elektor
DÉCOUVRIR CRÉER PARTAGER

elektor n° 479 – septembre/octobre 2019

Horticulture Box éclairage de plantes

à « LED horticoles »

MicroSupply

alimentation de labo pour objets connectés

générateur de fonctions à DDS

large gamme de fréquences et nombreuses options à petit prix

concentrateur de capteurs avec MQTT ⊣⊢ banc d'essai : nouveau Raspberry Pi 4 ⊣⊢ variateur pour LED sans scintillement ⊣⊢ conception matérielle avec (V)HDL (4) ⊣⊢ transmission d'informations en WSPR ⊣⊢ catamaran aéroglisseur ⊣⊢ convertisseur A/N à PLD, simple à construire ⊣⊢ voltmètre sans fil pour batterie de voiture ⊣⊢ reproducteur de basses de haute ponctualité ⊣⊢ passerelle USB/RS232 ⊣⊢ réduction du bruit de sortie de l'ICL7660 ⊣⊢ projet SCCC (4) ⊣⊢ Rétronique : des trésors ou des déchets électroniques ⊣⊢ Fab Academy ⊣⊢ une journée avec LoRa

ISSN 0181-7450

(B) 16,50 € • (CH) 29,00 FS • (CAN) 20.99 $ca • (And) 15,50 €
DOM surface 16,50 € • DOM avion 16,90 € • N Cal/S 2000 cfp

L 19624 - 479 - F: 15,50 € - RD

reichelt
elektronik – Tirer le meilleur parti de votre projet

- Plus de 45 ans d'expérience
- Grande disponibilité et livraison garantie
- Plus de 110 000 produits

DELOCK® – LA MARQUE FORTE POUR LA TECHNOLOGIE INFORMATIQUE, LES PÉRIPHÉRIQUES ET LES ACCESSOIRES

DELOCK® we move the world

Séparateur DisplayPort

Miroir ou étendre le signal du moniteur à 3 affichages en résolution 4K !

- Compatible Dual Mode (DP+++)
- DisplayPort 1.4 Multi-Stream Transport
- supporte HDR
- supporte HDCP 1.4 et 2.2

(Livré sans moniteurs)

Référence : DELOCK 87737

TECHNOLOGIE TIPP **89,65** (€ 68,47)

Antenne LoRa, 868 MHz

DELOCK we move the world

Connecteur SMA

- Gamme de fréquence : 860 - 870 MHz
- ZigBee 868 MHz, Z-Wave 868 MHz
- Gain d'antenne : 3 dBi
- Impédance : 50 Ohm
- Rendement : 45,91
- ROS : 1,5

Référence : DELOCK 89769

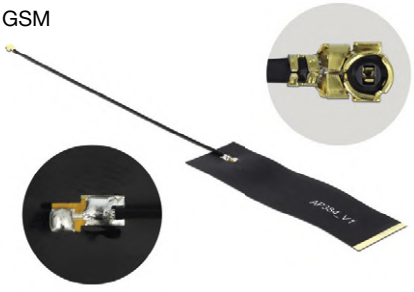
9,40 (€ 7,84) BEST SELLER

Antenne LoRa, 863-928 MHz

DELOCK we move the world

Connecteur MHF

- LoRa, NB-IoT, ZigBee, Z-Wave (908, 921), ISM, GSM
- Gain d'antenne : 1,68 dBi
- Impédance : 50 Ohm
- Montage adhésif
- Type de câble : 1.13



Référence : DELOCK 12540

7,08 (€ 5,90) BEST SELLER

ABONNEZ-VOUS À LA NEWSLETTER DÈS MAINTENANT ET PROFITEZ-EN !

Toujours être le premier à savoir - Top offres, sujets intéressants, actions et nouveautés.

INSCRIVEZ-VOUS MAINTENANT ► http://rch.lt/v3

Prix du jour! Prix à la date du: 1. 8. 2019

Types de paiement : Sofort. VISA MasterCard Maestro PayPal amazon pay

www.reichelt.fr

ASSISTANCE TÉLÉPHONIQUE: +33 97 518 03 04

42ème année,
n° 479 – septembre/octobre 2019

ISSN 0181-7450
Dépôt légal : août 2019
CPPAP 1120 T 83713

Directeur de la publication : Donatus Akkermans

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

@ : service@elektor.fr

Téléphone : (+33) 01 70 61 27 41
du lundi au vendredi de 10h à 13h

Télécopie : (+33) 01 70 61 27 42

www.elektor.fr | www.elektormagazine.fr

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

Publicité :
Margriet Debeij
Tél. : +49 (0)241 955 09 174
margriet.debeij@elektor.com

Imprimé aux Pays-Bas par
Pijper Media – Groningen
Distribué en France par M.L.P.
et en Belgique par A.M.P.

Les avez-vous comptées ? Vous en êtes à combien ?

La plupart sont cachées sous des capots à vis dans de petits objets souvent oubliés au fond d'un tiroir. Vous avez trouvé ? Ce sont les piles et les petites batteries. Dans chaque foyer français, il y en aurait une bonne centaine, sans compter celles engrangées dans votre atelier d'électronique.
Environ 70% des matériaux contenus dans une pile usagée peuvent être extraits et valorisés : plomb, cuivre, nickel, zinc, mercure, manganèse, fer, cobalt, lithium, cadmium… Ces métaux se retrouveront dans des tuyaux en cuivre, des pièces automobiles, des gouttières, des vélos et même dans vos boules de pétanque.
D'après un rapport de l'ADEME*, 45% des piles et accumulateurs portables (PA) sont aujourd'hui collectés. L'objectif des éco-organismes est de 50% en 2021. Hélas, la quantité moyenne de PA récupérée par habitant progresse peu d'une année à l'autre : 205 g/hab. en 2016, 209 g/hab. en 2017. Poursuivons nos efforts en privilégiant les piles rechargeables et en recyclant ! Il y a plus de 30.000 points de collecte. Donc surtout plus jamais de piles dans la poubelle !
Vous avez commencé à compter les vôtres ? N'oubliez pas les batteries des assistants vocaux (même le jeu du Monopoly en a un maintenant) ni celle de la clé de la voiture, ni celle de…
J'espère que vous avez bien rechargé vos batteries durant la pause estivale et vous souhaite une bonne reprise.

Mariline Thiebaut-Brodier

* « Piles et accumulateurs – synthèse – données 2017 », ADEME

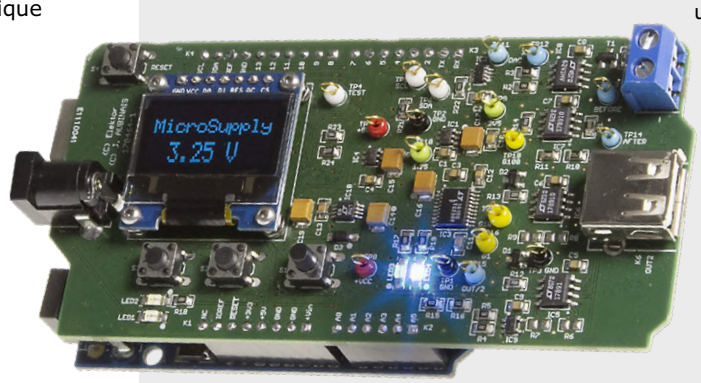
ce mois-ci

42e année – nº 479
septembre/octobre 2019

Horticulture
éclairage de plantes

MicroSupply
alimentation de labo pour objets connectés

Combien consomme mon objet connecté ? Est-ce que son mode de veille est conforme à la documentation du fabricant ? Combien de temps va tenir la pile CR2032 de mon objet ? Si vous vous posez ce genre de questions, voici une alimentation réglable, baptisée MicroSupply, sous forme de « shield » Arduino qui mesurera les courants très faibles. Grâce à un couplage avec un logiciel pour PC, vous pouvez visualiser et enregistrer la consommation de votre objet.



92

en coulisse

sur la scène : les projets

passerelle USB/RS232
appelez-moi BoB

76

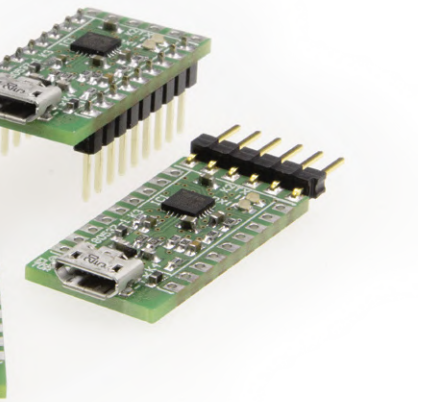
elektor
DÉCOUVRIR CRÉER PARTAGER

générateur de fonctions à DDS JDS6600 de JOY-IT

une large gamme de fréquences et de nombreuses options à petit prix

Dans le passé, tout électronicien sérieux possédait un générateur de fonctions à XR2206 dans son labo (amateur). Ce temps-là est révolu. Aujourd'hui, ce type de générateur utilise une puce électronique DDS (*Direct Digital Synthesis* ou synthèse numérique directe) qui, comme son nom l'indique, produit des fréquences à l'aide d'une technologie numérique. Malheureusement, les générateurs de fonctions DDS de qualité ne sont pas réellement abordables. Vraiment ? Nous avons testé pour vous le modèle proposé par JOY-iT, qui annonce une large gamme de fréquences et de très nombreuses options à un prix optimisé. Y aurait-il un loup ?

106



bientôt sur ces pages

Extrait du sommaire du prochain numéro :

- Automatisation avec Raspberry Pi
- Carte à pilote de moteur TMC2160
- Communication avec protocole UDP
- Compteur à 8 voies avec écran OLED
- Console de jeu Elektor
- Construire son votre propre lecteur/graveur RFID
- FreeRTOS pour ESP32
- Projet SigFox
- Radiamètre amélioré
- Récepteur radio à large bande SV3ORA

Sous réserve de modification.
Le numéro de novembre-décembre 2019 paraîtra le 17 octobre 2019.

PROJET DU LABO

concentrateur de capteurs avec MQTT

acquisition et transmission de valeurs de capteurs avec ESP32-PICO-KIT

Mathias Claußen (labo d'Elektor)

Dans le pénultième numéro d'Elektor, l'horloge à LED géante était l'un des projets à la une. Avec ses immenses afficheurs à sept segments, elle pouvait afficher non seulement l'heure, mais aussi des valeurs de mesure d'une carte à capteurs, reçues par Wi-Fi et MQTT. Dans cet article, nous proposons un microgiciel alternatif qui gère la connexion directe de capteurs à l'horloge. Les données sont envoyées à un courtier (*broker*) central, comme dans le cas d'un RPi avec Mosquitto et Node-RED. Pour le microgiciel de ce « concentrateur (*hub*) de capteurs avec MQTT », nous avons récupéré des modules logiciels de divers projets, d'où un énorme gain de temps pour le développement.

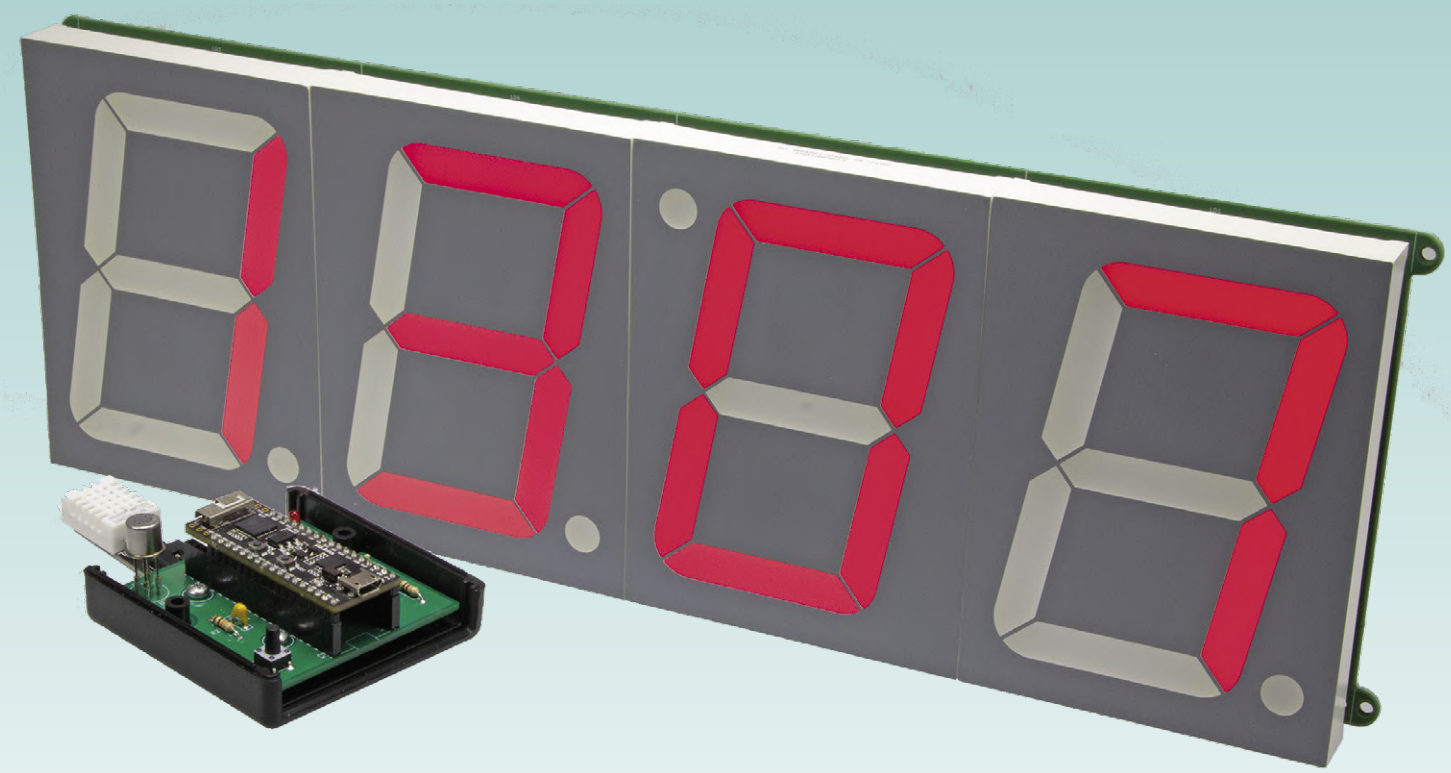
L'avantage du développement modulaire est qu'on peut réutiliser des parties déjà testées d'autres projets. Cela économise non seulement du temps de développement, mais aussi de l'effort dans le test et la documentation du logiciel.
Mathias Claußen du labo d'Elektor est en train de travailler, entre autres, à une deuxième version de la station météo Elektor à base d'ESP32 (voir [1]). Il y tient compte des désirs de lecteurs utilisateurs de la première version (on trouvera des améliorations et des mises à jour du microgiciel sous [2]). Il a été proposé, entre autres, la connexion directe à l'ESP32 des capteurs I2C suivants :

- BMP280, capteur de pression atmosphérique, de température et d'humidité de l'air [5]
- VEML6070, capteur d'ultra-violets [6]
- TSL2561, capteur de luminosité [7]

Comme l'horloge à LED géante [3] est également basée sur l'ESP32-PICO-KIT, il a été facile d'ajouter la prise en compte de ces capteurs dans le code de ce projet (à l'instar de la lecture de capteur utilisée dans le projet de surveillance du niveau de pollution de l'air [13]). Nous avons également repris du code de la station météo des parties du transport de données MQTT, qui servent à envoyer les données des capteurs à un courtier MQTT. Comme il est pénible, à chaque modification ou mise à jour, de décrocher l'horloge du mur pour lui téléverser un nouveau microgiciel, l'ESP32 peut maintenant être mis à jour par Wi-Fi grâce à la bibliothèque Arduino OTA (voir l'**encadré « Programmation OTA avec Arduino »**).

Connexion des capteurs

Il n'est pas prévu de connecteur particulier pour raccorder des capteurs. Les broches GPIO de l'ESP32 non utilisées pour l'affichage sont toutes amenées sur la face inférieure de la carte (**fig. 1**). On peut y souder directement des embases mâles ou des fils. Les broches I2C sont étiquetées ; pour les capteurs DHT11/DHT22, on utilise IO15 comme ligne de données. L'ESP32 offre une particularité avec sa matrice d'entrées-sorties : alors que pour d'autres microcontrôleurs, par exemple ceux de la série ATmega, les fonctions attribuées aux broches sont fixes et peuvent même conduire à des conflits lors de l'utilisation des fonctions spéciales, l'ESP32 fait mieux. Grâce à sa matrice d'entrées-sorties, on peut attribuer n'importe quelle fonction à pratiquement chaque broche de l'ESP32.
On peut donc non seulement utiliser l'I2C pour l'horloge, mais aussi associer l'un des UART à des broches encore disponibles pour l'utiliser. On pourrait même connecter un codec audio à des parties de l'interface I2S. On peut aussi piloter sans problème des extensions par SPI.



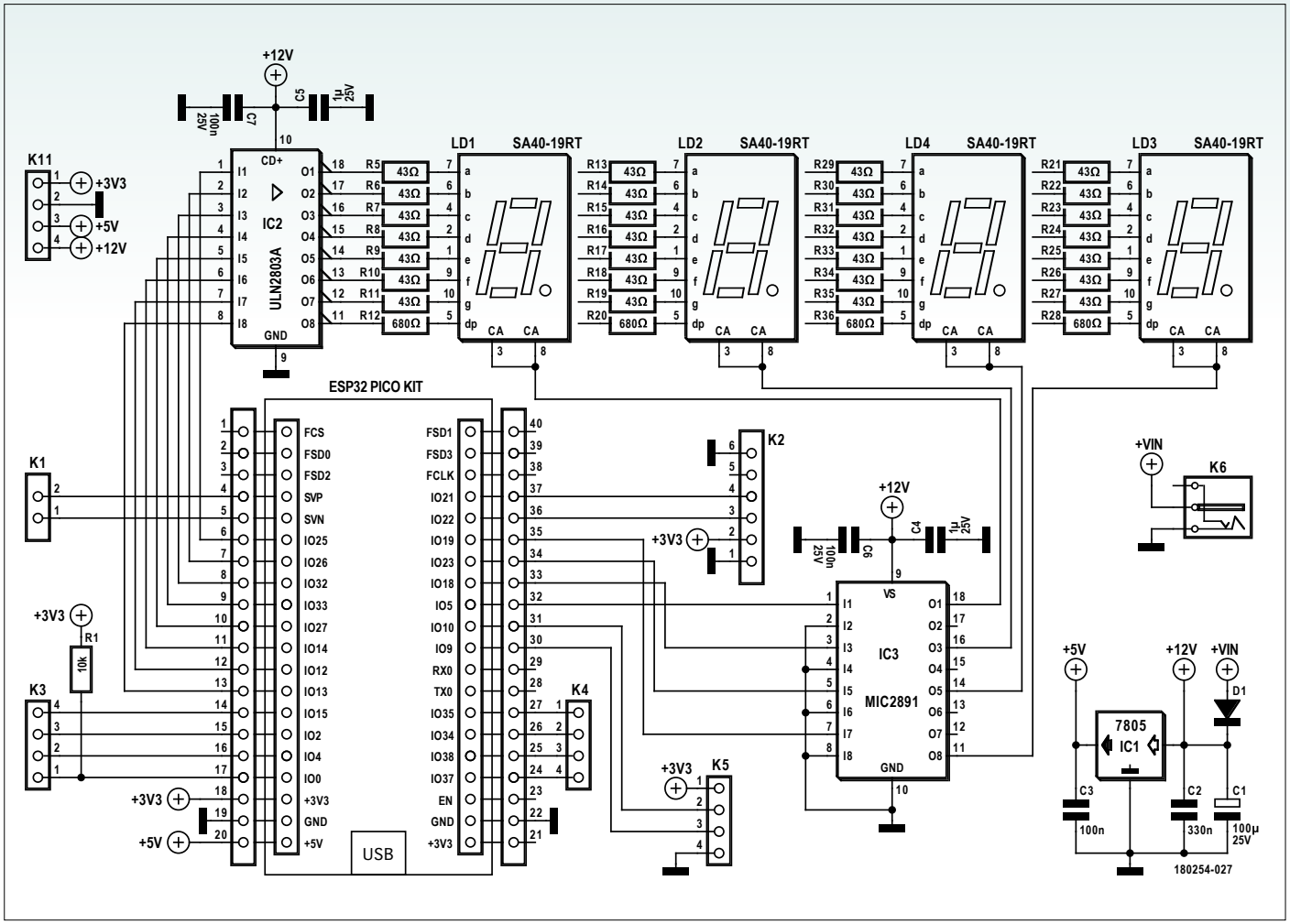


Figure 1. Schéma de l'horloge LED géante avec les broches libres de l'ESP32.

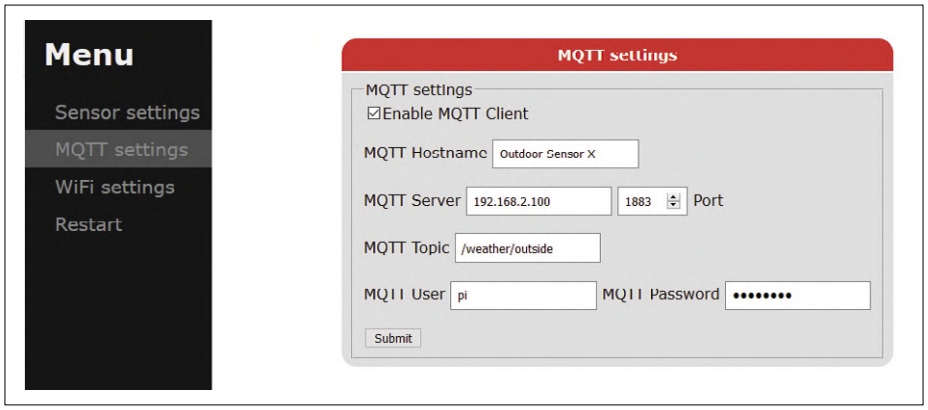


Figure 2. Paramètres MQTT configurables.

La configuration de la matrice est effectuée avec les bibliothèques *ESP32-Arduino-Core*. Grâce à des paramètres dans le code, on peut ainsi décider à quelle broche est attribuée une fonction donnée. On en a un exemple dans le projet « ESP32 comme serveur de temps » [8], où, pour la commande de l'afficheur OLED, les lignes SDA et SCL sont affectées aux broches GPIO 5 et 4 de l'ESP32. Avec l'instruction :

```
wire.begin(5,4,100000);
```

on choisit les broches, le troisième paramètre sert à spécifier la fréquence I2C souhaitée. On procède de manière analogue pour les UART de l'ESP.
Cette souplesse se paie par l'augmentation du temps de parcours du signal et la réduction de la fréquence utile. La structure matricielle impose une limite à 40 MHz, les signaux plus rapides ne pouvant plus être routés. En raison de latences, il peut y avoir, au-delà de 20 MHz, des décalages entre les données et le signal d'horloge conduisant à de possibles erreurs d'évaluation de bits.

Envoi MQTT

Venons-en à ce qu'envoie par MQTT le concentrateur de capteurs dans la nouvelle version de l'horloge géante. Il s'agit d'une chaîne de caractères JSON avec la structure suivante :

```
"{"data":
   {"temperature":31.1,
   "humidity":25.9,
   "airpressure":0,
   "PM2_5":-1,
   "PM10":-1,
   "Lux":-1,
   "UV":-1}
}"
```

L'objet data contient les valeurs de mesure. Si aucun capteur n'a été reconnu, des valeurs par défaut sont attribuées aux paramètres. Elles sont nulles pour la température, l'humidité et la pression atmosphérique. Toutes les températures ont pour unité le °C. Toutes les valeurs par défaut restantes sont mises à −1. Mais si les capteurs en question répondent présents au démarrage, on a affaire à de vraies valeurs de mesure.
Comme sur la première version du micrologiciel de l'horloge géante, le serveur (le courtier MQT*T) et* le sujet (topic) des données envoyées sont paramétrables sur une page de configuration fournie par le serveur **web de** l'ESP32 (fig. 2). Vous trouverez en [4] comment utiliser un RPi comme courtier MQTT, en [9] comment installer Node-RED, qui permet de réaliser rapidement des petites applications exploitant et affichant des données de capteurs.

Modules logiciels

Après une vue d'ensemble du contenu, venons-en au comment faire : un codeur n'est nullement tenu de réinventer la roue à chaque fois. Si l'on ne travaille que sur une seule plateforme (une famille de µC, etc.), on collectionne quasi automatiquement des routines, des bibliothèques et des modules logiciels qu'on réutilise fréquemment. C'est aussi le cas au labo Elektor pour l'ESP32, vu le nombre considérable de projets qu'il anime. Des projets en réseau analogues, tels que la station météo ou l'horloge géante, ont fondamentalement le même noyau constitué d'un serveur web, du paramétrage du Wi-Fi par navigateur web et de routines de confi-

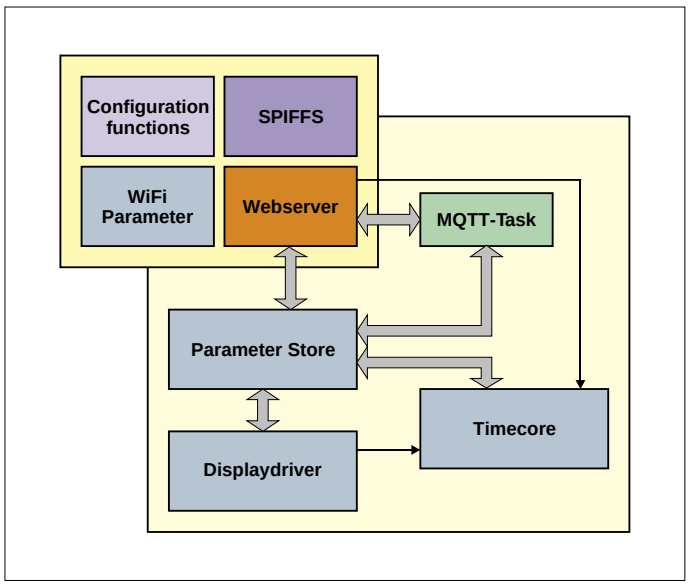


Figure 3. Modules logiciels utilisés.

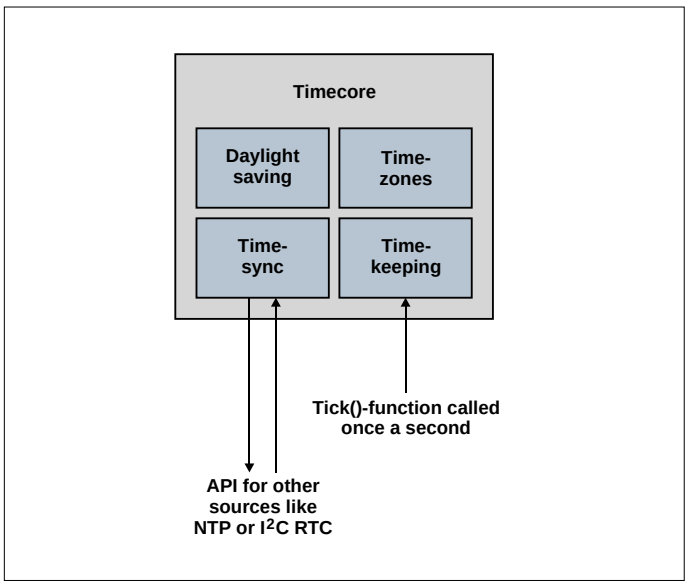


Figure 4. Fonctions de TimeCore.

guration. Il n'y a donc aucun besoin de redévelopper cela à chaque fois.

Horodatage

Pour l'heure, on a réutilisé une partie de l'horloge rétro à afficheurs de flipper [10], qui s'occupe de l'heure, de la synchronisation, des fuseaux horaires et du changement été/hiver. Ce module, développé *sous le* nom de TimeCore, est réutilisé dans différents projets. La **figure 3** en montre la structure générale. *La bibli*othèque TimeCore inclut un client NTP et peut référencer en cours d'exécution d'autres sources d'heure comme une horloge en temps réel I2C. Il suffit simplement de s'assurer qu'*une fonc*tion de TimeCore est appelée une fois par seconde pour incrémenter un compteur de secondes interne (**fig. 4**).
Pour cela, on utilise *la bib*liothèque ticker, qui possède la faculté simple d'exécuter des fonctions périodiques. Le code associé :

```
#include
…
Ticker OneSecondTick;
…
OneSecondTick.attach_ms(1000, FNC);
```

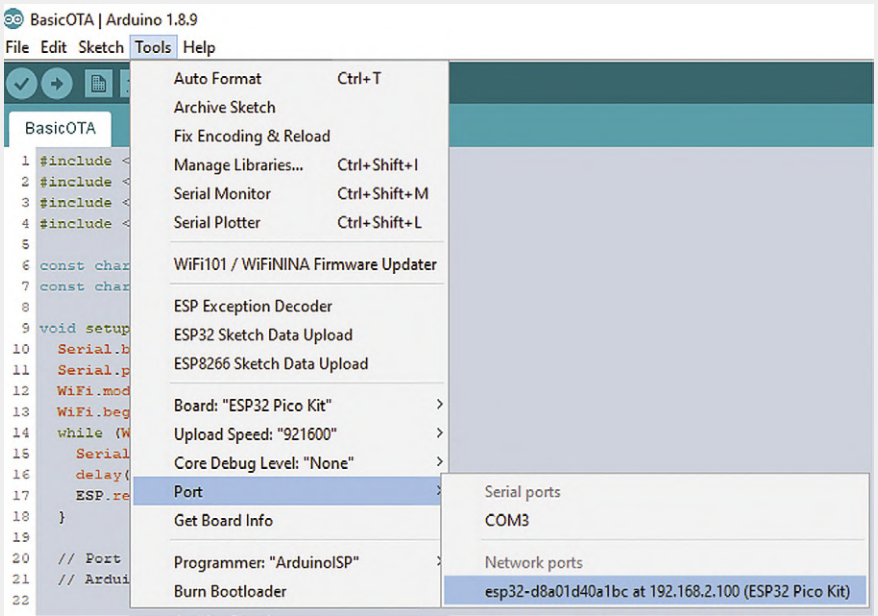
L'expression `FNC` tient lieu de fonction à appeler périodiquement. Celle-ci doit avoir le prototype `void fnc(void);` et donc n'accepter aucun paramètre ni ne retourner aucune valeur. Le premier paramètre de la fonction `attach_ms` est le délai entre deux appels en millisecondes. Cela permet aussi des appels récurrents à d'autres fonctions. Il faut toutefois prendre garde à une chose : sur l'ESP32 s'exécute le système d'exploitation FreeRTOS [11], alors qu'il est caché au programmeur dans l'EDI Arduino. Avec `Ticker`, on a affaire à une seconde tâche qui appelle la fonction spécifiée avec une très faible priorité. Il peut donc arriver que dans l'intervalle spécifié la fonction appelée soit interrompue par une tâche de plus haute priorité, étant donné qu'on n'utilise pas d'interruptions ici.

Les tâches

MQTT joue un rôle particulier dans le logiciel. On pourrait inclure toutes les fonctions dans la fonction `loop()`, mais l'ensemble en deviendrait beaucoup moins modulaire. C'est pourquoi on a préféré en faire une tâche séparée. Quand MQTT n'a plus de données à transmettre, cette tâche est mise en sommeil jusqu'à ce qu'il soit alimenté avec de nouvelles données. Quand la tâche MQTT établit une liaison avec un serveur, cela peut prendre plusieurs secondes. Cela se fait particulièrement sentir si à ce moment le *serveur* déclare un timeout. Si l'établissement de la liaison s'effectue dans une tâche séparée, FreeRTOS peut le gérer pendant les temps d'attente d'autres tâches (comme la fonction `loop`).
La tâche doit toutefois être informée de la disponibilité de nouvelles données. C'est facile à réaliser, car sous FreeRTOS chaque tâche *peut re*cevoir des *notifications* (**fig. 5***)* si l'identifiant (*handle*) de la tâche est connu. Le sixième paramètre

Programmation OTA avec Arduino

On peut téléverser un nouveau microgiciel sur une carte ESP32 par son port série. C'est spartiate, mais très souvent suffisant. Mais le vrai confort s'obtient par la programmation OTA (**O***ver* **T***he* **A***ir*). Pour que ça marche, il faut qu'un progiciel capable d'effectuer des mises à jour OTA soit déjà résident en mémoire flash. On en a un petit exemple dans l'EDI sous *Exemples -> ArduinoOTA -> BasicOTA*.
Si l'on intègre ces parties à son propre microgiciel et si le pare-feu du PC utilisé pour le développement est configuré en conséquence, on obtient un nouveau port réseau qu'on peut choisir dans l'EDI (voir la **figure**).
Pour téléverser un nouveau microgiciel, un croquis de téléchargement écrit les données dans une partition spécifique OTA de la mémoire flash. C'est la raison pour laquelle au maximum la moitié de la mémoire flash matériellement disponible est utilisable pour les applications. En plus du micrologiciel, on peut aussi renouveler le contenu du système de fichiers flash SPI (SPIFFS), qui contient normalement les pages web utilisées par le serveur web. Comme la fonction OTA est une partie du microgiciel, l'écriture ne peut être effectuée par le microgiciel que si au moins la partie OTA du nouveau microgiciel s'exécute sans crash.
Pour rendre le port visible dans l'EDI Arduino, l'ESP32 informe en arrière-plan par



`mDNS` tous les membres présents sur le segment de réseau de sa capacité OTA et de l'adresse IP associée. Au démarrage d'une mise à jour, il y a un échange de paquets UDP et éventuellement une demande de mot de passe. Si aucun mot de passe n'a été établi, n'importe qui sur le réseau local peut effectuer une mise à jour, ce qui ne va pas sans risque ! Toutefois, s'il y a un mot de passe, on ne peut plus mettre à jour le SPIFFS avec l'EDI Arduino. Sans mot de passe, on risque de voir des plaisantins ou des *hackers* charger du micrologiciel sur l'ESP32. C'est pourquoi il faut évaluer soigneusement où et comment employer l'OTA. Au besoin, il faut prendre des mesures complémentaires pour empêcher des mises à jour intempestives.

de la tâche `xTaskCreatePinnedToCore` est un pointeur vers une variable du type `TaskHandle_t`.

```
xTaskCreatePinnedToCore(
   MQTT_Task,
/* Function to implement the task
   */
   "MQTT_Task",
/* Name of the task */
   10000,
/* Stack size in words */
   NULL,
/* Task input parameter */
   1,
/* Priority of the task */
   &MQTTTaskHandle,
/* Task handle */
   0);
/* CORE to pin */
```

Si la création de la tâche réussit, la variable pointée par ce pointeur contient la valeur de l'identifiant, ce qui permet à d'autres fonctions d'envoyer des notifications à la tâche.
Une tâche peut même se mettre en attente de notifications, ce qui est réalisé pour la tâche MQTT avec l'instruction suivante :

```
ulNotificationValue =
   ulTaskNotifyTake(pdTRUE,
   portMAX_DELAY);
```

Le premier paramètre provoque la réinitialisation de tous les bits après la lecture, le second contient le délai d'attente maximal (infini ici). La fonction retourne une valeur sur 32 bits, dont les bits peuvent être positionnés par la fonction mqtt*settings_update dans* webserverfunctions.cpp :

```
if(MQTTTaskHandle != NULL){
   xTaskNotify(MQTTTaskHandle,
   0x01, eSetBits);
}
```

Dans ce fragment de code, on commence par vérifier si l'identifiant a été bien initialisé. Ensuite on positionne le bit 0 dans la valeur de notification de la tâche MQTT. Si celle-ci se trouve être en sommeil, elle est réveillée et exécutée. Cette notification informe la tâche MQTT que les paramètres pour le transport de données ont changé. La tâche peut alors lire et traiter les nouvelles valeurs.
Comme une description plus approfondie de FreeRTOS et de ses mécanismes de communication intertâches dépasserait le cadre de cet article, nous la réservons pour un article à paraître prochainement où seront traitées plus en détail les possibilités de FreeRTOS et la programmation des tâches.

Commande modulaire de l'affichage

La tâche la plus importante de l'affichage est la commande des afficheurs à sept segments à LED. Nous avons à nouveau choisi une démarche modulaire pour pouvoir réutiliser le code dans d'autres projets. Il y a une modification par rapport à la version d'origine : il faut passer la version du matériel en paramètre. Comme annoncé dans l'article de l'horloge à LED géante, il y a une version « mini » à venir, mais pour l'instant la carte ne donne pas encore entière satisfaction au labo d'Elektor. C'est pourquoi le pilote a été préparé pour prendre en compte des modifications de conception de la carte. Le pilote n'est ainsi plus lié à une horloge particulière, mais peut être facilement intégré à des projets personnels. Voici quelques indications qui devraient faciliter son adaptation à votre propre matériel.
La fonction `SevenSegmentSetup`(Bedroom clockHW_t HW) effectue dans la matrice *Array DisplaySegmentPins* l'affectation des broches aux segments de LED A à G et au point décimal.
Dans *sevensegments.cpp*, on trouve :

```
volatile int DisplaySegmentPins[]=
   {25,26,32,33,27,14,12,13};
```

Les paramètres correspondent aux broches pour les segments A à G, le dernier étant pour le point décimal. Une simple modification suffit à les adapter à une autre configuration. Cela vaut aussi pour les broches communes des afficheurs LED. Avec le tableau :

```
volatile int DisplayCommonPins[]=
   {23,19,18,05};
```

on fixe les broches communes des afficheurs 0 à 3.
À l'appel de la fonction `SevenSegmentSetup()`, on doit spécifier le type de matériel utilisé, ce qui a pour effet l'écriture des numéros de broches correspondants dans les deux tableaux précédents. Comme on utilise une inter*ruption* de cadenceur (timer), ces données, déjà présentes dans la mémoire flash du µC, doivent être recopiées en mémoire RAM.
La raison en est la suivante : pour que l'affichage s'ef*fectue* au bon moment (timing), on utilise un cadenceur de l'ESP32. Ce cadenceur déclenche la fonction de commande des différents segments à une cadence de 400 Hz. Dans le cas d'un µC AVR ou Cortex-M, il est facile d'aller lire les données dans la mémoire flash. Sur l'ESP32, c'est plus délicat, car le µC ne possède en fait aucune mémoire flash intégrée, mais seulement une ROM d'amorçage et de la RAM. Le logiciel est toujours résident dans une puce flash QSPI externe. Comme la vitesse de lecture de celle-ci est limitée, le µC est équipé d'un cache qui contient les données qui viennent juste d'être chargées ou d'utilisation prochaine probable. L'exécution du code est ainsi plus rapide que ne le permettrait sa lecture directe depuis la mémoire flash. Mais lors d'une interruption, il peut arriver que le code ne soit pas présent en cache et qu'il faille le lire depuis la mémoire flash, ce qui ralentirait l'opération, surtout si ont lieu en même temps d'autres opérations ou même une écriture vers la mémoire flash, empêchant la lecture. Ce serait nocif pour le traitement de l'interruption. Non seulement la lecture coûterait du temps, mais s'il y avait une collision entre la lecture et un processus d'écriture, cela provoquerait une exception et un redémarrage de l'ESP32. C'est pourquoi toutes les données nécessaires au traitement d'une interruption doivent être présentes en RAM. La fonction appelée par le cadenceur est déclarée avec `IRAM_ATTR ()`, ce qui demande au compilateur et à l'éditeur de liens d'en faire faire une copie en RAM au démarrage, laquelle sera exécutée. On s'affranchit ainsi totalement du problème de l'accès à la mémoire flash.
Il y a **en**core la question : qui traite l'interruption ? La réponse « le µC » appelle la question « oui, mais laquelle de ses parties ? », car l'ESP32 possède deux cœurs. La règle de base est qu'une interruption est traitée par le cœur auquel elle

a été assignée. Pour en savoir davantage sur les processus internes de l'ESP32, on peut se référer à la documentation de l'ESP-IDF [12].

Nous avons connecté un capteur DHT11 (température et humidité de l'air) à la carte prototype du labo d'Elektor. Nous avions un souci avec la bibliothèque utilisée : lors de la lecture des données du capteur, les interruptions sont brièvement masquées. Pendant le développement du logiciel, à chaque lecture du capteur, il se produisait une désynchronisation de l'interruption du cadenceur du multiplexage de l'affichage et de la tâche d'affichage de l'heure ou de la température. Nous avons résolu ce problème de manière simple, en transférant la tâche MQTT du cœur 1 vers le cœur 0. Ainsi, l'interruption n'est plus perturbée et les tâches ne se gênent pas mutuellement.

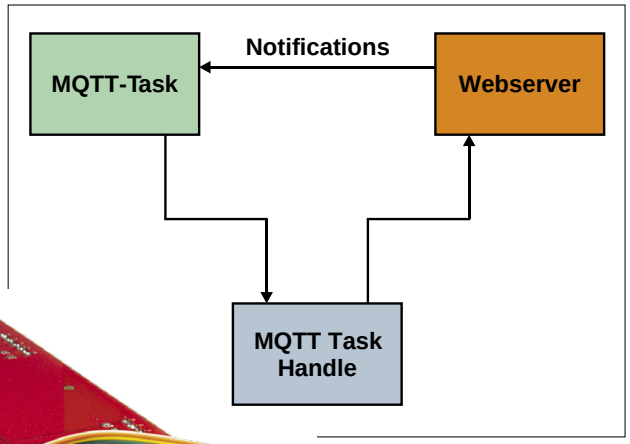


Figure 5. Informations passées aux tâches par notifications.

Conclusion

Cet article sur la mise à niveau d'un microgiciel illustre l'efficacité de l'usage de bibliothèques et de la réutilisation de code dans le développement de logiciel. Même si cela coûte un peu plus d'effort au début, on en sera largement récompensé dans les projets ultérieurs. Mais le recyclage de code et de bibliothèques ne doit pas dissuader de jeter un coup d'œil occasionnel sous le capot. Les environnements de développement Arduino et ESP-IDF épargnent beaucoup d'efforts, mais dissimulent les détails qu'il est parfois important de connaître. Beaucoup de phénomènes énigmatiques et perturbants et de dysfonctionnements qu'on passe des heures à traquer trouvent bien souvent leur explication dans les notes de bas de page des feuilles de caractéristiques ou de la documentation de l'environnement de développement – pourvu qu'on se donne la peine de les lire !

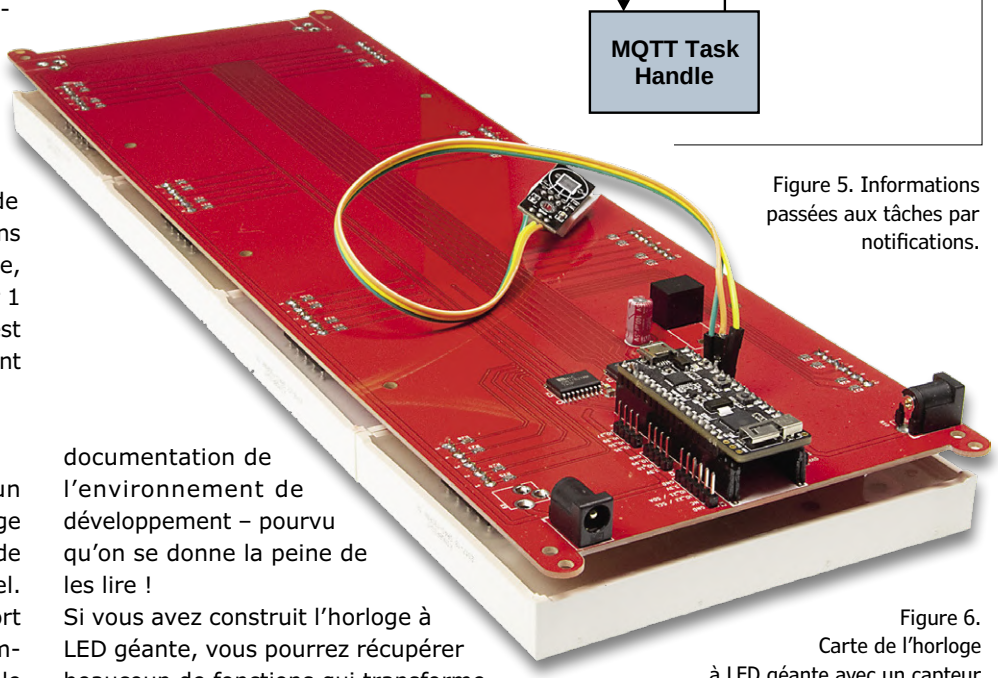


Figure 6. Carte de l'horloge à LED géante avec un capteur connecté directement.

Si vous avez construit l'horloge à LED géante, vous pourrez récupérer beaucoup de fonctions qui transformeront l'horloge en un concentrateur **MQTT de** capteurs. La figure 6 montre la carte de l'horloge avec le capteur DHT directement connecté à sa face inférieure. La souplesse des entrées-sorties permet d'ajouter bien d'autres extensions. Vous pourriez, par exemple, connecter un thermomètre 1-wire DS18B20 local, ou bien tenter de brancher un capteur distant par l'intermédiaire d'un module LoRa ; ce ne sont pas les bonnes idées qui manquent ! ⏮

(180254-B-03 – version française: Helmut Müller)

Liens

[1] « station météo à ESP32 », Elektor 01-02/2019 : www.elektormagazine.fr/180468-04

[2] « station météo à ESP32 » dans le labo d'Elektor : www.elektormagazine.fr/labs/esp32-weather-station-180468

[3] « horloge à LED géante avec Wi-Fi et mesures météo », Elektor 05-06/2019 : www.elektormagazine.fr//180254-04

[4] « horloge à LED géante avec Wi-Fi et mesures météo » dans le labo d'Elektor : www.elektormagazine.fr/labs/bedroom-clock-with-out-side-temperature-based-on-esp32

[5] BMP280 : www.bosch-sensortec.com/bst/products/all_products/bmp280

[6] VEML6070 : www.vishay.com/ppg?84277

[7] TSL2561 : https://ams.com/tsl2561

[8] « ESP32 comme serveur de temps », Elektor 07-08/2019 : www.elektormagazine.fr/180662-04

[9] Node-RED : https://nodered.org

[10] « horloge rétro à afficheurs de flipper », Elektor 03-04/2019 : www.elektormagazine.fr/180307-04

[11] FreeRTOS : www.freertos.org

[12] ESP-IDF : https://docs.espressif.com/projects/esp-idf/en/latest/

[13] « surveillance du niveau de pollution de l'air », Elektor 03-04/2019 : www.elektormagazine.fr/170182-04

PROJET DE LECTEUR

variateur pour LED sans scintillement

courant constant et haut rendement

Joost Waegebaert (Belgique)

Faire varier la luminosité d'une lampe à LED semble a priori assez simple : on prend un signal modulé en largeur d'impulsions (MLI) – tous les microcontrôleurs possèdent au moins une sortie de ce type, on amplifie le courant avec un MOSFET, et le tour est joué ! Nous avons préféré une autre approche, qui n'a rien de nouveau, mais qui permet une variation de luminosité parfaite et sans scintillement. Le tout avec des composants standard, car le montage se veut aussi éducatif.

On trouve des circuits pour faire varier la luminosité d'une LED dans tous les manuels d'électronique, certains appliquent la méthode proposée dans cet article. Elle est pourtant souvent ignorée, bien qu'elle ait des atouts importants par rapport à la MLI.

Le problème avec la MLI

L'utilisation de la modulation de la largeur d'impulsions (MLI) pour faire varier la luminosité d'une LED est bien connue. Un signal MLI est un signal rectangulaire dont la fréquence est fixe, mais dont le rapport cyclique varie. Plus ce rapport est élevé, plus longtemps la LED est alimentée, et plus sa luminosité moyenne est élevée [1].

Pendant les fractions de période où le signal MLI est à l'état bas, la LED n'émet aucune lumière, et il y a donc un scintillement. Lorsque la fréquence du signal est suffisamment élevée, notre œil ne le perçoit pas. Ces absences temporaires de lumière sont pourtant bien là, et à la longue elles peuvent occasionner une certaine fatigue oculaire, ainsi que d'autres phénomènes gênants.

Le scintillement peut être remarqué lors de la prise de photos ou d'une vidéo, si l'appareil n'est pas synchrone avec la modulation de la LED. Un objet que nous voyons parfaitement apparaîtra insuffisamment éclairé, ou pas du tout, si la prise de vue a lieu lorsque la LED n'éclaire pas. Il peut aussi y avoir un effet stroboscopique avec les objets en mouvement rapide.

La modulation rapide du courant de la LED crée des pointes de courant, avec des effets d'interférence électromagnétique (*Electromagnetic Interference*,

EMI). Il est donc hors de question de relier la LED à son pilote via de long fils ; ceux-ci deviendraient un parfait émetteur… d'interférences !

Une approche alternative

Un simple convertisseur Buck est tout à fait capable d'alimenter une LED avec un courant donné. Le principe du convertisseur est repris à la **figure 1** : l'interrupteur S1 laisse passer le courant périodiquement dans le circuit constitué par L1 et D2. Celui-ci, habituellement un MOSFET, est connecté au plus de l'alimentation, ce qui constitue un inconvénient : il faudrait idéalement le commander avec une tension supérieure à celle d'alimentation pour éviter des pertes excessives, ce qui est compliqué à réaliser.

Le problème est relativement simple à régler, il suffit de réarranger le circuit pour que l'interrupteur soit référencé à la masse. La **figure 2** en montre le principe. Il est appelé « flottant », car la charge (D2) n'est pas reliée directement à la masse. L'interrupteur S1 peut être un MOSFET à canal N standard.

Lorsque S1 est fermé, le courant dans L1 augmente ; on mesure ce courant, lorsqu'il atteint une certaine valeur, S1 est ouvert. Le courant continue à parcourir le circuit via la diode de roue libre D1 et va baisser. S1 est à nouveau fermé avant que le courant atteigne une valeur nulle. Le courant qui parcourt la LED D2 n'est pas intermittent, mais a une forme triangulaire comme montrée sommairement sur la figure 2. L'ondulation (*ripple*) du courant module bien entendu l'intensité lumineuse de la LED, mais la modulation est moindre de plusieurs ordres par rapport à un signal MLI (un signal triangulaire avec une faible ondulation comparé à un rectangulaire de forte amplitude). Le scintillement est virtuellement éliminé, de même que les interférences électromagnétiques via les fils de connexion de la LED.

Le circuit

Le schéma du convertisseur Buck flottant, appliqué à un variateur, est à la **figure 3**. Afin de bien illustrer le principe sous-jacent, nous avons utilisé des circuits classiques et des composants ordinaires.

Le comparateur IC1A compare le courant dans T1 à un courant de référence fixé par le potentiomètre R7. Lorsque T1 conduit, le courant augmente graduellement, ainsi que la tension aux bornes de R4 ; dès que celle-ci atteint celle du curseur de R7, la sortie du comparateur – à collecteur commun – bascule, C1 est

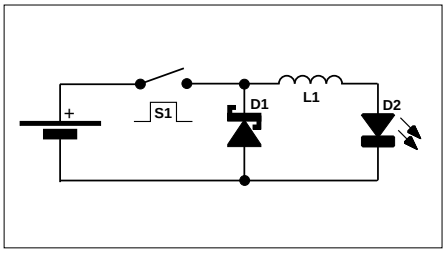


Figure 1. Le principe d'un convertisseur Buck.

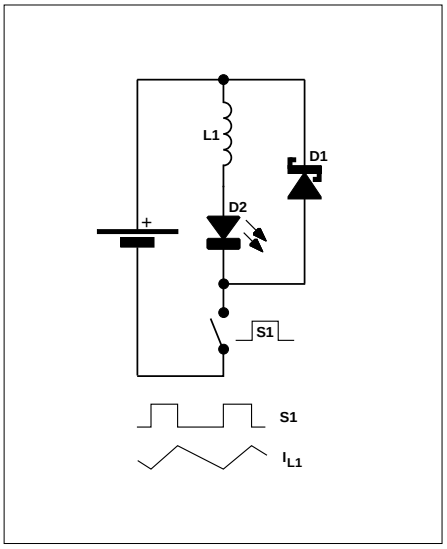


Figure 2. Le convertisseur Buck « flottant ».

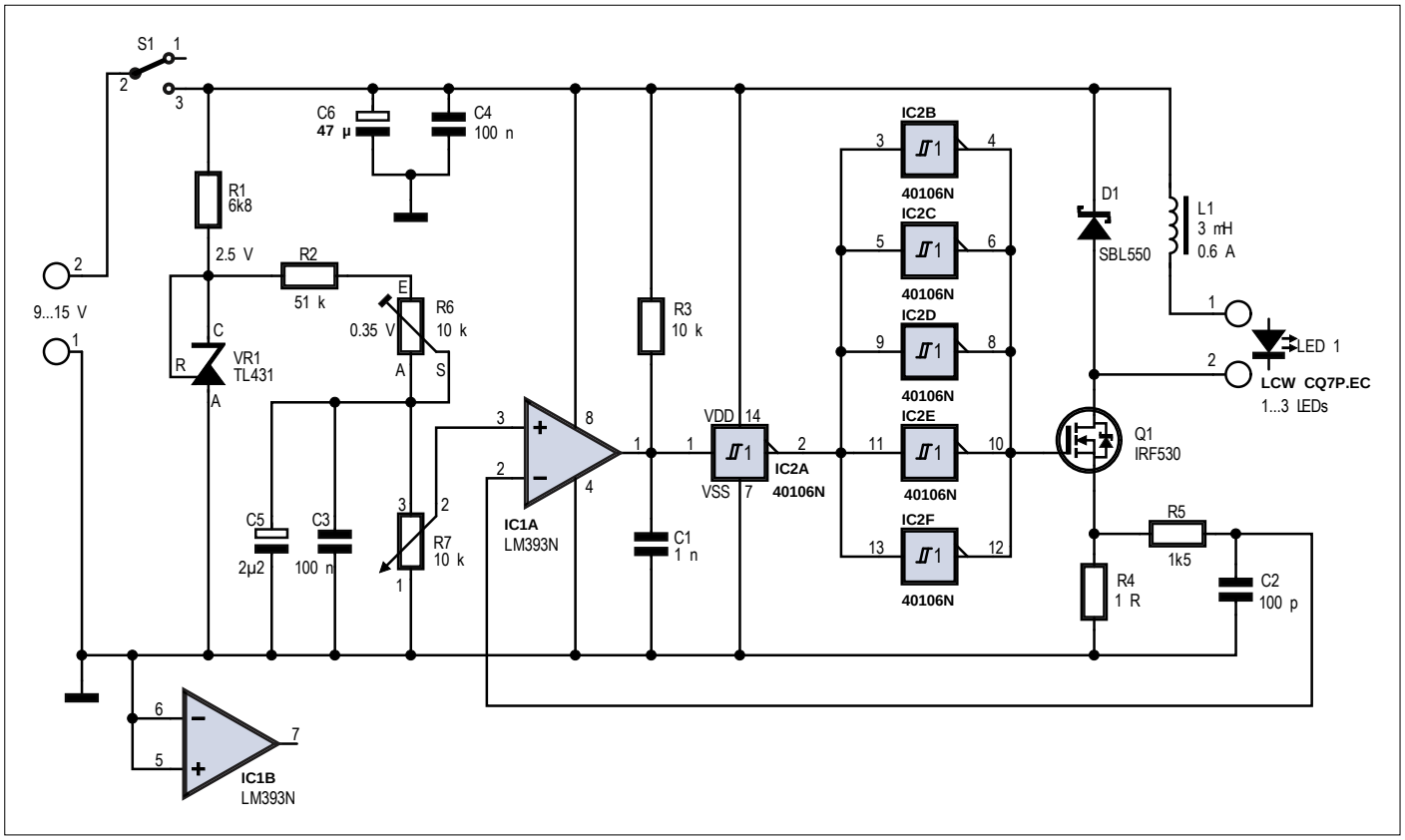


Figure 3. Le schéma du variateur à courant constant pour LED.

Un peu d'arithmétique

Le temps de blocage du MOSFET est fixé par la constante de temps R3 × C1. Le seuil de basculement des entrées du 40106 étant à environ la moitié de la tension d'alimentation, celui-ci est déterminé par la formule suivante :

$t_{OFF} \approx 0{,}7 \times R3 \times C1 = 0{,}7 \times 10^{4} \times 10^{-9} = 7\ \mu s$

Lorsque Q1 conduit, la tension aux bornes de L1 est déterminée comme suit :

$V_{L1\_ON} \approx V_{cc} - V_{LED} - I_{LED} \times R4 = 12 - 3{,}2 - 0{,}35 \times 1 = 8{,}45\ V$

Et lorsque T1 est bloqué :

$V_{L1\_OFF} \approx V_{LED} + V_{D1} = 3{,}2 + 0{,}5 = 3.7\ V$

L'évolution du courant dans L1 :

$\Delta I_{L1} \approx V_{L1\_OFF} \times t_{OFF} / L1 = 3{,}7 \times 7 \times 10^{-6} / 0{,}003 = 8{,}6\ mA$

Pourcentage de scintillement [3], en considérant que la luminosité de la LED (*luminous output*) est linéairement proportionnelle au courant I_{LED} :

$FP = (LO_{MAX} - LO_{MIN}) / (LO_{MAX} + LO_{MIN}) \approx (350 - 341{,}4) / (350 + 341{,}4) = 1{,}2\%$

Et lors d'une variation à la moitié et au cinquième de l'intensité maximale :

$FP_{50} \approx (175 - 166{,}4) / (175 + 166{,}4) = 2{,}5\%$
$FP_{20} \approx (70 - 61{,}4) / (70 + 61{,}4) = 6{,}5\%$

Le temps de conduction de Q1 :

$t_{ON} \approx L1 \times \Delta I_{L1} / V_{L1\_ON} = 0{,}003 \times 0{,}0086 / 8{,}45 = 3{,}1\ \mu s$

La fréquence du convertisseur :

$f = 1 / (t_{ON} + t_{OFF}) = 106 / (7 + 3{,}1) = 99\ kHz$

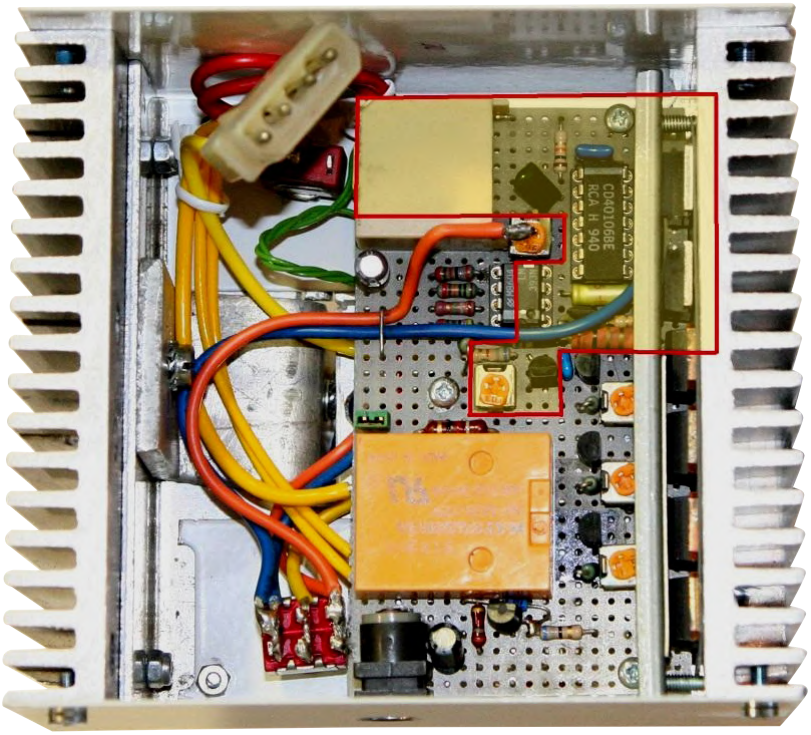


Figure 4. Le variateur de l'auteur, encadré en rouge, fait partie d'un circuit permettant d'alimenter deux LED de 1 W à partir de trois batteries lithium-ion. Les radiateurs refroidissent le régulateur linéaire d'un chargeur photovoltaïque.

déchargé et T1 est bloqué.
La tension aux bornes de R4 va alors baisser et la sortie du comparateur basculer à nouveau. Le condensateur C1 se charge via R3, et dès que le seuil de IC2A – un des six inverseurs du circuit CMOS 40106 – est atteint, T1 conduit à nouveau et le cycle recommence.
La tension sur le curseur de R7 fixe le courant maximal dans L1, et le courant minimal dépend de la constante de temps R3 × C1. La différence entre courant maximal et courant minimal constitue l'ondulation, que l'on peut facilement calculer (voir encadré).
L'intensité du courant dans la LED, fixée par R7, peut varier de 0 à une valeur déterminée par R6. Pour une LED de 1 W, le courant maximal sera de 0,35 A, avec une tension sur le curseur de R7 de 0,35 V (la valeur de R4 est de 1 Ω).
On peut connecter plusieurs LED en série, mais la chute de tension totale doit rester inférieure à la tension entre le drain de Q1 et la masse (moins la chute de tension aux bornes de L1, qui est en principe faible). On peut si nécessaire alimenter de manière indépendante cette partie du circuit avec une tension supérieure à 15 V.
La fréquence de commutation de Q1, fixée par L1, C1 et R3, est d'environ 100 kHz, ce qui est peu élevé par rapport à ce qu'il est possible de réaliser de nos jours. On pourrait l'augmenter, moyennant un comparateur plus rapide et une adaptation du circuit de commande du MOSFET, mais le plus simple est encore de se tourner vers un régulateur intégré, par ex. le ZXLD1350 [2] de Zetex (Diodes Incorporated) ; il peut fonctionner jusqu'à 1 MHz.

Les modèles choisis pour D1 et T1 ne sont pas critiques, ils étaient tout simplement disponibles chez l'auteur. Le transistor T1 doit supporter une tension drain-source (V_{DS}) d'au moins 50 V, un courant de drain (I_{DS}) d'au moins 2 A, et avoir une résistance à l'état passant ($R_{DS(on)}$) inférieure à 100 mΩ. La diode D1 est une diode Schottky supportant un courant direct d'au moins 2 A et une tension inverse de 50 V. Il n'est pas nécessaire de munir ces deux semiconducteurs d'un radiateur.

Lors de la construction – éventuellement sur un circuit imprimé – il faudra veiller à câbler toutes les masses en étoile (point central près du pied de R4).

Liens

[1] Modulation de largeur d'impulsion : https://fr.wikipedia.org/wiki/Modulation_de_largeur_d%27impulsion

[2] Feuille de caractéristiques du ZXLD1350 : www.diodes.com/assets/Datasheets/ZXLD1350.pdf

[3] Pourcentage de scintillement : www.energy.gov/sites/prod/files/2015/05/f22/miller%2Blehman_flicker_lightfair2015.pdf

Les résultats

Le pourcentage de scintillement (*Percent Flicker*) est inférieur à 1,5% avec le circuit décrit ici (voir encadré) ; c'est peu comparé aux 6,5% d'une lampe à incandescence. Ce pourcentage augmente lors de la variation de la luminosité, car le courant dans la LED est constant et la constante de temps n'est pas nulle. L'emploi du ZXLD1350 permettrait de meilleurs résultats, car il maintient l'ondulation à une fraction fixe du courant dans le circuit : un courant plus faible entraîne une plus petite ondulation, et donc un pourcentage de scintillement constant.

Conclusion

Ce circuit, conçu avec des composants traditionnels, régule la luminosité d'une LED avec un courant constant. L'effet de scintillement est virtuellement éliminé, quelle que soit l'intensité lumineuse choisie. Les fils de connexion de la LED ne produisent quasiment pas d'interférences électromagnétiques. La **figure 4** montre l'exemplaire construit par l'auteur.

(190062-04 –
version française : Jean-Louis Mehren)

Guiding Innovation Forward

Arrow Electronics
est un fournisseur global de produits,
de solutions et de services
destinés aux utilisateurs industriels
et commerciaux de composants électroniques
et de solutions informatiques professionnelles.

www.arrow.com

Nous sommes le seul partenaire à offrir :

- Un large panel de solutions électroniques,
  représentant plus de 250 fabricants de composants
- Une offre complète qui va du Capteur jusqu'au Cloud (From Sensor to Cloud)
- Un accès rapide et détaillé aux technologies de pointe les plus avancées

---

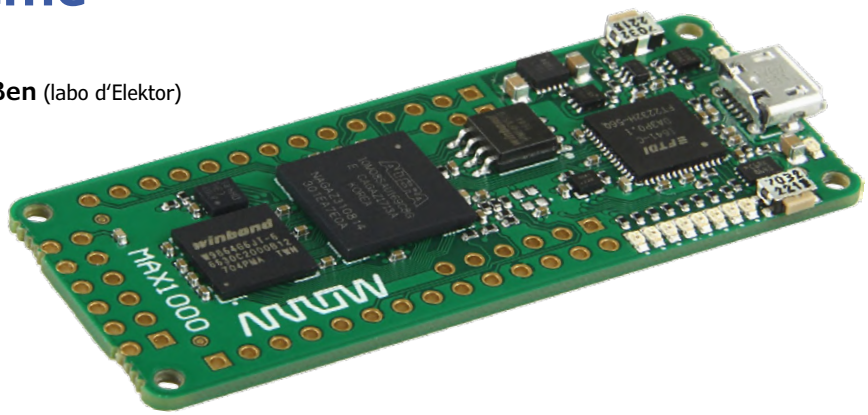
Le but d'Arrow est d'aider les startups à :

- Elaborer leurs solutions électroniques avec le support local d'ingénieurs d'applications
  (conseil technique en électronique gratuit)
- Développer leur POC (proof of concept/ prototype) en mettant à leur disposition
  un écosystème de partenaires (design house, sous-traitants)
- Les accompagner jusqu'à l'industrialisation de leur produit
  (support à la conception, optimisation des couts, gestion de la supply chain globale)

projet SCCC (4)

processeur *softcore* et compilateur C à construire soi-même

Martin Oßmann (Allemagne) et **Mathias Claußen** (labo d'Elektor)



Dans cet article de la série, nous nous occuperons de la sortie de signaux analogiques. Pour cela, nous utiliserons des CN/A sigma-delta et des interruptions.

Dans l'article précédent de la série [3], nous avons vu de quelles unités périphériques notre CPU à construire soi-même dispose en particulier d'un MLI (modulateur en largeur d'impulsion, en anglais PWM = pulse width modulator), d'un convertisseur A/N, d'une interface GPIO et enfin, mais ce n'est pas le moins important, d'une UART utilisateur. Ces unités sont déjà affectées à des voies d'entrée/sortie dans le code *Verilo*g. Nous avons aussi vu comment l'utilisateur peut interconnecter ces différents périphériques aux broches de la FPGA, pour exploiter (lire, commander) des éléments externes, par ex. un capteur d'accélération. Il s'agissait en fin de compte de voir comment exploiter la périphérie dans le programme C qui pilote notre processeur. Nous vous recommandons vivement de relire l'ensemble avant d'aborder l'intéressante expérience qui suit.

