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

ELECTRONIQUE

ELECTRONIQUE

ET LOISIRS

magazine

http://www.electroniquemagazine.com

n°128 AUTOMNE 2014

Carte d'extension A/D 16 bits pour RaspberryPi compatible Arduino

COURS ANDROID 1ère partie

- Emetteur 4 canaux codé sur 9 digits
- 3DRAG les circuits imprimés suite
- Testeur de servomoteur pour modélisme
- « Tracker » motorisé pour panneaux solaires
- Alimentation de laboratoire symétrique contrôlée par PIC

N° 128 Septembre 2014

M 04662 - 128 - F: 7,50 € - RD

France 7,50 € - DOM 7,50 € - CE 8,00 € - NC (S)900FP - Poly.fr 1200CFP - Canada 11,50 $C

Sommaire

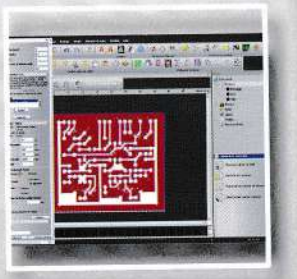
ARTICLES

Numéro 128
Automne 2014
100 pages

04 IMPRIMANTE 3D

RÉALISEZ LES CIRCUITS AVEC LA 3DRAG DE ARTCAM À REPETIER-HOST

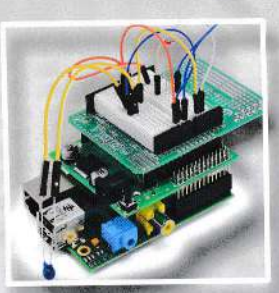
Dans cet article nous allons vous expliquer le cheminement pour passer d'un fichier image de type BMP à un fichier interprétable directement par Repetier-Host à l'aide du logiciel ArtCam. En d'autres termes vous serez capable à partir d'un fichier image (.bmp) de fabriquer directement avec la 3DRAG un PCB.



15 INFORMATIQUE

CARTE D'EXTENSION A/D 16 BITS POUR RASPBERRYPI COMPATIBLE ARDUINO

Cette carte d'extension est un convertisseur analogique-numérique d'une résolution de 16 bits. Elle étend les fonctionnalités du RaspberryPi et elle est compatible broche à broche avec la carte Arduino.



31 ALIMENTATION

ALIMENTATION DE LABORATOIRE CONTRÔLÉE PAR UN PIC16F876A

Cette alimentation professionnelle de laboratoire est capable de fournir une tension réglable de 0 à 25 V et un courant de sortie réglable de 0 à 2,5 A. Elle est gérée par un PIC 16F876A qui régule la tension et le courant, contrôle la température de l'étage de sortie et le désactive en cas de surchauffe. Elle dispose d'un afficheur LCD et peut être gérée à partir d'un PC via le port série .

46 MAISON

« TRACKER » OU SUIVEUR DE SOLEIL MOTORISÉ POUR PANNEAUX SOLAIRES

Un « tracker » solaire ou suiveur de Soleil est une installation de production d'énergie solaire utilisant le principe de l'héliostat. C'est une structure qui permet d'orienter des panneaux solaires en fonction de la position du soleil. Ce système oriente le panneau solaire tout au long de la journée vers la lumière du soleil de manière optimale, assurant ainsi le meilleur rendement possible.

61 MODELISME
TESTEUR DE SERVOMOTEURS POUR MODÉLISME

Ce montage permet de vérifier l'efficacité des servomoteurs analogiques pour modélisme. Grâce au programme, dont nous **publions le code source complet en BASIC**, ce montage permet de tester les servomoteurs afin d'identifier facilement le « neutre » et contrôler le temps de réponse ainsi que le bon fonctionnement des engrenages du « servo ».

74 TELECOMMANDE
ÉMETTEUR 4 CANAUX CODÉ SUR 9 DIGITS

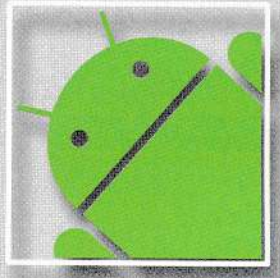
Dans cet article, nous vous proposons de réaliser un émetteur pour radiocommande avec 4 canaux indépendants et codifiables individuellement, facile à mettre en œuvre et de faible coût. Ce montage utilise un microcontrôleur Microchip et met en œuvre la génération d'un code à 9 digits de type « three-state » (3 états). Il fonctionne dans la bande de fréquence 433,92 MHz.

84 AUTOMATISME
TIMER PROGRAMMABLE DE 1 SECONDE À 60 HEURES

Ce montage est un temporisateur universel dont la sortie sur relais est activée soit de manière cyclique, soit à l'aide d'une simple impulsion avec une durée d'intervalle de temps ON et OFF réglable de 1 seconde à 60 heures.

89 COURS
PROGRAMMEZ AVEC ANDROID PREMIÈRE PARTIE

A travers cette série de plusieurs articles, nous allons vous apprendre à développer des applications pour smartphones, tablettes et systèmes « embarqués » sous la plate-forme Android. Cela vous permettra d'interfacer avec le monde extérieur vos projets électroniques.



L'EDITO AUTOMNE 2014

Chères lectrices, chers lecteurs,

c'est en cette rentrée scolaire que la Rédaction d'Electronique et Loisirs Magazine vous propose un nouveau Cours sur la Programmation Android. Celui-ci vous permettra de développer des applications sur tout type d'appareils (Smartphone, Tablette, RaspberryPi, etc.) fonctionnant sous Android. Pour compléter le cours nous vous proposons à partir de ce numéro une série d'interfaces pour le RaspberryPi, dont la première est une carte d'extension A/D 16 bits compatible Arduino. Elle vous permettra de faire communiquer le RaspberryPi avec l'Arduino. D'autre part nous continuons à développer l'imprimante 3DRAG pour réaliser des circuits imprimés à partir de fichiers images (.bmp, .jpg). Nous n'avons pas oublié les lecteurs qui désirent équiper leur laboratoire avec l'alimentation pilotée et ceux qui sont férus de modélisme avec le testeur de servomoteur. Enfin pour ceux qui désirent réaliser des économies d'énergie, nous leur proposons un système de production d'énergie à base de panneaux solaires avec un rendement exceptionnel car il a la particularité de suivre le Soleil en fonction de sa position tout au long de la journée.

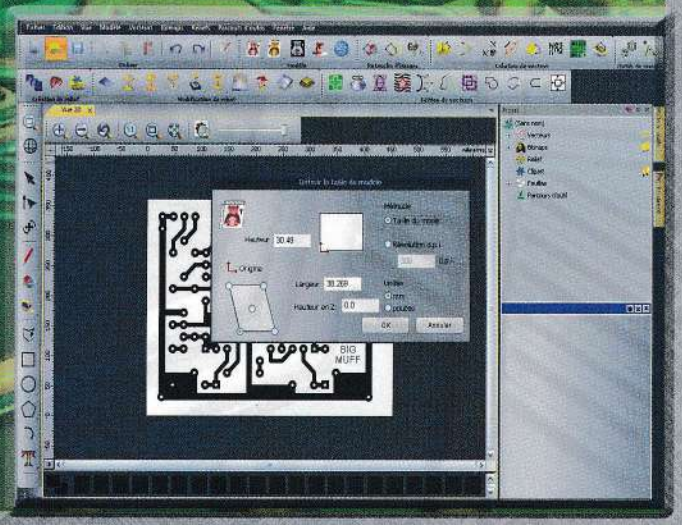
Les typons des circuits imprimés et les programmes **lorsqu'ils sont libres de droits** sont téléchargeables à l'adresse : **www.electroniquemagazine.com** dans le sommaire de la revue 128 et à l'onglet « Télécharger »

Transformez la 3DRAG pour réaliser des circuits imprimés

De ArtCam à Repetier-Host
Deuxième partie

De la Rédaction

Dans le précèdent numéro 127 d'Electronique et Loisirs Magazine, nous vous avons décrit comment transformer l'imprimante 3DRAG en une fraiseuse à commande numérique (CNC) pour réaliser des circuits imprimés sans résine photosensible, sans développement et sans bain d'acide, en envoyant tout simplement au contrôleur le fichier G-Code obtenu à partir du fichier GERBER généré par le logiciel Cadsoft Eagle. Dans ce numéro nous allons vous expliquer le cheminement pour passer d'un fichier image de type BMP à un fichier interprétable directement par Repetier-Host à l'aide du logiciel ArtCam. En d'autres termes vous serez capable à partir d'un fichier image (.bmp) de fabriquer directement avec la 3DRAG un PCB.



Le succès de la **3DRAG** est maintenant incontestable et est connu de nombreux lecteurs que la Rédaction tient à remercier. Nous avons écrit de nombreux articles qui ont démontré les capacités de cette imprimante en 3D et ainsi qu'en CNC. Beaucoup de nos lecteurs ont consacré du temps à l'amélioration de la 3DRAG, et plusieurs d'entre eux nous ont **soumis l'idée de graver directement avec la 3DRAG un circuit imprimé à partir d'un fichier image de type BMP**.

La 3DRAG est une imprimante très polyvalente, elle peut être transformée en une machine de fraisage (CNC), et nous nous sommes demandés à la Rédaction s'il était possible de fabriquer directement un circuit imprimé à partir d'un fichier « . jpg » ou « .bmp ».

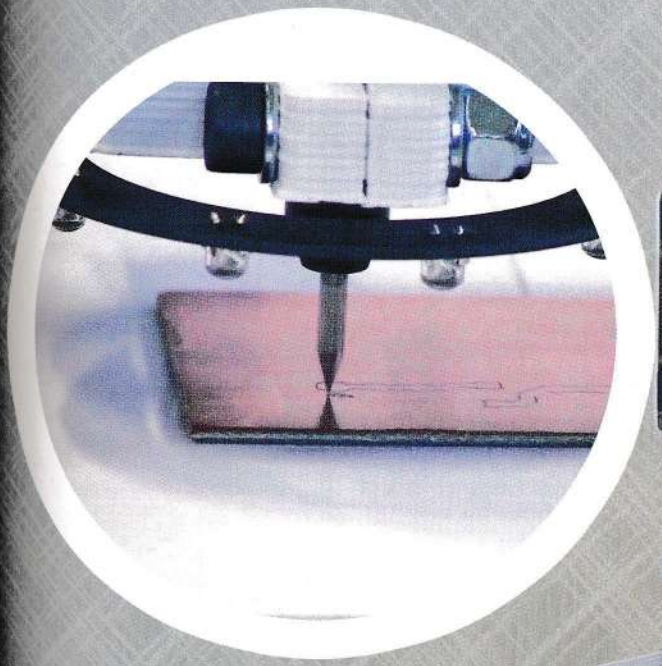
En effet l'électronique est devenue très importante de nos jours, car la plupart des appareils qui fonctionnent grâce à l'électricité disposent de composants ou d'un circuit intégré. Pour dessiner un circuit imprimé sur un ordinateur nous avons opté pour le logiciel « **DIY Layout Creator** » qui est un outil idéal

pour dessiner des circuits de toute sorte de manière intuitive car il possède une interface très simple qui dispose de tous les outils nécessaires pour pouvoir ajouter des éléments en quelques secondes. Le programme permet d'ajouter des circuits intégrés, des commutateurs, des résistances, des condensateurs, des diodes ou des transistors, etc. Une fois le dessin du circuit terminé, vous pouvez le sauvegarder comme un fichier image ou l'imprimer facilement.

Si vous êtes amateur d'électronique et que vous cherchez un programme qui puisse vous aider à créer des circuits de la manière la plus simple et intuitive possible, téléchargez « **DIY Layout Creator** » en version complète et totalement gratuite à l'adresse suivante :

https://code.google.com/p/diy-layout-creator/downloads/list.

Nous mettrons aussi le programme en téléchargement sur notre site **www.3dprint.electroniquemagazine.com**.



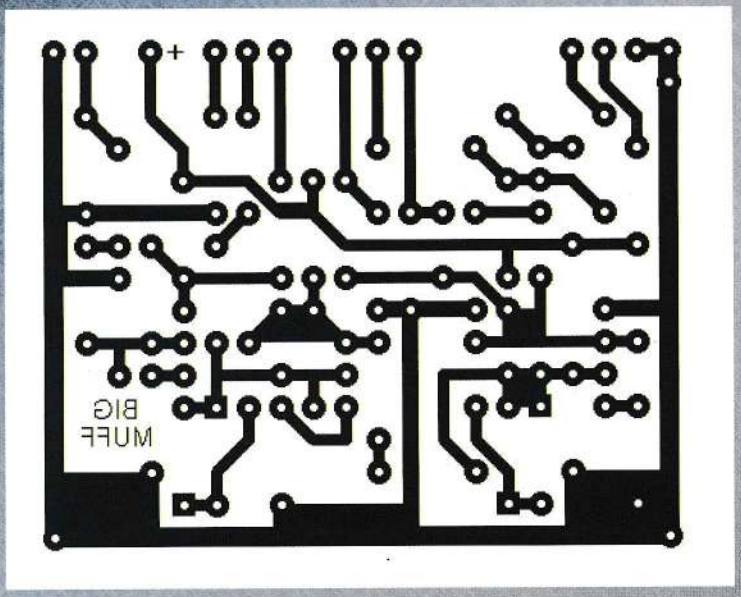


Figure 1 : ici nous apercevons le fichier image au format « .bmp» du circuit que nous devons graver, l'image est vue du « côté composants ».

découp
« Ajust
de cou
la sou

Ensuit
face d
droit d
suiven

Juste a
la sou
« Piste
tard. M
cliquar
le bas.

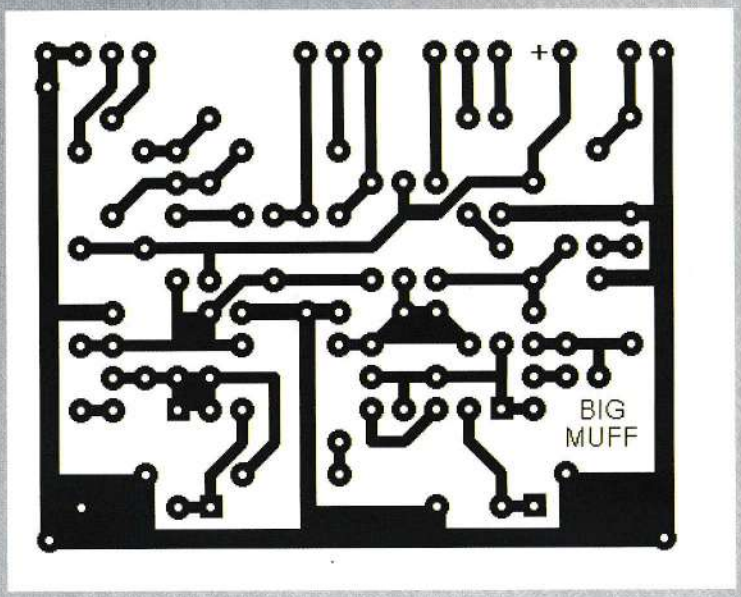


Figure 2 : pour être en mesure de fabriquer le circuit correctement sur une plaque de cuivre, nous devons faire un « miroir » du circuit en utilisant le programme « Paint » de Windows.

Malheureusement, nous n'avons pas la possibilité d'obtenir un fichier GERBER à partir de « DIY Layout Creator », et importer un fichier image dans Cadsoft Eagle pourrait devenir un véritable cauchemar. L'idéal à ce stade est d'utiliser un logiciel de « CAM » qui gère directement le fichier d'image. Il existe de nombreuses alternatives Open Source telles que « CamBam » (avec certaines limites), Freemill (un des meilleurs logiciels), G-simple, etc.

Mais le logiciel que nous avons retenu est « **ArtCam** », il est disponible en version d'évaluation avec laquelle vous pourrez tester la plupart des fonctions du logiciel. Il ne nécessite pas d'étapes intermédiaires, il accepte des fichiers d'importations dans pratiquement n'importe quel format et vous permet de réaliser la gravure de PCB, de dessins et de lettres.

Cependant rappelez-vous qu'il ne faut pas trop exagérer avec la 3DRAG, car elle ne supporte pas très bien les travaux sévères. Sa structure est robuste, mais elle n'est pas une vraie machine CNC.

La première étape consiste à importer votre dessin au format « .bmp » dans ArtCam, dans notre cas nous avons utilisé le dessin (circuit) nommé « BIG MUFF » visible sur la figure 1. « BIG MUFF » est une pédale d'effets conçue par la société Electro Harmonix ou EHX.

Les dimensions du circuit sont de 50 mm par 61,11 mm, l'image est vue du « côté composants », pour être en mesure de fabriquer le circuit correctement sur une plaque de cuivre, nous devons faire un « miroir » du circuit en utilisant le programme « Paint » de Windows. Ainsi l'image se trouve dans la bonne position pour la gravure, c'est-à-dire du côté cuivre comme vous pouvez le voir en figure 2.

Maintenant nous devons importer dans Artcam l'image. Cliquons sur le menu « Fichier » puis « Ouvrir », et sélectionnons « BIGMUFF.bmp », au bout de quelques secondes la figure 3 apparaît, définissons la taille réelle du dessin en cliquant sur « Ok ». Dessinons maintenant le bord en prenant la forme carrée sur le côté gauche (qui servira aussi à la

Figur
nons
et ch
avec
Ensu
qui s
sur l'
côté

découpe du PCB) et dans le menu « Vecteurs » sélectionnons « Ajuster les vecteurs aux frontières », et choisissons l'option de couleur qui nous plaît en cliquant avec le bouton droit de la souris sur le vecteur.

Ensuite sur la droite, cliquons sur l'ampoule qui se situe en face du menu « Bitmaps » (voir la flèche rouge sur le côté droit de la figure 4), nous voyons apparaître des lignes qui suivent le dessin.

Juste au-dessus de la flèche rouge, faisons un clic droit avec la souris sur « Vecteurs », et créons les nouveaux vecteurs « Piste », « Bord » et « Perçage » que nous utiliserons plus tard. Maintenant sélectionnons le vecteur « Bord », puis en cliquant avec le bouton droit de la souris déplaçons le vers le bas.

Faisons de même avec le vecteur « Piste » en sélectionnant toutes les pistes avec la touche « Shift », avec un clic droit déplaçons le vecteur. Enfin nous faisons de même avec le vecteur « Perçage » en sélectionnant cette fois tous les trous.

Nous pouvons supprimer des vecteurs inutiles et arranger ceux qui sont mal sortis. A l'aide des « Ampoules » faisons apparaître uniquement les vecteurs « Bord » et « Pistes » sur lesquels nous voulons effectuer des modifications. Après un long travail nous obtenons le résultat de la figure 5.

Choisissons le menu en haut à droite « Parcours d'outils », nous avons donc deux solutions, la première est de créer un profil sur les pistes et la seconde une gravure.

- 1er cas : dans le menu « Parcours d'outils » sélectionnons

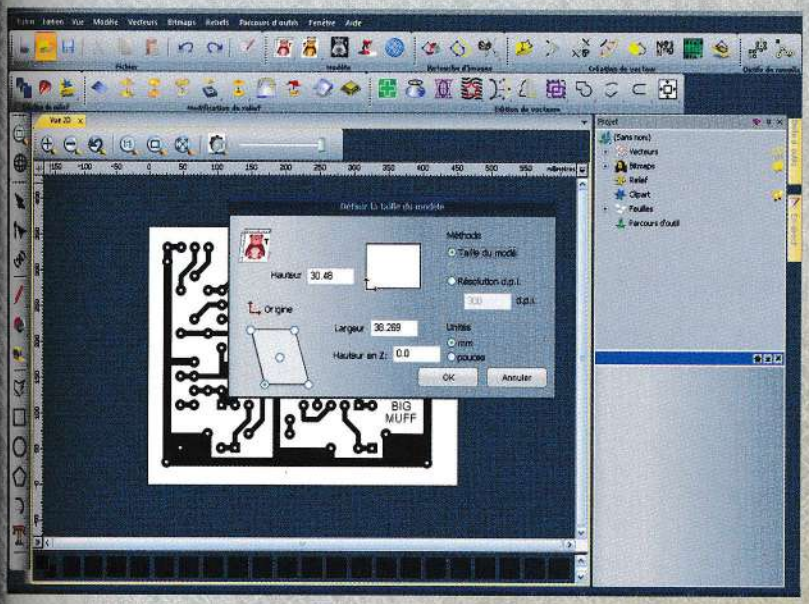


Figure 3 : pour importer l'image dans «Art-Cam» , cliquons sur le menu « Fichier » puis « Ouvrir », sélectionnons « BIGMUFF.bmp », au bout de quelques secondes l'écran ci-contre apparaît.

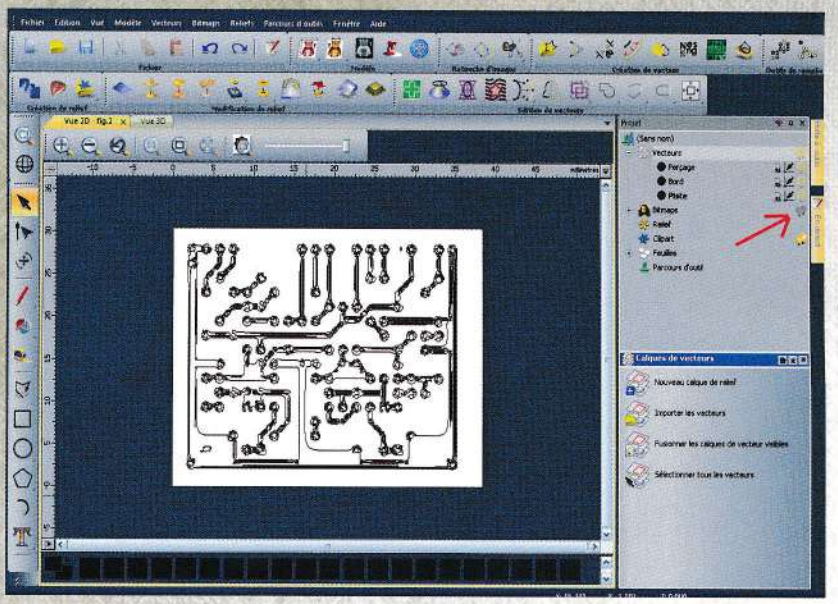


Figure 4 : dans le menu « Vecteurs » sélectionnons « Ajuster les vecteurs aux frontières », et choisissons l'option de couleur en cliquant avec le bouton droit de la souris sur le vecteur. Ensuite sur la droite, cliquons sur l'ampoule qui se situe en face du menu « Bitmaps » (voir sur l'image ci-contre la flèche rouge sur le côté droit).

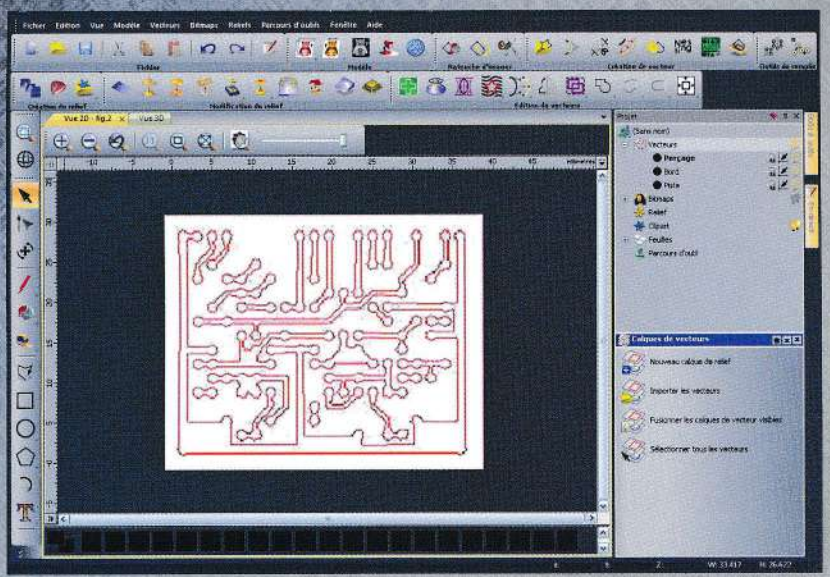


Figure 5 : à l'aide des « Ampoules » faisons apparaître uniquement les vecteurs « Bord » et « Pistes » sur lesquels nous voulons effectuer des modifications. Après un long travail nous obtenons le résultat ci-contre.

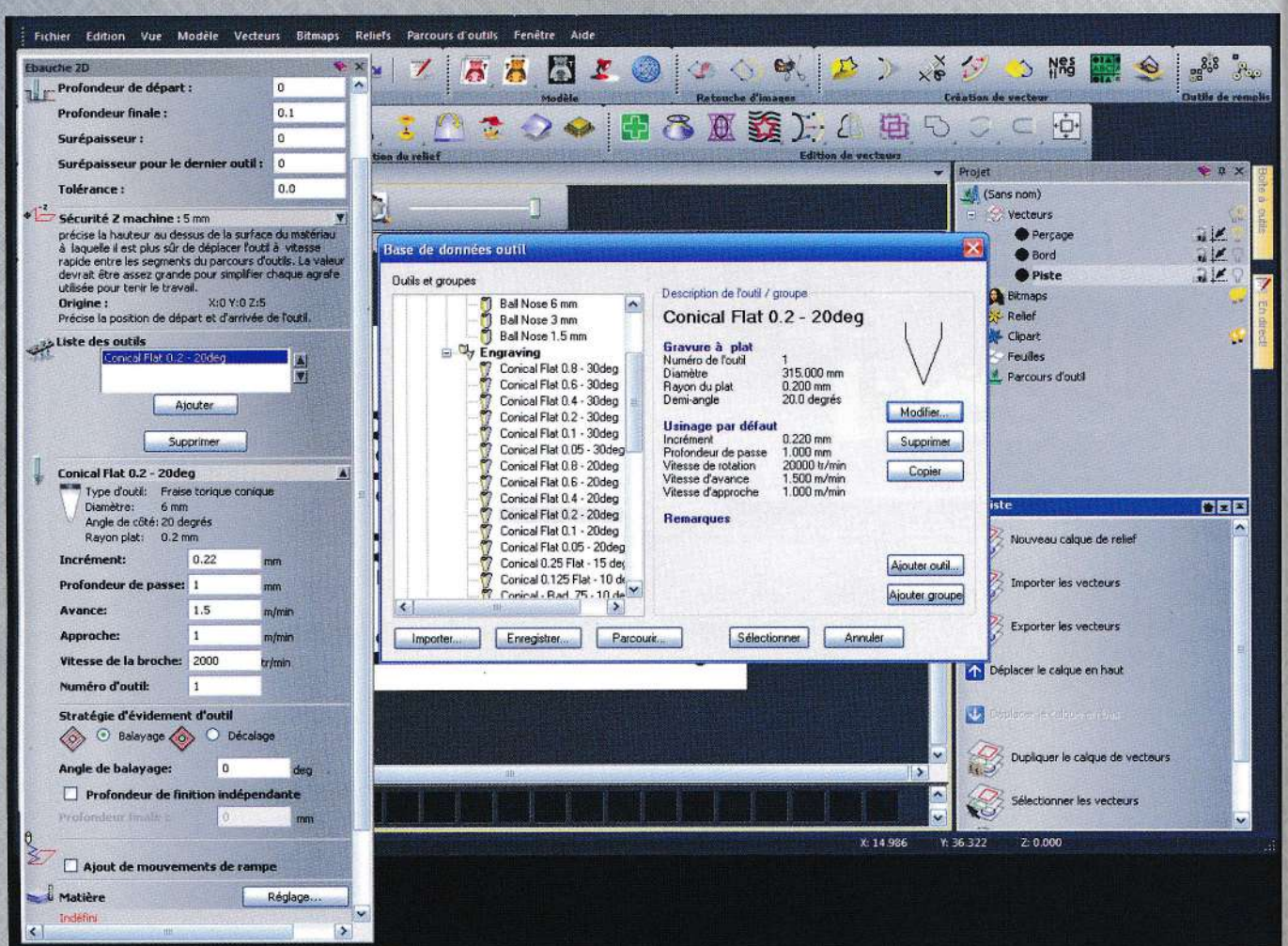


Figure 6 : 1er cas : dans le menu « Parcours d'outils » sélectionnons « Nouveau parcours d'outils 2D » puis « Ebauche 2D », insérons les paramètres suivants : profondeur finale 0,1, dans liste des outils cliquons sur « Ajouter » et dans « Metric Tools » puis « Engraving », sélectionnons « Conical Flat 0.2 – 20 deg ».

« Nouveau parcours d'outils 2D » puis « Ebauche 2D », insérons les paramètres suivants : profondeur finale 0,1, dans liste des outils cliquons sur « Ajouter » et dans « Metric Tools » puis « Engraving », sélectionnons « Conical Flat 0.2 – 20 deg » (voir la figure 6), ce qui correspond à une pointe dont le rayon du plat est de 0,2 mm et le demi-angle de 20 degrés. A la figure 7 vous pouvez voir le détail des paramètres de l'outil, vous pouvez aussi personnaliser le menu selon vos besoins.

Cliquons en bas sur l'onglet « Calculer maintenant », les vecteurs doivent être coloriés en rose, s'ils sont rouges cela veut dire qu'il existe des superpositions et que vous devez les éviter.

Nous obtenons le résultat de la figure 8. Tout le cuivre est éliminé pour ne laisser que les pistes. Nous pouvons procéder maintenant à la simulation, avec un clic droit de la souris sélectionnons « vidage » et « Simuler les parcours d'outils », nous obtenons le résultat de la figue 11.

- 2eme cas : dans le menu « Parcours d'outils » sélectionnons « Nouveau parcours d'outils 2D » puis « Profil », insérons les paramètres suivants en ayant au préalable sélectionné le vecteur « Piste » : dans « Profondeur de coupe » réglons « Profondeur de départ » à 0.1, « Profondeur finale » à 0.1 et « Tolérance » à 0.001. Nous obtenons la représentation de la figure 10, et la simulation nous donne le résultat de la figure 13.

Maintenant nous allons nous occuper du perçage, cachons le vecteur « Piste » à l'aide de « l'ampoule » sur le côté pour plus de commodité, nous constatons que seuls les trous réapparaissent.

Sélectionnons tous les trous qui deviennent de couleur rose et dans le menu « Parcours d'outils » sélectionnons « Nouveau parcours d'outils 2D » puis « Perçage », la fenêtre de la figure 12 apparaît dans laquelle nous allons ajuster les paramètres.

Dans notre cas nous choisissons un foret de 0,8 mm de diamètre et une profondeur finale de 1,6 mm. Aidez-vous de la figure 12 pour régler les autres paramètres. Ainsi, si nous effectuons une simulation dans le cas de l'ébauche ou du profil, nous voyons apparaître les trous comme sur la figure 13.

Maintenant nous allons couper le circuit imprimé (PCB), sélectionnez « Parcours d'outils » puis « Profil » et l'outil « End mill 1.5 mm », dans notre cas l'outil de coupe doit avoir un diamètre de 1 mm.

Donc, nous allons couper le bord du circuit selon nos besoins. Nous adaptons les réglages suivants de l'outil : « Diamètre », « Avance », « Approche » et « Vitesse de la broche » que vous devrez modifier selon vos besoins et en effectuant des essais.

Maintenant que tout est correct, sauvegardons le « Parcours d'outils ».

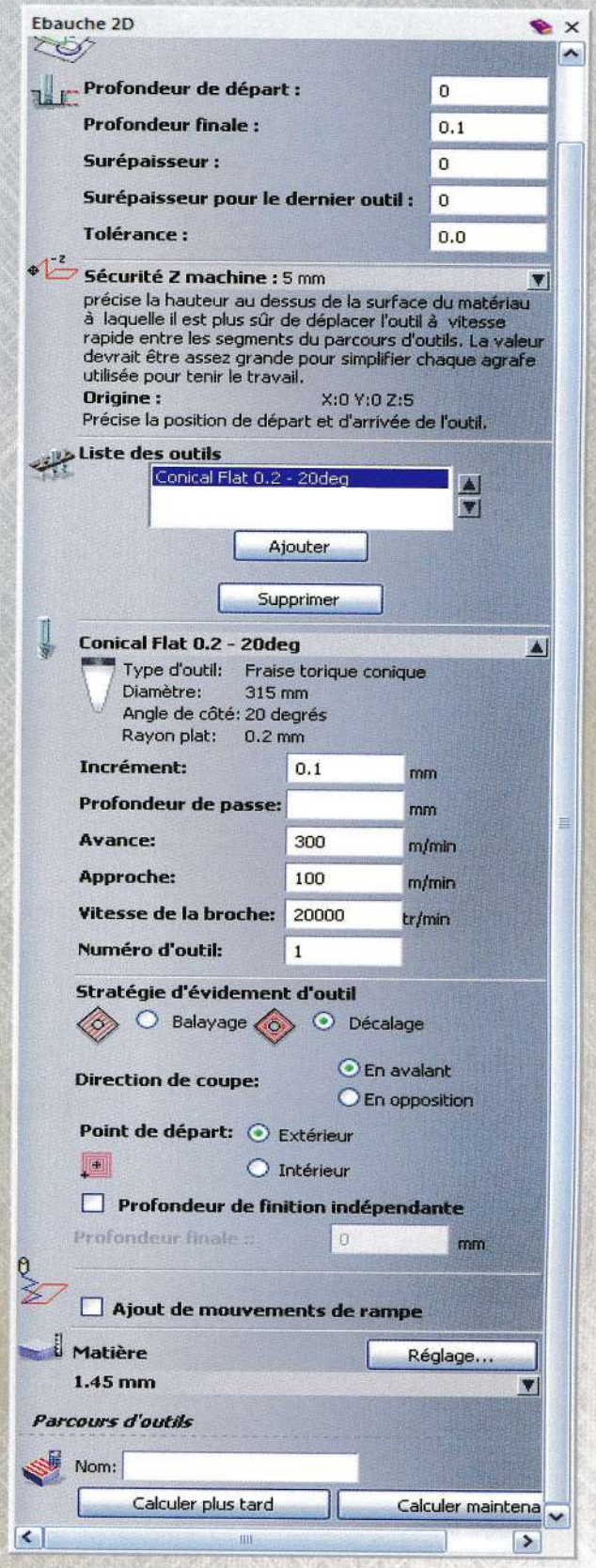


Figure 7 : détail des paramètres de l'outil, vous pouvez aussi personnaliser le menu selon vos besoins.

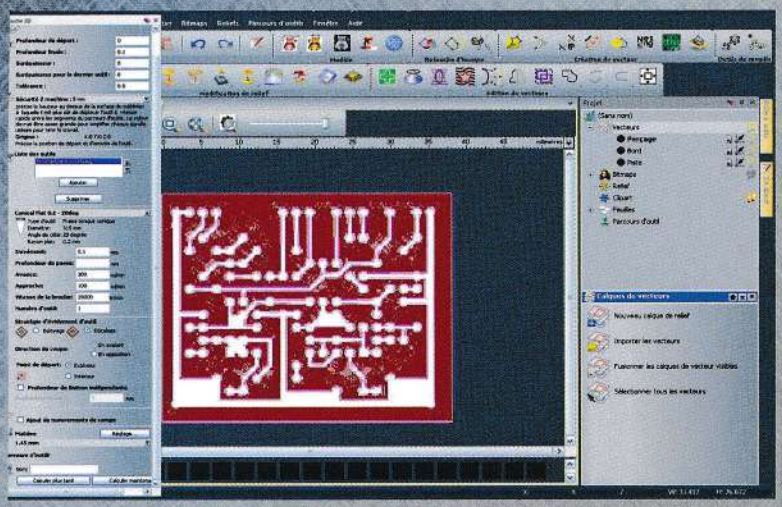


Figure 8 : en cliquant en bas sur l'onglet « Calculer maintenant », les vecteurs doivent être coloriés en rose, s'ils sont rouges cela veut dire qu'il existe des superpositions et que vous devez les éviter. Nous obtenons le résultat de la figure ci-contre.

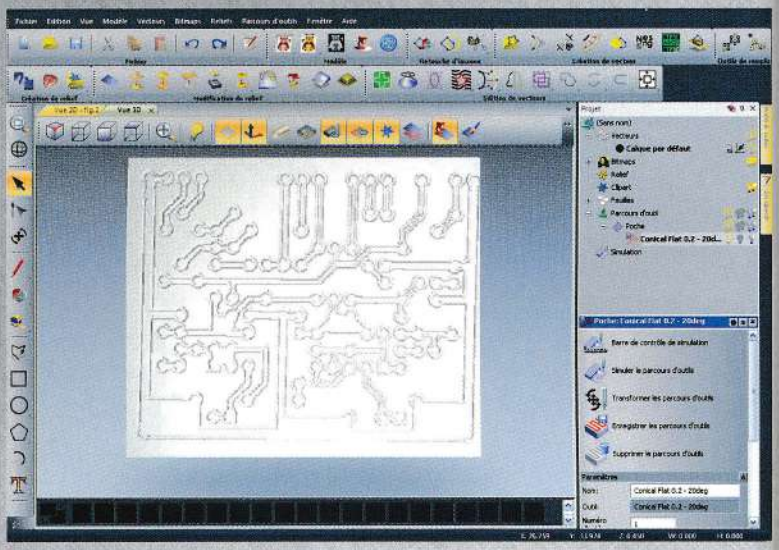


Figure 11 : en procédant à la simulation, avec un clic droit de la souris sélectionnons « vidage » et « Simuler les parcours d'outils », nous obtenons le résultat ci-contre.

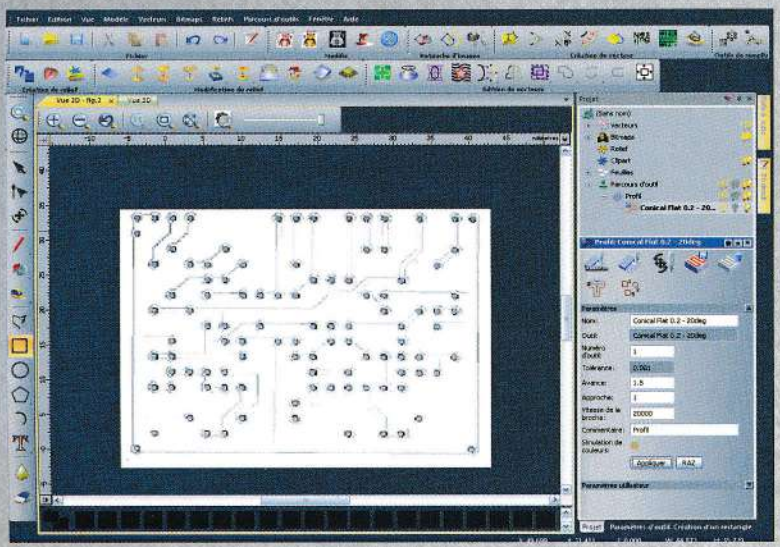


Figure 13 : dans le cas d'une simulation de l'ébauche ainsi que dans le cas d'une simulation du profil, nous voyons apparaître les trous comme sur la figure ci-contre.

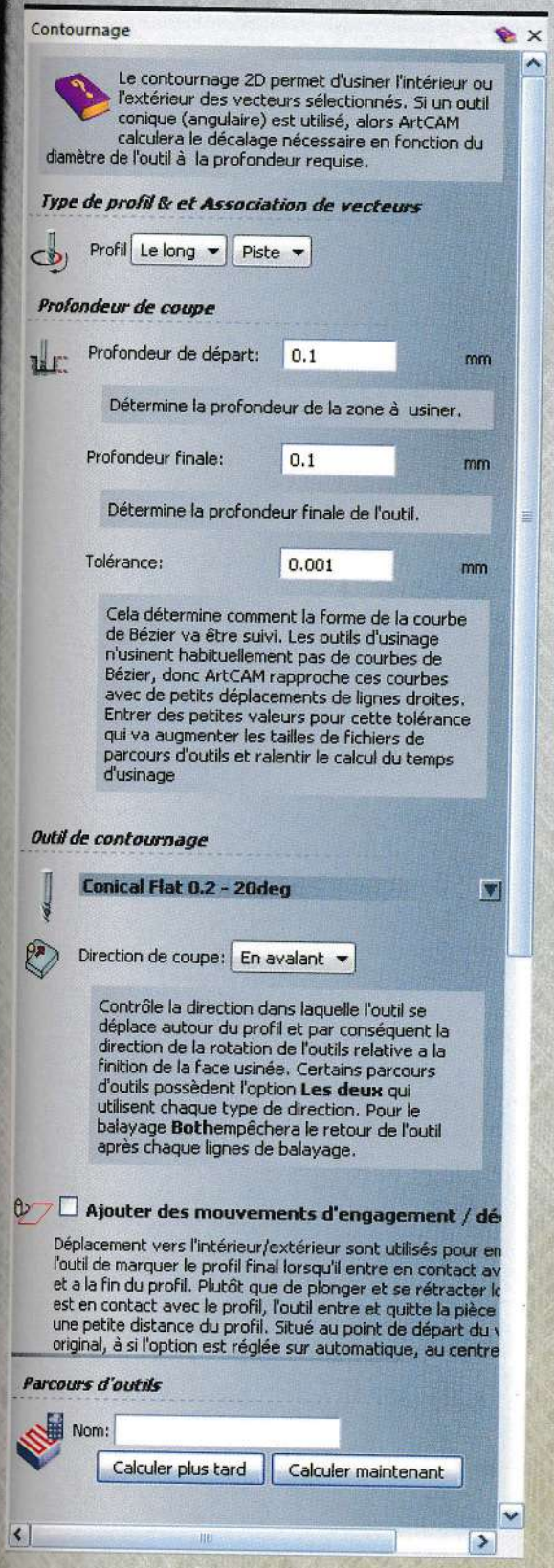


Figure 10 : 2eme cas : dans le menu « Parcours d'outils » sélectionnez « Nouveau parcours d'outils 2D » puis « Profil », insérez les paramètres visibles sur la figure ci-dessus en ayant au préalable sélectionné le vecteur « Piste ». Dans « Profondeur de coupe » réglez « Profondeur de départ » à 0.1, « Profondeur finale » à 0.1 et « Tolérance » à 0.001.

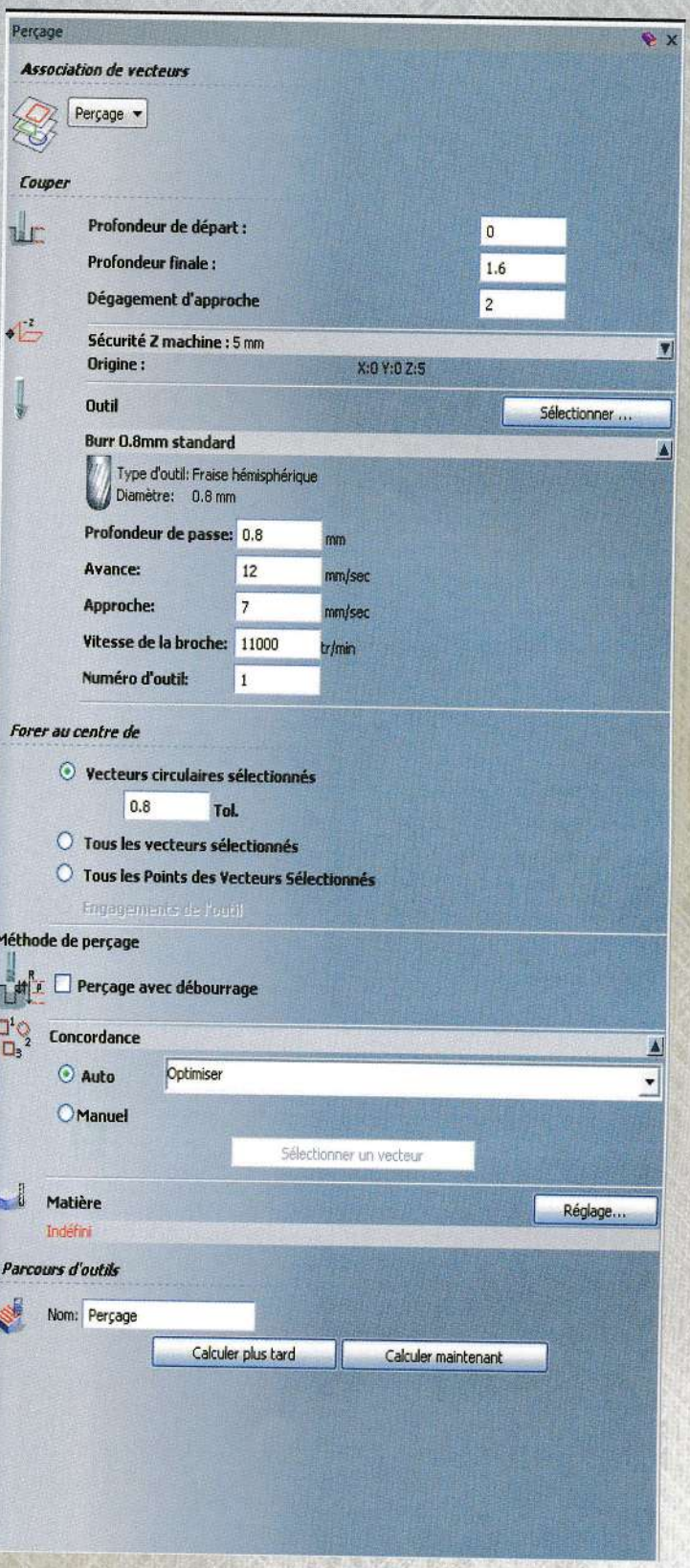


Figure 12 : dans le menu « Parcours d'outils » sélectionnez « Nouveau parcours d'outils 2D » puis « Perçage », la fenêtre ci-dessus apparaît dans laquelle vous ajustez les paramètres du foret à 0,8 mm de diamètre et une profondeur finale de 1,6 mm.

Nous obtenons un fichier de sortie (*.nc) compatible avec « Repetier-Host ». Nous enregistrons chaque fichier séparément, rappelons-nous que nous devons changer d'outil pour fraiser (détourer), percer et découper.

Ouvrons « Repetier-Host » et cliquons sur « Réglages imprimante » (dans le coin supérieur droit indiqué par la flèche rouge). Une fenêtre s'ouvre et cliquons sur l'onglet « Format imprimante », dans la liste déroulante « Printer Type » sélectionnons « Routeur CNC » et réglons les paramètres comme indiqués sur la figure 15.

Pour visualiser correctement la gravure et le parcours de l'outil, nous devons cocher « Afficher Filament » et « Afficher déplacements » (voir la figure 16).

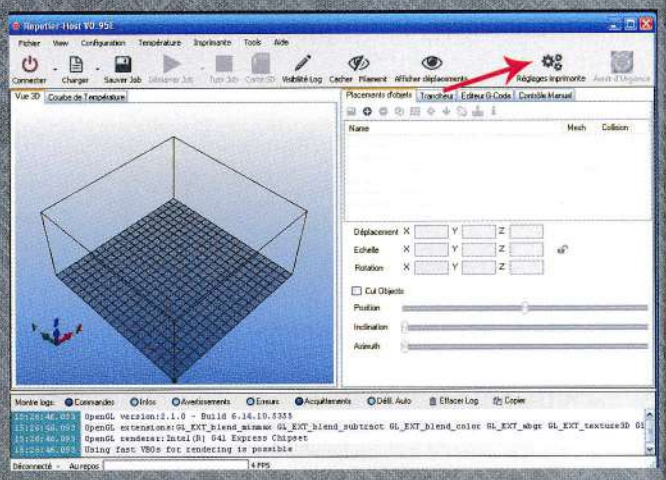


Figure 14 : ouvrez « Repetier-Host » et cliquez sur « Réglages imprimante » dans le coin supérieur droit indiqué par la flèche rouge.

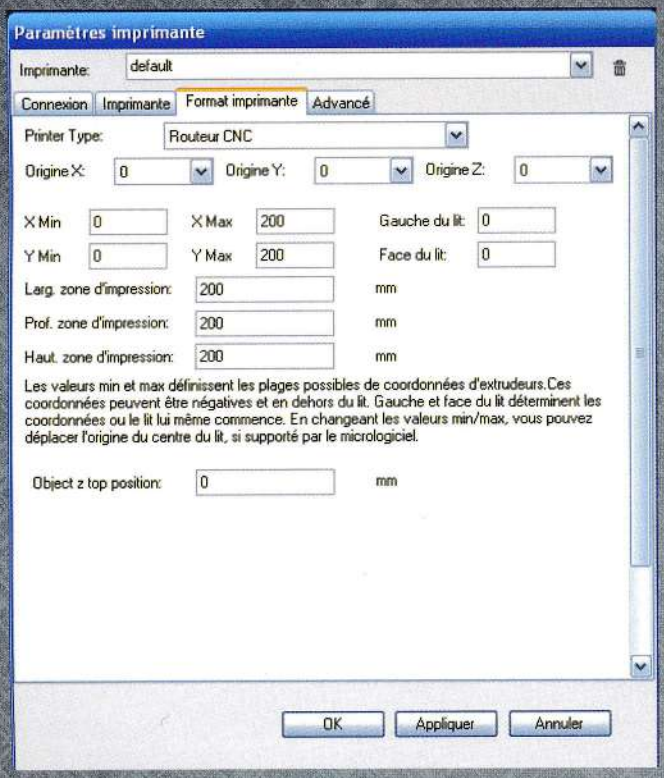


Figure 15 : cliquez sur l'onglet « Format imprimante », dans la liste déroulante « Printer Type » sélectionnez « Routeur CNC » et réglez les paramètres comme indiqué ci-dessus.

Figure 16 : pour visualiser correctement la gravure et le parcours de l'outil, vous devez cocher « Afficher Filament » et « Afficher déplacements » (cercle rouge).

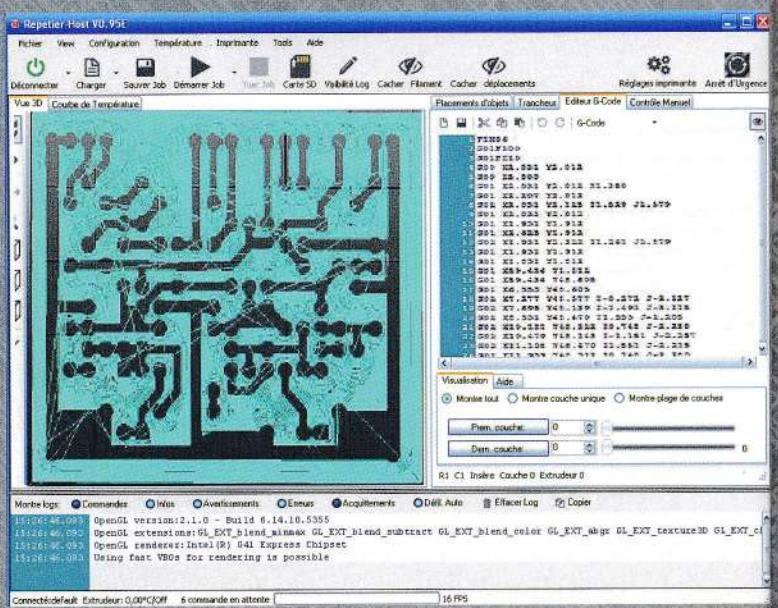


Figure 17 : en utilisant le bouton « Charger » en haut sur la gauche, vous chargez le fichier G-code qui vous intéresse.

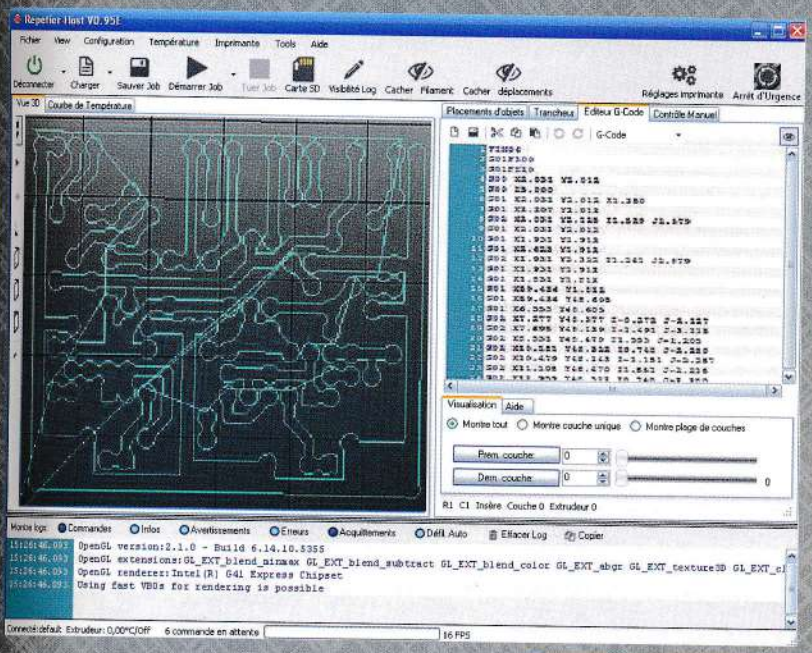


Figure 18 : ici vous voyez le « profilage du PCB.

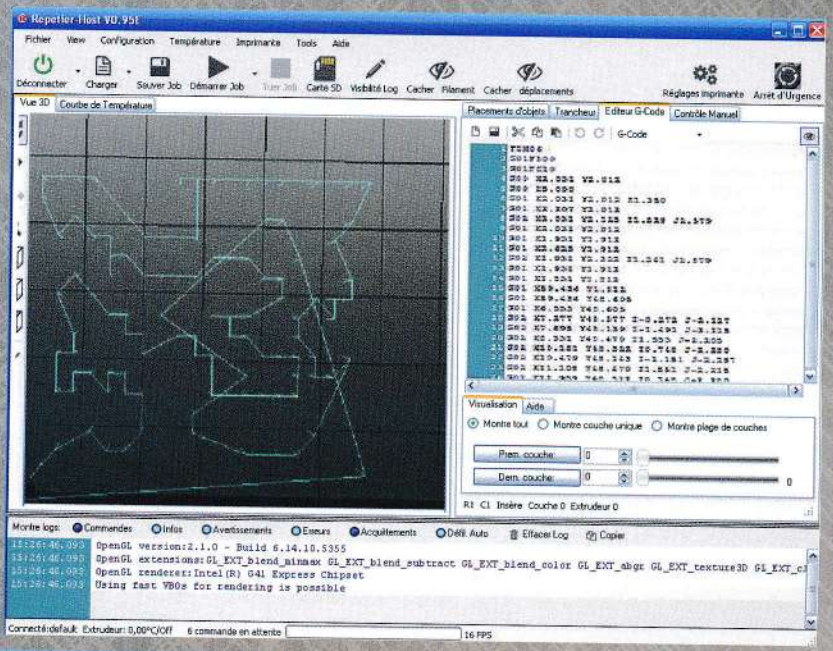


Figure 19 : ci-contre le « perçage du PCB »

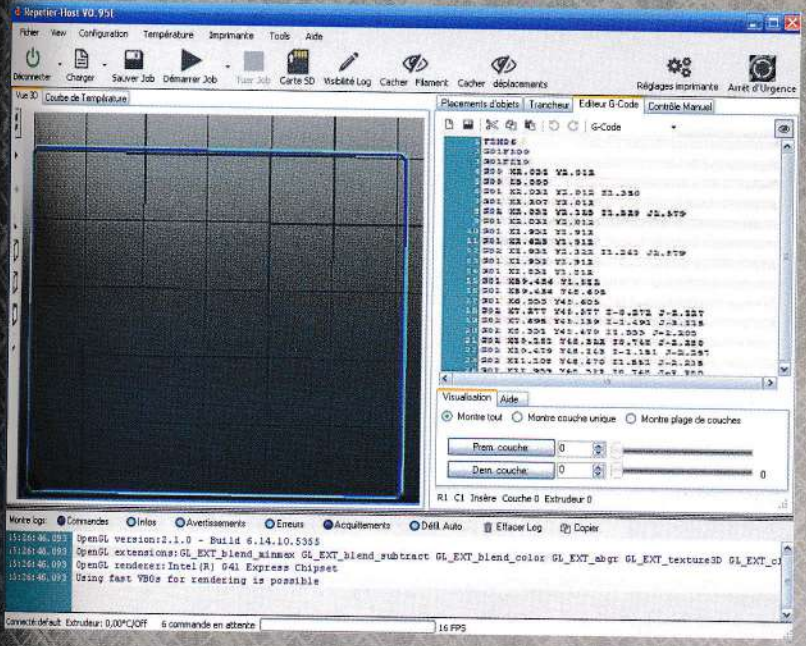


Figure 20 : et pour terminer ci-contre la découpe du circuit imprimé.

Le site de référence :

www. 3dprint.electroniquemagazine.com

Afin d'apporter un support à ceux qui ont commencé l'aventure ou qui sont sur le point d'acheter une imprimante **3DRAG**, mais aussi pour tous les utilisateurs de la communauté **RepRap**, nous avons créé un site internet dédié uniquement à l'imprimante **3D**.

Ce site est un espace où tous ceux qui souhaitent entreprendre la construction de notre imprimante peuvent trouver les informations nécessaires à la construction, au choix des programmes et les configurations optimales.

Dans les prochains mois, nous améliorerons et développerons le site avec notamment la **nouvelle version 1.2** de la **3DRAG**, tout en offrant un certain nombre de modèles prêts à être « imprimés » en téléchargement gratuit, et des détails relatifs à la configuration de votre imprimante.

Les modèles que vous pouvez trouver sur Internet, en fait, sont au format **STL** et doivent être « découpés » par des programmes appropriés. En gérant les paramètres et les caractéristiques de l'imprimante, il suffit de créer un fichier contenant les instructions **G-Code** qui, couche par couche, impriment l'objet.

Sur notre site vous trouverez des fichiers **G-Code de modèles testés** et basés sur notre expérience ainsi que sur les tests d'impression effectués avec la **3DRAG**. Les fichiers **G-Code** sont prêts à être imprimés avec les meilleurs réglages possibles pour obtenir un résultat parfait.

