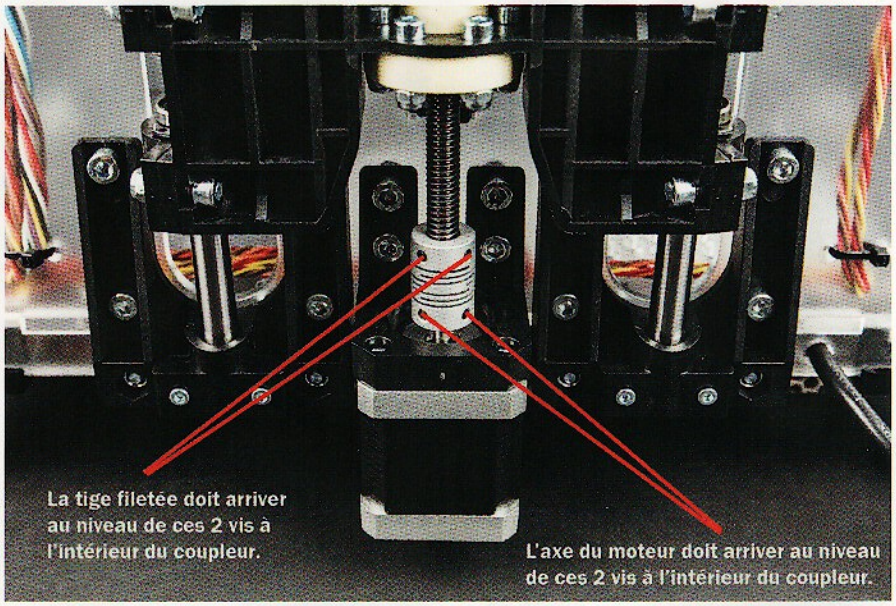


Figure 36 : vérifiez les positions de la tige et de l'axe du moteur, elles doivent tous les 2 arriver à l'emplacement des vis comme indiqué sur l'image.

Ensuite, placez le boulon de type M3 x 20 mm dans le dispositif de calibrage de la hauteur de l'axe des Z et insérez le ressort comme indiqué en figure 31.

Maintenant montez le dispositif que vous venez d'assembler dans le compartiment de l'axe des Z selon la figure 32. Vérifiez que l'écrou M3 ne tombe pas hors du compartiment, puis serrez le boulon M3.

Vérifiez lors du serrage que le dispositif de calibrage de la hauteur de l'axe des Z a avancé d'une distance comparable à celle de la figure 33.

Cette pièce détermine la hauteur du plateau d'impression et assure que l'objet en cours d'impression ne soit pas écrasé par l'extrudeuse (la tête d'impression).

Vous allez maintenant installer sur l'imprimante l'ensemble des pièces que vous venez d'assembler, pour cela vous avez besoin de 4 boulons M5 x 16 mm et de 4 écrous autobloquants M5.

Disposez l'ensemble compartiment de l'axe des Z et dispositif de calibrage dans les roulements situés sur l'imprimante, à l'aide des 4 écrous et des 4 boulons M5 fixez l'ensemble (voir la figure 34).

Ensuite vissez les écrous et les boulons situés à l'intérieur de l'imprimante. L'axe des Z doit coulisser en hauteur sans trop de frottements.

Si l'axe ne coulisse pas, desserrez les boulons et réinstallez les supports de roulements, refixez l'ensemble une nouvelle fois.

Une fois l'opération précédente effectuée avec succès, munissez-vous du coupleur de l'axe des Z et de la tige filetée (voir la figure 35).

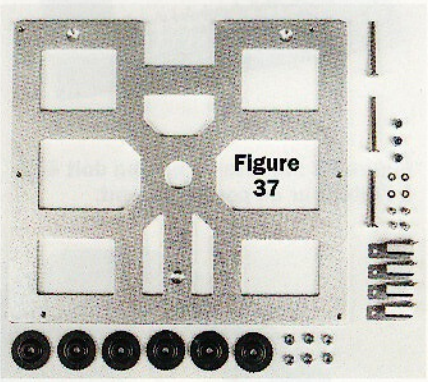


Figure 37

Vissez la tige filetée dans le raccord en iglidur (pièce jaune), puis insérez le coupleur de l'axe des Z à l'extrémité de la tige filetée, elle doit arriver au niveau des 2 premières vis du haut du coupleur.

Faites attention de ne pas l'endommager, serrez les vis de réglage du coupleur.

Descendez l'axe des Z, le coupleur doit s'insérer dans l'axe du moteur, puis serrez les vis du bas (voir la figure 36).

Assurez-vous que la tige filetée et l'axe du moteur soient parfaitement alignés, puis serrez toutes les vis y compris celles du support moteur que vous n'aviez pas serrées complètement (voir le texte plus haut dans l'article).