Expérience 6

Dans cet expérience 6, nous nous familiariserons avec l'utilisation d'un CN/A sigma-delta et des interruptions. Commençons par le CN/A. La FPGA *MAX1*0 elle-même ne comporte pas de sortie CN/A. Comme avec de nombreux microcontrôleurs, nous pourrions utiliser la modulation en largeur d'impulsions (MLI). Nous préférerons avoir recours à des CN/A de type sigma-delta. Dans le module de la couche supérieure, nous créons deux instances d'un module du genre `firstOrderSigmaDeltaDac1v01` qui envoient leurs signaux sur les broches PMOD1 et PMOD2 de la FPGA. Les données font 8 bits de large et sortent sur les voies n°19 et 20 avec l'instruction `OUTA` (pour les périphériques cités ci-dessus, seules les voies 0 à 7 et 18 sont utilisées) :

```
reg[8-1:0] aDacSignal ;
reg[8-1:0] bDacSignal ;
firstOrderSigmaDeltaDac1v01 #(.MSBI(7)) DACa
   (clk_100, aDacSignal, PMOD1) ;
firstOrderSigmaDeltaDac1v01 #(.MSBI(7)) DACb
   (clk_100, bDacSignal, PMOD2) ;

always @(posedge sCclk) begin
  if(outAstrobe) begin
    if( outAchannel==0) begin   ... end
     else if( outAchannel==19) begin aDacSignal
   <=outA[8-1:0] ;   end
    else if( outAchannel==20) begin  bDacSignal
   <=outA[8-1:0] ;  end
    end
  end
```

Le code sigma-delta lui-même est le suivant :

```
module firstOrderSigmaDeltaDac1v01  #(parameter
   MSBI=7)
 (input            clk ,
  input  [MSBI:0]  DACsigIn ,
  output reg       DACout
  );

reg [MSBI:0]   DACin;
// entrée CN/A (complément à 2 MSBI)
reg [MSBI+2:0] DeltaAdder;
// sortie de l'additionneur delta
reg [MSBI+2:0] SigmaAdder;
// sortie de l'additionneur sigma
reg [MSBI+2:0] SigmaLatch;
// sortie verrouillée de l'additionneur sigma
reg [MSBI+2:0] DeltaB;
// entrée B de l'additionneur sigma
always @(SigmaLatch)              DeltaB = <<
```

```
   (MSBI+1);
always @(DACin or DeltaB)          DeltaAdder = DACin
   + DeltaB;
always @(DeltaAdder or SigmaLatch) SigmaAdder =
   DeltaAdder + SigmaLatch;

always @(posedge clk) begin
  DACin <= DACsigIn ;
  SigmaLatch <= SigmaAdder;
  DACout <= SigmaLatch[MSBI+2];
  end

endmodule
```

Un CN/A sigma-delta présente moins de bruit à basse fréquence qu'un CN/A MLI. La **figure** 1 donne le schéma du raccordement. Dans le logiciel en C, nous utilisons les instructions `OUTA` à l'intérieur d'une fonction adéquate. Dans ce programme, cette fonction attend toujours que le programme d'interruption fasse passer à 1 la variable `mail`. Ensuite, cette variable est réinitialisée et les paramètres x et y de la fonction sont sortis respectivement sur les voies 19 et 20.

```
#asm
SigmaDeltaDacA     EQU  19
SigmaDeltaDacB     EQU  20
#endasm


outDACs(int x, int y) {
  while(mail==0){ }
  mail=0 ;
  x ;
#asm
  OUTA   SigmaDeltaDacA
#endasm
  y ;
#asm
```

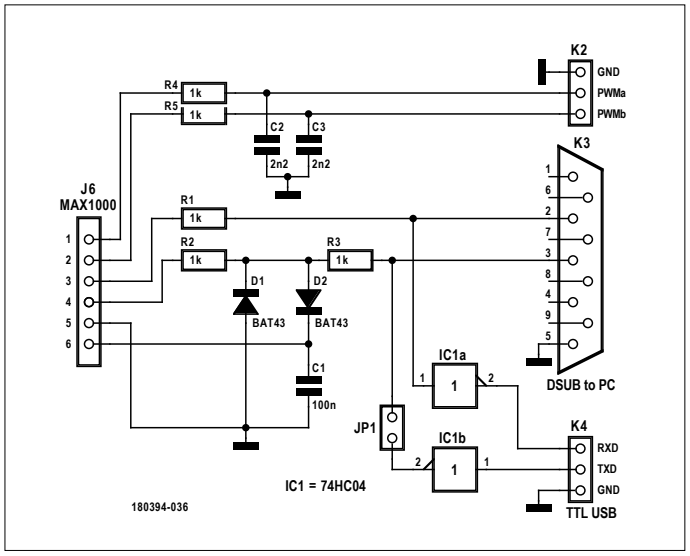


Figure 1. Sortie des signaux.

```
  OUTA   SigmaDeltaDacB
#endasm
  }
```

La routine `outDACs` est assez courte pour publier le listage complet produit par le compilateur :

```
outDACs        EQU $
// Début de la fonction outDACs
// Le paramètre x est à l'adresse 2 relativement à la
   pile
// Le paramètre y est à l'adresse 1 relativement à la
   pile
// L'adresse de retour est à l'adresse 0 relativement
   à la pile

__LBL00000     EQU $       ; whileStackBase=0
          LDPUSH      @ __mail
       // Valeur de mail sur la pile
          LDI         0  ; hex=00000000
    // Valeur de 0 dans R0
          POP         R1
// R1 = valeur de mail de la pile
         CMPEQ
// Comparaison sur ==
        JFALSE      __LBL00001
// Si FAUX, sort de la boucle
        JMP         __LBL00000
// Continue à exécuter la boucle
__LBL00001 EQU $
        LDI         0  ; hex=00000000
// Valeur 0 vers R0
       ST          @ __mail
       // Enregistre 0 de R0 dans mail
       GETLCL      2
      // variable locale x = 2, rel. à la pile zsp=0
       OUTA        SigmaDeltaDacA
      // Envoie la valeur au DACa
       GETLCL      1
       // Valeur y vers R0
       OUTA        SigmaDeltaDacB
     // Envoie la valeur au DACb
       RET
       // Retour de la fonction
```

Les instructions `LDPUSH` et `GETLCL` ont été implémentées par l'optimiseur, et ce selon l'idée suivante :

les instructions

```
    LD          @ __mail
    PUSH        R0
```

deviennent

```
    LDPUSH      @ __mail
```

et de même

```
    LDI         2
    IADD        SP
    LDIND       @R0
```

deviennent

```
    GETLCL      2
```

Grâce à cette optimisation, le programme est plus court et plus rapide. Venons-en aux interruptions.

Interruptions

Dans le CPU *sCCC*P actuel, nous avons prévu quatre interruptions, nous n'en avons utilisé cependant que deux (`RTCinterrupt` et `uartRXready-Interrupt`). Le traitement des interruptions est effectué par le CPU dans la phase *fetc*h (chargement d'instruction). Normalement, l'instruction effective y est récupérée. Cependant si une interruption est survenue (`selectedIRQ != 0`), le déroulement est différent. Le compteur de programme (CP) est sauvegardé sur la pile. Ensuite le CP prend la valeur 8 * numéro d'interruption. Pour l'interruption IRQ1, le compteur saute à 8, pour l'interruption IRQ2 il saute à 16, et ainsi de suite. L'écriture de la valeur sauvegardée du CP a lieu concurremment à la phase *fetc*h suivante qui survient directement à la fin de la phase *fetc*h en cours (celle de l'interruption). En code Verilog du CPU, le listage est le suivant :

```
if(selectedIRQ) begin
  writeAddress <= sCregSP-1'b1 ;
  writeValue   <= sCregPC ;
  writePending <= 1 ;
  sCregSP <= sCregSP-1'b1 ;
  sCregPC <= selectedIRQ << 3 ;
  sCstate <= sCstateFetch ;

  if(selectedIRQ==1) begin irq0reg<=0 ; end
  ...
  end
```

Pour exécuter une routine C déclenchée par une interruption, nous devons programmer le code ci-dessous pour l'interruption *RT*C.

```
#asm
RTCintVector       EQU   8

  ORG      RTCintVector
  PUSH     R0
  PUSH     R1
  CALL     RTCinterrupt
  POP      R1
  POP      R0
  RET
#endasm

...

int mail ;

RTCinterrupt(){
  mail=1 ;
  }
```

Dans la routine d'interruption elle-même, nous nous contentons d'attribuer la valeur 1 à la variable `mail`. Il nous manque cependant ce qui suit. Dans le programme principal, le `RTCtimerTOP` est initialisé à 50, c.-à-d. qu'une interruption intervient tous les 50 cycles. L'instruction `setCPUflags(RTCintEna)` autorise le fonctionnement sous interruption *RT*C. Ensuite, nous appelons la fonction `do Samples` qui envoie 32 valeurs successives aux deux voies du CN/A.

```
#asm
CPUflagsChannel    EQU   6
RTCtimerTopChannel EQU   7
#endasm

doSamples(){
// written by SinTabGen1v01.lua
// 128+127*sine(Xi)
while(1){
  outDACs(  128 ,    0 ) ;
  outDACs(  153 ,    8 ) ;
  .....
  outDACs(  103 ,  248 ) ;
  }
}

#define RTCintEna 1

main(){
  setRTCtimerTop(50) ;
  setCPUflags( RTCintEna ) ;
  doSamples() ;
  }
```

L'expérience elle-même est, là encore, relativement simple à conduire : nous synthétisons le projet Quartus *experiment6.qp*f et chargeons la FPGA avec le fichier *Bitfil*e. Nous compilons ensuite le code C *experiment6.*c et le chargeons dans le CPU à l'aide du chargeur de processus (*Processing Uploade*r). Ensuite, une sinusoïde et une dent de scie devraient respectivement apparaître sur les sorties MLI (PWM) 1 et 2.

Expérience 7 : oscilloscope à échantillonnage

Jusqu'à maintenant, les expériences ont revêtu un caractère essentiellement démonstratif. Nous allons maintenant faire un essai plus proche d'une application réelle. À l'aide du CA/N de la puce *MAX*10, nous représenterons des données et les enverrons à un PC par une interface série. Le PC affichera alors les données (fonction « oscilloscope »). Via une transformation de Fourier rapide (FFT), nous pouvons naturellement envoyer au PC le spectre du signal pour offrir une fonction « analyseur de spectre ».
Pour réaliser cette application, nous travaillons sur trois niveaux. Au niveau de la FPGA, nous élargissons notre CPU de sorte que nous écrirons 8192 échantillons dans un tampon FPGA, en faisant fonctionner le CA/N à sa vitesse maximale d'échantil-

lonnage, soit 1 Méch./s. Ensuite, nous lisons le tampon à l'aide d'un programme CPU et nous envoyons les données au PC. Le troisième niveau consiste pour le PC à afficher les valeurs, c'est-à-dire calculer le spectre.
La lecture des valeurs dans le FPGA se déroule comme suit : le pointeur d'écriture `traceWritePtr` joue un rôle central ; au repos, il a la valeur 8191. S'il est inférieur à 8191, des valeurs sont lues jusqu'à ce que le compteur atteigne à nouveau 8191. Pour démarrer la lecture, il suffit de ramener ce compteur à zéro ; ensuite, 8192 valeurs sont enregistrées dans le tampon `traceBuffer`. Le pointeur d'écriture est réinitialisé avec une instruction `OUTA`. Il est aussi possible de lire le pointeur d'écriture sur la voie 32, de sorte que l'on peut vérifier si toutes les valeurs ont déjà été lues.

Pour lire les valeurs du tampon, nous utilisons un pointeur de lecture, `traceReadPtr` que l'on peut initialiser par une instruction `OUTA` sur la voie 33. La valeur dans le tampon à cette adresse est lue et disponible (voie 33) via une instruction `INPA` :

```
parameter traceLength = 8192 ;
parameter traceLengthWidth = 13 ;
reg [12-1:0] traceBuffer[0:traceLength-1] ;
reg [traceLengthWidth-1:0] traceWritePtr ;
reg [traceLengthWidth-1:0] traceReadPtr ;
reg trigger ;

// Impulsion eocPulse de fin de conversion (provient
   du CA/N)

always @(posedge sCclk) begin
   if ( (outAstrobe) & ( outAchannel==32) ) begin
   trigger<=1 ; end
   if (eocPulse) begin
      if(trigger) begin
         traceWritePtr<=0 ;
         trigger<=0 ;
         end
      if(traceWritePtr != traceLength-1 ) begin
      traceBuffer[traceWritePtr] <= ADCval ;
      traceWritePtr <= traceWritePtr+1 ;
      end
     end
  end

always @(posedge sCclk) begin
  if(outAstrobe) begin
    if( outAchannel==33) begin traceReadPtr<=outA ;
      end
    end
  end

assign inpA=
  (inpAchannel==0) ?  { 32'h12341234   } :
  .....
  (inpAchannel==32) ? traceWritePtr :
  (inpAchannel==33) ? traceBuffer[traceReadPtr] :
  32'h12341111 ;
```

E*n Veri*log, le programme fait au total moins de 40 lignes. Lors de traitement de signaux, il arrive souvent que la fonction FPGA soit réalisée par une petite portion de code. Il faut maintenant envoyer des valeurs du tampon au PC. L'écriture en Verilog au niveau FPGA est relativement gourmande en ressources. Il est plus simple de mettre à contribution notre CPU logiciel. Le protocole série entre la carte *MAX1*000 et le PC se déroule comme ci-dessous. Lorsque le PC envoie u*n* x, la FPGA commence le processus d'échantillonnage et interroge le CA/N, jusqu'à écrire 8192 valeurs dans `traceBuffer`. Ensuite, le CP*U sC*CCP récupère les données du tampon FPGA et les envoie au PC par communication série à 115200 bauds. Il envoie ensuite une somme de contrôle. Dans notre exemple de programme *experiment*7.c, nous pouvons également voir comment lire les données par CP*U sC*CCP. Le code proposé est le suivant :

```
doCommand(){
  int c ;
  c=getCharPolling() ;
  if(c=='x') { execute() ; return ; }
  putChar('?') ;
  }

execute(){
  int data,k,i,chkSum ;
  traceTrigger() ;
  // reset traceWritePtr=0
  while( getTraceWritePtr() !=traceLength-1) { }
  // attendre jusqu'à la fin
  chkSum=0 ;
  putChar('[') ;
  // opens frame
  for(k=0 ; k<frameLength ; k++){
      // boucler sur les valeurs
      setTraceReadPtr(k) ;
      data=getTraceData() ;
      // récupère traceData[k]
      word12Out(data) ;
      // sortir 3x4 bits hex (12 bits)
      chkSum=chkSum+data ;
    // mettre la somme de contrôle à jour
      putChar('.') ;
      // mettre un . derrière chaque valeur
      if( (k&0x1F)==0x1F ){ crlf() ; }
     // améliore la lisibilité
    }
  word12Out(chkSum) ;
  // envoie la somme de contrôle au PC
  putChar(']') ;
  // ferme la trame de données
  crlf() ;
  }
```

L**a figur**e 2 donne le câblage de l'entrée analogique *AIN*0. Nous commençons par placer le cavalier JP1 et laissons l'entrée analogique *Analog* in en l'air. De cette manière, le signal MLI arrive sur le CA/N. Ultérieurement, nous retirerons le cavalier pour relier l'entrée *Ana*log in au signal via un condensateur

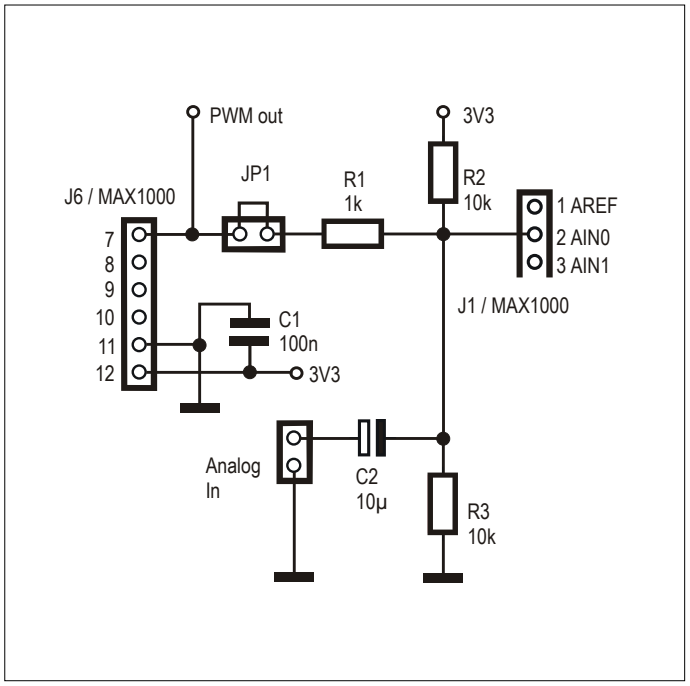


Figure 2. Câblage de l'entrée analogique *AIN0*.

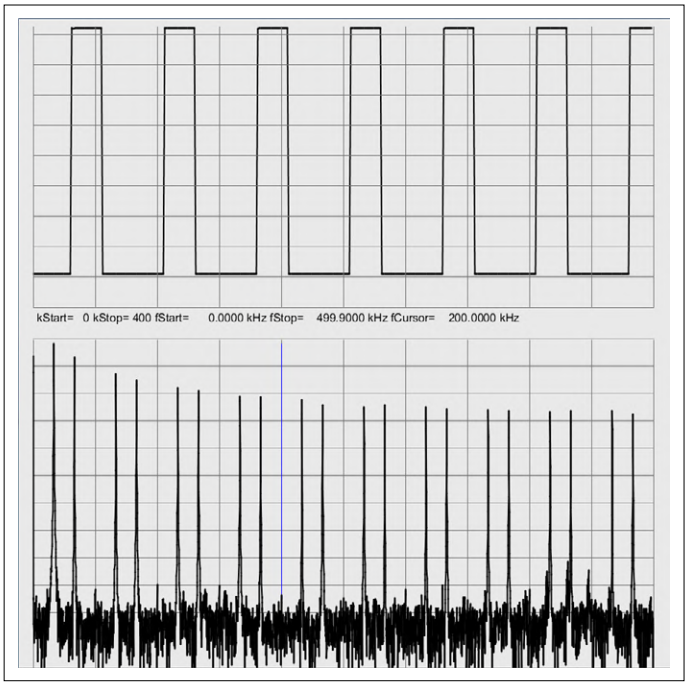


Figure 3. En haut, données échantillonnées ; en bas, le spectre correspondant.

(filtre passe-haut).
Le signal MLI est produit par le code ci-dessous :

```
setPWMtop(2000) ;
setPWMperiod(6000-1) ;
```

La période vaut 6000 fois le cycle MLI de 100 MHz, soit 60 µs. Le temps *on-T*ime est de 2000 cycles, c.-à-d. 20 µs. Nous avons maintenant besoin d'un programme côté PC pour récupérer les valeurs échantillonnées et les afficher. À cet effet, nous utilisons le croquis de traitement *C:/sCCCP/experiment7/Processing/ShowGraph1/ShowGraph1/ShowGraph1.*pde.
Avant de commencer, nous devons adapter la troisième ligne *(String serialPort=»COM*2») de façon à utiliser le port COM auquel le port utilisateur de l*a MAX10*00 est relié. À l'aide d'un programme de terminal, nous pouvons aussi vérifier que le programme qui tourne sur l*a MAX10*00 est correct. Si l'on envoie un x *(eXec*ute) à l*a MAX1*000, cette dernière doit répondre avec un grand nombre de données du style [804.842.....].
Le croquis de traitement d'affichage de ces valeurs produit un écran tel que celui de l**a figur**e 3. La moitié supérieure montre les données échantillonnées. La moitié inférieure constitue le spectre.
Avec cette application, nous sommes encore très loin d'un oscilloscope numérique à échantillonnage (DSO) ou d'un analyseur de spectre. Pour cela, il faudrait débuter l'enregistrement au moyen d'un signal de déclenchement. En outre, il faudrait programmer une interface graphique utilisateur (GUI) convenable. Comme amélioration, nous pourrions utiliser un CA/N nettement plus rapide pour représenter des signaux de plus haute fréquence. Et si, au lieu d'afficher les échantillons CA/N, nous affichions les entrées numériques, nous réaliserions un analyseur logique.
Dans le prochain et dernier article de la série, nous perfectionnerons notre matériel : l'émetteur *DD*H47 du service météo envoie des messages météorologiques sur 147,3 kHz. L'expérience 8 reçoit ces données et les affiche sur un moniteur VGA.

(180394-D-04 – version française : Yves Georges)

Liens

[1] « projet SCCC (1) », Elektor 03-04/2019 : www.elektormagazine.fr/180394-04
[2] « projet SCCC (2) », Elektor 05-06/2019 : www.elektormagazine.fr/180394-B-04
[3] « projet SCCC (3) », Elektor 07-08/2019 : www.elektormagazine.fr/180394-C-04
[4] Page de l'article : http://www.elektormagazine.fr/180394-D-04

PROJET DU LABO

Horticulture Box
éclairage de plantes
à « LED horticoles » de Würth Elektronik

Luc Lemmens et **Matthias Claußen** (labo d'Elektor), avec la contribution de **Würth Elektronik**

L'utilisation et la recherche de sources lumineuses électriques pour stimuler la croissance des plantes remontent à la seconde moitié du 19e siècle. À cette époque, l'éclairage électrique poursuivait son essor au travers de trois voies principales : les lampes à incandescence, à arc à l'air libre, et à décharge dans un gaz. Aujourd'hui les LED rayonnent à quasiment chaque coin d'appareil ou de rue, y compris au profit des plantes grâce à cet éclairage « horticool ».

La théorie qui sous-tend la croissance végétale par LED étant trop complexe pour être exposée ici, je vous renvoie d'emblée aux trois notes d'application ANO002, ANO003 et ANO004 de Würth Elektronik [1]. Je me contenterai ici d'évoquer l'avantage que procurent les LED par rapport aux sources de lumière conventionnelles.
Lorsqu'il est question de la croissance et du développement des plantes, le spectre de la lumière importe autant que son intensité. Voici les effets connus de quelques bandes de longueurs d'onde typiques :

- Le rouge (630-660 nm) est le principal moteur de la photosynthèse et un agent important de la croissance des tiges. Il peut également réguler la floraison, la dormance et la germination des graines.
- Le bleu (400-520 nm) joue également un rôle clé dans la photosynthèse, mais une surexposition à sa lumière peut inhiber la croissance. Cette partie du spectre doit donc être contrôlée et combinée à d'autres fréquences. Le bleu a également été associé à la régulation de la concentration en chlorophylle, à la croissance des bourgeons latéraux et à l'épaisseur des feuilles.
- Le rouge lointain (720-740 nm), qui fait partie du spectre IR, affecte la germination et peut avancer la date de floraison, mais aussi augmenter la longueur des tiges via un mécanisme appelé « évitement de l'ombre ».
- Le vert (500-600 nm) a longtemps été considéré comme ayant un effet négligeable sur le développement des plantes. Des études récentes ont cependant montré que le mécanisme d'évitement de l'ombre était déclenché par cette plage de longueurs d'onde chez des plantes subissant l'ombre d'autres plantes.
- La lumière UV (280-400 nm) reste très expérimentale en horticulture. Certains végétaux tels que la laitue et les tomates s'avèrent résistants à ces longueurs d'onde pourtant mutagènes. Le spectre UV peut déclencher la production de molécules protectrices bénéfiques pour l'homme, comme des antioxydants et des phénols.

Contrairement aux sources de lumière plus anciennes, il est possible de fabriquer des LED émettant des bandes de longueurs d'onde spécifiques. On peut donc satisfaire les besoins lumineux d'une plante en combinant des LED de différentes couleurs. Würth Elektronik offre à cet effet la série de LED CMS monochrome *WL-SMDC Waterclear* à boîtier céramique. Cette série a été étendue aux longueurs d'onde 450 nm (bleu foncé), 660 nm (hyper rouge) et 730 nm (rouge lointain) pour couvrir le spectre d'absorption des pigments photosynthétiques. Notre système comprend ces trois variantes de LED ainsi que des LED à blanc froid (6000 K).
Ces LED sont commandées via Wi-Fi par l'ESP32 et une interface web accessible depuis un ordinateur, un ordiphone ou une tablette. Comme l'interface permet de commander l'intensité totale ou individuelle de ces quatre canaux, vous pourrez facilement expérimenter différentes recettes d'éclairement.
Il y a bien d'autres avantages à utiliser des LED, en particulier leur taille, leur rendement et leur durée de vie.

INFOS SUR LE PROJET

LED PWM ESP32
éclairage pour plantes
horticulture

débutant
connaisseur
expert

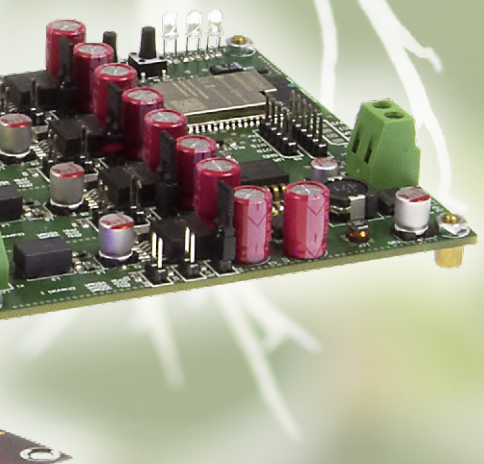
env. 2 h

EDI Arduino (optionnel)

env. 165 €

Carte pilote

Une grande partie du circuit provient du schéma de référence d'un pilote de LED par MLI à 4 canaux de Würth Elektronik. Luc Lemmens et Matthias Claussen, du labo d'Elektor, ont remplacé le contrôleur PIC et le module Bluetooth d'origine par le populaire module ESP32-WROVER-B.



Caractéristiques

- 4 canaux de couleur : rouge, rouge lointain, bleu, blanc
- LED haute luminosité WL-SMDC de Würth Elektronik
- Limitation de courant des LED : 300 mA / 350 mA / 450 mA
- 4 pilotes MLI (*PWM*)
- Commandé par ESP32
- Intensité et mélange de couleurs réglables
- Cycles jour/nuit définissables par temporisateur
- Surface d'éclairement : rayon de 55 cm
- Paramétrable par Wi-Fi (interface web / MQTT)
- Adapté à toute alimentation de 24 V, 50 W
- Carte pilote et carte des LED préassemblées disponibles

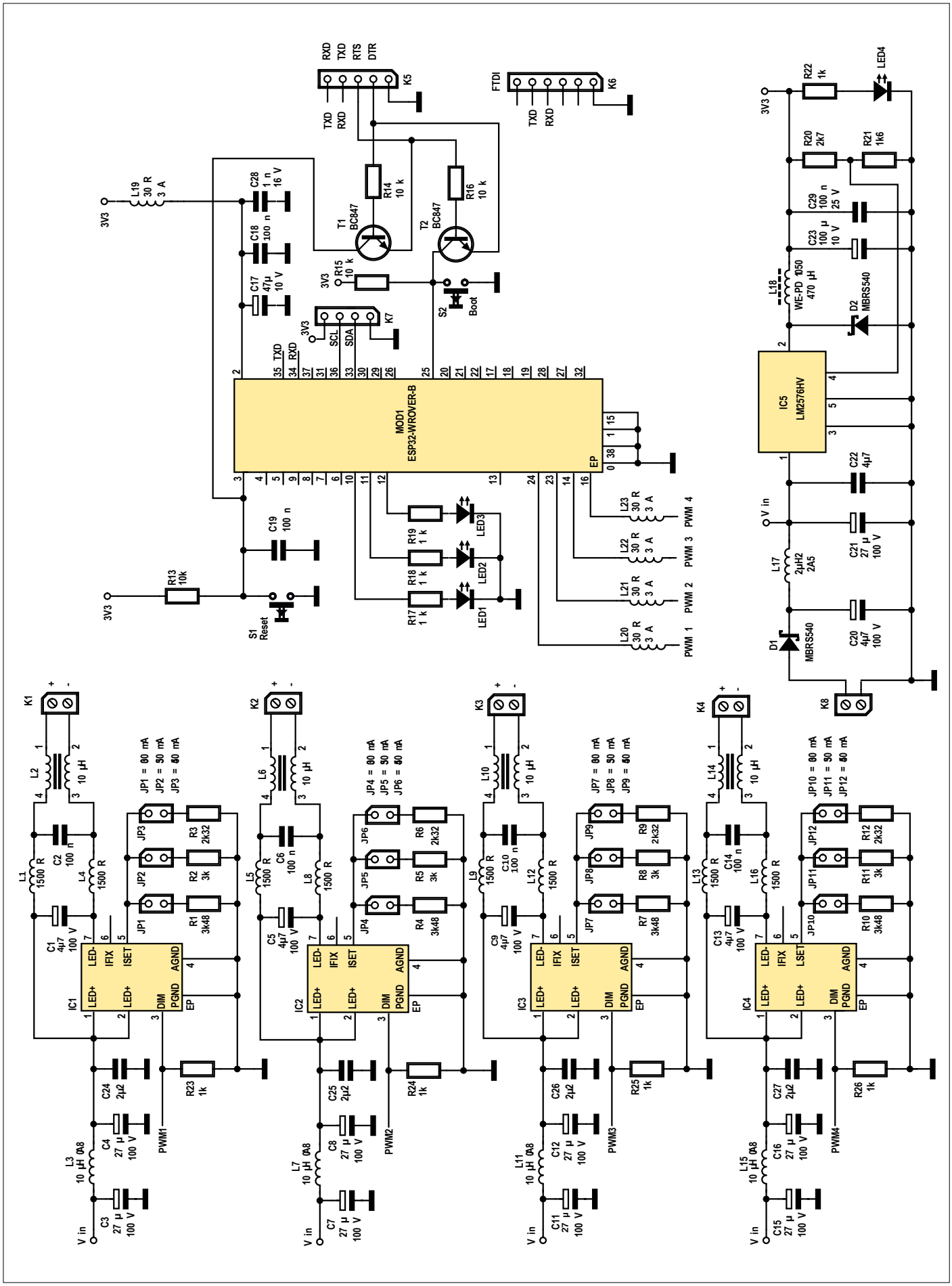


Figure 1. Schéma de la carte pilote du système d'éclairage horticole. Le module ESP32 y trône tel le Roi-Soleil.

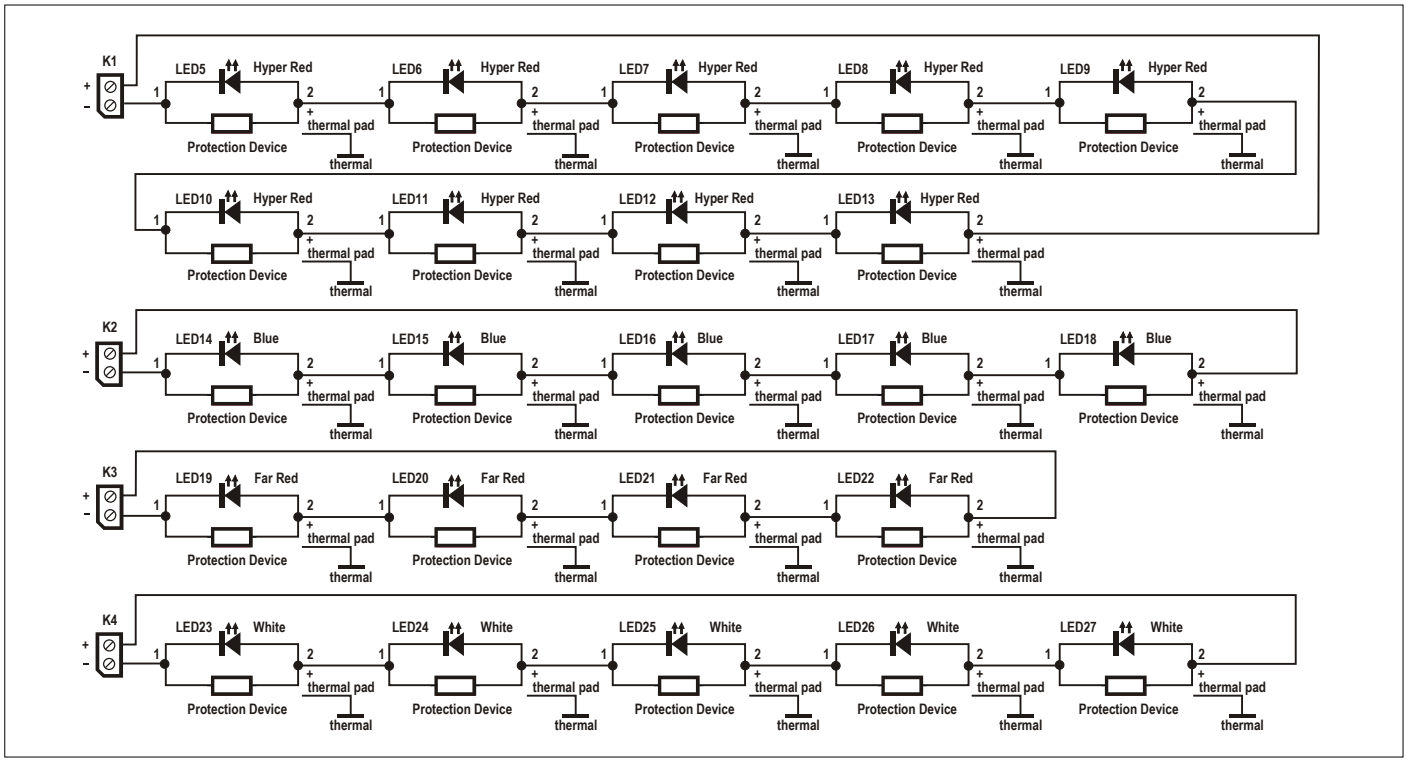


Figure 2. Schéma de la carte des LED.

Luc a aussi décidé d'alimenter l'ESP32 avec un convertisseur CC/CC LM2576 plutôt qu'avec le module de puissance original de Würth Elektronik.

Le schéma de la **figure** 1 montre le résultat de ces modifications. Il comprend l'ensemble des condensateurs et inductances implantés pour satisfaire la norme de compatibilité électromagnétique EN 55015/CISPR 32, même lorsque le gradateur est utilisé. Les quatre étages pilotes avaient passé sans problème les tests de préconformité, et pour cette partie nous avons pratiquement recopié le schéma et le dessin du circuit imprimé (CI) original. Il n'en a pas été de même pour l'ESP32, qui présentait de sérieux problèmes avec un cumul d'harmoniques de 80 MHz et plus dans le spectre. Sans surprise, 80 MHz correspondait à la fréquence d'horloge de l'ESP32. Une refonte partielle de la carte s'imposait. Nous avons notamment redéfini le plan de masse et ajouté des perles de ferrite.

Le module ESP32-WROVER produit les signaux MLI (*PW*M) contrôlant l'intensité des LED, et est bien sûr le cerveau de la carte pilote. Il régit l'interface utilisateur ainsi que la connexion au réseau Wi-Fi local. Le convertisseur abaisseur LM2576 (IC5) délivre la tension de 3,3 V au module ESP32 (MOD1). La diode LED4 (bleue) indique que la carte est sous tension. Même si la tension d'entrée de l'éclairage accepte jusqu'à 45 V, nous avons décidé de la limiter à 24 V afin de restreindre la dissipation de chaleur par le CI lorsque les LED sont allumées.

Il est possible d'accéder à l'UART de l'ESP via K5 ou K6 soit pour une communication série, soit pour la programmation de l'amorce (*bootloade*r). Les connexions aux broches DTR et RTS de K5 permettent à l'EDI Arduino de placer automatiquement l'ESP32 en mode chargeur d'amorce via T1 et T2. Vous aurez besoin d'une interface USB vers UART fournissant ces deux signaux de mise en liaison. La passerelle USB/série BoB-FT232R ou la carte de liaison FT231X feront l'affaire. Si vous optez plutôt pour un câble FTDI standard de 3,3 V relié à K6, vous aurez besoin – comme vous l'ont sans doute appris d'autres projets Elektor à ESP – de deux poussoirs S1 et S2 pour réinitialiser l'ESP et le mettre en mode chargeur d'amorce.

Vous n'aurez pas à vous soucier de la programmation du micrologiciel si vous utilisez les deux cartes préassemblées par Elektor, et la mise à jour du micrologiciel peut se faire par liaison sans fil. C'est exact, ni câble ni interface ne sont nécessaires pour effectuer une mise à jour.

Les signaux de 3,3 V des lignes SDA et SCL du bus I²C peuvent être reliés à K7 pour de potentielles évolutions futures. Des résistances de rappel pourraient s'avérer nécessaires si vous connectez un dispositif à ce bus I²C. Et souvenez-vous que les broches d'E/S de l'ESP32 ne tolèrent pas le 5 V, donc vous pourriez également avoir besoin de modules de décalage de niveau bidirectionnels 5 V vers 3,3 V.

L'ESP délivre les signaux MLI (*PW*M) aux entrées DIM des pilotes de LED IC1 à IC4. Le courant de sortie est limité à 300 mA, 350 mA ou 450 mA en fermant un des trois cavaliers associés à chaque pilote.

LED1 (verte) s'allume lorsque l'ESP32 est connecté à un réseau Wi-Fi et est en mode Station. Elle clignote lorsque l'ESP32 est en mode Point d'accès. LED2 (rouge) s'allume lorsqu'une mise à jour par liaison sans fil est en cours. LED2 (jaune) signale une connexion MQTT active.

Carte des LED

Le schéma de la **figure** 2 laisse peut-être à penser que l'élaboration de la carte des LED aura été la partie la moins compliquée du projet, mais les apparences sont trompeuses. Il nous a d'abord fallu décider des couleurs des LED et de leur nombre. N'ayant aucune expérience dans ce domaine, nous avons demandé aux

Figure 3.
Alimentation à découpage de 24 V / 2,2 A, modèle LRS-50-24 de Mean Well.

Gestion de la chaleur

Une conduction verticale de la chaleur au moyen de *vias* thermoconducteurs placés directement sous la surface de refroidissement des LED peut aider, de même qu'une conduction horizontale via de larges surfaces de cuivre sur les couches supérieure et inférieure.
C'est la méthode suivie par Würth Elektronik pour sa technologie *Heatsink*. La chaleur ainsi répartie est acheminée au travers d'une large surface vers un dissipateur en aluminium fixé sur un adhésif de transfert. L'avantage par rapport à la technologie IMS est la possibilité d'assembler des circuits multicouches.
Les « thermovias » sont des trous répartis sur le circuit imprimé pour faciliter le transfert de chaleur. On les place idéalement sous la source de chaleur pour obtenir une dissipation directe. Leur diamètre typique est de 0,30 mm, mais des diamètres plus grands sont possibles ; l'épaisseur de leur manchon de cuivre mesure au moins 25 µm.
Le remplissage et la fermeture de ces *vias* par un « couvercle » de cuivre évitent les flux de soudure, les composants pouvant sans problème être soudés sur les pastilles.
La résistance thermique et les températures sur et dans le CI peuvent être calculées avec une bonne approximation par analyse ou simulation numériques. En plus de la conduction verticale, la chaleur est répartie latéralement sur une plus grande surface du CI. Une grande surface est nécessaire pour accroître le transfert vertical en raison de la présence de matériaux moins thermoconducteurs, p. ex. les couches d'isolation entre CI et dissipateur thermique.
Un refroidissement par diffusion est souvent obtenu par dispersion de la chaleur dans les couches de cuivre, ou par fixation du CI sur un dissipateur thermique. Les dissipateurs en aluminium sont connus pour leur efficacité. Ils sont fixés soit par pressage avec un préimprégné FR4, soit en utilisant de l'adhésif de transfert. L'avantage de l'adhésif est que les coefficients de dilatation de l'aluminium et du CI peuvent être mieux compensés durant le brasage, favorisant ainsi sa dynamique.
Würth Elektronik fixe le dissipateur à froid et sous vide sur le CI au moyen d'un adhésif de transfert selon une technologie appelée *TWINflex*.
Idéalement, le verso de la carte est réalisé avec une couche de cuivre pleine surface. La chaleur s'y répartit et est ensuite transférée au dissipateur via l'adhésif de transfert. La surface plus large ainsi disponible diminue la résistance thermique de l'application. L'objectif est d'obtenir une épaisseur minimale de CI offrant une surface de dissipation maximale.

spécialistes de Würth Elektronik de nous préparer une « combinaison » de LED adaptée à une « mini-serre » d'intérieur. Le résultat est une carte dont les LED couvrent une zone circulaire d'environ 55 cm de rayon. La zone extérieure est moins éclairée que le centre.
Deuxième obstacle à surmonter : des LED de forte intensité, ça chauffe ! Là encore, Würth Elektronik sait fabriquer des circuits imprimés qui réduisent au maximum l'échauffement des LED, ce qui de plus ralentit leur vieillissement. La méthode consiste à coller un CI mince sur un dissipateur en aluminium (cf. encadré **Gestion de la chaleu**r). Abstraction faite du prix, plus élevé que celui d'un CI ordinaire, la solution est parfaite.
Pour que ce projet reste abordable, nous avons conçu un CI mince (0,5 mm !) doté de larges plans de cuivre sur chaque face. Les LED de la série WL-SMDC possèdent des pastilles à faible résistance thermique pouvant être soudées sur le plan de cuivre d'une des faces. Ce plan est relié au plan opposé par des *vias* thermoconducteurs, ou « thermovias ». Par défaut, les cavaliers de la carte pilote sont configurés pour absorber 300 mA sur les quatre canaux. Cette valeur permet de se passer de dissipateur, mais pourrait s'avérer un peu faible pour certaines plantes.

Assemblage

Si vous utilisez les cartes préassemblées de l'e-choppe, vous aurez juste à placer les cavaliers, relier les cartes entre elles et câbler une alimentation CC de 24 V et 2,5 A. Il existe pour cela de nombreuses alimentations prêtes à l'emploi et bon marché. Nous avons utilisé une alimentation secteur pour le prototypage, puis un modèle BI60-240250-E2 d'Egston pour les tests de préconformité. L'élégante petite alimentation à découpage LRS-50-24 de 24 V et 2,2 A de Mean Well (**fig.** 3) serait également un très bon choix.

Si vous décidez de souder les deux CI, nous vous conseillons vivement d'avoir au moins de la pâte à souder et une station à air chaud. Un fer à panne fine pourrait convenir, mais veillez à placer les composants par ordre de taille, les gros risquant de gêner l'accès aux pastilles des plus petits. Placez d'abord les CMS, et n'attaquez les traversants qu'après avoir doublement vérifié la soudure des CMS !

Programmation et mise à jour du micrologiciel

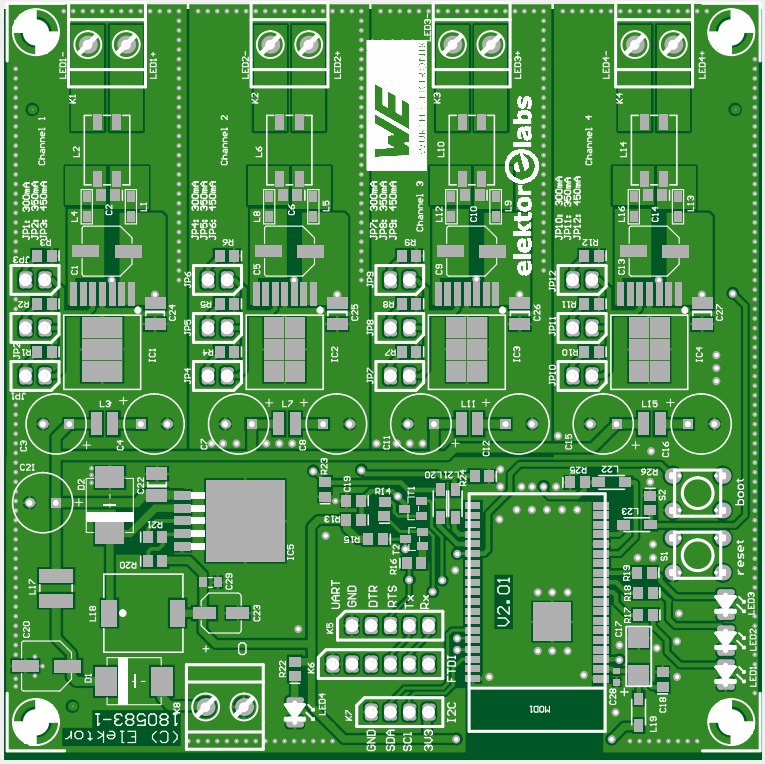
La carte pilote préassemblée de l'echoppe possède un module ESP32 programmé et prêt à l'emploi. Dans ce cas, l'EDI Arduino et son paquet pour ESP32 ne vous seront utiles que si vous souhaitez modifier le micrologiciel. L'installation du paquet *arduino-esp32* est décrite en [2].

Il vous faudra aussi télécharger le micrologiciel depuis la page du labo d'Elektor [3]. Le commentaire placé au début du croquis principal *Firmware.INO* liste les bibliothèques nécessaires à la compilation du code source.

Si vous disposez d'un UART USB avec signaux de mise en liaison RTS et DTR, reliez-le à K5. L'EDI Arduino sera alors en mesure de charger automatiquement le micrologiciel et le croquis. Si votre UART n'a pas ces signaux de mise en liaison (p. ex. un câble FTDI relié à K6), vous devez d'abord mettre l'ESP32 en mode chargeur d'amorce afin que l'EDI puisse transférer les données : maintenez S2 enfoncé tout en appuyant et relâchant S1, puis relâchez S2.

La liaison sans fil (OTA, *over-the-air*) est activée dans le micrologiciel par défaut, ce qui permet de flasher le système via Wi-Fi sans passer par la connexion USB ou les poussoirs. Connectez-vous à la carte pilote en mode Point d'accès, ou laissez le dispositif se connecter à votre réseau local. Lancez l'EDI Arduino, et depuis le menu *Outils* sélectionnez *ESP32 WROVER Module*. Dans le sous-menu *Port*, choisissez le port *HC-LED-XX-XX-XX*, soit l'ESP32. Cliquez sur le bouton *Téléverser* pour transférer le code dans la puce. Si tout s'est bien déroulé, la puce redémarre et le nouveau micrologiciel est en place. Attention, notez que si vous modifiez le micrologiciel et rompez la liai-

LISTE DES COMPOSANTS, CARTE PILOTE



60% of true size

Résistances

R1, R4, R7, R10 = 3,48 kΩ, couche épaisse, 1 %, 0,125 W, 150 V
R2, R5, R8, R11 = 3 kΩ, couche épaisse, 1 %, 0,125 W, 150 V
R3, R6, R9, R12 = 2,32 kΩ, couche épaisse, 1 %, 0,1 W, 150 V
R13, R14, R15, R16 = 10 kΩ, couche épaisse, 5 %, 0,1 W, 150 V
R17, R18, R19, R22, R23, R24, R25, R26 = 1 kΩ, couche épaisse, 5 %, 0,1 W, 150 V
R20 = 2,7 kΩ, couche épaisse, 5 %, 0,1 W, 150 V
R21 = 1,6 kΩ, couche épaisse, 1 %, 0,125 W, 150 V

Inductances

L1, L4, L5, L8, L9, L12, L13, L16 = ferrite anti-interférences, 1500 Ω@100 MHz, 0805, WE-CBF
L2, L6, L10, L14 = filtre en mode commun CMS, 10 µH, 1,6 A, WE-SL2
L3, L7, L11, L15 = inductance de puissance CMS, 10 µH, 800 mA, WE-PD2, boîtier 3521
L17 = inductance de puissance CMS, 2,2 µH, 2,5 A, WE-PD2, boîtier 4532
L18 = inductance de puissance CMS, 470 µH, 600 mA, WE-PD 1050
L19, L20, L21, L22, L23 = perle de ferrite, 31 Ω, 3 A, boîtier 1206

Condensateurs

C1, C5, C9, C13, C20 = 4,7 µF, 100 V, 7,7 × 6,3 mm
C2, C6, C10, C14, C18, C19, C29 = 100 nF, 100 V, X7R, 0805
C3, C4, C7, C8, C11, C12, C15, C16, C21 = 27 µF, 100 V, 20 %, 8 mm, radial
C17 = 47 µF, 10 V, 2312
C22 = 4,7 µF, 50 V, X7R, 1210
C23 = 100 µF, 10 V, 5,5 x 5,5 mm
C24, C25, C26, C27 = 2,2 µF, 100 V, X7R, 1210
C28 = 1 nF, 16 V, X7R, 0603

Semi-conducteurs

D1, D2 = MBRS540, 40 V, Vf = 550 mV @ If = 5 A
LED1 = LED, verte, 3 mm
LED2 = LED, rouge, 3 mm
LED3 = LED, jaune, 3 mm
LED4 = LED, bleue, 3 mm
T1, T2 = BC847C, 45 V, 100 mA, 250 mW, hfe = 400
IC1, IC2, IC3, IC4 = module MagI3C LED Step Down High Current 172946001, Würth
IC5 = LM2576HVS-ADJ, régulateur abaisseur, 4-60 V, 3 A
MOD1 = ESP-32-WROVER-B

Divers

S1, S2 = poussoir tactile, 12 V, 5 mA, 6x6 mm
K1, K2, K3, K4, K8 = bornier à vis, 2 voies, au pas de 5 mm, 630 V
K5 = barrette à 5 points, au pas de 2,54 mm, verticale
K6 = barrette à 6 points, au pas de 2,54 mm, verticale
K7 = barrette à 4 points, au pas de 2,54 mm, verticale
JP1, JP2, JP3, JP4, JP5, JP6, JP7, JP8, JP9 = cavalier, 1×2, au pas de 2,54 mm, vertical
4 cavaliers, 2 voies, au pas de 2,54 mm
Circuit imprimé, réf. 180583-1

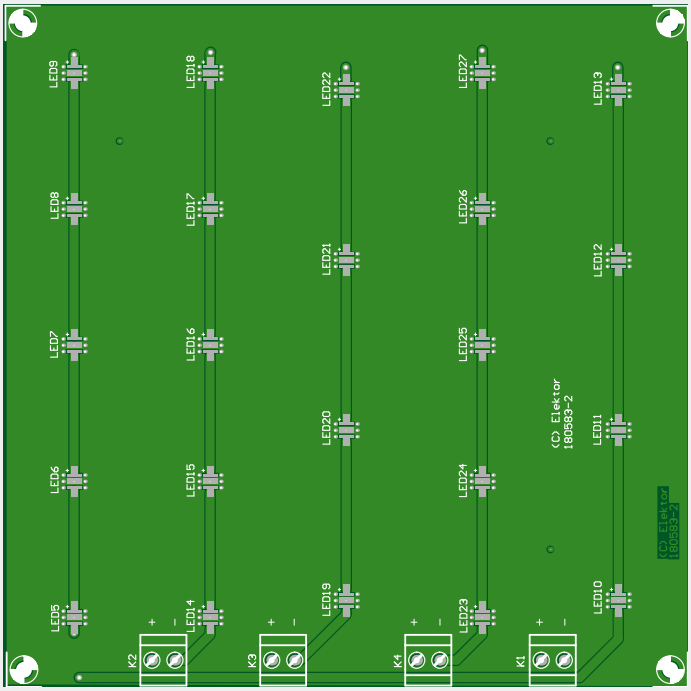
LISTE DES COMPOSANTS, CARTE DES LED

Semi-conducteurs

LED5, LED6, LED7, LED8, LED9, LED10, LED11, LED12, LED13 = LED à haute luminosité, série WL-SMDC, rouge, 660 nm, 125°, 700 mA
LED14, LED15, LED16, LED17, LED18 = LED à haute luminosité, série WL-SMDC, bleue, 450 nm, 125°, 700 mA
LED19, LED20, LED21, LED22 = LED à haute luminosité, série WL-SMDC, rouge, 730 nm, 125°, 700 mA
LED23, LED24, LED25, LED26, LED27 = LED à haute luminosité, série WL-SWTC, blanche, 120°, 121 lm, 6000 K, 700 mA

Divers

K1, K2, K3, K4 = bornier à vis, 2 voies, au pas de 5 mm, 630 V
Circuit imprimé, réf. 180583-2



60% of true size

Attention les yeux !

Ne regardez jamais directement des LED de forte intensité sous peine de lésions oculaires irréversibles. L'intensité de la lumière émise par les LED est en effet très grande, même à puissance effective réduite.
Le fait que les LED soient pilotées par MLI n'est pas non plus sans conséquence. Contrairement au cas des lampes à incandescence p. ex., l'intensité de la lumière émise par une LED peut varier très rapidement. Une LED apparemment éteinte peut ainsi émettre des impulsions lumineuses ultra-brèves et très intenses, mais imperceptibles. La dangerosité de ces impulsions n'est pas clairement établie, mais elles ne peuvent certainement pas être bénéfiques.
Le plus sûr est sans doute de ne jamais regarder directement des LED de forte intensité, même si elles semblent éteintes.

son OTA Arduino, vous devrez regraver entièrement le système par câble.
Le téléversement du croquis ESP peut lui aussi être effectué par liaison sans fil en cliquant sur *ESP32 Sketch Data Upload* depuis le menu *Outils*.

Partie logicielle

Nous avons allègrement conduit en parallèle les développements matériel et logiciel, ce qui a occasionné un bel échange d'informations tout au long du projet, en particulier lors du choix des broches d'E/S à utiliser. Certaines des broches ont ainsi été échangées en cours de route pour faciliter le routage du CI ; la modification d'une seule ligne de code nous aura épargné un travail compliqué sur le tracé du CI.

Pour en venir à la partie logicielle, vous avez sans doute remarqué que pas mal de nos derniers projets reposaient sur un ESP32. Alors bien sûr, comme tout bon développeur ne souhaitant pas réinventer la roue, nous avons cherché du côté de ces projets des éléments à réutiliser. Nous avons ainsi trouvé un serveur web et des routines de configuration par réseau sans fil dans la *station météo à ESP32* et dans *l'horloge à LED géante avec Wi-Fi et mesures météo*.

Notre système comprend aussi d'autres modules, dont « ledc », une unité MLI apportant d'utiles fonctions supplémentaires de pilotage des LED. Pour le configurer, nous lui indiquons la fréquence et la résolution souhaitées de la MLI ; le pilote de LED de Würth Elektronik repose en effet sur le rapport de gradation. Il y a aussi une durée d'activation minimale du signal (ou largeur d'impulsion) à observer. Comme pour toute unité MLI, nous cherchons à déterminer la résolution pour une fréquence MLI donnée. La fréquence d'entrée étant de 40 MHz, la résolution maximale en bits est donnée par :

$$\log_2 (40 \times 10^6 / 250) = 17{,}2$$

Nous disposons donc de 17 bits pour la résolution MLI, mais n'en utilisons que 8 dans le code en raison de la largeur d'impulsion minimale à respecter.

```
ledcSetup(CHANNEL, PWM_FREQ ,
   PWM_RES );
```

Cette instruction configure un des huit canaux de LED disponibles. À la différence d'autres systèmes, le canal *ledc* n'est pas lié à une broche particulière de l'ESP32. Nous pouvons donc utiliser n'importe quelle broche de sortie avec :

```
ledcAttachPin(GPIO_PIN, CHANNEL);
```

Nous sommes dès lors prêts à commander les broches MLI. Une valeur se définit avec :

```
ledcWrite(CHANNEL,  VALUE);
```

La valeur actuelle d'un canal *ledc* s'obtient avec :

```
ledcRead(CHANNEL)
```

Ce sont les commandes de base de l'unité *ledc*. Le module logiciel comprend les quatre canaux physiques des LED ; un cinquième, purement logiciel, contrôle l'intensité totale. Ce canal est appelé *intensity* dans l'interface web. Toute modification de sa valeur n'est pas immédiate. Le mécanisme est le suivant : une fonction appelée toutes les 100 ms par un temporisateur vérifie si la valeur réglée est égale à celle en cours. Si ce n'est pas le cas, cette valeur est incrémentée ou décrémentée selon un pas maximal donné. Le résultat est un certain affaiblissement, perceptible lors d'un changement d'intensité. Ce mécanisme est appliqué simultanément aux quatre canaux. L'intensité augmente ainsi lentement à la mise sous tension.
La carte exploite également, dans une version légèrement modifiée, le module temporel de l'horloge à LED géante. Certaines plantes ayant besoin d'un cycle jour/nuit, nous avons ajouté une fonction qui éteint les LED durant une période donnée de la journée.

Interface web

L'interface utilisateur est un recyclage de projets Elektor précédents (**fig.** 4). La page principale comprend cinq champs pour le réglage de la MLI : les champs *Channel* 1 à 4 représentent les canaux physiques, le champ *Intensity* commande la luminosité totale. Vous pouvez utiliser les boutons + et – pour changer la valeur par pas de 1 %, ou entrer directement une valeur comprise entre 0 et 100 %. Un changement de valeur est communiqué par *websockets*. Si plus d'une personne modifie les réglages, toutes les autres recevront automatiquement les nouvelles valeurs dans leur navigateur. Pour cela le code utilise un serveur WebSocket séparé qui envoie et traite les messages encapsulés en JSON.

Sous ces champs, une case à cocher permet d'activer/désactiver une plage horaire d'allumage des LED.
Le menu *Time settings* permet de régler manuellement l'horloge interne ou de sélectionner un serveur NTP (réglage automatique de la date et de l'heure).

Le menu *Wi-Fi settings* enregistrera vos identifiants de réseau. Cela permet de commander l'éclairage avec un autre appareil relié au réseau.

La partie MQTT vous semblera peut-être moins familière. Nous l'avons empruntée à la station météo à ESP32 comme moyen de régler les LED et l'éclairage automatique depuis un courtier MQTT. Pour cela nous avons utilisé les bibliothèques *PubSubClient* et *ArduinoJson*. Pour envoyer au système de nouveaux réglages, vous devez créer une chaîne JSON formatée comme suit :

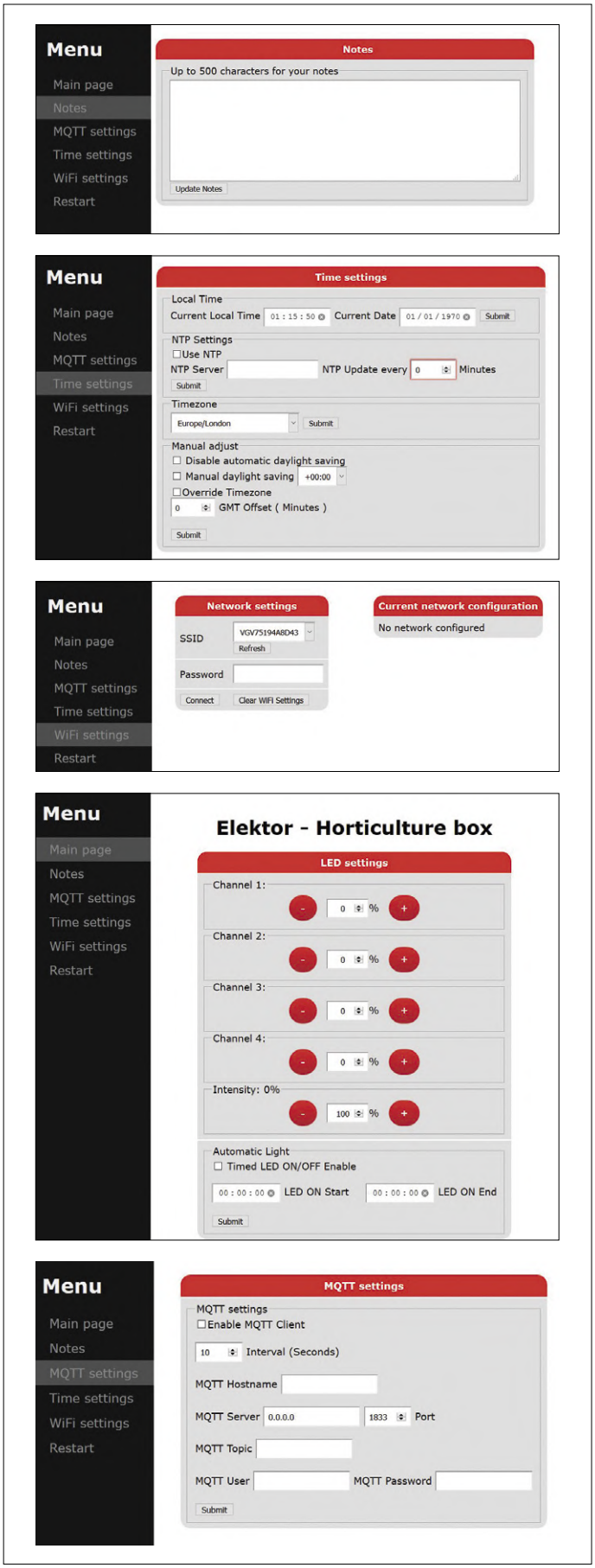


Figure 4. Copies d'écran de l'interface utilisateur de l'Horticulture Box (le nom anglais du projet) : page principale, notes, paramétrages date/heure, Wi-Fi et MQTT. Il s'agit d'une version modifiée de l'interface écrite pour deux autres projets Elektor.

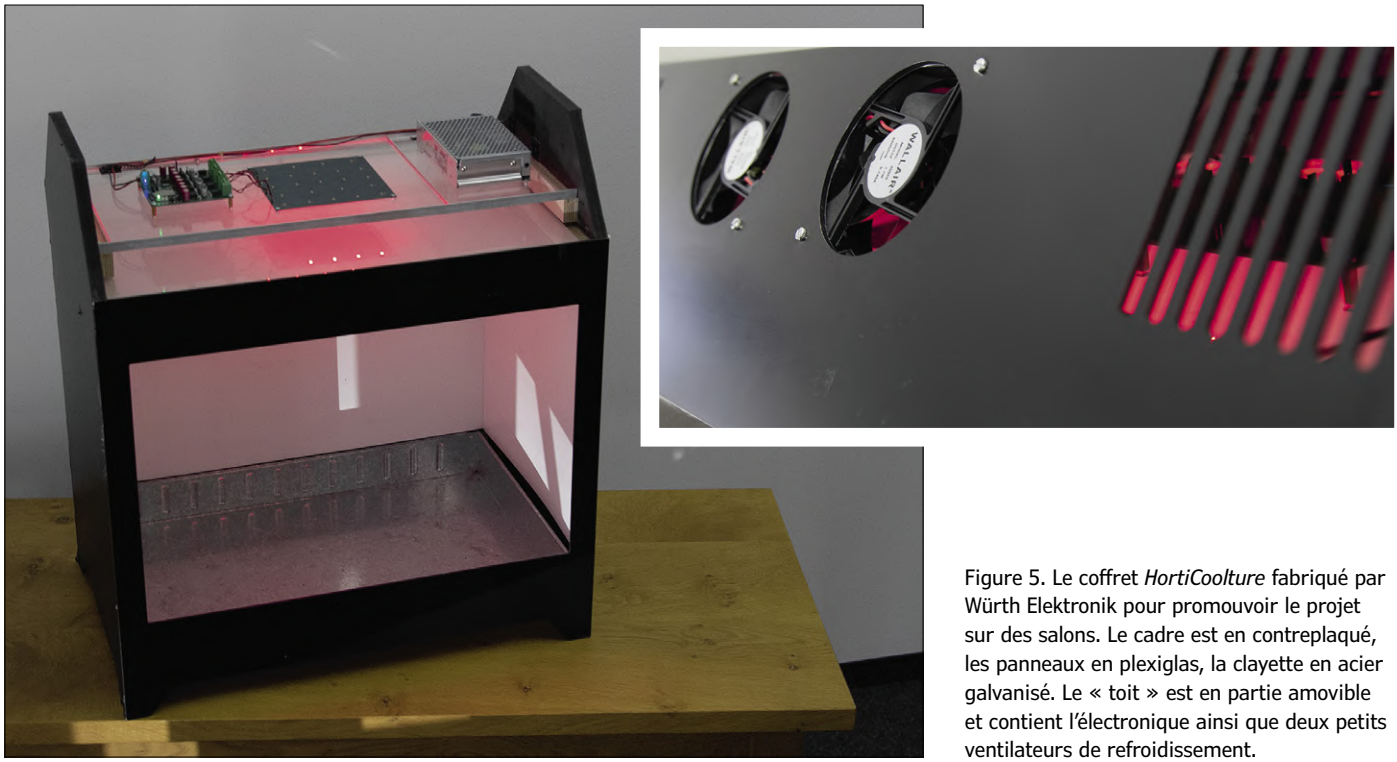


Figure 5. Le coffret *HortiCoolture* fabriqué par Würth Elektronik pour promouvoir le projet sur des salons. Le cadre est en contreplaqué, les panneaux en plexiglas, la clayette en acier galvanisé. Le « toit » est en partie amovible et contient l'électronique ainsi que deux petits ventilateurs de refroidissement.

Figure 6. La dernière déclinaison de l'Horticulture Box a été imprimée en 3D. (Produit et photo : Würth Elektronik)

```
{
    "light": {
        "ch0": 6400,
        "ch1": 0,
        "ch2": 0,
        "ch3": 0,
        "intense": 32000
    },
    "timer": {
        "enable": false,
        "start": {
            "hour": 0,
            "minute": 0,
            "second": 0
        },
        "end": {
            "hour": 0,
            "minute": 0,
            "second": 0
        }
    }
}
```

Les valeurs non signées de 16 bits sont autorisées pour les clés ch0 à intensity. Les nombres plus grands sont automatiquement limités à 65.535, la limite inférieure étant 0. Cela permet à Node-RED (par exemple) d'éclairer les plantes avec un mélange de couleurs différentes selon l'heure de la journée. La présence des

Liens

[1] Notes d'application des LED Horticulture : https://katalog.we-online.de/en/led/WL-SMDC_HORTICULTURE

[2] Paquet ESP32 pour EDI Arduino : www.elektormagazine.com/labs/esp32-getting-started

[3] Téléchargement du micrologiciel : www.elektormagazine.fr/180583-03

[4] « horloge à LED géante avec Wi-Fi et mesures météo », Elektor 05-06/2019 : www.elektormagazine.fr/magazine/180254-04

[5] « station météo à ESP32 », Elektor 01-02/2019 : www.elektormagazine.fr/180468-04

clés dont on ne modifie pas la valeur n'est pas obligatoire. On utilisera ainsi, p*our* modifier uniquement la clé ch0 :

```
{
   "light": {
        "ch0": 6400
   }
}
```

Ici le premier canal est réglé sur 6.400. Cette écriture est possible, car le code de traitement des messages MQTT vérifie si des objets individuels sont présents. Avec ArduinoJson, le code devient :

```
JsonObject light = doc["light"];
 JsonVariant light_ch0_var = light["ch0"];
    If (false == light_ch0_var.isNull() ){
    ……
```

La condition IF vérifie si une valeur est présente ou non dans le canal 0. Si oui, nous la traitons, sinon, nous l'ignorons. Si vous deviez traiter JSON vous-même, vérifiez si les valeurs dont vous avez besoin sont bien contenues dans les données JSON, sous peine d'utiliser involontairement des valeurs par défaut.

Entre plein d'oseille et plus un radis

Nous avons installé notre éclairage dans un coffret pouvant servir de paludarium, mais libre à vous bien sûr de concevoir le contenant le plus approprié à vos plantes, l'essentiel étant q**ue la ca**rte des LED les surplombe. La figure 5 montre le bâti « HortiCoolture » façonné par Würth Elektronik à des fins promotionnelles. Son capot est équipé de deux petits ventilateurs aidant au refroidissement de la carte des LED. Le fabricant allemand a ensuite conçu et réali**sé un** autre modèle par impression 3D (fig. 6).

Évidemment, tout le monde ne pourra pas reproduire ou s'offrir un tel modèle, mais ce n'est pas la fin des haricots : un aquarium déniché dans un dépôt-vente, acheté en jardinerie ou chez Ikea, formera une bonne base pour monter une « mini-serre » à peu de frais.

Quant à l'éclairement optimal pour une espèce particulière, même si la littérature scientifique abonde à ce sujet, nous vous encourageons à trouver votre propre « recette lumineuse ».

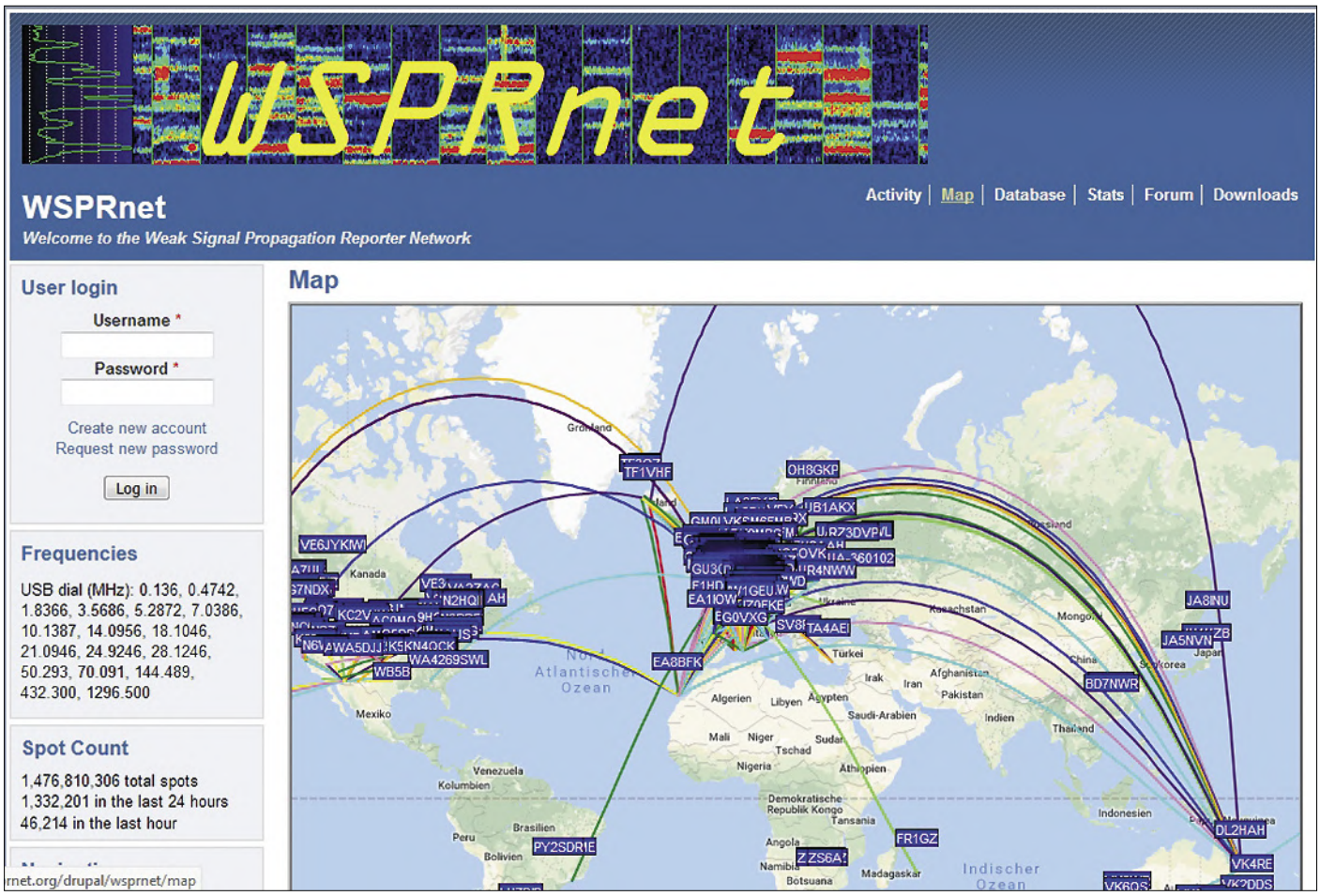
(180583-03 – version française : Hervé Moreau)

PROJET DE LECTEUR

transmission d'informations en WSPR

atteignez toute l'Europe avec le shield SDR d'Elektor

Burkhard Kainka



Le procédé de transmission WSPR a été conçu de manière à porter à grande distance avec une basse puissance et une bande étroite. Comment envoyer des messages WSPR avec le *shield* SDR d'Elektor ? Avec un logiciel gratuit que nous allons décrire.

