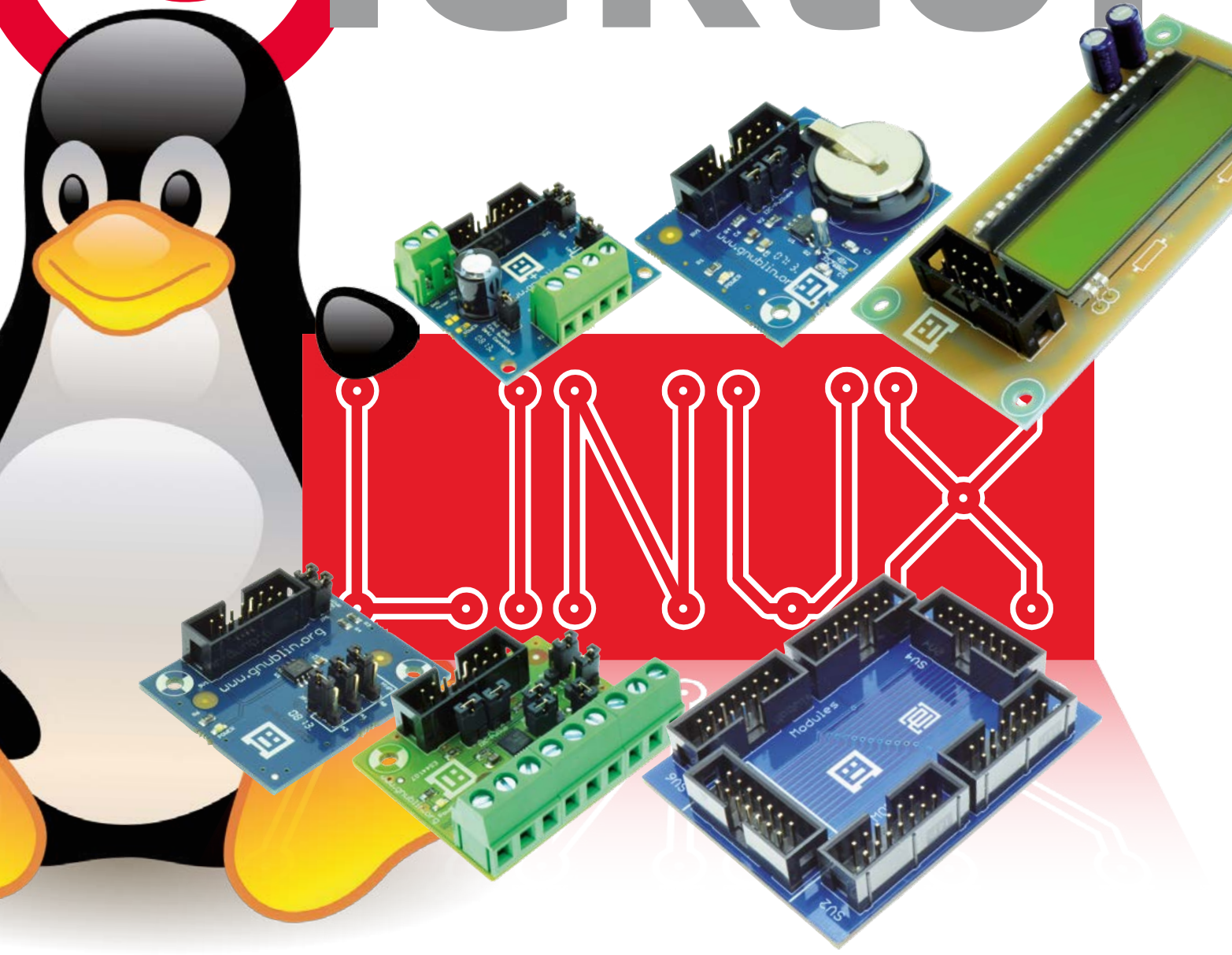
www.elektor-magazine.fr

.magazine

septembre 2013 | nº 423

elektor



● **chaîne d'outils et API C/C++** pour Linux et Raspberry Pi | **ampli audio de puissance 200 W**

elektorcardi♥scope Android 2e partie | **photodétecteur avec Arduino**

de BASIC à Python (3) | thermomètre USB | liaison RF modulaire à code Manchester (1)

● les bibliothèques Design Spark ● les dessous (physiques) de l'internet

L 19624 -423- F: 7,80 € - RD

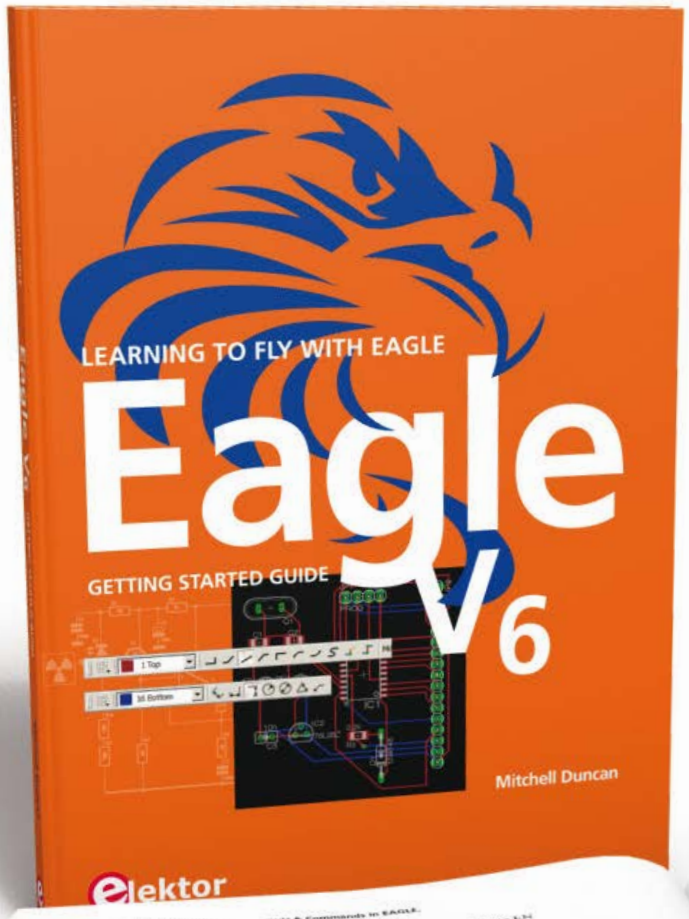
ISSN 0181-7450

(B) 7,80 € • (CH) 13,20 FS • (CAN) 11.99 $ca • (And) 7,80 €

DOM surface 8,80 € • DOM avion 9,90 € • N Cal/S 1050 cfp

Guide de démarrage pour Eagle V6

NOUVEAU



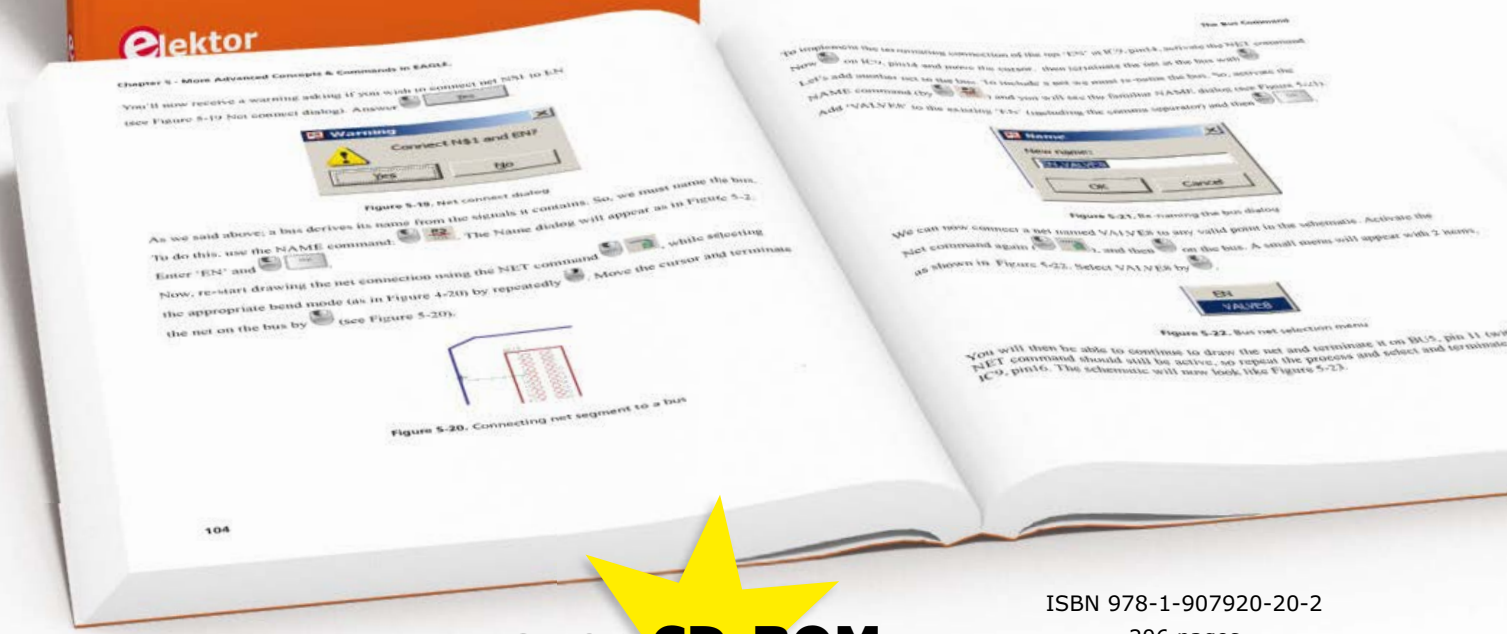
LEARNING TO FLY WITH Eagle V6

GETTING STARTED GUIDE

Que vous soyez électronicien amateur ou professionnel, ce guide de prise en main vous initiera rapidement au logiciel de conception de circuits imprimés EAGLE de CadSoft.

Ce livre (en anglais) a été écrit pour tous ceux qui souhaitent découvrir les possibilités d'EAGLE. L'auteur s'adresse aussi bien au novice en conception de circuits imprimés qu'au professionnel déjà familier d'un autre logiciel de CAO, mais qui envisage de passer à EAGLE.

Ce livre est accompagné d'un CD-ROM gracieusement fourni par CadSoft Computer GmbH (www.cadsoft.de). Il contient une version d'Eagle pour Windows, Linux et Mac.



avec CD-ROM

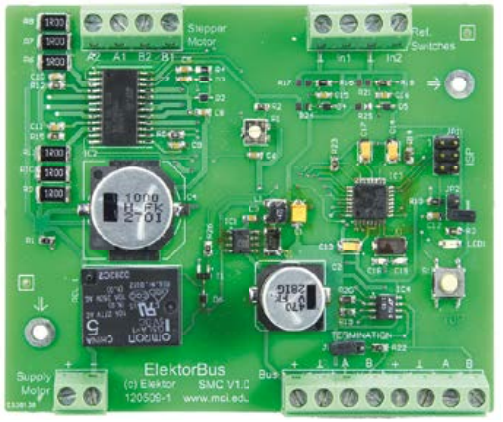
ISBN 978-1-907920-20-2
206 pages
34,50 €

elektor

www.elektor.fr/eagle

•Elektor e-CHOPPE

en partenariat avec Eurocircuits



1

3

2

4

5

1 Pilote de moteur pas à pas

Pour ElektorBus (module)

Voyager avec l'ElektorBus ouvre de nouveaux horizons. Grâce à sa modularité logicielle autant que matérielle, ce bus accélère le développement d'applications. Exemple pratique : la mise au pas des moteurs !

Réf 120509-91

2 Barostick

Clés USB, clés du succès ? Elles sont partout, elles sont notre album d'images, de vidéos, de musiques préférées, d'articles, de fichiers et même de températures. Et la pression atmosphérique, y aviez-vous pensé ? C'est fait : sur un baromètre sans mercure, avec un capteur Bosch, hectopascals et degrés Celsius rejoignent Windows pour se faire tirer le portrait.

Réf 120481-91

3 LCR-Mètre 0,05 %

Le luxe de la précision accessible à tous

La remarquable précision de cet appareil et son étonnant confort d'utilisation sont le résultat d'une étude soignée. Il marche si bien derrière sa façade dépouillée, qu'on en oublierait presque la subtilité des techniques de mesure mises en œuvre. L'occasion rêvée, pour nos lecteurs passionnés par la mesure, de se faire plaisir. Si, comme nous, les prodiges que les techniques modernes mettent à notre portée vous émerveillent, venez palper le pouième de volt.

Module carte principale réf 110758-91
Module de l'extension afficheur et clavier réf 110758-92
Les deux modules réf 110758-93

4 Module réseau universel

Connectivité internet pour les nuls

Le raccordement de sa propre électronique par une liaison internet est une fonction impressionnante. Ce module réseau universel - composé d'un circuit compact, d'une bibliothèque de logiciels libres et d'un microcontrôleur utilisable immédiatement comme serveur web - s'adresse même aux débutants. Quant aux électroniciens plus chevronnés, ils apprécieront à leur juste valeur les caractéristiques comme la communication SPI, PoE, et plus encore !

Réf 100552-91

5 Accéléromètre DIPlômé

Accessoire d'adaptation pour puce d'accéléromètre (dont les surfaces de contact se trouvent sous le boîtier de type LGA) à une carte de prototypage ordinaire.

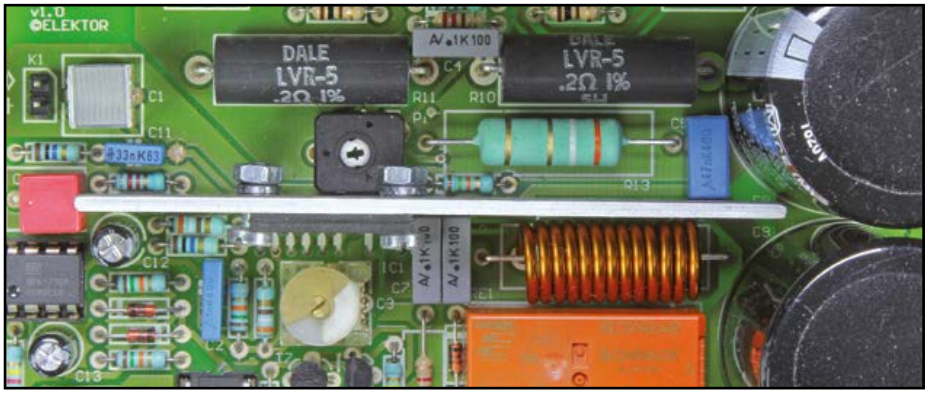
Réf 090535-91

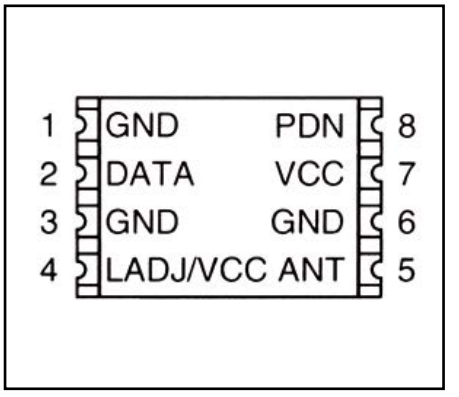
Informations et gamme complète sur :

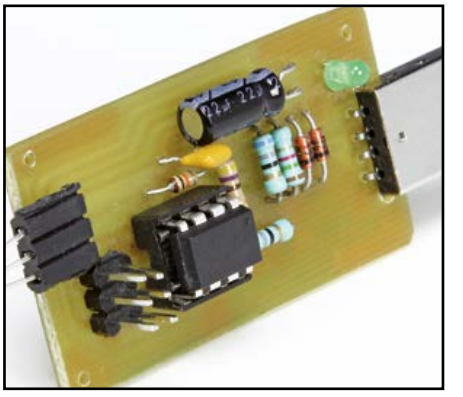
www.elektorpcbservice.com

sommaire

elektor magazine







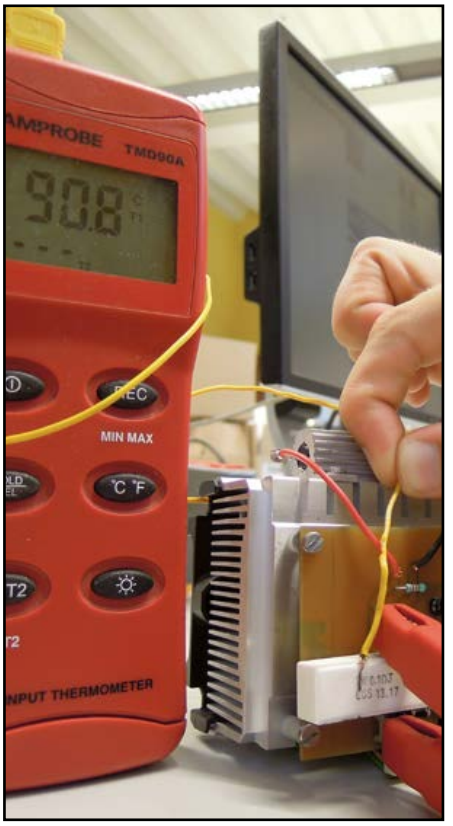
communauté

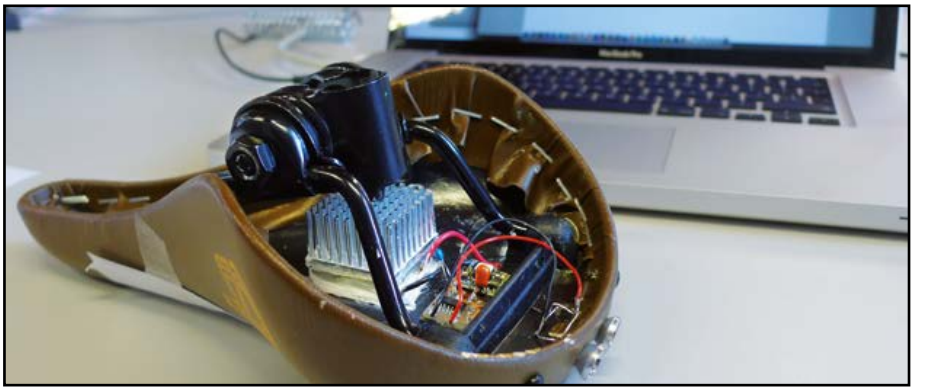
DesignSpark

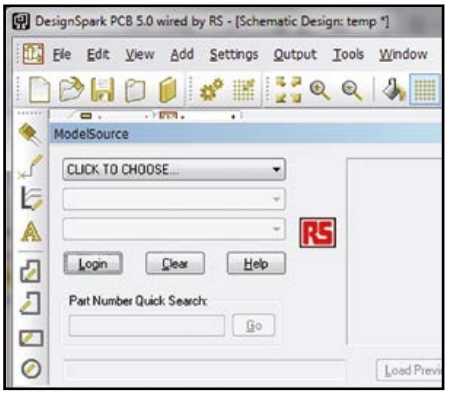
labs

industrie

projets







36ème année, n° 423 septembre 2013
ISSN 0181-7450
Dépôt légal : août 2013
CPPAP 1113 U 83713

ELEKTOR / 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
lundi, mardi et jeudi de 8h30 à 12h30
Fax : (+33) 01.49.19.22.37
www.elektor.fr

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

Elektor est édité par Elektor International Media B.V.
Siège social : Allee 1 - 6141 AV Limbricht, Pays-Bas

Imprimé aux Pays-Bas
par Senefelder Misset – Doetinchem

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

Le virtuel des bibliothèques et les constantes chimiques du sang

Vous voilà, lecteurs francophones, de retour des vacances, puisqu'elles sont obligatoires et que la France s'est arrêtée pendant deux mois. Elektor ne s'est pas arrêté et vous trouverez ici un sommaire équilibré entre clavier et fer à souder.
Fer à souder, réellement ?
Les composants modernes se soudent au four ou à l'air chaud. Au point qu'on précise dans l'article sur le **thermomètre USB** que tous ses composants sont « traversants ». L'**amplificateur audio de 200 W**, lui aussi, est complètement « traversant ». C'est donc un retour en force de l'analogique. Le **photodétecteur**, s'il utilise un Arduino pour commander l'obturateur de l'appareil photo, est un bel exemple de détournement de matériel : un détecteur de mouvement à infrarouges déclenche la prise de vue d'animaux. Dans ce cas, l'Arduino n'est qu'un *moyen*, simple, rapide et souple, de remplir les fonctions élémentaires d'un *trigger* de Schmitt et d'un monostable.
Les **cartes d'extension** pour la carte *Linux Elektor* et *Raspberry pi* sont des ponts entre le matériel et le virtuel. De même, les trucs et astuces de *DesignSpark* relient le virtuel des bibliothèques de composants au cuivre des circuits imprimés.
Pour le clavier – avant que tout passe par des écrans tactiles ou la reconnaissance vocale – ce numéro fait une belle place à la dernière livraison de la série **Python**.
Le logiciel du microcontrôleur de l'**elektorcardi♥scope** Android rassurera les hypocondriaques, mais ils devront encore se *ronger les sangs* en attendant le prochain numéro pour la réalisation et le mode d'emploi de cette interface d'ores et déjà disponible comme module assemblé. De l'hémoglobine, il y en a jusque dans la rubrique *Rétronique* qui présente un analyseur de pH sanguin. Ce sera le prétexte à s'intéresser aux constantes chimiques du sang, comme la pression partielle d'oxygène et de gaz carbonique, et à s'imaginer encore, les bienheureux, de nouvelles sources d'inquiétude.

Portez-vous bien

Jean-Paul Brodier

PS : Jean-Paul B. souhaite à Dominique C. une retraite longue, riche et paisible.

Notre équipe

Rédacteur en chef :	Denis Meyer (redaction@elektor.fr)
Directeur éditorial :	Wisse Hettinga
Rédaction internationale :	Harry Baggen, Jan Buiting, Thijs Beckers, Eduardo Corral, Jens Nickel, Clemens Valens
Laboratoire :	Thijs Beckers, Ton Giesberts, Luc Lemmens, Tim Uiterwijk, Jan Visser
Ont coopéré à ce numéro :	Jean-Paul Brodier, Robert Grignard, Hervé Moreau, Kévin Petit, NN
Service de la clientèle :	Jolanda van Kruchten
Graphistes :	Giel Dols, Jeanine Opreij, Mart Schroijen
Elektor *online* :	Daniëlle Mertens & Patrick Wielders

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

United Kingdom
Wisse Hettinga
+31 (0)46 4389428
w.hettinga@elektor.com

USA
Hugo Van haecke
+1 860-289-0800
h.vanhaecke@elektor.com

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

Netherlands
Harry Baggen
+31 46 4389429
h.baggen@elektor.nl

Spain
Eduardo Corral
+34 91 101 93 95
e.corral@elektor.es

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

Sweden
Wisse Hettinga
+31 46 4389428
w.hettinga@elektor.com

Brazil
João Martins
+35 19 14 46 55 77
j.martins@elektor.com

Portugal
João Martins
+35 19 14 46 55 77
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 · techthefuture

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 lundi, mardi et jeudi de 8h30 à 12h30 ou par courriel : service@elektor.fr

le monde d'Elektor

compilé par
Wisse Hettinga

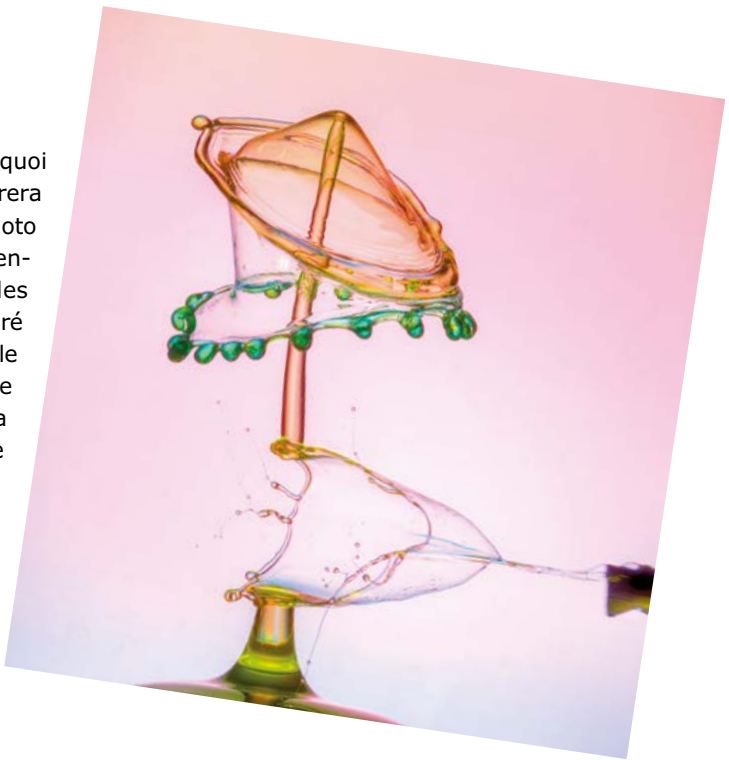
Le monde d'Elektor est un grand carrefour international où se croisent les voies les plus diverses et se créent les liens les plus improbables.

Monsieur CAN

Dans le cadre d'un projet d'article sur le bus CAN, Jan Visser du labo d'Elektor, a invité le concepteur Hugo Stiers à présenter son circuit dans nos locaux. La visite a duré tout un après-midi tant la matière s'est révélée intéressante. Hugo, ancien instructeur chez *DAF trucks*, expert dans le domaine du CAN, est un homme de terrain, qui ne se déplace jamais sans une *camionnée* de cartes, de câbles, de classeurs et de portables… pour le cas où…
Quel plaisir de voir naître et croître la complicité entre électroniciens passionnés ! Le temps, denrée autrement si précieuse, semble tout d'un coup disponible à profusion. Personne ne songe plus à regarder l'horloge. Le projet CAN de nos comparses devrait figurer au sommaire de ce numéro… sauf changement de dernière minute, auquel cas ce sera pour le prochain.

Goutte à goutte

Arduino vous tente, mais vous ne savez pas trop quoi en faire ? L'exemple de Huib Theunissen vous inspirera peut-être : photographe amateur, passionné de photo ultrarapide, il s'est construit un dispositif de déclenchement à base de carte Arduino dont il utilise les sorties pour commander les gouttes de liquide coloré qu'il photographie, l'éclairage, l'obturateur et, dans le cas très particulier de la photo ci-contre, une arme à feu. L'Arduino fait (presque) tout, certes, mais la patience et l'ingéniosité sont évidemment à mettre au crédit du seul photographe. Bravo ! Avec cette image intitulée *The Speed of Life*, Huib a emporté un prix au concours Nikon. Admirez le fruit de son travail sur *www.facebook.com/druppelfotos*.
Et si vous avez vous-même une application d'Arduino dont vous pensez qu'il faudrait en parler ici, faites-le nous savoir.

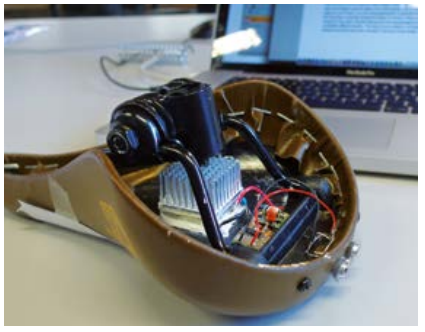


Où somme-t-on ?

Nous somme-t-au Kite, à Oxford, un pub à côté de la gare, avec quelques chambres pour voyageurs fatigués. Mon collègue Johan Dijk et moi sortons d'un rendez-vous d'affaires chez RS Components. Alors que nous discutons fébrilement des chances de concrétisation d'un ambitieux projet de coopération entre Elektor et RS, autour du brouillon de contrat que nous venions d'éplucher avec nos partenaires, la serveuse derrière le comptoir nous propose sa participation : « J'ignore si ça vous portera bonheur, mais je pourrais vous prendre en photo. Ça vous fera toujours un bon souvenir ! » ... *And here we are*!

Le fantôme de l'eau paiera

On rencontre de tout au château d'Elektor, même des membres de la LPI (*League of Paranormal Investigators*) sur les traces de fantômes ! J'en ai déjà croisé qui cherchaient les restes de soldats de Napoléon tués jadis aux abords du château, d'autres plus récemment qui cherchaient à entrer en contact avec l'esprit d'Entgen Luyten, la dernière «sorcière » de Hollande dont la légende rapporte qu'elle se serait pendue sous les combles du château. Tout cela en se prenant très au sérieux et à grand renfort de technique... charlatanesque.



Le feu au cul des Hollandais(es)

Les Néerlandais sont à n'en pas douter une nation de cyclistes économes. Chaque soir, au crépuscule, se pose pour des millions de personnes le problème de leur éclairage cyclopédique. Dynamos mues par la force du cycliste, ou piles et batteries rechargeables, il y a toujours un moment où ces trucs-là vous lâchent. Comme il est vain d'espérer attendrir la maréchaussée néerlandaise, Wouter Eisema, ingénieux élève ingénieur, a imaginé une solution à notre connaissance inédite. La chaleur de la selle (ou plus exactement du postérieur du cycliste) est convertie en électricité à l'aide d'éléments Peltier pour allumer des LED montées sous la selle. Rien ne se perd au royaume d'Orange… une trouvaille qui fera peut-être un article dans un prochain numéro !

De la table traçante Mondrian à une grosse CNC

En 1987, Elektor publiait *Mondrian*, une table traçante à trois couleurs, qui a suscité un vif intérêt, notamment chez des artistes tels que Jonas Vos. C'était le premier montage d'Elektor que Jonas assemblait et son premier pas dans le monde de l'art avec des machines. Aujourd'hui il enseigne à la Jan van Eyck Academy, célèbre école internationale d'art de Maastricht, et il s'est lancé dans la réalisation de sa propre machine à commande numérique, une grosse CNC qui fraise toutes sortes de matériaux sur 3 axes dans un volume de 180x240x80 cm. Sur la photo, Jonas aux commandes de sa machine.

Jan van Eyck Academy: www.janvaneyck.nl

l'éclairage à LED dans le collimateur
séminaire OSRAM

Mariline Thiebaut-Brodier
(Elektor)

En juin dernier, *OSRAM Opto Semiconductors* et ses partenaires (*Derelek*, *Gaggione* et *Texas Instruments*) organisaient à Paris leur 5e séminaire sur l'éclairage à LED au cours duquel les présentations étaient complétées par des ateliers. Ceux-ci favorisent les échanges entre experts.

Evolution et tendances du marché de la LED

Le rapport McKinsey 2011 sur le marché de l'éclairage montre que plus de la moitié des LED sont utilisées pour l'éclairage extérieur (7%) et l'éclairage architectural (46%). Les fabricants de LED prévoient une forte progression des LED (5% par an jusqu'en 2020) dans l'éclairage des boutiques et des bureaux ainsi que dans l'éclairage résidentiel. Dans un premier temps il s'agit de remplacer les lampes actuelles par des lampes à LED (c'est ce qu'on appelle le *retrofit*). Dans un deuxième temps, les nouveaux luminaires seront construits autour du composant (*LEDfit*). Alors que la course à l'efficacité lumineuse (lm/W) semble toucher à sa fin, les recherches actuelles portent sur la diminution des coûts de production grâce à :

- l'augmentation du diamètre des galettes (*wafer*) de production des puces (>6")
- l'augmentation du rendement de la production (sur une galette de 6", seuls 58% des puces sont utilisables)
- le remplacement du substrat en saphir par du silicium.

La standardisation est également un domaine en pleine effervescence, la Corée entre autres y est très active. Actuellement la norme LM80, publiée par l'*Illuminating Engineering Society of North America* (IESNA), est largement appliquée pour tester le maintien du flux lumineux des LED et calculer la durée de vie des lampes à LED. Cela ne suffit pas, c'est pourquoi de nouvelles normes sont en préparation pour qualifier la résistance à l'environnement (choc, eau…) des LED, la qualité de la lumière (indice de rendu des couleurs), la fiabilité à long terme… Sur les nouvelles générations de LED OSRAM, le flux lumineux et la couleur par exemple restent stables même lorsque la température ambiante s'élève (OSLON SSL, OSLON SQUARE).

Matières plastiques des systèmes optiques pour LED

La société *Gaggione* spécialisée dans l'injection plastique fabrique des boîtiers en plastique (marque *PlatiCase*) ainsi que des systèmes optiques (marque *LednLight*). Jean-Pierre Lauret nous a présenté les différents plastiques utilisés pour fabriquer des collimateurs ainsi que leurs avantages et inconvénients.

Le principe général de fabrication est le suivant : le plastique liquide est coulé dans un moule en métal. Il refroidit pour repasser à l'état solide. La peau extérieure se solidifie en premier. Ensuite c'est au tour du cœur mais comme le volume diminue (phénomène de retrait), cela provoque une déformation de la peau. Toute la difficulté est donc de limiter et de compenser la déformation (max. 25 µm) qui peut nuire à l'efficacité axiale (cd/lm) du collimateur.

Les deux matières plastiques transparentes les plus courantes pour les systèmes optiques sont :

- PMMA, polyméthacrylate de méthyle (*Plexiglas*®, *Altuglas*®) ; si on soumet le PMMA à une pression trop forte ou une température trop élevée lors du moulage, il devient marron.

- PC, PolyCarbonate ; très utilisé dans l'industrie automobile, aujourd'hui controversé parce qu'il contient du bisphénol A.

D'autres thermoplastiques sont disponibles mais réservés à des applications particulières :

- COP, Cyclo Oléfine Polymère (*Zeonor*®, *Zeonex*®) ;
- COC, Cyclo Oléfine Copolymère ; très fragile, très sensible aux UV et à la température (jaunissement), réservé à l'imagerie ;
- PS, polystyrène (c'est le plastique des boîtiers de CD) ; moins solide et moins résistant en température que le PC ;
- PEHD, polyéthylène haute densité ; utilisé dans les capteurs de mouvement IR passifs, transparence limitée (83% d'absorption sur 1 mm).

Bien entendu cette liste n'est pas exhaustive. Nous la fermerons avec le dernier arrivé : le silicone liquide ou LSR (*liquid silicone rubber*), très transparent (voir absorption volumique dans le tableau récapitulatif), résistant aux UV, aux températures élevées (jusqu'à 150 °C), aux agressions chimiques mais… mou !

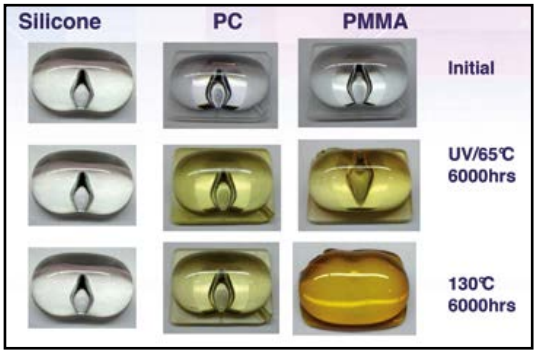


Fig. 1 : Comparaison du vieillissement de différents matériaux utilisés pour les systèmes optiques dans les lampes. Seul le silicone ne change pas de couleur. *(*Source *: Down Corning)*

Pour améliorer la diffusion de la lumière, il faut également dépolir la surface d'émission. Pour cela, on dispose de deux solutions simples :

- sablage/microbillage des moules
- technique de *Charmilles*® (électro-érosion des moules d'injection plastique).

Le numéro de *Charmilles* (Ch26, Ch31…) est proportionnel au logarithme de la rugosité ; plus la rugosité est élevée, meilleure est la diffusion.

Tableau comparatif de différentes matières thermoplastiques.

	PMMA	**PC**	**COC**	**LSR**
équivalent verre	*Crown glass*	*Flint glass*	*Crown glass*	pas d'équivalent
indice de réfraction, n	1,492	1,585	1,53	1,41
nombre d'Abbe (ou coeff. de dispersion), v	57	30	56	50
absorption volumique, a	0,0017 mm^{-1} 10% de pertes sur 62 mm	0,0083 mm^{-1} 10% de pertes sur 13 mm		0,000635 mm^{-1} 10% de pertes sur 166 mm
température en continu	90 °C	135 °C		150 °C
tenue aux UV	moyen -> utilisation à l'extérieur	mauvais -> utilisation à l'intérieur		bon
résistance au feu	HB s'enflamme et dégage une odeur d'ail	V0 ne s'enflamme pas mais se consume avec des gaz toxiques		HB à V1
résistance mécanique	rigide et cassant	souple et résistant	fragile	mou et élastique
autres caractéristiques		sensible au process de moulage, aux additifs anti-UV, aux additifs de tenue au feu	sensible aux UV et à la température ; très cher (10x plus que le PMMA)	

Fig 2 : À gauche, un collimateur avec une surface de type *Charmilles* ; à droite, un collimateur à microtexture. (Source : *Gaggione*)

Toutefois la technique de *Charmilles* présente des inconvénients (limites physiques, résultat aléatoire), c'est pourquoi elle est remplacée aujourd'hui par la microtexture.

La surface d'émission du collimateur est composée d'un réseau de sphères en rectangle, en hexagone…

Optique : taille et couleur d'un faisceau lumineux

Maintenant que nous avons choisi le matériau et le polissage du collimateur, intéressons-nous au faisceau. L'angle d'un faisceau lumineux dépend *grosso modo* de deux facteurs :

- Plus le faisceau collecté passe par un diamètre fin, plus il est divergent. L'angle du

Collimateur

Un collimateur est un dispositif optique qui produit un faisceau de rayons de lumière parallèles à partir d'une source de lumière. Il est constitué d'une lentille convergente achromatique et d'une fente éclairée, située dans le plan focal de cette lentille. (Le dispositif était connu dans les agrandisseurs photographiques sous le nom de condenseur.)
Dans le monde de l'éclairage à LED, un collimateur est un système hybride qui fonctionne à la fois en transmission et en réflexion totale interne, il est conçu pour récupérer un maximum de lumière de la source lumineuse.

Nombre d'Abbe

Un verre optique est caractérisé par son nombre d'Abbe (ou constringence) : $V = (N_D - 1) / (N_F - N_C)$.
Il permet de déterminer la dispersion, c'est-à-dire la variation de l'indice de réfraction avec la longueur d'onde.
N_D, N_F et N_C sont les valeurs de l'indice pour les radiations D (589,3 nm du sodium) F (486,1 nm de l'hydrogène) et C (656,3 nm de l'hydrogène). Plus la constringence est élevée, moins le verre présente de dispersion chromatique.
Il existe deux familles de verres :

- *flint* (silex en anglais), verre dont les constituants sont souvent le plomb, la silice et la potasse, avec V voisin de 35 et donc très dispersif ;
- *crown* (silicates alcalins) avec un V de l'ordre de 60, peu dispersif.

Absorption volumique

L'absorption volumique est la quantité de lumière absorbée par le matériau. Il s'agit d'une propriété intrinsèque au matériau, qui ne dépend que de l'épaisseur de matière traversée par la lumière. Sur le plan de la théorie, la quantité de lumière absorbée est définie par une exponentielle décroissante A(L)=A(0)*(1-exp(-L/a)) où « L » est l'épaisseur de matière traversée par la lumière et « a » l'absorption volumique, grandeur caractéristique du matériau, exprimée en mm^{-1}.

faisceau est donc inversement proportionnel au diamètre du collimateur.
Nous l'avons constaté lors des expériences de l'atelier d'optique :
OSLON SQUARE + optique de 32 mm de diamètre : surface éclairée plus petite mais par un faisceau de 970 lux au centre
OSLON SQUARE + optique de 45 mm de diamètre : surface éclairée par un faisceau de 1640 lux. Le faisceau est donc plus concentré dans ce cas.

- Plus l'aire du collecteur est grande, plus l'étendue de faisceau est grande. L'angle du faisceau est donc proportionnel à la surface émettrice de la LED.

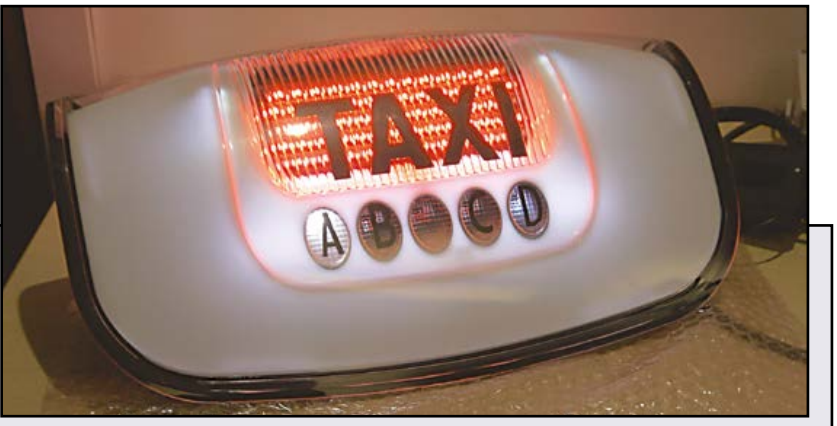
La taille n'est pas le seul critère de choix d'un collimateur. Il faut également que le modèle soit adapté à la LED pour éviter de faire ressortir les défauts de la LED (décomposition en taches jaune et bleu lorsque la surface de la LED n'est pas homogène par exemple).

