www.elektor-magazine.fr

.magazine

décembre 2014 | nº 438

elektor

CoCo-ri-Co

Cool Controller Concept

une approche nouvelle pour simplifier l'interface homme-microcontrôleur

sapin virtuel | **T-Board 28** | mesure de niveau et de distance | **module *Compute* pour Pi** | **VariLab 402 :** *alimentation de labo tandem de 40 V* | **modules de logiciel en C** | Red Pitaya | GestIC & 3D TouchPad | ***MagIC-VDRM :*** *régulateurs réglo !* | **commande de LED à la tablette** | **module Compute pour Pi** | Rétronique : ampli à tubes (1960)

L 19624 -438- F: 7,80 € - RD

ISSN 0181-7450

(B) 7,90 € • (CH) 13,20 FS • (CAN) 9,75 CAD • DOM surface 8,20 e • DOM avion 10,20 €
TOM surface 940 XPF • TOM avion 1620 XPF • Réu/A 9,70 € • Cal/S 950 CFP • Cal/A 1560 CFP • And 7,20 €

Au service du génie

National Instruments met à la disposition des étudiants le matériel et le logiciel dont ils ont besoin pour développer leur expérience, aller au-delà de la théorie, et réaliser l'importance du rôle de l'ingénieur dans la société.

>> Découvrez comment NI supporte la prochaine génération d'innovateurs, en visitant **ni.com/academic/f**

01 57 66 24 24

NATIONAL INSTRUMENTS

abonnez-vous à Elektor
devenez membre GOLD!

elektor GOLD membership

Les avantages de la formule Gold :

- 8 numéros et 2 numéros doubles sur papier et sous forme numérique
- 1 DVD-ROM annuel
- 10% de remise sur TOUS les produits sur www.elektor.fr
- l'accès direct en ligne à Elektor.LABS, le laboratoire virtuel d'Elektor
- des projets électroniques inédits exclusifs par voie de courrier électronique
- et enfin le bonus exclusif : une reliure gratuite (sur demande) pour des projets inédits.

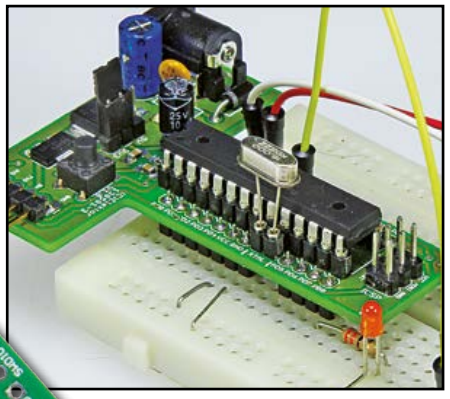
VOUS N'AVEZ PLUS BESOIN DU MAGAZINE IMPRIMÉ ?
Optez pour la formule Green
l'édition numérique (PDF) d'Elektor
sans papier ni DVD-ROM
économique et écologique
… avec les mêmes avantages !

elektor GREEN membership

Demandez votre carte de membre sur www.elektor.fr/membres

sommaire

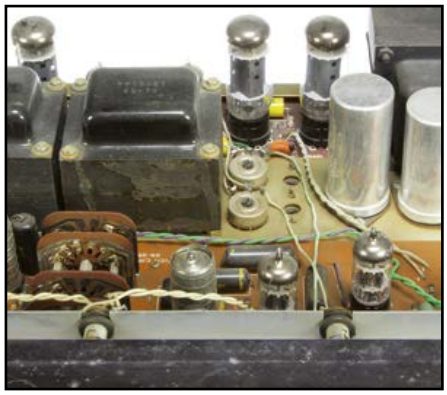
elektor magazine

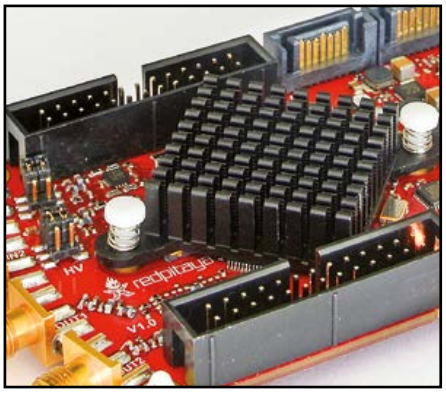


communauté

e-labs

projets





ne doit ni créer de perturbation ni surtout regimber lors d'une variation de consommation, aussi brutale soit-elle. Comme toutes bonnes mamelles qui vont par paire, celle-ci répartit la régulation sur deux étages, un premier à découpage suivi d'un circuit analogique, pour fournir une gamme étendue de 0 à 40 V et 2 A.

industrie

banc d'essai

magazine

37ème année, n° 438
décembre 2014

ISSN 0181-7450
Dépôt légal : novembre 2014
CPPAP 1113 U 83713
Directeur de la publication : Donatus Akkermans

Elektor est édité par :
PUBLITRONIC SARL
c/o Regus Roissy CDG
1, rue de la Haye
BP 12910
FR - 95731 Roissy CDG Cedex

Tél. : (+33) 01.49.19.26.19
du lundi, mardi et jeudi de 8h30 à 17h
le vendredi de 8h30 à 12h30
Fax : (+33) 01.49.19.22.37
www.elektor.fr | www.elektor-magazine.fr

Banque ABN AMRO : Paris
IBAN : FR76 1873 9000 0100 2007 9702 603
BIC : ABNAFRPP

Imprimé aux Pays-Bas
par Senefelder Misset – Doetinchem

Distribué en France par M.L.P.
et en Belgique par A.M.P.

Periculum in mora

Les lecteurs, c'est un peu comme les enfants, plus on en a, moins on a de théories sur la meilleure manière de se comporter avec eux. La pratique et le bon sens dictent de rechercher sans cesse le fragile équilibre entre la satisfaction des besoins et des désirs, et d'autre part l'épanouissement des qualités de chacun dans le respect de sa propre personnalité et de celle des autres. Autant dire que, pour les enfants comme pour les lecteurs, il est vain d'espérer trouver *la* méthode universelle, mais il n'est pas inutile de la chercher.
Ce qui ne laisse pas de me frapper dans ce métier, c'est l'aptitude de certains de nos lecteurs à interpréter ce qu'ils lisent et surtout ce que, par définition, ils ne peuvent pas lire puisque ce n'est pas même écrit. Pourquoi leurs prédictions sont-elles immanquablement alarmistes ? Ils conjecturent et, tels d'effrayants devins, semblent tirer gloire aujourd'hui de prédire les catastrophes qui, selon eux, se passeront demain... La fin prochaine d'Elektor, par exemple, une rengaine dont je connais quelques couplets, tous faux dans leur prétendue prémonition.
Quand les bons vieux transistors furent supplantés, un jour, sans merci, par des circuits intégrés au fonctionnement desquels les anciens craignaient de ne plus rien comprendre — en fait, ils s'y sont bien faits. Quand les microscopiques CMS ont chassé les composants traversants. Quand est apparue la norme « sans plomb », perçue alors comme une menace aujourd'hui bien oubliée. Quand les microcontrôleurs se sont mis à proliférer dans l'espace laissé libre par des microprocesseurs devenus indomptables... Arrêtons là cette litanie qui prend aujourd'hui de nouveaux accents de malédiction avec la grande (r)évolution de l'internet, exécrée à son tour comme catastrophe imminente. Des lecteurs ont relevé dans un récent édito de l'édition anglaise d'Elektor l'annonce de changements qu'ils jugent inquiétants. En effet, mon homologue J. Buiting y informait ses lecteurs anglophones non abonnés que l'année prochaine leur magazine serait disponible **sous forme numérique** et plus en kiosque. Cette mesure de bon sens concerne l'édition anglaise, diffusée dans le monde entier beaucoup plus largement que sur le seul territoire britannique. C'est une très bonne nouvelle pour la plupart de ces lecteurs qu'Elektor peut atteindre au quatre coins du monde. Il n'est pas question d'une telle mesure pour les territoires francophones (suffisamment bien desservis comme la France et la Belgique), mais, si son biotope change, Elektor s'adaptera, car c'est la loi de la vie. Loin de nous effrayer, cette aptitude nous réjouit. Elle est source de motivation et d'innovation. Et pour conclure sur une note allègre, Francis Blanche nous rappelle qu'*il vaut mieux penser le changement que [d'avoir à] changer le pansement.*
Bonne lecture !

Denis Meyer

Notre équipe

Rédacteur en chef :	Denis Meyer (redaction@elektor.fr)
Rédaction internationale :	Harry Baggen, Jan Buiting, Jaime González-Arintero, Jens Nickel
Laboratoire :	Thijs Beckers, Ton Giesberts, Wisse Hettinga, Luc Lemmens, Mart Schroijen, Clemens Valens, Jan Visser, Patrick Wielders
Coordination :	Hedwig Hennekens
Ont coopéré à ce numéro :	Rémi Descistes, Robert Grignard, Hervé Moreau, Kévin Petit, Guy Raedersdorf, NN
Service de la clientèle :	Cindy Tyssen & Vanessa Noville
Graphiste :	Giel Dols
Elektor *en ligne* :	Daniëlle Mertens

France
Denis Meyer
+31 46 4389435
d.meyer@elektor.fr

United Kingdom
Carlo van Nistelrooy
+44 20 7692 8344
c.vannistelrooy@elektor.com

USA
Carlo van Nistelrooy
+1 860-289-0800
c.vannistelrooy@elektor.com

Germany
Ferdinand te Walvaart
+49 241 88 909-17
f.tewalvaart@elektor.de

Netherlands
Ferdinand te Walvaart
+31 46 43 89 444
f.tewalvaart@elektor.nl

Spain
Jaime González-Arintero
+34 6 16 99 74 86
j.glez.arintero@elektor.es

Italy
Maurizio del Corso
+39 2.66504755
m.delcorso@inware.it

Sweden
Carlo van Nistelrooy
+31 46 43 89 418
c.vannistelrooy@elektor.com

Brazil
João Martins
+31 46 4389444
j.martins@elektor.com

Portugal
João Martins
+31 46 4389444
j.martins@elektor.com

India
Sunil D. Malekar
+91 9833168815
ts@elektor.in

Russia
Nataliya Melnikova
+7 (965) 395 33 36
Elektor.Russia@gmail.com

Turkey
Zeynep Köksal
+90 532 277 48 26
zkoksal@beti.com.tr

South Africa
Johan Dijk
+31 6 1589 4245
j.dijk@elektor.com

China
Cees Baay
+86 21 6445 2811
CeesBaay@gmail.com

Notre réseau

elektor · VOICE COIL · circuit cellar · hifi.nl MUZIEK - FILM - LIFESTYLE · audioxpress · techthe future

vous connecte à

Nos annonceurs

Pour placer votre annonce dans le prochain numéro d'Elektor

veuillez contacter Mme Ilham Mohammedi par téléphone au (+31) 6.41.42.25.25
ou par courrier électronique : i.mohammedi@elektor.fr

Vos correspondants

Nous sommes à votre service pour toute question relative à votre commande ou votre abonnement par téléphone au (+33) 01.49.19.26.19 du lundi au jeudi de 8h30 à 17h et le vendredi de 8h30 à 12h30 ou par courriel : service@elektor.fr

MADE IN MUNICH

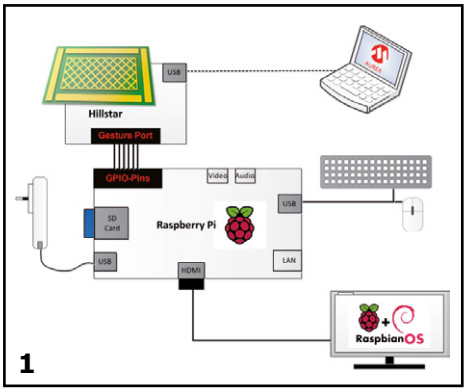
GestIC & 3D TouchPad : manip. n°1

Commande par pichenettes pour Raspberry Pi

Thomas Lindner & **Roland Aubauer** (Microchip GestIC® Team, Germany)
Jan Buiting (Elektor)

Après avoir lancé le mois dernier l'offre exclusive Microchip/Elektor du kit de commande gestuelle avec plaque 3D Touchpad, connectons le contrôleur MGC3130 GestIC® à un ordinateur Raspberry Pi pour en découvrir tout le potentiel.

Available in the Elektor Store



1

L'offre exclusive faite conjointement par Microchip et Elektor [1] réunit un kit d'évaluation Hillstar MGC3130 3D *Gesture Control Development* et une plaque 3D Touchpad prête à l'emploi. Le genre de circuits qu'on a tout de suite envie de connecter pour voir comment ça marche, avec un microcontrôleur ou avec un PC ! Ça tombe bien, le produit est maintenant disponible [2].
Au cours de cette première manip, nous proposons une approche pas-à-pas combinée, pour réaliser une commande tactile (2D) et par gestes (3D, donc sans contact) et — si ça marche — jouer au jeu '2048' avec un ordinateur Raspberry Pi. Comme ce sera une manip, ou des travaux pratiques si vous préférez, et qu'on suppose que vous participez activement, le tempo de la description ci-dessous sera assez rapide. Les informations fournies ici sont surtout censées faire jaillir de nouvelles idées d'utilisation du MGC3130 pour la commande gestuelle 3D et tactile (2D) dans d'autres contextes que celui que nous décrirons.

1. matos

Pour suivre, il faut au moins le matériel suivant :

- Raspberry Pi Model B v.2, RBCA000 … 2x USB 2.0 3,5 W
- Alimentation : micro USB 1200 mA 5 V pour RPi
- Kit d'évaluation MGC3130 Hillstar

La **figure 1** illustre la configuration matérielle photographiée (**fig. 2**).

Ce dispositif de reconnaissance de gestes fonctionne sans PC. Il suffit de l'alimenter au moyen d'un chargeur USB p. ex ou directement depuis le Raspberry Pi. La communication entre les circuits intégrés passe par les broches *GesturePort* de la petite carte MGC3130 incluse dans le kit Hillstar. Un PC n'est nécessaire que pour le paramétrage et pour flasher le progiciel du MGC3130.

2. connexions

La détection capacitive est gérée entièrement par le MGC3130. Les signaux EIO1, EIO2, EIO3, EIO6, EIO7 de la carte Hillstar MGC3130 sont appliqués au connecteur GPIO du Raspberry Pi comme le montre la **figure 3**. N'oubliez pas le potentiel de référence (GND), bien sûr.

3. paramétrage avec Aurea

Les pichenettes est-ouest et nord-sud de la main, reconnues par le MGC3130, se traduisent par les signaux EIOx. Afin d'obtenir un calibrage précis, l'environnement graphique gratuit **Aurea** pour la commande par gestes avec l'interface GestIC permet de paramétrer la taille et la configuration des plans de détection capacitive. Remarquez dans les paramètres de *GesturePort* réunis sur la **figure 4** la combinaison unique en son genre entre pichenettes 3D (*Flick E<-->W; Flick N<-->S*) **et** toucher 2D (*Center*).

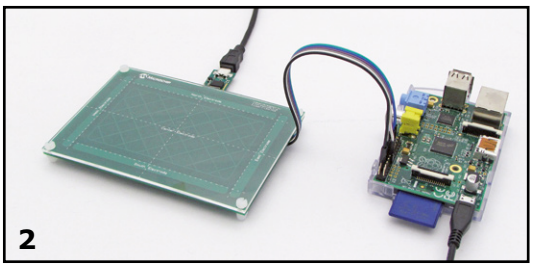
4. programmation

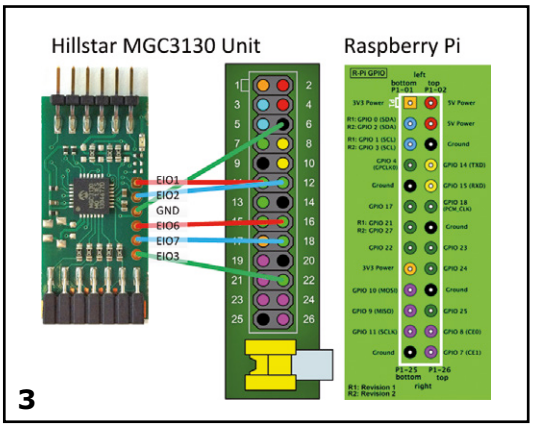
Comme logiciel il nous faudra :

- OS : Raspbian (Debian Wheezy) v. janvier 2014, date : 2014-01-07
- Python | v. 2.7.3 (default, Mar 18 2014, 05:13:23) (déjà installé sous Raspbian)
- bibliothèque RPi.GPIO | v. 0.5.4 (déjà installée sous Raspbian)
- Tkinter (déjà installé sous Raspbian)
- éditeur Leafpad (déjà installé sous Raspbian)
- le code du jeu '2048_with_Gesture_Port_Demo.py' (patience, ce sera pour le mois prochain)

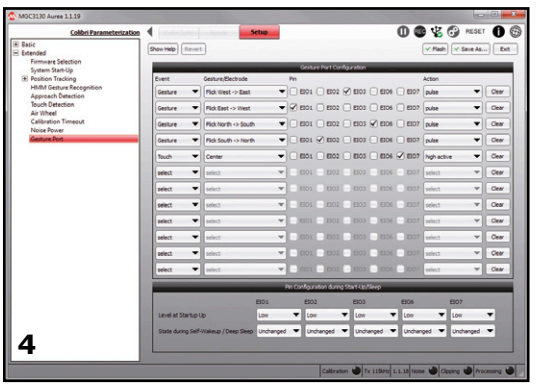
5. c'est parti

1. Installez Raspbian sur votre Raspberry Pi.
2. Connectez votre passerelle Hillstar I²C - USB à *GesturePort Demo*. Puis connectez la passerelle à l'USB de votre PC et lancez *Aurea*. La synchronisation se traduit par l'ouverture d'une fenêtre.
3. Cliquez en haut sur 'Setup' et choisissez 'Parameterization'.
4. Cliquez à gauche sur 'Extended', et cherchez dans 'Parameters' le fichier préconfiguré 'Gesture Port Demo to Raspberry Pi Demo 2048. enz' puis lancez le paramétrage, qui va prendre un peu de temps.
5. Une fois que c'est fini, cliquez à gauche sur 'GesturePort' et vous verrez la configuration (*events*). Si vous cliquez sur 'Flash' dans le coin en haut à droite, la configuration reproduite à la fig. 4 sera inscrite dans le circuit intégré MGC3130.
6. Une fois que sa configuration est flashée, *GesturePort Demo* est autonome. Débranchez la passerelle I²C - USB et connectez la carte *GesturePort Demo* à votre Raspberry Pi comme sur la fig. 1. Vérifiez que *GesturePort Demo* est alimenté par un chargeur USB (mini connecteur USB). Si vous n'en avez pas sous la main, siphonnez l'alimentation sous 5 V du Raspberry Pi.


2



3


4

La prochaine manipulation nous permettra d'approfondir, en nous attaquant notamment au jeu *2048*. Entretemps, s'il y a du nouveau, ce qui est probable, nous reparlerons du kit GestIC dans Elektor.POST, notre lettre d'information électronique hebdomadaire.

(140423 – version française : Rémy Descistes)

Liens

[1] commande par gestes 3D pour µC et PC
Elektor novembre 2014, p.75, www.elektor-magazine.fr/140408

[2] offre groupée exclusive : Microchip Hillstar GestIC dev kit & 3D Touchpad :
www.elektor.fr/microchip-dm160218-hillstar-development-kit-and-dm160225-3d-touchpad

[3] www.raspberrypi.org/downloads/

16e jour : travailler avec les composants

Neil Gruending
(Canada)

Nous avons déjà parlé un peu des composants dans les articles précédents ; entrons dans les détails.

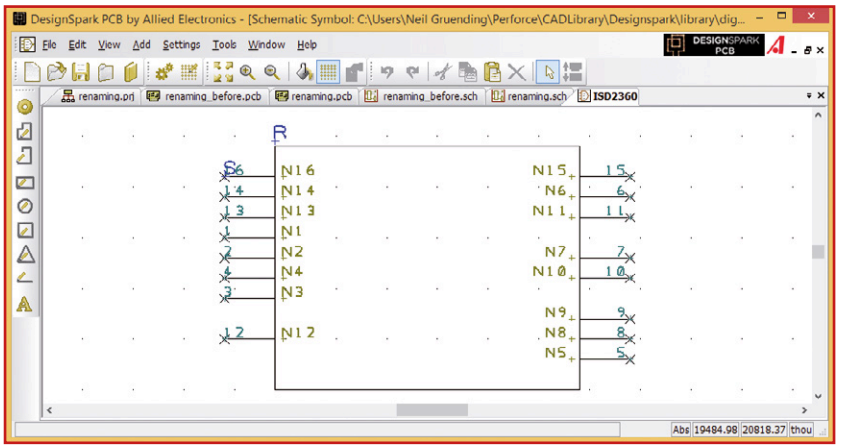


Figure 1.
Le symbole de schéma modifié avec l'assistant.

J'avais déjà parlé de la gestion des composants dans DesignSpark, mais sans aborder les détails d'une utilisation efficace. Dans le 3e volet de cette série (Elektor de sept. 2013) je vous avais parlé des bibliothèques et de la manière dont les composants DS agglutinent toutes les informations de conception d'un composant. C'est ainsi que les outils de conception de schémas et de circuits imprimés (C.I.) retrouvent facilement tout ce dont ils ont besoin.

Créer un composant prend du temps si vous partez d'une page blanche. S'il existe dans la bibliothèque un composant similaire à celui que vous souhaitez créer, il est plus facile de le copier à l'aide du bouton *Copy To* du *Library Manager*, puis de le modifier. Lorsque ce n'est pas possible, les assistants de création de composants de DS sont d'une aide précieuse et permettent d'accélérer l'opération. Pour apprendre comment fonctionnent les assistants, je vous propose de construire un composant modélisant une puce de lecture audio.

Commençons par le symbole en ouvrant l'assistant de schéma de DS : cliquez sur le bouton *Wizard* dans le *Library manager*. Répondez aux quelques questions et il créera automatiquement un symbole que nous pourrons modifier. Après un peu d'édition, je suis arrivé au symbole de la **figure 1**. Les broches sont là où je les veux, mais, si vous y regardez de plus près, vous remarquerez qu'elles possèdent les noms par défaut N1 à N16. Pour l'instant cela suffira ; nous les changerons plus tard au niveau du composant. Une astuce consiste à disposer dès maintenant des chaînes de caractères temporaires sur les noms des broches pour s'assurer que les noms définitifs tiendront dans le symbole.

DS possède également un assistant de création d'empreintes pour C.I. capable de créer pour vous une vaste gamme d'empreintes. Il est même capable de créer des pastilles avec l'espacement qu'il faut. La dernière étape consiste à utiliser l'assistant de composants pour associer le symbole de schéma à l'empreinte pour C.I.

Une fois que vous aurez choisi le symbole et l'empreinte pour C.I. que vous venez de créer, vous utiliserez l'écran d'attribution des broches *Assign Pins* (**fig. 2**) qui permet d'étiqueter chaque broche et d'associer les broches du symbole à l'empreinte pour C.I. Dans mon cas, j'ai conçu le symbole de telle façon que les numéros de broches correspondent à ceux de l'empreinte pour C.I. et il me suffit donc d'utiliser le bouton *Assign 1-to-1* pour réaliser l'association. Mais DS permet d'avoir toutes les associations dont vous pourriez rêver. C'est aussi le moment de saisir tous les noms des broches dans la colonne *Terminal Name*.

Cliquez sur le bouton *Next*, le composant sera créé et vous pourrez l'ajouter à votre bibliothèque. La **figure 3** montre l'aspect de notre composant dans un schéma. Voyons maintenant comment le modifier et le mettre à jour là où il est utilisé.

Lorsque l'on modifie un composant dans une bibliothèque, DS ne met à jour que le fichier de bibliothèque et pas les schémas ni les tracés de circuits imprimés qui l'utilisent. Cela est dû au fait que chacun des fichiers de conception conserve sa propre copie du composant afin que les changements dans une bibliothèque de composants ne puissent affecter accidentellement un projet existant. Comment mettre à jour ces copies locales avec la nouvelle version d'un composant ?

Une manière de faire est de supprimer le composant puis de l'ajouter à nouveau, mais cette opération dangereuse peut avoir des effets inattendus comme casser les liens entre les *nets* de votre projet. C'est particulièrement risqué dans un fichier C.I. où cela pourrait casser les liens avec le schéma. Utilisez plutôt la fonction *Update Components* du menu *Tools* et sélectionnez les composants à mettre à jour. Cela fonctionne dans les fichiers de schéma et de C.I.

Le plus souvent, vous souhaiterez soit parcourir les composants soit mettre à jour les composants sélectionnés. Habituellement il n'est pas question de mettre à jour *tous* les composants automatiquement à moins que vous ne vérifiiez tous les changements à venir pour vous assurer qu'il ne se passera rien d'inattendu. Dans tous les cas vous devrez utiliser la fenêtre *Update Components* (**fig. 4**) pour mettre à jour les composants d'un schéma. Sa contrepartie dans le programme d'édition de circuit imprimé est similaire.

L'option *Only update item if version is different in library* demande à DS de ne mettre à jour un composant que s'il est différent de celui qui se trouve dans la bibliothèque. L'option *Keep value positions* demande à DS d'essayer de conserver les champs visibles des composants à leur emplacement actuel. En général, cela fonctionne, mais vérifiez que rien n'a bougé dans votre dos : même si DS conserve le même point d'ancrage, la justification du texte changera parfois pour reprendre ce qui est spécifié dans la bibliothèque.

Si vous activez l'option *Keep existing component values,* les champs de valeur des composants seront conservés ; sinon les valeurs seront remplacées par celles de la bibliothèque. Cette option est importante lorsque vous modifiez les champs de valeur d'un composant dans la bibliothèque et que vous souhaitez transférer ces changements dans votre projet. Lorsque vous mettez à jour un circuit imprimé, la fenêtre *Update Components* contient également l'option *Remove pad style exceptions*. En général vous la laisserez sélectionnée afin que DS efface les erreurs détectées avec la version précédente des empreintes.

Parfois on a besoin de remplacer ou changer un composant au milieu d'un projet. Par exemple, j'aime avoir un seul composant *résistance* différent pour chaque valeur afin que, dans un projet, je puisse mettre toutes les références du fabricant dans les champs de valeur des composants et en tirer une liste plus tard. Cela implique que je change simplement le composant pour changer la valeur d'une résistance.

Tant que vous n'avez pas placé le composant sur le circuit imprimé, l'effacer pour le remplacer par un nouveau ne pose pas de problème. À l'inverse, si le composant est déjà sur le C.I., DS effacera le composant puis ajoutera le nouveau dans le conteneur des composants à placer.

Une meilleure solution consiste à remplacer un composant sans l'effacer. Pour ce faire, sélectionnez le composant et affichez ses propriétés. Cliquez ensuite sur le bouton *Change* qui ouvrira la fenêtre *Change Component* dans laquelle vous pourrez sélectionner le nouveau composant. Avec cette méthode, DS laissera le composant sur le C.I. et vous n'aurez pas à vous soucier de la correspondance des références.

La prochaine fois, nous verrons comment utiliser les composants à portes multiples et les connexions d'alimentation cachées.

(140367 – version française : Kévin Petit)

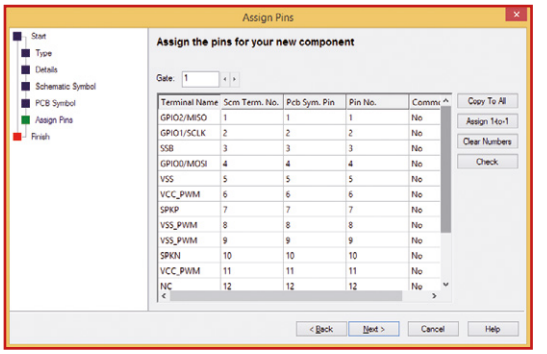


Figure 2.
L'écran d'attribution des broches *Assign Pins*.

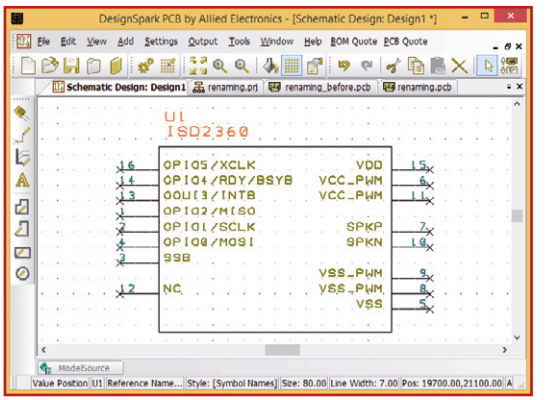


Figure 3.
Le composant terminé.

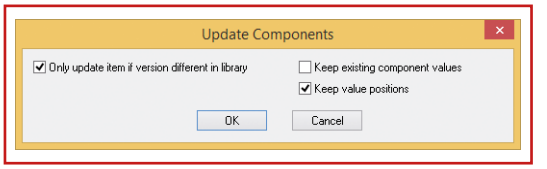


Figure 4.
La fenêtre de mise à jour de composant.

intégrés, mais pas comme les autres : MagI³C-VDRM

Viacheslav Gromov
(Allemagne)

Si je vous dis régulateur intégré, vous pensez plutôt à leurs inconvénients ou à leurs avantages ? Les progrès dans ce domaine sont indéniables, notamment la miniaturisation d'une régulation de plus en plus efficace. Nous ne sommes pas au bout de cette courbe d'amélioration, surtout si l'on s'intéresse aux circuits de faible puissance qui ont encore un bel avenir.
En voici un exemple insigne avec une nouvelle famille de convertisseurs abaisseurs basés sur le principe du VDRM : ***V**ariable Step **D**own **R**egulator **M**odule*.

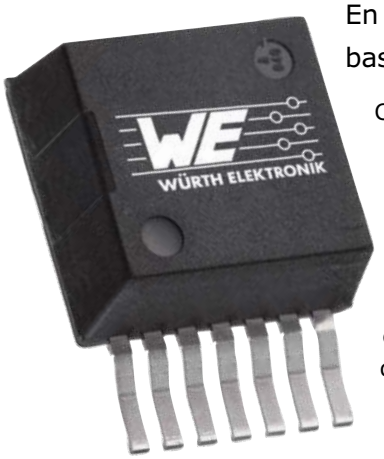


Figure 1.
Les régulateurs en boîtier TO263-7EP

Ces modules de puissance nous sont proposés par le fabricant allemand Würth Elektronik, dont la réputation dans le domaine des composants passifs de très grande qualité est solidement établie. Attention, n'attendez pas du composant bon marché, vous seriez déçu. La qualité se paie. Il faut dire que ces circuits intégrés sont pourvus de nombreux dispositifs de protection, et même de selfs incorporées, ce qui non seulement réduit l'encombrement du circuit dans lesquels les modules sont mis en oeuvre, mais simplifie la conception tout en augmentant la fiabilité. Le rendement de ces composants s'en trouve considérablement amélioré. J'avale ma salive avant de vous parler du prix... 25 € pour trois échantillons. On ne sera pas surpris, du coup, que la carte d'évaluation, bien utile pour tester et expérimenter ces modules, ne soit pas spécialement bon marché non plus.

Une histoire de famille

Cette famille de régulateurs offre des solutions variées autant pour les puissances que pour les tensions (**tableau 1**). Il faut déjà chercher pour trouver un cas de figure qui ne serait pas couvert. Le fabricant propose des brochures d'information gratuites [1] pour guider votre choix en fonction des exigences de votre application. Il y a cinq types en tout, tous en boîtier TO263-7EP (**fig 1**). Le rendement maximal dans cette famille atteint 97 % ; il dépend de divers facteurs, dont la puissance.
Ce qui distingue ces composants de ceux de la famille FDRM (***F**ixed Step **D**own **R**egulator **M**odule*) – hormis la tension de sortie variable et les autres possibilités de réglage des VDRM – c'est essentiellement la présence de condensateurs d'entrée et de sortie intégrés (**fig. 2**). Chacun des types est protégé contre les tensions excessives ou insuffisantes, contre les courants excessifs et les températures supérieures à 165 °C. Toutes ces caractéristiques facilitent la mise en oeuvre.
Le diviseur de tension avec R_{ENT} et R_{ENB} sur l'entrée EN des régulateurs variables VDRM (**fig. 3**) permet d'obtenir la mise hors service du circuit en-dessous d'un certain seuil de tension. Cette fonction dite UVLO, pour ***U**nder **V**oltage **L**ockout,* protègera p. ex. un accumulateur contre une décharge profonde. Le condensateur C_{SS} sur l'entrée SS (*soft start*) détermine la pente du démarrage en douceur. La fréquence de commutation est réglable entre 0,2 et 0,8 MHz à l'aide de R_{ON} sur la broche du même nom, sauf sur le modèle WPMDM1500602JT, qui lui autorise

Tableau 1. Caractéristiques

type	U_{in}	U_{out}	I_{out}	P_{out}	fréquence	self interne	réf. des cartes
WPMDH1200601JT	6 à 42 V	0,8 à 6 V	2 A	12 W	0,2 à 0,8 MHz	10 µH	178 020 601
WPMDH1102401JT	6 à 42 V	5 à 24 V	1 A	24 W	0,2 à 0,8 MHz	15 µH	178 012 401
WPMDM1500602JT	6 à 36 V	0,8 à 6 V	5 A	30 W	0,812 MHz	3,3 µH	178 050 601
WPMDH1152401JT	6 à 42 V	5 à 24 V	1,5 A	36 W	0,2 à 0,8 MHz	15 µH	178 012 402
WPMDH1302401JT	6 à 42 V	5 à 24 V	3 A	72 W	0,2 à 0,8 MHz	10 µH	178 032 401

la synchronisation de plusieurs exemplaires. La broche RON est remplacée par une broche SYNC, à laquelle on injectera une fréquence entre 0,65 et 0,95 MHz issue d'une source extérieure. Quand cette broche est forcée à la masse, la fréquence de commutation est de 0,812 MHz. La tension de sortie est fixée par le diviseur de tension R_{FBT} et R_{FBB} sur l'entrée de réinjection FB (*feedback*). La feuille de caractéristiques [1] indique comment calculer la valeur de ces composants, et donne bien d'autres informations.

La plage de la température de service est vaste : –40 à 125 °C. Pour éliminer ou du moins limiter le bruit, un régulateur ne peut se passer de condensateurs de lissage correctement dimensionnés, aussi bien à l'entrée qu'à la sortie. Leur résistance série équivalente sera, comme sur d'autres types de régulateurs, aussi faible que possible. Se prêtent bien à cette application les condensateurs céramique multicouche à diélectrique X7R ou X5R ou des condensateurs au tantale. Le tableau 1 confirme que les tensions d'entrée maximales admissibles et les fréquences de commutation sont identiques pour tous les modèles, à une exception près, déjà évoquée. Les synoptiques de la fig. 2 montrent comment sont connectés la self intégrée et les condensateurs externes. On y voit aussi les transistors MOSFET intégrés, qui sur des régulateurs à découpage conventionnels seraient des composants discrets.

La fiche technique aborde la question du tracé des pistes du circuit imprimé, en donnant des conseils dont l'application permettra de supprimer ou au moins de limiter les interférences. Le plan de masse des circuits intégrés devra évidemment être au potentiel de la masse. La ligne de réinjection (feedback) devra être aussi courte que possible. Si l'application envisagée se contente des tensions de sortie usuelles, il est préférable d'utiliser un type FDRM à tension fixe, moins encombrant, et dans lequel les condensateurs d'entrée et de sortie sont intégrés. Les cartes d'évaluation (**fig. 4**) seront utiles si on envisage de créer des séries ou pour une étude approfondie des possibilités de ces modules.

(130579 – version française : Rémi Descistes)

Lien

[1] http://katalog.we-online.de/en/pm/MagIC-VDRM

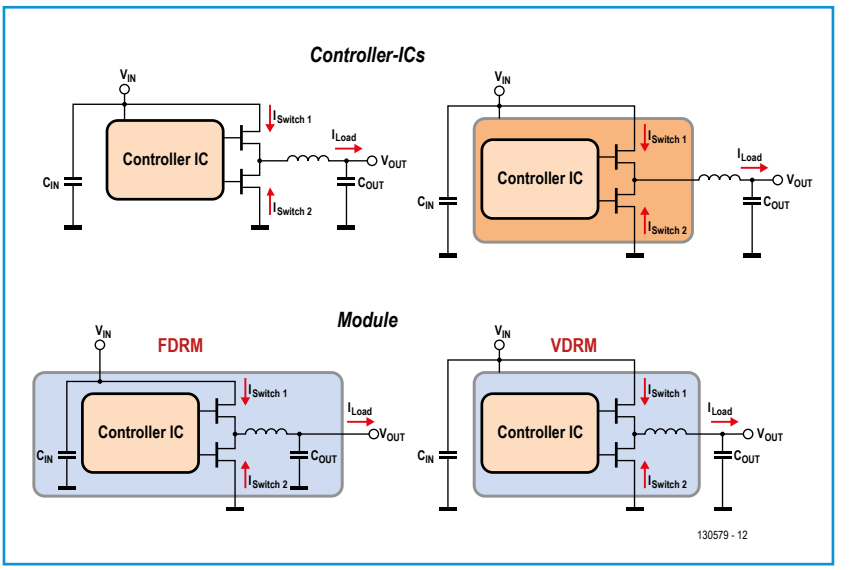


Figure 2. Différences entre FDRM / VDRM et les régulateurs et modules courants.

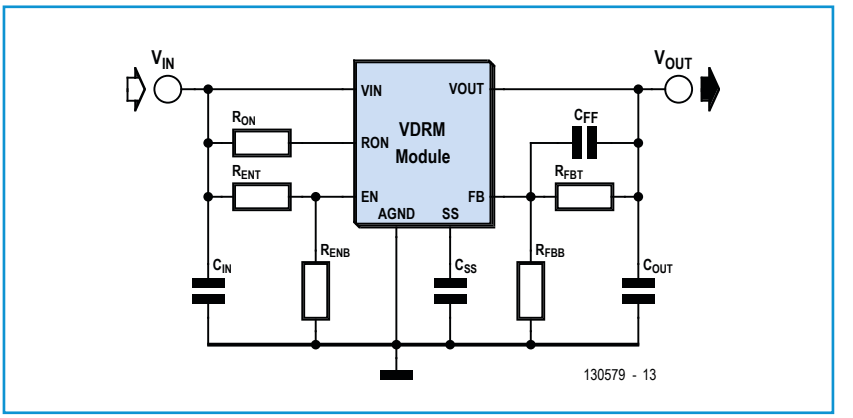


Figure 3. Configuration type du VDRM (sauf pour le WPMDM1500602JT).

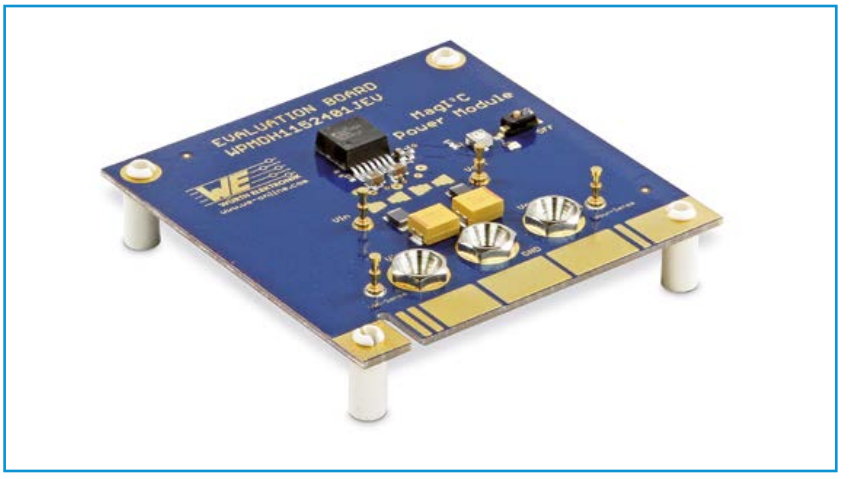


Figure 4. Carte d'évaluation.

commande de LED à la tablette

avec des E-blocks sous Flowcode

Bert van Dam
(Pays-Bas)

Apprenez à commander, à distance et sans fil, une rangée de LED depuis votre tablette ou votre PC où s'affiche d'ailleurs l'état de la commande. C'est facile et vite fait avec quelques modules E-blocks et Flowcode 6.

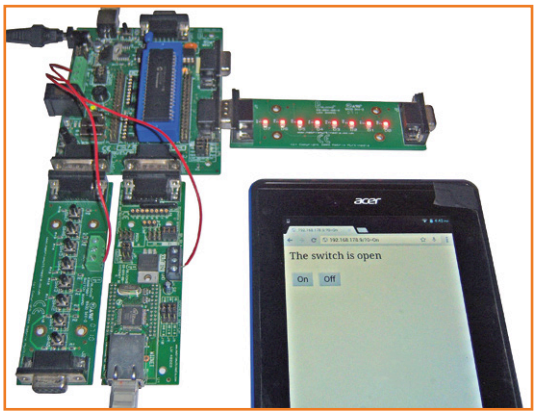


Figure 1.
Commande de l'allumage des LED.

On a déjà allumé et éteint des LED avec tout ce qui est imaginable comme matériel et logiciel. Et pourquoi pas avec une tablette ? Puisque nous disposons des E-blocks, ces modules électroniques qui permettent d'aller droit au but, n'attendons plus ! Notre logiciel graphique, ce sera évidemment Flowcode. Il nous faudra Flowcode V6 pour PIC, le multiprogrammateur de PIC EB006 avec un PIC 18F4620, une carte de LED EB004, une carte à boutons-poussoirs EB007, une carte internet EB023 et un câble Ethernet. Nous allons construire un internet local, un intranet à la maison.

Son intranet à soi

Commençons par raccorder à notre (modem) routeur la carte internet EB023 par un câble Ethernet. La plupart des appareils demandent eux-mêmes au routeur une adresse IP, mais avec un EB023, il faut le faire soi-même. C'est une adresse IP statique (elle ne changera donc pas). La manière la plus simple, c'est de s'identifier sur le routeur pour voir quelle adresse lui attribuer. Les trois premiers groupes de chiffres doivent rester les mêmes, chez moi, c'est 192.168.178, à vous de choisir le dernier chiffre, au moins 2, puisque le routeur est toujours le n°1. Le maximum possible dépend du routeur. Sur le mien, le maximum est réglable, il est sur 9. Si vous utilisez plusieurs adresses IP statiques, dressez-en la liste (comme celle du **tableau 1**) pour ne pas attribuer deux fois la même.
Pour la tablette, rien de particulier à faire, elle est normalement en liaison par WiFi avec le routeur. Si vous n'avez pas de tablette, ça marche aussi avec un téléphone tactile ou un PC.

Construction du projet

Comme nous utilisons des E-blocks (**fig. 1**), l'assemblage est simple et instantané.

- **Programmateur :** EB006. Commutateurs sur XTAL et Fast, cavalier LVP sur I/O-port, J29 sur PSU, J12-14 sur USB, utilisez une alimentation externe. Remarquez que c'est un PIC 18F4620 qui est utilisé, pas le PIC standard dont la mémoire est trop exiguë.

Tableau 1. Adresses IP statiques sur le routeur de l'auteur.

adresse IP	appareil
192.168.178.9	Flowcode EB023
192.168.178.8	ARM mbed
192.168.178.7	Raspberry Pi sur Wi-Fi

Tableau 2. Adresses IP dans le composant serveur web.

adresse IP	objet
192.168.178.1	passerelle (le routeur)
255.255.255.0	masque de sous-réseau
192.168.178.9	EB023

- **Port B** : EB004, carte à LED.
- **Port C** : EB023, carte internet. Cavaliers sur A, +5 V, adresse sur 1-1-1, câblée au réseau par le routeur. Cordon d'alim vers le programmateur.
- **Port D** : EB007, carte à boutons-poussoirs. Cordon d'alim vers le programmateur.

Mode opératoire logiciel

Le programme a été rédigé en Flowcode 6. C'est un bus I²C qui relie la carte internet EB023 au microcontrôleur. Le programme se sert du composant serveur web pour les déclarations (**figure 3**) de la carte EB023. La mise en place des cavaliers 1-1-1 pour fixer l'adresse I²C se traduit ici par *high-high-high*.
Il faut aussi introduire les données IP (**tableau 2**) telles qu'elles l'ont été dans le routeur. Pour l'adresse MAC, mieux vaut adhérer à la proposition par défaut qui figure dans le composant serveur web.