Le protocole de communication WSPR (*Weak Signal Propagation Reporter*, prononcer *whisper*, chuchoter) permet une longue portée sur un spectre étroit malgré la modestie de la puissance émise. De nombreuses stations actuellement en service rapportent les résultats de réception (QSA) via la page de wsprnet.org [1]. Vous y voyez directement où le signal est parvenu. Avec seulement 10 mW, on peut porter jusqu'à 1 000 km et avec 100 mW, on couvre sans difficulté toute l'Europe.

Il faut pour cela le programme WSPR 2.0 de Joe Tailor (K1JT) [2], le concepteur de WSPR. Le signal émis est en réalité

une tonalité constante de 1,5 kHz d'habitude transmise par un émetteur-récepteur BLU (*SSB*). Si vous écoutez attentivement, vous constaterez de très petites différences à répétition. C'est que WSPR utilise un signal 4FSK à quatre fréquences distantes de 1,46 Hz. Chaque fréquence reste stable pendant à peu près une seconde. De la sorte, le logiciel de réception peut exercer un filtrage à bande extrêmement étroite qui résulte en une bonne marge au bruit.
Elektor a présenté la technique de **réception** WSPR avec son *shield* SDR dans le dernier magazine de 2018 [3]. Il est temps de s'occuper maintenant de la technique **d'émission**. Elle met en œuvre le logiciel Arduino avec le codeur WSPR, un petit étage final HF et l'adaptateur d'antenne. Les essais de transmission présentés ici ne sont légalement permis qu'à des radioamateurs qui disposent d'une licence. Pour les autres que cela intéresse, il est toujours possible de conduire de petites expériences sans antenne d'émission raccordée, en écoutant le signal que l'on a soi-même émis.

Listage 1. L'initialisation dans si5351vfo2_1.ino

```
{
  Serial.begin(9600);
  Serial.println("Si5351 Clockgen"); Serial.
println("");
  si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);
  si5351.set_correction(162100,
SI5351_PLL_INPUT_XO);
  Serial.println(10100);
  si5351.set_freq(4040000000ULL,  SI5351_CLK1);
  si5351.output_enable(SI5351_CLK1, 1);
  freq = 10100;
  lcd.begin(16, 2);
  lcd.print(freq);
}
```

Commande du *shield* SDR

Le *shield* SDR s'utilise le plus souvent comme récepteur des ondes courtes. Mais le générateur asservi en phase (PLL) Si5351 dispose de deux autres sorties utilisables pour des mesures ou le pilotage d'un émetteur. Il peut aussi bien produire un signal WSPR. Le livre en anglais « SDR Hands-on Book » [4] le décrit, ainsi que bien d'autres expériences.
Lors du travail de rédaction de ce livre et pour les premiers articles sur le *shield* SDR, j'ai utilisé l'ancienne bibliothèque Etherkit Si5351 de Jason Milldrum en version 1.1.2. Les nouveaux utilisateurs auront recours à la nouvelle version 2.1.4. Avec la nouvelle version de la bibliothèque et le code source original, vous disposez des correctifs, vu que certaines structures ont légèrement changé.
L'initialisation réclame maintenant un paramètre supplémentaire 0 pour signaler que la fréquence précise du quartz (qui diffère de 25 MHz) sera indiquée plus tard.

```
si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);
```

L'étalonnage proprement dit doit intervenir après l'initialisation dans laquelle `...INPUT_XO` indique qu'il s'agit de l'oscillateur à quartz :

```
si5351.set_correction(162100, SI5351_PLL_INPUT_XO);
```

L'expression de la fréquence n'a plus besoin de l'ancien paramètre `PLL_FIXED`, puisque la bibliothèque fixe automatiquement le réglage de la PLL :

```
si5351.set_freq(freq*100ULL, SI5351_CLK0);
```

Entretemps, les programmes d'exemple du livre ont été révisés, y compris les applications de la série d'articles sur le *shield* SDR. L'archive de logiciels pour le livre [4] reprend à présent les versions d'origine et celles actualisées.

Au début, on n'a besoin que du programme d'accord si53510vfo2-1 (**listage 1**). Il s'agit de régler la fréquence du VFO sur 7 038,6 kHz (**fig. 1**) pour que les signaux WSPR dans la bande des 40 m soient mélangés vers le bas sur 1,5 kHz. Pour vérifier l'exactitude des réglages, on peut enclencher la sortie A. Un signal à 7 440,1 kHz se couple alors dans l'entrée, on peut le voir (**fig. 2**) dans WSPR 2.0 juste au milieu du domaine de réception large de 200 Hz. Toutefois, l'étalonnage de la fréquence de réception n'est pas encore assez précis.
On voit que la porteuse est reçue et donc qu'il y a aussi un couplage capacitif faible. Il ne nous manque plus que la modulation, c'est-à-dire la commutation voulue entre les quatre fréquences adjacentes du signal 4FSK.

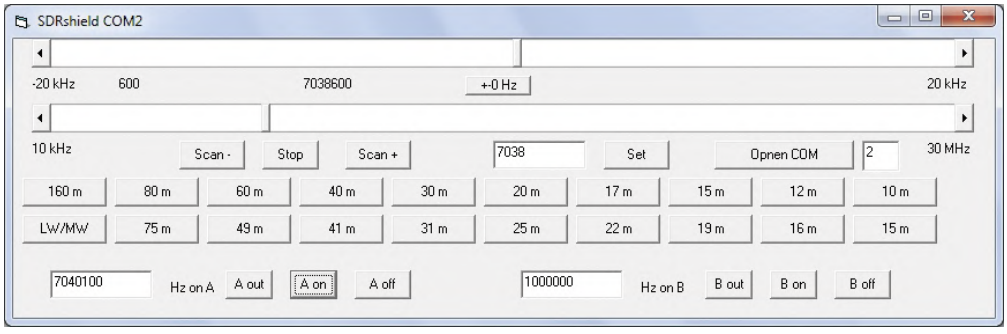


Figure 1. Réglage de la fréquence du VFO.

Listage 2. Transmission d'un signal WSPR (extrait de Si5351WSPR1.ino)

```
// Simple JT65/JT9/WSPR/FSQ beacon for Arduino, with the Etherkit
// Si5351A Breakout Board, by Jason Milldrum NT7S.

#include
#include
#include
#include
#include "Wire.h"
#define BUTTON                  12
#define LED_PIN                 13

…
char message[] = "EL2SDR JO31";
char call[] = "EL2SDR";  //das Rufzeichen
char loc[] = "JO31";    //der Lacotor wie z.B. JO31";
uint8_t dbm = 10;       //10 mW, 30 für 1W, 37 für 5 W
uint8_t tx_buffer[255];
…

void encode()
{
  uint8_t i;
  // Clear out the old transmit buffer
 memset(tx_buffer, 0, 255);
 jtencode.wspr_encode(call, loc, dbm, tx_buffer);
  // Reset the tone to the base frequency and turn on the output
  si5351.output_enable(SI5351_CLK0, 1);
  digitalWrite(LED_PIN, HIGH);
  for(i = 0; i < symbol_count; i++)
  {
      si5351.set_freq((freq * 100) + (tx_buffer[i] * tone_spacing), SI5351_CLK0);
      Serial.print (tx_buffer[i]);  Serial.print (",");
      proceed = false;
      while(!proceed);
  }
  // Turn off the output
  si5351.output_enable(SI5351_CLK0, 0);
  digitalWrite(LED_PIN, LOW);
}


void loop()
{
  if(digitalRead(BUTTON) == LOW)
  {
    delay(50);   // delay to debounce
    if (digitalRead(BUTTON) == LOW)
    {
      encode();
      delay(50); //delay to avoid extra triggers
    }
  }
}
```

La bibliothèque JTEncode

Pour produire les signaux numériques WSPR et autres avec le Si5351, la bibliothèque JTEncode [55] provient aussi de Jason Mildrum. Le premier programme pour le *shield* SDR a été développé à partir d'un de ses exemples. On fixe le récepteur sur la fréquence WSPR dans la bande des 20 m (VFO = 14 095,600 kHz), celle de l'émetteur, il démarre dès que la broche 12 de l'Arduino est mise à la masse. On peut alors entendre son signal à 1,5 kHz.

Pour le programme *Si5351WSPR1.ino* (**listage 2**), j'ai imaginé l'indicatif fantaisiste EL2SDR, lequel ne peut servir évidemment que pour des essais privés sans antenne. Il faut indiquer en outre un emplacement (JO31) et la puissance d'émission (10 dBm). La fonction de codage en fait un tampon d'émission avec les symboles WSPR 0, 1, 2 et 3 qui représentent les quatre fréquences adjacentes à distance de 1,46 Hz.

```
jtencode.wspr_encode(call, loc, dbm, tx_buffer);
```

Voici à quoi ressemble le contenu d'un tampon : 1,3,0,2,2,2,2,2,1,0,2,0,1,3,1,0… C'est lui qui régit la fréquence du signal FSK lors de l'émission des signaux (listage 2).

Un cycle de transmission dure 2 min et commence toujours par une minute paire. Aussi, pour que tout se déroule bien, il faut garder les horloges bien à l'heure. Avec le bouton-poussoir de la broche 2, vous lancez votre émetteur deux secondes après le début d'une minute paire.

Avec une antenne branchée sur le récepteur, on peut recevoir les vraies stations WSPR. En revanche, la sortie de l'émetteur reste libre, sans antenne. Le signal est couplé de manière capacitive entre la sortie A et l'entrée d'antenne, il est assez fort pour avoir une bonne réception avec WSPR 2.0. La **figure 3** montre le résultat du test. Le signal produit est reçu à 12 dB au-dessus du bruit. On voit aussi le décodage à −26 dB d'une station italienne d'une puissance de 37 dBm (5 W).

Étalonnage et démarrage temporisé

Impossible de faire fonctionner le WSPR sur toutes les bandes sans passer par l'écriture d'un micrologiciel étendu, *si5351vfo2_WSPR.ino* et d'un programme utilisateur sur PC, *SDRshield_WSPR*. La fréquence exacte est déterminante. C'est pourquoi j'ai

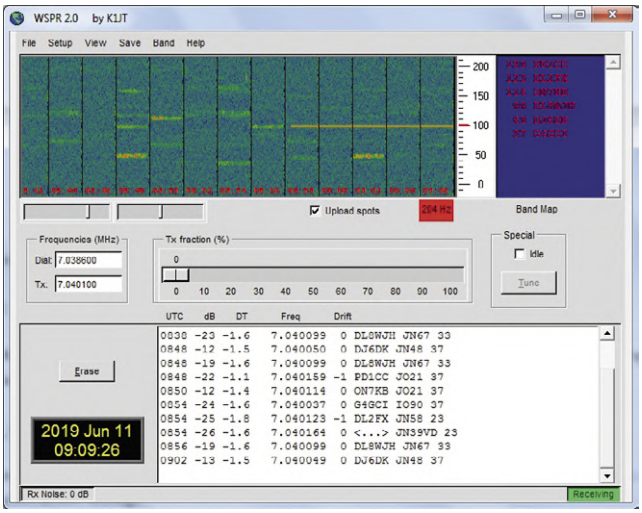


Figure 2. Les signaux WSPR et la porteuse auxiliaire A.

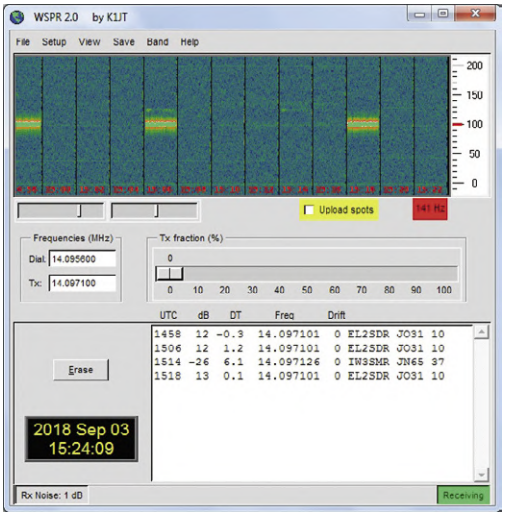


Figure 3. Réception de son propre signal.

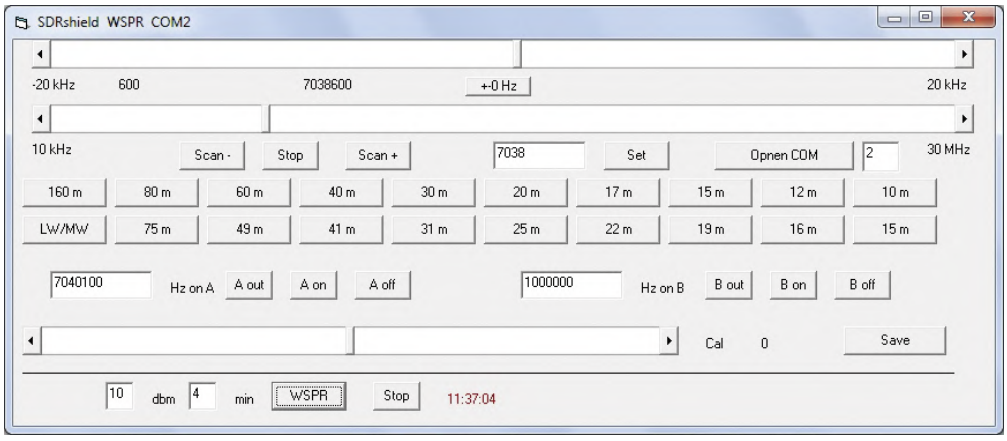


Figure 4. Fréquence WSPR sur 7 038,600 kHz en bande 40 m.

Listage 3. La commande de l'émetteur

```
    if (ch == 119) {       // w, Start wspr
      dbm = number;
      encode();
    }
   …
    if (ch == 102) {        //f
      si5351.set_freq(freq*400ULL, SI5351_CLK1);
      wsprfreq = freq;
    }

void encode()
{
…
      si5351.set_freq((wsprfreq * 100)+ 150000 + (tx_buffer[i] * tone_spacing),
                  SI5351_CLK0);
…
```

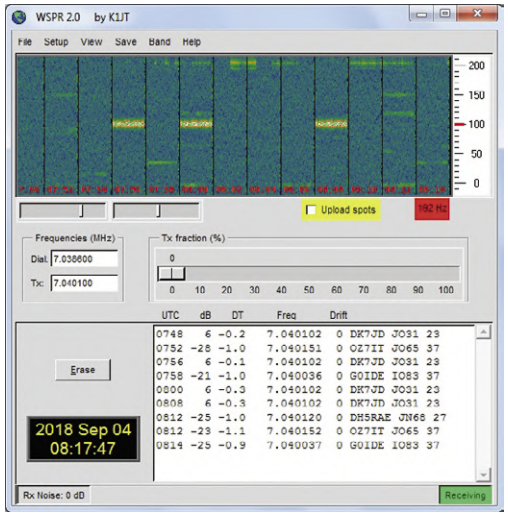


Figure 5. Réception de son propre signal.

intégré une fonction d'étalonnage. Comme étalon de fréquence, cherchez un émetteur de radiodiffusion en ondes courtes. À l'aide du curseur, syntonisez le récepteur exactement sur sa fréquence. Dans le spectre du logiciel SDR# [6], la porteuse doit être juste dans la grille de 5 kHz. On peut répéter le réglage sur d'autres stations de radio, la plupart d'entre elles travaillent avec grande précision. Le bouton *Save* enregistre la correction dans l'Arduino.

Le programme utilisateur VB a été modifié de sorte que les touches de raccourci pour les bandes de radioamateurs ne donnent plus le début de la bande, mais la fréquence WSPR correspondante. Réglez le VFO d'un clic sur 7 038,600 kHz dans la bande 40 m.

Un clic sur le bouton *WSPR* active la fonction d'émission (**fig. 4**). L'émetteur démarrera alors à la minute paire suivante puis encore après une période de 2 (par défaut), 4, 6, 8 ou 10 min, selon ce qui a été introduit. L'heure s'affiche en rouge pendant les phases actives.

La fréquence de réception courante est assignée à la fréquence d'émission dans le croquis Arduino. La commande *w* lance l'émission, donc plus besoin de la broche 12. Un seul paramètre l'accompagne, la puissance d'émission actuelle en dBm. Pendant la transmission, on ajoute à `wsprfreq` 1 500 Hz et la fréquence de modulation correspondante (**listage 3**).

Premier test en bande 40 m effectué sans antenne émettrice, la borne A en l'air. À la **figure 5,** quelques stations enregistrées et les rapports des signaux entre elles, avec mon propre indicatif DK7JD et l'antenne de réception. Une connexion sans fil, ici aussi, mais la portée de 2 cm est particulièrement limitée.

Amplificateur d'émission de 200 mW

À la **figure 6**, un petit étage de sortie pour émetteur avec un MOSFET BS170 qui, alimenté en 5 V, fournit jusqu'à 200 mW (23 dBm). Préférez-vous le sustenter en 3,3 V ? Il donnera encore 100 mW (20 dBm). Une résistance série de 100 Ω réduit encore la puissance de 10 dB, pour délivrer 10 mW (10 dBm) ou 20 mW (13 dBm) à l'antenne. On a ainsi un large éventail de puissances :

- 5 V : 200 mW
- 3,3 V : 100 mW
- 5 V via 100 Ω : 20 mW
- 3,3 V via 100 Ω : 10 mW

Le filtre passe-bas à la sortie de l'étage final est proportionné aux bandes 30 m et 20 m. Les bobines de filtre de 0,5 µH sont faites de dix spires sur un diamètre de 5 mm, sans noyau. La **figure 7** révèle la constitution compacte de l'étage de sortie.

Le filtre passe-bas a été conçu pour une fréquence typique de 10 MHz. Comme vérification, le *shield* SDR a été soumis à un programme de mesure de quadripôles. Résultat à la **figure 8,** une fréquence limite de 18 MHz. Donc, la bande de 20 m vers 14 MHz passe bien et celle de 17 m sur 18 MHz aussi, mais avec restrictions. La limite inférieure utilisable de l'ampli est à environ 2 MHz. Cependant, pour y brancher directement une antenne appropriée, on ne peut attendre une atténuation harmonique suffisante qu'à partir de 10 MHz. La limite basse autour

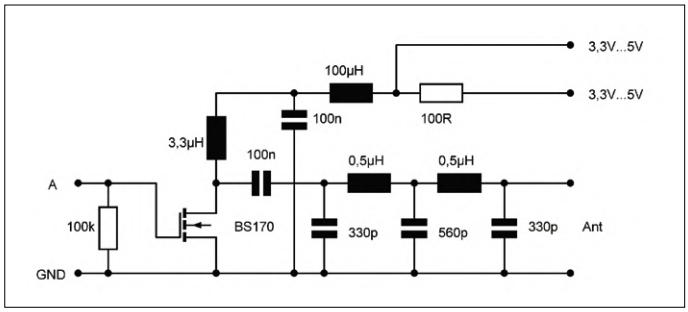


Figure 6. Étage de sortie 200 mW avec un BS170.

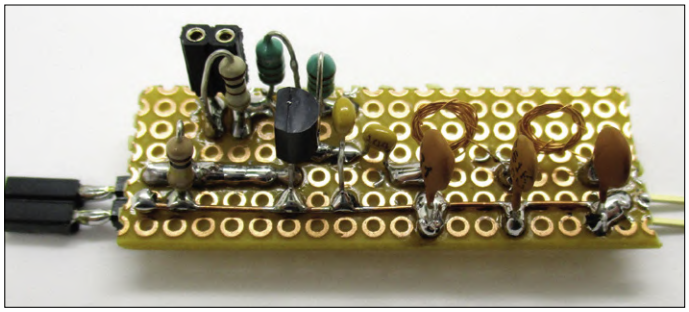


Figure 7. L'étage de sortie à FET avec filtre.

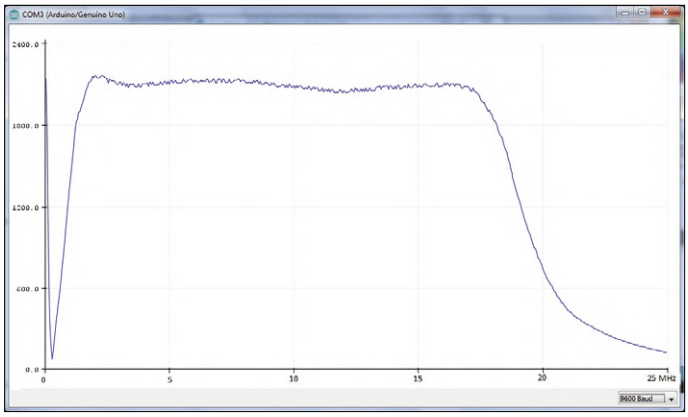


Figure 8. Réponse en fréquence de l'étage de sortie.

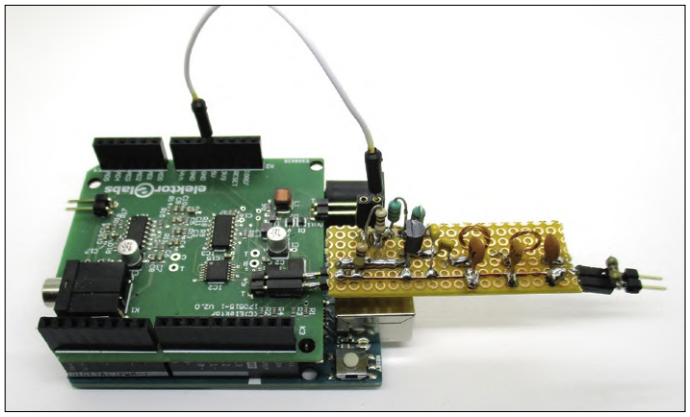


Figure 9. Alimentation en 5 V avec résistance de bouclage.

de 2 MHz est attribuable à la relativement faible inductance de 3,3 µH dans le circuit de drain de l'amplificateur.

L'examen à l'oscilloscope montre l'effet du passe-bas et la puissance de sortie accessible. Alimenter l'ampli sous 5 V et prévoir une résistance de bouclage de 51 Ω (**fig. 9**). Il y a sur la résistance de charge 8 V_{CC}, donc une amplitude de 4 V. On en déduit une puissance de : $P = U^2 / 2 R = (4 V)^2 / 100 \Omega = 0,16 W = 160 mW$. Monter l'alimentation à 7 V augmente encore la puissance. Les oscillogrammes des **figures 10 à 12** montrent l'amélioration du filtrage jusqu'à la sortie antenne.
Somme toute, on peut utiliser l'ampli sur toutes les bandes de 160 m à 20 m. Il faudra quand même veiller à une atténuation des harmoniques suffisante, mais c'est assez facile avec un adaptateur d'antenne sous la forme d'un filtre en π (pi).

Adaptateur d'antenne

Avec un adaptateur, on peut se servir d'antennes qui n'ont pas la bonne longueur et ne sont donc pas telles quelles en résonance. Seule une antenne dipôle optimale présente une réelle résistance au pied proche de 50 Ω. Dans tous les autres cas, vous avez une impédance plus élevée plus une composante capacitive ou inductive.
On peut adapter presque chaque antenne avec un filtre en π (**figures 13 et 14**). Composé de deux condensateurs variables et d'une bobine à prises intermédiaires, il fait varier l'inductance résultante selon les cavaliers mis. Suivant la configuration, la tension de sortie peut devenir supérieure ou inférieure à l'entrée. On adapte ainsi autant les antennes à basse qu'à haute impédance. Le filtre en π transforme l'impédance de l'antenne et compense les réactances restantes.
Le filtre en π est en même temps un passe-bas qui atténue les

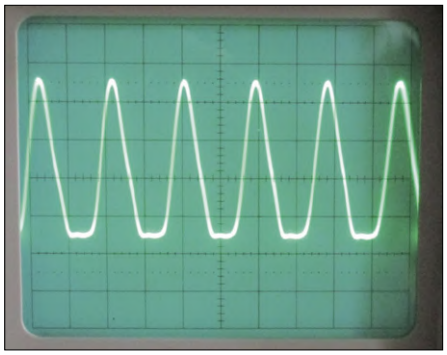


Figure 10. Le signal sur le drain du FET.

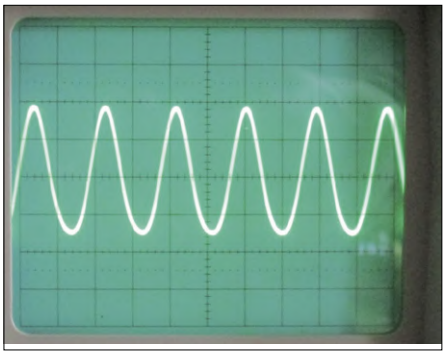


Figure 11. Au milieu du filtre.

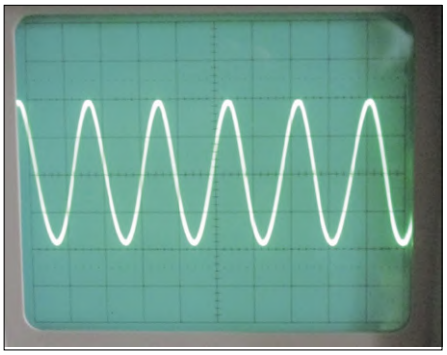


Figure 12. Sur la résistance de bouclage.

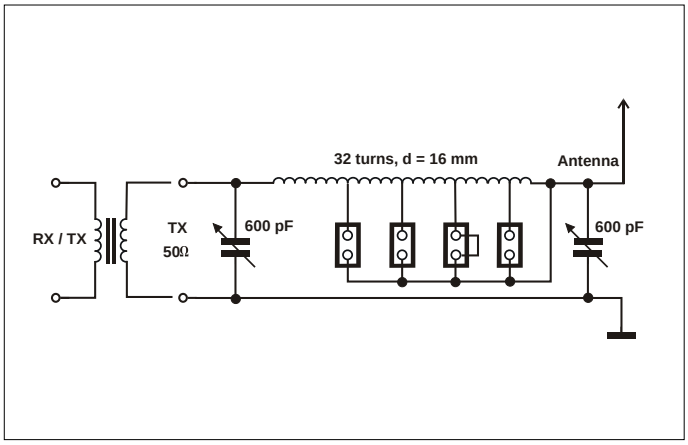


Figure 13. Filtre en π avec transformateur.

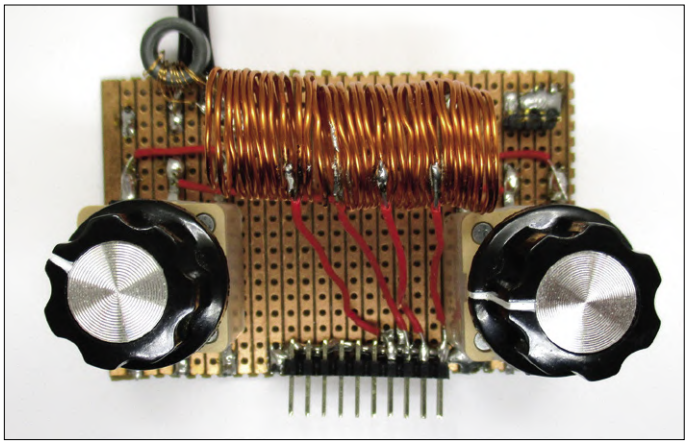


Figure 14. Construction avec bobine à air et condensateurs variables à feuilles.

Figure 15. Points de contact avec 200 mW.

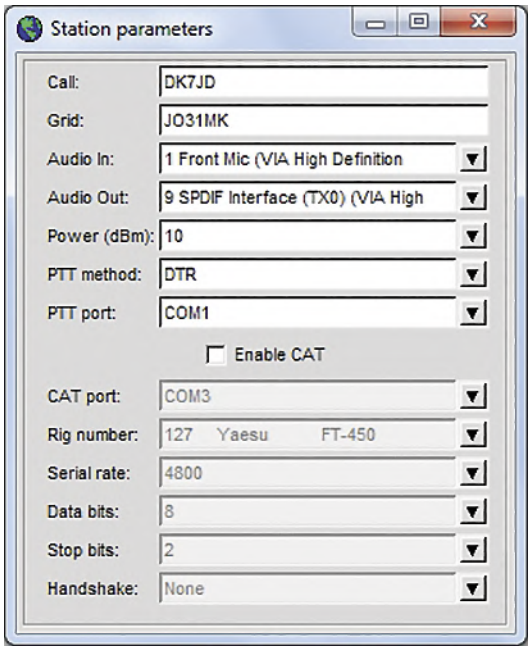


Figure 16. Paramétrage logiciel.

harmoniques du signal. Pour une efficacité optimale, il faut le doter du plus grand facteur de mérite (Q) possible. Cela ne réussit pourtant pas à toutes les fréquences si le filtre en π est situé directement sur l'étage final avec son impédance proche de 50 Ω. Aussi vaut-il mieux élever la tension de sortie pour l'adapter au filtre en π. L'émetteur WSPR est alors en mesure de travailler aussi dans les bandes 80 m, 60 m et 40 m. L'oscilloscope peut aider à faire l'accord pour trouver le signal d'antenne qui fournit la tension la plus grande. Du même coup, on peut mettre en évidence de grosses erreurs dans le filtrage, auquel cas, on voit une nette détérioration de la forme d'onde due à un excès d'harmoniques.

J'ai pu, avec ce petit émetteur WSPR, opérer avec succès dans toutes les bandes de 80 m jusqu'à 20 m. J'ai ainsi pu couvrir presque toute l'Europe et atteindre des portées jusqu'à 3 000 km (**fig. 15**).

Lors d'autres expériences, j'ai également établi des liaisons avec des stations aux États-Unis, au Canada et en Australie avec des puissances tout aussi faibles. Pour ce faire, il suffit que l'émetteur reste en opération assez longtemps sur la fréquence appropriée. Les fluctuations générales des conditions de propagation conduisent alors souvent à une connexion réussie, mais de courte durée. Un test avec seulement 10 mW sur la bande de 40 m a démontré des portées jusqu'à 600 km.

Contrôle du CAT

Le programme WSPR 2.0 est aménagé en vue du contrôle du CAT, *Computer Aided Transceiver* (ou *Tuning*), réglage (du trancepteur ou de l'accord) assisté par ordinateur, sur de nombreux appareils de radioamateurs qui disposent de l'interface correspondante. Ensemble, ils permettent de syntoniser l'émetteur-récepteur sur toutes les fréquences et de le faire passer en émission. Le logiciel produit la modulation dans le domaine de 1 400 à 1 600 Hz pour l'entrée microphone de l'émetteur BLU (*SSB*). Tous les réglages nécessaires et les paramètres sont enregistrés pour cela dans le menu *Setup/Station Parameters* (**fig. 16**).

Le *shield* SDR n'étant pas un émetteur BLU, la totalité de la commande automatique n'est pas assurée. Mais on peut aussi enclencher le démarrage automatique par la méthode PTT. Le réglage DTR avec le port sériel COM1 augmente le débit de sortie sur l'interface série COM1 à chaque phase de transmis-

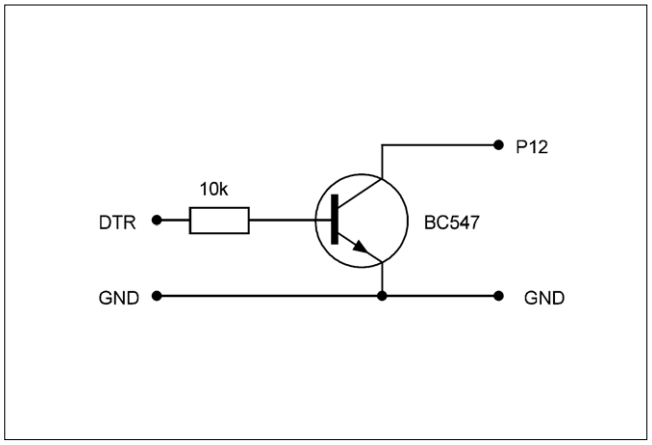


Figure 17. Interface de démarrage.

sion pour allumer un émetteur. On peut utiliser ce signal pour démarrer l'émetteur, il suffit d'un étage inverseur de commutation (**fig. 17**) pour commander l'entrée P12 de l'Arduino.
Avec cette liaison, le logiciel WSPR assume le contrôle de la phase d'émission. Il n'y a plus alors de réception de son propre signal. De plus, les intervalles de temps entre les phases de transmission peuvent être réglés, mais également dispersés de manière aléatoire. Cela évite que les stations éloignées doivent toujours émettre simultanément sur une plus longue période et donc ne puissent pas s'écouter les unes les autres. ⏮

(180709-03 – version française : Robert Grignard)

Liens

[1] Weak Signal Propagation Reporter Network : http://wsprnet.org/

[2] Logiciel WSPR 2.0 : https://physics.princeton.edu/pulsar/K1JT/wspr.html

[3] « shield SDR 2.0 d'Elektor (2) », Elektor 09-10/2018 : www.elektormagazine.fr/180284-04

[4] Livre SDR avec archive du logiciel : www.elektor.fr/sdr-hands-on-book

[5] Bibliothèque JTEncode : https://github.com/etherkit/JTEncode

[6] Logiciel SDR# : https://airspy.com

PROJET DE LECTEUR

catamaran aéroglisseur

avec courtier MQTT intégré

Walter Trojan

L'approche ludique est idéale pour se familiariser de façon durable avec les matériels et logiciels de l'IdO (Internet des Objets). L'auteur a eu l'idée d'un jouet dont la réalisation mécanique est à la portée des plus jeunes, par ex. de vos petits-enfants : un catamaran sur coussin d'air ! Le Wi-Fi et le protocole MQTT conviennent parfaitement pour le télécommander. Dans l'article, nous verrons également qu'un petit contrôleur comme l'ESP8266 peut prendre en charge la tâche d'un courtier MQTT.

« Papy, tu as déjà construit tellement de choses, tu pourrais quand même penser un peu à nous ! » Cette demande de mes petits-enfants me sortit de mes réflexions sur la commande vocale, l'intelligence artificielle et me fit redescendre sur terre. Mais ce n'était pas simple de la concrétiser. Pourtant, je leur avais déjà bricolé des robots, des feux tricolores et autres jouets. Une idée s'imposa lorsque je jetai un coup d'œil dans la caisse de bricolage contenant quelques ventilateurs et une télécommande d'hélicoptère orpheline après le crash de celui-ci. J'entrevis aussi un moyen pour que mes petits-enfants puissent faire

le montage mécanique : un catamaran sur coussin d'air ! Et cela me permettrait aussi d'améliorer mes connaissances et d'enrichir mon expérience. Comme je venais justement de lire un article à propos d'un courtier MQTT s'exécutant sur ESP8266, il serait facile de passer rapidement aux essais. Oui, vous avez bien lu : un courtier MQTT s'exécutant sur ESP8266 à la mémoire principale très limitée. Est-ce possible ? Vous verrez.

Caractéristiques

- Communication entre le catamaran et la télécommande au moyen du protocole MQTT pour l'échange des commandes et des informations d'état
- Réseau Wi-Fi privé pour être indépendant du lieu d'utilisation et d'un courtier MQTT existant
- Commande aisée à l'aide d'une console (transformée)
- Surveillance de la batterie du catamaran et de la console
- Utilisation de l'EDI Arduino comme environnement de développement simple du logiciel
- Construction mécanique simple et peu coûteuse

Une mécanique très élémentaire

La structure ne m'a pas demandé beaucoup d'investissement : les flotteurs sont constitués de deux bouteilles de polyéthylène (PET) de taille moyenne, thermocollées sur une planchette de contre-plaqué. Pour la propulsion, deux ventilateurs de 5 V eux-mêmes simplement thermocollés. L'alimentation est confiée à deux accumulateurs Li-Po de type 18650 (3400 mAh chacun). Ils suffisent à amuser les enfants pendant une durée plus que satisfaisante. Une petite carte accueillant l'ESP8266 est montée à côté des boîtiers des accus. La photo du chapeau de l'article montre l'ensemble et prouve clairement que l'on peut construire un jouet sophistiqué avec des moyens dérisoires. L'embarcation tient bien l'eau et les deux ventilateurs la font avancer assez vite. Mais il n'y a pas de marche arrière.

Commande du catamaran avec l'ESP8266 (un surdoué polyvalent)

Dans ce projet, l'ESP8266 est le chef de bord, son grand frère l'ESP32 ne fait que l'assister dans la console. Les deux puces sont fabriquées par le fondeur chinois *Espressif*, elles possèdent un ou deux cœurs à 32 bits et disposent d'un grand nombre de périphériques. Comme ces deux microcontrôleurs (µC) ont déjà été maintes fois mis à contribution dans les colonnes d'*Elektor*, je me contenterai de vous renvoyer vers la littérature disponible [1] et les nombreuses informations présentes sur l'internet. L'ESP8266 modèle 201 se trouve sur une petite carte d'évaluation comportant de nombreuses connexions d'accès aux broches du µC. Une platine avec régulateur de tension et commande des moteurs accueille la carte ESP.

La **figure 1** montre l'électronique installée sur le catamaran. Au-delà de la

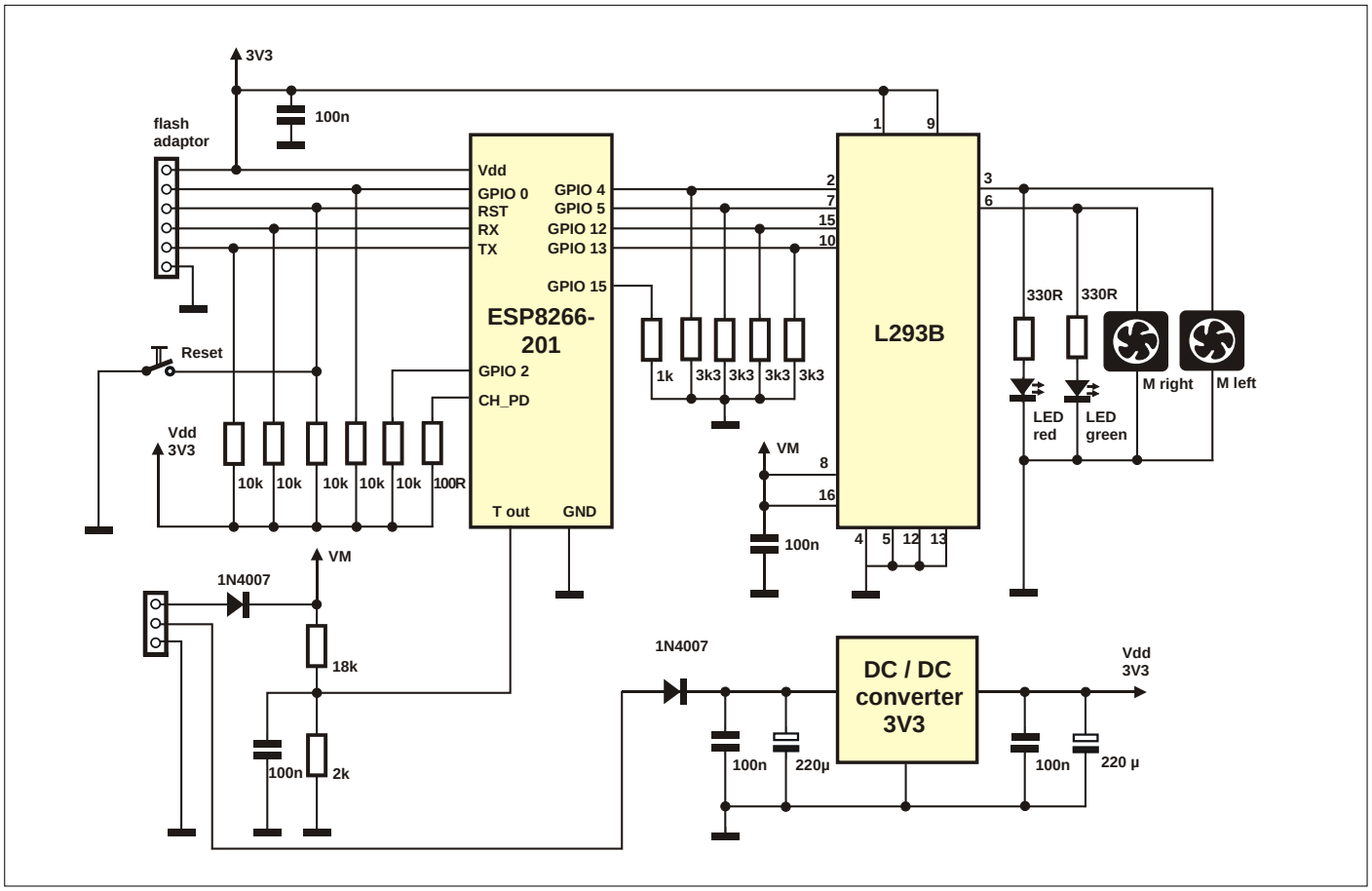


Figure 1. L'électronique installée sur le catamaran aéroglisseur.

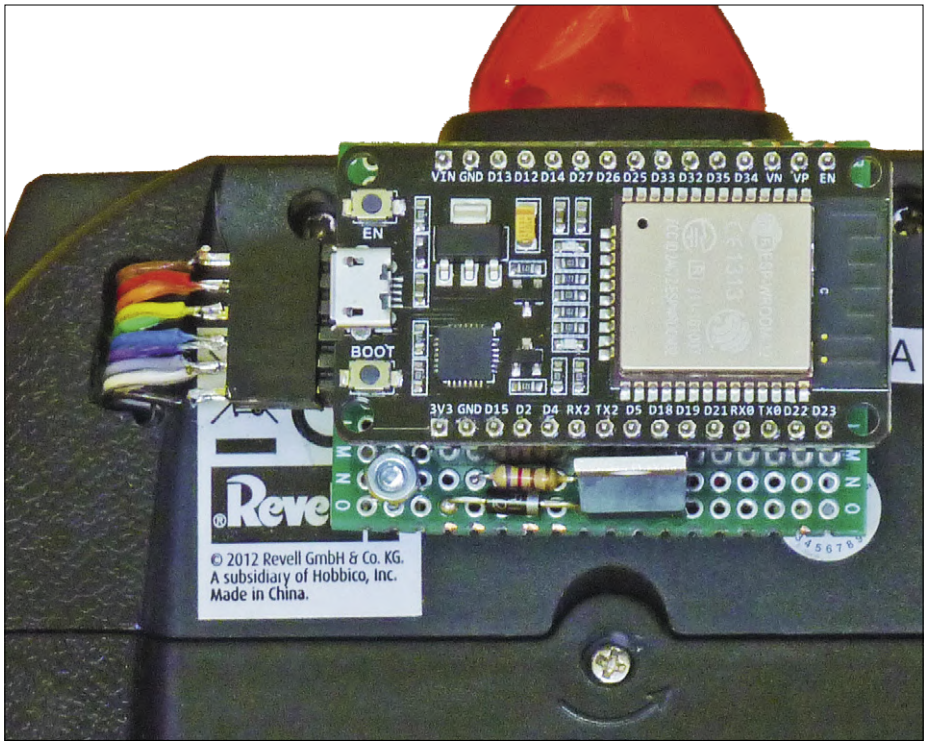


Figure 2. Le module ESP32 est à l'extérieur, adossé à la télécommande radio.

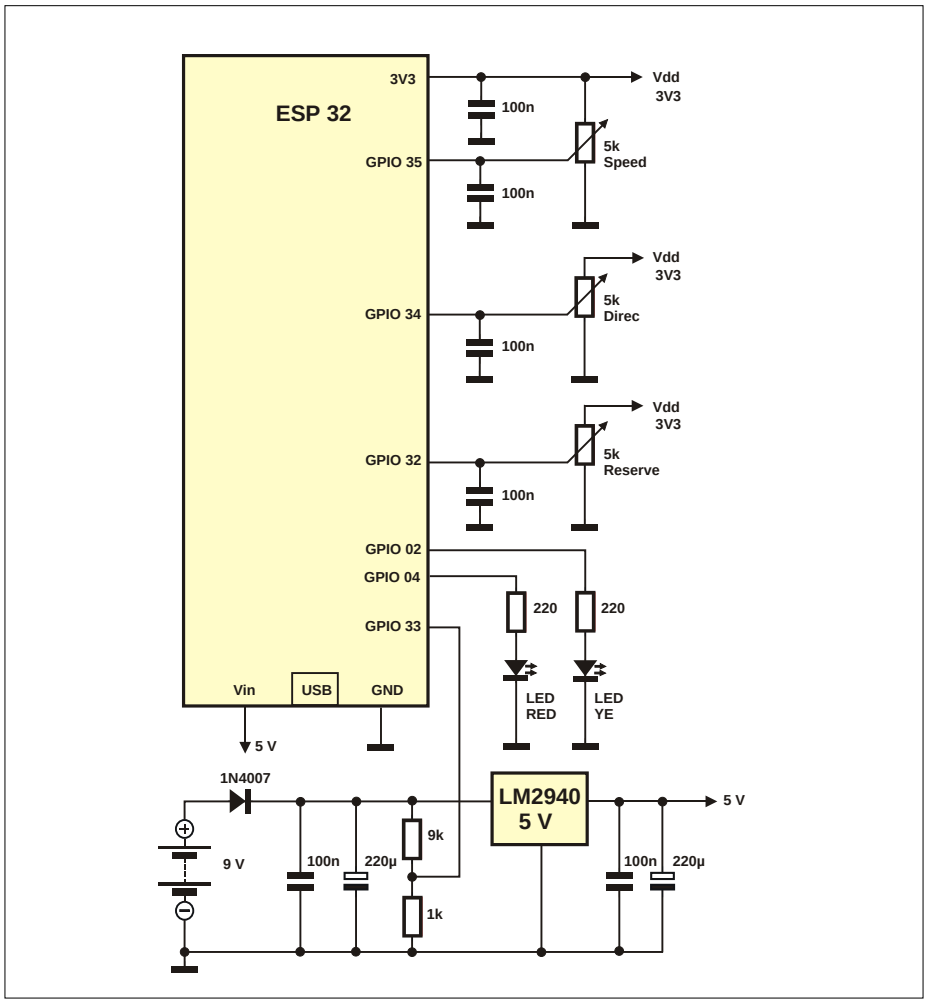


Figure 3. Deux potentiomètres commandent le catamaran. La troisième connexion sera utilisée dans une application future.

commande des ventilateurs du catamaran, l'ESP8266 déploie un réseau Wi-Fi privé et héberge également un courtier MQTT compact.

Un circuit intégré de puissance, le L293B commande les ventilateurs qui consomment jusqu'à 0,5 A à pleine charge. Nous n'utiliserons que deux des quatre pilotes intégrés dans le CI, les autres étant en réserve (par ex. pour un montage en pont avec marche avant/arrière). Des LED sont montées en parallèle sur les ventilateurs de sorte qu'il est facile d'apprécier la puissance fournie. Les moteurs des ventilateurs sont alimentés directement par la tension VM des batteries (de 7 à 8,4 V compte tenu des pertes de la diode de protection et des pilotes). Pour prendre un niveau déterminé au moment de la mise en marche, les lignes GPIO de commande 4, 5, 12 et 13 sont pourvues de résistances de rappel à la masse de 3,3 kΩ.

La tension des batteries est surveillée en permanence. Elle parvient à l'unique entrée analogique de l'ESP8266 via un diviseur de tension 10:1. Étant donné que la plage de mesure s'étend de 0 à 1 V, le diviseur permet d'acquérir des tensions atteignant 10 V. Pour un fonctionnement correct, d'autres ports du µC sont câblés : CH_PD et GPIO 2 sont reliés à Vdd et GPIO 15 à la masse (GND). Pour la programmation, les broches RST, GPIO 0, TX et RX sont reprises sur un connecteur rapide à six broches et rappelées à Vdd par des résistances idoines. Pour la tension d'alimentation du contrôleur, un régulateur linéaire LM2950-3.3 fournit une tension stabilisée de 3,3 V à partir des batteries. Des condensateurs de découplage placés avant et après le régulateur éliminent les parasites de commutation des moteurs. Deux diodes 1N4007 protègent l'électronique contre l'inversion accidentelle de la polarité de branchement des batteries.

Utilisation conviviale avec une console modifiée

La console de pilotage de l'hélicoptère mise au rebut trouve ici une seconde vie. La transmission infrarouge d'origine ne convenait pas pour ce projet, mais le compartiment à piles, les trois potentiomètres de commande et une LED double de couleur jaune et rouge ainsi qu'un commutateur marche/arrêt permettaient de constituer une « interface utilisateur » très agréable pour piloter l'aéroglisseur. Pour communiquer avec le catamaran,

il nous fallait une connexion Wi-Fi et un client MQTT. L'ESP32 convient parfaitement pour fournir ces fonctions. Et pourquoi son petit frère, l'ESP8266 monté sur le catamaran et qui a beaucoup plus à faire ne convient-il pas ici ? C'est simplement en raison du nombre des entrées analogiques : l'ESP8266 n'en a qu'une tandis que l'ESP32 en met plusieurs à disposition.

Comme la plus grande partie de la place disponible dans le boîtier de la console est occupée par le tiroir des piles (6×AA), l'ESP32 a dû être monté à l'extérieur. Comme on le voit sur la **figure 2**, la petite platine de développement de l'ESP32 avec port de programmation USB est reliée aux composants de la console par un câble plat.

La **figure 3** illustre bien la simplicité du câblage. La console dispose d'un potentiomètre de vitesse avant ainsi que d'un manche de direction équipé de deux potentiomètres. Il suffit donc de relier les curseurs à des entrées analogiques de l'ESP32. Pour piloter le catamaran, nous n'avons besoin que d'une commande de vitesse et d'une commande de direction gauche/droite. La commande avant/arrière du manche est mise en réserve et pourra éventuellement être utilisée pour un autre projet. Les entrées analogiques couvrent une plage de 0 à 3,3 V, c'est précisément la plage d'excursion des potentiomètres. La tension de la batterie est transmise à la broche d'entrée analogique GPIO 33 (ADC1_CH5) via un diviseur 10:1.

Les deux sorties tout ou rien GPIO 02 et GPIO 4 commandent la LED double. La jaune sert à indiquer le fonctionnement et la rouge avertit que la batterie du catamaran ou de la console est faible. Pour la tension d'alimentation, nous utilisons là aussi un régulateur linéaire, le LM2940-5 qui délivre une tension de 5 V. Un régulateur secondaire placé sur le module de développement alimente l'ESP32 sous 3,3 V. Cette double régulation augmente la stabilité !

Une bonne préparation = 50% du chemin

Pour le logiciel du projet, nous utiliserons l'EDI Arduino, l'environnement de développement bien connu et apprécié de tous. Non seulement ce dernier est convivial, mais il garantit l'accès à de très nombreuses bibliothèques et exemples d'applications. Par bonheur, les µC ESP8266 et ESP32 d'Espressif peuvent être intégrés à cet EDI grâce à des noyaux complémentaires. Pour démarrer le développement du logiciel, il faut préparer l'environnement :

- Installation de l'EDI Arduino : télécharger la version à jour depuis [2] et l'installer dans l'EDI.
- Installation du noyau ESP8266 : le lien [3] explique comment intégrer le module complémentaire dans l'EDI.
- Installation du noyau ESP32 : vous trouverez ici [4] d'excellentes instructions d'installation.

Une fois l'EDI installé avec les deux noyaux complémentaires, il faut encore intégrer des bibliothèques supplémentaires. C'est l'objet des paragraphes suivants qui indiquent également les paramètres à spécifier.

En bref : MQTT

Il était naturellement possible de réaliser la communication entre le catamaran et la console au moyen de simples paquets TCP, mais l'utilisation des fonctions d'un courtier MQTT sur le petit ESP8266 présentait un intérêt plus grand. Il y a en outre un autre avantage : il est possible de lancer un client MQTT sur un PC afin d'observer l'échange complet des données. Si vous ne connaissez pas du tout le protocole MQTT, vous avez intérêt à lire l'**encadré**.

Le courtier est la centrale de commande du protocole de messages MQTT. Il existe de nombreux courtiers MQTT, appartenant à des constructeurs ou disponibles en libre accès comme Mosquitto. Cependant, pour que MQTT fonctionne sur un µC aussi peu puissant que l'ESP8266, il faut utiliser un courtier spécial, réduit aux fonctions essentielles pour tenir dans la faible quantité de mémoire disponible. Malgré ces limitations, le courtier utilisé ici sur l'ESP8266 prend en charge :

- simultanément les versions MQTT v3.1 et v3.1.1 ;
- jusqu'à huit clients connectés ;
- le niveau 0 de qualité ;
- le testament avec dernière volonté ;
- la mise à disposition de messages, y compris pour les clients nouvellement connectés (*retained*) ;
- l'authentification au moyen d'un nom

En bref : MQTT

À l'inverse du protocole HTTP bien connu (basé sur un mode requête/réponse), le protocole développé par la société IBM (et désormais libre de droits) MQTT (*Message Queuing Telematic Transport*) possède une architecture publication/abonnement qui met en œuvre un courtier (serveur) central. Les appareils qui souhaitent communiquer quelque chose envoient (publient) leur message au courtier, qui le répercute vers les autres appareils qui se sont préalablement abonnés à ce type de message. La communication peut fonctionner dans les deux sens, un participant peut donc à la fois publier et être abonné. Un nœud peut être un simple microcontrôleur, un PC ou un serveur Linux. La condition sine qua non est simplement qu'une pile TCP et le protocole MQTT y soient implémentés.

L'adressage lors des envois et des réceptions se fait par l'intermédiaire de ce que l'on appelle des *topics* (sujet). Il s'agit de chaînes de caractères qui sont une sorte d'objet décrivant les messages, mais dont la construction rappelle celle d'une URL. Un capteur de température dans un bureau pourrait par ex. publier la température dans un *topic* comme « MaMaison/Bureau/Température ». Outre le *topic*, les données utiles (*payload*) et d'autres paramètres sont transmis. Les données peuvent être transmises sous forme binaire, de texte et même de structure XML ou JSON.

Il existe trois niveaux pour assurer la qualité de la transmission des messages.

- **Niveau 0** : absence de contrôle particulier de la qualité selon le principe : lancer et oublier.
- **Niveau 1** : transmission garantie d'au moins un message, sans exclure de copies.
- **Niveau 2** : transmission garantie d'exactement un message sans aucune copie.

Avez-vous déjà entendu parler d'un protocole avec testament ? C'est également possible avec MQTT. Si un nœud est défaillant, il peut envoyer au courtier ses « dernières volontés ». Elles devront être exécutées après la défaillance. Cela pourrait par ex. être l'ordre d'informer l'administrateur compétent.

Il y aurait encore beaucoup à dire à propos de MQTT : filtrage, programme de sécurité à plusieurs niveaux, protection en cas de défaillance, etc. Le mieux est de vous informer sur l'internet !

Listage 1. Initialisation.

```
#include
#include
#include "Ticker.h"
#include

#define speed_TOPIC    "speed"
#define direc_TOPIC    "direc"
#define katama_TOPIC   "katama"

const char* ssid = "ESP_HOST";
const char* password = "";
const char* mqtt_server = "192.168.4.1";
```

Listage 2. Instructions MQTT pertinentes.

```
WiFiClient espClient; // Définitions : clients WiFi et MQTT
PubSubClient client(espClient);

client.setServer(mqtt_server, 1883);
// Configuration : adresse et port du courtier (broker)
client.setCallback(receivedCallback);
// Configuration : démarrage de Callback pour l'abonnement

if (!client.connected()) {
// Configuration : connexion au courtier
  mqttconnect(); }

client.loop(); // Boucle (loop) : appel du client MQTT et
send_MQTT();  // exécution des actions nécessaires

client.publish(speed_TOPIC, spdmqt);
// Boucle (loop) : publication des données de réglage
delay(100);
client.publish(direc_TOPIC, dirmqt);
delay(100);

// Fonction de rappel (Callback)
void receivedCallback(char* topic, byte* payload, unsigned int
length)
```

Commandes MQTT utilisées

Topic	Payload	Valeurs
speed	speed_xxx	speed_000 = min., speed_100 = max.
direc	direc_xxx	direc_000 = gauche, direc_100 = tout droit, direc_200 = droite
vbatt	vbatt_xxx	vbatt_999 = 9,99 V

et d'un mot de passe.

C'est déjà très satisfaisant et nous nous accommoderons des lacunes ci-dessous :

- absence des niveaux de qualité 1 et 2 (garantie de livraison) ;
- absence de cryptage de sécurité TLS (*Transport Layer Security*) ;
- mémorisation permanente des publications (*non-clear sessions*).

Le développeur de ce logiciel libre [5], martin-ger, a choisi comme plateforme l'ESP8266 car ces µC sont utilisés dans les commutateurs Sonoff bon marché ce qui permet d'économiser un PC ou un Raspberry Pi complémentaire pour accueillir le courtier. Le programme développé est très efficace : selon mes mesures, le courtier parvient à exécuter environ 100 publications/abonnements par seconde. Chapeau Martin !
Sous les *topics* « speed » et « direc », la console publie les données du manche de commande à chaque fois que la position de ce dernier change, et au moins une fois toutes les 10 s. Le *payload* renferme aussi la description du *topic*, cette dernière sert simplement à mieux observer le déroulement de la communication. Le catamaran envoie également de façon cyclique la tension des batteries mesurée par la console, au cas où elle serait trop faible.

Logiciel de la console

Le logiciel de la console de commande est également réalisé sous forme de croquis Arduino. Il transmet via MQTT la position en cours du manche de commande pour indiquer au catamaran la vitesse et la direction. En outre, il vérifie en permanence la tension des batteries de la console et du catamaran et signale toute valeur trop faible en faisant clignoter la LED rouge. La LED jaune clignote toutes les secondes pour indiquer le fonctionnement correct.
Le croquis baptisé **Konsole.ino** et les bibliothèques nécessaires sont à votre disposition sous [6]. Dans l'EDI Arduino, sous « Outils/carte », sélectionnez votre module ESP32 (par ex. « ESP32 Dev Module ») et définissez votre numéro de port de communication ainsi que sa vitesse de transmission (115200 bits/s). Le client MQTT se trouve dans le fichier « pubsubclient-master.zip ». Il faut l'installer sous « Croquis/inclure une bibliothèque/.ZIB-Bibliothek installer ». En outre les deux fichiers « Ticker.h » et

« Ticker.cpp » doivent se trouver dans le dossier du projet.
La structure du programme est donnée par la **figure 4** et correspond à un modèle Arduino type avec routines de configuration (*setup*) et boucle (*loop*). Pour que l'attente de l'arrivée d'un message MQTT ne bloque pas la boucle, l'abonnement à la tension de la batterie du catamaran est hébergé dans la fonction de rappel `receivedCallback()`. L'arrivée d'un message active cette fonction qui évalue alors le message. Deux routines complémentaires sont lancées avec intervalles respectifs de 10 s et 1 s par les temporisateurs de la bibliothèque `Ticker`. La fonction `firemqtt()` lance au moins une fois toutes les 10 s une publication des données du manche de commande, tandis que la fonction `blinker()` fait clignoter les LED au rythme d'1 Hz. Comme d'habitude, la fonction `setup` effectue des initialisations : établissement de la liaison avec le réseau Wi-Fi, démarrage du client MQTT avec établissement de la liaison avec le courtier, lancement des fonctions `Callback`…
Dans la boucle sans fin, nous commençons par acquérir les valeurs analogiques des potentiomètres et en cas de modification, elles sont immédiatement publiées à destination du catamaran. Si les potentiomètres ne bougent pas, la transmission a lieu à intervalle de 10 s. Dans ce cycle, nous mesurons également les tensions des batteries (et si nécessaire, nous activons l'avertissement LED en alternance avec le clignotement jaune de fonctionnement).
Au début du programme (**listage 1**) sont incluses les bibliothèques nécessaires pour le Wi-Fi, MQTT et le Ticker de l'interruption du temporisateur. Ensuite, les topics MQTT ainsi que les données d'accès pour le réseau Wi-Fi exclusif du catamaran et du courtier sont élaborés. La protection d'accès n'est pas exploitée ici, le SSID du réseau est *ESP_HOST* et il n'y a pas de mot de passe. Pour un projet plus sérieux, il serait bien sûr nécessaire de définir un mot de passe, d'utiliser un cryptage convenable et de protéger l'accès au courtier par un nom et un mot de passe. L'adresse IP du serveur MQTT *192.168.4.1* est prédéfinie.
Vous trouverez ensuite les instructions MQTT idoines (**listage 2**, ici aussi, la position est indiquée dans le code source). Les publications interviennent dans la boucle toutes les 100 ms, afin que le catamaran ne soit pas surchargé

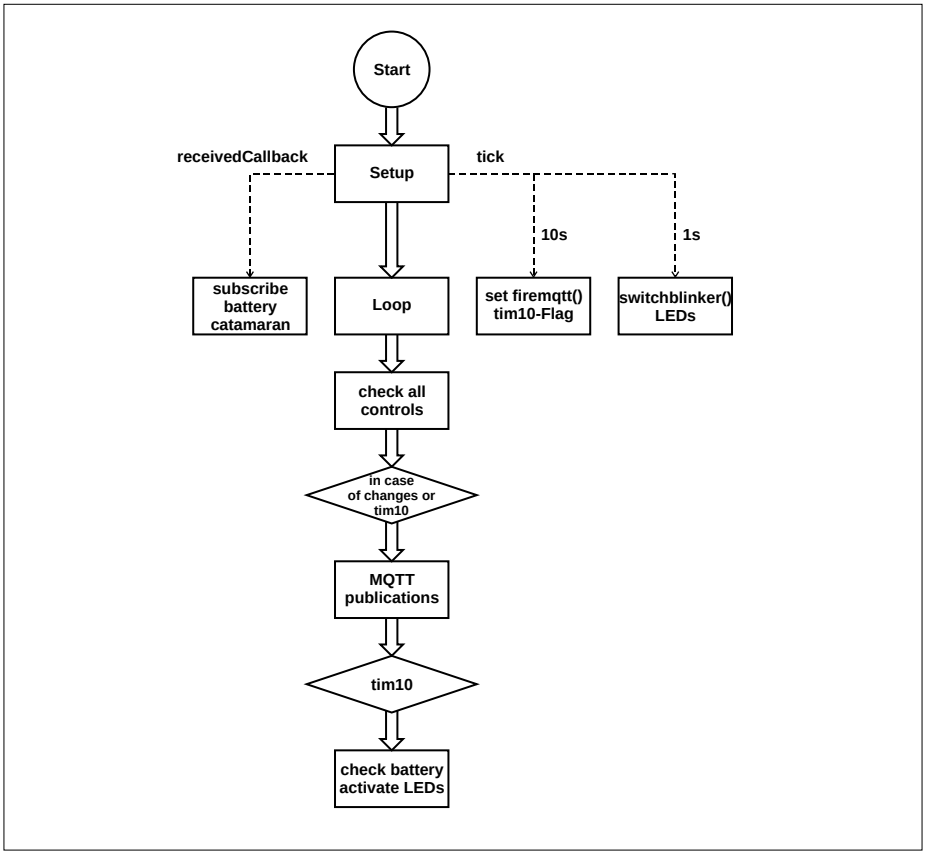


Figure 4. Le programme de la console : un modèle Arduino classique.

Listage 3. Les deux tickers (temporisateurs).

```
Ticker minmqt; // Définition : affectation du ticker
Ticker ledblink;

ledblink.attach_ms(1000, blinker);
// Configuration : intervalle de temps et fonction de service
minmqt.attach(10, firemqtt);

void firemqtt(){  // Fonction : firemqtt()
tim10 = true;
}
```

Liens

[1] Compilation ESP32/ESP8266, Elektor : www.elektor.fr/e-book-compilation-esp32-esp8266-fr
[2] EDI Arduino : www.arduino.cc/en/Main/Software
[3] Démarrer avec l'ESP8266 : https://f-leb.developpez.com/tutoriels/arduino/esp8266/debuter/
[4] Démarrer avec l'ESP32 (en anglais) : www.elektormagazine.com/labs/esp32-getting-started
[5] martin-ger : https://github.com/martin-ger
[6] Page de l'article : www.elektormagazine.fr/170198-03
[7] Courtier Arduino : https://github.com/martin-ger/uMQTTBroker
[8] Courtier en C : https://github.com/martin-ger/esp_mqtt/
[9] Courtier sur Youtube: www.youtube.com/watch?v=0K9q4IuB_oA

Listage 4. Détails du logiciel du catamaran.

```
#include
#include "uMQTTBroker.h" // Courtier MQTT avec client
#include "Ticker.h"
#include

char ssid[] = "ESP_HOST"; // SSID (nom) du réseau Wi-Fi du catamaran
char pass[] = ""; // Mot de passe Wi-Fi, (ouvert ici)
bool WiFiAP = true; // Un réseau Wi-Fi privé doit-il être
configuré ?

#define speed_TOPIC "speed" // Définition du topic
#define direc_TOPIC "direc"
#define katama_TOPIC "katama"

myMQTTBroker myBroker; // Définition de l'instance du courtier
```

Listage 6. Instructions WiFi et MQTT.

```
if (WiFiAP) // Wi-Fi propre…
startWiFiAP();
else
startWiFiClient(); // ou insertion dans le réseau privé

myBroker.init();
// Configuration : initialisisation du courtier

myBroker.subscribe(speed_Topic);
// Abonnement pour vitesse et direction
myBroker.subscribe(direc_Topic);

myBroker.publish(katama_TOPIC, batmqt);
 // Boucle (loop) : publication de la tension de la batterie
```

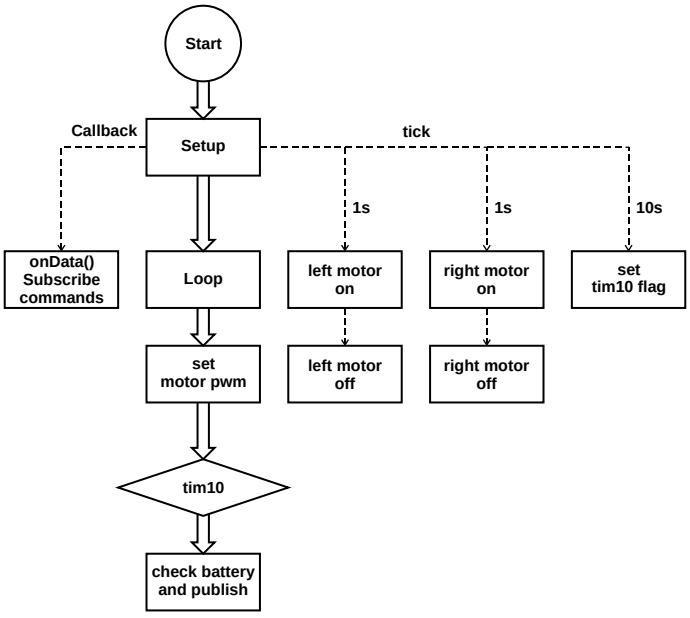


Figure 5. Organigramme de commande du catamaran.

par l'acquisition des données. Il ne reçoit ainsi qu'au maximum dix messages par seconde, ce que le courtier peut aisément fournir. La fonction `Callback` délivre toutes les données des abonnements reçus comme les *topics*, *payload* et la longueur du *payload*.

Les deux tickers sont définis au début du programme (**listage 3**). Dans le Setup, un intervalle en millisecondes et en secondes ainsi qu'une fonction de service vous sont attribués. Ces fonctions d'interruption devraient être aussi courtes que possible et s'exécuter rapidement. La fonction `firemqtt()` lève simplement un drapeau et la boucle en tient compte. Toutes les autres unités du programme sont disponibles en code source et commentées. Notez la présence de sorties `Serial.print` avec lesquelles vous pourrez suivre le déroulement du programme dans le moniteur Arduino.