En utilisant les fichiers **G-Code**, vous pouvez lancer immédiatement l'impression, épargner le temps de découpage, et aussi éviter d'avoir de mauvaises surprises à cause de certains paramètres qui ne sont pas définis de manière optimale.

Le site : www.3dprint.electroniquemagazine.com

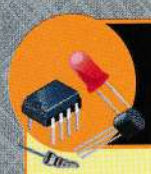
Les programmes ArtCam et DIY Layout sont disponibles en version d'évaluation à l'onglet « Logiciels ».

Le site comprend également une section consacrée au firmware de l'imprimante et au logiciel pour l'impression 3D, au fur et à mesure nous améliorerons les programmes en proposant des mises à jour.

NB : Attention la **3DRAG** est vendue uniquement par la société **COMELEC**, bien qu'elle utilise une mécanique commune avec la K8200 de Velleman, elle est très différente au niveau du firmware. Les modifications présentées dans cet article pour **fabriquer les circuits imprimés sans produits chimiques à partir d'un fichier GERBER** sont **uniquement applicables** à la **3DRAG**.

Maintenant, en utilisant le bouton « Charger » en haut sur la gauche, nous allons charger le fichier G-code qui nous intéresse (voir la figure 17). Sur la figure 18 nous avons chargé le « profilage du PCB », sur la figure 19 le « perçage du PCB » et sur la figure 20 la découpe du circuit imprimé.

Nous arrivons à la fin de la description du procédé du passage d'un fichier image à un fichier de type Repetier-Host. Sur notre site vous pourrez télécharger une **version d'évaluation** d'**ArtCam** ainsi que de « **DIY Layout** ». Ils vous aideront dans la compréhension du cheminement. Vous pourrez réaliser uniquement des circuits de petites dimensions à cause de la limitation d'ArtCam. Dans les **prochains numéros d'Electronique et Loisirs Magazine nous étudierons des interfaces afin de rendre la 3DRAG autonome.**



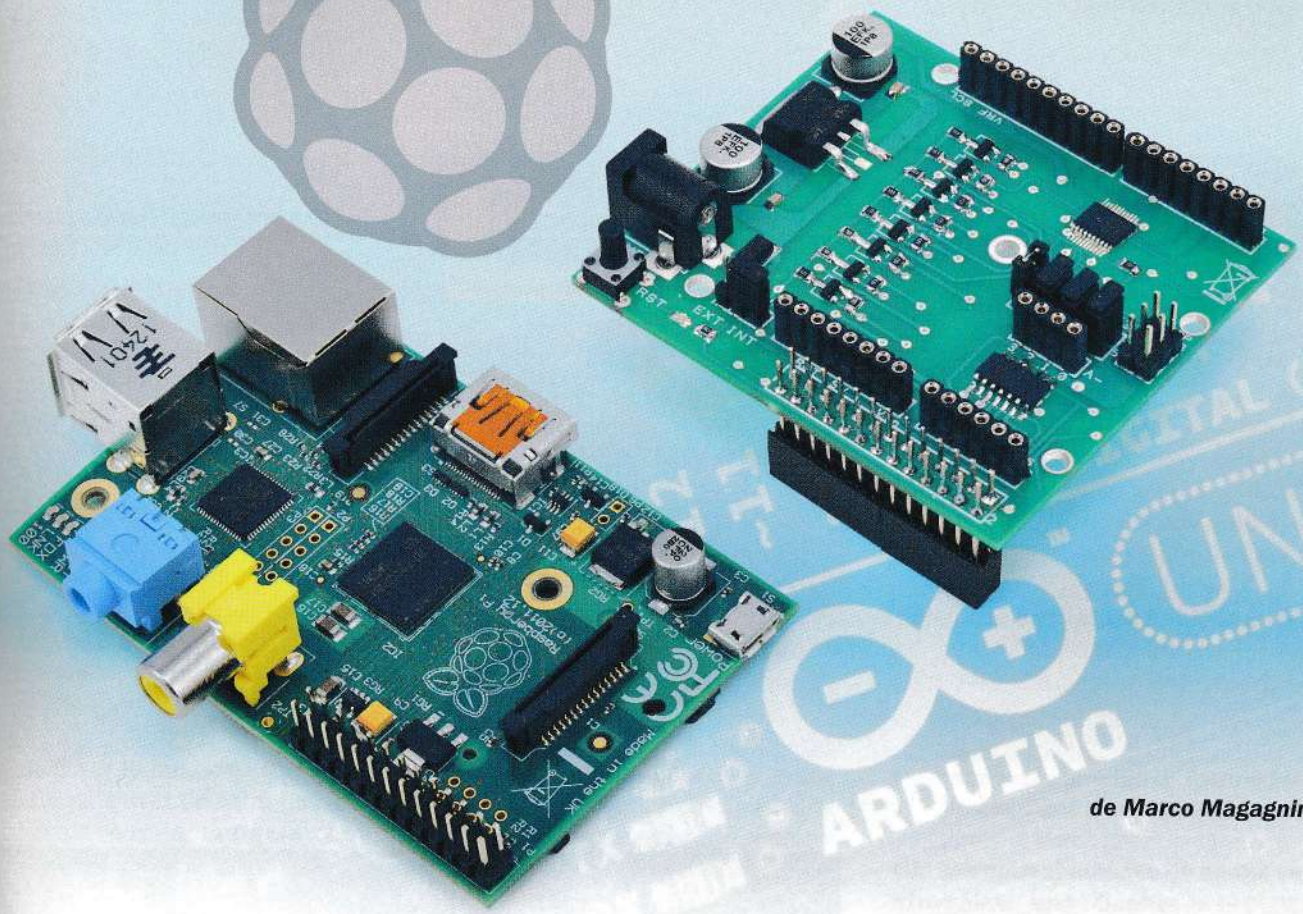
Pour le Matériel

Notre imprimante 3D est disponible auprès de notre partenaire COMELEC (voir les publicités) sous forme de kit complet comprenant tous les équipements mécaniques, électriques et électroniques (bloc d'alimentation inclus) nécessaires pour l'imprimante, à l'exclusion des fils PLA et ABS qui peuvent être commandés séparément.

COMELEC CD 908 13720 Belcodène
Tél. : 04 42 70 63 90 www.comelec.fr

ET 1041

Carte d'extension A/D 16 bits pour RaspberryPi compatible Arduino



de Marco Magagnin

Cette carte d'extension est un convertisseur analogique-numérique d'une résolution de 16 bits. Elle étend les fonctionnalités du RaspberryPi et elle est compatible broche à broche avec la carte Arduino. Nous vous présenterons dans l'avenir d'autres extensions telles qu'un convertisseur DAC, une extension d'entrées/sorties (I/O) numériques, et un adaptateur ayant la capacité de se connecter au PC sans le port GPIO.

Nous poursuivons notre série d'articles liés à l'univers et aux applications du **RaspberryPi**. Ici nous allons traiter plus en détail les **fonctionnalités** du **port GPIO**.

Dans les prochains articles nous vous proposons une **série d'extensions** qui vous permettront à la fois de connecter du matériel et des périphériques à travers le **port GPIO** du **RaspberryPi**, ou à d'autres « Micro-PC » qui ne disposent pas du port GPIO.

En figure 1, vous pouvez voir une photo de **toutes les intégrations possibles de la carte** que nous vous présentons dans cet article. Cette carte, qui est un **convertisseur analogique-numérique**, permet au **RaspberryPi d'effectuer des lectures de capteurs dotés de sorties analogiques**.

Comme nous le verrons plus loin dans l'article, cette carte peut être alimentée à la fois en **12 V** et en **5 V** et **convertit les niveaux de tensions de 3,3 V à 5 V pour le port GPIO et**

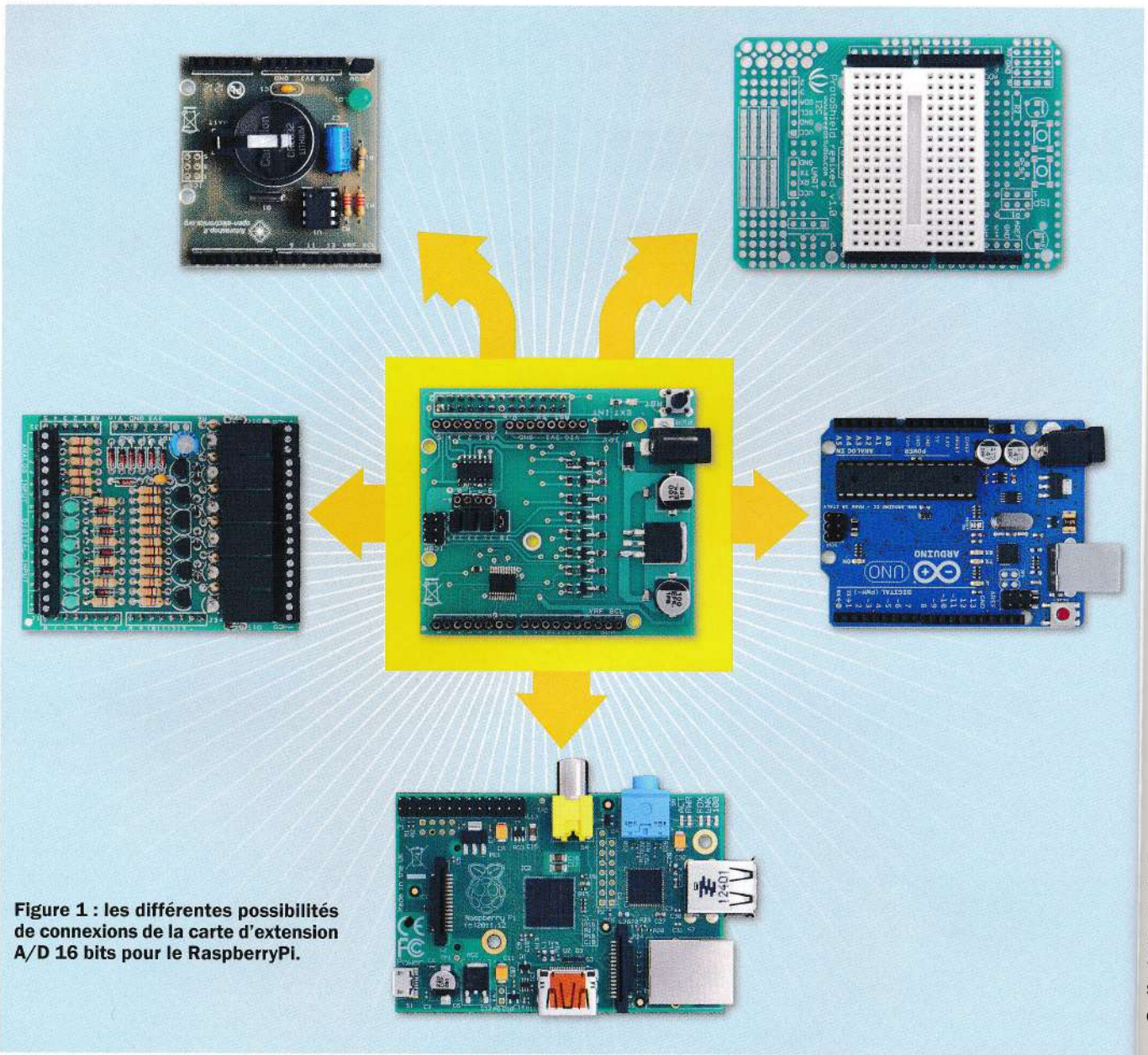


Figure 1 : les différentes possibilités de connexions de la carte d'extension A/D 16 bits pour le RaspberryPi.

les lignes de communication des bus **I2C**, **SPI** et **série**. De plus, cette **carte** dispose à la fois d'un **connecteur adapté pour être inséré dans celui du connecteur GPIO du RaspberryPi et aussi aux connecteurs de la famille des microcontrôleurs et extensions de type Arduino**.

Avec cette carte, vous pouvez **utiliser les cartes d'extensions prévues pour l'Arduino avec le RaspberryPi**, et ainsi **relier l'Arduino et le RaspberryPi en les faisant communiquer**.

Dans les prochains numéros, nous vous proposerons des cartes supplémentaires à savoir un « DAC » intégré

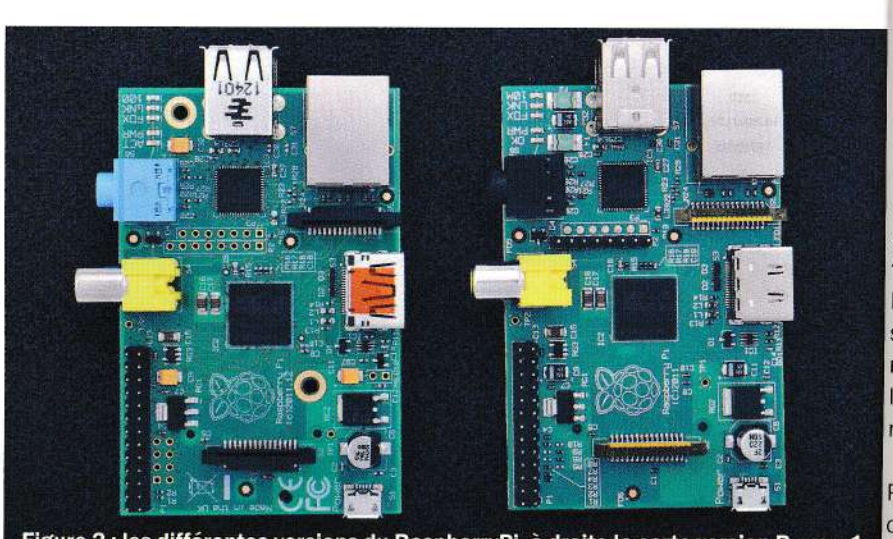


Figure 2 : les différentes versions du RaspberryPi, à droite la carte version B « rev. 1 » et à gauche la carte version B « rev. 2 » reconnaissable grâce à ses 2 trous de fixation.

et une carte d'extension d'entrées/sorties numériques dotée de 16 broches, toutes les deux contrôlées par le BUS I2C.

Pour les périphériques **LINUX** qui **ne disposent pas de port GPIO** tel que le « **Micro-PC** » **MK802** ou **MK802 III** et même un **PC normal**, il sera possible en utilisant cette carte convertisseur analogique-numérique et des convertisseurs USB/I2C et USB/SPI que nous vous présenterons par la suite, **de les relier entre eux ou vers d'autres dispositifs**.

Il est clair que nous allons terminer ce tour d'horizon avec des exemples de programmes pour utiliser cette architecture avec différents langages et périphériques **LINUX**.

Avant de commencer la description du schéma électrique de la carte, il est nécessaire d'examiner à nouveau les modifications et les mises à jour qui ont eu lieu dans ce monde en évolution rapide qu'est le système d'exploitation **LINUX** et qui n'épargnent pas le RaspberryPi.

La **carte RaspberryPi actuellement en vente** est la « **rev.2** » de la « **version B** » caractérisée principalement par rapport à la « rev.1 » par l'augmentation de la RAM de 256 à 512 Mo et de plusieurs modifications des broches I/O du connecteur GPIO.

Pour **identifier le modèle** « **RaspberryPi** » que vous avez en votre possession, référez-vous sur l**a figure 2**. A droite vous apercevez la carte « rev. 1 » et à gauche la carte « rev. 2 » reconnaissable grâce aux 2 trous de fixation.

La figure 3 montre le brochage du connecteur GPIO pour les 2 versions du RaspberryPi, les **principales différences se situent au niveau des broches 3, 5, 7, 13** des entrées/sorties numériques et les **canaux primaires** et **secondaires** du **BUS I2C** de la version « **rev.2** » **sont inversés** par rapport à la version « rev. 1 ».

Passons maintenant à la description du schéma électrique de la carte d'extension, ensuite nous étudierons les aspects logiciels.

Figure 3 : les différents brochages du connecteur GPIO pour les 2 versions du RaspberryPi.

Le schéma électrique

Cette carte d'extension du connecteur GPIO a été conçue pour offrir les possibilités suivantes :

- étendre les **fonctionnalités du port GPIO en le dotant d'un convertisseur A/D asymétrique ou différentiel à 4 canaux**, directement sur la carte, et avec une carte additionnelle que nous vous proposerons par la suite jusqu'à 16 E/S numériques supplémentaires et un DAC 12 bits (4096 pas), toutes les 2 gérées par BUS I2C ;
- **alimentation directe** de la carte prélevée sur le **5 V** du **RaspberryPi** ou avec une **alimentation externe** de **5 V** à **12 V** compatible avec la carte Arduino ;
- **conversion des niveaux** de **3,3 V** à **5 V** et **vice versa** des I/O digitales, des bus I2C, SPI et du port série, afin de répondre à la fois aux exigences électriques du RaspberryPi, des capteurs et périphériques externes ;
- possibilité d'utiliser le convertisseur A/D avec des **entrées asymétriques** ou des **entrées différentielles** ;
- **connecteur à 26 broches** pour la connexion du **port GPIO** du **RaspberryPi** ;
- **connecteurs pour la connexion directe d'une carte Arduino** et des cartes prévues pour ce dernier ;
- possibilité de connecter des **unités supplémentaires externes USB/I2C** et **USB/SPI**.

Analysons le schéma électrique visible sur la figure 3a afin de voir comment les exigences ont été réalisées dans la pratique. Nous partons de l'alimentation en configuration classique basée sur un régulateur de type 7805. Les condensateurs C1, C2, C3 et C4 filtrent et stabilisent la tension de sortie.

Les cavaliers « **EXT** » et « **INT** » permettent le choix d'une alimentation en 5 V à partir de la broche 2 du connecteur GPIO ou d'une alimentation externe

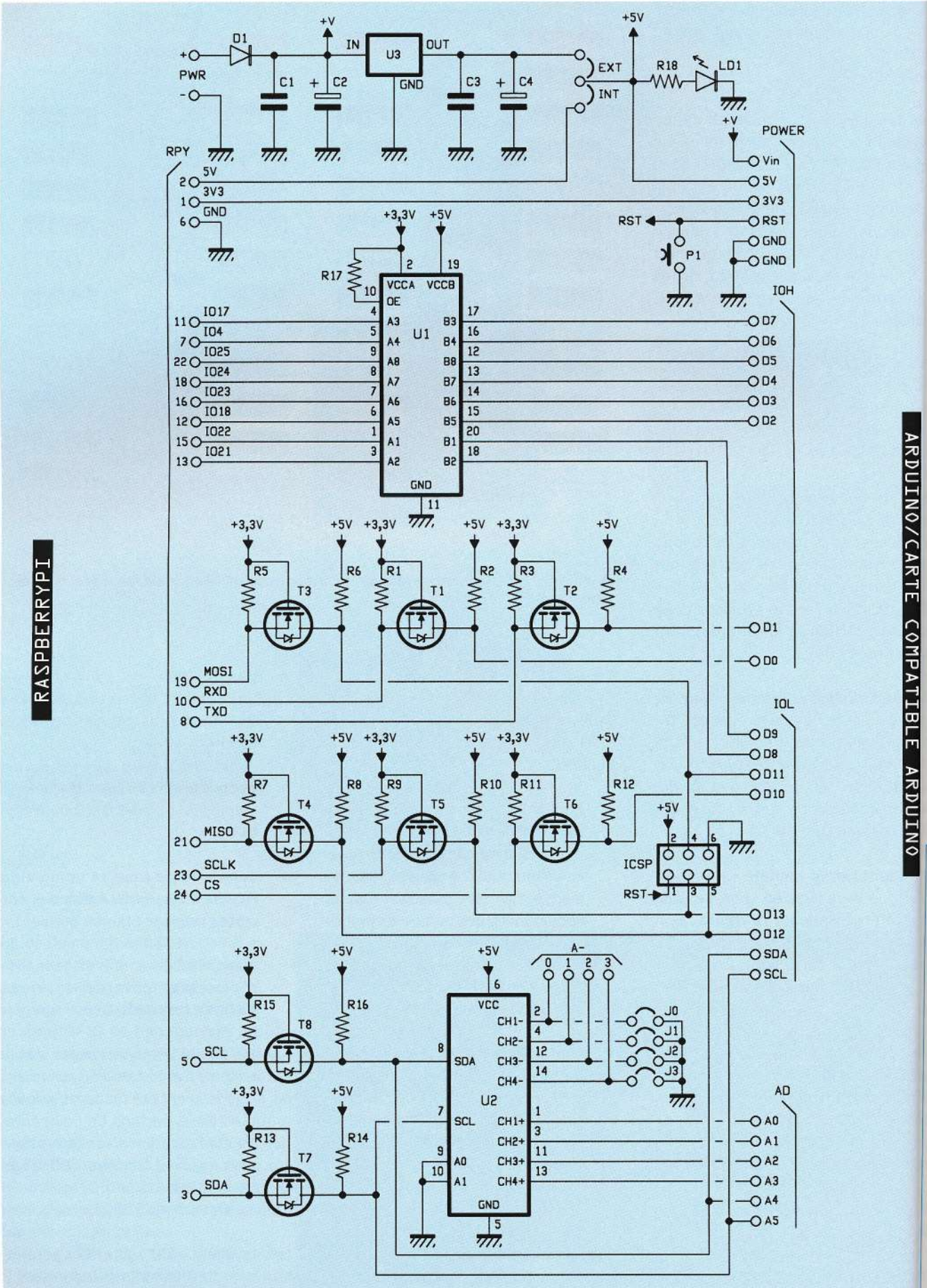


Figure 3a : schéma électrique de la carte d'extension A/D 16 bits pour le RaspberryPi.

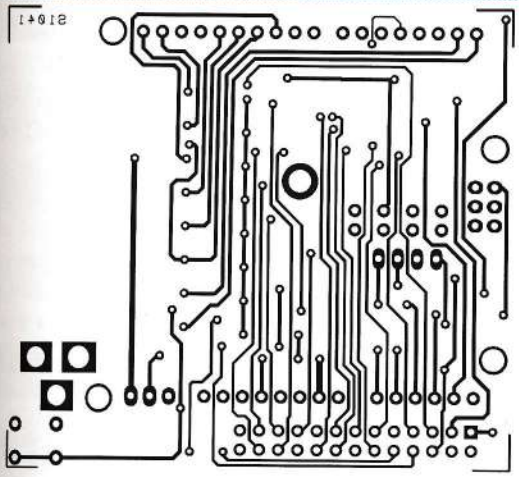


Figure 3b : circuit imprimé à l'échelle 1 : 1 côté soudures de la carte d'extension A/D 16 bits pour RaspberryPi compatible Arduino.

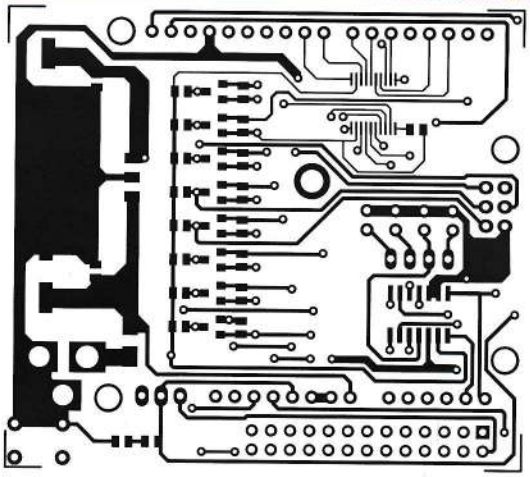


Figure 3c : circuit imprimé à l'échelle 1 : 1 côté composants de la carte d'extension A/D 16 bits pour RaspberryPi compatible Arduino.

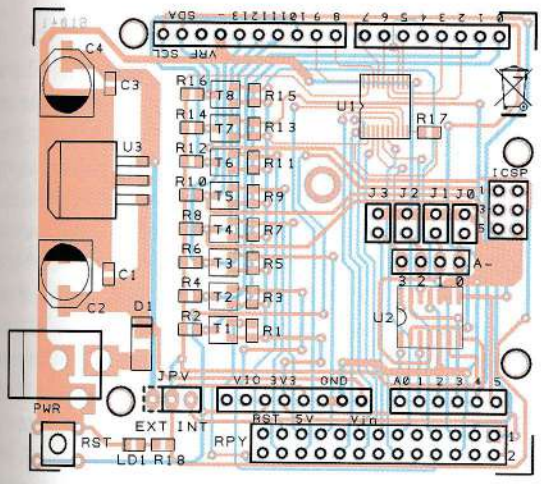


Figure 3d : schéma de câblage de la carte d'extension A/D 16 bits pour RaspberryPi compatible Arduino.

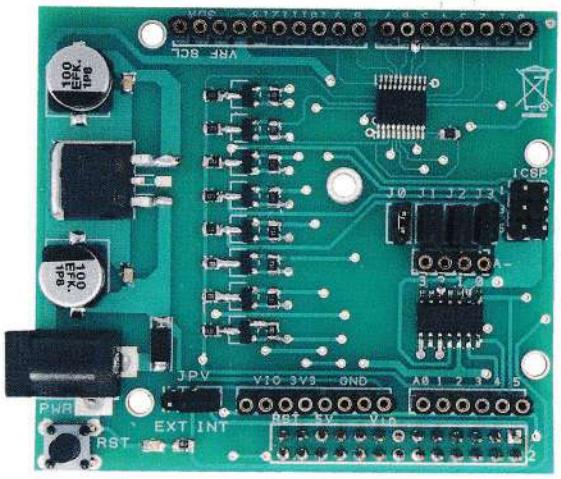


Figure 3e : photo de l'un de nos prototypes de la carte d'extension A/D 16 bits pour RaspberryPi compatible Arduino.

Liste des composants de la carte d'extension A/D 16 bits pour RaspberryPi compatible Arduino

R1.....10 kΩ CMS 0805
R2.....10 kΩ CMS 0805
R3.....10 kΩ CMS 0805
R4.....10 kΩ CMS 0805
R5.....10 kΩ CMS 0805
R6.....10 kΩ CMS 0805
R7.....10 kΩ CMS 0805
R8.....10 kΩ CMS 0805
R9.....10 kΩ CMS 0805
R10....10 kΩ CMS 0805
R11....10 kΩ CMS 0805
R12....10 kΩ CMS 0805
R13....10 kΩ CMS 0805
R14....10 kΩ CMS 0805
R15....10 kΩ CMS 0805
R16....10 kΩ CMS 0805
R17....10 kΩ CMS 0805
R18....470 Ω CMS 0805

C1.....100 nF céramique CMS 0805
C2.....100 µF 25 V électrolytique CMS Type E
C3.....100 nF céramique CMS 0805
C4.....100 µF 25 V électrolytique CMS Type E

U1..... TXB0108PWR CMS
U2..... MCP3428-E/SL CMS
U3..... MC7805ABD2T (D2PAK CMS)
LD1.... LED verte CMS (0805)

T1 ..... BSS138W-7-F CMS
T2 ..... BSS138W-7-F CMS
T3 ..... BSS138W-7-F CMS
T5 ..... BSS138W-7-F CMS
T6 ..... BSS138W-7-F CMS
T7 ..... BSS138W-7-F CMS
T8 ..... BSS138W-7-F CMS
D1..... GF1M-E3/67A CMS
RST.... Microswitch

Divers

PWR... Fiche d'alimentation pour circuit imprimé
Barette sécable 2 pôles mâles (x 4)
Barette sécable 4 pôles femelle (x 1)
Barette sécable 2 x 3 pôles mâles (x 1)
Barette sécable 2 x 13 pôles mâle/femelle (x 1)
Barette sécable 6 pôles femelle (x 1)
Barette sécable 8 pôles femelle (x 2)
Barette sécable 10 pôles femelle (x 1)
Jumpers (cavaliers)

stabilisée par le 7805. De cette façon, vous pouvez éviter l'utilisation d'une alimentation externe lorsque cela n'est pas nécessaire. La broche 2 du RaspberryPi est capable de fournir 500 mA pour la version « rev.1 » et 300 mA pour la « rev.2 ».

Pour des cartes et des capteurs qui nécessitent un courant plus élevé et/ou des tensions plus élevées, telles que des cartes avec des relais, vous devez utiliser l'alimentation externe.

Les circuits de conversion des niveaux sont réalisés en utilisant deux approches différentes. La **conversion des niveaux des broches E/S numériques est réalisée à l'aide du circuit intégré TXB0108** qui est un convertisseur (translateur) de niveau de tension à 8 bits avec 2 lignes d'alimentation, dont vous pouvez voir la description dans l'encadré dédié.

Sur le côté « **A** » du circuit intégré sont reliées les lignes provenant du **connecteur GPIO** du **RaspberryPi** indiqué sur le schéma par « **RPY** ». Les lignes du **côté** « **B** » sont reliées aux connecteurs « **IOL** », « **IOH** », « **POWER** » et « **AD** » qui caractérisent le brochage de l'**Arduino**.

La broche « **GND** » est connectée à la **masse**, les broches **VCCA** et **VCCB** sont les **références de tension pour la conversion des niveaux** et sont connectées respectivement aux lignes de tension **3,3 V** et **5 V**. La broche **OE** (« **Enable** »), quand elle est placée à un **niveau haut**, permet le **fonctionnement du circuit intégré** et est reliée au niveau **VCCA** (3,3 V).

Dans le tableau de la figure 4, vous pouvez voir les **correspondances entre les broches du connecteur GPIO** du **RaspberryPi** et les **broches des connecteurs de l'Arduino**. Les deux colonnes de droite du tableau donnent les correspondances à la fois pour les versions « rev.1 » et « rev.2 ».

Le circuit intégré **TXB0108 permet la conversion des niveaux dans les deux sens** (directions) de **manière automatique**. Le circuit est capable de déterminer la direction du flux de données pour chaque broche d'entrée/sortie (ou pin I/O en Anglais).

Arduino	RPI rev. 1	RPI rev. 2
D2	GPIO18	GPIO18
D3	GPIO23	GPIO23
D4	GPIO24	GPIO24
D5	GPIO25	GPIO25
D6	GPIO4	GPIO4
D7	GPIO17	GPIO17
D8	GPIO21	GPIO27
D9	GPIO22	GPIO22

Figure 4 : le tableau indique les correspondances entre les broches du connecteur GPIO du RaspberryPi (versions « rev.1 » et « rev.2 ») et les broches des connecteurs de l'Arduino.

Cette caractéristique est appelée « **détection automatique** » ou « **autosensing** » dans la langue de William Shakespeare. Cette caractéristique, utile pour la conversion des niveaux des broches d'entrées/sorties digitales, rend peu fiable l'utilisation du même circuit intégré pour la conversion des niveaux des lignes du BUS I2C.

En effet les lignes du **BUS I2C** du **RaspberryPi sont équipées de résistances de tirage** (ou pull-up) qui **interfèrent avec la fonction de détection automatique** du circuit intégré **TXB0108** (voir l'encadré dédié).

Pour les lignes du bus **I2C**, du bus **SPI** et du **port série** nous avons adopté une solution basée sur l'utilisation de transistors **MOSFET** de type **BSS138 canal N à effet de champ à enrichissement** avec un seuil **VGS** de **1,3 V**. Un **MOSFET à enrichissement est caractérisé par le fait qu'il ne conduit pas en l'absence de polarisation**. Le circuit de conversion de niveau est identique pour chaque ligne des différents signaux (différents bus et port série). A titre d'exemple nous nous référons à la ligne SDA du bus I2C.

La **grille** (**G**) du **MOSFET BSS138** (**T7**) est reliée à la ligne d'alimentation du **3,3 V**, la **source** (**S**) à la **ligne du signal d'un niveau inférieur** (3,3 V) et le **drain** (D) à la **ligne du signal de plus haut niveau** (5 V). Entre la source (S) et le 3,3 V est connectée la résistance R13 de tirage (pull-up) tandis qu'entre le drain (D) et le 5 V est connectée la résistance R14 de pull-up.

Analysons le fonctionnement dans l différentes conditions :

1. **Pas de communication en cour** aucun périphérique ne porte l lignes du BUS I2C à un niveau ba par conséquent la ligne du sign à 3,3 V est maintenue à un nive haut par la résistance de pull- R13. La grille et la source du M FET sont toutes les 2 à 3,3 V, sion « grille-source ») est **inférieu au seuil de conduction** et le **M FET est bloqué**. Dans cette con tion, la ligne du signal présente u tension plus élevée (5 V) et est ma tenue à un niveau élevé par la rés tance de pull-up R14. Les **deux s tions de la ligne du signal son des niveaux hauts, mais différen**
2. **Communication provena d'un périphérique à 3,3 V** : périphérique connecté à la lig 3,3 V la porte à un niveau ba La source (S) du transistor M FET T7 est également portée un niveau bas, tandis que la gr (G) est maintenue à un niveau tension de 3,3 V. La tension **devient supérieure au seuil conduction et le MOSFET devi passant** (conducteur). Dans ce condition, le drain de T7 est po à un niveau bas et un courant cule à travers R14 en fonction signaux émis par le périphéri le MOSFET T7 est « piloté » pa périphérique à 3,3 V. En con sion, les **2 sections de la ligne signal sont à un niveau bas**.

3. **Communication provenant d'un périphérique à 5V** : un périphérique connecté à la section de la ligne du signal du 5V la porte à un niveau bas. Dans cette condition, la section du 3,3 V est portée à un niveau bas à travers la diode drain-source du MOSFET, la tension **VGS dépasse alors le seuil de conduction et T7 devient conducteur** (passant). La section 3,3 V de la ligne du signal est maintenue à un niveau bas et « pilotée » par le périphérique du 5V qui fait entrer le MOSFET en conduction. En conclusion, dans ce cas également, les **2 sections de la ligne du signal sont à un niveau bas**.

De cette façon, les niveaux logiques sont convertis et transférés dans les deux directions (sens), quel que soit le périphérique qui commande la communication. Les conditions 2) et 3) réalisent la fonction « **wired AND** » (« connexion ET ») entre les 2 sections de la ligne du signal comme requis par la spécification du BUS I2C, et ne sont pas affectées par les résistances de pull-up du bus.

Nous arrivons maintenant à la **conversion analogique/digitale** (A/D) pour laquelle nous avons adopté le circuit intégré **MCP3428** de **Microchip** (U2). Lors de l'étude de la partie conversion analogique/digitale (A/D), nous avons dû accepter un compromis dans le brochage des connecteurs qui accueillent la carte Arduino.

La carte Arduino dispose de 6 entrées de conversion analogique/digitale (A/D), mais 2 d'entre elles (broches A4 et A5) sont partagées avec les lignes du BUS I2C.

Cependant dans l'**environnement Arduino**, les **différents emplacements des broches sont configurables par le programme**, en fonction des besoins de l'application.

Cette possibilité ne peut être reproduite sur notre carte, car elle utilise en permanence les lignes (broches) du BUS I2C pour communiquer avec le convertisseur analogique/digital (A/D).

Donc **nous ne disposerons que de 4 entrées pour la conversion analogique/digitale** (A/D). Par contre le circuit intégré **MCP3428** permet une conversion avec une **résolution de 16 bits**, aussi bien pour des signaux asymétriques que différentiels. Les broches « **CH1+** » à « **CH4+** » sont reliées respectivement aux broches **A0** à **A3** du connecteur « **AD** » de l'**Arduino**.

Les bornes « **CH1-** » à « **CH4-** » sont **répliquées sur le connecteur** « **A-** », et peuvent être **mises à la masse** (GND) **individuellement** à l'aide des cavaliers **J0**, **J1**, **J2** et **J3**. De cette manière, il est possible de configurer chaque entrée du convertisseur pour mesurer soit des signaux analogiques asymétriques ou différentiels.

Les broches **SDA** et **SCL** du **circuit intégré sont connectées d'un côté aux broches du connecteur correspondant** à l'**Arduino** et de l'**autre sur les drains des transistors T7** et **T8**. Les broches **Adr0** et **Adr1** permettent d'attribuer différentes adresses au circuit **MCP3428** en fonction de la combinaison des niveaux bas et haut présents sur ces broches.

Dans notre cas, **nous mettons les deux broches à un niveau bas en les reliant à la masse** (GND) pour configurer l'adresse à la valeur « **0x68** ». Bien évidemment, les broches VCC et VSS sont connectées respectivement au + 5 V et la masse (GND). Les lignes du bus SPI, MISO, MOSI, SCLK et CS vont aux broches 19, 21, 23 et 24 du connecteur GPIO et aux broches du connecteur SPI de l'Arduino ainsi qu'au connecteur ICSP.

Les lignes du bus série provenant des broches 8 et 10 du connecteur GPIO sont connectées respectivement au circuit à transistors qui gère les niveaux de conversion, ainsi qu'aux broches TXD et RXD du connecteur de l'Arduino.

Utilisons la carte avec le RaspberryPi

Nous allons procéder aux opérations suivantes en utilisant le RaspberryPi en mode terminal.

Les 2 fenêtres **SSH** et **SCP** s'ouvrent, connectons-nous en tant qu'utilisateur « root », **tel que décrit dans les articles précédents consacrés au RaspberryPi à partir de la revue n°123 d'Electronique et Loisirs Magazine.**

Tout d'abord si le RaspberryPi est en fonctionnement, éteignons-le avec la commande :

shutdown -h now ou ***halt***

puis coupons l'alimentation. Relions le connecteur « RPY » de la carte d'extension au connecteur GPIO du RaspberryPi et mettons sous tension.

Maintenant, nous devons activer les modules de gestion (appelés pilotes ou « drivers » dans le monde du système d'exploitation Windows) des bus I2C et SPI ou l'un des deux, en fonction de nos besoins (ou de vos besoins selon le cas).

À ce propos, il existe une multitude de modules de gestion des divers périphériques dans les mondes de WINDOWS et de LINUX, reportez-vous à l'encadré « Les modules de gestion sous LINUX ».

Comme vous pouvez le remarquer dans cet encadré, les modules de gestion des bus I2C et SPI sont compilés dans le noyau « **Raspbian** » en tant que modules externes et ne font pas partie intégrante du noyau.

En plus de cela, la distribution « **Raspbian** » que nous utilisons dans nos articles n'est pas optimisée pour une utilisation normale. En effet, les lignes des bus partagent des broches du connecteur GPIO qui sont privilégiées dans la configuration par défaut.

Pour utiliser un ou les 2 modules de gestion des bus I2C et SPI, il est nécessaire de les « ajouter » à l'ensemble des modules reconnus par le noyau. Étant donné que les **modules de gestion** des bus **I2C** et **SPI** ont été introduits dans les nouvelles versions de la distribution « **Raspbian** », nous commençons toujours par **une mise à jour de la distribution présente sur notre RaspberryPi** à l'aide des commandes suivantes :

Le circuit intégré TXB0108

Le circuit intégré **TXB0108** (figure A) est un **convertisseur** (translateur) de **niveau de tension à 8 bits** avec 2 lignes d'alimentation configurables. Il permet de communiquer avec des périphériques et des systèmes numériques caractérisés par des signaux dont les tensions ont des niveaux différents, par exemple 3,3 V et 5 V.

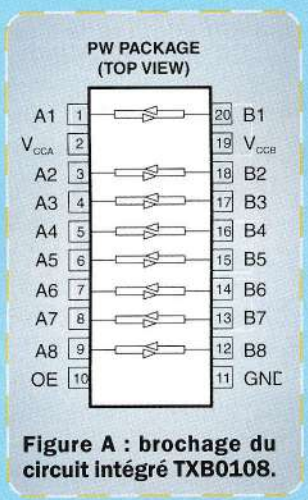


Figure A : brochage du circuit intégré TXB0108.

L'architecture interne du TXB0108, visible sur la figure B qui représente une section relative à une ligne du signal, est capable de déterminer automatiquement le sens du flux de données. Il ne nécessite donc pas de signal de commande pour définir la direction du flux de données d'un périphérique à l'autre. Lorsqu'une section de la ligne est activée pour transférer les données dans une seule direction, le driver de sortie du circuit intégré garde la ligne entière à un niveau fixe, bas ou haut. Compte tenu du faible courant de maintien, prévu à la conception du circuit, un périphérique externe qui « s'engage » dans la ligne dont la direction est opposée et avec un niveau de courant qui est supérieur à celui de maintien du circuit intégré, est capable de provoquer l'inversion du sens du flux de données. C'est pour cette raison que les périphériques qui pilotent les lignes d'entrées/sorties du circuit intégré doivent être capables de maintenir un courant d'au moins ± 2 mA.

Quand un périphérique « engage » la ligne dans une direction, le circuit monostable détecte le « front montant » ou le « front descendant », sur la porte A ou B en fonction de la provenance du flux de données. Lors de la détection du « front montant », le circuit monostable met en conduction les transistors PMOS (T1, T3) pendant une courte durée de manière à accélérer la transition des niveaux bas vers des niveaux hauts. D'une manière similaire, lors de la détection d'un « front descendant », le circuit monostable met en conduction les transistors NMOS (T2, T4) pendant une courte durée, ce qui accélère la transition des niveaux hauts vers des niveaux bas. Le TXB0108 est conçu pour piloter des charges capacitives jusqu'à 70 pf. Le « driver » de sortie du circuit intégré est caractérisé par un courant de commande très faible.

Pour la même raison, **vous ne devez pas utiliser le TXB0108 dans des applications qui nécessitent la conversion des niveaux d'un BUS I2C ou d'un BUS « One-wire »**, car les **lignes du BUS sont configurées en « Collecteur Ouvert »** avec des **résistances de pull-up**, celles-ci **interfèreraient** avec l'**auto-détection** du circuit intégré. La porte A est commandée par la tension présente sur la broche VCCA et accepte des niveaux de tension compris entre 1,2 V et 3,6 V. La porte B est commandée par la tension présente sur la broche VCCB et accepte des niveaux de tension compris entre 1,65 V et 5,5 V. La tension présente sur la broche VCCA ne doit jamais être supérieure à celle présente sur la broche VCCB.

Lorsque la broche « Output Enable » (OE) est à un niveau bas, toutes les

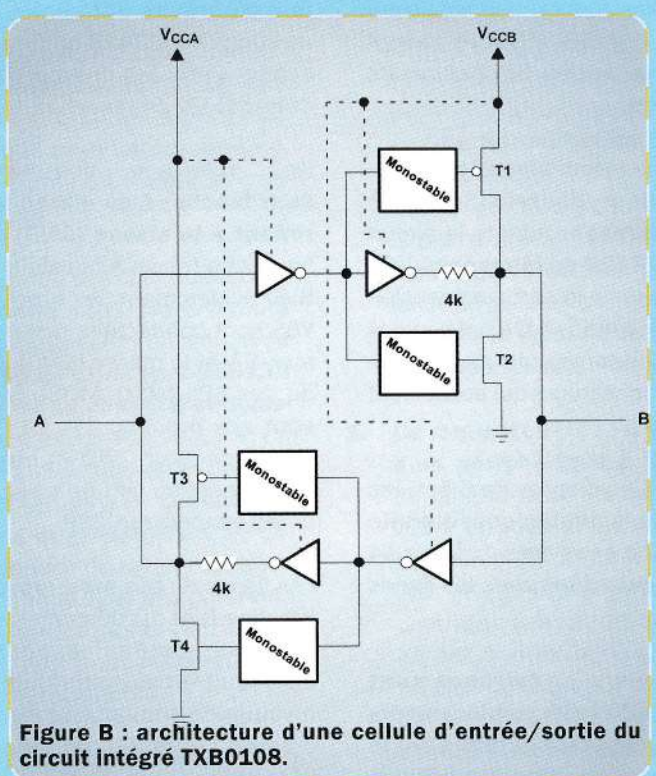


Figure B : architecture d'une cellule d'entrée/sortie du circuit intégré TXB0108.

Si vous devez connectez des résistances de « pull-up » (reliées au + VCC) ou « pull-down » (reliées à la masse) sur les lignes d'entrées/sorties, leur valeur doit être supérieure à 50 kΩ afin d'éviter tout problème avec le « driver » de sortie du circuit intégré.

sorties sont placées dans un état de haute impédance, et la conversion des niveaux est bloqué. En fonctionnement normal, la broche « OE » doit être maintenue à un niveau haut, connectée à la ligne d'alimentation VCCA.

apt-clean
apt-get update
apt-get upgrade

Maintenant, ouvrons le fichier de configuration qui contient la liste des modules, avec la commande :

nano /etc/modprobe.d/raspi-blacklist.conf

« **Nano** » est un **éditeur de texte minimaliste qui s'exécute sur le RaspberryPi**. Éliminons les modules I2C et SPI de la liste en supprimant les lignes ou, comme nous préférons le faire en mettant un « **#** » en **début de ligne** (figure 5).

Appuyons sur « **Ctrl-X** », et puis sur « **Y** » pour **enregistrer le fichier après la modification**. Maintenant, nous devons veiller à ce que les 2 modules soient chargés correctement et fassent partie intégrante du noyau.

Pour cela nous avons 2 possibilités, **la première est de charger les modules avec une commande**, et ils **resteront actifs tant que le RaspberryPi sera allumé**. Au prochain démarrage les modules devront à nouveau être rechargés avec la commande.

La seconde est de charger les modules directement au démarrage du système d'exploitation, et de rendre ainsi les applications disponibles immédiatement après le démarrage, ce qui est essentiel lors d'une utilisation en serveur sans surveillance.

La première possibilité nécessite l'utilisation de la commande ***modprobe***. Tapons la commande (voir la figure 6) :

modprobe i2c-dev

Nous pouvons vérifier l'activation correcte du pilote (« driver ») à l'aide de la commande affichant la liste de tous les modules installés (voir la figure 7) :

lsmod

Sous **LINUX** tout (ou presque) se présente sous la forme d'un fichier, si nous allons dans le répertoire « dev/ », nous verrons les fichiers des périphériques « **i2c-0** » et « **i2c-1** » connectés (voir la figure 8).

Figure 5 : éliminons les modules I2C et SPI de la liste en supprimant les lignes ou en mettant un « # » en début de ligne comme visible sur la figure.

Figure 6 : utilisation de la commande *modprobe*.

Figure 7 : affichage de la liste de tous les modules installés grâce à la commande *lsmod* .

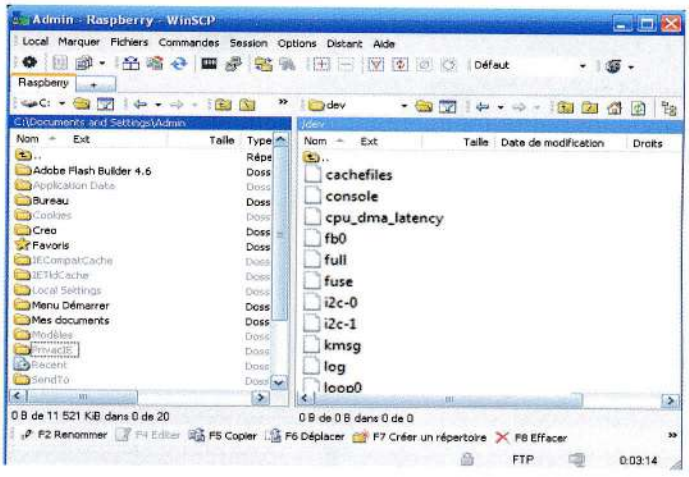


Figure 8 : dans le répertoire « dev/ », vous apercevez les fichiers des périphériques connectés « i2c-0 » et « i2c-1 ».

La commande « ***modprobe*** » permet de charger et de désactiver les modules lorsque nous l'exécutons, elle conserve ses effets **tant que le RaspberryPi est allumé**.

Dans le cas d'un arrêt ou d'une remise à zéro, les modules doivent être rechargés manuellement. Cette condition n'est pas prévue pour une application autonome qui doit fonctionner en mode automatique.

Le circuit intégré MCP3428

Le circuit intégré MCP3428 est un convertisseur analogique/digital de type « Delta-Sigma » à faible bruit et d'une résolution de 16 bits. Ce type de convertisseur est basé sur le principe du « suréchantillonnage » d'un signal d'entrée. Un comparateur est utilisé pour convertir sur 1 bit (c'est-à-dire 0 ou 1) la différence (delta) entre le signal d'entrée et le résultat de la conversion (0 = plus petit, 1 = plus grand). Le résultat de la comparaison parvient alors dans un filtre appelé le « décimateur », qui fait la somme (sigma) des échantillons du signal d'entrée. Cela revient à calculer l'intégrale de la différence entre l'entrée et la sortie. On obtient alors un système asservi (la sortie est rebouclée sur l'entrée) qui fait osciller la valeur de l'intégrale du signal à convertir autour d'une valeur de référence (le résultat de la conversion).

L'intérêt de ce genre de convertisseur réside dans sa grande résolution de sortie possible (16, 24, 32, 64 bits voire plus) pour des signaux d'entrée avec une bande passante modérée. Ces convertisseurs sont très adaptés à la conversion de signaux analogiques issus de capteurs dont la bande passante est souvent faible (par exemple les signaux audio). Les convertisseurs « Delta-Sigma » sont, par exemple, utilisés dans les lecteurs de CD dans le cas d'une conversion numérique/analogique.

SOIC, TSSOP

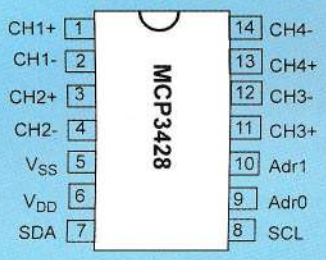


Figure C

Le MCP3428 dispose de 4 canaux avec des entrées différentielles, son brochage est représenté sur la figure C. La fréquence d'échantillonnage dépend de la résolution que l'on veut obtenir. Avec une résolution de 16 bits on peut obtenir 15 mesures par seconde, avec une résolution de 14 bits on obtient 60 mesures par seconde et avec une résolution de 12 bits on arrive à 240 mesures par seconde. Le circuit se calibre automatiquement en éliminant les erreurs d'« offset » (décalages) et de gain en raison des variations de la température et de la tension d'alimentation.

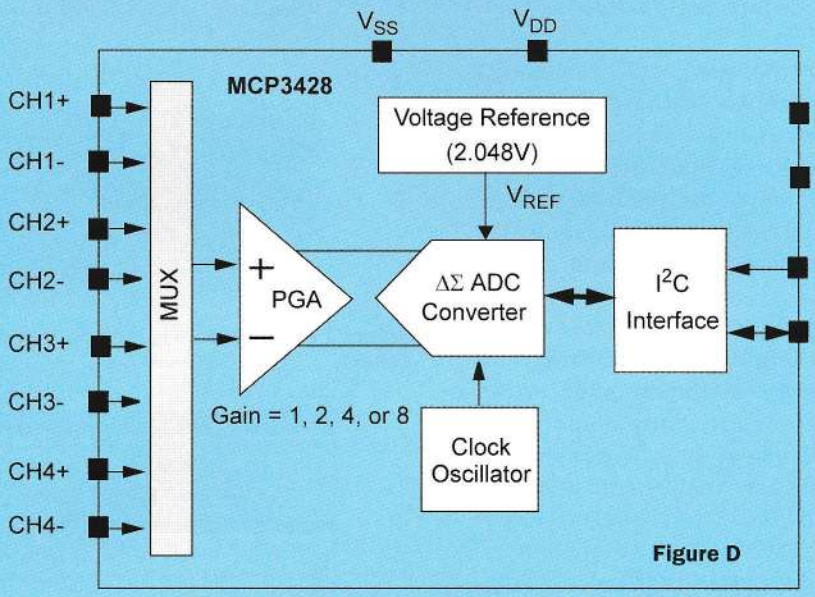


Figure D

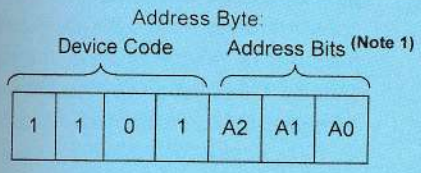
Comme vous pouvez le voir en figure D, la tension de référence d'une valeur de 2,048 V est gérée en interne dans le circuit, ce qui permet un intervalle des tensions d'entrées différentielles de ± 2,048 V. Le signal d'entrée peut être amplifié au moyen d'un amplificateur interne (PGA sur la figure D), dont le gain est programmable par l'utilisateur et peut varier de 1, 2, 4 ou 8 fois. L'utilisateur peut également programmer le mode de conversion, qui peut être soit un échantillonnage unique ou continu. En mode continu le circuit effectue des lectures en permanence et met à jour le tampon (« buffer ») avec la dernière lecture effectuée. En mode simple, la lecture est effectuée après qu'une commande soit reçue, ensuite le circuit se met en veille jusqu'à la prochaine commande. Ce mode permet de réduire considérablement la consommation de courant. La communication avec le circuit s'effectue à l'aide du bus I2C en envoyant sur le registre d'adresse et de commande deux octets à chacun.

I²C Device Address Bits			Logic Status of Address Selection Pins	
A2	A1	A0	Adr0 Pin	Adr1 Pin
0	0	0	0 (Addr_Low)	0 (Addr_Low)
0	0	1	0 (Addr_Low)	Float
0	1	0	0 (Addr_Low)	1 (Addr_High)
1	0	0	1 (Addr_High)	0 (Addr_Low)
1	0	1	1 (Addr_High)	Float
1	1	0	1 (Addr_High)	1 (Addr_High)
0	1	1	Float	0 (Addr_Low)
1	1	1	Float	1 (Addr_High)
0	0	0	Float	Float

Figure E

Le registre d'adresse

Le registre d'adresse est utilisé pour identifier le périphérique que vous souhaitez atteindre sur le bus I2C. Il se compose de 7 bits dont les 4 premiers sont définis en usine, ils représentent le code attribué par le fabricant au circuit et prend la configuration 1101. Les 3 derniers bits sont déterminés par la façon dont sont connectées les 2 broches (pins) d'adresse du circuit.



Le tableau de la figure E contient les différentes configurations des 3 bits en fonction de l'état des broches d'adresse. Dans notre cas, nous avons relié les 2 broches à la masse et donc la valeur des 3 bits vaut « 000 ». La configuration complète de l'octet de l'adresse du circuit devient « 1101000 », ce qui est équivalent à « 0x68 » en hexadécimal.

Le registre de configuration

La gestion du registre de configuration est un peu plus complexe . Ce dernier est utilisé pour définir le mode de fonctionnement du convertisseur A/D et pour lire le « buffer » (mémoire tampon) d'un canal en particulier. L'octet de configuration est le résultat de la concaténation de plusieurs paramètres exprimés dans un format binaire, comme indiqué sur la figure F.

bit 7 « RDY » ou Bit « Ready » :

ce bit indique que les données dans la mémoire tampon sont prêtes à être lues. En mode simple ce bit doit être à « 1 » pour indiquer au circuit de commencer une nouvelle lecture.

En lecture le bit « RDY » a les significations suivantes :
1 = le tampon de sortie n'a pas été mis à jour.
0 = le tampon de sortie a été mis à jour avec la dernière lecture.

En écriture le bit « RDY » a les significations suivantes :
En mode de conversion continu, il est ignoré.
En mode de conversion à échantillonnage simple :
1 = lance une nouvelle conversion.
0 = pas de signification.

bits 6-5 C1-C0 ou bits de sélection du canal de lecture :

00 = Canal 1
01 = Canal 2
10 = Canal 3
11 = Canal 4

bit 4 O/C ou bit qui indique le mode de conversion :

1 = mode de conversion à échantillonnage continu.
0 = mode de conversion à échantillonnage unique.

bit 3-2 S1-S0 ou bits de configuration de la résolution du convertisseur :

00 = 12 bits avec 240 conversions par seconde
01 = 14 bits avec 60 conversions par seconde
10 = 16 bits avec 15 conversions par seconde

bit 1-0 G1-G0 ou bits de configuration du facteur de gain de l'amplificateur PGA :

00 = x1
01 = x2
10 = x4
11 = x8

R/W-1	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
$\overline{RDY}$	C1	C0	$\overline{O}$/C	S1	S0	G1	G0
1 *	0 *	0 *	1 *	0 *	0 *	0 *	0 *
bit 7							bit 0

Figure F

Par exemple, si nous voulons lire le canal 1 en mode simple avec une résolution de 16 bits et un gain de « 1 » (PGA x1) nous devons insérer la séquence suivante dans le registre : « 1-00-0-10-00 » ce qui correspond à la valeur « 0x88 » en hexadécimal c'est-à-dire au registre utilisé dans notre programme.

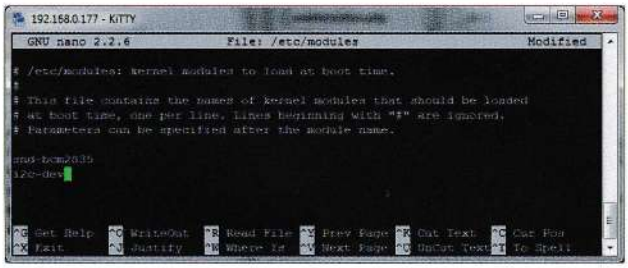


Figure 9 : ajout de la nouvelle ligne dans le fichier de configuration.

Figure 10 : l'adresse « 0x68 » identifie le convertisseur A/D.

La commande « ***modprobe*** », avec l'option de suppression (« remove »), peut également être utilisée pour désactiver un module déjà chargé, par exemple :

modprobe –r i2c-dev

Si nous voulons que les modules soient chargés à l'allumage du RaspberryPi, il est nécessaire d'activer le chargement automatique du « driver ».