Enfin le montage du collimateur est une opération qui influence aussi la qualité du faisceau : il faut que le système soit centré et aligné avec la source de lumière pour éviter une rupture de symétrie ou une perte d'intensité du faisceau.

Avec une LED de couleur (type RGBW), un simple collimateur ne suffit plus parce qu'on obtient une image des quatre puces (fig. 3).
C'est pourquoi dans les systèmes classiques, on intercale, entre la LED et une lentille, un barreau de mélange de couleur, en silicone (absorption volumique faible). Toutefois c'est une solution encombrante et avec un faible rendement (50%).
Gaggione a mis au point des collimateurs spéciaux pour les puces RGBW qui assurent plusieurs fonctions :

- homogénéisation (diffusion contrôlée)
- correction de l'image des puces (système confocal)
- compensation du décentrage.

Cela fonctionne si les puces de la LED sont équidistantes de l'axe optique et si les tolérances de fabrication sont maîtrisées. Là encore le positionnement du collimateur doit être très précis, un



Enseigne de toit pour taxi de la nouvelle génération ;
l'intensité lumineuse est passée de 1 cd à 7 cd. Éblouissant !

Enseigne de toit pour taxi, nouvelle génération

La société *Derelek* a présenté sa toute nouvelle enseigne de toit pour taxi, l'occasion de nous montrer combien l'intégration de LED peut être complexe. Les deux cartes (avant et arrière) regroupent plusieurs types de LED pour répondre à différentes exigences, comme par exemple les SIDELED pour un éclairage à 90° ou les OSLON SSL pour les couleurs des différents tarifs.
L'utilisation d'un circuit imprimé souple (*flex PCB* ou *circuit flex*) permet de ramener à la verticale les LED vertes et rouges. En effet, pour la certification par le LNE, les mesures de flux lumineux de l'enseigne sont effectuées à 30° sous l'horizontale. Si les LED étaient collées à la carte, elles seraient inclinées vers l'arrière et on ne les aurait plus vues lors des essais du LNE.
Pour obtenir un meilleur refroidissement par convection naturelle, les pilotes des LED sont montés en bas des cartes. Les découpes en haut et en bas des cartes créent des cheminées qui permettent à l'air de circuler et donc d'obtenir une bonne régulation thermique.
Chaque carte est recouverte d'une résine de tropicalisation noire, le boîtier terminé est parfaitement étanche. Cela permet de résister à différentes agressions : la température sur le toit d'une voiture atteint facilement plus de 50 °C ; le boîtier doit supporter un lavage avec un nettoyeur haute pression de type Kärcher… La couleur noire a une fonction bien particulière : elle absorbe la lumière du soleil et permet d'éviter l'effet « fantôme » au soleil. En effet, si la lumière du soleil est réfléchie avec la lumière verte ou rouge des LED, on ne distingue plus clairement l'enseigne. Avec cette résine noire, on a un bon contraste même en plein jour.

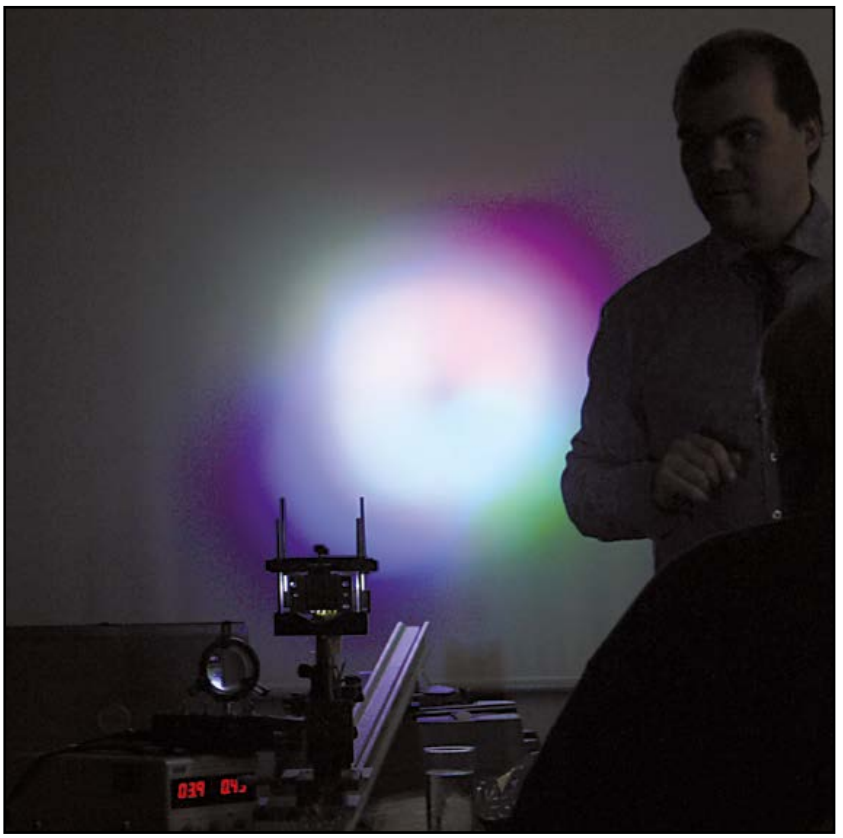


Fig 3 : Image des quatre puces d'une LED RGBW à travers un simple collimateur.

minuscule décentrage peut provoquer un changement de couleur.

Gaggione propose également un système de zoom, pour élargir le faisceau issu du collimateur. Le principe repose sur une lentille convergente suivie d'une lentille divergente ; quand la lentille divergente est éloignée de la convergente, le faisceau est plus large. L'éclairement qui en résulte est loin d'être homogène. *Gaggione* propose un zoom plus élaboré : chaque lentille est constituée d'un réseau de microlentilles (convergent/divergent). Chaque couple convergent/divergent voit localement un faisceau élémentaire uniforme (collimateur LLC49Z pour le spectacle/événementiel).

Enfin notre interlocuteur nous a mis en garde contre les contrefaçons venues entre autres de Chine et d'Italie. Même si à l'œil nu, les produits semblent identiques, leurs caractéristiques ne sont pas du tout équivalentes : l'intensité lumineuse des copies varie entre 13 et 22 cd/lm contre 38 cd/lm pour l'original.

Ces quelques lignes ne donnent qu'un aperçu de la richesse et de la variété des présentations et des ateliers de cette journée.

(130270)

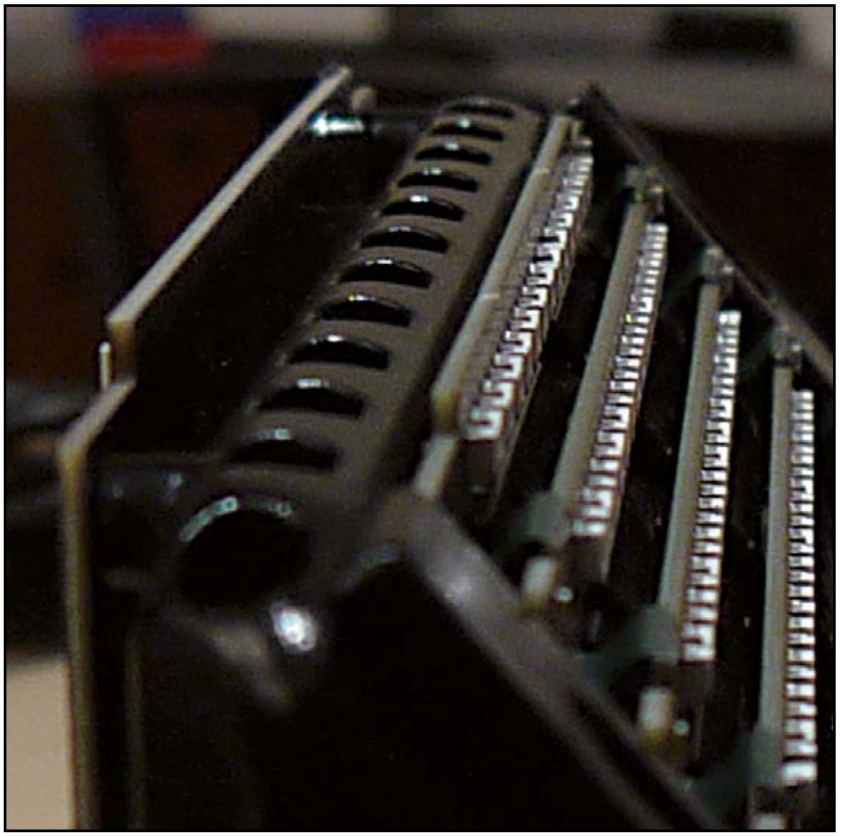


Fig 4 : Détail du haut de la carte. On y voit les trous de cheminée pour évacuer la chaleur ainsi que les LED ramenées à la verticale grâce au circuit imprimé souple.

photodétecteur avec Arduino

Une veilleuse qui s'allume quand un quidam approche, une LED IR, quelques résistances, un condensateur et un Arduino avec son logiciel : il n'en faut pas davantage pour photographier tout ce qui bouge.

Rolf Blijleven
(Pays-Bas)

Arduino, c'est comme Lego : chacun en fait ce qu'il veut. C'est amusant, instructif, utile. On s'en sert un bout de temps et puis, quand on veut, on le démonte pour en faire autre chose.
J'en avais fait une télécommande infrarouge pour mon Nikon D80. Pareil accessoire n'est pas tellement cher, mais le faire soi-même, c'est plus attrayant et Arduino ouvre d'autres perspectives. D'ailleurs, c'est facile comme tout : une LED IR, une résistance et un bout de logiciel trouvé sur l'internet, le tour est joué. Avantage appréciable d'Arduino sur d'autres plateformes embarquées, l'extraordinaire quantité de micrologiciels disponibles gratuitement sur la toile.

Puis, l'idée me vient de déclencher la télécommande par un détecteur de mouvement. On en trouve facilement dans les magasins et sur internet, mais ils travaillent généralement sur secteur. Or je ne veux pas de fil à la patte ! J'ai trouvé, pour 2,65 €, une veilleuse sur piles avec détecteur de mouvement (**fig. 1**).

Bidouillage de veilleuse

Première chose à faire, voir ce qu'il y a dedans. Pas trop mal ! Aucun CMS ni COB (*Chip On Board*), mais une puce à pattes, des résistances et des condensateurs normaux, un capteur PIR et une photodiode. Le circuit intégré porte l'im-

matriculation TL0001. Coup d'œil sur Google, la fiche de caractéristiques existe [1], en chinois. Copié-collé vers Google Traduction, résultat abominable, mais on devine. J'ai même mis la main sur la Note d'application d'un bidule pas vraiment identique à ma veilleuse, mais qui y ressemble. D'origine, la veilleuse fait trois choses indésirables pour mon projet : elle ne fonctionne que dans l'obscurité, donne une impulsion de quelques minutes alors qu'il ne m'en faut qu'une beaucoup plus courte et allume intensément trois LED quand on passe à proximité.
Le dernier inconvénient est simple à éliminer. Les trois LED partagent la même résistance talon. J'en ai supprimé deux et remplacé la résistance par une 2,2 kΩ (A dans la **fig. 2**). Ainsi la dernière LED s'allume lors d'un déclenchement, mais moins fort.

Figure 1.
Dans un supermarché ou un magasin discount près de chez vous, vous trouverez bien une veilleuse à détecteur de mouvement à prix cassé.

Ensuite, la fonction *inhibit* en présence de lumière du jour. Dans le schéma de la note (**fig. 3**) R3 est une LDR. Je n'en ai vu nulle part, mais plutôt une photodiode qui, elle aussi, présente une moindre résistance à mesure que la lumière augmente. Je l'ai remplacée par une 220 kΩ (B dans la fig. 2). Désormais, quand il y a du mouvement, la lampe s'allume même en pleine lumière.
Reste la durée de l'impulsion. La lampe s'allumait environ 5 s. Il doit y avoir une constante de temps RC ; mais quelle R et quel C ? J'ai dans l'idée que la lampe ne doit pas se rallumer immédiatement après l'extinction, ce qui implique une autre constante RC pour l'en empêcher, un *trigger inhibit*. Faut aller consulter les caractéristiques. Effectivement, c'est inscrit, mais en chinois. Même traduit par Google, ça reste du chinois :

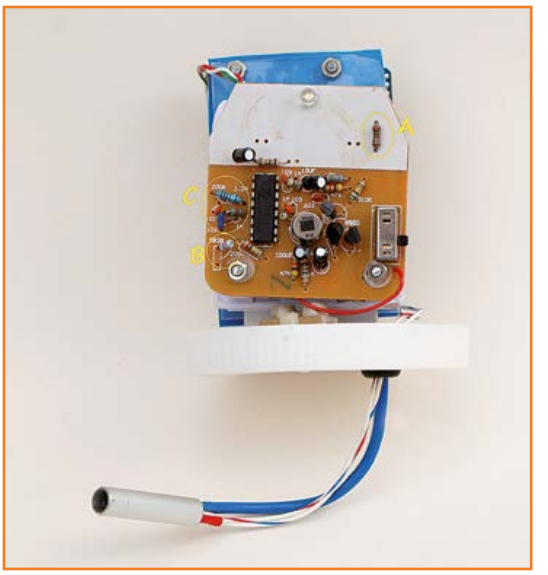


Figure 2.
La même lampe, avec quelques modifications.

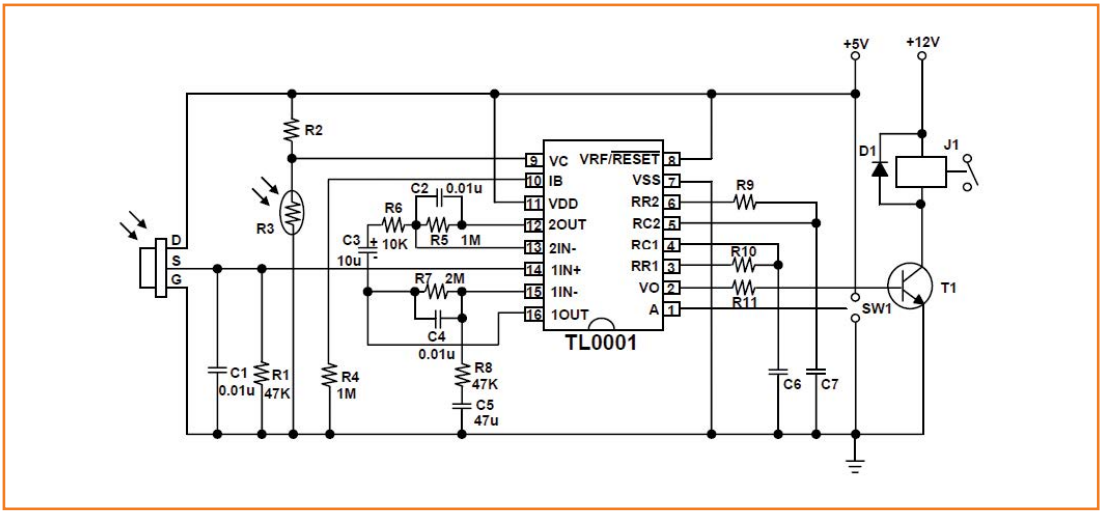


Figure 3.
Un exemple d'application tiré de la feuille de caractéristiques du TL0001 de la firme chinoise Treasure Link Technology. Il ressemble comme un frère à celui de la veilleuse, mais avec de légères variantes.

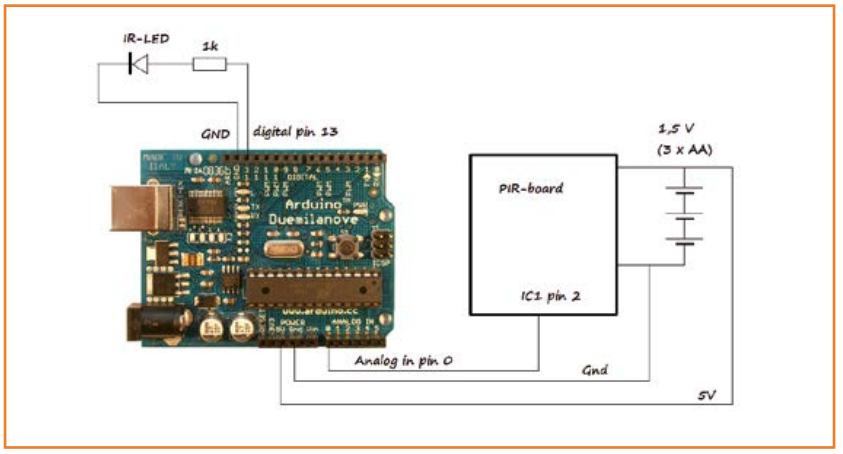


Figure 4.
Le circuit complet avec Arduino, LED IR et carte PIR, ex veilleuse.

« *Tx temps de retard de sortie à partir de l'extérieur pour s'adapter à la taille de la R9 et C7, d'une valeur de Tx ≈ 24576xR9C7; causés par blocage externe temps Ti. R10 et C6 de l'amplitude de la valeur de correction de Ti ≈ 24xR10C6*. »

Un peu d'ingénierie ***inverse*** à présent, qui consiste non pas concevoir un circuit imprimé à partir d'un schéma, mais à reconstituer un schéma à partir du circuit imprimé. Dans ces cas, pour pouvoir examiner simultanément les deux faces du circuit imprimé, faites une photocopie de la face cuivrée.
Après quelques recherches, j'ai identifié R9/C7 et R10/C6 que j'ai remplacés par les « bonnes » valeurs. Pensais-je. En fait, la petite veilleuse en avait perdu le nord. Je vérifie tout, puis une phrase sybilline de la feuille de caractéristiques chinoise attire mon attention : « *cela BISS0001 chip est entièrement compatible* ». En effet, les données du BISS0001 présentent les formules justes : Tx ≈ 24576xR10C6 et Ti ≈ 24xR9C7. Dans les caractéristiques, en revanche, R10/C6 et R9/C7 ont été intervertis ! Avec R10/C6 = 1 kΩ x 100 nF et R9/C7 = 270 kΩ x 1 nF, Tx ≈ 24 ms et Ti ≈ 0,5 s (C dans la figure 2). J'aurais pu laisser C7 en place.

Une tentative de stimuler l'amplification du capteur PIR n'a donné qu'un résultat mitigé. Le gain repose sur deux étages. Dans la puce, 1IN+, 1IN- et 1OUT appartiennent à un ampli op (cf. caractéristiques) dont le gain est voisin de R7/R8. Au départ, avec 2 MΩ et 47 kΩ, il était de 40. Avec 1 MΩ et 12 kΩ, il passe à 84. Cela n'augmente pas la sensibilité du PIR, mais permet d'amplifier de plus petits signaux. À l'usage, on remarque que les signaux plus grands verrouillent la sortie à la tension d'alimentation (*latch up*). Le second étage est aussi un ampli op dont le gain, par R6/R5, était de 100, devient égal à 67 avec 15 kΩ pour R6. Et plus aucun souci d'« écrêtage ».

Il en résulte que le capteur détecte les mouvements à l'intérieur à plus longue distance, mais ça ne veut rien dire. Les capteurs PIR sont sensibles aux *différences* de température. Un chat qui passe par un froid sibérien sera détecté de bien plus loin que lors d'une chaude journée d'été.

La sortie du circuit intégré est à la broche 2, soudée sur une piste de cuivre assez large pour y brancher facilement un fil. Deux autres fils pour le +5 V (après l'interrupteur !) et la masse transforment la veilleuse en carte à PIR. Pas mal pour un investissement de 2,65 € et un peu de recherche.
Vous aussi vous trouverez près de chez vous un modèle de capteur de proximité ou une veilleuse similaire. Mon récit vous aidera à y adapter le circuit.

Arduino, bête de somme

Il m'est apparu qu'une entrée analogique d'Arduino serait plus adéquate pour le PIR qu'une numérique. J'utilise une sortie numérique d'Arduino pour piloter la LED IR, avec résistance en série, pour commander l'appareil photo. Les trois piles AA de la veilleuse servent aussi à alimenter l'Arduino. La simplicité même (**fig. 4**).

La salve infrarouge pour déclencher l'appareil, j'en ai trouvé les détails sur [2] et [3]. Un mouvement à quelques mètres du capteur PIR produit un stimulus IR vers l'appareil qui peut également être distant de quelques mètres. La salve est capable de traverser une fenêtre, vous pouvez donc prendre la photo de l'intérieur alors que la télécommande est à l'extérieur. J'ai monté la LED IR sur un morceau de gros fil électrique, de manière à pouvoir le plier et orienter le capteur dans la bonne direction.

Mon appareil Nikon D80 a de drôles de lubies. Quand on le met en mode télécommandé par IR, il attend une commande. Si celle-ci tarde, il quitte le mode IR et ne répond plus. Pour moi qui fais de la photo de nature, c'est malcommode. On peut cependant régler sur l'appareil le temps d'attente jusqu'à 15 mn. Aussi ai-je inclus dans le

micrologiciel une commande « rester éveillé ! » si aucun mouvement n'a été détecté au cours des 14 minutes précédentes. On peut ainsi laisser l'appareil à longueur de journée en attente d'un animal rare.

Cet intervalle, vous pouvez le raccourcir ou l'allonger. Sans carte PIR, vous pouvez aussi faire de petits films en accéléré pour voir par exemple des fleurs croître et s'épanouir.

Reste à convertir cet intervalle en code avec perspicacité, ce qui demande un petit calcul. Utilisons Timer1, un temporisateur à 16 bits, qui compte de 0 jusqu'à 65 536. Il peut démarrer sur une valeur acquise, un *timerPreload* de 3 036, et donnera alors 65 536 - 3036 = 62 500 coups avant de produire une interruption. La carte Duemilenove tourne à 16 MHz, avec un pré-diviseur par 1 024, on est à 15 625 Hz, l'interruption aura lieu toutes les 62 500/15 625 = 4 secondes (compte non tenu de l'imprécision de l'horloge). Le code donne alors :

```
#define four_sec 1
#define twelve_sec 3 * four_sec
#define minute 5 * twelve_sec
#define quarter 14 * minute
```

C'est le quart de 14 mn, parce qu'à 15, l'appareil quitte le mode IR. Notre horloge, c'est *timeCounter*. Dans la routine de service d'interruption, nous donnons une valeur initiale *timerPreload* de 3 036 et nous incrémentons *timeCounter*. Le contenu du compteur fois quatre, c'est donc le temps écoulé en secondes.

```
ISR(TIMER1_OVF_vect) {
  TCNT1 = timerPreload;
  timeCounter +=1;
}
```

Dans la boucle principale, nous prenons donc une photo quand un déclenchement émane du capteur PIR ou si un quart d'heure s'est écoulé.

```
if (val > 200 || timeCounter == quarter
) {
      timeCounter = 0;
      takePicture();
      delay(500);
}
```

Le code source pour ce projet est sur le site d'Elektor [5]. Le fichier binaire tient dans 4 Ko. Dans les 32 Ko de mémoire flash d'un Arduino, il reste donc pas mal de place pour vos extensions personnelles. Un déclenchement sonore, par exemple, est également envisageable.

(130265 – version française : Robert Grignard)

Liens

[1] www.treaslink.com/Upload-Files/2010531152721141.pdf

[2] www.e-ele.net/DataSheet/BISS0001.pdf

[3] www.bigmike.it/ircontrol

[4] http://luckylarry.co.uk/arduino-projects/arduino-ir-remote

[5] www.elektor.fr/130265

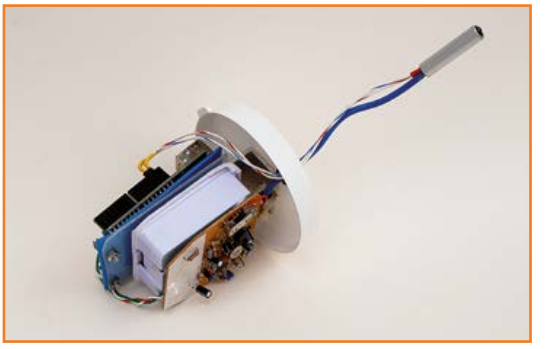


Figure 5.
Arduino et la carte à PIR avec le support de piles montés dos à dos sur un morceau de profilé en L.

Figure 6.
L'ensemble est ensuite logé dans un boîtier étanche aux projections d'eau.

ampli de puissance compact

le plein de puissance, distorsion en moins

Audiophiles, réjouissez-vous ! Tout droit sorti du labo d'Elektor, un nouvel amplificateur analogique de grande puissance et de la meilleure veine. Jugez plutôt : avec une seule paire de transistors de sortie pilotés par un circuit intégré spécial, il crache plus de 200 W sur 4 Ω avec un taux de distorsion harmonique exemplaire.

Ton Giesberts (Elektor)

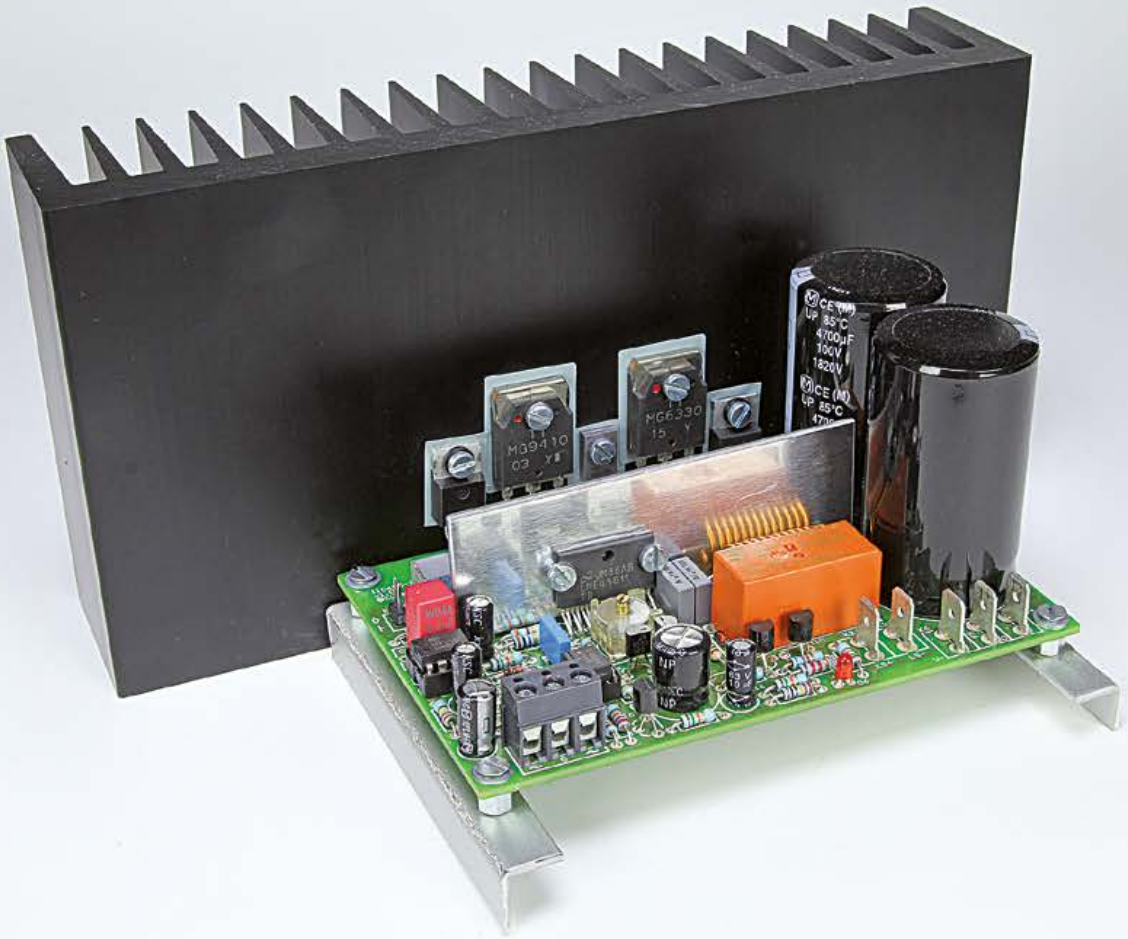
Les étages de puissance pour amplis audio, une longue tradition chez Elektor : on se souvient de projets légendaires, comme cet Edwin de mai 1970 ou le Titan en mai 1986, avec lesquels des milliers d'audiophiles ont grandi. Après une accalmie, plombée par les chaînes hifi bon marché des supermarchés, aux antipodes des normes d'excellence, l'engouement renaît. Vous êtes nombreux à désirer encore un amplificateur de valeur, créé de vos propres mains, qui vous offrira durablement une très haute fidélité sonore.

Notre choix s'est porté cette fois sur une structure semi-discrète, compacte, avec l'avantage

Résultats de mesures

(effectuées avec comme alimentation un transfo de 2 x 40 V/500 VA (Nuvotem 0500P1-2-040) et électrolytiques externes 4 x 10 000 µF/100 V)

• sensibilité d'entrée :	0,88 V (137 W/8 Ω, DHT+b = 0,1 %)
	0,91 V (145 W/8 Ω, DHT+b = 1 %)
• impédance d'entrée :	15 kΩ
• puissance de sortie permanente :	137 W sur 8 Ω (DHT+b = 0,1 %)
	145 W sur 8 Ω (DHT+b = 1 %)
	220 W sur 4 Ω (DHT+b = 0,1 %)
	233 W sur 4 Ω (DHT+b = 1 %)
• puissance pointe/musicale :	218 W sur 8 Ω (DHT+b = 10 %)
(alim. CC ±56,8 V)	175 W (8 Ω, DHT+b = 1 %)
	165 W (8 Ω, DHT+b = 0,1 %)
	395 W (4 Ω, DHT+b = 10 %)
	316 W (4 Ω, DHT+b = 1 %)
	299 W (4 Ω, DHT+b = 0,1 %)
• largeur de bande de puissance :	2,1 Hz à 125 kHz (50 W/8 Ω)
• vitesse de balayage :	26,7 V/µs
• temps de montée :	2,4 µs
• rapport signal/bruit :	> 94 dB (linéaire, B = 22 Hz à 22 kHz)
(référence 1 W/8 Ω)	> 97 dBA



• distorsion harmonique + bruit : (B = 80 kHz)	0,0033 % (1 kHz, 1 W/8 Ω)
	0,0006 % (1 kHz, 50 W/8 Ω)
	0,006 % (20 kHz, 50 W, 8 Ω)
	0,0047 % (1 kHz, 1 W/4 Ω)
	0,0009 % (1 kHz, 100 W/4 Ω)
	0,009 % (20 kHz, 100 W/4 Ω)
• distorsion d'intermodulation : (50 Hz : 7 kHz = 4 : 1)	0,002 % (1 W/8 Ω)
	0,0009 % (50 W/8 Ω)
	0,003 % (1 W/4 Ω)
	0,0026 % (100 W/4 Ω)
• distorsion d'IM dynamique : (ondes carrées 3,15 kHz + sinus 15 kHz)	0,0033 % (1 W/8 Ω)
	0,0022 % (50 W/8 Ω)
	0,0045 % (1 W/4 Ω)
	0,0027 % (100 W/4 Ω)
• facteur d'amortissement :	560 (1 kHz/8 Ω)
	311 (20 kHz/8 Ω)
• rendement : (alimentation CC)	70,6 % (8 Ω, DHT+b = 0,1 %)
	72,5 % (8 Ω, DHT+b = 1 %)
	68,5 % (4 Ω, DHT+b = 0,1 %)
	70,5 % (4 Ω, DHT+b = 1 %)
• protection CC :	+0,55 V/–0,86 V
• décalage CC en sortie :	0,2 mV (max. 0,6 mV)
• retard à l'allumage :	6 s

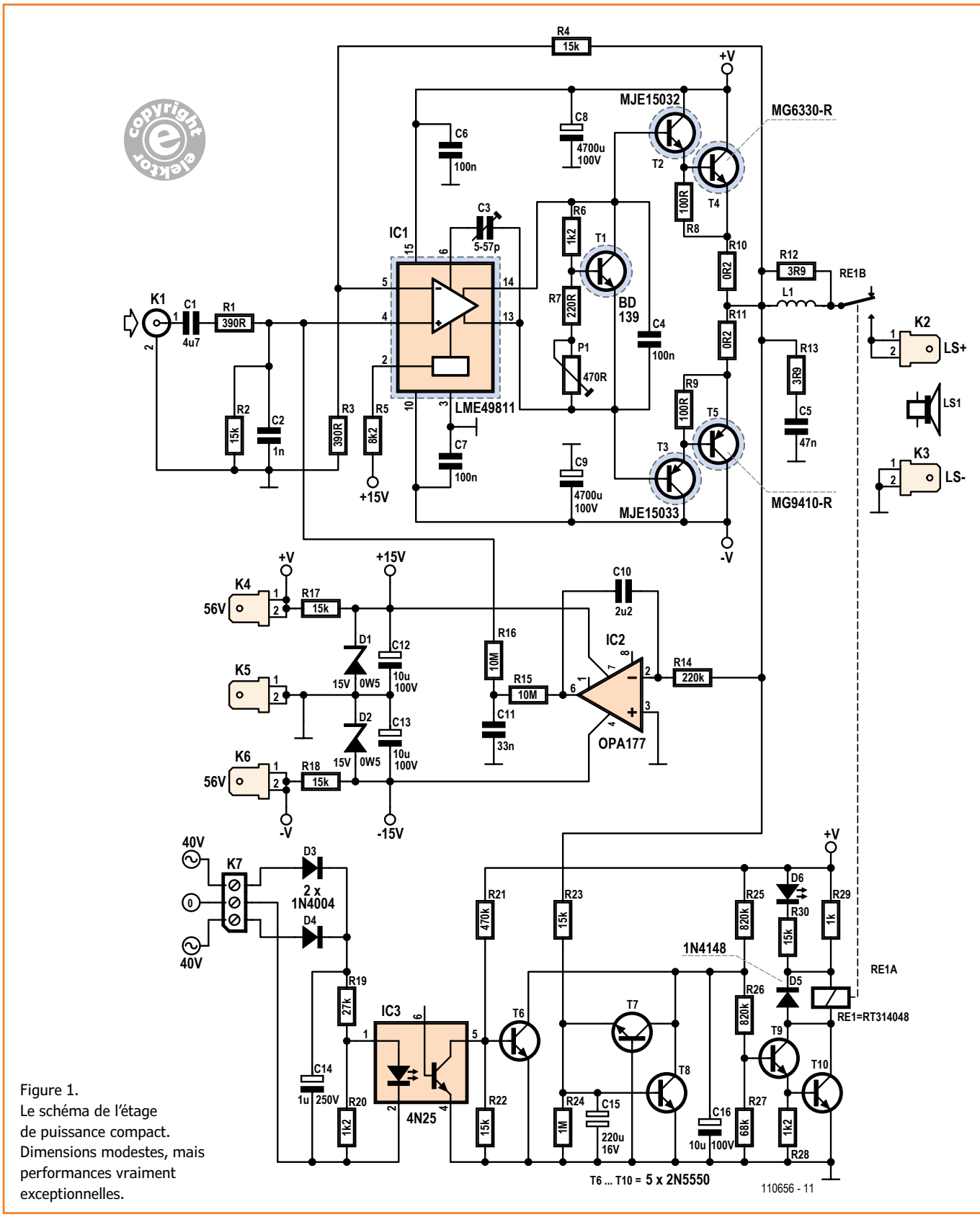


Figure 1.
Le schéma de l'étage de puissance compact. Dimensions modestes, mais performances vraiment exceptionnelles.

de la simplicité de construction. Restait à opérer le choix judicieux des autres composants pour atteindre des spécifications et une qualité sonore hors du commun.

L'historique

Tout a commencé avec un filtre de mesure pour la classe D publié dans le numéro double d'été 2011. Le Labo Elektor l'a conçu pour mesurer les hautes tensions de sortie, jusqu'à 70 V_{eff}, d'amplificateurs en classe D. Cependant, nous n'avons jamais réussi à tester le filtre sous de pareilles hautes tensions, par manque d'un étage de puissance approprié. Chez Elektor, un concepteur ne se laisse jamais abattre ; nous avons donc lancé un projet d'amplificateur haute tension, complètement discret, avec 23 transistors haute tension (MJE340, MJE350, MPSA42 et MPSA92), appelé à fonctionner sur une alimentation symétrique de ±110 V. Un circuit imprimé a été dessiné, un premier prototype construit, mais la complexité de l'entreprise a quelque peu refroidi l'ardeur initiale. Simplement pour tester un filtre, le jeu en valait-il la chandelle ?

Les spécifications du projet d'ampli sont ambitieuses. Il doit fournir un signal de sortie jusqu'à 20 kHz à 70 V_{eff} avec une distorsion extrêmement faible. L'impédance du filtre de mesure fait au moins 1 kΩ, il faut donc lui administrer un courant de sortie de 100 mA en pointe, plus si possible. En cherchant une autre solution plus simple, comme un circuit intégré capable de fournir une puissance suffisante sous une telle tension, nous tombons sur le LME49811 de *Texas Instruments,* au titre engageant : *Audio Power Amplifier Series High Fidelity 200 Volt Power Amplifier Input Stage with Shutdown*. Des spécifications excellentes, même si l'on ignore si les résultats de mesure annoncés sont obtenus avec ou sans l'étage de sortie externe. Cela vaut la peine de développer un amplificateur avec ce circuit intégré.

Les transistors ad hoc

Première démarche : sélectionner les transistors de puissance (T4/T5) pour l'étage de sortie. Dans un ampli audio, la priorité est accordée à une grande SOA, la zone de fonctionnement en toute sécurité. Il y a chez *Semelab* quelques exemplaires remarquables, le MG6330-R (NPN) et son complémentaire, le MG9410-R. Sous une tension collecteur-émetteur de 200 V, ils supportent un courant de collecteur de plus de 600 mA. C'est ce qui arrive quand l'amplificateur est sollicité à fond sans charge. On pressent la possibilité de les faire travailler en classe AB, avec de larges excursions en classe A pure. Leur gain en courant continu se maintient pratiquement linéaire jusqu'à quelques ampères, un peu moins pour le PNP. Prémices prometteuses de linéarité. Les mêmes exigences s'appliquent à l'étage qui précède, celui de T2 et T3. Les types choisis sont le MJE15032 (NPN) et le MJE15033 (PNP) qui tolèrent des tensions jusqu'à 250 V. Avec eux aussi, l'évolution du gain en courant continu est très linéaire. Les transistors intermédiaires comme ceux de sortie présentent de très hautes fréquences limites : 30 MHz pour les MJE, 60 MHz pour le MG6330-R et 35 MHz pour le MG9410-R. Le réglage du courant de repos est confié à un BD139 ordinaire.

La version audio

À la vue de ce projet, un de mes collègues rédacteurs me demande s'il ne pourrait pas servir de véritable ampli audio. Avec de telles qualités, il intéresserait bien plus de lecteurs qu'un amplificateur de mesure pour haute tension. Effectivement, il ne demanderait que quelques adaptations par rapport au projet d'origine : modifier la valeur de quelques composants et abaisser la tension d'alimentation (**fig. 1**). Sous une tension moindre, ±56 V obtenus avec un transfo de 2x40 V~ au secondaire, l'étage final délivre une puissance considérable, rien qu'avec une paire de transistors complémentaires : plus de 300 W musicaux sur 4 Ω.

Avec le LME49811 (IC1), l'étage final composé des 4 transistors T2 à T5 et un réglage du courant de repos par l'unique transistor T1, il ne manque plus que quelques composants.