Le logiciel du catamaran

Pour le programme du catamaran (**fig. 5**), il y a également quelques réglages de l'EDI Arduino à effectuer. Sous *Outils/Carte* nous choisissons le module ESP8286 correct (ici *Generic ESP8266 Module*) et nous définissons le numéro de port de communication et sa vitesse (115200 bauds). Pour la méthode de réinitialisation, le mieux est d'activer *nodemcu* afin de supprimer la pénible manipulation de touches lors du flashage. Il est également important de choisir *LwIP Variant* avec la valeur *v1.4 Higher Bandwidth* pour attribuer la taille maximale à la pile TCP.

Le logiciel courtier pour l'EDI Arduino est disponible sous [7], une alternative en version C existe sous [8]. Nous recommandons ensuite de suivre le film Youtube [9] de l'auteur qui présente une très bonne description des fonctions. Le logiciel prêt à l'utilisation est dans le dossier Elektor [6]. Il faut l'enregistrer dans votre dossier de projet, une installation dans l'EDI Arduino n'est pas obligatoire. Par défaut, l'adresse IP du serveur MQTT est *192.168.4.1*.

Le programme de commande du catamaran est baptisé **Katamaran.ino** et se trouve dans le dossier de projet. Son architecture est semblable à celle du logiciel de la console. Là encore, nous lançons dans Setup les routines `Callback` et `Ticker`, dans lesquelles la fonction `onData()` est toujours activée si un message arrive de la console et les fonctions Ticker démarrent avec les intervalles définis.

Étant donné que les ventilateurs utilisés tournaient avec difficulté aux fréquences MLI classiques, nous avons opté pour une fréquence très basse de 1 Hz. Ils sont mis en route à la cadence d'une fois par seconde puis coupés après une durée (<1 s) dépendant de la vitesse. L'inertie des rotors des ventilateurs procure une rotation sans à-coups.
Un autre ticker déclenche à intervalle de 10 s la mesure et la publication de la tension de la batterie. La boucle calcule les intervalles de démarrage et les passe aux fonctions Ticker.

Le **listage 4** propose d'intéressants extraits du logiciel. Dans les définitions, nous effectuons l'appel de la bibliothèque *WiFiClient*, du courtier MQTT et des fonctions Ticker. Le nom du réseau Wi-Fi du catamaran est *ESP_HOST*, aucun mot de passe ni aucun niveau de sécurité ne sont spécifiés. Le fait d'attribuer à *WiFiAP* la valeur *true* définit un point d'accès privé. Le courtier MQTT met à disposition trois fonctions Callback importantes (**listage 5**) :

- `onConnect` : s'active lors de la connexion d'un nouveau client et transmet son adresse IP ainsi que le nombre de clients connectés. Il est ainsi possible de déterminer si ce client est le bienvenu et s'il peut être servi.
- `onAuth` : lors de la connexion avec un nouveau client, cette fonction lui fournit nom et mot de passe (si définis). Si le retour est vrai (*true*), l'autorisation est accordée. Dans ce projet, tous les clients sont autorisés, avec ou sans données d'utilisateur.
- `onData` : reçoit tous les *topics* objets d'un abonnement et transmet la des-

Listage 5. Fonctions de rappel du courtier MQTT.

```
class myMQTTBroker: public uMQTTBroker
{
public: // Un client se connecte
virtual bool onConnect(IPAddress addr, uint16_t client_count) {
Serial.println(addr.toString()+" connected");
return true; // true = autorisé, false = non autorisé
}
// Authentification du client
virtual bool onAuth(String username, String password) {
Serial.println("Username/Password: "+username+"/"+password);
return true; // true = autorisé, false = non autorisé
}
// Début d'un abonnement
virtual void onData(String topic, const char *data, uint32_t length) {
char payload[length+1];
char payval[4] = "000";

os_memcpy(payload, data, length);
payload[length] = '\0';
if(topic == speed_TOPIC){ // Réception d'une commande de vitesse
payval[0] = payload[6];
payval[1] = payload[7];
payval[2] = payload[8];
spdval = atoi(payval);
if((spdval >= 0) && (spdval <= 100)) // Plage autorisée : 0 à 100
spdok = true;
}
if(topic == direc_TOPIC){
// Réception d'une commande de direction
payval[0] = payload[6];
payval[1] = payload[7];
payval[2] = payload[8];
dirval = atoi(payval);
if((dirval >= 0) && (dirval <= 200)) // Plage autorisée : 0 à 200
dirok = true;
}
Serial.println("received topic '"+topic+"' with data '"+(String)payload+"'");
}
};
```

cription du *topic* et le *payload* avec sa longueur. Ensuite intervient la transmission de la partie numérique du *payload* ainsi que sa conversion en variables entières (`spdval` ou `dirval`). Ensuite, on vérifie si cette valeur appartient à l'intervalle autorisé puis elle est transmise à la commande des moteurs au moyen de drapeaux.

Le **listage 6** comporte les instructions Wi-Fi et MQTT adéquates.
La commande proprement dite des moteurs est proposée au **listage 7** basé sur quatre tickers. Les fonctions de démarrage `Tlion et Treon` sont lancées à intervalles fixes de 1000 ms. Les fonctions de service `Molion` et `MoReon` démarrent les moteurs des ventilateurs et lancent les routines `Molioff` et `Moreoff` qui les arrêtent grâce aux longueurs d'impulsion calculées. Toutes les 10 s, un cinquième ticker commute le drapeau `tim10` pour déclencher la mesure de la batterie.
Nous avons observé que l'EDI Arduino s'accommode parfaitement de cet exercice et nous n'avons rencontré lors du développement aucun problème ni aucune restriction d'utilisation. Nous prouvons ainsi qu'un environnement complexe n'est pas obligatoire pour maîtriser des projets peu ou moyennement difficiles.

Comment les enfants ont-ils apprécié ce projet ?

Mes petits-enfants ont réagi avec enthousiasme et immédiatement essayé le catamaran dans la mare la plus proche. Le catamaran s'est comporté comme prévu. Il répondait encore bien à 50 m de distance. En raison de sa construction relativement haute, de son faible poids et de son faible tirant d'eau, le catamaran est assez sensible au vent. Mais cette embarcation pourrait être mécaniquement perfectionnée, par ex. en l'équipant d'hélices.

Pour ma part, le projet a montré que le courtier MQTT de l'ESP8266 peut bien être utilisé pour de petits projets, c'est juste une question d'imagination. Le plus important fut la grande joie que ce travail avec mes petits-enfants a pu me procurer. Je souhaite qu'il en soit ainsi pour vos propres projets ! ⏮

(170198-03 – version française : Yves Georges)

Listage 7. Commande du moteur.

```
// Configuration : définition du ticker
Ticker Tlion; // Mise en marche du moteur gauche
Ticker Tlioff; // Arrêt du moteur gauche
Ticker Treon; // Mise en marche du moteur droit
Ticker Treoff; // Arrêt du moteur droit
Ticker Tick10; // Ticker 10 s

Tlion.attach_ms(1000, Molion);
// Démarrer le moteur gauche toutes les secondes
Treon.attach_ms(1000, Moreon);
// Démarrer le moteur droit toutes les secondes
Tick10.attach(10, SendBatt); // Ticker 10 s

void Molioff(){ // Couper le moteur
digitalWrite(MOLI,LOW);
}
void Molion(){ // Mettre le moteur en marche
digitalWrite(MOLI,HIGH);
Tlioff.attach_ms(molipow, Molioff);
 // Démarrer le moteur pour une durée variable de 0 à 1000
}

void SendBatt(){ // Lever le drapeau du ticker
tim10 = true;
}
```

Conseils d'installation

Contenu du dossier du logiciel

`Konsole.ino` : programme de commande de la console sur ESP32
`Katamaran.ino` : programme de commande du catamaran sur ESP8266, courtier MQTT inclus
`uMQTTBroker-master.zip` : bibliothèque avec fonction courtier MQTT pour *Katamaran.ino*
`pubsubclient-master.zip` : bibliothèque avec client MQTT pour *Konsole.ino*
`Ticker.h / Ticker.cpp` : bibliothèque pour ticker
`mqtt-brok-E.zip` : version C du courtier MQTT pour utilisation dans l'EDI Espressif (non nécessaire avec l'EDI Arduino)

Installation du logiciel

EDI Arduino : installer les noyaux ESP8266 et ESP32
Catamaran : installer le fichier compressé *uMQTTBroker-master.zip* dans l'EDI Arduino
Console : installer le fichier *pubsubclient-master.zip* dans l'EDI Arduino

@ WWW.ELEKTOR.FR

→ Compilation ESP32/ESP8266
www.elektor.fr/e-book-compilation-esp32-esp8266-fr

convertisseur A/N à PLD, simple à construire

réaliser un CA/N sigma-delta avec le minimum d'effort

Guido Nopper

Vous pouvez faire plein de choses avec les PLD (*programmable logic device*), les réseaux logiques programmables. Par exemple, avec seulement 26 macrocellules, on peut réaliser un CA/N du type ΣΔM, un modulateur sigma-delta à 8 bits, avec en aval un filtre passe-bas et de conversion numérique. Nous en analysons ensuite la précision, la résolution et le coût. Il est curieux de constater que c'est le filtrage numérique qui réclame le plus gros de l'effort.

Il existe différentes façons de convertir des signaux analogiques en valeurs numériques. Pareil circuit s'appelle convertisseur analogique numérique que l'on raccourcit volontiers en CA/N, en anglais ADC (*analog to digital converter*). Celui décrit ici se base sur la technique du suréchantillonnage (*oversampling*).

Suréchantillonnage

Il consiste à échantillonner le signal d'entrée analogique à une fréquence suffisamment haute, comme l'exige le théorème d'échantillonnage de Nyquist-Shannon (au moins deux fois la fréquence d'échantillonnage de la partie spectrale la plus élevée du signal utile). La **figure 1** en symbolise l'idée. Le modulateur produit un flux de données de 1 bit avec le débit binaire $\mathbf{f_S}$. On se doute que la résolution d'un bit est très faible. En revanche, la résolution dans le domaine des fréquences est nettement plus élevée. On appelle **R** ou OSR (*OverSampling Ratio*) le rapport de suréchantillonnage ; il se situe souvent à plusieurs puissances de deux (32, 64, 128, etc.) au-dessus de la fréquence à traiter.

Le filtre de conversion qui suit a pour fonction d'augmenter la résolution de l'amplitude sur un certain nombre de bits du flux de données par calcul de la moyenne, ce qui correspond à une fonction passe-bas. Cela permet de réduire la fréquence d'échantillonnage jusqu'à la valeur $\mathbf{f_{SO}}$ (Nyquist) sans que le contenu du signal non désiré ne se replie trop dans la plage utile.

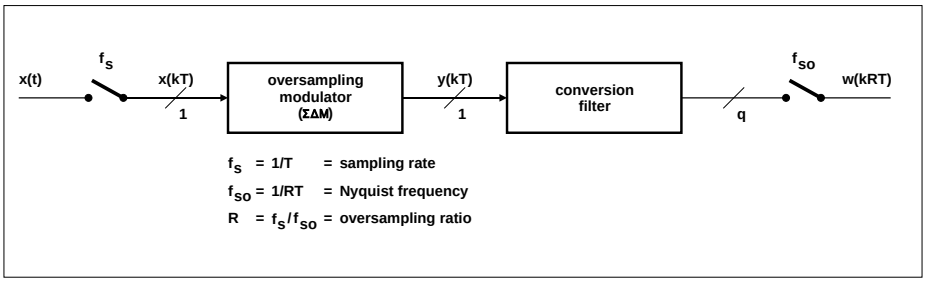


Figure 1. L'archétype du convertisseur analogique/numérique à suréchantillonnage.

La **figure 2** montre le spectre du signal de la figure 1. Le suréchantillonnage avec une f_S élevée a l'avantage de simplifier le filtre antirepliement du spectre, d'ordre

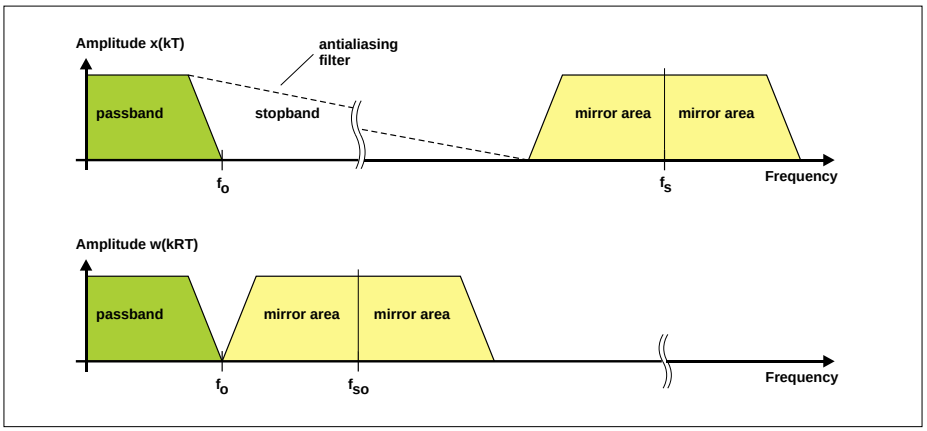


Figure 2. Les spectres des signaux de la figure 1.

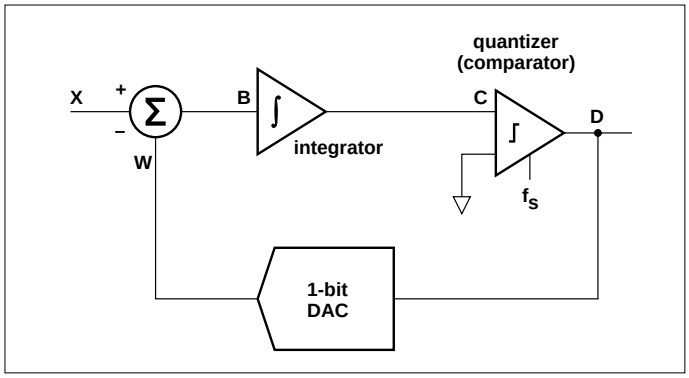


Figure 3. Diagramme fonctionnel d'un modulateur sigma-delta simple.

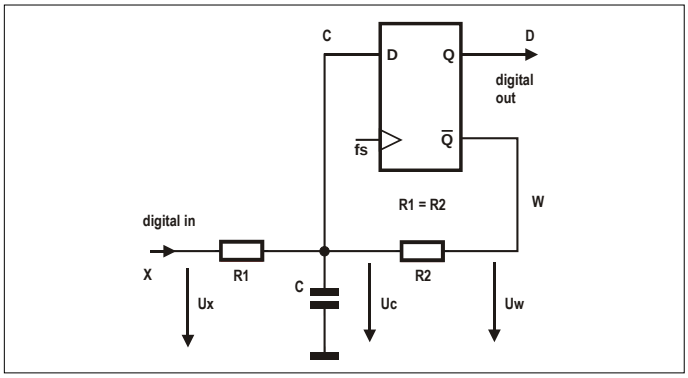


Figure 4. Le plus simple des modulateurs sigma-delta.

plus bas, voire simplement omis. Mais en réduisant la fréquence d'échantillonnage à la sortie sur f_{SO}, des composantes du signal non désiré risquent d'être repliées vers la bande utile.

Modulateur sigma-delta

Un ΣΔM (*Sigma Delta Modulator*) d'ordre un réunit un soustracteur, un intégrateur, un quantificateur (comparateur) et un CN/A à 1 bit. Diagramme fonctionnel à la **figure 3**. On peut aussi l'appeler « Delta Sigma Modulator », du fait que la différence entre entrée (point X) et sortie (analogique par le CN/A à 1 bit, point W), est formée dans le soustracteur (Δ) (point B) et ce n'est qu'ensuite que l'intégration (Σ) a lieu. Outre le signal de différence intégré (point C), le comparateur reçoit également la fréquence d'échantillonnage f_S. Cela signifie que le signal de sortie (point D) ne peut changer qu'au moment de l'échantillonnage et est donc quantifié avec f_S. Ces modulateurs sont connus depuis de nombreuses années [1].

Un modulateur simple de ce genre (**fig. 4**), c'est une bascule de type D, deux résistances et un condensateur. Si la fréquence d'échantillonnage f_S est nettement supérieure à la fréquence maximale du signal utile, on parle alors de conversion A/N par suréchantillonnage. Les points X, C, D et W de la figure 4 correspondent approximativement à ceux de la figure 3. Il est également facile de s'apercevoir que la simplicité du circuit se paie surtout par une plus grande imprécision.

Sur le ΣΔM (fig. 4), le signal de sortie numérique inversé (point W) subit une conversion N/A par le passe-bas RC et est ajouté au signal d'entrée. En raison de l'inversion, cela correspond à une soustraction et à une comparaison. Les niveaux au point W représentent les tensions de référence pour la conversion A/N et doivent donc satisfaire aux exigences analogiques. Une bascule CMOS permet d'approcher les niveaux haut VCC et bas GND.

L'entrée D de la bascule sert de comparateur analogique. Son niveau de déclenchement est donc déterminant pour la précision du système. La tension U_C sur le condensateur (point C) reste toujours proche de ce niveau. Dès que la tension d'entrée au point X augmente comme celle sur C, la bascule émet un pourcentage plus élevé de bits bas au point W, ce qui réduit U_C. Le ΣΔM essaie donc de suivre la tension d'entrée analogique avec la moyenne de ses niveaux de sortie.

Plage de tensions d'entrée et fonction de transfert

Pour que le ΣΔM de la fig. 4 ne la limite pas, la tension d'entrée ne doit être ni trop haute ni trop basse. La bascule n'émet que des niveaux hauts ou bas. Sans tenir compte du courant à l'entrée D ni du courant résiduel du condensateur céramique C, les courants moyens à travers les deux résistances R1 et R2 doivent être en moyenne les mêmes. Cela signifie :

$$I_{R1} = I_{R2}$$

$$\frac{U_X - U_C}{R1} = \frac{U_C - U_W}{R2} \quad \text{avec R1 = R2}$$

$$U_X - U_C = U_C - U_W$$

$$U_X = 2U_C - U_W$$

Pour une tension d'entrée maximale, le point *W* doit rester au niveau bas :

$$U_{Xmax} = 2U_C - U_{Wlow}$$

Pour une tension d'entrée minimale, le point *W* doit rester au niveau haut :

$$U_{Xmin} = 2U_C - U_{Whigh}$$

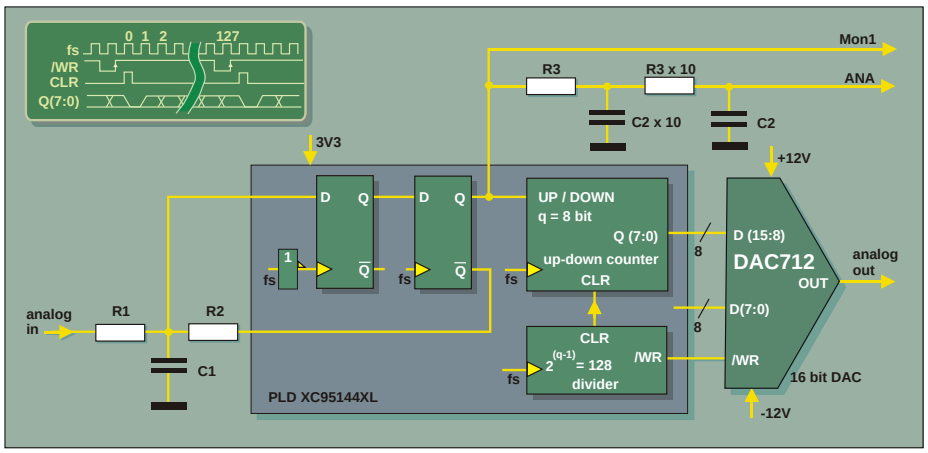


Figure 5. Diagramme fonctionnel d'un banc d'essai pour CA/N ΣΔ.

Dans les hypothèses U_{Wlow} = 0 à 5 %V_{CC}, U_{Whigh} = 95 à 100 %V_{CC} et le niveau de déclenchement U_C = 45 à 55 %V_{CC}, on obtient :

U_{Xmax} = 85 à 110 %V_{CC}

et :

U_{Xmin} = −10 à 14 %V_{CC}

L'augmentation de la tension d'entrée ΔUX est la différence entre U_{Xmax} et U_{Xmin} :

ΔUX = 90 à 100 %V_{CC}

La tension d'entrée peut ainsi, dans certaines circonstances, atteindre les limites et la hauteur de V_{CC}.
Lors de la détermination de la résistance dynamique d'entrée, on peut supposer que U_C est maintenue quasi constante et que ce point a donc une impédance très faible. Pour les tensions alternatives, la résistance d'entrée est donc également R_{INac} :

R_{INac} = R1

Pour ce qu'il en est de la tension continue, il faut tenir compte du fait que R1 à l'autre extrémité est au niveau de U_C. Par conséquent, avec $U_{IN} > U_C$, le courant circule dans l'entrée X ; avec $U_{IN} < U_C$, il en sort. Une difficulté survient de ce que l'entrée D de la bascule est toujours maintenue au niveau de déclenchement et fonctionne donc en dehors des spécifications. Les niveaux d'entrée nominaux haut ou bas ainsi que les temps de transition et de maintien ne sont pas respectés. La bascule peut donc réagir à des états de sortie métastables ; il se peut que ses deux sorties ne soient alors plus exactement en opposition de phase. Pour compenser cela, on peut échantillonner à nouveau le signal de sortie avec une autre bascule D comme l'indique la **figure 5**. Pour ne pas subir trop de retard du signal dans la boucle de régulation, on utilise le front opposé à celui du premier échantillonnage.
Pour éviter d'autres imprécisions, on pourrait désactiver les résistances de rappel haut ou bas du PLD sur l'entrée D-FF. Il y a une fonction « keeper » disponible sous la forme d'une résistance de réaction, dans la zone des kiloohms, au-dessus de l'amplificateur d'entrée interne non inverseur ; on peut l'accepter puisqu'elle ne peut provoquer que peu de variation de la tension du signal dans une horloge d'échantillonnage par rapport à la capacité d'entrée C relativement grande.
Le modulateur sigma-delta de la figure 4 permet d'obtenir un rapport signal/bruit S/N à pleine modulation de :

$$S/N = \frac{3}{4\pi}\left(\frac{f_s}{f_o}\right)^{\frac{3}{2}}$$

où f_S représente la fréquence d'échantillonnage et f_O la fréquence de coupure supérieure de la largeur de bande utile du signal (0 à f_O). On voit que doubler la fréquence d'échantillonnage f_S augmente le rapport signal/bruit d'un facteur de $2^{3/2}$ = 9 dB. Comment est-ce possible ? Avec les CA/N Nyquist conventionnels, le rapport S/N augmente d'un facteur 2 en raison du suréchantillonnage, mais ici seulement d'un facteur de $2^{1/2}$ = 3 dB ! L'explication : avec les CN/A courants, le bruit est réparti uniformément sur la fréquence. Avec le ΣΔM, en revanche, le bruit se déplace vers de plus hautes fréquences parce que les résistances et le condensateur du modulateur agissent comme un passe-bas pour le signal d'entrée et comme un passe-haut pour le bruit.
La **figure 6** montre que le bruit augmente à partir de f_O et reste constant seulement à $f_{S/2}$. La valeur pour f_O résulte de la combinaison de R1, R2 et C (fig. 4), où R1 et R2 sont pratiquement parallèles :

$$f_o = \frac{1}{2\pi\left(\frac{R1 \bullet R2}{R1 + R2}\right)C}$$

et où R1 = R2 = R alors :

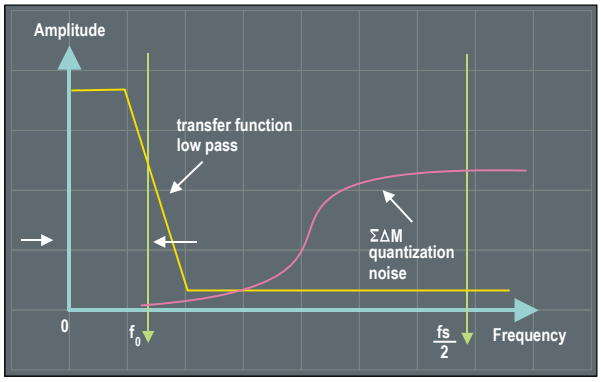


Figure 6. ΣΔM : bruit de quantification et courbe de fréquence du filtre numérique.

$$f_o = \frac{1}{\pi RC}$$

La formule du rapport S/N montre qu'à f_S = 12 MHz et une largeur de bande utile du signal f_O de 14,4 kHz, on peut obtenir un rapport S/N de 5,743 = 75,2 dB. Il est donc judicieux de traiter le signal de sortie du ΣΔM dans le filtre de conversion suivant avec 8 bits ou plus.

Filtre passe-bas et de conversion numérique

La tâche essentielle de ce filtre est d'atténuer le bruit de quantification avant que la fréquence d'échantillonnage ne soit réduite et que ce bruit ne soit replié dans la largeur de bande utile. Comme le bruit de quantification dans un ΣΔM du 1er ordre présente une caractéristique de fréquence ascendante de 20 dB par décade à partir de f_O, un filtre passe-bas numérique du 1er ordre suffit pour une application simple. Une autre fonction de ce filtre est l'augmentation de la largeur de mot de 1 bit sur q bits, ce que l'on appelle aussi un filtre de conversion. Différents types de filtre peuvent convenir. Un compteur/décompteur à 8 bits à réinitialiser, comme à la figure 5, représente un effort minimum pour un tel filtre. Sa fonction de transfert correspond à un **filtre sinc** du 1er ordre avec M = $2^{(q-1)}$ pas. Cependant, la moyenne des bits de sortie $2^{(q-1)}-1$ au lieu de $2^{(q-1)}$ est calculée comme avec le filtre sinc correspondant du ΣΔM. La réinitialisation du compteur après $2^{(q-1)}$ cycles génère une seconde base de temps (f_{SO}), ce qui signifie que la transformation en Z ne peut plus être appliquée en mode fermé [2]. L'avantage de la remise à zéro du compteur le temps d'un cycle est qu'il peut prendre la valeur ±($2^{(q-1)}$ − 1) au maximum/minimum côté sortie. Cela permet d'utiliser au mieux la longueur de mot du compteur et d'éviter de manière fiable le débordement du complément à deux. Le CA/N complet avec un tel filtre et un compteur/décompteur pour q = 8 bits est réalisable avec seulement 26 macrocellules de ce PLD. Chaque canal supplémentaire ne nécessite que 11 macrocellules de plus en partageant le diviseur d'horloge. En utilisant un filtre sinc du 1er ordre avec différenciateur et intégrateur, le CA/N avec filtre va jusqu'à M = 112 avec les 144 macrocellules dis-

ponibles [3]. M correspond au nombre de bits de sortie du ΣΔM, dont la moyenne est calculée en continu. En changeant l'ordre de l'intégrateur et du différenciateur, on pourrait même construire ce circuit avec seulement 50 macrocellules. Si le filtre numérique est en configuration d'accumulateur de pertes avec k = 0,99609375 (m = 8) et une largeur du mot de sortie de 8 bits, il faudra 61 macrocellules au total pour réaliser ce CA/N. Le facteur k est le coefficient multiplicateur par lequel le résultat précédent d'une accumulation doit être multiplié avant d'être ajouté à la nouvelle

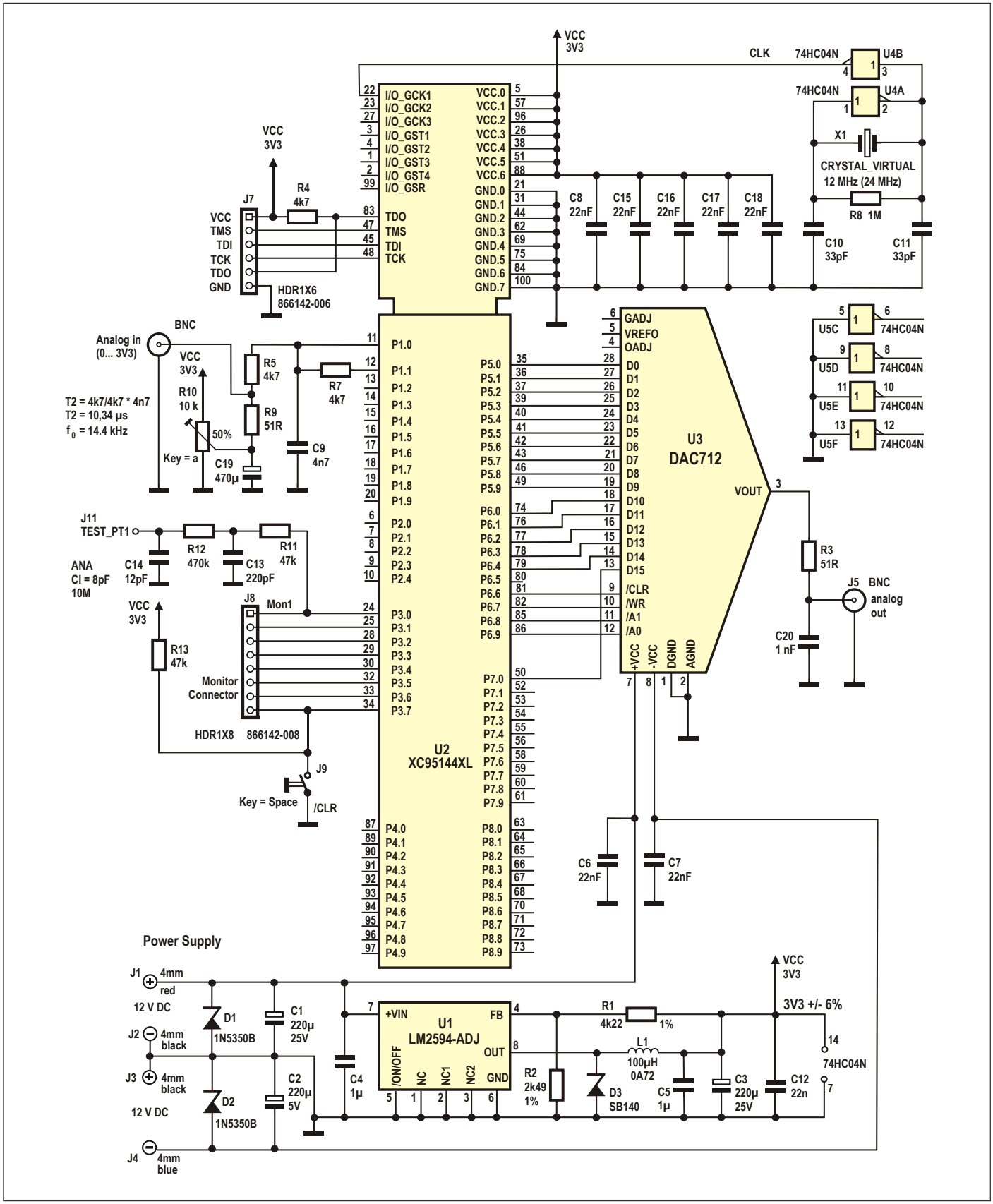


Figure 7. Le circuit du banc d'essai du CA/N ΣΔ.

valeur d'entrée. Si le facteur k est mis en œuvre par décalage et addition au lieu d'utiliser un multiplicateur complet, alors m correspond au décalage de bits. Ainsi, si vous voulez diviser par 2m, le mot de données binaire sera introduit décalé de m positions vers la droite.

Réalisation pratique

Voyez à la figure 5 le diagramme fonctionnel de l'installation de test du CA/N ΣΔ. Le modulateur ΣΔ se compose ici de deux bascules D déclenchées sur différents flancs de la fréquence d'échantillonnage f_S = 12 MHz pour modérer les états métastables. La combinaison de R1, R2 et C1 pour le ΣΔM est calculée pour f_O = 14,4 kHz. La sortie du ΣΔM est connectée à Mon1 pour le contrôle. Après filtrage passe-bas (3e ordre avec f_g = 15,4 kHz), le signal d'entrée analogique contenu dans la séquence d'impulsions de sortie du ΣΔM apparaît à la sortie ANA. Comme l'alimentation du PLD est en 3,3 V, la tension d'entrée analogique ne doit pas dépasser 3,3 V_{cc}.

La sortie du ΣΔM est acheminée vers l'entrée up/down du compteur à 8 bits correspondant. Ici, « up » est interprété comme « +1 » et « down » comme « −1 ». La sortie Q(7:0) du CA/N ΣΔ fournit les données en complément à deux.

Le compteur qui sert de filtre de conversion est d'abord lu tous les 128 coups d'horloge, puis réinitialisé par une longue impulsion d'horloge. Le compteur peut ainsi aller jusqu'à maximum/minimum ±127, profiter de la plage de valeurs de 8 bits tout en évitant le débordement d'un complément à deux. La valeur de sortie maximale ou minimale est obtenue lorsque le signal d'entrée analogique prend la valeur U_{Xmax} ou U_{Xmin}. Si le diviseur est toujours sélectionné 1 bit plus petit que le compteur up/down, le signal de sortie est symétrique et ne peut pas déborder. Le débordement dans un sens ou dans l'autre d'un complément à deux provoquerait le saut du signal du maximum au minimum ou l'inverse avec des harmoniques très désagréables. On le verra dans une capture d'écran.

La sortie du CA/N ΣΔ Q(7:0) va à l'entrée de l'octet haut d'un CN/A à 16 bits en complément à deux. L'entrée de l'octet de poids faible n'est pas utilisée ici. Mais elle est câblée sur le PLD en vue d'une plus grande largeur de mot du CA/N ΣΔ. La tension d'alimentation

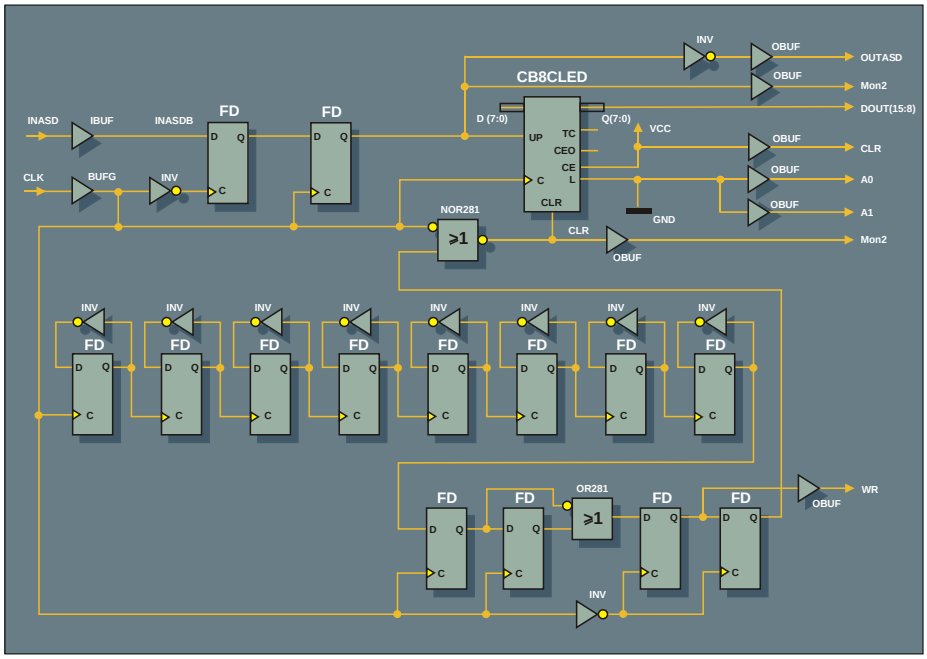


Figure 8. L'assemblage des 26 macrocellules de PLD pour le CA/N ΣΔ.

symétrique du CN/A de ±12 V permet une plage de sortie analogique de près de ±10 V. Le taux de conversion maximal de 100 kHz est presque atteint avec f_{SO} = 12 MHz / 128 = 93,75 kHz.

Concevoir le compteur comme un compteur de transfert pur de 8 bits et utiliser la sortie ΣΔM comme entrée d'activation donnerait un décalage binaire au lieu de la représentation en complément à deux. Cependant, le diviseur correspondant devrait alors être réglé sur 2^q.

La **figure 7** montre le dispositif de test. On y voit aussi des fonctions telles que l'alimentation, la production d'horloge ou la conversion N/A à 16 bits, qui n'appartiennent pas nécessairement au CA/N du ΣΔ. À la **figure 8**, on peut voir le circuit interne du PLD XC95144 pour le CA/N du ΣΔ. Il ne requiert que 26 des 144 macrocellules. Avec les trois composants externes R1, R2 et C1, il constitue le CA/N complet à 8 bits de résolution du ΣΔ. Toutefois, les données de sortie ne sont valables qu'au moment du front positif de /WR. Autrement, vous devez ajouter un registre D de 8 bits déclenché par ce front.

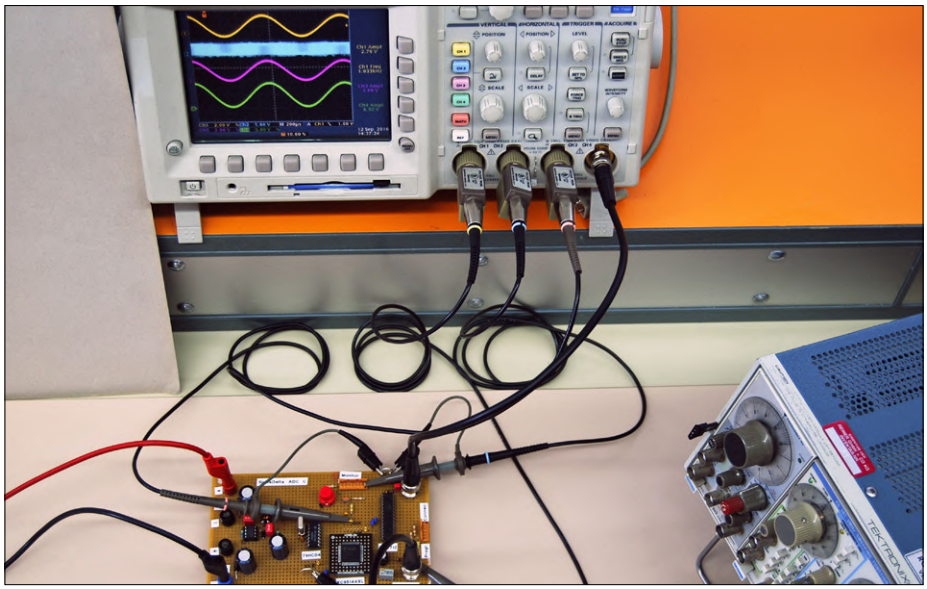


Figure 9. Le banc d'essai pour les mesures dans le domaine temporel.

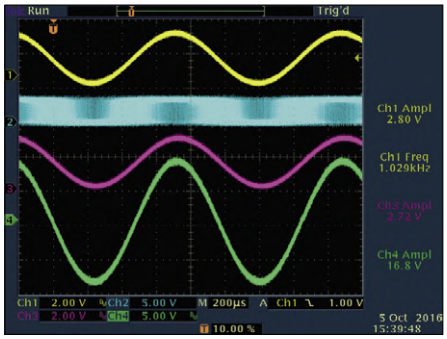


Figure 10. Les signaux sur banc d'essai en domaine temporel à la limite de la surmodulation. Ch1 : entrée analogique, Ch2 : Mon1, Ch3 : ANA et Ch4 : sortie analogique.

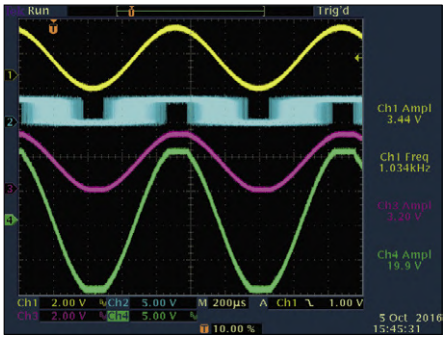


Figure 11. Signaux en surmodulation dans le domaine temporel. Ch1 : entrée analogique, Ch2 : Mon1, Ch3 : ANA et Ch4 : sortie analogique.

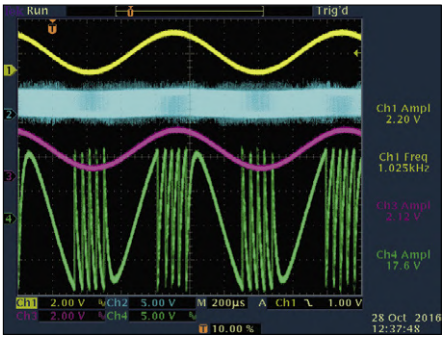


Figure 12. Signaux du banc d'essai en domaine temporel avec une longueur trop courte du mot pour le filtre numérique. Ch1 : entrée analogique, Ch2 : Mon1, Ch3 : ANA et Ch4 : sortie analogique.

Mesures dans le domaine temporel

Pour savoir ce qui se passe dans ce CA/N en PLD et quelles sont ses caractéristiques, on a effectué des mesures au banc d'essai. La **figure 9** montre la configuration du test pour le domaine temporel.

À la **figure 10,** on voit les signaux à l'entrée et aux sorties du CA/N du ΣΔ à une fréquence de signal de 1 kHz. Le signal d'entrée sinusoïdal (Ch1, jaune) de 2,8 V_{cc} est proche du maximum. Sur la sortie numérique (Ch2, bleu), voyez la variation de la densité d'impulsions proportionnelle à la tension analogique. À basse tension d'entrée, la sortie tend à être basse et elle est haute à plus haute tension d'entrée.

Dans le signal de sortie ANA (Ch3, rose), on voit que le signal d'entrée est représenté presque exactement dans la sortie numérique du ΣΔM. La légère atténuation de l'amplitude doit être due au passe-bas RC passif. Le signal de sortie analogique du CN/A (Ch4, vert) a une amplitude d'environ ±8,4 V, une modulation presque totale. Sa symétrie prouve que le signal d'entrée numérique du CN/A Q(7:0) est présent dans le complément à deux.

La **figure 11** montre le comportement en surcharge du CN/A du ΣΔ. Quand la sortie Mon1 de ΣΔM ne donne plus que des « hauts » ou des « bas » en phases, c'est qu'il y a surcharge. Du coup, la sortie analogique du CN/A reste également à sa valeur maximale ou minimale lors des pointes, mais il n'y a pas de dépassement du complément à deux. Le signal de sortie du CA/N, cadré exactement, prouve que la longueur des mots dans le filtre numérique est suffisante.

La **figure 12** montre le comportement du CA/N du ΣΔ quand la largeur du mot du filtre numérique est trop petite. Cela conduit à un débordement du complément à deux et le signal de sortie du DAW passe du maximum au minimum et vice versa avec beaucoup de puissants harmoniques.

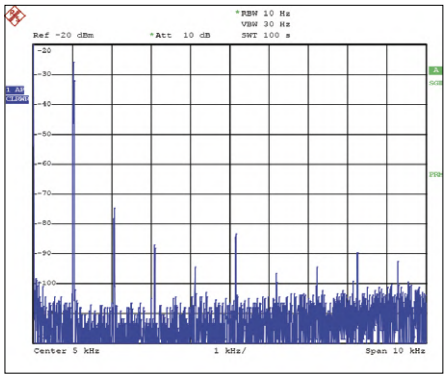


Figure 13. Spectre de la figure 10 avec 10 kHz de bande passante.

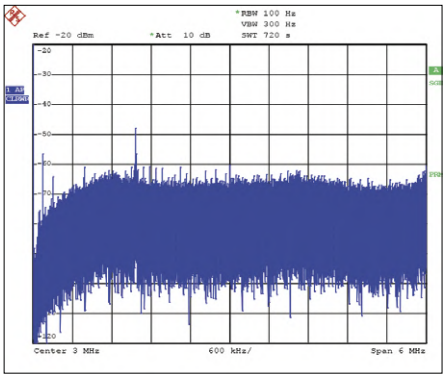


Figure 14. Spectre de la figure 10 avec 6 MHz de bande passante.

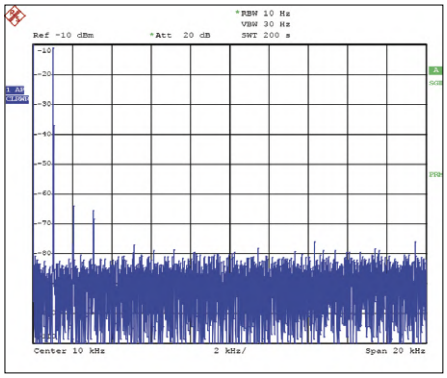


Figure 15. Spectre de la figure 10 avec 20 kHz de bande passante, mais sur la sortie analogique.

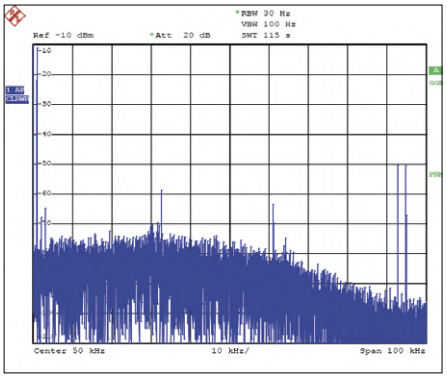


Figure 16. Même spectre qu'à la figure 15, mais avec une largeur de bande de 100 kHz.

Mesures dans le domaine de fréquence

Pour le domaine fréquentiel, le dispositif de mesure a été légèrement modifié par rapport à la figure 9. Un filtre passe-bas LC à la sortie du générateur atténue les harmoniques du signal de test. La sonde à l'entrée d'analyse atténue de 40 dB et adapte à l'impédance de 50 Ω.

La **figure 13** montre le spectre à la sortie numérique Mon1 dans la plage de 0 à 10 kHz avec un signal sinusoïdal de 1 kHz et 2,8 V_{cc} à l'entrée analogique. Le bruit à −100 dBm est inférieur d'environ 75 dB au niveau du signal utile de −25 dBm, en toute conformité avec les valeurs théoriques du S/B. La différence de niveau par rapport aux harmoniques de ≥50 dB indique une bonne linéarité du ΣΔM.

Remarque : une telle mesure avec un analyseur FFT peut produire des erreurs de mesure, car ces analyseurs ont souvent une fréquence d'échantillonnage trop faible par rapport à la plage de fréquence et le bruit est alors replié dans la plage utile aux plus hautes fréquences.

À la **figure 14**, le spectre pris à la sortie numérique Mon1, est maintenant compris entre 0 à 6 MHz. La croissance du bruit avec la fréquence y est manifeste. À partir de 1,2 MHz, le niveau de bruit reste voisin de −60 dBm. L'écart du signal utile ne représente encore que 35 dB.

La **figure 15** montre le spectre dans la gamme de 0 à 20 kHz à la sortie du CN/A « analog out », également pour un signal de 1 kHz et 2,8 Vcc à l'entrée analogique. Le niveau de bruit à −75 dB se situe environ à 65 dB sous celui du signal utile de −10 dBm.

La **figure 16** donne encore le spectre sur « analog out », mais dans la gamme de 0 à 100 kHz. L'effet du passe-bas numérique y est clairement exposé, puisque, contrairement à la figure 14, le bruit n'augmente pas avec la fréquence, mais diminue. À droite dans l'image, on aperçoit des produits de repliement symétriques de la fréquence d'entrée vers celle d'échantillonnage f_{SO} du CN/A à 93,75 kHz.

L'auteur

Guido Nopper est ingénieur diplômé de l'université technique de Furtwangen (Allemagne) et a travaillé de 1981 à 1984 comme ingénieur de développement en LSI MOS avec une spécialisation en convertisseurs A/N et filtres numériques. Depuis 1984, il est responsable de l'ensemble de la conception matérielle des machines de bureau pour le traitement du papier, en particulier des imprimantes de chèques et des scanneurs.

En conclusion

Ceux qui s'intéressent aux PLD, tentés par de nouvelles expériences, ont pu voir dans cet article comment réaliser des circuits relativement complexes et qui fonctionnent bien avec aussi peu de composants externes. Bien sûr, l'idée n'est pas de reconstruire le circuit tel quel, mais par exemple de s'en inspirer pour réaliser un CA/N avec une meilleure résolution (et une bande passante plus étroite). Je me suis limité à l'exposé des maths nécessaires à la compréhension, mais elles devraient vous suffire pour concevoir des circuits centrés sur vos aspirations propres. ⏮

(170566-03 – version française : Robert Grignard)

Liens

[1] 'The Evolution of Oversampling Analog-to-Digital Converters' (PDF) : https://bit.ly/2Z1q68G

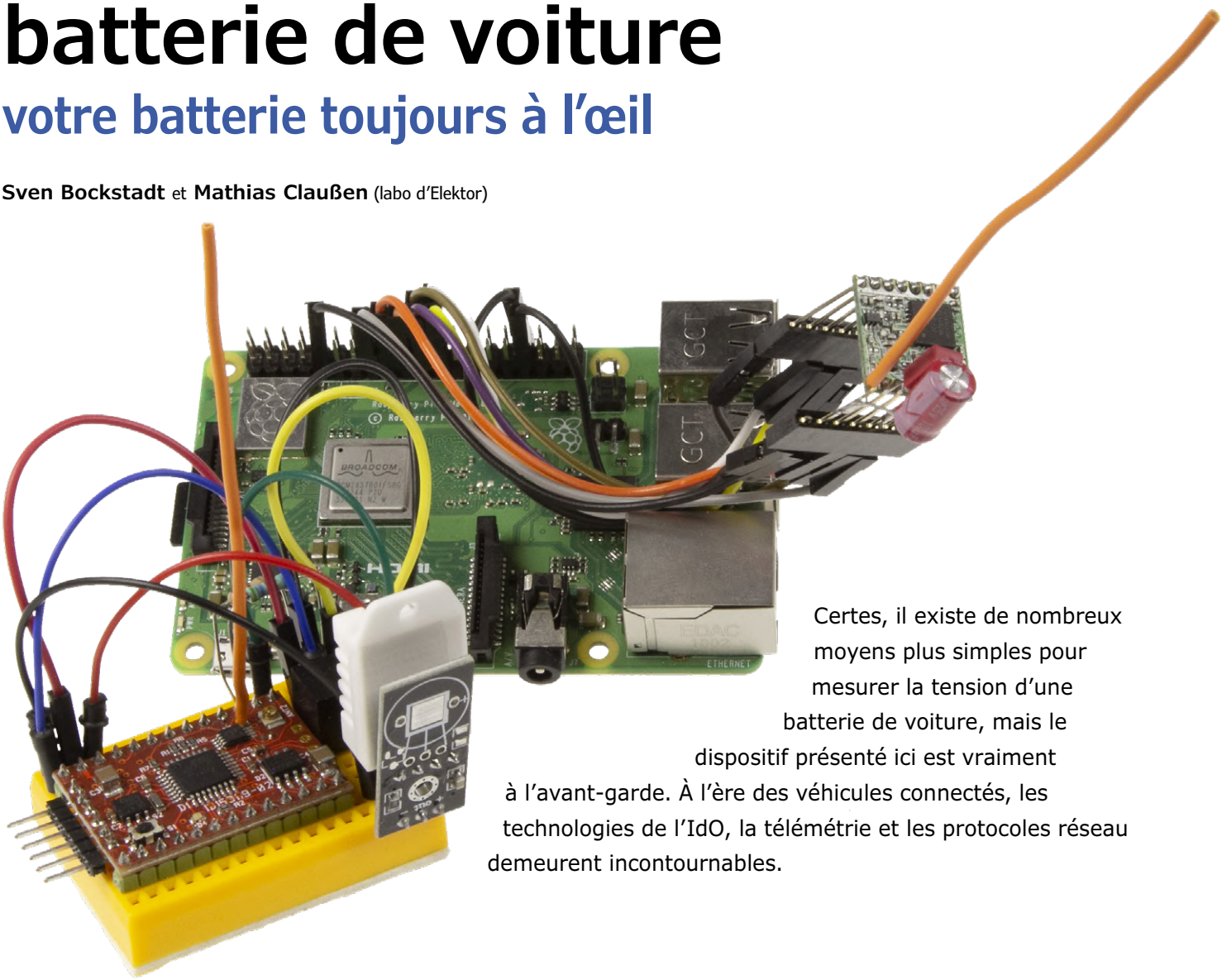
[2] 'Do Multirate Systems Have Transfer Functions?' : www.dsprelated.com/showarticle/143.php

[3] Fiche technique du CPLD XC95144XL : https://bit.ly/2XvhtTy

voltmètre sans fil pour batterie de voiture

votre batterie toujours à l'œil

Sven Bockstadt et **Mathias Claußen** (labo d'Elektor)



Certes, il existe de nombreux moyens plus simples pour mesurer la tension d'une batterie de voiture, mais le dispositif présenté ici est vraiment à l'avant-garde. À l'ère des véhicules connectés, les technologies de l'IdO, la télémétrie et les protocoles réseau demeurent incontournables.

Bien que les véhicules modernes disposent de systèmes électroniques de charge de plus en plus sophistiqués, cela n'implique pas que la batterie reste toujours chargée à 100%. Les mécanismes récents de marche/arrêt des véhicules et la réglementation en matière d'émissions font que la batterie n'est généralement chargée que lorsqu'il y a accélération. En outre, pendant la saison froide, il peut y avoir des défaillances spontanées ou des décharges profondes liées à la température.

Afin d'avoir toujours une idée de l'état de charge de la batterie de sa voiture, l'auteur Sven Bockstadt a mis au point une solution économe en énergie qui effectue de brèves mesures de tension à intervalles réguliers. La valeur de la tension est transmise du point de mesure à l'unité de calcul via une liaison radio à longue portée, selon le standard LoRa ; les valeurs mesurées y sont affichées et enregistrées pour une analyse graphique ultérieure.

Des plates-formes telles que Arduino ou Raspberry Pi proposent une multitude de cartes d'extension adaptées à la transmission radio, ce qui permet le transfert de données sans gros efforts. Afin de ne pas compliquer son projet, l'auteur a utilisé autant que possible des cartes standard pour Arduino. Ainsi, aucune nouvelle carte n'a dû être créée, et le câblage externe a pu être maintenu à sa plus simple expression.

Un logiciel de surveillance a également été développé pour le matériel de l'émetteur et du récepteur, afin de pouvoir piloter ceux-ci par une liaison série, mais aussi à des fins de débogage. La fonction d'enregistrement crée un fichier CSV qui croît en permanence, il peut être sauvegardé sur une carte micro-SD, puis ouvert sur PC avec Excel ou tout logi-

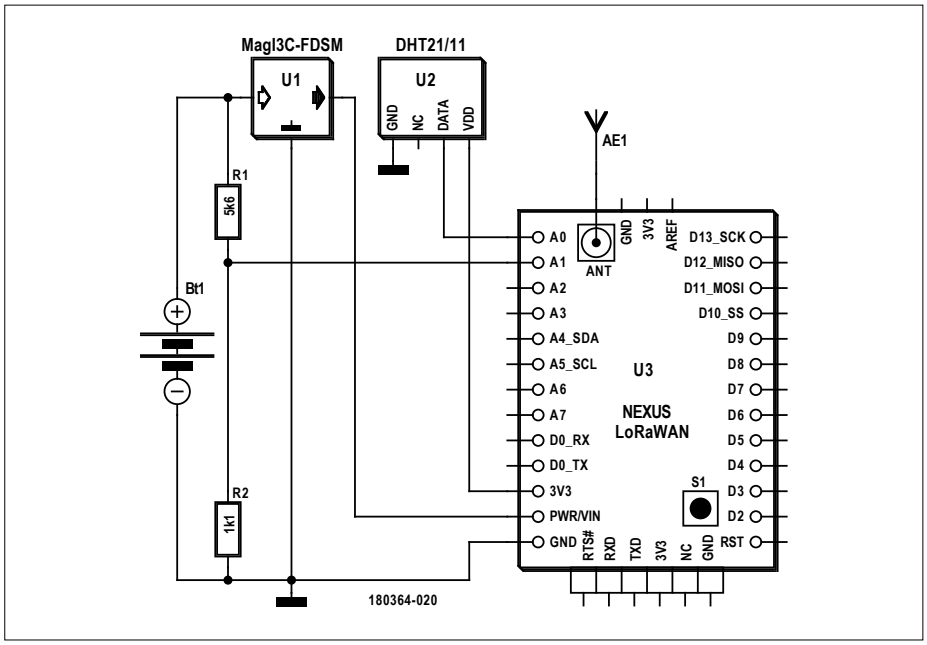
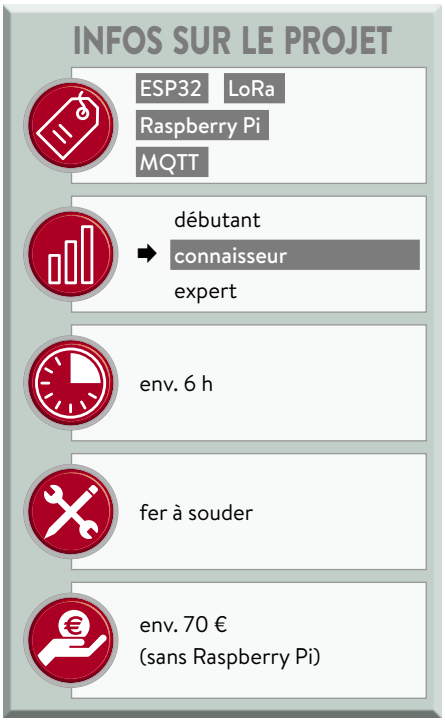


Figure 1. L'émetteur est composé d'un diviseur de tension et du module LoRa. Le capteur météo en option DHT21/11 utilise l'une des nombreuses entrées analogiques libres.



ciel de calcul équivalent, puis analysé en quelques clics avec des graphiques. L'auteur a présenté ce moniteur de tension automobile sans fil il y a quelque temps déjà sur la plate-forme du labo d'Elektor [1]. Pour cette version, matériel et logiciel ont été passés à la moulinette du labo d'Elektor, avec à la clé quelques modifications, améliorations et simplifications. Vous en trouverez le résultat ci-dessous.

L'émetteur

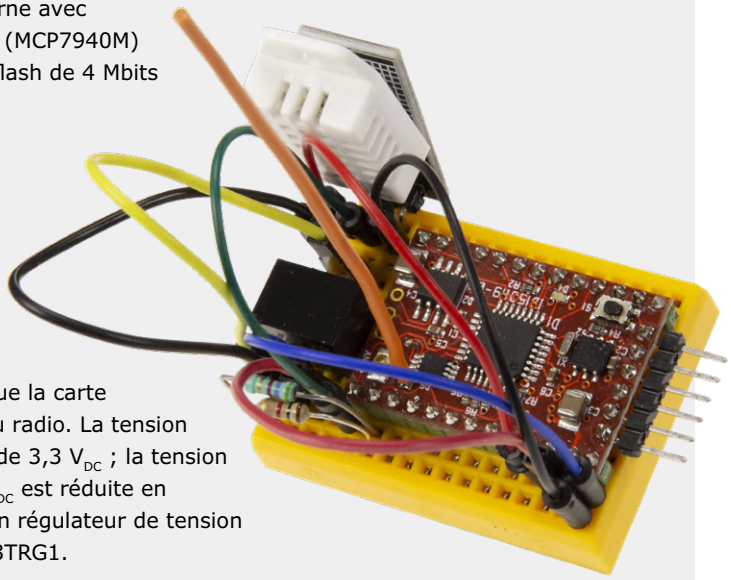
À l'origine, l'auteur utilisait comme émetteur une carte Adafruit Feather M0 comprenant un module radio LoRa RFM96 à 433 MHz de HopeRF [2]. Mais cette carte nécessitait un convertisseur analogique/numérique externe pour la fonction d'enregistrement, ainsi qu'un module d'horloge externe pour la synchronisation du trafic des données en basse énergie. Cela devenait trop coûteux, nous avons préféré remplacer la Feather M0 par une carte LoRa Nexus compatible Arduino ([3], voir encadré), qui non seulement inclut un module RFM95 LoRa à 868 MHz [2], mais qui est pilotée par un microcontrôleur ATmega328P, lequel est doté d'un CA/N interne avec une résolution de 10 bits. La **figure** 1 montre qu'il suffit de connecter un simple diviseur de tension avec R1 = 5,6 kΩ et R2 = 1,1 kΩ à l'entrée de conversion A1 du contrôleur pour adapter la tension de la batterie à la plage du CA/N. La carte LoRa Nexus possède une mémoire flash (W25X40CLSNIG) et une horloge en temps réel, éliminant ainsi le besoin de composants externes. La synchronisation du transfert de données est prise en charge par le micrologiciel du contrôleur.

Pour alimenter le module, on utilise un convertisseur continu-continu à tension fixe de Würth [4], qui produit une tension de sortie stabilisée de +5 V à un courant de 1 A, pour une tension d'entrée de 8 à 28 V. Si ces données vous rappellent le bon vieux 7805, sachez qu'en effet, Würth présente le petit convertisseur à découpage de cette série comme rempla-

Carte LoRa Nexus

La carte LoRa Nexus de 23×33 mm² s'appuie sur le µC à 8 bits ATmega328P d'Atmel, le même type de contrôleur que l'Arduino Nano. Ajoutez à cela un module radio RFM95W LoRa inclus sur la carte. Le contrôleur est équipé d'une horloge en temps réel externe avec 64 octets de SRAM (MCP7940M) et d'une mémoire flash de 4 Mbits (W25X40CL).

Le circuit intégré DS2401P est une particularité : il ne contient qu'un numéro de série unique de 48 bits. Il permet d'identifier de manière univoque la carte au sein d'un réseau radio. La tension d'alimentation est de 3,3 V_{DC} ; la tension d'entrée V_{raw} = 5 V_{DC} est réduite en conséquence par un régulateur de tension LDO AZ1117CR-3.3TRG1.



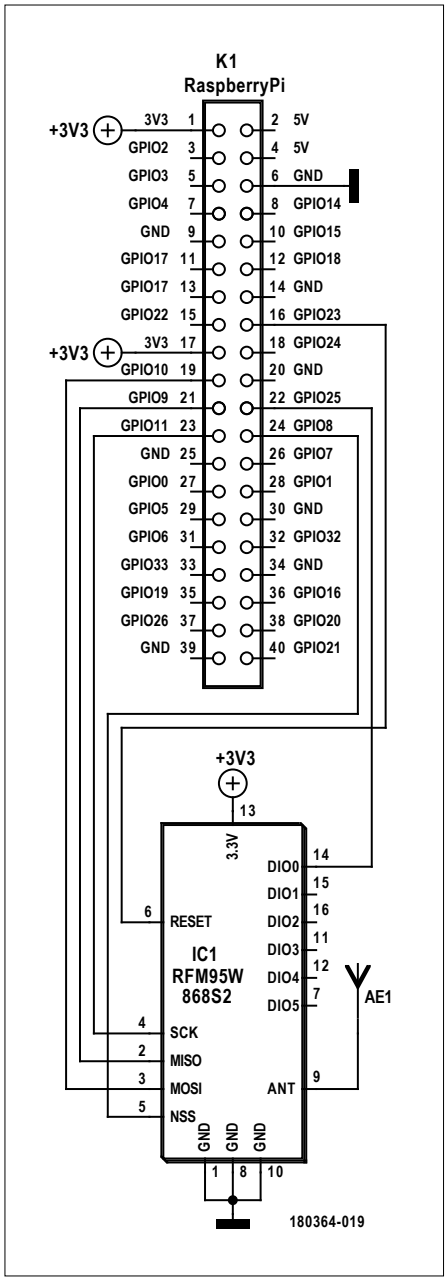


Figure 2. Le circuit se réduit à un récepteur RFM95 connecté au port GPIO du Raspberry Pi.

Tableau 1. Liaisons entre RFM95 et RPi

RFM95	Raspberry Pi
SCK	GPIO11
MISO	GPIO09
MOSI	GPIO10
NSS	GPIO08
RESET	GOIO22
DIO0	GPIO25

çant direct du L78xx. Il n'est même pas nécessaire de rajouter de condensateurs à l'entrée ou à la sortie du convertisseur. La tension de sortie de +5 V du régulateur à découpage est réduite à 3,3 V sur la carte par un autre régulateur de tension (linéaire) qui alimente tous les circuits intégrés du module. La tension délivrée par ce régulateur peut aussi servir de tension de bus I2C (ce n'est pas le cas ici), et simultanément de tension de référence pour le CA/N. La tension de 3,3 V du convertisseur intégré alimente également un « capteur météo » DHT21/11, qui mesure humidité et température. Le logiciel évalue ces données, et ainsi peut mieux estimer les paramètres à attendre de la batterie. Le capteur est connecté à l'entrée de CA/N A0.

Ces quelques composants, rassemblés sur une carte de test sont programmés à l'aide d'un câble FTDI approprié (VCC = 3,3 V) relié au connecteur FTDI de la carte Nexus.

Le récepteur

Initialement, le récepteur était lui aussi conçu autour d'une carte Feather M0, associée à un CA/N / enregistreur externe, un module d'horloge et une alimentation.

Après passage par le labo d'Elektor, ne subsiste que le module RFM95 sous la forme d'un modem LoRa [5] connecté à quelques broches GPIO d'un Raspberry Pi (voir la **figure** 2 et le **tableau** 1). L'unité de calcul repose sur un Raspberry Pi 3+ (au lieu d'un contrôleur Arduino), lequel offre bien plus de possibilités de traitement de données qu'un Arduino. Le choix s'est aussi porté sur le RPi parce qu'il est facile de lui ajouter un écran. En outre le RPi délivre une tension d'alimentation correcte de 3,3 V pour le module radio.

Les deux modules doivent être équipés d'antennes. Deux fils de 86 mm de long chacun suffisent amplement. Voilà pour le matériel, le reste est maintenant confié au logiciel.

Liaison par radio LoRa-PHY

La liaison radio s'appuie sur la norme LoRa, pour laquelle seule la couche physique (LoRa RF / PHY) est utilisée. Il s'agit principalement de la modulation à étalement de spectre de type CSS *(chirp spread spectr*um) à la fréquence de transmission dans la bande ISM ou SRD.

LoRa ne définit pas de protocoles plus complexes. Seul un en-tête avec l'adresse de l'expéditeur est spécifié. Vous pouvez définir une adresse du récepteur, mais ce n'est pas obligatoire : le récepteur (comme dans notre application) peut être configuré pour recevoir tous les paquets de données sur une porteuse à 869,5 MHz, indépendamment de l'en-tête d'adressage.

Il vous suffit de spécifier le nombre d'octets que vous souhaitez envoyer, la transmission est alors effectuée par la puce Semtech et ne requiert pas d'autres actions. Pour des raisons d'immunité aux interférences, les modules LoRa rajoutent un préambule que vous pouvez définir. Le récepteur est alors configuré selon ce même préambule. Si un paquet et son préambule arrive au récepteur, alors il délivre les octets de la charge utile.

Côté RPi, les paramètres sont configurés à l'aide d'une petite application basée sur la bibliothèque Radiohead [6]. Bien entendu, on choisit les mêmes paramètres par défaut pour l'émetteur.

Le logiciel du RPi lit les paquets de données LoRa reçus, et recherche un message avec 0x20 comme expéditeur, avec une longueur de charge utile de 12 octets, et avec 0x10 comme destinataire. Si un tel message est trouvé, alors un paquet de données utiles de structure similaire à celle de la **figure** 3 est attendu. L'identifiant des données est suivi successivement de la valeur de

LISTE DES COMPOSANTS

R = 1 kΩ, 5%, 0,25 W, 250 V
R = 5,6 kΩ, 5%, 0,25 W, 250 V
MOD1 = régulateur abaisseur fixe MagI³C-FDSM
MOD2 = DHT11 du kit ESP32 d'Elektor
MOD3 = carte LoRa Nexus
MOD4 = module émetteur-récepteur RFM95 Ultra LoRa (868 MHz)
Kit de démarrage du Raspberry Pi 3B+ + compilation RPi gratuite
J1,J2 = cavaliers du kit ESP32 d'Elektor
Grande plaque d'essais (optionnel)
Passerelle USB/série BOB-FT232R (Elektor, réf. 110553-91) (optionnel)

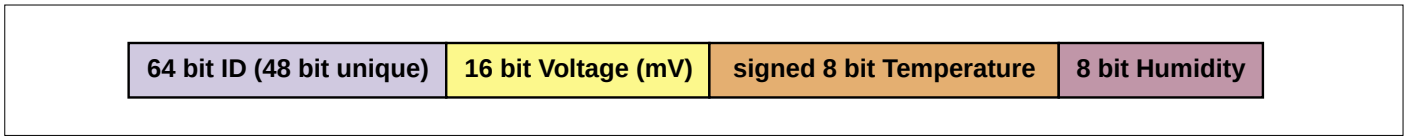


Figure 3. Le protocole simple de données utiles comprend un identifiant de 64 bits suivi de trois valeurs de mesure.

la tension (16 bits), de la température (8 bits) et de l'humidité de l'air (8 bits). L'ID à 64 bits se compose de l'adresse unique sur 48 bits spécifiée par le circuit d'authentification DS2401P sur la carte LoRa Nexus (généralement marquée sur un autocollant situé sur la carte, mais vous pouvez également la lire avec un Arduino), et deux octets, toujours identiques, pour l'application « Car-Voltmeter ».