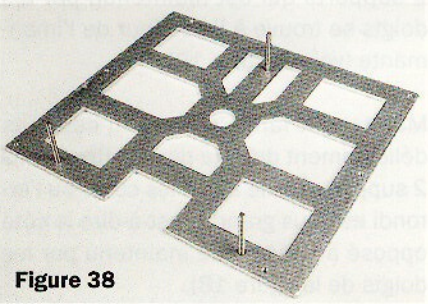


Figure 38

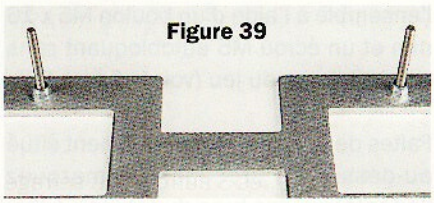


Figure 39


Figure 40

N'oubliez pas de lubrifier la tige filetée. Si vous entendez un bruit mécanique important lors du déplacement de l'axe des Z lorsque l'imprimante fonctionne, vous devez vérifier l'alignement de la tige et du moteur ainsi que la lubrification de la tige (éventuellement lubrifiez de manière plus abondante sans en faire couler dans le moteur ou à l'intérieur de l'imprimante).

Montage du plateau d'impression

Nous arrivons à la fin du montage mécanique de la 3DVERTEX (du moins pour la version à une seule tête). Pour le montage du plateau vous avez besoin des éléments suivants, visibles en figure 37 :

- 1 plateau d'impression ;
- 6 vis à serrage manuel (roues crantées noires sur l'image) ;
- 6 écrous M4 ;
- 3 vis à tête fraisée M4 x 40 mm ;
- 3 écrous autobloquants M4 ;
- 4 rondelles M3 ;
- 4 boulons M3 x 4 mm ;
- 4 clips de plateau.

Insérez les 3 vis à tête fraisées sur le plateau comme indiqué en figure 38 et à l'aide des 3 écrous autobloquants M4 serrez les vis (voir la figure 39).

Disposez 3 écrous M4 dans 3 vis à serrage manuel comme visible en figure 40.

Vissez les 3 vis à serrage manuel sur le plateau en respectant l'orientation de la figure 41.

Maintenant, retournez le plateau d'impression et fixez les clips de plateau à l'aide de 4 boulons M3 x 4 mm et des 4 rondelles comme illustré en figure 42.

L'orientation des clips doit correspondre aux figures 43 et 44.

Ensuite prenez les 3 autres vis à serrage manuel et insérez un écrou M4 dans chacune d'elle, comme vous l'avez fait précédemment (voir la figure 40).

Disposez le plateau à l'intérieur de l'imprimante, puis faites coulisser délicatement les axes des 3 vis à serrage manuel dans les 3 supports du bras en forme de « V ».

Vous devez obtenir le résultat de la figure 45.

Prenez les 3 autres vis à serrage manuel et fixez le plateau comme indiqué en figure 46.

Maintenant vous prenez la plaque de verre et la feuille en « buildtak », collez

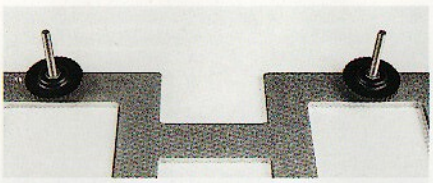


Figure 41 : vérifiez l'orientation des vis à serrage manuel, elle doit être conforme à l'image.

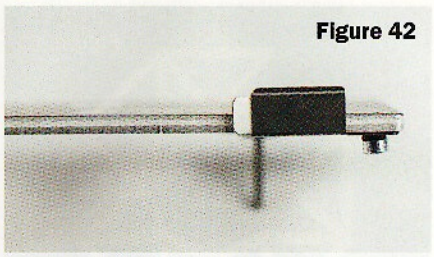


Figure 42

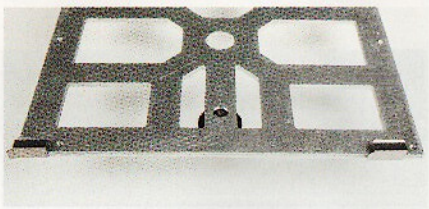


Figure 43 : orientation des clips sur la partie avant du plateau. Les languettes des clips pointent vers la gauche.

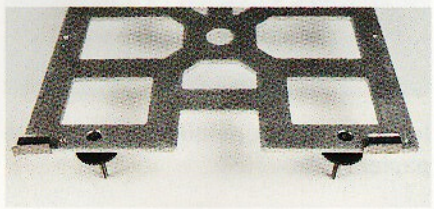


Figure 44 : orientation des clips sur la partie arrière du plateau. Les languettes des clips pointent vers la gauche.