L'étape suivante consiste à créer une page qui doit apparaître sur la tablette dès l'établissement de la liaison avec la carte EB023. Le but est d'allumer et d'éteindre les LED au moyen de deux touches sur l'écran de la tablette, mais aussi d'y voir l'état d'un interrupteur.
Dès qu'on passe sur l'adresse IP du EB023, le composant serveur web envoie une page avec deux boutons. Avec l'un d'eux, on appelle (par « _self ») ladite page, puis derrière l'adresse IP, vient le nom du bouton et sa valeur. Au premier bouton, donnons le nom « 0 » et la valeur « On ». L'entrée est donc « ?0=On ». Au complet, la demande au serveur s'écrit « 192.168.178.9/?0=On ». Demander en Flowcode au composant serveur web *GetInValue(0)* permet d'obtenir en réponse la valeur (en chaîne de caractères) de la variable avec l'indice 0 sur la ligne d'adresse, dans ce cas-ci « On ». Valeur qui sera utilisée pour allumer les LED.

Comme on ne doit rien faire d'autre avec ces boutons sur la page internet, on peut attribuer aussi au second bouton le nom « 0 », mais maintenant avec la valeur « Off » pour éteindre les LED.
Sur une tablette, la taille des caractères par défaut est petite ; il faut donc agrandir les boutons : *width:100px;height:50px;* et choisir une police plus grande : *font-size: 18pt ;*
Pour envoyer à la tablette une donnée, comme l'état de l'interrupteur, mettez un index en code

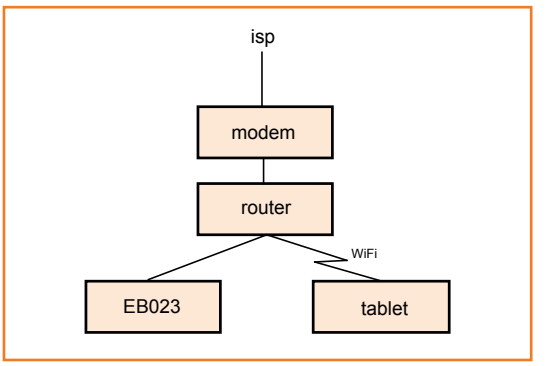


Figure 2.
Le module EB023 et la tablette communiquent tous deux avec le routeur, l'un par câble, l'autre sans fil.

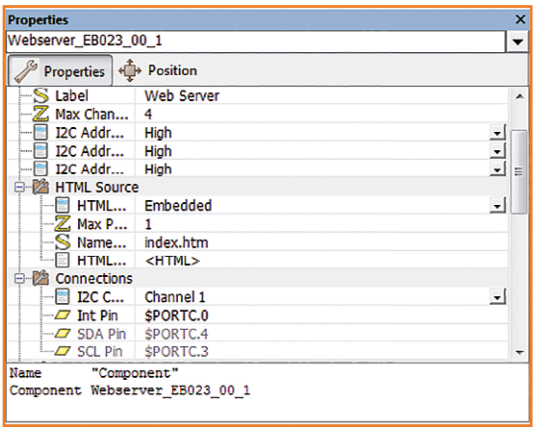


Figure 3.
Les déclarations du bus I²C pour le composant serveur web.

HTML, précédé d'un signe pour-cent. Voulez-vous donner l'état de l'interrupteur, utilisez la phrase « The switch is %1 » au sein de laquelle une macro remplacera %1 par la vraie valeur.

```
<HTML>
<FORM>
<P></P>
<P STYLE=»font-size: 18pt;»>The switch is
   %1</P>
<P><INPUT NAME=»0» TYPE=»submit» STYLE
   =»width:100px;height:50px;font-size:
   18pt» VALUE=»On» onClick=»window.
   open(‘’,’_self’)»>
<INPUT NAME=»0» TYPE=»submit» STYLE=»
   width:100px;height:50px;font-size:
   18pt» VALUE=»Off» onClick=»window.
   open(‘’,’_self’)»>
</P>
</FORM>
</HTML>
```

Saisissez ce code en HTML source dans la case HTML de la figure 3. Pour gagner de la place, nous ne respecterons pas toutes les conventions du

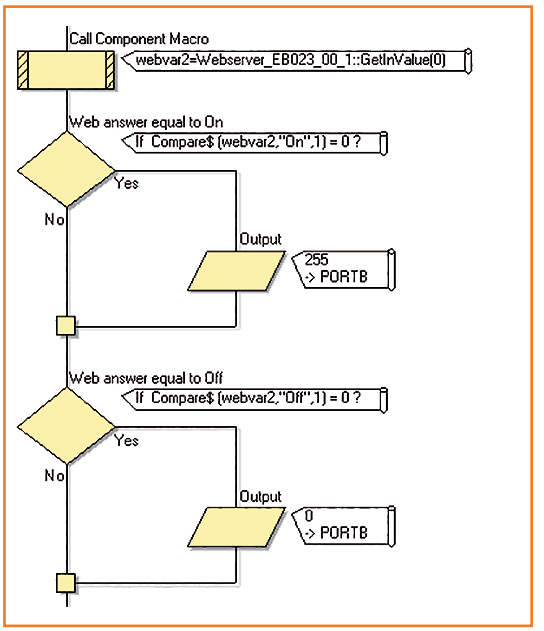


Figure 4.
En Flowcode, la réception de données envoyées par la tablette.

langage (il n'y a ni en-tête ni corps) mais cela, les navigateurs modernes l'acceptent.
L'utilisation est toute simple. Au démarrage, le programme ouvre un *socket* : c'est comme un téléphone. Pour communiquer, les deux parties ont besoin chacune d'un téléphone et les deux doivent être connectés ensemble. C'est pareil avec les sockets. Le EB023 ouvre un socket et écoute. Quand la tablette a établi la liaison, son socket est relié à celui du EB023 et la communication a lieu. Plus tard, dans la boucle principale du programme, la macro *CheckSocketActivity* vérifiera s'il s'y passe quelque chose, auquel cas, ce sera traité automatiquement.

La seule chose qu'il nous reste à faire, c'est d'aller voir avec une macro *GetValue(0)* s'il y a une nouvelle valeur, indexée 0, arrivée de la tablette et par *SetOutValue(1,stand)* remplacer le %1 par l'état de l'interrupteur.

Le maniement

Attendez que le programme soit lancé et laissez encore quelques secondes à EB023 et au routeur pour établir la liaison. Avec un navigateur sur la tablette, allez alors à l'adresse de EB023 ; chez moi c'est 192.168.178.9. La page internet se charge. Touchez l'un des boutons pour allumer ou éteindre les LED.
La page publie aussi l'état de l'interrupteur SW1 sur la carte à boutons-poussoirs EB007. Appuyez sur ce bouton-poussoir-là puis touchez l'un des boutons virtuels sur l'écran de la tablette pour rafraîchir la page de manière à ce que le nouvel état de SW1 soit affiché.

Vous pouvez aussi déclarer à votre routeur que EB023 doit être joignable par internet. Du coup, où que vous soyez dans le monde, vous pourrez commander vos LED et l'état de l'interrupteur affiché sur la tablette confirmera qu'elles sont allumées ou éteintes.

Bien sûr, les sempiternelles LED, c'était juste pour la démonstration. Remplacez-les par un relais, et vous pourrez, depuis votre tablette, commander à distance n'importe quel appareil chez vous. Attention, la carte à relais des E-blocks n'est pas compatible avec la tension du secteur, il lui faudrait une interface.

Matériel et logiciel

Le matériel nécessaire et le code source du programme en Flowcode, vous les trouverez sur les liens [1] et [2].
À découvrir aussi, d'autres idées dans mon livre en anglais « FLOWCODE V6, *30 projects for PIC microcontrollers* ».

(140351 – version française : Robert Grignard)

Liens

[1] www.elektor.fr/devtools/eblocks **et** www.elektor.fr/devtools/flowcode-1-user/
[2] www.elektor-magazine.fr/140351

L'auteur

Bert van Dam est auteur de livres, de matériel didactique et d'articles sur les microcontrôleurs PIC et ARM, Arduino, Raspberry Pi, PC, l'intelligence artificielle et les langages de programmation JAL, C, assembleur, Python et Flowcode.

condensateurs *goutte* au tantale
drôle de composant n° 11

Neil Gruending
(Canada)

Que vérifiez-vous en premier quand un appareil cesse de fonctionner ? Je parie que c'est les tensions d'alimentation et, si vous réparez une carte de plus de vingt ans, les condensateurs *goutte* au tantale. Les jours de chance, ils seront simplement en court-circuit et la limitation de courant de l'alimentation aura fait son travail. Les jours de malchance, ils explosent en petites boules de feu ! Mais pourquoi tant de haine ?

Le condensateur *goutte* (**fig. 1**) est un électrolytique dont le diélectrique est de la poudre de tantale oxydé compressée en une petite boulette. Ce procédé donne des condensateurs de capacité élevée pour leur taille. Comme le tantale est un solide, la gamme de température est très étendue. La propriété la plus intéressante de ces condos est sans doute leur très faible résistance série équivalente (ESR) comparée à celle d'autres électrolytiques comme les modèles à l'aluminium. La poudre de tantale est un diélectrique à cause de la couche d'oxyde isolant qui se forme lors de la fabrication. L'épaisseur de cette couche est choisie en appliquant au condensateur une tension polarisée de plusieurs fois la tension maximale d'utilisation, appelée tension de formation. C'est cette tension qui déterminera la polarité du condensateur. La couche d'oxyde est critique pour le bon fonctionnement d'un condensateur au tantale.

Ce sont leur capacité élevée et leur faible ESR qui font des condensateurs au tantale un bon choix pour les alimentations ; ils ont longtemps été utilisés pour remplacer les petits condensateurs à l'aluminium. Ce n'était pas forcément le choix le plus sage : les condensateurs au tantale sont sensibles aux transitoires de tension et aux courants d'ondulation. Ils peuvent absorber des petites transitoires de tension dans leur gamme de fonctionnement, mais si le pic est plus élevé, il percera la fine couche d'oxyde sur la poudre de tantale et endommagera le condensateur. La faible ESR peut entraîner l'amplification des crêtes (oscillations de dépassement) et produire des pics de courant dévastateurs. Une fois endommagé, un condensateur se dégradera jusqu'au court-circuit. Si l'alimentation est suffisamment puissante, il peut finir par exploser et mettre le feu au circuit imprimé.

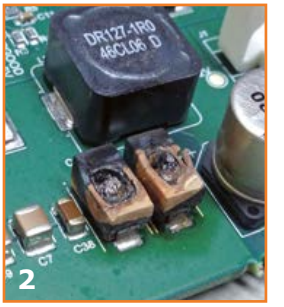
Les condensateurs au tantale peuvent être utilisés en toute sécurité tant que vous respectez les limites de tension et de courant. Alors qu'il est normalement possible d'utiliser un condensateur à l'aluminium près de sa tension maximale, il faudra garder une marge d'au moins 50 % avec un condensateur au tantale. Assurez-vous également que le courant d'ondulation ne dépasse pas les spécifications en ajoutant suffisamment de condensateurs en parallèle. La mise en parallèle réduira toutefois également l'ESR du condensateur et pourra créer un petit circuit bouchon avec l'inductance des pistes du C.I. et produire des pics de tension inattendus dans les applications d'alimentation.

De la fin des années 80 jusqu'au milieu des années 90, les condensateurs au tantale, en boîtier goutte, recouverts d'époxy ou à monter en surface, ont été pas mal utilisés dans les alimentations et pour le découplage. On sait aujourd'hui qu'il en a résulté un taux de pannes élevé (**fig. 2**). Lorsque l'on répare un appareil des années 80/90, on commence souvent par tester ou remplacer les condensateurs au tantale. Difficile de connaître la cause précise des pannes, mais si l'on en croit Kemet [1] les premiers condensateurs au tantale n'étaient pas prévus pour les applications d'alimentation et nécessitaient une résistance externe significative pour fonctionner de manière fiable (habituellement plusieurs ohms par volt aux bornes du condo). Ou bien beaucoup des projets conçus à cette époque utilisaient les condensateurs trop près de leur tension maximale et ils tombaient en panne simplement à cause d'une multitude de petites transitoires. Il était p. ex. courant d'utiliser des condensateurs 6,3 V pour des rails d'alimentation 5 V.

Les condensateurs au tantale sont moins utilisés dans les nouveaux projets, sans doute grâce aux céramique MLCC et aux meilleurs électrolytiques à l'aluminium, mais ils restent imbattables s'il vous faut une grande capacité dans un petit boîtier. Les condensateurs au tantale modernes, testés pour les pics de tension et munis de fusibles, ont résolu les problèmes décrits ici, mais cela ne vous dispense pas de respecter les spécifications.

(140366 – version française : Kévin Petit)

Source: Wikipedia



Lien : [1] http://canada.newark.com/pdfs/techarticles/kemet/Tantalum-in-Power-Supply-Applications.pdf

module *Compute* pour Raspberry Pi

une framboise avec encore plus de jus

Tony Dixon
(Royaume-Uni)

Pionnière dans son genre, peu chère et puissante, la plateforme Raspberry Pi a permis à bon nombre d'étudiants et amateurs d'entrer dans le monde de l'informatique embarquée. La carte a également conquis les professionnels puisqu'elle est au cœur de nombreux projets commerciaux. Consciente de ce fort potentiel commercial, la fondation Raspberry Pi a créé le module *Compute Module*, ainsi qu'une carte pour en exploiter les E/S.

Avec seulement deux puces, le module *CM* incarne la simplicité : la première est la puce Broadcom 2835 dotée de 512 Mo de RAM du modèle B du Pi, la seconde est une mémoire flash eMMC de 4 Go. Ces deux puces sont montées sur un module SO-DIMM à 200 broches semblable à celui des mémoires DDR2 des ordinateurs portables. Les caractéristiques de ce *Compute Module* sont données dans le **tableau 1**.

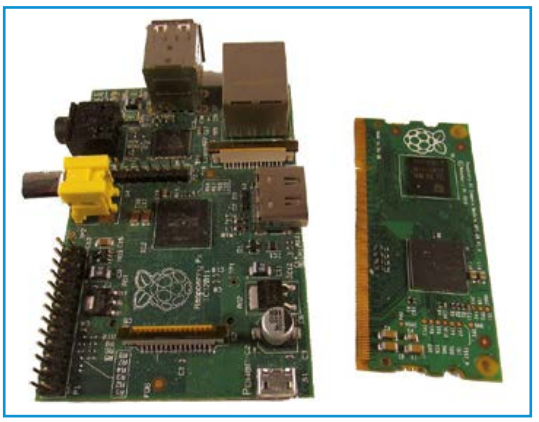


Figure 1.
À gauche l'ordinateur Raspberry Pi, à droite le module *Compute*.

Tableau 1. Caractéristiques du module *CM*

Système sur puce	Broadcom BCM2835
Architecture	ARM1176JZF-S
Horloge	700 MHz
GPU	VideoCore IV GPU
RAM	512 Mo
Interfaces	HDMI, TV Out, USB, 2x DSI LCD, 2x CSI caméra
Dimensions	67,6 mm x 30 mm (SODIMM)

Tableau 2. Fonctions GPIO supplémentaires du module *CM*

Nombre	Fonction
1	interface carte SD
2	UART (une avec signaux CTS et RTS)
2	I²C
2	SPI (une avec 2 sélections de puce, l'autre à 3 sélections de puce)
2	sorties MLI/PWM
1	I²S/PCM
1	adresse secondaire/bus de données
3	sorties d'horloge à usage général
46	GPIO

Se débarrasser des connecteurs au profit d'une empreinte SO-DIMM a permis de fabriquer un module beaucoup plus petit que le Raspberry Pi (**fig. 1**). La plateforme gagne également en polyvalence grâce à des accès à quasiment tous les signaux de la Broadcom 2835, y compris à la majorité des signaux GPIO. Au total, ces 46 signaux GPIO donnent accès à six interfaces supplémentaires (**tableau 2**).

Des GPIO à gogo

Les fonctions des ports GPIO du module *CM* sont reprises dans le **tableau 3**. On y retrouve les signaux des modèle A et B du Pi, autrement dit il est possible d'utiliser les cartes d'extension Pi existantes avec le module *CM*. Les signaux et interfaces supplémentaires élargissent les possibilités de commande et de contrôle.
Le **tableau 4** donne le brochage du connecteur SO-DIMM du module *CM*. Outre les signaux GPIO, on retrouve là aussi les ports standards du Pi : HDMI, sortie TV, USB, DSI (*display serial interface*) et CSI (*camera serial interface*), avec toutefois un port CSI supplémentaire et une seconde interface DSI. Le module peut être enfiché sur une autre nouveauté, la carte *Compute Module I/O* (**fig. 2**).

Le module *CM* nécessite trois alimentations séparées de 3,3 V, 1,8 V et 2,5 V, idéalement des

alimentations à découpage. Celle de 2,5 V n'est pas nécessaire si la sortie TV n'est pas utilisée, ce qui simplifie un peu les choses. Vous trouverez une documentation détaillée des deux cartes sur le site GitHub [1].
Le module *CM* est autonome en ce sens qu'il s'amorce depuis la mémoire eMMC.

Conclusion

Le module *Compute Module* et la carte d'E/S associée agrandissent de façon heureuse et

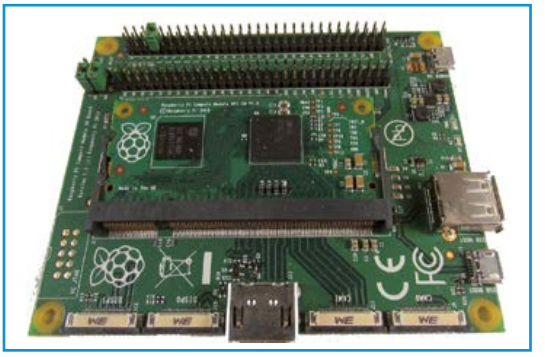


Figure 2.
Pour être exploitables, autant de lignes regroupées sur les 200 broches d'un si petit connecteur SO-DIMM nécessitaient une carte spéciale. Relié comme ici à la carte CM IO, le module CM devient le petit bras musclé d'une sorte de Super Framboise.

Tableau 3. Fonctions GPIO du module *CM*

GPIO ID	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO0	SDA0	SA5	<réservée>			
GPIO1	SCL0	SA4	<réservée>			
GPIO2	SDA1	SA3	<réservée>			
GPIO3	SCL1	SA2	<réservée>			
GPIO4	GPCLK0	SA1	<réservée>			ARM_TDI
GPIO5	GPCLK1	SA0	<réservée>			ARM_TDO
GPIO6	GPCLK2	SOE_N/SE	<réservée>			ARM_RTCK
GPIO7	SPI0_CE1_N	SWE_N/SRW_N	<réservée>			
GPIO8	SPI0_CE0_N	SD0	<réservée>			
GPIO9	SPI0_MISO	SD1	<réservée>			
GPIO10	SPI0_MOSI	SD2	<réservée>			
GPIO11	SPI0_SCLK	SD3	<réservée>			
GPIO12	PWM0	SD4	<réservée>			ARM_TMS
GPIO13	PWM1	SD5	<réservée>			ARM_TCK
GPIO14	TXD0	SD6	<réservée>			TXD1
GPIO15	RXD0	SD7	<réservée>			RXD1
GPIO16	<réservée>	SD8	<réservée>	CTS0	SPI1_CE2_N	CTS1
GPIO17	<réservée>	SD9	<réservée>	RTS0	SPI1_CE1_N	RTS1
GPIO18	PCM_CLK	SD10	<réservée>	BSCSL_SDA/MOSI	SPI1_CE0_N	PWM0
GPIO19	PCM_FS	SD11	<réservée>	BSCSL_SCL/SCLK	SPI1_MISO	PWM1
GPIO20	PCM_DIN	SD12	<réservée>	BSCSL/MISO	SPI1_MOSI	GPCLK0
GPIO21	PCM_DOUT	SD13	<réservée>	BSCSL/CE_N	SPI1_SCLK	GPCLK1
GPIO22	<réservée>	SD14	<réservée>	SD1_CLK	ARM_TRST	
GPIO23	<réservée>	SD15	<réservée>	SD1_CMD	ARM_RTCK	
GPIO24	<réservée>	SD16	<réservée>	SD1_DAT0	ARM_TDO	
GPIO25	<réservée>	SD17	<réservée>	SD1_DAT1	ARM_TCK	
GPIO26	<réservée>	<réservée>	<réservée>	SD1_DAT2	ARM_TDI	
GPIO27	<réservée>	<réservée>	<réservée>	SD1_DAT3	ARM_TMS	
GPIO28	SDA0	SA5	PCM_CLK	<réservée>		
GPIO29	SCL0	SA4	PCM_FS	<réservée>		
GPIO30	<réservée>	SA3	PCM_DIN	CTS0		CTS1
GPIO31	<réservée>	SA2	PCM_DOUT	RTS0		RTS1
GPIO32	GPCLK0	SA1	<réservée>	TXD0		TXD1
GPIO33	<réservée>	SA0	<réservée>	RXD0		RXD1
GPIO34	GPCLK0	SOE_N/SE	<réservée>	<réservée>		
GPIO35	SPI0_CE1_N	SWE_N/SRW_N	<réservée>			
GPIO36	SPI0_CE0_N	SD0	TXD0	<réservée>		
GPIO37	SPI0_MISO	SD1	RXD0	<réservée>		
GPIO38	SPI0_MOSI	SD2	RTS0	<réservée>		
GPIO39	SPI0_SCLK	SD3	CTS0	<réservée>		
GPIO40	PWM0	SD4		<réservée>	SPI2_MISO	TXD1
GPIO41	PWM1	SD5	<réservée>	<réservée>	SPI2_MOSI	RXD1
GPIO42	GPCLK1	SD6	<réservée>	<réservée>	SPI2_SCLK	RTS1
GPIO43	GPCLK2	SD7	<réservée>	<réservée>	SPI2_CE0_N	CTS1
GPIO44	GPCLK1	SDA0	SDA1	<réservée>	SPI2_CE1_N	
GPIO45	PWM1	SCL0	SCL1	<réservée>	SPI2_CE2_N	

bienvenue la famille des produits Raspberry Pi. Avec sa petite surface embroussaillée de broches GPIO et d'interfaces, le module CM est idéal pour construire une carte et un boîtier parfaitement adaptés à la taille d'un projet, en particulier lorsque l'encombrement du Raspberry Pi ne convient pas. Sans oublier les E/S supplémentaires, véritable atout pour les projets matériels ambitieux, pour ne pas dire soutien indispensable.

(140320 – version française : Hervé Moreau)

Référence

[1]. Guide de conception matérielle du Compute Module : https://github.com/raspberrypi/documentation/tree/master/hardware/computemodule

Tableau 4. Brochage du connecteur SO-DIMM du module *CM*

GND	1	2	EMMC_DISABLE	DSI0_DP0	101	102	DSI1_CN
GPIO0	3	4	non reliée	GND	103	104	GND
GPIO1	5	6	non reliée	DSI0_CN	105	106	DSI1_DP3
GND	7	8	non reliée	DSI0_CP	107	108	DSI1_DN3
GPIO2	9	10	non reliée	GND	109	110	GND
GPIO3	11	12	non reliée	HDMI_CK_N	111	112	DSI1_DP2
GND	13	14	non reliée	HDMI_CK_P	113	114	DSI1_DN2
GPIO4	15	16	non reliée	GND	115	116	GND
GPIO5	17	18	non reliée	HDMI_D0_N	117	118	DSI1_DP1
GND	19	20	non reliée	HDMI_D0_P	119	120	DSI1_DN1
GPIO6	21	22	non reliée	GND	121	122	GND
GPIO7	23	24	non reliée	HDMI_D1_N	123	124	non reliée
GND	25	26	GND	HDMI_D1_P	125	126	non reliée
GPIO8	27	28	GPIO28	GND	127	128	non reliée
GPIO9	29	30	GPIO29	HDMI_D2_N	129	130	non reliée
GND	31	32	GND	HDMI_D2_P	131	132	non reliée
GPIO10	33	34	GPIO30	GND	133	134	GND
GPIO11	35	36	GPIO31	CAM1_DP3	135	136	CAM0_DP0
GND	37	38	GND	CAM1_DN3	137	138	CAM0_DN0
GPIO0-27_VREF	39	40	GPIO0-27_VREF	GND	139	140	GND
GPIO28-45_VREF	41	42	GPIO28-45_VREF	CAM1_DP2	141	142	CAM0_CP
GND	43	44	GND	CAM1_DN2	143	144	CAM0_CN
GPIO12	45	46	GPIO32	GND	145	146	GND
GPIO13	47	48	GPIO33	CAM1_CP	147	148	CAM0_DP1
GND	49	50	GND	CAM1_CN	149	150	CAM0_DN1
GPIO14	51	52	GPIO34	GND	151	152	GND
GPIO15	53	54	GPIO35	CAM1_DP1	153	154	non reliée
GND	55	56	GND	CAM1_DN1	155	156	non reliée
GPIO16	57	58	GPIO36	GND	157	158	non reliée
GPIO17	59	60	GPIO37	CAM1_DP0	159	160	non reliée
GND	61	62	GND	CAM1_DN0	161	162	non reliée
GPIO18	63	64	GPIO38	GND	163	164	GND
GPIO19	65	66	GPIO39	USB_DP	165	166	TVDAC
GND	67	68	GND	USB_DM	167	168	USB_OTGID
GPIO20	69	70	GPIO40	GND	169	170	GND
GPIO21	71	72	GPIO41	HDMI_CEC	171	172	VC_TRST
GND	73	74	GND	HDMI_SDA	173	174	VC_TDI
GPIO22	75	76	GPIO42	HDMI_SCL	175	176	VC_TMS
GPIO23	77	78	GPIO43	RUN	177	178	VC_TDO
GND	79	80	GND	VDD_CORE	179	180	VC_TCK
GPIO24	81	82	GPIO44	GND	181	182	GND
GPIO25	83	84	GPIO45	1V8	183	184	1V8
GND	85	86	GND	1V8	185	186	1V8
GPIO26	87	88	GPIO46_1V8	GND	187	188	GND
GPIO27	89	90	GPIO47_1V8	VDAC	189	190	VDAC
GND	91	92	GND	3V3	191	192	3V3
DSI0_DN1	93	94	DSI1_DP0	3V3	193	194	3V3
DSI0_DP1	95	96	DSI1_DN0	GND	195	196	GND
GND	97	98	GND	VBAT	197	198	VBAT
DSI0_DN0	99	100	DSI1_CP	VBAT	199	200	VBAT

elektor.labs & l'électronique mettable

Clemens Valens
(elektor.labs)

Ôte-toi de là que je m'y mette

C'est drôle de voir Intel, le plus grand fabricant de µP au monde, s'intéresser aux fans d'électronique. Non seulement *Big Blue* a sorti la V2 de son Galileo et sa nouvelle carte Edison compatible Arduino, mais il parrainait aussi le *Maker Faire* de Rome évoqué ci-dessus. Il en a même profité pour y faire une annonce qui n'a pas eu l'écho mérité : Intel entend devenir un acteur principal dans le domaine de l'électronique mettable, celle que l'on met comme des chaussettes ou une montre… ou le bracelet d'Intel, appelé MICA, pour *My Intelligent Communication Accessory*. Intel

stimule des jeunes pousses et le partenariat avec des créateurs de mode pour étendre le marché de ces appareils en offrant des fonctions plus variées que les classiques fonctions biologiques (rythmique cardiaque et température). Voyez le *Mimo Baby Monitor*, basé sur Edison.

Elektor est déjà dans la course*. C'est pourquoi nous lançons ici un appel à projets d'**électronique mettable** à publier sur Elektor.Labs. Si vous avez des projets en cours, des idées, postez-les sur Elektor.Labs et nous verrons ensemble où ça peut mener. Nous mettrons les bouchées doubles pour en faire des produits… mettables !

www.elektor-labs.com/wearable

** vous vous souvenez du **compteur de longueur de natations** de Michel Kueneman dans le nº410 d'Elektor juillet-août 2012, p.21, une application mettable, exemplaire de la synthèse vocale, de l'accéléromètre et du magnétomètre. Ça donne des idées, non ?*

impression 3D — vous reprendrez bien des spaghettis

Après avoir rouspété, il y a quelques mois, sur l'intérêt généralement limité de l'impression 3D, j'avais suggéré (en riant) aux fans de l'impression 3D d'imprimer leur propre filament, ce qui mettrait un frein au gaspillage. Au *Maker Faire* tenu début octobre 2014 à Rome, j'ai de nouveau constaté la prolifération d'objets imprimés en 3D, toujours aussi dépourvus d'intérêt pour la plupart. J'y ai quand même vu des projets qui tenaient la route, mais pas beaucoup. Au programme de la conférence inaugurale, il y avait l'imprimante 3D des gagnants d'un concours de conception organisé par le magazine italien *Focus*. Et devinez ce qu'il imprime, leur engin (photos ci-contre à gauche) ? Du filament d'imprimante 3D ! C'est un peu comme une imprimante qui imprimerait de l'encre pour imprimer… Les Italiens jouaient à domicile ; ils ont donc mis le paquet pour épurer la silhouette de leur engin qui ne ressemble en rien aux horribles imprimantes 3D habituelles. En haut vous mettez le polymère brut, en bas sort le filament, sur une bobine amovible. Je le verrais bien dans ma cuisine. J'ai idée qu'il y aurait moyen d'imprimer des spaghettis avec.

www.elektor-labs.com/3d-printing

(140365)

Red Pitaya : pulpeux & juteux
bien plus qu'un oscilloscope USB !

Martin Oßmann
(Allemagne)

Red Pitaya est une plateforme de mesure *open source*, petite, compacte, simple, intuitive, puissante, polyvalente, configurable, compatible avec des applications prêtes à l'emploi ou développées soi-même. Alléchant, non ?
Alors épluchons ce pitaya pour voir ce qu'il a dans la pulpe !

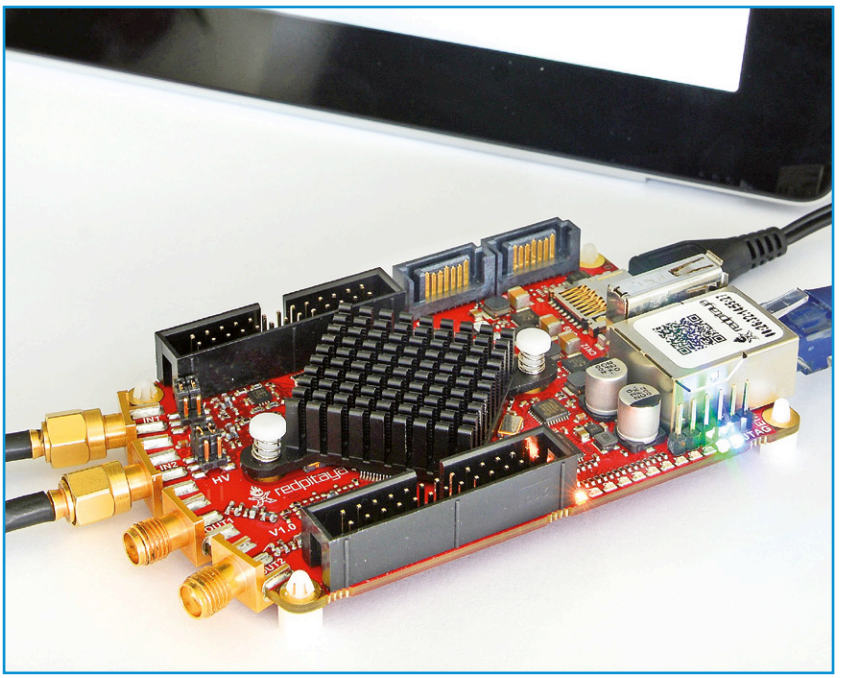


Figure 1.
Le Red Pitaya. Petit, compact, et rouge.

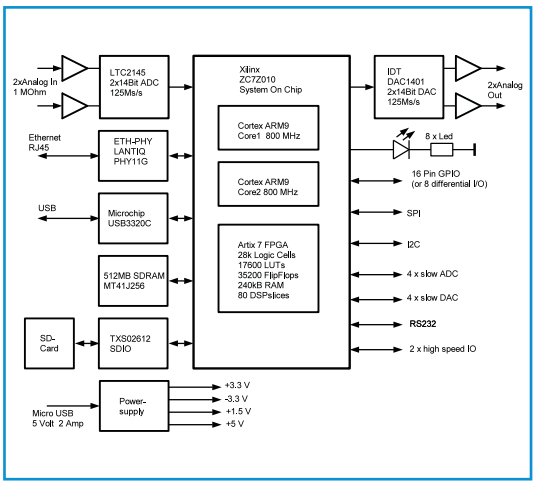


Figure 2.
Le diagramme fonctionnel du Pitaya.

Le diagramme fonctionnel de la **figure 2** donne un premier aperçu des possibilités du module Red Pitaya (**fig. 1**). La puce centrale est un ZC7Z010, un système sur puce (SoC) de Xilinx équipé d'un processeur ARM9 à double cœur, cadencé à 800 MHz pour faire tourner un système Linux taillé pour le Red Pitaya. La puce fournit en outre un FPGA doté de 28000 cellules logiques, de 80 tranches DSP ainsi que de nombreux blocs de RAM (BRAM). Ce FPGA permet de créer des fonctions très rapides, comme celles utilisées en traitement du signal numérique. Le système sur puce est entouré de toutes sortes de périphériques et d'une RAM de 512 Mo.

Pour les mesures, on dispose de deux convertisseurs A/N et N/A dont la fréquence d'échantillonnage peut aller jusqu'à 125 Méch/s. Ces deux unités permettent de réaliser des tâches d'autant plus complexes qu'elles peuvent être soutenues par la vitesse de traitement du FPGA. Le module fournit une connexion Ethernet, un port hôte USB et un port COM USB pour faciliter l'exploitation d'une interface série par PC, ainsi qu'un connecteur pour une carte SD servant de mémoire de masse à Linux. Des connecteurs d'extension donnent accès à des ports d'E/S à utilisation générale, à I²C, à RS-232 et à d'autres convertisseurs A/N et N/A, plus lents que les précédents.

Avec un équipement aussi complet, on peut effectuer des mesures complexes et bénéficier de performances élevées en traitement du signal, même à hautes fréquences. C'est précisément l'ambition du Red Pitaya : être une plateforme de mesure universelle dotée d'une largeur de bande de 50 MHz.

Mise en service

La mise en service est aisée grâce à l'excellent guide de démarrage du site Red Pitaya. On copie l'image du système pour le Red Pitaya sur une carte microSD que l'on l'insère ensuite dans le connecteur SD du module. Il suffit alors de relier le Pitaya à un réseau LAN et de le mettre sous tension pour inaugurer le premier amorçage.
Suivre le processus d'amorçage avec le port COM USB d'un PC permet de voir l'adresse IP affectée au module Pitaya. L'amorçage terminé, on peut communiquer directement avec Linux via le port COM USB ou, méthode préférée, passer par Ethernet pour établir d'autres communications.

Bienvenue au Bazaar

Un des objectifs du projet Red Pitaya était de fournir des applications web et des instruments virtuels que l'utilisateur peut télécharger et installer facilement. C'est chose faite avec le Bazaar, le marché en ligne des applications pour Pitaya. On y accède en tapant l'adresse IP du module dans la barre d'adresse de son navigateur.

La liste des applications présentes dans le Bazaar au moment de rédiger cet article (**tableau 1**) se sera sans doute bien enrichie quand vous la lirez sur le site. Les projets sont tous à code source ouvert et peuvent donc servir de point de départ à toutes sortes de développements.
J'ai testé l'oscilloscope et l'analyseur de spectre avec le circuit dont le schéma est représenté sur la **figure 3**, et l'assemblage sur la **figure 4**. Les entrées du convertisseur A/N du Pitaya sont équipées d'AOP à impédance d'entrée élevée (1 MΩ). On peut donc y relier directement des sondes d'oscilloscope. Deux gains sont disponibles pour les gammes 2 V_{CC} et 20 V_{CC}. Après avoir relié les sondes en TP1 et TP2 (fig. 3) et lancé l'oscilloscope, j'ai obtenu l'oscillogramme de la **figure 5**. Tous les paramètres habituels d'un oscilloscope sont réglables.

La **figure 6** montre l'écran de l'analyseur de spectre. Les deux spectres peuvent être suivis et comparés en temps réel. Les harmoniques de l'oscillateur sont facilement identifiables. On voit que ceux de la sortie de TPI ont une amplitude plus élevée que ceux de TP2.

Après ce premier aperçu des capacités de la plateforme Pitaya, voyons s'il est facile de construire sa propre application web.

Tableau 1. Applis disponibles dans le Bazaar

- Générateur de signal à 2 canaux, 0 à 50 MHz
- Oscilloscope à 2 canaux, 2 x 125 Méch/s
- Analyseur de spectre à 2 canaux
- Moniteur de réponse en fréquence
- Régulateur PID

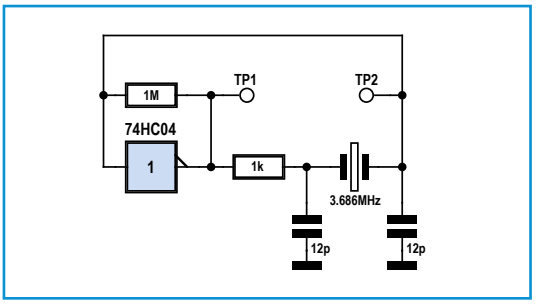


Figure 3.
Circuit d'un oscillateur à quartz.

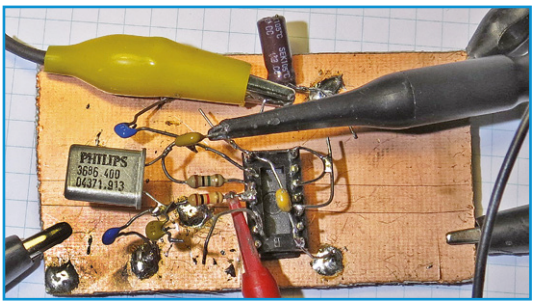


Figure 4.
Montage pour tester l'oscilloscope.

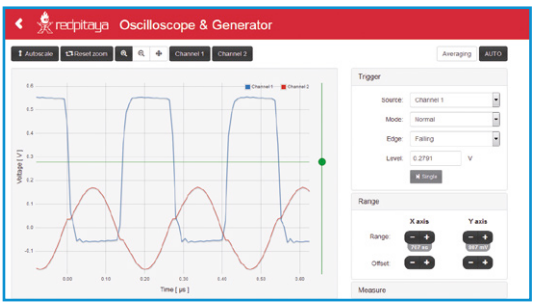


Figure 5.
Écran de l'appli Oscilloscope.

Figure 6.
Écran de l'appli Analyseur de spectre.

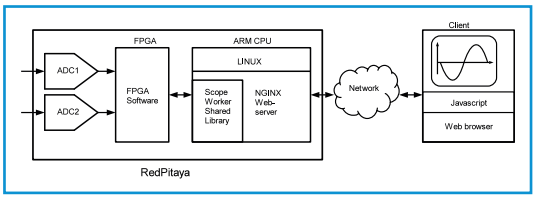


Figure 7.
Les composantes d'une appli web Pitaya.

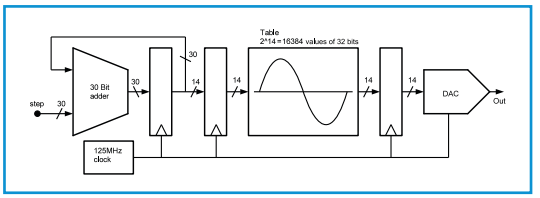


Figure 8.
Schéma du générateur de signal.

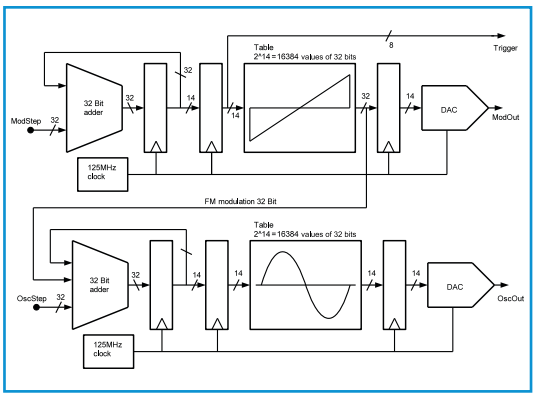


Figure 9.
Schéma des deux modules DDS formant le générateur FM.

Appareils de mesure par service web

La **figure 7** décortique les rouages d'une application Pitaya. Les convertisseurs A/N et N/A sont en général commandés par le FPGA, qui peut aussi avoir à effectuer un prétraitement, p. ex. enregistrer des images dans sa mémoire rapide. Pour ce type d'opération on doit écrire le code FPGA correspondant.

Le FPGA communique habituellement avec un programme exécuté par l'ARM9. Ce programme se charge des instructions plus lentes mais plus complexes. Le code, qui doit bien sûr être lui aussi écrit pour toute nouvelle application, est inclus dans le serveur Linux en tant que bibliothèque partagée. Le serveur transmet les données entre le navigateur client et le programme ARM. Le code de l'interface graphique d'une nouvelle application doit lui aussi être écrit, p. ex. en JavaScript. Si vous rêviez déjà de votre propre application de mesure, vous êtes sans doute refroidi par l'investissement intellectuel apparemment nécessaire. Les applications Pitaya sont heureusement à code source ouvert, et fournissent donc assez de tambouille pré-cuite pour que l'on puisse mitonner un projet sans avoir à sortir trop de casseroles.

Programmation de l'ARM

Voyons d'abord ce qu'une simple programmation du processeur ARM permet de faire en utilisant le code FPGA à disposition. Ce code contient des fonctions permettant de construire un générateur de signal à deux canaux par synthèse numérique directe (DDS, **D**irect **D**igital **S**ynthesis). Autrement dit chaque canal forme l'onde à partir de valeurs d'échantillonnage stockées dans une table. Ces 16384 valeurs sont traitées par le convertisseur N/A à 14 bits à une fréquence d'échantillonnage de 125 MHz. La **figure 8** montre le schéma de principe pour un canal.
Les paramètres se trouvent dans les registres du FPGA. Le processeur peut lire et écrire leur contenu en utilisant leurs adresses fixes. Cette tâche est confiée à un programme C.

Générateur FM-RTTY

La fréquence du générateur doit être commutée de façon logicielle entre deux valeurs. On obtient un générateur FM-RTTY en modifiant le pas (*step*) du générateur DDS. Le programme attend 20000 µs entre deux modifications, ce qui donne un débit de 50 bits/s. Les deux fréquences d'envoi peuvent être réglées entre 0 et 50 MHz. Dans le programme d'exemple, les bits à transmettre sont préalablement transcodés en code Baudot, puis traités dans une boucle (**listage 1**). Un simple accès à la variable *chb-count-step* permet de modifier directement le registre du FPGA. Cet exemple montre que quelques lignes de code C suffisent pour créer une application Pitaya intéressante. Le code C doit bien sûr être compilé avec un compilateur compatible avec l'ARM9. Pour ma part j'utilise l'option ARM9 pour le compilateur GCC d'une distribution *Lubuntu* que je fais tourner sous *VirtualBox*.
On peut aussi écrire un programme C pour la plateforme Pitaya depuis un environnement Windows, puis transmettre l'exécutable au module à l'aide d'un client SCP comme WINSCP.

Programmation FPGA

Quelques mots avant de voir un exemple de programmation FPGA. J'utilise la version gratuite de l'environnement *Vivado* de Xilinx, la WebPACK Edition, qui fournit elle aussi un code FPGA standard pour Pitaya en tant que projet *Vivado*. Un bon moyen de créer ses propres programmes,

et en particulier de ne pas avoir à recréer une interface pour le processeur ARM et les autres unités matérielles, est de partir de ce code et de le modifier au besoin.
Autre aide appréciable, on peut rendre visibles les registres FPGA et les signaux sous forme de variables dans la mémoire de l'ARM. Pouvoir « regarder » en direct dans le FPGA facilite considérablement le débogage.