Le réseau de contre-réaction R4/R3 est proportionné de façon à ce que, pour une excursion maximale de ±55 V avec une tension d'alimentation de ±60 V, la sensibilité d'entrée avoisine 1 V_{eff}. Ce niveau de ligne, tout préamplificateur moderne peut le délivrer facilement. Les valeurs des résistances sont choisies pour que la dissipation de R4 lors d'une excursion maximale reste sous 0,25 W. Pour garder optimale l'atténuation du mode commun à l'entrée du LME49811, on attribue aux résistances R1 et R2 les mêmes valeurs qu'à R3 et R4. Ce faisant, l'impédance d'entrée s'élève à 15 kΩ environ. La largeur de bande du signal d'entrée est limitée en bas par le

condensateur d'entrée C1 avec une fréquence de coupure théorique à 2,2 Hz, et en haut par C2. Il s'agit pour lui d'éviter tout souci avec des signaux d'entrée trop rapides, limités par la vitesse de balayage, et en plus de supprimer le bruit HF éventuel. Pour la compensation de fréquence du circuit intégré, le seul condensateur C3 est nécessaire. Aux fins d'expérimentation, on installe ici un ajustable au diélectrique en PTFE (idéal en audio) sur le circuit imprimé, lequel accepte aussi les condensateurs au mica métallisé au pas de 5,9 mm. Les tests ont montré que les résultats les meilleurs étaient obtenus au tiers de la course, soit à peu près 18 pF.

Le réglage de la polarisation de l'amplificateur, c'est le système d'asservissement centré sur IC2 qui le prend en charge. Il compare la tension de sortie à la référence de masse et applique, en fonction de la différence observée, un très petit courant de compensation à l'entrée non inverseuse du LME49811. Pourquoi cette entrée ? En raison de sa haute impédance, celle de l'entrée inverseuse vaut environ R3, qui n'est que de 390 Ω. La vitesse de réaction est de l'ordre de quelques dixièmes de seconde. L'amplificateur opérationnel qui la commande est un OPA177 connu pour ses excellentes caractéristiques en continu, avec moins de 2,8 nA et 60 µV comme courant de polarisation et tension de décalage. On en déduit que la tension de décalage maximale théorique de l'amplificateur final reste inférieure à 0,6 mV, aucun danger pour les haut-parleurs connectés. Sur notre prototype, elle ne dépasse pas 0,2 mV.
L'ampli op pour la correction en CC dispose de sa propre alimentation de ±15 V, dérivée de l'alimentation principale au moyen de quelques résistances et diodes zener : R17, R18, D1, D2. Avec une tension d'alimentation principale plus basse, il faudrait adapter R17 et R18, sans oublier le courant supplémentaire de 1,5 mA pris par la broche 2 de IC1 sur la ligne à +15 V.

À la sortie de l'amplificateur, on trouve un réseau Boucherot (R13 & C5) pour stabiliser l'amplificateur soumis à une charge inductive ou privé de charge. La bobine L1 fournit une protection supplémentaire avec une charge capacitive. Quant à R12, elle atténue de possibles oscillations ou des extra-courants. L'assemblage de R12 dans L1 fait gagner de la place sur le circuit imprimé. On remarque aussi sur le circuit imprimé deux gros condensateurs réservoirs de 4 700 µF. Ce sont des modèles à RSE (résistance série équivalente) basse. À charge du constructeur de l'ampli de prévoir à l'extérieur un transfo secteur, un redresseur en pont et 4 condensateurs électrolytiques de 10 000 µF/100 V.

Comme transformateur, j'ai choisi un type à deux enroulements secondaires de 40 V. Pour le pro-

Les lignes d'alimentation

Dans un étage de puissance, les pointes de courant sont fortes. Pour tamponner les lignes d'alimentation, il y a, en plus des condensateurs électrolytiques réservoirs externes, deux électrolytiques supplémentaires à faible RSE implantés directement au voisinage des transistors de sortie.
Sur un étage de puissance audio, il est essentiel que les lignes d'alimentation vers et sur le circuit imprimé ne produisent aucun champ magnétique parasite qui augmenterait la distorsion en induisant des courants dans la boucle de rétroaction ou d'autres parties de l'ampli. Une façon de réduire cet effet indésirable, c'est de garder aussi proches que possible les trois lignes d'alimentation et de les découpler à proximité immédiate de l'étage final. Sur le circuit imprimé, il y a des pistes parcourues par les courants d'alimentation sous forme d'alternances redressées en raison de la configuration même de la classe AB utilisée. Par le fait de garder les lignes d'alimentation positive et négative aussi voisines l'une de l'autre, le champ magnétique produit est presque sinusoïdal, ce qui réduit la distorsion. Sur un circuit imprimé à double face, il est astucieux de les superposer exactement.

Ces considérations sont cruciales pour la conception d'étages de puissance à très faible distorsion. La configuration en étoile de la masse est également fondamentale ; ici, elle est centrée près de C5 où convergent les lignes de masse de l'entrée, de la rétroaction, du circuit Boucherot, de la sortie haut-parleur et de l'alimentation. Cette carte est développée spécifiquement pour un amplificateur monophonique. Pour un ampli stéréo, il suffit d'en construire deux, rassemblés dans le même boîtier avec l'alimentation, de préférence deux alimentations séparées, d'ailleurs.

totype, on a utilisé au labo un transfo bon marché de 500 W dont la tension chute sévèrement sous forte charge. Avec un modèle plus stable, on atteindra de plus fortes puissances que ce que promettent les spécifications.

Sécurité

Comme tout appareil électronique — ce ne sont sûrement pas les circuits de puissance qui font exception — cet amplificateur peut connaître des anicroches. À pleine puissance, la température des transistors de sortie peut dépasser 70 °C, ce qui compromet leur longévité. Un transistor qui lâche forme souvent un court-circuit ; en l'absence de fusible, une forte tension règne alors à la sortie de l'ampli et ce sont nos (très) chers HP qui morflent. Il est donc hors de question de se passer de sécurité contre le continu dans un étage de puissance audio.

Après la mise sous tension de l'amplificateur, il lui faut quelques secondes pour stabiliser le réglage de la polarisation. Comme de coutume, le HP est relié à la sortie par un relais, activé une fois que l'alimentation de l'ampli est présente et qu'il n'y a plus de tension continue à sa sortie. Seule la tension d'alimentation positive est contrôlée dans ce cas, elle sert également de source pour le circuit de sécurité composé de T6 à T10. Sans elle, impossible d'exciter le relais. Le circuit de sécurité CC se compose de deux transistors et d'un filtre passe-bas, R23 et C15, d'une constante de temps de 3,3 s. Elle semble longue, mais plus la tension continue à la sortie est élevée, plus T7 et T8 entreront vite en conduction pour décharger C16. S'il y a une tension de décalage positive à la sortie de plus de 0,55 V, T8 conduit et entraîne la chute du relais via T9/T10. Avec un décalage négatif au-delà de –0,85 V, T7 réagit.

À côté de cela, on surveille aussi la présence de la tension au secondaire du transformateur pour faire chuter le relais lors du débranchement du transfo secteur ou si l'un des fusibles saute. Pour éviter une boucle de masse, la présence de la tension au secondaire est communiquée à T6 du circuit de protection par l'intermédiaire du photocoupleur IC3. D3 et D4 constituent avec C14 un redresseur à double alternance pour la LED du photocoupleur. Le rapport du diviseur R19/R20 est calculé pour que la LED s'éteigne directement lors de la disparition de l'une des tensions du transfo.

Avec les résistances R25 et R26, le condensateur C16 détermine le temps de commutation du relais après la mise sous tension, environ 6 s. Si vous avez des difficultés à vous procurer la version à 48 V prévue pour le relais, utilisez-en un de 24 V et prenez pour R29 une valeur de 2,2 kΩ/1 W.

Le refroidissement

Le circuit intégré autant que les transistors de sortie et d'étage pilote ont besoin d'être refroidis. Pour le CI, ce sera à l'aide d'une plaque d'aluminium de 2 mm d'épaisseur et 2,5 x 8 cm, vissée contre l'intégré. Elle permet d'évacuer les 2 W dissipés par le circuit sous une tension d'environ ±56 V.
Pour le radiateur de l'étage final, nous avons cherché le compromis entre dimensions et moyenne de puissance (estimée) demandée à l'ampli, car pour un usage permanent à pleine puissance, il faudrait un radiateur énorme ou une ventilation forcée. Qui a besoin de ça ?

Nous considérons donc que l'ampli ne délivrerait la pleine puissance que pendant de courtes périodes, disons quelques minutes, et trouvé un compromis intéressant avec un radiateur de *Fischer Elektronik*. Pas trop petit, veiller à éviter la surchauffe aux grandes puissances de sortie, donc une basse résistance thermique. Le modèle choisi a une hauteur de 10 cm et une résistance thermique de 0,7 K/W. Faisons le calcul. Avec une alimentation stabilisée de ±56,8 V, en restant sous la barre de 0,1 % de distorsion, l'ampli peut délivrer 300 W dans 4 Ω. Avec un rendement de 68,5 %, il y a 137 W à dissiper. En sinus permanent, à pleine puissance, la température grimperait à plus de 90 °C au-dessus de la température ambiante ! Les résistances d'émetteur aussi, R10 et R11, des modèles de 5 W, seraient au bord de la défaillance. Mais en réalité, le but est ici d'amplifier de la musique, c'est tout différent.

Saviez-vous qu'aucun fabricant d'amplis audio ne dimensionne ses radiateurs pour la pleine puissance annoncée ?

Liste des composants

Résistances :

5 %/0,25 W (sauf mention contraire)
R1, R3 = 390 Ω
R2, R4, R17, R18, R22, R23, R30 = 15 kΩ
R5 = 8,2 kΩ
R6, R20, R28 = 1,2 kΩ
R7 = 220 Ω
R8, R9 = 100 Ω
R10, R11 = 0,2 Ω, 1 %, 5 W, à faible induction (*Vishay Dale* LVR05R2000FE73)
R12, R13 = 3,9 Ω, 5 %, 5 W
R14 = 220 kΩ
R15, R16 = 10 MΩ
R19 = 27 kΩ
R21 = 470 kΩ
R24 = 1 MΩ
R25, R26 = 820 kΩ
R27 = 68 kΩ
R29 = 1 kΩ
P1 = 470 Ω aj. horiz.

Condensateurs :

C1 = 4,7 µF/63 V, MKT, au pas de 5/7,5 mm
C2 = 1 nF/400 V, MKT, au pas de 5/7,5 mm
C3 = ajust. 5 à 57 pF/250 V, horiz. (*Vishay BCcomponents* BFC280908003)
C4, C6, C7 = 100 nF/100 V, au pas de 5/7,5 mm
C5 = 47 nF/400 V, au pas de 5/7,5 mm
C8, C9 = 4700 µF/100 V, au pas de 10 mm, enfichable, Ø 30 mm (*Panasonic* ECOS2AP472DA)
C10 = 2,2 µF/63 V, au pas de 5/7,5 mm
C11 = 33 nF/63 V, au pas de 5/7,5 mm
C12, C13, C16 = 10 µF/100 V, au pas de 2,5 mm, Ø 6,3 mm
C14 = 1 µF/250 V, au pas de 2,5 mm, Ø 6,3 mm
C15 = 220 µF/16 V bipolaire, au pas de 5 mm, Ø 10 mm

Inductance :

L1 = 450 nH, 13 spires de fil de cuivre émaillé Ø 1,5 mm, Ø intérieur 7 mm

Semi-conducteurs :

D1, D2 = diode zener 15 V/0,5 W
D3, D4 = 1N4004
D5 = 1N4148
D6 = LED rouge, 3 mm
T1 = BD139
T2 = MJE15032
T3 = MJE15033
T4 = MG6330-R
T5 = MG9410-R
T6 à T10 = 2N5550
IC1 = LME49811TB/NOPB
IC2 = OPA177GPG4
IC3 = 4N25

Divers :

K1 = embase SIL à 2 picots, au pas de 2,54 mm
K2 à K6 = connecteur *Faston* encartable, au pas de 5,08 mm
K7 = domino à 3 contacts, au pas de 5 mm
RE1 = relais encartable, SPCO, 16 A, bobine 48 V/5,52 kΩ (TE *Connectivity/Schrack* RT314048)
plaquette d'isolation TO-220 pour T1 à T3, *Kapton* MT-film, 0,15 mm, 6 kV
plaquette d'isolation TO-3P pour T4, T5, *Kapton* MT-film, 0,15 mm, 6 kV
plaquette d'isolation TO-220 3 mm pour T2, T3
radiateur 0,7 K/W (p.ex. *Fischer* SK 47/100 SA)
radiateur pour IC1, dim. 2,5 x 8 cm, aluminium épais. 2 mm
circuit imprimé 110656-1, cf. www.elektor.fr/110656

Alimentation (pour 1 étage final) :

transfo secteur, sec. 2 x 40 V/500 VA (p.ex. *Nuvotem* 0500P1-2-040)
pont redresseur 200 V/35 A (p.ex. GBPC3502 (*Fairchild*)
4 x condensateur électrolytique 10 000 µF/100 V (2 pièces en // par ligne d'alimentation)

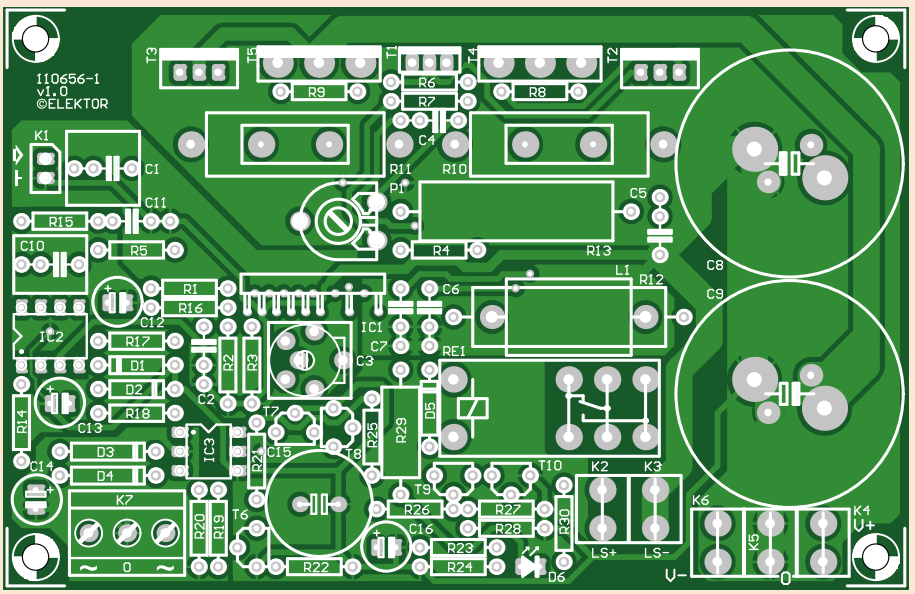


Figure 2.
Le circuit imprimé porte l'étage de puissance complet avec les condensateurs réservoirs et le circuit de protection et d'allumage.

Le circuit de protection est calculé pour une tension d'alimentation de ±56 V. Avec une tension plus basse, il faudra changer les valeurs de certaines résistances, ainsi que celles de contre-réaction pour garder la sensibilité d'entrée voisine de 1 V. Comptez que le LME49811 doit amplifier au moins 20 fois, soit 26 dB.

Construction

Comme promis, l'amplificateur est compact (**fig. 2**), l'implantation des composants est facile, mais certains points méritent l'attention. La plupart des composants se soudent directement sur le circuit imprimé, à l'exception de T1 à T5, IC1 et des condensateurs d'alimentation C8 et C9. Pour raccorder l'alimentation et les HP, des connecteurs plats (*Faston* 6,3x0,8 mm) sont soudés sur le circuit imprimé.

La bobine L1 compte 13 spires de fil de cuivre émaillé de 1,5 mm bobinées sur un foret de 7 mm. On lui laisse des terminaisons suffisamment longues pour pouvoir la monter à distance de la platine et pliées bien au milieu de la bobine pour qu'elles descendent verticalement. La résistance R12 se place à l'intérieur de L1 et on plie ses extrémités en concordance avec les trous de passage dans le circuit imprimé. Il faut les installer toutes deux en même temps pour que la bobine soit maintenue à quelque distance du circuit imprimé et que la résistance vienne bien dans l'axe de la self (**fig. 3**).

Avant d'aller plus loin, choisissez le modèle de boîtier pour voir comment implanter les radiateurs et la platine. Le plus simple est d'attacher deux supports au radiateur pour y monter le circuit imprimé, afin de ménager l'accès à la platine quand les transistors sont fixés au radiateur. Il faut monter le circuit imprimé contre le radiateur pour que les broches des transistors se trouvent le plus près possible de leurs pastilles respectives. Avec une pince, coudez les pattes de T1 à T5 en forme de S pour les faire coïncider avec l'écart des trous dans le circuit imprimé *sans leur imposer de contrainte mécanique permanente*. Le premier coude se pratique le plus près possible du boîtier. Ne jamais les tordre à ras, mais toujours intercaler une plaque métallique contre les pattes pour éviter de causer des microfissures dans le boîtier. La deuxième pliure est au niveau des trous dans la platine (**fig. 4b**). Vous pouvez installer provisoirement les plaquettes d'isolation entre les transistors et le radiateur pour localiser précisément la position du second coude, mais ce n'est utile qu'avec des isolants en céramique. Ne soudez les pattes des transistors au circuit imprimé qu'une fois que les transistors sont solidement fixés au radiateur avec le matériel d'isolation.

Figure 3.
Vue détaillée sur la self de sortie autour de la résistance de puissance R12.

En avant pour IC1 ! On commence par prendre une plaque de refroidissement en aluminium, 2 mm d'épaisseur et 2,5 x 8 cm, sur laquelle on vissera la puce. Il faut s'arranger pour que la puce reste à distance du circuit imprimé, de manière à ne pas venir en contact avec R1, R4 et R5. Important : la partie métallique à l'arrière de IC1 est reliée à la tension négative d'alimentation. En l'absence de matériau d'isolation, la

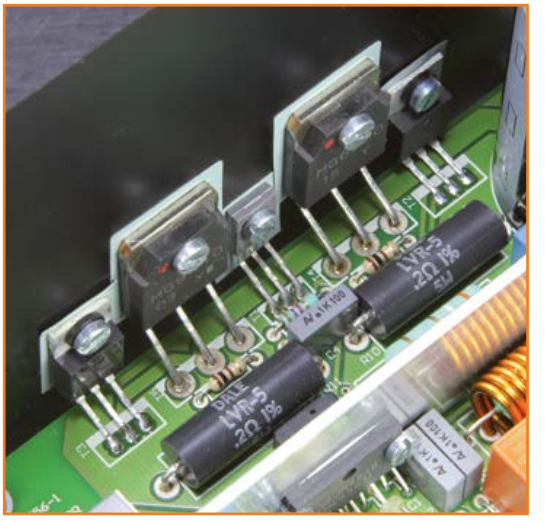


Figure 4.
Les pattes des transistors montés sur le radiateur sont coudées deux fois pour les adapter exactement aux trous dans la platine en évitant toute contrainte mécanique.

Figure 5.
Sur le circuit imprimé, il y a tout juste la place pour le radiateur de IC1.

pleine tension d'alimentation règne donc sur cette plaque ! Par précaution, il est recommandé d'utiliser une plaquette d'isolation. On soude ensuite la puce au circuit imprimé, sur lequel il y a tout juste la place (**fig. 5**). Au besoin, inclinez un peu L1 pour l'écarter du radiateur.
On termine par l'implantation des deux encombrants condensateurs C8 et C9.

Galop d'essai

Avant de mettre l'amplificateur directement sous tension, il faut régler le courant de repos de l'étage final. Pour cela, on intercale deux résistances de puissance de 47 Ω/5 W dans les lignes d'alimentation : si quelque chose va mal (un court-circuit…), les dommages seront limités. Dans le pire des cas, les résistances partiront en fumée. Vous pourriez aussi utiliser une alimentation stabilisée avec limitation de courant, mais pareils modèles sont rares pour des tensions de ±56 V. Insérez un ampèremètre dans la ligne positive d'alimentation.

Avant d'allumer, tournez P1 à fond vers la gauche et n'oubliez pas de câbler le bornier K7 avec les secondaires du transfo. Le courant dans la ligne d'alimentation positive après mise sous tension doit être de 30 mA environ, relais de sortie activé. Tournez lentement P1 vers la droite pour hausser le courant de 30 mA, donc 60 mA en tout. Ce faible courant de repos est très suffisant. Quand la température du radiateur croît, le courant de repos s'élève aussi. Il se stabilisera sous les 90 mA. À forte puissance de sortie, la température de jonction des transistors grimpe plus vite que celle du radiateur, et le transistor de courant de repos ne parvient pas à compenser totalement. On assiste alors à une hausse de quelques centaines de milliampères du courant de repos, mais lors de la baisse de température, il décroît vite aussi. C'est d'ailleurs une propriété intéressante de cet amplificateur : le réglage de la classe A de l'étage final s'élève avec la puissance de sortie fournie.

(110656 – version française : Robert Grignard)

Approfondissez vos connaissances et retrouvez cet ampli sur :

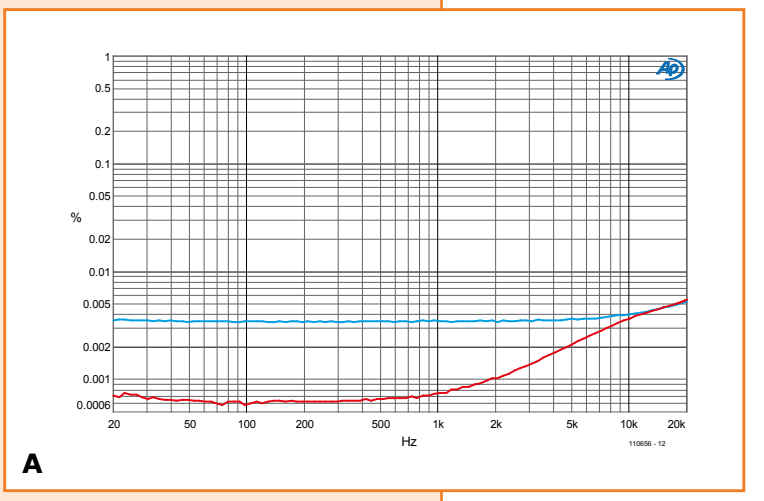
www.elektor-projects.com/project/110656-simple-audio-power-amplifier.13247.html

Quelques caractéristiques mesurées

Instrument de mesure : *Audio Precision System Two Cascade Plus 2722 Dual Domain*

Trace A

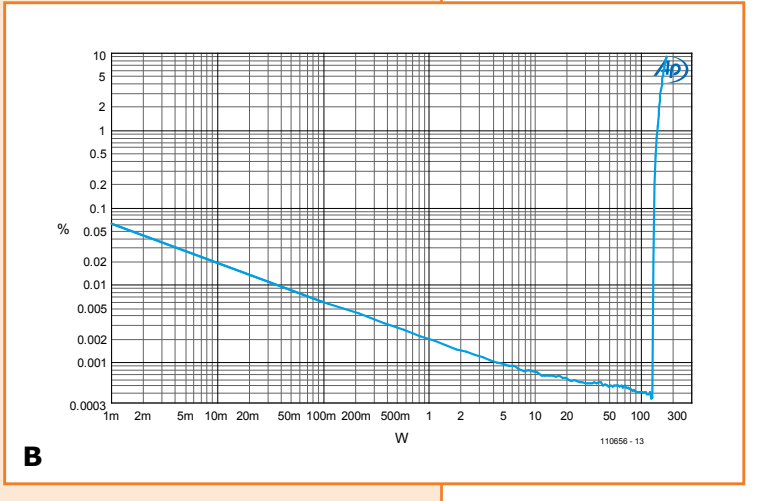
DHT+b pour une puissance de sortie de 1 W/8 Ω et 50 W/8 Ω, B = 80 kHz. La courbe pour 1 W contient essentiellement du bruit (DHT+b = 0,0034%). Ce n'est qu'au seuil de 20 kHz que la distorsion émerge à peine du bruit (DHT+b = 0,0052%). À 50 W (donc exactement à 20 V, si bien que ces résultats sont comparables aux données du cahier de caractéristiques du LME49811), le bruit de fond est nettement plus bas en comparaison de la tension de sortie. On voit que la distorsion augmente plutôt en haute fréquence. À 1 W, la distorsion est encore inférieure au bruit. Au-dessus de 10 kHz, elle est comparable à celle de la courbe à 1 W. La caractéristique à 100 W n'est pas représentée, elle est pratiquement identique à celle de 50 W. La distorsion à toutes les puissances de sortie reste très basse jusque sous le niveau de l'écrêtage.



A

Trace B

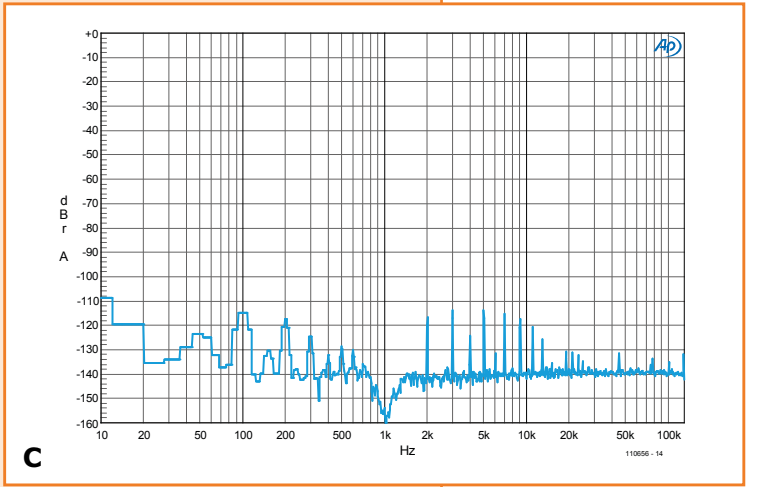
DHT+b comme fonction de la puissance de sortie (1 kHz/8 Ω, B = 22 kHz). La bande passante est réduite ici pour mieux mettre en lumière l'évolution de la distorsion. Ici aussi, on voit que la distorsion est extrêmement faible, alors que le bruit de fond diminue avec l'augmentation de la tension de sortie. À 127 W, le point d'écrêtage est atteint et au-delà, la distorsion s'accroît vite. À 137 W, DHT+b atteint une valeur de 0,1% (ce qui est encore utilisable dans une bonne reproduction sonore). En poussant la surcharge, on peut même sortir 174 W avec 10 % de DHT. On peut remarquer ici qu'avec le transfo secteur (bon marché) utilisé, la tension d'alimentation chute allègrement dès la saturation (avec 10% de DHT elle tombe à ±−51,5 V). Un transfo dont la tension de sortie est plus stable permet de tirer une plus grande puissance de sortie.



B

Trace C

Transformée de Fourier rapide (*FFT*) de 1 kHz à 50 W/8 Ω (20 V_{eff}). Les résidus de l'alimentation et les harmoniques de 1 kHz se situent à un niveau extrêmement bas, inaudible en pratique. Le troisième harmonique gît à -113,8 dB, ou 0,0002 % ! À cette puissance, DHT+b est à 0,0006 % (B = 80 kHz).



C

thermomètre USB

capturer facilement des données via USB

Michael Odenwald
(Allemagne)

L'électronicien s'est longtemps servi de l'interface série RS-232 comme d'une sorte de connexion universelle. Les connecteurs D-sub à neuf broches sont de plus en plus rares sur les ordinateurs, et pour relier nos circuits à un PC nous passons désormais par l'interface USB. Un pilote n'en reste pas moins nécessaire. Faut-il pour autant l'écrire pour chaque circuit ? Voici une méthode bien plus élégante.

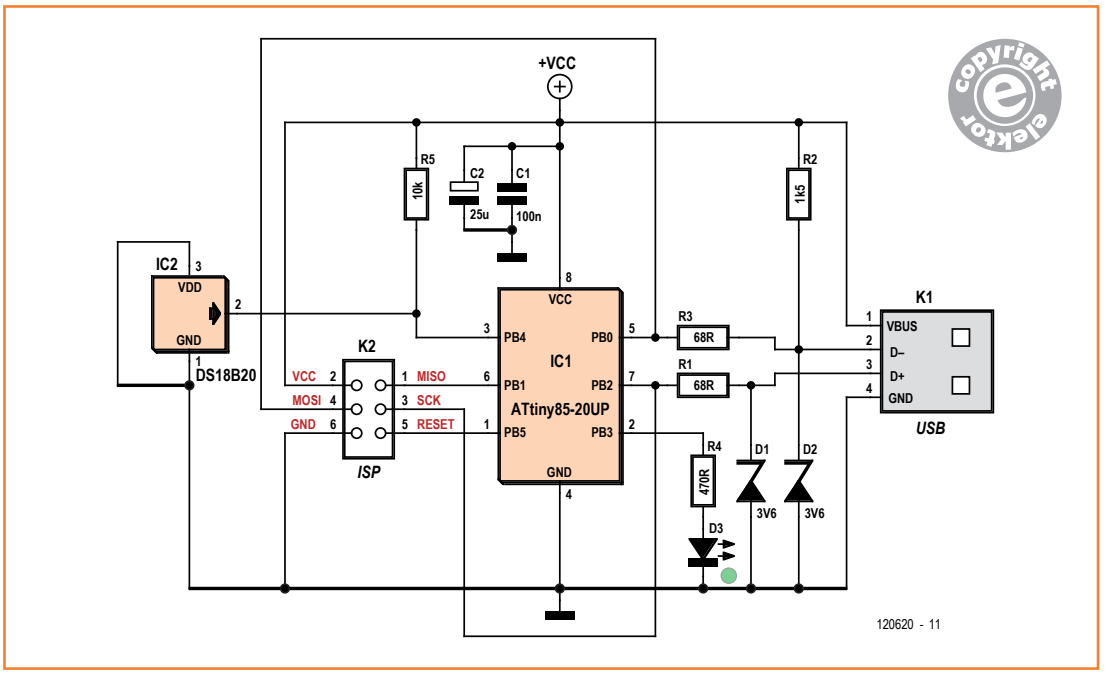


Figure 1.
Le schéma du thermomètre USB.

Quel que soit le système d'exploitation utilisé, l'écriture d'un pilote est loin d'être triviale. Certaines subtilités comme la signature numérique des pilotes compliquent par ailleurs la gestion de la chaîne « périphérique USB/pilote/système d'exploitation/programme utilisateur ». Pour les petits projets, on utilise donc en général le port COM virtuel. Pas de pilote à écrire, mais cette solution ne va pas sans autres inconvénients, notamment de configuration, et surtout ne tire pas parti de toutes les possibilités de l'USB.

Car il existe un mode USB qui fonctionne toujours et partout, qui prend en charge les prototypes tels que les petites séries, et qui de surcroît offre un grand confort : la classe de périphériques USB dite HID (***H**uman **I**nterface **D**evice*). Cette classe USB HID ne comprend pas que la souris et le clavier (voir **encadré**). D'autres périphériques, par exemple des actionneurs ou des capteurs, ont été prévus dans les spécifications de la norme USB. Autre aubaine, les systèmes d'exploitation actuels fournissent les pilotes USB HID. Les pilotes sont là, ne reste donc plus qu'à écrire l'application qui les exploitera !

Capture de données

L'USB HID se prête bien à la capture de grandeurs

externes sur PC. Illustrons cela avec la mesure de température. Les capacités d'un petit microcontrôleur de la famille ATtiny nous suffiront. Nous avons choisi un ATtiny85-20 (IC1 sur le circuit de la **figure 1**). Son rôle est triple : traiter la pile de protocole USB, gérer la communication avec le capteur, et acquérir et conditionner les données de ce capteur.
IC1 et le capteur de température IC2 sont alimentés sous 5 V par la prise USB K1. Le µC est cadencé à 16,5 MHz par une boucle à phase asservie interne ; le quartz n'est donc pas nécessaire, et la vitesse est suffisante pour l'USB.
IC2 est un capteur de température doté d'une interface à un fil, un DS18B20 de Dallas. Êtes-vous surpris de voir V_{DD} relié à GND ? Il s'agit du *parasite power mode* [2] : IC2 est alimenté par la ligne de données afin de maintenir l'échauffement propre aussi faible que possible. La liaison USB est établie via les résistances de protection R1 et R3 ; leur rôle est de limiter le courant de la ligne de données en cas de court-circuit. Les zeners D1 et D2 de 3,6 V confinent les niveaux des lignes de données dans l'intervalle des tensions conformes à USB.
La résistance R2 provoque l'énumération USB ; l'hôte USB (dans le PC) identifiera un *low speed device* doté d'un débit maximal de 1,5 Mbit/s.
Les condensateurs C1 et C2 lissent la tension d'alimentation de la prise USB. K2 est le support ISP à 6 contacts pour la programmation des contrôleurs AVR. La LED D3 indique qu'un cycle de mesure est en cours, d'une durée de 750 ms lorsque la résolution du capteur est maximale.

Micrologiciel

Le micrologiciel de notre thermomètre USB est écrit en C. Le code a été compilé avec WinAVR [3], puis transféré dans la mémoire Flash du contrôleur. La pile USB a été réalisée à l'aide du logiciel V-USB [4]. Les fonctions d'interrogation du capteur utilisent une bibliothèque écrite par Martin Thomas [5].
Le processus d'énumération USB débute après l'initialisation du matériel et de la pile logicielle USB. Le programme passe ensuite en mode d'exploitation interne, c'est-à-dire opère comme un automate fini dont les états sont : protocole USB, interrogation du capteur, et attente. Ces étapes sont exécutées l'une après l'autre avec une durée prédéfinie.
Le cycle complet dure 10 s. La valeur lisible par l'hôte n'est pas modifiée durant cet intervalle. Les nouvelles données n'arrivent donc après interrogation que toutes les 10 s. La mesure est commandée par le microcontrôleur et ne doit pas être initiée par le PC. Le capteur dispose ainsi de temps pour se refroidir.

L'élément le plus important du micrologiciel est le descripteur USB HID :

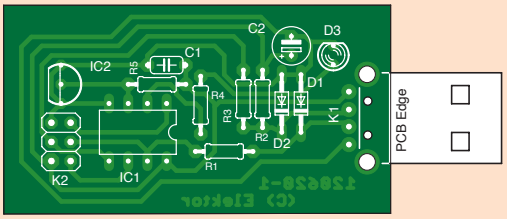
```
/*
 * The USB Hid report descriptor
 */
PROGMEM char usbHidReportDescriptor[33] =
{
    0x06, 0x00, 0xff,        // USAGE_PAGE (Generic Desktop)
    0x09, 0x01,              // USAGE (Vendor Usage 1)
    0xa1, 0x01,              // COLLECTION (Application)
    0x15, 0x00,              //   LOGICAL_MINIMUM (0)
    0x26, 0xff, 0x00,        //   LOGICAL_MAXIMUM (255)
    0x75, 0x08,              //   REPORT_SIZE (8)
    0x85, 0x0a,              //   REPORT_ID (10)
    0x95, 0x04,              //   REPORT_COUNT (4)
    0x09, 0x00,              //   USAGE (Undefined)
    0xb2, 0x02, 0x01,        //   FEATURE (Data,Var,Abs,Buf)
    0x85, 0x14,              //   REPORT_ID (20)
    0x95, 0x0a,              //   REPORT_COUNT (10)
    0x09, 0x00,              //   USAGE (Undefined)
    0xb2, 0x02, 0x01,        //   FEATURE (Data,Var,Abs,Buf)
    0xc0                     // END_COLLECTION
};
```

Classe de périphériques USB HID

La classe HID (***H**uman **I**nterface **D**evice*) est une extension de la norme USB. Elle fournit aux fabricants les informations nécessaires à la construction de périphériques d'entrée compatibles USB, et spécifie la façon dont un pilote HID peut extraire les données d'un matériel USB. Claviers et souris sont des représentants types de la classe HID, mais d'autres systèmes y sont définis ou prévus : capteurs, téléphones, lecteurs, articles promotionnels...
Propriété agréable des périphériques USB HID, leurs pilotes font déjà partie du système d'exploitation – au moins pour Windows, Linux et Mac OS X – et sont automatiquement chargés, sans intervention de l'utilisateur, lorsque le système détecte la connexion d'un nouveau périphérique USB HID.
Les périphériques HID ont aussi des inconvénients : leur débit n'est pas très élevé, et ils n'ont que peu de terminaison USB (*endpoints*), par lesquelles ne peuvent être transférées que peu de données.

Liste des composants

Résistances : (toutes de 0,25 W)
R1, R3 = 68 Ω
R3 = 1,5 kΩ
R4 = 470 Ω
R5 = 10 kΩ



Condensateurs :
C1 = 100 nF, céramique, pas de 5 mm
C2 = 25 µF/16 V, électrolytique, pas de 2,5 mm

Semi-conducteurs :
IC1 = ATtiny85-20PU, support 8 broches (programmé : 120620-41 [6])
IC2 = DS1820, boîtier TO92 à 3 contacts
D1, D2 = zener 3,6 V ; 0,5 W
D3 = LED, verte, 5 mm

Divers :
K1 = prise USB, type A, pour montage sur circuit
K2 = embase à 2x3 contacts, pas de 2,54 mm
circuit imprimé 120620-1 [6]

Figure 2.
Implantation des composants sur la carte.

Ce descripteur est composé de 33 octets. Il définit les identificateurs *Report* (10 et 20) que peut utiliser le programme utilisateur pour communiquer avec le thermomètre. Les *Reports* de type *Feature* sont constitués de blocs d'information de différentes longueurs (4 et 10 octets) et permettent à l'hôte et au périphérique d'échanger des données de configuration.
Le *Report_ID* 10 sert à demander la température et renvoie 4 octets. Le *Report_ID* 20 demande une chaîne d'identification au format date (*yyyy-mm-dd*) composée de 10 octets.

La classe centrale du thermomètre

```
namespace WindowsApp
{
  /// <summary>
  /// Implementation of the usbDevice with service methods
  /// based on the class usbGenericHidCommunication
  /// </summary>
  class usbDevice : usbGenericHidCommunication
  {
      private int tval;

      /// <summary>
      /// Class constructor - place any initialisation here
      /// </summary>
      /// <param name="vid"></param>
      /// <param name="pid"></param>
      public usbDevice(int vid, int pid)
          : base(vid, pid)
      {
      }

      /// <summary>
      /// USB HiD Temperatur Modul Methode GetTemeratur()
      /// </summary>
      public int GetTemperatur()
```

Du montage au pilote

Le circuit ne recourt qu'à des composants traversants. Son assemblage sur la carte de la **figure 2** devrait donc être un jeu d'enfant. Vous pouvez télécharger les fichiers de dessin depuis la page Elektor associée à cet article [6]. La **figure 3** montre le prototype équipé. Inutile de l'étalonner. Après avoir équipé et vérifié la carte, nous avons programmé le microcontrôleur via K2 à l'aide d'un programmateur ISP AVR adapté. Le micrologiciel peut être téléchargé gratuitement |6] sous forme de code source et de fichier hexadécimal directement programmable. Si votre ATtiny est vierge, pensez à désactiver le diviseur d'horloge « /8 » via le bit de configuration correspondant, et assurez-vous que la fréquence d'horloge interne est correcte. La plupart des outils de programmation devraient permettre d'entrer les bons paramètres : 0xE1 pour le *low fuse*, et 0xDD pour le *high fuse*.

Reliez le circuit au PC à l'aide d'un câble USB dès que le microcontrôleur a été programmé.

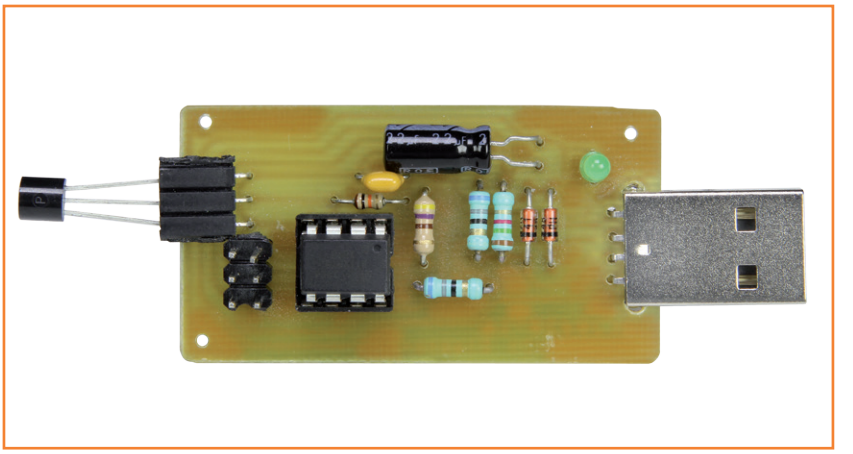


Figure 3. Le prototype du thermomètre USB.