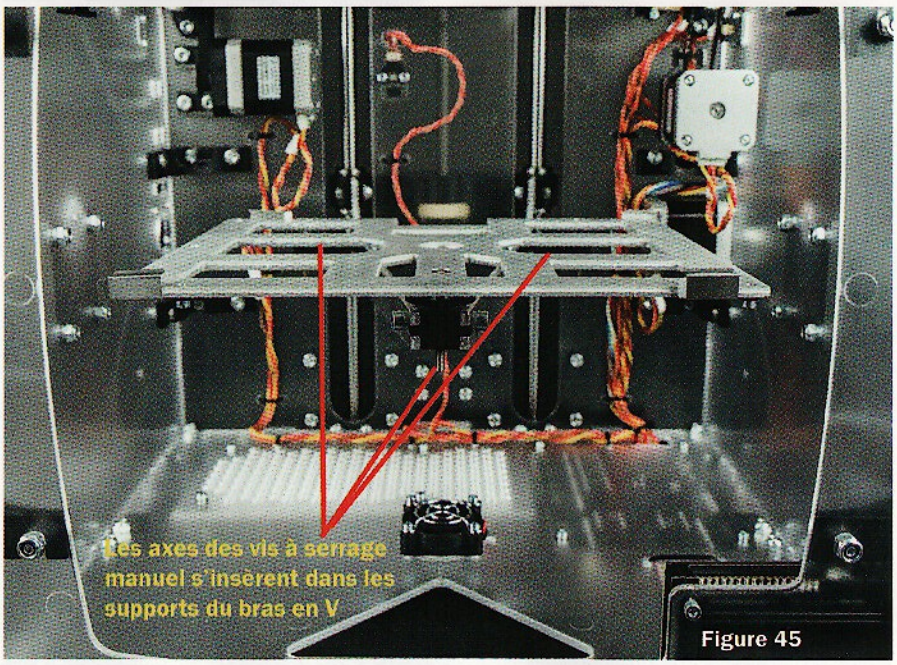


Figure 45

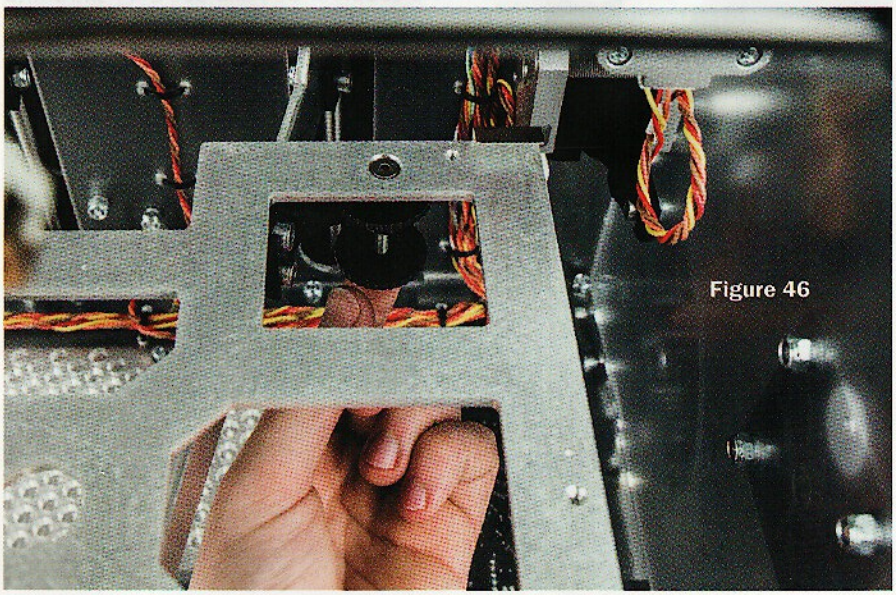


Figure 46

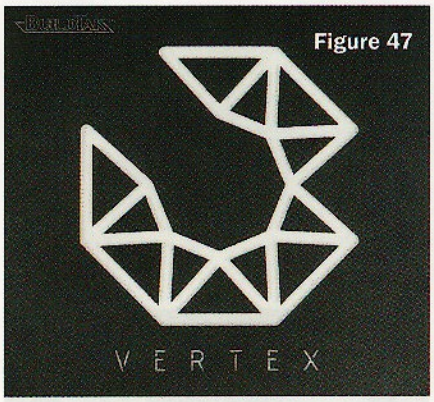


Figure 47

la feuille sur la plaque en retirant le papier blanc.

Faites attention à l'orientation, la plaque n'est pas carrée. Assurez-vous aussi qu'il n'y a pas de formation de bulles entre la feuille et le verre, utilisez une règle en plastique pour étaler délicatement la feuille. Vous devez obtenir le résultat de la figure 47.