Générateur FM

Le FPGA standard comprend un générateur de signal à deux canaux. La **figure 8** schématise le fonctionnement d'un canal. J'ai modifié l'ensemble de façon à ce que le signal d'un canal module la fréquence de l'autre. L'onde de modulation est stockée en mémoire, ce qui permet d'utiliser une modulation linéaire pour créer aussi un vobulateur.
Le schéma du générateur FM est représenté sur la **figure 9**. Le groupe d'éléments du dessus constitue le modulateur DDS. Les valeurs de la table sont représentées sur 32 bits afin que la fréquence puisse être modulée de façon précise. Ces valeurs sont transmises au deuxième convertisseur N/A pour y être contrôlées. Il est crucial que ces valeurs parviennent au sommateur du deuxième module DDS (en bas) afin que cet oscillateur soit effectivement modulé en fréquence. Pour qu'un oscilloscope puisse être déclenché en mode vobulateur, un signal de déclenchement (*trigger*) est prélevé sur la partie du haut, puis envoyé à la fois sur des broches GPIO et dans un registre qui pourra être lu de façon logicielle. Le code principal du modulateur (**listage 2**) ne comprend que quelques déclarations et un bloc de code de six instructions.
J'ai utilisé ce générateur FM pour produire un signal de 10 MHz modulé par un signal sinusoïdal de 1 kHz d'amplitude déterminée. La **figure 10** montre le spectre correspondant. L'excursion est réglée pour que manquent les fréquences de 10 MHz ±2 kHz.
Le code Verilog du générateur à modulation de fréquence (**listage 3**) est quasiment le même que celui du modulateur. La principale différence réside dans le calcul de la nouvelle position, à laquelle est ajouté le signal de modulation *fmInput-i*.
Lorsqu'on paramètre le générateur de façon à ce qu'il effectue un balayage entre 9,7 MHz et 11,7 MHz, on obtient le signal de la **figure 11**, qui représente la mesure d'un filtre FI THF de 10,7 MHz. La courbe bleue sous le signal est le signal de déclenchement.

Listage 1; Boucle d'envoi RTTY

```
uint32_t delay1=20000 ; // 50 Bit/sec
while(1){
  for(int k=0 ; k<nBits ; k++){
    int theBit=bitBuffer[k] ;
    if(theBit){ g_awg_reg->chb_count_step = step2 ; }
          else{ g_awg_reg->chb_count_step = step1 ; }
    usleep(delay1) ;
    }
  }
```

Listage 2. Code Verilog du modulateur DDS

```
reg [ 32-1: 0] dac_buf [0:(1<<14)-1] ; // data buffer
16384x32 Bit
reg [ 32-1: 0] dac_rd    ; // DAC value
reg [ 14-1: 0] dac_rp    ; // DAC read pointer
reg [ 32-1: 0] dac_pnt   ; // read pointer

always @(posedge dac_clk_i) begin
   dac_rp   <= dac_pnt[14-1+18:0+18];
   dac_rd   <= dac_buf[dac_rp] ;     // read data value
   signal_o <= dac_rd ;
   dac_o <=dac_rd[14-1+12:0+12] ;    // feed to output
   dac_pnt <= dac_pnt + set_step_i ; // get new position
   trigSignal_o <= dac_rp[8-1+6:0+6] ;
end
```

Listage 3. Code Verilog de l'oscillateur DDS

```
reg  [ 14-1: 0] dac_buf [0:(1<<14)-1] ;
reg  [ 14-1: 0] dac_rd ;
reg  [ 14-1: 0] dac_rp ;
reg  [ 32-1: 0] dac_pnt ; // read pointer

always @(posedge dac_clk_i) begin
  dac_rp    <= dac_pnt[14-1+18:0+18];
  dac_rd    <= dac_buf[dac_rp] ;
  dac_o <= dac_rd[13:0] ;
  dac_pnt <= dac_pnt + set_step_i + fmInput_i ;
end
```

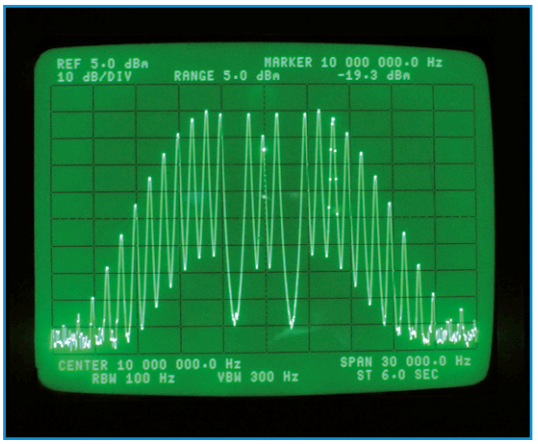


Figure 10.
Spectre d'un signal FM.

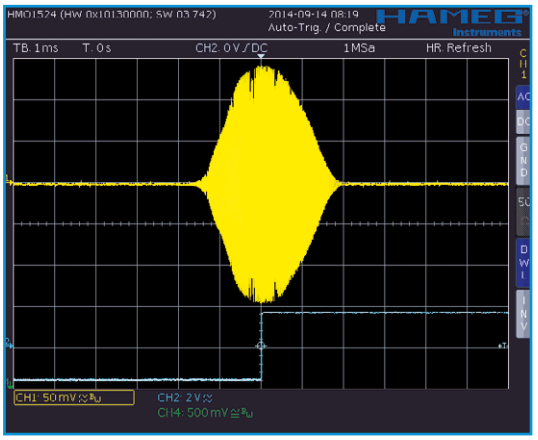


Figure 11.
Balayage FM d'un filtre FI de 10,7 MHz.

D'autres projets

On vient de le voir, de simples programmes FPGA et C permettent de créer des outils de mesure et d'essai pratiques et dotés de fonctions complexes. Voici, décrits en quelques mots, d'autres projets Pitaya déjà réalisés :

Générateur AM

Comme le générateur FM précédent, ce générateur AM utilise le convertisseur A/N pour moduler des signaux externes.

Voltmètre RMS

Les entrées sont échantillonnées à une fréquence de 125 Méch/s pour calculer puis afficher la valeur RMS. On peut de la même façon mesurer des valeurs RMS sur deux canaux avec une largeur de bande allant jusqu'à 50 MHz.

Analyseur de gain/phase/impédance

Cette application est décrite dans le blog de Red Pitaya et disponible sur GitHub. Pour mesurer les réponses en fréquence et en phase de filtres et autres amplificateurs dans l'intervalle de 1 kHz à 50 MHz, un signal sinusoïdal est produit, puis les modules de son intensité et de sa phase sont mesurés en entrée et en sortie. Les valeurs d'impédance sont calculées à partir de l'intensité et de la tension.

Récepteur AM de 0 à 50 MHz

Le signal d'une antenne filaire alimente un convertisseur A/N. Il est mélangé au signal d'un oscillateur sur la fréquence de réception (fréquence d'échantillonnage de 125 Méch/s). Les composantes en quadrature sont filtrées par un passe-bas et réduites jusqu'à 120 kéch/s. Ces signaux sont amenés par FIFO dans l'ARM pour être démodulés. Le signal démodulé sort par un convertisseur N/A. Le tout forme un récepteur AM logiciel simple.

Émetteur BLU de 0 à 50 MHz

Le signal vocal est échantillonné par le convertisseur A/N, puis converti par le processeur au moyen de deux filtres en une paire de filtres de Hilbert I/Q. Ces signaux I/Q arrivent au FPGA échantillonnés à 120 kéch/s, y sont interpolés et filtrés, puis modulent le sinus et le cosinus de la porteuse.

Perspective

À l'aune de son potentiel, il ne fait guère de doute que le puissant matériel de la plateforme Red Pitaya sera utilisé pour de nombreuses autres applications, et qu'avec le temps nous verrons fleurir de plus en plus « d'instruments » *open source*, ce qui de surcroît amortira le prix d'achat du module.

(140277 – version française : Hervé Moreau)

Liens

http://redpitaya.com

http://bazaar.redpitaya.com

ECD7

NOUVELLE EDITION

Base de composants d'ELEKTOR

Cet ensemble consiste en une quadruple banque de données (circuits intégrés, transistors, diodes et optocoupleurs) complétée par neuf applications satellites, au nombre desquelles on trouvera notamment de quoi calculer la valeur de la résistance associée à une diode zener, à un régulateur, à un diviseur, ou un multivibrateur astable, mais aussi le code de couleur de la résistance et de l'inductance. Avec ce CD-ROM, vous disposez donc de données fiables sur plus de 7.800 circuits integers ; plus de 35.600 transistors, FET, thyristors et triacs ; environ 25.000 diodes et plus de 1.800 optocoupleurs. Le clou, c'est que vous allez pouvoir rajouter dans la base de données ce qui y manque encore, car elle est interactive ! Ainsi chaque utilisateur pourra lui-même rajouter des composants, en modifi er les caractéristiques déjà enregistrées ou les compléter.

ISBN 978-90-5381-298-3 • 29,50 €

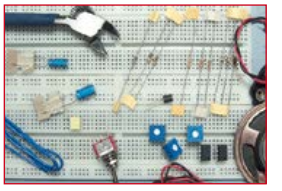
elektor **Pour commander en ligne :**
www.elektor.fr/ecd7

l'électronique pour les débutants

trois kits d'initiation disponibles

ISBN 978-2-86661-186-6 - 352 pages - **42,50 €**

Voici le cadeau idéal pour partager votre passion de l'électronique avec vos enfants, petits-enfants, neveux… et autres *geeks*



Fin pédagogue, Rémy Mallard écrit pour les débutants et répond d'abord aux questions prosaïques du néophyte : quel fer à souder acheter ? Un multimètre à 5 € peut-il suffire ? Et bien d'autres trop souvent laissées en suspens.

L'auteur démystifie l'électronique en n'utilisant que ce qu'il vous faut de théorie pour aborder la pratique : identifier les composants et leur rôle, les récupérer, les tester et les ranger ; lire un schéma ; choisir ses outils ; mettre en boîte ses montages…

Les kits permettent de réaliser quelques-uns des montages simples et ludiques présentés dans le livre.

Kit n°1 : sirène – réf. 119016-71 - **24,50 €**
Kit n°2 : chenillard & thermomètre - réf. : 119016-72 - **24,50 €**
Kit n°3 : orgue - réf. : 119016-73 - **24,50 €**

elektor **Offre spéciale : livre + deux kits = 99,- €** au lieu de 116,- €
Informations complémentaires et commande :
www.elektor.fr/debut

sapin virtuel : il azure

clarté bleutée à la veillée

Harry Baggen (Elektor)
en coopération avec
Eurocircuits

En octobre dernier, le prix Nobel de physique était attribué aux inventeurs de la LED bleue, ce composant aujourd'hui banal, mais toujours en voie d'amélioration après avoir tenu les chercheurs en échec pendant des décennies. Elektor fête l'événement ici même avec un projet de circonstance : une silhouette d'épicéa garnie de 62 LED avec lesquelles vous pouvez laisser libre cours à votre fantaisie pour en modifier les arabesques et les séquences lumineuses.

Nos lecteurs les plus fidèles se souviennent sans doute de cette couverture d'Elektor de mars 1982 qui titrait : ***La LED bleue existe-t-elle ?*** Cette question aujourd'hui dérisoire a longtemps taraudé bien des scientifiques dans le monde entier avant de trouver une réponse au Japon. Quelques décennies plus tard, la LED d'éclairage est littéralement partout. Elle est devenue *le* composant de ce début du XXIe siècle. Nous pensons que pour les fêtes de fin d'année on peut faire mieux que les grandes surfaces de bricolage ou de jardinage et leurs gadgets tout faits : des sapins en plastique avec boules et guirlandes, pas toujours à LED d'ailleurs. Avec le concours du fabricant de circuits imprimés Eurocircuits, nous avons élaboré une création originale que vous pouvez construire vous-même, s'il vous en reste le temps. Vous pouvez aussi la commander, prête à l'emploi et pour un bon prix, directement dans l'e-choppe d'Elektor. Et si par ailleurs on vous a déjà décerné le Nobel de l'esbroufe, vous n'hésiterez pas à bluffer vos visiteurs, autour d'un petit verre, en leur racontant que c'est sous microscope et avec un fer extrêmement fin que vous avez tout soudé vous-même…

Tout soudé, ici cela signifie 62 LED bleu clair... Plus l'électronique de commande, centrée sur un puissant microcontrôleur, sur l'autre face. Après avoir publié maints projets de sapins de Noël clignotants, en voici un qui offre un véritable spectacle lumineux avec des motifs extraordinairement variés ou des textes qui défilent en boucle. Certains sont programmés d'origine en mémoire, mais ce qui est remarquable, c'est **la possibilité qui vous est offerte de créer vous-même, simplement, une animation ou des effets particuliers sur PC pour les charger ensuite dans la mémoire de l'arbre** par une classique liaison USB.

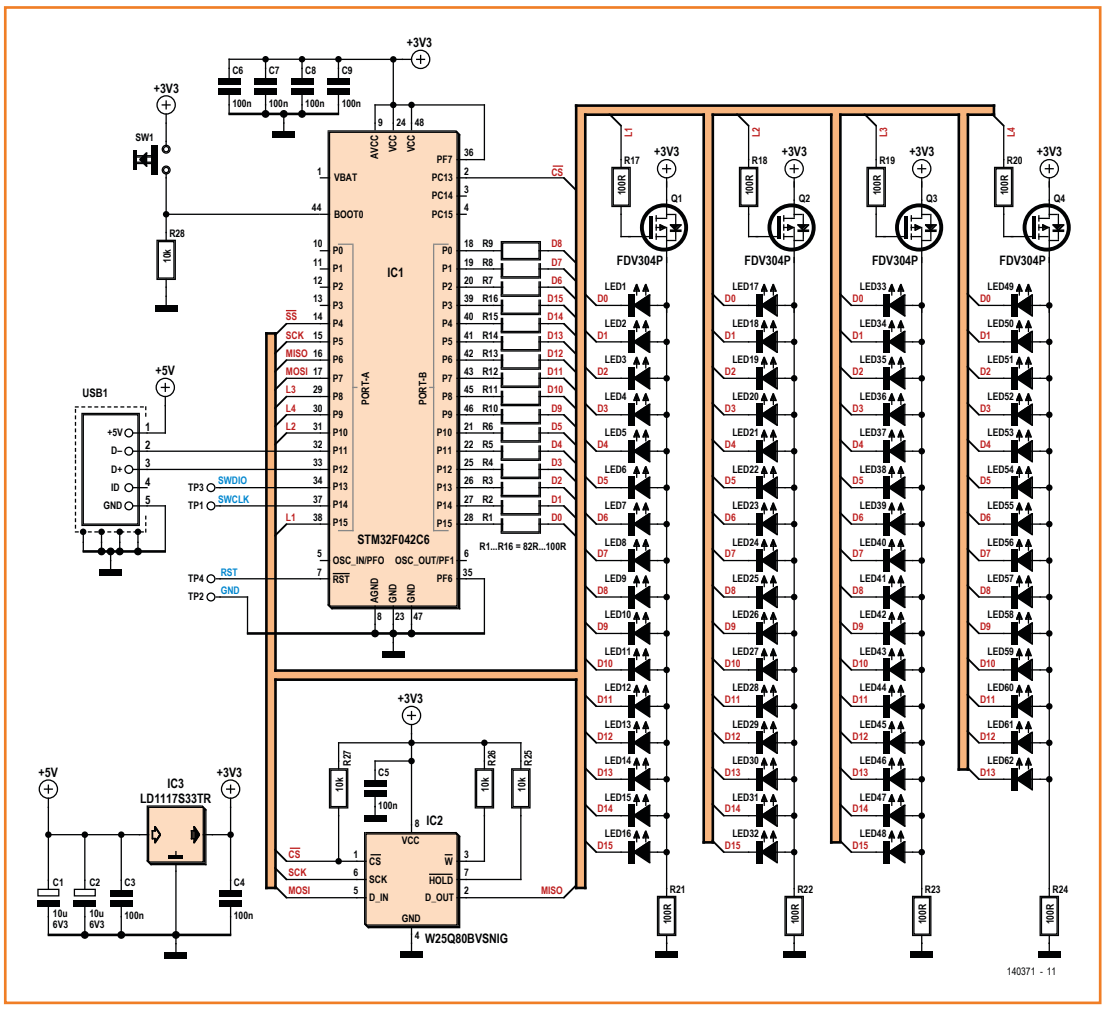


Figure 1.
Dans le schéma, peu de chose à voir d'autre que le microcontrôleur de STM et les faisceaux de LED bleues, 62 en tout.

Liste des composants

Résistances :
R1 à R16 = 82 à 100 Ω (CMS 0805)
R17 à R24 = 100 Ω (CMS 0603)
R25 à R28 = 10 kΩ (CMS 0603)

Condensateurs :
C1, C2 = 10 µF/6,3 V tant. (CMS A)
C3 à C9 = 100 nF/10 V cér. (CMS 0603)

Semi-conducteurs :
LED1 à LED62 = LED bleue (CMS 1206)
Q1 à Q4 = FDV304P (SOT23)
IC1 = STM32F042C6 (LQFP48)
IC2 = W25Q80BVSNIG (SO8)
IC3 = LD1117S33TR (SOT223)

Divers :
USB1 = connecteur micro USB encartable (47346-0001)
SW1 = bouton-poussoir encartable (TACTB-64K-F)
dessin de circuit imprimé réf. 140371-1

Puissant microcontrôleur

Pour commander les LED, nous avons choisi un microcontrôleur ARM Cortex-MO de STM (**fig. 1**), à 32 bits, relativement bon marché, qui assure. Il offre une grande puissance de calcul et dispose d'une interface USB 2.0 pour s'associer aisément au PC. Son chargeur d'amorce embarqué permet de charger aisément un autre micrologiciel ou de nouveaux motifs de LED dans une mémoire flash série, couplée au microcontrôleur, d'une capacité de 1 Mo (8x1 Mbit).

Habitué à la capacité **giga**ntesque des clés USB actuelles et des cartes mémoire, on peut trouver ça étriqué : la capacité n'en est pas moins de 90 000 motifs de LED complets !
Les LED sont groupées en matrice de 4x16 (enfin presque : 3x16 + 1x14). À tour de rôle, les lignes P0 à P15 du port B du contrôleur attaquent 16 (ou 14) LED. Les 4 colonnes sont continuellement pilotées en multiplex. La rémanence de l'œil humain fait le reste : les LED paraissent éclairées en permanence. Les colonnes de LED sont commutées par quatre MOSFET à canal P du type FDV304P. Les cathodes sont reliées par les résistances de limitation de courant R1 à R16, sans tampon, aux lignes de port du microcontrôleur. Malgré cela, il y a déjà une fameuse quantité de

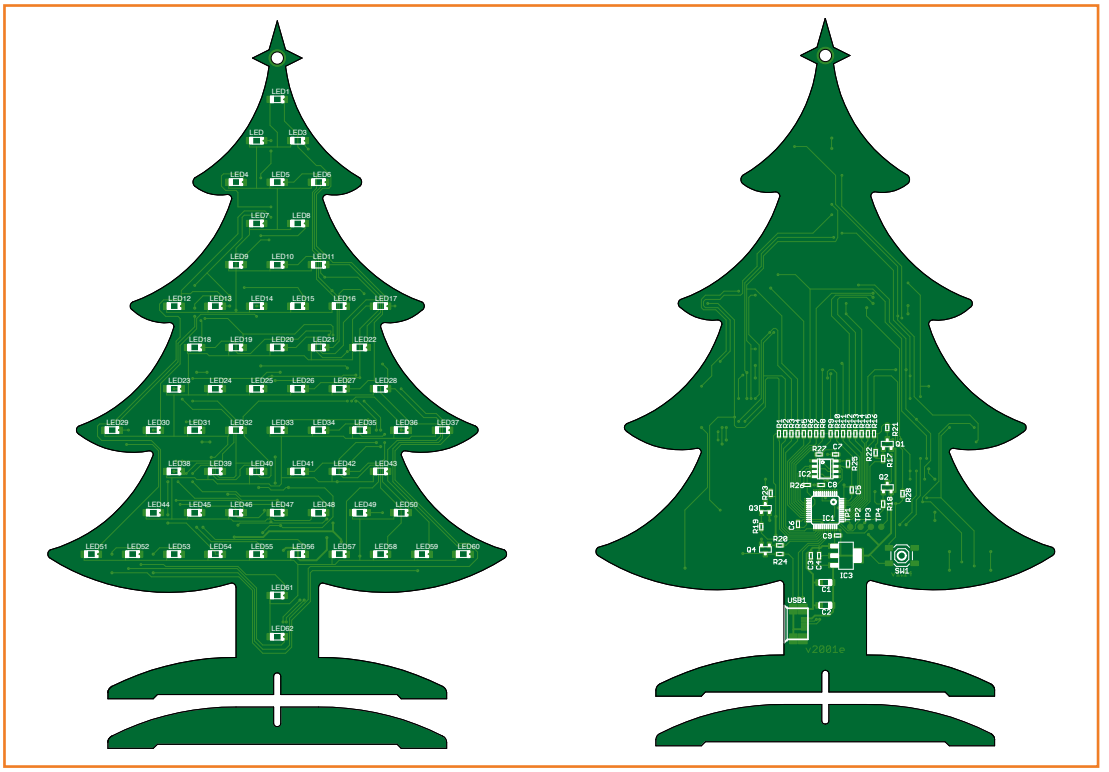


Figure 2.
Le circuit imprimé est à double face, la plus visible est décorée, saison oblige, d'un masque de soudage PIXture [1] à cristaux de glace.

composants, reste à savoir si le contrôleur peut supporter la charge. Il est capable de fournir en tout 80 mA, ce qui limite le courant à 5 mA par LED. Les exemplaires à haut rendement restent bien visibles dans un environnement raisonnablement éclairé.
Tout le circuit est alimenté à partir d'une prise micro USB. Le régulateur de 3,3 V IC3 assure une tension stable au µC et à la matrice de LED.
La **figure 2** montre le projet de circuit imprimé du sapin. Les LED sont en face avant, le reste des composants, y compris le connecteur micro USB, à l'arrière. Pour la circonstance, le circuit imprimé est revêtu d'un masque de soudage décoratif PCB PIXture [1] à cristaux de glace.
Si vous vous lancez dans la construction (gravure du circuit imprimé et implantation des composants), il faut savoir que certains CMS sont difficiles à souder à la main.
Les codes source et hexadécimal du micrologiciel sont disponibles sur le site [2]. Le logiciel ARM est rédigé en C avec un compilateur Keil. La plus grande part du code est consacrée à la communication par USB, basée sur la bibliothèque de micrologiciel ST.

Quand le sapin est en liaison avec le PC, appuyer sur le bouton-poussoir à l'arrière pour le mettre en mode d'amorçage : il charge ou met à niveau le micrologiciel à l'aide du logiciel DfuSe [3] de ST.

Logiciel

D'origine, le sapin prêt à l'emploi tel qu'il est vendu dans l'e-choppe dispose d'une gamme variée d'animations. Si elles vous satisfont, il n'y a rien d'autre à faire. Sinon, créez vos propres motifs lumineux, animations, textes en boucle et autres fantaisies. Il y a pour cela une page spéciale [4] en ligne, pour vous exercer à d'autres créations de façon interactive. À la souris, vous pouvez y sélectionner les LED pour réaliser vos propres animations. Vous obtiendrez des séquences d'images, dans lesquelles vous pourrez régler, à gauche, la durée voulue et l'intensité lumineuse, avant de passer au plan suivant. Toutes les images créées s'inscrivent côte à côte au bas de l'écran. Vous pouvez aussi rédiger un texte à faire passer en boucle, le symbole de texte est en haut, à gauche. Cet outil en ligne facilite la composition de vos nouvelles animations, que vous pourrez sauvegarder et modifier ultérieurement. À chaque étape, le bouton *Preview* permet de voir l'animation à l'écran.

Gagnez un prix avec votre animation !

Eurocircuits organise un concours de programmation pour ce sapin. Postez votre propre animation sur la page du sapin [4] et arrangez-vous pour que le plus possible de personnes « aiment » votre programme. Pour ceux qui auront obtenu le plus grand nombre de « j'aime », il y aura différents prix à gagner. La compétition sera clôturée le 7 janvier 2015, les gagnants seront avertis personnellement. À vos sapins !

Vous pouvez aussi composer ainsi différentes animations et les faire se succéder pour produire un spectacle lumineux. Il ne vous restera plus qu'à télécharger la série complète sur le sapin.
La liaison se fait par câble micro USB vers le PC. Lors de la première connexion, Windows reconnaît le sapin comme appareil HID et installe un pilote approprié. Ensuite, vous pourrez, depuis la page internet, flasher les nouvelles animations dans le sapin.
Pour le mettre en service à la maison, branchez-le sur un adaptateur secteur de 5 V pourvu d'un câble micro USB.

Nous vous souhaitons de joyeuses illuminations.

(140371 – version française : Robert Grignard)

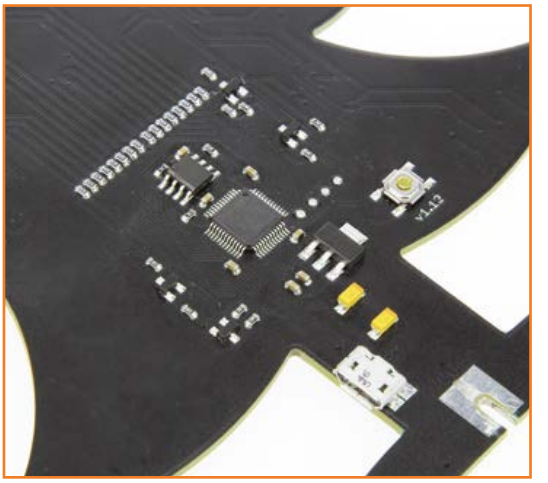


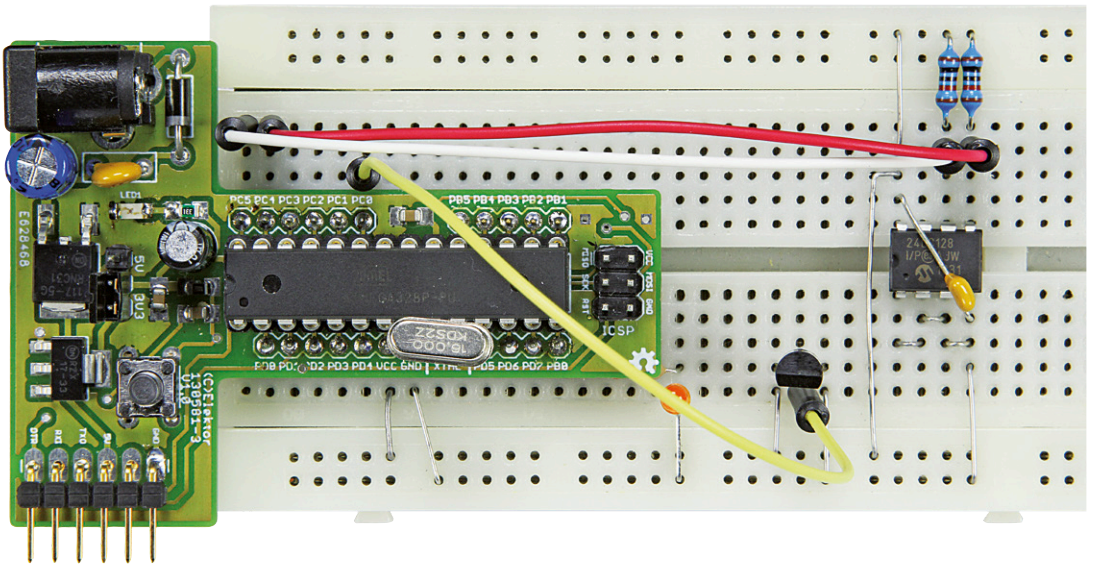
Figure 3.
Les autres composants sont installés sur la face arrière du sapin.

Liens

[1] www.eurocircuits.com/blog/171-PCB-PIXture-launched
[2] www.elektor-magazine.fr/140371
[3] www.st.com/web/en/catalog/tools/FM147/CL1794/SC961/SS1533/PF257916
[4] www.eurocircuits.com/x-mas

T-Board 28 : consommation minimale

moins c'est mieux, microscopique c'est fantastique



Andrew Retallack
(Afrique du Sud)

La consommation d'énergie est devenue cruciale depuis que l'Internet des Objets dope la création des projets à microcontrôleurs. Chaque milliampère compte lorsqu'un capteur distant doit fonctionner des mois durant sur une unique batterie. Comme nous allons le voir avec la construction d'un enregistreur de température, la carte T-Board 28 se prête particulièrement bien à ce type d'optimisation.

Un des avantages clés de la T-Board 28 [1] par rapport à d'autres plateformes figées comme les cartes Arduino est de permettre d'optimiser progressivement la consommation d'un circuit en y apportant de petites modifications et en mesurant les effets produits. Comme la Uno, la T-Board 28 est équipée d'un ATmega328. Étant donc à peu près sûr que les premiers utilisateurs des T-Boards viendraient en majorité du monde Arduino, je me suis dit qu'il serait judicieux d'illustrer ma façon de procéder à l'aide d'un enregistreur de température dont je cherche à réduire la consommation. Vous trouverez les moTules T-Board (ainsi qu'un t-shirt !) dans l'e-choppe.

Pourquoi un enregistreur de température ?

Pour me familiariser avec les différentes façons de réduire la consommation d'un circuit, je voulais un projet à la fois d'intérêt pratique et assez simple pour ne pas me distraire de mon objectif premier. Pour ça un enregistreur de température était parfait. Le montage devait mesurer la température à intervalles réguliers au moyen d'un capteur bon marché, et stocker les valeurs dans une EEPROM. Je voulais pouvoir charger les données dans un PC ou un portable via une liaison série, et j'avais pour cela d'abord pensé à une carte SD, mais elles ne tolèrent pas le 5 V et leur consommation est jusqu'à 3,5 fois supérieure à

celle des EEPROM, d'où mon choix final. Le projet pourrait être amélioré par une transmission radio des lectures, mais fignoler la consommation d'un circuit RF serait une tout autre histoire !

Petit tour des pistes

C'est là un autre des avantages des moTules, la conception est très simple (**fig. 1**). Une EEPROM 24LC128P de Microchip est reliée via I²C à une carte T-Board 28, avec les indispensables résistances de rappel vers le haut placées sur les lignes de communication. Le capteur de température est un LM60, que j'ai choisi pour son faible courant nominal, sa capacité à mesurer des températures négatives (j'ai pensé à tous ceux qui ne vivaient pas sous le soleil d'Afrique du Sud !), et sa prise en charge des tensions d'entrées à partir de 2,7 V. La LED ne fait pas partie de l'enregistreur final, elle m'a uniquement servi de témoin pour visualiser les tâches en cours lorsque je mesurais le courant consommé.
Comme tout composant I²C, le 24LC128 possède une adresse. La partie fixe de cette adresse à 7 bits est 1010, les bits suivants étant définis par le niveau haut ou bas des broches 3, 2 et 1 (A2, A1 et A0). La ligne d'adresse A0 étant ici tirée vers le haut, l'adresse de l'EEPROM est 1010001. Soyez attentif lorsque vous câblez la puce sur la carte, j'ai perdu des heures à chercher une erreur dans le code pour finalement découvrir que j'avais relié la broche 7 (*Write Protect*) à V_{CC} au lieu de V_{SS} (GND). Résultat, la puce était effectivement protégée en écriture ! Lire trop vite une fiche technique est… vite arrivé.

L'E/S série (pour le débogage et le vidage des valeurs de lecture stockées) se fait via les broches FTDI de la T-Board et l'UART intégrée à l'ATmega. Relié au connecteur FTDI de la T-Board, un module ou câble FTDI permet de relier la carte au port USB d'un PC. Une autre version de l'enregistreur pourrait inclure l'adaptation du code à la T-Board 8, ce qui en passant nécessiterait d'implanter le protocole série de façon logicielle.

Le programme

J'ai écrit le code avec la version 6.2 d'Atmel Studio. J'ai créé des modules simples pour manipuler le CAN (convertisseur analogique-numérique), les fonctions de l'UART ainsi que le bus TWI (*Two Wire Inferface*). TWI est un protocole compatible à 100 % avec I²C. Atmel a choisi ce nom pour ne pas entrer en conflit avec plusieurs noms de marque déposés autour du protocole I²C. Dans les fiches techniques et pour les noms des registres, c'est donc TWI qu'il faut chercher, pas I²C, mais c'est la même chose.

Vous pouvez télécharger le code depuis [7]. Rien d'abscons dans ce code, mais les symboles de compilation (auxquels on accède via l'option *Symbols* de la section *AVR/GNU C Compiler* des propriétés du projet) nécessitent peut-être quelques commentaires. `F_CPU` définit la fréquence du processeur en hertz pour que la vitesse de communication et les temporisations soient calculées avec précision. `SYSTEM_MILLIVOLTAGE` définit la tension en mV afin que les valeurs du CAN soient correctement converties en tensions. Et il existe un symbole `DEBUG` qui envoie les informations de débogage sur le port série lorsqu'il est mis à 1. Lorsque l'ATmega est mis sous tension, il affiche d'abord un menu via le port série. Ce menu permet d'imprimer le journal au format CSV ou de l'effacer. Le programme attend 10 s que l'utilisateur ait sélectionné une option, puis entre dans la boucle d'enregistrement.

Le code est décliné en deux versions : la première (**fig. 2**) n'est pas optimisée sur le plan de la consommation, la seconde l'est (**fig. 3**). L'optimisation de la version finale n'est pas parfaite,

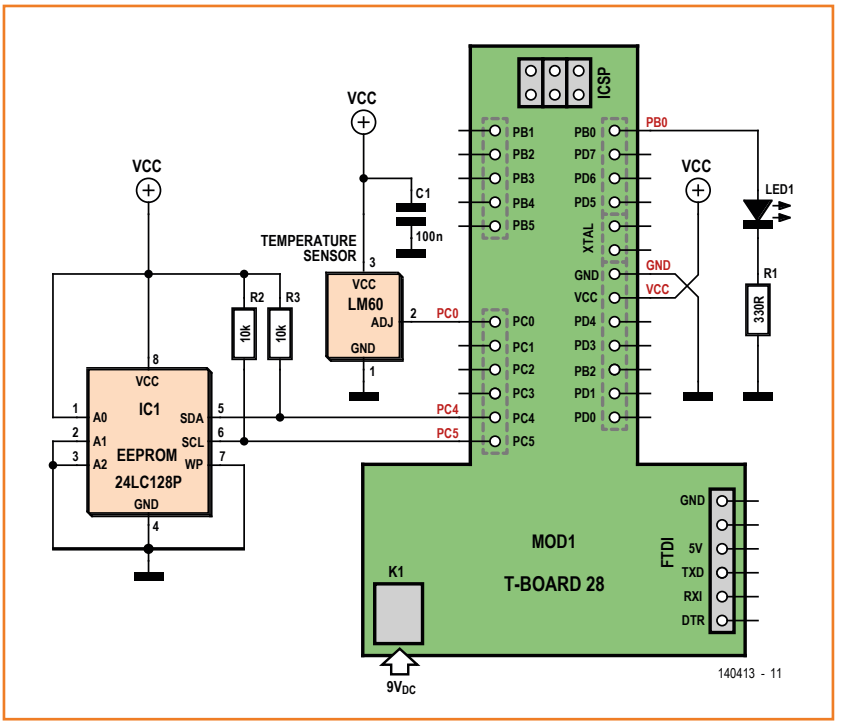


Figure 1.
Schéma du montage exploitant un moTule T-Board 28 pour tenter d'enregistrer à la fois la température et un record de (basse) consommation.

Ne grillez pas vos fusibles !

Les fusibles, ou bits de configuration (Atmel les appelle *word*), servent à paramétrer différentes fonctions ou options des AVR. Ces bits s'écrivent au moyen d'un programmateur ISP externe et, en dehors de quelques exceptions, ne peuvent pas être modifiés par le programme en cours d'exécution.

Ce type de configuration est souvent et à juste titre appréhendé, car un réglage incorrect des fusibles peut « démolir » un processeur. Le « retaper » nécessite alors un programmateur spécial fonctionnant à des tensions plus hautes.

L'Atmega328 de la T-Board 28 possède trois fusibles : le hfuse (*high fuse*), le lfuse (*low fuse*) et le efuse (*extended fuse*). Ici nous modifierons surtout le lfuse, le fusible qui détermine la source d'horloge. Les microcontrôleurs AVR peuvent en effet être cadencés par diverses sources d'horloges internes et externes, et comme nous en utiliserons plusieurs nous aurons à configurer en conséquence le processeur.

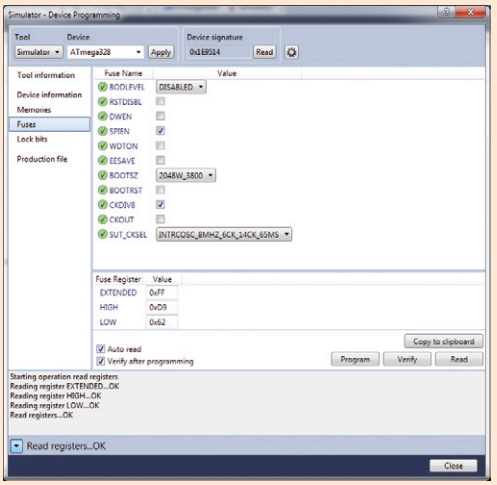
Pour calculer la valeur des mots de configuration, vous pouvez vous attaquer aux données souvent cryptiques des fiches techniques ou utiliser un calculateur. Il en existe plusieurs en ligne (j'aime bien le Engbedded [4]), mais aussi sous forme d'applis pour tablettes ou téléphones tactiles.

Régler des fusibles avec un programmateur pris en charge par Atmel Studio est simple, il suffit d'entrer les valeurs dans la fenêtre *Device Programming* :
Si votre programmeur n'est pas pris en charge par Atmel Studio, utilisez AVRDude [5] :

1. Ouvrez une invite de commandes et placez-vous dans le dossier contenant `AVRDude.exe`.
2. Entrez la ligne suivante pour lire le réglage d'un fusible :
   `avrdude -c <programmer> -p <MCU> -U lfuse:r:-:h`
3. Entrez la ligne suivante pour programmer un fusible :
   `avrdude -c <programmer> -p <MCU> -v -U lfuse:w:0xFF:m`

En remplaçant :
`<programmer>` par le code AVRDude de votre programmateur (p. ex. `usbtiny` pour un USBTinyISP) — référez-vous au fichier *avrdude.conf* pour trouver votre code ;
`<MCU>` par le code AVRDude du processeur à programmer (p. ex. `m328` pour l'ATmega328).
`lfuse` est ici le fusible à positionner ; utilisez `lfuse`, `hfuse` ou `efuse` selon le fusible cible ;
`0xFF` est un exemple de valeur pour le fusible à régler.

Faites une recherche en ligne si vous voulez tester et comprendre plus avant les fusibles, les ressources ne manquent pas.



Configuration des fusibles sous Atmel Studio

Tempérer le temporisateur

Le Timer2 est un temporisateur/compteur à 8 bits et ne peut donc compter que jusqu'à 255. Chaque coup d'horloge incrémente le compteur. Timer2 peut être configuré pour déclencher une interruption lorsqu'il atteint 255. Problème, avec un quartz cadencé à 1 MHz il ne mettra que 256 ms pour atteindre 255 ! Il existe heureusement deux façons de ralentir ce *timer*. La première est d'utiliser un prédiviseur (*prescaler*) pour diviser la fréquence d'horloge. Si nous utilisons la valeur maximale de ce diviseur, c.-à-d. 1024, le compteur atteint 255 en 0,26 s. C'est mieux, mais encore trop rapide. La seconde est de ralentir l'horloge.

Ralentir l'horloge principale serait une mauvaise idée, car cela mettrait en péril la fiabilité des communications via le port série. Le Timer2 peut cependant être cadencé par un quartz externe en mode asynchrone (asynchrone signifie que le quartz externe n'est pas synchronisé avec l'horloge principale du processeur).
Mieux, nous pouvons utiliser des quartz très lents. Avec p. ex. un quartz horloger de 32,768 kHz, le temporisateur/compteur mettra cette fois-ci 7,97 s à atteindre 255.
Enfin une valeur pratique ! Et bien sûr un cristal plus lent consomme moins de courant.

car je voulais que le code reste lisible et facile à comprendre. Si vous l'améliorez, et je vous mets au défi de le faire, n'hésitez pas à partager vos résultats sur www.elektor-labs.com et http://forum.elektor.com (dans le fil Microcontrôleurs & microprocesseurs).

T-Board ou l'optimisation facile

Pour réduire la consommation d'un projet à microcontrôleur, on peut agir sur quatre leviers élémentaires. Le premier, traduction (simpliste) de la loi d'Ohm, est de réduire la tension d'entrée. Le deuxième est de diminuer la fréquence d'hor-

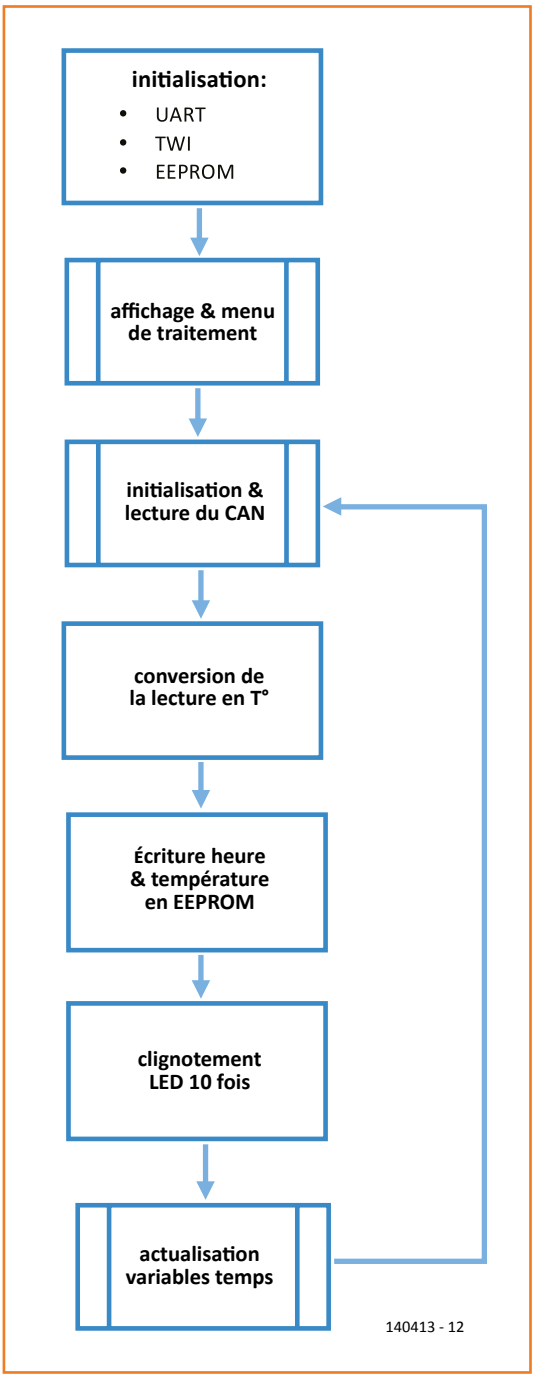


Figure 2. Organigramme de la « version 1 », la version non optimisée du micrologiciel.