MQTT et un courtier de données baptisé Mosquitto

Le reste du logiciel traite les nouvelles données entrantes et les envoie à un courtier MQTT à l'aide de la bibliothèque *libmosquitto*. La bibliothèque utilise l'ID à 64 bits pour décider si le message appartient à son propre émetteur. MQTT (voir **encad**ré) est un protocole simple et efficace pour échanger des données entre différents périphériques (IdO).

Avec MQTT, vous avez toujours besoin d'un serveur appelé courtier *(broke*r). Mosquitto, désormais inclus dans Raspbian, est un courtier MQTT à code source ouvert, multiplateforme, bien connu. Le courtier reçoit les messages MQTT de la source de données et les envoie aux récepteurs « abonnés » à ces messages. Dans ce cas, le récepteur LoRa publie les paquets LoRa reçus sur différents *topic*s MQTT « voiture/batterie/tension », « voiture/intérieur/température » et « voiture/intérieur/humidité ». Chaque valeur peut ainsi être traitée séparément. Les informations destinées au courtier MQTT sont écrites au format JSON, ce qui permet un traitement facile et souple de ces trois éléments.

Il est possible de se servir du langage de programmation graphique Node-RED, spécialisé dans les applications IdO, pour créer un logiciel lui-même utilisé pour s'abonner à des *topic*s MQTT. Vous pouvez ainsi réaliser les choses les plus extravagantes à partir des messages reçus, telles que piloter votre domicile, envoyer des e-mails ou même afficher les données sous forme d'élégants graphiques **(figure** 4). Il est très facile de développer une telle application avec Node-RED [8].

L'installation en pratique

L'installation de Node-RED et de MQTT sur un RPi a déjà été décrite dans l'article [8]. Lorsque l'installation de Node-RED et *mosquitto* est terminée, il reste encore quelques préparatifs pour compiler le code permettant au logiciel de fonctionner sur le RPi. Il suffit de saisir quelques commandes dans le terminal pour installer certaines dépendances du logiciel.

```
apt-get install libmosquittopp-dev
apt-get install libmosquittopp1
apt-get install libmosquitto1
```

Par ailleurs, il faut installer la bibliothèque `bcm2835` [9] sur le RPi. Extrayez le fichier et rendez-vous dans le répertoire. Puis exécutez `./configure` pour préparer le code en vue de la compilation. Une fois que c'est fait, et sans message d'erreur, vous pouvez saisir :

```
make
sudo make check
sudo make install
```

La bibliothèque devrait s'installer. Lorsque tout est fait, nous sommes prêts à compiler le code pour le récepteur. Récupérez le code de Git [10] et copiez-le dans le RPi. Allez dans le répertoire et lancez `make`. Après cela, le récepteur est prêt à l'emploi (espérons-le !). Vous devez malheureusement exécuter le programme avec les privilèges `root` pour

MQTT et Mosquitto

Selon le protocole MQTT (*Message Queuing Telemetry Transport*), les données et commandes échangées entre périphériques (IdO) ne le sont pas directement, mais transitent par un serveur MQTT central (courtier). MQTT est efficace, sécurisé et peu exigeant en ressources du µC. Le protocole est orienté message : une source envoie (publie) ses données uniquement au courtier. Toutefois, un client récepteur n'a pas à interroger constamment le serveur sur l'existence de nouvelles données provenant de l'émetteur, mais peut « s'abonner » (souscrire) aux messages de l'émetteur, de sorte que le courtier transfère automatiquement les nouvelles données de l'émetteur abonné au client. Émetteur et récepteur de messages sont donc complètement découplés par le courtier. Ainsi, tout émetteur de données n'a pas à se soucier de savoir qui recevra ces données.

Un message comprend entre autres les éléments suivants :

- *Topic* est le sujet du message. Les sujets sont de simples chaînes de caractères séparées par des barres obliques. Un sujet tel que **voiture/batterie/tension** contient une hiérarchie d'objets, à savoir l'émetteur dans la voiture ; il s'agit de la batterie, et nous nous intéressons à sa tension.
- *Payload* est le contenu utile du message, généralement des commandes ou données.

Pour utiliser MQTT (versions 3.1, 3.1.1 et 5.0), vous avez besoin d'un courtier tel que Eclipse Mosquitto [17] bien connu. Ce courtier à code source ouvert et léger peut être installé sur n'importe quel ordinateur monocarte à basse consommation, y compris un Raspberry Pi (absolument sans problème via le dépôt (*repository*) principal).

Source : FHEM [7].

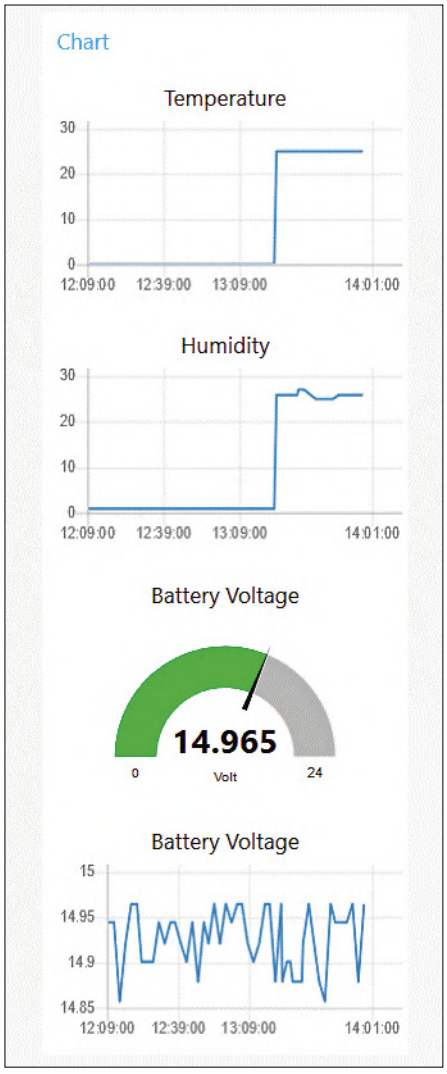


Figure 4. L'élégance de l'affichage des mesures.

accéder à certains sous-systèmes du RPi. Donc, `./sudo main` lance le logiciel. Vous pouvez utiliser `sqlite3` comme moteur de base de données pour enregistrer les données. Installez-le avec :

```
apt-get install sqlite3
```

La base de données SQLite permet d'enregistrer toutes les informations fournies par le « Car-Voltmeter ». Ensuite, nous allons dans le dossier personnel de l'utilisateur RPi et créons une base de données. Ouvrez le terminal et saisissez :

```
cd ~
sqlite3 carsensor.db
```

Ceci crée une nouvelle base de données à remplir avec trois tableaux :

```
sqlite> CREATE TABLE battery_
   voltage ( id INTEGER PRIMARY
   KEY AUTOINCREMENT, uuid
   NUMERIC, voltage NUMERIC,
   timestamp DATETIME DEFAULT
   CURRENT_TIMESTAMP);
sqlite> CREATE TABLE humidity
   ( id INTEGER PRIMARY
   KEY AUTOINCREMENT, uuid
   NUMERIC, humidity NUMERIC,
   timestamp DATETIME DEFAULT
   CURRENT_TIMESTAMP);
sqlite> CREATE TABLE temperature
   ( id INTEGER PRIMARY KEY
   AUTOINCREMENT, uuid NUMERIC,
   temperature NUMERIC,
   timestamp DATETIME DEFAULT
   CURRENT_TIMESTAMP);
sqlite> COMMIT;
sqlite> .exit
```

Les tableaux requis sont créés. À présent, reste à installer Node-RED :

```
cd ~/.node-red
npm i --unsafe-perm
   node-red-node-sqlite
```

Ensuite, redémarrez le RPi. Presque tous les composants fonctionnent désormais.

Après cela, vous devez importer le code de traitement [10] dans Node-RED, c'est pareil qu'avec l'horloge à LED géantes d'Elektor de mai/juin 2019.

Une dernière chose à faire côté récepteur : démarrer le logiciel compilé. Allez dans `~/LoRA_MQTT_VoltMeter` et démarrez le programme compilé avec `sudo ./main`. Toutefois ce n'est pas indispensable dans un premier temps, vous pourrez déplacer le logiciel plus tard dans la zone de démarrage automatique du RPi.

Il est temps maintenant de s'occuper de la partie **émetteur** avec la carte LoRa Nexus compatible Arduino [3]. Le code a pour tâche d'initialiser le matériel et de produire un jeu de nouvelles mesures toutes les 120 s. Toutefois il faut configurer quelques bibliothèques : une version corrigée de la bibliothèque `DS2401`, que vous trouverez en téléchargement dans GitHub [11], et une bibliothèque basée sur la bibliothèque `OneWire` de Paul Stoffregen [12]. En vue de pouvoir économiser de l'énergie, la bibliothèque `LowPower` [13] doit également être installée. Comme nous voulons aussi utiliser les capteurs d'humidité et de température, nous avons besoin de la bibliothèque `Adafruit DHT` [14]. Enfin pour le

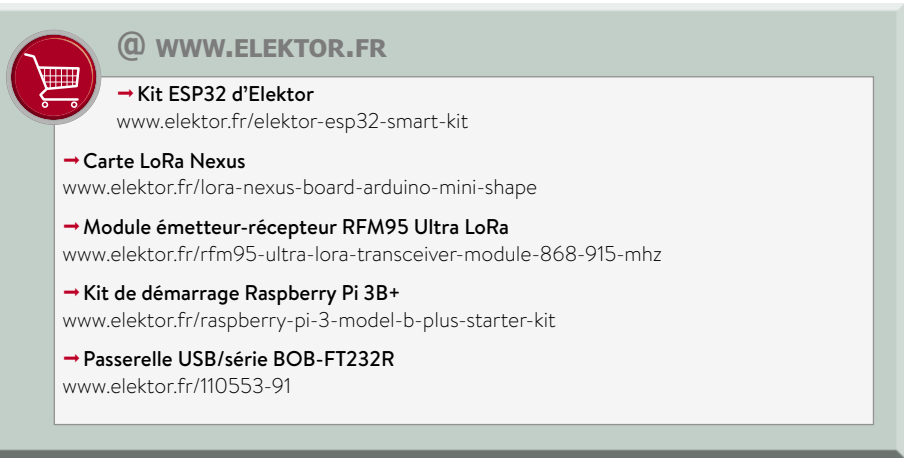
LoRa-PHY

La norme de communication LoRa (*Long Range*) est spécialement conçue pour envoyer des données de capteurs IdO, à des débits lents inférieurs à 50 kbit/s, mais à haute efficacité énergétique, à un récepteur, et ce sur de longues distances (10 km et plus, en fonction des conditions aux limites). LoRa est particulièrement adapté aux transmissions en temps non réel, où le délai importe moins. Les systèmes LoRa émettent en Europe dans la bande ISM (433,05 à 434,79 MHz) ou (comme ici) dans la bande SRD (863 à 870 MHz).

LoRa est la couche physique la plus basse du modèle OSI ; elle peut être utilisée par des couches supérieures telles que LoRaWAN (*Long Range Wide Area Network*) entre autres. Contrairement aux systèmes sans fil traditionnels basés sur la modulation FSK, LoRa utilise un type de modulation à étalement de spectre (CSS) inventé par Semtech. En faisant varier le facteur d'étalement, LoRa peut ajuster le débit de données selon la sensibilité à une largeur de bande passante d'un canal donné et atteindre ainsi une grande distance avec un bilan de transmission de puissance extrêmement élevé d'environ 155 à 170 dB. En outre, LoRa est moins sensible aux interférences grâce au codage avec correction d'erreur directe (*Forward Error Correction*).

Vous trouverez d'autres informations de base sur LoRa sur le portail des développeurs de la société Semtech [18].

fonctionnement du module LoRa, nous avons besoin de la version corrigée de Radiohead, incluse dans le paquet de téléchargement du projet [10].
Le micrologiciel du récepteur est chargé dans l'EDI Arduino, toutes les bibliothèques mentionnées sont intégrées et le programme est compilé. Un câble FTDI (3,3 V) sert à télécharger le micrologiciel complet sur la carte LoRa Nexus.
Tous les programmes mentionnés ici peuvent être récupérés dans le dépôt GitHub d'Elektor [11].



Prêt pour le montage ?

Un point que notre collègue Mathias Claussen n'a découvert que sur sa paillasse de laboratoire : le RFM95 ne devrait pas être connecté par câble, car cela peut entraîner instabilité et réinitialisations indésirables. Pour cette raison, la bibliothèque *Radiohead* a été légèrement corrigée afin de vérifier la configuration du modem et, le cas échéant, le réinitialiser si quelque chose ne va pas.
L'essai en laboratoire du prototype sur une petite plaque d'essai est donc terminé. Si vous vous demandez maintenant si le voltmètre est prêt à être utilisé dans un véhicule, la réponse est clairement : **non** ! Un circuit de protection approprié du matériel est indispensable lorsqu'on utilise de l'électronique dans une voiture. Il élimine les signaux d'interférence du système électrique et protège le circuit électronique.
L'article « Sources d'interférences en électronique automobile » [15] présente sans détour tous les problèmes auxquels vous vous exposez si vous tentez d'intégrer vos propres composants dans l'électronique de votre voiture. Mais les mesures à prendre pour protéger ce voltmètre des interférences dépendent (trop) du véhicule et des conditions d'installation, donc pas de recette miracle ici. ⏮

(180364-03 – version française : Xavier Pfaff)

Liens

[1] Projet dans le labo d'Elektor : www.elektormagazine.fr/labs/wireless-car-voltmeter

[2] Module LoRa : www.hoperf.com/modules/lora/index.html

[3] Carte LoRa Nexus : www.elektor.fr/lora-nexus-board-arduino-mini-shape

[4] Module DC/DC Würth 173010578 : https://katalog.we-online.de/en/pm/MAGIC_FDSM_FIXED_OUTPUT_VOLTAGE

[5] RMF95 en modem LoRa : www.elektor.fr/rfm95-ultra-lora-transceiver-module-868-915-mhz

[6] Bibliothèque Radiohead : https://github.com/hallard/RadioHead

[7] Présentation de MQTT : https://fr.wikipedia.org/wiki/MQTT

[8] « Horloge à LED géante avec Wi-Fi et mesures météo », Elektor 05-06/2019 : www.elektormagazine.fr/180254-04

[9] BCM2835 : www.airspayce.com/mikem/bcm2835/

[10] Dépôt GitHub du projet : https://github.com/ElektorLabs/180364-wireless-car-multimeter

[11] Bibliothèque DS2401 : https://github.com/sindrehal/Arduino_DS2401

[12] Bibliothèque OneWire : https://github.com/PaulStoffregen/OneWire

[13] Bibliothèque LowPower : https://github.com/rocketscream/Low-Power

[14] Bibliothèque du capteur DHT : https://github.com/adafruit/DHT-sensor-library

[15] « Sources d'interférences dans l'électronique automobile », Elektor 03-04/2019 : www.elektormagazine.fr/180345-04

[16] Mosquitto: http://mosquitto.org/

[17] Portail des développeurs LoRa : https://lora-developers.semtech.com/

Un événement oublié ?

Vous organisez une conférence, un salon… ou bien vous participez à un séminaire ou tout autre événement qui aurait sa place ici, partagez cette information avec tous les lecteurs. Envoyez-nous tous les détails à **redaction@elektor.fr**.

septembre 2019

◇ Biomim expo
11/09 - Paris
biomimexpo.wordpress.com/

◇ Nantes Digital Week
12 au 22/09 - Nantes
www.nantesdigitalweek.com/

◇ Semaine européenne de la mobilité
16 au 22/09
www.mobilityweek.eu/

◇ Focus IoT Valley
19/09 - Toulouse
www.focus.iot-valley.fr/

◇ Salon du jeu de café
21 au 22/09 - Pouilly en Auxois
www.pinballpassion.org/

◇ 19e Congrès International de Métrologie
24 au 26/09 - Paris
www.cim2019.com/

◇ 4th SD-WAN Summit
24 au 26/09 - Paris
www.uppersideconferences.com/sd-wan

Forum de l'électronique
Salon de l'innovation et des solutions électroniques

CAP'TRONIC
Systèmes électroniques et logiciel embarqué

◇ Forum de l'électronique
24 au 26/09 - Paris
www.forum-electronique.com/fr
Venez nous rendre visite sur le stand E71 !

◇ Mems & imaging sensors summit
25 au 27/09 - Grenoble
www.semi.org/en/connect/events/mems-imaging-sensors-summit

◇ Salon des véhicules de loisirs
28/09 au 06/10 - Paris
www.salonvdl.com/

◇ European Microwave Week 2019
29/09 au 04/10 - Paris
www.eumweek.com

◇ La Mêlée Numérique
30/09 au 07/10 - Toulouse
www.meleenumerique.com/

octobre 2019

◇ Mobility for business
01 au 03/10 - Paris
www.mobility-for-business.com/

◇ Salon professionnel de la sûreté et de la sécurité
01 AU 03/10 - Paris
www.salon-aps.com/

◇ IBS (Intelligent Buildings Systems)
02 au 03/10 - Paris
www.ibs-event.com/

◇ Smart City + Smart Grid
02 au 03/10 - Paris
www.smartgrid-smartcity.com/

◇ Code Week – Semaine Européenne du Code
05 au 20/10 - partout en Europe
codeweek.eu/

◇ Fête de la science
05 au 13/10 - partout en France
www.fetedelascience.fr/

◇ Paris manga & sci-fi show
05 au 06/10 - Paris
www.parismanga.fr/

◇ Radiomania 2018
06/10 - Combronde
radiomania.pagesperso-orange.fr/

◇ HAMexpo
12/10 - Le Mans
www.ref-union.org

◇ 10e Colloque national éolien
16 au 17/10 - Paris
www.colloque-national-eolien.fr/

◇ Les rendez-vous Carnot
16 au 17/10 - Paris
www.rdv-carnot.com/

◇ Salon du flipper et du jeu vidéo
19 au 20/10 - Sorgues
www.facebook/rpjgame

◇ Batteries event
22 au 24/10 - Nice
www.batteriesevent.com/

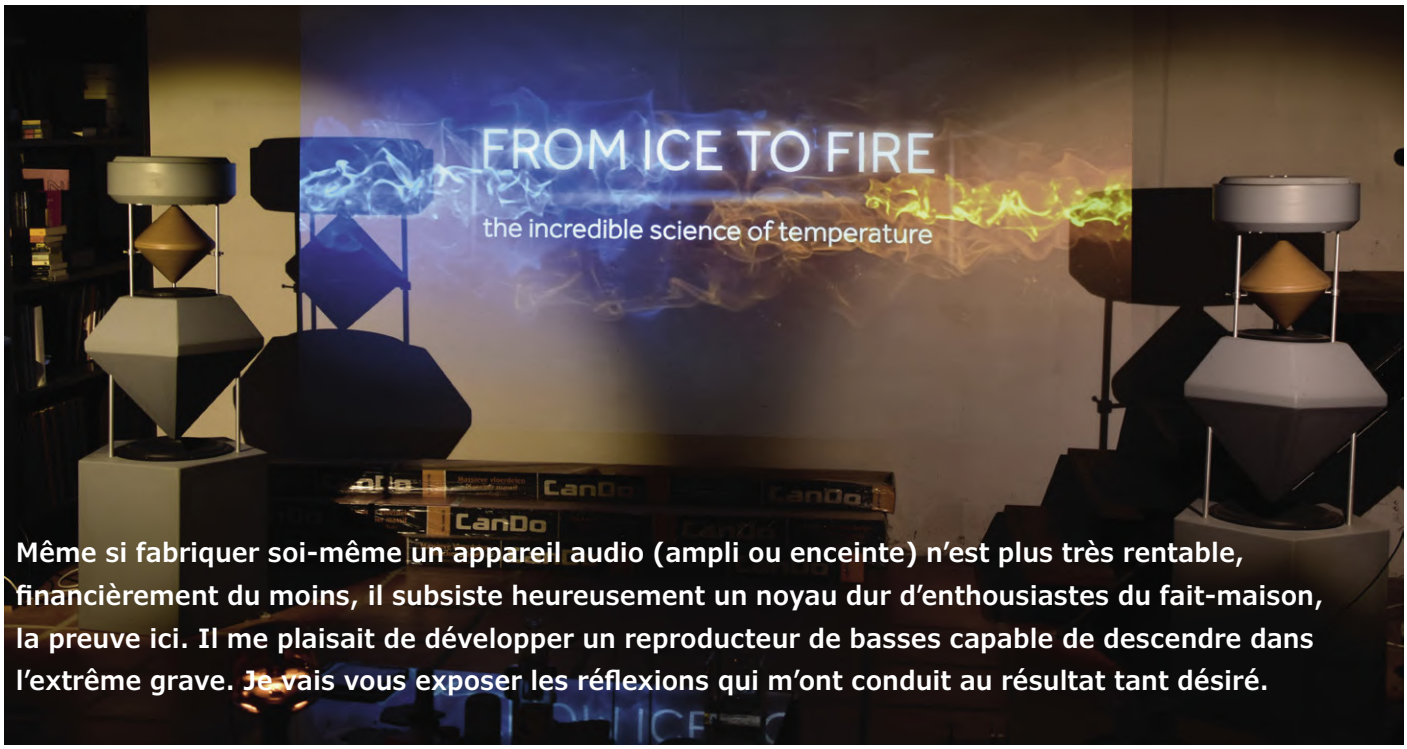
◇ Embedded Linux Conference Europe
28 au 30/10 - Lyon
elinux.org

◇ Paris game week
30/10 au 03/11 - Paris
www.parisgamesweek.com/

PROJET DE LECTEUR

reproducteur de basses de haute ponctualité

jusqu'à l'ultra-grave



Même si fabriquer soi-même un appareil audio (ampli ou enceinte) n'est plus très rentable, financièrement du moins, il subsiste heureusement un noyau dur d'enthousiastes du fait-maison, la preuve ici. Il me plaisait de développer un reproducteur de basses capable de descendre dans l'extrême grave. Je vais vous exposer les réflexions qui m'ont conduit au résultat tant désiré.

Jan Breemer (Pays-Bas)

Ne vous méprenez pas, vous ne trouverez pas dans cet article une solution toute faite à recopier. En revanche, je veux donner au lecteur intéressé les informations suffisantes pour concevoir et réaliser le reproducteur de basses qu'il souhaite.

Position du problème

La plupart des systèmes audio, qu'ils soient du commerce ou fait maison, se fondent sur un amplificateur stéréo ordinaire, donc avec, dans le même boîtier, un canal gauche et un canal droit qui couvrent chacun l'ensemble du spectre audio, suivis de deux groupes de haut-parleurs raccordés par des filtres passifs qui répartissent le spectre audio en deux ou trois bandes accordées aux spécificités des reproducteurs utilisés.

Et c'est avec ces filtres de sélection de voie que les soucis commencent. Dans la plupart des cas, ils altèrent l'amortissement des haut-parleurs. Ils compliquent aussi l'adaptation à la sensibilité propre de chaque haut-parleur. Finalement, les composants mis en jeu ne valent jamais plus que leur prix.

Une autre anicroche provient aussi du recours pour l'enceinte de grave à la structure *basreflex* qui met en relation l'intérieur du baffle avec le monde extérieur par l'intermédiaire d'un tunnel. Ces systèmes reflex ne rendent que peu ou pas du tout les très basses fréquences, sous la trentaine de hertz. Pire encore, sous la fréquence de résonance, ils se comportent comme un filtre passe-haut abrupt (du quatrième ordre). Pareil filtrage occasionne des suroscillations qui nuisent à la réponse impulsionnelle du système.

Finalement, si nous voulons malgré tout entraîner cette enceinte dans les fréquences extrêmement basses, le cône du haut-parleur va bien subir de très grands débattements, mais ils seront inaudibles en raison du court-circuit acoustique. De plus, ces excursions extrêmes déforment les hautes fréquences.

L'approche

L'enceinte totalement close offre de nombreux avantages, raison pour laquelle j'ai choisi cette piste pour mes cogitations. Sous la fréquence de résonance, la pression sonore descend avec une pente moins raide, 12 dB par octave, donc un filtre passe-haut d'ordre deux. En compensation, j'ai mis un filtre spécial

devant l'étage final, sachant que chaque HP, donc celui des graves aussi, a son propre amplificateur de puissance. La particularité de ce filtre réside dans la rectification de la courbe et un affaiblissement de la résonance propre du système. Et ce qui ne gâte rien, c'est qu'il est relativement aisé de modéliser le comportement d'une enceinte close avec une bonne précision.
Pour simplifier la vie des lecteurs intéressés, j'ai développé un logiciel pour ordinateur qui se charge des calculs difficiles et fournit des résultats graphiques. Il est disponible gratuitement au téléchargement sur la page de cet article [1], avec les fichiers exécutables sous Linux et Windows, ainsi que le code source.

Deux messieurs

La conception d'une bonne enceinte acoustique n'est plus aujourd'hui affaire de procédure fastidieuse d'essais, d'adaptation et de nouveaux essais à répétition, et cela grâce à deux messieurs, Neville Thiele et Richard Small. Les paramètres qui portent leurs noms permettent de prédire avec précision le comportement d'un reproducteur de grave, à savoir la combinaison du HP de grave (*woofer*) et de l'enceinte.
Les plus importants de ces fameux paramètres TS sont rassemblés dans le **tableau 1**.
Tous les paramètres TS ne sont pas représentés ici – il y en a bien d'autres, mais leur influence relative sur le résultat final est trop faible pour être prise en considération ici.

Le modèle de calcul

Nous voici face à une kyrielle de paramètres et de caractéristiques du HP et de son enceinte ; pour arriver à calculer nous-mêmes ou plus confortablement à expérimenter avec un simulateur, nous les convertissons en leurs équivalents électriques. C'est ce qui nous permettra après d'utiliser n'importe quel simulateur. *Le* et *Re* sont déjà des grandeurs électriques, elles entrent directement dans le modèle. Pour convertir les grandeurs mécaniques, adoptons une approche logique dans laquelle le courant qui passe dans la bobine mobile est proportionnel à la force exercée sur le cône. Il en découle alors que la masse du cône est équivalente à une capacité, tandis que la force du ressort de suspension et celle de l'air dans l'enceinte entrent dans le modèle sous forme d'inductances.

Tableau 1. Extrait des paramètres TS.

paramètre	description	unité
Mms	masse du cône + de l'air entraîné	kg
Cms	compliance (souplesse) de la suspension du cône	m/N
Rms	résistance mécanique (amortissement) du cône	kg/s
Re	résistance en continu de la bobine mobile	Ω
Le	inductance de la bobine mobile	H
Bl	facteur de puissance de la bobine mobile	N/A ou Tm
Sd	surface réelle du cône	m²
Xmax	course maximale permise du cône	±mm
Cbx	compliance de l'air dans l'enceinte	m/N
Rbx	atténuation du matériau de remplissage (p.ex. laine de roche) du boîtier	kg/s
Qtc	facteur de mérite total du châssis du haut-parleur	–

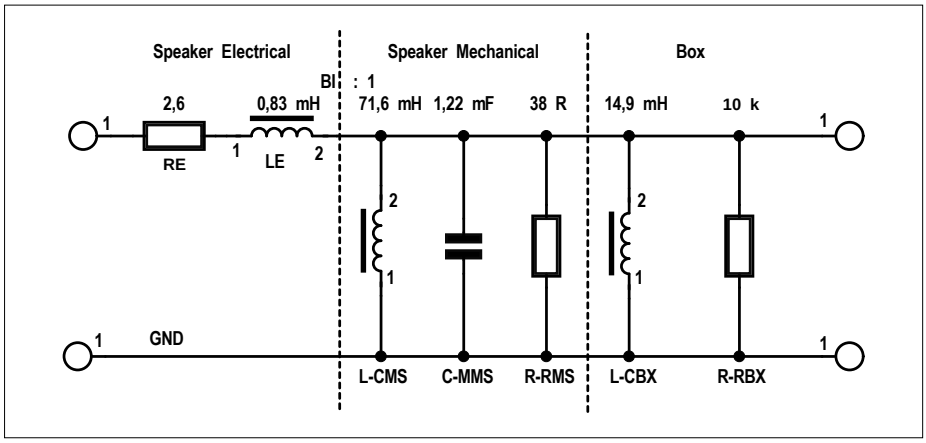


Figure 1. Le modèle mathématique pour la combinaison haut-parleur et enceinte. Les valeurs s'appliquent au ScanSpeak 30W4558T00 dans une enceinte close d'un volume de 41 l.

On peut s'imaginer le facteur de conversion comme un transformateur idéal dont le rapport du nombre de spires vaut *Bl*:1. Avec ce transformateur, la force mécanique se traduit en courant électrique et la vitesse en tension électrique et inversement, bien entendu. Et tout comme sur un vrai transfo, dans la conversion des impédances, le paramètre *Bl* apparaît au carré.

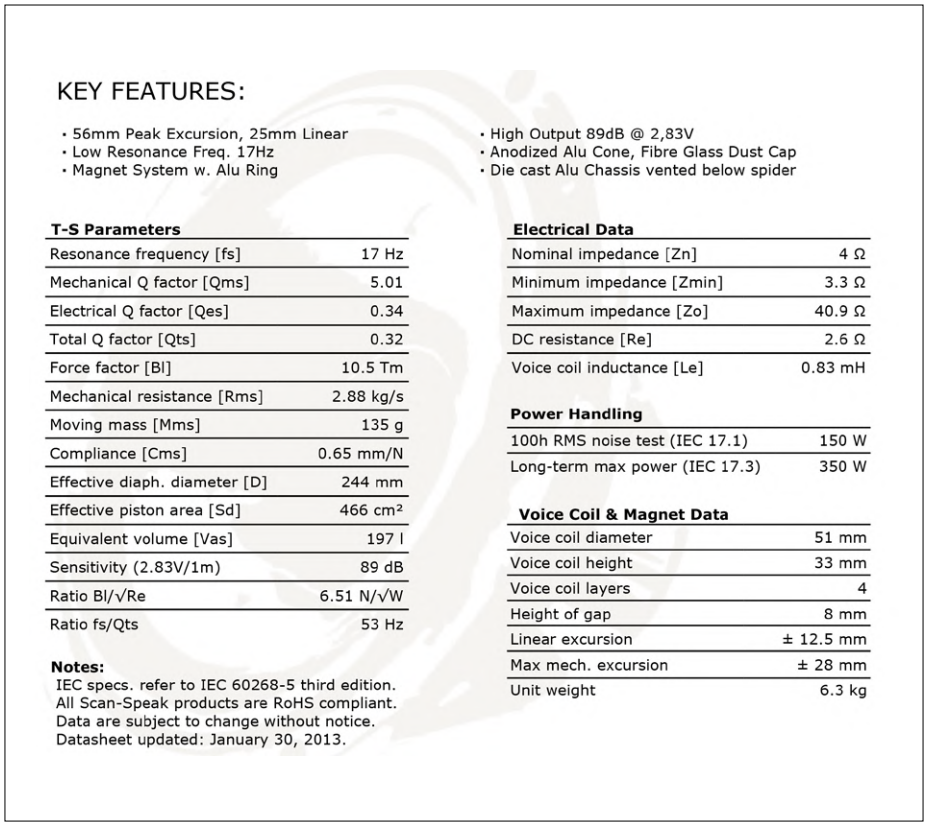
La **figure 1** représente le modèle de calcul sous la forme d'un schéma électrique équivalent.
En résumé, les conversions sont les suivantes :

- L-CMS = Cms * Bl^2
- C-MMS = Mms / Bl^2
- R-RMS = Bl^2 / Rms
- L-CBX = Cbx * Bl^2
- R-RBX = Bl^2 / Rbx

La compliance (mobilité) de l'air dans l'enceinte dépend – cela va de soi – du volume intérieur et du carré de la surface effective du piston, le cône du haut-parleur :
$Cbx = 7{,}14 * 10^{-6} * Vbox / Sd^2$ [m/N]

Passons à la pratique

Nous disposons ainsi de quelques relations théoriques et d'un modèle de calcul électrique, c'est très bien, mais encore insuffisant, que pouvons-nous en faire ? Clairement, où aller chercher les valeurs pertinentes à introduire dans les équations et dans le modèle mathématique ? Et que va-t-il en sortir ?
Commençons par le plus facile : tout fabricant de bons HP qui se respecte a mesuré avec tout son savoir-faire les paramètres TS de ses produits et les met gratuitement à disposition. C'est un haut-parleur ScanSpeak 30W4558T00 que



KEY FEATURES:

- 56mm Peak Excursion, 25mm Linear
- Low Resonance Freq. 17Hz
- Magnet System w. Alu Ring
- High Output 89dB @ 2,83V
- Anodized Alu Cone, Fibre Glass Dust Cap
- Die cast Alu Chassis vented below spider

T-S Parameters	
Resonance frequency [fs]	17 Hz
Mechanical Q factor [Qms]	5.01
Electrical Q factor [Qes]	0.34
Total Q factor [Qts]	0.32
Force factor [Bl]	10.5 Tm
Mechanical resistance [Rms]	2.88 kg/s
Moving mass [Mms]	135 g
Compliance [Cms]	0.65 mm/N
Effective diaph. diameter [D]	244 mm
Effective piston area [Sd]	466 cm²
Equivalent volume [Vas]	197 l
Sensitivity (2.83V/1m)	89 dB
Ratio Bl/√Re	6.51 N/√W
Ratio fs/Qts	53 Hz

Electrical Data	
Nominal impedance [Zn]	4 Ω
Minimum impedance [Zmin]	3.3 Ω
Maximum impedance [Zo]	40.9 Ω
DC resistance [Re]	2.6 Ω
Voice coil inductance [Le]	0.83 mH

Power Handling	
100h RMS noise test (IEC 17.1)	150 W
Long-term max power (IEC 17.3)	350 W

Voice Coil & Magnet Data	
Voice coil diameter	51 mm
Voice coil height	33 mm
Voice coil layers	4
Height of gap	8 mm
Linear excursion	± 12.5 mm
Max mech. excursion	± 28 mm
Unit weight	6.3 kg

Notes:
IEC specs. refer to IEC 60268-5 third edition.
All Scan-Speak products are RoHS compliant.
Data are subject to change without notice.
Datasheet updated: January 30, 2013.

Figure 2. On trouve aisément les paramètres TS des haut-parleurs de fabricants réputés.

j'avais choisi pour mon projet et je l'ai développé avec les équations ci-dessus. Aucune difficulté à dénicher sur l'internet (cf. **fig. 2**) les valeurs nécessaires. Les valeurs de la figure 2 sont également utilisées pour calculer celles des composants donnés à la figure 1.

C'est ici que tout devient chouette : nous allons déterminer le comportement du système de HP, parce que c'était bien le but, savoir comment la tension de sortie de l'amplificateur se traduit en pression sonore (SPL = *Sound Pressure Level*, spécifiée généralement à une distance de 1 m du HP).
À cette fin, nous considérons que la tension électrique à la sortie du modèle mathématique (fig. 1) est proportionnelle à la vitesse de déplacement du cône et personne ne s'étonnera que le facteur de proportionnalité soit encore une fois le facteur de force Bl. On note :

$$v = U / Bl \qquad [m/s]$$

avec v = vitesse du cône et U = tension de sortie du modèle électrique de calcul. Combinons cela avec la surface du cône pour trouver le volume d'air V_{air} déplacé :

$$V_{air} = v * Sd \qquad [m^3/s]$$

En 1954, Leo Leroy Beranek a déterminé que la pression sonore SPL à une distance *r* d'un radiateur sphérique (ou d'une source sonore ponctuelle) en plein air est égale à :

$$P_{SPL} = V_{air} * \rho * f / (2 * r) \qquad [N/m^2]$$

dans laquelle ρ est la densité de l'air (1,2 kg/m³). Dans l'équation ci-dessus, on présume que la source sonore est petite par rapport à la longueur d'onde du son, en prenant une fréquence de 30 Hz (on nage ici parmi les reproducteurs de grave) la longueur d'onde est de 10 m, on peut appliquer la formule l'esprit en paix.

Nous pouvons considérer la distance *r* comme le rayon d'une sphère dont la source sonore est le centre. Mais voilà, nous ne sommes pas vraiment en présence d'une source suspendue librement dans l'espace, mais bien d'une enceinte acoustique qui repose sur le sol. Ce qui signifie que l'énergie sonore ne se répand pas dans une sphère complète, mais dans une moitié seulement et donc que la pression sonore en est pratiquement doublée. Enfin, nous devons encore convertir la pression sonore en décibels en utilisant comme niveau de référence :

$$0\ dB = 2 * 10^{-5} \qquad [N/m^2].$$

Cellule de Boucherot

Les audiophiles aiment toujours savoir quelle impédance « voit » la sortie de leur amplificateur. La plupart des gens ne s'en soucient guère, en revanche, tout amplificateur préfère toujours attaquer une charge aussi ohmique que possible. La distorsion sera d'autant plus basse

Liens

[1] Page de l'article : www.elektormagazine.fr/180585-04
[2] Cellule de Boucherot : https://en.wikipedia.org/wiki/Boucherot_cell
[3] Gyrateur : https://en.wikipedia.org/wiki/Gyrator
[4] « Le haut-parleur manipulations et mesures électroacoustiques », J. D'Appolito, éditions Publitronic/Elektor, 1999, ISBN 978-2-86661-114-9 [livre épuisé, mais disponible sur le marché de l'occasion]
[5] « Enceintes acoustiques & haut-parleurs », Vance Dickason, éditions Publitronic/Elektor, 2006 (5e tirage), ISBN 978-2-86661-073-9 [livre épuisé, mais disponible sur le marché de l'occasion]
[6] Livre en néerlandais : 'Het luidspreker-bouwboek', H.H. Klinger, Elektuur BV, 1990, ISBN 9070160846
[7] Système de haut-parleurs Diamant : www.temporalcoherence.nl/cms/en
[8] Un système de haut-parleurs non conventionnel : www.breem.nl/lsp/LoudspeakerSystem.htm

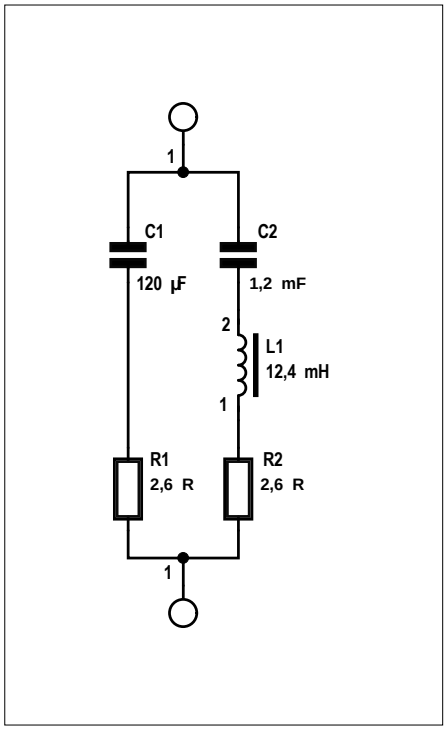


Figure 3. Une cellule de Boucherot sert à compenser la pointe de résonance d'un haut-parleur.

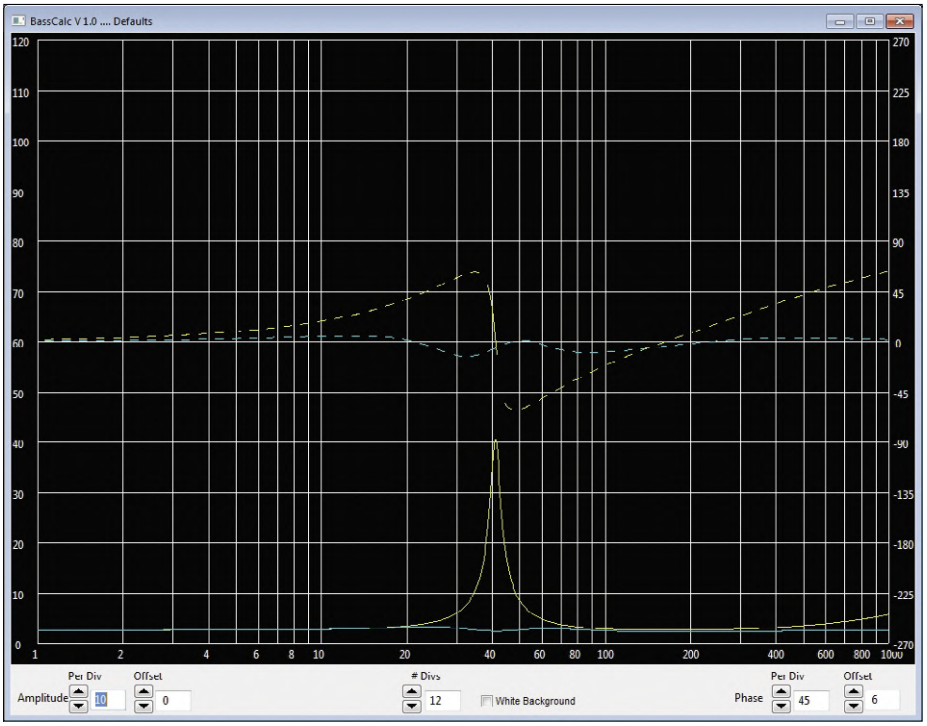


Figure 4. Ces courbes caractéristiques montrent clairement l'effet de la cellule de Boucherot : en jaune, le profil de l'impédance sans le réseau, en bleu, avec.

que la tension et le courant resteront à peu près en phase.

C'est ici qu'une cellule de Boucherot, aussi appelée *réseau de Zobel*, peut rendre de bons et loyaux services. Elle se branche en parallèle sur les bornes du HP, elle n'a que peu d'influence sur la charge totale, mais plutôt sur le courant que l'ampli doit délivrer.

Le réseau est fait en deux parties, l'une compense l'inductance *Le* de la bobine mobile alors que l'autre agit sur le résonateur formé par la masse et l'élasticité du HP dans l'enceinte.

Il n'est pas particulièrement difficile de calculer les valeurs des composants. On prend la même valeur que *Re* du haut-parleur pour les deux résistances (dans l'exemple du ScanSpeak 2,6 Ω donc). Pour C1 avec R1, la même constante de temps que *Re* avec *Le*. Nous trouvons alors :

$$C1 = Le / (Re * R1)$$

Pour le circuit du piston, nous prenons C-MMS (cf. fig. 1) comme valeur pour C2 et pour L1, la mise en parallèle de L-CMS et L-CBW (fig. 1 encore).

J'admets volontiers que C2, avec une valeur de 1,22 mF, serait un mastodonte en technologie à film plastique. Une autre solution consiste à mettre en série deux électrolytiques en opposition tout en reliant le nœud central à la tension d'alimentation de l'ampli pour qu'aucun d'entre eux ne se retrouve polarisé en inverse.

La bobine L1 de 12,4 mH n'est pas particulièrement svelte non plus. En théorie, un gyrateur pourrait apporter une solution, mais l'inconvénient serait de relier une des bornes du haut-parleur à la masse. J'ai donc préféré une bobine à noyau de fer (fermé), il ne cause aucune

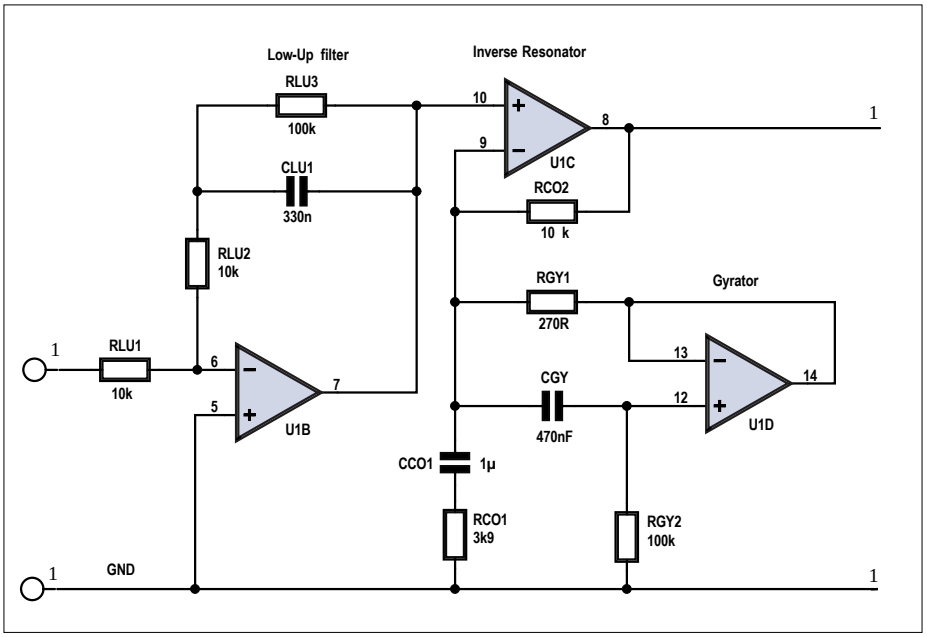


Figure 5. Le schéma du filtre de compensation. Les valeurs des composants ne sont valables que pour le haut-parleur ScanSpeak utilisé dans ce projet. Les dénominations correspondent à celles employées dans le programme BassCalc.

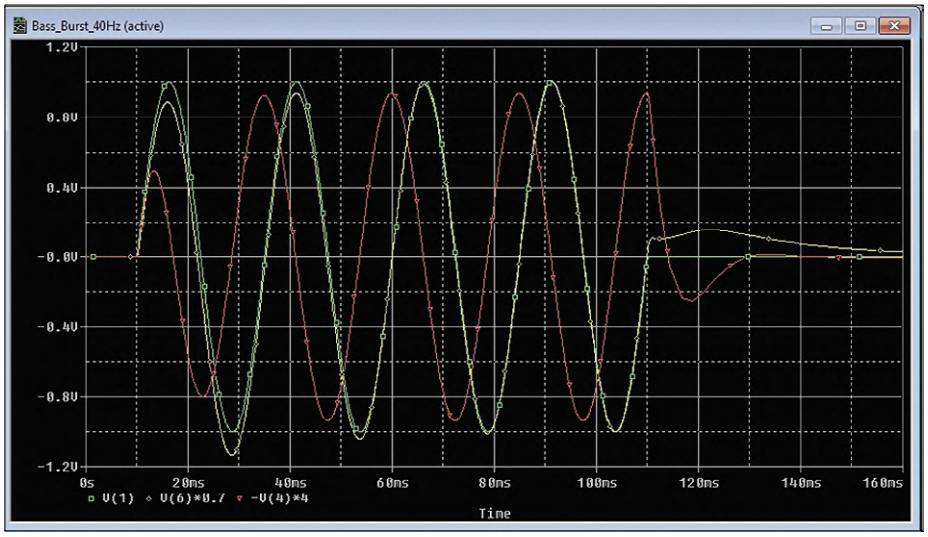


Figure 6. Simulation Spice du système de reproduction avec et sans filtre de compensation. En vert, le signal d'entrée, une salve de quatre périodes de sinusoïde à 40 Hz ; en rouge, la réponse sans compensation, en jaune, avec compensation. Se passe de commentaire...

gêne puisque la cellule Boucherot n'est pas dans le parcours du signal. Il suffit de s'assurer que le noyau n'atteint jamais la saturation complète.

On voit bien à la **figure 4** l'influence bénéfique de la cellule de Boucherot.

Filtre de compensation

Venons-en à la dernière partie des considérations annoncées : le filtre de compensation qui relève de 12 dB par octave la caractéristique de fréquence en dessous de la résonance. Le même filtre agit aussi sur la résonance propre du système, ce qui donne une reproduction extrêmement plate du registre grave.
Le schéma de ce filtre est à la **figure 5**. Il se compose d'un amplificateur accentuateur des basses et d'un circuit qui réfrène la résonance du haut-parleur dans l'enceinte.
L'ampli de rattrapage des basses fréquences, c'est U1B. RLU3 et CLU1 déterminent la fréquence à partir de laquelle la remontée s'opère. Les valeurs indiquées ici se rapportent expressément au haut-parleur de ScanSpeak dans une enceinte de 41 litres, la fréquence est de 4,8 Hz. Le même CLU1 et RLU2 cette fois fixent la fréquence charnière où la suramplification cesse, ici à 48,5 Hz environ.
Le circuit suivant s'occupe de la résonance du système, il est construit autour de U1C. On peut le décrire comme un

Le logiciel BassCalc

Ce programme, dont la **figure 7** illustre l'interface utilisateur, a été développé avec Free Pascal et l'EDI de Lazarus. Il s'agit d'outils à télécharger gratuitement et qui peuvent tourner sur presque toutes les plateformes (Windows, Linux, Mac, RPi, Android). Le slogan « Write Once, Compile Anywhere » correspond vraiment à la réalité, même si des surprises sont toujours possibles selon le système d'exploitation. Quoi qu'il en soit, le programme BassCalc a été testé sur WinXP, Win7 et Linux Mint.
Dans la fenêtre de paramétrage de la figure 7, on introduit les valeurs TS dans le champ supérieur gauche (les valeurs par défaut y sont celles utilisées pour le châssis ScanSpeak) et à côté, à droite, les valeurs équivalentes du modèle mathématique électrique plus les fréquences de résonance résultantes et la valeur totale du facteur Q.
On trouve dans le champ de droite les valeurs des composants pour le filtre de compensation et plus bas, celles pour le filtre accentuateur de grave.
Viennent ensuite (sous le bouton *Recalculate*) dans le champ de gauche les valeurs de la cellule de Boucherot (notée *Zobel*) ; en dessous, on peut indiquer le domaine

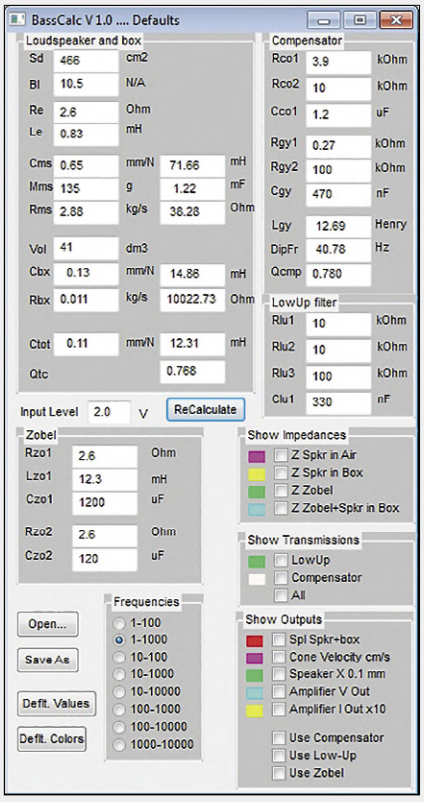


Figure 7. L'interface utilisateur du programme BassCalc. Les valeurs par défaut sont celles utilisées pour le châssis ScanSpeak utilisé.

de fréquence dans lequel il faut effectuer les calculs et inscrire les résultats. Dans les champs de droite, on peut encore choisir les graphiques intéressants (impédance, transmissions, sorties) et finalement prescrire les corrections à inclure. Les cases colorées permettent de changer les couleurs.
Les boutons *Open* et *Save* offrent la possibilité d'enregistrer un ensemble de données puis d'aller les rechercher. Les boutons *Deflt Values* et *Deflt Colors* servent à restaurer les valeurs par défaut du programme. Si les valeurs des composants ont changé, il faut pousser sur *Recalculate*.
Vous donnez dans *Input Level* la valeur de la tension à appliquer au haut-parleur. À 1 W sur 8 Ω correspond 2,38 V. Pour un HP de 4 Ω comme le ScanSpeak du projet, il faut 2 V pour 1 W. Remarque : on fait précéder ici les décimales d'un point, pas d'une virgule.

Il n'est pas nécessaire d'installer le programme : copiez le fichier exécutable dans un dossier et faites-le exécuter. Sachez que si vous entrez des données bizarres, le programme répondra dans la même veine ou se plantera.

antirésonateur centré sur U1C. Le gain de cet ampli op dépend de l'impédance entre son entrée (−) et la masse, elle est constituée du condensateur CC01, de la résistance RC01 et du gyrateur basé sur U1D. Le gyrateur, aussi appelé convertisseur d'impédance négative, est un circuit d'un intérêt fabuleux, mais il faudrait tout un chapitre pour en décrire le fonctionnement. Dans ce cas, rien de tel que d'aller consulter la littérature abondante à trouver sur l'internet, entre autres sous [3]. Le gyrateur est souvent utilisé pour convertir l'inductance d'un condensateur en celle d'une bobine dépourvue des inconvénients habituels de poids, d'encombrement et des parasites inhérents au bobinage. C'est le cas ici, et pour la figure 5, on exprime l'inductance équivalente par :

L = CGY * RGY1 * RGY2
[valeurs en farad, ohm et henry]

Cette inductance équivalente est en parallèle sur CCO1 et RCO1, de sorte qu'il se forme un circuit parallèle dont la fréquence de résonance est égale à :

$f = 1 / (2 * \pi * \sqrt{(L * C)})$

ce qui donne avec les valeurs de composants de la figure 5 environ 40 Hz.
Si vous voulez expérimenter sur ce schéma, mieux vaut partir, pour la fréquence de résonance et le facteur de mérite Q de l'antirésonateur, de valeurs égales à celles du modèle de haut-parleur. Pour régler le facteur Q, modifiez la valeur de RC01, qui n'a aucune influence sur la fréquence de résonance. La fréquence du pôle du filtre d'accentuation des basses devrait aussi se situer au voisinage de la fréquence de résonance. Dans le programme BassCalc (cf. encadré), vous pouvez jouer sur toute une série de valeurs de paramètres sans devoir faire chauffer votre fer à souder. Si vous décidez de construire votre version de ce circuit, utilisez de préférence des amplis op caractérisés par une distorsion extrêmement faible, tels que l'OPA134.

Le résultat

Après toute cette théorie, la **figure 6** montre le résultat de nos cogitations : une simulation Spice du reproducteur de graves de ma conception (cf. encadré) et qui constitue le sujet de cet article.
Le signal d'entrée, une salve de quatre périodes d'un signal sinusoïdal à 40 Hz, est tracé en vert. En rouge, on voit la réponse du haut-parleur de ScanSpeak dans l'enceinte de 41 l **sans** compensation. Clairement, le reproducteur a du mal à démarrer, puis il accuse un retard par rapport au signal d'entrée.
La réponse du système **avec** compensation est en jaune : la reproduction du signal suit presque à la perfection le signal appliqué. Ce constat donne tout son sens au titre de reproducteur de basses de **haute ponctualité** !
Bien entendu, une simulation n'est rien d'autre qu'une simulation. Néanmoins, le système que j'ai construit sonne remarquablement clair et détaillé. Le labeur investi, tant théorique que pratique, en valait la peine.
Vous le savez, ceci n'est pas un projet tout cuit, qu'il suffit de copier, mais il doit permettre à qui le désire d'intégrer les informations fournies pour réaliser son propre système. Vous avez d'ailleurs la possibilité de prendre contact avec moi via la rédaction. ⏮

(180585-04 – version française : Robert Grignard)

Un système de haut-parleurs non conventionnel

La **figure 8** montre un système de haut-parleurs qui concrétise le résultat des considérations théoriques exposées dans cet article. C'est une réalisation à trois voies, chaque haut-parleur dispose de son propre amplificateur en classe D. Le boîtier pentagonal en dessous abrite le reproducteur de grave de ScanSpeak, lequel produit dans cette configuration des fréquences jusque sous les 20 Hz environ. Le système dans son ensemble est inspiré de celui de Hans van Maanen appelé Diamond [7].
Vous trouverez une description complète (en anglais) de ce système de haut-parleurs sur l'internet [8].
S'il se manifeste pour lui un intérêt suffisant, Elektor pourrait y revenir dans une prochaine édition.

Figure 8. Le système à trois voies tel que l'auteur l'a construit d'après les considérations développées dans cet article.

amplificateurs opérationnels et charges capacitives

limiter les oscillations parasites

Robert Lacoste (Chaville)

Source : Emilian Robert Vicol de Pixabay

Dans le précédent article, je vous ai présenté les caractéristiques essentielles des amplificateurs opérationnels (AOP). Je vous ai également expliqué comment sélectionner celui qui sera adapté à votre projet. Évidemment, choisir le bon composant ne suffit pas, il faut ensuite concevoir le circuit autour de celui-ci. Cela peut sembler facile puisqu'on trouve sans soucis des tas d'exemples d'applications pour les AOP dans cette revue comme sur le web ou dans votre bibliothèque préférée. Il y a même des logiciels de conception qui permettent de synthétiser en quelques clics de souris le schéma pour un tas d'applications, par exemple pour réaliser un amplificateur ou un filtre actif.

Oui, c'est vrai, construire un circuit avec un AOP est beaucoup plus facile qu'avec des transistors. Mais est-ce toujours aussi simple ? Évidemment, la réponse est non : quelquefois l'électronique est un peu capricieuse et on peut avoir des surprises désagréables. Je vous parlerai ici de l'une des difficultés les plus fréquentes : le pilotage de charges capacitives. Conformément à sa description, ce problème peut apparaître dès que la sortie d'un amplificateur à contre-réaction, et en particulier d'un AOP, est connectée à quelque chose qui ressemble de près ou de loin à un condensateur. Et c'est fréquent : condensateurs de filtrage en sortie d'une alimentation, capacité d'entrée de l'étage suivant du montage, capacité parasite de longs fils de connexion, composants par nature capacitifs comme des transducteurs piézo, etc. De telles charges capacitives peuvent conduire à des cauchemars pour le concepteur : oscillations parasites, comportement erratique, voire instabilité complète ou même composants partant en fumée si vous n'avez vraiment pas de chance.
Bien entendu, tout ceci n'est pas mystique, et s'explique par la théorie des systèmes asservis. Mais pas de stress, il n'y aura pas l'ombre d'une formule mathématique dans cet article. Je vais essayer de vous expliquer simplement ce qui pourrait mal tourner, et surtout quelles solutions peuvent remédier au problème.

Un exemple…

Prenons un exemple très simple : un amplificateur non-inverseur réalisé avec un antique AOP, le UA741 [1] (**fig. 1**). Rien d'extraordinaire, le signal est connecté à l'entrée positive de l'AOP, tandis que son entrée négative est connectée à un pont de deux résis-

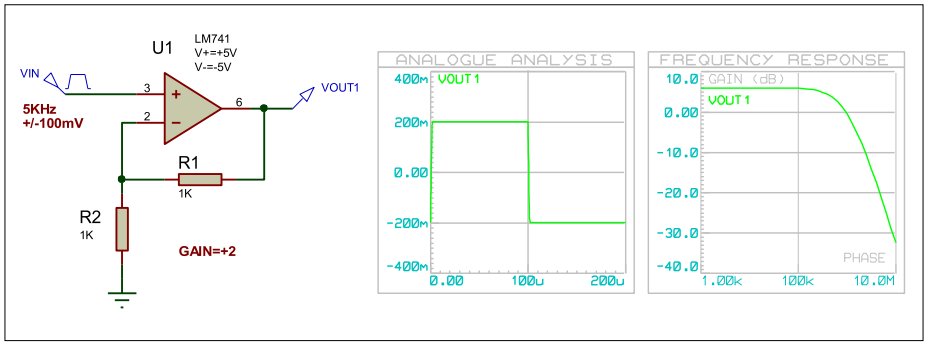


Figure 1. Un amplificateur non-inverseur de gain 2. La simulation montre que la tension du signal de sortie est bien deux fois plus élevée que celle du signal d'entrée. La réponse en fréquence est simulée à droite.

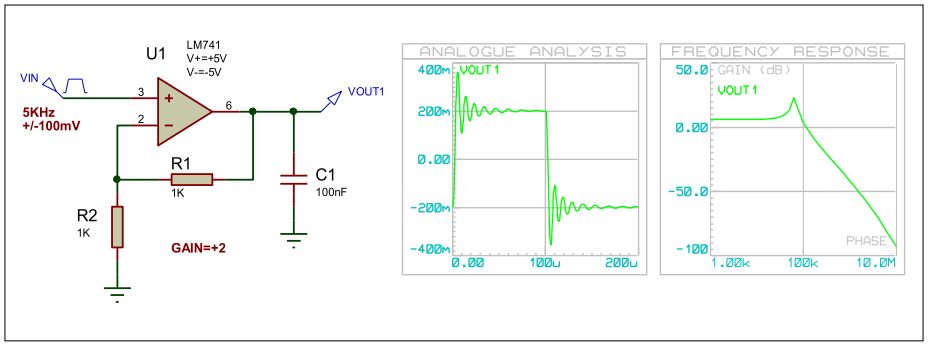


Figure 2. L'ajout d'une petite charge capacitive en sortie de l'amplificateur, ici C1, provoque de fortes oscillations parasites.

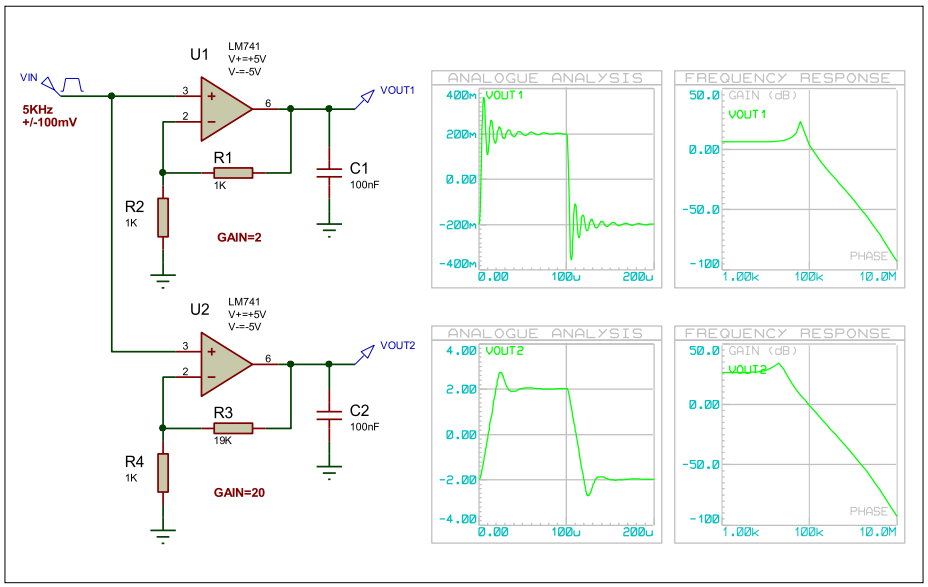


Figure 3. À charge capacitive égale (ici 100 nF), les oscillations sont beaucoup plus élevées avec un gain de 2 (en haut) qu'avec un gain de 20 (en bas).

tances de 1 kΩ branchées entre la sortie de l'ampli et la masse, ce qui crée une contre-réaction. Comme expliqué dans le précédent article, l'AOP fait tout son possible pour avoir la même tension sur ses deux entrées. Ici, il se débrouillera pour que la tension appliquée à l'entrée inverseuse soit la même que la tension d'entrée, c'est-à-dire, grâce au diviseur de tension, pour que la tension de sortie soit deux fois plus élevée que cette tension d'entrée. C'est donc un amplificateur non-inverseur avec un gain de 2 en tension.

À ce stade, je vous encourage chaudement à essayer de réaliser ce circuit, avec un fer à souder ou, plus simplement, en utilisant un logiciel de simulation sur PC. Pour ma part, j'ai utilisé le simulateur Spice intégré à ma CAO Proteus (Labcenter) [2], mais vous pouvez facilement reproduire cet exemple avec n'importe quel simulateur comme l'excellent et gratuit LT-Spice (disponible sur le site d'Analog Devices [3]). Regardez de nouveau la figure 1, j'ai connecté un générateur virtuel sur l'entrée (signal carré de 5 kHz, ±100 mV) et une sonde pour mesurer la tension de sortie (V_{out}). Il n'y a plus qu'à cliquer sur « Run » et le simulateur montre évidemment que le signal de sortie est un aussi un signal carré, mais d'amplitude ±200 mV. Le gain en tension est bien de 2. Le simulateur m'a également permis de tracer la réponse en fréquence de cet amplificateur : le gain est plat, +6 dB, au moins jusqu'à un peu plus de 100 kHz. Juste un mot pour ceux qui seraient surpris par ces 6 dB : « ×2 » en tension correspond à « ×4 » en puissance (souvenez-vous, $P=U^2/R$?). Et « ×4 » équivaut à +6 dB, car 10 × log(4) = 6.

Oscillations parasites ?

Il n'y a rien de passionnant jusque-là. Maintenant que se passe-t-il si on monte un condensateur entre la sortie de l'amplificateur et la masse ? Regardez la **figure 2**, c'est le même amplificateur avec un condensateur de 100 nF en sortie (C1). Le signal de sortie est encore plus ou moins un carré d'amplitude ±200 mV, mais avec des oscillations parasites impressionnantes à chaque transition du signal carré. Regardez aussi la réponse en fréquence et comparez-la à celle de la figure 1. Toute d'abord la fréquence de coupure est un peu plus basse. Ensuite il y a un pic étrange autour de 65 kHz. Cette fréquence correspond approximativement à la fréquence des oscillations parasites sur la sortie. Cela ne devrait pas vous surprendre, car l'amplificateur a envie d'osciller à cette fréquence en raison de son gain plus élevé.

D'où vient ce phénomène ? En deux mots, la charge capacitive, en combinaison avec l'impédance de sortie de l'AOP, provoque un retard dans la boucle de rétroaction. Ce retard équivaut à un déphasage qui transforme l'amplificateur en oscillateur plus ou moins amorti. Il semble logique qu'en raison de ce retard, la sortie de l'amplificateur soit déjà en

train de monter à une tension trop élevée lorsque l'entrée négative « sait » qu'il est temps de ralentir, et vice-versa.
Plutôt que de consacrer du temps à l'explication mathématique du phénomène, je vous propose d'examiner les paramètres qui l'influencent. Commençons par le gain de l'amplificateur. J'ai simulé pour vous l'effet de la même charge capacitive de 100 nF sur deux copies du même amplificateur, mais avec des gains respectifs de 2 et 20. Regardez la **figure 3** : je pense que le résultat n'est pas du tout intuitif. Les oscillations sont beaucoup plus faibles lorsque le gain de l'amplificateur est augmenté ! Cela signifie que le plus difficile en matière d'oscillations parasites n'est pas de construire un amplificateur à gain élevé, mais un simple suiveur de tension de gain 1 ! Pourquoi ? Rappelez-vous que le produit gain × bande passante d'un AOP est presque constant. Si on a un gain plus élevé, la bande passante est plus faible. C'est bien visible sur les réponses en fréquence de la figure 3 : la fréquence de coupure descend de 100 kHz à 10 kHz pour un gain passant de 2 à 20. Cette réponse en fréquence plus basse réduit à la fois l'amplitude et la fréquence des oscillations. C'est encore logique : puisque l'amplificateur est plus lent, l'effet d'un même retard sur la boucle de contre-réaction est moins critique.
Cela m'amène à un autre paramètre important : la vitesse de l'AOP lui-même. Si vous utilisez un composant très rapide, c'est-à-dire avec une très grande bande passante, la même charge capacitive pourra entraîner des problèmes plus fâcheux... Une bande passante plus élevée implique qu'une petite capacité parasite peut aussi créer des oscillations, mais à très haute fréquence. Juste pour le plaisir, j'ai simulé le même amplificateur en remplaçant le UA741 par un AOP beaucoup plus véloce, un AD8009AN [4] (Analog Devices). Ce composant a un produit gain × bande de l'ordre de 1 GHz, quelques milliers de fois meilleur que le vénérable UA741. Bien sûr, la fréquence de coupure est beaucoup plus élevée et les oscillations parasites sont bien présentes, mais à plus haute fréquence. J'ai essayé de réduire le gain de l'ampli afin d'aggraver la situation. Un phénomène intéressant est apparu pour un gain proche de 1 : comme le montre la **figure 4**, l'amplificateur devient alors un oscillateur, avec une fréquence de sortie de l'ordre de 300 MHz, et ce même avec une charge capacitive aussi faible que 1 nF !

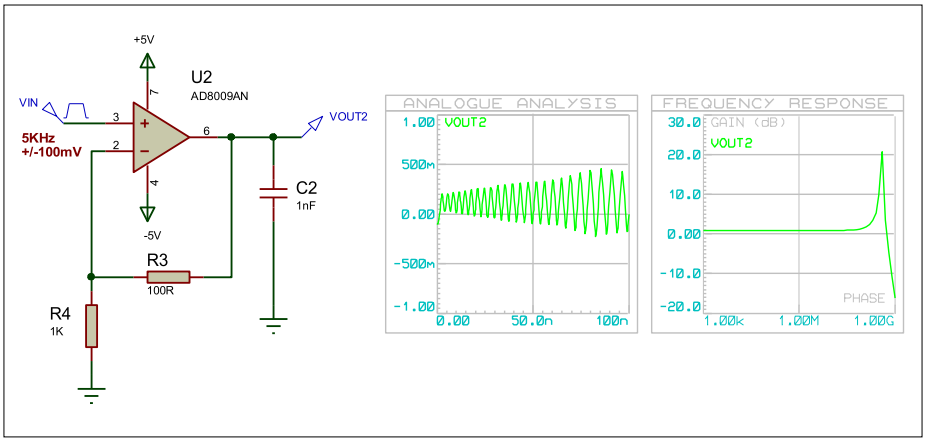


Figure 4. Un AOP plus rapide entraîne des oscillations à plus haute fréquence, voire une entrée en oscillation permanente. Ici un AD8009 est transformé en oscillateur dès que le gain est inférieur à 1,2, même avec une petite charge capacitive de 1 nF.