Le système d'exploitation va détecter un nouveau périphérique HID et immédiatement installer le pilote HID. Oui, c'est aussi simple que ça ! Windows 32 ou 64 bits, Linux ou OS X, les pilotes HID sont toujours présents, possèdent tous une signature numérique, et sont aussitôt

```
      {
          // Declare a input buffer
          Byte[] inputBuffer = new Byte[5]; // we expect 5 byte; 1 x ReportID and 4 Byte temperature

          inputBuffer[0] = 10; // Read ReportID 10

          // Perform the Read Command
          bool success;
          success = getFeatureReport(inputBuffer);

          if (success == false)
          {
              Debug.WriteLine("Error during getFeatureReport");
              return tval;  // Error during USB HiD_GetFeature Request so return the old value
          }

          tval  = inputBuffer[1] << 24;
          tval |= inputBuffer[2] << 16;
          tval |= inputBuffer[3] << 8;
          tval |= inputBuffer[4];

          return tval; // Return the new value
        }
    }
}
```

installés, sans intervention de l'utilisateur. Vous le constaterez vous-même, le circuit est prêt à être utilisé quelques secondes à peine après avoir été connecté.

Programme

Le programme (Windows) qui interroge le thermomètre USB et affiche la température est écrit en C#. Il illustre bien la méthode à employer pour communiquer avec un périphérique HID. Le programme utilise les fonctions pour périphériques USB HID génériques de la bibliothèque écrite par Simon Inns [7], une bibliothèque qui encapsule les fonctions de l'API HID de Windows. Le code peut être compilé avec la version Express de Visual Studio 2010 [8]. Le rôle de la classe de

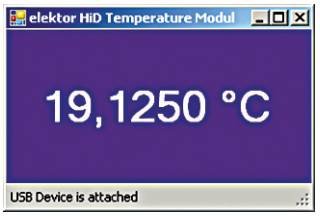


Figure 4.
Fenêtre d'affichage de la température sous Windows.

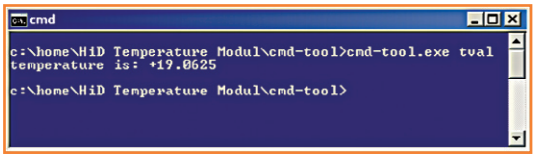


Figure 5.
Affichage de la température sur la ligne de commande.

base *usbGenericHidCommunication* est important, c'est d'elle qu'est dérivée une classe propre au périphérique HID. Les méthodes à exécuter sont ensuite implantées dans cette classe.

Le périphérique HID est identifié et appelé avec les paramètres *vid* = 0x0C7D et *pid* = 0x0011, respectivement les identificateurs fabricant et produit. La température s'obtient avec la méthode *GetTemperatur()*. La plage de mesure du capteur va de –55 °C à +125 °C. La température transmise est représentée par une variable de type *signed longint* qui peut prendre des valeurs allant de – 550000 à +1250000. Cette valeur est donc divisée par 10000, et au final la résolution est de 12 bits, ou 0,0625 °C. Ne la surestimez pas, la précision du capteur n'est que de 0,5 °C. La **figure 4** montre la fenêtre d'affichage de la température.

Le code source et la bibliothèque requise peuvent être téléchargés [6]. Nous avons choisi d'afficher la température dans une fenêtre graphique, mais la sortie peut aussi être dirigée vers une console à l'aide d'un court programme (**fig. 5**). Ici encore, vous pouvez télécharger [6] le code source et le code compilé de ce programme, ainsi qu'un exécutable pour Linux.

Bilan

Notre circuit et notre programme ont montré qu'il existe une méthode simple pour connecter un périphérique USB HID à un PC. Les modifier est facile, et si les données à transférer n'exigent pas une trop grande vitesse, vous disposerez d'un support matériel et logiciel pour relier sans tracas un périphérique à votre PC.

(120620 – version française : Hervé Moreau)

Liens

[1] www.usb.org/developers/hidpage/
[2] http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
[3] http://winavr.sourceforge.net/
[4] www.obdev.at/products/vusb/index.html
[5] www.siwawi.arubi.uni-kl.de/avr_projects/tempsensor/index.html
[6] www.elektor.fr/120620
[7] www.waitingforfriday.com/index.php/Open_Source_Framework_for_USB_Generic_HID_devices_based_on_the_PIC18F_and_Windows
[8] www.microsoft.com/visualstudio/fra/products/visual-studio-2010-express

90 ° ça craint

Thijs Beckers
(Elektor)

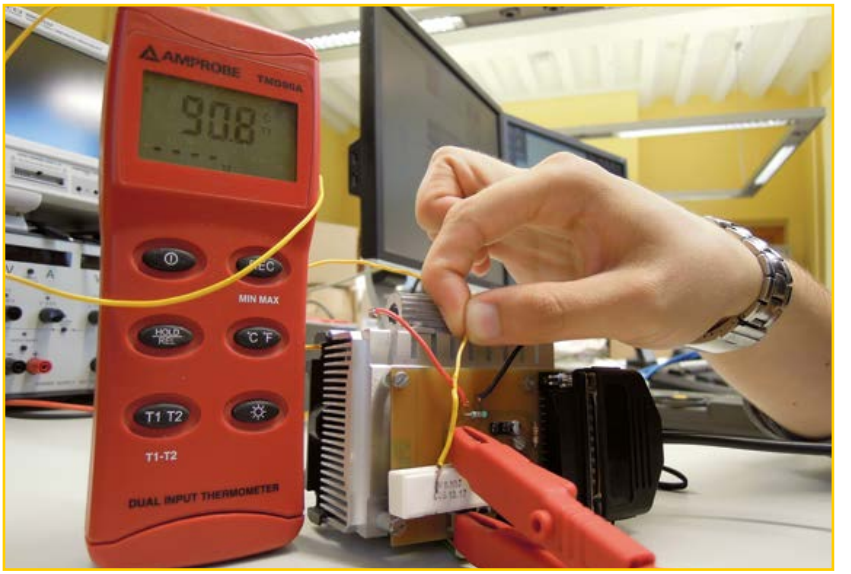
S'il a fait chaud, cet été, au labo d'Elektor, c'est la faute à la canicule, oui, mais aussi au prototype d'un circuit de test de batteries dont nous préparons la publication. Un beau jour, mon jeune collègue Tim a été surpris de mesurer plus de 90 º sur une résistance série de 7 W (le grand rectangle blanc, en bas, au milieu de la photo ci-dessous, contre lequel Tim tient la sonde), une température d'autant plus surprenante pour lui que P_{diss}, la dissipation escomptée et calculée selon la formule suivante, devait rester bien en deçà des 7 W de puissance nominale de la résistance :

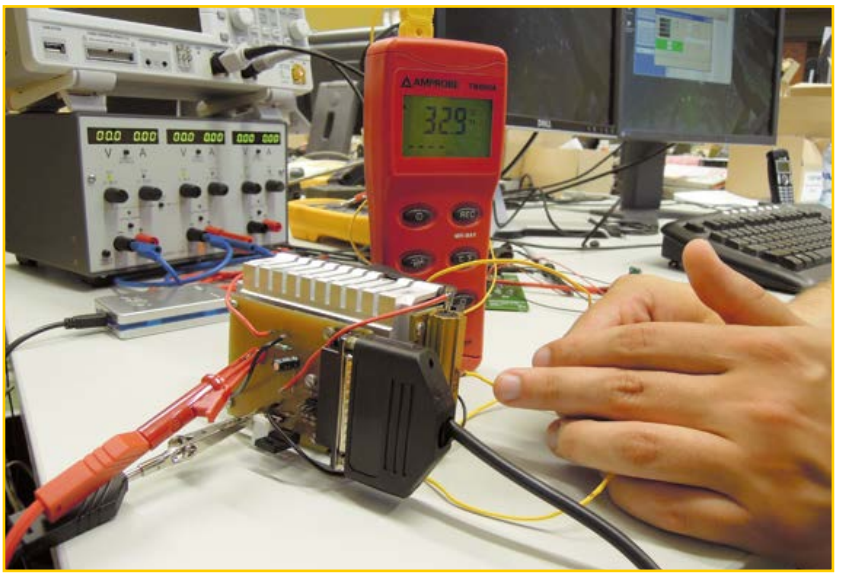
$$P_{diss} = I^2 \times R$$

Ce calcul donne pour limite supérieure une intensité de 8 A pour le courant à travers cette résistance, mais en pratique on n'a pas relevé plus de 2 A. Ainsi, avec I = 2 A et R = 0,1 Ω, soit à peine $2^2 \times 0{,}1 = 0{,}4$ W cette relativement grosse résistance en céramique a eu chaud bien rapidement. *Trop* chaud selon la règle de bon sens en usage dans notre labo qui considère comme critique tout composant dont la température dépasse 80 °C. Pour résoudre le problème, comme le montrent les illustrations, Tim a monté le circuit à même un radiateur, un modèle courant, avec ventilateur, pour les antiques unités centrales Intel P4. Grâce à sa ventilation forcée, ce type de radiateur ne présente qu'un faible indice de résistance thermique à l'air, de l'ordre de 0,40 K/W ; c'est le moyen par excellence pour obtenir une forte dissipation de chaleur. Il fallait bien ça pour les unités centrales de PC qui dissipent des 125 W et même plus. Sur la photo du haut, on voit qu'il reste de la place à côté de la résistance *shunt* déjà montée sur le radiateur.

Tim a remplacé la résistance enrobée de céramique par un modèle bobiné en boîtier alu, donné pour une dissipation de 50 W, puis il a monté les deux résistances sur les flancs du radiateur, là où le flux d'air est le plus fort (photo du bas). De cette manière, la résistance qui auparavant s'échauffait excessivement n'arrivait plus qu'à 33 °C dans des circonstances comparables. Avec l'intensité maximale (8 A, limités par le logiciel) l'échauffement de la résistance plafonne à 50 °C, conformément à notre règle des « 80 º max ». Il y a même de la marge. *En théorie*, la résistance de 50 W devrait passer des courants jusqu'à 22 A à des températures jusqu'à 250 °C (!), mais en pratique nous n'irons pas jusque là avec ce circuit-ci… que vous retrouverez dans une prochaine édition de ce magazine.

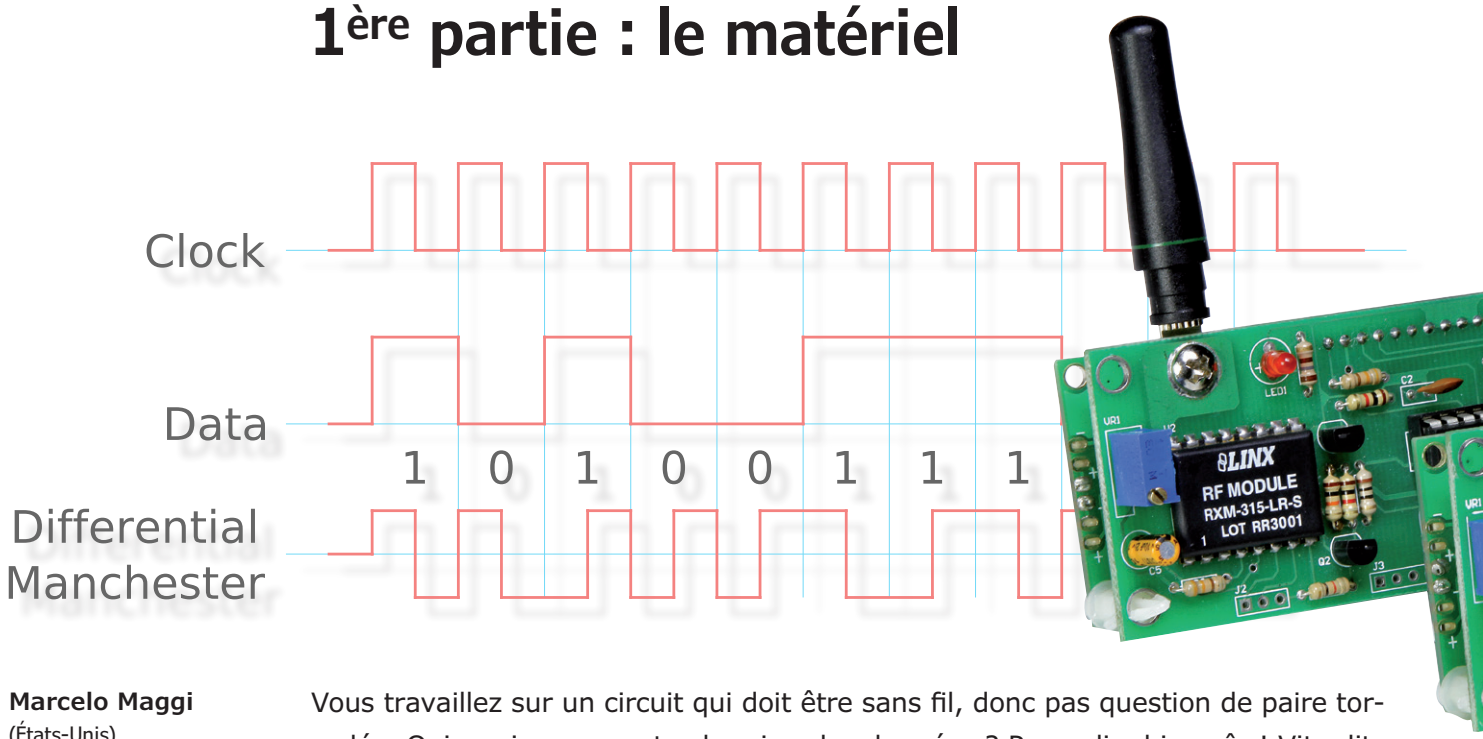
(130055)





liaison RF modulaire à code Manchester (1)

1ère partie : le matériel



Marcelo Maggi
(États-Unis)

Vous travaillez sur un circuit qui doit être sans fil, donc pas question de paire torsadée. Oui, mais comment acheminer les données ? Par radio, bien sûr ! Vite dit… et vite fait. Ne vous tracassez plus à concevoir vous-même une liaison RF fiable. Le travail a déjà été fait ! Suivez le guide : d'abord l'étude puis la soudure.

Voici une méthode pour envoyer des données par radiofréquences, à l'aide de composants bon marché, mais fiables, dans les bandes ISM à 315 ou 433 MHz avec une vitesse maximale de 5000 bps. Nous utiliserons un protocole très utile en RF, le code Manchester, pour atteindre une portée de 200 m et même plus.
Deux montages modulaires, l'un pour émettre, l'autre pour recevoir, ont été conçus et attendent d'être intégrés à votre application : il vous suffira d'adapter le code du microcontrôleur.

Émetteur, récepteur et antennes

La création d'un lien RF nécessite du matériel et du logiciel (ou micrologiciel), côté émission et réception. Ce premier volet de l'article décrira le matériel.

Concevoir un circuit RF avec des composants discrets n'est pas facile, et les résultats sont souvent (très) décevants. Par chance, *Linx Technologies* a fait le plus dur en offrant des modules RF complets en boîtier hybride. Ils sont disponibles pour une large gamme de fréquences, mais nous nous concentrerons sur les principales bandes accessibles là où vous vivez : 315, 418 et 433 MHz.
Jetez un œil aux extraits des feuilles de caractéristiques [1] de l'émetteur (**fig. 1**) et du récepteur (**fig. 2**). Peu de pattes sont actives. Le récepteur possède plus de broches, mais la plupart sont non connectées (NC). Pour le fonctionnement de base, il ne vous faudra, en plus de l'inévitable alimentation régulée, qu'une antenne.
Il est bon de rappeler que les antennes constituent un maillon très important, bien que trop

Transmission à 5000 bps sur plus de 200 m

souvent négligé, d'une liaison RF de bonne qualité. La division antennes de *Linx Technologies* fournit une solution appelée *Antenna Factor*. Nous utiliserons l'antenne monopole ¼ d'onde à hauteur réduite, dont vous trouverez en **figure 3** un extrait de la feuille de caractéristiques de la version 315 MHz.

La fréquence pour laquelle ces antennes sont optimisées est indiquée par une bande colorée sur leur corps : verte pour 315 MHz, bleue ou rouge pour 418 MHz ou 433 MHz.

Conception de l'émetteur (TX)

Le module émetteur transmet, à quelques restrictions près, bien entendu, le signal qui arrive sur sa broche DATA. Il ne contient aucune intelligence, c.-à-d. ni synchronisation ni protection des données. À l'utilisateur de s'en charger.

Le schéma de l'émetteur (**fig. 4**) est plutôt simple ; il n'y a qu'une seule connexion entre les deux composants principaux, le microcontrôleur et le module RF : la broche B0 (RB0/INT, broche 6) du microcontrôleur est reliée à la broche DATA du module RF. Le reste des composants est nécessaire pour assurer le bon fonctionnement, mais ils ne jouent aucun rôle actif dans la transmission des données. C1, C2, C3, C7 et C8, par exemple, sont des condensateurs de découplage en céramique de 0,1 µF. R5 sert à prévenir les remises à zéro intempestives du microcontrôleur (la broche 4 sert à la remise à zéro). L'oscillateur est asservi à 20 MHz par un quartz standard.

Le module émetteur fonctionne sous 3 V, c'est pourquoi la tension de notre alimentation 5 V (externe) passe par un régulateur fixe de type LP2950-30LPR qui la réduit à 3,0 V pour le module. C1, C5, C6 et R4 augmentent la stabilité, comme le recommande la feuille de caractéristiques du module.

Il y a deux résistances supplémentaires reliées au module. R3 relie la broche PD à l'alimentation en 3 V, ce qui la maintient au niveau haut. Au niveau bas, cette broche active le mode à basse consommation du module : il ne peut alors plus émettre. R1 est un simple morceau de fil (0 Ω). Il est possible de le remplacer par une résistance pour limiter la puissance d'émission, par exemple pour vous conformer à la réglementation des bandes ISM en vigueur dans votre pays. Avec 0 Ω, la puissance maximale est sélectionnée ; plus vous augmentez R1 et plus la puissance baissera (**fig. 5**).

Comme l'émetteur fonctionne sous 3 V, il fau-

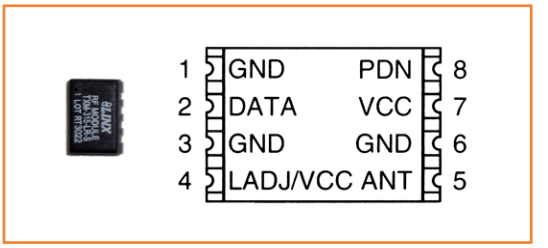


Figure 1.
Aperçu et brochage du module émetteur *Linx*, extraits de la feuille de caractéristiques. Il n'y a que peu de connexions à faire pour utiliser ce module.

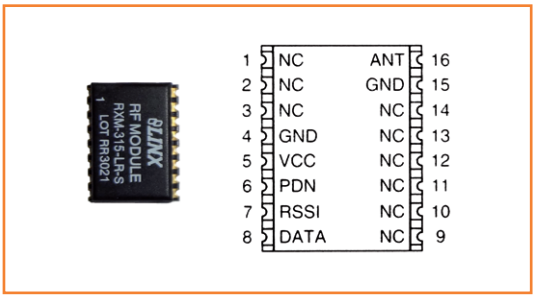


Figure 2.
Aperçu et brochage du module récepteur *Linx*. Beaucoup de broches ne sont pas reliées.

Figure 3.
L'antenne est un quart d'onde d'*Antenna Factor*.

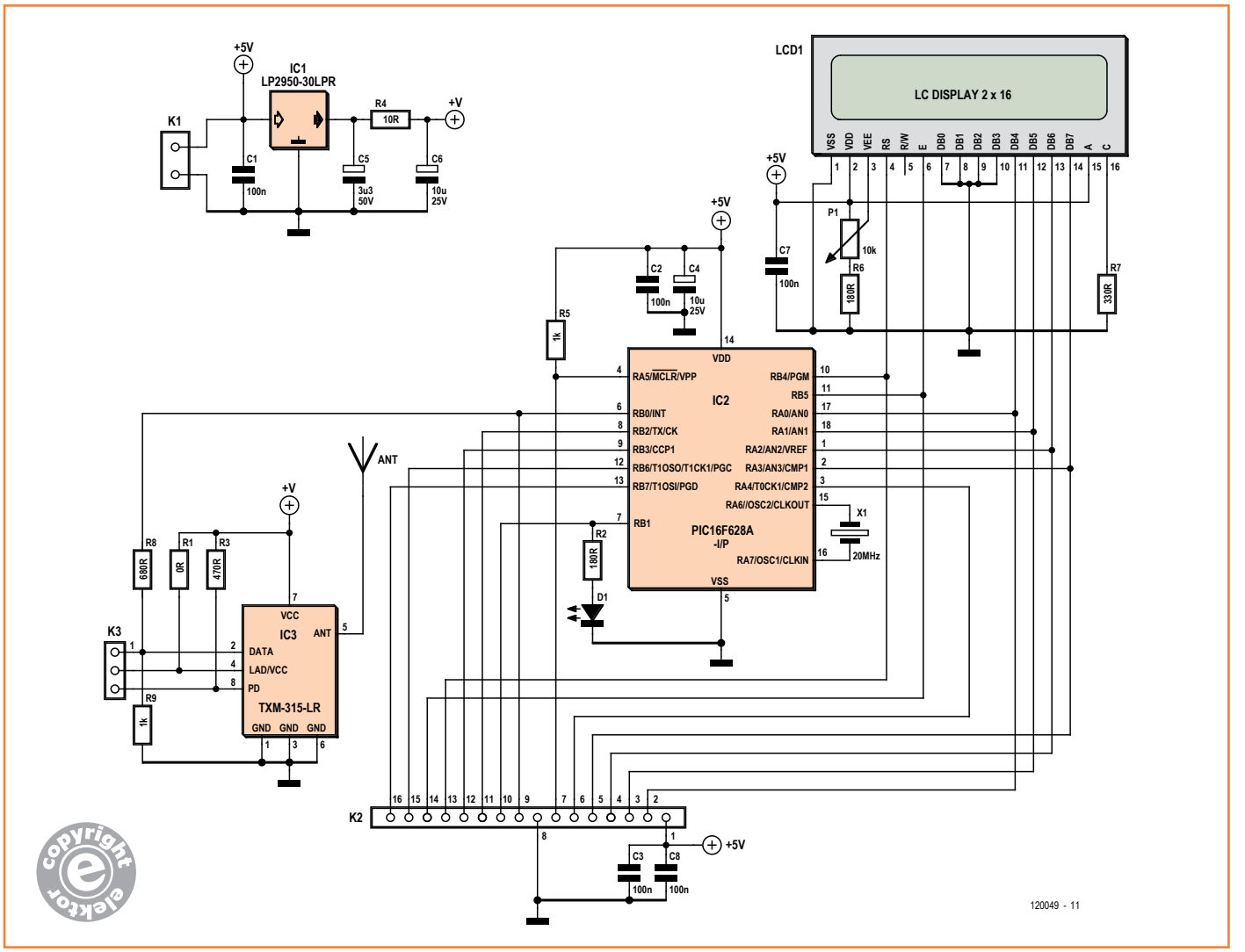


Figure 4.
Schéma de l'émetteur ; un PIC commande le LCD et la liaison de données.

dra veiller à ce que la tension présentée sur la patte d'entrée de données ne dépasse pas cette valeur ; c'est le rôle des résistances R8 et R9 qui forment un diviseur de tension.

Nous avons eu l'embarras du choix du µC ; le PIC Microchip retenu appartient à la sous-famille des PIC à 18 pattes. Le PIC16F628A est la version dotée de 2 kmot de mémoire de programme, mais vous pourriez tout aussi bien utiliser le PIC16F648A qui en possède 4 ainsi que plus de SRAM et d'EEPROM. Même le bon vieux PIC16F84A peut être utilisé, ce qu'apprécieront ceux qui le connaissent bien.

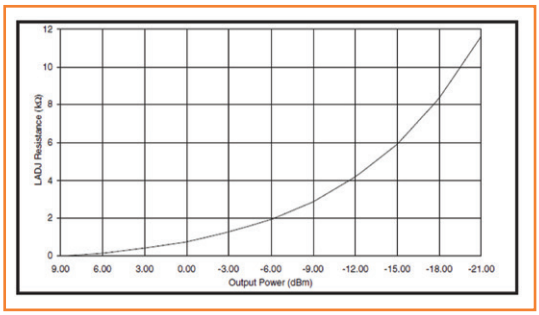


Figure 5.
La puissance de sortie peut être réduite à l'aide de R1.

Le schéma montre que l'accès à la plupart des broches importantes du micro et du module RF passe par K2 et K3 ; vous pourrez donc à peu près tout faire avec ces connecteurs : remettre à zéro le micro, injecter des données externes dans l'émetteur, etc. K3 permet même d'utiliser un micro complètement différent, p. ex. un modèle Atmel. Il vous suffirait de retirer le PIC de son support et commander le module RF directement depuis K3 (avec des signaux 3 V !). N'oubliez pas

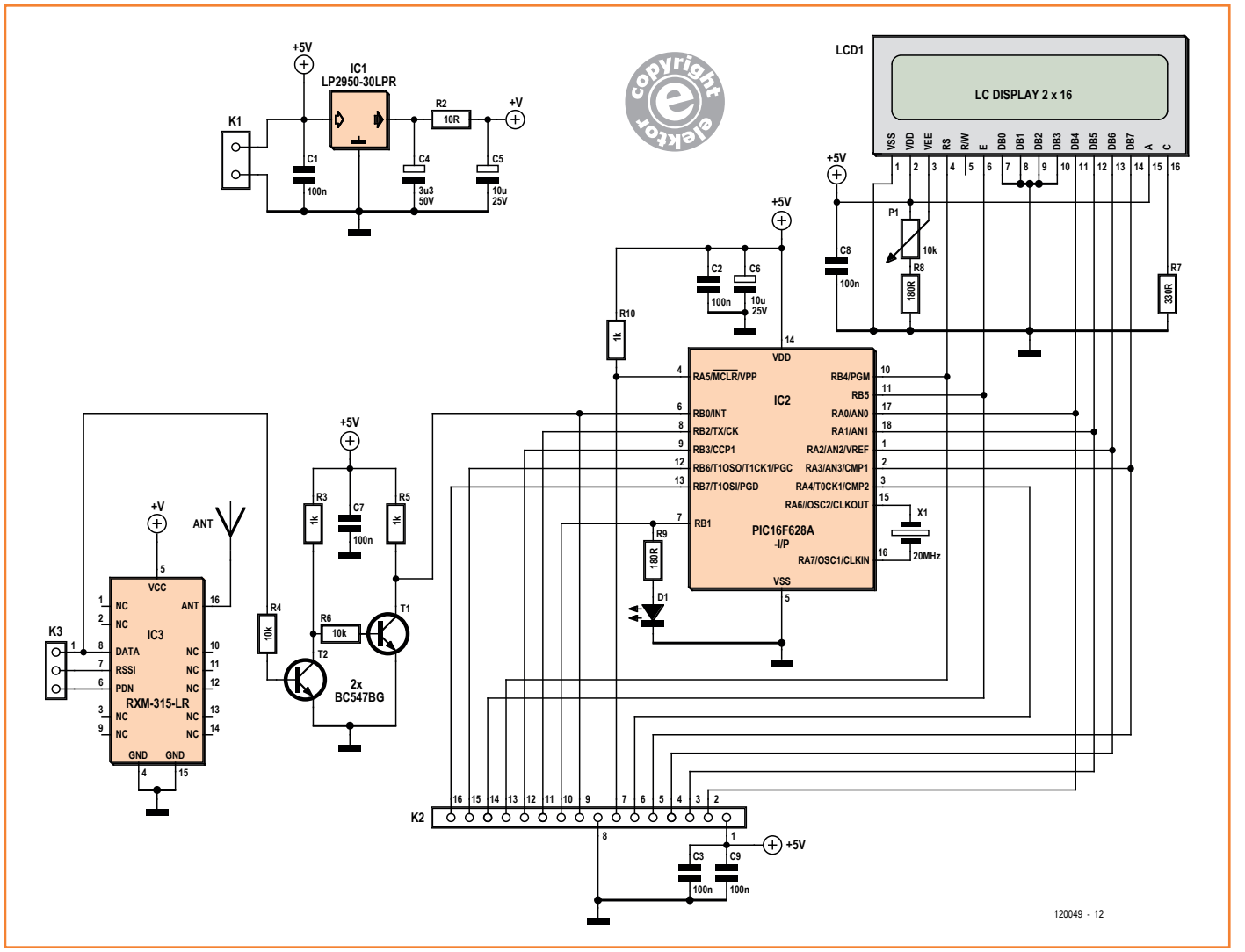


Figure 6.
Le schéma du récepteur ressemble en grande partie à celui de l'émetteur.

de fournir l'alimentation 5 V nécessaire et vous serez fin prêt pour des transmissions fiables.
Une dernière remarque : quiconque possède quelques rudiments de radio sait qu'il est vain d'espérer faire fonctionner un montage RF travaillant au-delà de 10 MHz sur une plaque d'essais. Le fonctionnement serait, au mieux, erratique. Par contre, ce montage, une fois sur son circuit imprimé, fonctionnera très bien associé à une plaque d'essais : toute la partie RF est gérée par le module émetteur.

Conception du récepteur (RX)

Plusieurs aspects du schéma (**fig. 6**) devraient vous sembler familiers, notamment les condensateurs de découplage (C2, C3, C7, C8 et C9), l'alimentation 3 V similaire à celle de l'émetteur (le récepteur fonctionne aussi sous 3 V), un condensateur de suppression de bruit (C6) relié aux pattes d'alimentation du micro, R10 qui empêche les remises à zéro intempestives ainsi qu'un quartz oscillant à 20 MHz. Lorsque je parlerai du micrologiciel dans le prochain volet, j'expliquerai l'importance de faire fonctionner les micros de l'émetteur et du récepteur à la même fréquence. Les connecteurs vers le monde extérieur possèdent une disposition similaire. La seule différence par rapport à l'émetteur est la présence sur le module de réception d'une sortie RSSI (received signal strength indicator) image de la puissance du signal reçu. Le signal RSSI analogique (utile pour ajouter un silencieux automatique, *squelch* pour les radiophiles) est routé sur la broche du milieu de K3 à la place de la

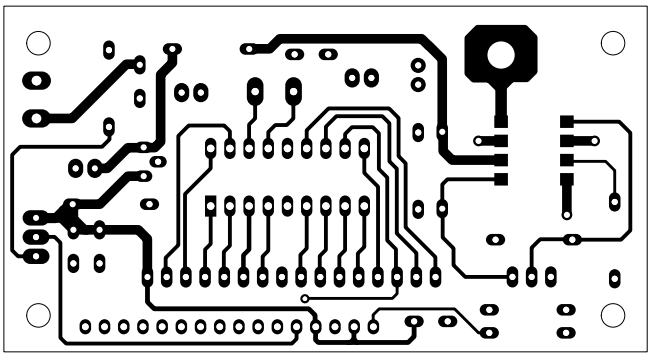


Figure 7. Côté composants du circuit imprimé de l'émetteur.

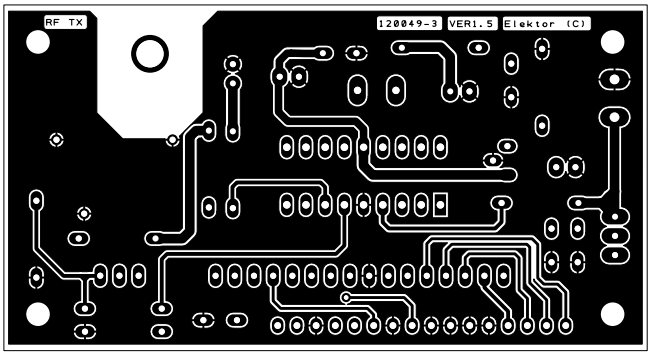


Figure 8. Côté cuivre du circuit imprimé de l'émetteur.

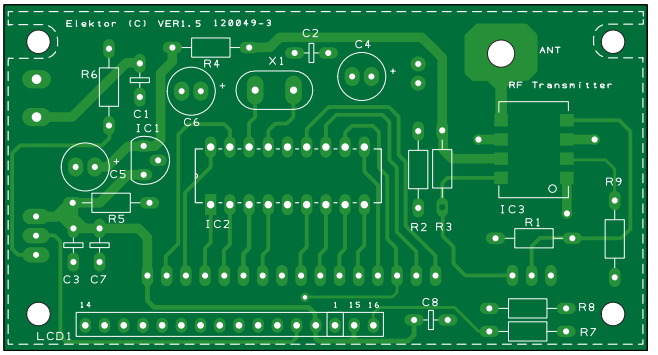


Figure 9. Disposition des composants sur la carte de l'émetteur.

broche de réglage de la puissance d'émission (qui n'existe évidemment pas sur un récepteur). R9 et D1, reliées à la broche 7 (RB1) d'IC2, permettent de voir d'un coup d'œil si la liaison radio est opérationnelle. Un programme très simple sur l'émetteur permet d'envoyer la commande nécessaire pour activer la broche RB7 côté récepteur. Si cette commande est transmise correctement, la LED s'illuminera. J'en reparlerai en détail lorsque je présenterai la partie logicielle.

Les transistors T1 et T2 forment un convertisseur de niveau non-inverseur qui relie la sortie 3 V du module RF à l'entrée 5 V du micro. Deux remarques :

- Oui, il aurait été possible d'alimenter le micro sous 3 V. Cela n'a pas été fait afin que le circuit reste plus universel et compatible avec les vieux PIC qui ne fonctionnent que sous 5 V.
- En utilisant un convertisseur inverseur, il aurait été possible d'économiser un transistor et deux résistances, mais il aurait fallu gérer

Liste des composants

Émetteur

Résistances

R1 = 0 Ω
R2,R6 = 180 Ω
R3 = 470 Ω
R4 = 10 Ω
R5,R9 = 1 kΩ
R7 = 330 Ω
R8 = 680 Ω
P1 = 10 k Ω, aj. multitours

Condensateurs

C1,C2,C3,C7,C8 = 100 nF
C4,C6 = 10 µF 25 V
C5 = 3,3 µF 50 V

Semiconducteurs

D1 = LED rouge 5 mm
IC1 = LP2950-30LPR
IC2 = PIC16F628A-I/P
IC3 = TXM-315-LR, *Linx Technologies* (version 418 ou 433 MHz)

Divers

ANT = ANT-315-PW-LP, *Linx Technologies*
K1 = bornier à vis pour C.I., deux voies au pas de 5 mm
K2 = barrette sécable 16 voies au pas de 2,54 mm
K3 = barrette sécable 3 voies au pas de 2,54 mm
LCD1 = 2x16 caractères, DEM16217, disponible sous la référence Elektor 120061-71
X1 = résonateur à quartz 20 MHz

circuit imprimé 120049-3

ça dans le logiciel. Je préfère simplifier. Si ce montage avait été conçu pour une production de masse, un directeur financier consciencieux aurait sans doute refusé ma solution.

Comme sur le module émetteur, vous pourrez utiliser le micro de votre choix simplement en retirant le PIC de son support et en utilisant les trois lignes du module RF. Le circuit imprimé s'accommodera parfaitement d'une utilisation sur plaque d'essais ; le relier à d'autres montages se fera en un clic de connecteur.

Assemblage de l'émetteur (TX)

Linx Technologies fabrique des modules émetteurs pour trois fréquences différentes, compatibles broche à broche et donc facilement interchangeables. La conception RF requiert des précautions particulières si l'on souhaite obtenir les performances désirées. *Linx Technologies* a fait un gros effort pour fournir des modules fiables et stables, il faut donc suivre leurs recommandations de tracé de circuit imprimé pour tirer les

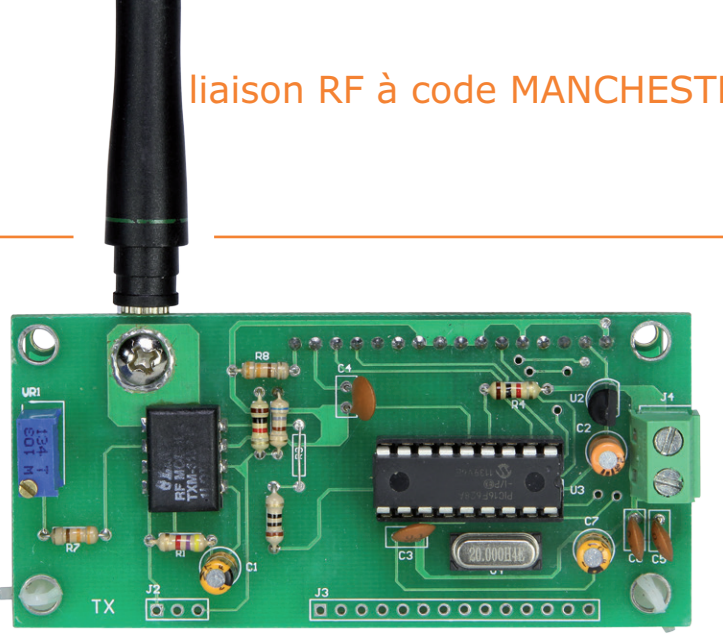


Figure 10. Notre premier prototype aura encore besoin de quelques changements.

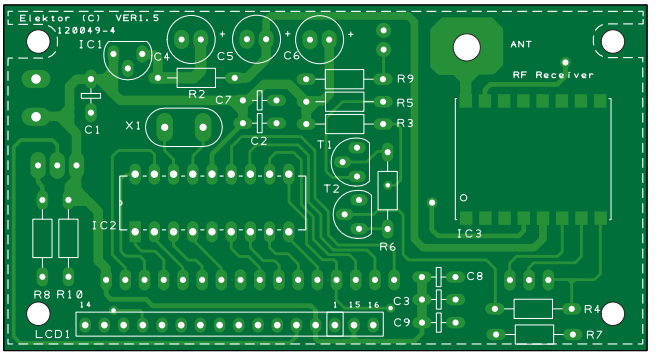


Figure 11. Disposition des composants sur la carte récepteur.

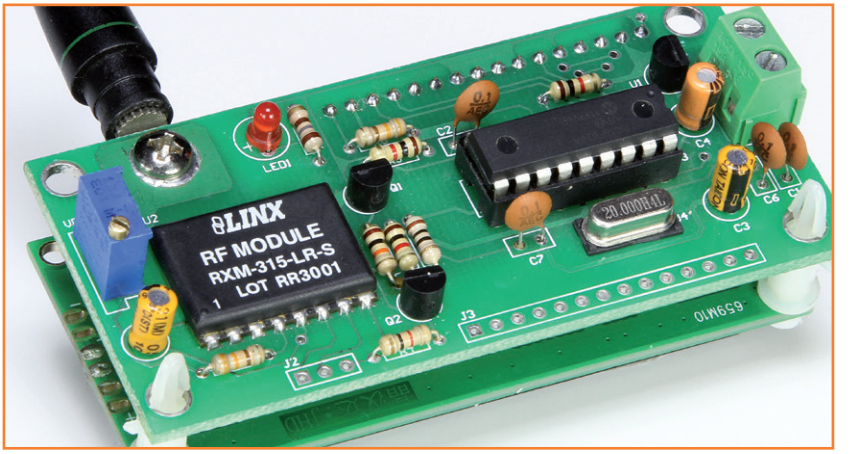


Figure 12. Le LCD est enfiché au dos du prototype de récepteur.

Liste des composants

Récepteur

Résistances

R1 = omise
R2 = 10 Ω
R3,R5,R10 = 1 kΩ
R4,R6 = 10 kΩ
R7 = 330 Ω
R8,R9 = 180 Ω
P1 = 10 kΩ, aj. multitours

Condensateurs

C1,C2,C3,C7,C8,C9 = 100 nF
C4 = 3,3 uF 50 V
C5,C6 = 10 uF 25 V

Semiconducteurs

D1 = LED rouge 5 mm
IC1 = LP2950-30LPR
IC2 = PIC16F628A-I/P
IC3 = RXM-315-LR, *Linx Technologies* (version 418 ou 433 MHz)
T1,T2 = BC547B

Divers

ANT = ANT-315-PW-LP, *Linx Technologies*
K1 = bornier à vis pour C.I., deux voies au pas de 5 mm
K2 = barrette sécable 16 voies au pas de 2,54 mm
K3 = barrette sécable 3 voies au pas de 2,54 mm
LCD1 = 2x16 caractères, DEM16217, disponible sous la référence Elektor 120061-71
X1 = résonateur à quartz 20 MHz

circuit imprimé 120049-4

meilleures performances de leurs modules. Il y a trois instructions clés à respecter :

- Plan de masse sur la face opposée au module.
- Aucune piste ne doit passer sous le module, et aucun conducteur ne doit se trouver à moins de 3,8 mm (0,15") du dessus ou des côtés du module.
- La piste allant à l'antenne doit être aussi courte que possible.