Pour cela nous devons modifier le fichier de configuration « /etc/modules » qui contient la liste des pilotes à charger au démarrage. Pour modifier le fichier, nous utilisons la commande :

nano /etc/modules

et ajoutons une nouvelle ligne dans le fichier de configuration (voir la figure 9) :

i2c-dev

Appuyons sur « CTRL-X » puis sur « Y » pour enregistrer les modifications apportées au fichier et sortir. Maintenant installons le paquet « i2c-tools » qui fournit un certain nombre de fonctions que nous pouvons appeler à l'aide d'une ligne de commande pour vérifier le fonctionnement du bus I2C :

apt-get install i2c-tools

Ajoutons à notre « utilisateur » le groupe i2c :

adduser pi i2c

Redémarrons le RaspberryPi pour activer la nouvelle configuration avec la commande :

reboot

Après le redémarrage du RaspberryPi et une fois que nous nous sommes connectés avez « Putty » ou « Kitty », vérifions si le convertisseur A/D est visible via le bus I2C à l'aide de la commande :

i2cdetect -y 0 pour le RaspberryPi « rev.1 »

ou

i2cdetect -y 1 pour le RaspberryPi « rev.2 »

Nous obtenons un résultat similaire à celui de la figure 10 où l'adresse « 0x68 » identifie le convertisseur A/D. Puisque nous réalisons un outil d'intégration avec la carte Arduino, pour tester le fonctionnement du convertisseur A/D, et plus généralement le BUS I2C, nous avons construit un petit circuit qui permet de mesurer la température d'une résistance CTN et la luminosité ambiante avec une photorésistance.

Le schéma du circuit est représenté sur la figure 11. Les résistances ont une valeur de 22 kΩ. La CTN est un modèle de marque Vishay référencé NTCLE-100E-3103-10 kΩ. La photorésistance est un modèle LDR04 ou GL5528 de 10 kΩ.

Le point milieu du diviseur de tension dont fait partie la CTN est relié au canal « CH1+ » (A0) du convertisseur, le point milieu du diviseur de tension dont fait partie la photorésistance est relié au canal « CH2+ » (A1).

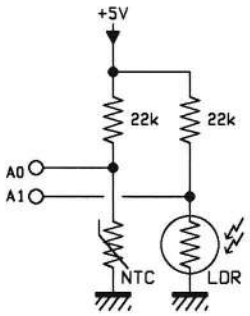
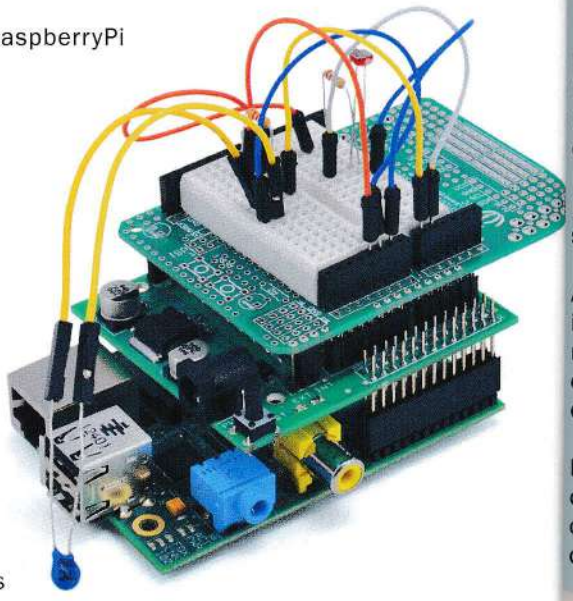


Figure 11 : schéma électrique du petit circuit qui permet de mesurer la température d'une résistance CTN et la luminosité ambiante avec une photorésistance.

Photo de la carte de prototypage pour Arduino avec une plaque d'expérimentation montée sur la carte d'extension, celle-ci est reliée au connecteur GPIO du RaspberryPi.



Les points « CH1- » et « CH2- » sont reliés à la masse.

Le circuit est réalisé à l'aide d'une carte de prototypage pour Arduino avec une plaque d'expérimentation.

La carte de prototypage est montée sur la carte d'extension, celle-ci est reliée au connecteur GPIO du RaspberryPi. Il faut toujours connecter le RaspberryPi et vérifier les connexions avant la mise sous tension de l'ensemble.

Maintenant nous allons faire fonctionner le convertisseur en utilisant la ligne de commande :

i2cget -y 0 0x68 0x88 w pour le RaspberryPi rev. 1

ou

i2cget -y 1 0x68 0x88 w pour le RaspBerryPi rev. 2

pour lire la valeur hexadécimale (figure 12) des 2 octets renvoyés par le canal 1 du convertisseur A/D qui représente la valeur numérique de la tension mesurée aux bornes de la résistance CTN.

Figure 12 : résultat de la conversion en valeur numérique de la tension mesurée aux bornes de la résistance CTN, exprimée par 2 octets hexadécimaux, dans notre cas « 5769 »

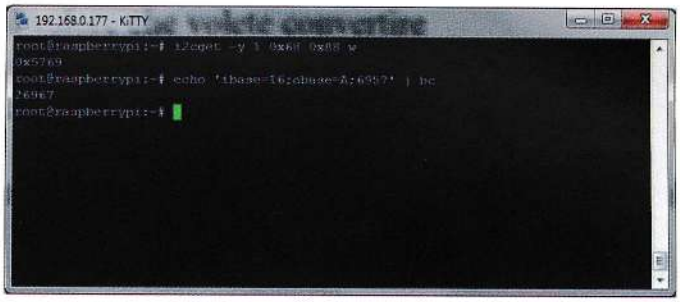


Figure 13 : conversion de la valeur hexadécimale correspondant à la valeur décimale, retourne la valeur « 26967 ».

Le 2eme paramètre indique si le bus I2C doit être interrogé, le 3eme indique le registre qui contient l'adresse du périphérique à lire (0x68), le 4eme indique le registre de commande qui représente l'entrée sur laquelle nous voulons effectuer la lecture et la modalité d'échantillonnage, le dernier paramètre « **w** » indique si nous voulons renvoyer les 2 octets de la mesure.

Les modules de gestion sous LINUX

Le monde GNU-LINUX est immense, et une analyse approfondie de son fonctionnement, en plus de ne pas être l'intérêt de ce magazine, prendrait des centaines de pages et de diagrammes. Cependant, nous pensons qu'il est intéressant d'aborder certains aspects qui peuvent faciliter la compréhension de ce montage ou qui peuvent donner une piste pour la résolution de problèmes qui pourraient survenir.

Par exemple la nécessité d'activer les modules de gestion des bus I2C et SPI afin de comprendre les aspects cachés de la gestion des pilotes (« drivers ») des périphériques sous LINUX. Nous avons décrit dans les précédents articles sur le RaspberryPi les caractéristiques du noyau de LINUX, qui permet le fonctionnement sur de nombreux processeurs différents, et donc le principal facteur de son utilisation.

Pour comprendre l'architecture du noyau de LINUX, vous pouvez considérer qu'elle se compose d'un certain nombre de sous-systèmes comme le montre la figure G.

Arrêtons-nous un instant sur 2 sous-systèmes particuliers, l'un appelé « Arch » visible sur la figure G. C'est une distribution légère et rapide dont le concept est de rester la plus simple possible. Elle peut être considérée comme le véritable cœur du noyau, et prend en charge le processeur spécifique auquel elle est destinée. Elle doit être réécrite, du moins en partie, pour chaque nouveau processeur.

L'autre sous-système est fortement intégré avec le premier et représente l'ensemble des pilotes des périphériques, c'est-à-dire l'ensemble des modules qui permettent de gérer des périphériques très différents à partir du même noyau et en fonction des applications de l'utilisateur.

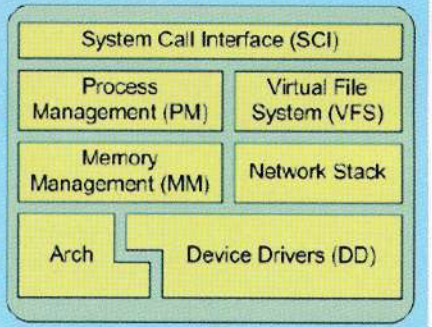


Figure G

La plupart des dizaines de millions de lignes de code qui constituent le code source du noyau GNU-LINUX sont concentrées dans le sous-système « pilotes des périphériques » (Device Drivers) et sont compilées avec le noyau. Il existe deux possibilités pour compiler un pilote de périphérique (Device Drivers), la première est de compiler le pilote de périphérique dans le noyau afin d'obtenir un seul objet monolithique. Cette option est utilisée pour intégrer les pilotes de périphériques dans le noyau et doit obligatoirement être utilisée avec une console pour rendre accessible les périphériques automatiquement (Plug and Play) tels que les périphériques USB, afin de rendre ces opérations totalement transparentes pour l'utilisateur.

La seconde possibilité permet de compiler le pilote de périphérique (Device Drivers) en tant que modules chargeables séparément dans le noyau, si nécessaire, par exemple les pilotes de périphériques pour les bus I2C et SPI. Comme nous l'avons déjà vu dans les articles précédents, les périphériques gérés par le pilote de périphérique sont ensuite disponibles pour l'application sous la forme de fichiers dans le répertoire « **/dev** » (par exemple « gpio4 » ou « ttyUSB0 » sont des exemples que nous avons déjà rencontrés).

Voyons brièvement la séquence des actions et des liens qui permettent de créer la chaîne de transmission de données et de commandes entre le périphérique matériel réel, géré par le noyau, et sa représentation applicative dans le répertoire « /dev ». Lors d'une compilation d'une nouvelle distribution LINUX, il faut commencer par choisir les sources que nous souhaitons inclure dans la compilation et les structurer dans des dossiers bien organisés.

Une des étapes les plus difficiles consiste à définir les caractéristiques de la distribution finale, un objectif qui est atteint en configurant des centaines d'options et de paramètres présents dans les fichiers sources. Pour cela nous devons utiliser un menu de configuration qui, comme vous pouvez le voir sur les figures H et I, représente 2 pages pour la section dédiée aux pilotes de périphériques (Device Drivers). A la figure H vous pouvez voir, par exemple, le pilote (driver) FTDI (celui utilisé pour la station météo du numéro 127 d'Electronique et Loisirs Magazine), tandis qu'sur la figure I est présent le pilote de périphérique (Device Driver) pour le BUS I2C.

Figure H

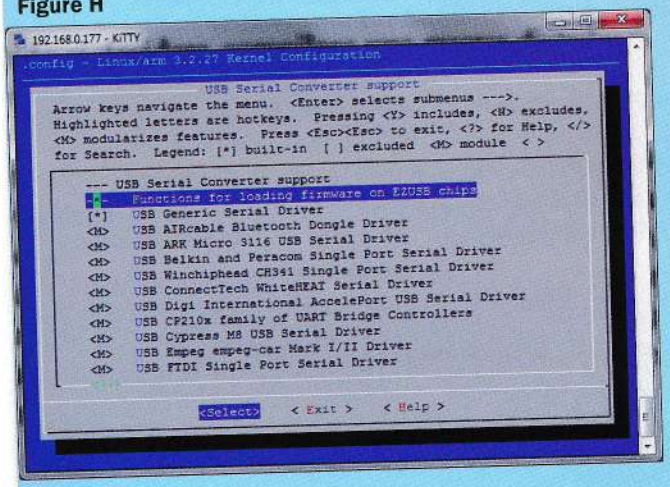
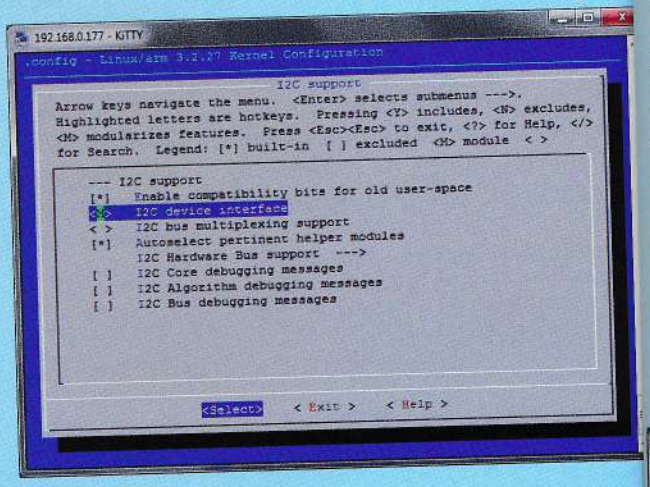


Figure I



Les modules à inclure dans le corps du noyau sont choisis par un astérisque « * », ceux qui doivent être compilés séparément sont marqués par un « **M** ». Lorsque nous avons terminé, et enregistré le fichier de configuration, nous devons compiler le noyau en utilisant le compilateur « C » (package g++). A la fin, nous installons les composants compilés du noyau respectivement dans les partitions « boot » (de démarrage) et « root » (partition racine). Le noyau reconnaît chaque pilote de périphérique (Device Driver) à l'aide d'un nombre majeur et mineur caractérisant chaque périphérique (Device). Ces nombres sont des identificateurs uniques pour chaque pilote de périphériques (Device Driver), ils sont définis dans la documentation du noyau. Ainsi chaque pilote (driver) est relié à l'aide d'un code unique à chaque type de périphérique. Ce qu'il faut retenir c'est que les périphériques sont caractérisés par des nombres majeurs (ou major device numbers).

Par exemple, tous les disques SCSI présentent un nombre majeur égal à 8. Par ailleurs, chaque périphérique individuel possède un nombre mineur. Ainsi, « /dev/sda » est caractérisé par un nombre mineur égal à 0. Les nombres majeur et mineur permettent au noyau d'identifier les périphériques. Le nom de fichier est arbitraire mais il est choisi de manière à apporter de la cohérence. Vous pouvez déterminer les nombres majeur et mineur (8, 0) grâce à la commande « ls » appliquée à « /dev/sda », nous obtenons par exemple :

```
brw-r-.-.-.-.- 1 root   disk   8,   0 May 14   2007 /dev/sd
```

Pour connaître tous les pilotes présents dans le noyau du RaspberryPi, tapons la commande :

```
cat /proc/devices
```

avec laquelle no
cation. Vous pou
généralement ré
magnétiques) et

L'accès aux péri
persistante. La l
octets). Ces péri

La commande

```
ls -l /dev/hda
```

affiche une ligne

```
brw-r-.-.-.-.- 1 ro
```

En revanche, l'a
seule fois seule
par un autre pr
caractérisés pa
dev/dsp » (*Digit

```
crw-r-.-r-.- 1 ro
```

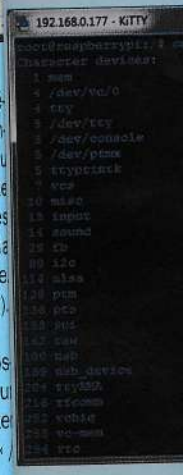
L'association de
toire « /dev », en

```
ls -l
```

Nous obtenons
qui identifie un
un « bloc », suiv

Ensuite nous a
ième le nombre

De cette manièr
dans le réperto
d'associer le pi



Lorsque nous
bre du périph
exemple dans

```
mknod /dev/
```

avec laquelle nous obtenons la liste de la figure J qui montre les pilotes disponibles avec leur nombre majeur d'identification. Vous pouvez remarquer que la liste est divisée en fichiers caractères et en fichiers blocs. Les périphériques sont généralement répartis en deux grandes catégories : les périphériques à accès aléatoire (disques, lecteurs de bandes magnétiques) et les périphériques série (souris, cartes son, terminaux).

L'accès aux périphériques en mode aléatoire se fait parmi de grands blocs continus de données stockées de manière persistante. La lecture y est réalisée par unités discrètes (pour la plupart des disques, cela se fait par groupe de 1024 octets). Ces périphériques sont appelés « périphériques blocs » (block devices).

La commande

ls -l /dev/hda

affiche une ligne avec une lettre « b » à gauche indiquant ainsi que votre disque dur est un périphérique bloc, par exemple :

brw-r-.-.-.- 1 root disk 3, 64 Apr 30 2008 /dev/hdb

En revanche, l'accès aux périphériques série se fait « un octet à la fois ». Les données peuvent être lues ou écrites une seule fois seulement. Par exemple, après qu'un octet ait été lu depuis votre souris, le même octet ne peut plus être lu par un autre programme. Les périphériques série sont appelés « périphériques caractères » (*character devices*) et sont caractérisés par un « c » sur la partie gauche de la ligne affichée par la commande « ls ». Par exemple le périphérique « /dev/dsp » (*Digital Signal Processor*, c'est-à-dire la carte son) se caractérise ainsi :

crw-r-.-r-.- 1 root sys 14, 3 Jul 09 2004 /dev/dsp

L'association des pilotes de périphériques (Device Driver) du noyau avec les fichiers des périphériques présents dans le répertoire « /dev », en fait utilisent leur propre numéro de périphérique. Allons dans le répertoire « /dev » et tapons la commande :

ls -l

Nous obtenons le résultat visible sur la figure K. Comme nous l'avons expliqué un peu plus haut, à chaque début de ligne qui identifie un périphérique se trouve la première lettre « c » ou « b » qui indique si le périphérique est un « caractère » ou un « bloc », suivie par les autorisations sur le fichier (lecture seule, écriture..), et enfin le « propriétaire » du fichier.

Ensuite nous apercevons 2 nombres séparés par une virgule, le premier est le nombre majeur du périphérique et le deuxième le nombre mineur. Après la date, nous trouvons le nom du fichier qui identifie le périphérique pour les applications.

De cette manière, lorsqu'une application nécessite l'utilisation d'un certain périphérique en rappelant son nom correspondant dans le répertoire « /dev », le système d'exploitation est capable, en remontant le long de la chaîne décrite précédemment, d'associer le pilote correct au périphérique pour sa gestion.

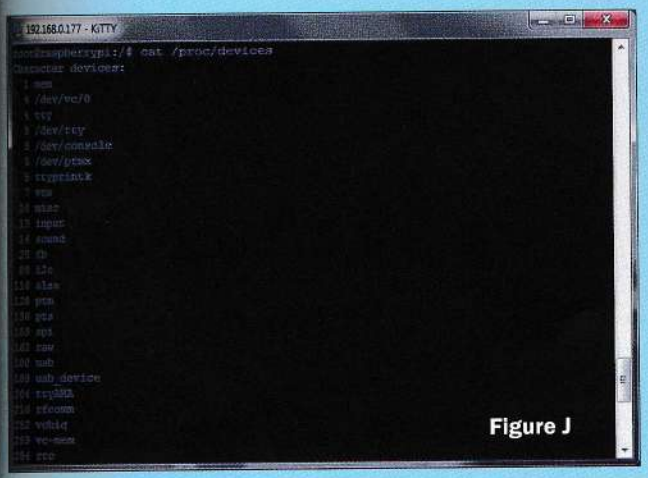


Figure J

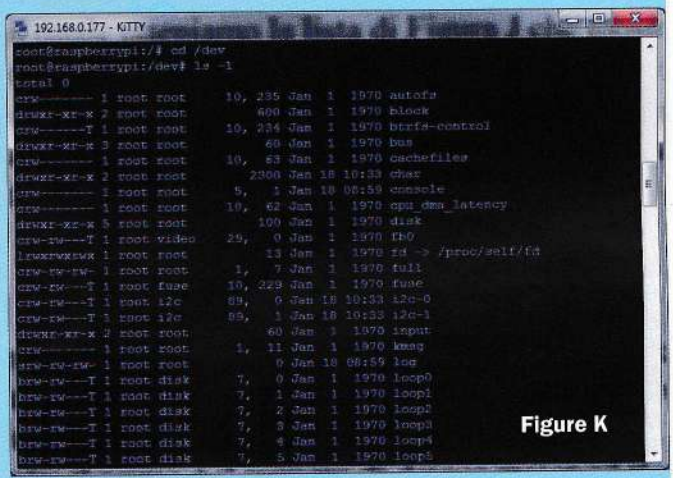


Figure K

Lorsque nous compilons un module pour un nouveau pilote de périphérique (Device Driver), l'association entre le nombre du périphérique et les entrées dans le répertoire « /dev » est créée en utilisant la commande « ***mknod*** », comme par exemple dans le cas du pilote (driver) du bus I2C :

mknod /dev/i2c-0 c 89 0

En fonction de la température, nous obtenons une valeur de 4 caractères, qui représentent la conversion en valeur numérique de la tension mesurée aux bornes de la résistance CTN, exprimée par 2 octets hexadécimaux, dans notre cas « 5769 » (voir la figure 12).

Rappelez-vous que le 1er octet (octet de poids faible) est le moins significatif, et le 2eme est le plus significatif (octet de poids fort). Avant de les utiliser, par exemple pour les convertir en une valeur décimale, nous devons les inverser pour obtenir dans notre cas : « 6957 ».

Pour convertir la valeur hexadécimale correspondant à la valeur décimale nous pouvons utiliser la commande :

```
echo 'ibase=16;obase=A;6957' | bc
```

qui retourne la valeur « 26967 » (voir la figure 13).

La commande affiche le résultat de la conversion de la base 16 à la base 10 (A en hexadécimal) de la valeur « 6957 » (il est obligatoire d'utiliser des majuscules pour les lettres) effectuée par le programme « bc ». Celui-ci est un programme générique sous forme de ligne de commande qui permet des calculs et des conversions dans toutes les bases, il vaut la peine d'être essayé car il peut résoudre de nombreux problèmes. Par exemple, pour la conversion binaire en hexadécimal, utilisons la commande :

```
echo 'ibase=2;obase=10000;bbbbbbbb' | bc
```

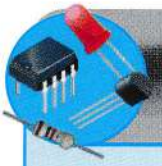
en remplaçant « bbbbbbbb » par la valeur que vous souhaitez convertir. Pour la conversion hexadécimal en binaire, utilisons la commande :

```
echo 'ibase=16;obase=2;hh' | bc
```

en remplaçant « hh » par la valeur que vous souhaitez convertir.

Nous pouvons effectuer le même test sur le canal CH2 du convertisseur A/D afin de vérifier le bon fonctionnement du circuit de la photorésistance.

Dans les prochains articles, nous approfondirons les calculs de conversion de la lecture numérique des valeurs de température et de luminosité, avec programme en Python. D'autre dans le **prochain numéro 129 d'E tronique et Loisirs Magazine nous liserons une carte d'extension un afficheur LCD pour le Raspber afin de créer une interface auton pour contrôler nos applications** la nécessité de connecter un moni un clavier et une souris.



Comment construi ce montage

La carte d'extension A/D pc RaspberryPi compatible Ardui ET1041 est disponible auprès **COMELEC**. Les typons des circu imprimés sont téléchargeables su **www.electroniquemagazine.co** dans le sommaire détaillé de revue numéro 128 section « Té charger ». Le programme est té chargeable sur : **www.raspberry electroniquemagazine.com**.

ET 726

Alimentation de laboratoire contrôlée par un PIC16F876A

.....de Mirco SEGATELLO

Cette alimentation professionnelle de laboratoire est capable de fournir une tension réglable de 0 à 25 V et un courant de sortie réglable de 0 à 2,5 A. Elle est gérée par un microcontrôleur PIC 16F876A, celui-ci régule à la fois la tension et le courant, contrôle la température du dissipateur de l'étage de sortie et le désactive en cas de surchauffe. Elle dispose d'un afficheur LCD utilisé à la fois comme « instrument de mesure » (lecture de la tension et du courant) et comme indicateur de messages d'erreurs (ou de services). Elle peut être gérée à partir d'un PC via le port série dont elle est équipée.

Une bonne alimentation est un instrument absolument indispensable dans tout laboratoire d'électronique, car elle vous permet de tester les circuits sur le banc d'essai ou de les réparer. Elle fournit aussi une tension de référence pour les circuits qui en ont besoin. Le projet que nous vous présentons dans cet article est une alimentation de laboratoire, qui sans être présomptueux, est vraiment professionnelle.

Tout d'abord concernant la régulation, nous pouvons **régler précisément** et de manière linéaire la **tension de sortie** continue ainsi que le **courant de sortie** de la valeur minimale à la valeur maximale.

Ensuite elle dispose de nombreuses caractéristiques telles que l'affichage **LCD rétroéclairé avec 2 lignes de 16 caractères** chacune pour visualiser tous les paramètres de fonctionnement, la **limitation de courant** définie par l'utilisateur et la **protection thermique** très utile pour éviter qu'en cas de forts courants ou dans des conditions particulièrement défavorables, l'étage de sortie ne soit endommagé.

Les prérogatives de la régulation et les nombreuses fonctions qu'offre l'alimentation, nous ont conduit à confier la gestion de l'ensemble à un **microcontrôleur PIC**. Celui-ci interprète et exécute les commandes, surveille le courant de sortie et en permanence la température des transistors de sortie.

Caractéristiques techniques

- **Tension de sortie réglable de 0 V à 25 V.**
- **Courant de sortie réglable de 0 A à 2 A.**
- **Régulation de la tension par pas de 100 mV via l'encodeur rotatif.**
- **Régulation du courant par pas de 10 mA via l'encodeur rotatif.**
- **Précision : meilleure que 2 % pour la tension, et meilleure que 5 % pour le courant.**
- **Tension d'ondulation en sortie inférieure à 10 $mV_{eff.}$**
- **Afficheur LCD 2 lignes de 16 caractères qui affiche :**
  - **les valeurs de consigne de la tension et du courant ;**
  - **la tension et le courant réels de sortie ;**
  - **les messages d'erreurs (ou de services).**
- **Possibilité de déconnecter instantanément la charge en appuyant sur un bouton.**
- **Stabilité de la tension de sortie maintenue : aucun pic anormal ou aucun changement brusque dans n'importe quelle situation, y compris pendant la phase d'allumage.**

En cas d'excès, il bloque l'alimentation en courant. Il gère également l'afficheur LCD qui est utilisé à la fois comme un voltmètre, un ampèremètre et un indicateur de messages d'erreurs (ou de services). Enfin, il reconnaît les commandes qui arrivent du PC via le port série dont l'alimentation est équipée.

Avant d'entrer dans les détails du fonctionnement du schéma électrique, vous devez comprendre la raison pour laquelle il est important d'avoir une alimentation réglable à la fois en tension et en courant.

Prenons un exemple simple, supposons que nous réglons la **tension de sortie à 10 V** et que nous utilisons une **résistance de 10** Ω, nous obtenons un courant de **1 A**. Si nous réglons la **limitation du courant à 1,5 A**, l'alimentation se comporte comme un **générateur de tension** simple. Maintenant supposons que nous réglons la **limitation du courant à 0,5 A**, l'alimentation se comporte comme un **régulateur de courant de 0,5 A** et par conséquent la chute de tension aux bornes de la résistance de charge (10 Ω) tombe à **5 V**.

Dans la plupart des cas, les circuits sont **alimentés en tension**, alors que dans d'autres cas plus particuliers la **régulation du courant est indispensable**, certains circuits électroniques doivent être alimentés à **courant constant**. Mais pas seulement certains circuits, les **batteries** aussi doivent être **chargées à courant constant**.

Cependant **dans un laboratoire** d'électronique, la **limitation du courant est souvent utilisée** lorsque nous effectuons des **tests sur des prototypes** qui peuvent ne pas fonctionner correctement et consommer plus de courant que prévu. Supposons que nous voulons alimenter pour la première fois un prototype nécessitant une tension de 5 V et consommant en théorie un courant de 100 mA.

Dans ce cas, nous réglons l'alimentation pour fournir la tension requise limitée à une valeur de courant légèrement supérieure (par exemple 150 mA). **Cela garantit que, même en cas d'erreurs lors du montage ou dans le schéma, le courant maximal circulant dans le circuit ne peut pas dépasser 150 mA**, car dans ce cas l'alimentation réduit la tension de sortie (ce qui protège aussi les composants du circuit). Dans le cas d'un **court-circuit**, cela **ne présente pas de danger pour l'alimentation**, car elle limite le courant à 150 mA.

Le fonctionnement

L'alimentation fonctionne à partir de la tension secteur **230 VAC** et **tire** la **puissance d'un transformateur principal**, pour fournir une **tension de sortie continue et stabilisée dont la valeur est contrôlée par le microcontrôleur par rapport à la position d'un encodeur rotatif**. Pour obtenir la valeur souhaitée le PIC commande de manière opportune l'étage de sortie à l'aide de **signaux PWM qui dépendent de la position de l'encodeur**. Le **courant consommé par la charge est constamment mesuré** afin de s'assurer que le **courant ne dépasse pas le seuil maximum fixé** par les réglages.

Pour **limiter la dissipation thermique de l'étage de sortie**, ce qui signifie une perte d'énergie sous forme de chaleur dans les jonctions des transistors qui alimentent la charge, le microcontrôleur **sélectionne la tension d'entrée en fonction de ce que nous voulons en sortie**. Pour cela nous disposons d'un transformateur de puissance avec un secondaire à double prise, à partir laquelle nous **prélevons la tension la plus basse** lorsque en sortie nous voulons une tension comprise entre **0** et **8 V**, et la tension la plus haute si nous voulons plus **à partir de 9 V.**

La connexion du transformateur est **gérée par le PIC à l'aide d'un relais qui est relié aux enroulements secondaires**, en fonction des besoins seul enroulement ou les deux sont connectés.

L'**hystérésis** est de **1 V** et garantit une **stabilité lors des phases de commutation des enroulements**. En effet, nous voulons une tension de sortie **supérieure à 9 V**, le relais se déclenche et **connecte les 2 enroulements secondaires**. Pour qu'**un seul secondaire soit connecté**, nous devons descendre **en dessous** de **8 V** (cela évite des commutations intempestives du relais).

L'avantage est facilement compréhensible à l'aide de l'exemple suivant :

- supposons que nous voulons obtenir en sortie une tension de 5 V à partir d'une tension d'entrée de 20 V. Si le courant consommé en sortie par la charge est de 1 A, la puissance prélevée en sortie sera de 5 W. Le régulateur de puissance (ici l'étage de sortie) maintient la tension de sortie à 5 V, mais il est traversé par le même courant, cependant il est alimenté par une tension de 20 V. A ses bornes il présente donc une chute de tension de : 20 - 5 = 15 V. La puissance qu'il doit dissiper est donc de 15 x 1 = 15 W.

- si au contraire, pour avoir 5 V en sortie nous partons d'une tension d'entrée de 12 V, la chute de tension aux bornes du régulateur est seulement de 7 V et, avec le même courant, nous obtenons une puissance dissipée de seulement 7 W.

Revenons sur les fonctions, nous voyons que **la tension de sortie peut être appliquée ou non à l'aide d'un bouton.** Cette fonction est particulièrement utile lorsque **vous testez un prototype et que vous voulez effectuer une modification sans retirer tous les fils de l'alimentation.** A l'aide du bouton vous isolez le circuit à tester et ensuite vous pouvez de nouveau l'alimenter pour vérifier le fonctionnement. Bien sûr, vous pouvez débrancher et rebrancher les fils, mais cela est gênant lorsque vous manipulez des charges inductives et/ou capacitives car il y a des risques d'étincelles.

Regardons maintenant la gestion de la température, comme mentionné ci-dessus, le **microcontrôleur déconnecte l'étage de sortie si la température du transistor de puissance devient supérieure à 50 °C.** Nous avons muni l'alimentation d'un **connecteur** pour un **ventilateur optionnel** (FAN) qui, s'il est connecté, permet un meilleur refroidissement des composants.

La dernière **particularité** de **cette alimentation** est qu'elle peut être **commandée** par un **PC** à l'aide d'un **programme spécial qui fonctionne sous Microsoft Windows** (téléchargeable sur notre site). A cet égard nous avons mis en œuvre une **interface série RS232 opto-isolée.**

Schéma synoptique des différentes sections de l'alimentation

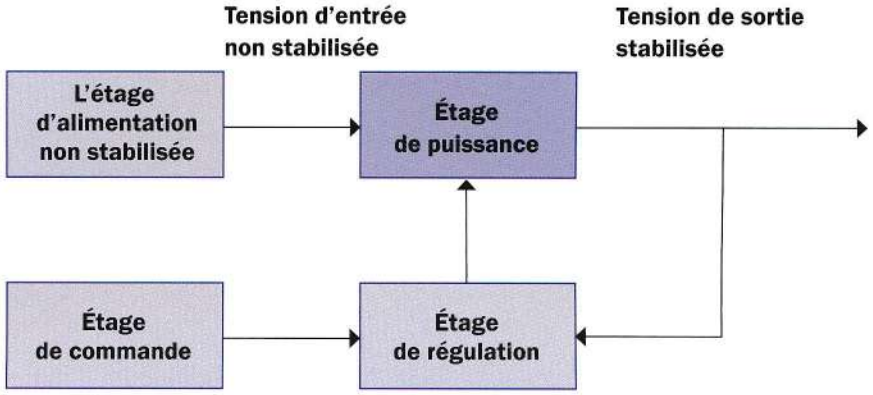


Figure 1 : L'étage d'alimentation fournit les tensions nécessaires aux étages de puissance et de commande. L'étage de commande génère la tension de référence, gère l'afficheur LCD, mesure la tension et le courant. Les étages de régulation et de puissance effectuent la stabilisation de la tension et du courant dans la charge.

Schéma électrique

Après la présentation générale, nous allons entrer dans le détail. Pour bien comprendre le fonctionnement de l'alimentation, référez-vous au schéma synoptique de la figure 1, dans laquelle nous avons décomposé le schéma en **3 sections** distinctes : l'**étage d'alimentation non stabilisée**, l'**étage de régulation** et de **puissance**, et enfin l'**étage de commande**.

1. **L'étage d'alimentation non stabilisée** :

Il se compose d'un **transformateur toroïdal** comprenant **deux secondaires** de **12 VAC** chacun et d'une puissance de **80 VA**. La tension est prélevée à partir de l'un des deux secondaires en fonction de l'**état du relais K1**, qui pour diminuer la perte de dissipation dans l'étage de puissance, **sélectionne la tension d'entrée en fonction de la tension de sortie**. Plus précisément, nous nous utilisons les deux secondaires en série lorsque nous avons besoin de tensions supérieures à 9 V et un seul secondaire pour des tensions de sortie inférieures à 8 V environ.

Il existe aussi un **deuxième étage d'alimentation de faible puissance composé du transformateur TF1** ayant une tension de sortie de 2 x 6 V (3VA). Il fournit les **tensions nécessaires** c'est-à-dire, « **Vcc+** » pour l'**étage de régulation** qui est **filtrée mais non stabilisée**, « **Vcc-** » pour le **ventilateur** et « **VS** » **stabilisée** par un **LM7805** pour l'**étage de commande** à microcontrôleur.

2. **L'étage de régulation et de puissance** :

Cet **étage** est **constitué** de **deux amplificateurs opérationnels U2** et **U3** (TL082) afin d'assurer les **ajustements de la tension et du courant de sortie**. Une **partie de la tension** de **sortie** présente sur le bornier **OUT est prélevée par le pont diviseur** formé par les résistances **R10**, **R11** et **R12** par rapport à la **tension de référence** fournie par **U4a** (V_{ref}**V**) et appliquée à la broche **3** de **U2a**. Celui-ci compare la tension sur sa broche 3 avec celle présente sur sa broche 2, qui est **équivalente à celle de la résistance de shunt** (résistances en parallèle R13 à R17) et qui suit l'**évolution du courant circulant dans la charge**.

La **comparaison des tensions** (broches 2 et 3) **détermine la variation de la sortie** de l'amplificateur opérationnel qui, à son tour, fait varier la **polarisation de l'étage de puissance** (Q2 et Q3) et **stabilise** ainsi la **tension de sortie** de l'alimentation. **Toutes les résistances ont une tolérance de 1% afin d'assurer**

Schéma électrique

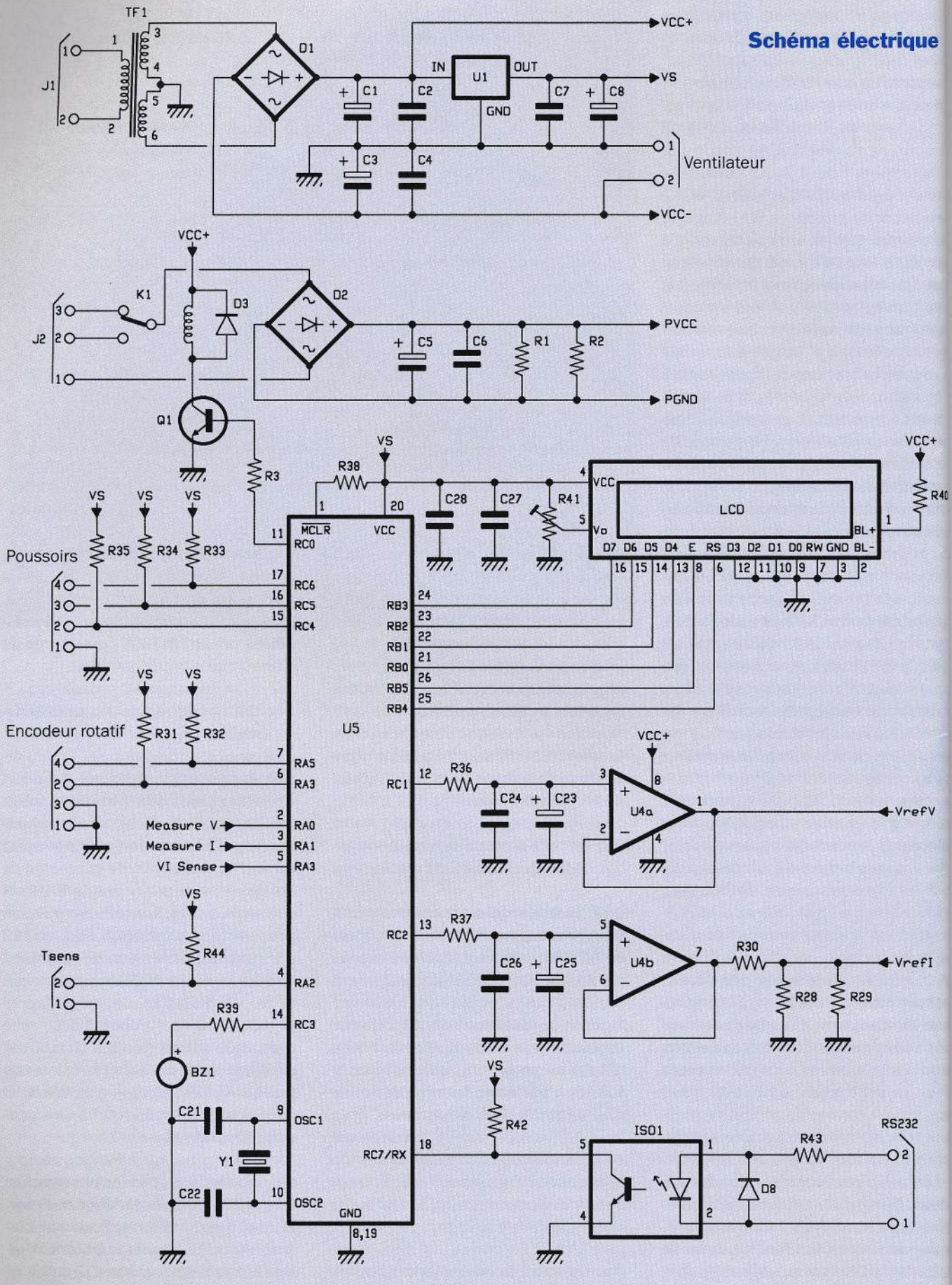
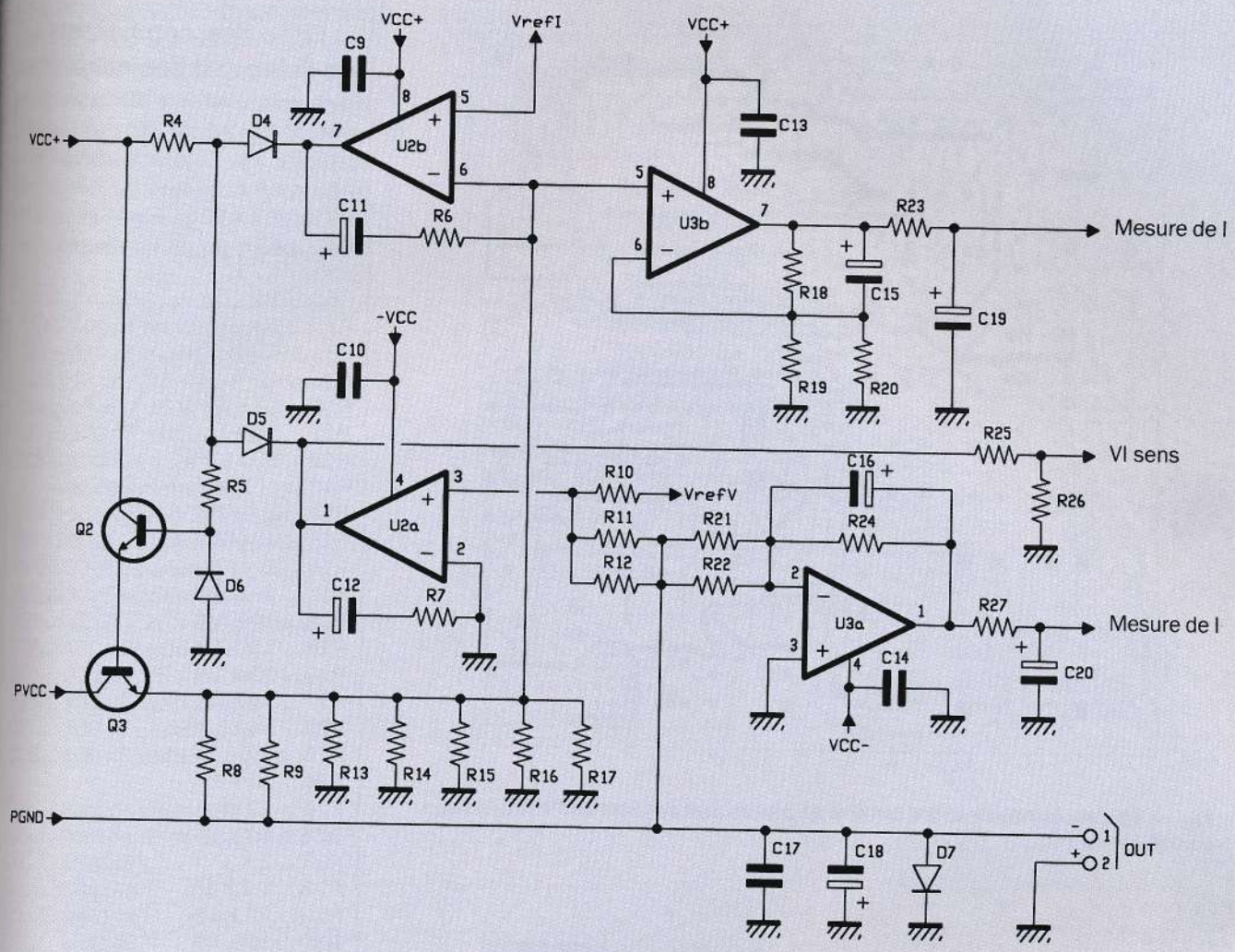


Figure 1a : schéma électrique de l'alimentation contrôlée par PIC.



la plus grande précision possible de la tension de sortie. Le circuit de stabilisation est également commandé à partir de la sortie de **U2b**, qui est chargé de la **régulation du courant**.

Cette dernière, basée sur la lecture de la chute de tension aux bornes du shunt (résistances en parallèle R13 à R17), est effectuée en même temps que la régulation en tension. L'amplificateur opérationnel **U2a fournit le courant de base à Q2** (et par conséquent à Q3) pour **maintenir une tension de sortie stable**, même lorsque la **charge varie**. **U2b limite le courant en fonction de la valeur de référence** ($V_{ref}I$) obtenue par **U4b**, même si le **courant de la charge dépasse la limite fixée**.

Les deux amplificateurs opérationnels **U2a** et **U2b gèrent** ainsi **le courant de base de Q2** par l'intermédiaire des deux diodes **D4** et **D5**, qui se comportent comme une **porte de type « OR » (OU)**, laissant le contrôle de la base du transistor **Q2** à l'amplificateur opérationnel avec la tension de sortie la plus basse ou à celui qui opère le réglage de la tension ou du courant nécessaire à cet instant. Remarquez la présence de nombreuses résistances de précision et certaines d'entre elles sont en parallèle, ce qui assure une plus grande précision dans le réglage de la tension et du courant dans chaque situation.

Pour **réaliser le shunt** nous avons préféré **utiliser 5 résistances en parallèle au lieu d'une seule de puissance**, ce qui permet d'obtenir un shunt ayant une **bonne précision** (les résistances de faible puissance ont une tolérance moins grande que celles de puissance).

Notez que l'utilisation de composants de précision a permis de réaliser un circuit qui ne nécessite pas d'étalonnage, même dans les deux étages utilisés pour la mesure de la tension (mesure de V prélevée à la sortie de U3b) et pour la mesure du courant de la charge (mesure de V à la sortie de U3a).

Notez également que **U3a** et **U3b** réalisent **deux fonctions** : ils **détectent** la **tension** et le **courant** et les adaptent de sorte qu'ils puissent être appliqués **directement sur les entrées analogiques du PIC**, car celles-ci ne peuvent accepter que des tensions positives.

3. L'étage de commande :

L'**étage de commande** doit fournir les **deux tensions de référence**, l'une pour la **régulation de la tension** ($V_{ref}V$: sortie U4a) et l'autre pour la **régulation du courant** ($V_{ref}I$: sortie U4b). Le cœur de cet étage est le microcontrôleur **PIC16F876A** qui, lorsqu'il est correctement programmé, **gère toutes les fonctions de l'alimentation**. Les **deux sorties PWM** du **PIC** sont filtrées à travers des filtres passe-bas qui permettent d'obtenir **deux tensions de référence précises** qui sont « bufférisées » et découplées par les deux amplificateurs opérationnels **U4**, formant ainsi l'**étage de régulation**.

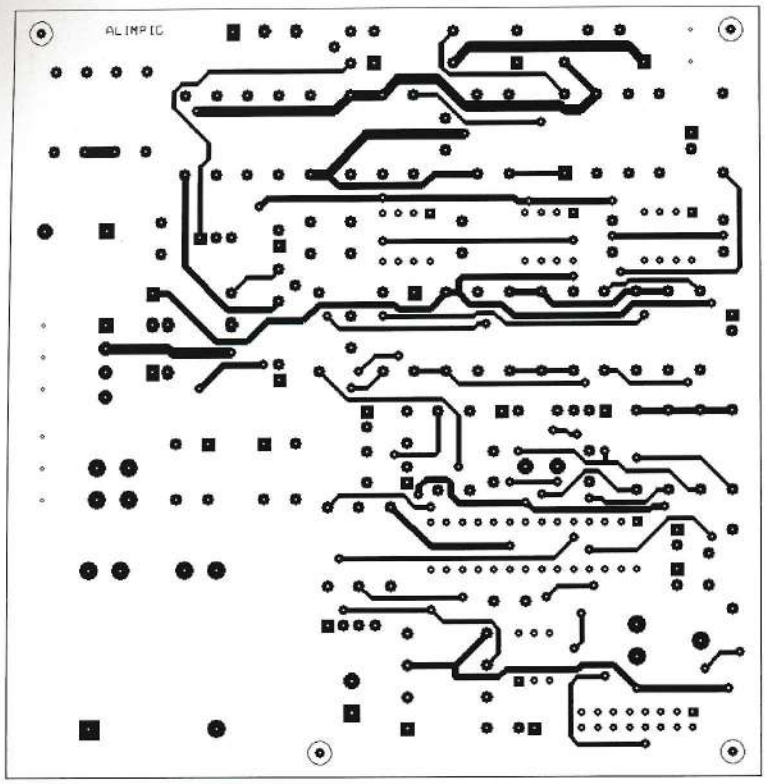


Figure 1b : circuit imprimé à l'échelle 1 : 1 côté composants de l'alimentation contrôlée par PIC.

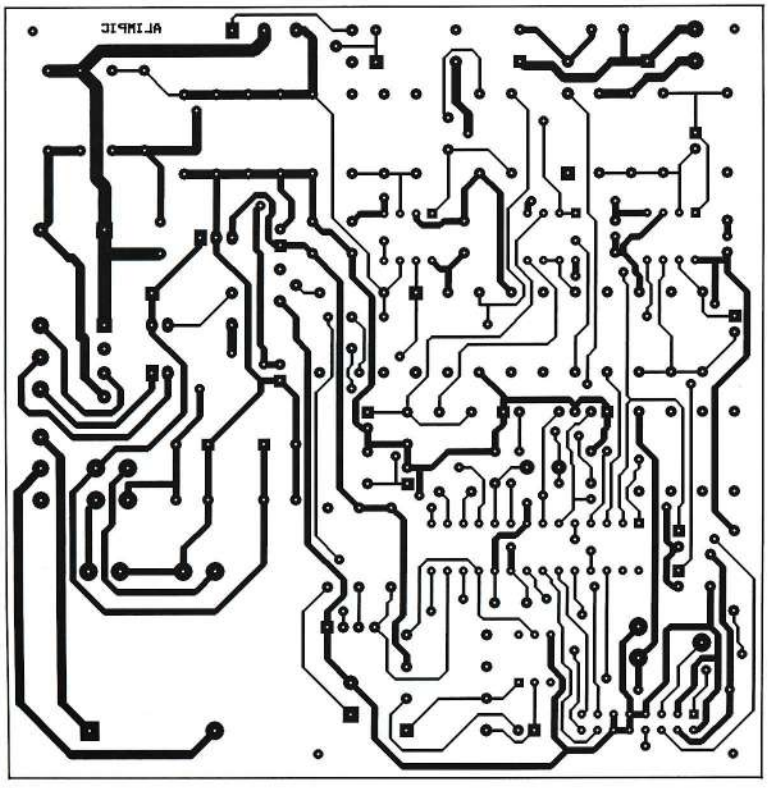


Figure 1c : circuit imprimé à l'échelle 1 : 1 côté soudures de l'alimentation contrôlée par PIC.

Liste des composants de l'alimentation contrôlée

R1..... 4,7 k
R2..... 4,7 k
R3..... 4,7 k
R4..... 47 k
R5..... 4,7 k
R6..... 10
R7..... 10
R8..... 4,7 k
R9..... 4,7 k
R10.... 10 k 1%
R11 ... 56 k 1%
R12 ... 560 k 1%
R13 ... 1 1/2W
R14.... 1 1/2W
R15 ... 1 1/2W
R16.... 1 1/2W
R17.... 1 1/2W
R18 ... 10 k 1%
R19 ... 1,2 k 1%
R20 ... 22 k 1%
R21.... 56 k 1%
R22 ... 560 k 1%
R23 ... 4,7 k
R24.... 10 k 1%
R25 ... 10 k
R26 ... 10 k
R27.... 1 k
R28 ... 1,2 k 1%
R29 ... 22 k 1%
R30 ... 10 k 1%
R31.... 10 k
R32 ... 10 k
R33 ... 10 k
R34 ... 10 k
R35 ... 10 k
R36 ... 10 k
R37.... 10 k
R38 ... 10 k
R39 ... 100
R40 ... 330
R41.... Trimmer multi-tour 10 K
R42 ... 2,2 k
R43 ... 3,3 k

C1..... 1000 µF / 25 V électrolytique
C2..... 100 nF multicouche
C3..... 1000 µF / 25 V électrolytique
C4..... 100 nF multicouche
C5..... 4700 µF / 50 V électrolytique
C6..... 100 nF multicouche
C7..... 100 nF multicouche
C8..... 10 µF / 25 V électrolytique
C9..... 100 nF multicouche
C10.... 100 nF multicouche
C11.... 22 µF / 25 V électrolytique
C12.... 100 µF / 25 V électrolytique

Pour les réglages des valeurs de la tension et du courant nous n'avons pas utilisé des boutons classiques, mais un **encodeur rotatif plus fin et plus précis** (figures 5a et 5b), qui **en combinaison avec 3 boutons** (figure 2 : les 3 boutons SW1 à SW3 sont reliés au connecteur « Poussoirs » broches RC4 à RC6), permet d'**obtenir un réglage très précis**

C13.... 100 nF multicouche
C14.... 100 nF multicouche
C15.... 10 µF / 25 V électrolytique
C16.... 10 µF / 25 V électrolytique
C17.... 100 nF multicouche
C18.... 1000 µF / 25 V électrolytique
C19.... 10 µF / 25 V électrolytique
C20.... 10 µF / 25 V électrolytique
C21.... 22pF céramique
C22.... 22pF céramique
C23.... 22 µF / 25 V électrolytique
C24.... 100 nF multicouche
C25.... 22 µF / 25 V électrolytique
C26.... 100 nF multicouche
C27.... 100 nF multicouche
C28.... 100 nF multicouche

D1..... Pont de diode 100 V / 1 A
D2..... Pont de diode 100 V / 5 A
D3..... 1N4007
D4..... 1N4148
D5..... 1N4148
D6..... 1N4148
D7..... 1N4007
D8..... 1N4148

ISO1.. 4N35
U1..... 7805
U2..... TL082
U3..... TL082
U4..... TL082
U5..... PIC16F876A (MF726)
Y1..... Quartz 4 MHz
K1..... Relais 12 V 1C 3 A
TF1.... Transformateur 230 VAC / 2 x 6 V 3 VA pour ci
Q1..... BC337
Q2..... BC337
LCD.... Afficheur 2 lignes 16 caractères

Divers

Support circuit 2 x 4 broches (3 pièces)
Support circuit 2 x 3 broches (1 pièce)
Support circuit 2 x 14 broches (1 pièce)
Connecteur dil 16 broches
Bornier 3 pôles (2 pièces)
Bornier 2 pôles (1 pièces)

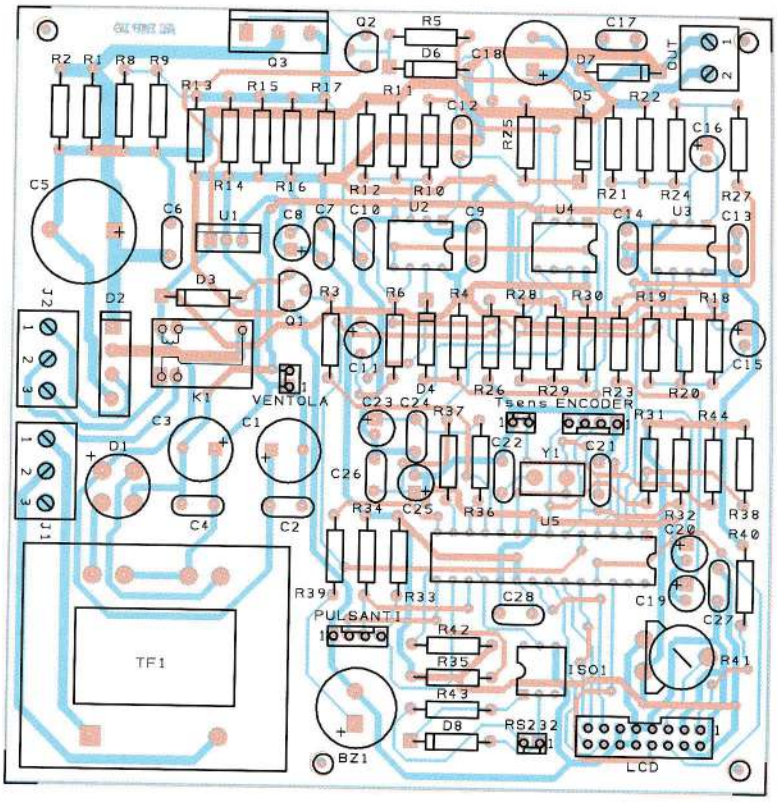


Figure 1d : schéma de câblage de l'alimentation contrôlée par PIC.

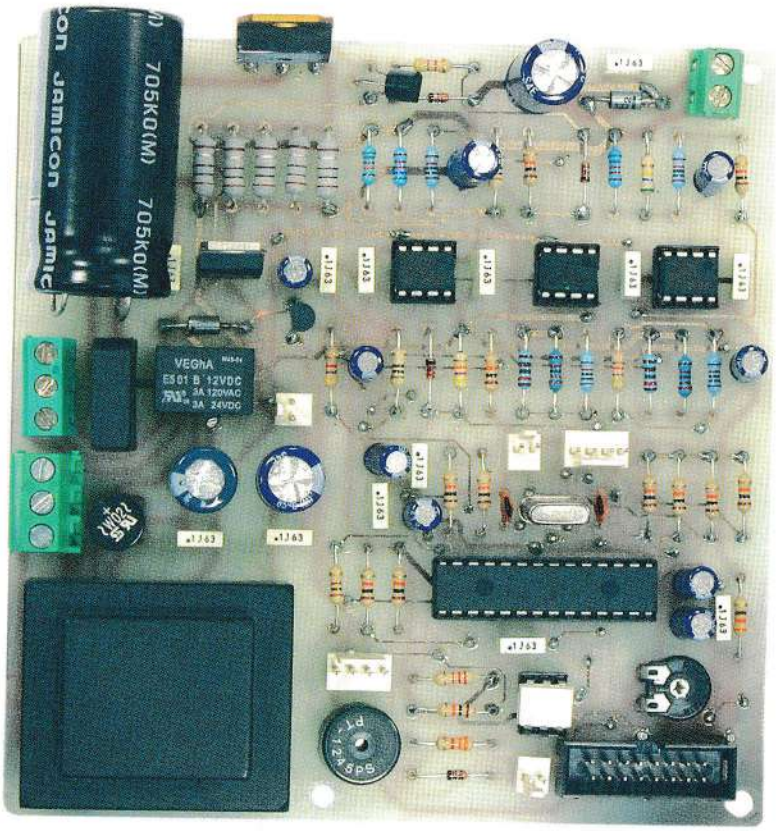


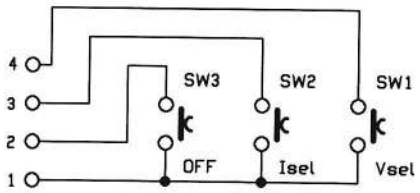
Figure 1e : photo de l'un de nos prototypes de l'alimentation contrôlée par PIC.

et rapide de tous les paramètres. Un **afficheur LCD rétroéclairé** de **2 lignes** de 16 caractères fournit toutes les **indications** de fonctionnement.