Contre-mesures !

Alors, comment faire ? Évidemment la bonne solution c'est d'éviter toute charge capacitive en sortie de l'amplificateur, mais c'est souvent impossible. Quelles sont les solutions pour limiter ces oscillations parasites ? Grâce aux exemples précédents, vous en connaissez déjà deux. Tout d'abord, opter pour un amplificateur aussi lent que possible, évidemment en fonction des exigences du projet. Deuxièmement, prendre particulièrement soin des étages à faible gain. Il faut choisir avec un soin tout particulier les suiveurs de tension, souvent utilisés comme étage final pour piloter une charge à courant élevé, car ils

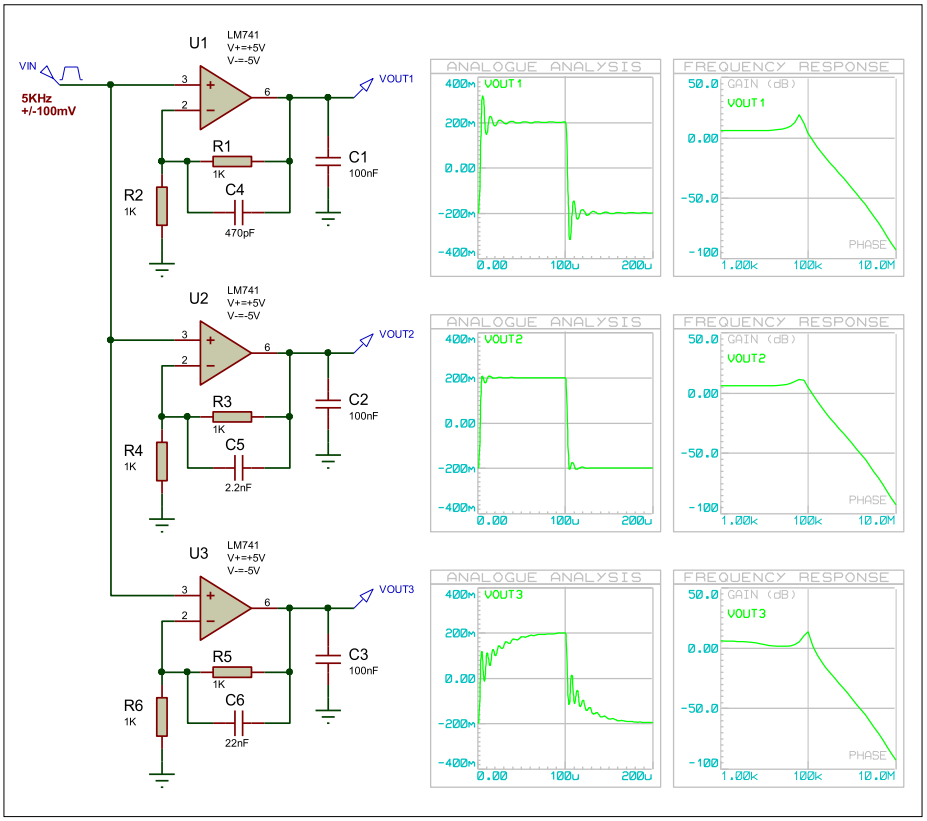


Figure 5. L'ajout d'un condensateur en parallèle de la résistance de rétroaction réduit la bande passante, mais aussi les oscillations parasites. Ici 470 pF est un peu trop faible (en haut), et 22 nF est trop élevé (en bas). Un 2,2 nF fait l'affaire (au centre).

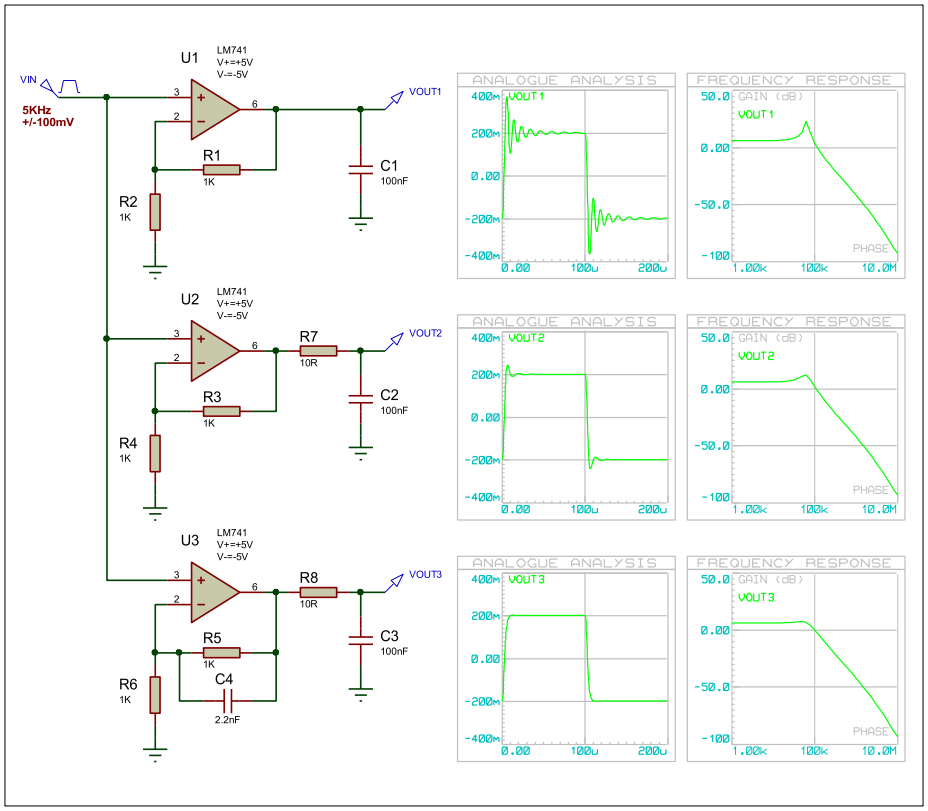


Figure 6. L'ajout d'une résistance série entre la sortie et la charge capacitive permet d'obtenir une amélioration impressionnante (au centre). Cette solution peut être combinée avec l'ajout d'un condensateur de rétroaction (en bas).

peuvent facilement osciller. Mieux vaut donc répartir le gain entre les différents étages. Notez que les circuits intégrés dits « suiveurs de tension » spécifiquement conçus pour cette application ont une impédance de sortie très faible, ce qui réduit les risques. Lisez aussi en détail les fiches techniques des AOP choisis, il y a souvent des diagrammes de stabilité en présence de charge capacitive.

À part ça, quoi faire ? La première technique consiste à réduire volontairement la bande passante de la rétroaction. Cela diminue la bande passante de l'amplificateur complet et, par conséquent, les oscillations parasites à haute fréquence. La façon la plus simple pour cela est d'ajouter un petit condensateur en parallèle de la résistance de contre-réaction, c'est-à-dire entre la sortie et l'entrée négative de l'amplificateur. Regardez la **figure 5**, où j'ai simulé trois copies du même amplificateur de gain 2 avec la même charge capacitive de 100 nF, mais avec respectivement un condensateur de contre-réaction de 470 pF, 2,2 nF et 22 nF. Si la valeur de ce condensateur de rétroaction est trop faible, les oscillations restent élevées. Réciproquement si elle est trop élevée, la forme d'onde de sortie est trop déformée. Une valeur intermédiaire fournit une réponse plus propre, même si les oscillations sont encore visibles. Pour ceux qui suivent, c'est exactement la même chose que le petit condensateur ajustable qu'on trouve sur certaines sondes d'oscilloscope et qui permettent de compenser leur réponse en fréquence.

Vous voulez une autre très bonne technique ? Regardez la **figure 6**. Ici, on ajoute une simple résistance série de 10 Ω (R7) entre la sortie de l'amplificateur opérationnel et la charge capacitive. L'effet est vraiment impressionnant, les oscillations disparaissent presque totalement. Pourquoi ? Comme le signal de rétroaction est un peu isolé du condensateur de sortie grâce à la résistance, l'information de rétroaction arrive plus vite à l'AOP… Cette méthode très efficace peut bien sûr être combinée avec la précédente. Regardez la branche du bas sur la figure 6 pour vous en convaincre.

D'autres soucis ?

Ajouter une résistance en série sur la sortie semble presque magique, mais cette technique a-t-elle des inconvénients ? Bien sûr que oui. Le problème est que

Liens

[1] AOP UA741 : www.st.com

[2] Logiciel PROTEUS VSM : www.labcenter.co.uk

[3] Logiciel LTSPICE : www.linear.com/designtools/software/

[4] Amplificateur à faible distorsion AD8009AN, 1 GHz (bande passante), 5,500 V/µs (vitesse de montée) : www.analog.com

[5] 'The Art of Electronics', 3rd edition, Paul Horowitz and Winfield Hill, Cambridge University Press, ISBN 978-0-521-80926-9
[la version française est disponible sous le titre « Traité de l'électronique analogique & numérique » (volume 1, ISBN 978-2-86661-1)]

[6] "Ask The Application Engineer : Practical Techniques to Avoid Instability Due to Capacitive Loading", Soufiane Bendaoud & Giampaolo Marino, Analog Devices Inc : www.analog.com/library/analogDialogue/archives/38-06/capacitive_loading.html

[7] "Does your Op Amp oscillate?", Bary Harvey, note d'application 148, Linear Devices : http://cds.linear.com/docs/en/application-note/an148fa.pdf

cette résistance augmente l'impédance de sortie de l'amplificateur. Si un courant circule dans la sortie, cette résistance provoque une chute de tension. Cette chute ne sera pas compensée par l'AOP, car cette résistance est en dehors de la boucle de contre-réaction. Voyons cela sur une dernière simulation (**figure 7**). Dans la branche supérieure, j'ai repris le circuit sur lequel on est arrivé, mais en ajoutant une charge de 50 Ω en sortie (R6). Il n'y a toujours pas d'oscillations parasites (c'est super), mais l'amplitude du signal est de ±180 mV et non de ±200 mV comme attendu. De plus, cette amplitude changerait si la charge était variable, ce qui n'est pas idéal.

Quelle solution peut-on essayer si cela n'est pas acceptable ? La première idée est de connecter la boucle de contre-réaction après la résistance de 10 Ω et pas avant, car cela permet à l'AOP de compenser tout décalage de tension sur la sortie. Cela fonctionne, mais malheureusement de petites oscillations parasites réapparaissent, comme illustré sur la branche centrale sur la figure 7.

Une meilleure idée ? Une solution très astucieuse consiste à diviser la boucle de contre-réaction en deux circuits, comme indiqué sur le schéma sur la branche inférieure de la figure 7. L'idée est qu'un « chemin rapide » à travers C7 permet d'arrêter rapidement les oscillations à haute fréquence, alors que la seconde branche (via R9 et C6) est connectée après la résistance de 10 Ω et annule tout décalage de tension statique. Cela donne un résultat quasi parfait en simulation. Bien sûr, dans la vraie vie, ce ne sera peut-être pas si simple, mais vous avez saisi le concept.

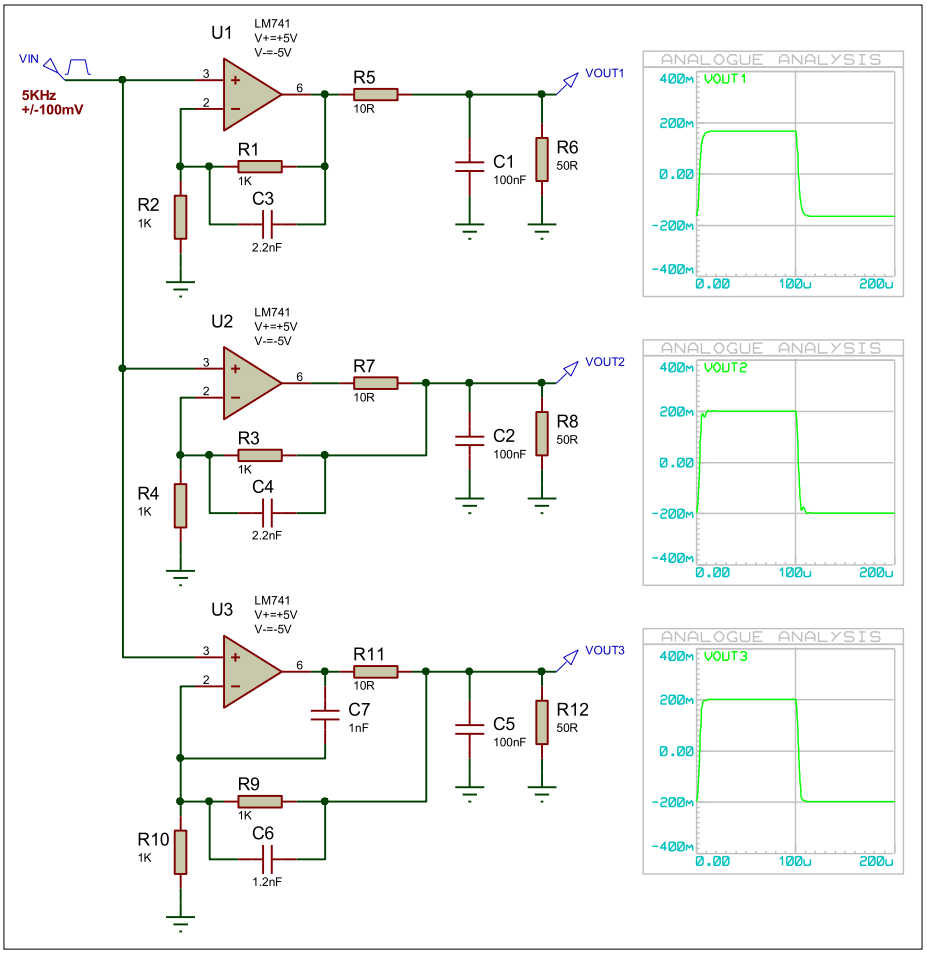


Figure 7. La résistance série R5 introduit une erreur sur la tension de sortie lorsqu'une charge résistive est présente (en haut). Brancher la contre-réaction après cette résistance corrige l'erreur, mais réintroduit des oscillations parasites (au centre). La meilleure solution est souvent de prévoir deux branches de contre-réaction (en bas).

Pour conclure

J'espère que vous avez une meilleure compréhension des soucis de coexistence entre charge capacitive et amplificateurs opérationnels. Si le sujet vous intéresse, je vous encourage vivement à aller plus loin après avoir lu cet article. Deux solutions : lire ou expérimenter.

Coté lecture et pour ceux qui sont plus ingénieurs que mathématiciens, je vous recommande de vous plonger dans la bible du sujet : « The Art of Electronics » de Paul Horowitz et Winfield Hill [5]. Ce livre n'est pas donné, mais vous ne le regretterez pas. D'ailleurs c'est cet ouvrage qui m'a fait découvrir la solution des deux branches de contre-réaction illustrée ci-dessus. De très bonnes références sont également disponibles sur le web, en particulier l'article écrit par Soufiane Bendaoud et Giampaomo Marino (Analog Devices) [6] et la note d'application 148 de Linear Technologies rédigée par Barry Harvey [7].

Coté expérimentation et comme toujours, je vous encourage fortement à tester par vous-même. Mettez sous tension votre oscilloscope et votre fer à souder, ou téléchargez LTSpice et reproduisez les exemples de cet article. Ensuite, modifiez les valeurs des composants et étudiez les résultats. Comme l'a dit Niehls Bohr : « Un expert est une personne qui a commis toutes les erreurs possibles dans un domaine restreint », alors n'ayez pas peur des erreurs et amusez-vous ! ⏮

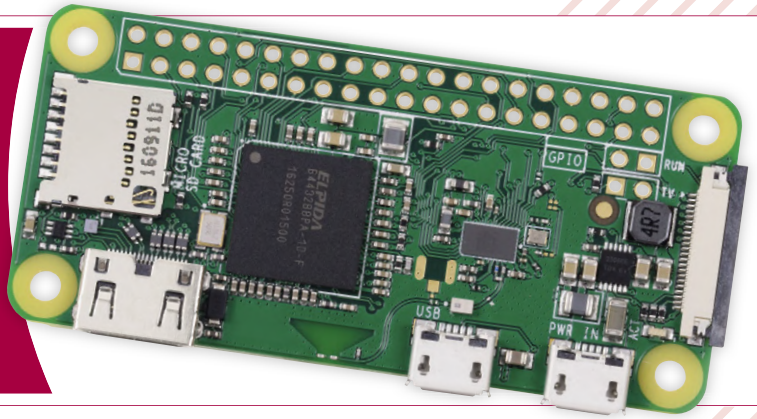
Cet article a été publié dans la revue Circuit Cellar (n°305, décembre 2015).

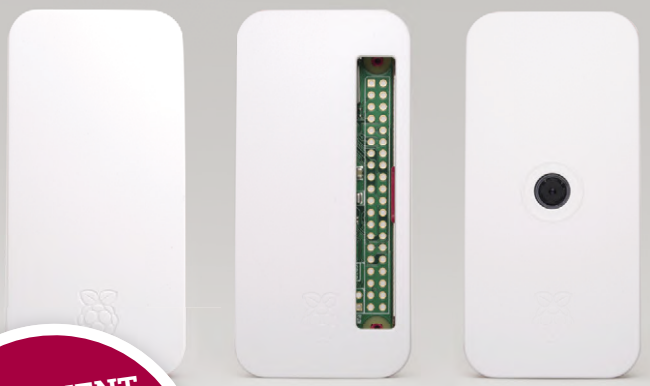
(190330-01)

Source : Emilian Robert Vicol de Pixabay

ABONNEZ-VOUS ET RECEVEZ

RPI ZERO W GRATUIT





SEULEMENT **54,95 €** PAR AN (6 NUMÉROS)

TOUS LES 2 MOIS, LES DERNIÈRES NOUVELLES DU RASPBERRY PI ET LES MEILLEURS PROJETS !

Souscrivez dès maintenant un abonnement d'un an au magazine MagPi, nous vous offrons :

- Six numéros du magazine MagPi
- Une carte Raspberry Pi Zero W
- Un boîtier avec trois couvercles différents
- Un connecteur pour module de caméra
- Un câble HDMI/mini-HDMI et un câble micro-USB/USB OTG

Vos avantages :

- Prix au numéro réduit
- Chaque numéro directement dans votre boîte aux lettres
- Tous les numéros disponibles sous forme numérique (PDF)
- Cadeau de bienvenue d'une valeur de 22,95 €
- Découverte de chaque nouveau numéro avant sa sortie en kiosque

ABONNEZ-VOUS : WWW.MAGPI.FR

productronica
fast forward
the start-up platform
powered by Elektor

PARTICIPEZ AU CONCOURS POUR LANCER VOTRE STARTUP SUR LE SALON **PRODUCTRONICA** 2019

Participez à l'édition 2019!

November 12-15, 2019
Munich

informations détaillées :
www.elektormagazine.fr/p-ffwd

Productronica Fast Forward is brought to you by

productronica

elektor
INNOVATION STARTUP TRADE

Platinum Sponsor:

DISTRELEC
Distribution with a difference

Silver Sponsor:

kurtz ersa
Weller
almit

Bronce Sponsor:

BERNSTEIN
TOOLS FOR ELECTRONICS

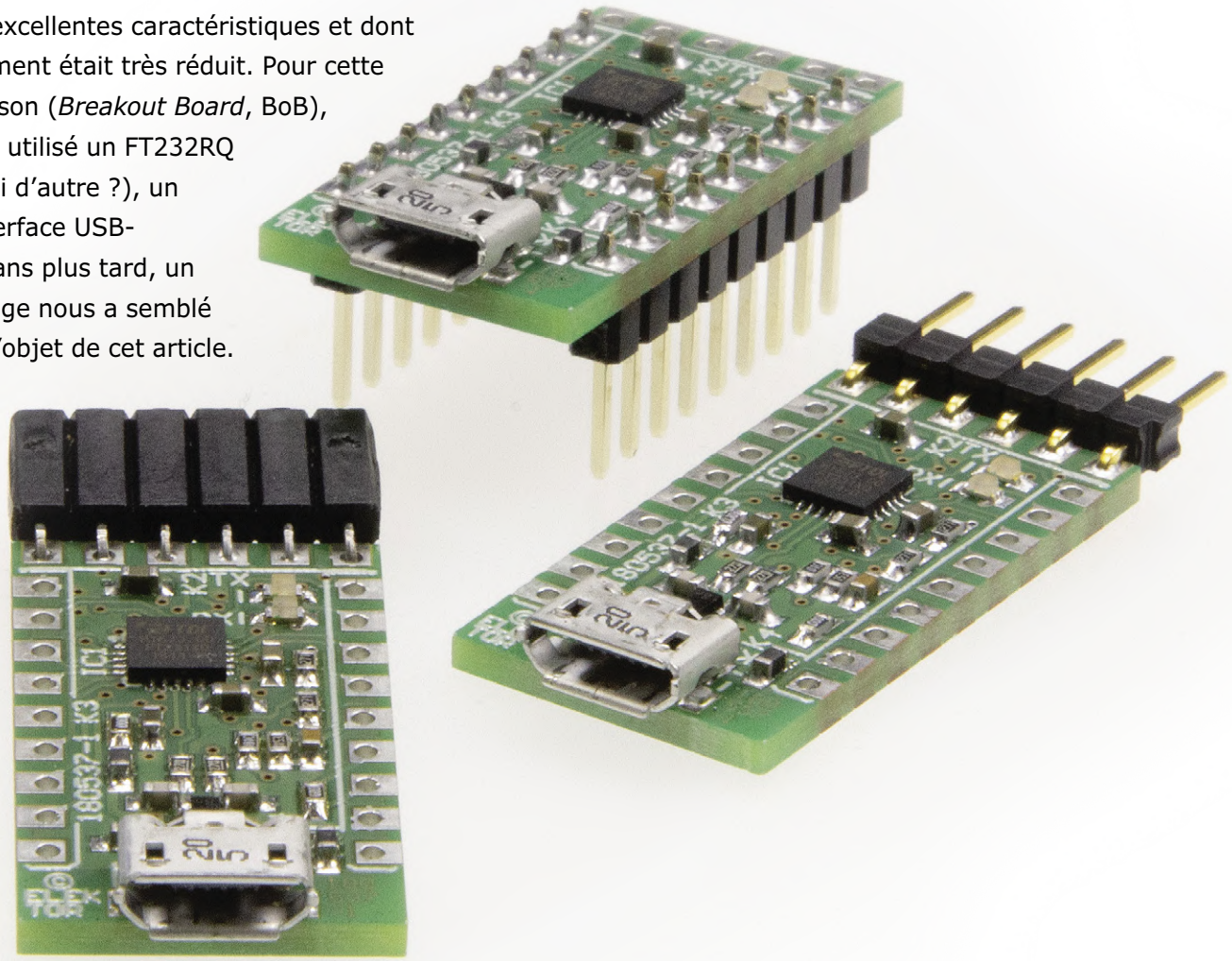
PROJET DU LABO

passerelle USB/RS232

appelez-moi BoB

Ton Giesberts (labo d'Elektor)

Dans le numéro de septembre 2011 du magazine, nous avons décrit un convertisseur USB-série qui avait d'excellentes caractéristiques et dont l'encombrement était très réduit. Pour cette carte de liaison (*Breakout Board*, BoB), nous avions utilisé un FT232RQ de FTDI (qui d'autre ?), un circuit d'interface USB-UART. Huit ans plus tard, un dépoussiérage nous a semblé utile, c'est l'objet de cet article.



Caractéristiques

- Connecteur micro-USB
- Entièrement compatible avec USB 2.0
- VCCIO : 1,8 à 3,3 V (max. 4 V)
- Sortie 3,3 V régulée (max. 50 mA)
- Vitesse de transmission de 300 à 3 Mbauds
- Compatible avec RS-232, RS-485 et RS-422
- Protocole d'échange de données USB->UART complet
- Quatre E/S programmables

Il y a bien entendu quelques modifications par rapport à notre convertisseur précédent [1]. Nous avons tout d'abord choisi un circuit intégré moins cher, le FT231XQ. Le raccordement de la carte est toujours possible avec des barrettes de picots à souder. Celles-ci sont moins espacées que sur l'ancien BoB, et la carte en est moins large ; elle est cependant un peu plus longue : 29,2 mm

contre 27 mm. Avec le connecteur micro-USB et une barrette de picots à souder coudés à l'opposé pour l'interface RS-232, la longueur totale est légèrement inférieure à 4 cm.
Nous utilisons effectivement un connecteur micro-USB plutôt qu'un mini-USB ; ces modèles sont désormais tout à fait fiables. Nous avons modifié l'ordre des signaux sur le connecteur RS-232, et nous avons ajouté le signal RTS ; le tout est maintenant compatible avec les câbles USB/RS-232 de FTDI. Un ensemble de diodes de protection contre les décharges électrostatiques (*Electrostatic Discharges, ESD*) complète les modifications.
Si vous êtes intéressé par le fonctionnement interne du FT231XQ, consultez la fiche de caractéristiques [2] et la note technique 140 de FTDI [3], qui comprend une liste des problèmes potentiels.

Le circuit

Le schéma de la **figure 1** reprend l'ensemble du circuit ; c'est bien moins compliqué que ça en a l'air. IC1, notre convertisseur, est le cœur du montage. Ses broches sont reliées à quatre connecteurs pour les liaisons vers l'extérieur : K1, le connecteur micro-USB ; K2, l'interface RS-232 ; K3, qui dispose de tous les signaux de l'UART ; et K4, qui reprend quatre E/S programmables d'IC1 et les lignes d'alimentation.
Les deux LED indiquent l'activité sur les lignes de données du bus USB : LED1 est connectée à CBUS1 et signale la transmission (TX) de données ; LED2 est connectée à CBUS2 et signale une réception (RX) de données. Le circuit pourrait aussi être configuré pour indiquer l'activité de l'interface RS-232. Il est à noter que l'indication « TX » sur le circuit imprimé se trouve près de K2, mais n'a rien à voir avec ce connecteur : il n'y avait pas assez de place ailleurs…
L'ensemble de diodes D1, de Nexperia, protège le circuit contre d'éventuelles décharges électrostatiques. Le circuit choisi a une très faible capacité parasite. On peut enfin noter la présence de deux résistances de valeur nulle (0 Ω), R1 et R8. Nous y revenons plus loin lors de la description de l'alimentation.

L'alimentation

Il est utile de s'attarder sur l'alimentation : toute erreur à ce niveau peut être fatale à notre BoB !

INFOS SUR LE PROJET

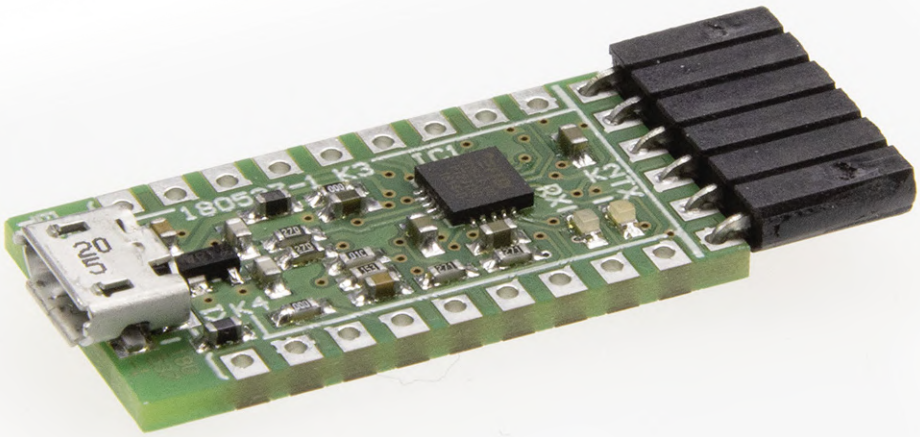
USB
RS232
FT231

débutant
connaisseur
➡ expert

env. 2 h

outillage CMS,
fer à souder à air chaud

env. 25 €



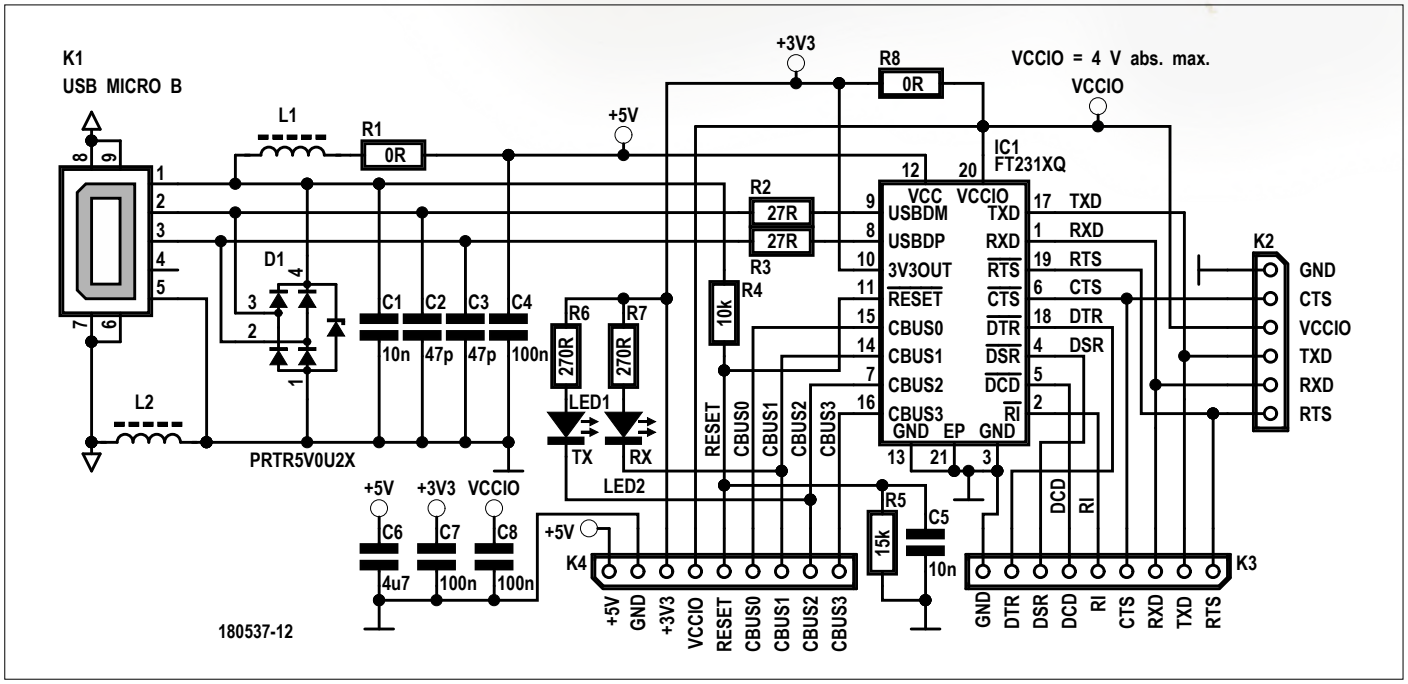


Figure 1. Le schéma du BoB : moins compliqué qu'il n'en a l'air.

Le circuit peut être alimenté via le bus USB (broche 1 du connecteur). C'est l'appareil auquel le BoB est connecté qui fournit la tension de 5 V, et la résistance R1 doit dans ce cas être installée sur le circuit imprimé. Il va de soi qu'aucune autre source de tension externe ne peut être reliée à la broche 1 de K4.
Si on préfère alimenter le circuit à partir d'une source externe (reliée à la broche 1 de K4), il faut retirer la résistance R1. Le diviseur de tension R4/R5, qui sert à détecter la présence d'un appareil connecté à K1, ne demandera à celui-ci qu'un courant d'une intensité de 0,2 mA. Il faut faire très attention ici : le circuit intégré est relié directement à la broche 1 de K4 !
Les E/S du circuit sont quant à elles alimentées en 3,3 V (VCCIO). Cette tension peut être fournie par le régulateur interne d'IC1, dont la sortie (broche 10) est reliée à VCCIO (broche 20) via R8. Cette tension est aussi disponible sur la broche 3 de K4. Le courant maximal pouvant être délivré par le régulateur est de 50 mA.
VCCIO peut également provenir d'une source externe, via la broche 4 de K4 ; dans ce cas, il ne faut pas installer la résistance R8. La tension acceptable varie de 1,8 à 3,3 V, et ne peut en aucun cas excéder 4 V : IC1 serait définitivement grillé…

Pour ce montage, pas question que la main tremble !

Notez que les broches utilisées par l'interface RS-232 résistent à une tension de 5 V ; ceci permet la connexion via l'UART d'appareils alimentés en 5 V.

Configuration

Nous l'avons signalé plus haut, il est possible de configurer quatre E/S d'IC1 (CBUS1 à CBUS4) en fonction de ses propres besoins ; elles sont disponibles sur K4. Le plus simple est d'utiliser l'outil gratuit FTPROG de FTDI [4]. Un manuel très détaillé (en anglais) est disponible sur cette même page, nous n'irons pas plus loin sur ce sujet, d'autant plus que la configuration standard est exactement ce que nous voulions. Attention tout de même : si vous expérimentez un peu trop avec ce logiciel, il est possible que le circuit ne réagisse plus du tout…

Montage

La **figure 2** montre le circuit imprimé à double face que nous avons conçu – et qui est disponible dans l'e-choppe. Il y a de nombreux *vias*, ce qui n'est pas surprenant pour un aussi petit circuit imprimé avec quatre connecteurs.
Si vous n'êtes pas à l'aise avec les CMS, un circuit imprimé sur lequel les composants « difficiles » sont déjà montés est aussi disponible dans l'e-choppe. Si vous êtes un amateur chevronné et que vous voulez câbler tous les composants, vous aurez besoin non seulement de l'outillage adapté, entre autres un fer à souder à air chaud, mais aussi d'une main sûre. Il est vivement conseillé d'agrandir le dessin du circuit imprimé – disponible sur la page du projet [5], pour bien localiser l'emplacement des composants ; le peu de place disponible fait que les composants masquent l'indication de leur emplacement. Ce dessin reprend également les signaux disponibles sur les broches des connecteurs.

LISTE DES COMPOSANTS

Résistances
Toutes CMS 0603
R1, R8 = 0 Ω, 75 V, 100 mW
R2, R3 = 27 Ω, 75 V, 100 mW
R4 = 10 kΩ, 50 V, 100 mW
R5 = 15 kΩ, 50 V, 100 mW
R6, R7 = 270 Ω, 50 V / 100 mW

Condensateurs
Tous CMS 0603
C1, C5 = 10 nF, 20%, 50 V, X7R
C2, C3 = 47 pF, 2%, 50 V, C0G/NP0
C4, C7, C8 = 100 nF, 10%, 50 V, X7R
C6 = 4,7 µF, 10%, 6,3 V, X5R

Inductances
L1, L2 = 330 Ω @ 100 MHz, 1,7 A (Murata BLM18KG331SN1D)

Semi-conducteurs
D1 = PRTR5V0U2X, diodes de protection ESD, CMS SOT-143B
LED1 (TX) = HSMG-C190, verte, CMS 0603
LED2 (RX) = HSMS-C190, rouge, CMS 0603
IC1 = FT231XQ, CMS QFN-20

Divers
K1 = connecteur micro-USB de type B, femelle, CMS pour circuit imprimé
K2 = barrette de 6 picots à souder coudés, au pas de 2,54 mm
K3, K4 = barrette de 9 picots à souder, au pas de 2,54 mm

Circuit imprimé, réf. 180537-1

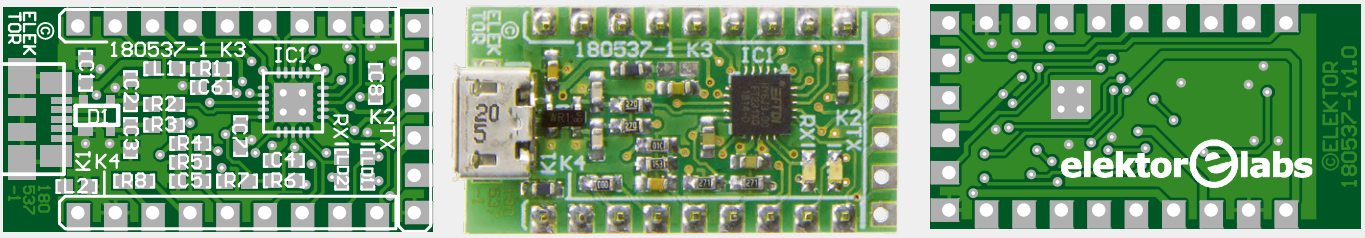


Figure 2. Si l'aventure CMS ne vous tente pas, un circuit imprimé partiellement garni est disponible dans l'e-choppe.

Il convient aussi de monter le connecteur micro-USB avant D1 (protection contre les décharges électrostatiques) : lorsque le circuit D1 est installé, il est pratiquement impossible de réparer un éventuel court-circuit entre les broches de K1 lors de leur soudage.

Conclusion

La liaison RS-232 la plus simple est sans modem, et elle est aussi appelée connexion croisée. La ligne de transmission d'un des deux appareils (TxD) est reliée à la ligne de réception de l'autre (RxD), et vice versa. Les masses sont – bien entendu – aussi reliées. En plus des données (jusqu'à 8 bits), on transmet également un bit de départ, un à deux bits d'arrêt et éventuellement un bit de parité.

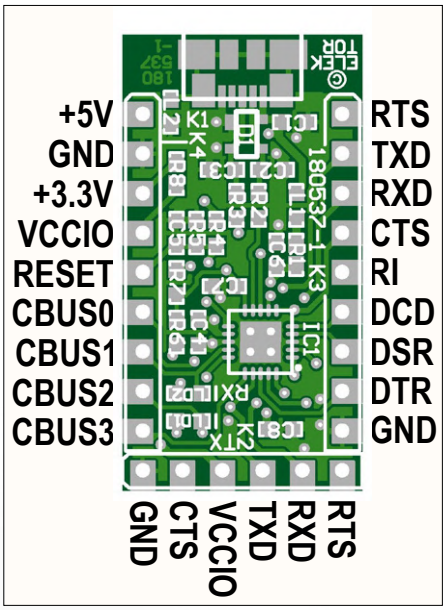
Pour établir la liaison de manière plus sûre, on a recours à une mémoire temporaire (*buffer*), matérielle ou logicielle, et un protocole pour l'échange de données (*handshaking*) évite le dépassement de la capacité.

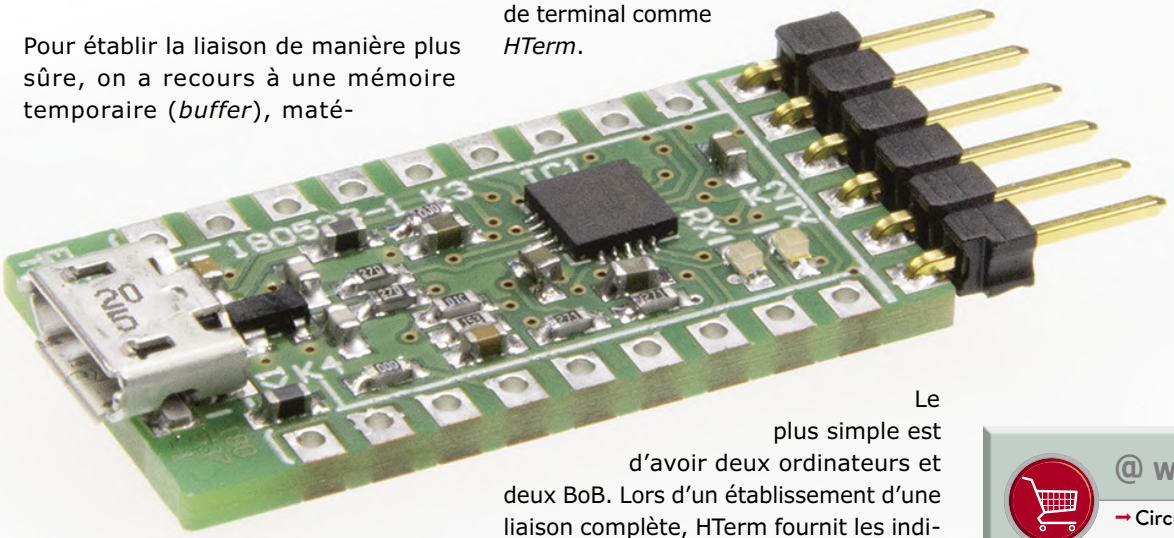
Pour la solution matérielle, on s'appuie sur les signaux DTR, DSR, RTS et CTS. Ce dernier est croisé avec RTS, et DTR avec DSR. S'il y a un modem dans l'un des deux appareils, les signaux DCD et RI sont aussi utilisés, connexion sans croisement.

Pour la commande logicielle (*software flow control*), deux caractères spéciaux sont utilisés pour le démarrage et l'arrêt : XON et XOFF (valeur décimale 17 et 19). Cependant, on ne peut pas employer ces caractères lors d'une transmission de données binaires, sinon il y a un problème.

Pour plus d'informations relatives à la norme RS-232, voir [6] et [7].

Après montage – et vérification très stricte des soudures – on peut tester le BoB avec un programme de terminal comme *HTerm*.





Le plus simple est d'avoir deux ordinateurs et deux BoB. Lors d'un établissement d'une liaison complète, HTerm fournit les indications nécessaires : un « rond » vert indique que DTR et/ou RTS sont aussi actifs de l'autre côté. ⏮

(180537-B-04 – version française : Jean-Louis Mehren)

Liens

[1] « BOB-FT232R, passerelle USB/série », Elektor 09/2011 : www.elektormagazine.fr/110553

[2] Feuille de caratéristiques du FT232XQ : www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT231X.pdf

[3] TN140, note technique pour FT231 : www.ftdichip.com/Support/Documents/TechnicalNotes/TN_140_FT231X%20Errata%20Technical%20Note.pdf

[4] Outil FTPROG : www.ftdichip.com/Support/Utilities.htm#FT_PROG

[5] Page de l'article : www.elektormagazine.fr/180537-04

[6] En savoir plus sur le RS232 : www.commfront.com/pages/3-easy-steps-to-understand-and-control-your-rs232-devices

[7] En savoir plus sur le RS232 : www.codrey.com/embedded-systems/rs232-serial-communication/

[8] Pilote FTDI : www.ftdichip.com/Drivers/VCP.htm

[9] Pilote FTDI : www.ftdichip.com/Drivers/D2XX.htm

[10] Page du projet dans le labo d'Elektor : www.elektormagazine.fr/labs/remake-110553-usb-rs232-converter-ft232-bob

vol tous azimuts

l'électronique par monts, maux et merveilles

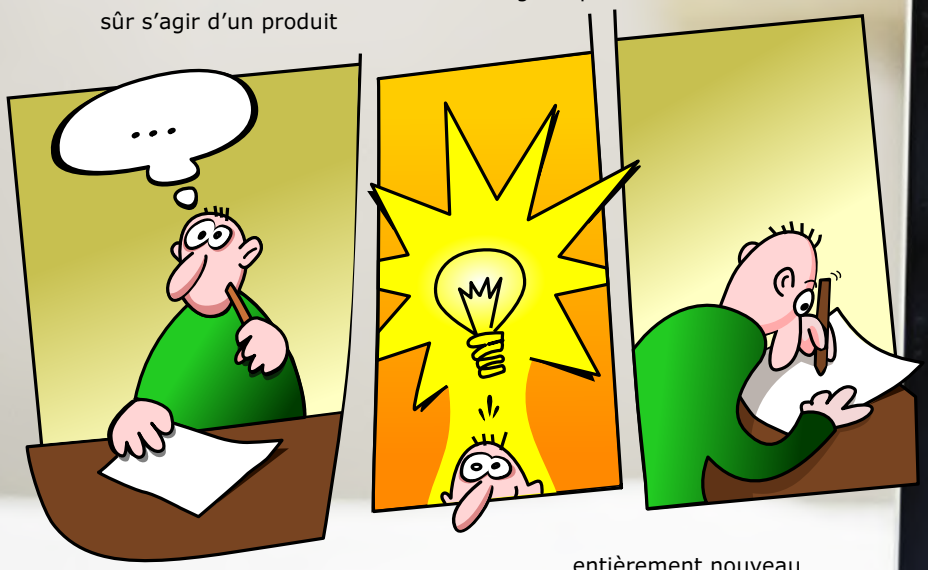
Clemens Valens

DE L'IDÉE AU PRODUIT 1RE PARTIE

On nous demande parfois : comment créez-vous un « produit » ? Avant d'aborder ce sujet intéressant, mais complexe, envisageons d'abord cette question sous un autre angle : pourquoi élabore-t-on un produit ? Dans un but commercial, sinon, généralement, un prototype suffit. Vue ainsi, la vie d'un produit ne commencera réellement qu'au moment où il sera prêt à être vendu. Sa création implique donc également celle d'un ou plusieurs canaux de vente, sans oublier les détails tels que l'emballage, la livraison, la facturation, le marketing et le service après-vente. Vous pouvez établir un canal de vente en vendant votre produit à la sauvette sur un parking, en créant votre propre boutique en ligne, en persuadant une chaîne de supermarchés de l'accueillir dans ses rayons, ou encore en le référençant sur Amazon ou Alibaba. Avant même d'être vendable, un produit doit toutefois être fabriqué, testé et certifié. Il faut aussi écrire son manuel de l'utilisateur. Sa production ne sera possible qu'après avoir mis au point un prototype dûment fonctionnel et validé, conçu avec ladite production en ligne de mire, autrement dit votre activité devra être industrialisée. Comme vous le constatez, la création d'un produit ne se résume pas à avoir une bonne idée.

AVANT TOUT, PRODUIRE UNE IDÉE

Il vous faut trouver une idée de produit (ou de service, mais je m'en tiendrai ici à des produits tangibles) susceptible de séduire un cercle d'acheteurs suffisamment large. Il peut bien sûr s'agir d'un produit



entièrement nouveau, mais aussi de quelque chose que vous avez déjà créé par ailleurs. De nombreux projets menés à bien dans un but particulier peuvent en effet être adaptés à un champ d'application plus large ou différent, et donc se révéler de bons produits candidats.

LE BON PLAN : LE BUSINESS PLAN

Vous avez trouvé une idée de produit intéressant ? Si votre âme d'ingénieur imagine déjà un moyen de le fabriquer en série, réfrénez vos ardeurs. Commencez par étudier le marché de façon objective. Vous trouvez peut-être votre idée géniale, mais qu'en pensent les autres ? Et parmi eux, combien ont déjà eu la même idée ? Le produit existe-t-il déjà ou non ? Si non, est-ce parce que votre idée est vraiment originale, ou bien est-ce parce que tous ceux qui l'ont déjà eue ont renoncé après avoir découvert qu'il n'y avait aucun marché pour elle ? Si le produit existe déjà, le marché visé laisse-t-il place à de la concurrence ? Et de quoi aurez-vous besoin ? Combien de temps, d'argent, d'outils, de collaborateurs ? Comment comptez-vous commercialiser votre produit ? Bref, vous devriez d'abord écrire un plan d'affaires. L'exercice n'est pas facile, les modèles et exemples de plans d'affaires que l'on trouve sur la toile vous en convaincront. En l'établissant, ne faites pas l'erreur de supposer que chacun voit le monde comme vous le voyez. Essayez aussi d'en dresser un couvrant plusieurs années. Où aimeriez-vous en être dans, disons, cinq ans ? Et comment y parviendrez-vous ? Pour toutes ces questions, les Chambres de commerce peuvent vous aider à trouver des réponses

AVEZ-VOUS VRAIMENT L'ÉTOFFE D'UN PDG ?

Supposons que vous soyez parvenu à écrire un plan d'affaires qui tienne la route. Il faut maintenant le confronter à la réalité. Cela pourrait signifier quitter votre travail, investir toutes vos économies, voire mettre sur le feu un motif de divorce ; êtes-vous prêt à tout ça ? À moins que soirées et week-ends ne suffisent à votre projet ? Gardez également à l'esprit qu'une fois votre entreprise viable, votre travail d'ingénieur pourrait bien se résumer à gérer une équipe, trouver des clients, chercher de l'argent, et plus généralement à faire des « trucs » d'homme d'affaires. Là encore, êtes-vous vraiment prêt à laisser à d'autres ce travail d'ingénieur que vous aimez tant ? Et d'ailleurs, feriez-vous un bon dirigeant d'entreprise ? Peut-être aurez-vous besoin d'associés. Évitez les amis si vous souhaitez qu'ils le restent, et à tout prix les partenariats 50/50 si vous ne voulez pas finir éjecté de votre propre entreprise ; le plus prudent est de rester majoritaire.

ET ENSUITE ?

Si la lecture de cet article n'a pas douché vos rêves d'entrepreneuriat, ne ratez pas les prochains épisodes, ils vous apprendront comment transformer une idée en produit « qui peut vraiment être produit ». Nous parlerons de produits électroniques puisqu'il s'agit du fonds de commerce d'Elektor, mais ce qui sera dit vaudra également pour d'autres types de produits. Quoi qu'il en soit, n'abandonnez jamais une idée, et laissez votre fer à souder l'exprimer : on ne sait jamais ce qu'il peut en sortir.

(180567-03 – version française : Hervé Moreau)

Alias, l'assistant anti-barbouze

Alias se monte sur un assistant personnel intelligent.
Crédit photo : Bjørn Karmann

Par principe, les assistants personnels intelligents tels que Google Home ou Alexa d'Amazon écoutent en permanence leur environnement dans l'attente de commandes. Le respect de la vie privée peut à cet égard poser problème puisque ces appareils sont connectés à l'internet, et vous ne devriez pas les installer si cet enjeu vous préoccupe. Il existe toutefois un moyen de rendre ces assistants moins intrusifs. Conçu par le designer danois Bjørn Karmann, le projet Alias s'inspire de la façon dont les champignons cordyceps et certains virus parasitent le cerveau de leur hôte pour en prendre le contrôle. Alias est plus précisément un dispositif qui rend les oreilles des assistants vocaux moins indiscrètes en brouillant leur microphone avec un bruit parasite permanent. Alias n'écoute que ses propres mots activateurs et n'est pas connecté à l'internet. Lorsqu'il détecte un mot activateur valide, il coupe le son parasite et passe le relais à l'assistant, qui dès lors exécute normalement les commandes transmises. Alias peut être programmé pour passer toutes sortes de commandes vocales à l'assistant, ce qui offre de nombreuses possibilités d'utilisations nouvelles. Guide d'assemblage, instructions d'installation et code source sont disponibles sur GitHub.

http://bjoernkarmann.dk/project_alias

LoRa gagne l'espace

Lacuna Space a conclu la première phase de test de sa passerelle spatiale à technologie LoRa en avril 2019 grâce à un satellite lancé depuis le centre spatial indien Satish Dhawan. Les systèmes de tests déployés autour du globe ont montré l'efficacité de la communication spatiale LoRa. Trois satellites supplémentaires rejoindront le premier au second semestre 2019 pour former une constellation opérationnelle.

La passerelle spatiale LoRa était logée dans un satellite CubeSat 6A.
Crédit photo : Nano Avionics

Outil de labo indispensable

La compagnie d'un chat peut réduire le stress et l'anxiété, faire baisser la pression artérielle et le rythme cardiaque, et par là même réduire le risque de maladie cardiaque. La recherche scientifique le confirme. Une étude de 2008 a même établi que la probabilité de mourir d'une crise cardiaque était 30 % plus élevée chez les personnes n'ayant pas de chat que chez celles en possédant un. Afin de réduire le stress au travail et augmenter la productivité, une entreprise informatique japonaise a adopté des chats et encouragé ses employés à prendre soin d'eux, offrant même une aide financière à ceux qui décideraient d'en emmener un chez eux. Un autre exemple intéressant est cet immeuble de bureaux à Las Cruces, Nouveau-Mexique, où une « félinothèque » permet aux salariés d'emprunter un chaton pour se relaxer. Ayant eu des chats toute ma vie, je ne peux que confirmer leurs qualités antistress. Je recommande donc chat-leureusement leur présence au labo ou au bureau. La seule chose à surveiller, c'est leur urine, nauséabonde à souhait et de surcroît assez corrosive pour rendre un circuit imprimé inutilisable en un rien de temps, pour ne pas dire en un jet. Si bien sûr vous ne pouvez pas sentir les chats, les effets pourraient être moins bénéfiques.

Chats de bureau : https://youtu.be/vd21SH6zAX4

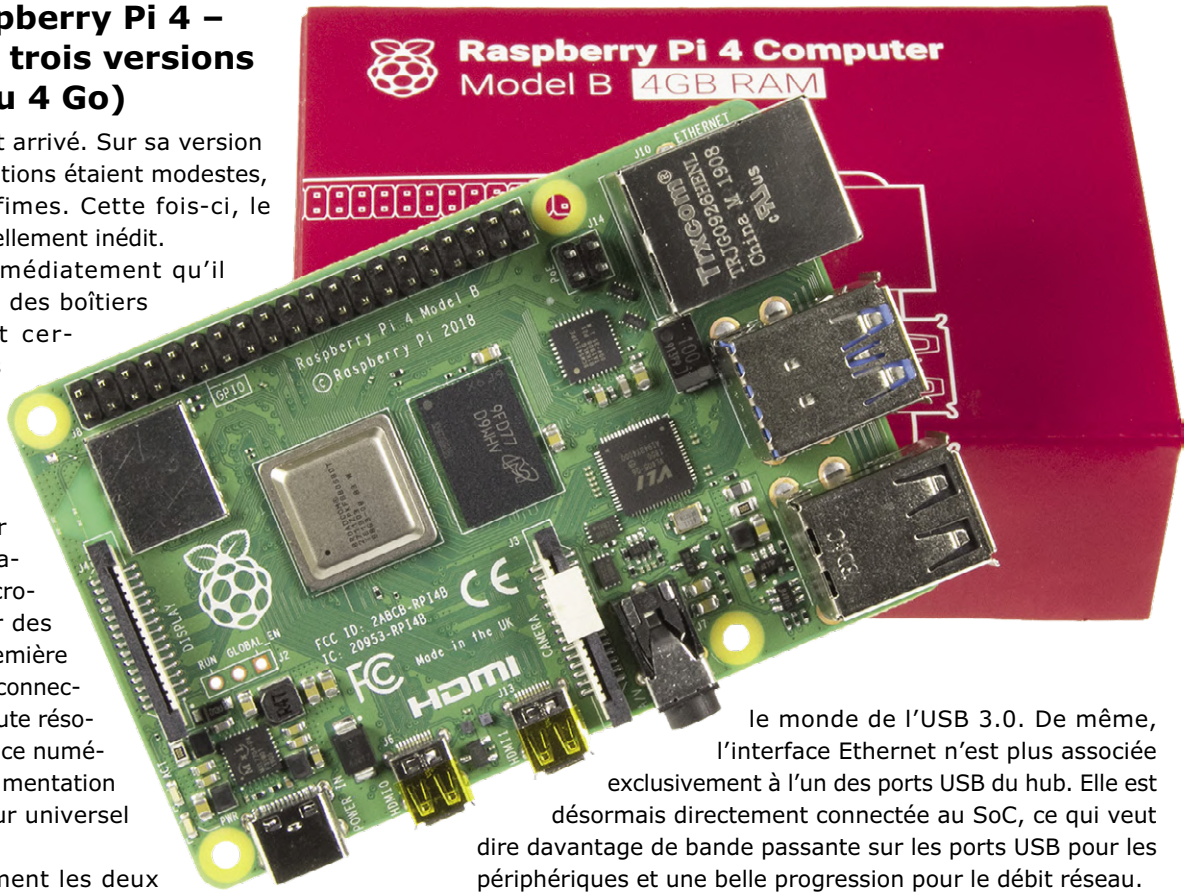
Chatons à rendre à 18 h : https://youtu.be/OKq22oF4FiQ

PETITE SÉLECTION DE LA BOUTIQUE EN LIGNE

Nouveau Raspberry Pi 4

...livre « SDR Hands-On », compteur Geiger à monter soi-même

Nouveau Raspberry Pi 4 – disponible en trois versions (1 Go, 2 Go ou 4 Go)



Le Raspberry Pi 4 est arrivé. Sur sa version précédente, les évolutions étaient modestes, pour ne pas dire infimes. Cette fois-ci, le Raspberry Pi 4 est réellement inédit.

Vous remarquez immédiatement qu'il ne s'adapte à aucun des boîtiers habituels. En effet certains changements majeurs portent sur la connectique pour l'écran et l'alimentation. Il y a maintenant un connecteur USB-C pour l'alimentation et deux ports micro-HDMI pour raccorder des écrans 4K. Pour la première fois, il est possible de connecter deux écrans en haute résolution grâce à l'interface numérique et d'assurer l'alimentation à partir du connecteur universel USB-C.

Vous noterez également les deux ports USB de couleur bleue : le RPi 4 dispose de ports USB 3.0 intégrés ainsi que d'une interface Ethernet native, ce qui règle définitivement tout problème de transfert de données entre un périphérique de stockage de masse et le réseau. Tout cela grâce au nouveau SoC BMC2711, processeur pivot du RPi 4, qui arbore quatre cœurs Cortex-A72 cadencés à 1,5 GHz max. et jusqu'à 4 Go de RAM.

Le RPi 4 conserve certaines fonctions bien établies : le port RCA, le connecteur pour l'écran et l'interface caméra. Également inchangée et rétrocompatible, la barrette à 40 broches est implantée sur le circuit imprimé. Les HAT (extensions matérielles) peuvent donc être connectés comme à l'accoutumée, au moins matériellement.

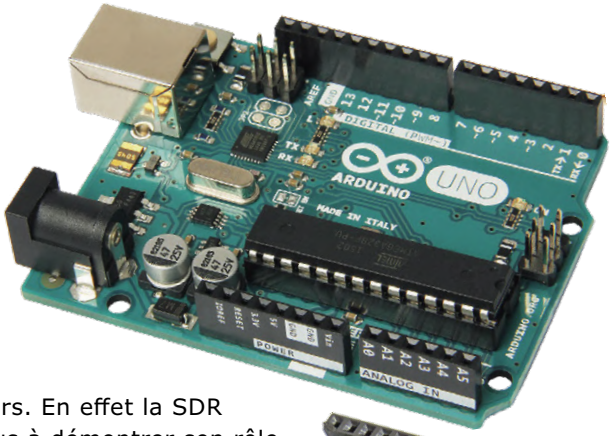
Enfin on dispose maintenant de ports USB dignes de ce nom, et non plus d'une simple connexion USB 2.0 avec une affligeante liaison montante vers le SoC. Avec les deux ports de couleur bleue et le nouveau hub USB, le RPi 4 est entré dans le monde de l'USB 3.0. De même, l'interface Ethernet n'est plus associée exclusivement à l'un des ports USB du hub. Elle est désormais directement connectée au SoC, ce qui veut dire davantage de bande passante sur les ports USB pour les périphériques et une belle progression pour le débit réseau.

Sur les versions antérieures du RPi, la possibilité d'une accé-

@ WWW.ELEKTOR.FR

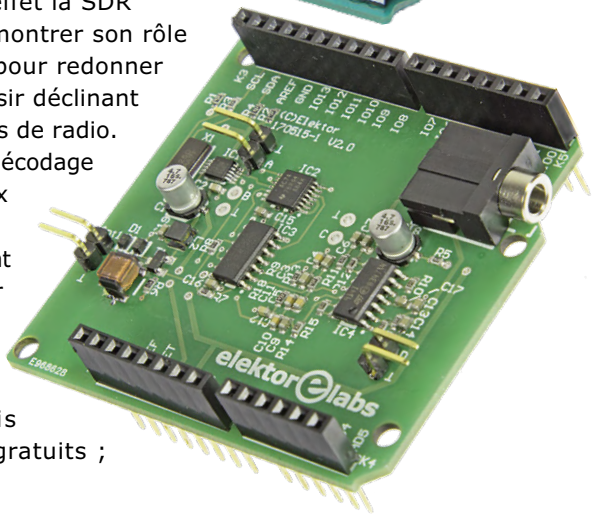
- Raspberry Pi 4 avec 1 Go de RAM
  www.elektor.fr/18966
- Raspberry Pi 4 avec 2 Go de RAM
  www.elektor.fr/18965
- Raspberry Pi 4 avec 4 Go de RAM
  www.elektor.fr/18964
- Boîtier officiel du RPi 4
  www.elektor.fr/18963
- Alimentation officielle du RPi
  www.elektor.fr/18962

lération 3D pour le Bureau manquait également. Avec le RPi 4 et le pilote de la partie graphique du processeur VideoCore VI, cette fonction est enfin de retour. Celle proposée (qui fonctionne parfaitement) est une accélération 3D en mode « fenêtré ».
Pour tous ceux qui possèdent déjà un HAT RPi ou envisagent d'en installer un, cette question est incontournable : pourrai-je encore connecter des produits additionnels sur le RPi 4 ? Oui, rassurez-vous. En revanche, s'il s'agit d'un écran TFT avec interface HDMI, il vous faudra au moins un nouveau câble.
Les évolutions tant attendues sont là : plus de vitesse pour les ports USB, le réseau et l'unité centrale, et plus de mémoire RAM. Tout ceci a évidemment un prix, en espèces sonnantes et trébuchantes, mais aussi en énergie consommée ! À partir des modèles d'entrée de gamme dotés de 1 Go de RAM à un tarif pratiquement identique au RPi 3 B+, le prix augmente en fonction de la quantité de mémoire désirée. Parallèlement, la consommation d'énergie et la dissipation thermique progressent aussi.



Livre « SDR Hands-on Book »

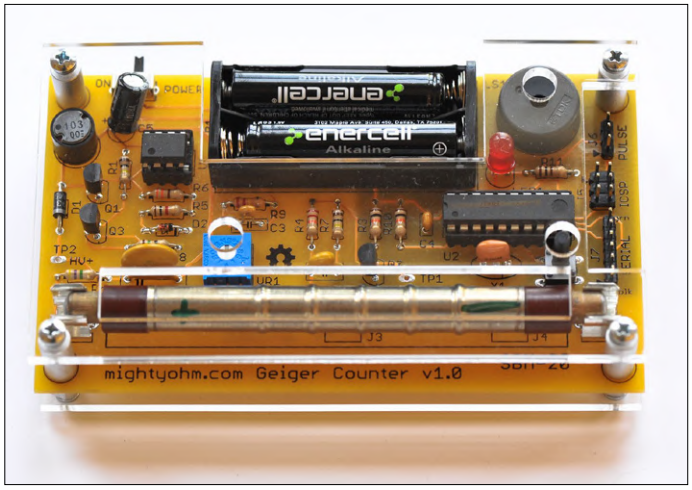
Dans la préface enthousiaste de son ouvrage en anglais « SDR Hands-on Book », l'auteur Burkhard Kainka explique pourquoi la radio logicielle (ou SDR – *software defined radio* en anglais) rencontre le succès bien au-delà de la communauté des radioamateurs. En effet la SDR n'a plus à démontrer son rôle exceptionnel pour redonner vigueur au loisir déclinant des passionnés de radio. L'accord et le décodage des signaux radio en ondes courtes reposent aujourd'hui sur un minimum de matériel et des logiciels à la fois puissants et gratuits ;

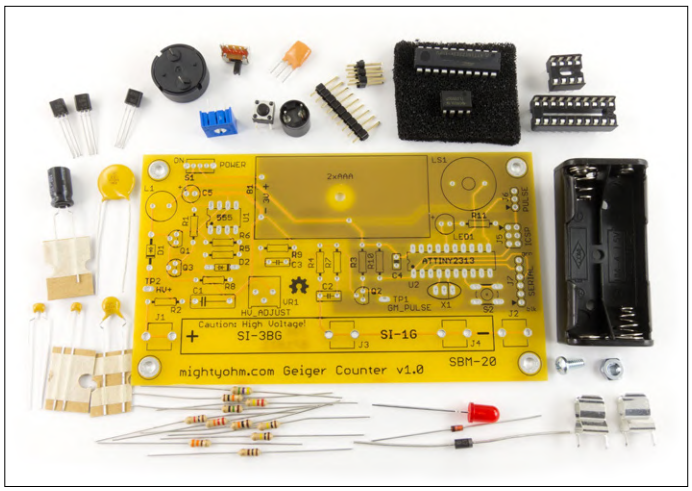


c'est pourquoi les nouveaux passionnés de radio ont adopté la radio logicielle.
L'époque des installations occultes en ondes courtes, formées de bobines complexes, de composants radio et de techniques de construction spécialisées est révolue. Aujourd'hui, une simple carte d'interface radio et un mélangeur IQ suffisent, associés évidemment… à un logiciel, installé en général sur un ordinateur portable doté d'une carte son de qualité. Toute la « matière grise » de la SDR est donc confiée au code.
Pour vous frayer un chemin vers les arcanes de la SDR, procédez ainsi : (1) montez le matériel en vous attachant à bien comprendre son fonctionnement, (2) exécutez le logiciel. L'ouvrage commence donc par présenter un *shield* (carte enfichable) pour l'Arduino Uno.
L'auteur aborde en détail le schéma et la théorie de fonctionnement du *shield* SDR.
Il est possible (mais optionnel) d'enficher le *shield* d'écran LCD d'Elektor au-dessus du *shield* SDR. Cette configuration permet d'afficher la fréquence SDR actuelle. C'est pratique et bon marché. Et en plus, l'Arduino a un peu plus à se mettre sous la dent !
Le chapitre 2 poursuit avec la procédure d'installation de l'un des logiciels SDR les plus anciens, (toujours) performant, et gratuit de surcroît : G8JCF. Le livre propose une procédure pas à pas pour relier le logiciel G8JCF à l'Arduino couplé au *shield* SDR, et faire tourner le micrologiciel de l'auteur. Un petit exercice de mise en route propose d'accorder l'appareil sur une fréquence de 7.000 kHz dans la bande des 40 m.
Le chapitre le plus efficace pour promouvoir la SDR auprès des novices concerne cet accessoire indispensable qu'aucun logiciel ne peut remplacer : l'antenne. Grâce à sa démarche progressive (propriétés des antennes de réception en ondes courtes, accord du récepteur), l'auteur calme les attentes des radioamateurs en appartement concernant la réception de signaux venant de l'autre bout du monde avec un simple fil de fer.
Ce livre est incontournable pour découvrir l'univers fascinant des ondes courtes avec un système SDR, constitué d'une carte Arduino, d'un *shield* SDR et du logiciel G8JCF, produits largement diffusés et très peu coûteux, voire gratuit concernant le logiciel.

Construire son propre compteur Geiger

Un compteur Geiger sert à détecter les rayonnements bêta et gamma. Ils sont émis par le césium 134 (^{134}Cs) et le césium 137 (^{137}Cs), isotopes radioactifs du césium, libérés dans l'environnement par les essais d'armes nucléaires et les accidents nucléaires, ainsi que par l'uranium (U) et le thorium (Th), deux des éléments radioactifs les plus courants sur Terre. Leurs chaînes de désintégration entraînent la production de radon (Rn), normalement gazeux, mais facilement inhalé, et donc dangereux pour la santé. À cause de ces risques relatifs de rayonnements, vous voudrez peut-être surveiller votre environnement et investir dans un détecteur comme le compteur Geiger MightyOhm.
Le détecteur produit un « clic » caractéristique ainsi qu'un éclair lumineux chaque fois que le gaz présent à l'intérieur du tube Geiger-Müller (G-M) est ionisé par le rayonnement appliqué. Pour cela, une tension de plusieurs centaines de volts sert à polariser le tube G-M. L'ionisation produit une courte impulsion de courant mesurable et transformable en son (les fameux « clics »).





Vous savez maintenant (à peu près) comment fonctionne un compteur Geiger et à quoi il sert. Construisez-en un avec le kit MightyOhm. Le kit est 100% ouvert (code source et matériel). Il comprend une LED et un haut-parleur piézoélectrique pour « voir »et « entendre » les niveaux de radioactivité.
La carte comporte des barrettes de connexion pour la communication série (signaux de 3,3 V), la programmation *in situ* (ICSP) du µC AVR et la production d'impulsions.
Le kit du compteur Geiger MightyOhm est facile à monter, et offre une bonne précision de mesure de la radioactivité. De plus, ses capacités d'enregistrement de données en font un instrument très pratique.