Avec ces instructions à l'esprit, j'ai commencé à dessiner la platine autour du module RF, puis positionné les autres composants en fonction. Vous trouverez en figure 7 le résultat pour l'émetteur vu côté composants et en figure 8 le côté cuivre. Vous remarquerez la disposition à l'écart du module RF, le plan de masse côté cuivre et le peu de pistes côté composants. Remarquez la piste courte menant à l'antenne fixée à la platine par une vis. N'hésitez pas à ajouter un peu de soudure pour améliorer le contact et la robustesse. Grâce au pas de 2,54 mm (0,1") des connecteurs K2 et K3, le circuit est facile à brancher sur une plaque d'essais.
La disposition des composants se trouve en figure 9. La carte, dont vous pourrez voir un premier prototype en figure 10, mesure 85 × 46 mm. Le µC sera monté sur un support afin qu'il soit facile de le retirer pour en faire évoluer le micrologiciel.

Assemblage du récepteur (RX)

Toutes les mises en garde faites pour l'assemblage de l'émetteur restent valables pour le récepteur : plan de masse étendu côté cuivre, mise à l'écart du module RF et piste d'antenne aussi courte que possible (antenne vissée et soudée).
Les typons de l'émetteur et du récepteur peuvent tous deux être téléchargés [2]. La disposition des composants du récepteur est représentée en **figure 11**. La carte fait la même taille que celle de l'émetteur.

Le pas des connecteurs est exactement le même (2,54 mm) que pour l'émetteur et compatible avec les plaques d'essais. Comme pour les émetteurs, *Linx Technologies* fournit des récepteurs pour trois fréquences des bandes ISM UHF.
Le premier prototype de notre récepteur est visible en **figure 12**, cette fois encore le microcontrôleur est monté sur un support afin de pouvoir être aisément reprogrammé.
C'est une évidence, mais toujours bonne à rappeler : pour qu'une liaison radio fonctionne, tous les composants RF (modules émetteur et récepteur, antennes) doivent être accordés sur la même fréquence. L'antenne, qui bien évidemment ne fait pas la différence entre émission et réception, sera du même modèle des deux côtés. Une petite remarque : si ces antennes sont plutôt bonnes, elles ne sont pas parfaites. Côté émission, vous voudrez peut-être limiter la puissance (et parfois le rendement de l'antenne) pour rester au-dessous des seuils prévus par la réglementation locale. Côté récepteur, vous voudrez sans doute améliorer le plus possible son efficacité. N'hésitez pas à expérimenter avec des monopoles en quart d'onde droits si vous avez besoin d'augmenter la portée.

Ça sera la conclusion de la première partie de cet article. Le mois prochain, nous parlerons du logiciel universel mis au point pour ces modules.

(120049 – version française : Kévin PETIT)

Liens

[1] www.linxtechnologies.com
[2] www.elektor.fr/120049

Télécharger des applications existantes
n'est pas le seul moyen de

rendre votre téléphone tactile encore plus intelligent !

Voici un nouveau livre, publié par Elektor, pour apprendre à construire des applications sous Android et à tirer le meilleur de ce système d'exploitation.
C'est un ouvrage d'initiation, avec des exemples simples, variés et concrets, qui montre de façon progressive comment la combinaison de briques de code permet de créer toutes sortes d'applis dans l'environnement de développement Eclipse :
calculette simple, interrogation des capteurs, exploitation des données GPS, communication par l'internet etc.

33,50 €

208 pages

format 17 x 23,5 cm

ISBN 978-2-86661-187-3

elektor

www.elektor.fr/android

les bibliothèques

Neil Gruending
(Canada)

Grâce aux précédents articles de cette série, vous savez désormais installer et configurer DesignSpark (DS). Ce mois-ci nous allons jeter un œil du côté des bibliothèques que nous utiliserons pour créer schémas et circuits imprimés.

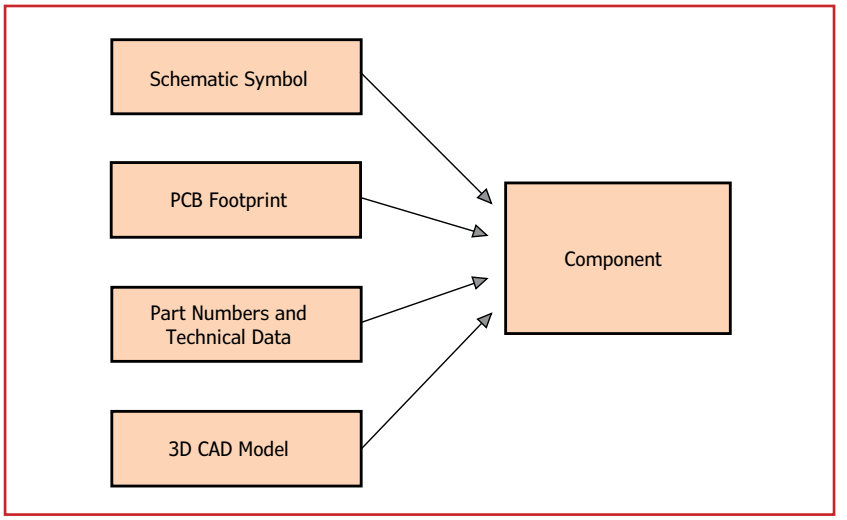


Figure 1.
Les différentes facettes d'un composant DesignSpark.

Les bibliothèques

Lorsque nous avons créé des cartouches le mois dernier, nous avons commencé par créer un symbole pour le schéma puis un composant pour référencer ce symbole dans les bibliothèques correspondantes. C'était un bon exemple de composant de documentation qui ne nécessitait pas l'association avec des informations relatives au PCB, une exception : le plus souvent on souhaite incorporer les informations de la figure 1.

Le symbole est stocké dans une bibliothèque de symboles (fichier *.ssl), l'empreinte dans une bibliothèque d'empreintes pour circuits imprimés et le modèle en 3D dans une bibliothèque de visualisation (*.pkg). Le composant qui rassemble le tout est lui stocké dans une bibliothèque de composants (*.cml) accompagné de sa références et des informations techniques associées. Bien entendu, cette dernière bibliothèque fait référence aux précédentes.

DS utilise différents fichiers pour les différents types de données afin de faciliter la réutilisation des données de conception dans de multiples composants. Vous pourriez p. ex. créer plusieurs composants résistance en ne créant qu'un seul symbole que vous réutiliseriez. Étant donné que tous les composants font référence au même symbole, toute modification serait automatiquement répercutée dans tous les composants qui l'utilisent. Il en est de même pour les empreintes pour PCB et les modèles en 3D. Vous trouverez un bon guide sur les bibliothèques et leur utilisation sur le site DesignSpark [1].

Leur organisation

Les bibliothèques livrées avec DesignSpark sont habituellement installées dans `C:\Users\Public\Documents\DesignSpark PCB 5.0\Library`. Elles constituent un bon exemple de la façon d'organiser par fabricant une grande bibliothèque de composants. Comme j'utilise également mes bibliothèques en guise de bases de données de références, je préfère les organiser par type de composant. Ma bibliothèque de transistors contient p. ex. un 2N3904 mais j'y associe plusieurs références de fabricants afin de ne pas avoir à mémoriser ceux que j'ai utilisés précédemment. J'essaie aussi de réutiliser les symboles et les empreintes que je place donc dans des bibliothèques génériques organisées comme suit :

- des bibliothèques de composants par groupes (transistors, résistances, condensateurs, etc)
- une bibliothèque générique de symboles (symboles résistance, condensateur, etc)
- une bibliothèque générique d'empreintes CMS (empreintes 0603, LQFP, etc)
- une bibliothèque générique d'empreintes pour composants traversants (empreintes DIP, résistance ¼ W, etc)

Voyons maintenant comment utiliser ces bibliothèques en commençant par *ModelSource*.

ModelSource

Pour celles et ceux qui ne le sauraient pas, *ModelSource* est une base de données de composants

en ligne utilisable avec beaucoup de suites logicielles de conception de circuits imprimés différentes, dont DS. J'aime bien le fait que DS puisse se connecter directement à *ModelSource*, ce qui permet de trouver des composants sans quitter l'application (un guide est disponible sur [2]). C'est l'endroit parfait pour trouver des empreintes conformes IPC qui respectent les contraintes de fabrication. Pour ouvrir *ModelSource* dans DS, cliquez sur le bouton *ModelSource* ou sélectionnez View->*ModelSource* Bar pour faire apparaître la barre d'outils de la **figure 2**.

Nous allons essayer de trouver un transistor NPN CMS MMBT3904 à l'aide du moteur de recherche paramétrique. Cliquez sur *CLICK TO CHOOSE* et identifiez-vous si nécessaire. Faites apparaître la liste des transistors bipolaires en sélectionnant *Semiconductors->Discrete Semiconductors->Bipolar Transistors* ; cela devrait afficher 740 transistors différents (**fig. 3**).
Réduisons maintenant le nombre de résultats en ajoutant des filtres sur les données. Pour le champ *Transistor Type* sélectionnez NPN, Surface *Mount* pour *Mounting Type*, SOT-23 pour *Package Type* et 40V pour *Maximum Collector Emitter Voltage*. Le deuxième transistor listé est le MMBT3904 que nous recherchions. En cliquant sur *Load Preview* vous obtiendrez l'écran suivant dans lequel s'afficheront le symbole de schéma, l'empreinte physique ainsi que quelques paramètres clés du composant (**fig. 4**).

Vous pouvez également effectuer des recherches en utilisant le champ *Part Number Quick Search* si vous connaissez déjà une partie de la référence. Maintenant que nous avons trouvé notre composant, il ne reste qu'à cliquer sur *Use Component* pour que DS télécharge le composant dans une bibliothèque placée dans le dossier des bibliothèques téléchargés (vous trouverez le chemin complet dans l'onglet *Folders* du *Library Manager*). DesignSpark vous donnera le nom de la bibliothèque une fois le composant téléchargé. Vous pouvez maintenant utiliser le transistor en le faisant glisser depuis la fenêtre *ModelSource* ou en utilisant l'habituel bouton *Add Component* de la barre d'outils.

C'est super, mais que faire lorsque *ModelSource* ne connaît pas le composant qu'il vous faut ou que vous souhaitez le modifier ? Par exemple, je souhaiterais modifier le composant MMBT3904

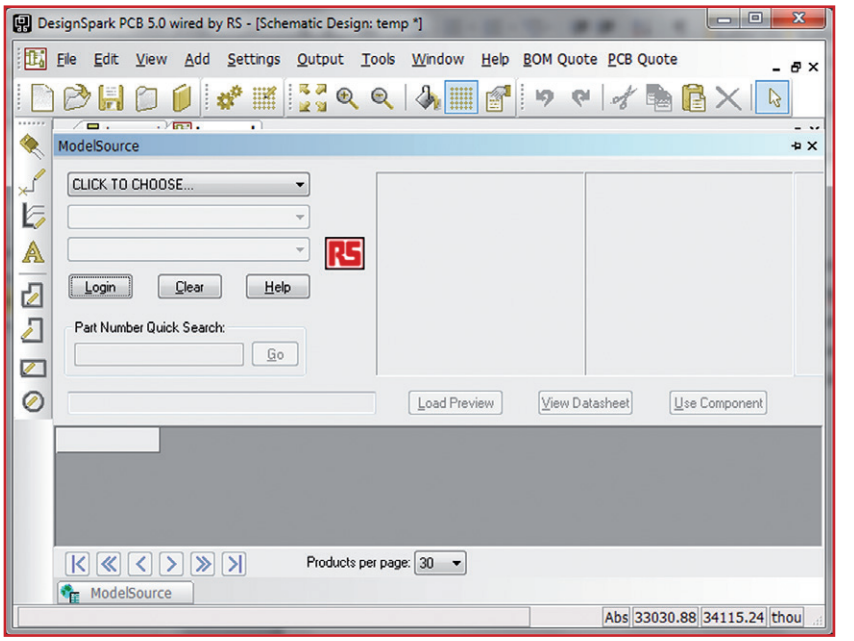


Figure 2.
Aperçu de *ModelSource* juste après ouverture.

que nous venons de télécharger pour le doter d'un symbole plus classique faisant apparaître l'émetteur. On utilisera pour cela les ...

Bibliothèques personnalisées

Même si cela représente beaucoup de travail, j'aime disposer de mon propre ensemble de bibliothèques. Lorsque c'est possible je préfère copier des composants depuis d'autres sources

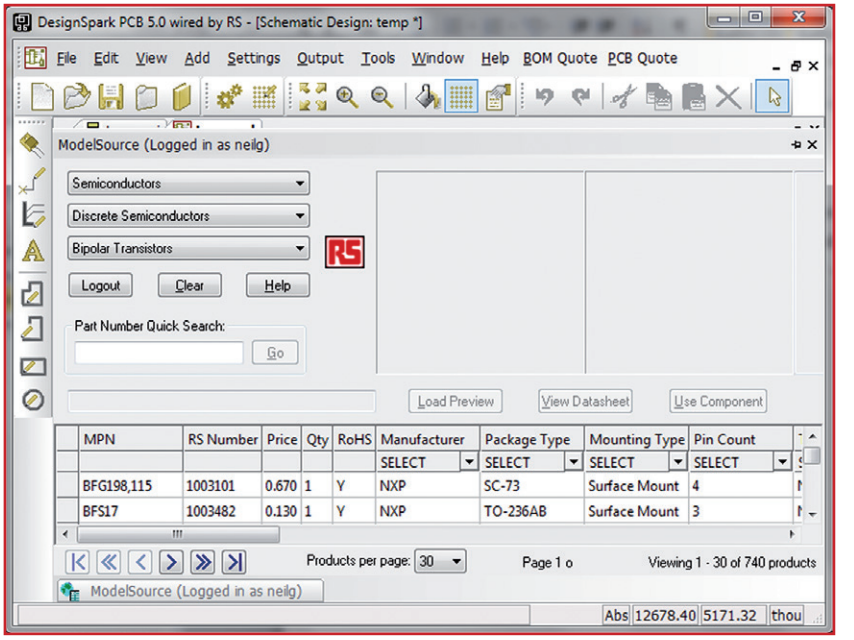


Figure 3.
Les résultats de recherche présentés par *ModelSource*.

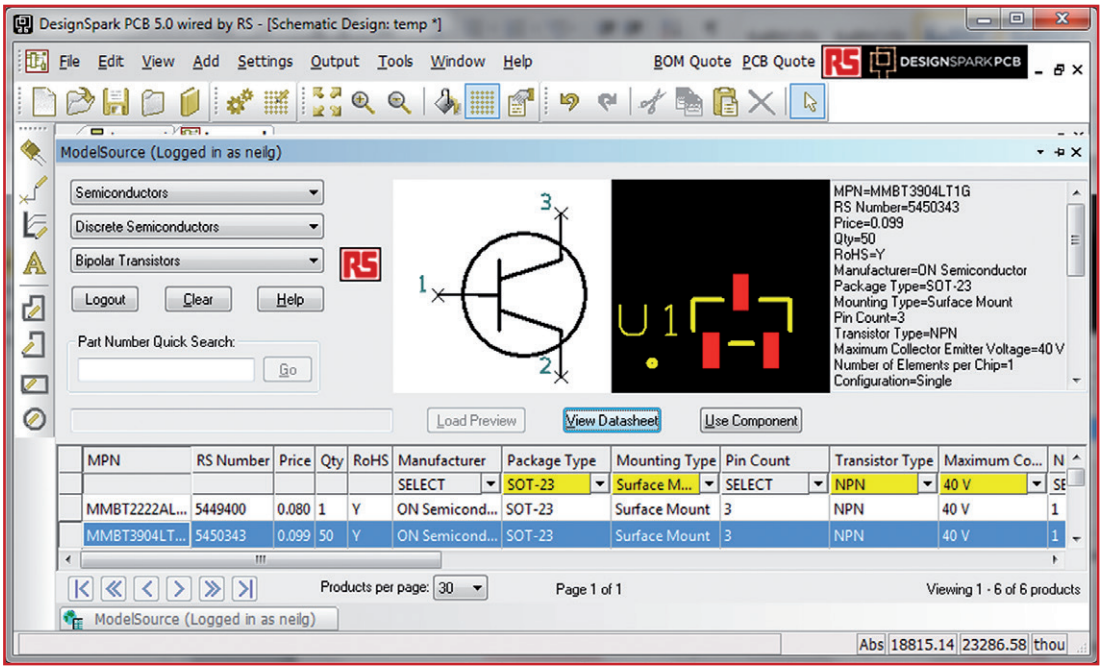


Figure 4.
Informations détaillées sur un transistor dans *ModelSource*.

pour les modifier ensuite. Dans le cas de notre MMBT3904, il suffirait de copier les informations que nous venons de télécharger dans nos propres bibliothèques à l'aide du *Library Manager* puis de modifier le composant à notre guise. C'est aussi le bon moment pour tout vérifier et repérer les erreurs éventuelles.

Le plus important, lorsque vous mettez en place vos propres bibliothèques, est d'utiliser des attributs communs pour chacun des composants ; cela vous permettra de produire des rapports tels que l'habituelle liste des composants (BOM). Comme je préfère stocker toutes les informations de fabrication d'un composant dans la bibliothèque du composant, j'ai habituellement plusieurs références constructeur que je stocke à l'aide d'attributs (**fig. 5**). Vous ne pourrez pas retirer les attributs par défaut si vous souhaitez utiliser la fonction de création de BOM de DesignSpark.

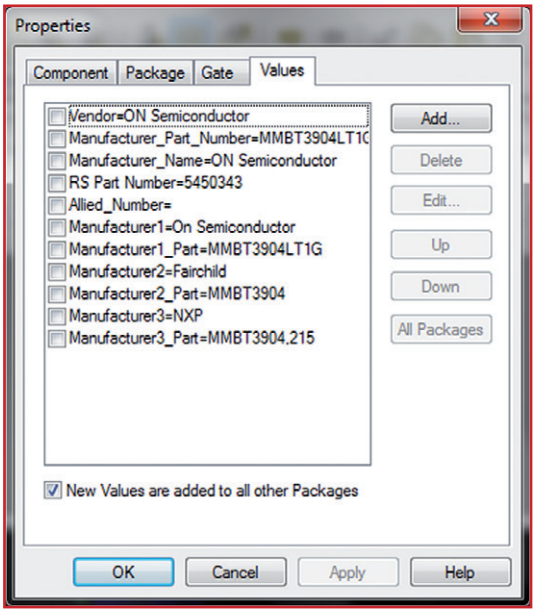


Figure 5.
Les attributs des composants peuvent être utiles.

Conclusion

ModelSource et les bibliothèques de DS sont une mine d'or pour qui veut créer ses propres bibliothèques ; ils vous feront gagner un temps fou. Vous pouvez maintenant utiliser tout cela pour créer vos schémas avec une liberté plus grande encore. La prochaine fois, je vous donnerai quelques astuces sur l'édition de schémas et vous expliquerai comment produire une BOM.

(130207 – version française : Kévin PETIT)

Liens

[1] www.designspark.com/tutorial/components-library-structure-library-manager

[2] www.designspark.com/eng/tutorial/components-downloading-from-modelsource-building-up-libraries

elektor cardi♥scope Android

sans fil, sans bouton : Bluetooth & écran tactile

2e partie

Marcel Cremmel
(en coopération avec Raymond Vermeulen)

Après la description du matériel de notre nouvelle interface pour ECG sur tablette ou téléphone tactile Android entreprise dans le numéro double de juillet-août 2013, nous revenons ici sur les fonctions du PIC et sur le déroulement de son programme, avant d'aborder l'application Android. Sans entrer dans les détails, nous en disons cependant assez pour inciter nos lecteurs à s'approprier le code et se lancer à leur tour dans le développement sous Android.

Que fait le PIC24 ?

Acquisition et transmission des échantillons (fig. 5)

Trois modules matériels intégrés au µC sont utilisés :

- le CAN à 10 bits et son multiplexeur analogique,
- l'UART (*Universal Asynchronous Receiver Transmitter*) pour la communication avec le module *Bluetooth* (= BT),
- le *Timer1* pour produire les signaux P2HZ et CAL.

Le multiplexeur analogique du CAN permet la conversion des trois entrées analogiques DI, DII et BATT_LEV. Ce dernier signal est produit par un diviseur résistif par 2 (R16/R17) à partir de la tension des piles.

Le CAN est configuré en mode *autoconversion* et *autoscan* : il assure la sélection, l'échantillonnage et la conversion des trois entrées sans intervention du processeur.

La fréquence d'échantillonnage de 2 kHz est largement suffisante pour l'acquisition d'un signal ECG.

Les résultats des conversions sont rangés dans trois variables codées sur 16 bits : Channel_DI, Channel_DII et Vbatt.

À la fin de chacune des trois conversions, soit à une fréquence de 2000 Hz, une interruption (_ADC1Interrupt) effectue le traitement suivant :

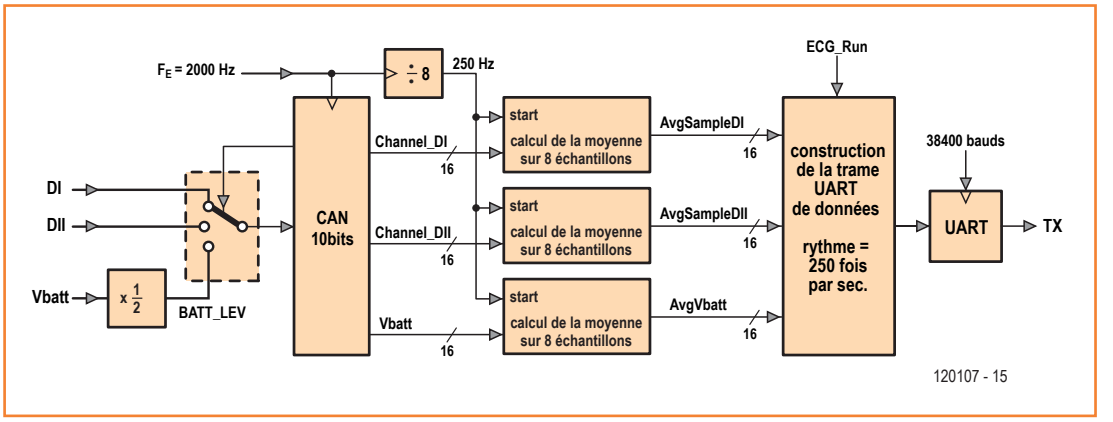


Figure 5. Acquisition et transmission des échantillons de DI et DII.

Atout cœur, ça tombe à PIC

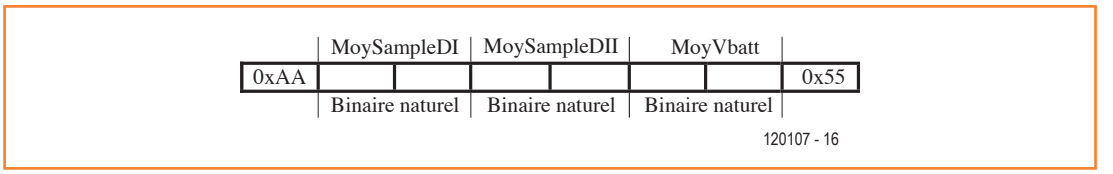


Figure 6. Format de la trame UART.

- Tous les 8 échantillons, soit à un rythme de 250 Hz : calcul des valeurs moyennes **AvgSampleDI**, **AvgSampleDII** et **AvgVbatt**. Ce traitement permet de réduire l'effet de parasites ponctuels.
- Construction et transmission de la trame série asynchrone de données vers le module BT.

La **figure 6** représente le format adopté pour cette trame de 8 octets. Les données sont encadrées par les octets 0xAA et 0x55. Ils serviront dans le terminal Android à réaliser la synchronisation de trame et ainsi à identifier et à prélever les échantillons. La valeur des échantillons est comprise entre 0x0000 et 0x03FF (conversion sur 10 bits en binaire naturel), une fausse synchronisation est impossible.

Sélection des constantes de temps d'auto-zéro (fig. 7)

Cette fonction logicielle adapte en permanence la vitesse d'alignement des signaux DI et DII (voir le paragraphe "Schémas à cœur ouvert" du 1er article) de façon à stabiliser chaque ECG à l'écran le plus vite possible.

Pour ce faire, la fonction **MovingAverageCalc()** calcule la moyenne glissante des signaux numériques **AvgSampleDI** et **AvgSampleDII**

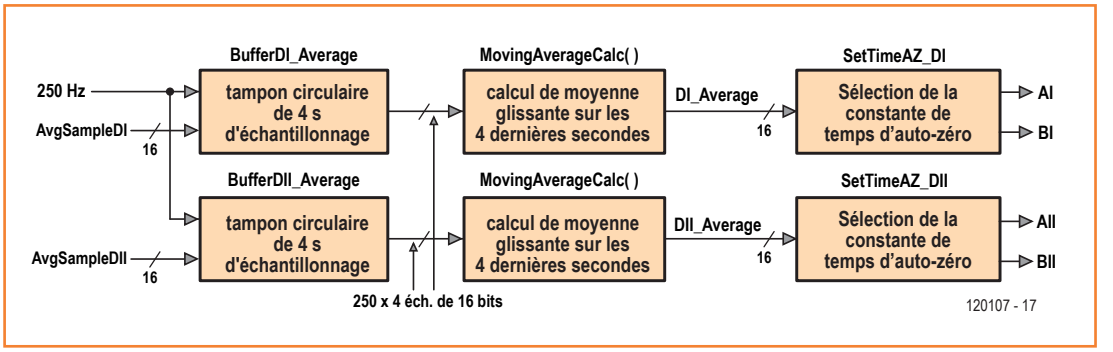


Figure 7. Sélection des constantes de temps d'auto-zéro.

NB : La numérotation des illustrations et des liens suit celle de la première partie de cet article.

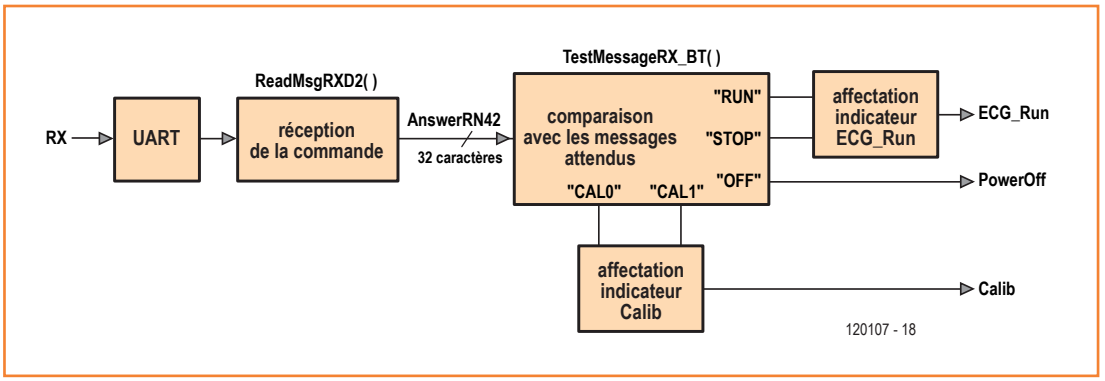


Figure 8.
Réception des ordres depuis la tablette ou le téléphone Android.

sur une durée de 4 s. Les résultats **DI_Average** et **DII_Average** sont comparés aux valeurs de repos attendues pour choisir, via AI et BI ou AII et BII, une constante de temps d'auto-zéro d'autant plus rapide que l'écart est grand.
Rappelons ce que signifie l'expression *moyenne glissante*. Les échantillons **AvgSampleDI** et **AvgSampleDII** sont accumulés dans un tampon circulaire de 4 s, soit ici 4 x 250 = 1000 mots de 16 bits. La fonction **MovingAverageCalc()** calcule alors à un rythme suffisant la moyenne arithmétique sur les 1000 derniers échantillons du tampon.
Le dernier échantillon correspond à l'instant du calcul et ***glisse*** donc avec le temps.

Réception des ordres depuis le terminal (fig. 8)
Peu d'ordres proviennent de l'utilisateur via le terminal :

- une commande Run/Stop pour valider ou bloquer la transmission des trames de données
- la mise hors tension de l'interface. Notez que la mise sous tension ne peut se faire que par le bouton poussoir M/A de l'interface
- les commandes **CAL0** et **CAL1** pour piloter la production des signaux de calibrage.

Le module UART se charge de la conversion série/parallèle de chaque octet du message reçu. Les fonctions de réception d'octets fournies dans les bibliothèques de *Microchip* n'exploitent pas les interruptions. Pour éviter les boucles d'attente de ces fonctions qui occupent inutilement le processeur, nous utilisons l'interruption en réception UART. La fonction associée, **_U2RXInterrupt**, accumule les caractères reçus dans un tampon de taille suffisante (256 octets). Ces caractères sont lus sans perte de temps par la fonction **ReadMsgRXD2()**.
La variable chaîne de caractères **AnswerRN42** est affectée dès qu'un message complet est reçu (séquence terminale = CR-LF). La fonction **TestMessageRX_BT()** compare alors celle-ci à une des commandes attendues.
Elle affecte en conséquence l'indicateur **ECG_Run** de validation de la transmission des trames de données (fig. 5), le signal /PowerOff de mise hors tension et l'indicateur **Calib** de validation de la production du signal de calibrage.

Production des signaux de calibrage (fig. 9)
Les signaux P2HZ et CAL agissent sur le multiplexeur analogique IC9 (fig. 3, F2) pour remplacer périodiquement les tensions prélevées par les électrodes par un signal de calibrage d'1 mV d'amplitude. La fréquence du signal P2HZ est de 2 Hz et son rapport cyclique de

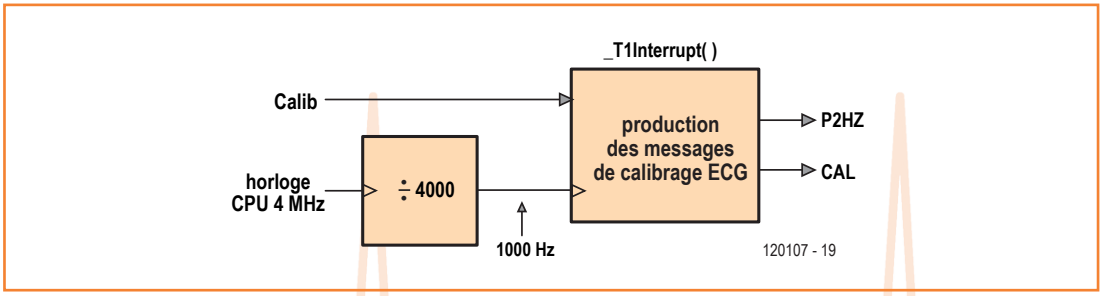


Figure 9.
Production des signaux de calibrage.

faites vous-même vos électrocardiogrammes sur votre tablette ou votre téléphone tactile !

20 %, proches de ceux d'un signal ECG. Le signal produit par le µC est atténué par le réseau R21-R22-R65 pour obtenir 1 mV d'amplitude et une valeur moyenne nulle.
Le signal CAL passe à 1 pendant 10 s chaque minute si l'utilisateur a validé sa production depuis la tablette ou le téléphone.
Ces signaux sont produits par un séquenceur implanté dans le µC. Il est constitué :

- d'un diviseur de fréquence par 4000 réalisé par une structure matérielle (le module *Timer1*)
- d'une fonction logicielle d'interruption **_T1Interrupt** activée 1000 fois par seconde. Si l'indicateur **Calib** est positionné, des variables de comptage sont incrémentées et comparées à des constantes pour produire les signaux P2HZ et CAL.

État de la liaison *Bluetooth* (fig. 10)

En l'absence de liaison BT, il est inutile de convertir les signaux ECG et de les transmettre. Le signal STATUS produit par le module BT donne cette information : liaison établie (1) ou rompue (0). La fonction d'interruption **_CNInterrupt** est activée à chaque changement d'état de STATUS et affecte en conséquence l'indicateur **ECG_Run** et le bit ADON de validation du convertisseur A/N (fig. 5). Le choix d'une fonction d'interruption évite la scrutation périodique du signal STATUS et, par conséquent, une perte de temps du processeur.

Déroulement du programme du PIC

L'architecture du programme est classique (contrairement à l'application Android, comme on le verra). Après la mise sous tension, les opérations suivantes sont exécutées :

- initialisation des variables, des ports d'entrée/sortie, du *Timer1* pour la production des signaux de calibrage (voir ci-dessus), du coupleur UART2 pour communiquer avec le module BT ;
- configuration du module BT pour passer à 38400 bauds ;
- initialisation du CAN à 10 bits : fréquence d'échantillonnage de 2000 Hz, auto-conversion et auto-scan des trois entrées analogiques ;
- validation de l'interruption CN ;
- le programme entre alors dans une boucle sans fin :
  - appel **TestMessageRX_BT()** : lecture et traitement de l'éventuel ordre reçu depuis le terminal
  - appel **MovingAverageCalc()** : calcul de la moyenne glissante **DI_Average**
  - appel **SetTimeAZ_DI()** : sélection de la constante de temps d'auto-zéro de la voie DI
  - appels **MovingAverageCalc()** et **SetTimeAZ_DI()** pour la voie DII.

Les fonctions de calcul des valeurs moyennes sont placées dans la boucle sans fin, car leur temps d'exécution est assez long (26800 cycles CPU, soit 6,7 ms). Selon les règles de programmation, on évite de confier des traitements longs à des fonctions d'interruption. En effet, pendant ce temps, les autres fonctions d'interruption, de priorité inférieure, ne seraient pas exécutées, ce qui pourrait entraîner une défaillance.
La fréquence d'exécution de la boucle sans fin est de 75 Hz environ, rythme suffisant pour le calcul des moyennes glissantes et la sélection des constantes de temps d'auto-zéro.

Interface homme-machine

Comme IHM, on aura du mal à trouver plus convivial (et meilleur marché) qu'un terminal Android (ou un *iPhone*). Elektor ne s'y est pas trompé en publiant déjà de nombreux articles dans ce sens et même un livre dont le succès confirme la forte demande : *Android | Apprendre*

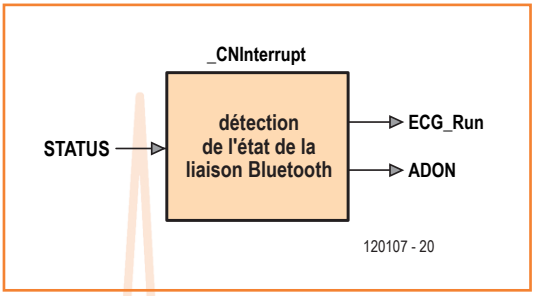


Figure 10. Détection de l'état de la liaison *Bluetooth*.

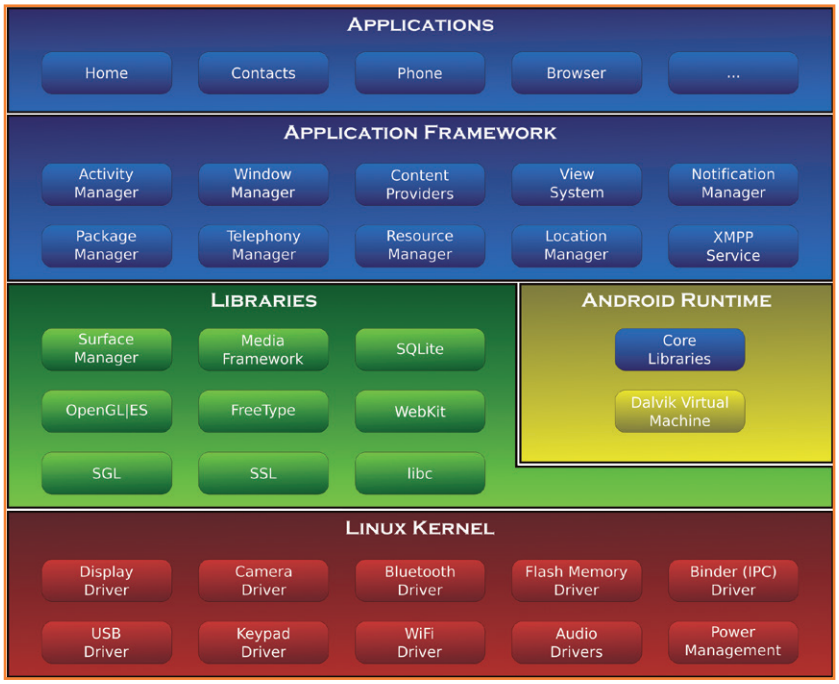
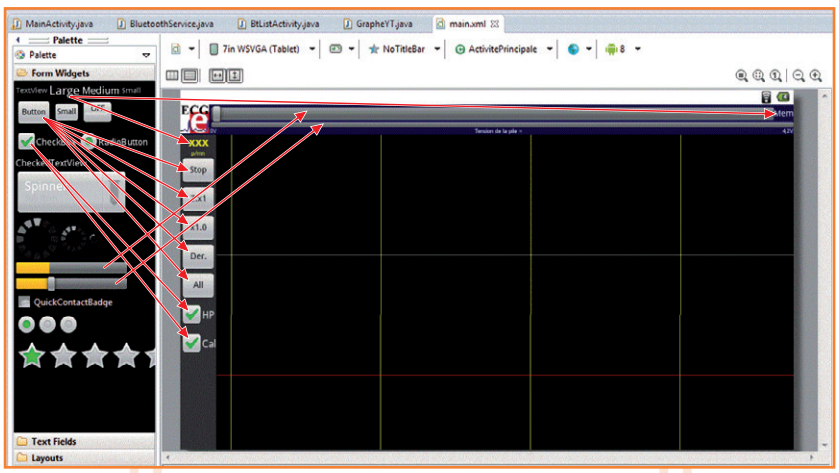


Figure 11.
L'architecture du système Android (cette image, reproduite ici en petit à titre indicatif, est téléchargeable sous forme de fichier à haute résolution).

Figure 12.
Il suffit de glisser-déposer sur l'écran (à droite) des éléments de la bibliothèque de composants graphiques choisis dans la palette (à gauche).

à programmer des applis de Stephan Schwark [4]. Nous invitons nos lecteurs intéressés par ce sujet à découvrir, approfondir, voire critiquer le code de l'elektorcardioscope disponible sur le site d'Elektor [3]). Comme il est impossible d'en décrire en quelques pages les 1900 lignes, nous donnerons ici assez d'informations pour encourager nos lecteurs à se plonger dans le code source afin d'y retrouver les fonctions décrites. Les programmeurs avertis pourront y apporter les modifications ou améliorations qu'ils souhaitent. D'autres y trouveront peut-être la motivation pour se lancer à leur tour dans le développement d'applications Android.



Comme le graphe déroulant fluide exige de la vitesse mais que les performances graphiques des applications développées sous *AppInventor* sont médiocres, j'ai dû renoncer à utiliser cet environnement gratuit. Mais je le recommande pour d'autres applications plus simples comme p. ex. le pilotage d'un robot *Mindstorm* en BT ou pour tous nos lecteurs qui désirent s'initier à la programmation.
J'ai opté ici pour l'Android SDK de *Google,* gratuit aussi. Les outils du SDK (sur PC, MAC ou LINUX) sont inclus dans un IDE populaire et gratuit : *Eclipse*. Son installation complète est longue, mais aisée si on suit la procédure décrite par *Google*. De bonnes connaissances sont nécessaires en Java, langage orienté objet (comme C++). Curiosité et goût de l'effort conjugués mettront son apprentissage à la portée de ceux qui savent déjà écrire des programmes en C. D'excellents tutoriaux sont disponibles [5] ainsi que mes propres documents sur mon site [6].

Développer pour Android

Le développement d'une application pour un système d'exploitation embarqué comme Android requiert une bonne connaissance de son architecture (**fig. 11**). L'utilisateur final n'accède directement qu'aux applications installées sur son terminal (la couche supérieure de l'illustration). Le développeur peut utiliser ces applications pour sa propre application, mais dispose surtout d'une riche collection d'API (*Application Programming Interface*) écrites en Java pour exploiter les ressources de la tablette. Elles sont regroupées dans le cadre des applications (*Application Framework*). Ces API font appel à des bibliothèques (en C et C++) qui se reposent elles-mêmes sur un noyau Linux.
L'originalité d'Android est son moteur d'exécution basé sur une machine virtuelle (= MV) *Dalvik VM*. Le principe est proche de la machine virtuelle Java (JVM) utilisée sur PC et MAC : le compilateur Java produit des fichiers exécutables en *bytecode* indépendamment du processeur utilisé. La MV, spécifique à chaque appareil, exécute alors les *bytecode* de l'application qui se comportera de la même façon quel que soit l'ordinateur cible.
Sous Android aussi, le *bytecode* produit par le compilateur pourra être exécuté sur tous les terminaux, quelque soit le processeur utilisé.
Chaque application Android tourne dans son propre processus, avec sa propre instance de machine virtuelle. Dalvik a été écrit de sorte qu'un

matériel puisse faire fonctionner efficacement plusieurs MV.