Dans l'étage de commande est également présent le **port de communication série** utilisé pour gérer l'**alimentation à partir d'un PC**. Ce port fonctionne **uniquement en réception**. Nous avons connecté l'interface **COM** de l'ordinateur avec la ligne **RX** de l'**UART** du microcontrôleur à l'aide d'un **optocoupleur**, dont la tâche est d'**adapter les niveaux RS232** à ceux **TTL** tolérés par le PIC.

Parce que comme dans toute **alimentation linéaire**, la **stabilisation est effectuée en dissipant la puissance en excès sous forme de chaleur**, nous avons muni l'alimentation d'un « **interrupteur thermique** » pour protéger le transistor **Q3** de sortie. Ce composant qui est relié au connecteur « **Tsens** », se comporte comme un **interrupteur fermé pour des températures normales**. Lorsque son boîtier atteint une température préréglée (en usine), il **ouvre son contact interne**.

Pour notre prototype, nous avons utilisé un interrupteur thermique à réarmement automatique dont la température d'activation est de 50 °C, mais il existe aussi des modèles 70 °C et 100 °C.



Réalisation pratique

Après avoir clarifié les aspects théoriques, passons à la pratique. Pour assembler cette alimentation vous devez en premier réaliser le circuit imprimé double face. Téléchargez sur notre site **www.electroniquemagazine.com** dans le **sommaire détaillé de la revue 128** à l'onglet « **Télécharger** » les typons du circuit imprimé. Vous y trouverez aussi le **programme** « **.hex** » du **PIC** ainsi que le **logiciel de gestion pour Windows**.

Une fois les deux faces gravées, à l'aide de morceaux de pattes de composants, commencez par souder les « vias » (petit trou que l'on complète par un lien métallique afin d'assurer la continuité des pistes d'une face à l'autre). Les composants doivent être soudés sur le côté composants, c'est-à-dire sur la partie supérieure, de manière à simplifier les opérations d'assemblage.

Une fois que tous les « vias » sont soudés, continuez par souder les composants ayant des profils bas : résistances, diodes (attention à l'orientation de la bague), supports de circuits intégrés, condensateurs non polarisés et borniers. Ensuite continuez avec les condensateurs polarisés (le négatif « - » est indiqué sur le boitier), le transistor Q3 et le transformateur.

Soudez maintenant le câble de l'afficheur LCD, à l'aide du brochage qui est indiqué sur le plan de montage du LCD visible sur la figure 6, en gardant à l'esprit que le résultat dépend du type d'afficheur utilisé. **Si vous n'avez pas l'intention d'utiliser le capteur de température**, il faut **court-circuiter les broches** « **Tsens** » de son connecteur, sinon la broche **RA2 du microcontrôleur croira que le capteur a atteint la température de déclenchement** par la présence de la tension due à la résistance R44 de pull-up (+ 5 V).

Pour la connexion des trois boutons poussoirs, reportez-vous au schéma de la figure 2. Le câblage de l'encodeur rotatif est indiqué sur la figure 5a (le composant est vu de face, c'est-à-dire que l'axe sort face à l'observateur). Le câble utilisé doit être assez long pour atteindre le connecteur de l'encodeur sur le circuit imprimé. Les derniers éléments à souder sont le transformateur et le condensateur de filtrage C5. Celui-ci est conçu pour être monté verticalement, cependant il est préférable de le souder en position horizontale, comme indiqué sur la figure 4, de manière à limiter la hauteur du circuit et ainsi utiliser un boitier de dimensions réduites.

Faites attention en pliant les pattes de C5 à ne pas les casser. Une fois les tests effectués, vous mettrez des points de colle pour maintenir C5 sur le circuit imprimé.

Pour le câblage correct de toutes le connexions de l'alimentation, repo tez-vous au schéma de câblage illust sur la figure 3.

Montage de la mécanique

Le circuit, une fois terminé et testé, pe être monté dans un boitier comme cel utilisé sur les photos de cet article. Il e important que le boitier soit en mét et pourvu de fentes pour la ventilatio afin de favoriser l'évacuation de la cha eur produite par les composants. Sur circuit imprimé est prévu un connecte pour alimenter un petit ventilateur en 1 VDC (maximum 100 mA), dans le ca où vous voulez augmenter la ventilatio interne. Cette solution est nécessair en cas d'usage intensif de l'alimenta tion, c'est-à-dire avec un courant sorti important pendant une longue durée. L capteur de température doit être fixé s le dissipateur, aussi près que possibl du transistor de puissance.

A l'arrière du boitier, il est conseillé d'i staller une prise à trois broches de typ VDE (comme sur les PC) avec un fusibl intégré de 500 mA pour protéger e cas de court-circuit le réseau 230 VAC

Une attention particulière doit êtr portée à la fixation du dissipateur. S votre boitier est assez grand, vous pou vez disposer le radiateur à l'intérieur e contact direct avec le transistor, san élément en mica isolant (mettez u peu de graisse silicone pour améliore l'échange thermique).

La circulation de l'air qui est limité dans le boitier, doit être améliorée pa un ventilateur fixé par exemple à l'a rière du boitier. Si nécessaire, le tran sistor Q3 peut être disposé hors d circuit imprimé et relié par une napp de 3 fils (au moins du fil multibrins d 0,75 mm²) sans dépasser une longue excessive (20 cm environ).

Le transistor Q3 peut également êtr monté sur un dissipateur situé à l'e térieur du boiter, c'est la solution qu nous avons adoptée pour notre prot type. Dans ce cas, vous devez placer un feuille de mica isolante entre la part métallique du transistor et le dissipate (mettez un peu de graisse silicone).

Figure 3 de câbla les élé l'alimen

Figure 4 à souder C5. Il es capacité ment due pas soud monter h les patte

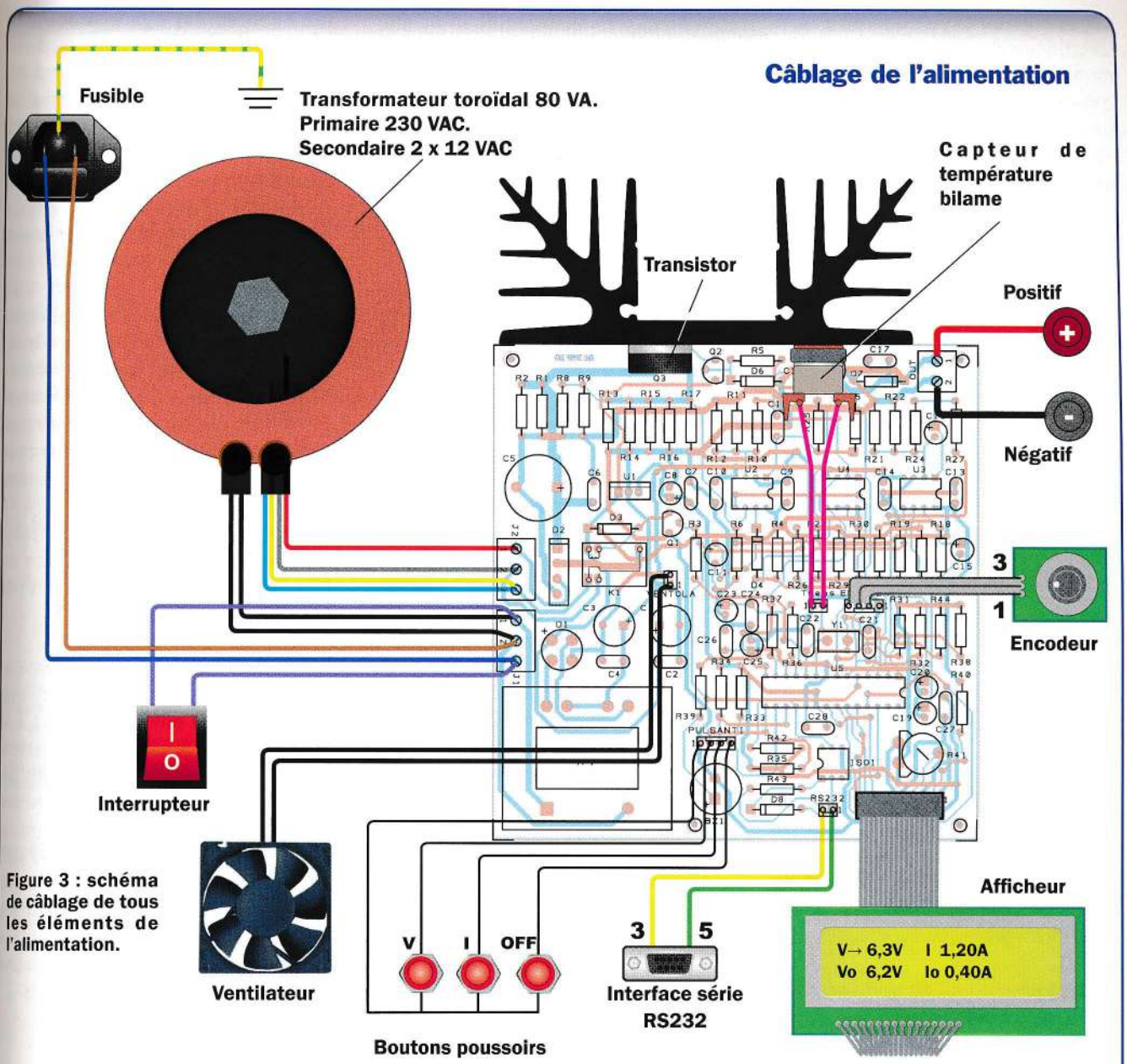


Figure 3 : schéma de câblage de tous les éléments de l'alimentation.

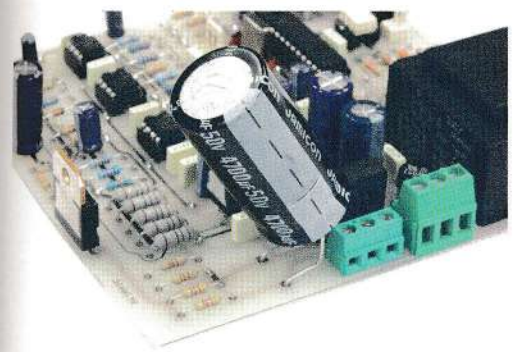
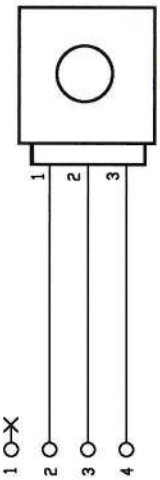


Figure 4 : le circuit imprimé est conçu de manière à souder verticalement le condensateur de filtrage C5. Il est de grandes dimensions en raison de sa capacité élevée. Si pour des raisons d'encombrement dues aux dimensions du boitier vous ne pouvez pas souder C5 verticalement, vous pouvez alors le monter horizontalement, en prenant soin de plier les pattes comme montré dans l'image ci-dessus.



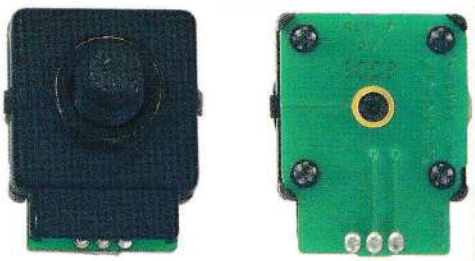


Figure 5a et 5b : schéma de câblage de l'encodeur (vu du côté de l'axe) au connecteur « ENCODEUR » sur le circuit imprimé (5a). L'encodeur ressemble à un potentiomètre, il est équipé d'un écrou de fixation et de 3 contacts, dont l'un est la masse (5b).

L'ensemble du circuit doit être isolé électriquement du boitier. Pour cela vérifiez à l'aide d'un multimètre, sur la position « diode », que la résistance entre le boitier en différents points et les douilles de sortie ont une valeur infinie (normalement le multimètre ne doit pas biper).

Figure 6 : l'afficheur est relié à un connecteur à l'aide d'un câble plat de 16 conducteurs. La broche 1 du connecteur doit être reliée à la broche 1 de l'afficheur et la numérotation suit la séquence 1 à 16. La connexion de l'afficheur dépend de son brochage, donc vous devez vous référer aux caractéristiques de l'afficheur. Pour le brochage du connecteur, consultez le schéma de câblage de l'alimentation.

Les tests

Tout d'abord **vérifiez soigneusement toutes les soudures** et le **placement des composants**. Ensuite **sans mettre aucun circuit intégré dans les supports, ni l'afficheur**, et **sans brancher le transformateur torique de puissance**, vérifiez les tensions fournies au primaire du transformateur TF1. **Attention vous êtes relié au secteur 230 VAC**, le **multimètre doit être sur le calibre V~** (alternatif 750 V ou 1000 V) et les broches parfaitement isolées.

Vous devez mesurer aux environs de **230 VAC**. Vous pouvez ensuite mesurer les tensions après le pont de diode D1, le **multimètre doit être sur le calibre V⎓** (continu 20 V ou plus). Le « commun » du multimètre doit être relié à la masse qui se trouve sur le « - » de C1 et le fil rouge (+) sur « VCC+ » (+ de C1), « VS » (+ de C8) et « VCC- » (- de C3).

Pour « **VCC+** » vous devez avoir une tension de **10 VDC**, pour « **VCC-** » aux environs de **– 10 VDC** et pour « **VS** » la tension doit être de **+ 5VDC**. **Vérifiez également sur le support du PIC que vous avez + 5VDC entre les broches 20 et 8,19**. De même vous devez avoir sur le support de U4 + 10VDC entre les broches 8 et 4 et pour U2 ± 10 VDC entre les broches 8 et 4.

Court-circuitez les broches 11 et 20 du support du PIC pour vérifier que la commande du relais fonctionne correctement.

A ce stade, **éteignez l'alimentation et retirez la prise du secteur**. **Attendez que les condensateurs se déchargent** et insérez tous les circuits intégrés dans les supports ainsi que le connecteur de l'afficheur LCD. **Allumez et ajustez le contraste de l'afficheur à l'aide du trimmer R41**.

L'afficheur doit indiquer les valeurs de la tension et du courant (toutes les 2 à 0.0 V), elles peuvent être ajustées par le codeur rotatif. Faites varier la position du codeur si nécessaire, et vérifiez que les boutons poussoirs fonctionnent correctement.

Maintenant éteignez l'alimentation et connectez le transformateur torique de puissance. Rallumez et **vérifiez que la tension de sortie présente sur les douilles est égale** (à peu près à la **valeur indiquée par l'afficheur**. Vous pouvez tester le circuit sans dissipateur pour Q3, en prenant soin de ne pas dépasser quelques mA. Les 2 amplificateurs opérationnels sont de type TL082, mais il est préférable de choisir un modèle à faible « offset » (faible tension de décalage) tel que des LF412.

La tension d'ondulation à la sortie (limitée à environ 8 mV crête à crête comme le montre la figure 7) est essentiellement due à l'ondulation résiduelle des tensions de référence des sorties PWM. Cette valeur peut être réduite en augmentant les valeurs des condensateurs C23 et C25 à 47 µF ou jusqu'à 100 µF, en acceptant l'inconvénient d'avoir un temps de réponse plus grand lors du réglage de la tension et du courant.

Après avoir monté Q3 sur le dissipateur, faites un test avec une charge qui consomme environ 0,5 A à 1 A et contrôlez que la température reste à une valeur acceptable. Les points critiques sont le pont de diode et le transistor Q3 de sortie. Le courant maximal de sortie qui peut aller jusqu'à un maximum de 2,5 A, doit être prélevé sur de courtes périodes pour que les composants parviennent à dissiper la chaleur produite, sauf si vous avez une excellente ventilation.

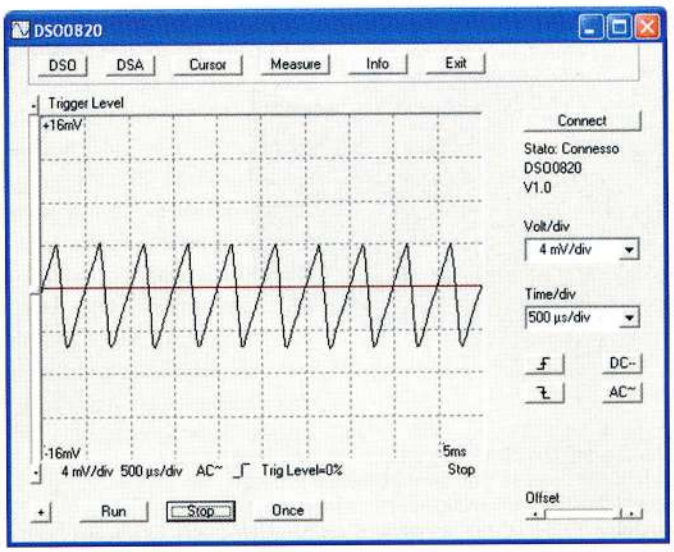


Figure 7 : ondulation de la tension de sortie pour Vout = 5 V et Iout = 0,6 A. La valeur mesurée qui est d'environ 8 mV crête à crête provient des sorties PWM de U4a et U4b. Elle peut être réduite en augmentant les valeurs de C23 et C25, mais cela ralentit le temps de réponse des valeurs de consignes imposées.

Utilisation

Lorsque vous allumez l'alimentation, l'**afficheur LCD indique le nom du firmware et sa version**. Au bout de **2 secondes**, temps nécessaire pour stabiliser la tension, l'**alimentation passe à l'état « OFF »** et affiche les valeurs de la tension et du courant mémorisées précédemment, **sans activer la sortie** (figure 8A, « OFF » signifie que la tension et le courant sont à 0). La **sortie prend les valeurs affichées par le LCD en appuyant sur les boutons « Vsel »** ou « **Isel** ».

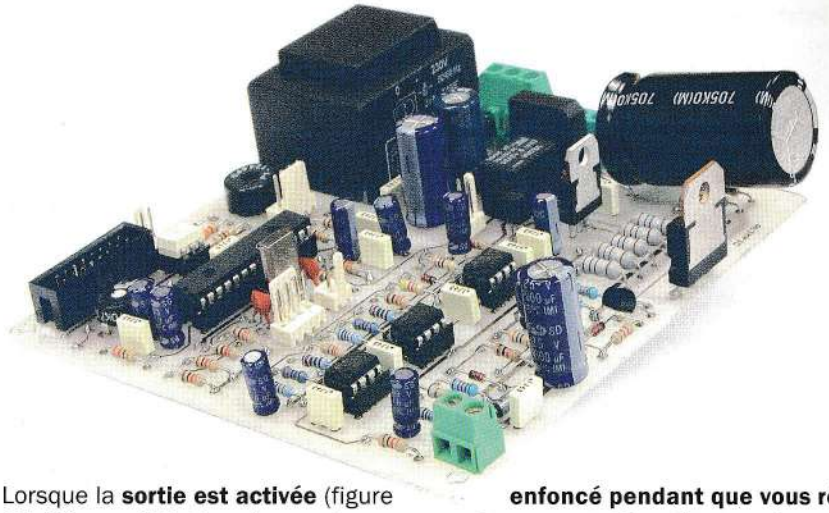


Fig. 8A

Fig. 8B

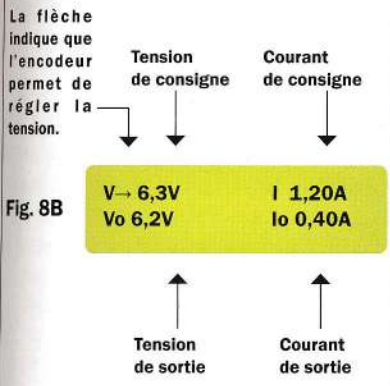


Fig. 8C

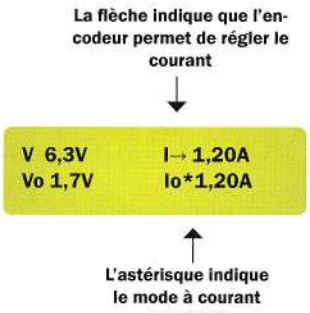


Fig. 8D

Lorsque la **sortie est activée** (figure 8B), l'écran affiche les valeurs suivantes :

- **tension de consigne**
- **courant de consigne**
- **tension réelle en sortie**
- **courant réel en sortie**

Dans le cas où le **courant de sortie dépasse la valeur de consigne**, l'**alimentation passe automatiquement en mode de fonctionnement** à « **courant constant** », ce qui est indiqué par un **astérisque sur l'afficheur à côté de l'indication « Io »** (figure 8C). Le bouton « **Vsel** » permet de **modifier la tension de consigne en utilisant l'encodeur**. Sur l'afficheur à côté du symbole « **V** » apparaît le symbole « → » pour indiquer ce mode.

De même, en appuyant sur le bouton « **Isel** », l'**encodeur modifie la valeur du courant de consigne** et la flèche apparaît à côté du symbole « **I** ». En faisant tourner l'encodeur, vous pouvez modifier la valeur de consigne affichée par pas d'un chiffre, ce qui correspond à **0,1 V** pour la tension et **0,01 A** pour le courant. Si vous **maintenez le bouton « Vsel » enfoncé tout en ajustant la tension**, l'**incrémentation se fait par pas de 1 V**. De même, si vous **maintenez le bouton « Isel » enfoncé pendant que vous réglez le courant**, l'**incrémentation se fait par pas de 0,1 A**.

En appuyant sur le bouton « **OFF** » la **sortie est désactivée**, et la **tension est immédiatement amenée à 0 V**. L'écran affiche **toujours les valeurs de consignes** de la tension et du courant, mais la **deuxième ligne indique « OFF »**. Vous pouvez modifier les valeurs de consignes de la tension et du courant.

Pour **réinitialiser la sortie à la valeur précédente**, appuyez simplement sur le bouton « **Vsel** » ou le bouton « **Isel** ». L'état « **OFF** » implique la **mémorisation des valeurs de consignes**, valeurs qui seront restituées au prochain allumage.

L'**intervention de l'interrupteur thermique met l'alimentation immédiatement à l'état « OFF »**. La **tension de sortie tombe à 0 V**, et le message « **Temp over!** » apparaît sur l'écran (figure 8D). Pour **tester cette fonction**, il suffit de **débrancher le connecteur du capteur thermique**. Si ce message apparaît pendant une utilisation normale, il est conseillé de couper l'alimentation et de chercher la cause de la surchauffe. Laissez refroidir afin que la sonde thermique restaure le contact.

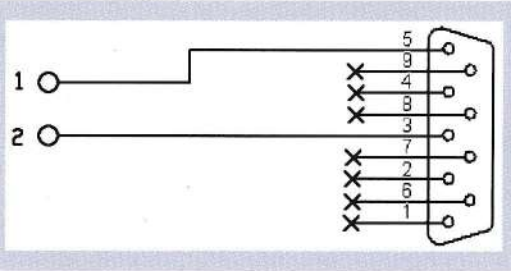


Figure 9 : le connecteur DB9 est relié par 2 fils au connecteur RS-232 du circuit imprimé.

Utilisation de l'interface série

L'alimentation **dispose d'une liaison opto-isolée** qui permet de la **connecter au port série** d'un PC. Vous pouvez envoyer les paramètres de fonctionnement à distance via le PC. Pour cela il est nécessaire de **relier à l'aide de 2 fils le connecteur RS232** présent sur le circuit imprimé à un **connecteur DB9 femelle qui sera ensuite fixé à l'arrière du boitier**. Le **schéma de câblage est visible sur la figure 9**. Faites attention en particulier à l'optocoupleur utilisé (ISO1) qui est très commun, mais pas très sensible.

Par conséquent, avec des signaux séries ayant une tension basse en dessous de 6 V, la gestion des signaux peut ne pas être optimale. Dans ce cas, il suffit de diminuer la valeur de R43 à une valeur de 2,2 kΩ ou jusqu'à un minimum de 1 kΩ.

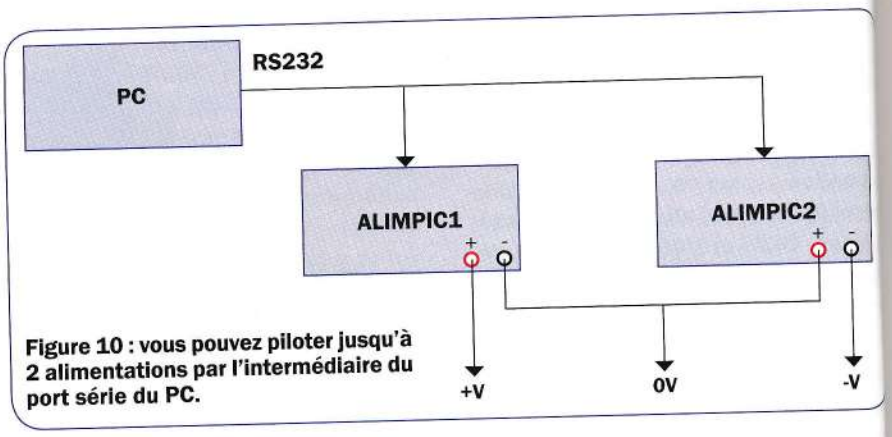


Figure 10 : vous pouvez piloter jusqu'à 2 alimentations par l'intermédiaire du port série du PC.

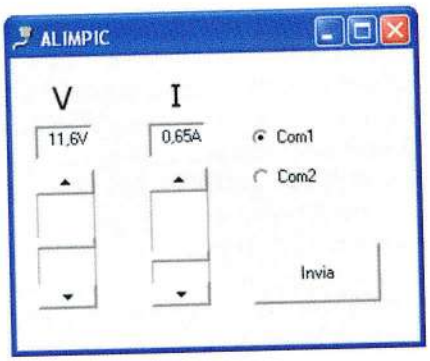


Figure 11 : le logiciel « ALIMPIC.msi » est écrit en Visual Basic, et est très simple d'utilisation.

Sur la figure
triquement a
la double for

Le boitier

Fig. A

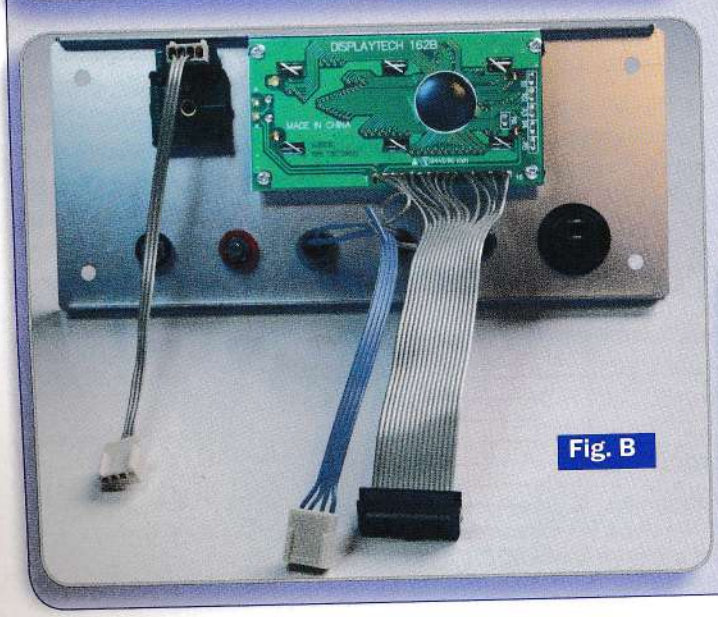


Fig. B

L'alimentation décrite dans ces pages peut être installée dans n'importe quel boitier de taille appropriée. Pour obtenir un résultat ayant un bel aspect, nous l'avons placée dans un boîtier métallique de dimensions très compactes. Sur les figures A et B vous apercevez la face avant vue de l'extérieur et de l'intérieur du boitier. Etant donné le petit nombre d'éléments (3 boutons, 2 prises, 1 interrupteur, l'encodeur et l'afficheur) à assembler, nous avons tiré le meilleur parti de l'espace disponible sur la face avant en aluminium.

Sur la figure C vous apercevez le radiateur de dimensions généreuses et la prise VDE à 3 pôles avec le compartiment contenant le fusible. Remarquez que sur la figure D, vous voyez la disposition du circuit imprimé à l'intérieur du boitier. Le transformateur de puissance, avec le primaire 230 VAC et le secondaire 2 x 12 V / 3 A (voir du côté droit le groupe de fils) est placé sous le circuit imprimé et l'étage de sortie comprenant Q3 est proche de la face arrière.

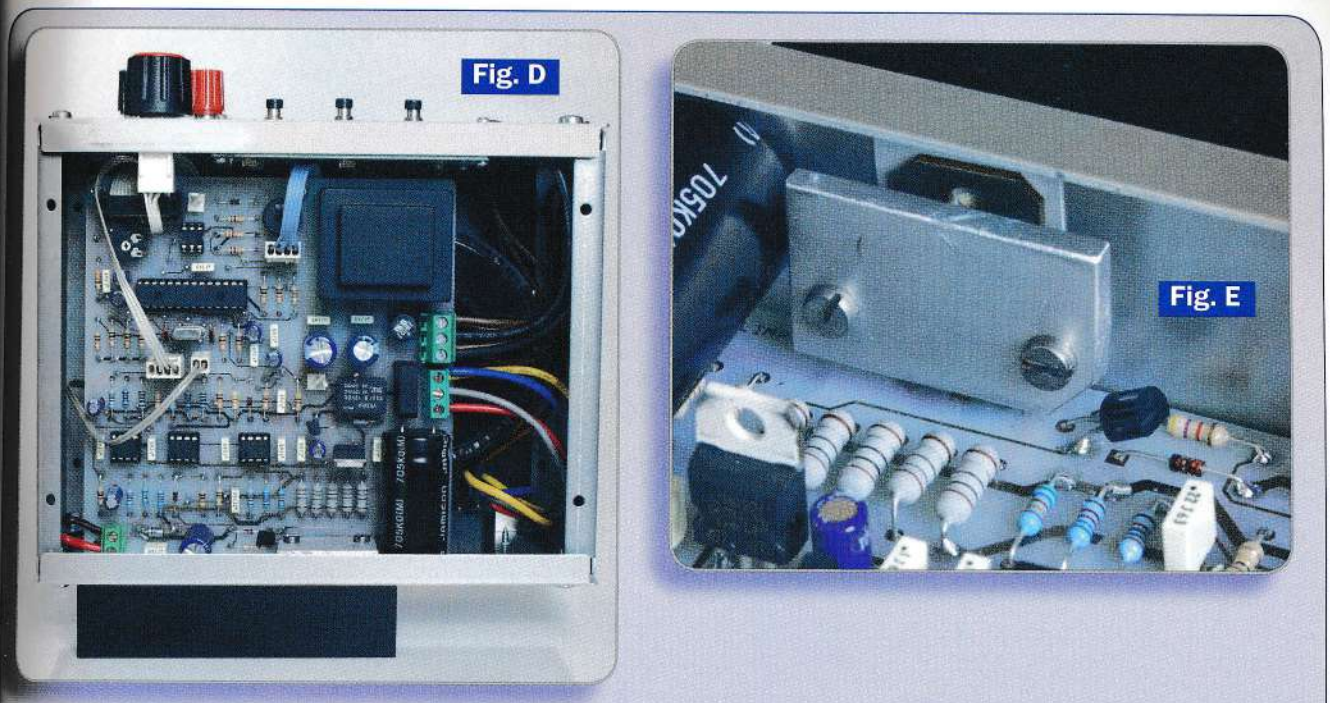
Fig. C

Le **protocol**
prévoit que les
la tension et
mises sous la
ASCII. Par co
eur est envoy
bauds, avec
sans parité. L
la valeur de
celle du **coura**

Prenons un ex

- si nous voul
eurs 11,6 V e
déterminer le
au code ASC
en supprimar
cas, 116 (la
caractère « T
courant) corre
Nous devrions
« TA » si nous
gramme d'ém

Mais pour sim
du PC en vou
ler avec un ta
nous avons
spécifique qu
sions ASCII. L
Basic, se nom



Sur la figure E enfin, vous apercevez la plaque métallique utilisée pour maintenir le transistor de puissance (isolé électriquement avec une feuille de mica ou de Teflon sur les 2 côtés avec de la graisse silicone) sur la face arrière qui réalise la double fonction de support mécanique et dissipateur de chaleur.

Le **protocole de communication** prévoit que les valeurs de consignes de la tension et du courant soient transmises sous la **forme de caractères ASCII**. Par conséquent, chaque valeur est envoyée sur **2 octets à 9600 bauds**, avec **un bit d'arrêt** (stop) et **sans parité**. Le **premier octet** contient la valeur de la **tension** et le **second** celle du **courant**.

Prenons un exemple :

- si nous voulons transmettre les valeurs 11,6 V et 0,65 A, il faut d'abord déterminer les caractères associés au code ASCII de ces deux valeurs, en supprimant la virgule. Dans notre cas, 116 (la tension) correspond au caractère « T » et 65 (0,65 A pour le courant) correspond au caractère « A ». Nous devrions alors envoyer la chaîne « TA » si nous travaillions avec un programme d'émulation de terminal.

Mais pour simplifier le contrôle à partir du PC en vous épargnant de travailler avec un tableau de codes ASCII, nous avons développé un **logiciel spécifique qui effectue les conversions ASCII**. Le logiciel, écrit en Visual Basic, se nomme « **ALIMPIC.msi** » et peut être téléchargé gratuitement sur notre site. Comme vous pouvez le voir sur la figure 11, le logiciel est très simple. Il suffit de définir le port COM, les valeurs de la tension et du courant à l'aide des 2 curseurs et d'appuyer sur le bouton « **Envoyer** ».

Si la **commande a été reçue correctement, vous entendrez un bip court généré par l'alimentation** et les 2 nouvelles valeurs seront affichées immédiatement sur l'écran LCD. En disposant d'un contrôle logiciel, vous pouvez programmer des valeurs très précises en fonction de vos besoins spécifiques en mode automatique et temporisé, comme c'est le cas, par exemple, lors des tests de circuits électroniques.

Une autre application intéressante est la **gestion simultanée de 2 alimentations**, comme indiqué sur le schéma synoptique de la figure 10. En faisant arriver le **signal série aux 2 alimentations**, vous pouvez les commander simultanément avec les mêmes valeurs et **réaliser ainsi une alimentation parfaitement symétrique**.

Comme l'interface série est opto-isolée, les alimentations peuvent être connectées en série pour obtenir une tension totale qui peut varier de 0 V à 50 V ou une tension symétrique réglable de ± 0 V à ± 25 V. Dans les deux cas, le courant est réglable entre 0 et 2 A. Vous pouvez aussi les relier en parallèle, afin d'obtenir un courant variant entre 0 et 4 A avec une tension maximale de 25 V.

Comme vous pouvez le constater, la souplesse d'utilisation de cette alimentation est remarquable. La seule contrainte est la limitation du courant fourni par le port série du PC, qui vous permet de contrôler simultanément plus de 4 alimentations. ■

Comment construire ce montage

Les typons des circuits imprimés et les programmes du microcontrôleur ainsi que du PC sont téléchargeables gratuitement à l'adresse : **www.electroniquemagazine.com** dans le sommaire détaillé de la revue numéro 128 section « Télécharger ».

PACK DE DÉMARRAGE RASPBERRY PI

Le Raspberry Pi est un ordinateur monocarte à processeur ARM pas plus gros qu'une carte de crédit. Il réalise de nombreuses fonctions comme se connecter à une télévision, une souris ou un clavier.
Il permet également l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux et des logiciels compatibles.

Réf. RASPKIT-PACK
84,00 €

CARTE D'EXTENSION A/D 16 BITS POUR RASPBERRYPI COMPATIBLE ARDUINO

NOUVEAU !

Réf. ET1041M
27,90 €

Le pack comprend :

Le Raspberry Pi modèle B, (512 Mo de RAM),
Le boîtier pour Raspberry PI,
Une carte micro SDCard HC de 4GB avec le logiciel nécessaire pour exécuter les expérimentations.
Une alimention à découpage ultra compacte, (68 x 35 x 14 mm) avec sortie USB 5VDC/1A,
Un câble HDMI d'une longueur de 1,5 mètre,
Un câble USB M(A), / Micro(B), de longueur 0,75 mètre et un câble FTP CAT5E de longueur 0,75 mètre.
Contenu de la carte SD
Système opérationnel Raspbian wheezy 2-9-2013 mis à jours le 18 mai 2013
Server SSH activé
Server web apache2 installé et configuré
Server MySQL installé et configuré
Utilisateur «root» activé avec mot de passe «root»
Server emoncms installé et configuré
Les bases de données RaspiBase et emoncms chargées et configurées
Paquet espeak installé
Inclus des langages de développement et des librairies de support

COMELEC

CD 908 - 13720 BELCODÈNE Tél. : 04 42 70 63 90 Fax : 04 42 70 63 95 www.comelec.fr

MINI WEBCAM USB COMPATIBLE AVEC LINUX

NOUVEAU !

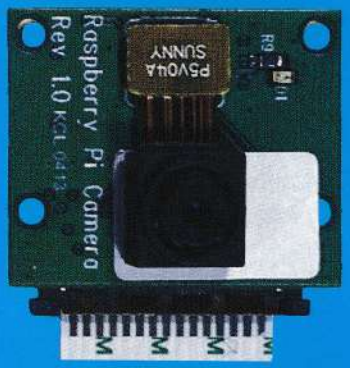
Cette mini caméra USB couleur est particulièrement adaptée pour s'utiliser avec PC Duino, Raspberry, Arduino Yún ou simplement connectée à un PC.
Elle dispose d'une résolution maximale de 640 x 480 pixels, d'un microphone intégré, de la fonction snapshot, d'un clip de fixation à ressort et d'un câble enroulable d'une longueur max de 91cm. Dimensions : 39.35 (largeur) x 29.45 (hauteur) x 14.55 (épaisseur sans l'objectif).

Réf. WEBCAMUSB 23,50 €

MODULE CAMÉRA POUR RASPBERRY PI

NOUVEAU !

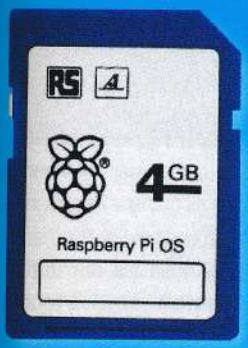
Petit module pour raspberry pi équipé d'une caméra de 5 méga pixels, et de résolution 2592 x 1944 pixels (omnivision 5647).
Ce petit module, de dimensions 25 x 20 x 10 mm pèse seulement 3 grammes. Il est capable d'effectuer une reprise vidéo hd au format 1080p à 30 fps, donnant la possibilité aux utilisateurs de cartes raspberry pi modèle a et b de réaliser des applications vidéo.
Le module vient se mettre sur le connecteur csi actuellement non utilisé sur le raspberry pi et se contrôle par bus i²c.



Réf. 7757731RS
37,90 €

CARTE SD 4GB, OS RASPBERRY PI

Réf. 7631030RS
10,90 €



Carte mémoire SD de 4GB avec système d'expoitation Raspberry Pi pré-installé.

BOITIER POUR MODULE CAMÉRA DU RASPBERRY PI

Réf. RPI-CAM.9 6,80 €

Au-delà de contenir le module camera du raspberry pi, ce coffret en plastique noir peut être fixé à n'importe quelle surface par velcro ou vis.

BOITIER TEKBERRY

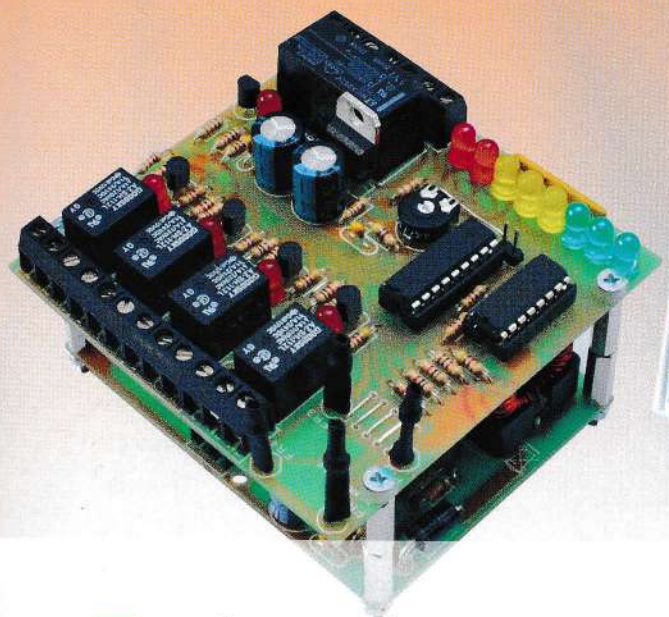
Réf. TEKBERRY-BLK 6,80 €

Boitier plastique spécialement conçu et percé pour recevoir la carte Raspberry PI.
Existe en blanc Réf: TEKBERRY

ET 755

« Tracker » ou suiveur de Soleil motorisé pour panneaux solaires

Un « tracker » solaire ou suiveur de Soleil est une installation de production d'énergie solaire utilisant le principe de l'héliostat. C'est une structure portante qui permet d'orienter des panneaux solaires en fonction de la position du soleil. Conçu pour de petites centrales électriques, ce système oriente le panneau solaire tout au long de la journée vers la lumière du soleil de manière optimale, assurant ainsi le meilleur rendement possible.



de Mirco Segatello

Parmi les sources d'**énergies renouvelables**, l'**énergie solaire est certainement la plus intéressante**, car elle est utilisable dans la pratique par des systèmes thermodynamiques et/ou des systèmes photovoltaïques. Les **panneaux photovoltaïques** (ou panneaux solaires) sont des **générateurs de tension** et **de courant** capables d'**obtenir de l'électricité à partir de la lumière**, en particulier celle du **Soleil**. De nos jours, nous voyons de plus en plus sur les toitures des maisons des panneaux solaires (fixes et donc ne captant qu'une partie de la lumière), des supports sur des terrains (toujours fixes), et même sur petits bateaux. Cependant **ces systèmes ne captent pas de manière optimale la lumière du soleil toute la journ**

Plus précisément en ce qui concerne le **positionnemen** rappelons que selon la **Loi de Lambert** * (une des règ fondamentales de l'optique ...), l'**énergie transférée par faisceau de lumière sur une surface est directement portionnelle au cosinus de l'angle formé entre la directi de la lumière et la normale à cette surface.**

Donc pour qu'un panneau solaire reçoive l'énergie maximale, ou travaille avec la meilleure efficacité possible, **il est nécessaire qu'il soit perpendiculaire à la direction des rayons solaires** et donc à la lumière du soleil.

* **Enoncé de la Loi de Lambert** :

La quantité d'énergie émise à partir d'un élément de surface dans une direction déterminée est proportionnelle au cosinus que fait cette direction avec la normale à la surface. La loi de Lambert est également appelée « Loi du cosinus ».

C'est pourquoi, pour obtenir le **rendement maximal** d'un panneau solaire ou d'un système de panneaux photovoltaïques, **il est nécessaire** que ce(s) dernier(s) **suive**(nt) la **trajectoire du Soleil tout au long de la journée**, de manière à être **perpendiculaire**(s) ou presque au Soleil.

Cette fonction est réalisée à l'aide d'appareils appelés « **trackers** » **solaires**. Ce sont des dispositifs capables de détecter la direction à partir de laquelle le Soleil est perçu avec la plus grande intensité lumineuse, et par conséquent ces dispositifs contrôlent un **actionneur** (moteur ou vérin) qui permet de faire **varier l'inclinaison** afin d'orienter les panneaux dans cette direction.

Son principe de fonctionnement est de s'orienter vers le Soleil tout au long de la journée, ce qui a pour effet d'augmenter la production d'énergie de manière substantielle. En effet, **la position du Soleil varie constamment, à la fois pendant la journée, mais aussi pendant les différentes périodes (saisons) de l'année**.

Le « **tracker** » permet ainsi de placer au mieux le panneau par rapport au positionnement du Soleil (perpendiculaire au rayonnement si possible). Suivre le soleil peut se faire sur **deux axes** : en **azimut** (d'Est en Ouest, à mesure de l'avancée de la journée) et en **hauteur** (selon la saison et, de nouveau, l'avancée de la journée). L'idéal est d'utiliser un « **tracker** » à **deux axes**, mais il en existe aussi avec un seul (typiquement avec un suivi seulement en azimut), l'angle par rapport au sol étant fixé selon l'optimum local, qui dépend de la latitude.

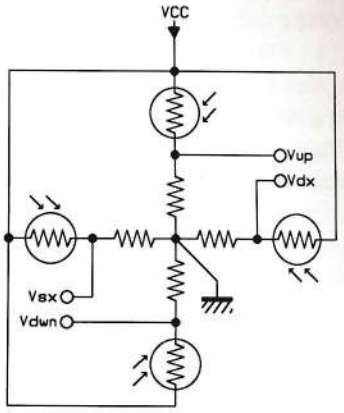


Figure 1 : pour détecter la position du soleil, nous utilisons 4 photorésistances disposées aux sommets d'un losange (points cardinaux) qui se déplacent en même temps que le panneau solaire. Les photorésistances disposées en haut et en bas détectent l'élévation verticale du soleil au-dessus de l'horizon, alors que celles disposées sur les côtés détectent le mouvement horizontal du déplacement du soleil de gauche et à droite. Chaque photorésistance fonctionne comme une résistance variable d'un diviseur de tension et produit une tension qui dépend de la quantité de lumière qu'elle reçoit. Lorsque les 4 tensions ont des valeurs égales (ou très proches), le panneau solaire se trouve perpendiculaire au soleil. Lorsqu'une des tensions varie (par exemple Vup> Vdw), cela signifie que le soleil est déplacé par rapport au panneau, le microcontrôleur qui contrôle les tensions corrige alors la position en déplaçant le panneau dans la direction de la photorésistance ayant la tension la plus élevée.

Dans cet article, nous allons décrire un « **tracker** » (ou suiveur) solaire à **deux axes**, dimensionné pour mouvoir des panneaux relativement légers. Plus précisément, le système permet d'orienter des panneaux photovoltaïques d'une **cinquantaine de watts** et d'un **poids** variant de **5** à **7 kg**. En installant plusieurs dispositifs côte à côte, vous pouvez obtenir un système de plusieurs centaines de watts avec un rendement exceptionnel.

Le fonctionnement

Pour connaître la direction privilégiée à un instant donné vers laquelle il faut orienter le panneau solaire, notre « **tracker** » utilise un **microcontrôleur qui analyse les signaux des 4 capteurs optiques disposés selon la figure 1** et assemblés de manière à détecter la lumière seulement de face.

Le microcontrôleur lit périodiquement les signaux des capteurs et détermine la direction privilégiée qui correspond au **capteur le plus « éclairé »** (recevant le plus de lumière).

Pour être sûr que le panneau soit orienté dans la bonne direction, le microcontrôleur vérifie l'état des autres capteurs qui se trouvent dans la direction opposée à celle la plus éclairée.

Le **panneau solaire est monté sur un support articulé** (motorisé) ayant un **mouvement de type « PAN-TILT »**. Le terme « **PAN** » correspond à un mouvement en rotation autour d'un point fixe : **rotation de gauche à droite** (ou l'inverse). Le terme « **TILT** » correspond à un **mouvement en rotation de haut en bas** (ou l'inverse). Le **panneau solaire est donc capable de faire des mouvements dans toutes les directions**, en combinant des mouvements horizontaux et verticaux, avec un **angle suffisamment grand pour couvrir la trajectoire du Soleil du lever au coucher**. La tension de commande est fournie par le microcontrôleur en fonction des informations provenant des capteurs de lumière.

Comme le **support fonctionne avec une tension alternative**, mais que le « **tracker** » puise son énergie dont il a besoin dans la **batterie que le panneau solaire charge**, nous avons utilisé un **convertisseur spécial continu-alternatif** géré directement par le microcontrôleur.

De quoi se compose l'ensemble ?

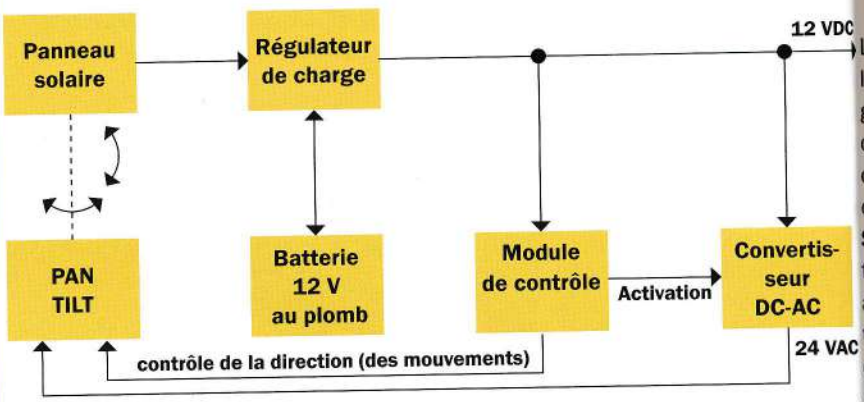
Le schéma de principe de notre « **tracker** » solaire est représenté sur la figure 2. Les éléments de base sont le **module de contrôle** contenant le microcontrôleur et les capteurs de lumière, ainsi que le **module convertisseur DC-AC** fonctionnant en **PWM** (modulation de largeur d'impulsion), contenant également son propre microcontrôleur et directement interfacé avec le module de contrôle.

Nous avons, enfin, le module « **panneau** », composé du **panneau solaire**, du **régulateur de charge** (de type **shunt**), de la **batterie** et du **support inclinable** « **PAN-TILT** ». Ce dernier est relié par 5 fils (4 pour la commande du mouvement dans une direction et un commun) vers le module de contrôle, d'où il **tire son alimentation générée par le convertisseur DC-AC**.

Le module de contrôle et le convertisseur DC-AC peuvent être montés sur le panneau solaire (fixés derrière), mais d'un point de vue pratique, il est **plus commode de fixer au panneau solaire les photorésistances**, en **laissant toute l'électronique et la batterie à l'abri des intempéries** (par exemple le grenier). Nous allons maintenant analyser en détails les différents modules.

Figure 2 : schéma de principe du « tracker » solaire

La tension provenant du panneau solaire est utilisée par le régulateur de charge dont le rôle est de charger la batterie afin d'alimenter le « tracker » solaire, le convertisseur DC-AC ainsi que la charge connectée à la sortie 12 VDC. Le « tracker » contrôle les mouvements du support « PAN-TILT » en activant le convertisseur DC-AC qui alimente le commun.



L'unité de contrôle

Elle est représentée sur la figure 3, c'est la partie la plus importante du système, car elle contrôle l'ensemble des fonctions. Elle est entièrement gérée par un microcontrôleur **MICROCHIP PIC16F88**, basée sur une architecture 8 bits, avec une mémoire Flash-EPROM organisée en « mots » (« words ») de 14 bits et fonctionnant dans notre montage avec l'horloge interne. Il dispose également d'un convertisseur A/D à 7 canaux de 10 bits et 16 entrées/sorties numériques (ports A et B).

Lors de l'initialisation, le programme définit **la ligne RB3 en entrée pour lire le cavalier J1** et les lignes **RA6**, **RA7**, **RB4**, **RB5**, **RB6** en **sorties**, destinées **à commander les 5 relais qui contrôlent physiquement le support inclinable** « **PAN-TILT** » et le **convertisseur DC-AC**. Plus précisément, l'activation d'une des sorties a lieu lorsque la ligne correspondante est portée à un niveau logique haut, le transistor NPN correspondant devient alors conducteur et alimente la bobine du relais qui est connectée entre le collecteur et + 12 VDC. Les circuits des relais sont identiques et chacun comprend une LED qui indique l'état actif.

En ce qui concerne la signification des 5 sorties et par conséquent des 5 relais, vous devez savoir que **seulement 4 relais gèrent les mouvements du support « PAN-TILT »**, tandis que le dernier est utilisé pour **commander le convertisseur DC-AC**.

Celui-ci n'est alimenté que lorsqu'il est nécessaire d'effectuer un mouvement. Il s'agit du relais **RL5** géré par la ligne **RB4** du microcontrôleur **U1**, et qui contrôle le convertisseur DC-AC.

Concernant les autres relais, **RL2** commande le déplacement horizontal vers la **gauche**, **RL4** le déplacement horizontal vers la **droite**, **RL1** et **RL3** les mouvements vers le **haut** et vers le **bas**. Chaque relais ferme en alternance le circuit d'alimentation des moteurs du support.

Poursuivons avec l'analyse des autres lignes du microcontrôleur. Comme vous pouvez le voir les **convertisseurs A/D** sont affectés aux broches 1, 2, 17, 18 (respectivement **AN2**, **AN3**, **AN0**, **AN1**) pour **lire l'état des photorésistances** (comme représenté sur la figure 1) et aux **broches 3** (**AN4**) et **12** (**AN5**) pour **détecter les tensions** respectivement, du **panneau solaire** et du circuit d'étalonnage des **seuils de détection des photorésistances**.

Il reste à examiner deux autres broches, **RB0** et **RB1**, qui constituent le **BUS I2C** du **PIC16F88** afin de **piloter** les **8 LED** à travers le **PCF8574** qui visualisent approximativement le niveau de la tension de sortie du panneau photovoltaïque. Ceci, bien sûr, en fonction de la tension lue par le convertisseur A/D aux points « + » et « - » de l'entrée « SOLAIRE » (broche 3, entrée AN4).

La visualisation est réalisée grâce au **PCF8574** qui est un circuit d'extension d'entrées/sorties fabriqué par Philips et que nous avons déjà utilisé dans de nombreuses autres applications. Nous ne pouvions pas faire autrement parce que le microcontrôleur n'a pas assez de lignes d'entrées/sorties pour gérer directement la barre de LED.

Le **PCF8574** est **tionnel**, en ce sens chacune des 8 lign driver logique et peut alors envoyer les **conditions** log ligne **SDA** du BUS peut **transmettre SDA**, les états des 9, 10, 11, 12.

Le mode de fonctio la commande reçu gère les instructio c'est le microcontrô que la norme du BU dispositifs connect **SCL** reçoivent dire tions de contrôle, **adresse qui permet** tant que « cible » de dispositif **I2C** poss permettent de défi **8 combinaisons** po

Dans le cas du PCF correspondantes sont vement **A0**, **A1**, **A2**, E schéma électrique, voi le circuit **U2** (PCF8574 gique « **000** » ce c **adresse** « **000** » (ég sorties de **U2** pilotent **LED**, le courant étan sur une sortie est niveau) peut atteind

Une fois les ent **isées**, le microc

Figure 3 : ici montée. Elle dé et active à la fo et gère l'inclina

Le **PCF8574** est un **circuit bidirectionnel**, en ce sens qu'il possède, pour chacune des 8 lignes P0 à P7, un étage driver logique et un buffer de lecture. Il peut alors **envoyer aux lignes P0 à P7** les **conditions logiques** arrivant sur la ligne **SDA** du **BUS I2C**, mais aussi il peut **transmettre** sur la même ligne **SDA**, les états des broches 4, 5, 6, 7, 9, 10, 11, 12.

Le mode de fonctionnement dépend de la commande reçue par l'élément qui gère les instructions, dans notre cas c'est le microcontrôleur **PIC16F88**. Puisque la norme du **BUS I2C** prévoit que les dispositifs connectés aux lignes **SDA** et **SCL** reçoivent directement les instructions de contrôle, ils disposent d'une **adresse qui permet de les identifier** en tant que « cible périphérique ». Chaque dispositif **I2C possède 3 broches** qui permettent de définir une adresse parmi **8 combinaisons possibles**.

Dans le cas du **PCF8574**, les broches correspondantes sont 1, 2, 3 (respectivement A0, A1, A2). En vous référant au schéma électrique, vous remarquez que le circuit **U2** (PCF8574) a ses **3 broches A0**, **A1** et **A2** à la **masse** soit au niveau logique « **000** » ce qui correspond à l'**adresse** « **000** » (adresse « 0 »). Les **sorties** de **U2 pilotent directement les LED**, le courant absorbé par chacune sur une sortie en mode « **sink** » (bas niveau) peut atteindre 25 mA.

Une fois les **entrées/sorties initialisées**, le microcontrôleur commence

Figure 3 : Ici l'unité de commande montée. Elle détecte la position du Soleil et active à la fois le convertisseur DC-AC et gère l'inclinaison du panneau solaire.

à **exécuter le programme principal**, qui prévoit un **test cyclique de l'état des photorésistances** et des **lignes RB3** (broche 9) et **AN5** (broche 12). La routine qui concerne la mesure des tensions provenant des convertisseurs A/D de FR1, FR2, FR3 et FR4 est particulièrement intéressante.

Comme nous l'avons déjà mentionné, ces composants **sont disposés en quadrilatère et assemblés de façon à détecter la lumière seulement de manière frontale**. Lorsqu'elles sont convenablement orientées vers le haut et inclinées à environ 45°, chacune des photorésistances détecte l'éclairement dans une direction déterminée.

D'un point de vu électrique **FR1**, **FR2**, **FR3** et **FR4** sont connectées avec des résistances en série formant chacune **un diviseur de tension**, lorsque l'**éclairement augmente**, la **résistance de la photorésistance diminue** donc la **tension en sortie du diviseur** appliquée aux entrées correspondantes (AN3, AN2, AN0, AN1) **augmente**. Vice versa, lorsque la **luminosité diminue**, la résistance **augmente** et la **tension du diviseur diminue**.

Pour **connaitre la trajectoire du Soleil** et **orienter correctement** le panneau solaire, le microcontrôleur **lit cycliquement les tensions des diviseurs formés par les 4 photorésistances et extrait la valeur la plus élevée**, ce qui correspond à la photorésistance la plus éclairée.

Le PIC estime alors que cette direction est la plus éclairée et va orienter le panneau dans cette direction grâce à la « **routine mouvement** ». Le microcontrôleur fournit alors un niveau logique **haut** sur la **sortie RB4** ce qui rend le transistor **T1 conducteur** et le relais **RL5 est activé**. Celui-ci **alimente le convertisseur DC-AC**, le PIC active ensuite le relais correspondant à la direction souhaitée afin de positionner correctement le panneau solaire. **RL5 est activé pendant toute la durée du mouvement du support**.