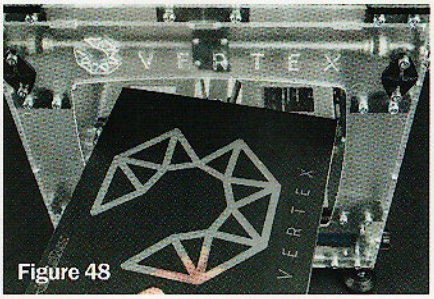


Figure 48

Eventuellement, à l'aide d'un cutter, enlever l'excédent de la feuille qui est légèrement plus grande que la plaque de verre.

Insérez la plaque sur le plateau d'impression, à l'intérieur de l'imprimante, comme indiqué en figure 48.

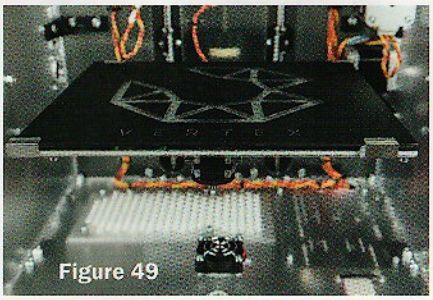


Figure 49

La plaque de verre est maintenue à l'aide des clips (voir la figure 49).

À ce stade le processus de construction de l'imprimante 3DVERTEX à une seule tête d'impression est terminé.

Votre imprimante doit ressembler à celle de la figure 50. Gardez les pièces en supplément, il se peut que vous en ayez besoin par la suite.

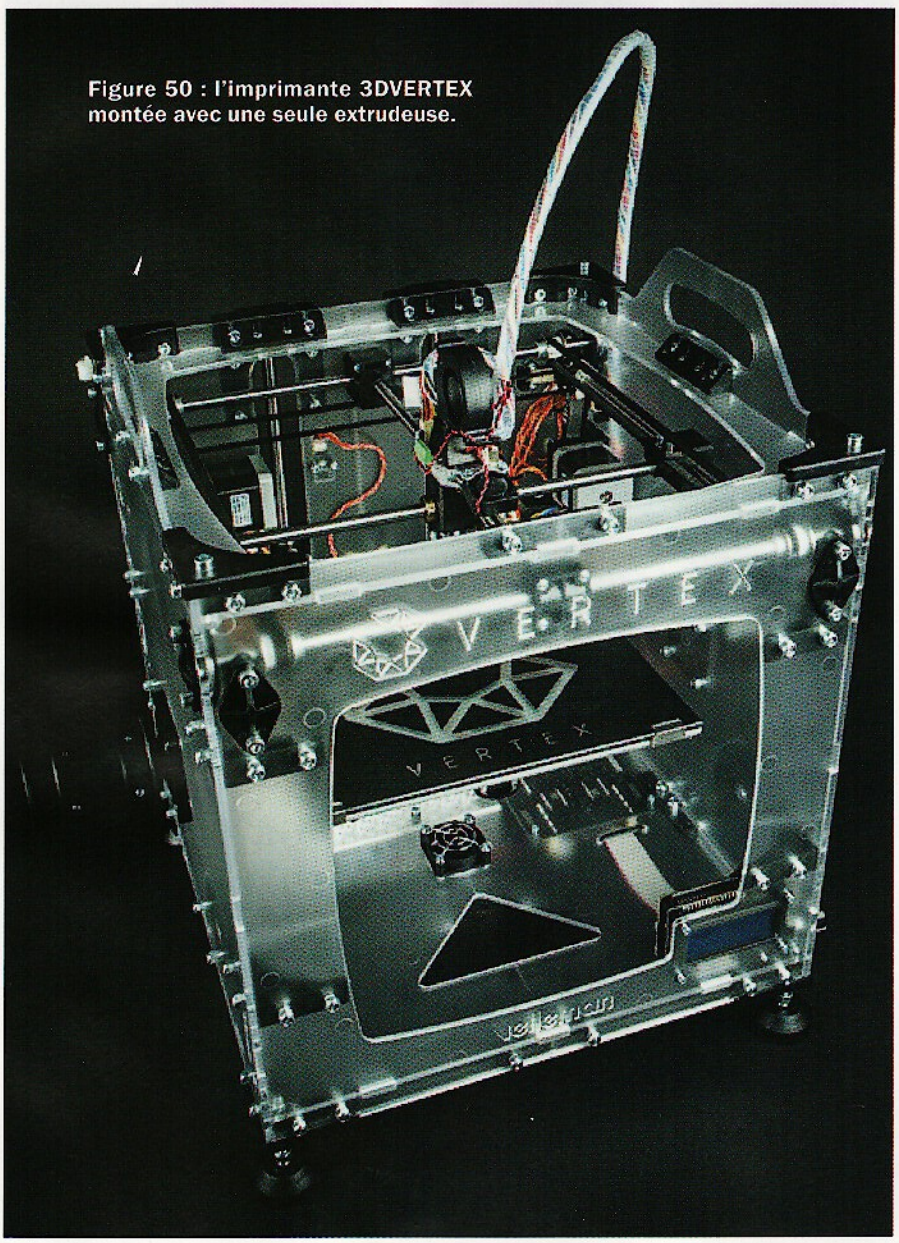


Figure 50 : l'imprimante 3DVERTEX montée avec une seule extrudeuse.