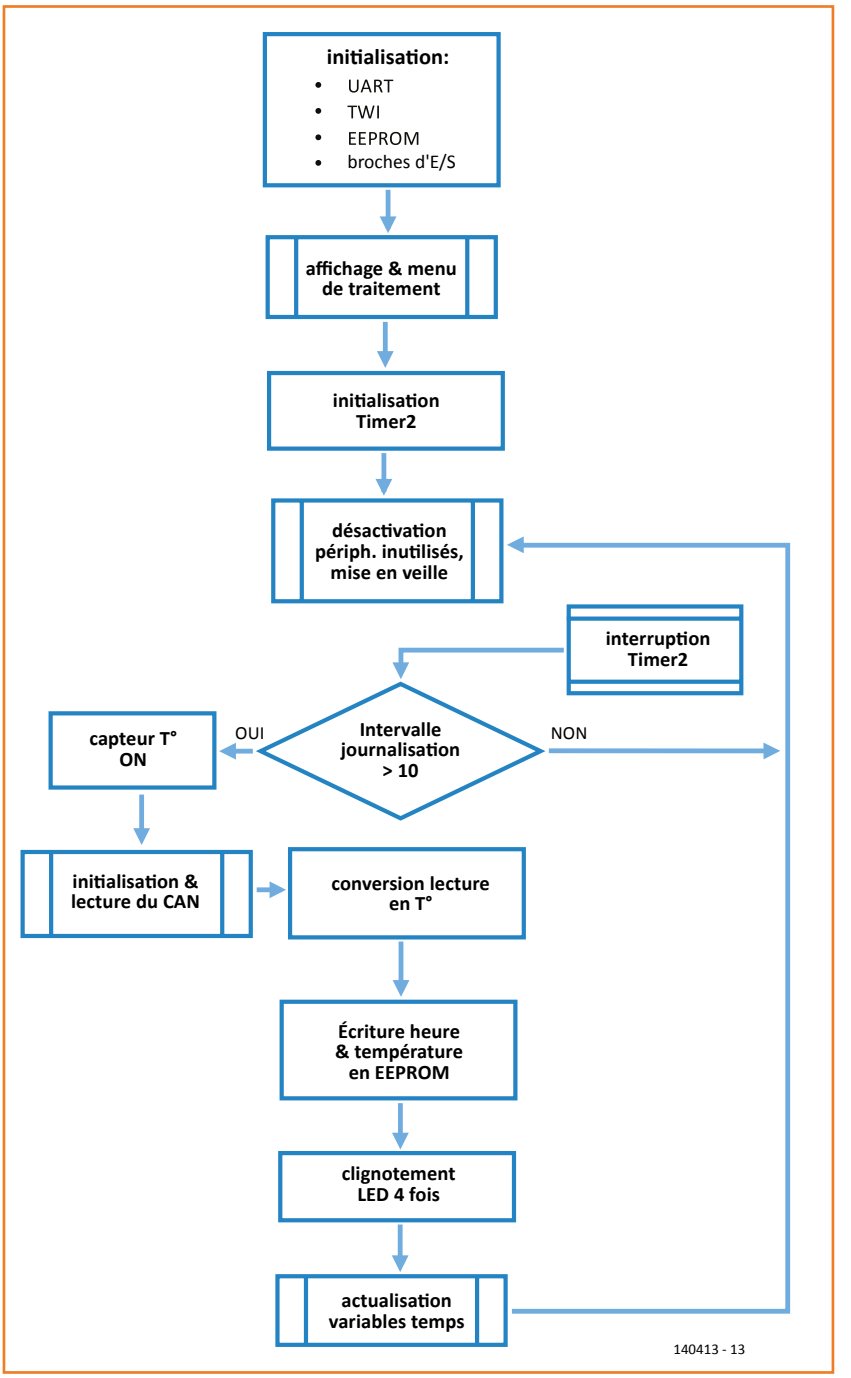


Figure 3. Organigramme de la « version 2 » optimisant la consommation de la T-Board 28.

Les 10 étapes de l'optimisation

1. Définir une référence

La figure 1 montre le schéma d'assemblage du projet. La tension d'entrée doit être définie à 5 V à l'aide du cavalier de sélection de tension, et un quartz externe de 16 MHz doit être relié. Comme nous devons mesurer le courant, le cavalier doit être retiré pour pouvoir connecter un ampèremètre à la broche de sélection de la tension (le noir à la broche centrale, le rouge à la broche de 5 V).

Les fusibles doivent ensuite être réglés selon les valeurs du **tableau 1**.

Pour finir, il nous faut charger la « version 1 » (non optimisée) du code (à télécharger [7]) dans Atmel Studio, la compiler puis la transférer dans le contrôleur. Cette version utilise la fonction `_delays_ms` dans une boucle de temporisation entre deux lectures. L'organigramme de la figure 2 schématise la structure du code.

Mes premières mesures ont révélé une consommation d'environ 12,4 mA à l'état de repos, état dans lequel entre le circuit après 10 clignotements de LED ; j'ai effectué les mesures après ces 10 clignotements, une fois la lecture du multimètre stabilisée.

Un autre clignotement de la LED indique l'imminence d'une lecture de la température, suivie de son écriture en EEPROM. La durée de cette lecture s'est toutefois avérée trop courte pour que la valeur lue sur le multimètre soit valide.

2. Réduire la tension d'entrée

C'est l'optimisation la plus simple qui a permis les plus grandes économies. Placez le cavalier sur la position 3,3 V pour que la T-Board fonctionne sous 3,3 V. Nous n'utilisons bien sûr pas le cavalier pour nos optimisations, donc reliez le fil rouge du multimètre à la broche de 3,3 V.

Le code doit être modifié de façon à ce que les calculs donnant la température soient corrects lorsque la référence vaut 3,3 V. Nous devons d'abord changer le symbole de compilation `SYSTEM_MILLIVOLTAGE` : dans l'explorateur, clic droit sur le projet, puis *Properties*. Ensuite, dans l'onglet *Toolchain* et sous la section *AVR/GNU C Compiler*, cliquez sur *Symbols*. Remplacez la valeur 5000 UL de `SYSTEM_MILLIVOLTAGE` par 3300 UL (UL force le type en *unsigned long*). Reste à compiler le code et à l'écrire dans la mémoire du contrôleur.

Avec ces 3,3 V, j'ai mesuré un courant de 5,76 mA, soit un gain de 6,6 mA. Autrement dit, ce simple changement a réduit de plus de moitié la consommation !

3, 4, 5. Ralentir le processeur.

Nous allons maintenant étudier l'effet d'un ralentissement du processeur sur la consommation. Pour cela nous allons changer le quartz externe et baisser la fréquence d'horloge. Pour chacune de ces trois étapes (**tableau 2**), le symbole de compilation `F_CPU` doit être modifié pour correspondre à la fréquence d'horloge effective (dans Atmel Studio, entrez les fréquences de la colonne F_CPU sans mettre d'espace entre les zéros). Suivez la procédure décrite plus haut pour chacune des lignes du tableau : connexion du programmateur, réglage des fusibles, changement de `F_CPU` dans le code, compilation et transfert du code, déconnexion du programmateur et, le cas échant, changement de quartz.

Aucun doute d'après le tableau, fréquence du processeur et consommation de courant sont liées. La fréquence de 500 kHz semble le meilleur choix, malheureusement le fonctionnement de l'USART de l'Atmega n'est pas fiable à des fréquences aussi faibles (cf. la fiche technique, ainsi que le « calculateur de débit binaire » du lien [6]). Arrière toute, donc.

6. Utiliser le quartz interne

J'ai ensuite utilisé le quartz interne de 8 MHz avec un diviseur de 8 pour obtenir une fréquence réelle de 1 MHz. Cette valeur est suffisante pour établir des communications série fiables à des débits aussi bas que 4800 bauds – valeur qui du reste suffit pour le menu du démarrage. Vous pouvez retirer le quartz externe après avoir positionné le fusible LFuse sur 0x62, paramétré le symbole `F_CPU` sur 1000000UL et transféré le code compilé dans l'ATmega.

J'ai mesuré une consommation de 748 µA. C'est moins bien qu'avec le cadencement à 500 kHz, mais beaucoup mieux que les 12,36 mA du départ !

7. Désactiver le BOD

L'ATmega possède un détecteur de baisse de tension appelé BOD (*Brown Out Detector*), chargé de déclencher une initialisation de la puce lorsque la tension d'entrée devient inférieure à un certain niveau. Les quelques microampères à payer pour bénéficier de cette fonction peuvent être évités en positionnant le fusible EFuse sur 0x07, ce qui désactive le BOD. Le gain obtenu n'est que de 6 µA, mais chaque économie compte.

8. Veille

Il est temps de passer à l'optimisation du code. Dans la version non optimisée (fig. 2), la temporisation entre deux lectures s'effectue à l'intérieur d'une boucle. A priori une boucle d'attente ne fait… qu'attendre, mais le processeur doit bien égrener les secondes, ce qui, bien sûr, consomme du courant. Pour réduire leur consommation, la plupart des contrôleurs peuvent être mis en veille (ce qui éteint le processeur et certains périphériques) et être réveillés à l'aide d'interruptions déclenchées soit par le changement de niveau d'une broche, soit par une communication entrante, ou encore au bout d'un laps de temps spécifique. Parmi les six modes de veille dont dispose l'AVR, j'ai choisi celui appelé *Power-save*. Il permet d'utiliser le temporisateur Timer2 pour réveiller le processeur au bout d'un certain temps. Le Timer2 est en fait le seul à pouvoir sortir le processeur de sa veille. Pour économiser l'énergie et travailler avec des échelles temporelles plus faciles à manipuler (voir **l'encadré** *Tempérer le temporisateur*), le Timer2 est configuré pour être cadencé

par un quartz horloger.
Aucun fusible à positionner pour cette optimisation, mais donc un quartz de montre de 32,768 kHz à insérer dans la T-Board 28. Veillez à ne pas endommager les fines connexions et à ce que le contact avec le connecteur de la carte soit correct. Ensuite chargez, compilez et transférez la version 2 (optimisée) du code. Ici la LED ne clignote que quatre fois à la fin de chaque lecture. J'ai attendu que le courant se stabilise après ces quatre clignotements. Résultat de la mesure : 59 µA. Cette consommation permettrait à une pile de 800 mAh d'alimenter le circuit pendant un an et demi, mais on peut encore faire mieux !

9. Désactiver les périphériques inutiles

L'ATmega328 possède un registre pour désactiver les périphériques non utilisés, le *Power Reduction Register* (PRR). La fonction `reducePower()` désactive Timer0, Timer1, SPI et USART. Nous ne touchons ni au bus TWI ni au Timer2 puisque nous en avons besoin pour communiquer avec l'EEPROM et réveiller le processeur. Le CAN fait partie du processus de conversion, donc aucune optimisation supplémentaire n'est à chercher de ce côté-là. La fonction `reducePower()` active en outre les résistances de rappel vers le haut associées aux broches inutilisées, ce qui évite le surplus de courant que pourraient engendrer des broches laissées dans un état indéfini. Je n'ai pas réussi à observer la moindre différence avec cette configuration, mais d'après les faibles pourcentages d'économie que donne la fiche technique, l'économie n'est sans doute réelle qu'avec un processeur en mode actif.

10. Une dernière modification

J'ai comparé mes résultats avec les données de la fiche technique et me suis senti insatisfait. Le processeur n'était bien sûr pas le seul composant à consommer du courant, il y avait aussi l'EEPROM et le LM60. Après quelques tests rapides et épluchages de fiches techniques, j'ai trouvé que l'EEPROM ne consommait que 100 nA en mode veille, et que le courant de repos du LM60 valait environ 82 µA. J'ai alors modifié le montage pour que le LM60 soit alimenté par la broche PD7 de la T-Board (elle peut délivrer jusqu'à 20 mA), puis changé le code de façon à ce que le capteur ne soit alimenté que le temps nécessaire à une lecture. J'ai ainsi ramené la consommation à un incroyable et unique microampère (la précision maximale de mon ampèremètre). Champagne ! Après avoir trinqué au succès de ma méthode, j'ai effectué quelques mesures de plus pour finalement évaluer le pic de courant absorbé durant une mesure de température à 65 µA. En l'arrondissant à 1 mA pour faire bonne mesure, et après avoir calculé quelques moyennes, j'en suis arrivé à la conclusion qu'une pile de 800 mAh pourrait alimenter le capteur durant six années, une valeur plus qu'acceptable pour ce type de circuit.

Tableau 1. Résultats des optimisations

Étape	Description	F_CPU	SYSTEM_ MILLIVOLTAGE	LFUSE	EFUSE	Courant (mA)
1	16 MHz et 5 V	16 000 000 UL	5000 UL	0xFF	0x05	12,360
2	réduire à 3,3 V	16 000 000 UL	3300 UL	0xFF	0x05	5,760
3	utiliser un quartz externe de 8 MHz	8 000 000 UL	3300 UL	0xFF	0x05	3,360
4	utiliser un quartz externe de 4 MHz	4 000 000 UL	3300 UL	0xFD	0x05	2,130
5	diviser l'horloge par 8	500 000 UL	3300 UL	0x7D	0x05	0,560
6	quartz interne 8 MHz + division par 8	1 000 000 UL	3300 UL	0x62	0x05	0,748
7	désactiver BOD	1 000 000 UL	3300 UL	0x62	0x07	0,742
8	veille avec Timer2	1 000 000 UL	3300 UL	0x62	0x07	0,059
9	désactiver les périphériques non utilisés	1 000 000 UL	3300 UL	0x62	0x07	0,059
10	désactiver le capteur quand non utilisé	1 000 000 UL	3300 UL	0x62	0x07	0,001

Tableau 2. Paramètres du quartz pour les étapes 3,4,5.

Étape	Quartz	Diviser par 8 ?	Fréquence réelle	F_CPU	LFUSE	Courant
3	8 MHz	non	8 MHz	8 000 000 UL	0xFF	3,36 mA
4	4 MHz	non	4 MHz	4 000 000 UL	0xFD	2,13 mA
5	4 MHz	oui	500 kHz	500 000 UL	0x7D	0,56 mA

Liens

[1] T-Boards 8/14/28, Elektor septembre 2014, www.elektor-magazine.fr/130581

[2] Atmel Studio : www.atmel.com/atmelstudio

[3] USB Tiny : www.crash-bang.com/using-usbtiny-with-atmelstudio/

[4] Fuse Calculator : www.engbedded.com/fusecalc/

[5] AVRDude : http://savannah.nongnu.org/projects/avrdude

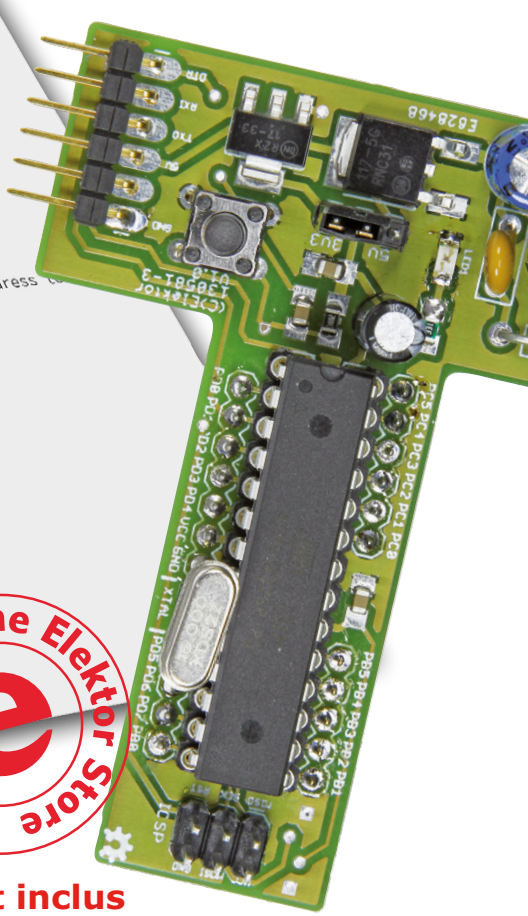
[6] Baud Calculator : www.wormfood.net/avrbaudcalc.php

[7] Téléchargement versions 1 et 2 : www.elektor-magazine.fr/140413

loge du processeur, car moins de cycles d'horloge par seconde équivalent à moins de courant consommé. Le troisième consiste à mettre le processeur en veille lorsqu'il n'est pas utilisé. Enfin, dernier levier, on peut désactiver les périphériques et composants non utilisés pour encore diminuer la consommation.

Il est facile d'agir sur les deux premiers leviers grâce à la conception flexible de la T-Board 28 : on peut sélectionner la tension d'entrée, et on peut relier un quartz externe pour changer de fréquence d'horloge. Les deuxième et troisième leviers sont implantés dans le code. Autre aide de la T-Board, le sélecteur de tension : c'est lui qui nous servira de point utile pour mesurer la consommation lorsque nous chercherons à évaluer l'efficacité des optimisations apportées. En plus des quatre points précités, on peut encore améliorer le rendement en écrivant p. ex. un code

```
void printLog(void)
{
   uint16_t addressCounter;
   uint8_t readData = 0;
   uint8_t result;
   uint8_t iCount;

   //Print Header
   UART_writeString("******\r\n");
   UART_writeString("Printing Log\r\n");
   UART_writeString("******\r\n\r\n");
   UART_writeString("Memory_Location, Day, Hour, Minute, Second, Log_Value\r\n");


   //Send START Condition
   result = I2C_sendStart();

   //Send the device Address with WRITE - we need to write in order to specify the address
   result = I2C_send(EEPROM_DEVICE_ADDRESS|TW_WRITE);

   //Send Memory Location Address to read from (High)
   result = I2C_send(EEPROM_FIRST_ADDRESS >> 8);     //Address High

   //Send Memory Location Address to read from (Low)
   result = I2C_send((uint8_t)EEPROM_FIRST_ADDRESS); //Address Low


   //Send RESTART Condition - now we read from the memory address
   result = I2C_sendStart();

   //Send the device Address with READ
   result = I2C_send(EEPROM_DEVICE_ADDRESS|TW_READ);
```



plus efficace pour réduire le nombre de cycles d'horloge nécessaire à la réalisation de certaines tâches, et ainsi réduire les exigences en courant. Comme je l'ai dit plus haut, l'optimisation du code n'a pas été ma priorité ; ce serait d'ailleurs un sujet à lui tout seul !

Mesure de la consommation de courant

J'ai supposé que beaucoup d'utilisateurs de la T-Board n'étaient pas équipés d'un vrai laboratoire de mesure. Puisque c'est aussi mon cas, pour mesurer le courant je me suis servi d'un multimètre numérique bon marché. Je l'ai mis en mode ampèremètre et, à l'aide de connecteurs femelles, j'ai relié les sondes au connecteur de sélection de tension de la T-Board pour mesurer la consommation de la carte à un instant donné. En effet, la position des cavaliers dans le circuit d'alimentation électrique des moTules T-Boards est telle que si on les remplace par un ampèremètre, cet instrument donnera une indication relativement fiable de la consommation du processeur AVR pendant l'exécution du code. La présence de l'ampèremètre n'a pas d'effet sur la tension.

Pour mesurer le courant avec plus de précision, j'ai dessoudé et retiré la LED témoin, car elle absorbait beaucoup plus de courant que le contrôleur. La consommation ainsi réduite et optimisée, j'ai pu mettre mon multimètre sur le calibre µA et mieux évaluer l'effet des petits changements apportés.

Hormis son manque de précision, une des limites de cette méthode de mesure est bien sûr que le courant n'est mesuré qu'à un instant donné. L'idéal aurait été d'enregistrer le courant consommé sur un certain intervalle de temps afin de pouvoir en calculer la moyenne (projet du reste suffisamment utile pour faire l'objet d'un futur article). C'est sur ce point que la LED reliée à PB0 s'est révélée pratique puisqu'elle m'indiquait les moments actifs du programme, et donc le contexte dans lequel je mesurais le courant.

Optimisation pas-à-pas

Je vais maintenant montrer comment je suis parvenu à réduire progressivement le courant consommé, quels changements j'ai effectués, et quels ont été leurs effets. J'ai suivi la procédure suivante pour chacune des dix étapes d'optimisations décrites :

1. changement des fusibles avec le programmateur ISP connecté ;
2. modification du code, compilation et transfert dans la T-Board ;
3. déconnexion du programmateur ISP ;
4. modifications physiques du circuit ;
5. pile de 9 V reliée au jack de 2,1 mm et mesure du courant avec le multimètre.

Aucun câble/module FTDI ne doit être connecté durant les mesures.

Avant de commencer, une remarque sur les programmateurs : Atmel Studio [2] en prend en charge un certain nombre, qui ne nécessitent donc que peu ou pas de configuration spéciale. D'autres, souvent bon marché, ne sont pas pris en charge de façon native mais peuvent tout de même être configurés pour fonctionner avec l'EDI (USBTiny, USBasp, etc.) Le chargement du code dans le contrôleur mis à part, la différence principale réside alors dans la façon de configurer les fusibles (voir **l'encadré** *Ne grillez pas vos fusibles !*) Différentes ressources en ligne expliquent comment paramétrer Atmel Studio pour qu'il prenne en charge ces programmateurs tiers [3].

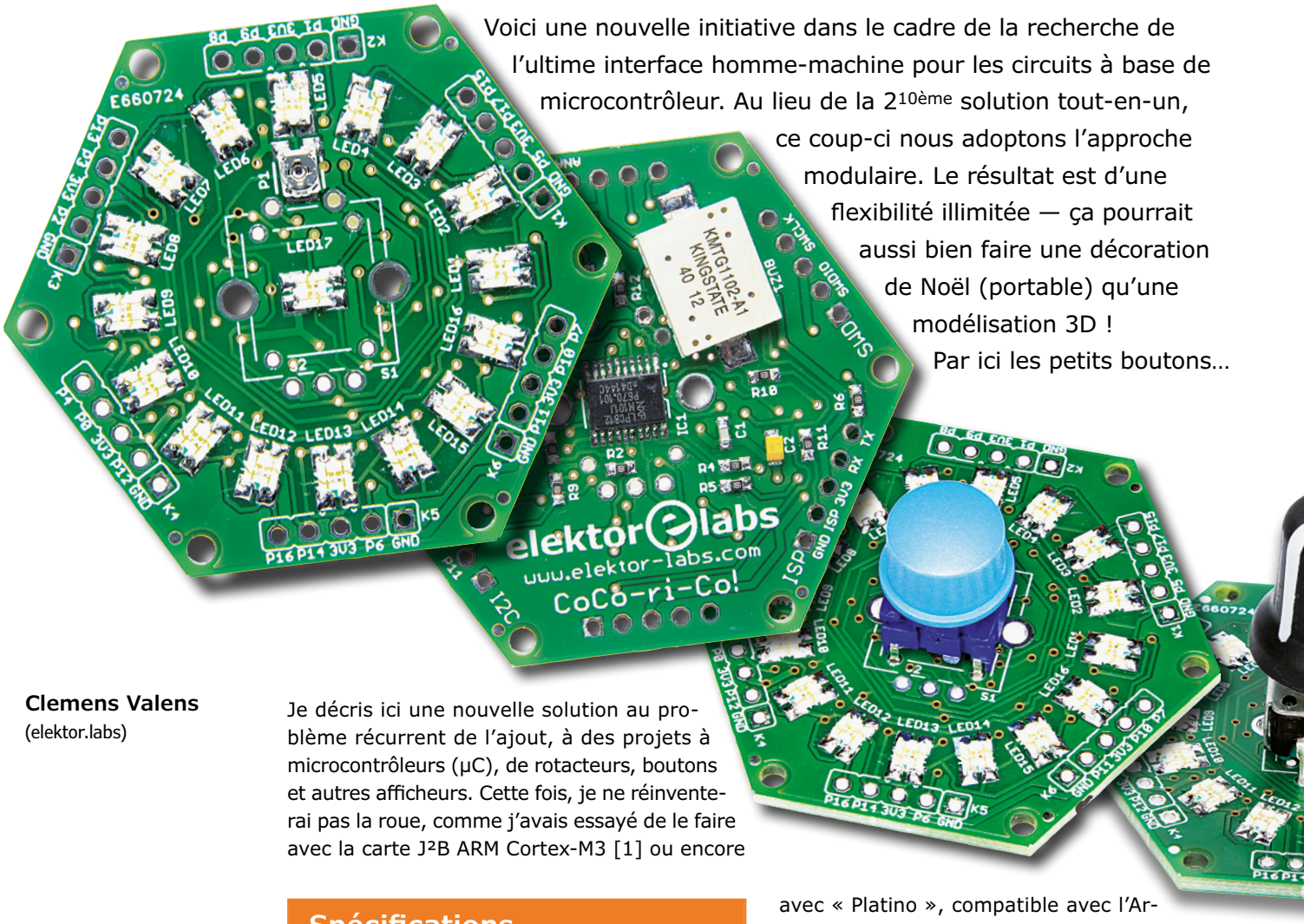
Faites le maximum pour le minimum

Concluante, utile, cette méthode aura aussi mis en lumière les avantages que procure une plateforme à la fois flexible et rationalisée comme les moTules T-Board. Parti d'une consommation « Arduino » de 12,3 mA, je suis arrivé à 13 µA (valeur moyenne établie en fonction des temps passés en modes veille et actif), soit une réduction de 99,89 % !

Sans oublier d'enfiler votre t-shirt bien sûr, tentez d'autres optimisations, p. ex. avec une T-Board 8 ou avec un capteur RF pour transmettre les mesures.

(140413 – version française : Hervé Moreau)

CoCo-ri-Co:
Cool Controller Concept



Voici une nouvelle initiative dans le cadre de la recherche de l'ultime interface homme-machine pour les circuits à base de microcontrôleur. Au lieu de la 2[10ème] solution tout-en-un, ce coup-ci nous adoptons l'approche modulaire. Le résultat est d'une flexibilité illimitée — ça pourrait aussi bien faire une décoration de Noël (portable) qu'une modélisation 3D !

Par ici les petits boutons…

Clemens Valens
(elektor.labs)

Je décris ici une nouvelle solution au problème récurrent de l'ajout, à des projets à microcontrôleurs (µC), de rotacteurs, boutons et autres afficheurs. Cette fois, je ne réinventerai pas la roue, comme j'avais essayé de le faire avec la carte J²B ARM Cortex-M3 [1] ou encore avec « Platino », compatible avec l'Arduino, la carte ATmega universelle [2]. Ces cartes offrent un nombre variable de codeurs rotatifs et de poussoirs, ainsi qu'un choix de tailles de LCD, afin de satisfaire *tous* les besoins du concepteur pour ses commandes.

Spécifications

- NXP LPC812 32-bit ARM Cortex-M0+
- Toutes les 18 broches d'E/S du microcontrôleur sont accessibles via des connecteurs d'extension
- Jusqu'à 17 LED bicolores
- Codeur rotatif et/ou bouton-poussoir
- Vibreur acoustique (*buzzer*)
- Supporte I²C, la communication sérielle SPI synchrone et asynchrone
- Port ISP compatible avec câble sériel USB FTDI 3,3 V (hormis alimentation 5 V)

Au lieu de nombreux organes de commande implantables de différentes façons, ce projet n'en comporte qu'un seul. Au lieu d'un grand circuit imprimé polyvalent censé tout intégrer, ce sera un petit module à répéter autant de fois qu'il le faut. Plus de volumineux LCD, aux messages cryptiques, en polices laides, difficiles à lire. Cette

fois, rien que des LED. C'est plus cool : visibles de très loin, clignotant rapidement, lentement, voire pas du tout.

Compromis passéiste

Les appareils électroniques utilisent des boutons, rotatifs ou non, pour que l'utilisateur ajuste tel ou tel paramètre(s). Pour certains de ces organes, peu importe l'indication précise de la valeur du paramètre réglé ; la commande de volume d'un amplificateur en est un bon exemple. Une fois que le niveau souhaité est atteint, on cesse de tourner le bouton. Pour d'autres commandes – les sélecteurs de fonction par exemple – il faut une indication pour que l'utilisateur visualise leur position, mais sa résolution est secondaire ; quelques LED font l'affaire. Pour les réglages précis, la commande est associée à un affichage (alpha)numérique.

Les commandes rotatives, pourtant rapides et confortables, sont souvent remplacées par des poussoirs +/–. Ceux-ci, certes petits, bon marché et faciles à commander par µC, sont en fait le résidu d'un compromis datant de l'époque où les ressources du µC (mémoire, E/S, puissance de traitement, etc.) coûtaient cher. Ce n'est plus le cas aujourd'hui. Débarrassons-nous de ces compromis passéistes.

Réinventer la roue

Le concept décrit ici est révolutionnaire (jeu de mots) : un seul codeur rotatif, jusqu'à 17 LED bicolores (c.-à-d. 34 LED au total) et un vibreur, tous commandés par le même µC à base d'ARM Cortex-M0+ à 32 bits. Excessif ? Peut-être, si l'on ne tient compte que de la puissance de traitement, mais pas si l'on met les coûts et le potentiel d'interconnexion dans la balance. Le µC que j'ai choisi pour ce projet est le LPC812M101JDH20FP, le LPC812, à 18 ports d'E/S, 16 Ko de mémoire Flash, 4 Ko de RAM, un bon nombre de temporisateurs (*timer*), un comparateur analogique, une interface I²C, deux contrôleurs SPI et trois interfaces de communication série asynchrone. De plus, grâce à une matrice de commutation intégrée, les broches de ces périphériques de communication peuvent être connectés à *n'importe lequel* des 18 ports d'E/S disponibles. Cette fonction très pratique sous-entend que l'on peut construire un port de communication multi-protocole à peu de lignes, configurable à la volée par logiciel, pour parler I²C ou encore sériel synchrone (SPI ou similaire) ou asynchrone. Une broche de plus, et la communication devient *full duplex*. Inutile de parler du prix, on perdrait de l'argent.

Tous les composants sont montés sur un tout petit circuit imprimé (PCB) : le µC, une LED et le codeur rotatif au centre, les 16 autres LED autour du codeur. Sur le bord, des embases permettent de relier le module à d'autres pour créer une surface de commande, voire pour le câbler à une autre carte-hôte.

C'est cool tout ça : la carte, le µC ARM Cortex-M0+, les LED bicolores, le codeur rotatif. Alors je l'ai appelé *Cool Controller Concept*. Et comme la carte est hexagonale, c'est devenu « CoCo-ri-Co ».

Pour qui ?

L'application type de CoCo-ri-Co, c'est le potentiomètre numérique avec indication de la position, utilisable sur tout appareil qui pourrait tirer bénéfice d'une ou plusieurs commandes rotatives pour le réglage de ses paramètres. Commande de volume et de tonalité des amplificateurs audio, sélecteurs de fonction, commandes de vitesse d'appareils ménagers, commande de température de chauffage, réglages de luminosité et de couleur de lampes – toutes ces applications conviendraient pour un ou plusieurs Coco-ri-Co. Ce module est en mesure de communiquer par protocoles série, I²C, SPI, ou d'autres protocoles personnalisés. Il utilise pour cela un seul port à 3 broches, facile à connecter à d'autres applications-hôtes à µC. Il peut aussi faire office de contrôleur principal pour des applications n'en possédant pas.

CoCo-ri-Co est assez petit pour tenir même dans de petits boîtiers et dans les boîtes électriques encastrables. En interconnecter plusieurs permet de créer un tableau de bord par exemple pour un véhicule électrique (**fig. 1**). Un CoCo-ri-Co avec commande par codeur en tant qu'indicateur de vitesse, un autre – sans codeur – pour l'affichage de la tension de la batterie et un troisième visualisant le régime (tr/min) du moteur ou sa température (voire les deux).

Il est possible d'alimenter CoCo-ri-Co par pile, une pile-bouton de 3 V p. ex., ce qui le rend portable. Les LED bicolores permettent des applications

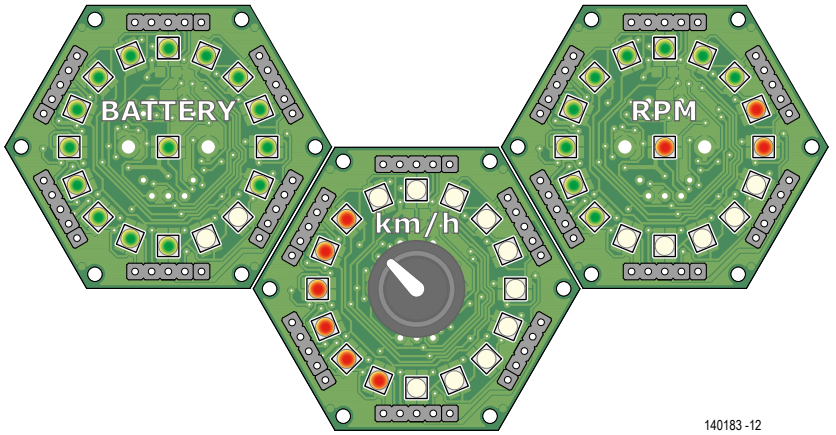


Figure 1.
Trois CoCo-ri-Co assemblés forment le tableau de bord d'un go-kart électrique ou tout autre véhicule futuriste.

décoratives, même une guirlande de Noël, sonore grâce à son vibreur acoustique (voir encadré). L'assemblage de plusieurs CoCo-ri-Co donne des tableaux de commande, pas nécessairement plans. *Cool Controller Concept*, ce pourrait être aussi des cartes pentagonales, carrées ou triangulaires, des formes susceptibles de composer ce que l'on appelle des solides d'Archimède. Le plus connu des polyèdres convexes, semi-réguliers, symétriques, composés de deux types de polygones réguliers ou plus, réunis en sommets identiques, est le ballon de foot. Cet icosaèdre tronqué comporte 20 hexagones et 12 pentagones. Un ballon de foot fait de cartes CoCo-ri-co (**fig. 2**) aurait un diamètre de quelque 12 cm (arête de 25 mm). Dotez-le d'un accéléromètre 3D et il peut visualiser des animations de LED et produire des sons en fonction de sa position, sa vitesse et son mouvement. CoCo-ri-Cool !

Figure 2.
Laissez votre créativité s'exprimer par platines polygonales interposées. Voici un quasi ballon (expérimental) fait de 20 CoCo-ri-Co hexagonaux et trois pentagonaux.

Comment ça marche ?

Chaque fois que la position du codeur rotatif change, CoCo-ri-Co envoie une valeur (croissante ou décroissante, selon le sens de rotation) au port de communication (par défaut : UART, 115200n81). Cette valeur se situe entre deux limites (de 0 à 100 par défaut). L'anneau de LED affiche la progression de la valeur comme un barregraphe (par défaut) ou un point, rouge (par défaut) ou vert. La LED centrale est commandée séparément. Une action sur le bouton-poussoir du codeur produit un message *touche enfoncée* (*Key-Down*) et son relâchement un message de *touche relâchée* (*Key-Up*). Les messages peuvent avoir les formats suivants :

Mode ASCII

- Codeur rotatif : E*xxxxx*, où *xxxxx* est un champ 5 chiffres véhiculant des valeurs comprises entre « 00000 » et « 65535 ».
- Bouton-poussoir : B*x*, où *x* = « 0 » (bouton enfoncé) ou « 1 » (bouton relâché).

Mode binaire

- Codeur rotatif : E*yz*, où la valeur est calculée en tant que 256 *y* + *z*.
- Bouton-poussoir : B*y*, où *y* = « 0 » (bouton enfoncé) ou « 1 » (bouton relâché).

Vous pouvez modifier le comportement du programme à la volée par l'envoi de commandes à CoCo-ri-Co. Cela vous permet de modifier certaines choses comme la couleur des LED, les limites de la plage du codeur ou la valeur du codeur lui-même, le mode de ce dernier, la gamme de LED à utiliser (est, par défaut 0 à 15). Vous pouvez également faire pivoter le barregraphe, produire un signal sonore, etc. Au moment d'écrire ces lignes, je rajoute sans cesse des commandes utiles ; la liste complète sera dans le téléchargement gratuit associé à cet article. Les mises à jour pour ce projet seront postées sur Elektor.Labs [3]. Le **tableau 1** répertorie les commandes implémentées à ce jour.

Configuration

La configuration de CoCo-ri-Co au moyen de P1 se résume au choix du type de port de communication à utiliser (UART, SPI/synchrone ou I²C).

Tableau 1. Commandes pour modifier les paramètres à la volée (* identifie la valeur par défaut). Chaque commande requiert 2 octets.

Commande	Donnée	Description
LED		
0x00 (0)	0|1*|2	Couleur anneau de LED : arrêt | rouge* | vert
0x01 (1)	0*|1	Mode anneau de LED : barregraphe* | point
0x02 (2)	0*-15	Première LED anneau de LED (0*)
0x03 (3)	0-15*	Dernière LED anneau de LED (15*)
0x04 (4)	0*|1	Direction anneau de LED : horaire* | anti-horaire
0x05 (5)	0*-15	Rotation anneau de LED (0*)
0x06 (6)	0-15	Anneau de LED, LEDx allumée
0x07 (7)	0-15	Anneau de LED, LEDx éteinte
0x08 (8)	0|1*|2	LED du centre : arrêt | rouge* | vert
Codeur rotatif		
0x10 (16)	0*|1	Mode codeur : normal* | accéléré
0x11 (17)	0-255 (0)	Minimum codeur, octet poids faible (0*)
0x12 (18)	0-255 (0)	Minimum codeur, octet poids fort (0*)
0x13 (19)	0-255 (64)	Maximum codeur, octet poids faible (64*)
0x14 (20)	0-255 (0)	Maximum codeur, octet poids fort (0*)
0x15 (21)	0-255 (0)	Valeur codeur, octet poids faible (0*)
0x16 (22)	0-255 (0)	Valeur codeur, octet poids fort (0*)
Vibreur acoustique		
0x20 (32)	0|1*	Vibreur activé | désactivé*
0x21 (33)	1	Bip vibreur maintenant
0x22 (34)	0-255 (232)	Fréquence bip vibreur [Hz], octet poids faible (232*)
0x23 (35)	0-255 (3)	Fréquence bip vibreur [Hz], octet poids fort (3*)
0x24 (36)	0-55 (64)	Durée bip vibreur [Hz], octet poids faible (64*)
0x25 (37)	0-255 (0)	Durée bip vibreur [Hz], octet poids fort (0*)
Divers		
0xfe (254)	0*|1	Mode en sortie : ASCII | binaire
0xff (255)	1	Restaurer valeurs par défaut

Avec P1, vous pouvez, en théorie, sélectionner jusqu'à 32 valeurs, mais pour des raisons pratiques – dont principalement l'imprécision de P1 – ce nombre a été limité à huit plages. Le **tableau 2** récapitule les fonctions activées par une plage. Les plages sont définies de telle sorte que la rotation de P1 à sa valeur minimale sélectionne le mode UART, la mise au centre donne SPI/synchrone et, en butée au maximum, CoCo-ri-Co parlera à I²C. Il existe également une procédure plus précise ; voilà comment y accéder :

1. Couper l'alimentation de la carte.
2. Implanter un cavalier entre P6 et GND (K5, broches 1 et 2).comprises entre « 00000 » et « 65535 ».
3. Mettre la carte sous tension.
4. Jouer sur P1 pour que, sur l'anneau de LED, la

Tableau 2. Configurations fixées par P1. En mode SPI/synchrone, s'assurer que, lors de la mise sous tension du système, la ligne SCK (P12) est au niveau logique haut (3,3 V), sinon CoCo-ri-Co démarrera en mode ISP.

Plage	Configuration	Note	Signaux
1	UART, sans parité, 8 bits de données, 1 bit d'arrêt (n81)	débit = 115200	P4=TxD, P0=RxD
2		débit = 38400	
3		débit = 9600	
4	SPI/synchrone, Esclave		P4=MISO, P0=MOSI, P12=SCK (SSEL=?)
5			
6	I²C, Esclave	adresse 0x1b (29)	P4=SDA, P0=SCK (P12=IRQ)
7		adresse 0x3a (58)	
8		adresse 0x74 (116)	

plage voulue s'allume. Restez au milieu de la plage pour éviter des changements de mode aléatoire.

5. Couper l'alimentation de la carte.
6. Enlevez le cavalier entre P6 et GND (connecteur K5, broches 1 et 2).
7. Remettez la carte sous tension et faites vos essais.

Ménagez P1, c'est un petit composant fragile. Remarquez que les modes SPI/synchrone et I²C Esclave requièrent que l'application hôte interroge CoCo-ri-Co à intervalle régulier. Vous pourriez utiliser un troisième port pour interrompre l'hôte en cas de disponibilité de nouvelles données. À noter également que le mode SPI Esclave réel utilisant le périphérique SPI du µC nécessite le signal SSEL. À vous de décider quel port utiliser pour cela. Le « SPI » implémenté utilise l'USART en mode synchrone capable de travailler sans signal SSEL. Si le mode Esclave ne vous convient pas, il y a aussi le mode Master, mais il faudra programmer un peu.

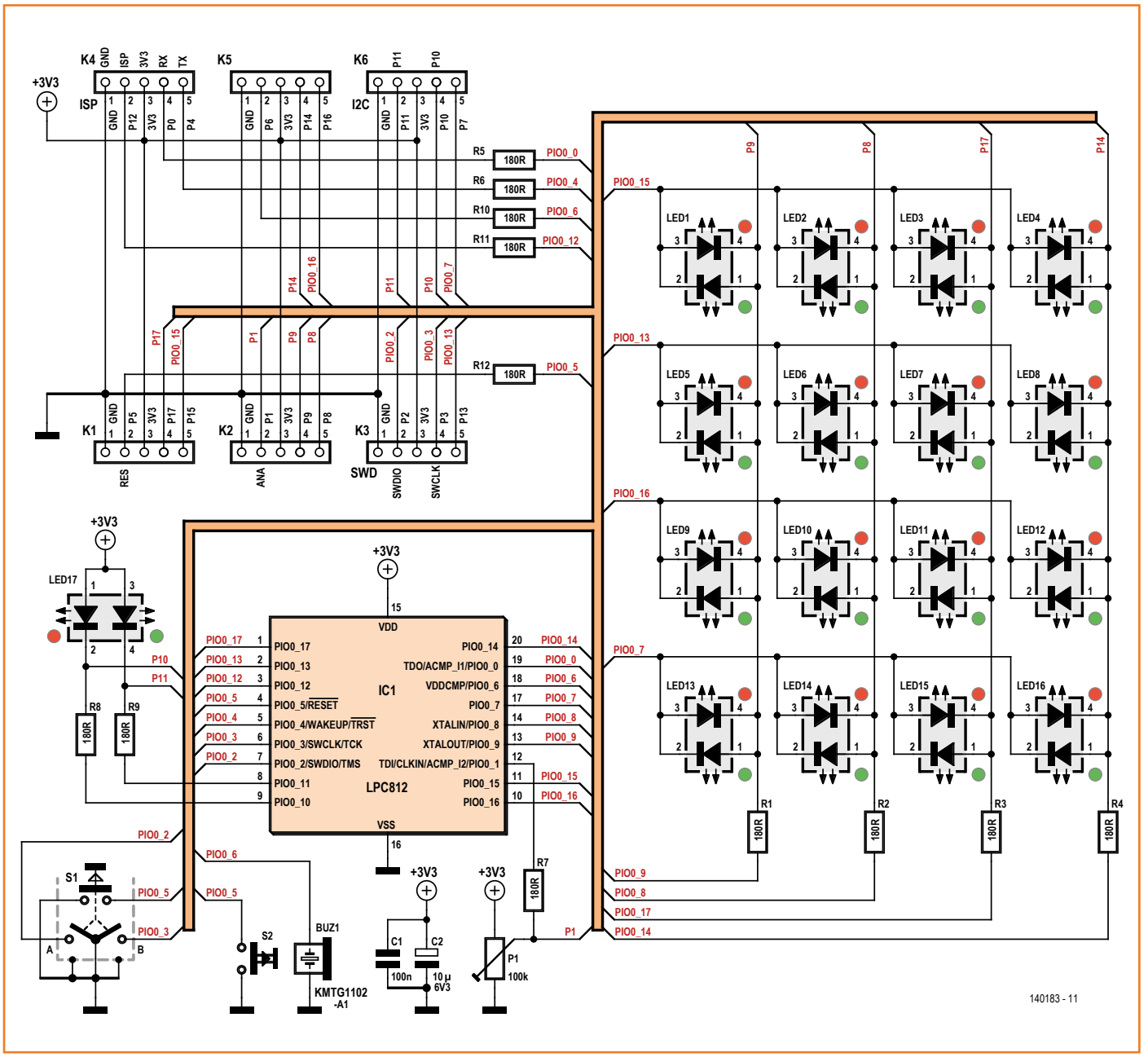


Figure 3.
Le schéma de CoCo-ri-Co. Le µC LPC812 ARM Cortex-M0+ possède largement assez de cervelle et de muscles de se charger de tout.