Quand s'arrête le mouvement ? Il est clair que si le mouvement prend en compte uniquement l'éclairage reçu, à un moment donné le **support** « **PAN-TILT** » **continuera à osciller** pour trouver une position d'équilibre, car le moindre changement d'éclairage dû par exemple à un passage de nuages provoquerait une modification de l'éclairement de la photorésistance la plus éclairée (elle ne le serait plus pour une courte durée).

Mais pas seulement, les photorésistances étant sur le bord du panneau, elles se déplacent toutes dans la direction du Soleil, ayant pour résultat, que pendant le mouvement, la différence d'éclairement entre les photorésistances diminue car elles sont de plus en plus illuminées. Ceci ferait revenir le support dans la direction d'origine, sans s'arrêter.

Le problème est donc d'**éviter des déplacements intempestifs**, qui sont non seulement inutiles et n'apportent aucune augmentation sensible à l'efficacité du système photovoltaïque, mais aussi consomment une part importante de l'énergie stockée dans la batterie.

En fait le **support doit se déplacer pendant de courtes périodes** et **non de façon continue**.

Nous avons **résolu le problème en imposant une différence minimale entre les niveaux d'éclairement** (et donc des tensions) de **2 photorésistances placées sur une diagonale** (FR1-FR3, FR2-FR4). Si la **différence n'est pas atteinte, le support n'est pas activé**.

Mais pas seulement, en effet pour que le système soit stable, le programme impose une certaine distance entre la valeur d'éclairement des photorésistances opposées qui provoque le mouvement du support et qui déterminent l'arrêt.

Dans la pratique, le **panneau est mis en mouvement lorsque la différence d'éclairement dépasse un certain seuil** (fixé par le potentiomètre R7) et s'arrête seulement **si la différence mesurée entre 2 photorésistances en diagonale devient inférieure au seuil minimal qui a engendré le mouvement**. Ainsi, vous obtenez un changement discontinu mais stable.

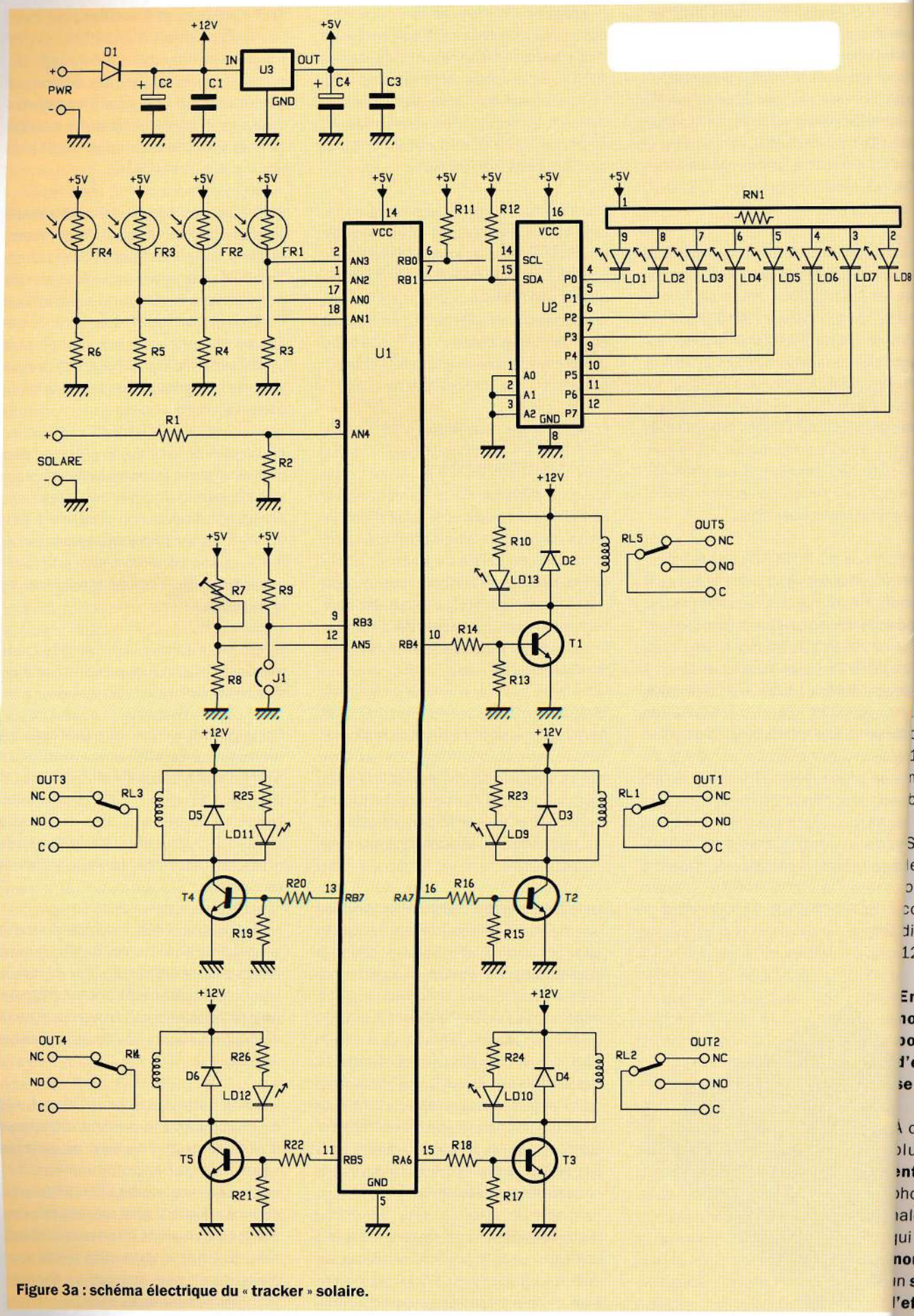


Figure 3a : schéma électrique du « tracker » solaire.

Prenons un exemple concret, supposons que le système se trouve au repos et dans la condition d'équilibre, avec toutes les photorésistances exposées à un éclairement de 5000 lux.

A un certain moment le Soleil se déplace, FR1 et FR3 (qui se trouvent sur le même axe diagonal du mouvement du Soleil) voient leur luminosité diminuer de manière identique, tandis que l'éclairement de FR2 qui est face à la direction du soleil augmente de manière significative par rapport à FR4.

Or celle-ci étant du côté opposé, est beaucoup plus affectée que les autres (FR1 et FR3) par la diminution de l'éclairement. Disons simplement que FR1 et FR3 reçoivent 4500 lux, FR2 5500 lux et FR4 seulement 4000 lux.

Le microcontrôleur détecte que la cellule la plus éclairée est FR2 et mesure la différence par rapport à celle qui fait face soit FR4. Il actionne le support pour faire pivoter le panneau dans la direction vers laquelle « on regarde FR2 » si la différence est supérieure au seuil défini par R7.

Ainsi, si le réglage est tel que la distance minimale entre les valeurs d'éclairements semble inférieure à 1500 lux, le microcontrôleur commande le support, sinon rien ne bouge.

Si le seuil est inférieur à 1500 lux, le support « PAN-TILT » commence à orienter le panneau solaire, le microcontrôleur l'arrête seulement si la différence est inférieure, par exemple 1200 lux.

En procédant de cette manière, nous sommes surs d'arrêter le support dans un délai raisonnable et d'éviter que le panneau continue à se déplacer autour d'une position.

À cet égard, il convient de noter que plus vous **augmentez la distance entre les valeurs d'éclairement** des photorésistances (opposées en diagonale) qui détermine le mouvement et qui provoque l'arrêt, **plus stable est le mouvement** du support. En revanche, un **seuil trop élevé entraîne une perte d'efficacité en raison du fait que le panneau photovoltaïque est orienté avec un plus grand retard**, ce qui ne se produit pas si la distance est plus courte.

Notez que le programme du **PIC16F88** prévoit que la **fréquence de lecture des tensions des photorésistances soit liée à la grandeur de la distance entre les valeurs de l'éclairement** des photorésistances opposées (seuils). Sinon il est inutile d'imposer des limites inférieures dans la variation de l'éclairement. Elle permet au système de suivre plus rapidement les changements.

Après avoir apporté quelques précisions, continuons par l'étude du mouvement du support. Comme nous l'avons déjà mentionné, lorsque le microcontrôleur détecte le dépassement de la distance prévue entre les deux seuils, il active le relais RL5 ainsi que le relais correspondant à la direction dans laquelle le panneau solaire doit être orienté. Le mouvement est interrompu lorsque la différence d'éclairement tombe en dessous de la valeur minimale.

Nous devons insister sur une chose, bien que **les photorésistances soient**

Liste des composants du « tracker» solaire

R1..... 39 kΩ
R2..... 10 kΩ
R3..... 1 kΩ
R4..... 1 kΩ
R5..... 1 kΩ
R6..... 1 kΩ
R7..... trimmer mono-tour 10 MΩ
R8..... 470 Ω
R9..... 4,7 kΩ
R10.... 1 kΩ
R11.... 4,7 kΩ
R12.... 4,7 kΩ
R13.... 10 kΩ
R14.... 4,7 kΩ
R15.... 10 kΩ
R16.... 4,7 kΩ
R17.... 10 kΩ
R18.... 4,7 kΩ
R19.... 10 kΩ
R20.... 4,7 kΩ
R21.... 10 kΩ
R22.... 4,7 kΩ
R23.... 1 kΩ
R24.... 1 kΩ
R25.... 1 kΩ
R26.... 1 kΩ
RN1 ... réseau de résistances 8 x 470 Ω + C

C1..... 100 nF multicouche
C2..... 470 µF / 25 V électrolytique
C3..... 100 nF multicouche
C4..... 470 µF / 16 V électrolytique

U1..... PIC16F88 (MF755)
U2..... PCF8574
U3..... 7805
D1..... 1N4007
D2..... 1N4007
D3..... 1N4007
D4..... 1N4007
D5..... 1N4007
D6..... 1N4007
LD1.... LED 5 mm verte
LD2.... LED 5 mm verte
LD3.... LED 5 mm verte
LD4.... LED 5 mm orange
LD5.... LED 5 mm orange
LD6.... LED 5 mm orange
LD7.... LED 5 mm rouge
LD8.... LED 5 mm rouge
LD9.... LED 5 mm rouge
LD10.. LED 5 mm rouge
LD11.. LED 5 mm rouge
LD12.. LED 5 mm rouge
LD13.. LED 5 mm rouge

T1 ..... BC547
T2 ..... BC547
T3 ..... BC547
T4 ..... BC547
T5 ..... BC547
FR1.... photorésistance 2 -20 kΩ
FR2.... photorésistance 2 -20 kΩ
FR3.... photorésistance 2 -20 kΩ
FR4.... photorésistance 2 -20 kΩ
RL1.... relais miniature 1 RT AZSH112L
RL2.... relais miniature 1 RT AZSH112L
RL3.... relais miniature 1 RT AZSH112L
RL4.... relais miniature 1 RT AZSH112L
RL5.... relais 1 RT FTR-H1CA012V 10A/250 VAC

Divers

Support ci 2 x 8 broches
Support ci 2 x 9 broches
Bornier 2 pôles (x 2)
Bornier 3 pôles (x 5)
Barrette sécable
Cavalier (jumper)

* toutes les résistances sont des ¼ w

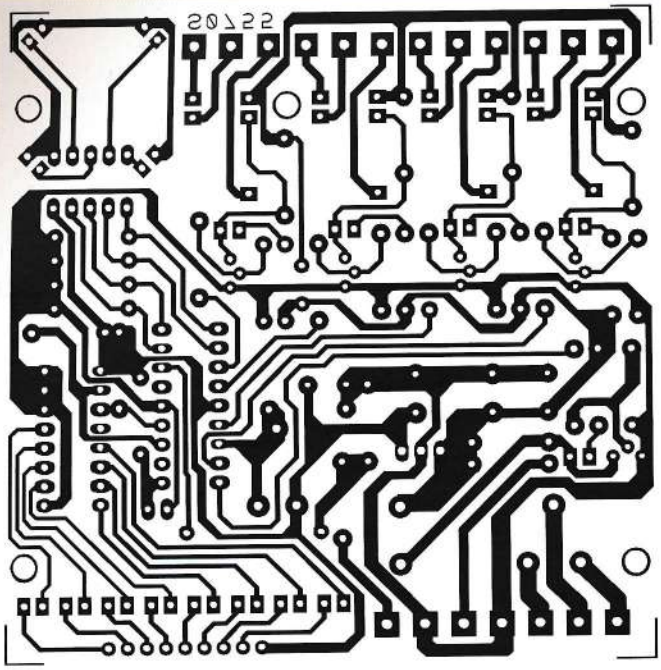


Figure 3b : dessin du circuit imprimé simple face à l'échelle 1 : 1 du « tracker » solaire.

Plan de montage du « tracker » solaire.

Le « tracker » solaire est relativement facile à réaliser. Tous les composants utilisés sont traditionnels, le circuit imprimé est un circuit simple face. L'angle en haut à gauche du circuit imprimé (voir la photo ci-dessous) est destiné à être découpé et relié à la base par l'intermédiaire d'un câble à 5 conducteurs, de manière à fixer les photorésistances sur le panneau solaire.

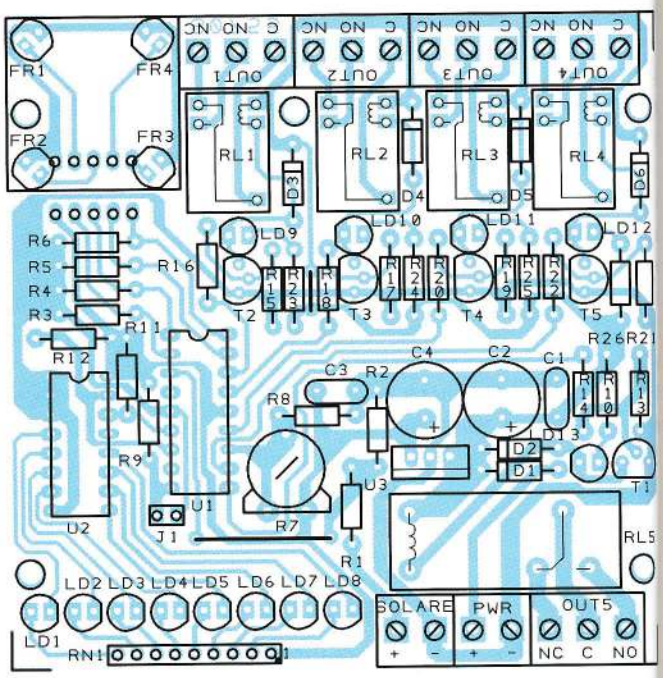


Figure 3c : schéma de câblage du « tracker » solaire.

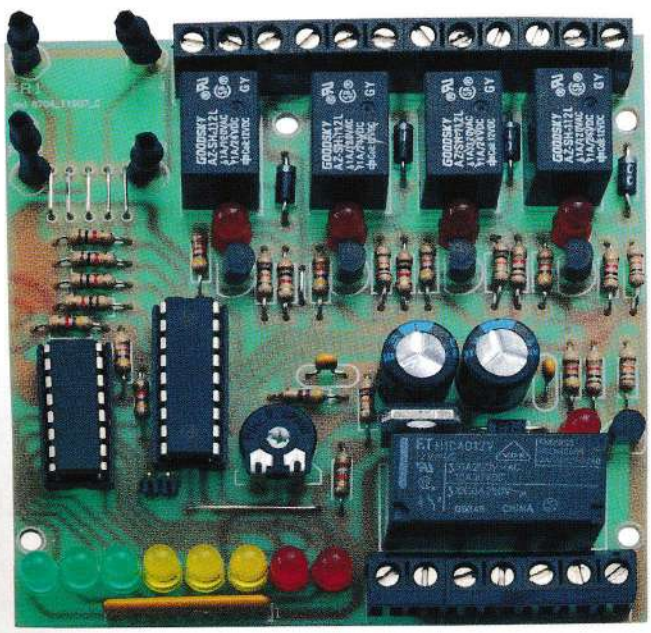


Figure 3d : photo de l'un de nos prototypes du « tracker » solaire.

du même modèle, comme tous les composants électroniques **leurs caractéristiques diffèrent en fonction de leur tolérance et de la dérive de la température.**

Cela signifie que, pour un **éclairement donné, chacune présente une valeur** de sa résistance différente de celle de l'autre et donc cela va **engendrer sur les entrées des convertisseurs A/N des niveaux de tensions différents.**

Cela n'est pas favorable à la précision du système, c'est pourquoi nous avons pensé à une manière de **compenser cette erreur à l'aide d'une procédure d'étalonnage** qui permet au microcontrôleur de définir les valeurs des photorésistances à un même niveau d'éclairement.

La **procédure est activée lorsque nous fermons J1**, le programme principal est alors interrompu et part dans une routine appropriée dans laquelle les **convertisseurs A/N lisent séquentiellement les valeurs des tensions des 4 photorésistances**, et mémorise les données.

Bien évidement **FR1, FR2, FR3** et **FR4 doivent être éclairées de la même manière.**

Avant de conclure la description de l'unité de contrôle, faisons une remarque sur le voltmètre constitué par la barre de LED. Le panneau solaire doit être connecté à la fois à l'entrée du régulateur de charge et à l'entrée de la carte de contrôle.

Comme le régulateur est de type « shunt », lorsque la batterie est chargée, il court-circuite le panneau photovoltaïque, ce qui fait apparaître une tension nulle. Il peut donc arriver que le voltmètre indique « 0 » même lorsque la batterie est complètement chargée, vous n'avez pas à vous en soucier.

Le convertisseur DC-AC

Le **convertisseur DC-AC est capable de fournir à partir d'un panneau solaire** et/ou d'une **batterie** de **12 VDC une tension de 24 VAC** (**alternative**) **nécessaire au support inclinable « PAN-TILT ».**

Le convertisseur, dont le schéma électrique est représenté sur la figure 4a et une photo d'un prototype en figure 4, est constitué d'un élévateur de tension stabilisée qui fournit 24 V à partir de la batterie de 12 V.

Il dispose d'un **étage de puissance** constitué d'un **driver en pont** et d'un **étage de commande** réalisé au moyen du microcontrôleur **U2** (un autre **PIC16F88**) programmé de manière appropriée.

Après l'initialisation des entrées/sorties, le **PIC génère une tension rectangulaire** avec laquelle il **pilote le MOSFET T5.** Celui-ci conjointement avec la **bobine L1** et la **diode D8** forment un **convertisseur continu-continu** (DC-DC) qui charge l'inductance à partir de laquelle on obtient **24 VDC.**

Le fonctionnement du convertisseur DC-DC est le suivant :

- lorsque le MOSFET reçoit du microcontrôleur une impulsion d'un **niveau logique haut**, T5 passe à l'**état « ON »** (il devient **conducteur**) et met pratiquement à la masse l'inductance L1.

- lorsque le MOSFET reçoit du microcontrôleur une impulsion d'un **niveau logique bas**, T5 passe à l'**état « OFF»** (il est **bloqué**) et **L1 se décharge** en quelque sorte. Elle **génère une tension de polarité inverse et d'amplitude plus grande que celle avec laquelle le MOSFET l'a alimentée.** La tension inverse s'écoule à travers D8 et charge

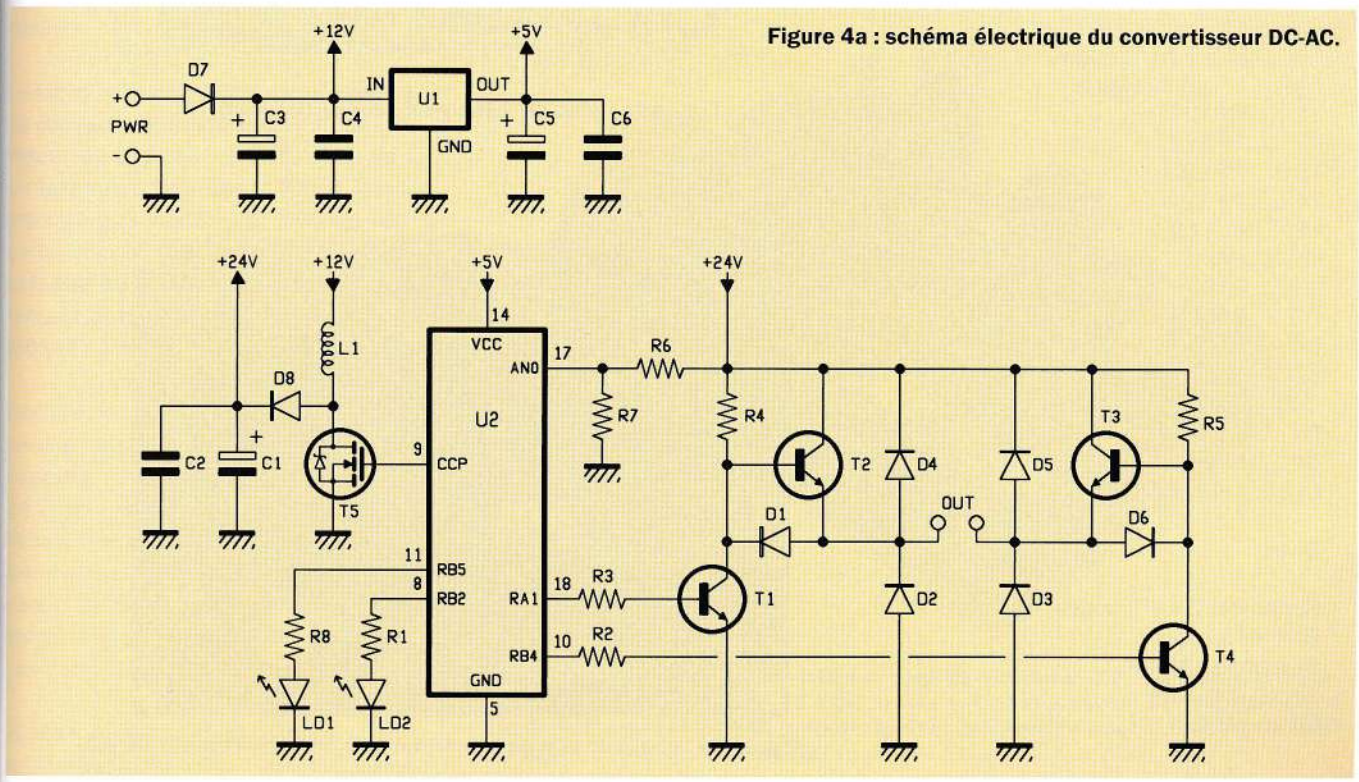


Figure 4a : schéma électrique du convertisseur DC-AC.

Plan de câblage du convertisseur

Comme pour le « tracker », le convertisseur est réalisé avec des composants traditionnels et donc facile à reproduire. Le seul composant auquel vous devez prêter attention est la bobine L1 qui, en raison de sa taille, doit être fixée sur le circuit imprimé avec de la colle.

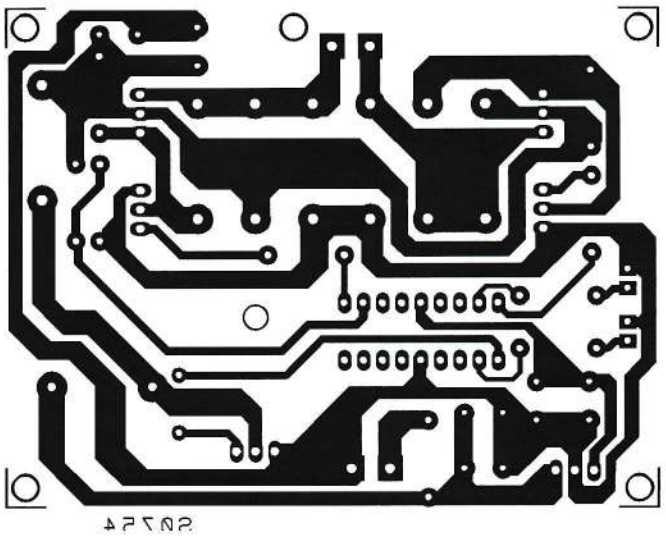


Figure 4b : dessin du circuit imprimé simple face à l'échelle 1 : 1 du convertisseur DC-AC.

Liste des composants du convertisseur DC-AC

R1..... 470 Ω
R2..... 1 kΩ
R3..... 1 kΩ
R4..... 1 kΩ
R5..... 1 kΩ
R6..... 330 Ω
R7..... 47 kΩ

C1..... 470 µF / 35 V électrolytique
C2..... 100 nF multicouche
C3..... 470 µF / 35 V électrolytique
C4..... 100 nF multicouche
C5..... 470 µF / 16 V électrolytique
C6..... 100 nF multicouche

U1..... 7805
U2..... PIC16F88 (MF754)
LD1.... LED 5 mm rouge
LD2.... LED 5 mm verte
T1 ..... TIP122
T2 ..... TIP122
T3 ..... TIP122
T4 ..... TIP122
T5 ..... BUZ11
D1..... 6A60
D2..... 6A60
D3..... 6A60
D4..... 6A60
D5..... 6A60
D6..... 6A60
D7..... 1N4007
D8..... SB540
L1 ..... bobine 68 µH

Divers

Support ci 2 x 9 broches
Bornier 2 pôles
Dissipateur ML26
Vis 10 mm 3MA
Ecrou 3MA

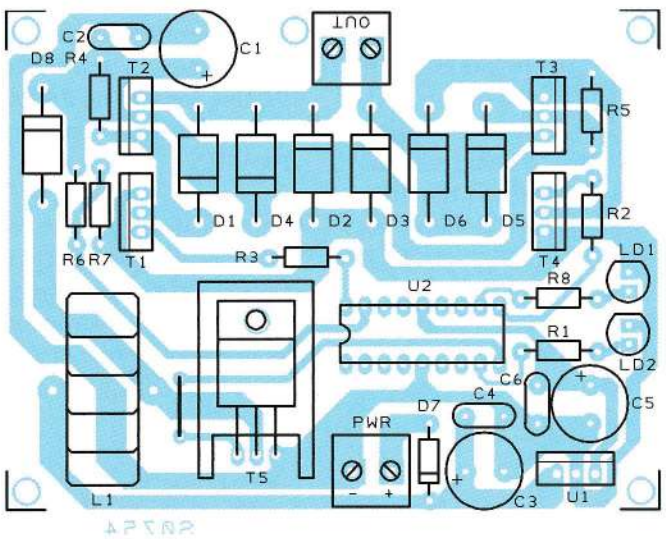


Figure 4c : plan de câblage du convertisseur DC-AC.

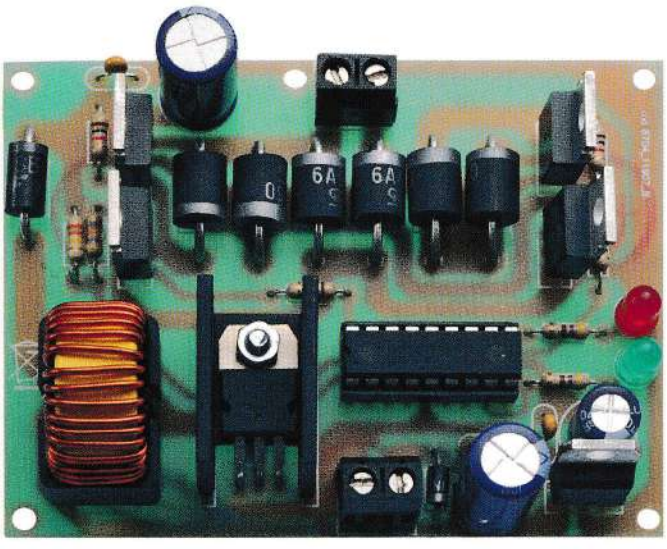


Figure 4d : photo de l'un de nos prototypes du convertisseur DC-AC.

les **condensateurs C1** et **C2**, cette **énergie stockée sera restituée lors de la période de conduction du T5**.

En résumé un « **1** » logique sur la grille du MOSFET fait **conduire le transistor et charge l'inductance L1**. Pendant la période de pause qui correspond à un « **0** » logique sur la grille du MOSFET, le **transistor est bloqué et L1 charge les condensateurs C1** et **C2** qui **ne peuvent pas se décharger** lorsque le **MOSFET est conducteur** à cause de la présence de la **diode D8**.

Aux extrémités des condensateurs se crée ainsi une tension continue que le **microcontrôleur stabilise à 24 VDC** à travers un sous-programme qui gère la **largeur des impulsions** qui commandent la grille du transistor MOSFET en fonction de la tension mesurée sur la broche 17 (AN0) du convertisseur A/D. Plus précisément, à l'initialisation le microcontrôleur fournit sur la broche 9 une composante rectangulaire d'un certain rapport cyclique, puis détecte la tension présente aux bornes des condensateurs C1 et C2.

Si la valeur est **supérieure à 24 V**, il **réduit le rapport cyclique**, alors que s'il détecte une tension **inférieure à 24 V** en raison de la charge due l'effort du support, il **augmente la largeur des impulsions** pour obtenir la tension désirée.

Le logiciel du microcontrôleur **réalise un convertisseur continu-continu élévateur de tension** « **DC-DC step-up** » en **largeur d'impulsion PWM** aussi stabilisé que possible et protège contre les surcharges. La **protection est obtenue en imposant une limite à la largeur des impulsions envoyées** au **MOSFET T5**.

Dans le cas d'un courant excessif ou en cas de court-circuit au niveau du support, la sortie n'atteindrait pas 24 V et le microcontrôleur devrait en théorie augmenter la largeur des impulsions au point que le courant circulant dans le MOSFET deviendrait excessif et entraînerait une surchauffe ainsi que des dommages.

Jusqu'à présent, nous avons vu comment obtenir 24 VDC, c'est-à-dire une tension continue, mais les **moteurs du support fonctionnent avec une tension alternative**. Mais alors, d'où vient-elle ?

La réponse est simple, nous l'**obtenons** à l'aide de **4 transistors bipolaires montés en « pont » et alimentés par la tension continue obtenue à partir du convertisseur DC-DC**. En sortie nous obtenons une tension alternative dont la **forme des alternances est carrée** et de **basse fréquence en opposition de phase**, elles sont générées par le même **PIC16F88**.

Lorsque la broche **RA1** est à **niveau haut** soit un **état logique « 1 »**, **RB4** est, en conséquence est à un **état logique « 0 »**. **T1** se trouve alors en **saturation** alors que **T2 est bloqué**. La **cathode de D1 se trouve alors proche de la masse** et donc aussi l'**émetteur de T2**. Le **côté gauche du pont** de la sortie (OUT) se trouve à un **niveau bas**. Simultanément **T4 étant bloqué**, **R5 polarise la base de T3** et le fait entrer en **saturation**. L'émetteur de **T3** porte alors à un **niveau haut le côté droit de la sortie du pont**.

Pendant la **demi-période suivante**, les **états logiques sont inversés**. La broche RA1 est mise à un niveau bas de sorte que T1 se trouve bloqué. T2 est saturé, car R4 peut maintenant polariser sa base, et provoque un niveau haut sur le côté gauche de la sortie du pont. La broche RB4 est à un niveau haut de 5 V et sature T4 de sorte que son collecteur amène à environ 0 V la base de T3 (qui devient non conducteur). La cathode de la diode D6 se trouve aussi à 0 V, et donc la sortie droite du pont se trouve à un niveau bas.

Le **cycle est répété tant que le microcontrôleur est alimenté** par le régulateur U1, un classique 7805. Ce cycle se produit lorsque l'unité de contrôle alimente le convertisseur en 12 VDC par l'intermédiaire du relais RL5, période pendant laquelle le support doit effectuer un mouvement. Les **LED** pilotées par les broches **RB2** et **RB5** indiquent la **présence de la tension de travail**.

La composante alternative présente au niveau de la **sortie « OUT »** du pont alimente directement d'un côté le **« commun »** du **support « PAN-TILT »** et de l'autre côté les **« communs »** (points « C ») des **contacts** des **4 relais RL1** à **RL4** de l'unité de contrôle.

Le support inclinable « PAN-TILT »

Le support inclinable « PAN-TILT » que nous avons choisi pour notre « tracker » solaire est le modèle **VISP2**. Ce modèle visible sur la figure 5 est du

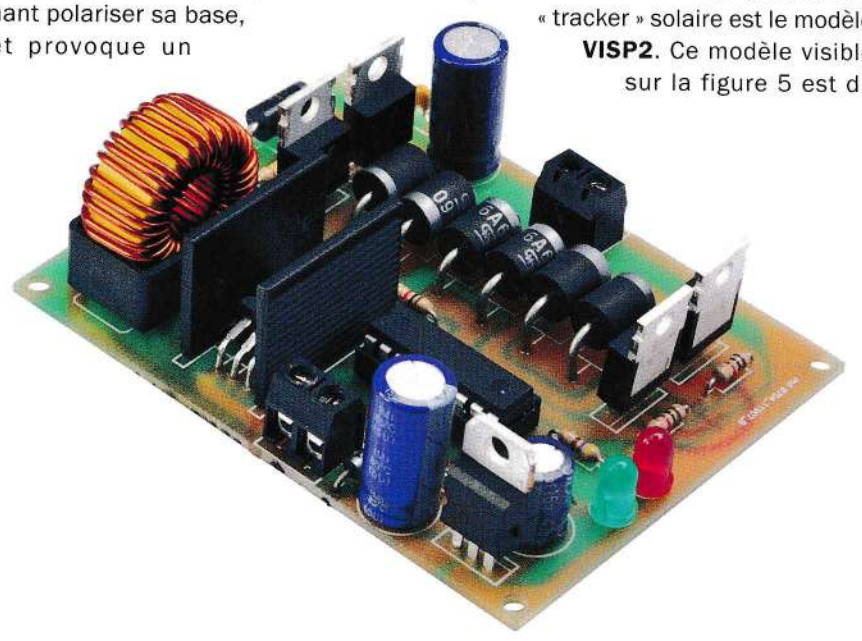


Figure 4 : prototype du convertisseur continu-alternatif (DC-AC) monté. Il transforme la tension continue 12 VDC provenant du régulateur de charge en une tension alternative 24 VAC (ondes carrées) nécessaire pour le fonctionnement du support inclinable « PAN-TILT ».

1 = COMMUN (rouge)
2 = 24VAC/AUTO (noir)
3 = INCLINAISON (TILT) BAS (bleu)
4 = INCLINAISON (TILT) HAUT (jaune)
5 = BALAYAGE (PAN) DROITE (vert)
6 = BALAYAGE (PAN) GAUCHE (blanc)

Caractéristique techniques du VISP2

Angle de balayage Horizontal : 355°/Vertical : ±100°
Arrêt : Réglage externe
Vitesse : Balayage hor/vert: 5° / seconde
Charge estimée : 7 kg
Tension d'entrée : 24VAC
Consommation : 7W
Câblage requis : 6 fils, non blindés
Température de fonctionnement : -5°C à + 60°C
Structure : plastique
Dimensions : 179 x 115 x 180mm
Poids : 1.6kg

Figure 5 : voici le support inclinable « PAN-TILT ». Il a été choisi pour sa puissance qui est adaptée à la taille du panneau solaire.

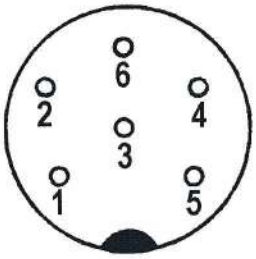


Figure 5a : brochage du connecteur du support.

type extérieur fonctionnant en **24 VAC**. Il est commandé par 5 fils qui représentent l'alimentation commune et la commande des moteurs haut, bas, gauche et droite.

Il peut maintenir et déplacer un objet pesant **plusieurs kilos** et dont la superficie n'excède pas **0,5 m²**. Il est à noter qu'une rafale de vent peut empêcher un déplacement correct du panneau solaire.

Il **doit être connecté comme indiqué dans le schéma de câblage général** avec le « **commun** » à une extrémité de la **sortie du convertisseur DC-AC** et les **4 autres fils de manière ordonnée**, aux contacts « **normalement ouverts** » (NO) des relais **RL1** à **RL4** de l'unité de contrôle. L'**autre extrémité de la sortie du convertisseur DC-AC** (bornier « OUT ») se connecte aux **points** « **communs** » (« C ») de chaque relais, de manière à fermer le circuit d'alimentation.

Le panneau solaire et le régulateur

Pour notre application, nous avon utilisé un panneau solaire d'une puis sance de **24 W**, référence **MCRY24** de dimensions 60 cm x 30 cm (vo la figure 7), capable de fournir une ten sion en plein Soleil de l'ordre de **17,4** avec un courant maximal de **1,38 A**. est directement connecté au bornie « **SOLAIRE** » de l'unité de contrôle, d

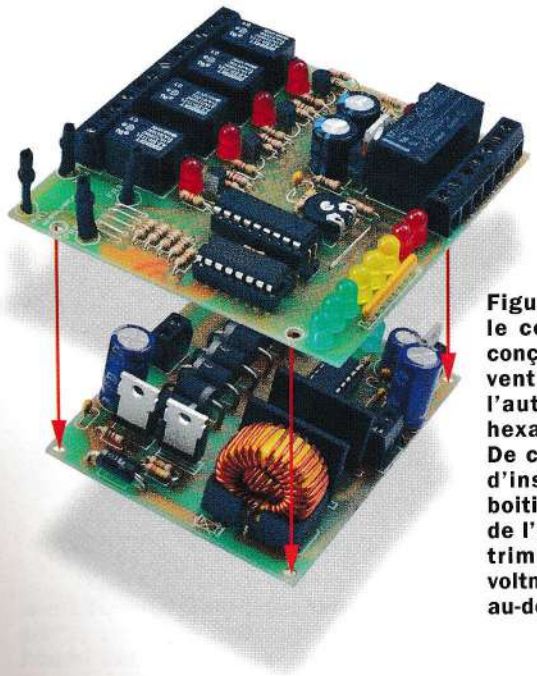


Figure 6 : l'unité de contrôle et le convertisseur DC-AC ont été conçus de telle manière qu'ils peuvent être fixés l'un au-dessus de l'autre à l'aide de 4 entretoises hexagonales de 30 mm de long. De cette manière, il est possible d'insérer les 2 circuits dans un boitier étanche à l'eau. Le circuit de l'unité de contrôle possède un trimmer de réglage ainsi qu'un voltmètre, il devra donc être placé au-dessus du convertisseur.

Figure 7 : notre prototype a été réalis avec un panneau fournissant 24 W a maximum (17,4 V/1,38 A max.)

Le régulateur SOL4UCN2

Ce régulateur est utilisé dans notre système pour maintenir la batterie chargée, c'est un régulateur de type « shunt » commercialisé par la société COMELEC et conçu pour être couplé avec des panneaux solaires fournissant jusqu'à 24 V et des batteries de 12 V d'une capacité allant de 4 à 40 Ah. Il dispose d'un boitier spécial muni de bornes à vis d'entrées (pour le panneau) et de sorties (pour la batterie et/ou la charge). Il dispose également d'un commutateur de dérivation (« bypass ») de la charge et de trois sorties de tensions stabilisées.

Les caractéristiques techniques sont les suivantes :

- tension de sortie de la charge : 12 V max / 4 A
- tensions de sorties auxiliaires CC : 3V, 6V, 12V (± 10%)
- courant de sortie CC : 4A max.
- tension de charge de la batterie : 12 V à 14 V ; optimale 13.5V ± 10%
- tension d'entrée maximale : 24 VDC
- protection contre la décharge: <= 11.0 V ±0.3V
- protection contre la surcharge : >= 15 V
- panneau solaire :
  courant de travail : <= 5A
  tension ouverte : 21 à 24V
- type d'accu recommandé: 12V/10 à 40Ah batterie scellée plomb-acide
- température de travail : -10 °C ~ +42 °C, humidité : <= 80%
- dimensions : 120 x 80 x 22mm

Le support du panneau

La manière dont le panneau solaire est fixé au support dépend à la fois de la forme et de la disposition des trous des deux ensembles. N'étant pas en mesure de savoir à l'avance quel type de panneau et de support vous allez utiliser, nous ne pouvons pas fournir d'instructions détaillées sur la manière de fixer mécaniquement les différentes pièces. Nous pouvons, cependant vous dire comment nous avons fait pour l'étrier de fixation. Nous avons acheté dans un magasin de bricolage une équerre en aluminium de 40 mm x 20 mm que nous avons coupé en 3 morceaux de longueur appropriée à assembler (avec vis et écrou) pour former une structure en « T » visible sur la figure B. Un 4eme morceau a été utilisé pour fixer le boitier contenant les photorésistances à 45 ° par rapport au panneau, afin d'effectuer les différents réglages. Une fois les réglages terminés, le boitier doit être fixé convenablement afin qu'il ne puisse pas bouger. Notez que le reste de l'électronique ainsi que le régulateur et la batterie doivent être placés à l'abri, dans le grenier par exemple.

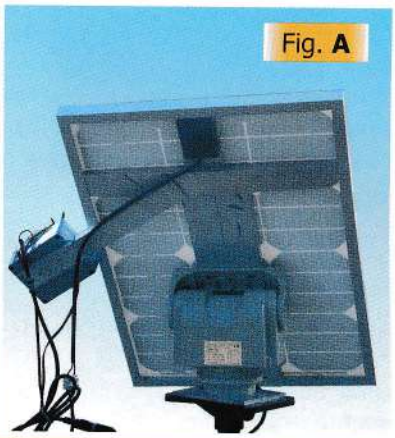


Fig. **A**

En principe, tous les panneaux solaires peuvent être fixés au support à l'aide d'une structure en « H » de taille appropriée, telle que celle représentée schématiquement sur la figure A. En choisissant une équerre convenable, vous pouvez obtenir un support de même épaisseur que le panneau solaire (figure C).

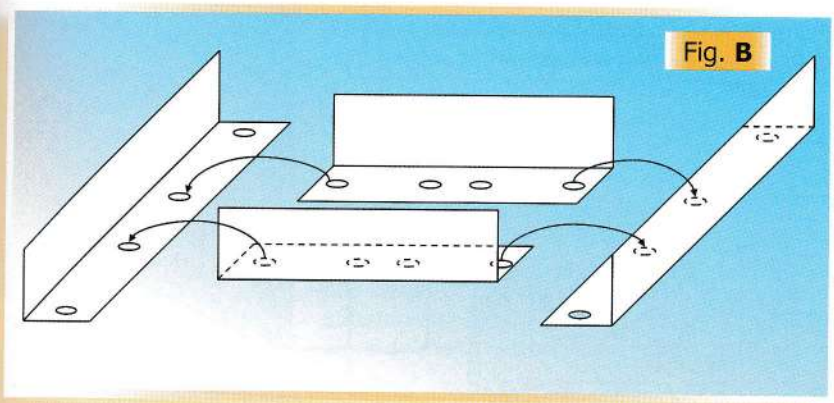


Fig. **B**

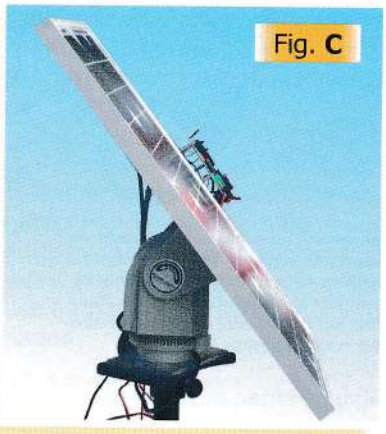


Fig. **C**

façon à permettre au microcontrôleur de mesurer et de contrôler la tension de sortie du panneau photovoltaïque.

En parallèle nous avons connecté un **régulateur de charge** modèle « **SOL4UCN2** » (voir l'encadré consacré au régulateur de charge) sur lequel est reliée une batterie au plomb de 12 V de 7 Ah à 30 Ah, bien que pour les tests en laboratoire, nous avons utilisé un modèle plus petit de 4 Ah.

Le « **SOL4UCN2** » est un régulateur de type « **shunt** » qui, lorsque la **batterie est complètement chargée, dérive le courant** du panneau photovoltaïque de façon à le réduire progressivement jusqu'à le court-circuiter (shunt). Le régulateur court-circuite de manière périodique et augmente la fréquence pour réduire le courant du panneau. En résumé, le régulateur absorbe en excès le courant provenant du panneau photovoltaïque afin de ne pas **endommager la batterie et maintenir sa charge**.

Le « SOL4UCN2 » peut accepter en entrée une tension allant jusqu'à 24 et charger des batteries de 12 V.

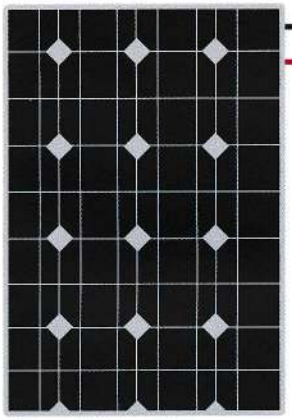
Réalisation pratique

Passons maintenant à la pratique pour construire ce « tracker » solaire.

Matériel à installer à l'extérieur

Matériel installé dans un endroit à l'abri des intempéries.

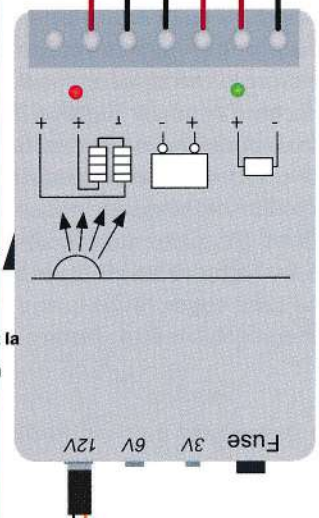
Panneau solaire, celui du prototype est un 24 W (17,4 V max. ; 1,38 A max.).



Batterie au plomb 12 V 4 Ah

Positif

Négatif



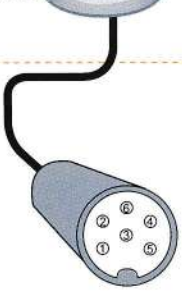
ON = Système allumé

OFF = Uniquement la charge de la batterie

Régulateur de charge pour panneaux solaires, modèle SOL4UCN2. Il dispose d'une sortie directe pour alimenter une charge (stabilisée à 12 V, 4 A max), cette sortie est déconnectée lorsque la tension de la batterie descend à 10,5 V. Il dispose aussi de 3 sorties stabilisées de 3 VDC, 6 VDC et 12 VDC (cette dernière alimente les circuits électroniques) lorsque la tension est présente. Il dispose aussi d'un interrupteur d'allumage (ON/OFF).

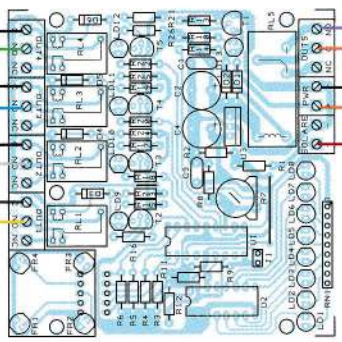
Connecteurs pour le raccordement de la charge externe, ils fournissent 12 VDC/4 A max. Les charges nécessitant un courant plus élevé doivent être connectées en parallèle avec la batterie.

Support « PAN-TILT » à l'extérieur alimenté en 24 VAC.

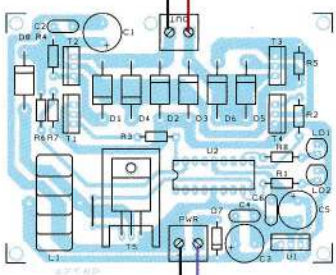


1 Commun
5 Droite
3 Bas
6 Gauche
4 Haut
2 Nc.

Connecteur du câble de commande du support. Le support « PAN-TILT » dispose d'un connecteur DIN à 6 pôles + détrompeur. Attention la taille du connecteur n'est pas à l'échelle, c'est pour rendre plus compréhensible le brochage.



Circuit de l'unité de contrôle, alimentée avec la sortie stabilisée 12 VDC provenant du régulateur.



Convertisseur DC-AC : il utilise l'alimentation stabilisée de l'unité de contrôle et alimente en 24 VAC le support « PAN-TILT »

Tout d'abord, procurez-vous tous les composants (panneau solaire, régulateur de charge « **SOL4UCN2** », batterie et support inclinable « PAN-TILT » VISP2).

Vous devez juste réaliser l'unité de contrôle et le convertisseur DC-AC, téléchargez les **typons des circuits imprimés** ainsi que les **programmes** « **.hex** » des microcontrôleurs sur notre site web **www.electroniquemagazine.com** dans le sommaire détaillé du **numéro 128** à l'onglet « **Télécharger** ».

Gravez et percez les circuits (n'oubliez pas que vous pouvez aussi les fabriquer directement avec la 3DRAG sans produits chimiques).

Découpez la partie (selon vos besoins) où doivent être implantées les photorésistances et mettez-là de côté. Sur la partie principale commencez par souder les résistances et les diodes en prenant soin de l'orientation des bagues.

Ensuite soudez les supports du microcontrôleur et du PCF8574, puis continuez avec les condensateurs non polarisés et enfin les électrolytiques.

Une attention particulière doit être accordée aux cellules photoélectriques, prenez la partie que vous avez découpée précédemment. Avant de souder les photorésistances vous devez introduire dans chaque broche (ou patte) un morceau de gaine thermorétractable afin d'éviter des court-circuits.

Ensuite chaque photorésistance doit être insérée dans un morceau de gaine thermorétractable noir d'environ 2 cm de long, de sorte que la surface sensible à la lumière arrive sur le bord supérieur de la gaine.

Chauffez légèrement la gaine pour solidifier l'ensemble. Cette opération est nécessaire pour que la lumière frappe de manière frontale chaque photorésistance, c'est-à-dire pour les rendre directives de sorte que chacune détectera la lumière dans une seule direction et le « tracker » sera très précis. Finissez par souder tous les relais, les borniers ainsi que les LED.

Procédez de la même manière pour le circuit imprimé du convertisseur DC-AC, n'oubliez pas en dernier de fixer le radiateur sur le circuit et la self. Après avoir vérifié que tous les composants des circuits sont correctement soudés, et avant d'effectuer le câblage général, les photorésistances de l'unité de contrôle doivent être étalonnées.

Prenez une alimentation, de préférence stabilisée, capable de fournir 12 VDC et connectez respectivement le « + » au point « + PWR » et le « - » au point « - PWR » du bornier (proche du relais RL5).

Orientez les photorésistances temporairement vers le même côté (en ayant au préalable installé correctement les gaines) et éclairez-les à l'aide d'un projecteur.

Fermez le cavalier J1 et attendez que toutes les LED soient allumées pendant 2 secondes.

A ce stade, le microcontrôleur est calibré. Retirez J1 et inclinez la lampe à 45° vers l'extérieur comme visible en figure 8.

En ce qui concerne le raccordement électrique des différents ensembles, reportez-vous au schéma de câblage général.

Pour la partie mécanique, le plus pratique et facile à faire, est de laisser à l'extérieur le panneau solaire, le support et les 4 photorésistances (la partie du circuit imprimé contenant les photorésistances peut-être découpée et reliée par un câble blindé).

Tu ne nous en veux pas Galilée ?

Si ... dans ces pages, nous avons dit à plusieurs reprises que de l'aube au coucher du Soleil, celui-ci se déplace, nous savons que c'est un abus de langage (et nous espérons que vous le savez aussi ...) car en fait c'est la Terre qui, en tournant sur elle-même et autour du Soleil, nous fait voir celui-ci dans différentes positions de l'hémisphère céleste. Nous avons cependant préféré parler de mouvement du Soleil, car cela est plus simple pour la compréhension de l'article, la mécanique céleste étant une science compliquée. Pour nous, simples mortels, le Soleil se déplace dans le ciel.

Galilée était un mathématicien, géomètre, physicien et astronome italien du XVIIe siècle, né à Pise le 15 février 1564 et mort à Arcetri, près de Florence, le 8 janvier 1642 (78 ans). Il a inventé la lunette astronomique pour procéder à des observations qui ont bouleversé les fondements de la discipline astronomique.

Cet homme de sciences s'est ainsi posé en défenseur de l'approche modélisatrice Copernicienne de l'Univers, proposant d'adopter l'héliocentrisme et les mouvements satellitaires, et ses observations et généralisations se sont alors heurtées aux critiques des philosophes partisans d'Aristote, proposant un géocentrisme stable, une classification des corps et des êtres, un ordre immuable des éléments et une évolution réglée des substances, ainsi qu'aux théologiens Jésuites de l'Église Catholique romaine, soucieux alors de préserver les fondements de la transsubstantiation.

Galilée, qui ne disposait pas de preuves directes du mouvement terrestre, a parfois oublié la prudence de ses protecteurs religieux.

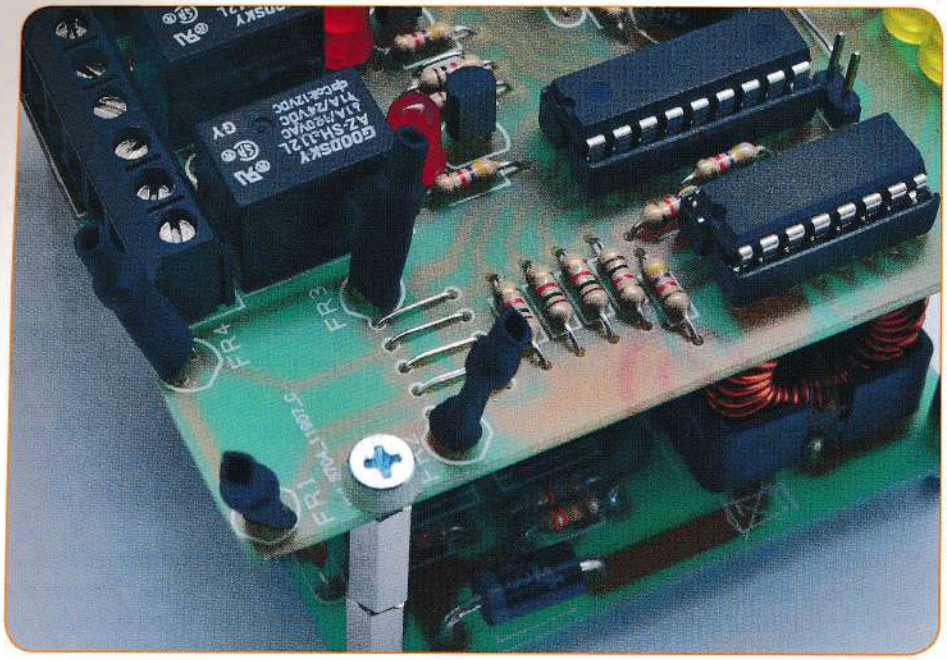


Figure 8 : les photorésistances sont rendues « directives » à l'aide d'une gaine thermorétractable noire. Elles sont inclinées de 45° à 60 ° par rapport au circuit.

Tandis que l'unité de contrôle, le convertisseur DC-AC, le régulateur de charge SOL4UCN2 et la batterie seront mis à l'abri (par exemple dans le grenier) et installés dans des boitiers appropriés.

Tout d'abord, procurez-vous un boitier en plastique avec un couvercle transparent et insérez l'unité de contrôle et le convertisseur DC-AC, montés comme illustré sur la figure 6.

Pour la fixation des 2 circuits dans le boitier, utilisez des entretoises hexagonales de longueur adéquate.

Percez des trous sur les côtés du boitier de manière à faire passer les câbles suivants :

- les câbles du support inclinable « PAN-TILT » VISP2 ;

- les câbles du panneau solaire ;

- les fils des 4 photorésistances, celles-ci doivent être fixées solidement sur le panneau solaire et protégées contre les intempéries par un 2ème boitier étanche transparent (IP68) de dimensions appropriées.

Si vous le souhaitez, ou si vous avez assez d'espace à l'intérieur du premier boitier, vous pouvez également intégrer le régulateur de charge et la batterie, mais dans ce cas, vous devrez percer des trous supplémentaires pour les câbles.

Lors du câblage du régulateur de charge et de l'ensemble du système, veillez à respecter la polarité de toutes les connexions car cela pourrait provoquer des dommages.

À ce stade, vous avez juste à fixer le panneau solaire à la plaque du support. L'ensemble dépend à la fois du panneau et du support que vous allez utiliser, nous ne pouvons pas vous dire de manière précise comment vous devez procéder pour l'assemblage.

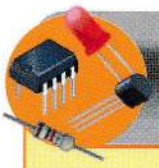
Reportez-vous à l'encadré nommé « Le support du panneau » dans lequel nous donnons quelques lignes directrices que nous avons suivies au cours de la mise en œuvre de notre prototype, cela vous sera certainement utile.

Choisissez enfin l'endroit où vous allez installer le support, cela peut être un poteau, un pilier ou un bras métallique, ensuite fixez la base.

Une fois que toutes les connexions sont effectuées, vous verrez normalement le panneau solaire suivre le mouvement du Soleil, si cela n'est pas le cas vérifiez à nouveau les connexions.

Repérez la position et vérifiez au bout de 10 ou 20 minutes que le panneau soit toujours orienté de la même manière.

Si vous constatez que le **panneau s'adapte trop lentement**, où s'il reste trop longtemps dans une position fixe, **agissez sur le trimmer R7** pour obtenir un mouvement plus approprié.



Comment construire ce montage

Le **panneau solaire MCRY24**, le régulateur de charge « **SOL4UCN2** », la **batterie RMS12V6AH5** sont disponibles auprès de **COMELEC**.

Les typons des circuits imprimés ainsi que les **deux programmes** au format « **.hex** » des microcontrôleurs sont téléchargeables sur notre site : **www.electroniquemagazine.com** dans le sommaire détaillé de la revue numéro 128 section « Télécharger ».

ET 741

Testeur de servomoteurs pour modélisme

Ce montage permet de vérifier l'efficacité des servomoteurs analogiques pour modélisme. Grâce au programme, dont nous publions le code source complet en BASIC, ce montage permet de tester les servomoteurs afin d'identifier facilement le « neutre » et contrôler le temps de réponse ainsi que le bon fonctionnement des engrenages du « servo ». A l'aide de 3 boutons, il affiche les paramètres de fonctionnement sur un afficheur LCD alphanumérique.

de Mirco Segatello

Le modélisme radiocommandé est un passe-temps passionnant pratiqué par des personnes de tous âges et de niveaux différents dans le modélisme. Les jeunes, la plupart du temps au début, réalisent les modèles les plus simples de manière à être en mesure de les piloter rapidement (typiquement ce sont des voitures de course ou des tous terrains).