Notez que le montage de la seconde tête d'impression est identique à celui décrit dans les articles précédents (reportez-vous aux numéros 136 et 137 d'Électronique et Loisirs Magazine).

Rappelez-vous que la 2ème extrudeuse est une option, elle n'est pas obligatoire pour faire fonctionner l'imprimante. Vous pourrez ainsi l'acquérir par la suite, selon vos besoins.

Dans le prochain numéro et les suivants, nous étudierons de manière détaillée l'utilisation de l'imprimante ainsi que les problèmes que vous pourrez rencontrer avec leurs solutions. ■

INTERFACE USB POUR LE BRAS ROBOTISÉ

Cette interface permet de contrôler, à partir d'un PC, le bras robotisé décrit dans le précédent numéro 137 d'Électronique et Loisirs Magazine. Elle autorise un mode de contrôle manuel ou automatique en effectuant des mouvements programmés (et donc répétitifs), comme par exemple une machine à commande numérique.

de Davide Scullino

Si vous vous souvenez, dans le précédent numéro, nous vous avons proposé un bras robotisé en kit conçu pour les débutants en robotique. Celui-ci, certes de conception simplifiée, offre des possibilités intéressantes dans le monde de la robotique.

Ce kit ne ressemble pas à un robot d'usine en acier dédié au montage de véhicules. Bien qu'il soit miniature par rapport aux vrais robots, il conserve la structure et les dispositifs techniques (moteurs, système d'entraînement par friction, articulations et base rotative) des produits industriels. C'est donc une sorte de modèle réduit avec lequel vous pouvez reproduire les mouvements effectués par un robot industriel.

Ce bras robotisé dispose d'une télécommande filaire pour effectuer tous les mouvements.

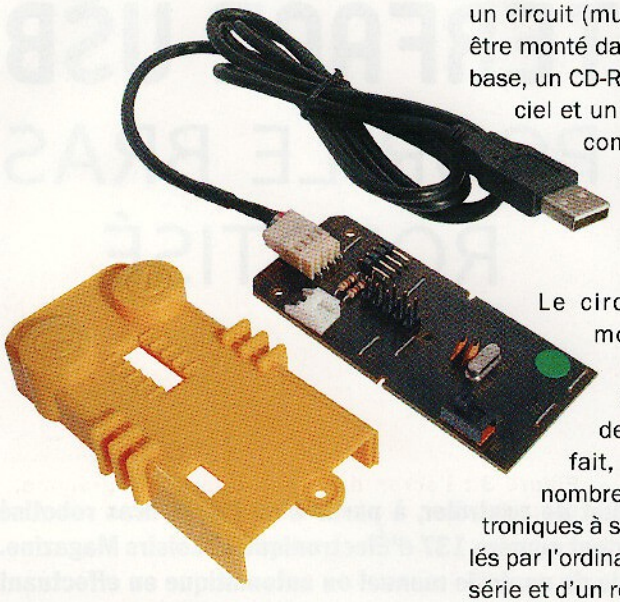


Figure 1 : le kit de modification qui ajoute au bras robotisé l'interface USB est composé d'une carte avec un câble et d'un couvercle, afin de pouvoir être inséré dans la partie arrière de la base.

La commande manuelle à fil est certainement pratique, cependant s'il est possible de gérer tous les mouvements du bras robotisé à partir d'un PC cela est certainement mieux pour l'utilisateur.

Tout d'abord, par rapport au mode manuel, vous pouvez utiliser la souris qui est plus intuitive (grâce notamment à la roulette) pour commander une rotation, un abaissement du bras, etc.

En outre, avec un ordinateur, vous avez la possibilité de mémoriser une série de mouvements (voire une séquence entière de mouvements) puis de les répéter autant de fois que vous le désirez.

Pour mémoriser une séquence il suffit d'activer la fonction appropriée, de piloter le bras selon les mouvements désirés, puis d'enregistrer le tout afin de reproduire la séquence.

Le fait de pouvoir créer des séquences de mouvements est la chose qui indubitablement rend unique le contrôle par un ordinateur. Nous sommes persuadés que cette fonction vous fera franchir le pas afin de tester le bras robotisé avec l'interface USB pour PC. Le module de commande à partir de l'ordinateur est disponible dans le commerce sous la référence « **KSR10/USB** » et comprend un circuit (muni d'un couvercle) pour être monté dans la partie arrière de la base, un CD-ROM pour installer le logiciel et un manuel. Ce module est compatible « USB 2.0 », il dispose d'un câble USB se terminant par un connecteur de type « A ».