Composer les écrans

Composer les écrans de l'application grâce au SDK Android avant même d'avoir écrit la moindre ligne de code, quelle étape gratifiante ! Le programmeur dispose d'une bibliothèque de composants graphiques qu'il suffit de déposer sur l'écran à sa convenance (**fig. 12**). Les flèches représentent quelques exemples de glisser-déposer entre palette de composants et écran. L'écran a déjà son aspect final, mais l'activité n'existe pas, car à ce stade pas une seule ligne de code n'a encore été écrite !

Évènements

Une application en C comporte toujours une fonction *main()* suivie d'une boucle sans fin qui appelle successivement les fonctions principales à traiter tandis que l'architecture d'une application Android est basée sur les évènements. En Java sous Android, les fonctions sont toujours exécutées suite à un évènement (touché sur l'écran, réception d'un SMS, etc.) et ne comportent jamais de boucle sans fin. Même la fonction d'initialisation, au lancement de l'application, se termine à la fin de son traitement et rend la main à Android. Le moteur d'exécution peut alors s'occuper des autres applications en cours. Les évènements sont gérés par le système d'exploitation et sont faciles à utiliser dans l'environnement de développement.

Activités

Une application Android comporte autant d'activités que d'écrans différents affichés pendant son exécution. Chacun de ces écrans est constitué de boutons, de textes, de graphes dont les traitements associés font partie de l'activité. Android étant multitâches, une activité connaît plusieurs états :

- active : elle est en cours d'exécution
- suspendue : elle est mise en pause suite à un évènement de priorité supérieure (affichage d'un SMS par exemple)
- arrêtée : une autre activité prend le focus. Le système retient son état pour pouvoir la reprendre, mais il peut arriver qu'il mette fin à l'application pour libérer de la mémoire système.

La **figure 13** illustre le cycle de vie d'une activité,

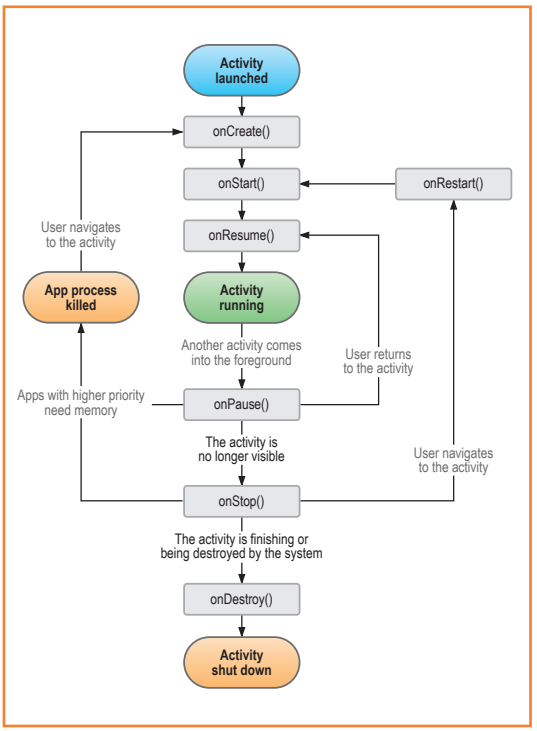


Figure 13.
Cycle de vie d'une activité.

typique dans un système multitâche.

Notre application ANDROECG comporte trois activités :

- **MainActivity** démarre au lancement de l'application. Elle affiche l'écran principal et les boutons de contrôle (voir les copies d'écran) et crée les services nécessaires à l'application.
- **BtListActivity** est démarrée sur demande pour afficher la liste des périphériques BT appariés et sélectionner celui de notre interface.
- **FileListActivity** est démarrée sur une demande de sauvegarde ou de lecture de données ECG. Elle affiche la liste des fichiers existants ainsi qu'une fenêtre d'édition pour créer un fichier.

Services

Il s'agit de tâches de fond ne nécessitant ni écran ni action de l'utilisateur. Les services peuvent communiquer avec les activités via des *Intents* (intentions).

Dans l'application ANDROECG, le service ***BluetoothService*** p. ex. s'occupe de la gestion du module BT : établir la connexion, émettre et recevoir des données et rompre la liaison. Le service ***Timer1Service*** est une tâche périodique

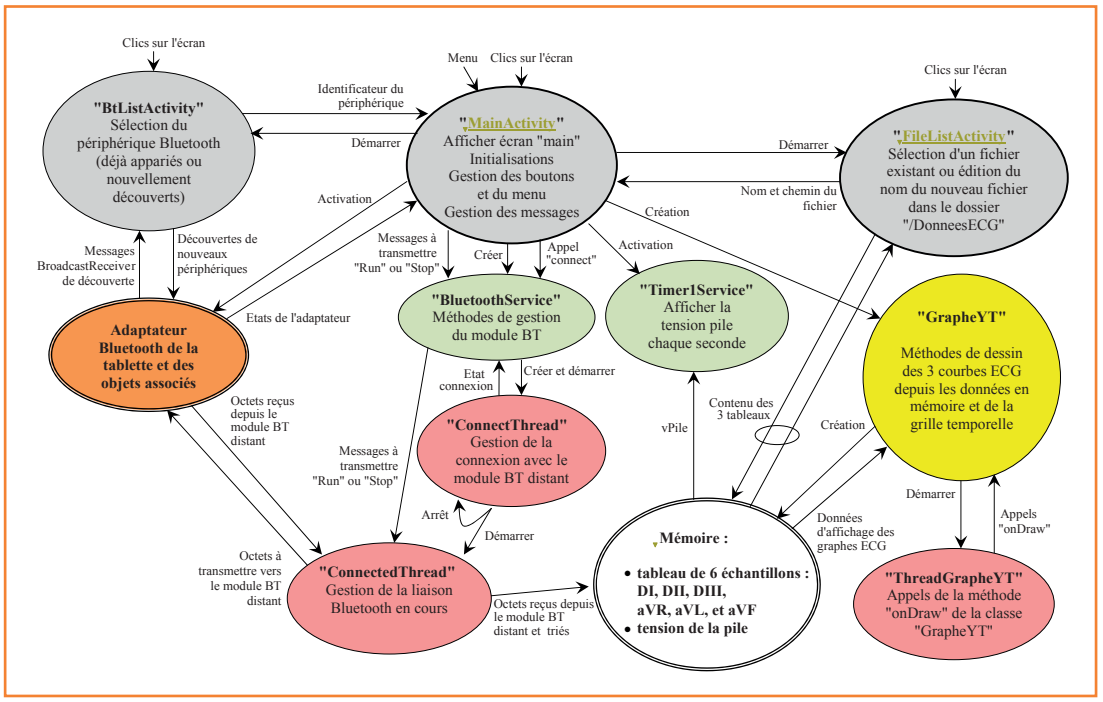


Figure 14.
Organisation des activités, services et *threads* de notre application.

chargée d'afficher la tension de la pile chaque seconde.
Dans les Paramètres de votre téléphone Android, le menu Applications donne à tout moment la liste des services en cours.

Threads

Le *thread* (prononcer *srèd*), ou tâche, est la base de la programmation concourante qui consiste à développer une application dont les tâches, du point de vue de l'utilisateur, s'exécutent simultanément. Chacune des tâches réagit indépendamment des autres à des évènements (clic sur l'écran, réception d'un message BT etc.) et réalise les opérations associées.
Chaque *thread* comporte une méthode (fonction) *run()* qui joue un peu le rôle de la fonction *main()* en C, mais en programmation concourante, il y a autant de *run()* que de *thread*. Un service, par exemple, s'exécute dans un *thread*. Au lancement d'une application, Android crée le *thread* UI (***User Interface***) chargé de détecter tous les évènements utilisés par l'activité (action sur les boutons par exemple) et d'agir en conséquence. Chaque activité ou service peut créer de nouveaux *threads* pour y réaliser des traitements spécifiques.

Notre application ANDROECG comporte les *threads* supplémentaires suivants :

- **ThreadGrapheYT** chargée de l'affichage déroulant des ECG. Pour obtenir une bonne fluidité, on lui attribue une haute priorité.
- **ConnectThread** établit la connexion avec le module BT distant.
- **ConnectedThread** gère la liaison BT en cours, notamment la réception et la transmission des données.

Organisation de l'application ANDROIDECG

L'organisation des activités, services et *threads* de notre application, ainsi que les liens entre eux (*Intents*) est moins compliquée qu'on pourrait le craindre à première vue (**fig. 14**). Observez également les copies d'écran (fig. 15).

MainActivity : Android crée cette activité au lancement de l'application et exécute la méthode ***onCreate()*** (fig. 13). Elle effectue toutes les initialisations nécessaires et crée, entre autres, les services *BluetoothService* et *Timer1Service*. Les autres méthodes (ou fonctions) de l'activité s'occupent des actions sur les boutons tactiles et des fonctions du menu. Les deux dernières méthodes se chargent des messages renvoyés à la fermeture des autres activités et par les services pour réagir en conséquence et/ou informer l'utilisateur (perte de connexion BT p. ex.).

BtListActivity : Cette activité est créée quand on appuie sur le bouton du menu «Paired BT Devices» (fig. 15). Elle ouvre une nouvelle fenêtre et interroge l'adaptateur BT du terminal et affiche la liste les périphériques appariés (**fig. 16**). Un bouton permet de lancer une nouvelle recherche. La fenêtre et l'activité se ferment au choix du périphérique après l'envoi vers l'activité principale d'un message comprenant son identificateur. L'activité principale démarre alors le *BluetoothService* pour établir la liaison avec notre interface ECG.

BluetoothService : Ce service est créé par l'activité principale dès que l'adaptateur BT est activé. Il est chargé d'établir la liaison, puis de la gérer. Pour cela, il crée deux *threads* :

- **ConnectThread** est démarré dès le choix du périphérique. Ce *thread* demande à l'adaptateur BT l'établissement d'une connexion avec le profil SPP. Quand c'est le cas (cela peut durer plusieurs secondes), ce *thread* est supprimé avant de démarrer le suivant.

- **ConnectedThread** reste actif durant tout le temps de connexion avec l'interface ECG. Il comporte notamment les méthodes *write* et *run* chargées respectivement de la transmission et de la réception des données échangées en BT. La méthode *run* détecte dans le flux reçu chaque trame d'échantillons transmis par l'interface au rythme de 250 Hz (fig. 6) pour l'affecter en temps réel à l'un des 6 tableaux d'échantillons. La taille de ces tableaux permet d'enregistrer 10 min d'activité cardiaque. Le *thread* est supprimé à la perte de la liaison ou à la fermeture de l'application.

GrapheYT : instanciée (c'est le jargon Java…) dans l'activité principale, cette classe comporte les déclarations des variables et méthodes nécessaires au dessin des ECG. On peut citer en particulier :

- Les 6 tableaux utilisés pour mémoriser 10 min de tracés ECG
- La méthode *onDraw()* appelée périodiquement par le thread *ThreadGrapheYT*, chargée de dessiner les ECG choisis, ainsi que les axes (**fig. 17** dans l'encadré).

Figure 15.
Les fonctions du menu : connexion BT, quitter, sauver et relire les ECG, effacer la mémoire ECG.

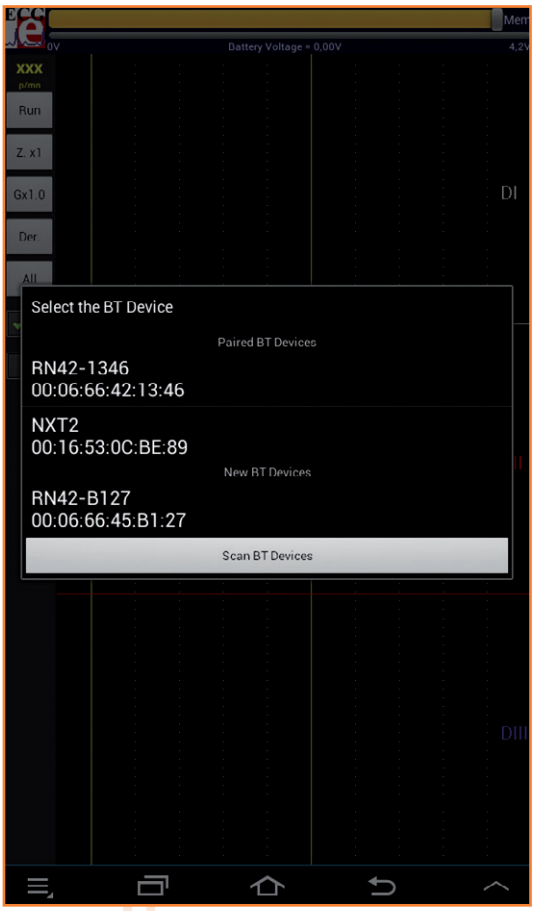


Figure 16.
L'activité **BtListActivity** affiche la liste les périphériques déjà appariés et recherche sur demande de nouveaux périphériques à portée.

Algorithme de rafraîchissement déroulant des graphes ECG

Pour bien comprendre l'algorithme, il faut avoir à l'esprit l'usage des tableaux mémorisant les 10 dernières minutes d'activité cardiaque :

- chacune des dérivations DI, DII, DIII, aVR, aVL et aVF comporte son propre tableau de 10 min d'échantillons ;
- chacun est affecté par un nouvel échantillon ECG à chaque trame de données reçue par le module BT, soit 250 fois par seconde ;
- en usage normal (curseur Mem à droite), le dernier échantillon acquis doit toujours être affiché à l'extrême droite de l'écran ;
- pour obtenir un graphe déroulant dynamique, la fonction d'affichage (*onDraw*) représente les derniers échantillons mémorisés dans les tableaux, en commençant par le dernier. On remonte le temps en quelque sorte.

La vitesse de défilement est ainsi de 250 pixels par seconde (Zoom x1).

Quel est le travail demandé aux processeurs du terminal Android pour afficher un ECG déroulant ? Dans cet exemple, la taille du graphe ECG est de 722 x 403 pixels. Dans ces conditions, à chaque appel de la méthode *onDraw*, il faut :

- effacer l'ensemble de l'écran, soit les 722 x 403 = 290.966 pixels,
- dessiner les noms des dérivations,
- dessiner les axes qui se déplacent avec les courbes,
- dessiner jusqu'à trois ECG, soit pour chacun 722 segments de droite,
- calculer le rythme cardiaque et l'afficher !

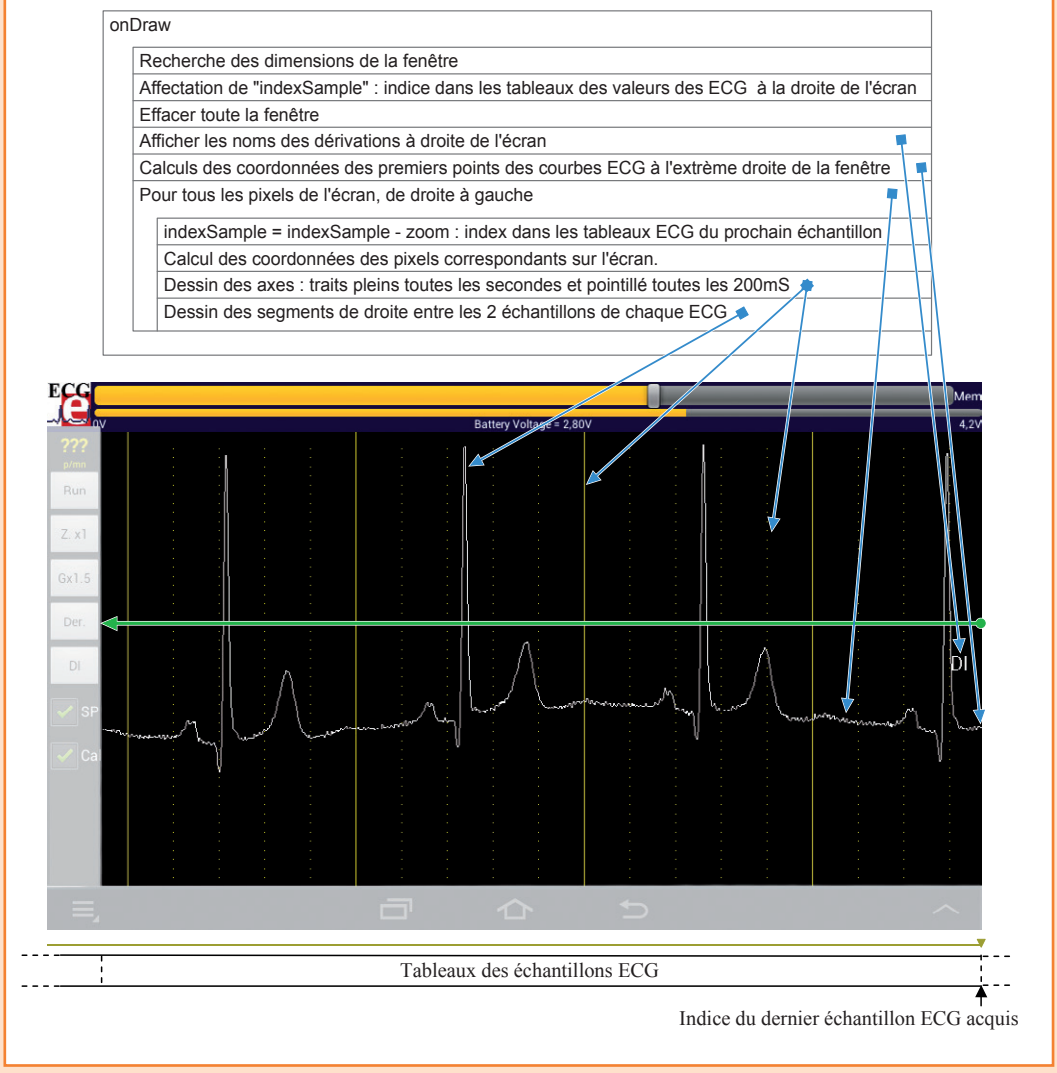


Figure 17. L'algorithme de rafraichissement déroulant du graphe.

Rien que ça… Le nombre d'instructions exécutées par le processeur, aidé dans certains cas par son coprocesseur graphique, est gigantesque. De plus, pour obtenir un défilement fluide, la fréquence d'appel de la méthode *onDraw* doit être largement supérieure à 10 Hz ! Il y a encore quelques années, un gros PC de bureau était incapable de soutenir un tel rythme de travail. Aujourd'hui, une de ces petites merveilles qui tiennent dans la poche y arrive facilement tout en s'occupant des autres applications actives…

ThreadGrapheYT démarre à la création de la classe *GrapheYT*, donc au lancement de l'application. Il comporte dans sa méthode *run* l'appel de la méthode *onDraw* citée ci-dessus. Une priorité élevée lui est donnée pour obtenir une bonne fluidité des graphes déroulants. Toutefois, la fréquence d'exécution de sa méthode *run* est déterminée par le système Android. Si d'autres *threads* sollicitent fortement le CPU du terminal, le défilement des graphes peut devenir saccadé.

Maman, maman, je n'ai rien au cœur !

Timer1Service : Cette classe crée un service qui exécute toutes les secondes une tâche relativement simple : afficher la tension de la pile de l'interface sous forme numérique et graphique (affichée en haut de l'écran).

FileListActivity : Cette activité est créée quand l'utilisateur choisit de sauvegarder ou de lire des relevés d'ECG depuis le menu. Une nouvelle fenêtre affiche la liste des fichiers existants ainsi qu'un cadre pour éditer le nom d'un nouveau fichier (**fig. 18**). La fenêtre et l'activité se ferment au choix du fichier après l'envoi vers l'activité principale d'un message comprenant son nom et la nature de l'opération (*save* ou *load*). L'activité principale réalise alors l'opération demandée.

J'ai le cœur qui fait boum

Sans avoir épuisé le sujet, nous sommes au terme de la description de l'elektorcardioscope. Le mois prochain nous passerons enfin à la pratique avec la réalisation, les réglages et un mode d'emploi. Pour ce qui est de l'interface, ce sera vite fait puisque le circuit, parfaitement au point, est d'ores et déjà disponible auprès de notre service ***elektorPCBservice*** sous la forme d'un module assemblé, prêt à l'emploi [5]. Les réglages ne demanderont pas non plus d'expertise particulière, mais nous ne manquerons pas de nous intéresser aux électrodes. Car la vocation de cet appareil conçu et construit avec soin est bel et bien de mettre l'ECG à la portée de tous.

(130227)

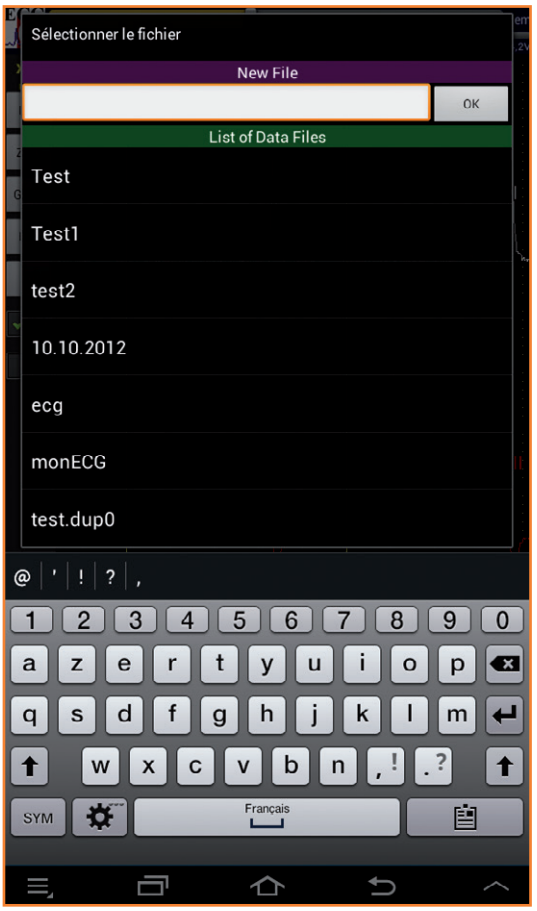


Figure 18.
Choix du fichier pour la sauvegarde ou la lecture des relevés d'ECG.

Liens

[3] www.elektor.fr/120107 et www.elektor.fr/130227

[4] Android | Apprendre à programmer des applis de Stephan Schwark www.elektor.fr/android.

[5] Le Site du Zéro www.siteduzero.com/informatique/tutoriels/apprenez-a-programmer-en-java ou http://goo.gl/OVZQY

[6] site de l'auteur http://electronique.marcel.free.fr/

[7] www.elektorpcbservice.com/

chaîne d'outils et API C/C++

pour cartes d'extension Linux et *Raspberry Pi*

Benedikt Sauter [1]

Le module à 8 relais présenté dans le dernier numéro fait partie d'une petite ménagerie de cartes d'extension pour la carte Linux Elektor, pour *Raspberry Pi* ou d'autres cartes à microcontrôleur. Nous les passerons en revue puis découvrirons les outils en ligne de commande et l'API C/C++ que l'équipe *d'embedded projects* a développés pour simplifier leur pilotage.

Dès lors que tout le monde s'accorde sur une spécification de connecteur, il devient possible de combiner entre elles des cartes à µC et des cartes d'extension. Nous avons déjà mentionné le connecteur Gnublin à 14 points de la carte Linux Elektor [2]. Sous le nom d'Embedded Extension Connector, il équipe également la carte Xmega Webserver que nous mettrons en vedette prochainement. Nous allons ici présenter d'autres cartes, mais surtout montrer combien il est facile de dialoguer avec ces extensions depuis Linux. L'équipe des développeurs a concocté des outils en ligne de commande qui permettent de tester rapidement les fonctions de chaque module. Une API C/C++ complète a également été conçue pour simplifier l'écriture d'applications [3]. Elle repose sur les pilotes standard de périphériques pour I²C, SPI, GPIO etc., et permet à l'électronicien de piloter son matériel (p. ex. un moteur pas à pas) à l'aide d'appels de fonctions. Fini donc l'étude laborieuse du code des pilotes du noyau (**fig. 1**). Notez que l'équipe a aussi développé une API Python [4].

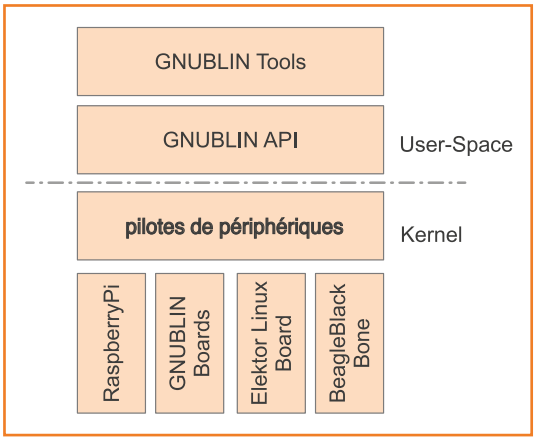


Figure 1.
L'API Gnublin évite d'avoir à se frotter aux pilotes SPI, I²C et autres pilotes de périphériques.

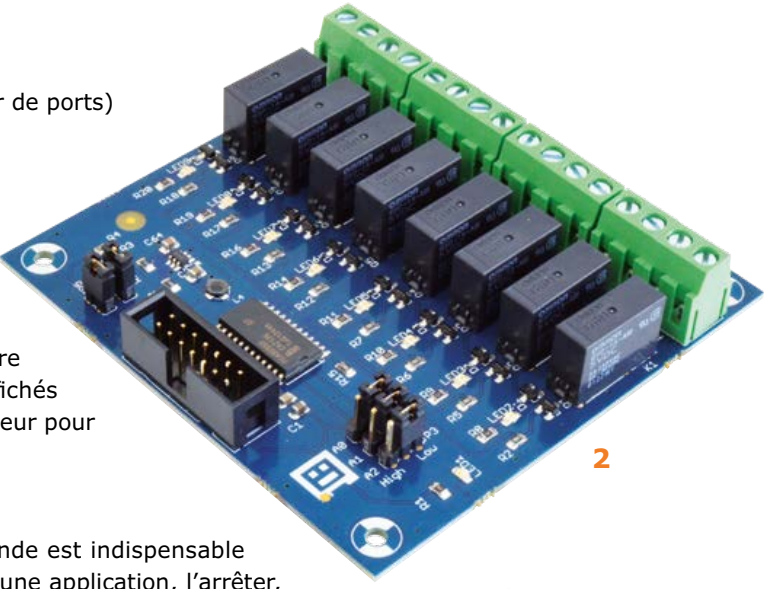
Le concept

Linux représente une excellente couche d'abstraction pour l'écriture d'applications indépendantes du processeur utilisé : on les développe simplement « pour Linux ». Grâce aux nouvelles cartes d'extension, disponibles depuis www.elektor.fr/gnublin, ce principe s'applique désormais aussi aux circuits qui utilisent des moteurs, des afficheurs, des capteurs de température et autres relais. Les modules suivants se connectent simplement à la carte Linux Elektor via un câble en nappe :

- 8 relais (130212-91, fig. 2)
- Capteur de température (130212-95, fig. 3)
- Afficheur LCD 4x20 caractères (130212-92, fig. 4)
- Pilote de moteur pas à pas (130212-93, fig. 5)

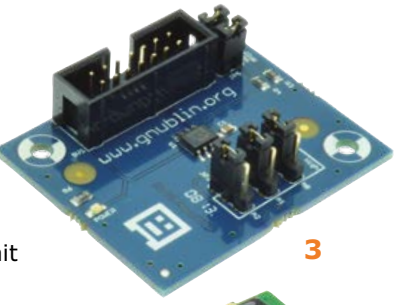
- IO-Expander (16 E/S numériques) (130212-94, fig. 6)
- Extension (afficheur, poussoir, RTC, bipeur et duplicateur de ports) (120596-91, fig. 7)

La carte de répartition Bridge Module (disponible chez Elektor : 130212-71) permet de relier plusieurs modules à la carte Linux Elektor (**fig. 8**).
N'oublions pas non plus l'adaptateur *Raspberry Pi* (130212-72), déjà mentionné dans le dernier numéro, et qui une fois enfiché sur le très tendance *Raspberry Pi* offre une rangée de connecteurs sur lesquels peuvent être enfichés d'autres modules. Dernière nouveauté, un module adaptateur pour la carte BeagleBone Black (130212-74).


2

Outils en ligne de commande

Véritable couteau suisse du pingouin, la ligne de commande est indispensable lorsqu'on travaille avec une carte Linux. Elle sert à lancer une application, l'arrêter, administrer Linux, lire les messages du système...
L'équipe d'embeded projects a donc créé une trousse à outils pour les modules Gnublin et pour certaines fonctions internes de la carte Linux. Pour voir la liste de ces mini-programmes, entrez `gnublin-` sur la ligne de commande et appuyez sur la touche de tabulation. Le **tableau 1** en présente quelques-uns.
Ces outils en ligne de commande sont pratiques pour effectuer un premier test, p. ex. pour s'assurer que le matériel est connecté correctement : qui n'a pas désespérément cherché une erreur dans son code avant de réaliser que son matériel n'était pas alimenté ? On a tous vécu ça !



3

L'API C/C++

Un module une fois connecté et testé, on peut passer à l'écriture du programme. Ici pas de pointeurs ou autres structures complexes à dompter, l'API C/C++ fournit des appels de fonctions simples et explicites. Chaque carte d'interface et d'extension possède son propre module logiciel (**tableau 2**). Lorsque vous utilisez la carte Linux Elektor, il suffit d'inclure le fichier d'en-tête gnublin.h, comme le montre le **listage 1**. La page du wiki de Gnublin [3] propose de nombreux autres exemples de code. Vous pouvez étudier la dernière version du code source de l'API sur la page GitHub associée [5]. Le wiki Gnublin [3] explique également comment installer un environnement de développement afin de pouvoir utiliser l'API.


4

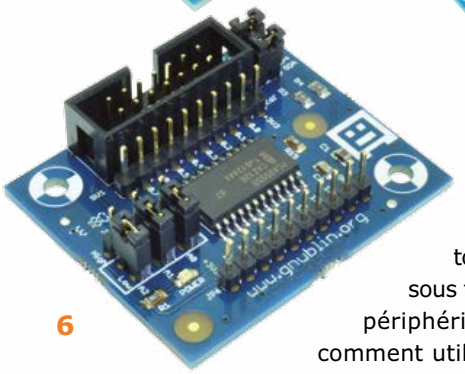
Installer les outils et l'API

L'API peut en principe être utilisée avec toute carte Linux qui dispose de pilotes pour I²C et SPI. Comme la plupart des processeurs incluent ces interfaces, leur accès est presque

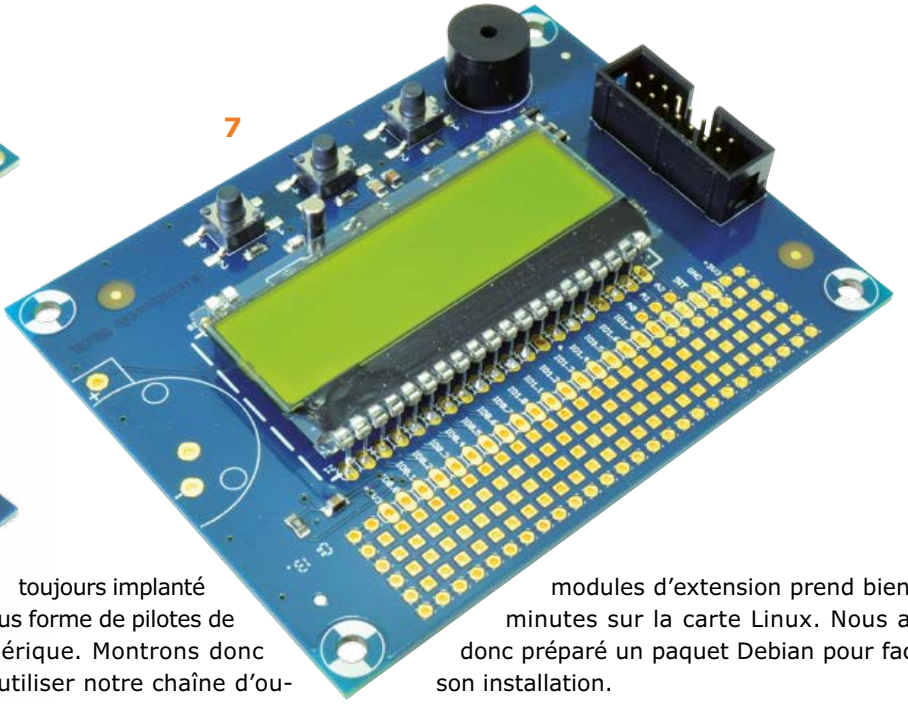
Tableau 1. Les outils Gnublin en ligne de commande (extrait).

outil	exemple de commande	description
gnublin-lm75	appel sans commande	sortie de la température
gnublin-relay	gnublin-relay -p 1- o 1	enclenche le relais 1
gnublin-adcint	gnublin-adcint -c 1	interroge le CAN interne, canal 1
gnublin-step	gnublin-step -p 3000	moteur pas à pas sur position 3000

5



6



7

toujours implanté sous forme de pilotes de périphérique. Montrons donc comment utiliser notre chaîne d'outils et l'API avec la carte Linux Elektor et le *Raspberry Pi*.

La carte Linux Elektor

La carte Linux Elektor est dotée d'un connecteur enfichable depuis la première génération. La marque rouge du câble en nappe doit être du même côté que le port GPA0 (l'ergot de repérage du câble doit être orienté côté carte).
Nous avions d'abord livré les premières versions de la carte avec un système de fichiers ELDK, mais finalement nous avons opté pour une image Debian. Le lien [6] explique comment mettre à jour une « vieille » carte mémoire.
La compilation de la chaîne d'outils pour les modules d'extension prend bien cinq minutes sur la carte Linux. Nous avons donc préparé un paquet Debian pour faciliter son installation.

Commencez par le télécharger depuis un PC, puis servez-vous d'un lecteur de cartes pour l'enregistrer sur la carte SD.
Vous pouvez aussi vous servir de la ligne de commande pour le télécharger si votre carte est connectée au réseau :

```
wget https://github.com/embeddedprojects/
gnublin-api/raw/master/gnublin-
tools_0.1-1_armel.deb
```

Pour ensuite installer le paquet .deb, entrez :

```
root@gnublin:~# dpkg -i gnublin-
tools_0.1-1_armel.deb
```

Si un jour vous souhaitez supprimer ce paquet :

```
root@gnublin:~# dpkg -r gnublin-tools
```

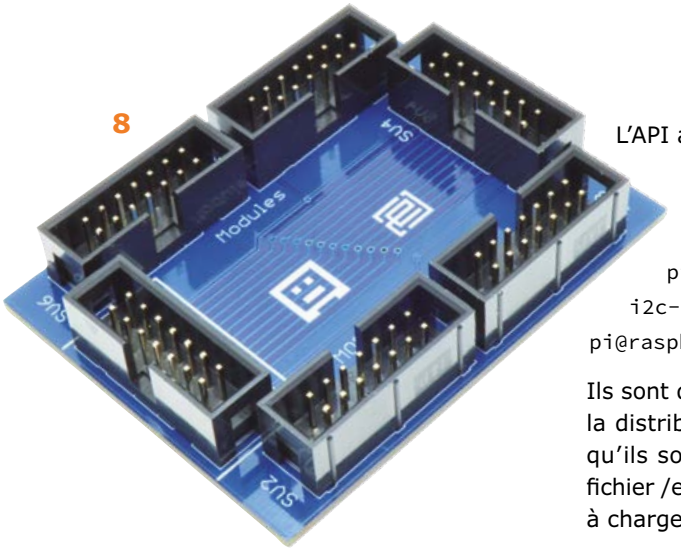
Listage 1. Commande d'un composant I²C.

```
#define BOARD_GNUBLIN
//#define BOARD_RASPBERRYPI

#include “gnublin.h”

int main()
{
   gnublin_adc ad;

   while(1){
        printf(“AD value %i \n”,ad.getValue(1));
   }
}
```

Raspberry PI

Pour installer les modules logiciels sur le *Raspberry Pi*, le plus simple est de se connecter directement au dépôt du code source. Le *Raspberry Pi* doit être sous tension et relié à l'internet. Vous pourrez alors cloner le dépôt avec le programme Git. Si Git n'est pas installé sur votre Pi, servez-vous d'apt-get :

```
pi@raspberrypi ~ $ sudo apt-get install
git
```



Pour récupérer le dépôt :

```
pi@raspberrypi ~ $ git clone https://
github.com/embeddedprojects/gnublin-api.
git
```

Placez-vous ensuite dans le répertoire gnublin-api :

```
pi@raspberrypi ~ $ cd gnublin-api
```

Saisissez la commande suivante pour compiler puis installer le programme, les exemples et l'API :

```
pi@raspberrypi ~ $ make && sudo make
install
```

L'API a besoin des pilotes suivants :

```
pi@raspberrypi ~ $ sudo modprobe
spi-bcm2708
pi@raspberrypi ~ $ sudo modprobe
i2c-bcm2708
pi@raspberrypi ~ $ sudo modprobe i2c-dev
```

Ils sont déjà présents sur la version actuelle de la distribution *Raspberry Pi*. Si vous souhaitez qu'ils soient chargés au démarrage, éditez le fichier /etc/modules et spécifiez le(s) module(s) à charger (un module par ligne) :

```
spi-bcm2708
i2c-bcm2708
i2c-dev
```

Avec la chaîne d'outils, vous pourrez tester une carte d'extension sitôt la carte connectée.

(130212 – version française : Hervé Moreau)

Liens

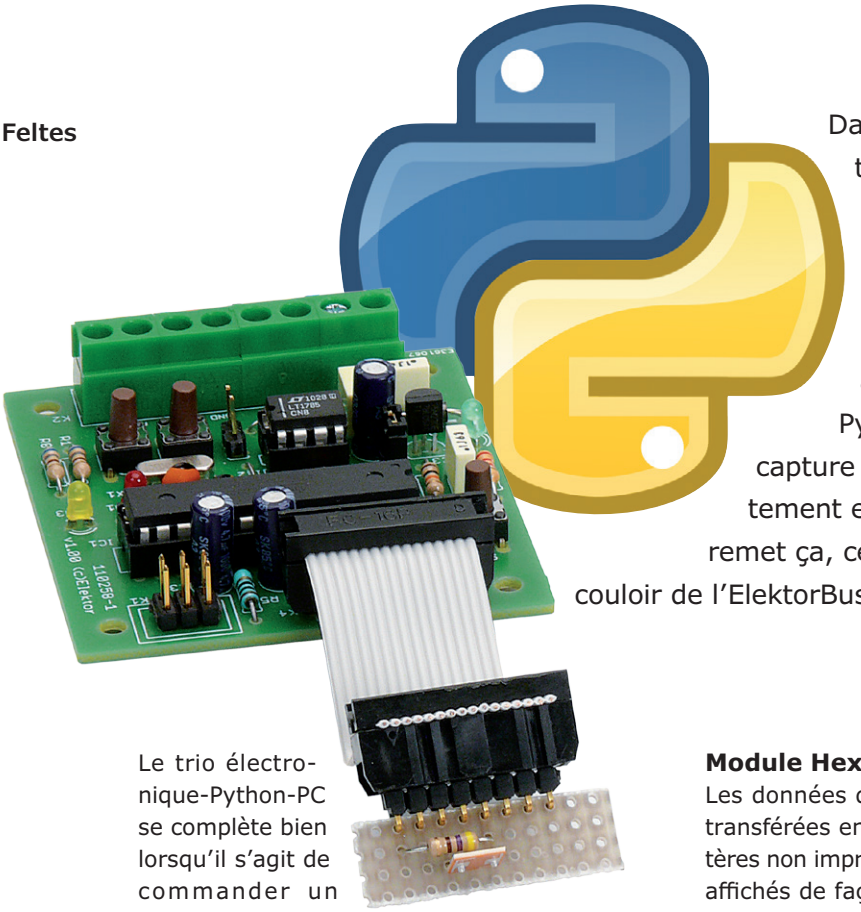
[1] sauter@embedded-projects.net
[2] www.elektor.fr/130157
[3] http://wiki.gnublin.org/index.php/API
[4] http://en.gnublin.org/index.php/API_Python
[5] https://gitbhub.com/embeddedprojects/gnublin-api
[6] http://en.gnublin.org/index.php/GNUBLIN-Elektor

Tableau 2. Classes de l'API (extrait).

module	interface	remarque
gnublin_gpio	interne	
gnublin_adc	interne	actuellement uniquement sur la carte Linux Elektor (pas sur le *Raspberry Pi*)
gnublin_i2c	I²C	bus I²C standard
gnublin_spi	SPI	périphériques SPI standard
gnublin_pwm	interne	actuellement uniquement sur la carte Linux Elektor
gnublin_module_lm75	I²C	capteur de température
gnublin_module_relay	I²C	carte à relais
gnublin_module_pca9555	I²C	duplicateur de ports avec 16 E/S numériques
gnublin_module_step	I²C	moteur pas à pas
gnublin_module_lcd	I²C	afficheur 4x20 caractères

de BASIC à Python (3)
communiquer avec l'ElektorBus

Jean-Claude Feltes
(Luxembourg)



Dans les deux premières parties, nous avions découvert le langage Python, apprécié ses aptitudes mathématiques, et utilisé ses bibliothèques graphiques. Nous avions surtout montré que Python se prêtait bien à la capture des données, à leur traitement et à leur visualisation. On remet ça, cette fois en empruntant le couloir de l'ElektorBus.