Par contre les professionnels et les « anciens » du modélisme passent souvent plus de 10 ans de leur vie pour réaliser des modèles radiocommandés des plus complexes (typiquement des navires, avions, sous-marins et hélicoptères), parfois si précis et raffinés qu'ils peuvent mériter la médaille d'or des championnats nationaux et internationaux de modélisme.

Entre ces deux extrêmes, nous trouvons le reste des fans du modélisme radiocommandé, avec plus ou moins de qualité et de patience, ainsi que des différences considérables en termes de types, de détails du modèle et de capacités du maquettiste.

Cependant une situation particulière unit tous les fans : soit le jeune qui a

fait des sacrifices pour épargner afin d'acquérir sa première voiture radiocommandé, soit celui qui a passé de nombreuses années pour construire un modèle hors du commun ; ils sont inquiets de perdre ou d'endommager leur modèle à cause d'un défaut de l'un des dispositifs de contrôle.

Nous parlons ici des servomoteurs, des récepteurs et émetteurs qui, malheureusement au cours du temps, s'abiment lorsqu'ils sont soumis à des allumages fréquents et des chocs lors des déplacements. Dans ces cas, il peut arriver surtout avec les avions, les hélicoptères et les bateaux, que lors du déplacement ils deviennent incontrôlables, et donc il suffit d'imaginer ce qu'il se passe, c'est-à-dire une perte assez importante en coût et en temps. Parmi les différents dispositifs, les servomoteurs sont parmi les plus sensibles, car ils comprennent des pièces électroniques et des pièces mécaniques,

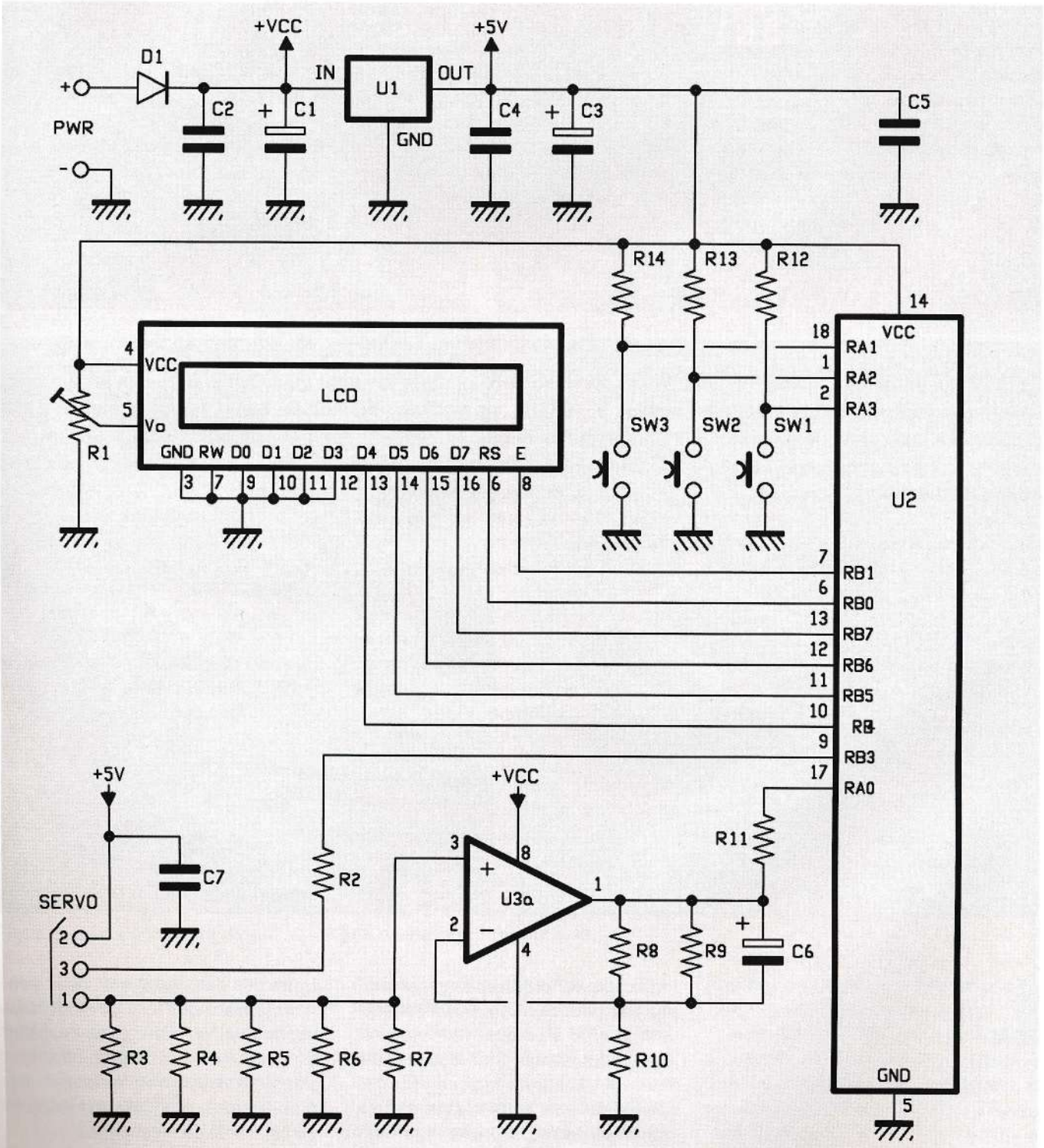


Figure A : schéma électrique du testeur de servomoteurs, il est très simple et se compose d'un microcontrôleur PIC16F818 (U2) et d'un amplificateur opérationnel LM358 (U3a).

ces dernières étant beaucoup plus sensibles aux chocs et à l'usure.

C'est pour cela que nous vous proposons dans cet article un circuit très utile, avec lequel **il est possible de tester un grand nombre de servomoteurs analogiques pour modélisme**.

Ce montage effectue le **test complet** à l'aide d'un microcontrôleur relié à un afficheur LCD et 3 boutons. Il vous permet de **contrôler précisément le mouvement du servomoteur**, et affiche sur l'écran la **durée de l'impulsion** de commande et la **consommation de courant en fonctionnement**. La **mesure du courant est une excellente indication sur le bon fonctionnement de la mécanique**, un **servomoteur en parfait état** à vide **consomme peu de courant**.

Si cela n'est pas le cas, c'est **symptomatique de problèmes mécaniques** (tels que le manque de lubrification, le grippage dû à la poussière, des engrenages avec des dents cassées, etc.) qui obligent le servomoteur à toujours surconsommer du courant pour compenser les problèmes mécaniques, il doit par conséquent être remplacé.

En ce qui concerne notre montage, nous allons voir comment avec 3 boutons et quelques réglages simples, vous pouvez effectuer à la fois le **contrôle manuel et automatique du mouvement du servomoteur**. La dernière caractéristique est particulièrement utile pour vérifier le fonctionnement correct une fois que le servomoteur est installé sur le modèle. Nous avons conçu en plus, un circuit flexible qui vous permet de procéder à la **vérification des régulateurs de vitesse** (Electronic Speed Control ou Contrôle Electronique de la Vitesse) **montés sur le modèle, sans avoir besoin de récepteur radio**.

Lorsque les tests des moteurs électriques sont effectués sur le banc d'essai ou lors de la phase de rodage et du test de traction, ce circuit est très utile pour commander avec précision le régulateur de vitesse. Nous allons maintenant détailler les possibilités de ce testeur pour servomoteurs.

Le schéma électrique

Comme vous pouvez le voir sur la figure A, le schéma électrique du circuit est très simple et se compose essentiellement d'un microcontrôleur **PIC16F818** (U2) de chez Microchip et d'un amplificateur opérationnel **LM358** (U3a). Le PIC dont le coût est faible, est complet et dispose d'un oscillateur interne et de plusieurs entrées analogiques.

Il correspond exactement aux fonctions dont nous avons besoin. En outre il est doté d'une sortie PWM, qui malheureusement ne peut être utilisée car sa résolution n'est pas suffisante.

Le générateur interne PWM du PIC a une résolution maximale de 10 bits, ce qui nous permettra d'obtenir un rapport cyclique (duty-cycle) en sortie du signal carré de 1023 valeurs.

Pour bien gérer un servomoteur, le **signal de commande doit être une onde carrée d'une impulsion d'une durée comprise entre 1 et 2 ms, répété toutes les 20 ms** en fonction de la durée de l'impulsion. Le **servomoteur doit garder une certaine position entre ± 60 °** (l'angle du mouvement dépend du type de servomoteur).

En ne disposant que de 1023 pas de 19,5 ms (20 ms/1023 = 19,5 ms environ), nous aurions seulement 51 pas pour gérer l'impulsion de 1 à 2 ms (51 x 19.5ms = 1ms).

Lorsque nous programmons le PIC afin de générer une impulsion d'une durée variable grâce à la fonction « **PULSOUT** » du **compilateur** « **PicBasic** », nous obtenons **une précision qui est 2 fois plus grande que l'impulsion que nous voulons obtenir**. En effet l'instruction a une résolution de 10 microsecondes ce qui correspond à une centaine de pas toutes les millisecondes.

Revenons maintenant au circuit, il a été conçu pour être alimenté avec un bloc secteur économique non stabilisé de **9 V** à **12 V** et de **10 W** à **20 W** (2 A). L'étage d'alimentation est constitué de la **diode D1 de protection contre les**

Caractéristiques techniques du testeur de servomoteurs :

- Visualisation par afficheur LCD rétroéclairé alphanumérique (2 lignes de 16 caractères) ;
- réglages par 3 boutons poussoirs « P+ », « P- » et « Enter » ;
- Modes de test : manuel et automatique ;
- Test automatique : lent (« Slow ») et rapide (« Fast ») ;
- Paramètres ajustables : valeur minimale et maximale des impulsions de commande ;
- Vérifications possibles sur le servomoteur :
- vérification de la vitesse de rotation ;
- identification de la position du « neutre » ;
- mesure du courant consommé au repos (max 1A) ;
- mesure du courant consommé en mouvement à la fois à vide et en charge (max 1 A) ;
- vérification de l'état des engrenages lors du test lent (« Slow ») ;
- vérification du temps de réponse lors du test rapide (« Fast ») ;
- Alimentation de 9 VDC à 12 VDC, 100 mA minimum, 1 A maximum. Le courant maximal est approximativement égal à celui consommé par le servomoteur.

inversions de polarité et par un régulateur de tension **7805** (U1) qui fournit une tension de 5 V stabilisée au circuit et au servomoteur. Comme vous pouvez le remarquer sur le schéma, nous avons 2 tensions d'alimentation, le **+ 5 V** qui est destiné au PIC et à l'afficheur LCD et le **+ VCC** qui alimente l'amplificateur opérationnel. Nous verrons plus tard à quoi sert cette deuxième tension d'alimentation.

Le **PIC gère directement l'afficheur LCD** (le contraste est ajusté avec le trimmer R1), ainsi que les 3 boutons (connectés aux broches RA1, RA2 et RA3) utilisés lors de la lecture du courant consommé par le servomoteur.

La **mesure est effectuée grâce à l'amplificateur opérationnel U3a** et les **5 résistances de 1 Ω en parallèle** (R3 à R7) qui constituent une résistance de dérivation d'une valeur de **0,2 Ω**. On utilise le terme de résistance de « **shunt** » qui est une résistance de faible valeur connectée en série avec le dispositif dont on veut mesurer le courant. La **chute de tension** aux bornes de la **résistance de « shunt »** est **proportionnelle au courant circulant** dans celle-ci (ce qui correspond au courant consommé par le dispositif, dans notre cas c'est le servomoteur), et doit être de faible valeur par rapport à la tension d'alimentation du circuit (par exemple 1/20).

Nous avons utilisé **5 résistances en parallèle de faible valeur capables de mesurer un courant maximal de 1 A**, avec une bonne précision de manière à rendre le circuit de mesure suffisamment précis sans la nécessité de réglages supplémentaires.

Les résistances de puissance, bon marché et facilement disponibles, ont une tolérance de ± 10 %, et **ne conviennent pas à notre cahier des charges**. Nous avons pris 5 résistances de « 1/4 W » et « ± 5% » placées en parallèle, nous obtenons une précision et une puissance 5 fois plus grandes (théoriquement ± 1% et 1,25 W).

La **chute de tension aux bornes de la résistance de shunt**, qui est de **quelques centaines de mV par rapport à la masse**, est trop faible pour être lue directement par l'entrée analogique du PIC, elle est donc amplifiée.

Pour **réaliser l'amplification**, nous utilisons un amplificateur opérationnel **LM358** (U3a) pouvant **fonctionner sous une tension unique par rapport à la masse** (alimentation simple ou asymétrique) et permet d'amplifier des signaux à partir de 0 V par rapport à la masse.

Avec un dimensionnement approprié les **résistances de contre-réaction déterminent le gain du LM358**, il est possible de prélever **en sortie de U3a une tension comprise entre 0 V et 5 V** et qui peut être **appliquée à l'entrée analogique** du microcontrôleur **PIC16F818** (U2).

Cependant il faut garder à l'esprit un détail très important, la tension minimale que l'on peut prélever en sortie du LM358 est de 0 V, mais le maximum est d'**environ de 2 V inférieur à la tension d'alimentation**. Cela signifie que pour obtenir en sortie de U3a une excursion de 5 V de la tension, l'**amplificateur opérationnel doit être alimenté avec une tension légèrement plus élevée** (d'au moins 2 V à 3 V).

Voici pourquoi la **seconde alimentation « + VCC » est utilisée pour alimenter l'amplificateur opérationnel U3a**, elle est prise avant le régulateur de tension U1 et varie de 9 V à 12 V en fonction du bloc secteur. Le condensateur **C1 filtre l'ondulation résiduelle** provenant du bloc secteur non stabilisé, de manière à alimenter l'amplificateur opérationnel avec une tension assez stable.

Compte tenu que la tension d'alimentation est supérieure à 5 V (9 V à 12 V), la sortie de l'amplificateur opérationnel pourrait théoriquement être supérieure à 5 V. Dans ce cas, que se passerait-il sur l'entrée du microcontrôleur ? Serait-il détruit ? Pour répondre à ces questions, **notez la présence de la résistance R11** (1 kΩ).

Dans les conditions normales, le courant circulant dans l'**entrée analogique** du PIC est d'environ **500 nA** et **la chute de tension aux bornes de R11 est négligeable**, nous pouvons affirmer que toute la tension de sortie de l'amplificateur opérationnel atteint l'entrée du PIC.

Voici un magnifique modèle à l'échelle 1 : 100 de la frégate F576 ESPERO (MMI). Fabriquée par E. Effalli en 8 ans, il a obtenu une médaille d'or aux championnats d'Italie. Cette frégate navigue et dispose de 14 servomoteurs pour le déplacement, ainsi que des moteurs, du gouvernail, de l'hélicoptère sur le pont d'envol, des ancres, des canons et des radars.

Ici un modèle de radiocommande très répandu dans le monde du modélisme, le Futaba 14MZ. Il dispose de 14 canaux programmables, de commandes vocales, d'un écran LCD couleur tactile, et des programmes pré-chargés pour Avion - Hélicoptère – Planeur. Ce modèle est équipé d'un bi-processeur et d'une transmission à 2,4 GHz qui en fait l'une des radiocommandes les plus évoluées pour le modélisme.

Si l'amplificateur opérationnel présente en sortie une tension supérieure à 5 V (donc trop élevée pour le microcontrôleur), les **diodes de protection internes du PIC deviendraient conductrices** et le courant dans ces diodes **serait limité par la résistance R11 afin d'assurer une protection efficace de l'entrée analogique du PIC**.

A la place du LM358, nous aurions pu utiliser un amplificateur opérationnel « **rail-to-rail** » qui garantit une tension de sortie comprise entre le potentiel de masse et la tension d'alimentation (par exemple s'il est alimenté en 5 V, la tension de sortie peut varier entre 0 V et 5 V), mais ce type d' amplificateur opérationnel est plus couteux et difficile à trouver dans les magasins.

A ce stade, nous devons calculer la valeur de l'amplification de U3a en considérant certains facteurs, la résolution du convertisseur interne du PIC est de 10 bits, la conversion fournit donc un nombre compris entre 0 et 1023.

Le **PIC convertit** les signaux numériques entre **0 V** et **5 V**, où le « **0 V** » correspond à la valeur numérique « **0** » et le « **5 V** » correspond à la valeur numérique « **1023** ». Pour éviter d'avoir à effectuer des calculs compliqués dans le microcontrôleur, nous utilisons une échelle directe :

- **0 mA correspond à la valeur décimale 0** ;

- **1023 mA correspond à la valeur décimale 1023**.

À ce stade, les calculs sont simples car l'**amplificateur opérationnel doit fournir une tension de 5 V en sortie lorsqu'un courant de 1,023 A circule dans la résistance de shunt**. Sachant que la résistance de shunt a une résistance de 0,2 Ω, la chute de tension maximale à ses bornes est égale à :

0,2 Ω x 1,023 A = 0,2046 V

pour laquelle le gain de l'amplificateur opérationnel doit être de :

5 V / 0,2046 V = 24,45.

L'amplificateur opérationnel U3a est utilisé dans une configuration « non-inverseur » et son gain est donné par la formule :

Av=[(R8 // R9) / R10] + 1

où le terme « **R8 // R9** » signifie que la valeur de la résistance équivaut à **R8 en parallèle avec R9**. Par conséquent le rapport [(R8 // R9) / R10] doit valoir **23.45**. Nous avons dû utiliser 2 résistances en parallèle (R8 et R9) pour obtenir le rapport 23,45, car la valeur unique de la résistance n'est pas disponible dans les gammes standard et donc dans le commerce.

Avec les valeurs choisies, le montage en parallèle de R8 (2,7 kΩ) et R9 (18 kΩ) donne la valeur de 2348 Ω et donc par rapport à R10 qui vaut 100 Ω, nous obtenons la valeur de 23,48 et un gain de 24,48, ce qui est proche de **24,45**.

Le condensateur **C6 réalise un filtre « passe-bas »** utile pour atténuer les signaux parasites générés par les pointes de courant dans le servomoteur, toujours présentes pendant les phases de fonctionnement normal.

Réalisation pratique

Tout d'abord vous devez réaliser le circuit imprimé simple face que vous pouvez télécharger sur notre site www.electroniquemagazine.com dans le sommaire détaillé du numéro 128 à l'onglet « Télécharger ».

Une fois le circuit gravé et percé, commencez par souder les 4 ponts à l'aide de pattes de composants dont 3 se situent en dessous de l'afficheur et le 4eme proche du régulateur U1 (aidez-vous des figures C, D et E du « Plan de montage »).

Ensuite continuez par souder les composants ayant un profil bas : les résistances, la diode D1 en respectant sa polarité (la bague qui représente la cathode doit être orientée vers l'afficheur), les condensateurs non polarisés, le trimmer R1 et les 3 boutons poussoirs.

Soudez les supports du microcontrôleur U2 et de l'amplificateur opérationnel U3, puis les condensateurs polarisés (électrolytiques, le « - » est indiqué sur le boitier), ensuite la fiche d'alimentation.

Terminez le montage en soudant le régulateur de tension U1 qui doit être monté verticalement avec la face métallique orientée vers l'extérieur du circuit imprimé, ensuite les 16 broches femelles de l'afficheur et les 3 broches mâles pour le servomoteur au pas de 2.54 mm. Enfin soudez sur l'afficheur 16 broches mâles au pas de 2,54 mm sur le côté opposé de l'écran, elles viendront s'enficher dans les 16 broches femelles du circuit, attention respectez bien le brochage, la broche 1 est vers l'intérieur de l'afficheur. Selon le type d'afficheur dont vous disposez, il se peut que le brochage soit différent, dans ce cas vérifiez toujours le brochage à partir de la documentation du constructeur.

Avant les tests, vérifiez soigneusement le montage et les éventuelles erreurs. Sans insérer le microcontrôleur et l'afficheur, alimentez le circuit. Vous devez avoir une tension continue de 9 V à 12 V.

Assurez-vous que sur la broche 8 de l'amplificateur opérationnel U3a soit présent une tension de 9 V à 12 V (moins la chute de tension de la diode D1) et que sur la broche 14 du support du microcontrôleur et la broche 4 de l'afficheur vous ayez une tension de 5 V.

Coupez l'alimentation et attendez quelques minutes pour que les condensateurs électrolytiques se déchargent complètement, puis insérez dans le support le microcontrôleur correctement programmé avec le fichier MF741.hex (également téléchargeable gratuitement sur notre site avec le circuit imprimé). Le microcontrôleur programmé MF741 en usine est disponible auprès de COMELEC.

Mettez le circuit sous tension et ajustez le contraste de l'afficheur à l'aide du trimmer R1, vérifiez que l'écran affiche le message « TDS V1 » (voir la figure 2). Le circuit est maintenant prêt à être utilisé.

Utilisation

Le **firmware**, dont le **code source complet est reporté dans l'encadré nommé « Listing 1 »**, n'est pas très complexe. Il doit tout simplement gérer de manière appropriée le signal de commande du servomoteur, ainsi que l'afficheur LCD et les boutons poussoirs.

Lorsque le circuit est mis sous tension, l'écran LCD affiche la version du firmware et l'écran suivant (voir la figure 3), qui apparaît quelques secondes après, visualise la durée de l'impulsion de commande présente en sortie ainsi que la valeur du courant consommé par le servomoteur. Le **montage doit être connecté au servomoteur par l'intermédiaire du connecteur « SERVO »** en respectant les **instructions de la figure 1**.

Si le servomoteur est « libre » et n'est pas en situation d'effort, le courant

Plan de montage

Lors du montage, n'oubliez pas de souder les 4 ponts à l'aide de pattes de composants sous l'afficheur et proche du régulateur de tension. L'afficheur LCD peut être fixé mécaniquement à l'aide de 2 entretoises hexagonales de type 3MA placées de chaque côté des broches. Vous pouvez compléter par une 3eme entretoise du côté droit de l'afficheur (vu de dessus), attention lorsque vous percez le trou correspondant à l'entretoise sur le circuit imprimé de ne pas couper la piste de cuivre qui est proche.

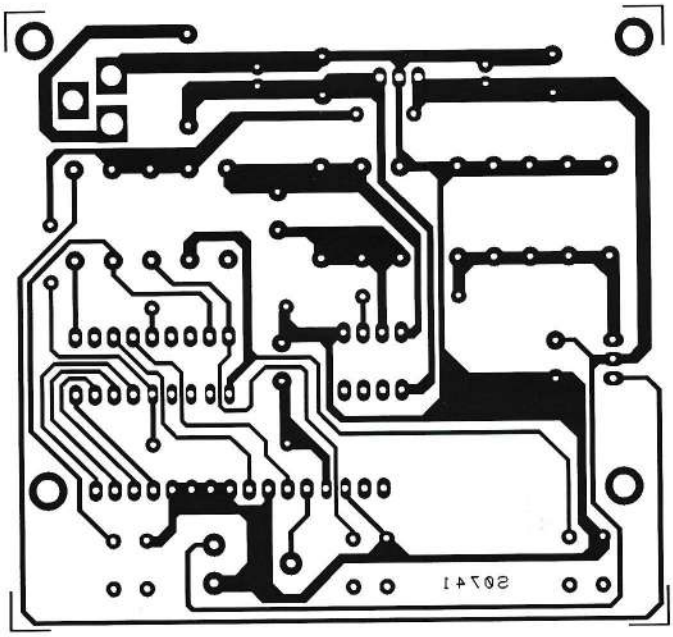


Figure B : circuit imprimé à l'échelle 1 : 1 du testeur pour servomoteur.

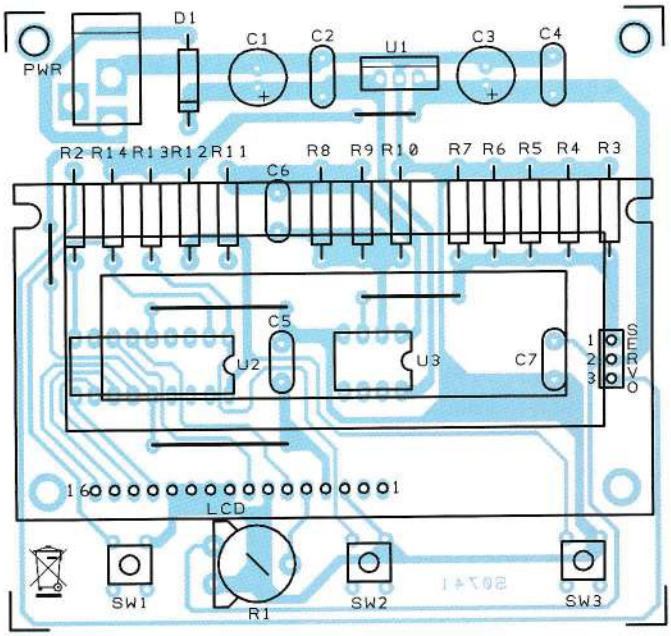


Figure C : schéma de câblage du testeur, n'oubliez pas de souder les 4 ponts.

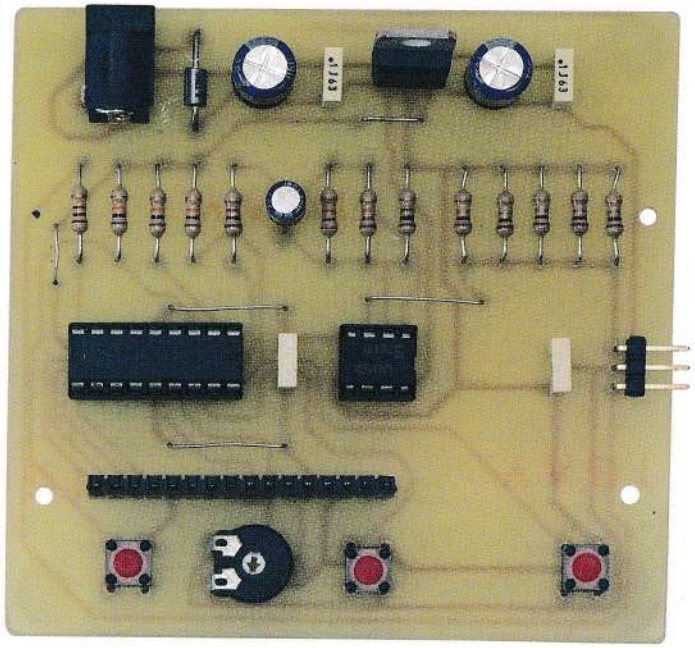


Figure D : photo de l'un de nos prototypes sans l'afficheur.

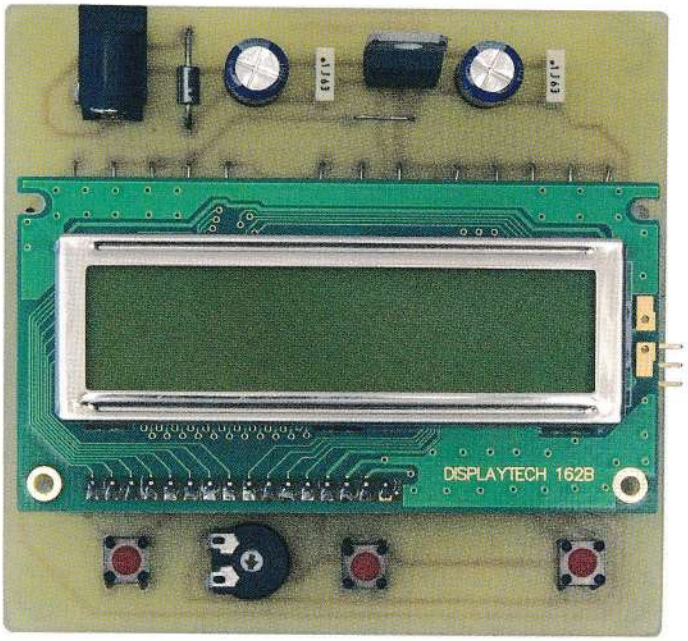


Figure E : photo de l'un de nos prototypes avec l'afficheur en place.

Liste des composants du testeur de servomoteurs

R1..... trimmer 10 kΩ monotour
R2..... 100 Ω
R3..... 1 Ω
R4..... 1 Ω
R5..... 1 Ω
R6..... 1 Ω
R7..... 1 Ω
R8..... 2,7 kΩ
R9..... 18 kΩ
R10.... 100 Ω
R11 ... 1 kΩ
R12 ... 10 kΩ
R13 ... 10 kΩ
R14.... 10 kΩ

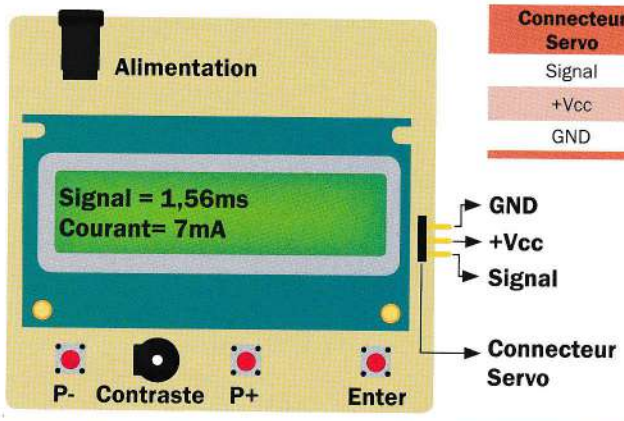
C1..... 220 µF / 25 V électrolytique
C2..... 100 nF multicouche
C3..... 100 µF / 25 V électrolytique
C4..... 100 nF multicouche
C5..... 100 nF multicouche
C6 ..... 10 µF / 25 V électrolytique
C7 ..... 100 nF multicouche

U1..... 7805
U2..... PIC16F818 (MF741)
U3..... LM358
D1..... 1N4007
LCD.... Afficheur 2 lignes 16 caractères
SW1... bouton poussoir
SW2... bouton poussoir
SW3... bouton poussoir

Divers

Fiche alimentation pour circuit imprimé
Support ci 2 x 4 broches
Support ci 2 x 9 broches
Barrette male 3 pôles coudée
Barrette male 16 pôles
Barrette femelle 3 pôles

Lorsque vous branchez le connecteur du servomoteur, vous devez prêter attention au sens d'insertion et à la couleur des fils.



Connecteur Servo	Couleur des fils du servomoteur
Signal	Jaune ou orange
+Vcc	Rouge
GND	Noir ou marron

Figure 1

Listing 1

Vous pouvez voir ci-dessous le code source complet du programme en **BASIC** commenté à chaque ligne. Notez que le **firmware effectue 2 tests**, un **test lent** et un **test rapide**, au cours desquels le servomoteur est mis en mouvement entre 2 positions (minimale et maximale définies par l'utilisateur). Dans les 2 cas au cours des tests, sont indiquées à la fois la durée de l'impulsion de contrôle en temps réel ainsi que la consommation de courant moyenne pendant le cycle de déplacement. Cette dernière est mesurée à chaque changement de direction du mouvement.

```
'*************************************************
'* Name         : TESTSERVO.BAS                  *
'* Author       : Mirco Segatello                *
'* Version      : PIC 16F818 V1.0                *
'*************************************************
@     DEVICE  INTRC_OSC
@     DEVICE  BOD_OFF
@     DEVICE  LVP_OFF
@     DEVICE  CPD_OFF
@     DEVICE  WDT_OFF
@     DEVICE  PWRT_OFF
@     DEVICE  MCLR_OFF
'*****************************************************************
'  Définitions
'*****************************************************************
DEFINE  LCD_DREG PORTB                  ' Afficheur sur PORTB
DEFINE  LCD_DBIT 4                      ' Initialise bit 4.
DEFINE  LCD_RSREG PORTB                 ' Ligne RS PORTB
DEFINE  LCD_RSBIT 0                     ' Bit 0.
DEFINE  LCD_EREG  PORTB                 ' Ligne E PORTB
DEFINE  LCD_EBIT  1                     ' Bit 1.
DEFINE  LCD_BITS  4                     ' Données afficheur sur 4 bits.
DEFINE  LCD_LINES 2                     ' LCD avec 2 lignes.
DEFINE  LCD_COMMANDS 2000               ' Retard de commande.
DEFINE  LCD_DATAUS 50                   ' Retard des données.
DEFINE  ADC_BITS 10                     ' Résolution ADC 10 bits.
DEFINE  ADC_CLOCK 3                     ' Horloge ADC
                                        ' (OSC/32= 2, rc = 3).
DEFINE  ADC_SAMPLEUS 500                ' vitesse de conversion.
DEFINE  OSC 4                           ' Horloge du PIC à 4 MHz.

'*****************************************************************
'  Paramétrages des registres internes et des variables
'*****************************************************************
OSCCON= $60                             ' Réglage de l'horloge interne à 4 MHz.
ADCON0= %11000001                       ' Habilite le convertisseur ADC.
ADCON1= %10001110                       ' Définit les entrées : RA0 analogique, RA1,RA2, RA3 digitales
TRISA   = %11111111                     ' RA0 à RA5 en entrées
TRISB   = %00000000                     ' RB0 à RB7 en sorties.
ISV     VAR byte                        ' Courant du servomoteur.
PWO     VAR byte                        ' largeur de l'impulsion PWM.
int     VAR byte                        ' Partie entière
fra     VAR byte                        ' Partie décimale (reste de la division).
Delay   VAR byte                        ' Durée répétition poussoir
Speed   VAR byte                        ' Vitesse de mouvement du servomoteur.
TV      VAR byte                        ' Vitesse de rafraîchissement de l'afficheur.
PWUP    VAR byte                        ' Limite maximale du mouvement.
PWDW    VAR byte                        ' Limite minimale du mouvement.
B0      VAR byte                        ' Variable à usage interne.
ISVM    VAR word                        ' Courant maximal du servomoteur.
```

```
i       VAR byte                        ' Variable de service.
TMP     VAR byte                        ' Variable de service.
symbol  PEN = PORTA.1                   ' Poussoir «Enter».
symbol  PUP = PORTA.2                   ' Poussoir augmente (+).
symbol  PDW = PORTA.3                   ' Poussoir diminue (-).
symbol  SVR = PORTB.3                   ' Sortie du signal vers le servomoteur
PORTB   = 0                             ' Initialise le PORTB.
PWO     = 150                           ' Met PWO à 150.
PWUP    = 200                           ' Met PWUP à 200.
PWDW    = 100                           ' Met PWDWM à 100.
ISV     = 0                             ' Initialise ISV à 0
Delay   = 0                             ' Initialise à 0 variables
TV      = 0                             ' les variables
Speed   = 0                             ' Delay, TV, Speed

'*****************************************************************
'  Message de mise sous tension
'*****************************************************************
pause   200                             ' Pause de 200 ms
lcdout  $FE,1                           ' Efface l'afficheur.
lcdout  $FE,2                           ' Curseur en début de ligne.
lcdout  "TDS V1"                        ' Ecrit le message «TDS V1»
pause   2000                            ' Pause de 2 secondes.
call    Lcd_New                         ' Met à jour l'afficheur.
'*****************************************************************
'  Programme principal
'*****************************************************************
Main                                    ' Lance la boucle principale.
 TV=0                                   ' Réinitialise l'afficheur.
 if PEN=0 then Main                     ' Si le poussoir «ENTER» retourne à «Main».
Ripeti                                  '
 if Delay<5 then Delay=Delay+1          ' Augmente Delay.
 TV=TV+1                                ' Augmente la vitesse de rafraîchissement de l'afficheur.
 if TV=5 then                           ' Si vitesse de rafraîchissement de l'afficheur = 5
  TV=0                                  ' Met la vitesse de rafraîchissement de l'afficheur à 0
  call Lcd_New                          ' Met à jour l'afficheur.
 endif                                  '
 if PUP=0 and Delay=5 then              ' Si UP pressé et delay=5
  if PWO < 250 then PWO=PWO+1                   ' augmente de 1 PWO
  Delay=0                               ' et met delay à 0.
 endif                                  '
 IF PDW=0 and Delay=5 then              ' Si DW pressé et delay=5
  if PWO > 50 then PWO=PWO-1            ' diminue de 1 PWO
  Delay=0                               ' et met delay à 0.
 endif                                  '
 ADCIN 0, ISV                           ' Lit le courant de l'ADC.
 PULSOUT SVR, PWO                       ' Génère l'impulsion du servomoteur
 pause 20                               ' Pause de 20 msec
 If PEN=0 then Menu_PWUP                ' Si «Enter» pressé alors change l'affichage
goto Ripeti                             ' Répète le cycle de la lecture.
'*****************************************************************
'  Mise à jour de l'affichage
'*****************************************************************
Lcd_New                                 ' Routine de mise à jour de l'affichage
 lcdout  $FE,1                          ' Efface l'afficheur.
 lcdout  $FE,2                          ' Curseur en début de ligne.
 int=PWO/100                            ' Calcule la partie entière.
 fra=PWO//100                           ' Calcule la partie décimale (reste de la division)
```

```
  lcdout "Signal=", #int, "."                    ' Affiche la partie entière sur le LCD.
  if fra<10 then lcdout "0"                      ' Affiche la partie décimale sur le LCD
  lcdout #fra, "mS"                              ' + unité "msec".
  lcdout $FE, $C0, "Courant=", #ISV, " mA"       ' Affiche le courant sur le LCD.
return                                           ' Retourne.
'****************************************************************
'  Réglage du mouvement maximal
'****************************************************************
Menu_PWUP                                        '
  lcdout $FE,1, $FE,2                            ' Prépare l'afficheur.
  int=PWUP/100                                   ' Calcule la partie entière.
  fra=PWUP//100                                  ' Calcule la partie décimale (reste de la division)
  lcdout "Max=", #int, "."                       ' Affiche la partie entière sur le LCD.
  if fra<10 then lcdout "0"                      ' Affiche la partie décimale sur le LCD.
  lcdout #fra, "mS"                              ' + unité "msec".
Push_1                                           ' Attend le relâchement du poussoir.
  pause 100                                      ' Pause de 100 msec.
  if PEN=0 then goto Push_1                      ' Répète si «Enter» est pressé.
Push_2                                           '
  button PUP,0,255,0,B0,1,incPWUP                ' UP? Va à incPWUP.
  button PDW,0,255,0,B0,1,decPWUP                ' DW? Va à decPWUP.
  Button PEN,0,255,0,B0,1,Menu_PWDW              ' ENTER? Va au Menu_PWDW.
  goto Push_2                                    ' Si aucun poussoir est pressé on répète.
incPWUP                                          ' Incrément de la limite maximale du mouvement.
  if PWUP < 250 then PWUP = PWUP + 1             '
  goto Menu_PWUP                                 ' Va au menu «Menu_PWUP».
decPWUP                                          ' Décrément de la limite maximale du mouvement.
  if PWUP > PWDW then  PWUP = PWUP - 1           '
  goto  Menu_PWUP                                ' Va au menu «Menu_PWUP».
'****************************************************************
'  Réglage du mouvement minimal
'****************************************************************
Menu_PWDW                                        '
  lcdout $FE,1, $FE,2                            ' Prépare l'afficheur.
  int=PWDW/100                                   ' Calcule la partie entière.
  fra=PWDw//100                                  ' Calcule la partie décimale (reste de la division)
  lcdout "Min=", #int, "."                       ' Affiche la partie entière sur le LCD.
  if fra<10 then lcdout "0"                      ' Affiche la partie décimale sur le LCD.
  lcdout #fra, "mS"                              ' + unité "msec".
Push_3                                           ' Attend le relâchement du poussoir.
  pause 100                                      ' Pause de 100 msec.
  if PEN=0 then goto Push_3                      ' Répète si «Enter» est pressé.
Push_4                                           '
  button PUP,0,255,0,B0,1,incPWDW                ' UP? Va à incPWDW.
  button PDW,0,255,0,B0,1,decPWDW                ' DW? Va à decPWDW.
  Button PEN,0,255,0,B0,1,Menu_Time              ' ENTER? Va au Menu_Time.
  goto Push_4                                    ' Si aucun poussoir est pressé on répète.
incPWDW                                          ' Incrément de la limite minimale du mouvement.
  if PWDW < PWUP then PWDW = PWDW + 1            '
  goto  Menu_PWDW                                ' Va au menu «Menu_PWDW».
decPWDW                                          ' Décrément de la limite minimale du mouvement.
  if PWDW > 50 then  PWDW = PWDW - 1             '
  goto  Menu_PWDW                                ' Va au menu «Menu_PWDW».
'****************************************************************
'  Temps de réglage
'****************************************************************
Menu_Time                                        '
  lcdout $FE,1, $FE,2                            ' Prépare l'afficheur.
```

```
 if Speed=0 then lcdout "Slow"                    ' Test Lent.
 if Speed=1 then lcdout "Fast"                    ' Test Rapide.
Push_5                                            ' Attend le relâchement du poussoir.
 pause  100                                       ' Pause de 100 msec.
 if PEN=0 then goto Push_5                        ' Répète si «Enter» est pressé.
Push_6                                            '
 if PUP=0 then                                    ' Si UP pressé
  Speed=1                                         ' alors test Rapide
  goto Menu_Time                                  ' et retourne à «Menu_Time».
 endif                                            '
 IF PDW=0 then                                    ' Si DW pressé
 Speed=0                                          ' alors test Lent
  goto Menu_Time                                  ' et retourne à «Menu_Time».
 endif                                            '
 if PEN=0 then Run_Servo                          ' Si Enter pressé démarre le servomoteur.
 goto Push_6                                      ' et retourne à «Push_6»
'*****************************************************************
'  Execution du test
'*****************************************************************
Run_Servo                                         '
 lcdout  $FE,1, $FE,2                             ' Prépare l'afficheur.
 lcdout "Run"                                     ' Affiche "Run".
Push_7                                            ' Attend le relâchement du poussoir.
 pause  100                                       ' Pause de 100 msec.
 if PEN=0 then goto Push_7                        ' Répète si «Enter» est pressé.
Speed0                                            ' Test Lent.
 ISVM=0                                           ' Réinitialise le courant maximal.
 if Speed=0 then                                  ' Test Lent?
  for i=PWDW to PWUP                              ' Cycle d'essais de PWDW à PWUP.
   pulsout SVR, i                                 ' Génère l'impulsion pour le servomoteur.
   pause 18                                       ' Pause de 18 msec.
   ADCIN 0, TMP                                   ' Mesure le courant
   ISVM=ISVM+TMP                                  ' Additionne pour le calcul de la moyenne.
   if PEN=0 then Main                             ' Enter pressé alors fin de test.
   PWO=I                                          ' Lit la largeur de l'impulsion.
   CALL Lcd_New                                   ' Met à jour l'afficheur.
  next i                                          ' Répète la boucle FOR~NEXT
  ISV=ISVM/(PWUP-PWDW)                            ' Calcule la moyenne du courant.
  call Lcd_New                                    ' Met à jour l'afficheur.
  ISVM=0                                          ' Remet à 0 la moyenne du courant.
  for i=PWUP to PWDW  step -1                     ' Cycle d'essais de PWUP à PWDW.
   pulsout SVR, i                                 ' Génère l'impulsion pour le servomoteur.
   pause 18                                       ' Pause 18 msec.
   ADCIN 0, TMP                                   ' Mesure le courant.
   ISVM=ISVM+TMP                                  ' Additionne pour le calcul de la moyenne.
   if PEN=0 then Main                             ' Enter pressé alors fin de test.
   PWO=I                                          ' Lit la largeur de l'impulsion.
   CALL Lcd_New                                   ' Met à jour l'afficheur.
  next i                                          ' Répète la boucle FOR~NEXT
  ISV=ISVM/(PWUP-PWDW)                            ' Calcule la moyenne du courant.
  call Lcd_New                                    ' Met à jour l'afficheur.
  goto Speed0                                     ' Remet à 0.
 endif                                            ' Fin du test Lent.
Speed1                                            ' Test Rapide.
 TV=0                                             ' Réinitialise la variable de service.
 ISVM=0                                           ' Réinitialise le courant maximal.
 repeat                                           ' Lance le test minimal.
  TV=TV+1                                         ' Incrémente TV.
```

```
   pulsout SVR, PWDW          ' Génère l'impulsion minimale.
   pause 18                   ' Pause di 18 msec.
   ADCIN 0, TMP               ' Mesure le courant.
   ISVM=ISVM+TMP              ' Additionne pour le calcul de la moyenne.
   if pen=0 then Main         ' Enter pressé alors fin de test.
  until TV>100                ' 100 répétitions.
  ISV=ISVM/100                ' Calcule la moyenne du courant.
  PWO=PWDW                    ' Lit la largeur de l'impulsion.
  call Lcd_New                ' Met à jour l'afficheur.
  TV=0                        ' Remet à 0 TV.
  repeat                      ' Lance le test maximal.
   TV=TV+1                    ' Incrémente TV.
   PULSOUT SVR, PWUP          ' Génère l'impulsion minimale.
   pause 18                   ' Pause 18 msec.
   ADCIN 0, TMP               ' Mesure le courant.
   ISVM=ISVM+TMP              ' Additionne pour le calcul de la moyenne.
  until TV>100                ' 100 répétitions
  ISV=ISVM/100                ' Calcule la moyenne du courant.
  PWO=PWUP                    ' Lit la largeur de l'impulsion.
  call Lcd_New                ' Met à jour l'afficheur.
  goto Speed1                 ' Répète le test.
end                           ' Fin du programme.
```

affiché correspond au courant de « repos » (voir la figure 3). En appuyant sur les touches « P+ » et « P- » (référez-vous toujours sur la figure 1), vous pouvez modifier manuellement la durée de l'impulsion sur une plage allant de **0,5 ms** à **2,5 ms**. La plage prévue (0,5 ms à 2,5 ms) est la maximale utilisable, et certains servomoteurs pourraient atteindre leur positionnement en buté avant celle limite.

En effet, notre testeur permet d'évaluer l'amplitude de la rotation de tout type de servomoteur. Le positionnement manuel est pratique pour les vérifications préalables, mais pour rendre notre dispositif plus complet, nous avons voulu donner la possibilité de programmer des « excursions » automatiques en agissant opportunément sur les boutons poussoirs.

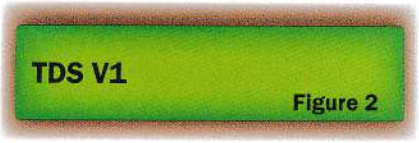


Figure 2 : à la mise sous tension, l'écran affiche le message « TDS V1 ».

Supposons que nous voulions contrôler un servomoteur dont la plage maximale est obtenue pour des impulsions de commande dont la durée minimale est de 0,6 ms et la durée maximale de 2,30 ms. Pour cela, nous effectuons le test automatiquement en appuyant sur la touche « Enter », l'écran de la figure 4 apparaît et affiche la durée maximale de l'impulsion de sortie.

Cette valeur peut être modifiée à l'aide des touches « **P+** » et « **P-** ». Nous réglons la valeur à 2,28 ms, ce qui correspond presque à notre limite. En appuyant de nouveau sur la touche « **Enter** », nous confirmons le réglage du paramètre affiché, ensuite un second affichage « demande » la valeur minimale de l'impulsion de sortie (voir la figure 5).

Cette valeur doit toujours être inférieure à la valeur maximale, appuyons de nouveau sur les touches « **P+** » et « **P-** », nous réglons notre limite inférieure à 0,62 ms. Lorsque nous appuyons sur la touche « **Enter** » une nouvelle fois, la valeur minimale est mémorisée.

Figure 3 : visualisation de la durée de l'impulsion de commande en sortie et de la valeur du courant consommé par le servomoteur.

A ce stade, un nouvel écran apparaît (voir la figure 6) dans lequel est demandé si le mouvement du servomoteur doit être lent (touche « P- ») ou rapide (touche « P+ »).

Figure 4 : exécution du test automatique, l'écran affiche la durée maximale de l'impulsion de sortie.

Figure 5 : l'afficheur « demande » la valeur minimale de l'impulsion de sortie.

Supposons que nous choisissions le **test lent**, il vérifie alors toutes les **positions du servomoteur comprises** entre les valeurs de l'**impulsion minimale** et **maximale**.

En appuyant de nouveau sur la touche « **Enter** », nous lançons le **test automatique** avec les paramètres définis. L'écran affiche le message « **Run** » qui au bout d'un moment sera remplacé

par les paramètres de fonctionnement (voir la figure 7).

Le **test se termine par une nouvelle pression** sur la touche « **Enter** ».

Après avoir choisi le **test lent** (« **Slow** »), le servomoteur commence à se déplacer lentement et effectue un va et vient entre les deux positions extrêmes, tandis que l'écran LCD affiche la valeur de l'impulsion de sortie en temps réel et le courant moyen consommé par le servomoteur à chaque changement de direction.

Cette fonction permet de **vérifier le fonctionnement « à vide »** et « **en charge** ». Lorsque le servomoteur est essayé au « banc d'essai », le test permet de déterminer si les engrenages fonctionnent correctement et si les mouvements sont fluides et précis.

Toutefois, lorsque le servomoteur est installé dans une maquette, le même test permet de vérifier si la course est optimale et correcte ou si, en raison de problèmes mécaniques sur des leviers, des points de surcharge sont présents.

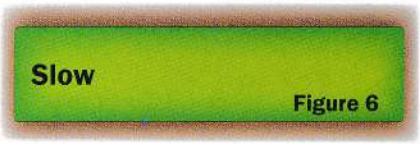


Figure 6 : le mouvement lent (« Slow ») est sélectionné.

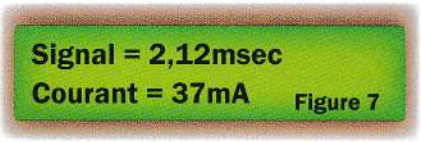


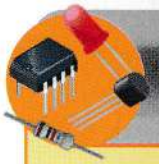
Figure 7 : affichage des paramètres de fonctionnement du test automatique.

En choisissant le **test rapide** (« **Fast** »), l'écran affiche toujours la valeur de l'impulsion de sortie en temps réel et le courant moyen consommé par le servomoteur, mais celui-ci oscillera entre les valeurs minimales et maximales sans valeurs intermédiaires, espacées d'un léger retard afin de vérifier la vitesse réelle de rotation de l'axe.

Lorsque le servomoteur est installé dans une maquette, au moyen de ce test il est possible de déterminer s'il est adapté à votre application.

Dans le cas où il serait trop faible, le servomoteur ne sera pas en mesure de se repositionner à la vitesse nécessaire et, par conséquent, il sera incapable de suivre le signal de commande.

Cela provoque des difficultés évidentes et des efforts considérables de la part du moteur interne, dans le milieu du modélisme, on dit que le servomoteur va **flotter** ou va « **flutter** ». ■



Comment construire ce montage

Le microcontrôleur programmé **MF741** est disponible auprès de **COMELEC**.

Le typon du circuit imprimé ainsi que le code source et le programme « .hex » sont téléchargeables sur : **www.electroniquemagazine.com** dans le sommaire détaillé de la revue numéro 128 section « Télécharger ».

TX4C9B

Émetteur 4 canaux codé sur 9 digits

Dans cet article, nous vous proposons de réaliser un émetteur pour radiocommande avec 4 canaux indépendants et codifiables individuellement, facile à mettre en œuvre et de faible coût. Ce montage utilise un microcontrôleur Microchip et met en œuvre la génération d'un code à 9 digits de type « three-state » (3 états). Il fonctionne dans la bande de fréquence 433,92 MHz.

de Roberto Prestianni

Par le passé, nous vous avons présenté un récepteur simple canal avec un codage de type **Motorola**, fonctionnant à l'aide d'un PIC. Dans la même lignée, nous vous présentons maintenant un émetteur 4 canaux simple à reproduire et de dimensions compactes. La combinaison de cet **émetteur TX** avec un **récepteur** équipé d'un module **Aurel AC-RX2**, permet de couvrir une distance de plus de **20 mètres** (à vol d'oiseau et sans aucun obstacle), ce qui est plus que suffisant pour des applications typiques domestiques comme l'ouverture à distance d'un portail ou la commande de lampes ou d'autres dispositifs dans la maison.

La réalisation de cet émetteur a émergé **de la nécessité de commander plus de 2 récepteurs ayant un codage sur 9 digits de type Motorola à 3 états et une fréquence d'horloge de 1,7 kHz**, en utilisant le **même émetteur**. En effet, la possibilité de coder individuellement chacun des 4 canaux permet de contrôler autant de récepteurs utilisant des codes Motorola complètement différents l'un de l'autre.

Beaucoup d'émetteurs à plusieurs canaux existent sur le marché, et génèrent de base un code (réglable par DIP switch ou par une procédure de codage), suivi par un ou plusieurs « digits » qui tiennent compte de la touche pressée. Par exemple, l'émetteur à deux canaux **TX2CSAW génère deux codes qui diffèrent par le neuvième « digit »**. En effet celui-ci vaut « **1** » pour le bouton de gauche et « **0** » pour le bouton de droite. De cette façon, les deux codes sont liés par un code de base personnalisable par l'utilisateur. Dans le cas de **l'émetteur décrit dans cet article, nous pouvons utiliser toutes les configurations possibles** (soit **19 683** combinaisons) obtenues avec les 9 digits du codage Motorola.

L'utilisation d'un **microcontrôleur**, en plus de fournir la possibilité d'**implémenter des codes différents** sans l'aide d'un circuit de codage spécifique (par exemple un MC145026), permet la **personnalisation de chacun d'eux** au moyen d'une **procédure de codage simplifiée**, réalisée avec les 4 mêmes touches de commande. La suppression du DIP switch permet un « **codage secret** » des différents canaux, ce qui signifie qu'en cas de vol ou de perte de l'émetteur, il sera très difficile de lire les codes qui y sont stockés, voire même impossible ! Le microcontrôleur choisi est bien connu, il s'agit d'un **PIC12F675** dans un boitier DIP plastique à 8 broches.

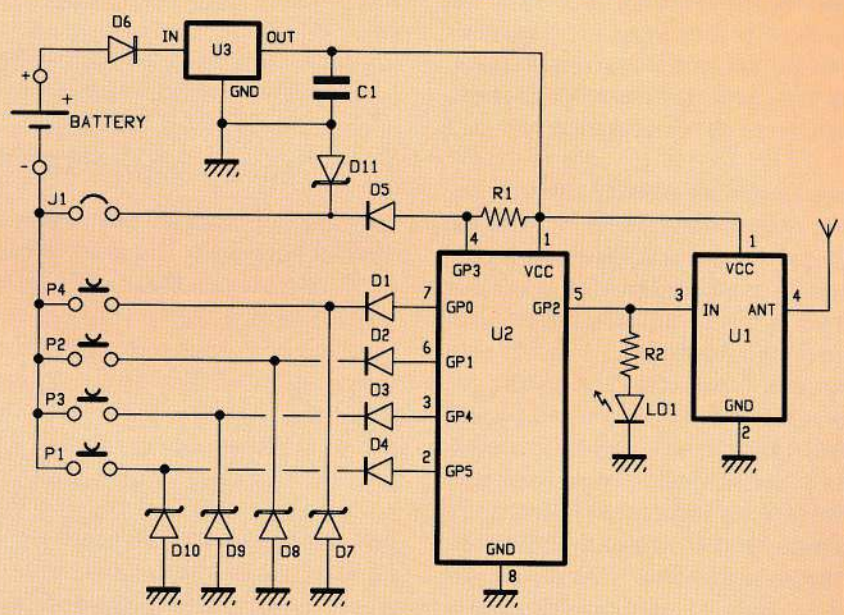
Schéma électrique

Le schéma électrique de cet émetteur est vraiment simple et basique, malgré la présence d'un réseau de diodes qui pourrait induire en erreur. Ces composants, dont le nombre est relativement élevé pour un si petit circuit, trouve sa justification dans le fait que notre émetteur doive être en mesure de fonctionner avec une pile de 12 V typique des télécommandes, condition qui requiert une économie d'énergie la plus grande possible.

Voici donc sur la figure 1 le schéma électrique de l'ensemble du circuit utilisé lorsque nous pressons l'une des 4 touches. Si vous voulez que votre pile que votre pile dure aussi longtemps que

Schéma électrique

Figure 1 : schéma électrique de l'émetteur 4 canaux codé sur 9 digits.



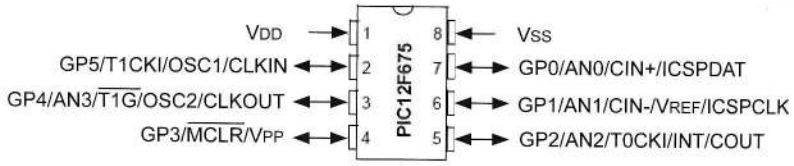
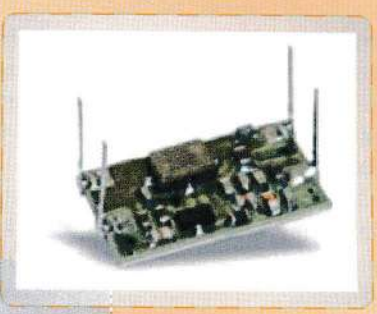


Figure 2 : brochage du PIC12F675 vu de dessus.

Module Aurel TX-4MDIL

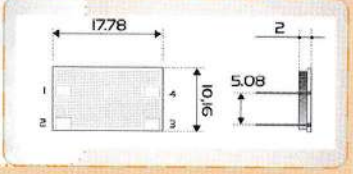
Il s'agit d'un petit émetteur (10,16 mm x 17,78 mm) de données numériques fonctionnant en **modulation « OOK »** (« **On Off Keying** » ou « **tout ou rien** ») sur une porteuse de **433.92 MHz**. Il se distingue par une très faible consommation (maximum de 6 mA sous 5 V). Il peut être alimenté avec une tension continue comprise entre 3 V et 5 V, et est capable de délivrer une puissance maximale de 2 dBm.