Le circuit contenu dans le module est essentiellement fait pour remplacer les interrupteurs à levier de la commande filaire. En fait, il dispose d'un certain nombre de commutateurs électroniques à semi-conducteurs contrôlés par l'ordinateur via l'interface USB/série et d'un registre qui peut transformer les données séries en parallèle.

La chose intéressante est que, par rapport à la télécommande filaire, l'ordinateur permet l'exécution simultanée de plusieurs mouvements. Par exemple, la fermeture de la pince, la rotation de la base et le soulèvement du bras peuvent être exécutés simultanément. Cette possibilité est due au circuit qui est capable, en même temps, de mémoriser les commandes et gérer les données arrivant du port USB pour effectuer d'autres mouvements.

Pour monter l'interface USB, vous devez retirer la carte d'origine du bras robotisé située dans la partie arrière. Retirez les connecteurs des moteurs et des LED après avoir repéré tous les fils à l'aide d'étiquettes (cela vous permettra d'effectuer correctement les branchements). Ensuite enlevez le couvercle en plastique et retirez la carte, après avoir débranché le connecteur d'alimentation provenant du compartiment à piles.

Retirez la carte originale, insérez la carte USB en prenant soin de faire passer le câble USB à l'arrière de la base. Connectez le connecteur d'alimentation à celui situé sur le circuit imprimé et fixez le couvercle fourni avec le kit USB avec les mêmes vis du couvercle original.

À ce stade, il suffit d'insérer les connecteurs des moteurs et des LED, chacun à sa place grâce aux étiquettes. La modification est terminée. Avant de relier le bras robotisé à l'ordinateur,

Figure 2 : l'interface USB est livrée avec un CD-ROM contenant les drivers et le programme de gestion du bras robotisé.

placez l'interrupteur sur la position « **OFF** », de façon à éviter une consommation inutile des piles.

Le software du PC

Pour piloter le bras robotisé à partir du port USB du PC, vous devez installer dans un premier temps les pilotes (drivers), et ensuite le logiciel de commande présent sur le CD-ROM fourni avec le kit. Notez que les pilotes se trouvent dans le dossier « **USB Driver** » et le logiciel de commande dans le dossier « **Robotic Arm** ».

Notez aussi que l'**interface USB ne fonctionne pas sur les systèmes Windows 8 & 10 64 bits** (uniquement sur les versions 32 bits).

Avant d'installer le pilote, **connectez le câble USB** et mettez l'interrupteur sur la position « **ON** ». Installez d'abord le driver USB, vérifiez dans le « **Gestionnaire de périphérique** » de Windows que le périphérique USB est correctement installé (il ne doit pas y avoir de point ? jaune). Si cela n'est pas le cas, faites un clic droit sur le périphérique USB et **mettez à jour le pilote** en sélectionnant le dossier « **USB Driver** » du CD-ROM.

Une fois cette étape effectuée, allez dans le dossier « **Robotic Arm** » et cliquez sur « **Install** ». Cliquez sur « **Next** » plusieurs fois en complétant éventuellement les champs. Il se peut aussi que le programme vous demande d'installer « Flash Player » selon la configuration de votre PC, suivez alors les instructions.

Une fois l'installation terminée, une icône doit apparaître sur le « Bureau Windows » nommée « **Robotic Arm** ».

Effectuez un double clic sur l'icône, la fenêtre principale du programme

s'ouvre, avec en son centre le bouton « **PLAY** » (voir la figure 3).

Pour commencer à travailler, vous devez donc cliquer sur le bouton « **PLAY** », un nouveau menu apparaît contenant 3 boutons : « **BASIC** », « **PROGRAM** » et « **QUIT** » (voir la figure 4). En cliquant sur le 1er bouton « **BASIC** », vous accédez à une nouvelle fenêtre relative au **contrôle manuel** (voir la figure 5).

Vérifiez que vous pouvez commander le bras à l'aide du clavier et de la souris. Pour cela, commencez par sélectionner le **mode de commande** (clavier ou souris) en cliquant sur le bouton « **MODE** » en bas à gauche. Si par exemple la souris apparaît, en cliquant sur le bouton « **MODE** » vous passez en mode clavier, et en cliquant une fois de plus vous revenez en mode souris, et ainsi de suite.

Pour **transmettre des commandes** à partir du clavier afin d'effectuer des mouvements, vous devez **appuyer sur les touches** correspondant aux **lettres affichées** à l'écran. Si vous ne connaissez pas leur signification, passez en mode de commande avec la souris et cliquez sur la barre des commandes en bas de l'écran où sont reportées les lettres avec les flèches indiquant le mouvement. Chaque fois que vous survolez un groupe de deux lettres, les moteurs correspondant au mouvement et donc à la commande seront mis en évidence.