180689-E-04

PROJET DE LECTEUR

réduction du bruit de sortie de l'ICL7660

avec un filtre en Pi du second ordre

Hesam Moshiri

Certains circuits comme les amplificateurs à base d'ampli-op sont sensibles aux perturbations des lignes d'alimentation, ce qui oblige les concepteurs à limiter au minimum les niveaux de bruit. Nous cherchons ici à supprimer efficacement le bruit de sortie de l'ICL7660 tout en apprenant comment connecter la partie convertisseur de tension au circuit principal pour une meilleure conformité CEM. Tout cela pour éviter que les courants de commutation du « 7660 » n'affectent en rien le circuit principal.

Fournir une ligne d'alimentation négative propre est un défi imprévu dans beaucoup d'applications alimentées par une seule batterie. L'ICL7660/MAX1044 est une puce bien connue pour « faire chuter la tension » et réaliser facilement une ligne d'alimentation négative. Une seule puce avec quelques composants passifs semble un bon choix, mais n'exempte pas de quelques défis de conception. Bien sûr cette puce, un convertisseur de tension monolithique à capacités commutées, a un taux de conversion de tension efficace, mais malheureusement elle ajoute un bruit de commutation à la tension de sortie. Nous présentons ici une façon d'obtenir une tension de sortie vraiment propre à partir du 7660.

Filtre analogique passif

Le filtre analogique est un module électronique important avec de nombreuses applications dans le traitement du signal. On l'utilise aussi comme module de préfiltrage dans beaucoup de convertisseurs analogique/numérique (CA/N).
Il existe trois principaux types de filtres : passe-haut, passe-bas et passe-bande/coupe-bande. Dans notre application, le filtre doit être simple, pas cher et ne pas limiter du tout le courant de sortie. De plus, il ne doit pas introduire d'instabilité ou de chute significative de tension à la sortie.

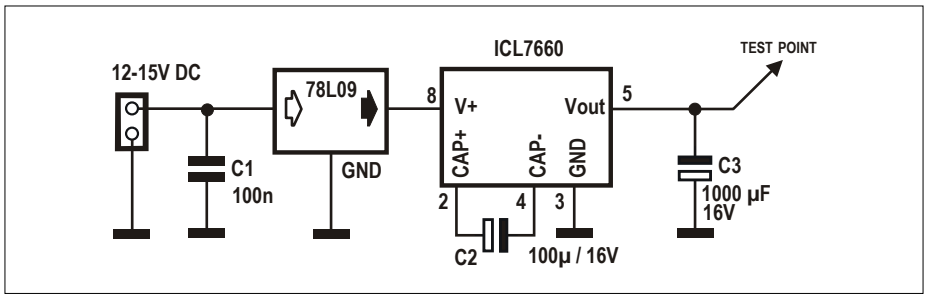


Figure 1. Puce ICL7660 configurée en inverseur de tension.

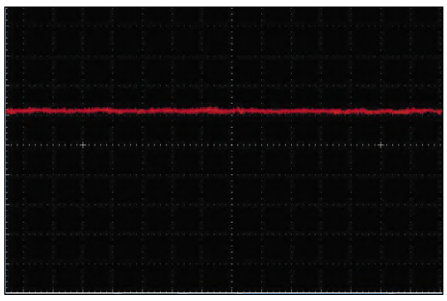


Figure 2. Mesure du bruit de fond de l'oscilloscope (sans sonde connectée, 2 mv/div).

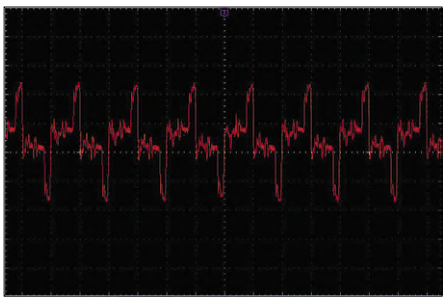


Figure 3. Le bruit à 50 Hz induit par le secteur est inévitablement capté par la sonde de l'oscilloscope.

Bruit en provenance de l'ICL7660

La **figure 1** montre l'ICL7660 dans une configuration typique « d'inverseur ». Il est configuré ici pour produire −9 V en sortie. Les valeurs des condensateurs du convertisseur (C2 et C3) ont été choisies pour optimiser une application pratique. Avant de connecter l'oscilloscope au circuit, vérifions d'abord le bruit interne de l'oscilloscope dans le contexte de laboratoire. La **figure 2** montre le bruit de fond interne de l'oscilloscope en mode monocanal, avec une échelle de 2 mV/div et une bande passante limitée à 20 MHz. Il présente un décalage d'environ 2 mV CC. Avec la sonde connectée, l'oscilloscope capte un niveau de ronflement (50 Hz) significatif, ce que montre la **figure 3**. Une méthode pour éliminer un bruit de type ronflement et évaluer les niveaux de bruit ambiants avec une sonde consiste à y connecter une batterie. La batterie est

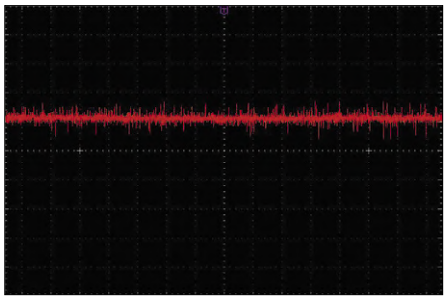


Figure 4. Le bruit d'ambiance avec une batterie connectée ; réglage de la sonde ×1 ; entrée CA sur l'oscilloscope.

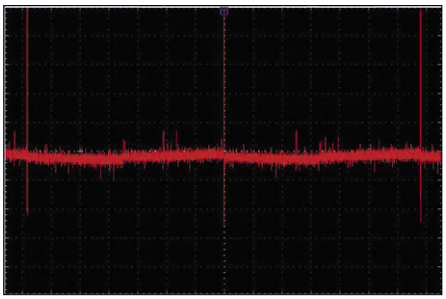


Figure 5. Tension de sortie de l'ICL7660 sans filtrage additionnel ; 2 mV/div.

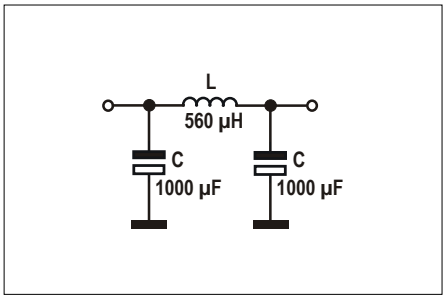


Figure 6. Filtre en Pi passe-bas du premier ordre, pour commencer.

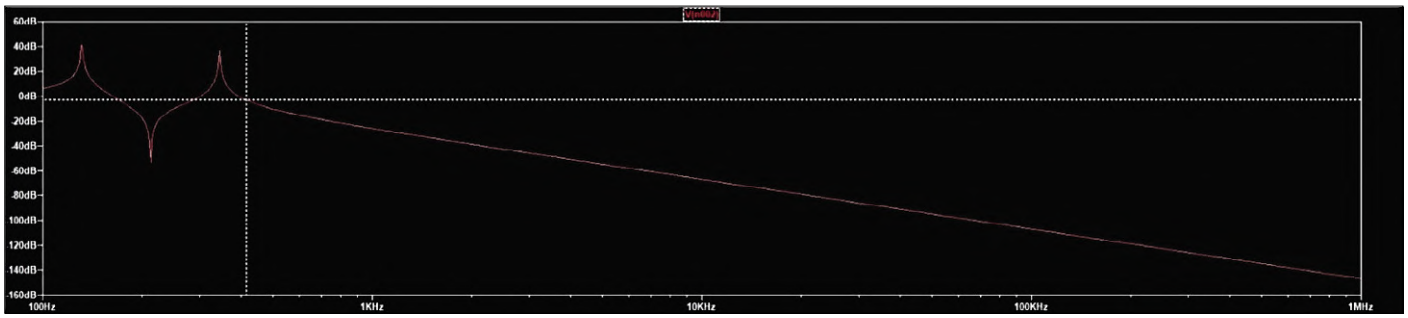


Figure 7. Réponse en fréquence du filtre du premier ordre.

une source pure de tension continue, donc le bruit observé sur l'entrée CA sera dû aux perturbations qui existent dans l'environnement de test, et apparaîtra naturellement sous une forme alternative. Par conséquent, l'entrée de l'oscilloscope doit être réglée sur le mode CA. La **figure 4** montre le bruit réel présent lorsque la batterie est connectée à la sonde.

Après cette étape, nous sommes prêts à réaliser la première phase du circuit de conversion de tension à ICL7660 et à connecter la sonde de l'oscilloscope au « point de test ». Le circuit doit être alimenté par une batterie, car nous voulons être absolument sûrs que le circuit est alimenté par une source sans bruit, et que ce qui est mesuré au-dessus et au-dessous du bruit sur la figure 4 est bien le bruit de commutation créé par le circuit.

La **figure 5** montre le signal du « point de test » sur l'oscilloscope (2 mV/div). Malgré un sérieux filtrage en sortie avec un condensateur de 1.000 µF, la ligne d'alimentation est toujours infestée de bruit. Les mesures à l'oscilloscope montrent que la fréquence du bruit de commutation principal est d'environ 7,3 kHz avec une variation d'amplitude pouvant atteindre 15 mV_c. Pour obtenir une tension continue pure (autant que possible), nous devons utiliser un filtre passe-bas et bloquer tout le bruit à cette fréquence et au-dessus.

Nous avons le choix ici : filtres RC et LC. J'écarte les filtres RC, car la composante « résistance » limite inévitablement le courant et introduit une chute de tension, particulièrement dans les filtres d'ordre élevé. On préfère donc le filtre de type LC. En pratique, un filtre à trois éléments se comporte plus efficacement, donc nous conservons le condensateur C3 et ajoutons un étage de filtre LC à la sortie comme le montre la **figure 6**. C'est ce qu'on appelle un CLC ou filtre en « Pi ».

La plupart des filtres sont caractérisés par leur fréquence de coupure. Par exemple, un filtre passe-bas bloque les signaux au-dessus de sa fréquence de coupure tout en laissant passer les signaux qui restent en dessous de ce « seuil ». Nous pouvons utiliser un simulateur de circuit comme LTSpice pour prédire le comportement du filtre. Nous pouvons aussi examiner l'efficacité du filtre en modifiant les valeurs des composants L et C. Dans notre cas, nous devrons choisir des valeurs qui produisent une fréquence de coupure basse avec un filtrage plus « raide » juste avant le bruit. La capacité du condensateur de sortie (C3) était de 1.000 µF, donc continuons avec cette valeur qui n'est toutefois pas obligatoire. Surtout si le circuit imprimé est de grande taille, vous pouvez prendre des valeurs de capacité de votre choix et simuler la réponse du filtre avant de passer à la conception du circuit imprimé et à son assemblage.

La **figure 7** montre la réponse en fréquence du filtre. La valeur de l'inductance est fixée à 560 µH. Le balayage de fréquence va de 100 Hz à 1 MHz.

La fréquence de coupure du filtre est la fréquence sur l'axe des X qui croise le niveau −3 dB sur l'axe des Y.

LTSpice effectue allègrement les calculs illustrés sur la **figure 8**. La fréquence de coupure est d'environ 414 Hz et le

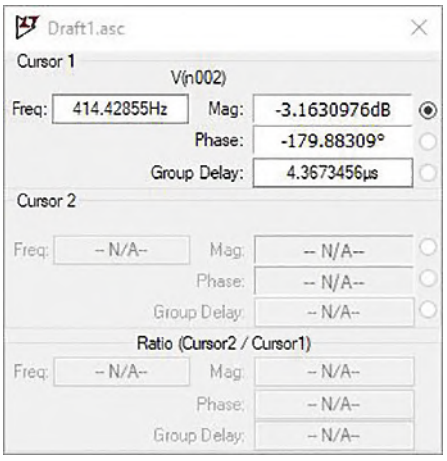


Figure 8. Fréquence de coupure du filtre et déphasage selon LTSpice.

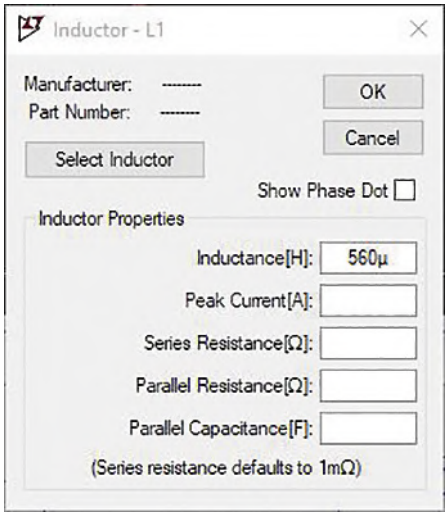


Figure 9. Paramètres modifiables d'une inductance dans LTSpice.

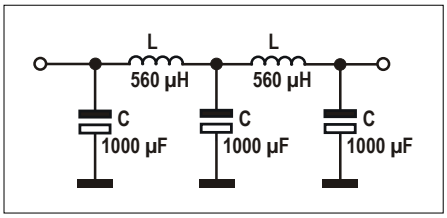


Figure 10. Filtre en Pi passe-bas du second ordre avec les valeurs choisies.

déphasage de presque −180°. Notez que ces calculs résultent de composants théoriques idéaux. Si vous voulez faire des simulations plus réalistes, modifiez les valeurs des composants avec les paramètres des fiches techniques. La **figure 9** montre les paramètres fournis par LTSpice pour l'inductance, comme la résistance série, le courant de crête, etc. Nous avons un filtre CLC ou en Pi du premier ordre. Nous pouvons rendre ce filtre plus puissant en augmentant l'ordre du filtre. Nous pouvons ainsi construire un filtre du second ordre avec les mêmes valeurs de composants. La **figure 10** montre le schéma du circuit.

Si nous examinons le comportement du filtre dans LTSpice, il est clair que le filtre du second ordre fournit une réponse plus raide et plus stable. En termes plus simples, il fait mieux le travail de filtrage. Le balayage de fréquence de la **figure 11** montre la réponse du filtre du second ordre (en vert) comparée à celle du filtre du premier ordre (en rouge).

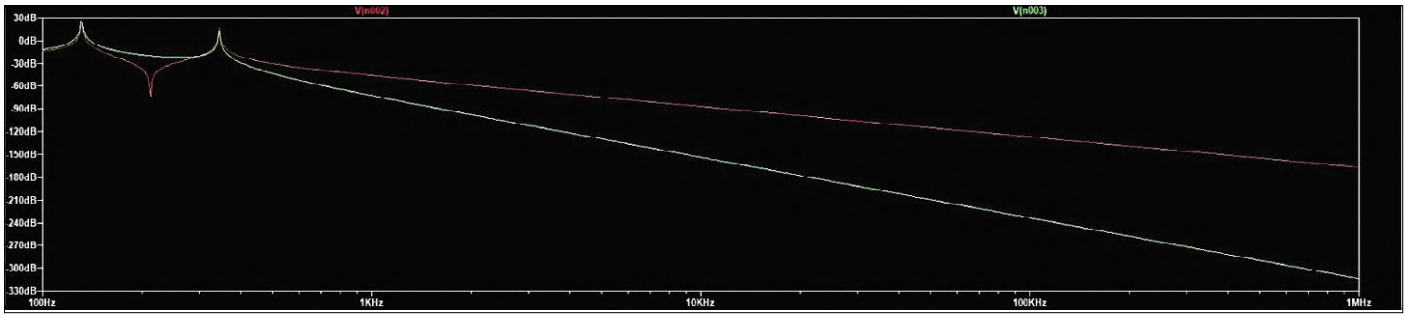


Figure 11. Réponse du filtre en Pi du second ordre (ligne verte).

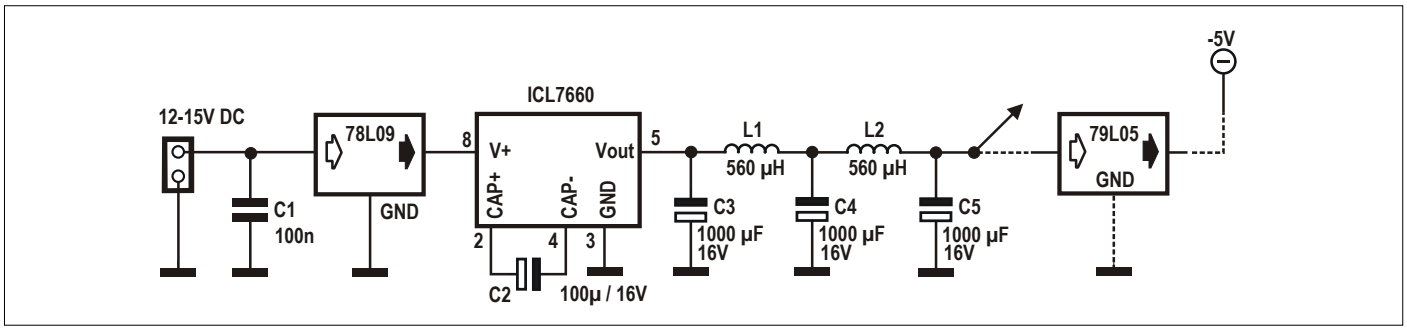


Figure 12. Schéma complet du l'inverseur de tension à 7660, y compris un régulateur de tension optionnel.

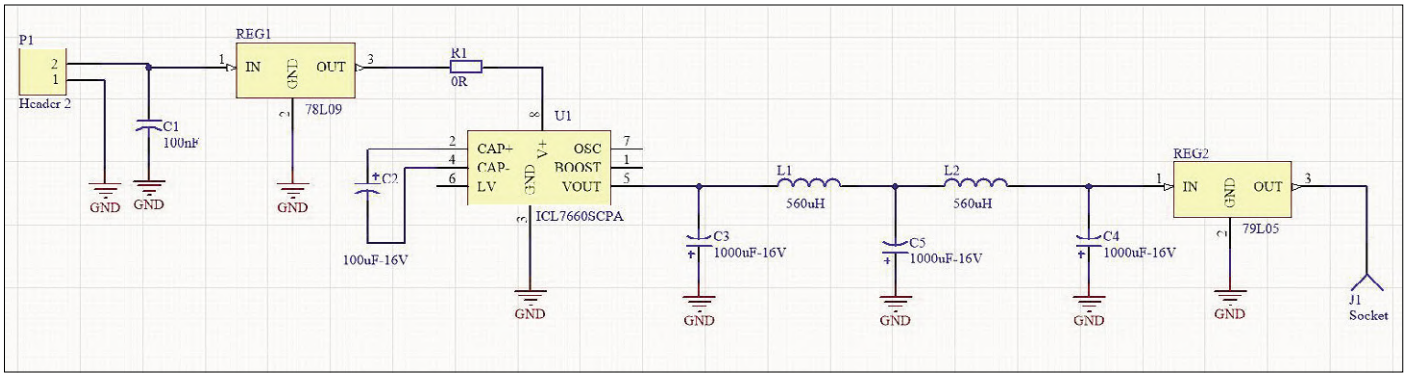


Figure 13. Schéma du circuit de conversion d'alimentation à faible bruit réalisé avec Altium Designer.

En pratique

Il est temps maintenant de tester le circuit convertisseur à ICL7660 avec le filtre mis au point. La **figure 12** montre le circuit complet. Il faut noter que la tension de sortie de l'ICL7660 n'est pas régulée. Par conséquent, s'il vous faut une tension régulée stable, comme −5 V, il est toujours sage d'utiliser un régulateur de tension négative à la sortie du filtre, comme indiqué.

La façon correcte de réaliser et tester le circuit est de concevoir son propre circuit imprimé. J'ai utilisé Altium Designer pour cela. J'ai aussi eu recours au moteur de recherche de composants SamacSys (greffon pour Altium) pour trouver les empreintes de composants. L'utilisation de ce service réduit significativement le temps de création du circuit imprimé en éliminant la tâche répétitive de conception de la bibliothèque de schémas et d'empreintes.

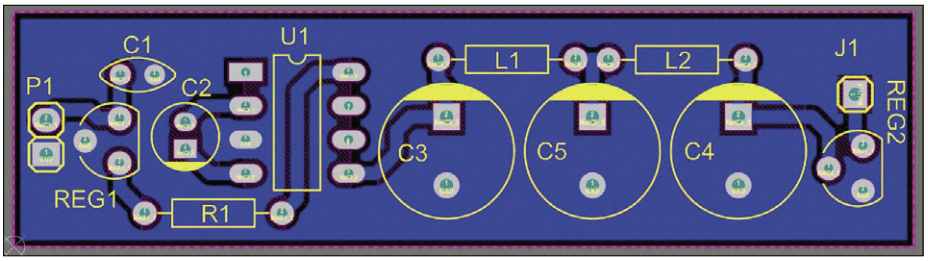


Figure 14. Circuit imprimé conçu pour le circuit convertisseur avec Altium Designer.

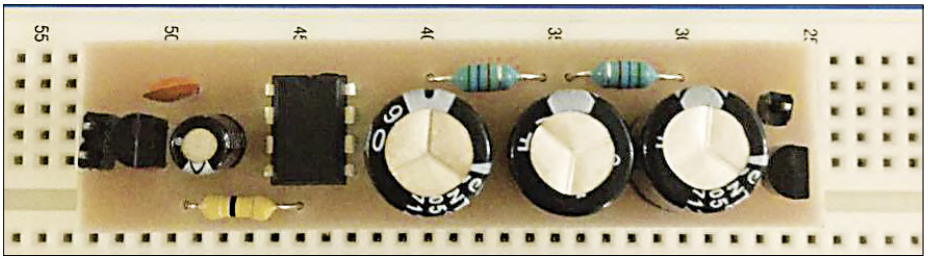


Figure 15. Circuit imprimé garni.

La **figure 13** montre le schéma dessiné dans Altium et la **figure 14** le circuit imprimé correspondant. Les fichiers de conception du circuit imprimé sont en téléchargement gratuit en [1]. La première astuce de conception d'un circuit imprimé est d'utiliser un bon plan de masse en cuivre, c.-à-d. remplir les zones libres. Le plan de masse aide à réduire le bruit et les transitoires. La résistance R1 est une résistance de 0 Ω utilisée ici pour éviter toute coupure inutile du plan de masse au-dessous du condensateur C2. Le solide plan de masse maximise la zone à basse impédance de passage du courant dans le circuit.

Selon les directives de CEM, une alimentation à découpage (*SMPS*) ne doit pas partager une masse commune avec le reste du circuit. Par conséquent, la connexion entre la masse de la SMPS et le plan de masse du circuit principal ne doit exister qu'en un seul point. Même si j'ai utilisé un plan de masse pour le circuit SMPS, la connexion de masse avec le circuit externe pouvait n'être faite qu'avec une seule barrette.

La **figure 15** montre le circuit imprimé garni. Et maintenant, pour le bruit mesuré à la sortie du circuit (à 2 mV/div), voyez la **figure 16**. En comparaison avec la figure 4, il est clairement proche du bruit de fond de l'oscilloscope !

(190127-03 – version française : Denis Lafourcade)

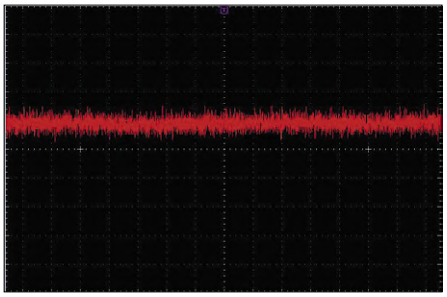


Figure 16. Bruit de sortie du convertisseur avec le filtre auxiliaire mis au point (2 mV/div ; ×1), résultat final.

Lien

[1] Fichiers de conception du circuit imprimé (Altium) : www.elektormagazine.fr/190127-03

« Du bon usage de l'oscilloscope »

Pour des mesures réellement précises, il faut enlever le fil de connexion de masse de la sonde de l'oscilloscope et la connexion avec le corps de la sonde doit être réalisée avec un ressort de masse. La **figure 17** montre ce type de connexion. De cette façon, vous serez sûr que le bruit du signal vu sur l'écran de l'oscilloscope provient uniquement du point de test dans le circuit, et que les effets du bruit ambiant restent à leur niveau minimal.

une journée avec LoRa

l'Alliance LoRa se réunit à Berlin

Jens Nickel

L'Internet des Objets se propose de rendre notre vie plus confortable, plus sûre et surtout moins consommatrice de ressources. Il se nourrit des données d'une myriade de capteurs sans fil pouvant passer l'année sans changer de piles. Les applications vont de la gestion de parkings et d'entrepôts au suivi de marchandises et d'animaux, en passant par le large domaine de l'acquisition de données environnementales. Les exigences de portée du signal, de passage à travers les murs d'immeubles et de faible consommation excluent les technologies sans fil classiques telles que le Wi-Fi et le Bluetooth. Mais pour la transmission sans fil efficace à faible débit, il est apparu de nouveaux standards tels que LoRa, Sigfox et NB-IoT, qui se font concurrence sur un marché en croissance permanente. L'architecture est basée sur des stations réceptrices de base (aussi appelées passerelles (*gateways*)) qui captent les données pour ensuite les envoyer sur l'internet par Wi-Fi ou Ethernet. L'écosystème de chacune de ces technologies comprend des serveurs qui rendent les données accessibles aux propriétaires des capteurs (par une interface de programmation web (API), le protocole MQTT, des courriels, etc.). Les concepteurs bénéficient de cartes de développement à prix intéressant, d'une utilisation gratuite de l'infrastructure à des fins d'évaluation, de bibliothèques logicielles et d'exemples d'applications.

LoRa s'est assuré une position enviable sur ce marché. Du point de vue physique, ce standard repose sur un procédé de modulation CSS mis au point par le fabricant de semi-conducteurs Semtech, qui l'intègre à des systèmes sur une puce qu'il produit et licencie. Contrairement au concurrent Sigfox, il n'y a pas là un réseau unique, mais toute une ribambelle de réseaux petits et grands, publics et privés. Pour que des nœuds de capteurs puissent rejoindre ces divers réseaux, il faut d'abord que les émetteurs-récepteurs s'entendent au niveau physique pour pouvoir s'échanger des données telles quelles (ce qui fonctionne, comme nous le montrons dans ce numéro avec le projet « voltmètre sans fil pour batterie de voiture »). Ensuite il faut des procédures de connexion, de la synchronisation et une gestion des collisions, ainsi que des mécanismes de sécurité. Tout cela est compris dans le standard LoRaWAN, que promeut une Alliance importante et croissante de sociétés des plus diverses [1][2]. L'Alliance LoRa [3] comprend, outre Semtech, différents fabricants de semi-conducteurs ainsi que des sociétés de télécommunications, des développeurs de logiciel et beaucoup d'utilisateurs, en tout plus de 500 membres. L'Alliance se réunit chaque année dans une ville différente ; en 2019, ce fut le tour de Berlin. Pendant les deux premiers jours, l'Alliance travaille en sessions fermées pour continuer à améliorer le protocole et échanger de l'expérience. Le troisième jour, elle invite les représentants des médias et des clients potentiels. Bien entendu, cela ne saurait se passer d'un peu d'autocélébration. Après le discours d'introduction de la dirigeante de l'Alliance, Donna Moore, qui portait surtout sur la croissance du consortium au cours de l'année précédente, des prix ont été attribués aux experts qui se sont particulièrement distingués dans la poursuite du développement du protocole et du support. La parole est ensuite passée aux sponsors de l'événement. Les conférenciers ont évidemment abordé la façon de se distinguer de la concurrence. La société d'études de marché ABI Research a publié une étude qui examine surtout les différences entre LoRa et NB-IoT (**fig. 1**) [5]. Il doit certes être douloureux pour les partenaires de LoRa que beaucoup de sociétés de télécommunications ne fassent pas partie de l'Alliance et promeuvent un standard basé NB-IoT. Celles-ci font valoir qu'il existe déjà une infrastructure de réseau et des domaines de fréquences spécialement réservés alors que LoRa utilise la bande des 868 MHz en Europe. Ce à quoi il fut répondu à Berlin que les nœuds de capteurs LoRa s'avèrent moins énergivores, meilleur marché et plus compacts, ce qui put être démontré avec un nœud à capteurs multiples sur l'exposition de produits attenante (**fig. 2**).

Figure 1. Étude comparative de LoRa avec NB-IoT, présentée à Berlin.

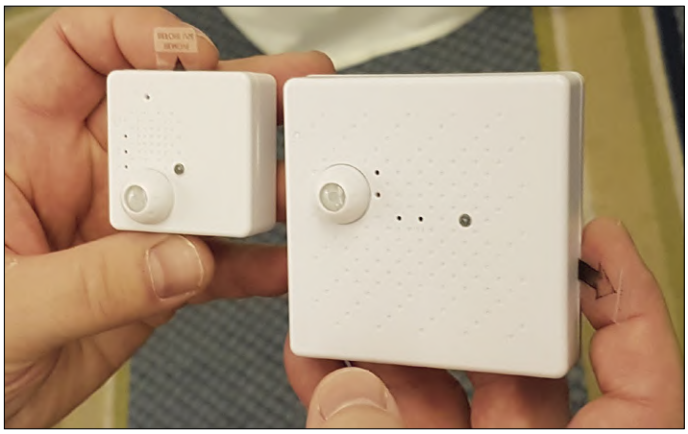


Figure 2. Deux variantes du même nœud de capteurs, l'une avec LoRa, l'autre avec NB-IoT (www.tektelic.com).

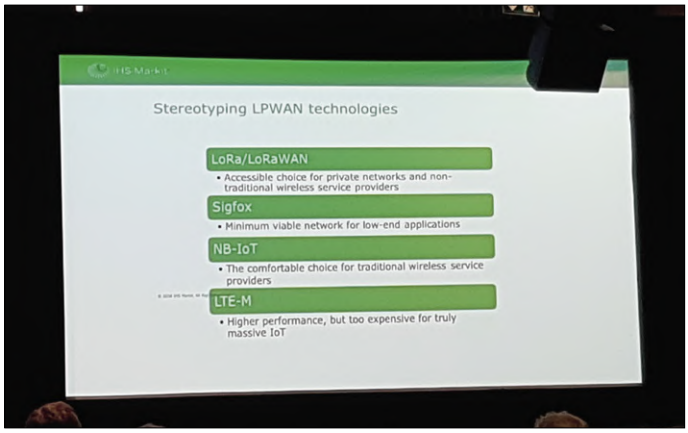


Figure 3. LoRa comparé à NB-IoT et Sigfox.

Figure 4. La startup Polygravity a présenté une solution pour les communications en itinérance et le « paiement transréseau ».

À Berlin, l'Alliance a situé le concurrent Sigfox de l'autre côté du spectre des applications, c'est-à-dire adapté aux applications de bas de gamme, où il ne s'échange que quelques octets par heure ou par jour (**fig. 3**). L'infrastructure Sigfox séduit par une certaine simplicité : il n'y a qu'une seule et même société pour licencier les puces, gérer les stations de base et les serveurs sur lesquels les utilisateurs déclarent leurs nœuds de capteurs et récupèrent les données. LoRa est par contre beaucoup plus hétérogène. Les utilisateurs doivent soit installer leur propre réseau (passerelles et serveurs), soit enregistrer leurs nœuds de capteurs chez un opérateur commercial. L'étendue de ces réseaux est le plus souvent limitée ; pour des applications exigeant un haut degré de mobilité, comme le suivi du transport lointain de marchandises, il existe bien entendu déjà des applications d'itinérance (*roaming*) (**fig. 4**).
TheThingsNetwork (TTN) [6] est un réseau LoRa libre, prétendant à une couverture globale. Il est géré par des bénévoles qui font fonctionner chacun une ou plusieurs passerelles. Il comprend également une plate-forme de réseau sur laquelle on peut récupérer ses propres données à travers une interface applicative. Le fondateur de TTN, Wienke Giezeman était également à Berlin. Un autre serveur de réseau ouvert est géré sur LoRaServer.io [7]. Dans un atelier, nous avons vu comment, avec un Raspberry Pi, un module passerelle connecté et du logiciel à code source ouvert, on peut réaliser une passerelle LoRaWAN capable d'échanger des données avec le serveur LoRa aussi bien qu'avec TheThingsNetwork (**fig. 5**). Si l'on ne compte pas le temps passé à télécharger les données depuis Github et à les installer sur le RPi, cela ne prend que quelques minutes. Cet article se limite à effleurer nombre de sujets. Ce thème fera l'objet d'autres articles dans le magazine Elektor ; des projets LoRaWAN sont d'ores et déjà en préparation.

Figure 5. Pour l'atelier « LoRaWAN in a box », la société indienne ICFOSS avait apporté des Raspberry Pi et des modules passerelles.

(190340-03 – version française : Helmut Müller)

Liens

[1] Spécifications LoRaWAN : https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf
[2] Boîte à outils logiciels : https://lora-developers.semtech.com/resources/tools/basic-mac/welcome-basic-mac/
[3] Alliance LoRa : https://lora-alliance.org
[4] L'Alliance LoRa se réunit à Berlin : https://lora-alliance.org/events/berlin-annual-members-meeting-2019
[5] Étude comparative LoRaWAN et NB-IoT : https://lora-alliance.org/resource-hub/lorawan-and-nb-iot-competitors-or-complementary
[6] The Things Network (« Le Réseau des Objets ») : https://www.thethingsnetwork.org/
[7] Serveur LoRa : https://www.loraserver.io/loraserver/overview/

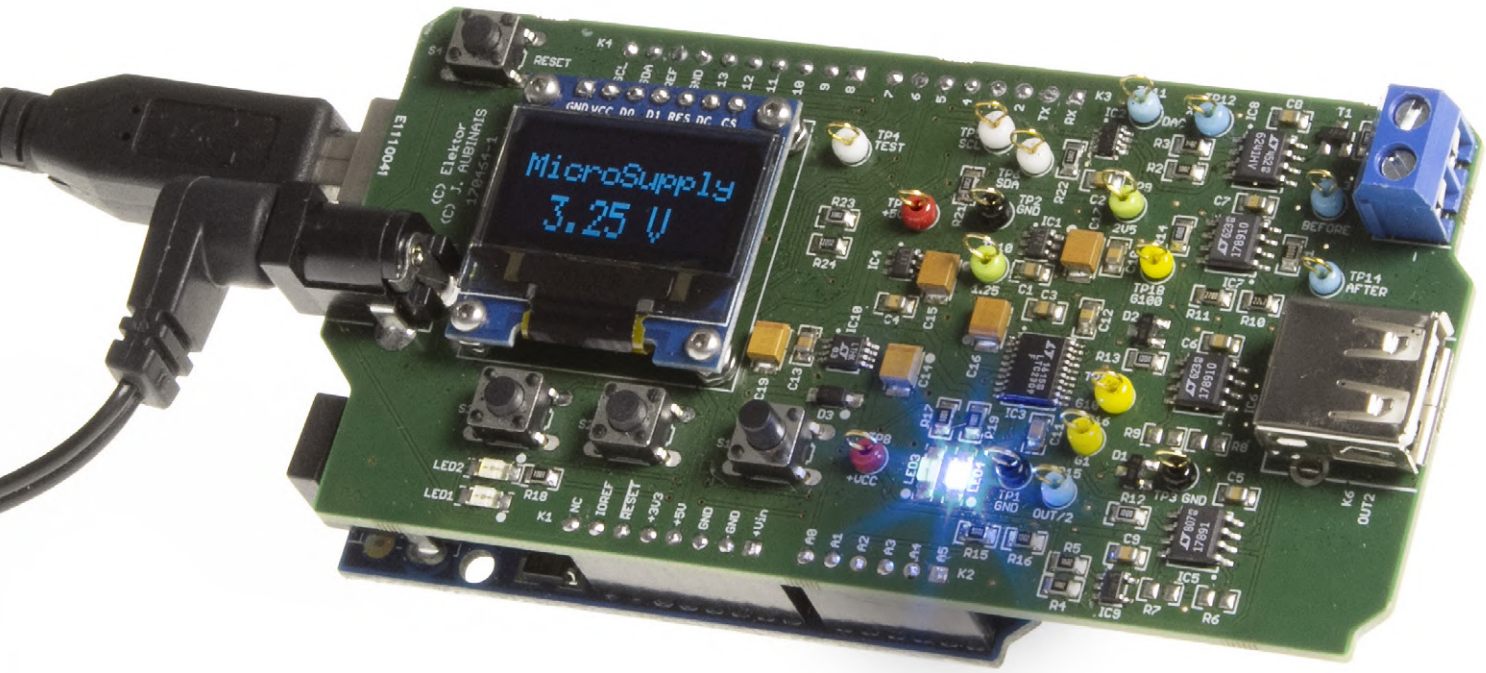
PROJET DU LABO

MicroSupply

alimentation de labo pour objets connectés

Jennifer Aubinais (Paris), remerciements à **Jean-Luc** et **Philippe**

Combien consomme mon objet connecté ? Est-ce que son mode de veille est conforme à la documentation du fabricant ? Combien de temps va tenir la pile CR2032 de mon objet ? Si vous vous posez ce genre de questions, voici une alimentation réglable, baptisée MicroSupply, sous forme de « shield » Arduino qui mesurera les courants très faibles. Grâce à un couplage avec un logiciel pour PC, vous pouvez visualiser et enregistrer la consommation de votre objet.



C'est un projet avec une longue histoire puisque cette version est déjà la 5e ! Le montage a évolué à cause des nombreuses difficultés techniques rencontrées lors de sa conception, mais aussi grâce aux échanges de l'auteure lors de séminaires et de salons. Des améliorations seront toujours possibles, mais il faut savoir mettre un point final…

Les caractéristiques

- *Shield* pour Arduino Uno.
- Tension de sortie réglable de 1,5 à 5 V au pas de 0,05 V.
- Courant de sortie mesuré affiché de 1 µA à 40 mA.
- Cadence de mesure de 1 kHz.
- Logiciel PC pour afficher les mesures reçues par le port série.
- Disjoncteur de courant par logiciel de 40 mA.
- Disjoncteur de tension par logiciel (inférieur à 100 mV).
- Nécessite une alimentation externe de 10 à 12 V.

Synoptique

Avant de détailler l'électronique du projet, voici les différents modules qui le constituent (**fig. 1**) :

- La tension d'alimentation pour notre objet connecté est réglable à l'aide d'un convertisseur numérique-analogique (CN/A).
- Comme le convertisseur ne fournit pas assez de courant, un petit amplificateur de « puissance », avec un maximum de 40 mA, est mis en place.

- La « résistance de shunt », comme sa définition l'indique, permet de mesurer le courant à travers celle-ci par la mesure de la tension à ses bornes. Comme cette résistance sera souvent citée, elle fait l'objet d'un module.
- Avant de convertir la tension aux bornes de notre résistance de shunt, plusieurs amplificateurs d'instrumentation sont implémentés. Pourquoi plusieurs ? Le courant mesuré peut varier de 1 µA à 40 mA, soit de 1 à 40.000. Il n'est pas facile de couvrir une telle amplitude avec précision avec une seule gamme. C'est pourquoi trois amplificateurs d'instrumentation se partagent les gammes nécessaires : ×1, ×10 et ×100.
- Un convertisseur analogique-numérique (CA/N) convertit les différentes tensions (les mesures de courant) en sortie des trois amplificateurs d'instrumentation.
- Enfin, un écran OLED permet l'affichage de la tension et du courant en sortie. L'écran est également utilisé pour montrer de nombreux messages comme les étapes de vérification du bon fonctionnement de l'ensemble du projet lors de la phase de test.

La première version…

… était un simple *shield* Arduino qui fournissait une tension de sortie de 0 à 5 V et qui mesurait le courant aux bornes d'une petite résistance de shunt. Un afficheur OLED de 0,96 pouce affichait périodiquement la mesure.

Ensuite il est apparu dommage que les mesures ne soient pas transmises à un PC pour pouvoir les visualiser, et, qui sait, les traiter ou les stocker. L'objectif n'était pas de réaliser un oscilloscope numérique, mais d'avoir une fréquence d'échantillonnage intéressante. Si c'est trop lent, ce n'est pas très utile puisqu'on ne cherche pas à visualiser la consommation toutes les heures ou tous les jours ; un rythme trop rapide, toutes les microsecondes par exemple, entraînerait une réalisation complexe. Le choix s'est alors porté sur une fréquence d'échantillonnage de 1 kHz, c.-à-d. 1.000 mesures par seconde. Le CA/N initialement choisi n'était pas assez rapide pour des conversions toutes les millisecondes, il a donc fallu le remplacer.

La résistance de shunt

Un autre problème est posé par la gamme des mesures souhaitée, de 1 µA à 40 mA. Dans la première version, la résistance de shunt avait une valeur de 10 Ω. Un courant de 40 mA produit alors une tension de 400 mV aux bornes de cette résistance, pour 1 µA c'est 10 µV. C'est peu. Une valeur de shunt plus élevée, p. ex. 100 Ω, permet de remédier en partie à ce problème, mais il ne faut pas dépasser la tension d'entrée maximale du CA/N. En plus, la tension aux bornes du shunt s'ajoute à la tension d'alimentation de la charge. Si cette dernière doit être alimentée sous 5 V et qu'elle consomme un courant de 40 mA, avec un shunt de 100 Ω, il faudra une tension de 9 V (5 + 0,04 × 100) pour alimenter la charge. Bref, plus la valeur de la résistance de shunt est grande, plus la tension aux bornes de celle-ci pour les courants faibles sera élevée, mais plus la tension de sortie de l'alimentation doit être élevée.

Trois amplificateurs d'instrumentation

Un amplificateur à gain variable est la solution idéale. Par exemple, un gain de 1 pour le haut de la gamme, un gain de 10 pour le milieu et un gain de 100 pour le bas permettront de mesurer avec la même précision le courant qu'il soit faible ou fort. Pour mesurer la tension sur les bornes du shunt, nous avons besoin d'un amplificateur différentiel, de bonne

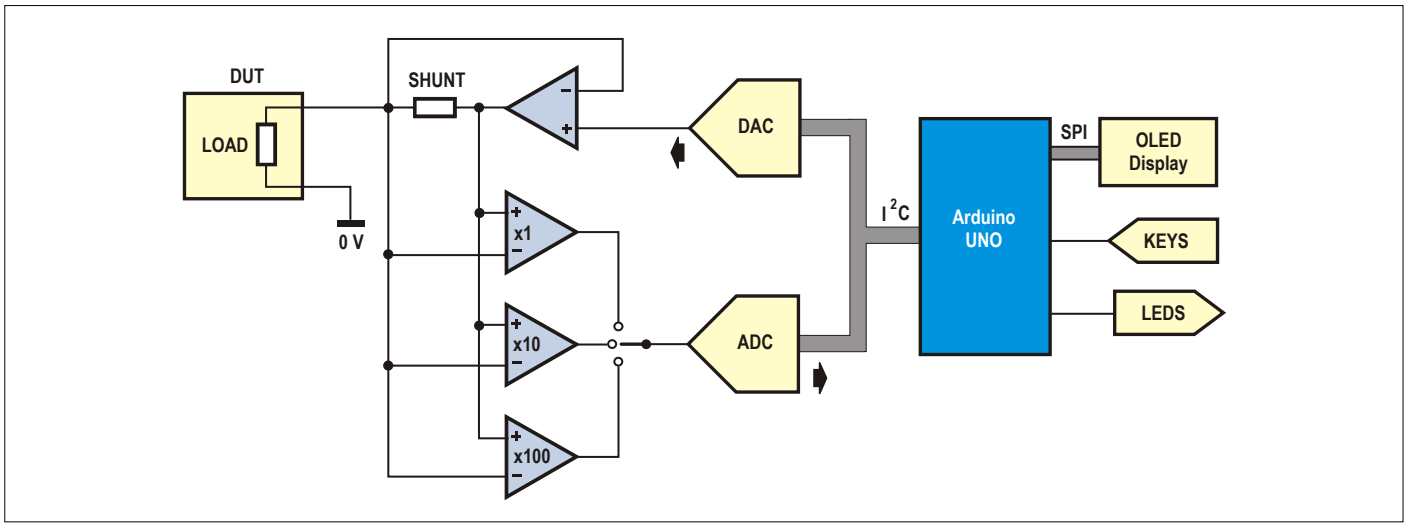


Figure 1. Le synoptique du MicroSupply.

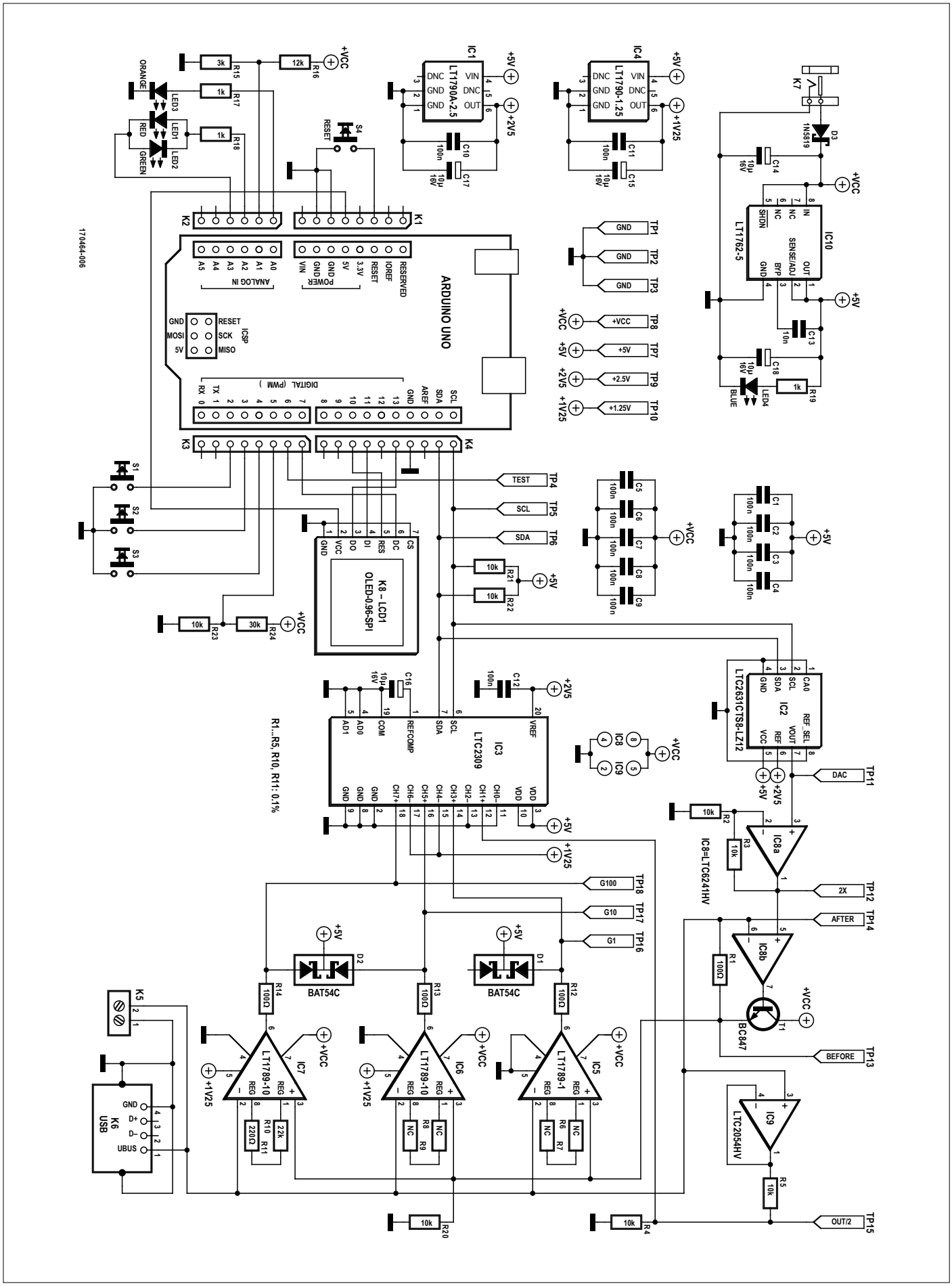


Figure 2. Le schéma du *shield*.

qualité de préférence et c'est pourquoi nous avons opté pour un amplificateur d'instrumentation. Malheureusement, ce genre d'amplificateur possède un gain fixe. Il est possible de modifier le gain en commutant des résistances, mais cela nécessite plusieurs commutateurs et des signaux pour les piloter (plus un peu de logiciel). La solution retenue est l'utilisation de trois amplificateurs avec des gains différents en parallèle.

La tension de sortie

Un convertisseur numérique-analogique (CN/A) est chargé de créer une tension réglable de 0 à 5 V, soit, dans la version finale du projet, de 1,5 V (valeur supérieure à la tension de référence 1,25 V des amplificateurs d'instrumentations, voir plus loin) à 5 V. Avec une tension de référence de 2,5 V et une précision de 12 bits, le convertisseur fournit alors une tension de sortie de 0 à 2,5 V au pas de 0,61 mV ($2{,}5 / (2^{12}-1) = 2{,}5 / 4.095$). Pour obtenir une tension jusqu'à 5 V, on utilise un amplificateur opérationnel avec un gain de 2. Du coup, le pas est de 1,22 mV.

Supprimer la résistance de sortie de notre alimentation

Reste à régler le problème de la résistance série élevée de l'alimentation due à la valeur du shunt. L'objectif est de fournir une tension de sortie constante, indépendamment du courant consommé par la charge, et il faut donc compenser cette résistance. C'est possible en créant une boucle à rétroaction autour du shunt. Ainsi, un ampli-op à sortie renforcée (par T1, pour pouvoir fournir les 40 mA max) fait en sorte que la tension de sortie reste constante, même si le courant varie entre 1 µA et 40 mA. Outre le fait qu'un courant maximal de 40 mA est intéressant, cela correspond aussi au courant maximal à travers la résistance de shunt de 100 Ω sans dépasser la tension maximale en entrée du CA/N (4,096 V).

Il faut une alimentation externe

Désormais, nous disposons d'un système capable de fournir une tension stable de 1,5 à 5 V et de mesurer le courant consommé. Le prix à payer, en dehors de la complexité du circuit, est une alimentation d'entrée élevée d'au moins 9 V (voir plus haut). La première version disposait pour cela d'un doubleur de tension, le LTC660, qui transformait les +5 V disponibles sur la carte Arduino en +10 V. Or, la tension ainsi obtenue comportait trop de bruit que nous n'avons pas réussi à réduire suffisamment. Las, nous avons fini par supprimer le doubleur de tension et le remplacer par une alimentation externe. Cette décision se justifie par le fait que le MicroSupply est un instrument de labo et que chaque labo qui se respecte est censé disposer d'une alimentation de labo digne de ce nom.

Pour la petite histoire, au départ nous étions partis sur une tension d'alimentation externe de 22 V ce qui permettait une résistance de shunt de 400 Ω, mais cette valeur atypique et relativement élevée était considérée comme trop contraignante, c'est pourquoi nous avons descendu la tension à 10 V minimum (9 V plus la tension aux bornes de D3 et T1), 12 V maximum, à cause de la limite d'alimentation d'IC8.

Notez que la carte Arduino est alimentée soit par son port USB que vous utilisiez le programme sur le PC ou pas, soit par une alimentation externe qui peut être la même que celle du *shield*.

Le circuit en détail

Commençons pour une fois par l'alimentation (**fig. 2**). Puisque celle-ci est censée être fournie par une alimentation de labo de qualité correcte, son traitement sur la carte est resté limité à un régulateur (IC10) et quelques condensateurs pour produire les 5 V nécessaires pour le CN/A et le CA/N. Les amplificateurs sont alimentés directement par la tension d'entrée (il y a toutefois une protection contre une inversion de polarité : D3). L'afficheur OLED est quant à lui alimenté par les 5 V de la carte Arduino. Ainsi les alimentations pour les parties analogique et numérique sont séparées sans trop d'efforts.

Convertisseur numérique-analogique

IC2 est le CN/A qui produit la tension de sortie de notre MicroSupply. Le circuit offre une résolution de 12 bits et possède en interne une tension de référence de 2,5 V. Il fournit alors une tension de sortie (« visible » sur TP11, nommé « DAC ») de 0 à 2,5 V au pas de 0,61 mV ($2{,}5 / (2^{12} - 1)$). Pour obtenir une tension de sortie jusqu'à 5 V, on utilise un ampli-op, IC8, avec un gain de 2 (TP11, « 2× »). La résolution passe alors à 1,22 mV.

Comme dit plus haut, le CN/A IC2 dispose d'une source de référence interne, mais sa fiche technique ne donne pas beaucoup de détails sur la qualité de cette référence. C'est pourquoi nous avons rajouté IC1, une source de référence à haute précision bien documentée. Ainsi nous savons mieux à quoi nous en tenir. Vous pouvez l'observer sur le point de test TP9 (« 2.5V »).

Amplificateur de « puissance »

L'autre partie de IC8 pilote un transistor pour débiter le courant de sortie maximal de 40 mA. Puisqu'il s'agit d'un ampli-op, il va tenter de régler sa sortie de telle façon que les deux tensions sur ses entrées, les broches 5 et 6, deviennent identiques. Grâce à la connexion de l'entrée inverseuse de l'ampli (broche 6) à la sortie du MicroSupply (TP14, « AFTER »), ce mécanisme fait en sorte que la tension de sortie soit identique à celle sur l'entrée non-inverseuse (broche 5), indépendante du courant à travers R1 et T1. Si l'ampli-op est assez rapide par rapport aux variations du courant demandé par la charge, cette dernière aura l'impression d'être alimentée par une source à très faible impédance de sortie, ce qui était notre objectif. IC9 divise la tension de sortie par deux (TP15, « OUT/2 ») pour pouvoir l'afficher et activer le disjoncteur logiciel.

La résistance de shunt et ses amplificateurs

R1 est la résistance de shunt évoquée plus haut. Le courant consommé par la charge passe à travers R1 et produit sur ses bornes (TP13, « BEFORE » et TP14, « AFTER ») une tension que nous pouvons mesurer. Pour cela, nous utilisons trois amplificateurs d'instrumentation en parallèle : un avec un gain de 1 (IC5), un deuxième avec un gain de 10 (IC6), et un dernier pour un gain de 100 (IC7). Faites bien attention aux suffixes des références de ces composants au moment de les commander ou de les souder, car ils n'ont pas tout à fait la même. En effet, IC5 a le suffixe « -1 » tandis que IC6 et IC7 ont comme suffixe « -10 ». Le suffixe indique le gain préprogrammé. À l'aide d'une résistance, il est possible de le modifier, ce que nous avons fait pour IC7 avec R10 et R11 pour arriver à un gain de 100. Il aurait été possible de n'utiliser qu'un seul amplificateur d'instrumentation et de commuter cette résistance de gain pour arriver à nos fins, mais il n'est pas sûr qu'au final nous aurions vraiment

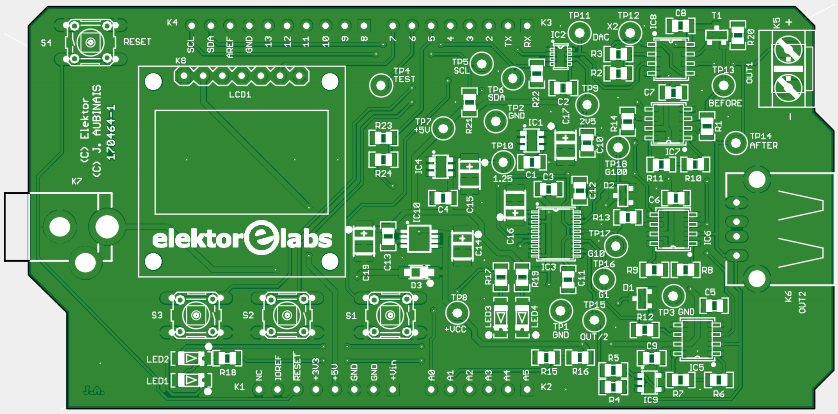
LISTE DES COMPOSANTS

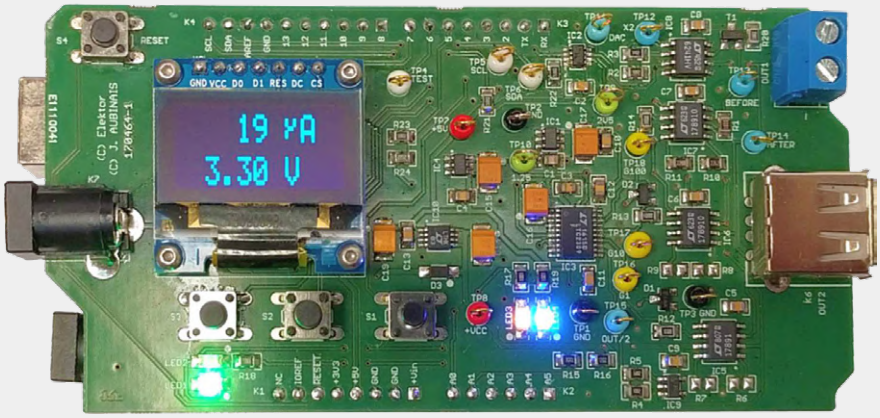
Résistances (0805)
R6, R7, R8, R9 = NC
R12, R13, R14 = 100 Ω
R1 = 100 Ω 0,1%
R11 = 220 Ω 0,1%
R17, R18, R19 = 1 kΩ
R15 = 3 kΩ
R20, R21, R22, R23 = 10 kΩ
R2, R3, R4, R5 = 10 kΩ 0,1%
R16 = 12 kΩ
R10 = 22 kΩ 0,1%
R24 = 22 kΩ

Condensateurs (0805)
C13 = 10 nF
C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12 = 100 nF
C14, C15, C16, C17, C18 = 10 µF 16 V, tantale, CMS 1411

Semi-conducteurs
D3 = 1N5819HW-7-F
D1, D2 = BAT54C
T1 = BC847C
LED4 = LED bleue, 1206
LED2 = LED rouge, 1206
LED3 = LED jaune, 1206
LED1 = LED verte, 1206
IC10 = LT1762EMS8-5
IC5 = LT1789CS8-1
IC6, IC7 = LT1789CS8-10
IC4 = LT1790BCS6-1.25
IC1 = LT1790ACS-2.5
IC9 = LTC2054HVCS5
IC3 = LTC2309CF
IC2 = LTC2631CTS8-LZ12
IC8 = LTC6241HVCS8

Électromécanique
K8 = connecteur femelle à 7 contacts, au pas de 2,54 mm pour LCD1
K2 = connecteur mâle à 6 contacts, au pas de 2,54 mm
K1, K3 = connecteur mâle à 8 contacts, au pas de 2,54 mm
K4 = connecteur mâle à 10 contacts, au pas de 2,54 mm
S1, S2, S3, S4 = commutateur tactile, 6x6 mm
K6 = embase USB de type A, coudée
K5 = bornier fil-à-carte, à 2 voies, au pas de 5,08 mm
K7 = connecteur d'alimentation CC, broche centrale de 1,95 mm





Circuit imprimé, réf. 170464-1, www.elektor.fr

Divers
LCD1 = écran OLED, 0,96 pouces, SPI, à 7 broches
4x boulon, M2 x 8 mm
12x écrou, M2
Résistances de précision de 1 kΩ, 100 kΩ et 1 MΩ, 0,1%, pour les tests et le calibrage.

gagné quelque chose.
TP16 (« G1 »), TP17 (« G10 ») et TP18 (« G100 ») offrent des points de test pour les trois amplificateurs. Attention, pour les mesures « G10 » et « G100 », il faudra soustraire la tension de référence de 1,25 V, TP10.

Numériser les mesures

Les signaux sur les sorties des amplificateurs, plus celui produit par IC9, sont numérisés par le convertisseur analogique-numérique (CA/N) IC3. Les entrées de ce circuit peuvent être utilisées comme paires différentielles ou comme entrées simples. Nous les utilisons en mode simple pour les signaux de sortie de IC9 et IC5. En revanche, les signaux de sortie de IC6 et IC7 sont mesurés en mode différentiel par rapport à une tension de référence de 1,25 V, produite par IC4 (TP10, « 1.25 »). Pourquoi? Parce que les sorties de IC6 et IC7 ne peuvent pas descendre en deçà de 110 mV, ce qui est un peu embêtant quand on essaie de mesurer de petites tensions à peine au-dessus de 0 V. Heureusement, ces circuits disposent d'une entrée nommée « référence » qui permet de rehausser le signal à la sortie. À l'aide d'une entrée différentielle du CA/N, nous pouvons ensuite soustraire la tension de référence pour la ramener vers une mesure par rapport au zéro. Nous n'appliquons pas cette astuce au signal produit par IC5, car cela permet d'avoir une tension maximale de 4,096 V en sortie de l'amplificateur d'instrumentation quand le courant fourni par l'alimentation est de 40 mA.

Comme le CN/A, le CA/N a une résolution de 12 bits. Et, tout comme le CN/A, il dispose d'une référence interne que nous avions préféré remplacer par notre propre référence IC1. Enfin, tout comme le CN/A (ils ne seraient pas de la même famille, par hasard ?), le CA/N communique par bus I2C (avec points de test TP5, « SCL » et TP6, « SDA »).

L'afficheur sur le bus SPI

Ainsi nous arrivons à l'écran OLED qui, lui, communique par bus SPI. Ce type

d'afficheur existe aussi avec interface I2C, mais nous avons voulu séparer les bus de communication, d'autant plus que le bus SPI est plus adapté à des applications graphiques, car il permet des vitesses de transfert de données nettement plus élevées.

Et le microcontrôleur alors ?

Ainsi se termine notre tour du schéma du MicroSupply, nous avons tout vu. Vraiment ? Mais non, il manque la description du microcontrôleur, le composant qui anime notre montage ! Vous l'avez deviné (en même temps, nous ne l'avons pas caché non plus), il s'agit d'une carte Arduino Uno sur laquelle se trouve un ATmega328.

À part piloter notre sh*ield,* ce qui consiste entre autres à contrôler le CN/A, récupérer les échantillons et afficher les données sur l'écran OLED, le µC s'occupe aussi de la communication avec un ordinateur par un port série USB.
Afin d'être sûr que la tension d'alimentation du sh*ield* soit suffisante pour alimenter la charge (> 10 V), le µC mesure la tension d'entrée via son entrée analogique A1. Il ne continuera pas son initialisation tant que la tension d'entrée n'est pas supérieure à 10 V.
La broche D5 de l'Uno cherche aussi à détecter la tension d'entrée, mais plus grossièrement. Si le µC lit un niveau logique bas sur cette entrée, la tension d'entrée est inférieure à 9,6 V environ, alors le système s'arrête.
La tension à la sortie est surveillée par le µC (à l'aide d'IC9) afin de vérifier qu'il n'y a pas de baisse de tension supérieure à 100 mV (limitée dans le programme). Si cela arrive, la sortie est coupée immédiatement et un message d'erreur est affiché.

Quelques mots sur le logiciel

Le croquis Arduino n'est pas complexe en soi, mais quelques lignes de code doivent attirer votre attention :

- Le **système antirebond des touches.** Les boutons-poussoirs S2 et S3 (« Do*wn* » et « Up ») permettent de choisir la tension de sortie entre 1,5 et 5 V au pas de 0,05 V. Ils sont gérés par la bibliothèque de Jack Christensen [3], légèrement modifiée, qui évite les rebonds et offre une accélération de l'incrémentation ou la décrémentation pendant une pression prolongée.
- Les **interruptions de 1 ms (1 kHz),** le temps entre chaque mesure. Lors des premiers tests, le montage a été soumis à une fausse charge réalisée avec un signal sinusoïdal de 1 Hz protégé par des résistances. Puisque 1 kHz est un multiple exact de 1 Hz, nous nous attendions à un signal parfaitement stable. Or, l'affichage sur le PC montrait une sinusoïde qui se décalait lentement. Après quelques recherches, il s'est avéré que la période de notre interruption était en réalité de 1,024 ms. Cela est dû à la carte Arduino qui utilise le même compteur pour produire les signaux à modulation de largeur d'impulsion (PW*M* [4]). Pour corriger cela, il faut configurer le registre OCR0A à 249 : 16.000.000 Hz (fréquence du quartz) / 64 (diviseur interne) / (249+1) = 1.000 Hz.
- Les **polices de caractères.** En plus des bibliothèques disponibles sur l'internet, nous avons dû créer une nouvelle police de caractères pour améliorer l'affichage sur l'écran OLED.
- La **bibliothèque I2C.** Étant donné que le projet n'est réalisé qu'avec des circuits intégrés d'Analog Devices (Linear Technology), nous avons recours à la bibliothèque « LT_I2C » [5].

La fonction loop

Comme son nom l'indique, la fonction *loop* fait tourner le µC en rond continuellement. Elle connaît trois modes :

- **Stand-by.** Ce mode est activé/désactivé par S1 (le plus à droite). Dans ce mode, S2 et S3, sous l'afficheur, permettent d'ajuster la tension à la sortie.
- **Local.** Les mesures sont affichées sur l'écran OLED, elles ne sont pas transmises sur le port série. La réception d'un caractère 'E' sur le port série permet d'activer ce mode (c'est le programme du PC qui s'en charge).
- **Déporté.** Les mesures sont transmises sur le port série, elles ne sont pas affichées sur l'écran OLED (qui affiche alors « SE*RIAL* »). La réception d'un caractère 'B' sur le port série permet d'activer ce mode (c'est le programme du PC qui s'en charge).

Autodiagnostic intégré

Si une des touches est pressée lors du démarrage de l'Arduino, un petit programme de test du sh*ield* est lancé. Il suffit de brancher une résistance de 100 kΩ (0,1%) puis de 1 kΩ (0,1%) sur la sortie et de suivre les instructions sur l'écran OLED.

Il est également possible de calibrer le sh*ield.* Pour lancer la procédure, il faut appuyer simultanément sur les touches S2 et S3 (« Do*wn* » et « Up ») lors du démarrage de l'Arduino. Branchez une résistance de 1 MΩ (0,1%) à la sortie et suivez les instructions sur l'afficheur.

Application PC

Puisque le MicroSupply envoie les mesures à une cadence de 1 kHz, il faut que le programme PC arrive à les afficher sous forme graphique en temps réel. L'auteure, qui programme principalement en C#, a d'abord cherché une solution avec ChartControl, mais le rafraîchissement et surtout la mise à l'échelle automatique étaient trop lents. Après de très longues recherches, elle a découvert la bibliothèque Nebula. Cette bibliothèque répond très bien aux besoins, mais demande un développement en Eclipse/Java. Les premiers tests ont été concluants, sauf pour la communication série avec notre Arduino. Alors, comment faire sans opter pour une solution payante ? Eh bien, avec IKVM [6], une machine virtuelle Java (JVM) pour les environnements d'exécution .NET, autrement dit, Visual Studio. L'outil IKVM.NET (développé par Jeroen Frijters) permet de migrer une application de base de données Java existante vers .NET. Comme Elektor n'est pas une revue d'informatique, nous vous épargnerons tous les détails sur l'utilisation de ce formidable outil, mais, en résumé, celui-ci produit une DLL écrite en Java (DL*L MicroSupply_JA.dll)* facile à exploiter avec C# dans Visual Studio. Notre application PC utilise donc deux langages de programmation différents.

L'application stocke 4.000 mesures, soit 4 s, qui pourront être visualisées par tranches de 1, 2 ou 4 s. Attention : l'application n'affichant que 1.000 points, pour les choix de 2 et 4 s certaines mesures n'apparaissent pas ce qui peut entraîner une erreur d'analyse du fonctionnement de votre objet connecté.

Windows 10 en 64 bits exigé

Pour que le programme fonctionne sur votre PC, vous aurez besoin de Windows 10, 64 bits et mis à jour plus Java (64 bits) version 8, update 191 ou supérieur. Téléchargez le fichier d'archive depuis [1] ou [2] et décompressez-le dans un répertoire de votre choix. Vous y trouverez ensuite :

- Le programme Mi*croSupply.exe*
- Les deux DLL IK*VM.OpenJDK.Core.dll* et ik*vm-native-win32-x64.dll*
- Les DLL de la fonction SWT (répertoire « filesSWT » du téléchargement [1]) avec les noms des répertoires « .swt\lib\win32\x86_64 » sont à copier dans le répertoire « C:\Users\YourUserName ».
- La DLL Mi*croSupply_JA.dll* (il n'est pas nécessaire de la copier, car elle a été intégrée dans le programme MicroSupply.exe avec de nombreuses autres DLL IKVM).

La première fois, dans une ligne de commande, il est souhaitable de lancer le programme avec l'argument « de*bug* ». Celui-ci affichera alors plus d'informations au cas où vous auriez un problème de configuration de votre PC pour l'exécution du programme.

Manuel d'utilisation

Voici rapidement à quoi servent les divers boutons et curseurs de l'application pour PC :

- « Re*fresh* » : rafraîchir la liste des ports séries du PC.
- « Ma*x Value* » et « Mi*n Value* » : intensités de courant mesurées, maximale et minimale sur la seconde affichée.
- « 1 *second* », « 2 *seconds* » et « 4 *seconds* » : mode d'affichage, afficher les données sur 1, 2 ou 4 s.
- Curseur long (de 0 à 9) : permet de choisir une période des données à afficher (max. 4 s). En mode « 1 *second* » par pas de 0,25 s ; en mode « 2 *seconds* », le pas est de 0,5 s.
- Curseur court (de 0 à 1 ou de 0 à 3) : sélectionner les mesures mémorisées. En mode « 2 *seconds* », le curseur permet d'extraire une mesure sur deux, soit la première mesure soit la deuxième. En mode « 4 *seconds* », c'est une mesure sur quatre : la première, la deuxième, la troisième ou la quatrième.
- « Sa*ve* » : sauvegarde dans un fichier CSV de toutes les mesures (valeurs en µA), soit 4.000 données maximum.