Le schéma

Jetons un coup d'œil aux schémas (**fig. 3**). Connecter 34 LED (bicolores) directement à 18 ports d'E/S est possible sous la forme de deux matrices 4 x 4 (16 LED par matrice) anti-parallèles. Elles n'occupent que 8 broches maximum. Ces 16 LED forment le cercle. Il nous reste donc une LED bicolore placée au centre de la carte qu'il nous faudra relier d'une autre façon. Ce que l'on sait moins est que le module I²C du µC LPC812 peut prendre en charge deux types d'interfaces : « normale », en mesure d'utiliser n'importe quelle broche d'E/S et « dédiée » qui insiste sur l'utilisation des ports 10 et 11. Ces deux ports ont des capacités de drain de courant élevées et conviennent donc parfaitement à la commande des LED ; j'ai donc câblé les LED bicolores restantes aux dits ports. Nous nous satisferons, pour la communication, de l'I²C normal. Des résistances en série avec les LED limitent leur courant. Normalement, les LED rouges et vertes nécessiteraient des résistances de valeur différente pour produire la même perception de luminosité (le rouge étant généralement plus brillant que le vert), mais, dans le cas de la matrice de LED, ceci est impossible. Pour des raisons pratiques, je n'ai qu'une seule valeur de résistance pour toute la carte : 180 Ω.

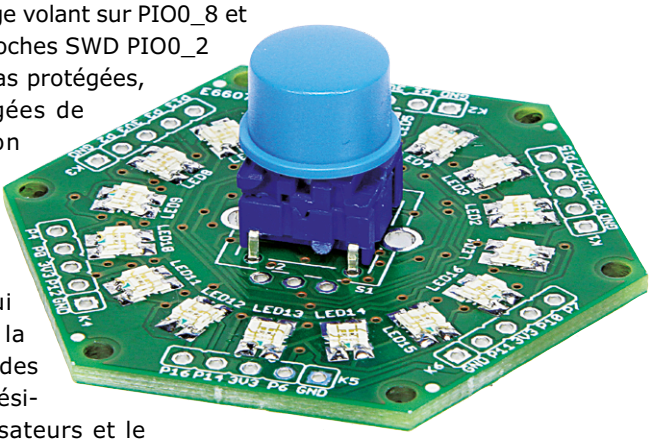
Le codeur rotatif est un modèle à bouton-poussoir intégré ; c'est plus cher, mais j'aime la technique de réglage des paramètres qui consiste à tourner et à pousser en même temps. Le bouton-poussoir est relié à la broche *Reset* (réinitialisation) du µC. Le logiciel permet de désactiver la fonction de réinitialisation de cette broche. Normalement, au cours du développement du logiciel, vous conserverez cette fonction de réinitialisation, mais une fois l'application fonctionnelle, vous pouvez la supprimer. Ou installer un codeur sans bouton-poussoir. Pour les applications ne nécessitant qu'un seul bouton-poussoir, la carte comporte aussi une empreinte spéciale bouton-poussoir, câblée en parallèle avec le bouton-poussoir du codeur.

Pourquoi le codeur se trouve-t-il au centre de la carte à cheval sur LED17 ? Primo, il se peut que votre application n'utilise que l'un des deux composants ; ne montez que celui dont vous avez besoin. Secundo, nous pouvons créer un éclairage de l'axe du codeur. LED17 vous permet d'utiliser l'axe comme un guide de lumière s'il est transparent. L'empreinte du bouton-poussoir se trouve aussi au centre de la carte, mais pour autant que je sache, il n'existe pas de version à boîtier transparent du modèle que j'ai choisi. Quoi qu'il en soit, LED17 peut produire une sorte de lueur d'arrière-plan pour l'organe de commande.

Un module de commande universel doit être « audiblement rétroactifs ». D'où la présence d'une empreinte pour un vibreur acoustique. Pour des raisons de place, c'est un vibreur CMS, malheureusement assez cher.

Une résistance variable, P1, est connectée au port PIO0_1, relié, dans les entrailles du µC, à un comparateur analogique capable de différencier 32 niveaux ; on pourra l'utiliser, par exemple, comme un commutateur à 32 positions permettant de sélectionner différentes fonctions logicielles. Pour faciliter son accès (même s'il est assez petit), il est monté sur la face « LED ». À n'implanter que si vous en avez besoin.

Le quartz intégré dans le µC est excellent. Si vous souhaitez une meilleure horloge, il vous faudra faire un câblage volant sur PIO0_8 et PIO0_9. Les deux broches SWD PIO0_2 et PIO0_3 ne sont pas protégées, les broches de rangées de matrice de LED non plus.



Connecteurs d'extension

Le µC se trouve, lui aussi, au centre de la carte, mais du côté des soudures, avec les résistances, les condensateurs et le vibreur. La totalité de ses 18 broches d'E/S ont été câblées jusqu'à six connecteurs d'extension à 5 broches, chacun donnant accès à trois ports d'E/S (6 x 3 = 18), plus de 3,3 V et la masse (GND). Ces connecteurs n'ont pas été câblés au petit bonheur la chance, mais de façon à remplir une fonction utile. K1 et K4, par exemple, mettent à disposition des fonctions de programmation in-situ (ISP). K4 est le vrai connecteur ISP, compatible (presque !) avec un câble de convertisseur sériel-USB FTDI 3,3 V et K1 y ajoute la broche de *Reset*. Ils se trouvent sur des bords opposés de la carte, écartés de 3,81 cm (1,5 pouce) ; il est donc facile, si vous le souhaitez, de réaliser un adaptateur de programmation plus élaboré sur carte de prototypage. Un tel adaptateur comporterait probablement un régulateur de 3,3 V pour abaisser le 5 V du câble FTDI 3,3 V (oui, méfiez-

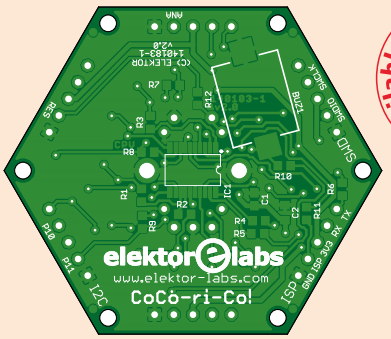
Liste des composants

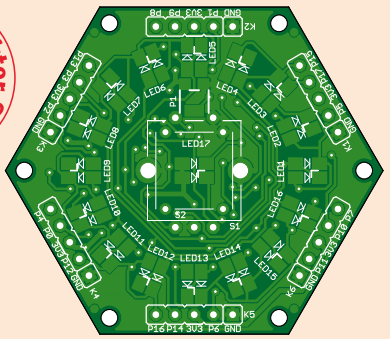
Résistances
R1–R12 = 180 Ω, SMD 0603
P1 = 100 kΩ aj., 3 mm, par ex. Bourns TC33X-2-104E

Condensateurs
C1 = 100 nF, SMD 0603
C2 = 10 µF/6,3 V, SMD 0805

Semi-conducteurs
IC1 = LPC812M101JDH20FP (TSSOP20)
LED1–LED17 = LED, bicolore rouge/vert, Dialight 5988610207F

Divers
BUZ1 = vibreur, Kingbright KMTG1102-A1
S2 = bouton-poussoir, Multimec 3FTL6





S1 = codeur rotatif avec/sans bouton-poussoir intégré, par ex. Alpes EC12E2424407
K1 à K6 = embase mâle à 5 contacts, horizontale ou verticale (optionnel)

Module assemblé : e-choppe Elektor ref. 1401830-91, Cf. www.elektor.fr
Optionnel : circuit imprimé nu : e-choppe Elektor ref. 140183-1 (v2.0)

vous !) à 3,3 V. En fait, l'adaptateur USB-sériel Elektor 110553 BoB-FT232R [4] est préférable au câble FTDI car il peut fournir également le 3,3 V. K2 est considéré comme étant un port analogique, car relié à l'entrée 2 du comparateur analogique du µC (ACMP_I2, PIO0_0 peut être utilisé comme entrée 1). Vous pouvez l'utiliser, en combinaison avec PIO0_8 ou PIO0_9, pour, par exemple, créer un capteur tactile capacitif. Si P1 est montée, vous disposerez d'une tension variable sur la broche 2 de ce connecteur.

K3 fournit un accès au port SWD (*Serial Wire Debug*) du µC. Il partage ce port avec le codeur rotatif, mais, au repos, ses contacts devraient être ouverts (si vous utilisez un modèle avec crans et qu'il est positionné entre les crans). Le brochage de K3 ne respecte aucun standard de *pod* SWD qui me soit connu, de sorte qu'il vous faudra probablement un adaptateur ou un autre.

K5 est destiné aux E/S numériques. Sa broche 2 (PIO0_6) a une résistance de limitation de courant, car elle ne supporte pas le 5 V (toutes les autres broches, si). Les autres résistances de limitation de courant sont prises sur les ports fréquemment connectés et déconnectés.

K6 est l'interface I²C à pleine vitesse, connectée aux broches 10 et 11 (et à la LED au centre).

Bien sûr, la plupart des ports des connecteurs d'extension sont partagés avec la matrice de LED et les autres périphériques embarqués, alors méfiez-vous lorsque vous optez de câbler les ports d'extension à toutes sortes d'autres composants. Le **tableau 3** récapitule les relations entre les ports d'E/S, les connecteurs et les fonctions.

Il y est clair que la fonction du connecteur K4 est la communication avec l'application-hôte. En mode ISP (c.-à-d. lorsque le µC est réinitialisé tout en forçant PIO0_12 au niveau bas),

Tableau 3. Ports d'E/S en relation avec les connecteurs d'extension et les fonctions

PIO	K	Fonction	PIO	K	Fonction	PIO	K	Fonction
0	4	RxD/Libre	6	5	Vibreur acoustique (buzzer)	12	4	ISP/Libre
1	2	P1/Analogique	7	6	Rangée 3	13	3	Rangée 1
2	3	SWDIO/S1A	8	2	Colonne 1	14	5	Colonne 3
3	3	SWCLK/S1B	9	2	Colonne 0	15	1	Rangée 0
4	4	TxD/Libre	10	6	LED17R/I²C	16	5	Rangée 2
5	1	Reset/ Bouton-poussoir	11	6	LED17G/I²C	17	1	Colonne 2

PIO0_0 et PIO0_4 deviennent un port sériel. En mode de fonctionnement normal, ces broches sont libres et peuvent parfaitement être attribuées, par exemple, au module SPI ou I²C voire à quoi que ce soit d'autre. Une fois sorti de *Reset*, PIO0_12 devient un port d'E/S normal. Notez que PIO0_12 n'a pas besoin d'une résistance de rappel au niveau haut, car le µC en comporte déjà une (comme tous les ports d'E/S d'ailleurs, à l'exception de PIO0_10 et PIO0_11), de sorte que le mode ISP ne sera pas activé accidentellement lorsque cette broche n'est pas connectée.

Le logiciel

NXP a un site dédié aux produits à base de µC LPC [5] ; vous pouvez y trouver des bibliothèques pour chacun d'entre eux ; elles vous éviteront le travail de création de la couche d'abstraction matérielle de bas niveau. Ces bibliothèques sont proposées par paires, l'une pour la famille de puces proprement dite et l'autre pour la carte d'évaluation et/ou de démonstration. Comme ce projet utilise le LPC812 il nous faut jeter un coup d'œil aux sets LPC8xx, et comme nous utilisons les outils de compilation LPCXpresso (**fig. 4**) il nous faudra choisir le set ayant trait à cet outil. Les bibliothèques sont incluses (version 2.01) dans le téléchargement pour le présent article [6].

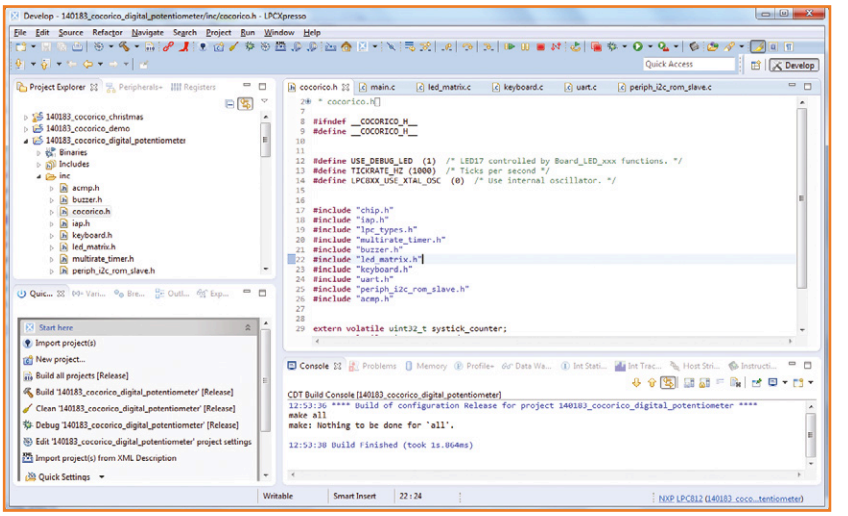
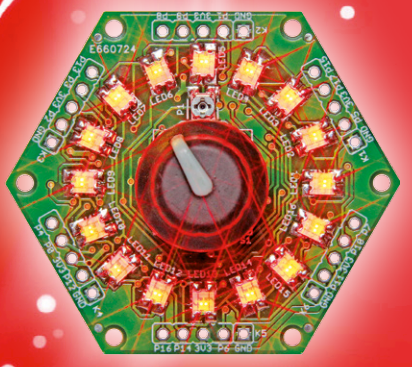


Figure 4. L'environnement (IDE) LPCXpresso IDE ayant servi à développer le logiciel de CoCo-ri-Co. Bien que vous ne puissiez pas le voir dans cette capture d'écran, le projet nommé lpc_chip_8xx_lib fait lui aussi partie de l'environnement de travail.

Je vous invite instamment à ne pas les remplacer par des versions fraîchement téléchargées sans faire, auparavant, une comparaison de fichiers exhaustive, sachant que j'ai corrigé, dans mes bibliothèques, quelques bogues rencontrés ici et là.

L'environnement de travail LPCXpresso pour CoCo-ri-Co comporte deux projets : `lpc_chip_8xx_lib` (la bibliothèque NXP) et `140183_cocorico_digital_potentiometer` (le matériau



Karaoke de Noël

CoCo-ri-Co rassemble 17 LED bicolores (rouge/vert) et un vibreur acoustique sur une minuscule platine, verte, alimentable par pile bouton 3 V. Que vous faut-il de plus pour créer des objets de Noël amusants ? Accrochez-le dans le sapin de Noël, portez-le comme une broche, comme une boucle d'oreille, à votre poignet… emballez-en un, c'est le cadeau original parfait.

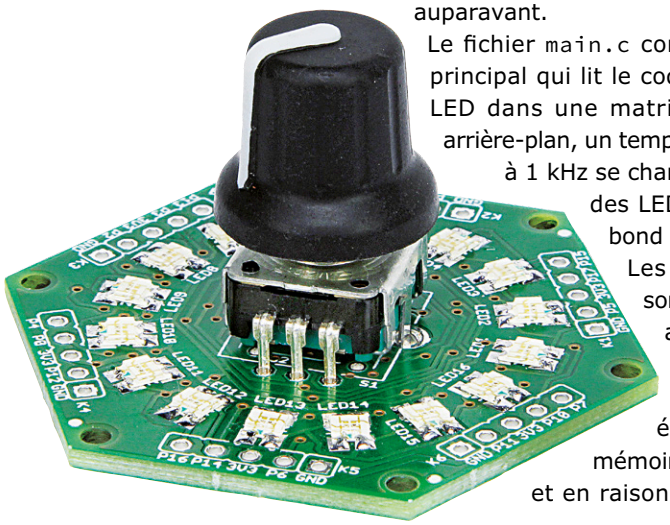
Pour vous lancer, j'ai créé une version CoCo-ri-Co de Noël qui joue *Jingle Bells* tout en clignotant joyeusement. À l'aide d'un adaptateur approprié, connectez le port série à un PC ou un ordinateur portable et utilisez un programme de terminal, tel que Tera Term, pour afficher les paroles en style Karaoké. Rassemblez vos proches autour de l'écran de l'ordinateur (ou TV, ou projetez-le sur un mur) et chantez tous en chœur avec CoCo-ri-Co... Ambiance garantie.

La LED du centre visualisé le rythme ; le codeur rotatif permet de le régler (le monter à l'arrière pour que la LED du centre reste visible). Cf. [3] et/ou [6] pour les fichiers. La structure du projet est telle qu'il est facile de modifier le morceau et les paroles. Après cette expérience, Noël ne sera plus jamais comme avant.

Ce qu'il faut savoir au sujet des ports d'E/S du LPC812

- Dans le tableau 1, les ports 10 et 11 sont répertoriés comme I²C sans spécification de signal (SDA ou SCK). C'est parce qu'il est de votre ressort de décider quel port véhicule quel signal. Cf. le chapitre *Switch Matrix* du guide de l'utilisateur du LPC81x. Ces ports ont des sorties à drain ouvert et ne devraient pas être utilisés pour les modes SPI ou UART rapides.
- Les ports 2, 3 et 5 reviennent, par défaut, à leurs fonctions SWD et de *Reset* à la mise sous tension et doivent être reconfigurés dans le logiciel avant de pouvoir être utilisés à d'autres fins. La ligne *Pin Enable Register 0* commande lesdites broches.
- En mode numérique, tous les ports supportent le 5 V, exception faite de PIO0_6. Les ports pouvant remplir des fonctions analogiques (PIO0_0, PIO0_1 et PIO0_6) ne supportent pas le 5 V (***5-V intolerant*** en anglais) lorsque leur fonction analogique est activée.
- Si le logiciel désactive l'entrée *Reset*, il reste possible de passer en mode ISP ; il suffit de maintenir PIO0_12 au niveau bas, lors de la mise sous tension du µC (*off-on*).
- La famille LPC81x se décline en plusieurs boîtiers. Sur les premières versions des boîtiers à 16 et 20 broches, la broche d'entrée en mode ISP était PIO0_1 lieu de PIO0_12. Si votre µC refuse de passer en mode ISP, vérifiez le code de révision. Il doit être « 4C » ou plus récent pour que PIO0_12 fonctionne.
- Un µC LPC81x vierge démarre directement dans le mode ISP ; vous devriez donc être en mesure de programmer le µC au moins une fois.

proprement dit). Ils font tous deux partie de l'archive téléchargeable que vous pouvez importer dans LPCXpresso sans avoir à la décompacter auparavant.



Le fichier `main.c` contient le moteur principal qui lit le codeur et place les LED dans une matrice virtuelle. En arrière-plan, un temporisateur *SysTick* à 1 kHz se charge de l'allumage des LED et de l'anti-rebond du codeur rotatif. Les bips du vibreur sont du ressort d'un autre temporisateur (*Multi-Rate Timer*). Pour économiser de la mémoire du programme et en raison de leur disponibilité, les communications sont confiées, autant que possible, aux pilotes intégrés dans le µC. Jetez un coup d'œil au code source (du C pur) et jouez avec. Copiez, en lui donnant un nouveau nom, le projet de potentiomètre numérique dans le même espace de travail et modifiez-le. Ce n'est pas trop difficile, c'est abondamment commenté. Lorsque vous créez un projet, il va créer un fichier HEX que vous pouvez programmer dans le µC via un port sériel et l'outil Flash Magic [7].

CoCo-ri-Co est proposé sous forme de **module prêt à l'emploi** dans l'e-choppe d'Elektor. Les circuits imprimés nus sont également disponibles. Vos questions et vos contributions sont attendues sur la page Elektor.Labs du projet [3].

(140183-I – version française : Guy Raedersdorf)

Liens

[1] J²B : Elektor septembre 2011, www.elektor-magazine.fr/110274; www.elektor-labs.com/node/3832

[2] Platino : Elektor octobre 2011, www.elektor-magazine.fr/100892; www.elektor-labs.com/node/2288

[3] CoCo-ri-Co on Elektor.Labs : www.elektor-labs.com/node/4257

[4] Elektor 110553 BoB-FT232R : www.elektor.fr/110553

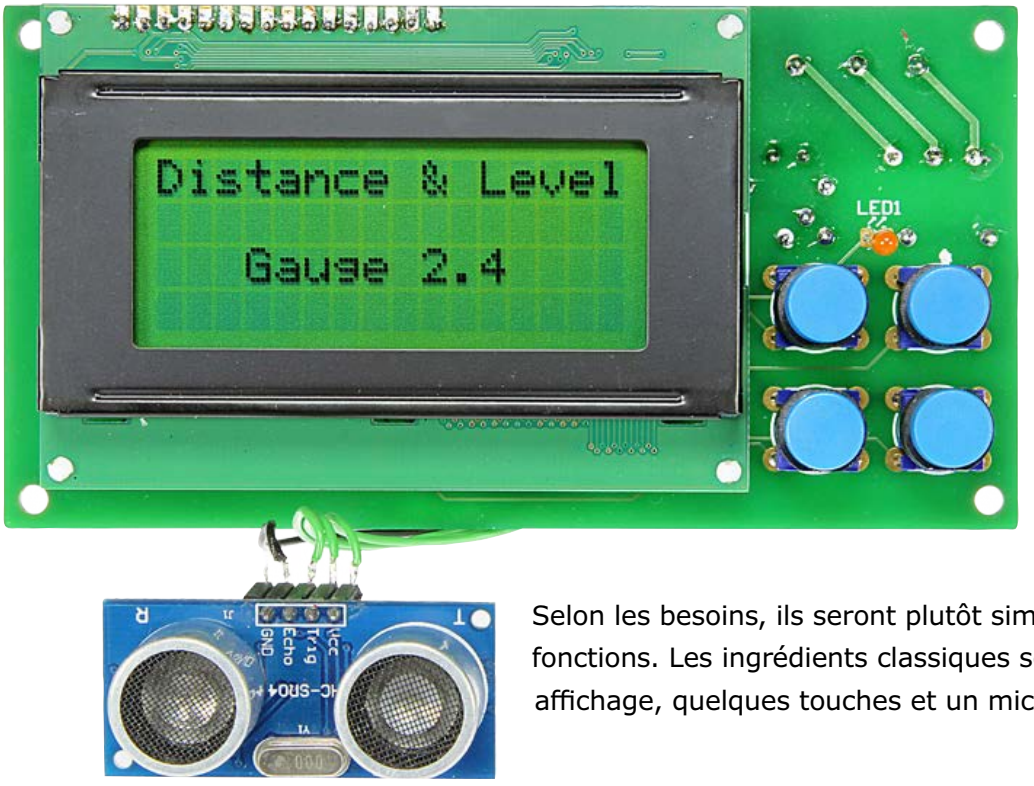
[5] LPCOpen : www.lpcware.com

[6] Téléchargements : www.elektor.fr/140183

[7] Outil de programmation : www.flashmagictool.com

mesure de niveau et de distance

avec fonction d'alarme



Jörg Trautmann (Allemagne)

Les capteurs à ultrasons compacts sont facilement disponibles et peu chers, et permettent de concevoir des appareils de mesure de distance variés. Selon les besoins, ils seront plutôt simples ou avec plein de fonctions. Les ingrédients classiques sont, outre le module US, un affichage, quelques touches et un microcontrôleur (µC).

Voici trois exemples de mise en œuvre de l'instrument de mesure de niveau ou de distance décrit ici : déterminer le niveau d'une cuve de fioul ou celui d'une citerne d'eau de pluie ; commander une électrovanne pour qu'elle arrête l'arrivée d'eau quand un niveau donné est atteint ; mesurer la distance entre deux objets. Il n'est pas difficile d'en imaginer encore bien d'autres. Le niveau d'un récipient cylindrique ou parallélépipédique peut être déterminé mécaniquement par flotteur et potentiomètre, par capacité électrique, par ultrasons ou par laser. Les variantes mécanique et capacitive impliquent des composants additionnels tels que flotteur et capteurs. Si le critère le plus important est la précision, un laser constitue indéniablement la solution optimale (abstraction faite de certains pièges, car des vapeurs peuvent perturber la réflexion du laser et fausser le résultat).
L'effet de tels facteurs sur les ultrasons est, en

Caractéristiques

- Mesure de niveau de fluides
- Fonction de surveillance du niveau avec sortie relais et visualisation par LED
- Niveau d'alarme Mini/Maxi programmable continûment
- Mémorisation de valeurs d'étalonnage Mini/Maxi de 10 récipients ou cuves
- Mesure de distance
- Navigation de menu intuitive sur afficheur LC

pratique, négligeable. D'où mon choix de cette option économique à base de capteur à ultrasons. Autre avantage : aucun contact avec le fluide. Et, en plus, le même instrument permet d'effectuer la mesure de distance.

La fréquence de 40 kHz à laquelle travaillent les transducteurs à ultrasons conventionnels correspond, à une température de l'air de 20 °C, à une longueur d'onde de 8,5 mm. Les ondes sonores situées dans cette gamme de fréquences peuvent facilement être focalisées ; l'angle d'ouverture de leur cône de propagation est de 15 ° environ. Des conditions idéales pour l'application envisagée, vu que les parois latérales du réservoir n'ont pratiquement pas d'influence sur la réflexion du signal.

Comme le transducteur ultrasonore utilisé ici a fait l'objet d'une description exhaustive dans le numéro d'été 2014 d'Elektor [1], nous nous limiterons ici à un résumé succinct de son fonctionnement : une impulsion de déclenchement commande au module US d'envoyer une salve (*burst*) d'ondes. Ce signal est réfléchi par l'objet dont on souhaite mesure la distance (la surface de l'eau dans un tonneau de récupération d'eau de pluie par exemple), et capté comme signal d'écho par la partie récepteur du module US. Dans la durée d'écho, la distance par rapport à l'objet source de la réflexion est parcourue deux fois. On calculera donc :

distance =
0,5 x vitesse du son [m/s] x durée d'écho [s].

Circuit et composants

Le circuit de la **figure 1** repose sur trois composants, le µC ATMEGA8 justement apprécié, un affichage LCD connecté à l'électronique et le module à ultrasons. Le module HC-SR04 (**fig. 2**) peut également être remplacé par un US-020,

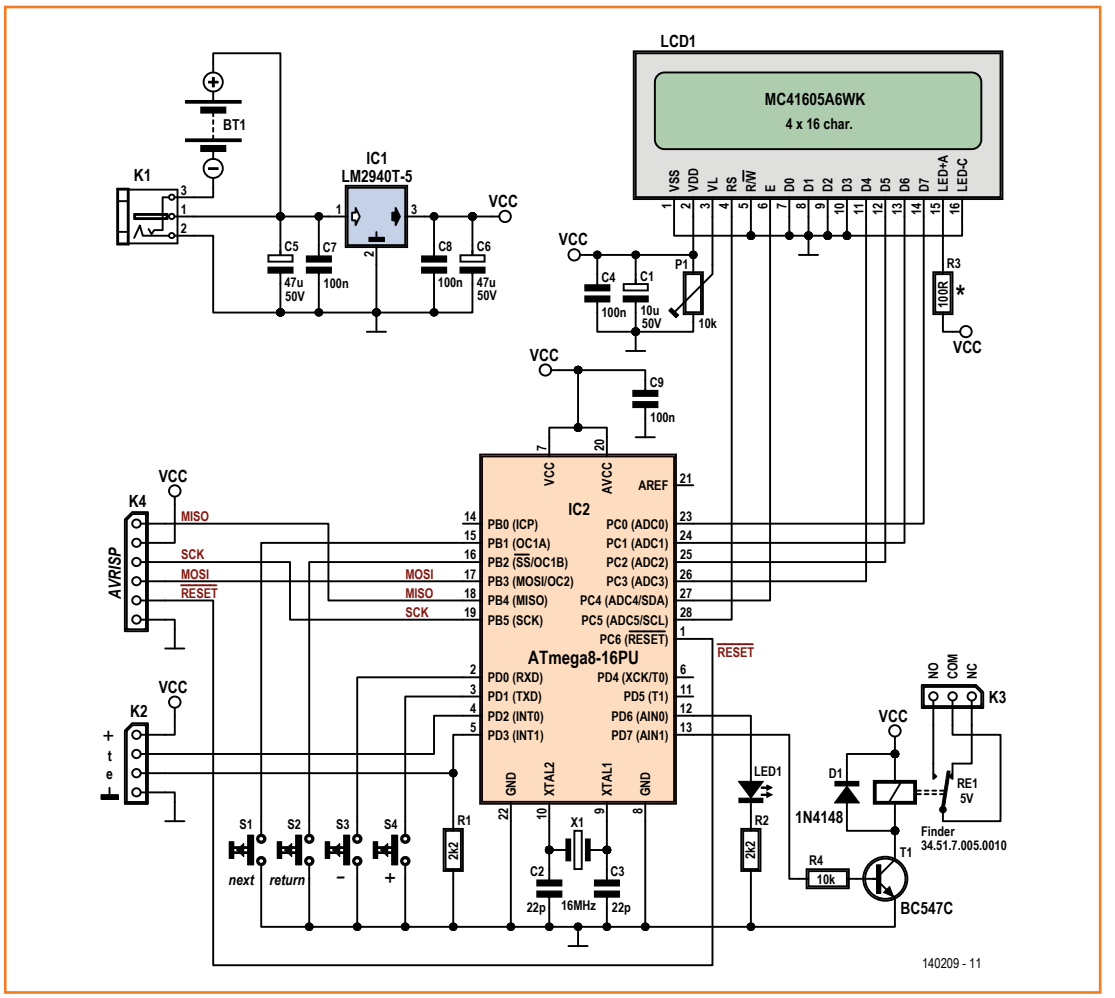


Figure 1.
L'électronique du circuit de mesure de niveau et de distance est simple.

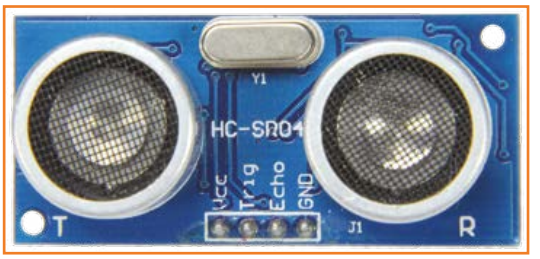


Figure 2.
Le minuscule module US HC-SR04.

de construction à peu près identique. Leur prix de quelque 4 € fait de ces deux modules des champions de l'économie pour un instrument de fabrication maison.

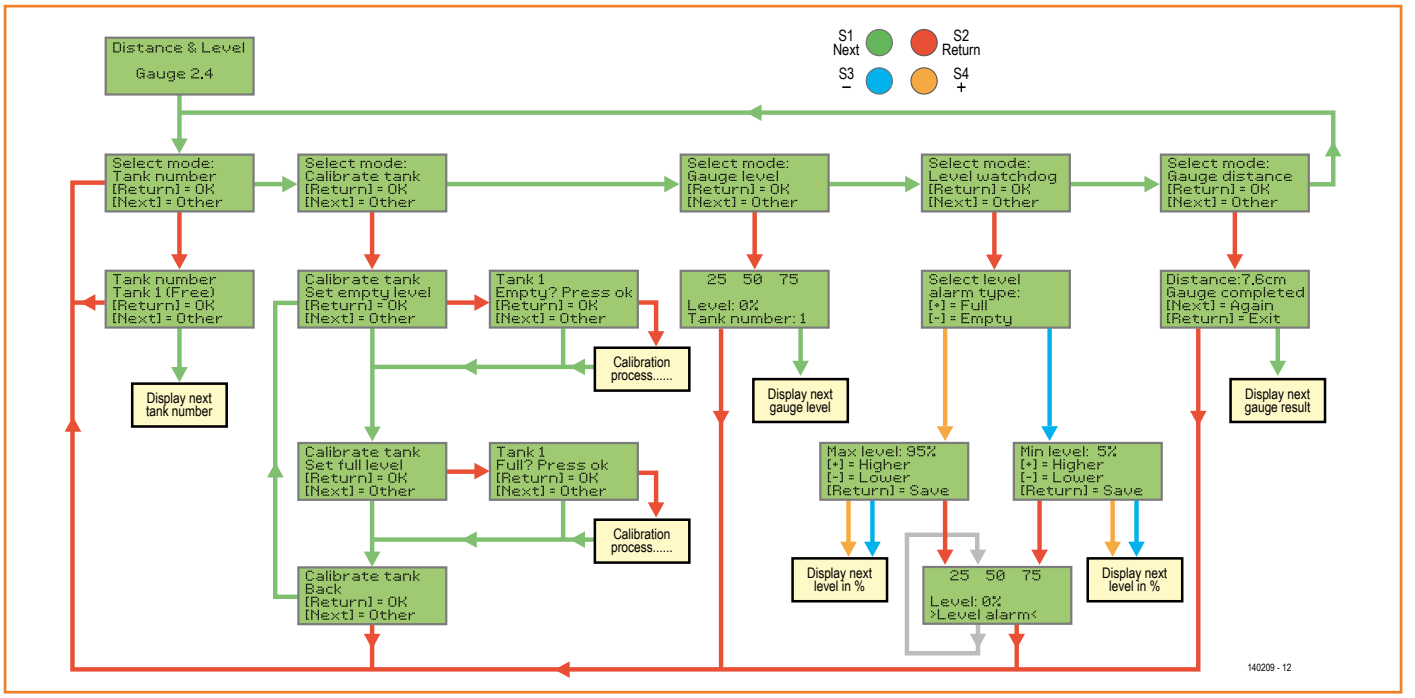
S'il est difficile de déterminer qui est au juste le fabricant des modules, il est facile d'en trouver chez les détaillants de composants électroniques, en particulier chez les spécialistes en robotique. La comparaison des fiches de caractéristiques permet de découvrir quelques différences. Ainsi, la distance de détection des deux modèles est, pour l'un de 2 cm à 3 m, pour l'autre de 3 cm à 7 m. Quelques recherches plus poussées permettent de découvrir, pour un seul et même modèle, d'autres différences encore. Selon l'origine de la fiche de caractéristiques, la consommation de courant se situera entre 3 et 15 mA. Pour être fixé, il ne reste qu'à la mesurer soi-même. Les fournisseurs des modules sont, en gros, d'accord sur les caractéristiques suivantes :

- Tension de service : 5 VCC
- Température de service : 0 à 70 °C
- Angle de propagation : 15°
- Fréquence de travail : 40 kHz
- Signal de déclenchement : 10 µs

Sur le module HC-SR04, la distance minimale entre les capsules de l'émetteur et du récepteur est si faible que l'on peut effectuer facilement des mesures à l'intérieur de bidons. En présence de liquides inflammables, il faut déporter le module US, en raison du risque d'explosion. Si l'on veut procéder à des mesures dans des récipients à diamètre d'ouverture très faible, on pourra opter pour le module US SRF02 ([4] ; non compatible broche à broche, doté d'une interface sérielle/ I²C !), il ne fait appel qu'à une seule capsule pour l'émission et la réception du signal US. Comme l'angle d'émission/de réception de ce modèle est très aigu, il faut donc accepter une distance minimale à mesurer de 16 cm.

Pour faciliter la navigation dans le menu, nous utilisons un écran LCD de quatre lignes de 16 caractères. P1 permet de régler le contraste. R3 est la résistance série du rétroéclairage (R3 = (5V-V_f)/I_{nom}). Consulter la fiche de caractéristiques de l'afficheur. Certains afficheurs LCD ont déjà une résistance série interne ; on pourra dans ce cas-là, remplacer R3 par un pont de câblage. Un interrupteur permettra de réduire la consomma-

Figure 3.
Ordinogramme du logiciel.



tion entre deux mesures.
Le relais commutateur RE1, dont les contacts sont accessibles sur K3, est piloté par la broche de port PD7 et le transistor de commande T1 ; la LED d'alarme (à faible courant) l'est par le biais de la broche de port PD6. Un régulateur à faible chute de tension LM2940CT-5 fournit une tension d'alimentation stable de 5 V à partir de quatre piles de 1,5 V. Ce régulateur de tension quelque peu exotique fournit jusqu'à 1 A et supporte donc bien mieux les charges que le classique 78L05 (100 mA seulement). Bien entendu, on pourra également utiliser l'instrument avec un adaptateur secteur (<26 V CC) branché sur K1. Relais activé et rétroéclairage en fonction, le circuit consomme environ 150 mA. Pour obtenir la meilleure précision possible, nous avons cadencé le µC par quartz externe de 16 MHz. Quelques composants passifs viennent compléter l'électronique.

Le logiciel

Le programme du µC doit remplir les fonctions suivantes :

- Mesure du niveau de fluides
- Fonction de surveillance du niveau avec sortie relais et visualisation par LED
- Niveau d'alarme Mini/Maxi programmable continûment
- Mémorisation de valeurs d'étalonnage Mini/Maxi d'un maximum de 10 récipients ou cuves
- Mesure de distance
- Navigation de menu intuitive par affichage LC
- Correction de décalage pour les mesures de distance
- Correction manuelle du facteur de division pour une adaptation à la température (+ faible/+ élevée)

La **figure 3** montre l'ordinogramme du logiciel. Compte tenu de l'espace mémoire limité de l'ATmega8, tout est en anglais. La commande se fait par les quatre touches *Return*, *Next*, *Plus* et *Minus*. En raison de la complexité de son code, le programme, disponible au téléchargement [3], avec fichier hex et dessins de la carte, a été subdivisé en procédures majeures et pourvu de commentaires.
Le processus de mesure proprement dit : Pour commander au module US connecté à K2 d'émettre un signal, on envoie, par le biais de la broche de port PD2, une impulsion longue de 10 µs, à flanc descendant, à l'entrée de déclenchement. Le module US envoie alors, après quelque 250 µs, une salve de 40 kHz de 200 µs. La sortie Echo, connectée au µC via la broche de port PD3, bascule alors au niveau haut, et le module US attend la réception du signal réfléchi. S'il arrive un tel signal, la sortie Echo rebascule au niveau bas. Pendant ce temps le compteur Timer1 compte. Une fois arrêté, on peut calculer la distance à partir du résultat du comptage. Si au bout de 50 ms aucun signal d'écho n'a été reçu, le processus de mesure pour le cycle en cours est interrompu. Pour une précision maximale, un cycle de mesure complet comporte 16 mesures. On pourra découvrir le code de programme dans la procédure `Gauge_distance()`.
Attention ! K4 donne accès à l'interface de programmation du µC, mais comme le programme est complexe, la mémoire Flash de 8 Ko est pleine à 100 %. Or, la version de démonstration de BASCOM-AVR ne peut pas être utilisée pour la compilation vu qu'elle ne permet de traiter que

Encadré 1. Influence de la température sur la vitesse du son

La vitesse du son c_0 dans l'air est, à une température T = 0 °C, de 331,5 m/s. Si la pression atmosphérique n'influe pas sur la vitesse du son, la température de l'air ϑ, elle, exprimée en degrés Celsius, la fait varier :

$$c_\vartheta = c_0 \cdot \sqrt{1+\alpha \cdot \vartheta}$$

Le coefficient de dilatation est ici de α 1/273,15 = 3,661x10^{-3} 1/°C.
Ce qui donne l'approximation :

$$c_\vartheta = 20{,}063 \cdot \sqrt{\vartheta + 273{,}15} \text{ [en m/s]}$$

Quelques exemples de la durée de trajet du son à 0 °C et à 20 °C. L'effet de la température n'est pas négligeable pour qui cherche la précision.

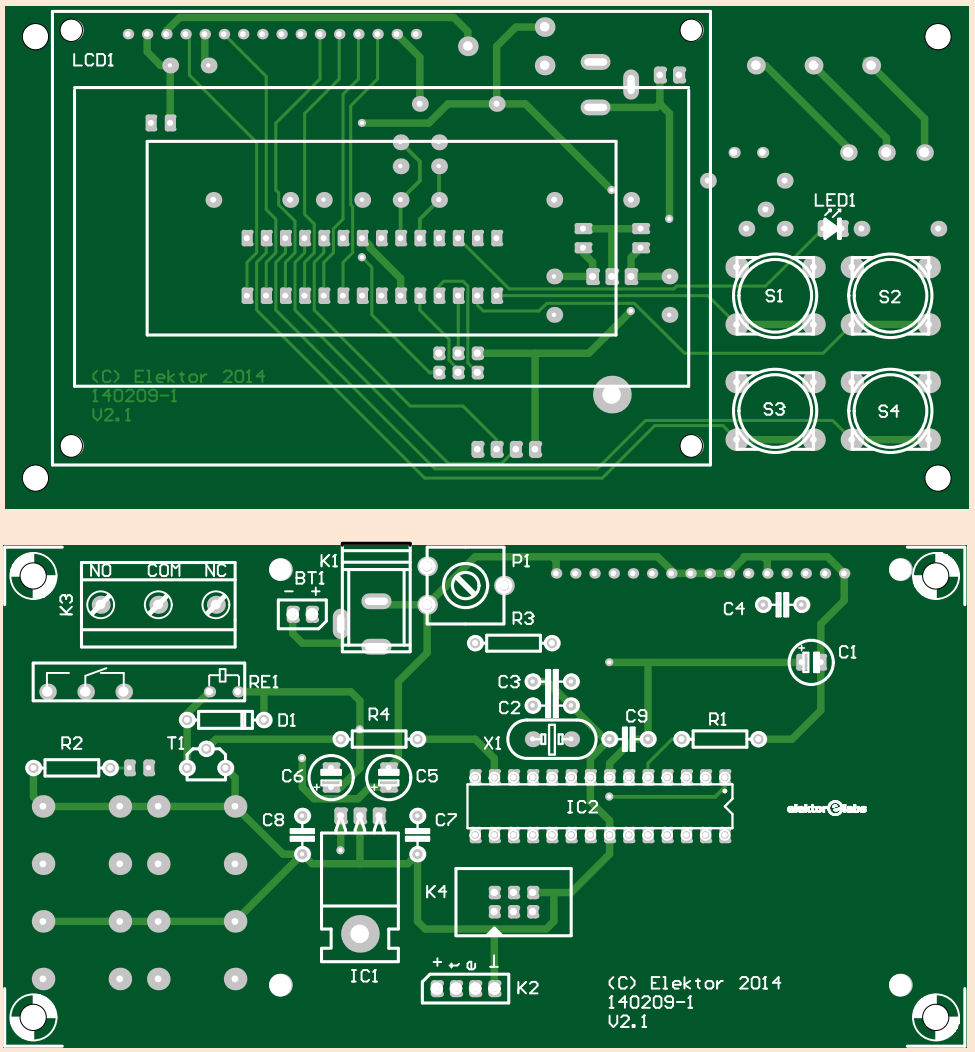
Distance en cm	Durée de trajet à 20°C [en ms]	Durée de trajet à 0°C [en ms]
2	0,117	0,121
10	0,583	0,603
50	2,915	3,017
100	5,831	6,033
200	11,662	12,066
300	17,492	18,100

4 Ko, soit seulement 50 % de la mémoire disponible. Lors de la programmation, un piège nous guette : comme l'ATmega8 travaille ici avec un quartz de 16 MHz externe, il faut positionner deux fusibles. `CKOPT` devrait être activé pour garantir le fonctionnement de l'oscillateur. Si ce choix accroît légèrement la consommation de courant, il est garant de la stabilité de l'oscillation. Il est impératif de positionner le fusible `SUT_CKSEL` sur `Ext. Crystal/Resonator High Freq.: Start-up time: 1K CK + 64 ms`! La liste des options de paramétrage est longue ; en cas d'erreur, il peut arriver que l'ATmega8 soit irrécupérable après le processus de programmation. Il faut donc vérifier et revérifier les options de paramétrage sélectionnées. On trouvera, dans le téléchargement, une capture d'écran d'AVR Studio qui donne le paramétrage correct des fusibles.

Réalisation et mise en service

La réalisation ne devrait pas, vu le faible nombre de composants, poser de problème. On pourra acheter la platine double face (**fig. 4**) chez Elektor ou la graver soi-même. Les fichiers correspondants sont inclus dans téléchargement [2]. Faire les découpes, dans le boîtier, pour l'affichage, les capsules ultrasons, les boutons et l'interrupteur nécessite une certaine dextérité. La disposition des capsules US est une affaire de goût. Si transducteur US doit être déporté loin du reste de l'électronique, il faudra l'y relier par un câble blindé.

Figure 4.
Le circuit imprimé, recto et verso.