Le contrôle du bras robotisé à partir de la souris est sans doute le mode le plus pratique. La **molette**, en la faisant tourner dans les 2 sens, permet le **déplacement du bras** dans la direction opposée. Chaque fois que vous pointez un segment du bras, une boîte de dialogue apparaît en vous montrant les commandes à utiliser (boutons et molette).

Par exemple, en pointant la pince, une boîte de dialogue apparaît en vous indiquant que la molette permet l'ouverture et la fermeture de la pince et que le bouton de gauche contrôle l'allumage et l'extinction de la LED (chaque clic inverse l'état).

Pour **sortir du mode** « commande manuelle » et revenir au menu de sélection (celui de la figure 4), cliquez simplement sur le bouton « **CLOSE** » à droite de l'écran.

Le plus intéressant est certainement le mode automatique, qui est accessible en cliquant sur le bouton « **PROGRAM** ». Il est possible de **construire des séquences de mouvements** du bras très diverses, simplement en spécifiant le nombre de secondes que durera chaque mouvement.

Par exemple, rotation de la base à gauche pendant 1 seconde, ensuite ½ seconde de levage, puis fermeture de la pince pendant 1 seconde et ouverture de la pince pendant 1 seconde, enfin ½ seconde d'abaissement et 1 seconde de rotation de la base dans le sens opposé.

La **séquence** peut être **testée** et **enregistrée dans un fichier**, en utilisant les commandes appropriées. Plus précisément avec le bouton « **NEW** » (en haut à droite) vous initiez la création de la séquence et, en définissant les mouvements et leurs durées respectives, avec le bouton « **SAVE** » vous accédez à une boîte de dialogue dans laquelle vous nommez le fichier contenant la séquence et choisissez son emplacement sur le disque dur (voir la figure 6).

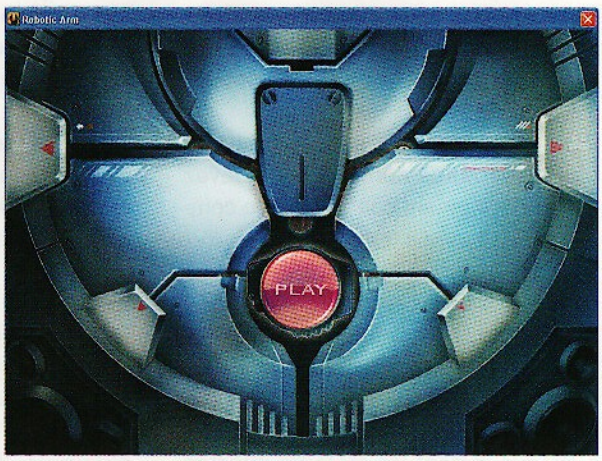


Figure 3 : l'écran de démarrage du programme, commencez en cliquant sur le bouton « PLAY ».

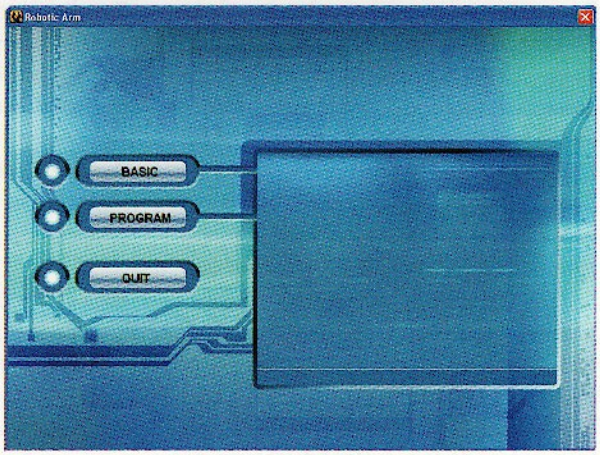


Figure 4 : avec le bouton « BASIC » vous entrez en mode de commande manuelle et avec le bouton « PROGRAM » vous accédez à la création de séquences de commandes.

Figure 5 : le menu du contrôle manuel à l'aide de la souris ou du clavier.

Pour **exécuter une séquence enregistrée**, vous devez ouvrir le fichier correspondant avec la commande « **OPEN** ». Le volet à droite affiche tous les mouvements contenus dans la séquence avec leurs durées respectives. Si le fichier n'a pas encore été créé, en cliquant sur « **ENTER** » vous mémorisez chaque séquence de mouvements qui apparaît dans le volet de droite.

Chaque séquence est composée d'un segment, dans lequel, pour chaque partie du bras il est possible de définir un seul mouvement.

Par exemple, vous ne pouvez pas faire tourner la base dans le sens horaire et anti-horaire pour le même mouvement.