Le trio électronique-Python-PC se complète bien lorsqu'il s'agit de commander un circuit à l'aide des données envoyées depuis un PC. Il se montre toutefois encore plus efficace lorsqu'il s'agit de transférer et visualiser sur PC des données en provenance de l'extérieur. Pour communiquer avec un matériel externe, nous pouvons utiliser une des interfaces série habituelles ou, lorsque les opérations en jeu sont plus complexes, un bus. C'est ce que nous ferons ici pour récupérer les données d'une carte, que nous représenterons ensuite sur une interface graphique.

Comme bus, nous avons choisi l'ElektorBus, passé onze fois sous vos yeux entre janvier 2011 et janvier 2012 [1]. Le matériel utilisé sera celui de l'article *Le bus arrive ! (6)* [2], c'est-à-dire un nœud expérimental sous forme de carte avec microcontrôleur, touches et LED. Le nœud se relie au PC à l'aide du convertisseur USB/RS-485 (**fig. 1**).

Module Hexfunctions

Les données qui passent par l'ElektorBus sont transférées en mode binaire, et certains caractères non imprimables ne pourront donc pas être affichés de façon optimale dans une fenêtre de terminal. C'est pourquoi nous les convertirons en hexadécimal à l'aide de la fonction *translate-2hex()* enregistrée dans le module *Hexfunctions.py* (**listage 1**).

Stocker une fonction dans un module extérieur est pratique, cela évite d'avoir à la redéfinir à chaque utilisation dans un programme. Autre avantage des modules, un code plus court, et donc plus lisible.

Hexfunctions.py contient aussi, à la mode Python, une fonction « de test ». Elle débute par la ligne : *if __name__ == «__main__»*. Si le module est importé depuis un programme quelconque, c'est-à-dire si *Hexfunctions.py* n'est pas le programme *main*, alors les lignes qui suivent cette instruction *if* ne sont pas exécutées. Si par contre *Hexfunctions.py* est lancé directement depuis une console, la variable *name* prend la valeur *main*, et le code de la condition *if* est exécuté :

HELLO
48 45 4C 4C 4F 0A

Avec cette ligne, un fichier Python peut donc servir à la fois de programme autonome ou de module réutilisable. Pour utiliser la fonction *translate2hex()* depuis p. ex. *Test_hexfunctions.py*, nous écririons :

```
from hexfunctions import *
s="HELLO\n"
print s, translate2hex(s)
```

Un module importé doit être situé dans le même répertoire que celui du programme principal, ou dans les chemins de recherche de Python.

Interfaces graphiques avec wxPython

Il existe de nombreuses façons de créer une interface graphique attrayante. En Python, aucune n'est, hélas, aussi pratique que l'utilisation des objets *Form* de Visual Basic. Les outils graphiques de Python permettent néanmoins un plein contrôle du résultat final, et avec de l'entraînement on parvient toujours à ses fins. J'ai fait mes premiers essais avec l'éditeur *Boa Constructor*. J'ai pu créer des fenêtres graphiques sans être un spécialiste, mais leur débogage était assez laborieux. *Python Card* m'a semblé plus simple, très pratique pour qui ne se soucie pas des détails, mais doit hélas être installé sur l'ordinateur cible.

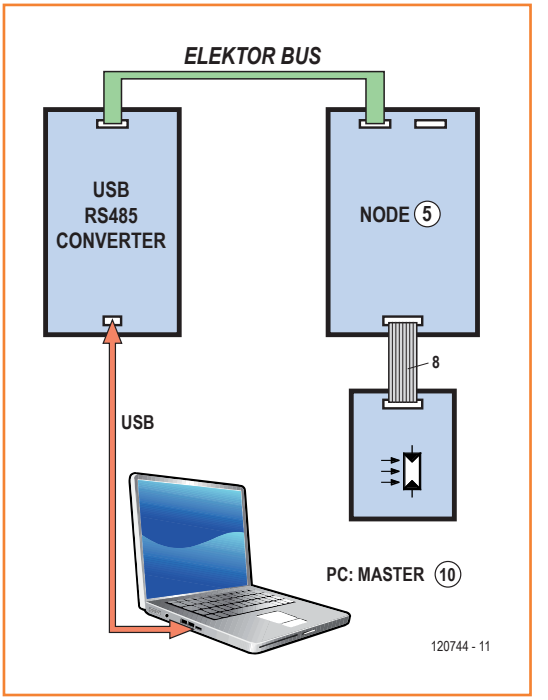


Figure 1.
Le PC reçoit via l'ElektorBus les données d'une petite carte à laquelle est reliée une photorésistance.

Tkinter est la bibliothèque graphique fournie par défaut avec Python. *Tkinter* est assez simple, mais dispose de moins d'objets que *wxPython*. Ce qu'il me manquait surtout, c'était un accès au presse-papier, donc j'ai fini par me tourner vers *wxPython*.
Nous allons construire pas à pas notre interface graphique en nous appuyant sur le code du **lis-**

Listage 1 : Hexfunctions.py

```
def translate2hex(c):
    """ translate character string c to hex representation string
    e.g ABC -> 41 42 43 """

    h=""
    for ch in c:                # iterate over all characters
        b=hex(ord(ch))          # get hex value
        b=b.replace("0x","")    # take away leading "0x for better overview"
        b=b.upper()             # all in upper characters
        if len(b)<=1:
            b="0"+b             # e.g. make "0A" out of "A"
        h=h+b+" "               # separate bytes by space

    return h

# test:
if __name__ == "__main__":
    s="HELLO\n"
    print s, translate2hex(s)
```

Listage 2 : GUI_template.py

```
import wx

#  GUI
class MyFrame(wx.Frame):

    def __init__(self, **kwargs):
        # create frame
        wx.Frame.__init__(self, None, **kwargs)

        # text box with fixed width font for nice data representation
        self.textbox=wx.TextCtrl(self, style = wx.TE_MULTILINE,
            pos = (5,5),size=(300, 200))
        myfont = wx.Font(12, wx.MODERN, wx.NORMAL, wx.BOLD, False, u'Courier')
        self.textbox.SetFont(myfont)

        self.button=wx.Button(self, -1, "TEST", pos=(100,230))

        # Bindings
        self.Bind(wx.EVT_IDLE, self.OnIdle)
        self.Bind(wx.EVT_WINDOW_DESTROY, self.OnDestroy)
```

Listage 3 : la classe Serialthread

```
class Serialthread(serial.Serial):
    def __init__(self, port, baud, **kwargs):
        # Initialization of port + baudrate
        serial.Serial.__init__(self)
        self.sCOM =serial.Serial(port)
        self.sCOM.setBaudrate(baud)

        # open port if not already open
        if self.sCOM.isOpen()==False:
            self.sCOM.open()
        if self.sCOM.isOpen()==True:
            print „connected to“, self.sCOM.port
        else:
            print „Error opening port“

        # Counter for received data blocks
        self.ctr=0

        # Create stop event (to terminate endless receiving loop)
        # and message queue for thread (to transmit received text to TextCtrl)
        self.stopevent=threading.Event()
        self.msgQueue=Queue.Queue()

    def disconnect(self):
        # set stop event so endless receiving loop can be interrupted
        self.stopevent.set()

    def connect(self):
        # create a new thread object that runs serial thread
```

```
        self.Bind(wx.EVT_BUTTON, self.OnButton)

    def OnIdle( self, event):
        # if nothing else to do, update text from message queue
        pass

    def  OnDestroy(self, event):
        print “Exit”

    def OnButton(self, event):
        self.textbox.AppendText (“Button pressed\n”)
#---------------------------------------------------------------------

# Main program
if __name__ == “__main__”:
    app = wx.App(redirect = False)
    frame = MyFrame(title=”GUI”, size = (320,270))
    frame.Show(True)
    frame.Centre()
    app.MainLoop()
```

```
        # to read serial characters
        self.serialthread = threading.Thread(target=self.readSerial)

        # clear stopevent and Connect thread
        self.stopevent.clear()
        self.serialthread.start()

    def readSerial(self):
        # endless receiving loop
        while not self.stopevent.isSet():
            data=““

            # read from port
            c = self.sCOM.read(1)

            # synchronize
            if ord(c) == 0xAA:
                self.ctr += 1
                rest = self.sCOM.read(15)
                data=c+rest

            # format c to 16 bytes output
            datastring=str(self.ctr) + „\t“ + translate2hex(data) + „\n“

            # update message queue
            self.msgQueue.put(datastring)
            wx.WakeUpIdle()         # wake up to update text

        # end serial thread
        print „disconnected“
        self.sCOM.close()
```

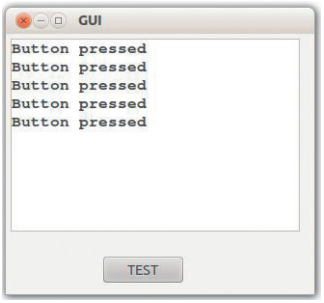


Figure 2.
La fenêtre créée par le code du listage 2.

tage 2, mais avant toute chose vous devez installer la bibliothèque *wxPython*.

wx.Frame est la classe de base pour les fenêtres standard. On commence donc par construire une classe *MyFrame* qui en dérive, et dont l'instance sera notre fenêtre principale. Notre classe *MyFrame* hérite de toutes les propriétés de la classe *Frame :* fonctions d'agrandissement, de déplacement, etc. Les propriétés de la fenêtre sont définies dans la fonction __ *init*__. Y sont en particulier créés deux éléments graphiques (*widgets*), une zone de texte et un bouton. Les dimensions et la position de la fenêtre ont été prédéfinies par souci de simplicité, mais nous aurions pu recourir aux mécanismes de placement et de dimensionnement automatiques que sont les *sizers*.

Pour finir, trois gestionnaires d'événements sont créés. L'usage veut que les méthodes associées à ces événements soient préfixées par *On*, p. ex. *OnButton* pour la méthode qui est appelée par un clic du bouton. Événements et méthodes sont liés à l'aide de *Bind()*.
La classe *MyFrame* peut maintenant être utilisée dans le programme principal. Il crée d'abord un objet *wx.App*, qui pour l'essentiel s'occupe de la gestion des événements. La fenêtre est ensuite instanciée, affichée, puis centrée. Les événements sont traités dans la boucle sans fin *app.MainLoop*.
La fenêtre qui apparaît au lancement du programme (**fig. 2**) réagit bien aux clics sur le bouton. Un clic droit ouvre en outre un menu d'édition pour la sélection, la copie, la suppression et le collage de texte.
Ajouter d'autres fonctions à ce programme ne serait pas compliqué, p. ex. pour la sauvegarde du texte dans un fichier.

ElektorBus : lecture

Le nœud expérimental se relie au PC avec le convertisseur USB/RS-485. Équipez la carte avec un microcontrôleur ATmega328, que vous programmerez avec le fichier hex à télécharger [4]. Reliez par exemple un potentiomètre ou une photorésistance au connecteur d'extension ADC0 du nœud expérimental (cf [2a] et [2b]).
Il faut ensuite identifier le port série utilisé. Sous Windows, il suffit de regarder quel nouveau port COM apparaît dans le gestionnaire de périphériques après l'insertion du convertisseur. Sous Linux, on peut passer par la ligne de commande :

```
ls /dev/tt*U*
```

On obtiendra par exemple :

```
/dev/ttyUSB0
```

Vous pouvez aussi utiliser le script *ScanSerial.py* de la partie 1 [3] (le listage imprimé dans le magazine contenait une erreur d'indentation, mais la version à télécharger est correcte.).
Si l'un des scripts ci-dessous ne fonctionne pas, assurez-vous d'abord d'avoir indiqué le bon port. Lors d'essais, il peut arriver que le système d'exploitation change le numéro de port « dans votre dos ». Cela arrive lorsque vous lancez un script avec *ttyUSB0*, que vous retirez la clé pendant son exécution, puis que vous la réinsérez. *ttyUSB1* est alors affecté au convertisseur, et le script ne peut plus recevoir de données. Une situation toutefois exceptionnelle.
Nous voici prêts à compléter le fichier modèle *GUI_template.py*. Commençons par importer quelques modules et paramétrer l'interface série :

```
COMport = "/dev/ttyUSB0" # à adapter!
Baud = 9600

import threading, Queue
import serial
import time
```

Nous avons besoin du module *threading,* car la réception série doit se faire dans un *thread* séparé : la réception nécessite en effet une boucle sans fin, et cette boucle entrerait en conflit avec la boucle principale de *wx* si elle n'était pas séparée. Il s'agit de la partie la plus complexe de la programmation.

Pour accéder à l'interface, nous créons la classe *Serialthread* (**listage 3**). Un objet de cette classe ouvre l'interface, lit les données entrantes, les formate, puis les envoie aux autres parties du programme via une file d'attente des messages. L'opération se déroule à l'intérieur d'une boucle sans fin indépendante, jusqu'à ce que le thread soit stoppé.

Un objet *Serialthread* hérite de toutes les propriétés et méthodes de la classe de base *Serial* : fonctions d'écriture et de lecture, nom de port, débit, etc. L'objet *Serial* créé dans la méthode *__init__* gère toutes les opérations relatives au port, en particulier son ouverture en cas de besoin, ainsi que le paramétrage du débit.
Le compteur *self.ctr* n'est pas important, il ne servira qu'à numéroter les blocs de données.
Sont ensuite créés deux objets utilisés par le thread : *stopevent*, pour quitter le thread, et *msgQueue*, la file d'attente des messages pour le transfert des données à l'interface graphique.

Depuis l'extérieur, le thread se lance et s'arrête à l'aide des méthodes *connect* et *disconnect*. La méthode *disconnect* ne fait qu'interrompre la boucle sans fin de réception. La méthode *connect* lance un nouveau thread qui exécute la fonction *readSerial*. Celle-ci lit les octets de l'interface série à l'intérieur d'une boucle sans fin, jusqu'à l'appel du gestionnaire d'arrêt (*stop event*). Le thread est alors arrêté et le port fermé.
La fonction *readSerial* contient un mécanisme de synchronisation des données. Chaque paquet de données de l'ElektorBus commence par 0xAA. Si cette valeur est lue, le compteur de paquets est incrémenté, puis 15 nouveaux octets de données sont lus. Ces octets sont formatés puis placés dans la file d'attente des messages en tant que chaînes. La méthode *wx.WakeUpIdle()* signale à la partie GUI du programme que des données sont à lire dans la file d'attente.

La classe *Serialthread* permet de lire des données en continu sans perturber le déroulement du reste du programme : la classe est quasiment exécutée en parallèle des autres parties du programme. Pour rendre cela possible, un objet de la classe *Serialthread* doit être créé puis lancé à l'intérieur du programme principal. Complétons donc le code de *MyFrame* (la classe qui a servi à créer la fenêtre principale) en ajoutant les lignes suivantes à la méthode *__init__* :

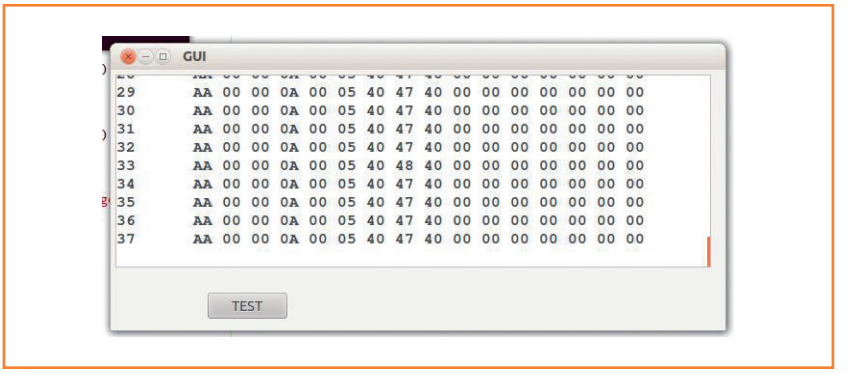


Figure 3.
L'affichage au format hexadécimal des données reçues.

```
class MyFrame(wx.Frame):

    def __init__(self, **kwargs):
        ...

        # Bindings
        ...

        # serial thread
        self.serialreceive =
Serialthread(COMport, Baud)
        self.serialreceive.connect()
```

Nous avons maintenant un objet *serialreceive* relié au port défini par la variable *COMport* et qui écrit les données entrantes dans la file d'attente. Pour rendre l'opération visible, le texte doit être extrait de la file, puis écrit dans la zone de texte. Pour cela nous faisons de nouveau appel à *OnIdle()* :

```
class MyFrame(wx.Frame):

    def __init__(self, **kwargs):
        ...

    def OnIdle( self, event):
        # if nothing else to do, update
text from message queue
        while not self.serialreceive.
msgQueue.empty():
            msg=self.serialreceive.
msgQueue.get()
            self.textbox.AppendText(msg)
```

Le mot-clé *pass* que nous avions utilisé (il désigne une opération nulle) ne servait que de garde-place. Avec le nouveau code, les données entrantes sont numérotées et affichées comme sur la **figure 3**.
Pour éviter l'apparition d'un message d'erreur à

la fermeture de la fenêtre, nous modifions également la méthode *OnDestroy* :

```
def OnDestroy(self, event):
    self.serialreceive.disconnect()
    time.sleep(1)
```

Grâce à ces lignes, le thread sera proprement arrêté avant la fermeture de la fenêtre. Comme le processus peut prendre un certain temps, nous avons inséré une pause avec *time.sleep(1)*. L'ensemble de ces modifications forme un nouveau programme, que nous avons appelé *Serialreceive1.py* et que vous pouvez télécharger [4].
Une fois la classe *Serialthread* entièrement définie, une bonne idée est de l'enregistrer sous forme de module (*Serialthread.py*). Nous pourrons alors l'importer depuis le programme principal et ainsi garder un code propre et lisible.
Avant de l'enregistrer, nous devons toutefois encore y ajouter les lignes qui importeront les modules que lui-même utilise :

```
import threading, Queue
import serial
import time
from hexfunctions import *
import wx

class Serialthread(serial.Serial):
    def __init__(self, port, baud,
**kwargs):
        # Initialization of port +
baudrate
        serial.Serial.__init__(self)
        ....
        # end serial thread
        print “disconnected”
        self.sCOM.close()
```

Dans le nouveau programme principal *Serialreceive2.py* qui importe ce module, la définition de la classe *Serialthread* peut bien sûr être omise, de même que sont inutiles les lignes qui importent les modules utilisés par *Serialthread.py*.
Même si nous nous en contenterons ici, cette séparation n'est à vrai dire pas entièrement satisfaisante, puisque le module *Serialthread.py* dépend de la nature de ce projet et n'est pas d'application universelle.

ElektorBus : écriture

Le nœud expérimental est équipé d'une LED rouge. Essayons de l'allumer et de l'éteindre depuis le PC à l'aide de deux boutons. Pour cela nous devons envoyer deux séquences d'octets via l'ElektorBus (cf. [1] pour les spécifications du bus) :
• sous tension : AA 00 00 05 00 0A 00 00 00 00 **60 01** 00 00 00 00
• hors tension : AA 00 00 05 00 0A 00 00 00 00 **60 00** 00 00 00 00

Commençons par ajouter un second bouton à notre programme, ainsi qu'un autre gestionnaire d'événement. N'oublions pas non plus d'attribuer des noms distincts aux boutons. Voici les modifications à apporter à la méthode *__init__* de la classe *MyFrame* :

```
    def __init__(self, **kwargs):
        # create frame
        ....
        buttonOn=wx.Button(self, -1, “LED
ON”, pos=(100,230))
        buttonOff=wx.Button(self, -1,
“LED OFF”, pos=(200,230))

        # Bindings
        ....
        buttonOn.Bind(wx.EVT_BUTTON,
self.OnButtonOn)
        buttonOff.Bind(wx.EVT_BUTTON,
self.OnButtonOff)
```

Le gestionnaire d'événement appelle les fonctions *self.OnButtonOn* et *self.OnButtonOff*. Complétons-les :

```
    def OnButtonOn(self, event):
        self.textbox.AppendText (“LED
ON\n”)
        data=b”\xAA\x00\x00\x05\x00\x0A\
x00\x00\x00\x00\x60\x01\x00\x00\x00\x00”
        self.serial_thread.sCOM.
write(data)

    def OnButtonOff(self, event):
        self.textbox.AppendText (“LED
OFF\n”)
        data=b”\xAA\x00\x00\x05\x00\x0A\
x00\x00\x00\x00\x60\x00\x00\x00\x00\x00”
        self.serial_thread.sCOM.
write(data)
```

Elles se servent d'une fonction *write* qui ne semble définie nulle part. L'explication est simple, *write* est héritée de la classe de base *serial.Serial* et n'a donc pas besoin d'être définie.

Enregistrons nos modifications sous *Receive_send.py*. Nous pouvons maintenant commander la LED rouge à l'aide des deux boutons. Notre programme ne contient aucun mécanisme de synchronisation avec le nœud et n'est donc pas parfait. Si nœud et PC émettent au même moment, il se produit une collision sur le bus, et les données transmises sont incorrectes. Ces collisions peuvent être évitées par envoi des messages du PC juste avant que les données en provenance du nœud ne soient reçues (mode *DirectMode*, cf. les spécifications de l'ElektorBus [1]). Pour n'envoyer la séquence de commande que sitôt reçues les données, nous pourrions p. ex. définir un drapeau.

Visualisation

Pour rendre notre fenêtre plus attrayante, nous allons y ajouter une courbe représentant les valeurs du convertisseur A/N.
La bibliothèque *matplotlib* est le module standard pour les tracés en 2D. Installez-la si ce n'est déjà fait. Dans l'article précédent, il ne nous avait fallu que 6 lignes de code pour tracer la fonction sinus avec l'interface procédurale *pyplot* de *matplotlib*. Ici, nous ne pouvons toutefois pas bénéficier de la simplicité de *pyplot*. Pour insérer une fenêtre d'affichage à l'intérieur de la fenêtre principale, nous avons besoin de l'interface orientée objet de *matplotlib*. La documentation correspondante se trouve sur le site de *matplotlib* [5]. Attention, cette bibliothèque est si riche qu'il est facile de s'y perdre quand on débute. Vous trouverez un exemple d'insertion de graphique avec *wxPython* en [6].
Enregistrons le module *Serialthread* sous un autre nom avant de le modifier, p. ex. sous *Serialthread_diagram.py*. Les valeurs du convertisseur A/N que nous voulons représenter graphiquement sont contenues dans les octets 5 et 6 des blocs de données reçus. Nous devons donc les extraire, puis les transmettre au programme principal. Une façon de le faire est d'ajouter les tableaux *x* et *y* à l'objet *Serialthread* en tant qu'attributs. Complétons donc *__init__* comme suit :

```
class Serialthread(serial.Serial):
    def __init__(self, port, baud,
**kwargs):
        ...

        ## init arrays and timer for data
        self.x=[]
        self.y=[]
        self.starttime=time.time()
        ...
```

Nous avons créé deux tableaux vides pour stocker les valeurs *x* et *y*. L'attribut (ou variable d'instance) *starttime* sert au calcul du temps écoulé (exprimé en secondes) car nous souhaitons remplacer les numéros de paquet associés à la valeur *x* par une variable temps.
Les valeurs du CAN sont extraites dans la fonction de réception, combinées, puis stockées dans les tableaux *x* et *y* :

```
    def readSerial(self):

        # endless receiving loop
        while not self.stopevent.isSet():
            ...
            # synchronize
            if ord(c) == 0xAA:
                self.ctr += 1
                rest = self.sCOM.read(15)
                data=c+rest

                ## update x,y
                lbyte = ord(rest[6])
                hbyte = ord(rest[5]) & 7
                adc  =  lbyte + hbyte
*256
                t=time.time()-self.
starttime
                self.x.append(t)
                self.y.append( adc)
                print t,adc

                # format c to 16 bytes
output
                ....
                wx.WakeUpIdle()         #
wake up to update text
```

Avec ces transformations, le module fournit les valeurs à la fois pour leur représentation hexadécimale et graphique.

Attaquons-nous enfin au programme principal. Commençons par importer le module sous son nouveau nom :

```
from serialthread_diagram import  *
```

Pour le tracé de la courbe, nous avons également besoin des modules suivants :

Sous Python, l'indentation a valeur de syntaxe ! Le moindre décalage se traduit par un message d'erreur ou un comportement erratique du programme !

```
from matplotlib.backends.backend_wxagg
import FigureCanvasWxAgg as FCanvas
from matplotlib.figure import Figure
```

Sous *matplotlib*, la sortie graphique est gérée par un « backend ». Il existe un type de *backend* pour chaque sortie, par exemple pour l'affichage d'un graphique avec *wxPython*, ou encore pour l'impression d'une image au format PNG. La première ligne crée un objet *FCanvas* pour *wx*, sur lequel *matplotlib* pourra tracer la courbe. *FCanvas* hérite des méthodes et propriétés de *wxPanel*, p. ex. le dimensionnement automatique.
L'objet *Figure* est le conteneur dans lequel *matplotlib* dessinera, mais... *Figure* ne représente toutefois pas le tracé proprement dit. Dans un objet *Figure* peuvent être insérés un ou plusieurs objets de type *Axes*, et ce sont eux qui seront les véritables surfaces de dessin. Déroutant, certes, mais la puissance de *matplotlib* est à ce prix.
L'agencement des fenêtres nécessite encore quelques paramétrages de leurs éléments graphiques, y compris des tailles de police (voir le code ci-dessous).
La fenêtre qui accueille le tracé (**fig. 4**) est construite dans la fonction *__init__* de la classe *MyFrame* :

```
class MyFrame(wx.Frame):

  def __init__(self, **kwargs):
    # create frame
    wx.Frame.__init__(self, None, **kwargs)

      # text box with fixed width font
for nice data representation
      self.textbox=wx.TextCtrl(self,
style = wx.TE_MULTILINE,
           pos = (5,5),size=(420, 200))
       myfont = wx.Font(10, wx.MODERN,
wx.NORMAL, wx.BOLD, False, u'Courier')
       self.textbox.SetFont(myfont)

       buttonOn=wx.Button(self, -1, "LED
ON", pos=(100,230))
        buttonOff=wx.Button(self, -1,
"LED OFF", pos=(200,230))

        ## diagram
        self.figure = Figure()
        self.axes = self.figure.
add_subplot(111)
        self.canvas = FCanvas(self, -1,
self.figure)
        self.canvas.SetPosition( (450,5))
        self.canvas.SetSize((300,250))
        ....
```

Un *Subplot* est d'abord créé dans une instance de *Figure*. Un graphique peut contenir plusieurs *Subplot*, p. ex. pour représenter plusieurs séries de données. La syntaxe est :
`figure.add_subplot(numrows, numcols,fignumber)`

Ici nous n'avons qu'un seul graphique (*fignumber = 1*), et par conséquent une seule ligne (*numrows = 1*) et une seule colonne (*numcols = 1*). Les trois dernières lignes créent la surface de dessin du *backend wx* avec l'objet *FCanvas*, la positionne et la dimensionne. Le graphique peut maintenant être tracé.
La fonction *OnIdle* a elle aussi été modifiée. Elle est appelée chaque fois que le programme est inactif – en particulier lorsque de nouvelles données sont en attente.

```
def OnIdle( self, event):
        # if nothing else to do, update
text from message queue
        while not self.serial_thread.
msgQueue.empty():
            msg=self.serial_thread.
msgQueue.get()
            self.textbox.AppendText(msg)

            ## display values in diagram
            self.axes.plot(self.serial_
thread.x, self.serial_thread.y)
            self.canvas.draw()
```

La dernière partie place sur la courbe les valeurs *x* et *y* précédemment reçues (le tableau *x* contient la valeur temps). La commande *canvas.draw()* trace la courbe finale.
L'ensemble de nos modifications est enregistré dans le fichier *Diagram py*. Nous n'avons pas cherché à paramétrer l'échelle du graphique. La mise à l'échelle s'adapte continuellement aux données. Autre effet de nos simplifications, la courbe change constamment de couleur car un nouveau graphique est tracé à chaque fois. Dès que l'on souhaite personnaliser l'aspect d'un graphique, l'étude de la documentation de *matplotlib* est incontournable.

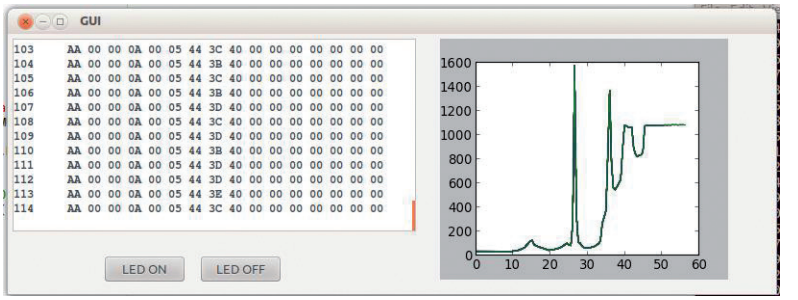


Figure 4. Une fenêtre pour deux représentations des données : texte et graphique.

Bilan & perspectives

Vous devriez maintenant vous faire une idée de l'approche à adopter pour récupérer des données et les visualiser. Notre programme est toutefois loin d'être parfait. Le traitement des erreurs est en particulier trop rudimentaire. L'interpréteur ne fait qu'envoyer un message d'erreur dès que quelque chose part de travers, rien d'autre.
Autre inconvénient, le programme devient inactif au bout d'un certain temps. Au rythme où le nœud envoie des valeurs, deux fois par seconde, il arrive en effet un moment où les tableaux *x* et *y* en contiennent tant que le tracé de la courbe dure trop longtemps. Si de nouvelles valeurs arrivent durant le tracé, le thread « GUI » est surchargé et la fenêtre se grise. Notez que le thread « réception » ne dépend pas de ce thread et reste donc toujours actif.
Ce défaut peut être contourné en modifiant par exemple le code de façon à ce que la courbe ne soit retracée que toutes les x secondes. Essayez de trouver une valeur de x satisfaisante !

(120744 – version française : Hervé Moreau)

Liens & références

[1] Page web de l'ElektorBus :
www.elektor.com/elektorbus

[2a] Le bus arrive (6) :
www.elektor.fr/110258

[2b] Le bus arrive (8) :
www.elektor.fr/110428

[3] De BASIC à Python (1) :
www.elektor.fr/110483

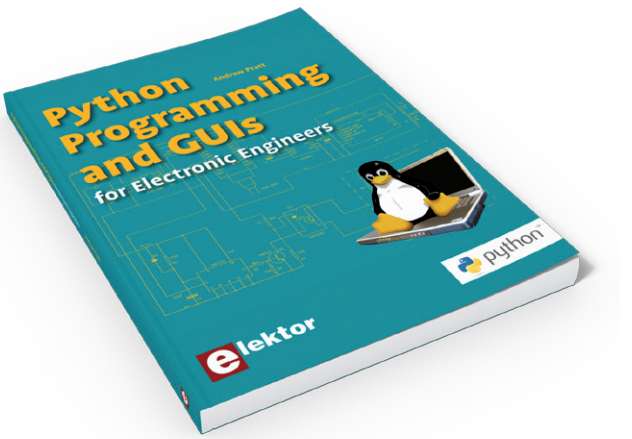
[4] De BASIC à Python (3) :
www.elektor.fr/120744

[5] Documentation matplotlib :
http://matplotlib.org/contents.html

[6] Sandro Tosi : Matplotlib for Python Developers

[7] Page de l'auteur (en allemand) :
http://staff.ltam.lu/feljc/home.html

[8] Python pour l'électronicien :
Andrew Pratt, Python Programming and GUIs for Electronic Engineers
www.elektor.fr/python-programming



L'auteur

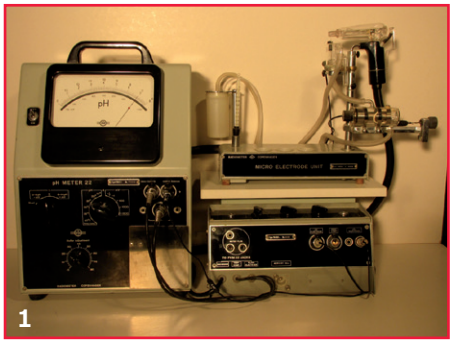
Jean-Claude Feltes enseigne l'électronique au Lycée Technique des Arts et Métiers de Luxembourg, un établissement public qui forme des artisans, des artistes, ainsi que des techniciens supérieurs. Jean-Claude se passionne encore pour l'électronique et l'informatique durant son temps libre [7].

analyseur de pH sanguin /O_2/CO_2 Radiometer PHM22 / PHA928a

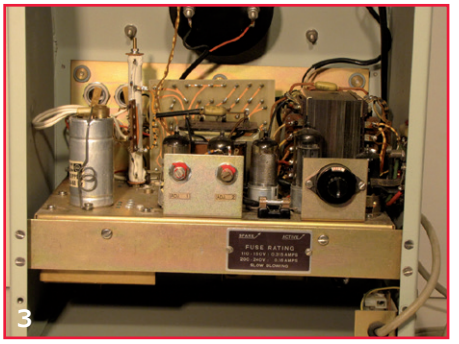
Patientez pour les résultats de vos tests sanguins

Seppo Lindeman (Finlande)

L'appareil est-il d'un vert vif ? Non, plutôt feldgrau. Alors c'est un *Radiometer*, fabriqué à Copenhague, c'était leur couleur favorite pendant plus de cinquante ans de production d'appareils de mesure électronique de haute qualité, au XXe siècle. La première fois que je suis tombé sur un appareil *Radiometer*, c'était en 1961, à Helsinki, à mon premier travail à l'usine de radio et télé *Helvar*. On trouvait beaucoup de voltmètres et de générateurs de signaux *Radiometer* « à lampes » un peu partout – faciles à repérer grâce à l'attroupement autour des prises de courant. Ce n'est que plus tard que j'ai remarqué que tous les appareils *Radiometer*, qu'il s'agisse d'analyse médicale ou de mesure électronique, étaient de ce vert-là. En ce temps-là :


1


2


3

- les transistors commençaient la concurrence commerciale avec les tubes à vide et personne n'avait entendu parler de décharges électrostatiques (ESD) ;
- les transistors au germanium claquaient souvent « mystérieusement » ;
- les amoureux des tubes à vide vantaient un des avantages des transistors : « ils prennent moins de place dans la poubelle quand ils sont morts ».

Le rêve du professeur Nimbus

Cet analyseur de pH/O_2/CO_2 sanguins n'est pas un appareil simple – il se compose de plusieurs ensembles distincts. Les premiers modèles dateraient des années 1950, cet exemplaire plutôt des années soixante. Ils se présentent soit sous forme de modèles de table soit comme « mobiles » sur des chariots. Le groupe d'appareils de la **figure 1** rassemble un pH-mètre sanguin de type PHM22t avec électrodes en verre et calomel (chlorure mercureux), une unité à micro-électrodes, un détecteur d'oxygène PHA928a avec une électrode pO_2 (pression partielle d'oxygène) et une cellule D616 commandée par thermostat, un tonomètre et deux humidificateurs en verre, un thermostat VTS13, une extension d'échelle de mesure et deux bouteilles de gaz.

pH-mètre 22 (version 1966)

Le cœur du système d'analyse est le PHM22t (**fig. 2**). Le mesureur contient cinq tubes à vide (**fig. 3**). Le schéma partiel de la **figure 4** permet de distinguer sept sections : amplificateur d'entrée, hacheur (*chopper*), amplificateur alternatif, démodulateur, afficheur, alimentation et diviseur de tension pour la compensation. Le schéma complet est disponible au téléchargement [2].

La pente de la tension de sortie de l'électrode de verre est de 61,54 mV.pH^{-1} à 37 °C. L'appareil a une résolution de 0,001 pH, soit 61 µV. Un amplificateur à tubes en continu pur n'est pas facile à stabiliser à des niveaux de tension aussi bas. De plus, la sortie en courant de l'électrode de verre est si faible qu'elle impose à l'amplificateur une impédance d'entrée de l'ordre de 50 à 500 MΩ. Si vous regardez la **figure 5**, vous trouvez le principe de fonctionnement d'un ancien pHmètre *Radiometer* de type PHM12. Il contient des tubes à vide configurés en amplificateur à grand gain. Il y a aussi un élément de référence (« Normal ») de type pile de Weston. Avant de commencer une mesure, les utilisateurs devaient appuyer sur le bouton *Test* et repérer la position de l'aiguille analogique par une marque sur le cadran. Il s'agissait de s'assurer qu'aucune dérive n'est due à l'amplificateur.

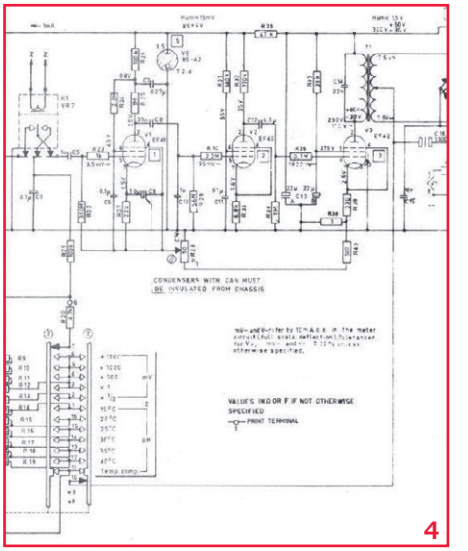
À l'intérieur du PHM22t

L'amplificateur d'entrée du pH-mètre comporte un circuit hacheur. Le hacheur convertit le signal continu d'entrée en un signal alternatif exempt de dérive et plus facile à amplifier. Le hacheur est semblable au convertisseur Brown présenté dans Rétronique de mai 2013 [1]. Le hacheur mécanique original de cet appareil a été remplacé ensuite par un hacheur photo-électrique (VR7). Malheureusement, un hacheur mécanique que l'on fait ronfler à 50 Hz pendant des années ne tardera pas à souffrir de problèmes de contacts.

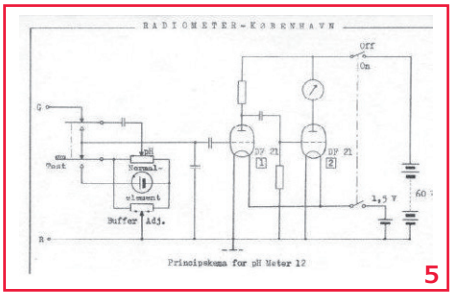
L'amplificateur de tension alternative comporte trois étages à tube avec contre-réaction de la cathode du tube de sortie à la cathode du tube d'entrée. La sortie de l'amplificateur attaque le circuit démodulateur par un transformateur. Le démodulateur convertit le signal alternatif en un signal continu virtuellement proportionnel à la tension continue à l'entrée du pH-mètre (fig. 6).

Le premier tube du PHM22t, V1 (absent de la **figure 4**), présente une particularité intéressante. Sa tension de filament est inférieure à 6,3 V, du fait d'une résistance série de 3 Ω, R48 – V1 est un redresseur. Le but est d'augmenter l'impédance d'entrée du tube en maintenant sa cathode « plus froide » que la normale. Au chapitre des inconvénients, la réduction de l'émission de la cathode est susceptible de réduire la durée de vie du tube, en permettant la contamination de la cathode. Quand j'ai ouvert le capot arrière pour la première fois, j'ai remarqué que l'un des tubes luisait moins que les autres et j'ai cru, à tort, qu'il était défectueux.