Liste des composants

Résistances :
R1,R2 = 2k2
R3 = 0 Ω (pont de câblage)
R4 = 10 kΩ (à couche de carbone 250 mW, 250 V)
P1 = ajustable 10 kΩ

Condensateurs :
C1 = 10 µF, 50 V, RM 2 mm
C2,C3 = 22 pF, 50 V, RM 2,5 mm, C0G/NP0
C4,C7,C8,C9 = 100 nF, 50 V, 20 %
C5,C6 = 47 µF, 50 V, RM 2,5 mm

Semi-conducteurs :
D1 = 1N4148
LED1 = LED faible courant rouge, 3 mm
T1 = BC547C
IC1 = LM2940T-4.0
IC2 = ATmega8-16PU, programmé (140209-41)

Divers :
K1 = fiche femelle basse tension, 1,9 mm (1217037)
K2 = embase 1x4 contacts
K3 = bornier encartable à 3 contacts RM 5 mm
K4 = connecteur mâle 2x3 contacts avec cavalier
LCD1 = afficheur LCD 4x16
RE1 = relais 5 V SPCO 1x (Finder 34.51.7.005.0010) (1169338)
S1...S4 = poussoir Multimec RA3FTL6 (1132885)
4 capuchons pour Multimec 1D03 (1132887)
X1 = quartz 16 MHz, 18 pF
Module US HC-SR04 ou US-020 ou SRF02

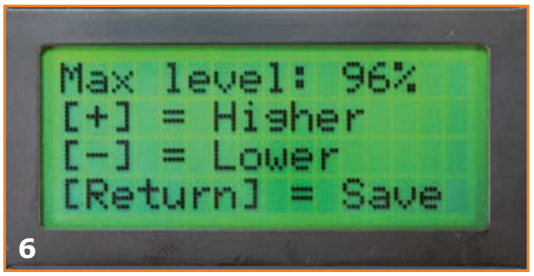
Références du catalogue Farnell entre parenthèses

Une fois l'ATmega8 (programmé !) mis en place et le module US connecté, mettre l'appareil sous tension. Le rétroéclairage de l'affichage LC doit s'allumer. Si ce n'est pas le cas, couper l'alimentation et vérifier l'implantation des composants. Lors du démarrage, on devrait voir s'afficher, pendant 2 secondes, le message *Distance & Level Gauge 2.4* (**fig. 5**) rassurant.

On peut alors passer au paramétrage de base. Pour ce faire, couper l'alimentation de l'appareil et le remettre sous tension en maintenant enfoncé le bouton *Return*. Après relâchement de ce bouton, le programme demande l'épaisseur du boîtier. On peut ici, par action sur les boutons *Plus* et *Minus* saisir, par pas de 5 mm, un décalage qui correspond à l'écart entre la surface du module US le fond du boîtier. Ce tarage permettra de mesurer une distance mur à mur. Cet étalonnage sera fait de préférence sur une distance de référence de 2 m vérifiée à l'aide d'un mètre-ruban. Confirmer la valeur par *Return*.

Le programme demande ensuite la valeur du diviseur à utiliser pour une correction de la valeur de mesure. On a besoin de ce facteur vu que la vitesse du son varie en fonction de la température. Sur une plage de température de –20 °C à +40 °C la vitesse du son croît de 312,85 m/s à 349,32 m/s. La dérive est pratiquement linéaire ; on peut donc envisager une imprécision de l'ordre de 2 % par 10 °C de variation de température. L'**encadré 1** propose quelques formules et un tableau des durées de trajet prévisibles. Note : La pression atmosphérique n'influence pas la vitesse du son.

Si un écart de 5 % maximum ne vous pose pas de problème, vous ne faites rien. Cependant, si vous souhaitez mesurer avec précision à des températures de l'air inférieures à 15 °C ou supé-

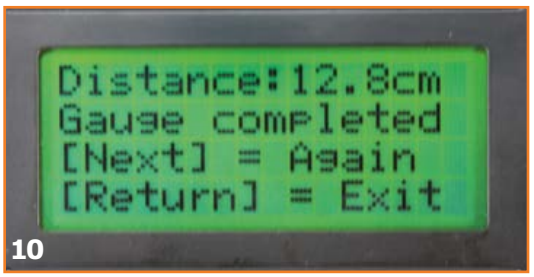


6

7

8

9



10

rieures à 25 °C, il faut saisir un facteur de division. Le diviseur croît avec la température. Les valeurs d'épaisseur du boîtier et du diviseur restent en mémoire même après un changement de pile ; elles peuvent cependant être redéfinies quand on le veut.

Vous devriez maintenant voir s'afficher l'option de menu *Select mode*. Actionnez le bouton *Return* pour passer au menu de sélection de réservoir. Il est possible d'étalonner jusqu'à dix réservoirs, cuves ou autres récipients, différents. Une seule condition à remplir : Le récipient doit être rectangulaire ou cylindrique ; avec une autre forme, une mesure linéaire devient impossible.

Après sélection du numéro de réservoir, on revient au menu principal. Une action sur le bouton *Next* fait s'afficher le point de menu *Calibrate tank*. Un *Return* et l'on se trouve dans le menu d'étalonnage. Après sélection de l'étalonnage à vide, le programme demande s'il peut procéder à l'étalonnage. Il n'est pas nécessaire, à cet effet, que l'on ait vidé le réservoir en question ; on pourra tout simplement mesurer, à l'extérieur du réservoir, la hauteur de son orifice de remplissage par rapport au sol.

Une fois l'étalonnage à vide effectué, on entre dans le menu de l'étalonnage à niveau plein (**fig. 6**). La valeur par défaut est fixée à un minimum de 3 cm ; on peut l'accepter en appuyant sur *Next*. Si l'on veut effectuer un étalonnage à niveau plein, il faudra positionner l'instrument au-dessus du récipient rempli et appuyer sur *Return*.

Une fois revenu dans le menu principal, on pourra effectuer la première mesure de niveau. Choisir à cet effet le point de menu *Gauge level* et le confirmer par un *Return*. La mesure est lancée et son résultat s'affiche. *Next* démarre une nouvelle mesure, *Return* fait revenir au menu principal.

Pour un test de la surveillance de niveau, ouvrir le menu *Level watchdog*. On peut, dans le sous-menu (**fig. 7**), sélectionner le type d'alarme : *Full* signifie alarme en cas de dépassement du niveau de consigne, *Empty* signifie que le niveau mesuré est sous le niveau de consigne. Une fois effectué le choix du type d'alarme, on pourra, dans le menu *Level-Alarm* (**fig. 8**), modifier le niveau de la valeur de seuil. Les valeurs par défaut sont prédéfinies à 95 % pour le niveau maximum et à 5 % pour le niveau minimum ; on pourra les modifier à l'aide des boutons *Plus* et *Minus*. Toutes les valeurs d'étalonnage étant sauvegardées dans la mémoire EEPROM, elles restent donc conservées même après un changement de pile.

La première mesure peut commencer, le niveau actuel s'affiche (**fig. 9**). Une nouvelle mesure est faite toutes les 5 s. Si le niveau de consigne n'est plus atteint ou s'il est dépassé, l'écran affiche > *Level alarm* <, la LED rouge s'allume et le relais s'enclenche, pour, par exemple, la mise en route d'une pompe. Si, dans les 5 s, le niveau est revenu dans la plage autorisée, la fonction d'alarme se désactive. Grâce à ce laps de mesure de 5 s, on se passe de fonction d'hystérésis tout en évitant le battement du relais.

Pour quitter ce menu et revenir au menu principal, il faut appuyer sur le bouton *Return* pendant 5 s. Il est possible alors de sélectionner le menu *Gauge distance*. Chaque action sur le bouton *Next* lance une mesure de distance (**fig. 10**). À nouveau, le bouton *Return* permet de revenir au menu principal.

(140209 – version française : Guy Raedersdorf)

Liens

[1] www.elektor-magazine.de/130546

[2] www.elektor-magazine.fr/140209

[3] HC-SR04: www.cytron.com.my;
SRF02: http://rn-wissen.de/wiki/index.php/Sensorarten#SRF02_Ultraschallsensor
(en allemand)

elektor PCB Service
powered by Eurocircuits

PCBs
Multilayer
Prototypes
Small series

Professional Quality
Trusted Service
Secure Ordering

Elektor PCB Service at a glance:

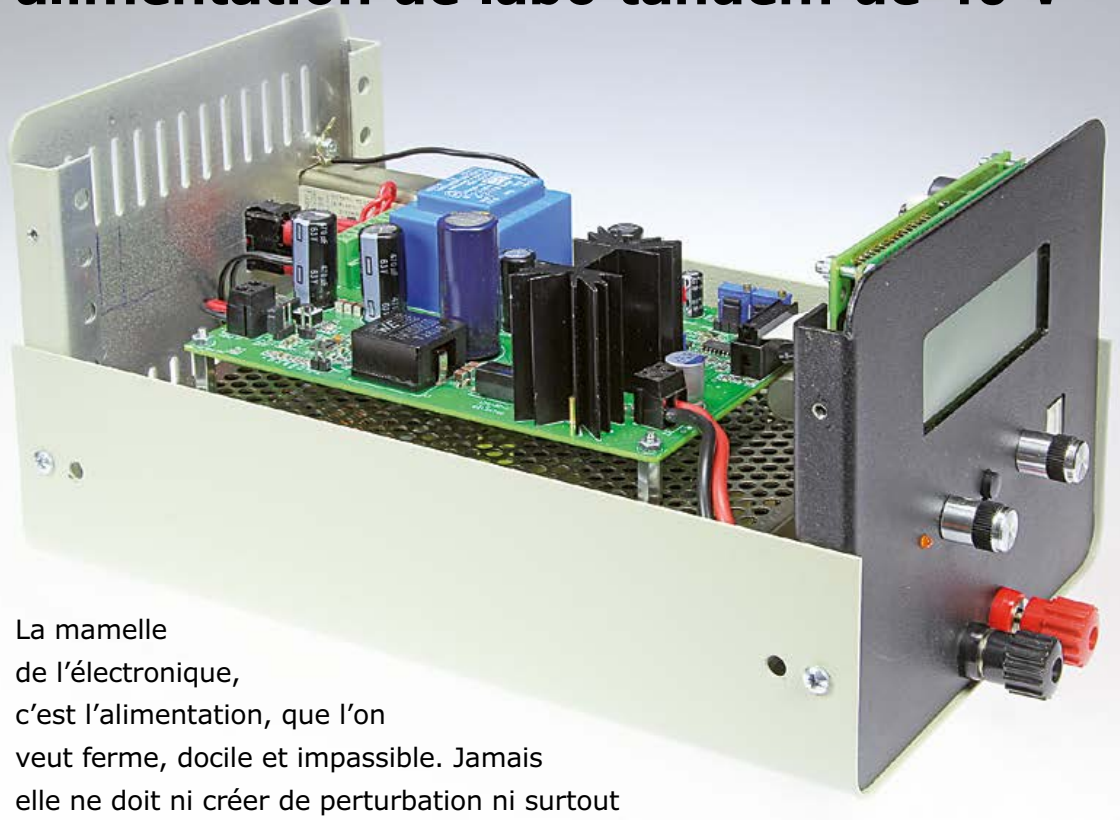
- 4 Targeted pooling services and 1 non-pooling service
- Free online PCB data verification service
- Online price calculator available
- No minimum order value
- No film charges or start-up charges

Delivery from 2 working days

Further information and ordering at www.elektorpcbservice.com

VariLab 402

alimentation de labo tandem de 40 V



Ton Giesberts
(Laboratoire Elektor)

La mamelle de l'électronique, c'est l'alimentation, que l'on veut ferme, docile et impassible. Jamais elle ne doit ni créer de perturbation ni surtout regimber lors d'une variation de consommation, aussi brutale soit-elle. Comme toutes bonnes mamelles qui vont par paire, celle-ci répartit la régulation sur deux étages, un premier à découpage suivi d'un circuit analogique, pour fournir une gamme étendue de 0 à 40 V et 2 A.

Tout électronicien praticien doit s'entourer d'outils appropriés pour mesurer parfaitement, souder parfaitement et alimenter ses circuits non moins parfaitement et en toute circonstance. Tout manquement à cette règle risque de coûter cher. Question prix, on trouve des alimentations dites de laboratoire à peu de frais.

Tant qu'elles sont éteintes, elles satisfont toujours aux exigences de stabilité, de précision, de régularité et de durabilité indispensables. Une fois allumées, c'est une autre affaire. Réunir toutes ces qualités n'est ni simple ni bon marché. Les vrais électroniciens savent pourquoi, de toutes les branches de l'électronique, celle qui traite des alimentations est la plus alambiquée !

En cinquante ans d'existence, Elektor a étudié et publié quantité d'alimentations. Ces derniers temps, nous en avons encore proposé plusieurs de différents types. Ce modèle-ci a été conçu et mis au point intégralement dans notre labo pour tenter d'offrir ce qu'on fait de mieux. Toutes les versions standard d'un prix abordable se limitent à une tension de 30 V, ce qui est parfois un peu juste. Aussi avons-nous voulu monter à 40 V tout en autorisant une consommation d'au moins 2 A. C'est un microcontrôleur qui commandera la régulation, assurera l'affichage et permettra la communication avec un PC.

Il restait à trouver un concept qui s'y prête et des composants à la hauteur de la situation. Première

réflexion : avec 40 V et 2 A, une électronique linéaire devrait pouvoir dissiper en permanence plus de 80 W, parfois en pure perte, faire appel à d'énormes radiateurs, voire à une ventilation forcée, donc bruyante.
Pour diminuer la dissipation, il existe bien des ficelles comme de travailler par échelons, avec différentes sources à tension fixe et commutables, mais les pertes en chaleur sont encore trop grandes. Quitte à passer par une solution alambiquée, nous avons fini par nous résoudre à utiliser une alimentation à découpage, dite SMPS (*switched mode power supply*), dont le gros avantage est son haut rendement. Mais il faudra se débarrasser de l'ondulation de sa tension de sortie et des parasites de commutation. Pour s'en affranchir, rien de tel que de passer par un étage tampon linéaire. Au total, cela donne un joyeux tandem, une solution bien hollandaise.

Caractéristiques techniques

- Stabilisation de la tension de sortie par régulateur à 2 étages
- Tension d'entrée fournie par module d'alimentation standard de 48 V
- Tension réglable entre 0 et 40 V, courant entre 0 et 2 A
- Réglage de tension et de courant par potentiomètres sur carte ou par microcontrôleur
- Commande par microcontrôleur ATxmega128A4U-AU
- Maniement par 2 codeurs rotatifs et bouton-poussoir
- Affichage sur 4 lignes de U_{set}, I_{set}, U_{out}, I_{out}, puissance délivrée et facteur de crête
- Connecteur USB pour liaison au PC
- Marche/arrêt par bouton-poussoir
- Ondulation de sortie < 15 mV_{pp} à 40 V/2 A
- Variation de sortie sous charge pulsée 220 mV_{pp} (charge 100 mA/1 A, fréquence 50 Hz, rapport cyclique 50 %)
- Consommation de l'alimentation sans charge < 4 W
- Haut rendement : 92 % à 40 V/2 A, 61 % à 5 V/1,75 A (mesuré entre la sortie et l'entrée du circuit imprimé d'alimentation)

Diagramme fonctionnel

La **figure 1** ébauche le diagramme fonctionnel de VariLab 402. S'il vous semble en avoir déjà vu un semblable, rien d'étonnant, l'alim de labo *PRO* de septembre dernier présente une structure comparable et ce n'est pas par hasard, elle aussi a été développée, en partie du moins, dans nos murs !

Une section d'alimentation à l'entrée fournit, à partir du secteur, une tension de 48 V pour 3 A. Cette tension est ensuite appliquée à un convertisseur à découpage qui la ramène entre 3 et 43 V, selon le réglage introduit. Puis cette tension est appliquée à un régulateur linéaire qui réduit l'ondulation et les parasites éventuels. La chute de tension sur cet étage est maintenue à environ 3 V (cf. description du schéma), suffisamment pour assurer un filtrage efficace, mais assez basse pour réduire la dissipation au minimum dans cette section : 6 W pour 2 A. La tension ainsi conditionnée passe par la section de mesure pour rejoindre la sortie.

Le microcontrôleur s'affaire aux tâches de service et à la régulation des différentes sections. Il obtient des informations sur la tension et le courant de sortie et, selon les instructions de l'utilisateur, commande les sections de régulation à découpage et linéaire. Toutes les valeurs, réglées sur deux codeurs rotatifs et un bouton-poussoir, et celles mesurées sont affichées sur un afficheur de 4 lignes. Le microcontrôleur dispose aussi d'un connecteur USB pour la liaison au PC, ce qui permet d'échanger les données de l'un à l'autre.
Il est aisé et rapide de dessiner un diagramme fonctionnel, mais les choses se compliquent quand il faut passer à la réalisation pratique. Aussi, tirons d'abord les grandes lignes du projet, nous décrirons ensuite le schéma final. Cet article se limitera à l'alimentation proprement dite, la partie informatique et l'affichage viendront le mois prochain.
Pour éviter les transformateurs secteur gros et chers, nous avons choisi un module d'alimentation

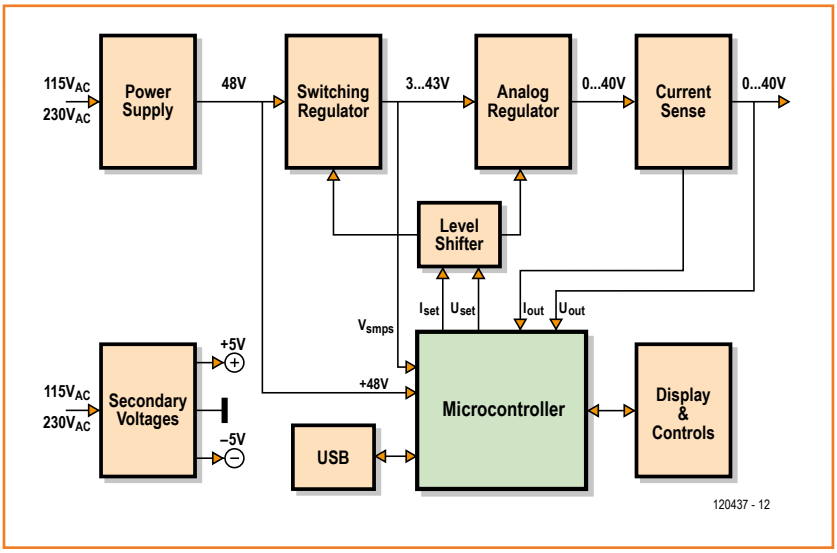


Figure 1.
Diagramme fonctionnel de VariLab 402. Le régulateur complet se compose d'une section à découpage et d'une section analogique l'une derrière l'autre (en tandem).

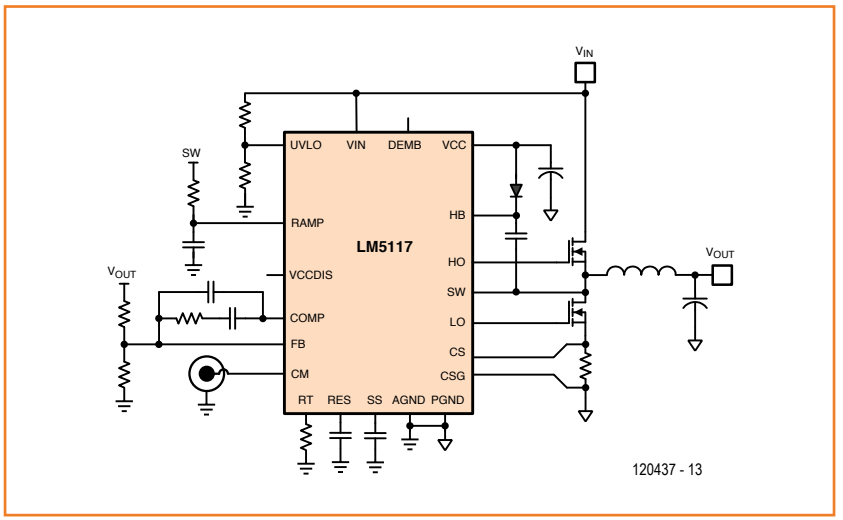


Figure 2.
Schéma de principe du circuit autour du convertisseur dévolteur LM5117.

standard de 150 W de *Mean Well*. Il peut fournir plus de 3 A sous une tension de 48 V.
La suite logique, c'est le passage par un régulateur dévolteur à découpage. Or il n'y a pas foule de puces qui acceptent de travailler sous 48 V. Nous avons opté pour le LM5117 [1] [2]. Le squelette du schéma d'un dévolteur avec cette puce est à la **figure 2**. On le reconnaîtra plus habillé dans le schéma complet. La puce commute, en externe, une paire de MOSFET de puissance qui attaque une cellule LC sur une haute fréquence de commutation, environ 100 kHz. Le circuit mesure le courant dans ces transistors sur une résistance dans la liaison de source de l'un d'eux. Les autres composants déterminent la tension de sortie et la chronométrie.

Pour de vrai

La figure 3 représente le vrai schéma de la partie alimentation. L'histoire du contrôleur et de l'affichage (carte µC/afficheur) viendra au prochain épisode.

Section commutation

C'est par la gauche du schéma qu'entre la tension continue de 48 V de l'alimentation secteur. Le convertisseur dévolteur centré sur IC1 la découpe en tranches et l'envoie par les FET de puissance T1 et T2 sur le circuit LC composé de L1/C13 à C16. L'épaisseur des tranches dépendra du signal de rétroaction appliqué à la broche 8 de la puce.
Le schéma autour de IC1 correspond largement à l'application standard de TI [2].
Nous avons choisi une fréquence de commutation plus basse, 100 kHz, qui cause moins de bruit HF sur un circuit à double face et moins de pertes de commutation. Mais la raison principale, c'est qu'à 100 kHz, on peut encore atteindre un rapport cyclique de 96 %. La valeur de R4 définit la fréquence de commutation.
Le condensateur C4 détermine la durée du démarrage en douceur de la puce. Avec 470 nF, la période est assez courte : 38 ms.
Le diviseur de tension R2/R1 indique à quelle tension d'entrée la puce commencera à travailler. Il est prévu pour une valeur d'environ 44 V.
Il y a un filtre formé de R7, R8 et C8 pour la mesure de la tension sur R9, laquelle indique l'intensité du courant dans les FET. Selon les données du constructeur, ce n'est pas nécessaire, mais nous l'avons quand même installé.
Pour régler la largeur d'impulsion, on se base sur la tension de la dent de scie prélevée par R6 et C7, lequel a une valeur de 1 nF, soit la moitié de la valeur maximale. D'après les données du fabricant, R6 devrait être de 820 kΩ, mais une valeur de 1,2 MΩ a donné un réglage plus stable pour différentes tensions de sortie.

Pour les condensateurs de découplage C9 sur V_{CC} et C10 sur HB, on a conservé les valeurs indiquées, il n'y avait pas de raison d'en changer. La diode de roue libre D1, en revanche, est un modèle à très basse tension directe : 0,57 V @ 1 A.

Les MOSFET de NXP renseignés dans la note d'application pour T1 et T2 sont pratiquement les meilleurs que l'on puisse trouver pour cette fonction. Les résistances de grille R11 et R10 préviennent les oscillations parasites, mais raccourcissent aussi le temps mort. En effet, comme les courants fournis par les sorties HO et LO sont plus forts pour le débit que pour le drain de courant par les transistors, T1 et T2 sont plus vite bloqués que mis en conduction.
Le circuit d'amortissement R13/C12 atténue les pics de tension qui peuvent survenir à la sortie des FET à cause de l'inductance L1.
La bobine de sortie L1 est calculée avec la formule de la feuille de caractéristiques. On table sur une tension fixe de 48 V et l'on se permet un courant d'ondulation de 40 % du maximum en sortie, donc 0,8 A. Le résultat donne 83,33 mH, très proche de la valeur existante en série E12.
La bobine de *Würth Elektronik* choisie est un peu surdimensionnée, mais l'avantage est ainsi d'y

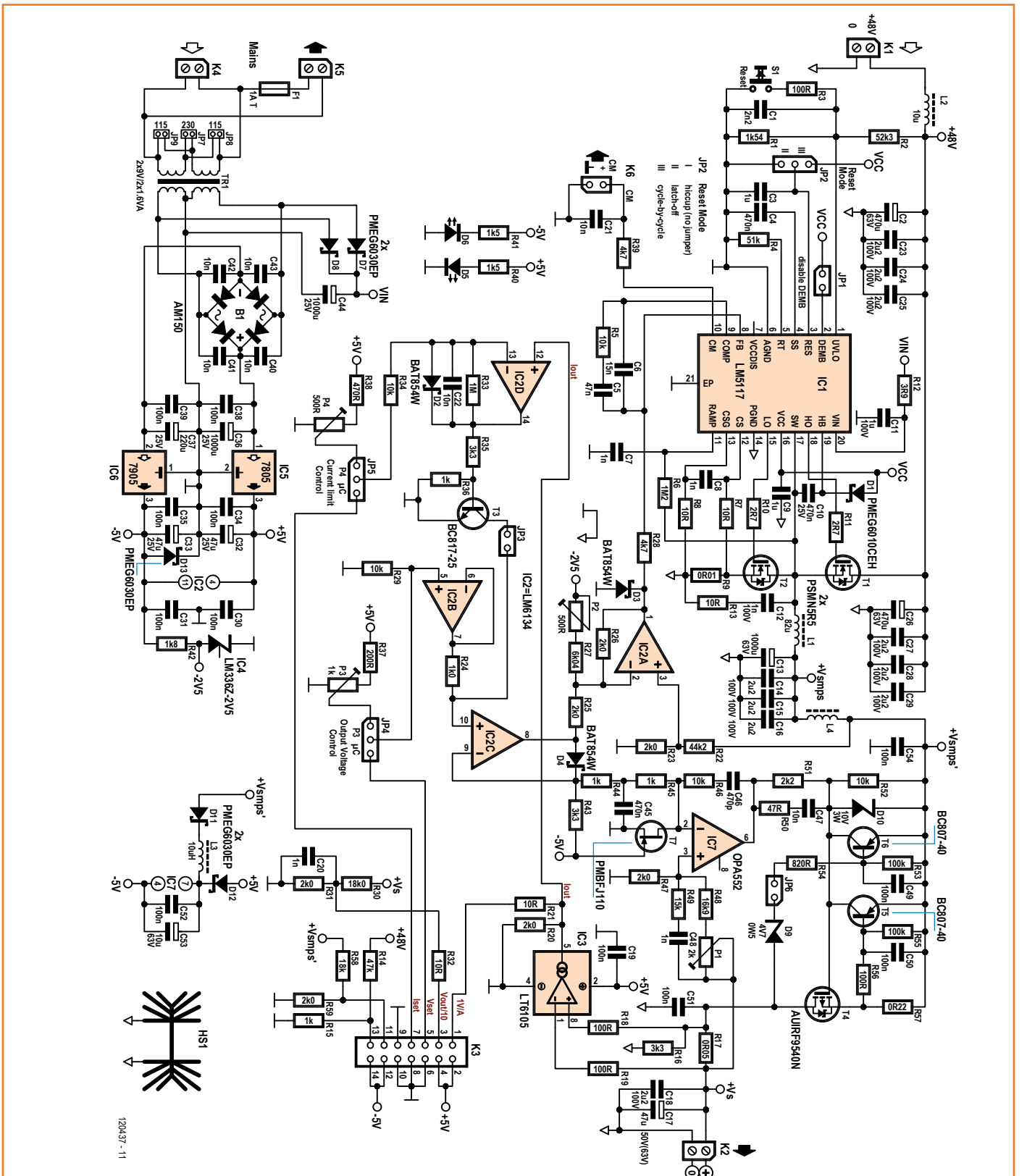
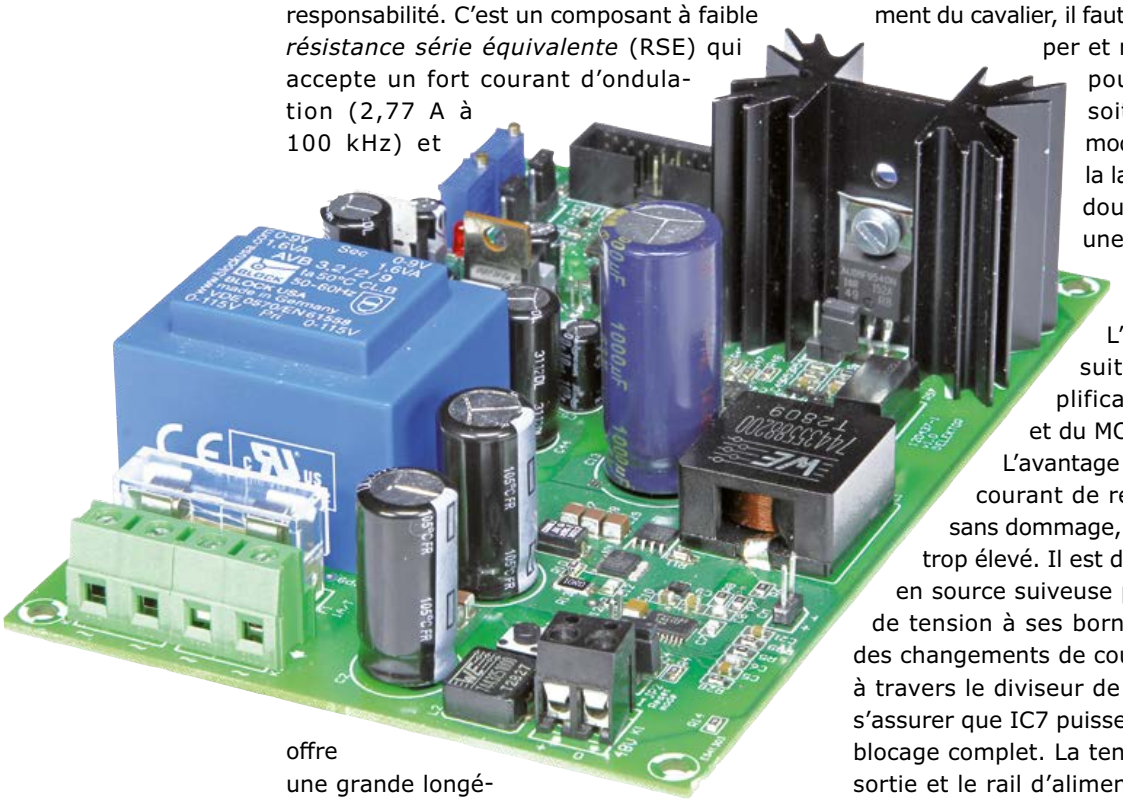


Figure 3. Le schéma complet avec, en haut, à gauche la section à découpage centrée sur IC1 et à droite la section analogique transcendée par T4.

réduire les pertes en puissance. Elle peut supporter le double du courant de sortie sans approcher de la saturation. Dans l'absolu, la partie SMPS peut fournir beaucoup plus que ce dont la partie linéaire pourrait avoir besoin. C'est volontaire : le SMPS n'aura ainsi qu'une influence moindre sur la régulation totale, surtout quand l'alimentation se trouve en mode de régulation du courant. Nous avons instauré comme limite la valeur de 12 A, trop haute semble-t-il, mais les tests ont montré des prestations moins bonnes avec des limites plus basses.

Le condensateur réservoir C13 porte une lourde responsabilité. C'est un composant à faible *résistance série équivalente* (RSE) qui accepte un fort courant d'ondulation (2,77 A à 100 kHz) et offre une grande longévité : 3 000 h à 105 °C. Une grande tolérance au courant d'ondulation permet aussi d'alimenter des appareils dont la consommation comporte de grosses impulsions. En parallèle sur C13, on trouve C14, C15 et C16 pour abaisser encore la RSE et malgré tout, on mesure encore une ondulation de 40 mV avec 20 V en sortie. La partie analogique qui suit l'éliminera.



Le régulateur de tension (de 7,6 V) interne du LM5117 est raccordé à une alimentation séparée fournie par le transformateur TR1, celui qui produit les tensions symétriques de ±5 V pour les amplificateurs opérationnels. On évite de la sorte de partir du 48 V disponible, ce qui occasionnerait à IC1 une grosse dissipation. Le filtre R12/C11 atténue les parasites HF de commutation. La sortie CM, broche 10 du LM5117, donne une information sur le courant de sortie moyen, mais elle n'est valable que pendant une période stable de fonctionnement. Ce signal est disponible sur K6 pour un test.

Le LM5117 dispose de trois modes de mise à zéro : *hiccup*, *latch-off* et *cycle by cycle*. *Hiccup* est normalement le meilleur mode, vous en aurez toutes les informations via [1]. On sélectionne le mode par le cavalier JP2. Après un déplacement du cavalier, il faut appuyer sur S1 ou couper et remettre l'alimentation pour que la modification soit prise en compte. En mode *hiccup*, C3 détermine la latence du démarrage en douceur après avoir défini une limitation de courant.

Section analogique

L'étage analogique qui suit se compose de l'amplificateur opérationnel IC7 et du MOSFET de puissance T4. L'avantage du MOSFET, c'est qu'un courant de retour peut le traverser sans dommage, pourvu qu'il ne soit pas trop élevé. Il est du type à canal P, monté en source suiveuse pour que les variations de tension à ses bornes restent petites lors des changements de courant. IC7 pilote le FET à travers le diviseur de tension R52/R51 pour s'assurer que IC7 puisse mener le transistor au blocage complet. La tension minimum entre la sortie et le rail d'alimentation de IC7, environ 2,7 V, n'est pas assez basse pour que le FET bloque totalement dans le cas le plus défavorable : U_{GS}, min. –2 V.

Le petit réseau R50/C47 est mis en parallèle sur R51 pour accélérer la boucle de régulation et C47 a été choisi plusieurs fois plus grand que la capacité d'entrée du MOSFET (1 300 pF).

L'amplificateur opérationnel IC7, un OPA552, est d'un type particulier ; alimenté entre ±4 V et ±30 V, il peut fournir jusqu'à 200 mA en sortie. Comme la tension finale doit pouvoir descendre jusqu'à 0 V et que, pour cette raison, la grille du

MOSFET doit être portée à une tension négative, l'ampli op est alimenté en –5 V.
Le réseau R45/R46/C46 veille à la stabilité de la régulation, surtout en mode de réglage du courant pour lequel la boucle est plus complexe. R49 et C48 dans la rétroaction entre la sortie de l'alimentation et l'entrée + de IC7 stabilisent aussi le comportement de sortie, spécialement lors de transitoires dans la charge. Attention : l'entrée + de IC7 est en fait l'entrée non inverseuse de l'étage de sortie linéaire, composé de IC7 et de T4, du fait que T4 inverse le signal. C45 filtre l'entrée de l'ampli op, mais fait aussi partie de la boucle de réglage en mode de courant. C51 atténue le bruit HF.

Pour alimenter IC7, il y a un petit circuit à part composé de D11, L3 et D12. Il évite que lors du réglage d'une tension de sortie proche de 0 V, la tension d'alimentation de IC7 soit trop basse pour bien piloter T4. Normalement, IC7 est mis sous tension par $V_{smps'}$ avant T4. Avec D12, la tension d'alim de 5 V prend le relais. Pour éviter que la tension de sortie de IC7, dans le cas où les tensions de ±5 V disparaîtraient, ne soit plus suffisante pour être pilotée par IC2C avec des tensions de sortie inférieures à 3 V, on a ajouté le FET T7 à l'entrée inverseuse de IC7. Si le –5 V s'effondre, le FET commence à conduire et dès lors, T4 bloque et la tension de sortie s'annule.
La résistance R17 est un *shunt* en série avec la sortie pour y mesurer le courant. Habituellement, cette mesure s'opère dans le conducteur de retour, mais cette méthode-ci permet à toutes les mises à la masse du circuit d'être directement reliées entre elles. IC3 mesure la tension sur R17 et fournit une tension directement proportionnelle au courant qui traverse R17. Finalement, C17 et C18 découplent et filtrent la tension de sortie.
Insérer R17 augmente quelque peu l'impédance de sortie. Pour en compenser l'effet, la rétroaction vers IC7, avec R47, R48 et P1, est directement prélevée du connecteur de sortie K2, pour éliminer l'influence de la moindre résistance, soudures comprises, entre le MOSFET et K2. Le potentiomètre P1 ajuste la tension de sortie ; éventuellement, un microcontrôleur pourrait aussi le faire.
La mesure du courant est confiée à IC3, une puce spécialisée dans cette tâche, c'est un LT6105 doté d'un très large domaine de tensions d'entrée : pas moins de 44 V, alors qu'il est alimenté sous 5 V seulement. Un coup d'œil à ses caractéristiques [3] vaut la peine. L'amplification est fixée par le rapport entre R20 et R18 ou R19, elle est donc de 20 fois ici. Il en résulte une tension de sortie sur R20 de 1 V/A. Des mesures précises nous ont indiqué que ce facteur pouvait légèrement varier avec la tension de sortie, mais en limitant la lecture à 3 chiffres, il n'y a pas lieu d'en tenir compte. Tout ceci est expliqué en détail sur notre site Elektor-labs [4].

Un court-circuit à la sortie est vite arrivé, comment protéger le MOSFET de puissance ? Il compte sur les réflexes d'un garde du corps personnel. Il y a déjà R57, une résistance de 0,22 Ω pour mesurer le courant. Quand il y passe 2,7 A, la chute de tension sur R57 est suffisante pour faire conduire T5, lequel diminue la tension entre grille et source. Difficile de réagir plus vite pour limiter le courant. Puis le contrôleur prend la main et veille à ce que la tension sur le MOSFET et la dissipation dans le radiateur restent dans des limites sures. Même topo, mais ici pour la tension sur T4. Normalement, elle ne doit pas dépasser 2,6 V. Si jamais elle les dépasse, c'est T6 qui, alerté par D9 et R54, entre en conduction et prend aussi le MOSFET en tenailles. C'est d'habitude au contrôleur que revient ce rôle, mais si vous utilisez VariLab 402 sans la carte µC/afficheur, vous pouvez activer cette sécurité avec le cavalier JP6.

Section régulateur

Il y a deux manières de régler tension et courant max. de sortie : par les potentiomètres multitours P3 et P4 du circuit imprimé ou par le microcontrôleur de la carte µC/afficheur. Le choix s'opère avec les cavaliers JP4 et JP5. Le réglage de la tension sur JP4 est tamponné par IC2B. La résistance R24 en série avec la sortie limite le courant si T3 est conducteur : il met le signal à la masse quand la limite de courant est dépassée. Si le cavalier JP3 est retiré, il n'y a plus de limitation matérielle, seul le microcontrôleur peut encore limiter le courant.

L'ampli op IC2D compare la tension sur JP5 à la mesure du courant de sortie effectuée par IC3. Pour être bien assuré de la stabilité de la régulation, tant le gain que la bande passante de l'amplificateur opérationnel sont limités par R33, R34 et C22. La diode Schottky limite l'excursion négative de IC2D. Si le courant de sortie excède la limite fixée, IC2D active T3 qui abaisse l'entrée + de IC2C.

Comme annoncé dans la description du diagramme fonctionnel, nous voulons maintenir sur le régulateur analogique et T4 une chute de tension constante. En fait, notre « changeur de niveau » est ici IC2C et D4. La tension de réglage pour la section analogique (R25) est un peu plus haute à cause de la chute sur D4. La valeur de R43 détermine le courant dans D4 et de là, la tension sur D4. Sur notre prototype, pour une tension de sortie de 0 V, il y a donc 5 V sur R43, nous avons mesuré sur D4 0,252 V et pour 40 V en sortie, 0,268 V ; il y a alors 9 V sur R43. C'est suffisamment constant pour notre objectif.

Puisque IC2A, l'ampli qui commande IC1, et IC7, celui qui commande T4, ont le même gain de 10, la différence de tension entre les deux sections reste dans la fourchette de 2,52 à 2,68 V.

L'ampli op IC2A fournit le signal de réglage pour l'entrée de rétroaction FB de IC1. Nous avons ajouté R27 et P2 pour compenser la tension de référence interne de IC1 (0,8 V). Ils sont branchés sur une tension de référence stable fournie par IC4. En principe, on pourrait par là ramener complètement à 0 V la sortie de la section commutation, mais avec la section analogique qui

Liste des composants

Résistances (1 %, 1/8 W, CMS 0805, sauf mention contraire) :

R1 = 1,54 kΩ
R2 = 52,3 kΩ
R3, R18, R19, R56 = 100 Ω
R4 = 51 kΩ
R5, R29, R34, R46, R52 = 10 kΩ
R6 = 1,2 MΩ
R7, R8, R21, R32 = 10 Ω
R9 = 0,01 Ω 5 %, 1 W (TE Connectivity/CGS, CGSSL1R01J)
R10, R11 = 2,7 Ω
R12 = 3,9 Ω 5 %
R13 = 10 Ω, 0,75 W (Vishay Draloric, CRCW201010R0FKEF, 2010)
R14 = 47 kΩ
R15, R24, R36, R44, R45 = 1 kΩ
R16 = 3,3 kΩ 1 W (Multicomp, RCL1218 3K3 1% 100 PPM/K E3, 1218)
R17 = 0Ω 05, 1 W (Bourns, CRM2010-FZ-R050ELF, 2010)
R20, R23, R25, R26, R31, R47, R59 = 2,00 kΩ
R22 = 44,2 kΩ
R27 = 6,04 kΩ
R28, R39 = 4,7 kΩ
R30, R58 = 18,0 kΩ
R33 = 1 MΩ
R35, R43 = 3,3 kΩ 5 %
R37 = 200 Ω
R38 = 470 Ω
R40, R41 = 1,5 kΩ 5 %
R42 = 1,8 kΩ 5 %
R48 = 16,9 kΩ
R49 = 15 kΩ 5%
R50 = 47 Ω 5 %
R51 = 2,2 kΩ 5 %
R53, R55 = 100 kΩ 5 %
R54 = 820 Ω 5 %
R57 = 0,22 Ω, 5 %, 5 W (TE Connectivity/CGS, SMW5R22JT, SMW5R22JT)
P1 = 2 kΩ 10 %, 0,5 W, ajust. 25 tours, vis en haut (Bourns, 3296Y-1-202LF)
P2, P4 = 500 Ω 10 %, 0,5 W, ajust. 25 tours, vis en haut (Bourns, 3296Y-1-501LF)
P3 = 1 kΩ 10 %, 0,5 W, ajust. 25 tours, vis en haut (Bourns, 3296Y-1-102LF)

Condensateurs (10 %, X7R, CMS 0805, sauf mention contraire) :

C1 = 2,2 nF/50 V, 5 %, C0G/NP0
C2, C26 = 470 µF/63 V, 20 %, RSE 0,027 Ω, I_{CA} 2 A (Panasonic, EEUFR1J471)
C3, C9 = 1 µ F/16 V,
C4, C10, C45 = 470 nF/25 V
C5 = 47 nF/50 V
C6 = 15 nF/50 V, C0G/NP0
C7, C8, C12, C20, C48 = 1 nF/100 V, 5 %, C0G/NP0
C11 = 1 µ F/100 V (Murata, GRM32CR72A105KA35L, 1210)
C13 = 1 000 µ F/63 V, 20 %, Ø 16 mm, au pas de 7,5 mm, I_{CA} 2,77 A (Panasonic, EEUFC1J102)
C14 à C16, C18, C23 à C25, C27 à C29 = 2,2 µF/100 V (TDK, C3225X7R2A225K, 1210)
C17 = 47 µ F/50 V, 20 %, Ø 10 mm, au pas de 5 mm, RSE 29 mΩ (Nichicon, PLX1H470MDL1TD)
C19, C30, C31, C49, C50 = 100 nF/25 V
C21, C22, C47 = 10 nF/50 V
C32, C33 = 47 µF/35 V, 20 %, au pas de 2,5 mm
C34, C35, C38, C39 = 100 nF/63 V, 5 %, au pas de 5/7,5 mm
C36, C44 = 1 000 µF/25 V, 20 %, Ø 10 mm, au pas de 5 mm, RSE 0,02 Ω (Panasonic, EEUFR1E102)
C37 = 220µ/F25 V, 20 %, Ø 8 mm, au pas de 3,5 mm, I_{AC} 950 mA (Panasonic, ECA1CM221)
C40 à C43 = 10 nF/100 V, au pas de 5 mm
C46 = 470 pF/100 V
C51, C52, C54 = 100 nF/100 V
C53 = 10 µ F/63 V, 20 % Ø 6,3 mm, au pas de 2,5 mm (Rubycon, 100YXF10MEFC6.3X11)

Inductances :

L1 = 82 µH, 7 A, CMS, haut courant, R_{CC} 30,4 mΩ (Würth Elektronik, 74435588200)
L2 = 10 µH, 7,2 A, CMS, haut courant, R_{CC} 16,3 mΩ (Würth Elektronik, 7443251000)
L3 = 10 µH, 950 mA, radial, au pas de 2 mm, RCC 0,14 Ω (Murata Power Solutions, 11R103C)
L4 = 70 Ω @ 100 MHz, 3,5 A, 0,022 Ω, 0603 (Murata, BLM18KG700TN1D)

Semi-conducteurs :

D1 = PMEG6010CEH (SOD-123F)
D2, D3, D4 = BAT85W (SOT-123)
D5 = LED rouge 3 mm, à fils
D6 = LED verte 3 mm, à fils
D7, D8, D11, D12, D13 = PMEG6030EP (SOD-128)
D9 = zener 4V7/500 mW (SOD-123F)
D10 = zener 10 V/3 W (SMB)
B1 = redresseur 50 V/1,5 A (Multicomp, AM150)
T1, T2 = PSMN5R5-60YS (LFPAK/SOT669)
T3 = BC817-25 (SOT-23)
T4 = AUIRF9540N (TOP -220)

suit, ce n'est pas nécessaire. S'il le faut, on peut avec P2 légèrement décaler la tension de sortie du SMPS. La diode Schottky D3 à la sortie de IC2A l'empêche de devenir négative, parce que IC1 n'est pas protégé contre cette éventualité. Vous trouverez davantage de détails sur le calcul de cette section sur [4].