Caractéristiques techniques :

- **Fréquence de fonctionnement : 433.92 MHz**
- **Alimentation : 3 à 5 VDC**
- **Courant consommé : 3 à 6 mA**
- **Puissance HF de sortie : - 1 à + 2 dBm**
- **Fréquence maximale de modulation : 4 kHz**
- **Niveau logique d'entrée typique : Vcc**
- **Température de fonctionnement : - 20 °C à + 80 °C**

Brochage :
1 : + V
2 : masse
3 : entrée des données
4 : sortie HF



possible, il serait absurde d'alimenter continuellement l'émetteur ! Et ceci, pas tant pour le PIC (U2) ou le module AUREL HF (U1), mais pour le régulateur 12 V (U3) qui consomme à lui seul environ 3 mA, ce qui réduirait drastiquement l'autonomie de l'émetteur.

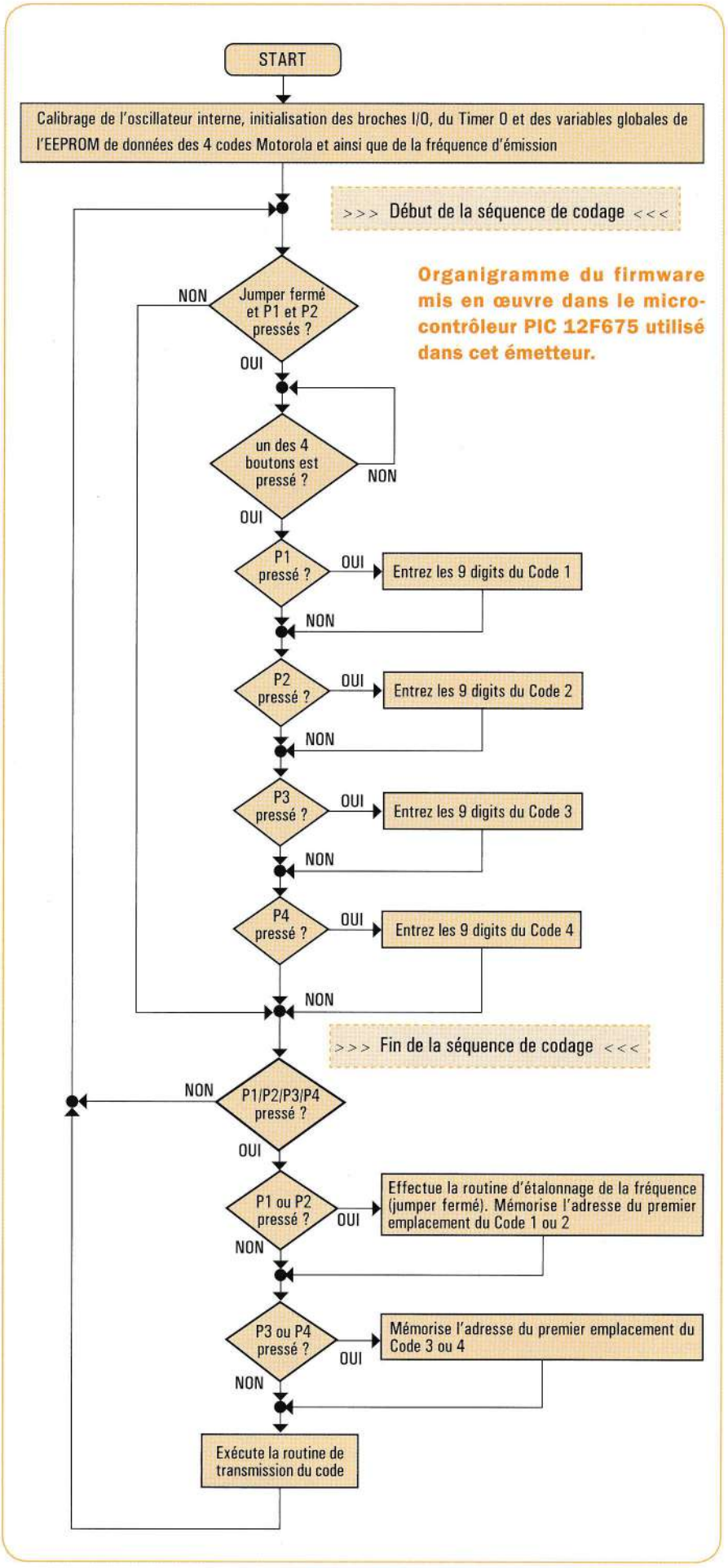
En regardant la configuration du circuit, chaque fois que nous pressons l'un des 4 boutons (ou fermons le cavalier J1) le pôle négatif de la pile est relié à la masse du circuit. Le régulateur de tension U3 est alors alimenté ainsi que l'ensemble du circuit.

Prenons par exemple le poussoir P4, une de ses extrémités est reliée au pôle négatif de la pile et l'autre aux cathodes de D1 (1N4148) et de D7 (BAT85 Schottky). Quand nous **appuyons sur P4**, la diode **D7 est polarisée directement** « engendrant» ainsi la **masse**. Nous obtenons alors un **potentiel positif** d'environ **0,4 V par rapport à la borne négative de la pile** (notez la différence avec une chute de tension classique d'une diode au silicium qui est d'environ 0,5 à 0,6 V).

Cela signifie que la tension présente à l'entrée de la **broche GP0** de **U2**, **référencée à la masse**, a une valeur égale mais de **signe opposé**, soit **- 0,4 V**. Afin que le **PIC puisse interpréter correctement le niveau logique « 0 », VIL** (niveau d'entrée bas) ne peut être négatif (il ne doit pas dépasser 0,2 Vdd). Grâce à la **présence** de **D1** (avec un seuil de 0,5 V), le potentiel de la broche **GP0** est ramené de - 0,4 V à **+ 0,1 V**. Pour chaque ligne d'entrée de **U2**, nous devons **activer les résistances internes de pull-up**.

Parallèlement à cette situation, sur les **lignes d'entrées des autres poussoirs** (P1, P2, P3, et J1) la **présence** des **diodes Schottky** (polarisées en inverse), ne permet pas au potentiel de masse d'atteindre ces lignes qui, par conséquent, seront **maintenues à un niveau élevé** en raison des résistances de **pull-up internes**.

Le reste du schéma montre la présence de **R1** qui fait office de **résistance** de **pull-up** pour la ligne **GP3** (elle est dépourvue de résistance interne de pull-up), la broche **GP2** est **configurée en sortie** et permet l'allumage de la



Organigramme du firmware mis en œuvre dans le microcontrôleur PIC 12F675 utilisé dans cet émetteur.

LED **LD1** ainsi que la **transmission de la trame de codage au module AUREL**. Le signal est donc appliqué à la **broche 3** du module **AUREL TX-4MDIL** qui correspond à l'entrée des « données ». L'émission de la trame est réalisée en **modulation** « **OOK** » (« **On Off Keying** ») avec une **porteuse** centrée sur **433,92 MHz**.

Le **PIC12F675** doit être alimenté avec **5 V** et cette valeur est obtenue par **U1**, un régulateur classique **78L05** en boitier plastique **TO92**. L'alimentation est complétée par la **diode de protection D6** qui agit en cas d'**inversion de polarité**, le condensateur **C1 filtre la tension** des éventuels parasites.

Etant donné que la source d'alimentation est une **pile de 12 V, il n'est pas nécessaire dans ce cas d'utiliser des condensateurs électrolytiques de filtrage** pour les tensions **12 V** et **5 V**.

Comme vous l'aurez certainement constaté, notre schéma ne comporte pas de réseau RC, ni de quartz, pour la génération du signal d'horloge du microcontrôleur. En effet, ce dernier est configuré pour fonctionner avec l'**oscillateur interne**, ceci permet d'utiliser les **broches 2** et **3** (OSC1 et OSC2) comme les **entrées /sorties numériques GP4** et **GP5**. En outre, grâce à l'utilisation de la **réinitialisation interne**, la ligne **MCLR**, normalement utilisée pour le « **RESET** » externe, **est utilisée ici comme une ligne d'entrée/sortie** (**GP3**). Sur la figure 2, vous trouverez le brochage du PIC et dans l'encadré celui du module AUREL TX-4MDIL.

Le firmware

Le **firmware** utilisé dans le **PIC12F675** se nomme « **TX4CH.hex** », vous pouvez le **télécharger** sur notre site **www.electroniquemagazine.com**.

Nous allons publier dans l'article, une **partie du code source** afin d'expliquer certaines routines. Cependant sur notre site nous proposerons à la vente à l'onglet « **Top projets** », le projet complet contenant les fichiers sources en **langage C** ainsi que le compilateur en version lite.

Le **programme** (firmware) a été développé dans l'environnement **MICROCHIP MPLAB** et dans un langage **ANSI C**. Le compilateur utilisé est le **PICC 9.60** de chez **Hitech Software** et le projet se compose de deux fichiers. L'un appelé « **TX_Motorola_4CH.c** » et l'autre « **pic12f6x.h** » qui correspond au fichier de la librairie.

Ce dernier **définit les noms des différents modules et les registres de configuration du PIC utilisés**, tandis que le fichier en « **.c** » contient le **programme principal** « **main** » et toutes les routines comme « **transmit_code()** » pour la transmission du **code Motorola**, « **code_TX()** » pour la génération d'un digit unique (« 1 », « 0 », « open » (ouvert)), « **lamp_led()** » et « **more_lamp()** » sont des routines pour **allumer** une ou plusieurs fois la LED, « **set_freq()** » permet la **modification de la fréquence de transmission des codes**.

Enfin, nous avons les **routines de temporisation**, « **halfperiod()** » pour le **taux** (cadence) d'**émission des bits**, « **DelayMs(n)** » et « **Delay100Us(x)** » respectivement pour les temporisations de « **n** » millisecondes et de « **x-multiples** » de **100 microsecondes**. Le **code source** commence par l'**inclusion** des fichiers « **header** » (**.h**), la **configuration** des **bits**, la **définition** de certaines **constantes** et l'**initialisation** de **4** blocs de **2 octets** (byte) de l'**EEPROM**.

Lors de la **programmation du PIC**, nous définirons les **4 codes Motorola** par défaut, et le paramètre « **delay_timer0_def** » définira la **fréquence par défaut** de l'horloge utilisée (environ 1,7 kHz).

En ce qui concerne les bits de configuration, nous devons tenir compte des points suivants :

- « **_CONFIG : INTIO** » indique au compilateur le choix de l'oscillateur interne pour la génération du signal d'horloge (qui libère automatiquement les lignes d'E/S GP4 et GP5) ;

- « **_CONFIG : PWRTE** » ou « **Power-Up Timer Enable** » permet d'activer le **registre** de « **RESET** » intégré dans le PIC ;

- « **_CONFIG : MCLRDIS** » désactive le « **RESET** » externe de la broche **MCLR** et permet de libérer une autre ligne d'E/S (GP3).

Ensuite nous avons les déclarations des **variables globales** (y compris le tableau à 4 dimensions de 9 éléments pour la mémorisation des 4 codes), la **définition des noms mnémoniques relatifs aux 6 lignes d'entrées/sorties** de **U2** et les fonctions mentionnées ci-dessus. Poursuivons l'analyse du firmware en observant l'organigramme figurant dans les pages précédentes et qui décrit le fonctionnement général du microcontrôleur.

En appuyant sur l'une des 4 touches ou en fermant le jumper, le circuit est alimenté et l'émetteur est en fonction. Le microcontrôleur exécute alors le programme. Il commence par le **calibrage** de l'**oscillateur interne**, qui est mis en œuvre par le **registre** « **OSCCAL** » en chargeant les données préenregistrées par Microchip.

Ensuite sont configurés les ports d'entrées/sorties à l'aide des **registres** « **TRIS** » ainsi que le registre « **TIMER0** ».

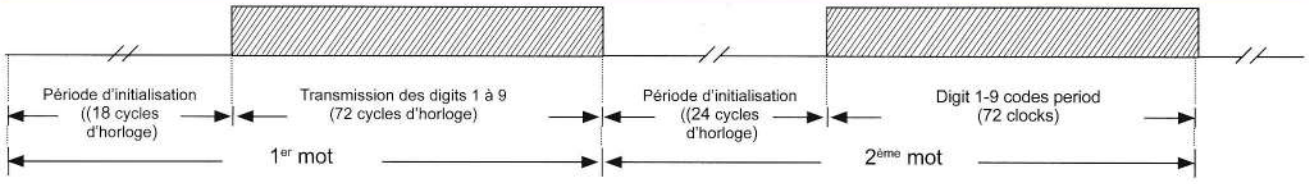


Figure 3 : chronogramme de la transmission du code 9 digits 3 états de Motorola.

Après l'initialisation des variables globales, **4 tableaux sont chargés contenant les 4 codes Motorola précédemment mémorisés dans l'EEPROM**, et la fréquence d'émission des bits est définie.

À ce stade, nous pouvons **commencer le codage des canaux, après avoir fermé le jumper et activé les touches P1 et P2 pendant une durée d'environ 2 secondes**. En appuyant sur l'**une des 4 touches**, nous **communiquons au PIC** le canal que nous souhaitons encoder et, à ce moment, commence la séquence d'**émission des 9 digits du code Motorola** et sa **mémorisation** dans la mémoire **EEPROM**. Après cette procédure, il vérifie l'état des 4 boutons pour générer le code qui lui est associé.

P1 et **P2** sont utilisés pour l'**exécution de la routine de calibrage de la fréquence d'émission des bits**, alors que **P3** et **P4** sont utilisés pour **augmenter**, **diminuer** ou **revenir à la valeur par défaut** de cette fréquence.

Dans le cas où l'**un des boutons est pressé**, une **variable temporaire est mémorisée** dans laquelle se trouve l'**adresse du tableau contenant le code souhaité** et, peu de temps après, au moyen de la routine « **transmit_code()** » la génération du code vers le module HF **U1** est effectuée.

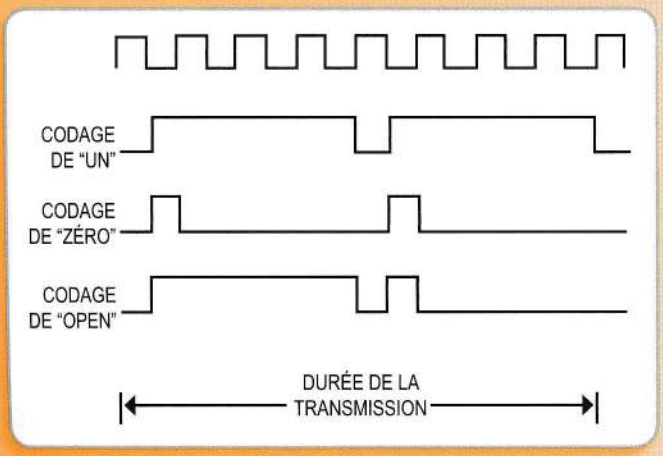
Le **Listing 1** montre le mode opératoire de la fonction « **transmit_code ()** » chaque fois qu'elle est appelée pour générer la séquence binaire du code désiré. Elle reproduit fidèlement les diagrammes temporels des codeurs de type **Motorola** à **9 digits / 3 états**. Dans un premier temps, le **signal est maintenu à un niveau bas** pendant une durée de **18 cycles d'horloge**, suivi par la génération du **1er mot contenant les 9 digits**. Ensuite une autre pause (niveau bas) pendant **24 cycles d'horloge** supplémentaires a lieu, représentant le **signal de synchronisation**, et enfin le **2eme mot identique au premier est envoyé**. Ces deux mots sont générés en utilisant une **boucle « for () »** qui, cycle après cycle, **appelle la routine « code_TX »** pour l'**émission de la séquence d'un seul bit à la fois**, et le **pointage vers le bit suivant**.

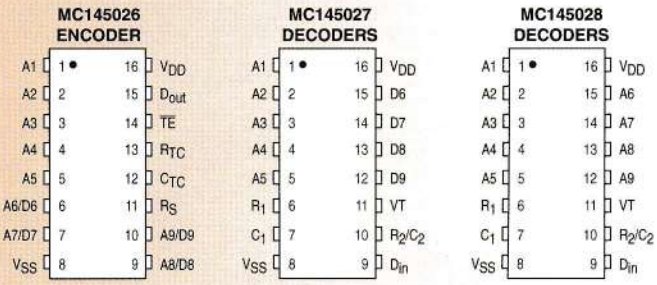
Le **Listing 2** montre le fonctionnement de la **routine « code_TX (m) »** qui supporte la **routine « transmit_code () »**. Pour mieux comprendre comment cette partie du firmware fonctionne, reportez-vous à l'encadré intitulé « **Le codage 3 états Motorola** ». Cette fonction reçoit de la routine, en appelant un paramètre, le digit qui doit être

Le codage 3 états Motorola

Le **codage 3 états Motorola** est une séquence de **9 digits consécutifs**, chacun représentant un **état logique particulier** de l'une des 9 lignes du codage : « **1** », « **0** », « **Open** ». A chacun de ces 3 états est associée une **séquence binaire particulière**, contenant un **signal de synchronisation**. Ce signal, dans notre cas, a une fréquence de **1,7 kHz**. En réalité les séquences durent T/2 = 294 µS (où T est la période de cette onde). Ensuite, cette période correspondra à la cadence d'émission du code Motorola. Dans le diagramme ci-contre, les 3 séquences sont représentées. Imaginez que nous divisions chacune de ces séquences par des durées de T/2 et en associant un statut binaire. Nous obtenons ainsi une **séquence précise de 16 bits** pour simplifier, que nous divisons en **2 octets** :

"UN" : digit=$(11111110)_2$=254 digit=$(11111110)_2$=254
"ZERO" : digit=$(10000000)_2$=128 digit=$(10000000)_2$=128
"OPEN" : digit=$(11111110)_2$=254 digit=$(10000000)_2$=128





Voici donc l'émission de l'un des 9 digits d'un code Motorola, qui est équivalent à une transmission bit par bit d'un octet. Pour cela, nous « chargeons » simplement les digits dans un registre, puis nous les faisons « glisser » dans une direction pour les transmettre tour à tour sous forme de bits en sortie.

Cette opération est évidemment **répétée 2 fois** pour **chaque digit**, par exemple pour transmettre « OPEN », nous devons charger la valeur 254 et la transmettre, ensuite charger la valeur 128 et la transmettre. La totalité de la séquence de 16 bits doit être répétée 9 fois, car il y a autant de digits constituant un « mot » (ou « word »). Celui-ci doit être délivré une ou plusieurs fois pour s'assurer que le récepteur reconnaisse la séquence. Pour plus d'informations sur le codage 3 états Motorola nous vous recommandons de télécharger sur le web la documentation technique des circuits intégrés MC145026/27/28.

Listing 1

Pour générer une chaîne de commandes tout à fait semblable à un code binaire 3 états de type Motorola, nous utilisons la routine « transmit_code () » que vous pouvez voir ci-dessous.

```
//****************** Fonction pour la transmission d'un Code Motorola*****************************
void transmit_code()
{
                                                //NOTE : le premier bit à émettre est le MSB
                                                //Impulsion de réveil  du module TX
        RF_out = 0;                             //RESET du signal modulé


//********************** PAUSE 18T ***************************************************************

        for (counter=0;counter<18;counter++)
        {
                                                // Retard T
                halfperiod();                   // Retard 1/2 T
                halfperiod();                   // Retard 1/2 T
        }


//********** Transmission du 1er mot (9 bits) ****************************************************

        for (index=0;index<9;index++)
        {
                code_TX(*EEADD_temp);           //Transmission des bits
                ++EEADD_temp;                   //Sélectionne le bit suivant
        }
        EEADD_temp -= 9;                        //Pointe le 1er bit



//********************** PAUSE 24T ***************************************************************
*****

        for (counter=0;counter<24;counter++)
        {
                                                // Retard T
                halfperiod();                   // Retard 1/2 T
                halfperiod();                   // Retard 1/2 T
        }


//********** Transmission du 2eme mot (9 bits) ****************************************************

        for (index=0;index<9;index++)
        {
                code_TX(*EEADD_temp);           //Transmission des bits
                ++EEADD_temp;                   //Sélectionne le bit suivant
        }
        EEADD_temp -= 9;                        //Pointe le 1er bit
}
```

Listing 2

Ci-dessous le fonctionnement de la routine « code_TX(m) » impliquée dans la routine « transmit_code () ». Cette fonction reçoit le paramètre indiquant le bit à transmettre (1, 0, Open), et sur cette base au moyen de l'instruction C switch(), génère la séquence.

```
//************************** Fonction générant le code Motorola '1', '0', 'OPEN' **********************

void code_TX(unsigned char m)

//Note: le paramètre 'm' représente le bit à émettre

{
        switch(m)
        {
                // "1" Émission du double octet 'ONE' (65278)

                case 1 :
                        digit = 254;
                        for (counter=0;counter<=7;counter++)
                        {
                                asm("rlf _digit");              //Rotation du registre
                                RF_out = CARRY;                 //Emission du bit
                                halfperiod();                   //Retard T/2
                        }
                        digit = 254;
                        for (counter=8;counter<=15;counter++)
                        {
                                asm("rlf _digit");              //Rotation du registre
                                RF_out = CARRY;                 //Emission du bit
                                halfperiod();                   //Retard T/2
                        }
                        break;

                // "0" Émission du double octet 'ZERO' (32896)

                case 0 :
                        digit = 128;
                        for (counter=0;counter<=7;counter++)
                        {
                                asm("rlf _digit");              //Rotation du registre
                                RF_out = CARRY;                 //Emission du bit
                                halfperiod();                   //Retard T/2
                        }
                        digit = 128;
                        for (counter=8;counter<=15;counter++)
                        {
                                asm("rlf _digit");              //Rotation du registre
                                RF_out = CARRY;                 //Emission du bit
                                halfperiod();                   //Retard T/2
                        }
                        break;

                // "OPEN" Émission du double octet 'OPEN' (65152)

                case 2 :
                        digit = 254;
                        for (counter=0;counter<=7;counter++)
```

```
                    {
                        asm("rlf _digit");          //Rotation du registre
                        RF_out = CARRY;             //Emission du bit
                        halfperiod();               //Retard T/2
                    }
                    digit = 128;
                    for (counter=8;counter<=15;counter++)
                    {
                        asm("rlf _digit");          //Rotation du registre
                        RF_out = CARRY;             //Emission du bit
                        halfperiod();               //Retard T/2
                    }
                    break;
            default:
                    break;
        }
        return;
}
```

transmis (1, 0, Open), et est basée sur l'utilisation de l'**instruction C « switch() »**, qui permet la **génération d'un bit et du suivant**. En résumé, les **2 octets sont associés à un « digit »** et sont **émis bit par bit** à l'aide d'une **rotation vers la gauche du registre « asm ("rlf_digit") »**, les bits résultants sont envoyés vers le module TX.

Réalisation pratique

Le montage est très simple et ne présente aucune difficulté pour sa réalisation. Tout d'abord, vous devez fabriquer le circuit imprimé soit par un procédé de gravure chimique ou à l'aide de la 3DRAG si vous en avez une. Pour cela téléchargez sur notre site **www.electroniquemagazine.com** dans le **sommaire détaillé du numéro 128** à l'onglet « **Télécharger** » le typon du circuit imprimé à l'échelle 1 : 1. Une fois le circuit gravé, nous vous recommandons d'utiliser un foret de 0,9 mm pour les trous de la pile, des 4 boutons et de JP1, et un foret de 0,8 mm pour le reste des composants.

Commencez par souder les composants ayant un profil bas, c'est-à-dire les résistances, les 11 diodes (D1 à D11) en faisant attention à l'orientation de la bague. Ensuite continuez avec le support de U1, les boutons, le jumper, la LED et les supports de la pile, le régulateur et enfin le condensateur C1.

Terminez par le module AUREL en prêtant une attention particulière à son orientation (vérifiez le brochage dans l'encadré intitulé « Module Aurel TX-4MDIL»).

Maintenant, nous allons fabriquer l'antenne. Procurez-vous un morceau de fil de 17cm de long semi-rigide isolé par une gaine. A l'aide d'une pince de précision, disposez le fil de manière à former des spires rectangulaires espacées du circuit imprimé de quelques millimètres. Nous vous conseillons d'enrouler le fil de sorte que les 4 boutons passent à l'intérieur des spires (et non le module U1). Le câblage étant terminé, vérifiez le travail effectué ainsi que les soudures. A ce stade, nous pouvons faire un premier essai. Insérez la pile de 12V en respectant la polarité et fermez le jumper (cavalier). A l'aide d'un multimètre, vous devez mesurer 5 V entre les broches 1 et 8 du support du PIC. Si c'est le cas, retirez le cavalier et montez le PIC en vérifiant que le repère détrompeur en forme de rond (qui indique la broche 1 du PIC) soit du côté de R1.

Par défaut les 4 codes ont tous les 9 digits dans l'état « OPEN » (ouvert). Si vous avez un récepteur avec un codage de type Motorola, configuré de la même manière, vous pouvez immédiatement vérifier le fonctionnement de l'émetteur. Si vous disposez d'un récepteur monocanal, il suffit tout simplement de l'alimenter.

Dès que vous appuyez sur l'un des 4 boutons de l'émetteur, la LED du récepteur clignotera pour confirmer l'apprentissage correct du code Motorola généré par l'émetteur.

Nous concluons le montage de notre émetteur en installant les 4 capuchons des micro-interrupteurs sans appuyer trop fermement. Le circuit est placé dans un boîtier en plastique pour télécommandes, disponible dans le commerce.

Programmation et utilisation

Maintenant nous allons décrire le fonctionnement de notre émetteur à 4 canaux en commençant par la programmation du code pour chaque canal. La **modification des codes** ou de la **fréquence d'émission prend effet après la fermeture du cavalier**,

Plan de montage

Les dimensions du circuit et la position des 4 boutons sont parfaitement adaptées pour un boitier en plastique de type Teko.

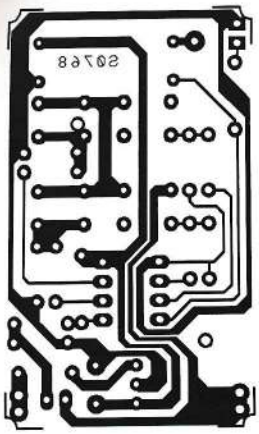


Figure 4 : dessin à l'échelle 1 : 1 du circuit imprimé simple face de l'émetteur 4 canaux.

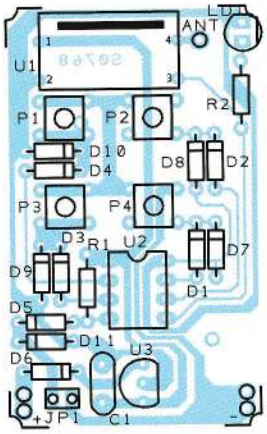


Figure 5 : schéma de câblage de l'émetteur 4 canaux.

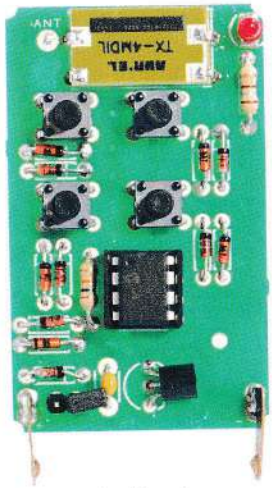


Figure 6 : photo de l'un de nos prototypes de l'émetteur 4 canaux.

Liste des composants de l' émetteur 4 canaux

R1.....10 kΩ ¼ W
R2.....180 Ω ¼ W
C1.....100 nF multicouche
LD1....LED rouge 3 mm
D1.....1N418
D2.....1N418
D3.....1N418
D4.....1N418
D5.....1N418
D6.....1N418
D7.....BAT85
D8.....BAT85
D9.....BAT85
D10....BAT85
D11....BAT85
U1.....TX-4M-DIL
U2.....PIC12F675
U3.....78L05
P1.....microswitch
P2.....microswitch
P3.....microswitch
P4.....microswitch

Divers

Support ci 2 x 4 broches
Pile 12 V pour télécommande
Connecteur ci pour pile 12 V
Barrette mâle 2 pôles
Cavalier (jumper)
Boitier Teko 11124.4

l'événement est immédiatement suivi par **5 clignotements** de la LED.

Nous vous rappelons que lorsque le cavalier est présent (J1 fermé), le circuit est alimenté de manière continue, par conséquent une fois que vous avez terminé les réglages vous devez enlever le cavalier sous peine de décharger la pile rapidement.

La procédure est très simple, supposons que vous voulez définir le premier code correspondant au bouton P1, vous devez procéder comme suit :

1. appuyez simultanément sur P1 et P2 pendant environ 2 secondes ; la LED du TX se met à clignoter rapidement ;

2. relâchez P1 et P2 et appuyez sur P1 (ou P2/P3/P4 pour les autres canaux) afin de sélectionner le canal 1 que vous désirez coder, vous voyez clignoter la led 4 fois rapidement ;

3. maintenant entrer un à la fois les 9 digits qui composent le code Motorola et utilisez la séquence suivante : P1 = « 1 », P2 = « 0 », P3 = « Open » (P4 non utilisé). Pour chaque digit inséré, vous avez la confirmation par un seul clignotement de la LED ;

4. après que vous ayez entré le dernier digit du codage, la séquence se terminera et la LED se mettra à clignoter rapidement comme à l'ouverture.

Une fois que vous avez effectué le paramétrage du premier code, **avec la même procédure vous pouvez programmer les 3 autres codes** ou **terminer la procédure en enlevant le cavalier** (selon vos besoins).

Comme nous l'avons mentionné précédemment, il existe **une procédure qui permet de modifier la fréquence d'émission des codes**. Cette possibilité a été prévue pour deux raisons, d'abord puisque le **PIC fonctionne avec son oscillateur interne**, il existe une probabilité minime que le récepteur ne fonctionne pas correctement à la fréquence d'émission généré par l'émetteur (dû à la tolérance des composants).

En outre, étant donné que nous sommes en mesure de modifier cette fréquence d'un minimum de 1 kHz à un maximum de 3 kHz (à peu près), nous avons la possibilité **de faire fonctionner notre télécommande à**

Le fil d'antenne

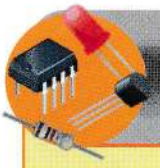
Pour fonctionner correctement, le module émetteur **Aurel TX-4MDIL** utilisé dans notre projet doit être équipé d'une antenne adéquate (c'est-à-dire ayant une longueur correspondant à la fréquence de travail). Dans notre cas, nous avons utilisé un morceau de fil (du câble électrique normal) de **17 centimètres** de longueur, égale à **1/4 de la longueur d'onde de la fréquence** de travail (433,92 MHz correspond à environ 70 cm). En divisant par 4, nous obtenons exactement 17 cm. Comme vous pouvez le constater dans l'image ci-contre, le fil est enroulé à l'intérieur du boitier. Cette solution permet une portée, avec un récepteur standard, d'au moins 20 mètres.

une fréquence différente de la norme (1,7 kHz), **ce qui rendra difficile son piratage**.

Pour cela, vous devrez procéder de la manière suivante : commencez par **fermer le cavalier** (si vous ne l'avez pas déjà fait) et, immédiatement après, **appuyez** sur **P1** ou **P2** (et seulement ceux-ci), puis appuyez sur **P3 pour augmenter la fréquence** ou **P4 pour la diminuer**. Une fois que vous arrivez aux extrémités de la plage de fréquences, vous verrez clignoter la LED rapidement.

Cependant **vous pouvez revenir à tout moment à la valeur par défaut** soit 1,7 kHz, en **maintenant enfoncés P3 et P4** et en **appuyant immédiatement sur P1** ou **P2**. La **restauration de la fréquence** par défaut est confirmée par **5 clignotements lents** de la LED.

Veuillez noter que lors de la procédure de modification de la fréquence d'horloge décrite ci-dessus, le transmetteur génère la séquence relative aux canaux 1 ou 2, étant donné que dans cette phase les boutons P1 ou P2 sont pressés, cela vous permet de tester la nouvelle fréquence d'horloge. ■



Comment construire ce montage

Tous les composants sont disponibles auprès de COMELEC.

Le typon du circuit imprimé ainsi que le programme sont téléchargeables gratuitement sur : **www.electroniquemagazine.com** dans le sommaire détaillé de la revue numéro 128 section « télécharger ».

Timer programmable de 1 seconde à 60 heures

Ce montage est un temporisateur universel dont la sortie sur relais est activée soit de manière cyclique, soit à l'aide d'une simple impulsion avec une durée d'intervalle de temps ON et OFF réglable de 1 seconde à 60 heures.

De FRANCESCO DONI

Lorsque nous devons commander un dispositif électrique sur une longue période, nous pouvons utiliser deux types de temporisation : la « **one-shot** » (un seul coup) et la cyclique. La différence entre les deux réside dans le fait que la première active le dispositif pendant un laps de temps puis le met au repos, tandis que la seconde peut périodiquement répéter la séquence d'activation, ou commuter le dispositif pour une période déterminée avec une pause de manière bien définie. Le tout à l'infini, ou bien de l'allumage à l'extinction du dispositif dès la réception d'une commande d'enclenchement de la séquence.

Nous avons souvent décrit des montages de temporisateurs de toutes sortes, mais la quasi-totalité d'entre eux étaient soit du type à simple impulsion (**one-shot**) soit de simples temporisateurs, soit des dispositifs capables d'activer une charge avec une pause, à l'aide d'un niveau de tension ou d'un bouton.

Avec ce montage nous vous proposons de réaliser un temporisateur bimodal, c'est-à-dire capable de travailler de deux manières différentes. La première commande un dispositif pendant une certaine période, ou immédiatement après une pause, tandis que la deuxième active cycliquement le dispositif.

La commande est réalisée par l'intermédiaire d'un relais inclus dans le circuit, qui active ou désactive la charge, en fonction de la connexion choisie et de la configuration du circuit.

Par conséquent, nous disposons d'un temporisateur universel avec une large gamme de réglages. Pour **chaque mode, nous pouvons définir les intervalles de temps d'activation et de pause sur une période de 1 seconde à 60 heures**. Le réglage se fait très facilement au moyen de deux trimmers (RV1 est utilisé pour définir la durée de la pause et le RV2 la durée d'activation) et de deux ensembles de cavaliers. Le montage est opérationnel dès qu'il est alimenté et ne nécessite donc pas de contrôle externe.

En mode « **one-shot** », vous pouvez opter pour deux fonctions : la première active le dispositif et une fois l'intervalle de temps atteint désactive le dispositif, la seconde c'est l'inverse elle désactive le dispositif et une fois l'intervalle de temps atteint l'active.

En **mode cyclique**, le temporisateur active et désactive périodiquement le relais de sortie en fonction de la séquence définie. Dans ce mode, vous pouvez déterminer si la séquence doit au début activer le relais ou doit

le laisser au repos. Les modes de fonctionnement sont définis à l'aide de deux cavaliers.

Schéma électrique

Afin de simplifier le montage, nous avons opté pour un microcontrôleur **PIC16F676** de **Microchip** que vous pouvez voir sur le schéma électrique (figure 1), dans un boitier de 14 broches avec une architecture RISC 8 bits et une mémoire programmable de type **Flash**.

Comme il est déjà programmé en usine, dès la mise sous tension, après le « **power-on-reset** », le **PIC** initialise les entrées **RA0** et **RA1** utilisées en **mode convertisseur A/D** et la **sortie RA2** qui commande le relais.

Les **broches RA4** et **RA5** sont initialisées en tant qu'**entrées avec résistance de pull-up**.

Elles sont dédiées à la lecture des cavaliers **SK9** et **SK10** (jumpers) pour le réglage du mode de fonctionnement.

Le **port RC** est représenté par les **broches RC0** à **RC5** qui sont **toutes initialisées en entrée sans résistance de pull-up**, et sont utilisées par le firmware pour **lire l'état de chaque cavalier** qui, lors du réglage du temps, **définit si le trimmer respectif travaille en heures, minutes ou secondes**.

Pour être précis, les cavaliers **SK3**, **SK5**, **SK7** concernent **respectivement** le **réglage,** des **secondes**, **minutes** et **heures** de la **pause** associé à RV1, **SK4**, **SK6**, **SK8** et **RV2** rconcernent le réglage des **secondes**, **minutes** et **heures** de **l'activation**.

Les résistances de « **pull-up** » sur les lignes **RC0**, **RC1**, **RC2**, **RC3**, **RC4**, **RC5** sont confiées à un **réseau de résistances « 8 x 1 »** (8 résistances avec un commun), car le **port RC** du **PIC ne dispose pas de résistances internes de « pull-up »** (résistance de tirage).

Une fois l'initialisation terminée, le programme va dans la boucle principale qui consiste à vérifier de manière cyclique l'état des entrées **RA1**, **RA0**, **RC0**, **RC1**, **RC2**, **RC3**, **RC4**, **RC5**. Plus précisément, le firmware vérifie les paramètres des cavaliers **SK9** et **SK10** pour décider dans quel mode de temporisation se positionner.

Si **SK9 se trouve ouvert**, le **temporisateur fonctionne en simple impulsion**, à savoir un seul cycle, tandis que si **SK9 est fermé le temporisateur répète la fonction de manière cyclique** jusqu'à ce que le circuit soit privé d'alimentation. Si **SK10** est ouvert, le **temporisateur commence par la pause puis active le relais**, si **SK10 est fermé le temporisateur commence par activer le relais puis le met au repos** (fonctionnement inverse).

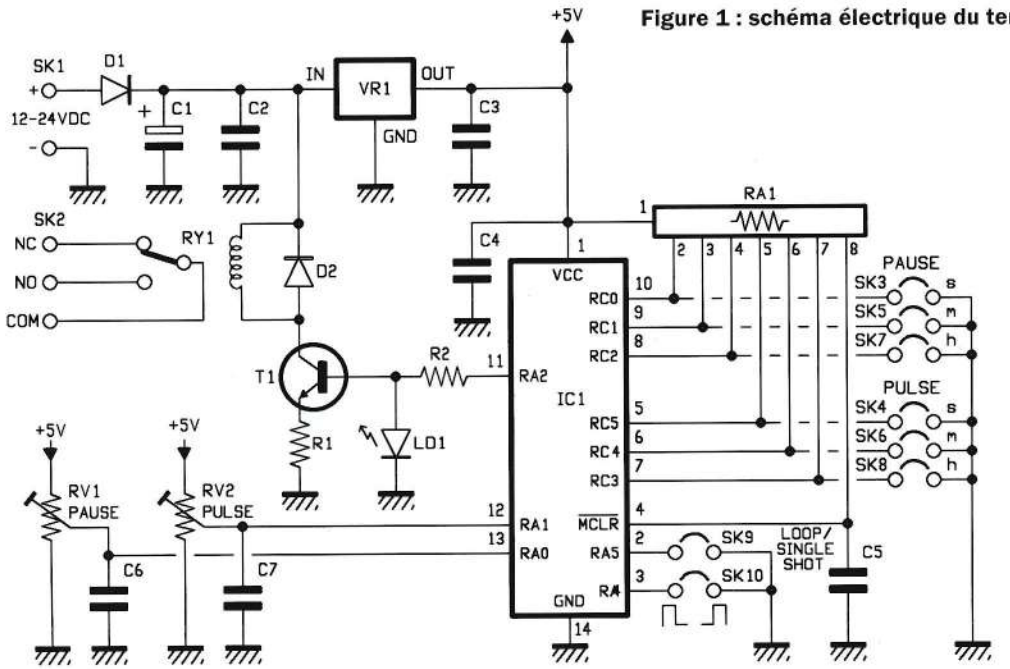


Figure 1 : schéma électrique du temporisateur de 1 s à 60 h.

Liste des composants du MK188

R1.. 33
R2.. 330
RA1..réseau de résistances 8 x 47 k + C
RV1.. Trimmer 4,7 k multitour
RV2.. Trimmer 4,7 k multitour
C1.. 220 µF 16 V électrolytique
C2.. 100 nF multicouche
C3.. 100 nF multicouche
C4.. 100 nF multicouche
C5.. 100 nF multicouche
C6.. 100 nF multicouche
C7.. 100 nF multicouche

D1.. 1N4007
D2.. 1N4007
IC1.. PIC16F676-I/P (VMK188)
VR1..7805
LD1.. LED 3 mm rouge
T1.. BD139
RY1.. Relais 12V VR17V121C

Divers :

Bornier 2 pôles au pas de 5 mm
Bornier 3 pôles au pas de 5 mm
Support ci 2 x 7 broches
Barrette sécable 2 pôles x 8
Cavaliers x 4

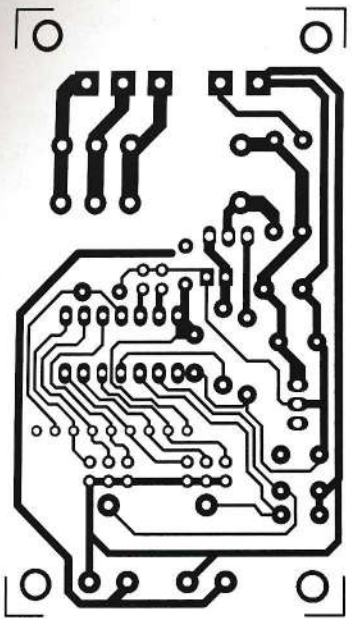


Figure 2 : dessin du circuit imprimé à l'échelle 1 : 1 du temporisateur.

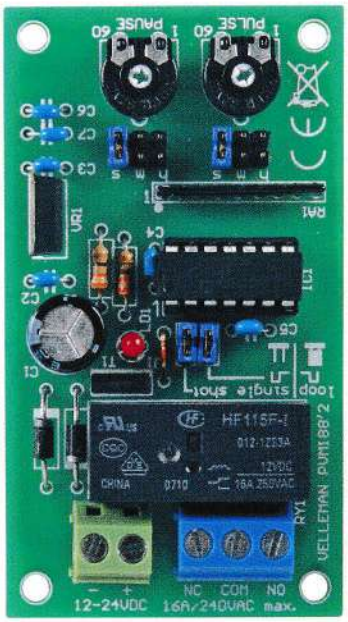


Figure 4 : photo de l'un de nos prototypes du temporisateur de 1 s à 60 h.

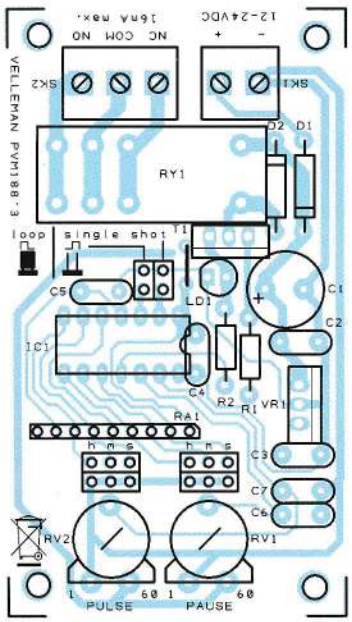


Figure 3 : schéma d'implantation des composants du temporisateur.

À tout moment il est possible de modifier le temp alloué à la pause et à l'activation, en utilisant le trimmer correspondant. A cet égard nous avons utilisé une astuce pour limiter le nombre de trimmers sur le circuit imprimé du temporisateur. Chaque trimmer définit les heures, les minutes et les secondes de la période avec laquelle il est associé, mais un seul à la fois.

Dans la pratique, le trimmer définit la période correspondante, de tour en tour, si le **cavalier est fermé** il y a une correspondance de **RV1** avec **SK3**, **SK5** et **SK7** et pour **RV2** avec **SK4**, **SK6** et **SK8**.

Notez que le curseur de chaque trimmer se règle dans le sens des aiguilles d'une montre, ce qui signifie que si le curseur est à l'extrême gauche la durée est minimale alors que s'il se trouve vers la droite la durée est maximale.

Pour **définir la durée de la pause**, nous utilisons le trimmer **RV1** : si **SK3 est fermé RV1 règle les secondes**, si **SK5 est fermé RV1 règle les minutes** et si **SK7 est fermé RV1 règle les heures**.

Si le **curseur du trimmer** est tourné dans le **sens anti-horaire** cela correspond à la **position (1)** qui est le **minimum** et quand il est **tourné vers l'extrémité opposée** cela correspond au **maximum (60)**.

A **mi-course** nous avons **30 secondes** ou **30 minutes** ou **30 heures**, à **1/3 de la course** de l'**extrême gauche** cela correspond à **20 secondes** ou **20 minutes** ou **20 heures**, au **2/3 de la course dans le sens horaire** nous avons **40 secondes** ou **40 minutes** ou **40 heures**.

Par exemple, si nous voulons régler un temps de pause de 1 heure 30 minutes et 30 secondes, nous devons fermer SK7 et tournez le curseur du trimmer RV1 complètement vers la gauche, puis ouvrir SK7 et fermer SK5, positionner le curseur de RV1 à mi-course, ensuite ouvrir SK5 et fermer SK3 avec RV1 à mi-course.

Une fois les réglages effectués, vous devez retirer le cavalier de SK3 et laisser ouvert SK3, SK5 et SK7.

Le **réglage de la durée de la pause est similaire**, nous fermons **SK4** et nous tournons le curseur du trimmer **RV2** pour régler les **secondes**. Ensuite nous **ouvrons SK4** et nous **fermons SK6** en tournant le curseur pour le **réglage des minutes**, et enfin nous **ouvrons SK6** et nous **fermons SK8** en tournant le curseur pour **régler les heures**.

Pour terminer le réglage nous ouvrons SK8 et nous laissons également SK4 et SK6 ouverts.

La figure 5 illustre les réglages du temps d'impulsion et de pause ainsi que les modes de fonctionnement du circuit.

Après avoir réglé les nouveaux paramètres de la temporisation souhaitée, pour **que les changements prennent effet**, vous devez **débrancher le circuit, attendre 20 secondes, puis le réalimenter.**

Nous terminons la description du circuit par l'étage de sortie et le bloc d'alimentation. Comme mentionné, la **sortie du circuit est un relais à un seul contact normalement fermé** (**NC**) dont la bobine est alimentée en **12 V** et pilotée par la broche **RA2** du microcontrôleur via le transistor **T1** qui agit comme un **amplificateur de courant**, ou si vous préférez comme un interrupteur statique (fonctionnement en commutation).

Chaque fois que le relais **RY1 doit être fermé**, c'est-à-dire que le contact s'établit entre les points **C** et **NO**, la broche **11** du **PIC16F676** passe à un état **logique haut** et **polarise la base** du transistor **NPN** à travers la résistance

Les réglages

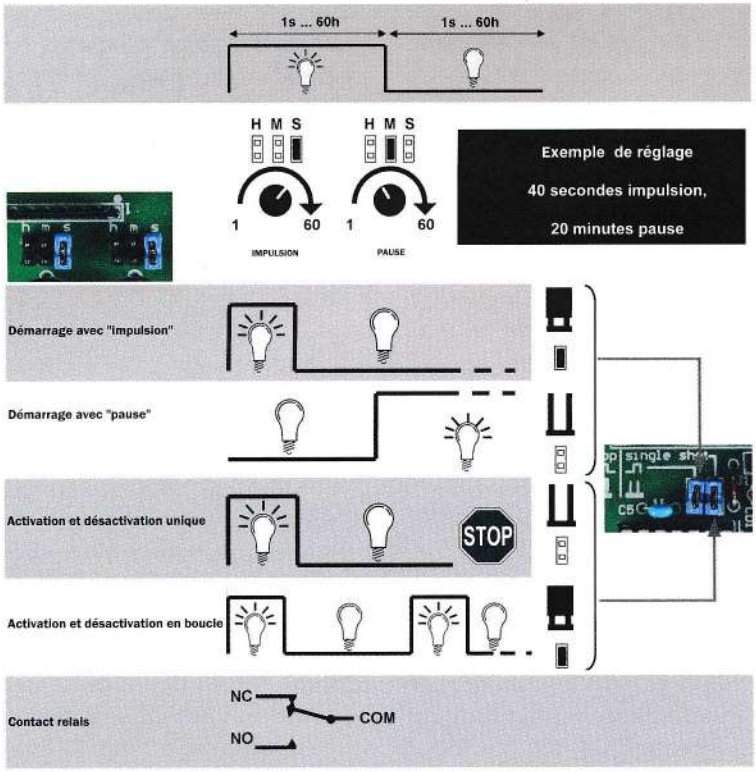
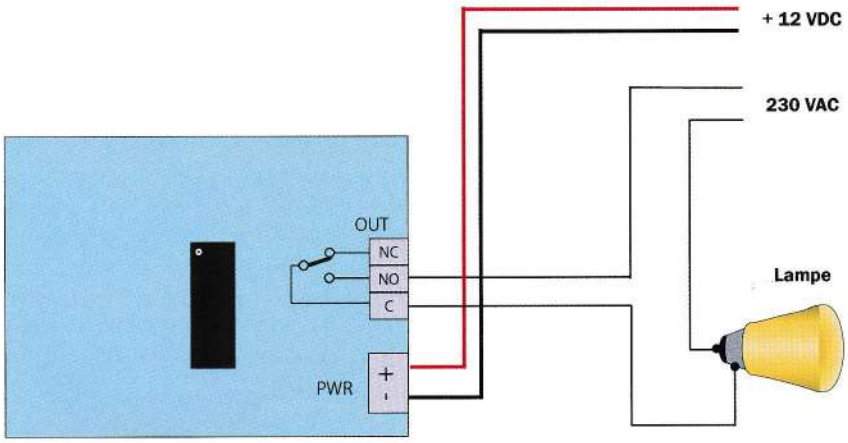


Figure 5 : les différents réglages et modes de fonctionnement du temporisateur ainsi que les définitions des durées d'impulsion et de pause : SK9 définit le mode cyclique ou one-shot (un seul coup), SK10 définit si le temporisateur doit commencer avec le relais activé ou au repos. Les trimmers règlent les heures, minutes et secondes (un pour l'impulsion d'activation du relais et l'autre la période de repos) sur la base du cavalier qui est actuellement fermé.



R2 qui a la double fonction de **limiter le courant de base** de **T1** et le **courant direct** de la LED **LD1**, celle-ci indique visuellement que le relais est activé.

Notez que **la présence de la résistance R1 est indispensable pour maintenir un potentiel élevé de la base du transistor** et permettre ainsi à la LED d'être allumée. Etant donné que c'est une LED rouge, elle nécessite une tension « anode – cathode » d'au moins 1,8 V. Si l'émetteur de T1 était directement relié à la masse, la tension Vbe du transistor serait de 0,6 à 0,7 V, ce qui ne serait pas suffisant pour allumer la LED.

En plaçant R1, la somme de la **tension Vbe** de **T1** et de la **chute de tension aux bornes** de **R1** atteint les **1,8 V** nécessaires pour **allumer LD1**. Lorsque la base est polarisée, un **courant suffisamment élevé traverse le collecteur de T1** pour **alimenter la bobine du relais** et par conséquent enclencher le contact. Lorsque la broche **RA2** du microcontrôleur retourne à un **niveau logique zéro**, le transistor **T1 est bloqué**, la **LD1 est éteinte** et le **relais est au repos**.

Notez que la diode **D2** qui est connectée entre l'alimentation et le **collecteur** de **T1** sert à **protéger** la **jonction « base – collecteur »** du transistor lorsque celui-ci est **bloqué**.

En effet, dans cette situation la bobine du relais, comme toutes les inductances, réagit à l'interruption brusque du courant et génère un pic tension de polarité opposée à celle reçue à l'instant précédent. La diode D2 devient conductrice et évacue la surtension, sinon cela provoquerait des dommages dans de la jonction « base – collecteur » de T1.

Nous concluons par l'analyse du schéma de l'alimentation, le circuit peut être alimenté par une tension continue ou alternative. Dans le premier cas il est nécessaire d'appliquer une tension de **12** à **14 VDC** non stabilisée ou à partir d'une **batterie** de **12 V**. Dans le second cas la **tension alternative doit être au maximum de 11 VAC efficaces**.

Lorsque nous utilisons une alimentation continue, la diode **D1 protège le circuit contre les inversions de polarité**, tandis que dans le cas d'une **alimentation alternative**, la diode **D1** sert de **redresseur mono alternance** (redresseur demi-onde) qui élimine les demi-ondes de polarité négative.

Le condensateur **C1 filtre la tension redressée** pour la rendre **continue**. Dans tous les cas, la tension présente sur les condensateurs C1 et C2 (celui-ci est utilisé pour filtrer les

parasites hautes fréquences présents sur la ligne d'alimentation) arrive sur l'entrée du régulateur **VR1** (un 7805), qui délivre à sa sortie une tension de 5 V parfaitement stabilisée alimentant le microcontrôleur IC1.

Notez que vous pouvez également **alimenter le circuit en 24 VDC**, dans ce cas vous devez **utiliser un relais dont la tension de la bobine est de 24 V** (compatible avec la tension d'alimentation).

Réalisation pratique

Maintenant que nous vous avons expliqué le fonctionnement du temporisateur, nous allons dire quelques mots sur la réalisation pratique. Le fabricant du kit ne propose pas la licence du programme, ni même l'achat uniquement du **PIC16F676** programmé.

Nous conseillons à nos fidèles lecteurs vu le prix du kit (une vingtaine d'euros) de l'acheter auprès d'un revendeur (voir les publicités), le circuit imprimé est déjà fabriqué et percé et le PIC est correctement programmé.

Pour nos lecteurs chevronnés, le typon du circuit imprimé simple face (voir la figure 2) est disponible sur notre site **www.electroniquemagazine.com** dans le **sommaire détaillé du numéro 127 à l'onglet « Télécharger »**.

Cependant il faudra écrire le programme, en utilisant un compilateur BASIC qui est plus simple que le langage C, cela ne devrait pas être compliqué.

Commencez par souder les composants qui ont le profil le plus bas (résistances, support du microcontrôleur et la diode) et ensuite continuez avec les trimmers, le réseau de résistance (attention à l'orientation : le point indique la broche 1).

Soudez en premier les condensateurs non polarisés et ensuite ceux polarisés.

Soudez les cavaliers à l'aide de barrettes sécables au pas de 2,54 mm et utilisez des straps au pas de 2,54 mm pour fermer les cavaliers pendant la phase de réglage.

Continuez le montage en insérant la LED (le méplat vers les cavaliers), le transistor BD139 (la face métallique vers LD1) et le régulateur 7805, celui-ci doit être soudé avec sa partie métallique vers le bord du circuit.

Enfin montez le relais, n'oubliez pas de souder le pont près du transistor T1, en utilisant un morceau d'une patte d'une résistance ou d'un condensateur.

Pour les connexions de la charge à piloter (bornes C, NO et NC du relais) et de l'alimentation, soudez les borniers au pas de 5 mm dans les emplacements correspondants. Vérifiez le positionnement correct de tous les composants en vous reportant au schéma d'implantation des composants visible sur la figure 3.

Une fois que vous avez vérifié les soudures et que tout était correctement en place, vous pouvez insérer dans le support le microcontrôleur déjà programmé, la broche 1 est repérée par le « rond » qui doit être positionné vers le relais.

À ce stade, vous êtes prêt pour installer votre temporisateur et le faire fonctionner sur l'intervalle de temps que vous voulez.

En figure 6 nous donnons un exemple de câblage d'une lampe dont vous pourrez vous inspirer pour faire fonctionner toute autre charge.

Rappelez-vous que le **circuit est alimenté avec une tension comprise entre 12 VDC et 24 VDC et nécessite un courant d'environ 100 mA avec le relais activé**, les contacts du relais peuvent supporter une **charge de 16 A** sous une tension de **230 VAC**.

Lors de l'installation du temporisateur dans des systèmes **soumis à des tensions du réseau EDF**, n'oubliez pas d'être prudent, car une erreur peut provoquer des pannes ou **même une électrocution. Vous devez toujours débrancher en premier l'alimentation du système du réseau EDF** et **ensuite vérifier les connexions**.

Soignez l'isolation du circuit imprimé avec une surface en plastique, car une fois alimentées les **pistes de cuivre sont soumises au 230 VAC**.

Placez le temporisateur dans un boitier en plastique que vous aurez au préalable percé pour rendre accessible les trimmers et permettre le passage des câbles d'alimentation et de la charge.

Comment construire ce montage

Le kit complet est disponible sous la référence MK188 auprès de la socité COMELEC. Le typon du circuit imprimé est téléchargeable gratuitement sur notre site Internet à la nouvelle adresse : **www.electroniquemagazine.com** dans le sommaire détaillé de la revue numéro 128 section « Télécharger ».

PROGRAMMEZ AVEC ANDROID

Première partie

A travers cette série de plusieurs articles, nous allons vous apprendre à développer des applications pour smartphones, tablettes et systèmes « embarqués » sous la plate-forme Android. Cela vous permettra d'interfacer avec le monde extérieur vos projets électroniques.

de Andrea Chiappori

Android, qui appartient au géant de l'internet **Google**, est sortie à la fin de l'année **2007**. Il a été conçu comme un **système d'exploitation Open-Source** pour systèmes embarqués et les années suivantes il a obtenu un énorme succès, ce qui le rend actuellement le concurrent le plus important face au système **iOS** d'**Apple** pour **iPhone**.

De nos jours, il existe énormément de smartphones et de tablettes fonctionnant sous **Android** sur le marché. Dernièrement des systèmes de développement sont apparus pour soutenir ce système d'exploitation, en effet étant **Open Source**, il permet une personnalisation et une modularité de nombreux périphériques avec des fonctionnalités et des caractéristiques différentes répondant ainsi aux besoins de chacun.

Présentation du cours

Dans ce cours, qui s'étalera sur plusieurs articles, nous allons vous plonger dans le monde d'**Android** et vous familiariser avec les outils disponibles gratuitement grâce à la communauté de développement de ce nouveau système d'exploitation. Puis nous développerons notre première application qui fonctionnera sur n'importe quel smartphone ou tablette, et avec lequel (ou laquelle) nous serons en mesure de contrôler à distance nos projets électroniques, de commander, d'envoyer et de recevoir des données utiles qui pourront être affichées sur l'écran de notre dispositif.