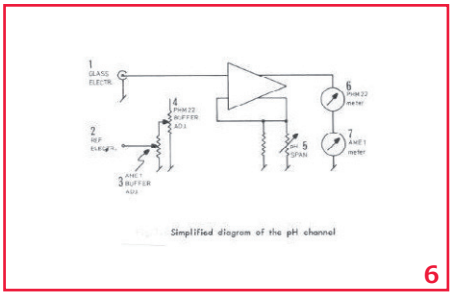
La tension d'alimentation continue de l'amplificateur vient d'un



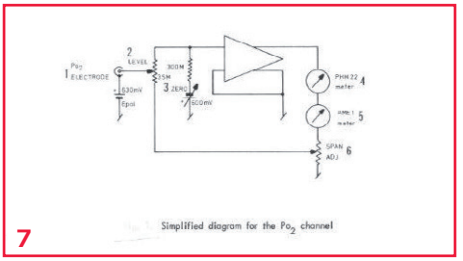
4



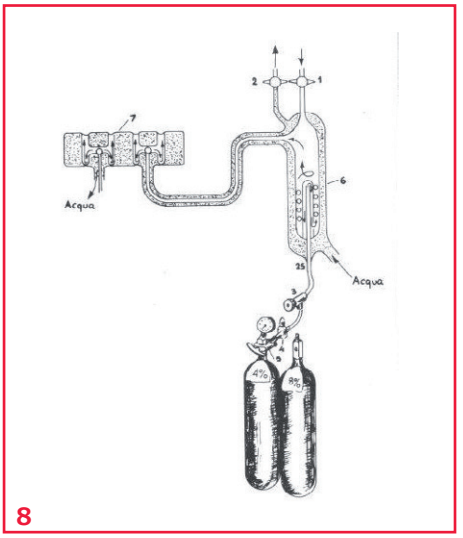
5



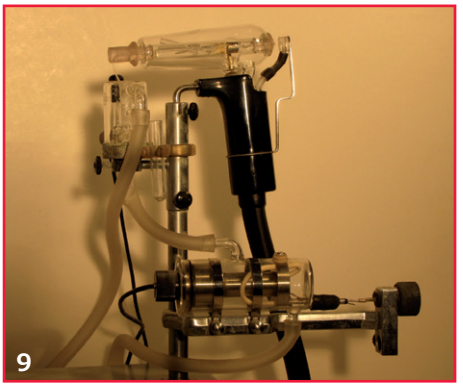
6



7



8


9

redresseur double alternance à tube. L'étage d'entrée fonctionne sous une tension de 85 V stabilisée par un tube.

Détecteur d'oxygène PHA928a

Le détecteur d'oxygène est un appareil entièrement passif qui ne contient que les circuits de contre-réaction qui se connectent dans le circuit du pH-mètre PHM22 et dans le mesureur extérieur.
L'électrode pO_2 fonctionne sous une polarisation de 630 mV tirée d'une pile au mercure de 1,35 V. Tout le circuit qui l'entoure est à basse impédance et faible courant. La **figure 7** montre le schéma simplifié du canal pO_2. Les **figures 6 et 7** montrent comment un simple pH-mètre peut être utilisé pour deux mesures différentes.

Dioxyde de carbone (pCO_2)

La grappe d'appareils ne comporte aucune électrode pCO_2. Le niveau de CO_2 dans le sang est mesuré indirectement au moyen de trois échantillons de sang. Le premier subit une mesure directe du pH. Les autres échantillons sont placés dans des chambres distinctes du tonomètre, qui reçoivent aussi des flux gazeux à 4 % et 8 % de CO_2 (**fig. 8**). Au bout d'environ 4 min d'équilibrage, on peut mesurer le pH des deux échantillons. La valeur de pCO_2 peut être obtenue par la méthode d'Astrup (et al.) au moyen du nomogramme Siggaard-Anderson qui établit la corrélation entre pH artériel et pCO_2 [3]. Il y a beaucoup plus à glaner de ce même nomogramme en matière de physiologie humaine, mais nous ne sommes pas entre carabins et ce sujet déborde du cadre de cet article.

Unité à micro-électrode

La cause principale des erreurs de mesure est la dérive. Les électrodes sont sensibles à la température et aux charges statiques du fait de la très haute impédance. Les échantillons de sang laissent des traces sur le verre et dégradent aussi (dans une certaine mesure) les membranes en polypropylène des électrodes. Les parties électroniques provoquent aussi des dérives. Toutes les électrodes sont protégées par des enveloppes de verre et plongées dans de l'eau thermostatée. L'eau est légèrement saline, ce qui crée un bouclier liquide autour de l'électrode – comme une enveloppe métallique protège un amplificateur sensible (**fig. 9**).
Comme la sensibilité des électrodes est susceptible de changer, le calibrage est nécessaire, au moyen de solutions de référence pour le pH et de solutions tampons et saturées pour la mesure de pO_2.

Cinquante ans plus tard

Aujourd'hui, les laboratoires d'hématologie disposent d'ampoules de contrôle et du confort de systèmes de test de haute qualité. Rien de tel dans les années 50 et 60 quand le kit *Radiometer* PHM22/PHA928a était en service. Un hôpital d'Helsinki était renommé pour ses tests de qualité et de calibrage dans le style « Rétronique » : au moindre soupçon de résultats de mesure incorrects avec les appareils PHM22/PHA928a, le personnel du laboratoire attendait qu'une certaine dame de l'équipe de nettoyage de l'hôpital prenne son service. Corpulente et toujours vaillante, elle passait pour un parangon de bonne santé. Une infirmière prenait donc simplement des échantillons du sang de la brave dame et les faisait analyser. Si le personnel du laboratoire trouvait cohérents les résultats de pH, pO_2 et pCO_2, tout le matériel *Radiometer* était aussitôt déclaré conforme.

(130132 – version française : Jean-Paul Brodier)

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

Liens

[1] La malédiction du collectionneur, Elektor mai 2013. www.elektor.fr/120753
[2] www.elektor.com/130132
[3] http://goo.gl/ICx3XI ou www.anaesthesia.med.usyd.edu.au/resources/lectures/acidbase_mjb/description.html

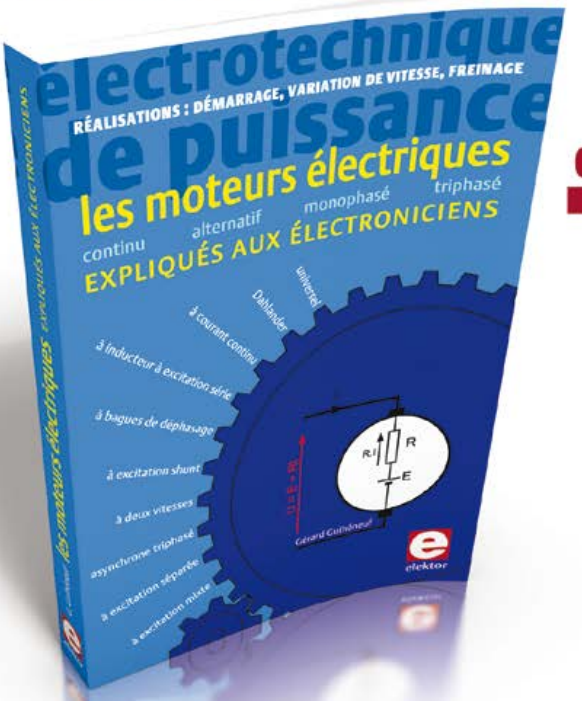
nouveau livre

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

Le premier détaille les principes de variation de la vitesse des moteurs à courant continu : conversion alt./continu (redresseurs commandés par des thyristors) et conversion CC (hacheurs à transistors IGBT). Mise en pratique avec des variateurs pour mini-perceuse et pour train miniature.

Les moteurs à alimentation alternative monophasée (à induction, à bagues de déphasage, universel) du 2e chap. font appel à une électronique de puissance. La puissance des réalisations proposées s'exprime en kW : démarreur à contacteur statique à deux points de commande pour moteur asynchrone monophasé à induction et variateur de vitesse pour moteur universel.

Reste le moteur électrique le plus utilisé dans l'industrie : le moteur asynchrone triphasé et ses différents principes de démarrage, de variation de vitesse et de freinage : démarreur électromécanique à contacteurs, démarreur-ralentisseur, convertisseur de fréquence ou couplage des pôles pour la variation de vitesse, moteur frein, freinage par injection de courant... Construisez le démarreur inverseur statique pour moteur asynchrone triphasé et découvrez la proximité entre électronique et électrotechnique : des portes NON-OU alimentées sous 12V commandent le sens de rotation d'un moteur de 1,5 kW alimenté sous 400 V en triphasé.



les moteurs électriques expliqués aux électroniciens - Gérard Guihéneuf

isbn 978-2-86661-188-0 | 320 pages | 37,50 €

www.elektor.fr/moteurs

nouvelle édition revue et augmentée du livre

l'électronique pour les débutants

deux kits d'initiation disponibles

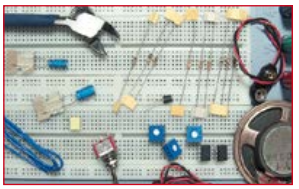
Voici le cadeau idéal pour partager votre passion de l'électronique avec vos enfants, petits-enfants, neveux… et autres *geeks*

Fin pédagogue, Rémy Mallard écrit pour les débutants dans un style inédit, 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 interrogations 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 deux kits disponibles séparément permettent de réaliser, sur une plaque d'expérimentation sans soudure, quelques-uns des montages simples et ludiques présentés dans le livre.

nouveau kit !



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

Kit n°1 : sirène – réf. 119016-71 - **22,50 €**

Kit n°2 : chenillard & thermomètre - réf. : 119016-72 - **22,50 €**

elektor

Offre spéciale : livre + deux kits = 81,50 € au lieu de 87,50 €

Informations complémentaires et commande :
www.elektor.fr/debut

les dessous (physiques) de l'internet

Tessel Renzenbrink
(Elektor)

L'internet est constitué de 40.000 réseaux administrativement séparés. Comment fonctionne physiquement ce système de systèmes ? Est-il fragile et peu fiable comme les quelques entreprises qui restent attachées à leurs connexions privées le croient ? Peut-il s'accommoder de la croissance continuelle des volumes de données ? S'étend-il pour atteindre les milliards d'habitants des pays en développement ?

Ces questions je les ai posées à Henk Steenman, directeur technique du nœud d'échange Amsterdam Internet Exchange (AMS-IX) et à James Cowie, cofondateur et directeur technique de Renesys, une entreprise qui mesure et analyse l'internet.

Nœuds d'échange

Au début des années 90, une grande partie du trafic internet européen local était routé vers la Virginie aux États-Unis via des câbles sous-marins transatlantiques. L'un des premiers nœuds d'échange (Internet Exchange ou IX), MAE-East, qu'y s'y trouvait abritait les liens physiques permettant de router le trafic d'un réseau à un autre. Pour beaucoup de petits fournisseurs d'accès européens, c'était le seul point d'échange disponible. En 1997, vingt fournisseurs d'accès et opérateurs concurrents ont fondé l'AMS-IX afin d'interconnecter localement leurs réseaux [1]. L'AMS-IX a permis de réduire le coût des échanges de données, la latence et a soulagé le nœud américain fortement surchargé.
Henk Steenman appartient depuis le début à l'organisme à but non lucratif hollandais. Avec son aide, l'AMS-IX s'est développé jusqu'à devenir l'un des plus gros nœuds d'échange au monde. En constante concurrence avec le DE-CIX à Francfort, l'AMS-IX est actuellement deuxième avec 595 réseaux et un trafic qui peut atteindre 2,3 Tb/s.

Mesurer l'internet

Renesys est une entreprise américaine qui collecte et analyse des données sur la structure physique et logique de l'internet [2]. « La carte logique nous apprend quelles sont les routes que l'internet croit devoir utiliser pour router le trafic », nous dit James Cowie. « En gros, elle vous dira que pour acheminer une information vers telle

personne depuis tel endroit vous pourrez passer par telles organisations. La carte physique est plus détaillée et implique de déterminer quelles adresses IP et quels routeurs sont reliés et lesquels seront les plus utiles pour rapprocher le trafic de sa destination. Nous mesurons activement des millions de points à des centaines d'endroits autour du globe pour réaliser une carte précise de ce qui se passe sur l'internet.
Nous utilisons ces informations pour aider nos clients qui ont besoin de savoir comment utiliser l'internet efficacement pour leur entreprise. Chacun a tendance à étudier avec grand soin sa partie de l'internet, mais personne ne regarde au-delà. Les entreprises sont de plus en plus mondialisées et nous leur fournissons cette vison globale. L'internet n'est pas un système centralisé et nous remédions à ce manque de transparence. »

Couche physique

« Il est intéressant de voir ce que l'on apprend de la couche physique avec la grille logique et des données de performance relevées par des capteurs », nous dit Cowie. « Un jour, par exemple, nous avons vu plusieurs réseaux en Irak et en Iran disparaître simultanément. Nous avons trouvé cela étrange et avons épluché les nouvelles le lendemain pour y trouver une explication. Il y a un pipeline de gaz qui part d'Iran et traverse la frontière turque pour alimenter les marchés européens ; or, comme la construction des pipelines est longue et laborieuse (droits de passage, sécurisation, ensevelissement), on en profite souvent pour poser des fibres optiques, en même temps et dans la même tranchée, pour un coût marginal quasi nul. Ce jour-là, une bombe avait endommagé le pipeline… et la fibre optique ! »
« Ce qui est encouragent, c'est que cet incident n'a pas affecté durablement l'internet ; celui-ci s'accommode bien et souvent de ce genre de perturbations. Une autre route fibrée a probablement été capable de prendre rapidement le relais ; dans nos relevés de données apparaissent une coupure puis un rétablissement. L'internet est bien plus résistant que l'on essaye de nous le faire croire. »

Réglementation gouvernementale

Les gouvernements du monde entier cherchent de plus en plus à réguler l'internet au niveau des utilisateurs. J'ai demandé à nos deux spécialistes s'ils observaient la même tendance au niveau des infrastructures.
Cowie : « L'UIT, l'agence de l'ONU responsable de la réglementation mondiale des télécommunications, n'a pas prêté attention à la période de croissance déterminante durant laquelle l'internet est devenu quelque chose qu'ont ne pouvait plus réguler facilement. Ce qui a été, à mon avis, un excellent coup de chance : il est maintenant plus difficile de faire machine arrière. Il est possible que l'intervention des gouvernements entraîne une charge législative croissante : on est toujours sous une juridiction. Mais je pense que les gouvernements réalisent que la fluidité de l'internet permet aux services de s'installer n'importe où. Tout ce qui rendrait les marchés locaux moins attirants pour les investisseurs suscitera des réticences.

Henk Steenman, directeur technique de l'AMS-IX.

Les gens me demandent souvent si « leur partie » d'internet peut être déconnectée de la même manière que ce qui s'est passé en Égypte ou en Syrie. Je pense qu'en Europe occidentale ou aux États-Unis, il n'y a plus vraiment de menaces réelles pour l'internet. Il s'y est tellement développé qu'il n'est plus imaginable de l'attaquer ou le mettre hors-ligne. Il est au-delà de ça. »
Le directeur technique de l'AMS-IX ne souhaite pas non plus voir les réglementations augmenter : « Actuellement le législateur hollandais se tient à distance d'AMS-IX, mais cela pourrait changer. Si nous nous voyions imposer des réglementations et de la bureaucratie, cela se ferait au détriment de la flexibilité et de la simplicité qui nous

James Cowie, directeur technique de Renesys.

permettent de fonctionner. L'une de nos plus grandes qualités est d'être un nœud d'échange qui ne discrimine pas les opérateurs, ce qui implique que n'importe quel fournisseur peut s'y relier pour échanger du trafic. Nous aimerions propager notre neutralité autant que possible, et j'ai bien peur que nous en perdions si les gouvernements venaient à s'en mêler. »

Déferlante de données

Le trafic passant par l'AMS-IX double environ tous les deux ans. Le défi pour Henk Steenman et ses collègues est de trouver des solutions techniques pour gérer cette croissance. « Nous mettons en place un équipement Ethernet 100 Gb/s disponible depuis l'année dernière », nous dit Steenman. « Jusque là nous utilisions le standard 10 GbE ; la capacité de notre réseau va donc décupler. Comme nous sommes l'un des plus importants nœuds d'échange, nous faisons en permanence face aux limites de la technique. Nous participons au groupe IEEE qui développe le prochain standard : 400 GbE. Le débit de données atteignable a toujours décuplé avec chaque nouveau standard Ethernet, mais la technologie n'est pas encore prête pour passer à 1 Tb même si notre croissance le justifiait. D'un autre côté, la croissance fonctionne dans les deux sens, le trafic ne peut pas croître plus vite que l'infrastructure disponible ne le permet, je ne prévois donc pas de pénurie sérieuse. »

James Cowie ne s'inquiète pas non plus de la capacité. « Une fraction seulement des fibres reliant les continents est effectivement utilisée. La réserve de bande passante est énorme. À l'intérieur des continents, surtout en Europe, la quantité de bande passante en réserve, prête à être mise en service si le trafic croît, est tout simplement ahurissante. Je pense que ça ne sera jamais un problème. »

Fracture numérique

Dans la plupart des pays développés, une infrastructure internet bien établie fournit un accès rapide et bon marché. Cependant, il en va autrement dans les pays en développement où l'infrastructure peine à se développer, ce qui crée une fracture numérique. La différence s'atténue-t-elle ?

James Cowie : « La tendance est que les pays les plus à la traine sont à la pointe dès que l'internet y arrive. L'Afrique orientale en est le parfait exemple. L'accès à l'internet y était très limité, souvent par l'intermédiaire de connexions par satellite, lentes et chères. Puis les câbles sous-marins ont été reliés. En moins de trois mois, le marché de l'internet tout entier avait complètement changé. Les internautes résiliaient leurs contrats d'accès par satellite et tous se ruaient sur ce câble. Les débits sont passés de dizaines de kilobits par seconde à un gigabit. C'est ce qui s'est passé à l'échelle des semaines et mois.

Tout compte fait, ce qui s'est passé c'est que les utilisateurs ont sauté plusieurs générations de technologie. Ils n'auront peut-être jamais d'ordinateur de bureau et passeront directement au smartphone. En ne passant pas par toutes les évolutions que l'Europe occidentale a connues, ils ont pu sélectionner les meilleures technologies et au meilleur prix. En fait, c'est très positif. La fracture numérique est encore très profonde, mais l'internet la comble avec brio. »

Henk Steenman : « Maintenant que les réseaux croissent rapidement en Afrique orientale, la prochaine étape en terme d'infrastructure sera les nœuds d'échanges régionaux. Au Kenya, par exemple, une grande partie du trafic local destiné aux pays voisins est routé par l'Europe à cause du manque de nœud d'échange régional. Ils font face aux mêmes problèmes que ceux qui nous ont motivés à fonder l'AMS-IX dans les années 90. Nous avons pensé : "ça a marché pour nous, pourquoi pas le refaire pour eux ?" Nous développons actuellement un nœud d'échange à Mombasa en collaboration avec l'association des fournisseurs de services de télécommunication du Kenya (TESPOK) afin d'améliorer la connectivité régionale. »

Lorsque j'ai demandé à James Cowie quel serait selon lui l'avenir de l'internet, il a répondu : « Mouais, je ne suis pas devin. La seule chose dont je suis sûr, c'est que ce sera inattendu, et rien de ce que nous imaginons. Nous nous trompons tout le temps. J'imagine que l'innovation viendra de tous ces Africains de l'est qui font déjà partie de l'internet, mais qui ont de vrais besoins. Ce sera quelque chose qui ne nous serait pas venu à l'esprit ; car nous n'avons pas de besoins : la plupart sont déjà satisfaits. L'avenir de l'internet, c'est eux. »

(130130 – version française : Kévin PETIT)

Liens

[1] www.ams-ix.net

[2] www.renesys.com

hexadoku casse-tête pour elektorniciens

Les vacances tirent à leur fin, mais ne nous précipitons pas. Qu'une grille hexadécimale soit résolue d'une traite ou qu'elle traîne trois semaines au fil de nos trop rares moments de détente, l'essentiel est de se faire plaisir. Et celui qui en arrive à bout, en plus de la satisfaction, aura peut-être la chance d'emporter un bon–cadeau de 100 €.

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 envoyer la série de chiffres sur fond grisé.

Participez et gagnez !

Nous tirons au sort l'une des réponses internationales correctes reçues dans les délais ; son auteur recevra un chèque-cadeau d'une valeur de **100 €** à valoir sur des circuits imprimés **elektorPCBservice** (**Eurocircuits)**.
Nous offrons en outre 3 chèques-cadeaux à valoir sur des **livres d'Elektor** d'une valeur de **50 €** chacun.

Où envoyer ?

Envoyez votre réponse (les chiffres sur fond grisé) avec vos coordonnées par courriel, télécopie ou courrier avant le **1er octobre 2013** :

Elektor c/o Regus Roissy CDG – Le Dôme – 1, rue de La Haye
BP 12910 – 95731 Roissy CDG
Courriel : **hexadoku@elektor.fr** | www.elektor.fr/hexadoku

Les gagnants

La solution de la grille du numéro de juin (420) est : **F9407**
Le gagnant des **100€** à valoir sur des circuits imprimés **Eurocircuits** est Ciril Zalokar (Slovénie).
Les 3 chèques-cadeaux Elektor d'une valeur de 50 € chacun vont à :
Arne Jansson (Suède), Gerard Yvraut (France) et Philippe Monnard (Suisse).
Bravo à tous et félicitations aux gagnants !

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

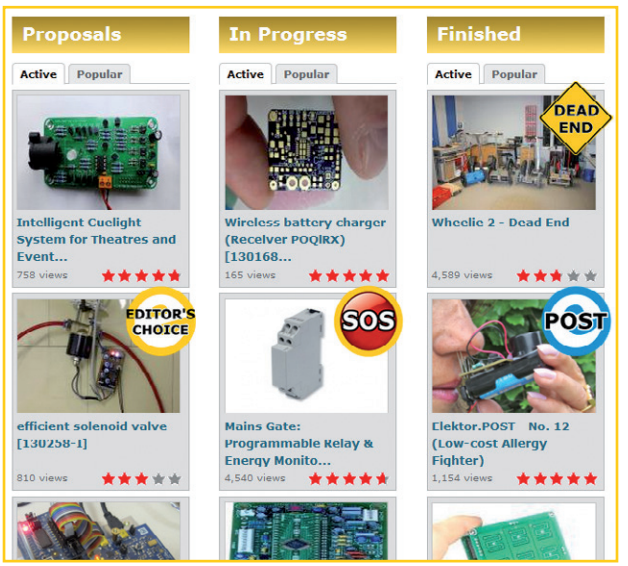
5	F	D	4	9	0	2	A	6	7	3	B	E	1	8	C
C	E	6	3	D	8	1	5	F	9	4	0	7	A	B	2
7	B	2	8	6	3	E	4	A	C	1	5	F	D	0	9
A	9	0	1	7	B	C	F	8	2	D	E	3	4	5	6
F	4	C	D	3	6	5	8	B	E	7	2	A	0	9	1
0	3	8	6	A	7	4	B	C	D	9	1	2	5	E	F
E	7	5	2	F	9	0	1	4	6	8	A	B	3	C	D
9	1	A	B	C	E	D	2	0	F	5	3	4	6	7	8
4	C	3	7	5	F	6	D	E	1	2	8	0	9	A	B
8	A	F	5	B	1	7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

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.

vivement la récré !

Clemens Valens
(elektor.labs)

L'été est fini, vous avez pondu de nouveaux projets, c'est le moment de (bien) les documenter sur elektor.labs. Voici quelques trucs et astuces pour tirer le meilleur parti de vos publications en ligne.



Les nouveaux icônes sur la page d'accueil d'elektor.labs.

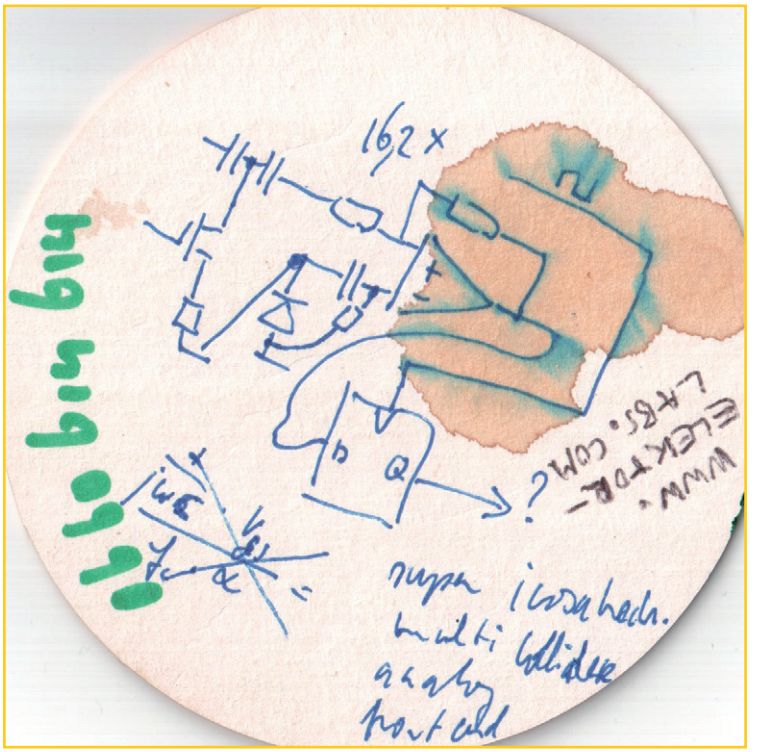
Icônes braillards

Nous améliorons sans cesse le site elektor.labs en y ajoutant des fonctions utiles, au fur et à mesure des besoins et selon nos possibilités. C'est le cas p. ex. des quatre icônes qui sur la page d'accueil permettent d'attirer l'attention sur un projet. Le *Post* bleu et l'icône jaune pour le choix de l'éditeur sont sous le contrôle exclusif de l'équipe d'Elektor. Les icônes *Dead End* (= *impasse*) et *SOS* sont pour tout le monde : ils informent les autres utilisateurs que vous êtes dans une impasse et que vous avez besoin d'aide. Attention ! Utilisez *Dead End* avec précaution, car s'il reste trop longtemps dans l'impasse, votre projet risque de passer sans autre forme de procès dans la catégorie *Finished*.

Réglage des nouveaux icônes pour mettre en valeur votre projet.

Visibilité de vos projets

Sur www.elektor-labs.com, les illustrations d'en-tête sont facultatives, mais vivement recommandées, car un projet bien illustré jouira automatiquement d'une meilleure visibilité. Chaque fois que vous cliquerez sur le bouton « sauvegarder » pour mettre à jour un projet, celui-ci sera déplacé vers le haut de la liste s'il a une photo d'en-tête. Par défaut, le site propose l'image d'un circuit griffonné sur un rond de bière taché. Pour le classement, le mécanisme de remise en haut de la liste ne tient pas compte de cette image-là.



Remplacez la photo du rond de bière taché par une photo de votre projet pour améliorer sa visibilité.

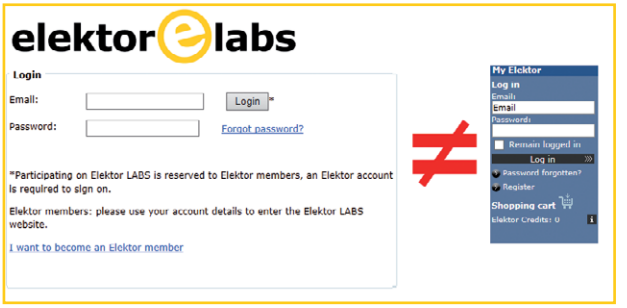
www.elektor-labs.com

Zèle récompensé, abus sanctionnés

Les contributions de certains internautes très actifs sont précieuses pour la vie du site .labs. Pour les distinguer, voici un **zélomètre** : un mécanisme qui identifie les contributeurs actifs, auxquels Elektor offrira des bonus occasionnels (cartes électroniques, livres etc). Le barème : 4 points pour la mise en ligne d'un projet, 2 points pour une contribution et 1 point pour un commentaire. Bientôt ces scores apparaitront sur le site (bricolage en cours). Ce mécanisme de valorisation est automatisé, mais l'intelligence humaine interviendra aussi pour une large part : nous filtrerons les contributions et les commentaires bidonnés. Cette pratique détestable pourrait même se retourner contre ses auteurs : les abus seront sanctionnés.

Mots de passe et @dresses

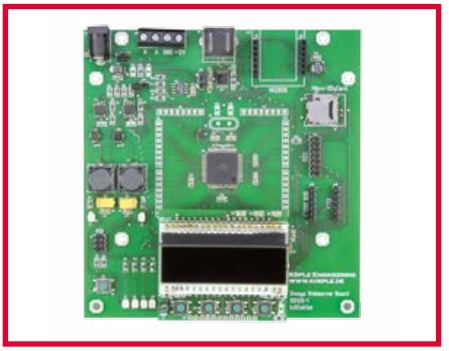
Pour des raisons historiques qu'il serait trop long d'élucider ici, notre base de données est à cheval sur deux systèmes informatiques dotés chacun de son protocole d'identification propre. L'unification de cette tour de Babel est en cours, mais dure plus longtemps que prévu. C'est pourquoi, pour l'instant, votre identification sur elektor.fr, elektor.post et elektor.store n'est pas (forcément) la même que sur elektor.labs et elektor.magazine. En attendant la fin de cet embrouillamini, simplifiez-vous la vie en utilisant la même @dresse électronique (et le même mot de passe) pour tous les sites d'Elektor.



Il suffit pour cela d'envoyer **depuis votre ancienne adresse** un message mentionnant votre ancienne adresse et votre nouvelle adresse, à service@elektor.fr ou labs@elektor.com.

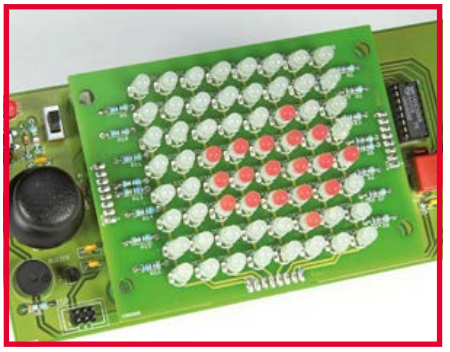
Pour atténuer le désagrément causé par les deux procédures d'identification différentes, utilisez la même @dresse et le même mot de passe pour les deux.

bientôt dans Elektor



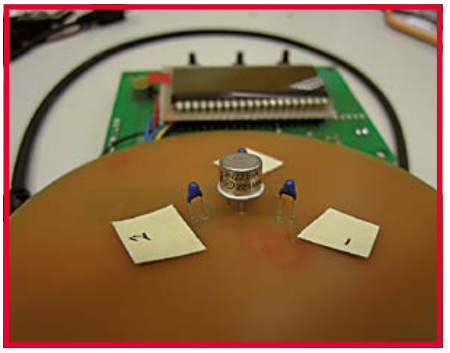
serveur web Xmega

Ce n'est pas la première fois qu'on l'annonce, mais cette fois c'est la bonne : elle arrive, **la carte polyvalente autour d'un AVR plutôt balaise**. Pour communiquer : 4 LED, 4 poussoirs, un afficheur et le choix entre RS485 et des connecteurs UART/TTL. Plus le connecteur d'extension, le socle micro-SD et la place pour le module TCP/IP.



matrice de 8×8 LED bicolores avec ATmega328P

Ce projet a surtout pour ambition de vous donner envie de programmer un µC Atmel. La matrice est donc un prétexte agréable **et** utile, car il ne sera pas difficile de lui trouver une application. Si vous avez quelques notions de base de la programmation C/C++, vous aurez tôt fait de maîtriser le principe du décalage de bits (*bitshift*).



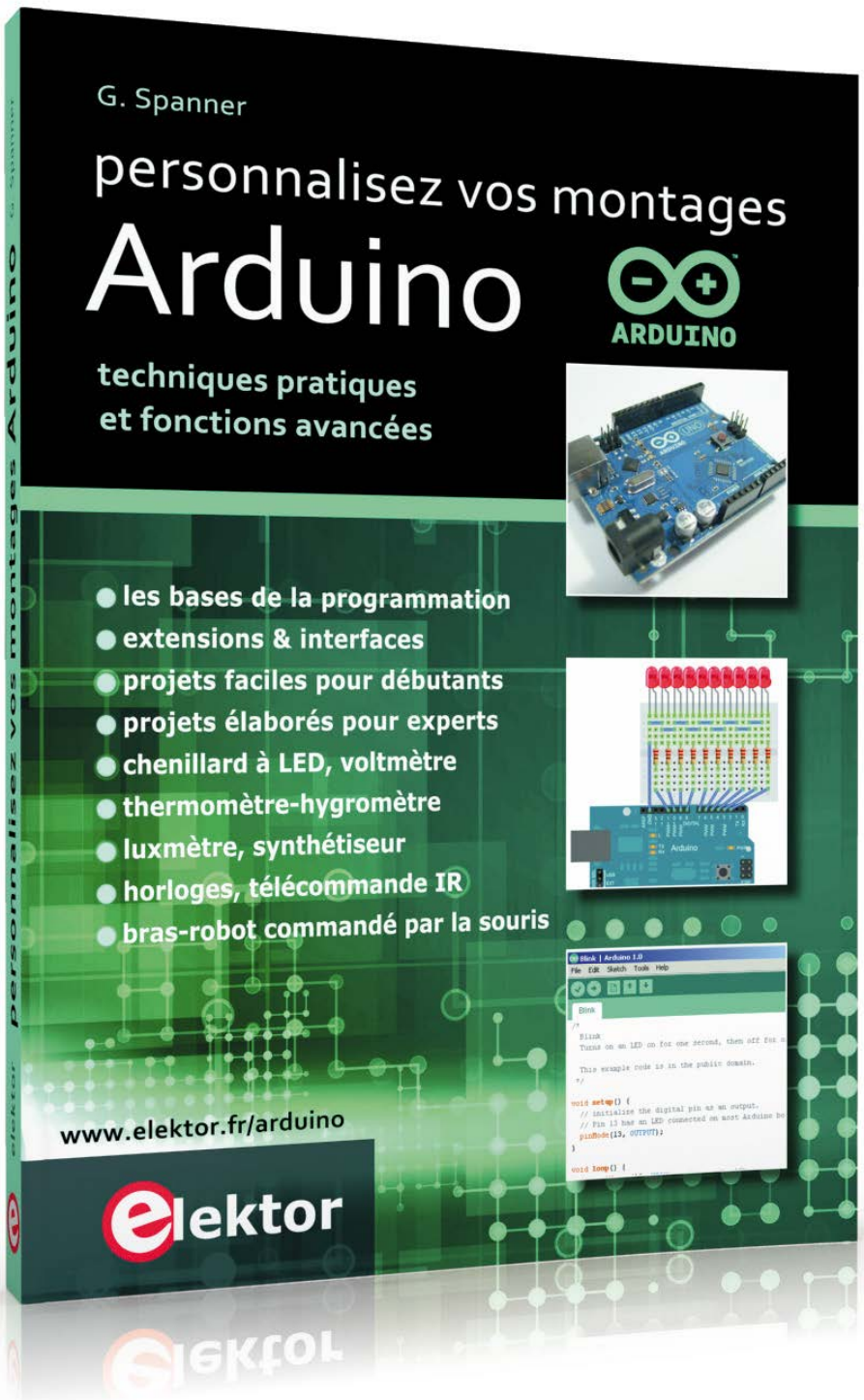
girouette-anémomètre sans pièce mobile

L'intérêt de ce projet réside dans l'absence de pièces mobiles. La méthode est connue, inspirée de la mesure de flux d'air, brillamment appliquée en avril 2012 dans le **double-anémomètre à fil chaud & anémomètre à tube de Pitot** de Marc Gérin dans le numéro 406 d'Elektor. (www.elektor.fr/110343). À l'aide de logiciel, bien sûr.

Informations préliminaires non contractuelles
Parution du numéro d'octobre : 24 septembre

Personnalisez vos montages Arduino

techniques **pratiques** et fonctions **avancées**



NOUVEAU

L'objectif de ce livre est de vous emmener à pas guidés vers la maîtrise d'Arduino.

Les projets, regroupés par thème, accompagnés de bases théoriques, sont des applications concrètes : chenillard à LED, voltmètre, thermomètre numérique, horloges sous différentes formes, ou encore bras de robot commandé par la souris.

Vous apprendrez ainsi à exploiter des techniques essentielles comme la conversion analogique-numérique, la modulation de largeur d'impulsion, ou encore les interruptions.

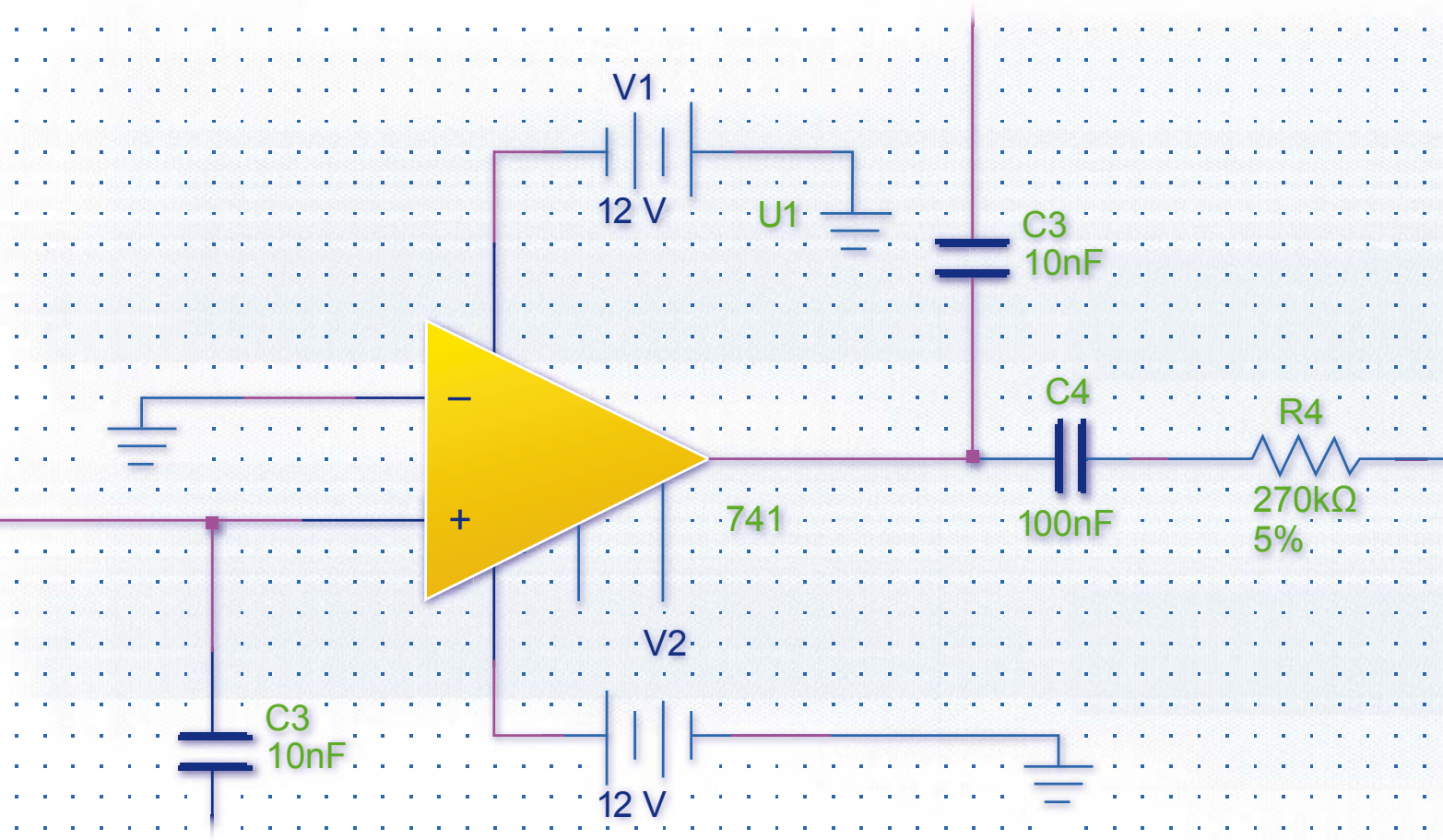
Après avoir mené à bien tous ces projets vous maîtriserez les fondamentaux de la technique des microcontrôleurs.

ISBN 978-2-86661-191-0
272 pages
34,50 €

elektor

www.elektor.fr/arduino

Une meilleure conception pour l'enseignement de l'ingénierie



Faites davantage qu'enseigner l'ingénierie. Pratiquez-la.

Enseigner la conception de circuits sans moyen efficace pour passer du concept à l'expérimentation, c'est comme expliquer à une personne comment faire un créneau sans lui laisser le volant. National Instruments propose les matériels et logiciels dont les étudiants ont besoin pour réaliser des expériences, afin d'aller au-delà de la théorie et de la simulation, et de prendre conscience de ce que la pratique de l'ingénierie signifie.

OUTILS PÉDAGOGIQUES

NI LabVIEW

NI myDAQ

NI ELVIS

NI Multisim

>> Découvrez comment NI soutient la prochaine génération d'innovateurs sur **ni.com/academic/f**

01 57 66 24 24

NATIONAL INSTRUMENTS