Liaison au microcontrôleur

Le connecteur K3 sur le circuit imprimé de l'alimentation rassemble tous les signaux et tensions nécessaires à la liaison avec la carte µC/afficheur qui sera publiée le mois prochain. Sur les broches 1 et 3 on trouve le courant (1 V/A) et la tension (V_{out}/10 par le diviseur R30/R31), tandis que sur la 5 et la 7, le microcontrôleur peut envoyer les signaux de commande de tension et de courant. La tension de sortie de la section commutation ($V_{smps'}$/10 par R58/R59) est disponible sur la broche 11 et la tension d'entrée de 48 V sur la 13 : +48 V/48 par R14/R15.

Alimentation

Nous avons donc choisi pour fournir le 48 V un module tout fait de *Mean Well* qui donne 3 A, mais vous pouvez évidemment choisir une autre

T5, T6 = BC807-40 (SOT-23)
T7 = PMBFJ110 (SOT-23)
IC1 = LM5117 (MXA20A)
IC2 = LM6134 (SO-14)
IC3 = LT6105 (MSOP-8)
IC4 = LM336Z-2.5 (TOP -92)
IC5 = 7805 (TOP -220)
IC6 = 7905 (TOP -220)
IC7 = OPA552UA (SO-8)

Divers :

K1, K2 = domino à 2 vis au pas de 5 mm
K3 = embase à 14 picots avec col, au pas de 2,54 mm
K4, K5 = domino à 2 vis au pas de 7,5 mm
K6, JP1, JP3, JP6 = embase à 2 picots au pas de 2,54 mm
JP2, JP4, JP5 = embase à 3 picots au pas de 2,54 mm
JP1 à JP6 = cavalier 2,54 mm
S1 = bouton-poussoir 6x6 mm SPST-NO (TE Connectivity/Alcoswitch, FSM4JH)
F1 = porte-fusible encartable avec capot et fusible 1 AT
TR1 = transfo secteur encartable, prim. 2x115 V, sec. 2x9 V, 3,2 VA (Block, AVB3.2/2/9)
HS1 = radiateur encartable ; p.ex. Fischer SK 129 38,1 STS, 6,5 K/W
module d'alimentation 48 V/3 A, p.ex. Mean Well S -150-48
circuit imprimé réf. 120437-1

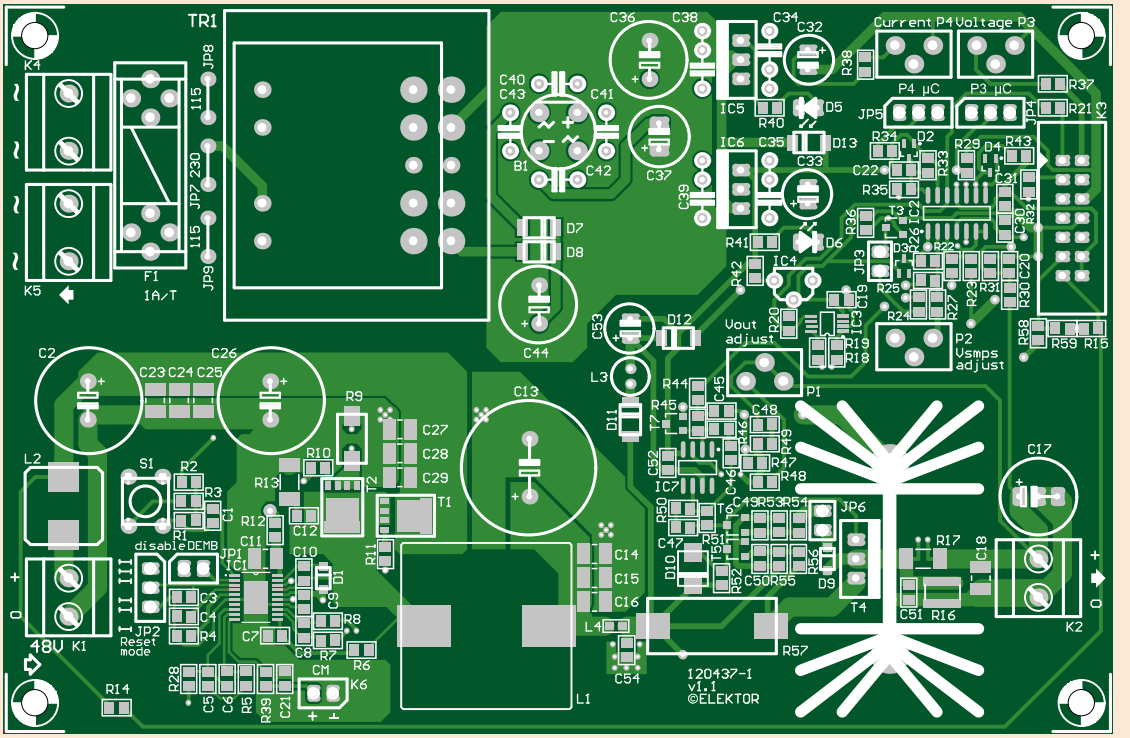


Figure 4.
Le circuit imprimé pour VariLab 402 porte un mélange de CMS et de composants traversants ; sûrement pas la tasse de thé du débutant !

solution pour disposer des mêmes valeurs.
L'alimentation des amplificateurs opérationnels, pour s'affranchir des parasites de commutation et du bruit, provient d'un petit circuit indépendant avec le transformateur encartable TR1, adaptable au 115 et au 230 V par des ponts de câblage sur JP7, 8 et 9, le pont redresseur B1, deux stabilisateurs de tension IC5 et IC6 et leur entourage habituel. Il fournit aussi la tension de service VIN à IC1 par D7 et D8.

La diode Schottky D13 à la sortie de IC6 protège la sortie du régulateur négatif contre toute tension positive, p.ex. lors de l'extinction. C'est ainsi que la tension ne montera jamais à plus de 0,2 V. Les tensions de 5 V arrivent aussi au connecteur K3 pour la carte µC/affichage et les LED D5 et D6 témoignent de leur présence effective. On raccorde au secteur le module d'alimentation de 48 V sur K5. Le fusible F1 fournit une sécurité supplémentaire parce que bien des choses peuvent arriver avec une alimentation externe.

Tant à l'entrée de la ligne d'alimentation à 48 V que près des MOSFET de commutation, les mesures de découplage sont prises. D'abord la bobine série L2 (10 µH, R_{CC} 16,3 mΩ) suivie par la mise en parallèle de C2/C23/C24/C25. Pour les MOSFET, il y a la combinaison C26 à C29. À noter que C2 et C26 présentent une RSE d'à peine 27 mΩ et soutiennent près de 2 A à 100 kHz. Les autres condensateurs sont des CMS dont l'efficacité est particulièrement grande en HF.

Construction

Ce fut une longue explication, mais au moins avons-nous analysé à la loupe (elle est utile) pratiquement tous les composants du schéma. La figure 4 révèle le tracé des pistes et l'implantation des composants sur le circuit imprimé à double face conçu pour VariLab 402. Pour construire soi-même ce projet, il faut déjà une solide expérience dans le soudage des CMS. Il importe de s'en tenir strictement aux composants indiqués dans la liste et surtout s'interdire d'installer des condensateurs électrolytiques de plus faible valeur. Le MOSFET T4 est doté d'un petit radiateur duquel le transistor doit être isolé, parce que le radiateur est mis à la masse par les broches de fixation.

Il faut des ponts de câblage pour sélectionner la tension secteur appropriée, sur JP7 pour le 230 V sur JP8 et JP9 pour le 115 V.

On commence par raccorder à l'entrée un module d'alimentation de 48 V pour vérifier si le circuit fonctionne. Mais avant cela, il faut installer des cavaliers sur JP3, JP4 (liaison à P3), JP5 (liaison à P4), et JP6. Mettez provisoirement P1 et P2 en position médiane avant de brancher sur le secteur. Raccordez une (petite) charge à K2 avec un voltmètre et un ampèremètre et voyez si vous pouvez régler la tension de sortie avec P3. Il doit aussi être possible de voir s'il y a bien limitation de courant avec P4, alors la tension de sortie diminue.
Si tout marche bien, vous êtes alors prêt pour la seconde partie dans laquelle nous explorerons la carte µC/affichage et son logiciel, avant de voir comment loger l'alimentation de labo complète dans un boîtier.

(120437 – version française : Robert Grignard)

Liens

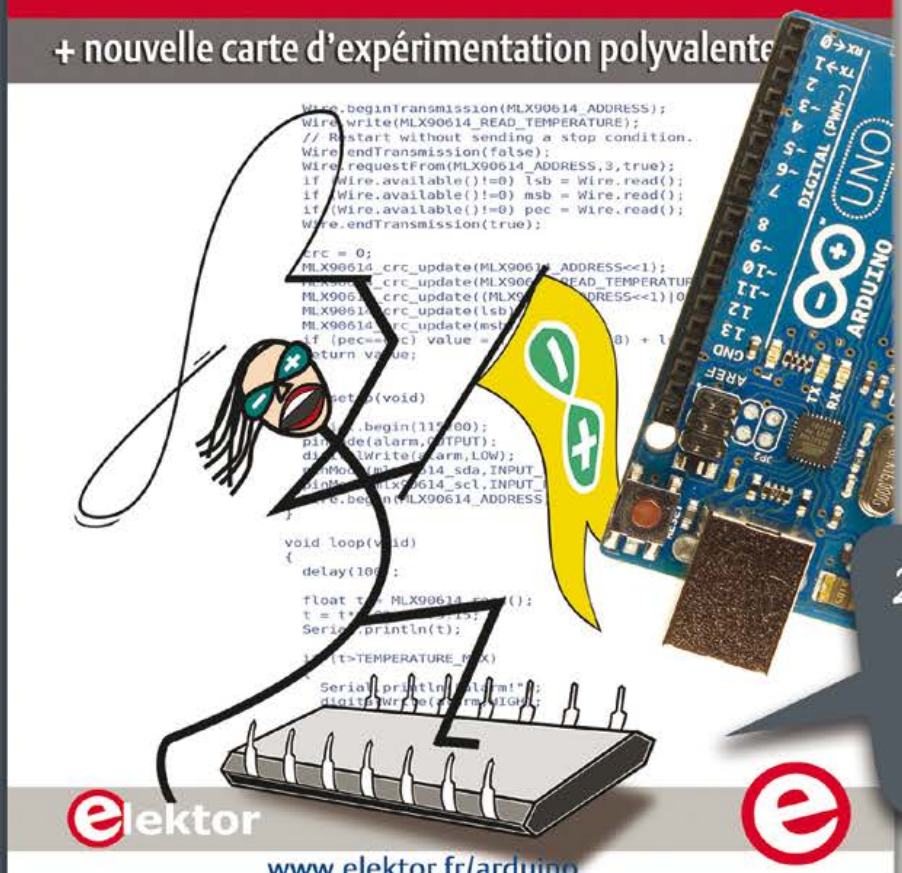
[1] www.ti.com/lit/ds/symlink/lm5117.pdf
[2] www.ti.com/lit/ug/snva466b/snva466b.pdf
[3] http://cds.linear.com/docs/en/datasheet/6105fa.pdf
[4] www.elektor-labs.com/120437

Clemens Valens

maîtrisez les microcontrôleurs à l'aide d'Arduino

2e édition revue et augmentée

+ nouvelle carte d'expérimentation polyvalente



elektor

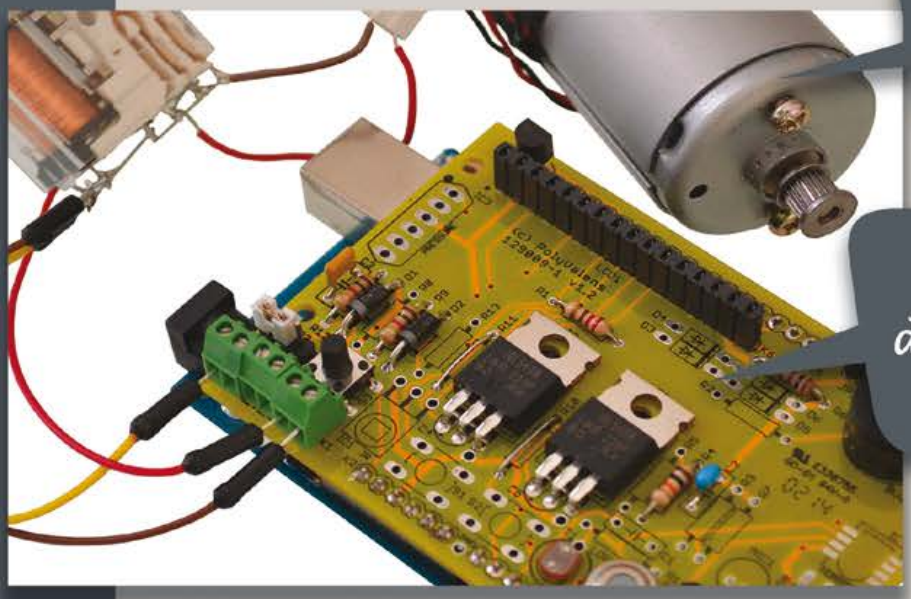
www.elektor.fr/arduino

www.elektor.fr/arduino

+35 pages

Témoignage :
« je prête volontiers mes bouquins, mais celui-là... je veux pas m'en passer »
Bernard P.

2e édition revue et augmentée par l'auteur
avril 2014



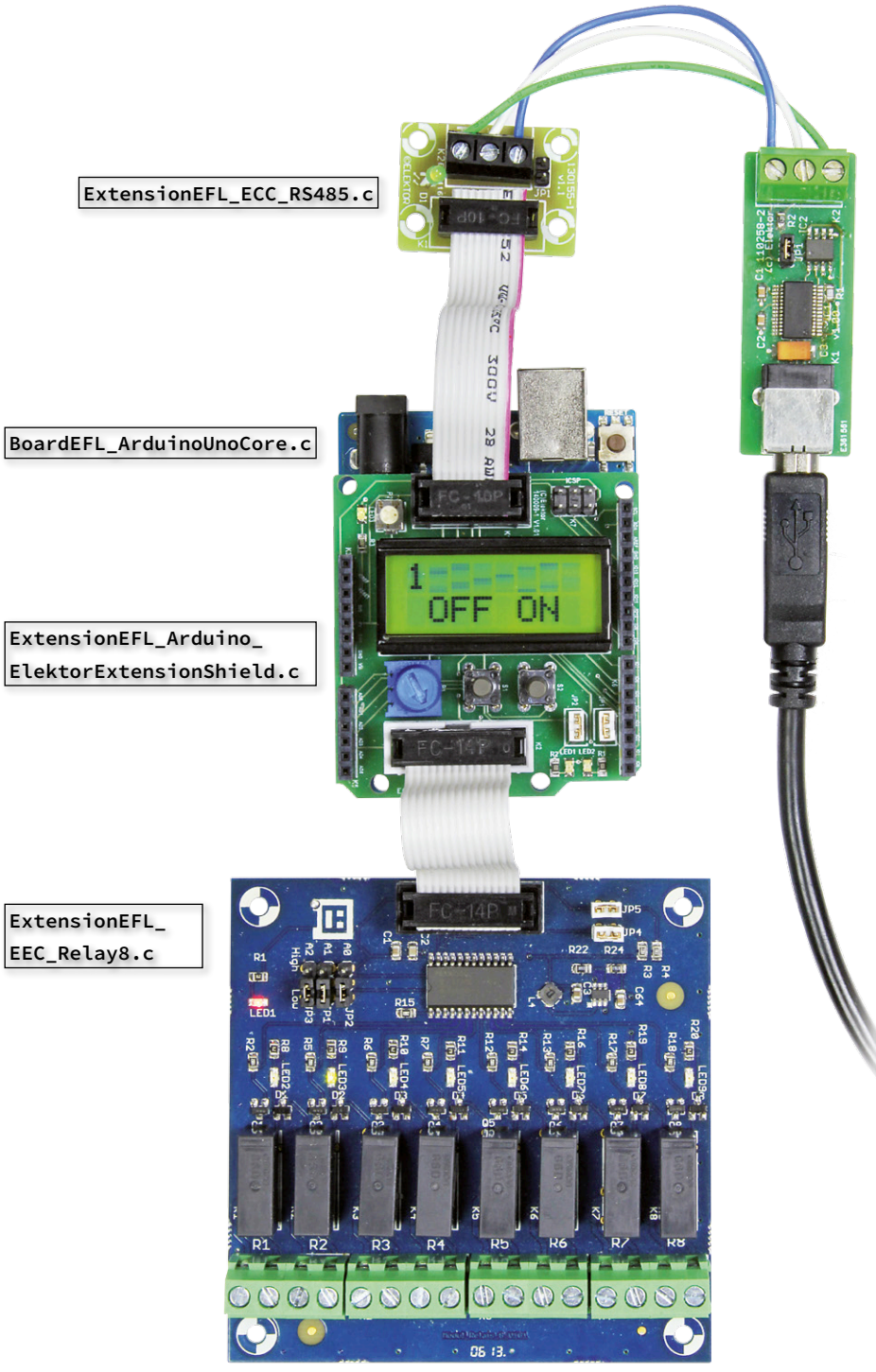
nouvelles applications

nouvelle carte d'expérimentation polyvalente

385 pages - 42,50 €
isbn 978-286661-195-8

modules de logiciel en C

Logicieux pour la carte *shield* d'Elektor, la carte relais et bien plus...



Jens Nickel (Elektor)

Le matériel modulaire accélère la réalisation d'un prototype adapté ; il suffit de raccorder des modules prêts à l'emploi. Le langage de programmation C est idéal pour transposer ce principe au monde des logiciels. Pour notre *shield* Elektor et trois cartes d'extension, nous vous proposons des modules logiciels et des applications de démo.

Dans le numéro double de l'été 2014, nous avons décrit une carte d'extension (*shield*) enfichable pour l'Arduino Uno, avec affichage, deux LED, deux boutons et un potentiomètre [1]. Vous pourrez vous lancer dans la programmation de microcontrôleurs (µC), paramétrer des sorties numériques (LED, on/off) et effectuer la lecture d'entrées logiques (boutons). Pour les plus expérimentés d'entre vous, la carte comporte une paire de connecteurs d'extension. L'ECC (*Embedded Communication Connector*) à 10 broches offre les signaux TX/RX de l'UART et deux broches GPIO. On peut ainsi, via un câble plat, connecter un module RS-485 grâce auquel on pourra, de façon fiable, envoyer et recevoir des octets sur de longues lignes. Nous développons actuellement d'une passerelle NFC qui permettra, par commandes ASCII simples, la lecture et l'écriture de cartes NFC et la communication avec un téléphone tactile compatible NFC. Plus de modules ECC au planning.

L'EEC à 14 broches, alias connecteur Gnublin, parce qu'il permet la connexion des modules Gnublin de Benedikt Sauter. C'est à lui aussi que l'on doit la spécification du connecteur femelle ; dans

le cas de la carte *shield* d'Elektor, seules quatre broches sont utilisées, le 3,3 V, la masse (GND) et les deux lignes I²C. La plupart des cartes Gnublin [2] n'utilisent cependant que les broches I²C, telles, par exemple, la carte d'extension de port, la carte du capteur de température, une carte à huit relais [3] et le module CA/N pour Arduino du numéro de septembre [4] d'Elektor.

Module logiciel

Pour son dernier *atelier du microcontrôleur* publié le nº 437 (nov. 2014), Burkhard Kainka a développé sous Bascom plusieurs applications intéressantes pour la carte *shield* d'Elektor ; nous ne manquerons pas, occasionnellement, de vous en proposer d'autres.

Grand absent, un support logiciel en C. Le langage C excelle dans les projets de logiciels modulaires. On peut, une fois des modules logiciels (personnels ou d'une tierce partie) mis au point et testés, les réutiliser indéfiniment dans ses propres projets. Le gain de temps est considérable ; dans le meilleur des cas, une grosse application parfaitement fonctionnelle est même réalisable en quelques minutes. En cas de modification du cahier des charges du projet, suite par exemple à l'adjonction tardive d'une interface RS-485 – devenue indispensable entretemps, mais pas prévue au départ – l'adaptation d'un tel projet logiciel pour le rendre opérationnel est l'affaire de quelques minutes.

Au niveau du matériel, nous disposons en effet d'un système déjà parfaitement au point, reposant sur des modules interconnectables. On implante, sur une carte Arduino Uno, une carte *shield* Elektor à laquelle on peut, par exemple, connecter un module RS-485, une carte de relais, voire une carte CA/N. Nous voulons maintenant transposer ce principe au logiciel ; il nous suffira, dans le cas d'une application aux exigences bien définies, d'accoler les modules de code correspondants pour les fusionner en un projet logiciel. Idéalement, il existe une association spécifique entre un module matériel et un module logiciel à utiliser telle quelle, quelle que soit la constellation dont fait partie le matériel. Le code de la carte d'extension à relais, par exemple, ne changera pas, que nous connections les relais à la carte *shield*, à la carte Xmega ou à la carte Linux d'Elektor.

Nouvelle structure de fichier

L'EFL convient maintenant aussi aux grands projets modulaires, de plusieurs cartes d'extension raccordées à une carte à microcontrôleur (même en chaîne). Il a fallu, élargir et restructurer quelque peu la bibliothèque d'origine [5].

- Le système hiérarchique des *Includes* a été restructuré. Maintenant, seuls sont encore intégrés, partout, des fichiers d'en-tête (*Header*) provenant du dossier « Common ». Ces fichiers sont identiques pour tous les projets EFL, quels que soient les cartes et le µC utilisés.
- Un nouveau fichier d'en-tête a été créé pour cela dans le dossier « Common » (ControllerDefinesEFL.h). Il comporte toutes les définitions de fonctions de l'API du microcontrôleur, qui restent en effet les mêmes. Le fichier d'en-tête spécifique au µC (ancien ControllerEFL.h) ne contient plus que des définitions spécifiques au µC, telles celles des registres.
- Nous remplaçons les noms ControllerEFL.h/.c ou encore BoardEFL.h/.c par des dénominations explicites pour les fichiers de code spécifiques au µC et aux cartes. Le fichier d'implémentation de l'API du microcontrôleur pour le ATmega328 s'appelle maintenant ControllerEFL_ATmega328, le fichier, dérivé du câblage de la carte de l'Arduino Uno, BoardEFL_ArduinoUnoCore.
- Nous pouvons alors intégrer plusieurs cartes d'extension dans notre projet sachant que les fichiers de code associés, ne s'appellent plus « ExtensionEFL.c/.h », mais ont chacun un nom différent. Exemple :
  - ExtensionEFL_Arduino_ElektorExtensionShield.c/.h
  - ExtensionEFL_ECC_RS485.c/.h
  - ExtensionEFL_EEC_Relay8.c/.h

  Entre les deux tirets bas (_) on lit toujours le type du connecteur d'extension.
- Les fonctions Init des cartes, qui transfèrent le câblage des cartes dans les tableaux et préparent les unités périphériques, ont aussi eu des noms plus spécifiques ; exemple : ExtensionEFL_Arduino_ElektorExtensionShield_Init(0) au lieu de ExtensionEFL_Init(). Pour garantir l'homogénéité, la fonction Init du µC s'appelle maintenant, par exemple, ControllerEFL_ATmega328_Init() au lieu de ControllerEFL_Init().

Voici à quoi pourrait ressembler une initialisation du µC et de toutes les cartes :

```
ControllerEFL_ATmega328_Init();
BoardEFL_ArduinoUnoCore_Init();
ExtensionEFL_Arduino_ElektorExtensionShield_Init(0);
ExtensionEFL_ECC_RS485_Init(2);
ExtensionEFL_EEC_Relay8_Init(3);
```

* Les fonctions Init des cartes d'extension se voient attribuer successivement le numéro d'ordre de bloc du connecteur d'extension auquel elles sont reliées. Ainsi, le câblage (des broches du µC à la périphérie extérieure) est cartographié correctement dans les tableaux EFL internes. On peut aussi enchaîner des cartes d'extension, comme ici (cf. le corps du texte et la fig. 1).

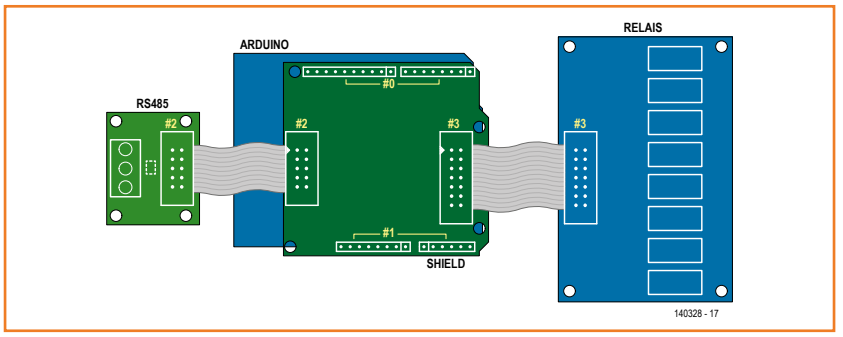


Figure 1.
Mise en chaîne de cartes d'extension : La carte *shield* est insérée dans deux embases de l'Arduino Uno (#0 = broches numériques, #1 = broches analogiques). À son tour, la carte *shield* met à disposition les connecteurs #2 (ECC) et #3 (EEC).

Quasi idéal

À l'aide du langage de programmation C et de l'EFL (*Embedded Firmware Library*) décrite dans Elektor [5], nous pouvons approcher l'idéal de très près. Jusqu'à présent, l'EFL ne pouvait être utilisée qu'avec un duo constitué d'une carte avec le µC et d'une carte d'extension. Pour pouvoir utiliser plusieurs cartes d'extension dans un même projet, il a fallu doter la bibliothèque de nouvelles fonctions et la restructurer quelque peu. Pour ne pas ennuyer nos lecteurs experts par moult détails (connus), les explications sont encadrées hors-texte. Voici le résumé des modifications les plus importantes :

- Jusqu'à présent, une paire de fichiers nommés ControllerEFL.h/.c contenait le code source spécifique au microcontrôleur, alors que, parallèlement, étaient données à chaque µC des fonctions standard telles que p. ex. IO_SetPinLevel(…) pour la commande des entrées et des sorties. Une paire de fichiers, toujours nommés BoardEFL.h/.c, rendait compte du câblage de la carte (évitant au développeur de l'application d'avoir à en prendre connaissance). Au lieu de noms (cryptiques) uniformes, ces fichiers portent dès à présent des noms parlants. Le fichier ayant trait à l'ATmega328 s'appelle maintenant ControllerEFL_ATmega328. Celui qui concerne la carte Arduino Uno, BoardEFL_ArduinoUnoCore.

- Nous pouvons maintenant utiliser plusieurs cartes d'extension dans notre projet, car les fichiers de code correspondants portent tous des noms différents, par exemple :
  - ExtensionEFL_Arduino_ElektorExtensionShield.c/.h
  - ExtensionEFL_ECC_RS485.c/.h
  - ExtensionEFL_EEC_Relay8.c/.h

- Les fonctions Init à appeler au départ ont été rebaptisées aussi de façon plus spécifique. Voici l'initialisation du µC et de toutes les cartes du premier exemple d'application décrit plus loin :

  - ControllerEFL_ATmega328_Init();
  - BoardEFL_ArduinoUnoCore_Init();
  - ExtensionEFL_Arduino_ElektorExtensionShield_Init(0);
  - ExtensionEFL_ECC_RS485_Init(2);
  - ExtensionEFL_EEC_Relay8_Init(3);

Les fonctions Init des cartes d'extension se voient attribuer le numéro du premier connecteur d'extension auquel elles sont reliées ; le numéro est incrémenté à l'initialisation de la carte (**fig. 1**).
La carte *shield* est montée sur l'Arduino Uno, qui définit deux connecteurs d'extension, auxquels sont donnés respectivement les numéros #0 (broches numériques) et #1 (broches analogiques). La carte *shield* est connectée à #0 et à #1 et possède, à son tour, deux connecteurs d'extension, à savoir #2 (ECC) et #3 (CEE). Le module RS-485 est connecté au premier, le second recevant le module de relais.

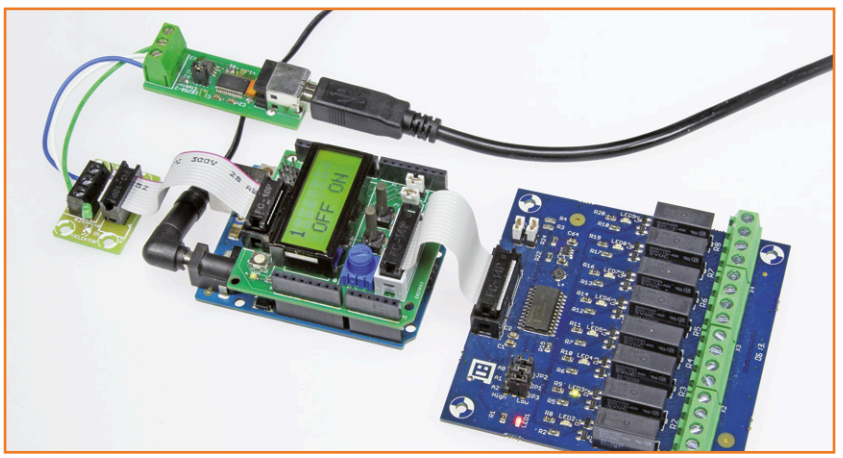


Figure 2.
Application 1 : Une interface utilisateur peut activer huit relais localement par télécommande via RS-485.

(Télé)commande de relais

Nous avons préparé trois projets de démo pour Atmel Studio 6, repris dans le téléchargement de cet article [6]. Dès la conception, un configurateur défini pour le PC regroupe automatiquement les fichiers dont a besoin un projet EFL sous Atmel Studio et qui les lie entre eux. Il reste à lui préciser la carte à utiliser pour le projet.

Commençons par le premier projet, une petite application de commande. Huit relais peuvent être activés localement par le biais d'une interface utilisateur ou télécommandés depuis un PC par un programme de terminal. Nous enfichons sur un Arduino Uno une carte *shield* à laquelle nous relions, par câble plat, une platine Gnublin à huit relais (**fig. 2**). Le module ECC RS-485, également relié au *shield*, est optionnel ; via RS-485 (deux lignes plus masse), on se connecte alors à un convertisseur RS-485/USB et au PC. L'application fonctionne aussi si l'on relie l'Arduino Uno directement au PC par USB.

On retrouvera le logiciel nécessaire dans le dossier ElektorShieldRelay du téléchargement. Un clic sur ElektorShieldRelay.atsln ouvre le projet dans Atmel Studio (**fig. 3**). On retrouve les fichiers intégrés dans Solution Explorer à droite. Il faudra impérativement jeter un coup d'œil au dossier Hardware ; outre les fichiers mentionnés pour le µC, la carte du microcontrôleur et les cartes d'extension, on y trouve également les nouveaux fichiers BlockEFL chargés des fonctions de bas niveau pour le pilote RS-485, les entrées et sorties numériques et l'affichage (cf. l'encadré « Fichiers BlockEFL »). Nous recommandons aux experts d'examiner le code du fichier ExtensionEFL_Arduino_ElektorExtensionShield.c (fig 3). Les blocs périphériques du *shield* sont enregistrés par le code dans un tableau de blocs EFL interne (**fig. 4**). Les boutons de la carte sont maintenant accessibles par le numéro de bloc #0, les LED par le numéro de bloc #1, vu qu'il y a déjà une LED sur la carte Arduino, dont la commande se fait par le numéro de bloc #0. Pour le potentiomètre, on crée un bloc CA/N #0 ; on passe ensuite à l'affichage qui reçoit le numéro #0. Pour finir, on reproduit le câblage des connecteurs ECC et EEC, par la création de deux nouveaux blocs pour connecteur (#2, #3). Les dites entrées sont, à leur tour, utilisées par le code des modules ECC et EEC. Finalement, par ces tableaux, chaque périphérique peut retrouver la broche du microcontrôleur à laquelle il est relié.

Jetons maintenant un coup d'œil au programme principal dans le fichier ElektorShieldRelay.c. La routine ApplicationSetup() contient toutes les initialisations, du µC aux bibliothèques, en passant par les cartes. Tout ceci peut rester en l'état, si tant est que l'on travaille avec une configuration matérielle identique.

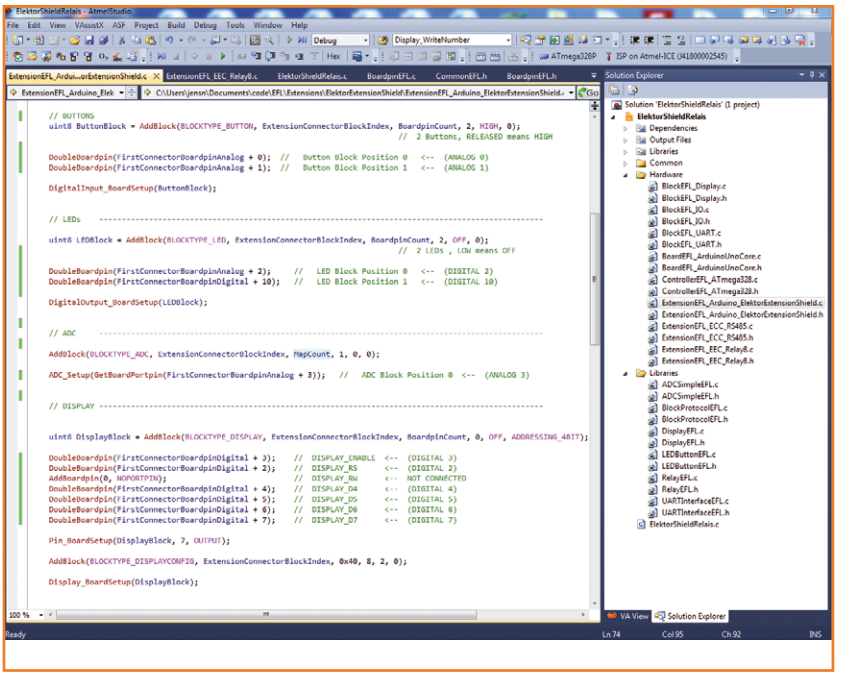


Figure 3.
La commande de relais sous Atmel Studio 6. Le code proposé ici est celui de l'initialisation de la carte *shield*.

Code compact

Le **listing 1** illustre l'application proprement dite. On retrouve, dans la fonction ApplicationLoop(), toutes les commandes à répéter continûment. Tout commence par l'interrogation des boutons ; on passe ensuite aux chaînes de caractères au format BlockProtocol [7], reçues par le biais de l'UART. La fonction ADCSimple_GetRawValue(0, 0) retourne la valeur de l'entrée du CA/N dans le bloc CA/N #0, puisque c'est en effet là qu'est connecté le potentiomètre. La valeur peut aller de 0 à 1023 (10 bits) ; par décalage à droite nous réduisons de 7 bits ce nombre de façon à obtenir un nombre compris entre 0 et 7. Nous pouvons ainsi sélectionner l'un des huit relais par la position donnée au potentiomètre. La valeur sélectionnée est visualisée sur la ligne 0 de l'affichage #0. On trouve, dans la fonction

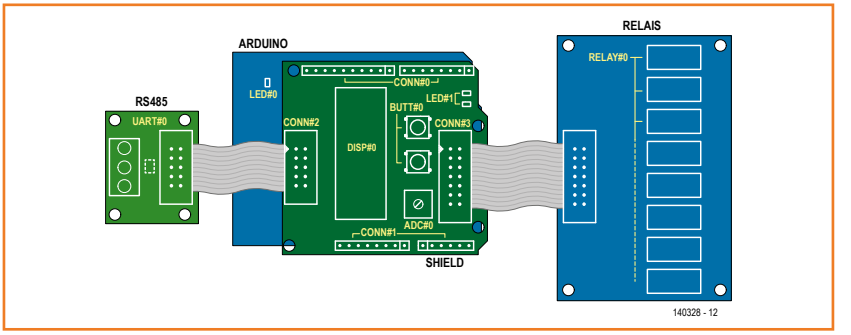
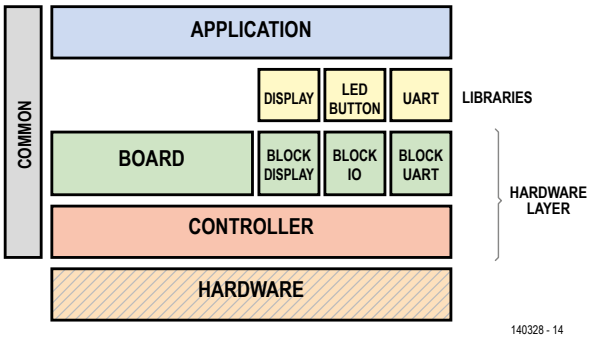


Figure 4.
L'ensemble des éléments périphériques est accessible facilement et de façon homogène par le biais des numéros de bloc.

Fichiers BlockEFL

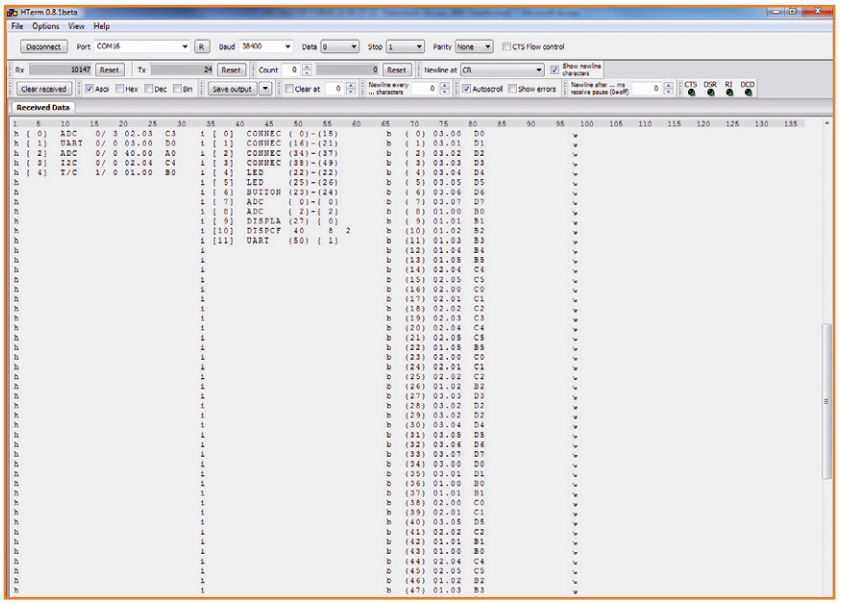
Jusqu'à présent, les fichiers `BoardEFL` et `ExtensionEFL` comportaient les fonctions dites `Low-Level/Block`, telle que `void Display_SendByte(uint8 DisplayBlockIndex, uint8 ByteToSend, uint8 DATABYTE_COMMANDBYTE)` extraite du câblage entre le microcontrôleur et l'affichage (4 bits et SPI respectivement). Une bibliothèque d'affichage comme `DisplayEFL` doit uniquement veiller à ce que les octets corrects soient envoyés à l'affichage, la technique d'envoi des octets vers l'affichage n'ayant aucun effet sur le code de ce dernier. Cette fonction de `Block` ne doit être définie qu'une seule fois dans le projet, si une de ses cartes comporte un affichage.



Judicieusement, nous créons pour cela, à l'intention des fonctions `Low-Level/Block` requises par un affichage, une paire de fichiers spécifiques dans le dossier `Hardware`. Si le projet comporte un affichage, il faudra, de plus, intégrer `BlockEFL_Display.h/.c`. Un autre fichier `BlockEFL` a charge des fonctions RS-485. Les fonctions `Block` pour les entrées et sorties numériques, telles que, p. ex. `SwitchDigitalOutput(uint8 BlockIndex, uint8 Position, uint8 ON_OFF)`, ont elles aussi été extraites du fichier `BoardEFL` et déposées dans un nouveau fichier `BlockEFL_IO`.
Normalement, les fichiers de carte ne contiennent plus maintenant que les fonctions `Init`, qui définissent le câblage et préparent les `Blocks` de périphériques intégrés tels que l'affichage. Pour ce faire, on appelle la fonction

```
void Display_BoardSetup(uint8 DisplayBlockIndex);
```

qui se trouve à présent dans `BlockEFL_Display.h/.c`.
Tous comptes faits, il y a bien quelques fichiers supplémentaires, mais l'EFL y a gagné en modularité.
Nous avons, en développement, un configurateur de projet, qui regroupe et lie automatiquement entre eux les fichiers nécessaires à un projet.



ButtonEventCallback(…), le code exécuté lors d'une action sur l'un des boutons. La variable ButtonPosition permet de différencier le bouton actionné (0 ou 1). Si c'est le bouton gauche (0), nous désactivons le relais sélectionné, si c'est le bouton droit (1), nous l'activons. Comme la fonction SwitchRelay(…) ne requiert qu'un 0 = OFF ou un 1 = ON en tant que troisième paramètre, le code est compact. On trouvera, dans la documentation Doxygen (cliquer sur Index.htm dans le téléchargement), plus d'infos au sujet de cette fonction et des autres fonctions EFL.

Figure 5. À l'aide d'un programme de terminal, on pourra visualiser les tableaux EFL. Au centre, on retrouve tous les blocs créés.

Le BlockProtocol [7] permet la télécommande des relais à partir d'un programme de terminal. La commande

R 0 2 + <CR>

active, dans le bloc de relais #0, le troisième relais (la base est zéro, rappelons-le, donc 0, 1, 2…). R 0 2 - <CR> désactive le dit relais. La commande L 0 0 + <CR> permet d'allumer les LED qui se trouvent sur la platine de l'Arduino. L 1 0 + <CR> et L 1 1 + <CR> sont les commandes correspondantes pour les LED de la carte *shield*. Essayez donc la commande x <CR> ; vous verrez s'afficher le contenu des tableaux EFL avec, au centre, tous les blocs créés (**fig 5**).

Listing 1. Commander un relais (à distance).

```
uint8 RelayPosition;

void ApplicationLoop()
{
  ButtonPoll(0);
  BlockProtocol_Engine();

  RelayPosition = ADCSimple_GetRawValue(0, 0) >> 7;
  Display_WriteNumber(0, 0, RelayPosition);
}


void ButtonEventCallback(uint8 BlockType, uint8
ButtonBlockNumber, uint8 ButtonPosition, uint8 Event)
{
  if (Event == EVENT_BUTTON_PRESSED)
  {
    ToggleLED(1, 0);
    SwitchRelay(0, RelayPosition, ButtonPosition);
  }
}
```

Mesures précises

Pour notre seconde application, nous débranchons la carte de relais et la remplaçons par le module CA/N à 16 bits présenté dans le numéro de septembre [4]. Nous relions ici la sortie A3 de l'embase femelle inférieure de l'Arduino (doublée sur le *shield*) par un fil volant à l'entrée AIN0 de la carte CA/N (**fig. 6**). Nous sommes alors en mesure de numériser la position du potentiomètre à l'aide du CA/N externe très précis.

Un double-clic sur ElektorShieldADC.atsln fait apparaître le code source. Comme le montre Solution Explorer, on trouve maintenant dans le projet, au lieu des deux fichiers ExtensionEFL_EEC_Relay8.c/.h, les deux fichiers ExtensionEFL_EEC_ADCModule16bit.c/.h, auxquels se sont en outre ajoutés ADC_ADS1x1xEFL.c et BlockEFL_DeviceRegister16.c. Le fichier BlockEFL_DeviceRegister16.c du dossier Hardware incorpore des fonctions de bas niveau (*Low-Level*) pour la commande d'une puce I²C à registre 16 bits. Le fichier ADC_ADS1x1xEFL.c se charge de la constitution correcte des octets écrits dans ce registre (la bibliothèque de Clemens Valens a été adaptée à l'EFL [4]).