La communauté de développement du système **Android** est, en fait, en perpétuelle évolution (comme vous pouvez le constater en visitant le site **http://developer.android.com**) et nous donne tous les outils et librairies nécessaires pour prendre pleine possession de notre smartphone et de l'ensemble de ses capteurs et périphériques.

Le smartphone (ou la tablette), sur lequel nous allons développer notre application, possède un **module Bluetooth**, un **récepteur GPS intégré**, un **module Wi-Fi**, un **accéléromètre**

Kits de développement pour systèmes Android

Android n'est pas seulement un projet logiciel, mais implique également un certain matériel pour son fonctionnement. De nombreuses plates-formes de développement basées sur ce système d'exploitation Open Source peuvent exploiter et contrôler divers périphériques.

FIG. 1

Par exemple vous pouvez voir sur la figure 1 la carte de développement « **IOIO** » pour **Android**. C'est une interface spécialement conçue pour être pilotée par le système **Android** via le port USB. Le contrôle de la carte est réalisé à l'aide d'une interface de programmation (API) JAVA simple et intuitive.

Le processeur embarqué de l'interface « IOIO » interprètera ainsi les commandes provenant de l'application **Android** de votre smartphone. Le module « **IOIO** » permettra à votre système **Android** d'interagir avec le monde extérieur en lui permettant de disposer de ports d'entrée/sorties en tout ou rien, de sorties PWM, d'entrées analogiques, de liaisons SPI™, I2C™, UART...

Microchip a également développé des systèmes pour **Android**, avec des kits de développement basés sur des microcontrôleurs 16 bits de la série PIC24F visibles sur la figure 2.

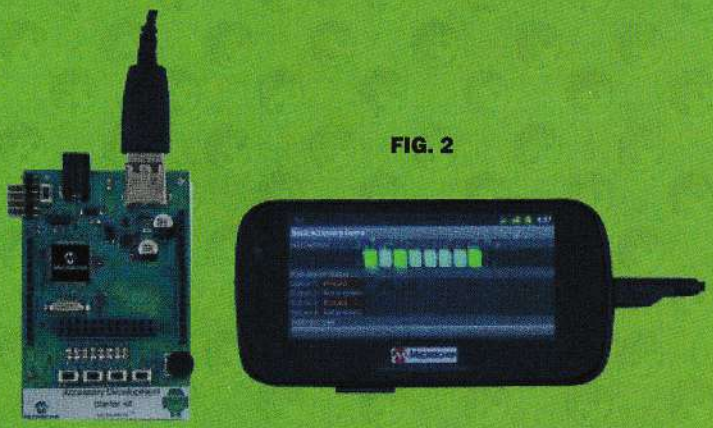


FIG. 2

Sans oublier l'une des plus récentes familles « Arduino » qui, en plus d'intégrer par rapport aux versions antérieures un contrôleur hôte USB pour s'interfacer avec **Android**, dispose d'un **ATmega2560** plus puissant et avec une plus grande connectivité.

C'est la carte **Mega ADK** pour **Android** visible sur la figure 3, elle dispose de 54 entrées/sorties numériques (dont 14 peuvent être utilisées comme sorties PWM), 16 entrées analogiques, 4 UARTS, un oscillateur à quartz de 16 MHz, une connexion USB, une prise d'alimentation, un connecteur ICSP, et un bouton de reset.

FIG. 3

3 axes, un appareil photo, et bien sûr un affichage à écran tactile, sans oublier le haut-parleur, le microphone, et le connecteur pour une carte mémoire externe microSD de quelques gigaoctets (Go).

Il y a simplement quelques années, toutes les fonctions présentes aujourd'hui sur la plupart des smartphones, n'auraient été disponibles que sur des systèmes professionnels réservés à des développements de haut de gamme.

Avec **Android** et un outil de développement, nous aurons la possibilité de faire interagir entre eux et piloter des modules à volonté pour créer des interfaces avec le monde extérieur et ajouter ainsi une valeur à nos projets électroniques. En particulier dans les articles suivants, nous réaliserons une carte dotée d'un module Bluetooth que nous pourrons piloter avec notre smartphone **Android**, et avec un module Wi-Fi nous procèderons à l'échange de données entre plusieurs périphériques.

Il existe aussi des cartes de développement fabriquées par des entreprises extérieures, telles que Atmel ou Microchip, que nous pouvons interfacer directement via un câble USB à n'importe quel dispositif afin de le personnaliser à notre goût. Certaines de ces cartes de développement (demoboard) sont présentées dans l'encadré « Kits de développement pour systèmes **Android** ».

La programmation sous Android

Pour programmer sous **Android** nous devons utiliser le langage « **Java** ». En fait, précisons que le langage utilisé n'est pas exactement du « **Java traditionnel** » que certains d'entre nous connaissent. Normalement lors de la compilation d'un programme écrit en Java, un « **bytecode** » est généré et exécuté par une **Machine Virtuelle Java** ou « **JVM** » (**Java Virtual Machine**).

Le « **bytecode** » est un **code binaire qui permet un traitement plus rapide** que le **code source Java**, et qui rassemble (compile) tous les codes

dispersés dans différents fichiers lors de l'écriture du programme. La « **JVM** » peut être représentée comme une implémentation logicielle qui permet au code Java de communiquer avec le système d'exploitation.

Sous **Android** le « **bytecode** » généré après la compilation n'est pas tout à fait le même que celui généré par un compilateur JVM classique, mais plutôt un **DVM** ou **Dalvik Machine Java**. **Dalvik est une machine virtuelle destinée aux smartphones et tablettes tactiles**, qui est incorporée dans le système d'exploitation **Android**.

Dalvik est destiné à permettre l'**exécution simultanée de plusieurs applications sur un appareil de faible capacité** (peu d'espace mémoire et peu de puissance de calcul). Elle a été créée par **Dan Bornstein**, Dalvik est le nom du village de pêcheurs en Islande d'où est originaire l'auteur.

Dalvik offre une alternative à la **machine virtuelle Java**, le « **bytecode** » est transformé et consolidé en un fichier « **.dex** » (**Dalvik Executable**). **Le but de la machine virtuelle Dalvik est de permettre d'exécuter le même programme sur une grande variété d'appareils, quelles que soient leurs caractéristiques techniques**.

Le code exécutable du programme (« bytecode ») est transformé à la volée en instructions spécifiques pour l'appareil sur lequel le programme est exécuté. On parle alors de **compilation à la volée** (en Anglais « **just-in-time compilation** » ou « **JIT compilation** »). Chaque appareil **Android** a sa **propre DVM** qui est capable d'effectuer un « **bytecode** » particulier.

Dalvik exécute un « **bytecode** » dont les **instructions sont basées sur des registres**, alors que la **machine virtuelle classique JVM** de technologie **Java** est basée sur le principe de la « **pile** » c'est-à-dire une **structure de données fondée sur le principe du « dernier arrivé, premier sorti »** (les derniers éléments ajoutés à la pile seront les premiers à être récupérés).

Une **machine à registres nécessite moins d'instructions pour effectuer les mêmes opérations qu'une machine à pile** et est par conséquent **mieux adaptée à un appareil ayant peu de puissance de calcul**.

En raison de cette différence, les **fichiers « bytecode » ordinaires Java ne peuvent pas être exécutés par Dalvik**, et un programme inclus dans le **SDK Android** transforme au préalable ces fichiers en « **.dex** » (Dalvik Executable).

Les fichiers exécutables pour **Dalvik** (**.dex**) sont créés par **consolidation des fichiers de type « bytecode » Java**. La taille du fichier « **.dex** » est de l'**ordre de la moitié de la taille des fichiers « bytecode » classiques Java**. Tous les services fournis par **Android** ainsi que les capacités matérielles des appareils sont mis à disposition à travers **Dalvik**. Cette machine virtuelle joue le rôle d'écran qui cache les caractéristiques techniques de l'appareil sur laquelle elle est exécutée.

Pour résumer, nous pouvons dire que la différence entre les deux machines virtuelles consiste dans le fait que la DVM est conçue pour des appareils embarqués dans lesquels la quantité de mémoire ne peut pas être upgradée comme celle d'un PC pour améliorer les performances.

Bien sûr, au niveau de la programmation, si vous avez déjà des connaissances en Java vous n'aurez pas de problèmes pour passer à **Android**. Nos lecteurs qui programment en « **C** » ou « **C ++** » devront utiliser un outil de développement appelé « **NDK** » (**Native Development Tool)** ou outil de développement natif que vous pourrez télécharger sur :

***https://developer.android.com/tools/sdk/ndk/index.html*.**

Le **NDK est un ensemble d'outils qui vous permet de mettre en œuvre votre application** en utilisant des **langages de code natif** comme « **C** » et « **C ++** ». Pour certains types d'applications, cela peut être utile car vous pouvez réutiliser des bibliothèques de codes existantes écrites dans ces langages.

Vous pourrez ainsi écrire vos applications en code natif, ce qui augmente la performance, la vitesse d'exécution et facilite la réutilisation de votre code « C » ou « C ++ ».

La plateforme SDK Android

Maintenant, nous allons installer sur notre ordinateur (PC, MAC, LINUX) tout ce dont nous avons besoin pour développer sous la plateforme **Android**. Les étapes sont très simples, tout d'abord nous devons installer le « **Run Time Java** » (JDK) qui est essentiel pour la réalisation de nos projets. Ensuite nous devrons installer l'outil de développement pour **Android**, appelé « **plateforme SDK** » qui se trouve sur le site :

http://developer.android.com/sdk/index.html

Lors de l'installation, nous allons choisir les « paquets » que nous allons installer parmi ceux disponibles. Il existe plusieurs versions de la « **plateforme SDK** » qui correspondent à des versions d'**Android** disponibles sur le marché. Vous pouvez également toutes les choisir, mais au début, afin de ne pas créer trop de confusion, vous devez vous concentrer sur une seule version en fonction de la version du système (smartphone ou tablette) à notre disposition et sur laquelle nous allons développer dans l'avenir.

Pour avoir une idée générale sur les versions d'**Android** et leurs noms de code, regardez l'encadré intitulé « Les versions d'**Android** et leurs noms de code ». Pour l'instant vous devez savoir que les versions 1.5 et 1.6 sont de nos jours obsolètes, les versions **2.1** à **2.3** sont **essentiellement présentes sur des smartphones relativement anciens mais qui représentent une grande partie du marché**, tandis que les versions **3.0**, **3.1** et + sont plutôt réservées aux **tablettes**.

Enfin, les **nouvelles versions 4.0** et supérieures **fonctionnent à la fois sur les tablettes et smartphones haut de gamme** en utilisant les mêmes « fragments » de code. Elles disposent de fonctionnalités étendues telles que le support du « **Wi-Fi Connect** » c'est à dire la **possibilité de se connecter directement à deux appareils**

Les versions d'Android et leurs noms de code

A chaque version du système d'exploitation **Android** développé par **Google**, est associé un **nom de code** en se basant sur des « desserts ». Au **Googleplex**, qui est le **siège social de Google** (en **Californie**), sont présentes des sculptures qui représentent ces produits (figure 4). Comme vous pouvez le voir dans le tableau 1, les initiales du nom de chaque version sont classées par ordre alphabétique. La **version 2.1** (« **Eclair** ») a gardé le même nom en raison de certains problèmes de la version 2.0 qui était instable. Les versions « **HoneyComb** » ont été spécialement conçues pour les tablettes, tandis que les versions « **IceCream** », dernières sorties, sont réservées pour les appareils haut de gamme. Elles disposent d'une interface unique à la fois pour les smartphones et les tablettes.

Le plus grand bon technologique en **termes d'améliorations des performances est le passage de la version « Eclair »** à « **Froyo** ». Cette dernière version permet de transformer un appareil **Android** connecté en réseau en un « hotspot Wi-Fi » (cette propriété est appelée « **Tethering** » qui veut dire « **modem affilié** » et permet à un appareil de procurer à un autre appareil l'accès à Internet.) ainsi que le transfert des données via le module « Bluetooth » et pas seulement par la « voix » comme dans les versions précédentes. Cette dernière fonctionnalité est très utile si nous voulons utiliser notre appareil **Android** pour contrôler à distance nos projets, comme nous le verrons dans les prochains articles. Vous pouvez toujours mettre à jour votre système Android vers une version plus récente que celle préinstallée, vous devez pour cela visiter le site Web du fabricant de votre appareil. **Android** est un système **Open Source**, la plupart des connaisseurs (« geeks ») peuvent également modifier et installer d'autres systèmes d'exploitation alternatifs, mais en prenant le risque d'avoir un dispositif instable.

Tableau 1

VERSIONS D' ANDROID		
Numéro de la version	**Nom de code**	**Versions d'API**
1.0	Apple Pie	1
1.1	Banana Bread -	2
1.5	Cupcake -	3
1.6	Donut	4
2.0.x	Eclair	5, 6
2.1.x	Eclair	7
2.2.x	Fro Yo (Yaourts glacés)	8
2.3.x	Gingerbread	9, 10
3.x	Honeycomb	11, 12, 13
4.x	Ice_Cream_Sandwich	14, 15

« Wi-Fi » sans devoir utiliser un routeur central.

Cela ne signifie pas cependant qu'une application développée en utilisant les bibliothèques de la version 2.1 ne peut pas « tourner » sur un smarthpone disposant d'une version 2.3 d'**Android**. Elle fonctionnera aussi correctement sur une tablette, mais ne pourra pas profiter des fonctionnalités qu'offrent les dernières versions.

C'est pour cette raison et le fait que la grande majorité des smartphones disposent des **versions 2.2** ou **2.3**, que nous vous recommandons de choisir les **bibliothèques** de développement de la « **plateforme SDK 2.2** », elles peuvent être installées sur des versions plus récentes.

Parmi les différents packages installables, sont aussi présentes des bibliothèques « **Google** » qui peuvent être utilisées si vous souhaitez développer des applications ayant des Parmi les différents packages installables, sont aussi présentes des bibliothèques « **Google** » qui peuvent être utilisées si vous souhaitez développer des applications ayant des fonctionnalités de navigation avec des cartes géographiques, tel que le moteur « **Google Maps** ».

A la fin de l'installation, nous aurons tous les outils nécessaires pour construire notre première application, même si pour l'instant cela ne sera qu'à partir de lignes de commandes.

En fait, grâce à une série de commandes présentes dans le répertoire « **Tools** » (Outils), nous allons maintenant chercher l'endroit où nous avons déjà installé la plateforme **SDK-Android**, nous pourrons ainsi créer un projet simple et le compiler. Pour plus de détails référez-vous à l'encadré « Compilation à partir de la ligne de commande ».

« ECLIPSE », l'environnement de développement graphique

Compiler à partir de la ligne de commande est utile pour comprendre les différentes étapes du processus et les

Compilation à partir de la ligne de commande

Vous pouvez compiler un projet directement à partir de la ligne de commande en utilisant deux scripts simples. Vous devez d'abord créer le projet avec tous ses répertoires (dossiers) nécessaires en exécutant par exemple la commande suivante :

```
Android.bat create project -target 1 -name AndroidApplication -path ./myProject -activity AndroidActivity -package com.packagename.android.
```

Voici une brève description des paramètres :

- ***target*** : spécifie la version des bibliothèques utilisées, par exemple pour la librairie API 8 (Android 2.2), la valeur doit être égale à 1 ;
- ***name*** : indique le nom du projet ;
- ***path*** : indique le chemin vers le répertoire dans lequel le projet sera créé avec ses sous-répertoires ;
- ***activity*** : spécifie le nom de la classe principale qui sera créée automatiquement ;
- ***package*** ; doit être un nom unique dans le cas où vous souhaitez publier l'application sur le market « Google Play » et normalement pour cette raison nous utilisons une adresse web inversée telle que « com.packagename.android ».

A ce stade, nous avons juste à exécuter la deuxième commande :

```
ant debug ou ant release
```

Suivons les étapes de la compilation. A la fin de celle-ci, nous trouverons dans le dossier « bin » le résultat final qui se présente sous la forme d'un fichier portant l'extension « .apk ». Nous verrons plus tard comment le tester directement sur un terminal ou sur un PC à l'aide d'un émulateur. Pour plus de commodité, nous vous suggérons d'ajouter à la variable d'environnement « **PATH** » de **WINDOWS** le répertoire dans lequel vous avez installé les « **Tools** » (Outils) d'**Android** de sorte que vous pouvez exécuter les commandes à partir de n'importe quel endroit.

différents fichiers qui entrent en jeu lors de la compilation du projet, mais si nous voulons avoir un environnement de développement pratique et intégré qui nous permette d'éditer le code source, compiler et déboguer sur un émulateur ou directement sur le périphérique physique en temps réel, contrôler pas à pas le programme, nous devons utiliser « **ECLIPSE** ».

C'est un environnement de développement (IDE) utilisé pour programmer les systèmes embarqués et développer en **Java**. Il existe plusieurs versions d'« **ECLIPSE** » à télécharger, voir cette adresse :
http://www.eclipse.org/downloads.

Mais pour notre projet il suffit d'une version de base, tel que « **Eclipse Standard 4.4** » (au moment où ces lignes sont écrites, il peut arriver que les recopies d'écran diffèrent par rapport à votre version).

Malheureusement « **ECLIPSE** » n'est pas en mesure de compiler nativement les sources pour **Android**, mais vous pouvez télécharger un plugin qui reconnaîtra l'outil de développement **Android** installé sur le système et le

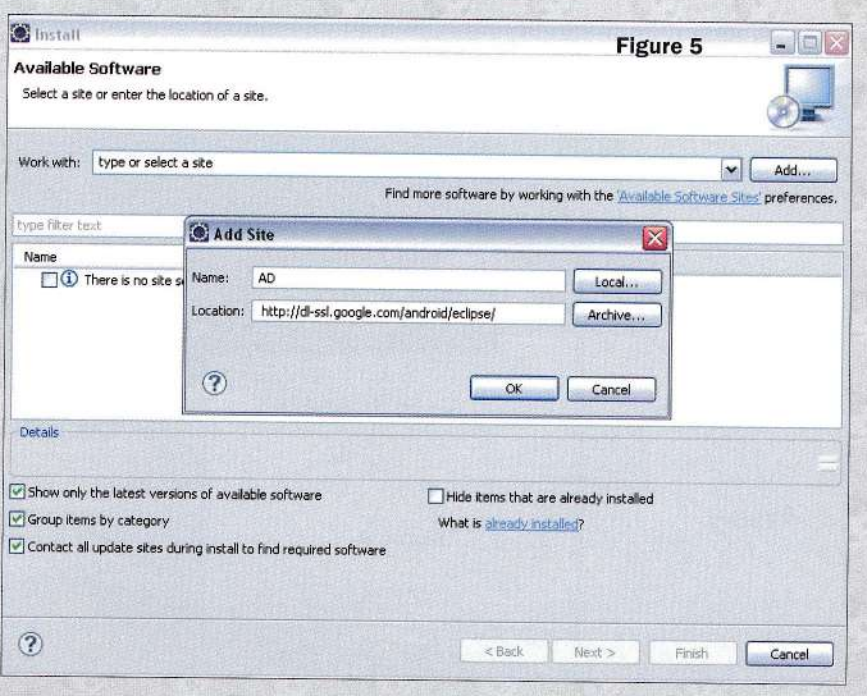


Figure 5

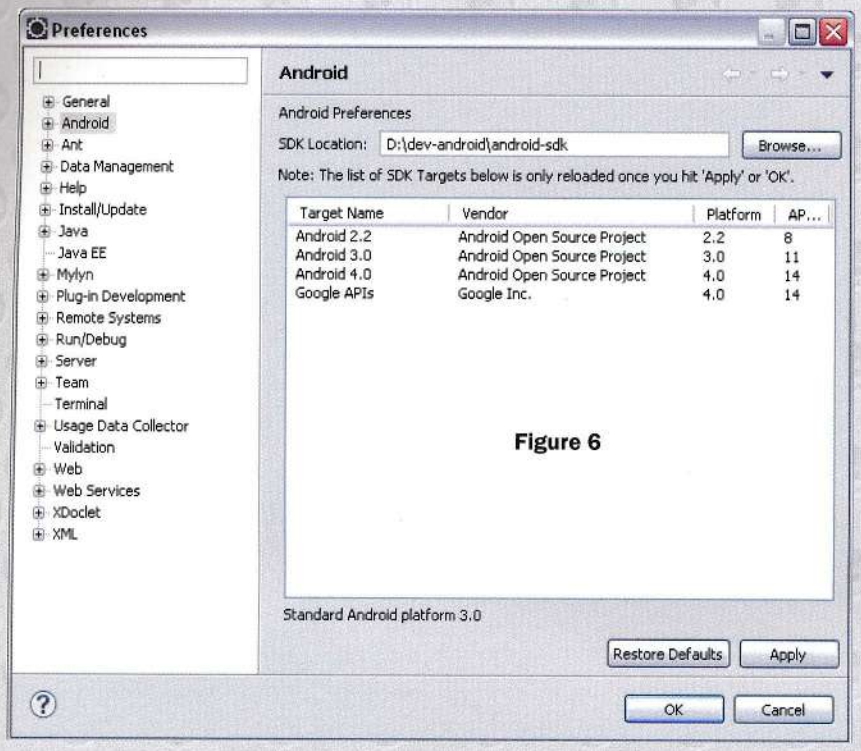


Figure 6

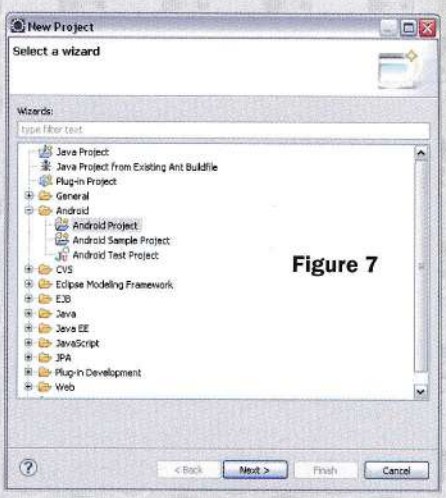


Figure 7

configurera de manière appropriée. Ce plugin est appelé « **ADT Plugin** » et peut être téléchargé à l'adresse suivante : ***http://developer.android.com/sdk/eclipse-adt.html***, ou plus facilement en ligne à partir de l'IDE en cliquant sur le menu « **Help** » (Aide), puis en cliquant sur « **Install New Software** » (installer un nouveau logiciel) et remplissez les champs avec l'adresse appropriée, comme indiqué sur la figure 5.

Une fois l'installation terminée, ouvrons la boite de dialogue « ***Preferences*** » accessible depuis le menu « ***Window*** »

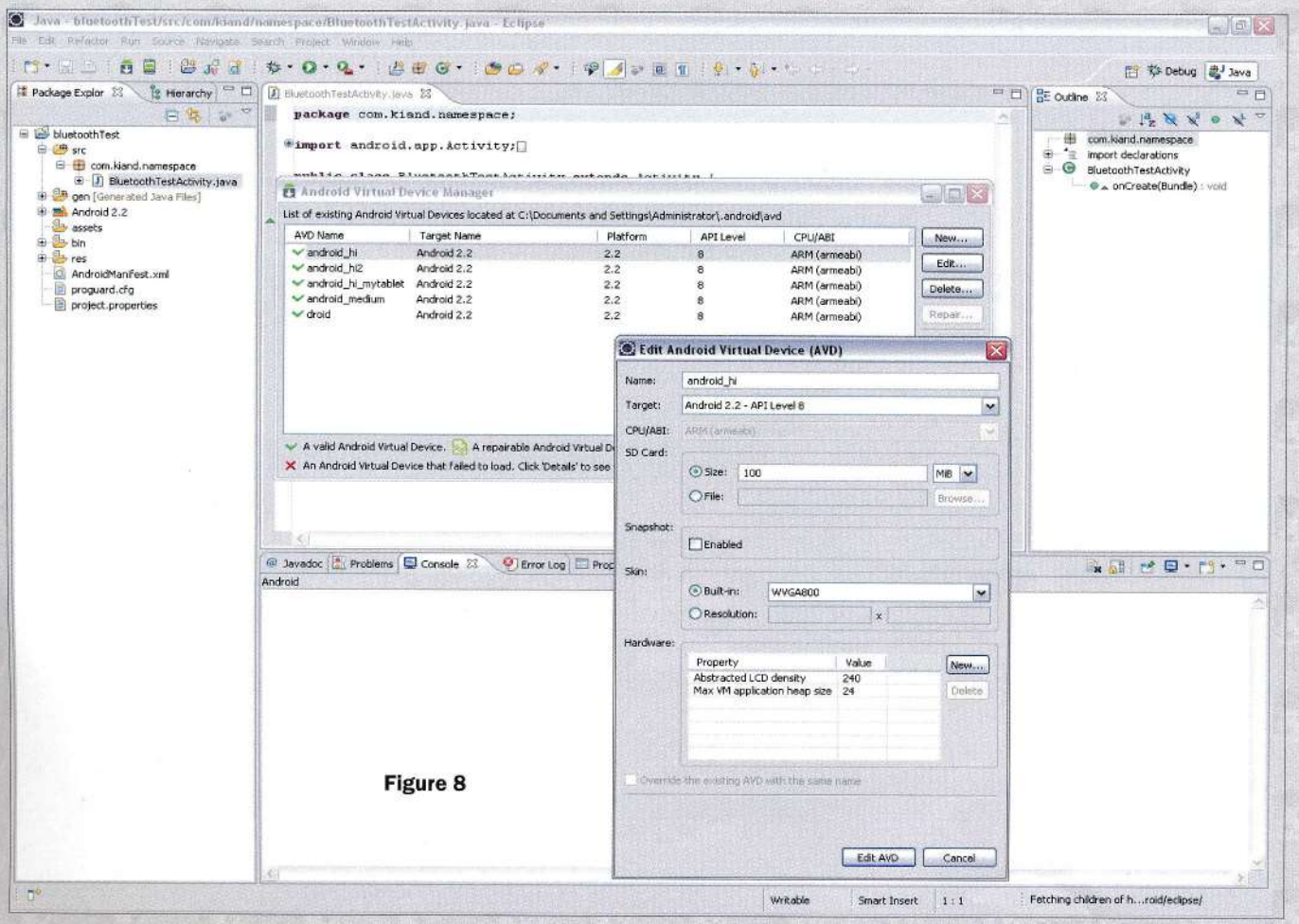


Figure 8

et indiquons dans le champ « **Android** » dans quel répertoire se trouve la **plateforme SDK**. Nous obtenons une liste de librairies installées similaires à celle de la figure 6. A ce stade, nous pouvons créer un nouveau projet **Android** comme le montre la figure 7. Suivons les étapes de l'assistant (« **Wizard** »).

Comme vous pouvez le voir, nous devons entrer les mêmes paramètres que dans le cas de la création avec les lignes de commandes. Avant d'effectuer un « **Build** » c'est-à-dire une **version exécutable** (ou **compilation**) d'une partie du projet ou du projet entier, cliquons sur le bouton droit de la souris en pointant sur le nom du projet dans la fenêtre « ***PackageExplorer*** » sur le côté gauche de l'IDE, puis en cliquant sur « ***Build Project*** ».

Ne nous attardons pas dans cette première partie du cours sur toutes les fonctionnalités d'« **ECLIPSE** », dans un premier temps il suffit de regarder dans la partie inférieure de l'IDE si un problème survient. Des informations sur la compilation du projet sont affichées, en particulier si elle a réussi ou non. Dans ce dernier cas les erreurs et/ou les avertissements sont signalés.

L'émulateur Android

Dans les prochains articles nous vous expliquerons en détail « **ECLIPSE** », mais dans cette introduction du **Cours**, nous devons vous parler d'un autre outil important qui est contenu dans le package « **SDK Android** » et que vous venez d'installer, à savoir l'émulateur. Celui-ci vous permettra de tester et debugger l'application sur votre PC sans avoir nécessairement un dispositif physique fonctionnant sous **Android**. Cela vous permettra d'accélérer le développement de l'application avec un gain de temps.

L'émulateur permet de développer également des applications **Android** pouvant fonctionner sur des appareils ayant des caractéristiques matérielles différentes, par exemple des appareils ayant une résolution et une taille d'écran différents d'une marque et d'un modèle à l'autre, ce qui procure un avantage significatif.

En réalité, nous avons l'intention de publier notre application sur le market ou « store » « **Google Play** » (le magasin **Android** à l'adresse :

https://play.google.com.

Elle devra être capable de fonctionner et de s'adapter avec n'importe quel appareil, comme nous ne pouvons pas avoir tous les appareils fonctionnant sous **Android** (en raison du coût global de ceux-ci), nous pourrons créer l'application et la tester dans l'émulateur à volonté pour chaque type d'appareils.

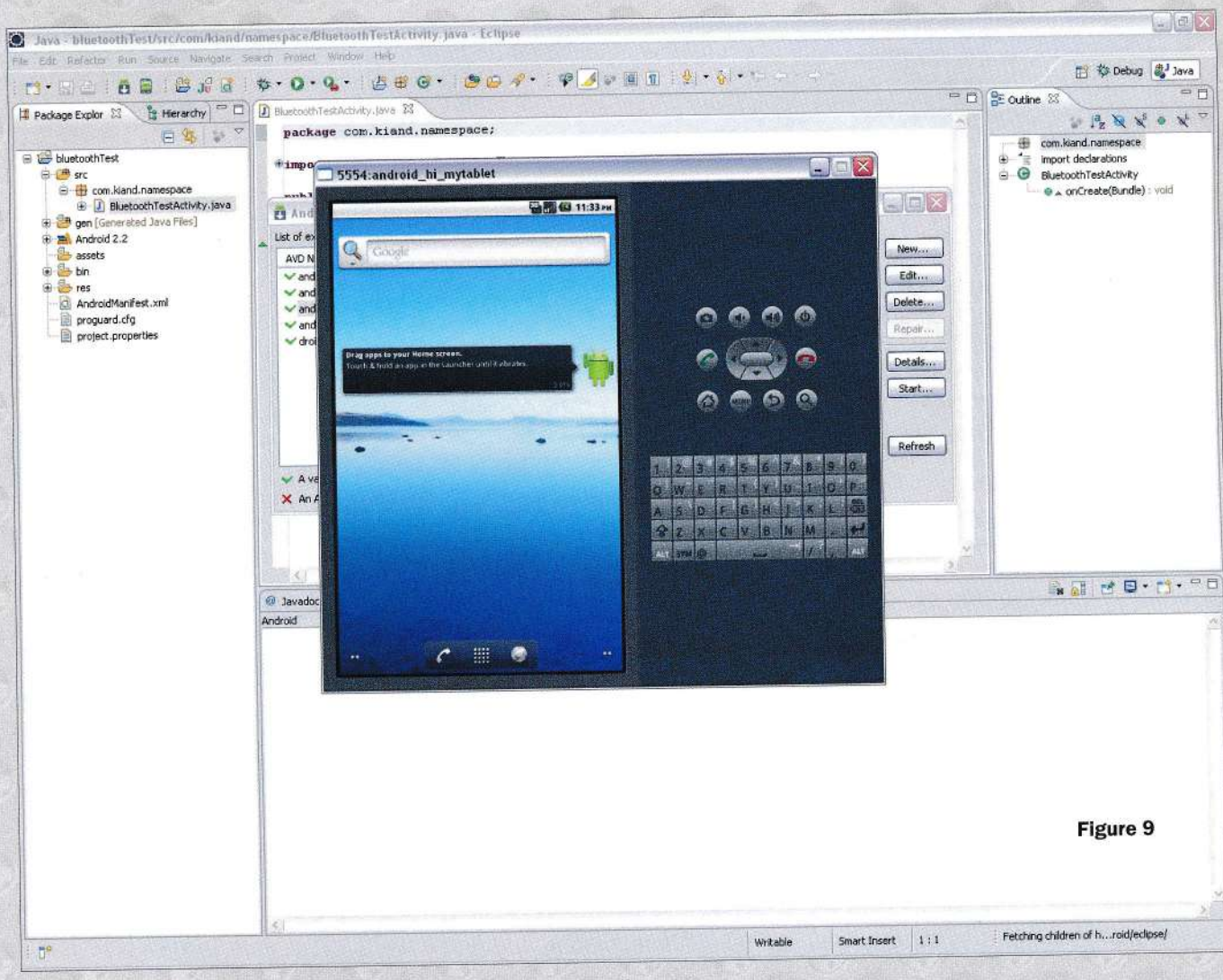


Figure 9

Ceci est possible grâce à l'exécutable « **AVD Manager** » (**Android Virtual Device**) présent dans le dossier d'installation (ou directement depuis « **ECLIPSE** ») et illustré sur la figure 8.

Dans la fenêtre principale, vous pouvez créer de nouvelles applications virtuelles, chacune avec une résolution différente et les associer à une version spécifique d'une librairie **SDK** (**API**) choisie parmi celles déjà installées (dans notre cas « **Android 2.2** », « **API Level 8** »).

Une fois que vous avez créé l'application désirée, vous pouvez choisir une librairie et commencer directement à partir de cette fenêtre. Après quelques minutes, vous serez en mesure de naviguer au sein de votre nouvelle application en utilisant la souris et des « touches virtuelles », vous pourrez lancer les différentes applications du système comme sur un vrai smartphone fonctionnant sous **Android** (voir la figure 9).

Ceci est très utile pour se familiariser avec le nouvel environnement **Android**, parce que, au-delà de notre application, vous pouvez naviguer dans les dossiers et « applications systèmes » comme s'il s'agissait d'un véritable smartphone.

Bien sûr, l'émulateur peut être exécuté directement à partir de l'environnement de développement « **ECLIPSE** » en effectuant une nouvelle compilation. Pour cela sélectionnez le nom et cliquez avec le bouton droit de la souris.

Ensuite cliquez sur « **Run As - Application Android** » (Exécuter en tant qu'application Android). Une fenêtre s'ouvre où vous pourrez choisir l'application que vous souhaitez émuler parmi celles précédemment créées et votre application sera exécutée directement dans l'émulateur.

Notez qu'il n'est pas conseillé de fermer la fenêtre de l'émulateur à la fin du test de l'application, mais plutôt vous devrez la garder ouverte afin de réduire le temps de chargement pour un nouveau test de l'application (« Build »). Le système trouvera l'émulateur déjà ouvert et l'application se chargera plus rapidement pour l'exécuter et la tester de nouveau. Avec les dernières versions de la plate-forme « **SDK Android** » quelques bugs ont été corrigés notamment sur le temps de chargement d'une application virtuelle qui était très long et parfois il était impossible de la tester et de la déboguer.

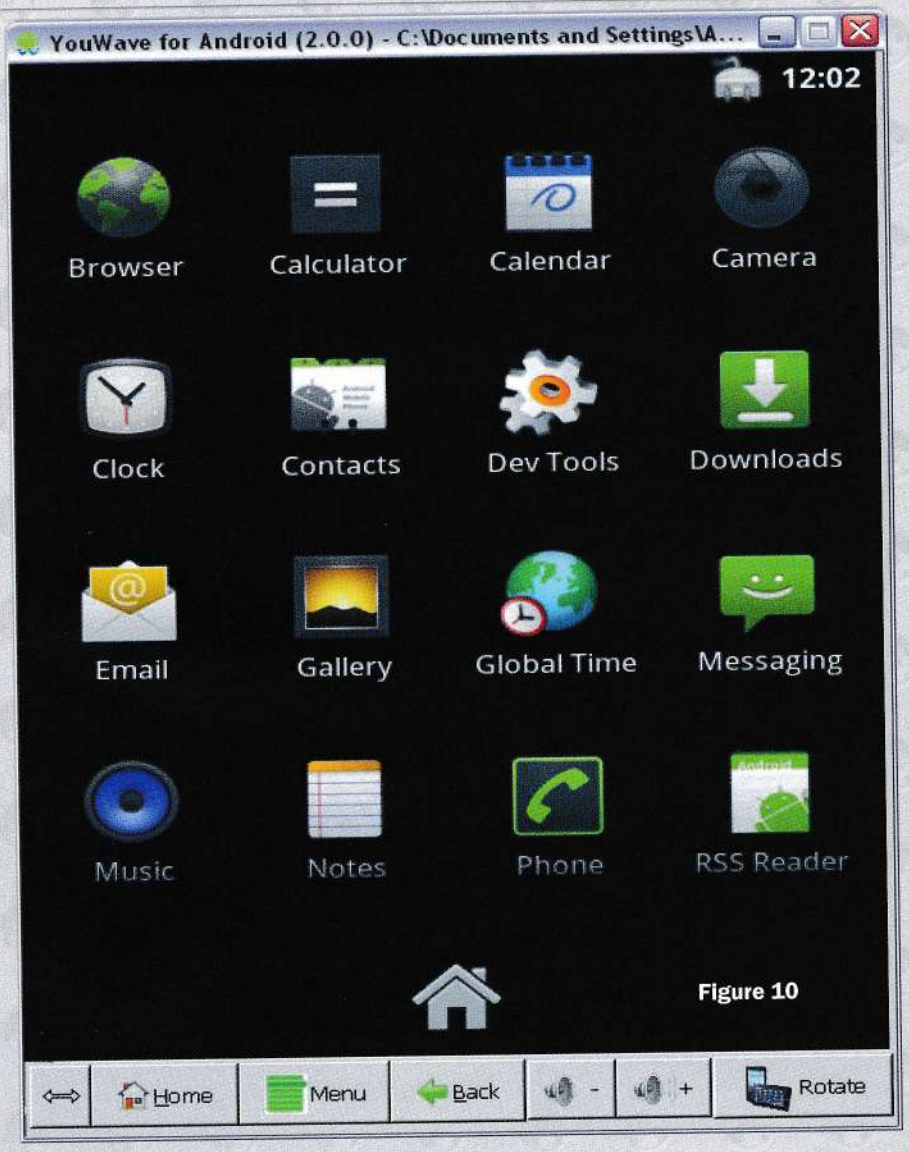


Figure 10

Cependant sur certains ordinateurs lents, il peut se produire ce genre de problème, nous vous suggérons d'installer un émulateur tiers qui est largement utilisé et surtout beaucoup plus rapide. Il s'appelle « **YouWave Android** » et peut être téléchargé à l'adresse suivante :

http://youwave.com.

Même s'il n'est pas libre de droit, il n'est pas très coûteux et vous pouvez l'essayer pour une période limitée de 7 jours (voir la figure 10). En copiant l'application générée à partir d'« **ECLIPSE** » (il s'agit d'un fichier « **.apk** » que nous étudierons plus tard) dans un dossier spécifique de « **YouWave** » vous serez en mesure de tester cet émulateur pendant une période limitée.

Conclusion

Dans cette première partie du **Cours Android**, nous vous avons présenté tous les outils pour développer sous **Android**, mais pour le moment nous nous arrêtons ici. Cela vous permettra d'intégrer et de tester ce nouvel environnement. Dans le prochain numéro nous aborderons l'aspect programmation avec des exemples de code **Java** et les premiers tests sur des dispositifs physiques.

ABONNEZ-VOUS

OUI, Je m'abonne à ELECTRONIQUE ET LOISIRS magazine — LE MENSUEL DE L'ÉLECTRONIQUE POUR TOUS

A PARTIR DU N° 129 ou supérieur → N°

E0128

Ci-joint mon règlement de ..... € correspondant à un abonnement de 4 revues Annuel

Règlement CB directement sur le site **www.electroniquemagazine.com rubrique Abonnement**

Adresser mon abonnement à :

Nom _____ **Prénom** _____

Adresse _____

Code postal _____ **Ville** _____

Tél. _____ e-mail _____

*Date, le*_____

Signature obligatoire ▷

TARIFS FRANCE

❑ **4 numéros** **28€,00**

TARIFS CEE/EUROPE

❑ **4 numéros** **32€,00**

DOM-TOM/HORS CEE OU EUROPE:

NOUS CONSULTER SUR www.electroniquemagazine.com rubrique Abonnement

L'ASSURANCE de ne manquer aucun numéro en recevant votre revue directement dans votre boîte aux lettres près d'une semaine avant sa sortie en kiosques.

BÉNÉFICIER de 50% de remise** sur les CD-ROM des anciens numéros

POUR TOUT CHANGEMENT D'ADRESSE, N'OUBLIEZ PAS DE NOUS INDIQUER VOTRE NUMÉRO D'ABONNÉ (INSCRIT SUR L'EMBALLAGE)

Bulletin à retourner à: JMJ – Abo. ELM

B.P. 20025 - 13720 LA BOUILLADISSE – Tél. 0820 820 534 – Fax 0820 820 722

Directeur de Publication
Rédacteur en chef
Jean Marc MOSCATI

Directeur Technique
Charles CARDONA

Direction - Administration
JMJ éditions
B.P. 20025
13720 LA BOUILLADISSE
Tél.: 0820 820 534

Secrétariat - Abonnements
Petites-annonces - Ventes
A la revue

Vente au numéro
A la revue

Publicité
A la revue

Maquette - Illustration
Composition - Photogravure
JMJ éditions sarl

Impression
Print Courtage
25 Bd Bouès
13003 Marseille

Distribution
Presstalis

Hot Line Technique
0820 820 534 *
du lundi au vendredi de 16 h à 18 h

Web
www.electroniquemagazine.com

e-mail
support@electroniquemagazine.com
* prix d'un appel local

JMJ éditions
Sarl au capital social de 7800 €
RCS MARSEILLE: 421 860 925
APE 221E
Commission paritaire: 1015T79056
ISSN: 1295-9693
Dépôt légal à parution

ELECTRONIQUE ET LOISIRS magazine — LE MENSUEL DE L'ÉLECTRONIQUE POUR TOUS

EST RÉALISÉ EN COLLABORATION AVEC :

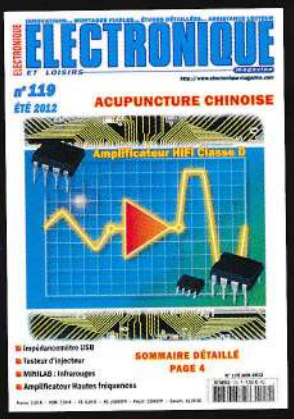
NUOVA ELETTRONICA
Elettronica In

10,50 €* la revue frais de port inclus pour la France Métropolitaine

CEE, les DOM-TOM et autres pays calcul sur le site www.electroniquemagazine.com

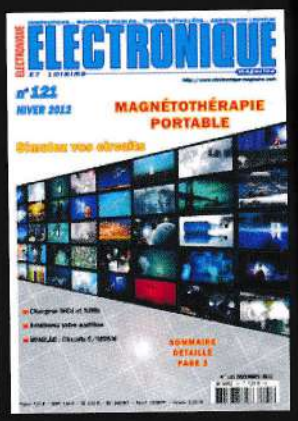
sommaire : «Theremin» en on professionnel - Deux alimentations écoupage avec dimensions réduites et possibilité d'obtenir une vaste gamme tensions - Emetteur FM 88-96 MHz onstruire sur la plaque d'essais du lab - Convertisseur N/A USB, avec ce oscopique convertisseur R2R nous allons sformer notre interface USB EN1741 en ertisseur N/A Numérique/Analogique) - nétiseur de 143 MHz à 970 MHz qui, relié générateur DDS, peut fournir n'importe le fréquence comprise entre 143 MHz 70 MHz avec une résolution de 10 Hz. - ne universelle pour LM358 - Antenne e pour ondes courtes.. Etc...

Au sommaire : Surveiller les fissures des murs avec l'USB. Stand-by (veille) off réactivable avec la télécommande: Réduisez votre facture d'électricité. Mesurer la distorsion avec un simple multimètre - Un selfmètre pour mesurer l'inductance des selfs - Mesurer la température avec le Minilab - Platine universelle pour LM358 - un amplificateur différentiel avec alimentation simple. Un sommateur inverseur et non inverseur avec alimentation double un convertisseur tension / courant un comparateur trigger de Schmitt - un intégrateur inverseur - un dérivateur inverseur - un amplificateur pour DDS. Etc...



Au sommaire : ÉLECTRORÉFLEX le générateur d'ondes «chinoises» utilisés pour contrer les douleurs aiguës de différentes origines - Amplificateur Hi-Fi stéréo 2x20 W en classe D, amplificateur aux dimensions réduites, que vous pourrez relier à votre Ipod, mp3 - Les rayons infrarouges avec le Minilab, expérimentations qui vous aideront à comprendre comment fonctionnent les dispositifs électroniques utilisant ces invisibles radiations électromagnétiques . Impédancemètre USB pour PC Seconde partie, le logiciel - Testeur d'injecteur pour automobile - Les amplificateurs RF à MMIC. très intéressants pour celui qui opère dans le domaine de la radiofréquence . Etc...

Au sommaire : Analyseur de spectre pour PC - Un récepteur FM à super-réaction avec une série de propositions d'applications pour le Minilab - Une barre lumineuse à LED pour téléviseur - Amplificateur linéaire RF large bande avec une paire de MOSFET PD55015 - Applications XOR et XNOR avec le programmateur CPLD, dédié aux applications pratiques réalisées avec notre programmateur pour dispositifs CPLD EN1685 - Un montage à ultrasons intéressant - Un antivol à ultrasons - Mini alimentation 9-12-15 V 0,4 A, conçue pour alimenter de petits circuits expérimentaux réclamant une tension de 9-12-15 V et un courant ne dépassant pas 0,4 A. Etc...



Au sommaire : Le QR CODE pour accéder rapidement à des contenus internet OPEN SOURCE : logiciel open ou free - La simulation de la 3D, est une technique de réalisation et de visualisation d'images, de dessins, photographies et films - Un micro stéréo préamplifié disposant de deux canaux indépendants - MINILAB : Expérimenter les CMOS - LTSpice pour apprendre à simuler vos circuits - Chargez les NiCd et NiMh avec votre alimentation. Il s'agit d'une manière intelligente et écologique de produire de l'énergie - Magnétothérapie RF professionnelle et portative - Le sismographe... ou ces secondes maudites - COURS: Le bruit des résistances. Etc...

Au somma GPS, combine les sur le réseau ce satellite GPS - L depuis des dé professionnel, ma commence à f monde des amat Box sur clé usb 1 LED comme veil de lumière - miniature - Raspb de crédit : Premi fonctionnant sou libre GNU/Linux W - Cours : Tuto

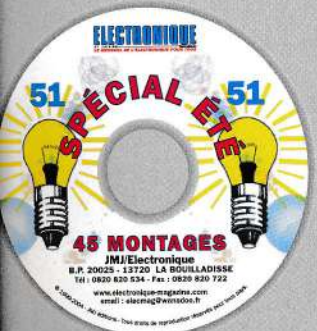
* 14,10 € pour la revue 127 frais de port inclus pour la France Métropolitaine

CD-ROM ENTIÈREMENT IMPRIMABLE

50 € Les 3 CD du Cours d'Électronique en Partant de Zéro

COURS Niveau 1,2 ou 3

19 € l'unité



Numéros spéciaux l'unité 5,50 €

CD - FRAIS DE PORT INCLUS POUR LA FRANCE (DOM-TOM ET AUTRES PAYS: NOUS CONSULTER.)

JMJ/ELECTRONIQUE - B.P. 20025 - 13720 LA BOUILLADISSE *règlement par Chèque à l'ordre de* ***JMJ ÉDITIONS***

Règlement par Carte Bancaire sur notre site : **www.electroniquemagazine.com** - Téléphone : 0820 820 534

Articles, Revues et CD téléchargeables au format PDF sur Internet

www.electroniquemagazine.com

LISEZ ET IMPRIMEZ VOTRE REVUE SUR VOTRE ORDINATEUR PC OU MACINTOSH

Au sommaire : Localisateur GSM/GPS, combine les techniques de localisation sur le réseau cellulaire GSM et le réseau satellite GPS - L'imprimante 3D qui existe depuis des décennies dans le monde professionnel, mais depuis deux ans environ commence à faire partie intégrante du monde des amateurs - Un lecteur MP3 Juke-Box sur clé usb 16 go - Comment utiliser une LED comme veilleuse et comme détecteur de lumière - Interrupteur crépusculaire miniature - RaspberryPi un PC au format carte de crédit : Première partie, PC de faible coût, fonctionnant sous un système d'exploitation libre GNU/Linux - Amplificateur stéréo 2 x 10 W - Cours : Tutoriel EAGLE CAD V 6

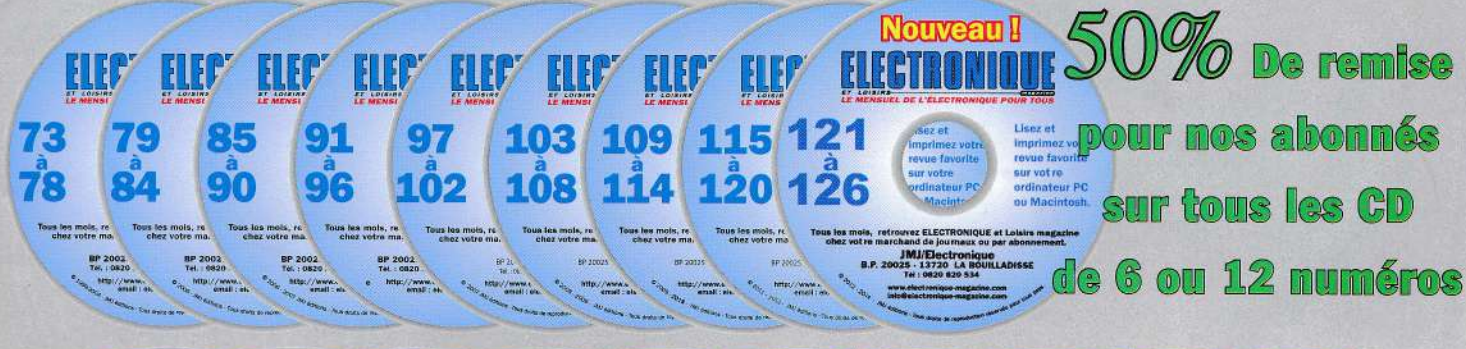
Au sommaire : Antivol pour panneaux solaires - L'imprimante 3D, le montage et les premiers pas, nous avons finalement décidé de vous dévoiler notre interprétation du projet d'imprimante 3D - RaspberryPi la programmation : 'il fonctionne comme un système embarqué, objectif, écrire un premier programme - MINIBUS, la qualité du signal a été améliorée afin de couvrir de longues distances entre les divers équipements. - Transmission audio par LED, utiliser son émission de lumière pour transmettre un signal audio en FM (Modulation de Fréquence). - Récepteur 4 canaux compatible MM53200, UM3750 et UM86409 Etc...

Au sommaire : Un serveur web avec le RaspberryPi, transformer le RaspberryPi en un système complet de gestion du port GPIO à distance (en réseau) - Chargeur d'accumulateur universel - Préamplificateur stéréo avec commande de tonalité numérique - 3DRAG 3 ème partie de l'idée à l'objet , nous allons nous concentrer sur le processus qui permet de passer de l'idée à un objet fini - MINIBUS, un bus pour l'automatisation, cet appareil permet de transmettre un signal audio en utilisant la ligne secteur 230 VAC de sorte qu'il soit disponible partout dans la maison sans avoir à tirer des câbles ou acheter des émetteurs/récepteurs radios Etc...

Au sommaire : La 3DRAG sans secrets, dans ce numéro nous allons aborder les logiciels et les aspects matériels de l'imprimante 3DRAG. Synthèse vocale pour RaspberryPi, transformez le RaspberryPi en un serveur web contrôlant à distance des entrées et des sorties - Capacimètre numérique de 10 pF à 10 000 µF, réalisez un capacimètre digital sans microcontrôleur et donc facile à reproduire - Chargeur d'accumulateur universel - MINI BUS, un bus pour l'automatisation : troisième partie. Alimentation symétrique de ± 1,25 V à ± 18 V / ± 1 A - Amplificateur pour MP3 & Smartphones Etc...

Au sommaire : Réalisez vos circuits imprimés avec la 3DRAG - Carillon électronique - Amplificateur Hi-Fi à MOSFET 2 x 200 W/4 Ω ou 350 W/8 Ω en pont - Protection pour amplificateur BF de puissance - Alarme pour voiture avec antidémarrage. Répétiteur de sonnerie pour téléphone. - Station météo avec RaspberryPi - Détecteur de gaz pour camping-cars et caravanes - Booster Classe H 2 X 70W pour voiture - Capacimètre numérique de 10 pF à 10 000 µF : 2ème partie - VU-mètre à LED - MINIBUS, un bus pour l'automatisation : 4 ème partie Etc...

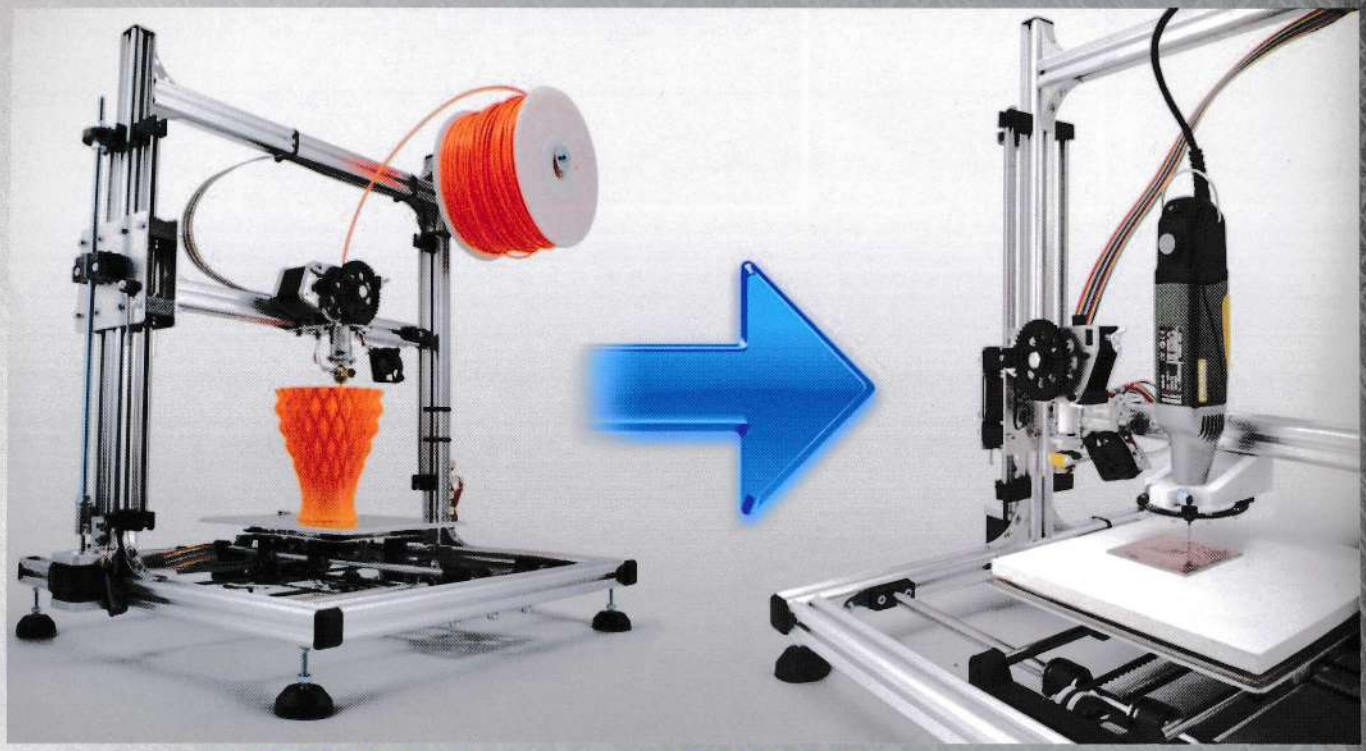
CD 6 Numéros 25 € / CD 12 Numéros 45€



50% De remise pour nos abonnés sur tous les CD de 6 ou 12 numéros

Transformez en Graveuse Votre imprimante 3DRAG

Cette imprimante 3D permet de créer des objets en plastique de formes diverses d'une taille maximum de 20 x 20 x 20 cm à l'aide de fil en ABS ou PLA de 3mm. L'impression est extrêmement rapide et précise, même à vitesse élevée.



3DRAG en Kit 599,90 € ou montée 789,00 €

Découvrez la mini perceuse professionnelle avec arbre de précision en acier rectifié monté sur roulements à billes. vitesse 5000-20000 tr / min, mandrin de 1 à 3,2 mm, sytème de maintien MICROMOT pour une utilisation verticale ou horizontale. Alimentation: 12 Vdc (via l'adaptateur de réseau 220-240 Vac), consommation maximale: 100 watts, longueur: 230 mm, poids: 500 grammes. Idéale pour le perçage, fraisage, rectification, polissage, gravure de circuit imprimé, pour la mécanique de précision, le modélisme, les bijoutiers, les opticiens, les artistes et les électroniciens.
Le kit comprend la mini perceuse et 34 accessoires dans un boîtier en plastique. Réf. **PROXXON1** **117,00 €**

Anneau LED POUR TÊTE D'IMPRESSION 3D
Réf. **VM8202** **17,00 €**

Plateau chauffant pour imprimante 3D
Réf. **A3D-3DHEATERPLATE** **28,90 €**

Extrudeur
à section chauffante pour le 3DRAG
Réf. **A3D-31** **59,80 €**

Lot de 5 feuilles
adhésives pour imprimante 3D
Réf. **A3D-FEUILLES3D** **10,90 €**

Nouveauté Bobine PLA FLUO

Bobine de fil PLA fluorescent pour imprimante 3D.
1KG – 3mm : orange , bleu , rouge , jaune **39,80 €**

COMELEC CD 908 - 13720 BELCODÈNE Tél. : 04 42 70 63 90 Fax : 04 42 70 63 95 www.comelec.fr