Pour retourner à la position initiale d'une séquence donnée, vous devez définir un segment contenant une commande opposée à la précédente.

Lorsque vous exécutez la séquence (en cliquant simplement sur le bouton « **START** »), le programme effectue tous les segments dans l'ordre dans lesquels ils ont été créés en montrant à chaque instant les moteurs affectés par les mouvements du bras.

Par exemple, si vous voulez ouvrir la pince (elle est au préalable fermée), vous devez enregistrer la séquence correspondante, puis en créer une nouvelle dans laquelle la pince se ferme. En cliquant sur le bouton « **START** », le programme exécutera tous les mouvements du bras sans pause, la pince reviendra à sa position initiale (fermée).

Etant donné que le robot ne dispose pas de capteurs de position, en exécutant une séquence qui implique des mouvements dans un sens, à un certain moment les motoréducteurs se trouveront en fin de course. Vous pouvez revenir à la position initiale tout simplement en cliquant sur le bouton « **REVIEW** », qui exécute la séquence dans le sens inverse.

Programmation du bras en langage Python

Nous allons maintenant vous faire découvrir un autre moyen de contrôler le bras robotisé à l'aide du langage **Python**. Pour cela vous pouvez utiliser un RaspberryPi en connectant l'interface USB du robot à l'un des ports USB du RaspberryPi.

Vous devez aussi avoir quelques connaissances en langage Python. Reportez-vous éventuellement au numéro 123 et suivants de la revue, nous avons décrit de nombreux exemples de programmes en Python fonctionnant sous RaspberryPi.

Vous devez installer le module « **PyUSB** » sur le RaspberryPi qui est disponible en téléchargement sur notre site dans le sommaire détaillé du numéro 138.

Ce dernier est un module python permettant d'utiliser un périphérique USB avec le langage Python.

Ce module permet de créer un objet (du point de vue programmation) du bras robotisé. L'objet ainsi créé dispose de plusieurs méthodes et options de contrôles.

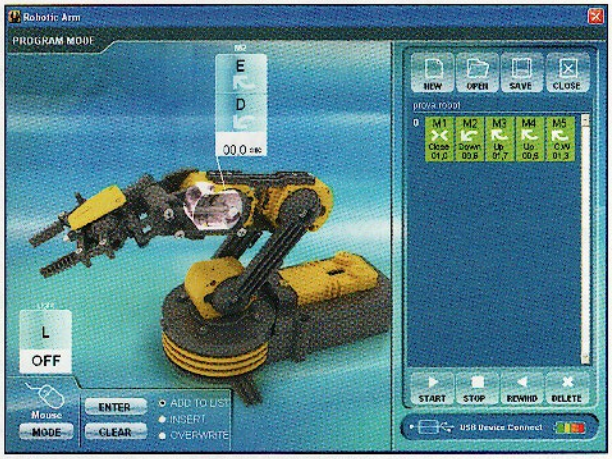


Figure 6 : construction de la séquence des mouvements du bras robotisé.

Ensuite décompressez l'archive « **ksr10.zip** » dans un répertoire de votre choix. Cette archive se trouve dans le dossier que vous venez de télécharger.

Ouvrez votre interpréteur Python et importez le fichier « **ksr10.py** » avec la commande :

```
import ksr10
```

Puis créez un nouvel objet, tel que :

```
ksr = ksr10.ksr10.ksr10_class()
```

Ensuite, les méthodes suivantes s'appliquent à l'objet :

- **ksr.stop()** ; arrête tous les mouvements
- **ksr.lights()** ; allume et éteint la LED
- **ksr.move(part,direction)** ; déplace un segment du bras dans une certain direction
- **ksr.move("shoulder","down")** ; déplace le bras vers le bas.

Un exemple de programme contenant les commandes basiques est visible dans le listing 1. Vous remarquerez en début de programme l'importation du fichier « ksr10 ».

À travers ces exemples de commandes, nous espérons vous avoir apporté quelques éclaircissements concernant le contrôle de ce bras robotisé.

Vous vous rendrez compte du potentiel réel de celui-ci qui au premier abord ressemble à un jouet, mais qui en réalité est bien un robot miniature.

Listing 1 : Exemples de commandes simples pour contrôler le bras robotisé en Python

```
#!/usr/bin/python

import ksr10
import time

# crée un objet
ksr = ksr10.ksr10.ksr10_class()

# allume la LED
ksr.lights()

# rotation de la base vers la gauche
ksr.move("base","left")

# attend ½ seconde
time.sleep(.5)

# arrête tous les mouvements
ksr.stop()

# rotation de la base vers la droite et
# levage du bras simultanément

ksr.move("base","right")
ksr.move("elbow","up")

# attend ½ seconde
time.sleep(.5)

# arrête tous les mouvements
ksr.stop()

# éteint la LED
ksr.lights()
```