Le développeur d'application n'a pas à s'en soucier, il suffit de savoir qu'un bloc CA/N additionnel, numéroté #1, est créé après initialisation de la carte (**fig. 7**). Il est possible maintenant d'accéder au CA/N externe, exactement comme on le ferait avec un CA/N interne du µC. La bibliothèque d'application ADCSimpleEFL.c met à disposition p. ex. la fonction ADCSimple_GetMillivoltValue(..) qui nous permet de déterminer la tension [mV] présente à l'entrée analogique d'un bloc CA/N déterminé.

Le code d'application proprement dit se trouve dans le fichier ElektorShieldADC.c dont le **listing 2** ne reproduit que la fonction essentielle. Il est facile deviner ce que fait l'application : numérisation de la tension présente sur la broche A3 de l'Arduino, tant par le CA/N interne que par le CA/N externe, puis affichage. Vérifiez avec

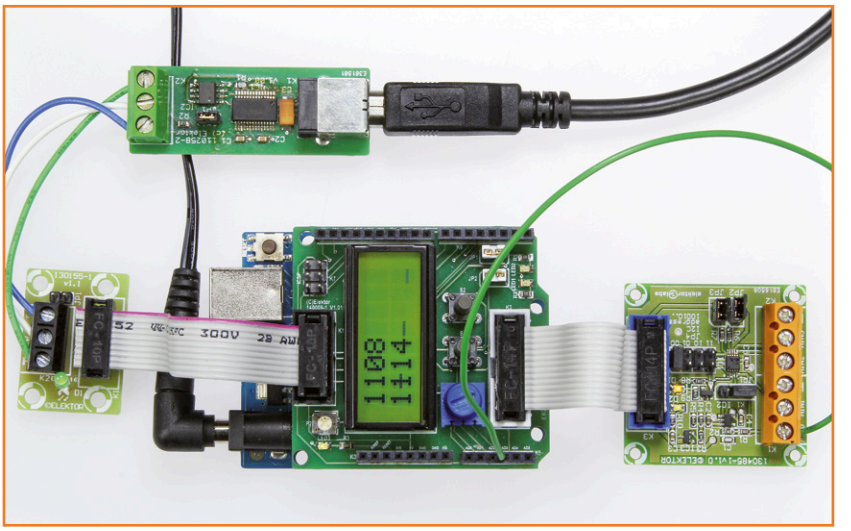


Figure 6.
Application 2 : Connexion d'un CA/N 16 bits externe à l'Arduino Uno. La ligne transmet la tension aux bornes du potentiomètre, ce qui permet une comparaison des résultats fournis par les CA/N interne et externe.

Virtualisation

Un expanseur de port et un CA/N externe sont deux exemples de puces offrant de nouvelles fonctions à un µC. Souvent, l'accès à ces puces se fait par le biais d'un bus I²C ou autre interface sérielle.

Une bibliothèque de prototypage modulaire devrait, le plus possible, effectuer une extraction du matériel utilisé. Pour le développeur d'une application (ou d'une bibliothèque de commande de périphérique), la cartographie des liaisons sur la carte, c-à-d. les broches du µC auxquelles sont connectés les périphériques, ne devrait pas avoir d'importance. Idéalement, il ne devrait pas être nécessaire de savoir si une entrée ou une sortie numérique est connectée à une broche de réelle du µC ou à une puce d'extension de port. Nous pouvons résoudre le problème en attribuant au µC, en plus de ses ports 0, 1, ..., etc., des ports virtuels qui commencent, pour les différencier des ports réels, par le nombre 0x40 = 64.

Si l'on voulait p. ex. activer un relais connecté à une extension de port, l'accès au relais est, comme d'habitude, transmis par la bibliothèque RelayEFL à la fonction `SwitchDigitalOutput(…)` qui se trouve maintenant dans le fichier `BlockEFL_IO.c`. Cette fonction s'informe, dans les tableaux EFL, à quel port et broche du µC correspond le relais et appelle la fonction `IO_SetPinLevel (...)` dans le fichier du µC. Pour le relais, un port 0x50 a été prévu dans les tableaux. En principe, le µC ne connaît pas de port 0x50. Cependant, pour de tels cas, lors de l'initialisation de la carte de relais, une fonction spéciale du code de la carte de relais été communiquée au code du µC pour qu'il l'appelle. C'est précisément cette fonction qui envoie ensuite les commandes I²C appropriées aux extensions de port présentes sur la carte afin de paramétrer la broche de sortie.

Le concept de la virtualisation s'est ainsi vu élargi aux entrées analogiques. Dans notre cas, le CA/N externe se trouve sur la carte d'extension CA/N EEC/Gnublin, de sorte que le code dans le fichier `ExtensionEFL_EEC_ADCModule16bit.c` doit se charger de la commande de son CA/N. Lors de l'initialisation de la carte par la fonction `ExtensionEFL_EEC_ADCModule16bit_Init(…)`, le CA/N externe est saisi comme un CA/N interne dans les tableaux EFL (dans notre cas le numéro de bloc 1), mais, à cet endroit il est fait référence à un port virtuel 0x40. Parallèlement prend place la préparation de l'interface I²C, le code du µC se voyant en outre communiquer la fonction `Virtual_ADC_GetValue(…)`.
Par la fonction `ADCSimple_GetRawValue(uint8 ADCBlockNumber, uint8 ADCPosition)`, l'utilisateur a maintenant accès à une broche de CA/N dans un bloc CA/N spécifique, indépendamment de la connexion au µC. Cette fonction appelle directement la fonction `ADC_GetValue(…)` du µC. Celui-ci effectue une recherche dans les tableaux EFL et reconnaît, sur la base du numéro de ports élevé, qu'il ne s'agit pas d'un CA/N interne, mais qu'il faut appeler la fonction `Virtual_ADC_GetValue(…)`, qui se trouve dans le code de la carte d'extension. Dans le cas présent, l'accès au CA/N externe se fait donc via I²C.
Dans notre application, non seulement de la fonction `ADC_GetValue(…)` est virtualisée, mais aussi la fonction `ADC_GetParameter(…)`, qui permet d'interroger la résolution et la plage de tension d'un CA/N. C'est ainsi qu'il devient possible de calculer les valeurs de CA/N en millivolts.

un bon multimètre : le CA/N externe est précis. Dans cette même application, nous obtenons un accès à distance par le BlockProtocol. La commande A 0 0 # <CR> instruit l'Arduino de renvoyer sous la forme de chiffres hexadécimaux la valeur tout juste échantillonnée par le CA/N interne. A 1 0 # <CR> renvoie la dernière valeur du CA/N externe. Les données diffèrent totalement, car il s'agit de valeurs brutes et non pas de données en millivolts. Pour éviter de trop gonfler la bibliothèque BlockProtocol, nous n'avons pas intégré de fonction de conversion des valeurs du CA/N en millivolts.

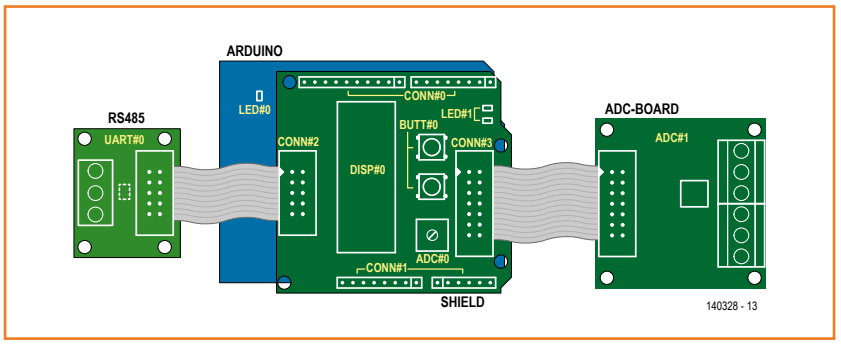


Figure 7. Les CA/N interne et externe sont accessibles par les mêmes fonctions, la distinction se faisant simplement par le biais des numéros de blocs #0 et #1.

Protocole maison

Puisque nous calculons déjà les données en millivolts dans l'application, ne pourrions-nous pas les récupérer simplement à cet endroit par le biais de l'UART ? Bricolons pour cela un petit protocole maison. Quand nous enverrons la commande 0 <CR>, il faudra que soit retournée la valeur [en millivolts] du CA/N interne. La commande 1 <CR> demande à l'Arduino de nous renvoyer la valeur du CA/N externe. C'est tout. Ainsi il sera possible de comparer les deux valeurs dans le programme de terminal.

Le **listing 3** reproduit le code correspondant (projet ElektorShieldADCMiniProtocol.atsln du téléchargement). ReceiveRingbuffer comporte l'adresse du tampon d'entrée en anneau dans lequel sont écrits les caractères qui, au travers de l'UART (UART-Block #0), parviennent à l'Arduino.

Dans la routine d'application SendMillivolt(…), la valeur [millivolts] en chiffres hexadécimaux est convertie et envoyée par le biais de l'UART #0. Dans le prochain numéro, nous vous présenterons le configurateur grâce auquel vous pourrez produire vous-même un projet EFL. Nous vous proposerons également un petit guide pour l'écriture de vos propres fichiers *Board*. Restez à l'écoute !

(140328 – version française : Guy Raedersdorf)

Listing 2. Mesure par CAN interne et externe.

```
void ApplicationLoop()
{
  ButtonPoll(0);

  BlockProtocol_Engine();

  uint16 ADCValue1 = ADCSimple_GetMillivoltValue(0, 0);
  Display_WriteNumber(0, 0, ADCValue1);

  uint16 ADCValue2 = ADCSimple_GetMillivoltValue(1, 0);
  Display_WriteNumber(0, 1, ADCValue2);
}
```

Listing 3. Lecture à distance des valeurs par un mini-protocole.

```
void ApplicationLoop()
{
  ButtonPoll(0);
  //BlockProtocol_Engine();

  uint16 ADCValue1 = ADCSimple_GetMillivoltValue(0, 0);
  Display_WriteNumber(0, 0, ADCValue1);

  uint16 ADCValue2 = ADCSimple_GetMillivoltValue(1, 0);
  Display_WriteNumber(0, 1, ADCValue2);

  while (Ringbuffer_IsEmpty(ReceiveRingbuffer) == FALSE)
  {
    uint8 ReceivedChar =
Ringbuffer_GetByte(ReceiveRingbuffer);
    if (ReceivedChar == '1')
    {
      SendADCValueOverUART(ADCValue2);
    }
    if (ReceivedChar == '0')
    {
      SendADCValueOverUART(ADCValue1);
    }
  }
}


void SendADCValueOverUART(uint16 ADCValue)
{
  uint8 sd[3];

  sd[0] = (ADCValue & 0xFF00) >> 8;
  sd[1] = ADCValue & 0x00FF;
  sd[2] = 13;

  UARTInterface_Send(0, sd, 3);
}
```

Liens

[1] www.elektor-magazine.fr/140009
[2] www.elektor.fr/developpement/gnublin
[3] www.elektor-magazine.fr/130157
[4] www.elektor-magazine.fr/130485
[5] www.elektor-magazine.fr/120668
[6] www.elektor-magazine.fr/140328
[7] www.elektor-magazine.fr/130154

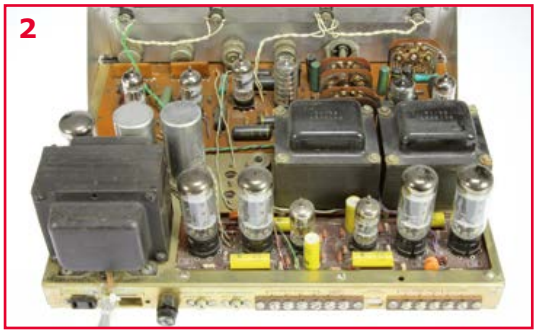
ampli HiFi à tubes
Heathkit AA-100 (1960)
ce kit à 85$ donnait du fil à retordre aux amplis commerciaux

Jan Buiting (Grand Manitou-ès-Antiquités)

Aujourd'hui objet de collection, l'amplificateur à tube AA-100 d'Heathkit brillait à sa sortie en 1960 par chacun de ses 25 W stéréo, ses tubes de sortie 'doux' et ses entrées. Sans oublier la joie de le monter soi-même pour pas cher.

Heathkit tient une place de choix dans l'histoire de l'électronique. Pratiquement n'importe quel électronicien de plus de 50 ans associera cette marque à un kit qu'il ou elle a construit, possédé—puis vendu—ou tout simplement admiré et rêver de posséder. La qualité des manuels *Heathkit* est sans doute le summum en termes de clarté et d'illustrations. Ils savent y faire, les Américains !


1


2

Histoire belge

Raymond, un lecteur néerlandais désireux de se défaire de certains de ses appareils, m'a proposé d'en décrire les plus remarquables dans Rétronique. Professeur émérite d'informatique, il a étudié puis enseigné dans plusieurs universités américaines, dont les célèbres californiennes, avant de retourner enseigner à l'université de Nimègue en Hollande. Retraité maintenant en Belgique, il fait toujours de l'électronique, avec une préférence pour les interfaces entre radios et ordinateurs.

Parmi la grande quantité d'appareils sans emballage qu'il m'a légués se trouvait un carton très lourd.

Ça doit être américain

Après avoir déballé le carton et vu "Heathkit/Daystrom" j'ai fait ce que la plupart des restaurateurs et collectionneurs font : chercher de l'aide auprès du fabricant d'origine, qui n'est plus très vaillant. Google m'a appris que : "L'ampli stéréo Heathkit AA-100 Deluxe dépasse de loin n'importe quel autre ampli stéréo en kit. Doté de nombreuses fonctions excitantes, cet ampli à haute fidélité de 50 W surpasse des modèles valant deux fois son prix. Il ne coûte que 84,95$ en kit, soit juste 1,70$ par watt. Avec deux canaux 25 W de puissance, il s'accommodera sans distorsion de toutes les tâches d'amplification dans votre chaîne stéréo."

Plus loin, je lis que " […] le boîtier en acier recouvert de vinyle marron clair ressemble à s'y méprendre à du cuir. Le fabuleux AA-100 est le jumeau du célèbre tuner stéréo AM/FM Heathkit AJ-30... ils se ressemblent, s'assemblent, et ne dépareilleront aucune pièce de votre maison." [1] J'ai également parcouru des forums audio et électronique rétro. J'y ai trouvé de bons conseils sur les améliorations à faire et sur les problèmes de restauration. J'ai ignoré les débats stériles sur les remplaçants des tubes, le son transparent, les solos de violon, condensateurs supersoniques et autres câbles à l'hélium.

X pour...

Une fois le AA-100 sur l'établi, je constate qu'il n'a pas de panneau arrière (!) et pas d'interrupteur de mise sous tension. L'arrière a été modifié au niveau de l'alimentation AC (beurk !) ; l'étiquette Heathkit/Daystrom est du mauvais côté ; les entrées de signaux sont au dessous (!); et les enceintes sont pathétiques comparées à l'apparence faste de l'ampli.

L'intérieur de cet AA-100 est soigné, sans surprise un peu poussiéreux (**fig. 1, 2**) et toutes les soudures sont aussi propres que le câblage et le positionnement des composants. Les boutons apparaissent blancs, mais des images retrouvées grâce à Google suggèrent qu'ils devraient être translucides et d'une couleur dorée. Fascinant.
Comme le savent mes lecteurs, je résiste à la tentation d'alimenter les vieux appareils aussitôt sortis de leur hibernation. Mon hésitation a été renforcée par la modif visible à l'arrière, au niveau de l'alimentation AC (**fig. 3**). La prise d'origine à deux voies SWITCHED n'était pas montée du tout et la prise NORMAL était surmontée d'une inscription manuelle indélébile : "Remote Switch" (NdT : interrupteur déporté). Ces prises, l'une avec interrupteur, l'autre sans, servent à l'origine à alimenter des appareils auxiliaires tels que le beau tuner AJ 30. Sans surprise, le carton contenait l'interrupteur déporté — rien de plus qu'un bout de câble à deux conducteurs avec une prise secteur américaine à un bout et un interrupteur à glissière en ligne en plastique à l'autre.

Sachant que Heathkit = États-Unis = 117 VAC / 60 Hz, là où en Europe c'est 230 V 50 Hz je suis resté méfiant vis-à-vis de l'alimentation secteur de l'ampli. J'ai trouvé dans le carton un autre objet fait-maison, un boîtier en plastique dur (**fig. 4**) avec un interrupteur, un indicateur de mise sous tension et une prise secteur IEC sur le dessus. Était-ce le transformateur 230 V/115 V ? Non, un autotransformateur permettant de réduire la tension du secteur de 9 ou 18 V (choix avec un fil).
Les amateurs de tubes et radios rétro d'Europe continentale ont, à raison, pris leurs précautions contre l'effet de l'augmentation graduelle de la tension du secteur de 220 V à 230 V nominaux (240 V maximum). Nous mesurons 228 V au labo d'Elektor.

3

4

Caractéristiques de l'AA-100

Puissance en sortie :	2 x 25 W stéréo ou 50 W mono
Puissance musicale :	30 W stéréo (0,7 % DHT à 1 kc)
	60 W mono (0,7 % DHT à 1 kc)
Sensibilité d'entrée	(V_{RMS} pour 25 W en sortie par canal)
	PHONO* en mono (seulement canal L) : 1,5 mV
	PHONO* en stéréo : 1,5 mV
	*pour têtes magnétiques
TAPE HEAD :	1,0 mV
TUNER :	0,2 V
AUXiliary 1 :	0,2 V
AUXiliary 2 :	0,2 V
Impédances d'entrée :	PHONO : 47 kΩ tel quel, adaptable aux têtes
	TAPE HEAD : 470 kΩ
	TUNER et AUXiliary : 250 kΩ chacune
Impédances de sortie :	4, 8 et 16 Ω par canal
Sortie d'enregistrement de cassette :	environ 0,5 V max. pour une résistance source de 600 Ω
	depuis un montage en cathode suiveuse. Min. 150 kΩ.
Réponse en fréquence :	± 1 dB 30 – 15.000 Hz à 25 W, depuis les entrées AUX
Séparation des canaux :	42 dB min. à 1 kc
Facteur d'amortissement :	15
Distorsion harmonique :	<0,5 % à 25 W, 1 kHz. <2 % à 25 W, 30 – 15.000 cps
Distorsion d'intermodulation :	<1 % à 25 W, 60 cps et 6000 cps mélange 4:1.
Bruit :	PHONO : 55 dB
	TAPE HEAD : 35 dB
	TUNER et entrée AUXiliary : 70 dB
Égalisation :	PHONO : courbe RIAA
	TAPE HEAD : courbe de lecture de bande NARTB
Réglage de tonalité :	Basses 15 dB ampl. /17 dB att. Aigus 12 dB ampl. / 20 dB att.
Tubes utilisés :	2x EF86, 4x 12AX7, 2x 7199, 4x 7591, 1x GZ34
Dimensions :	38 cm (L) x 11 cm (H) x 34 cm (P) (max., pieds compris)
Poids net :	13 kg

5

6

Je me suis dit que le transformateur d'alimentation dans l'AX-100 possédait certainement deux enroulements 115 V avec une prise milieu et que les primaires seraient reliés en série pour fonctionner avec du 230 V. Je m'étais encore planté : après avoir inspecté le câblage, très simple, côté secteur du transformateur, j'ai remarqué qu'il n'y a que deux fils pour le primaire et qu'ils sont reliés exactement comme sur le schéma Heathkit à l'exception de l'interrupteur de mise sous tension. Une seule petite différence : l'un des fils n'est pas noir, mais vert et noir. Ce n'est pas le transformateur mentionné dans le manuel ! Et en effet sa référence est 54X-89 (**fig. 5**), et pas 54-89. Modèle d'eXport ? Employés d'Heathkit qui lisez ces lignes, votre aide est la bienvenue.

Le câblage vers l'interrupteur de mise sous tension était installé à l'intérieur, mais pas l'interrupteur. Il n'était pas non plus dans le carton. Le trou rectangulaire que son absence laissait dans la face avant était couvert par l'étiquette Heathkit/Daystrom.
Arrivé là, j'étais sûr que l'ampli AA-100 utilisait du 230 VAC et que je pourrai le relier à mon bon vieux Variac (un autotransformateur toroïdal) pour le ramener à la vie. Plus tard.

En résumé

Le AA-100 combine circuit imprimé et câblage en l'air, mais sans faisceau de câbles. Seuls quelques composants sont montés en l'air ou sur une cosse à souder. Fait notable, deux composants à film épais sont utilisés dans les circuits de commande de tonalité G et D. Heathkit les appelle des PEC, *packaged electronic circuit*. Je ne suis pas fan des tubes montés sur circuit imprimé (surtout quand il s'agit de Pertinax fin), mais l'AA-100 n'a pas l'air de souffrir d'un quelconque problème, pas même sur les tubes de l'étage de sortie qui peuvent chauffer très fort. Les supports des tubes sont de bonne facture et possèdent de gros contacts à ressort.

Raymond m'avait fourni quelques notes indiquant que la carte de l'ampli avait été dotée de nouveaux condensateurs et résistances en 1992. Il vaut mieux prévenir que guérir. Des condensateurs au polypropylène ont été utilisés pour remplacer les « bombes noires » qui fuyaient. De nouvelles résistances sont venues relever les vieux modèles au carbone qui avaient vu leur valeur exploser. Les capas d'origine fournies par Heathkit arborent des noms absurdes : PYRAMID, SANGAMO, MICAMOLD TROPICAP (quand même !) qui feront grincer des dents les audiophiles. Beaucoup de condensateurs ont été mis de côté lors de la restauration, je les ai trouvés dans une boîte séparée (**fig. 6**). Malheureusement, tous les condensateurs électrolytiques tout neufs de la photo sont trop grands pour le boîtier. Plus triste encore, les bleus sont des électrolytiques de 180 µF, ce qui est trop pour le tube redresseur GZ34. La tension de 450 V qu'ils supportent est également trop faible.

Lors de la restauration de 1992, les tubes ont été changés, toutes les tensions mesurées et enregistrées, et les tubes d'origine ont été stockés dans des boîtes dépareillées (**fig. 7**). Heathkit a obtenu de Mullard UK qu'ils renomment des 12AX7 (ECC83) pour eux. J'ai vérifié tous les tubes d'origine sur mon testeur de tubes uTracer ; seul un 7591 était un peu faiblard au niveau de l'émission sans être pour autant inutilisable. Ça m'a fait plaisir de voir un tube redresseur GZ34 (les modèles Phillips d'origine valent leur pesant d'or) et un autre joyau de Philips Hollande : le EF86 ! En fait, seuls les tubes du circuit d'attaque et de l'étage de sortie sont des modèles américains.
Je prévois de réviser la carte tonalité/préampli de l'AA-100 ; je m'attends également à y trouver des condos et résistances en fin de vie.

Stéréo pré-HiFi & fioritures

Les schémas du circuit d'attaque et de l'étage final de l'AA-100 sont sur la **figure 8**, le schéma complet est facile à trouver, p. ex. sur Vintage Radio [2]. C'est un ampli de puissance *push-pull* classique dont les tubes de l'étage de sortie sont un peu inhabituels. Le double réglage (oui, gauche et droite séparées) graves/aigus est également remarquable, sans oublier la curieuse

ESTD 2004

Rétronique est une rubrique mensuelle sur les pages glorieuses et jaunies de l'électronique, avec occasionnellement des montages de légende décrits dans Elektor. Si vous avez des suggestions de sujets à traiter, merci de les télégraphier à redaction@elektor.fr

fonction SEPARATION et les connexions CENTER SPEAKER, le tout étant là, selon Heathkit, pour réduire l'effet de « creux au milieu » de certains programmes stéréo. Autre particularité : il était possible de décaler la phase du canal gauche. L'interrupteur et les fils ont été retirés par le précédent propriétaire.

Sur le schéma, la partie autour des pentodes 7199 (V7 et V8) sert d'amplificateur de tension. Elle est directement couplée à la partie autour de la triode configurée en diviseur de phase qui fournit des signaux en opposition de phase aux tubes 7591 de l'étage final (V9/V10 et V11/V12). Leur puissance se combine dans le transformateur de sortie dont la sortie est couplée à la charge : un HP 4, 8 ou 16 Ω. Les tubes fonctionnent comme des pentodes classiques dont les potentiels de plaque, écran et polarisation ont été choisis pour maximiser la puissance de sortie sans distorsion. Une polarisation fixe (tension de grille en entrée) de –16 V est dérivée d'un redresseur simple alternance au sélénium.

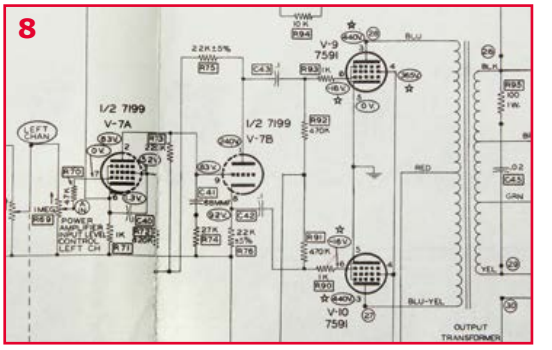
La valeur des résistances de fuite de grille R91/R92, R85/R86 a été largement critiquée pour être trop forte : les feuilles de caractéristiques de 7591 spécifient un maximum de 300 kΩ là où Heathkit utilise 470 kΩ. Une modif à faire absolument sur le AA-100 est de diminuer ces résistances à 220 kΩ ou 270 kΩ. Sur mon AA-100 le redresseur au sélénium été remplacé par une diode au silicium et les résistances associées ont été modifiées pour maintenir une polarisation à –16 V.

Le plastique c'est fantastique

Les boutons de la face avant et même certains des fils internes présentent de petites tâches qui font penser à de la moisissure ou de fines éclaboussures de peinture. Notamment les six gros boutons de plastique, lequel paraît blanc et non translucide et dorée. Les boutons noirs étaient couverts de petites tâches brunes heureusement superficielles, parties avec du détergent et une brosse à dents (**fig. 9**).

Le nettoyage des autres boutons et de la face avant a été remis à plus tard.

Maintenant que j'ai retrouvé confiance en l'alimentation et que la carte pilote/puissance possède de nouveaux condensateurs, je pouvais relier l'AA-100 à mon Variac et l'alimenter doucement en augmentant la tension par paliers de 20 V pour arriver à la tension nominale au bout d'une journée. Rien de fâcheux ne s'est produit. J'ai mis les haut-parleurs minables qui accompagnaient le AA-100 au grenier dont j'ai descendu, en échange, une paire de Philips 22RH427 retapés [3] (1973, boîte fermée, 35 litres, 3 voies, double woofer 20 cm). Ce gros ampli américain, si loin de chez lui et de ses 60 Hz, méritait un son *indie US*, des voix captées par un micro de 1946 et le son d'une guitare semi-acoustique Gretsch. J'ai donc passé *16 Horsepower Live [4].*



Comparé à mon Philips AG9015 (1966) révisé, le son est grave et pêchu. L'AA-100 est bien plus fort que l'AG9015 et devrait bien s'en sortir pour sonoriser mes petites fêtes. Même si son transformateur donne à mon AA-100 le facteur X, il faudra encore travailler sur le préamplificateur et lustrer le boîtier pour en faire la star de mon salon !

(140333 – version française : Kévin Petit)

Liens

[1] Musée Heathkit : www.heathkit-museum.com/hifi/hvmaa-100.shtml

[2] Schéma de l'AA-100 : www.vintage-radio.info/heathkit/

[3] 22RH427 : http://www.oudio.nl/speakers/22rh427.htm

[4] 16 HorsePower live @ Montreux 2010: http://youtu.be/G5pV2aQdf3o

hexadoku c'est l'heure des petits plaisirs

Ce jeu, on a beau savoir le goût qu'il a, on en redemande, mais on aime bien que ça ne vienne pas tout de suite, que ça résiste un peu, que ça fasse semblant d'être infaisable. Ça excite. Et puis on peut toujours s'y mettre à deux : « Chéri, je suis vraiment mal barré, tu voudrais pas jeter un coup d'œil à ma grille ? Je suis sûr que toi… »
Et souvent ça marche. De fil en aiguille, ça donne même des idées polissonnes. Taillez votre crayon, détendez-vous, mettez une musique bien douce et goûtez aux plaisirs partagés. Remplissez la grille selon les règles, envoyez-nous votre solution. Vous serez peut-être l'un des **cinq gagnants** d'un cadeau d'une valeur de **50 €.**
Après ça, vous serez d'attaque pour reprendre vos cogitations électroniques.

Une grille hexadoku est composée de chiffres du système hexadécimal, de 0 à F. Remplissez le diagramme de 16 x 16 cases de telle façon que **tous** les chiffres hexadécimaux de 0 à F (0 à 9 et A à F) n'apparaissent **qu'une seule et unique fois** dans chaque rangée, colonne et carré de 4 x 4 cases (délimités par un filet gras). Certains chiffres, déjà placés dans la grille, en définissent la situation de départ. Pour participer, inutile de nous envoyer toute la grille, il suffit de nous donner **la série de chiffres** sur fond grisé.

Participez et gagnez !

Nous tirons au sort cinq des réponses internationales correctes reçues dans les délais ; leurs auteurs recevront un bon d'achat de livres Elektor d'une valeur de **50 €.** À vos crayons !

Où envoyer ?

Envoyez votre réponse (les chiffres sur fond grisé) avec vos coordonnées par courriel, avant le **1er décembre 2014** à l'adresse **hexadoku@elektor.fr**

Les gagnants

La solution de la grille du numéro d'octobre (436) est : **C03EF**
Les cinq bons Elektor d'une valeur de **50 €** vont à :
Philippe Monnard (Suisse), **Ivan Corneillet** (États-Unis), **Siegfried Kepper** (Allemagne), **Steve Hasko** (Royaume-Uni) et **Jean-Paul Lagaisse** (Belgique).
Bravo à tous les participants et félicitations aux gagnants !

7	2						5	A						4	9
				6			9	D			5				
5	B	6			3					9			C	D	0
	0		9	4	C					F	1	5		E	
	1	7	A			6	C	E	5			B	4	3	
B	9		0			A			2			6		F	E
	3		8	5		9			6		B	C		1	
				B	D					A	C				
				A	5					1	2				
	6		E	C		1			9		D	8		5	
A	7		1			3			E			9		C	2
	D	F	C			4	0	8	A			E	3	B	
	8		B	D	7					C	0	3		A	
3	A	0			1					2			B	7	F
				F			6	7			3				
6	F						3	9						2	C

0	B	6	7	5	1	D	8	A	3	4	E	9	2	C	F
2	9	8	1	3	A	B	C	5	F	7	0	D	6	E	4
A	E	C	3	4	6	7	F	2	8	9	D	0	1	5	B
D	4	F	5	2	E	0	9	6	B	1	C	3	7	8	A
7	5	B	2	8	9	E	1	C	D	6	4	A	0	F	3
9	8	D	A	6	5	C	0	3	E	F	B	1	4	2	7
C	3	1	6	D	F	4	7	0	2	5	A	B	E	9	8
E	F	0	4	A	B	2	3	9	1	8	7	5	C	D	6
F	0	2	B	7	D	1	A	8	9	E	6	C	3	4	5
1	D	3	9	0	2	F	4	B	7	C	5	6	8	A	E
8	6	4	C	E	3	5	B	D	0	A	F	7	9	1	2
5	7	A	E	9	C	8	6	1	4	2	3	F	B	0	D
6	C	9	F	1	7	3	2	E	A	D	8	4	5	B	0
3	1	E	D	B	4	6	5	F	C	0	2	8	A	7	9
4	2	5	0	C	8	A	D	7	6	B	9	E	F	3	1
B	A	7	8	F	0	9	E	4	5	3	1	2	D	6	C

Tout recours est exclu de même que le sont, de ce jeu, les personnels d'Elektor International Media et leur famille. Un seul gagnant par foyer.

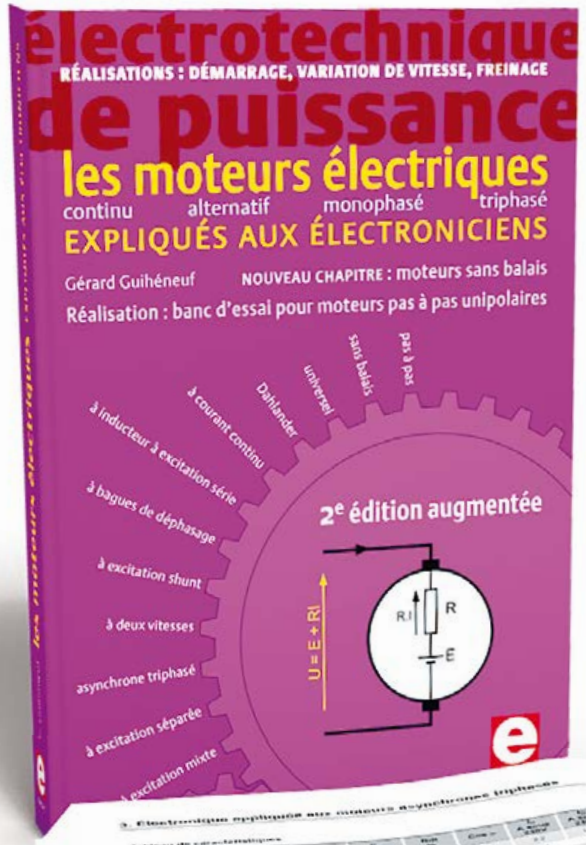
Nouvelle édition augmentée

100 pages supplémentaires !

électronique de puissance

les moteurs électriques

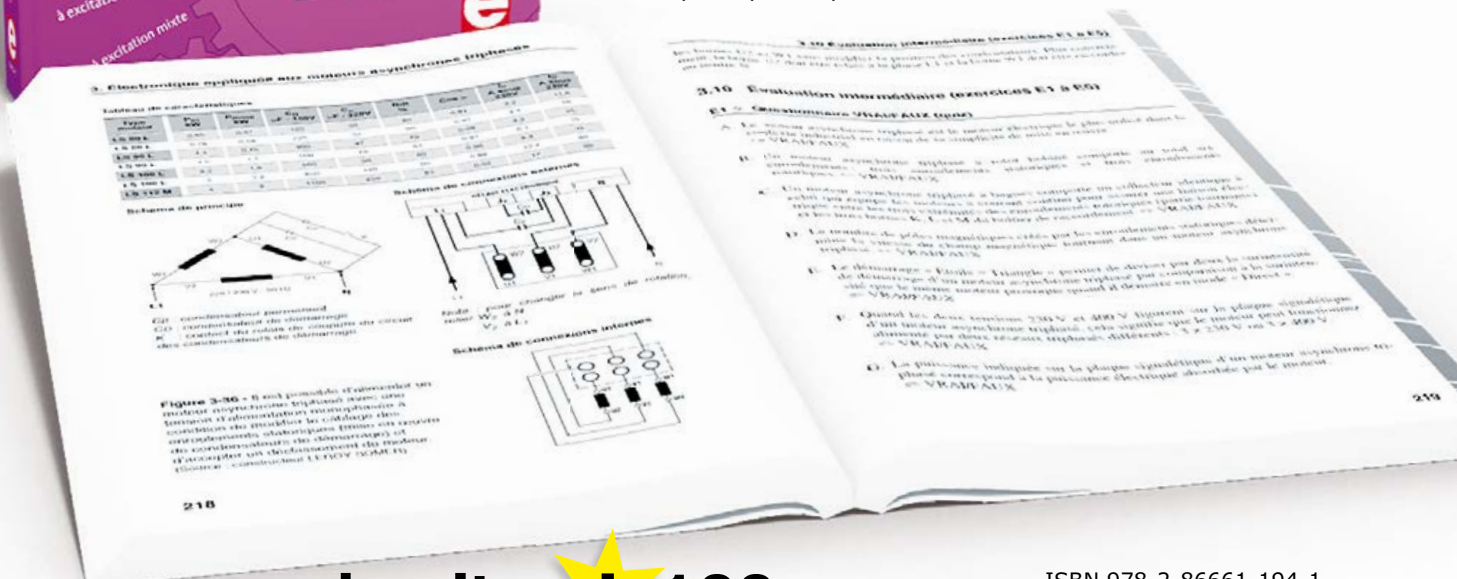
expliqués aux électroniciens



Ce livre en quatre chapitres offre une information accessible et digeste : constitution, fonctionnement, caractéristiques, domaines d'utilisation des moteurs, pour proposer aussi des réalisations électroniques simples et concrètes.

Le premier détaille les principes de variation de la vitesse des moteurs à courant continu. Les moteurs à alimentation alternative monophasée du 2e chapitre font appel à une électronique de puissance. Dans le 3e chapitre sont décrits les moteurs asynchrones triphasés, le moteur électrique le plus utilisé dans l'industrie.

Cette nouvelle édition s'enrichit d'un chapitre consacré aux **moteurs sans balais (*brushless*), du moteur synchrone triphasé de plusieurs centaines de kW au moteur à courant continu de quelques centaines de watts, en passant par les moteurs pas à pas ou encore le surprenant moteur linéaire.** Le lecteur découvrira leurs modes de commande : codeurs incrémentaux ou absolus associés à un onduleur commandé en courant ou en tension ou bien capteurs à effet Hall pour l'autopilotage, commande en pas entiers, demi-pas, micro-pas… L'auteur propose également de réaliser un banc d'essai pour moteurs pas à pas unipolaires.



nouveau chapitre + 100 pages

ISBN 978-2-86661-194-1
432 pages | 46,50 €

elektor

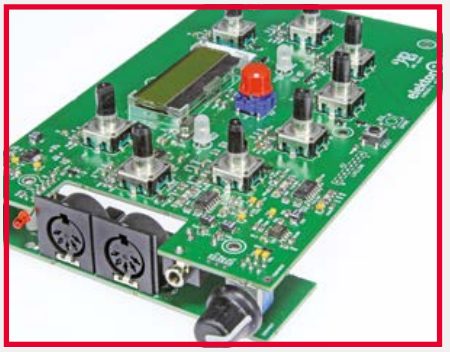
www.elektor.fr/moteurs

bientôt dans Elektor

numéro double janvier/février 2015

La tradition des numéros doubles d'hiver est désormais établie. Pour commencer 2015, nous vous proposerons donc un numéro spécialement riche et varié à souhait. La puissance des µC et de leurs fonctions auxiliaires de plus en plus nombreuses est irrésistible, mais l'électronique analogique et les petits circuits d'expérimentation ne perdent pas leurs droits. un échantillon prélevé au moment de boucler cette avant-première :

récepteur FM simple – ***booster*** DC/DC pour LED – BoB de **capteur d'humidité** – module **True-RMS** pour multimètre numérique – **modulateur IR** CMOS – moTule **T-Board XBee** – affichage des bits du **signal horaire atomique** – multitesteur de composants universels ELPP



synthé de musique J:B

Enfin un nouveau synthé dans Elektor ! En dépit de velléités de retour à l'analogique, ce sera un instrument résolument et intégralement numérique. Pas de monstre à boutons ni de commutateurs par douzaines ni de salade de cordons, non, rien qu'une carte compacte, basée sur le projet J²B ARM Cortex-M3 de septembre 2011, entièrement revu, même le contrôleur a changé. Comme il se doit pour un synthé Elektor, celui-ci sera orienté vers l'expérimentation, à code source ouvert et l'accent sera mis sur la portabilité du code.

générateur de fonctions Platino

Cette fois la carte polyvalente Platino devient un géné de fonctions, utile pour tester toutes sortes de circuits. Différentes formes d'ondes sont disponibles : sinus, carré, triangle, dents de scie, impulsion et formes d'onde arbitraires que l'utilisateur peut créer lui-même, sans oublier l'entrée de modulation de fréquence. Commande simple grâce à un codeur rotatif unique associé à un confortable LCD.

Sous réserve de modification
Parution du numéro de janvier-février : 6 janvier

reichelt.fr elektronik

Votre partenaire en ligne compétent pour

Eléments de construction
Alimentation électrique
Technique de mesure
Technique atelier & de brasage
Technique domotique & sécurité
Technique réseau
Technique PC
Technique Sat/TV
Communication

LE TOUT DERNIER ORDINATEUR MONOPLATINE

À UN SUPER PRIX !

S'adapte
parfaitement à Banana Pi

Kit refroidissement
en 4 parties

- Montage simple
- Profil extrêmement réduit

TEK-BERRY COOL **4,65**

Adaptateur WLAN-USB,
150 Mbits/s

- Compatible WEP, WPA, WPA2
- Compatible WPS

EDIMAX EW-7811UN **6,90**

Adaptateur de secteur micro USB
6W, 5V, 1,2A

- Double isolation
- Résistant aux surcharges et court-circuits

HNP06 MICRO USB **6,40**

Carte micro SDHC
Classe 10

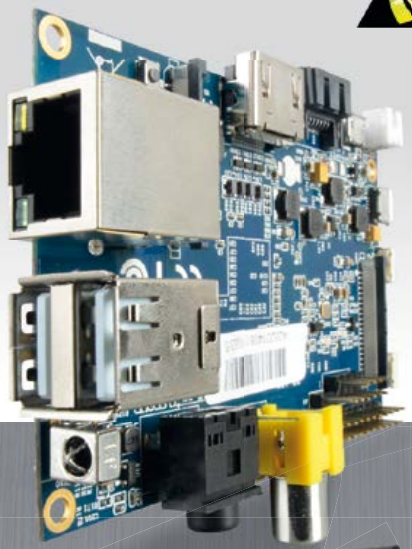
- Adaptateur inclus
- Capacité : 8 GB

SDC10/8GB **6,80**

Banana Pi
Compact comme le Raspberry Pi, mais nettement plus de puissance !

- ☑ **CPU plus rapide**
- ☑ **Plus de mémoire de travail**
- ☑ **Ethernet rapide**
- ☑ **Récepteur infrarouge**

- CPU : A20 ARM® Cortex™-A7-Dual-Core
- Mémoire de travail : 1 GB DDR3 SDRAM
- Prises : fente à carte SD, port SATA, HDMI, éthernet 10/100/1000 Mbits/s, 2 ports USB, prise caméra CSI, 26 PIN externes avec I²C, SPI, UART, CAN, sortie audio 3,5 mm
- Alimentation électrique par micro USB



BANANA PI **42,90**

Adaptation nec plus ultra !
Boîtier en plastique pour votre Banana Pi

Combinez les couleurs selon vos préférences. Grâce aux rainures clip-in, pas besoin de vis ni de colle désagréable.

- Toutes les découpes sont prédécoupées
- Pieds en caoutchouc antidérapant
- Surface très brillante

4,20

Dessus	CB DBBP AA GE	
Dessous	CB DBBP BA GE	6,20
Dessus	CB DBBP AA TR	4,70
Dessous	CB DBBP BA TR	6,90
Dessus	CB DBBP AA SW	4,20
Dessous	CB DBBP BA SW	6,20
Dessus	CB DBBP AA CF	4,70
Dessous	CB DBBP BA CF	6,90

livré sans Banana-Pi

Commander maintenant! **www.reichelt.fr**

Assistance téléphonique en anglais : **+49 (0)4422 955-333**

Modes de paiement internationaux: VISA MasterCard PayPal

Prix du jour ! Prix à la date du : 10. 10. 2014

Les langues de notre boutique:

Intégration de la reconnaissance de gestes 3D et du suivi des mouvements

grâce à la technologie GestIC® de Microchip

La technologie brevetée GestIC® 3D de Microchip offre une solution embarquée sur puce, de reconnaissance des gestes en temps réel. Le MGC3230 permet des interfaces utilisateur ultra innovantes sur tout type de produit pour un niveau de prix très économique.

MGC3130

- Le MGC3130 fonctionnant avec la bibliothèque de gestes Colibri, il permet la reconnaissance avancée de gestes 3D complexes sans pour autant nécessiter un processeur hôte
- Le MGC3130 envoie les données détectées (gestes, coordonnées XYZ) à l'hôte via un bus I²C™ ou des E/S à usage général (GPIO)

microchip DIRECT
www.microchipdirect.com

MICROCHIP

microchip.com/gestic