Les LED et les boutons

- LED1 (verte) : l'objet connecté est alimenté.
- LED2 (rouge) : le disjoncteur logiciel a été déclenché, l'objet connecté n'est plus alimenté.
- LED3 (jaune) : clignote si l'objet connecté est alimenté. En cas d'un dysfonctionnement du programme, elle restera fixe (allumée ou éteinte).
- LED4 (bleue) : le *shield* est alimenté par l'alimentation extérieure. Cela ne veut pas forcément dire que celle-ci est suffisante.
- S1 : activation / désactivation de la sortie
- S2 : *Up* / *Next* / *Yes*
- S3 : *Down* / *No*
- S4 : *Reset*.

Conclusion

Le MicroSupply est capable d'alimenter une charge sous 5 V maximum et de fournir un courant maximal de 40 mA. Dans le même temps, il mesure le courant consommé par la charge et il affiche cette valeur sur son écran OLED. Les données capturées peuvent également être transmises sur un port série, à un ordinateur par exemple. Si le récepteur de données est un PC tournant sous Windows 10, une application spécialement développée permet de visualiser ces données sous forme de graphe. Voici donc un petit outil très pratique pour vérifier la consommation de vos objets connectés (actuels ou à venir).

(170464-01)

Liens

[1] Page de l'article : www.elektormagazine.fr/170464-01

[2] Code source : https://github.com/jenniferaubinais/MicroSupply

[3] Bibliothèque Button : https://github.com/JChristensen/Button

[4] Livre 'Maîtrisez les microcontrôleurs à l'aide d'Arduino' : www.elektor.fr/maitrisez-les-microcontroleurs-a-l-aide-d-arduino-3e-edition

[5] LT_I2C & Linduino : https://github.com/analogdevicesinc/Linduino

[6] IKVM : www.ikvm.net/

SIGLENT®

The Best Value in Electronic Test & Measurement

SIGLENT : Une gamme d'instruments de mesures pour réussir un projet IoT

Oscilloscope 2 et 4 voies
De 50MHz à 1GHz

- Analyse de signaux analogiques et numériques
- Décodage et analyse de bus série
- analyse de la qualité et de l' intégrité du signal
- Electronique de puissance (diagramme de Bode)

Générateur de signaux et ondes arbitraires 1 et 2 voies 20Vpp 10MHz à 500MHz

- Modulation analogique et numériques
- Génération d'un signal en bande de base IQ
- USB/LAN (IEEE en option)

Analyseur de spectre jusqu'à 3. 2GHz
En option : analyse vectorielle du réseau jusqu'à 1, 5GHz (seul SVA1000X)

- Pré-conformité EMV
- Analyse du signal (par ex. PSK, QAM, etc.)
- Adaptation d'antenne
- Amplificateur RF et analyse de filtre

Conception d'un objet connecté

Alimentation de laboratoire Programmable

- 1 à 3 sorties indépendantes
- Contrôle fonctionnel
- Test et analyse de la consommation d'énergie
- Test de batterie

Générateurs de signaux HF jusqu'à 3, 2GHz

- Génération de signal IQ couplé au SDG 6000
- Mise au point d' un récepteur
- Test d'immunité CEM

Charge électronique 200W / 300W

- Simulation d' une charge
- test de batterie
- Calcul de la puissance absorbée
- Qualification d' alimentation de convertisseur DC/DC ….

Multimètre numérique 4 ½ -, 5 ½- et 6 ½ digits

- Fonction histogramme,
- Graphique de tendance et statistiques
- Centrale d' acquisition multicanal (opt)

SIGLENT®

Siglent Technologies Germany GmbH
Adresse: Liebigstrasse 2-20, Gebäude 14, 22113 Hamburg Deutschland
Tel.: +49(0)4081995946 E-mail: info-eu@siglent.com http://www.siglenteu.com

conception matérielle avec (V)HDL (4)

mesure de temps de réaction

Jörg Zollmann

Dans l'article précédent de la série [1], nous avons piloté une matrice à 8×8 LED pour afficher l'heure. Cette fois-ci, nous utiliserons pour l'affichage des afficheurs à 7 segments. Mais à la place de l'horloge, nous réaliserons une logique d'exécution un peu plus compliquée et programmerons un appareil de mesure de temps de réaction.

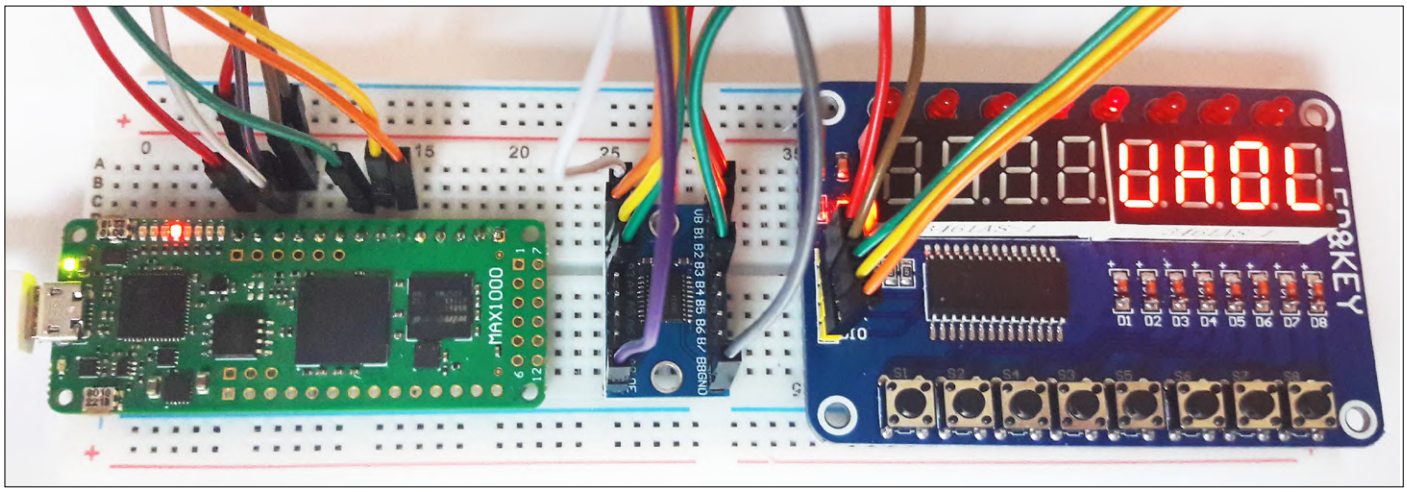


Figure 1. Réalisation du jeu de mesure de temps de réaction sur une platine d'essai.

Comme toujours, le code source du projet est téléchargeable sur sa page web d'Elektor [2]. Pour le matériel du projet de démo VHDL, on peut utiliser soit la carte CPLD présentée dans les articles précédents, soit la plateforme MAX1000 utilisée dans le projet SCCC (voir l'encadré « @www.elektor.fr »). En outre, nous utiliserons une carte d'extension sur laquelle une puce TM1638 prend en charge un groupe de huit LED et de huit boutons-poussoirs ainsi qu'un afficheur à 7 segments (**fig. 1**). La communication entre la FPGA et la TM1638 repose sur un protocole sériel. Comme la FPGA est alimentée en 3,3 V et qu'il faut du 5 V pour la TM1638, il est nécessaire d'inclure un convertisseur de niveau (TXS0108E) dans notre réalisation (**fig. 2**).

Description du jeu

Nous allons décrire pas à pas les composants matériels nécessaires à la réalisation du jeu de mesure du temps de réaction et expliquer brièvement l'analyse temporelle, que nous n'avons pas encore abordée jusqu'ici, mais dont aucun circuit numérique ne saurait se passer.

Le jeu se compose de plusieurs blocs fonctionnels divisés en différentes entités et paquets. Les fonctions sont : le pilotage du déroulement du jeu, un chenillard à LED, le pilotage de l'affichage, y compris son pilote (maître SPI), un générateur de nombres aléatoires, un compteur de délai d'attente et un chronomètre. Le jeu de mesure du temps de réaction est démarré par la pression du bouton KEY0 sur la carte d'extension, ce qui active le chenillard à LED. Après un temps aléatoire, ce chenillard se fige et la mesure du temps de réaction commence. Le joueur doit alors appuyer aussi vite que possible sur le bouton devant lequel le chenillard s'est arrêté. Si, par exemple, la LED3 est maintenant fixe, le joueur doit appuyer sur le bouton KEY3 pour arrêter le chronomètre. Pour démarrer un nouvel essai, il faut appuyer à nouveau sur KEY0.

Le chronomètre et le chenillard

Le chronomètre se compose de plusieurs compteurs tels que décrits dans les articles précédents. La configuration des compteurs est effectuée dans le paquet `count_pkg`. On configure un compteur pour servir de prédiviseur, dont la valeur de diviseur dépend de la carte utilisée. Outre ce prédiviseur, il y a encore un compteur pour les centièmes de seconde (`csec`),

un autre pour les dixièmes de seconde (`dsec`) et deux pour le comptage des secondes. Comme les valeurs du chronomètre doivent être affichées sur des afficheurs à 7 segments, ces compteurs sont directement configurés en compteurs DCB. Chaque compteur fournit directement les codes nécessaires pour les afficheurs et compte de 0 à 9. Si par exemple le compteur des centièmes de seconde a compté de 0 à 9, son débordement est utilisé comme signal de validation (*enable*) pour le compteur de dixièmes de seconde, qui est alors incrémenté de 1.

Pour que le chenillard soit reconnu comme tel, il ne doit changer d'état ni trop vite ni trop lentement. Un délai de 10 ms semble approprié. L'impulsion de cadencement nécessaire (*enable*) est transmise au circuit de niveau supérieur par le chronomètre avec le signal `t10ms`. Le processus `led_chaser` réalise la fonction du chenillard (**listage 1**). Des variables y sont utilisées.

En VHDL, on peut toujours avoir recours à des variables quand le signal n'est utilisé que localement par son propre processus. L'affectation à des variables s'effectue avec « `:=` » et non « `<=` » comme pour les signaux. Les variables prennent toujours immédiatement la valeur affectée, ce qui n'est le cas des signaux (`signal`) que si l'on décrit une partie combinatoire avec le signal. Sinon, les signaux ne prennent leur nouvelle valeur qu'à l'impulsion d'horloge suivante. Dans le processus du chenillard, la LED à allumer est stockée dans le signal de type `unsigned led_r`. Les variables `dir` et `pos_led` décident de la position de la LED à allumer, à droite ou à gauche de la LED courante. Pour bien voir la différence entre une variable et un signal, on peut essayer d'observer ce qui se passe quand on remplace les variables par des signaux.

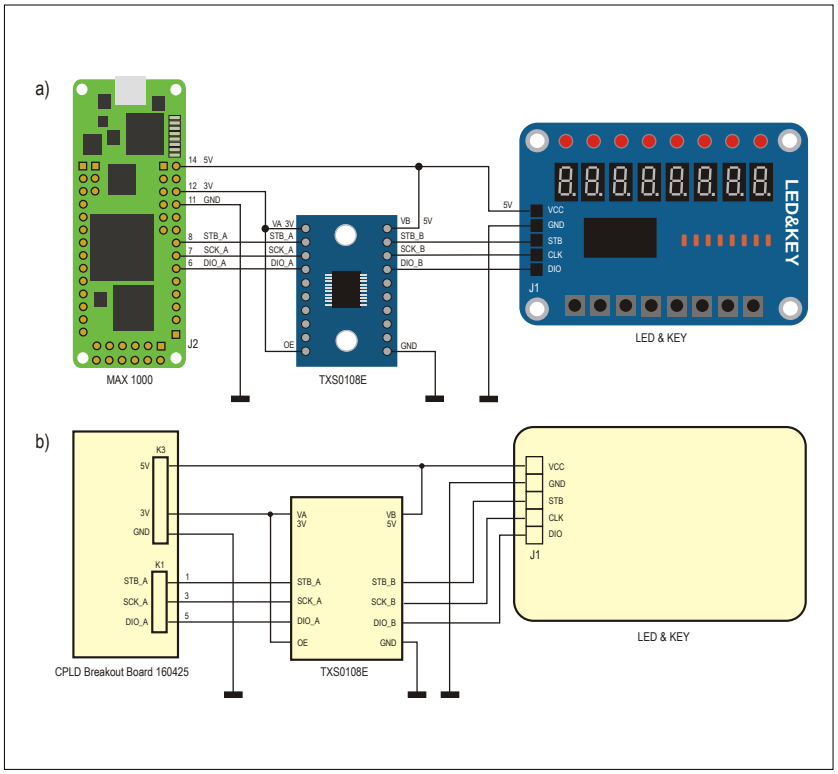


Figure 2. Le convertisseur de niveau convertit les signaux à 3,3 V de la FPGA en 5 V.

Listage 1. Chenillard.

```
led_chaser: process (rst, clk)
  variable led_pos : unsigned(3 downto 0);
  variable dir     : bit;
begin
  if (rst = '1') then
    led_r              <= 8B"0000_0001";
    led_pos            := (others => '0');
    dir                := '0';
  elsif rising_edge(clk) then
    if (t10ms) then
      if (run_chaser = '1') then
        if led_pos = 7 then
          dir := '1';
        elsif led_pos = 0 then
          dir := '0';
        end if;
        if dir = '0' then
          led_r    <= shift_
 left(unsigned(led_r), 1);
          led_pos  := led_pos +1;
        else
          led_r    <= shift_
 right(unsigned(led_r), 1);
          led_pos  := led_pos -1;
        end if;
      end if;
    end if;
  end if;
end process;
```

Le générateur de nombres aléatoires

Pour que le chenillard ne s'arrête pas toujours dans la même position, sa durée d'activité est laissée au hasard. Pour cela, il faut deux composants. L'un est un compteur de délai, qui reçoit à chaque nouveau démarrage sa valeur de fin de comptage, l'autre est un générateur de nombres aléatoires. Lorsqu'on démarre une nouvelle mesure, le nombre aléatoire courant est passé au compteur de délai et son signal de réinitialisation (*reset*) est annulé. À la différence de nos compteurs précédents, ce compteur a alors un *reset* synchrone et son `entity` reçoit la valeur `reset_val` par un port d'entrée. La valeur aléatoire est produite par un générateur PRBS (*Pseudo Random Binary Sequence*). Il s'agit là d'un registre à décalage à 7 bits avec rétroaction (**listage 2**). Ce registre décale continuellement la valeur d'entrée (bit 0) d'une position. Les données à l'entrée sont égales au résultat de la rétroaction, soit le OU exclusif des deux bits de poids fort du registre à décalage (bits 5 et 6). Ces « branchements » sont aussi nommés polynôme de génération. En fonction de ce polynôme, le générateur PRBS peut produire jusqu'à 2^{N-1} séquences de bits différentes. Dans le cas de notre appareil de mesure du temps de réaction, nous avons donc jusqu'à 127 temps d'attente différents possibles. Ce point nous donne une fois de plus l'occasion de revenir sur la réalisation d'un registre à décalage. On commence par exemple

par mémoriser dans deux affectations les « anciennes » valeurs pour ensuite déterminer la nouvelle valeur à la sortie du registre à décalage. Cette méthode est utilisée dans le **listage 2** ainsi que par le maître SPI. VHDL offre encore la possibilité de réaliser un registre à décalage avec des fonctions de la bibliothèque *IEEE.NUMERIC.STD* où sont définies les fonctions `shift_left` et `shift_right`. On peut passer comme premier paramètre à ces fonctions une valeur de type `signed` ou `unsigned`. Comme deuxième paramètre, on passe un entier représentant le nombre de pas de décalage. Selon que le premier paramètre est signé ou non, on charge des zéros (décalage logique) ou le résultat prend le bon signe (décalage arithmétique).

Listage 2. Générateur PRBS.

```
prbs_lfsr: process (rst, clk)
begin
  if (rst = '1') then
    d(6 downto 1) <= (others => '0');
    d(0)          <= '1';
  elsif rising_edge(clk) then
    d(6 downto 1)   <= d(5 downto 0);
    d(0)            <= d(5) xor d(6);
  end if;
end process;
```

Listage 3. Fonction rdata2slv.

```
function rdata2slv(
  rd : in std_logic_vector(31 downto 0)
)
return std_logic_vector is
  variable v_keys   : std_logic_vector(7 downto
 0);
begin
  v_keys := rd(17)& rd(21) & rd(25)
 & rd(29) & rd(19) & rd(23)& rd(27)
 &                   rd(31);
  return std_logic_vector(v_keys);
end;
```

Listage 4. Production de l'horloge SPI.

```
-- f_ena_int = 2 x f_sck
-- use sck_r1 in non delayed version -> CHPA = 0
-- use sck_r2 to delay sck --> CHPA = 1
-- use CPOL=1 to invert sck (switchable inverter)

  sck  <= ((sck_r1 and not CPHA) or (sck_r2 and
   CPHA)) xor CPOL;
  sck_proc : process (clk,rst) is
  begin
   if (rst = '1') then
     sck_r1 <= '0';
     sck_r2 <= '0';
   elsif rising_edge(clk) then
     if (ena_int) then
       if (frame_enable)then
         sck_r1  <= (not sck_r1);
         sck_r2  <= (sck_r1);
       end if;
     end if;
   end if;
  end process sck_proc;
```

La puce de pilotage des LED

La carte LED&KEY est pilotée par une puce immatriculée TM1638 disponible chez Amazon ou en Extrême-Orient pour 5 € environ. Ce composant provient de la société chinoise Titan Micro qui propose toute une série de puces pilotes de LED et d'afficheurs. La feuille de caractéristiques de la TM1638 n'est pas vraiment un modèle de clarté, en partie sans doute à cause des insuffisances de la traduction du texte chinois. Mais comme la carte est très appréciée dans l'environnement Arduino, on trouve sur l'internet de nombreux blogs où le fonctionnement de la carte est nettement mieux expliqué [4][5][6]. La TM1638 comprend 16 registres internes inscriptibles de largeur 8 ou 2 bits. Elle dispose aussi d'un tampon de quatre octets qu'on peut relire. Si l'on veut allumer une LED ou un afficheur à 7 segments, il faut commencer par régler l'intensité lumineuse au moyen d'une commande séparée, puis spécifier le mode d'adressage des registres internes. Il faut enfin écrire les valeurs désirées dans les registres et donner l'ordre d'affichage.
Les valeurs contenues dans les seize registres inscriptibles sont émises en permanence par la TM1638 en mode multiplex. Parmi les seize registres, on combine toujours un registre à 8 bits avec un registre à 2 bits pour former un registre étendu. Pendant l'émission, ces registres étendus sont adressés successivement et le contenu de la « grille » active est présenté sur le bus de données, tout en mettant la broche de sélection de grille au niveau bas. La largeur de l'impulsion de grille fixe ainsi la luminosité de la LED connectée, qui peut donc être réglée indépendamment comme mentionné plus haut. En fin de phase de sortie a lieu la phase de consultation des boutons-poussoirs (*keyscan*), lors de laquelle les sorties K de la puce fournissent l'une après l'autre une courte impulsion de niveau haut. Les lignes *keyscan* sont reliées à travers les boutons et une diode de protection aux broches SEG. Si pendant la phase de balayage, on appuie sur un bouton, le bit correspondant de l'un des quatre registres accessibles en lecture est mis à 1.

Fonctions

La commande de la carte LED&KEY est prise en charge par un automate à états finis simple dans le processus `ctrl_fsm`. Cet automate passe ses commandes, qui sont déclarées dans le paquet `tm1638_pkg` sous forme de constantes, au pilote de matériel de niveau inférieur (maître SPI). On commence par commuter le mode de fonctionnement de la TM1638 sur l'adressage direct et régler la luminosité. Ensuite, on adresse les segments individuels et on leur passe les valeurs correspondant au chiffre à afficher, puis les LED sont successivement allumées et éteintes, en fonction de la valeur du signal `led_r` gérée dans le processus du chenillard. Enfin, on lit les registres d'état des boutons, dont les valeurs sont copiées dans le signal interne `rdata`. Pour retrouver le bouton pressé à partir de la relecture du tampon de quatre octets de la TM1638, on utilise la fonction `rdata2slv`. La fonction `f_bcd_2_segment` se charge de convertir les valeurs DCB du chronomètre en valeurs pour les afficheurs à 7 segments. Comme dans tous les langages de programmation, les fonctions VHDL permettent la réutili-

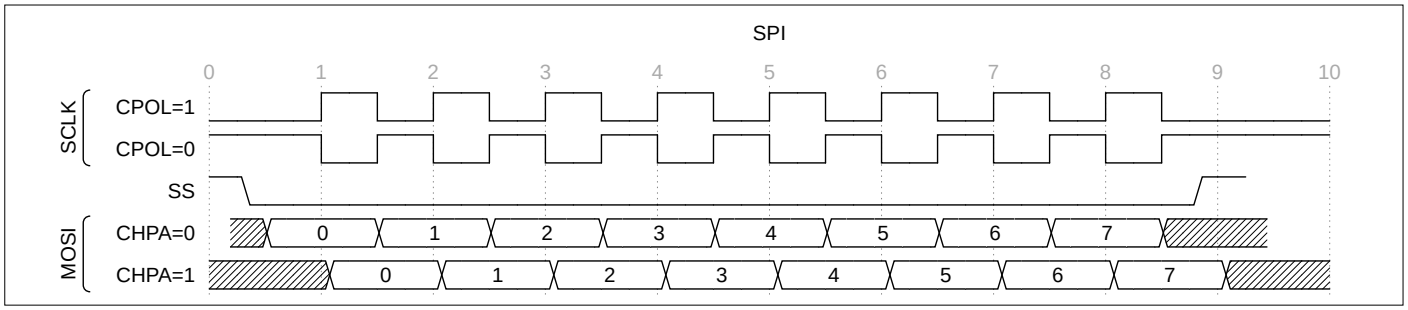


Figure 3. Cadencement des signaux SPI.

sation de parties de code, ce qui rend le code plus compact et plus lisible. De même qu'avec le langage C, les fonctions sont appelées par leur nom, suivi de parenthèses encadrant des paramètres optionnels. Les fonctions peuvent aussi retourner des valeurs. La ligne

```
key_r <= unsigned(rdata2slv(rdata));
```

appelle la fonction `rdat2slv` (**listage 3**), convertit sa valeur de retour (`std_logic_vector`) en une valeur non signée qui est affectée à `key_r`. La conversion en une valeur non signée est nécessaire pour rendre possible sa comparaison avec la valeur de LED courante stockée dans le signal non signé `led_r`. Les deux types `std_logic_vector` et `unsigned` sont des tableaux *unconstrained* de type `std_logic`. Inversement, pour comparer `led_r` et `key_r`, il aurait suffi de convertir `led_r` en type `std_logic_vector` au moyen de la fonction `std_logic_vector(led_r)`.

Maître SPI

La TM1638 est pilotée par trois lignes : ligne d'horloge, ligne d'activation (*strobe*) et ligne de données bidirectionnelle. Un coup d'œil à la feuille de caractéristiques de la TM1638 montre que la transmission de données respecte quasiment le protocole SPI, sauf que les données circulent sur une ligne bidirectionnelle. Techniquement, le standard SPI demeure très simple. Il ne spécifie l'échange de données qu'au niveau le plus bas, c'est-à-dire l'échange d'octets entre un appareil maître et un appareil esclave. Il ne contient aucune spécification électrique (par exemple de niveau de signal). En principe, il y a deux signaux de commande (SCK, SS) et deux lignes de données MOSI (*Master-Out Slave-In*) et MISO (*Master-In Slave-Out*). Il y a des composants qui, comme la TM1638, confondent MISO et MOSI en une seule ligne bidirectionnelle (à drain ouvert). SPI est un protocole sériel synchrone, c'est-à-dire que l'échange de données est régi par un signal d'horloge produit par le maître. Du fait d'une spécification un peu lâche, il existe quatre modes qui se distinguent par la polarité au repos et la position de la phase du signal d'horloge par rapport aux données (**fig. 3**). La feuille de caractéristiques de la TM1638 précise que le mode à employer est CPHA=1 et CPOL=0. Dans ce projet, le maître SPI est réalisé de manière très générique et est donc réutilisable dans beaucoup d'autres projets. Il est composé de quatre domaines fonctionnels principaux. Le diviseur de fréquence dans le processus `ckdiv_proc` divise le signal d'horloge du système et fournit pour chaque top SPI deux impulsions de validation (*enable*). C'est à partir de ce signal que sont dérivées les actions du générateur d'horloge SCK et des FSM des registres à décalage. La **figure 4** (`SPI_master_ctrl_fsm`) montre le diagramme d'état de `ctrl_fsm`. L'automate pilote essentiellement deux registres à décalage. À chaque impulsion *enable,* soit l'automate envoie un bit du registre à décalage TX vers la ligne de données, soit il lit un bit sur la ligne de données pour le pousser dans le registre à décalage RX. Le nombre de bits total à envoyer est spécifié par le signal d'entrée `frame_bytes`. Le signal `first_rx_byte` donne le nombre d'octets après lequel le transfert change de sens. Le processus `sck_proc` s'occupe de produire le signal d'horloge SCK (**listage 4**). Il est composé d'une bascule (`sck_r1`) qui change d'état à chaque impulsion *enable*, et d'une autre bascule (`sck_r2`), qui enregistre la valeur

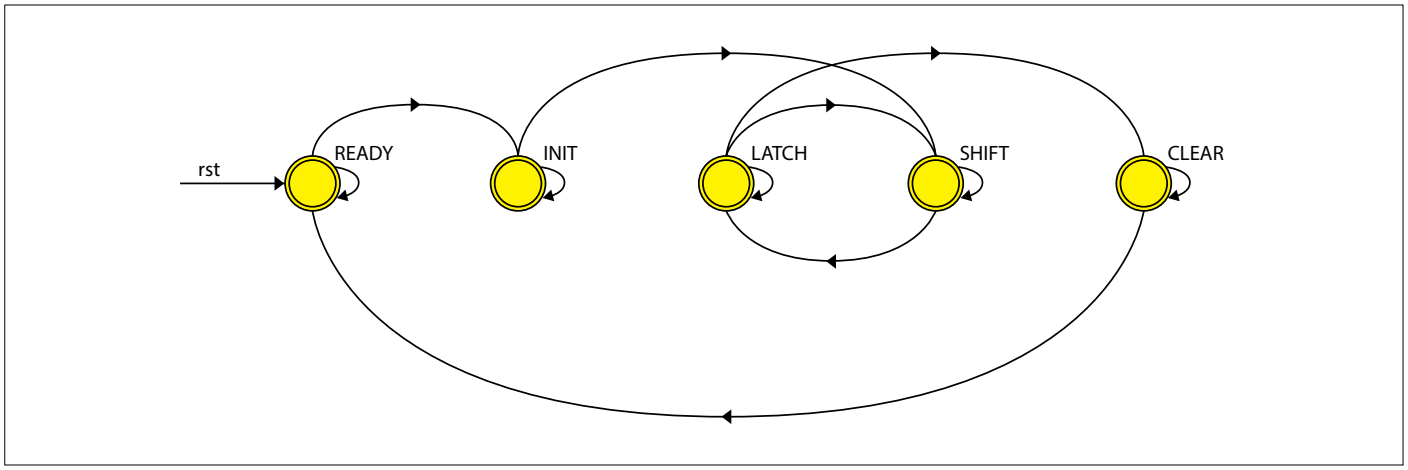


Figure 4. Diagramme d'états de l'automate de gestion du protocole SPI.

de la première et la décale ainsi d'un top d'horloge. Le signal d'horloge SPI à la sortie du maître SPI (sck) est soit la sortie de sck_r1 (CPHA=0), soit celle de sck_r2 (CPHA=1). Avec le signal CPOL, on peut encore inverser la sortie sélectionnée. Le dernier bloc fonctionnel important commande l'inversion de la ligne de données dans le processus (doe_proc). Ce processus, assisté du signal interne d_oe, gère l'inversion du sens de circulation des données sur la broche DIO en fonction de l'état de l'automate et du nombre de bits déjà envoyés (**listage 5**).

Listage 4. Production de l'horloge SPI.

```
-- f_ena_int = 2 x f_sck
-- use sck_r1 in non delayed version -> CHPA = 0
-- use sck_r2 to delay sck --> CHPA = 1
-- use CPOL=1 to invert sck (switchable inverter)

  sck  <= ((sck_r1 and not CPHA) or (sck_r2 and
   CPHA)) xor CPOL;
  sck_proc : process (clk,rst) is
  begin
   if (rst = '1') then
     sck_r1 <= '0';
     sck_r2 <= '0';
   elsif rising_edge(clk) then
     if (ena_int) then
       if (frame_enable)then
         sck_r1  <= (not sck_r1);
         sck_r2  <= (sck_r1);
       end if;
     end if;
   end if;
  end process sck_proc;
```

Listage 5. DIO.

```
-------------------------------------------------
-- Output enable generation for DIO
-------------------------------------------------
din    <= dio  when d_oe = '0' else '0';
dio    <= dout when d_oe = '1' else 'Z';

doe_proc : process (clk,rst) is
variable rx_bit : integer := 0;
variable send_bits : integer := 0;
begin
 if (rst = '1') then
   d_oe      <= '0';
 elsif rising_edge(clk) then
   rx_bit    := to_integer(unsigned(first_rx_
   byte)& "000");
   send_bits := to_integer(unsigned(frame_bytes)&
   "000");
   if (ena_int) then
      if (fsm_state = READY) then
        d_oe      <= '0';
      elsif (fsm_state = INIT) then
        d_oe      <= '1';
      elsif (to_integer(send_bits-bit_cnt) = rx_
   bit and (fsm_state = SHIFT)) then
        d_oe      <= '0';
      end if;
   end if;
 end if;
end process doe_proc;
```

Analyse temporelle TimeQuest

Pour que Quartus puisse convertir le code VHDL en un circuit numérique opérationnel, il faut compléter la description fonctionnelle donnée par VHDL avec d'autres objectifs de conception. Ceux-ci sont enregistrés dans un fichier séparé **.sdc*. SDC est l'acronyme de *Synopsys Design Constraint,* une norme de l'industrie basée sur TCL. Les contraintes les plus importantes concernent le diagramme temporel (outre l'affectation des broches déclarée chez Quartus dans le fichier *.qsf*). La **figure 4** montre un exemple simplifié de chemin parcouru par un signal dans notre projet et le diagramme temporel associé. Soit, par exemple, deux registres FF1 et FF2 représentant une partie de l'état de notre automate qui pilote le déroulement du jeu. Dans l'étape *Place and Route,* Quartus doit placer et connecter les bascules de sorte à garantir que les données envoyées par la première bascule (on parle de *launch-edge*) soient reçues de manière fonctionnellement correcte par la seconde bascule sur le front suivant du signal d'horloge système (*latch-edge*). S'il se produit sur le chemin une violation du *setup*, c'est-à-dire de la stabilité du signal (*negative Slack*, voir plus bas), il peut arriver qu'à cause du temps de parcours, les données qui doivent être enregistrées dans la seconde bascule ne soient pas encore disponibles et que soient incorrectement enregistrées les anciennes données. Une altération du délai de rétention (*Hold Time*) peut se traduire par l'enregistrement immédiat de données qui n'aurait dû se produire qu'à l'impulsion d'horloge suivante. De plus, une *Setup and Hold Time Violation* peut se traduire par le passage de la bascule dans un état indéterminé métastable [8]. Une altération du *setup* ne peut être évitée que si la période d'horloge est suffisamment longue par rapport au temps de propagation des données, c'est-à-dire si l'on a :

```
tclk > tsu + th + tpd + tco
```

tsu : *setup time* ; temps pendant lequel les données doivent être stables avant le front actif.
tpd : *propagation* (parfois aussi *path*) *delay* ; temps de propagation du signal de A à B.
tco : *clock to out* ; temps nécessaire au signal pour apparaître en sortie de la bascule après le front actif.
th : *hold time* ; temps pendant lequel le signal doit rester stable après le front actif.

Pour un *Setup Timing Check,* l'outil d'analyse des temps doit calculer le temps de parcours du signal d'une bascule à l'autre pour chaque chemin interne (*data arrival path*). Pour cela, on prend en considération les temps de parcours à travers le câblage utilisé ainsi que le retard des signaux d'horloge (*clock arrival path*). La valeur qui indique que la condition ci-dessus est satisfaite pour un chemin est appelée *slack* (ici *setup slack*). Elle est positive quand la condition est remplie, négative s'il y a une violation du *timing*. Pour indiquer à Quartus la fréquence d'horloge à utiliser, il suffit d'inclure l'instruction suivante dans le fichier **.sdc* :

```
create_clock -name  clk  -period  83
   [get_ports {clk12m}]
```

On installe ainsi *une Base C*lock avec une période de 83 ns dont les impulsions sont présentes sur le signal `clk12m`. Ceci n'est valable que pour la carte MAX1000 ; pour la carte CPLD, la valeur de la période doit être réduite à 25. Ce premier pas suffit pour la logique interne. Pour le deuxième pas, il nous faut « contraindre » les entrées et sorties SPI, donc informer l'outil qu'il existe une certaine relation temporelle entre les signaux sur les lignes STB, DIO et Clock, qui doit être respectée par le routage dans la puce FPGA/CPLD.
Pour cela, nous définissons une autre horloge appelée spi_clk :

```
create_clock -name  spi_clk -period  $spi_per
   [get_ports {sck}]
```

Comme *le* format .sdc est basé sur TCL, on peut y définir des variables avec set et accéder à leurs valeurs par la référence $ (il est recommandé d'utiliser des noms de variables significatifs). Ensuite, il nous manque encore les relations temporelles entre le circuit extérieur et les chemins d'entrée et de sortie. Cela est réglé par les instructions set_output_delay et set_input_delay. On voit si Quartus est en accord avec les contraintes temporelles spécifiées dans *la fenêtre Compila*tion Report, sous-*répertoire TimeQuest Timi*ng Analyzer, ou directement *dans le Timing* Analyse Tool qu'on démarre en c*liquan*t sur *Tools -> TimeQuest Timing* Analyser. Il n'est pas possible de discuter ici en détail de toutes les contraintes. Toutes les contraintes temporelles nécessaires dans ce projet se trouvent dans *le fichier reac*tionTimer.sdc. Il est important de souligner qu'avec un code VHDL complet, on est encore loin d'un circuit numérique terminé. Pour en savoir p*lus sur les Timing* Constraints, on peut consulter [7][9][10] sur internet.

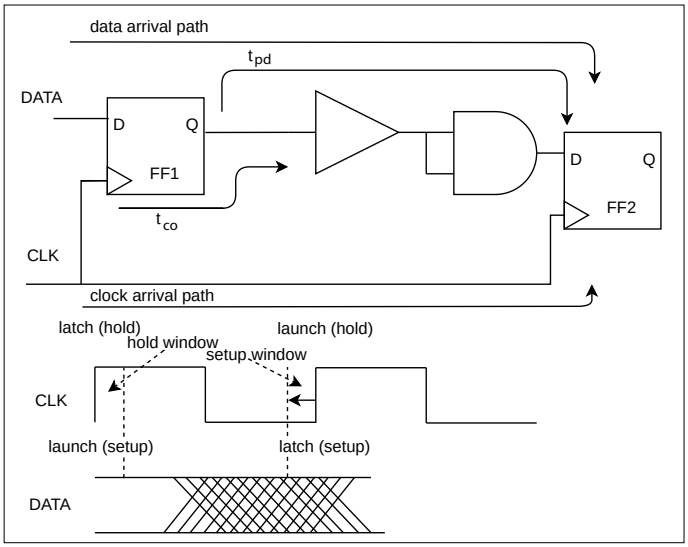


Figure 5. Exemple de chemin de signal dans notre conception et diagramme temporel correspondant.

Conclusion

L'instrument de mesure de temps de réaction de notre série sur VHDL est un bon exemple pour évaluer encore une fois les avantages et les inconvénients d'un langage de programmation de matériel. Pour la réalisation de ce jeu avec un croquis Arduino, il suffit sans doute de 50 lignes de code. Avec VHDL, on ne va pas loin avec 50 lignes de code ; nous en sommes déjà à plus de 1200 au total. Pour bricoler rapidement un circuit de mesure, commande et régulation, on n'envisagera donc certainement pas une CPLD ou FPGA. Mais si un projet exige des vitesses d'entrées-sorties élevées ou un comportement déterministe (ou des temps de latence faibles ou un grand nombre d'unités de traitement en parallèle), alors l'emploi de FPGA peut techniquement s'imposer. Les *processeu*rs softcore tels que dans le projet SCCC dans ce numéro fournissent une synthèse impressionnante des deux mondes. Si vous avez d'autres idées sur la mise en œuvre de VHDL, nous vous encourageons à nous en faire part !

(180285-D-04 - version française : Helmut Müller)

Liens

[1] « conception matérielle avec (V)HDL (3) », Elektor 03-04/2019 : www.elektormagazine.fr/180285-C-04

[2] Page de l'article : www.elektormagazine.fr/180285-D-04

[3] LED&KEY TM1638 : www.amazon.de/RoboMall-TM1638-Taster-8-stelligem-Display/dp/B0117BUAPU

[4] LED&KEY TM1638, démo 1 (en allemand) : https://draeger-it.blog/arduino-lektion-42-tm1638-led-taster-shield/

[5] LED&KEY TM1638, démo 2 (en allemand) : www.forum.g-heinrichs.de/viewtopic.php?t=108

[6] LED&KEY TM1638, démo 3 (en allemand) : www.led-treiber.de/html/leds_display.html

[7] Guide de l'utilisateur de TimeQuest : https://fpgawiki.intel.com/wiki/File:TimeQuest_User_Guide.pdf

[8] Livre blanc (Altera), 'Understanding Metastability in FPGAs' : www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01082-quartus-ii-metastability.pdf

[9] 'Quartus Timing Analysis on set_input_delay and set_output_delay constraints', Eli Billauer : http://billauer.co.il/blog/2017/04/io-timing-quartus-calculation/

[10] 'Constrain SPI Core', wiki Intel FPGA : https://fpgawiki.intel.com/wiki/Constrain_SPI_Core

générateur de fonctions à DDS JDS6600 de JOY-IT

une large gamme de fréquences et de nombreuses options à petit prix

Harry Baggen (labo d'Elektor)

Dans le passé, tout électronicien sérieux possédait un générateur de fonctions à XR2206 dans son labo (amateur). Ce temps-là est révolu. Aujourd'hui, ce type de générateur utilise une puce électronique DDS (*Direct Digital Synthesis* ou synthèse numérique directe) qui, comme son nom l'indique, produit des fréquences à l'aide d'une technologie numérique. Malheureusement, les générateurs de fonctions DDS de qualité ne sont pas réellement abordables. Vraiment ? Nous avons testé pour vous le modèle proposé par JOY-iT, qui annonce une large gamme de fréquences et de très nombreuses options à un prix optimisé. Y aurait-il un loup ?

Commençons tout de suite par les mauvaises nouvelles. Le boîtier du générateur de fonctions JOY-iT JDS6600 a un aspect assez « bon marché » et le fabricant aurait pu investir davantage de ce côté. La bonne nouvelle maintenant : pour moins de 140 €, l'appareil couvre une large gamme de fréquences, est assez précis et offre toute une palette d'options de réglage. Nous y reviendrons.

Matériel

Le générateur de JOY-iT est installé dans un simple boîtier en matière plastique de dimensions modestes. La face avant comporte un écran LCD de petites dimensions, mais lumineux, pour visualiser tous les réglages, ainsi qu'un certain nombre de boutons de commande, un bouton rotatif et trois prises BNC, avec deux sorties et une entrée de fréquencemètre. L'appareil peut donc produire deux signaux de sortie réglables de manière indépendante ou appairés. Un adaptateur secteur livré avec l'appareil fournit son alimentation. La livraison comporte également deux câbles BNC avec pinces crocodiles, un câble BNC-BNC et un câble USB, ce qui est peu usuel pour un appareil de ce prix.

Le générateur JDS6600 atteint 60 MHz pour les signaux sinusoïdaux, 15 MHz pour les signaux carrés et triangulaires et 6 MHz pour toutes les autres formes d'ondes. Le système permet de sélectionner près de 15 formes d'ondes prérogrammées et jusqu'à 60 formes d'ondes reprogrammables. La tension maximale de sortie est de 20 V_{cc} au-dessous de 10 MHz. Elle est de 10 V_{cc} (jusqu'à 30 MHz) ou 5 V_{cc} (jusqu'à 60 MHz). Les valeurs correspondant à la tension de décalage réglable sont similaires. En outre, le générateur JDS6600 permet de réaliser des salves de signaux et des balayages de fréquences. Le fréquencemètre intégré convient pour des signaux atteignant 100 MHz et 2 à 20 V_{cc}.

Pratique

Le fonctionnement du JDS6600 s'appuie sur une disposition soignée : quatre touches de fonction à proximité de l'écran (la fonction active dépend des paramètres sélectionnés). L'écran indique la configuration des deux voies, et le signal représenté dans la partie haute montre également la forme du signal sélectionné. La plupart des touches ont plusieurs fonctions, c'est un peu déroutant au début. Par exemple, si vous appuyez une fois sur une touche *CH*, vous basculez sur l'autre voie pour la configurer. Si vous appuyez sur une voie déjà sélectionnée, celle-ci est désactivée. En outre, une pression prolongée sur la touche *CH2* permet l'affichage de la voie 2 dans la partie haute de l'écran, avec sa forme d'onde. Enfin, si vous maintenez la touche *CH1* enfoncée, la voie 1 revient en haut de l'écran. C'est clair ?

Les touches fonctionnent parfaitement, à ceci près que lorsque vous appuyez, les touches adjacentes se déplacent également, ce qui aurait pu être résolu par une solution mécanique plus robuste. Les valeurs affichées à l'écran se définissent d'abord en sélectionnant un nombre à l'aide des deux touches de curseur, puis en fixant la valeur à l'aide du bouton rotatif. Selon la fréquence, l'affichage des chiffres est dense, ce qui rend le réglage un peu difficile. Le bouton *OK* permet d'activer ou de désactiver les deux sorties (sa dénomination est un peu

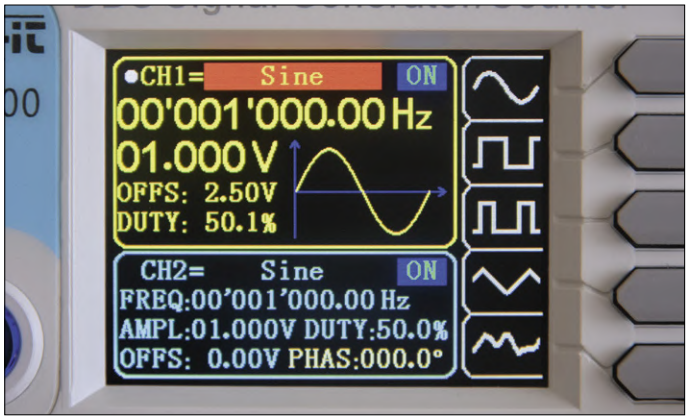


Figure 1. Tous les réglages relatifs aux deux signaux de sortie apparaissent à l'écran.

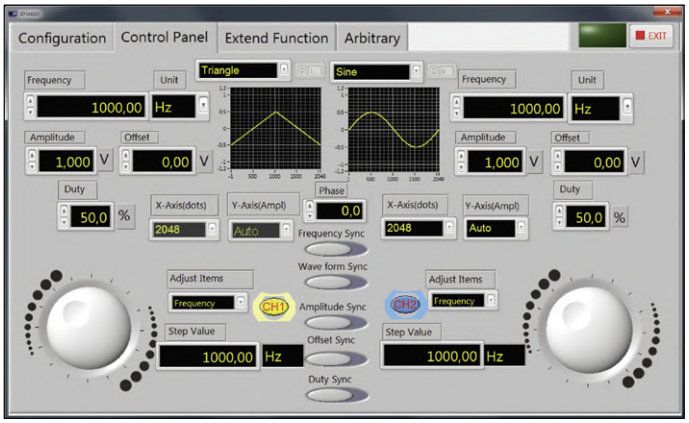


Figure 2. La configuration est également possible sur un PC.

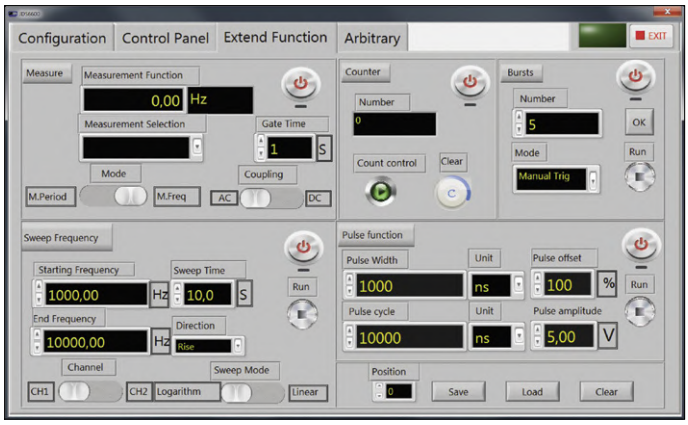


Figure 3. Cet onglet contient toutes les fonctions supplémentaires disponibles, notamment les balayages de fréquence.

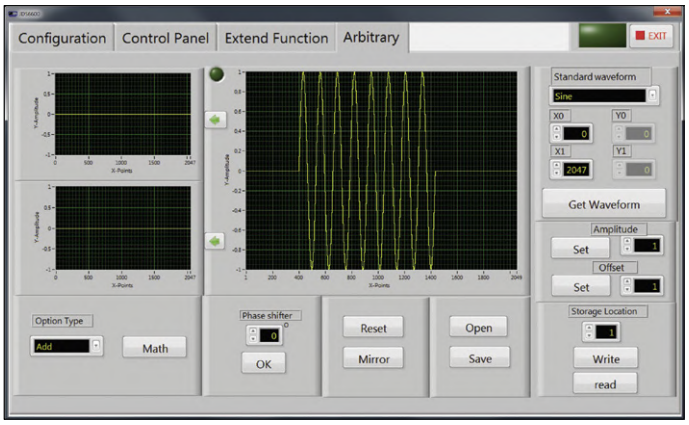


Figure 4. La programmation spécifique d'une forme d'onde est plutôt facile.

étrange pour cette fonction). Le bouton *MOD* permet de régler les balayages, les salves et les largeurs d'impulsions, mais sans modulation. C'est peu de chose, et l'appareil est globalement très facile à utiliser une fois ces particularités identifiées.

Les signaux de sortie sur l'écran de l'oscilloscope ont un aspect plutôt satisfaisant et conforme aux spécifications (fréquence d'échantillonnage 266 Méch./s, longueur d'onde 2048 points, résolution 14 bits), même si je m'attendais à des signaux plus lisses. Les mesures en FFT d'une onde sinusoïdale dans le domaine audio montrent des résidus de distorsion un peu supérieurs à 1%. Le constructeur mentionne moins de 0,8%, que je n'ai pas atteint avec mon échantillon de test. Les ondes carrées, par ailleurs, ont plutôt bel aspect, avec des fronts assez raides et peu de dépassement transitoire. La progression des fréquences est quasiment linéaire : au-dessus de 30 MHz, la tension de sortie augmente d'environ 2 dB, puis diminue d'environ −1 dB à 60 MHz. Pas mal du tout !

Vous pouvez télécharger sur le site web de JOY-iT un programme [1] qui permet de faire fonctionner le générateur JDS6600 depuis votre PC à l'aide d''une connexion USB. Comme pour la plupart des logiciels venus de Chine, la conception n'est pas exceptionnelle, mais tout fonctionne parfaitement. Vous disposez de plusieurs onglets pour les différentes fonctions, notamment celui qui permet de composer ses propres formes d'onde. Après l'avoir essayé un certain temps, j'ai créé en un rien de temps une salve sinusoïdale pour tester des enceintes.

Conclusion

Pour moins de 140 €, le générateur de fonctions JOY-iT JDS6600 a beaucoup à offrir. Un certain nombre de points pourraient être améliorés, mais au final vous en avez vraiment pour votre argent. Vous n'avez certes pas de possibilité de modulation, mais tout le reste permet de produire les signaux dont on a besoin occasionnellement dans un laboratoire amateur. Est-il vraiment nécessaire d'acheter un appareil professionnel qui aurait pour lui un élégant boîtier, mais au prix d'une facture nettement plus élevée ?

(190310-03)

Lien

[1] Programme pour PC : http://anleitung.joy-it.net/?goods=jds6600

des trésors ou des déchets électroniques ?

à vous de voter !

Jan Buiting (rédacteur de la rubrique Rétronique)

Les instruments de mesure disparaissent de nos établis ; on préfère numériser puis traiter les signaux avec du logiciel, ou tout simplement simuler. Des appareils de précision – analogiques ou du début de l'ère numérique – prennent désormais la poussière dans des greniers ou des placards. J'ai pu récupérer quelques appareils il y a peu (voir photos) : lesquels voudriez-vous que nous réparions ?

Les appareils décrits ci-après ne représentent qu'un échantillon de ce qui remplissait presque le coffre de ma Peugeot 308 SW, pourtant assez grand (1 775 l). On m'avait dit qu'ils étaient défectueux d'une manière ou d'une autre, mais il s'est avéré qu'il ne s'agissait pour la plupart que d'un problème d'étalonnage. Les appareils peuvent en effet être fournis avec un certificat d'étalonnage, qui expire après quelques années. L'instrument ne peut alors plus être utilisé pour des mesures scientifiques dans un laboratoire officiel, et si on ne refait pas l'échantillonnage, il est mis au rebut.
Certains semblaient effectivement en panne : aucune réaction à la mise sous tension. Notre labo a réussi à réparer et restaurer pas mal d'appareils par le passé, il suffit de relire d'anciens articles de la rubrique Rétronique pour s'en convaincre. Nous voudrions maintenant sonder les lecteurs, pour savoir si la remise en état complète d'un des appareils les intéresserait ; à vous de voter !

- Adresse : jan.buiting@elektor.com
- Objet : trésors ou déchets électroniques ? (*Treasures or E-Debris?)*

Si en plus vous avez une certaine expérience en la matière (ce n'est pas toujours évident de savoir par où commencer), votre aide sera appréciée.
Les appareils ont été nettoyés, et nous n'avons décelé aucun problème majeur de sécurité. Dans la mesure du possible nous avons obtenu les manuels et quelques avis d'experts. Chaque instrument est brièvement décrit, et je donne un avis (éclairé) quant aux pannes probables. Les piles vont de gauche à droite, les appareils de bas en haut.

1ère pile

Tektronix Type 11802 Digital Sampling Oscilloscope
Cet oscilloscope, qui était de haut de gamme – et très cher,

date de la seconde moitié des années 90. Cet exemplaire possède un module d'extension SD-24 (24 GHz). Son certificat d'étalonnage a expiré en 2008.
État : écran toujours bon ; message d'erreur lors de l'autotest « Error E1811, Subsys Executive ».
Pannes possibles : connecteurs, condensateurs électrolytiques (CMS), batterie de la mémoire non volatile (NVRAM).

Philips PM 2436
DC-Micrometer
Cet instrument, de la fameuse série « grise » PM 24xx, date du début des années 70. Il est resté pendant longtemps indispensable dans les labos de physique et de chimie pour des mesures de courant de faible intensité.
État : mise sous tension normale ; l'aiguille part en butée (*full scale deflection*) sur toutes les gammes et dans tous les modes.
Pannes possibles : soudures, condensateurs électrolytiques, semi-conducteurs.

2e pile
Marconi Instruments TF2173 Digital Synchronizer & TF2016 AM/FM 10 kHz – 120 MHz Signal Generator
Un couple idéal des années 80 pour les mesures en HF, avec boucles à verrouillage de phase et oscillateur à quartz thermostaté, mais seuls les grands labos pouvaient se l'offrir.
État : dégagement de chaleur et d'un peu de fumée à l'arrière du TF2016 à la mise sous tension ; verrouillage impossible et fréquence instable sur toutes les gammes.
Pannes possibles : alimentations défectueuses.

Philips PM 2504 Electronic VAΩ Meter
Un multimètre portable et alimenté par batterie de la série « noire » PM 25xx. L'adaptateur secteur permettait aussi la charge de la batterie.
État : mesures de tension et de courant a priori correctes ; l'aiguille part en butée sur les gammes de mesure de résistance.
Pannes possibles : ???

3e pile
Schomandl ND100M Frequency Decade 300 Hz – 100 MHz
Cet appareil imposant (27 kg) des années 60-70, garanti avec « rigueur germanique » (*Deutsche Gründlichkeit*) promettait une résolution de 0,1 Hz grâce à un oscillateur à quartz thermostaté ultra-stable.
État : fréquence et amplitude instables ; indicateur d'alarme hors d'état.
Pannes possibles : alimentations et/ou oscillateurs à quartz défectueux.

Tektronix TDS 520B Two-Channel Digitizing Oscilloscope
État : autotest passé avec succès ; distorsion importante sur le Canal 2, et pas de différence entre modes AC et DC.
Pannes possibles : condensateur d'entrée du Canal 2, condensateurs électrolytiques (CMS).

4e pile
Advantest R9211A Digital Spectrum Analyzer
Cet instrument « portable » couvre la gamme de 10 mHz (ce n'est pas une faute de frappe !) à 100 kHz et permet l'analyse de spectre par transformée de Fourier rapide (*FFT*) en temps réel. Le modèle R9211A possède une fonction de zoom avec résolution de 10 ou 100 mHz, et est un des moins chers de la série.
État : le fusible de l'alimentation secteur (2 A) grille.
Pannes possibles : alimentations défectueuses. Nous n'avons pas le manuel ; si vous pouvez nous aider…

Keithley Instruments 410A Picoammeter
Un instrument simple et populaire qui, avec sa gamme de 0,3 pA (10^{-12}), fait encore mieux que le Philips PM 2436. On est presque aux fA magiques (10^{-15}, billiardième d'ampère)…
État : mesures erratiques sur toutes les gammes.
Pannes possibles : ??? ⏮

(180574-04 – version française : Jean-Louis Mehren)

ESTD 2004
www.elektor.tv

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

QUESTIONS D'ÉTHIQUE

Fab Academy

cours intensif de fabrication numérique

Tessel Renzenbrink

Une machine à cocktails automatique (française), une machine à calligraphier pilotée par ordinateur (japonaise) ou une machine à commande numérique (CNC) (péruvienne). C'était la 17e semaine de la Fab Academy, et des étudiants du monde entier ont exposé leurs réalisations dans une salle de classe virtuelle. Chaque équipe avait deux minutes pour présenter son projet en ligne. Le professeur Neil Gershenfeld, qui enseigne à l'académie, prodiguait conseils et compliments, ou – plus rarement – quelques remontrances : « Il y a encore du pain sur la planche… ».

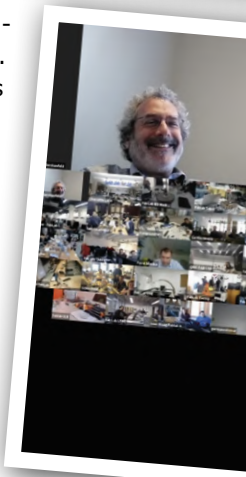
La Fab Academy n'a rien à cacher, et tout se trouve sur https://fabacademy.org, que ce soit les vidéos de cours du professeur Gershenfeld ou la documentation des projets de tous les étudiants qui y ont un jour participé.
Nous avons visité la Fab Academy à la Fondation Waag à Amsterdam [1]. Ce cours de la semaine 17, aussi appelé le tour du monde (*lap around the world*), illustre bien le parcours à la Fab Academy [2]. En vingt semaines, les étudiants apprennent comment on peut (à peu près) tout réaliser soi-même : la découpe au laser, les montages électroniques, la programmation de systèmes embarqués ou encore l'impression 3D. Le tout à raison d'une nouvelle compétence par semaine, qu'il faut bien entendu mettre en œuvre dans un projet, qui devra aussi être testé, débogué et documenté. Henk Buursen, mentor à l'académie d'Amsterdam, rappelle que c'est vraiment très intense : « 32 h par semaine, et plus pour beaucoup. On se lève avec l'académie en tête, et elle y est encore lorsqu'on se couche ! ». « Sans parler des rêves… », nous confie Rutger, un étudiant.

Du Fab Lab à la Fab Academy

La Fab Academy est une émanation du Fab Lab (*Fabrication Laboratory*). L'atelier dispose d'appareils commandés par ordinateur, comme des machines CNC, des découpeuses à laser, ou des imprimantes 3D, le but étant d'apprendre à tout un chacun à utiliser les techniques de fabrication numériques. Le concept de Fab Lab est né au *Center for Bits and Atoms* du MIT en 2001, sous la direction du professeur Gershenfeld ; aucune création n'en est alors sortie, mais c'est bien le labo lui-même qui s'est échappé des murs de l'université. Il y a depuis plus de 1 000 Fab Labs dans le monde. En 2003, Neil Gershenfeld a eu l'idée d'apprendre aux étudiants du MIT à se servir des instruments du Fab Lab ; il s'agissait d'un cours d'un semestre, intitulé *How To Make (almost) Anything*. En 2009, les utilisateurs d'autres Fab Labs ont pu suivre le cours via un canal vidéo, à titre d'essai ; la Fab Academy était née !
Entretemps le principe de formation à distance a été remplacé par l'apprentissage réparti *(distributed learning*). Les connaissances sont échangées hors des frontières, mais aussi localement : les étudiants peuvent alors apprendre les uns des autres et de leur mentor. Henk et ses étudiants se réunissent chaque jeudi au labo de la Fondation Waag pour s'entretenir de leurs projets et de l'utilisation des machines. Les différentes académies se « réunissent » virtuellement pour que tous puissent partager leurs expériences et documenter les projets plus en détail.

Un concept ouvert

Le partage des projets est essentiel à la Fab Academy ; cette ouverture permet à tous d'apprendre des autres. Le principe est semblable à celui du logiciel à code source ouvert, et chaque projet peut ainsi être utilisé, modifié ou amélioré par tout un chacun. Ceci poursuit un autre but que le partage des connaissances : la possibilité de produire localement. Plus besoin d'exporter des conteneurs entiers pleins d'appareils divers : on partage le projet et on fabrique au Fab Lab.



Des machines pour créer des machines

« L'idée maîtresse de l'académie est bien la production à titre individuel », nous dit Henk, et il ajoute : « Nous sommes devenus des consommateurs, incapables de produire quoi que ce soit. Avec l'équipement d'un Fab Lab, on peut fabriquer pratiquement tout ce qu'on veut, et ce n'est pas nécessairement onéreux. Les machines étaient autrefois très coûteuses et prenaient beaucoup de place ; il aurait fallu plus d'un million de dollars pour un Fab Lab. L'inventaire actuel d'un atelier [3] est estimé à un peu plus de 100 000 $, et on vise à terme les 10 000 $, en fabriquant les machines pour équiper un nouveau Fab Lab avec celles d'un atelier existant.

C'est d'ailleurs le projet de l'académie du Pérou : fabriquer une machine à commande numérique ».

Devenir producteur

Henk nous décrit ensuite le processus : « Le cours semestriel (payant, 5.000 € par Fab Lab) est réparti en vingt modules, ce qui permet d'apprendre à utiliser l'inventaire complet de l'atelier. On regagne son autonomie en passant directement de l'idée à la production, et on sait exactement comment ça marche. Je trouve par ex. important que les jeunes démontent entièrement un ordinateur pour en comprendre son fonctionnement. Lorsqu'on sort un appareil de sa boîte, on ne s'intéresse pas à ce qu'il contient, et c'est une des influences néfastes de la technologie : on a de moins en moins de prise sur ce qui nous entoure. C'est un de mes buts principaux en tant que mentor à la Fab Academy : l'autonomisation (*enpowerment*), en apprenant le fonctionnement intime des appareils. Si on possède les connaissances et l'outillage ad hoc, on peut comprendre et refaçonner ce qui nous entoure ».

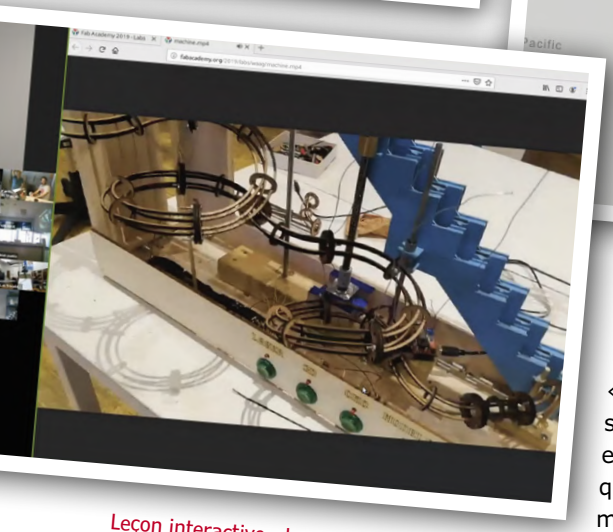
Apprentissage intensif

Suivre un cours à la Fab Academy n'est pas une sinécure conclut Henk : « Vous pouvez lire le manuel d'utilisation d'une machine, ou je peux vous expliquer son fonctionnement, mais dès que vous appuyez sur le bouton, elle se met à faire autre chose que ce que vous vouliez. C'est à vous de rechercher la cause et de résoudre le problème ; il faut être inventif et oser prendre des décisions. La pression est forte à l'académie, et on apprend très vite à déboguer. On a tous eu un projet qui ne fonctionnait pas correctement, et qu'on a laissé sur le côté pendant plusieurs mois, en espérant sans doute qu'un miracle allait se produire… Quel que soit le niveau lorsqu'on commence le cours, l'important est de réussir à apprendre quelque chose de nouveau, et à se dépasser. Je ne connais pas un seul étudiant qui durant les six mois n'ait fait une dépression, une crise de larmes ou de panique… ».

À la Fab Academy d'Amsterdam, nous pouvons assister au « tour du monde » (*lap around the world*) du professeur Gershenfeld : « Fab Lab Aachen, do you have a machine? ». Soupir de soulagement dans le local. « C'est par ordre alphabétique », dit Rutger, sachant qu'il a encore presque deux heures pour des corrections de dernière minute, avant de présenter le projet en public.

(180572-04 – version française : Jean-Louis Mehren)

Rutger, un étudiant, met la touche finale à un projet de groupe. Ses condisciples suivent d'autres démos en temps réel par vidéo.



Leçon interactive : le professeur Gershenfeld est en haut à gauche, la machine de Waag à droite, les canaux vidéos des académies participantes en bas à gauche.

Carte des Fab Labs du monde entier.

Fab Academy

La Fab Academy n'a rien à cacher, et tout se trouve sur https://fabacademy.org, que ce soit les vidéos de cours du professeur Gershenfeld ou la documentation des projets de tous les étudiants qui y ont un jour participé.

Liens

[1] Fondation Waag : https://waag.org/en/project/fab-academy

[2] Vidéo d'un cours mondial : https://vimeopro.com/academany/fab-2019/video/338245042

[3] Inventaire complet du matériel d'un Fab Lab : https://is.gd/XauCaK

bienvenue dans votre e-choppe

la rédaction recommande

Earth Listener de Vellemann

L'Earth Listener est un appareil qui mesure différentes valeurs environnementales pour indiquer la qualité de l'air. Il lit la température, le taux d'humidité, la pression atmosphérique, le taux de CO_2 et le taux de substances organiques volatiles (TVOC, *Total Volatile Organic Compound*) ; le résultat des mesures est affiché sur un écran tactile TFT et peut être sauvegardé sur une carte mémoire SD.

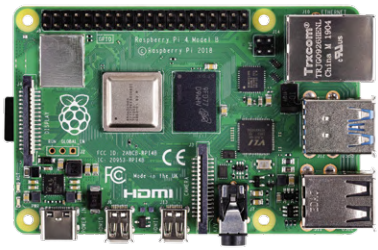
L'Earth Listener est un appareil pratique et bien conçu, qui trouvera sa place aussi bien à la maison qu'au bureau. Il est prêt à l'emploi tel quel, mais il offre également des possibilités d'extension : à vous d'imaginer lesquelles...

Luc Lemmens (labo d'Elektor)

www.elektor.fr/velleman-earth-listener-kit

Vos favoris :

1. Raspberry Pi 4 B
   www.elektor.fr/rpi4



2. SDR Hands-on Book (*livre en anglais*)
   www.elektor.fr/sdr-hands-on-book
3. Kit PCBite
   www.elektor.fr/pcbite-kit
4. *Shield* Elektor SDR 2.0
   www.elektor.fr/170515-91
5. Raspberry Pi Zero WH
   www.elektor.fr/rpi-zero-wh
6. Oscilloscope USB SmartScope
   www.elektor.fr/smartscope
7. Camera Projects Book (*livre en anglais*)
   www.elektor.fr/camera-projects-book

Primo de Volumio

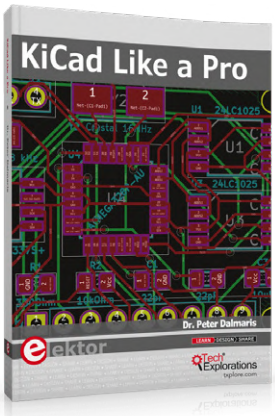
Primo est un lecteur de musique diffusée en *streaming*. Il est animé par un logiciel développé par Volumio, son fabricant. Ce boîtier permet de lire des fichiers musicaux stockés sur un disque dur USB local ou sur un serveur multimédia en réseau. Il est également possible de récupérer le flux des radios de l'internet et d'autres services de *streaming*. Selon vos préférences, Volumio est capable de livrer la musique sous forme numérique ou analogique.

Prix (membres) : 431,10 €

www.elektor.fr/volumio-primo

KiCad Like a Pro

(livre en anglais)



Ce livre (en anglais) vous apprendra à utiliser KiCad. Que vous soyez amateur ou ingénieur en électronique, grâce à ce livre, vous serez rapidement en mesure de dessiner vos schémas électroniques et de concevoir vos circuits imprimés. L'approche pratique de cet apprentissage repose sur quatre projets de difficulté progressive.

Prix (membres) : 35,96 €

www.elektor.fr/kicad-like-a-pro

SDR Hands-on Book

(livre en anglais)

Le *shield* SDR d'Elektor (réf. 18515) permet de construire un récepteur à ondes courtes jusqu'à 30 MHz. Avec une carte Arduino et le logiciel approprié, il est possible de recevoir des stations radio et SSB, du morse et des signaux numériques. Dans ce livre en anglais, Burkhard Kainka décrit comment pratiquer la radio logicielle avec le *shield* SDR d'Elektor, donne une formation théorique et détaille l'utilisation d'outils logiciels libres.

Prix (membres) : 35,96 €

www.elektor.fr/sdr-hands-on-book

nouveau

Camera Projects Book

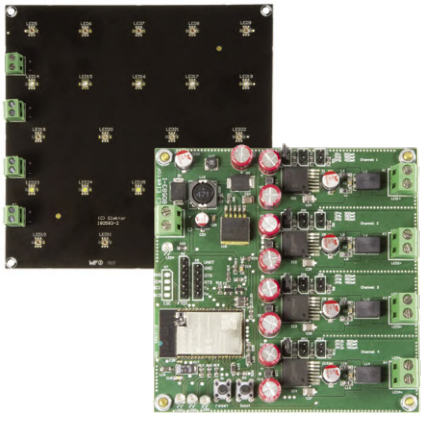
Ce livre (en anglais) explique en termes simples et avec des exemples testés et fonctionnels, comment configurer et utiliser une caméra Raspberry Pi et une webcam USB dans des projets qui combinent vision et Raspberry Pi.

Les exemples montrent comment capturer des images, prendre des photos à intervalles réguliers, enregistrer ou diffuser en direct des vidéos, créer des applications de sécurité et de surveillance, poster des images sur Twitter, enregistrer des animaux sauvages, utiliser une caméra à vision nocturne, envoyer des photos à des ordiphones, détecter des visages et des yeux, reconnaître des formes et des couleurs, lire des plaques minéralogiques et les codes à barres, etc.

Prix (membres) : 26,96 €

www.elektor.fr/camera-projects-book

Horticulture Box



Faites pousser vos plantes plus rapidement grâce à ce kit d'éclairage horticole à LED. Le kit contient les deux circuits imprimés garnis. Il faut une alimentation externe de 24 V, 50 W pour alimenter les cartes. Attention : ne regardez jamais directement les LED lorsqu'elles sont allumées ! Même à des niveaux de luminosité apparemment faibles, l'intensité de la lumière pulsée est très élevée et peut endommager vos yeux.

Prix (membres) : 148,46 €

www.elektor.fr/horticulture-box

The Complete ESP32 Projects Guide

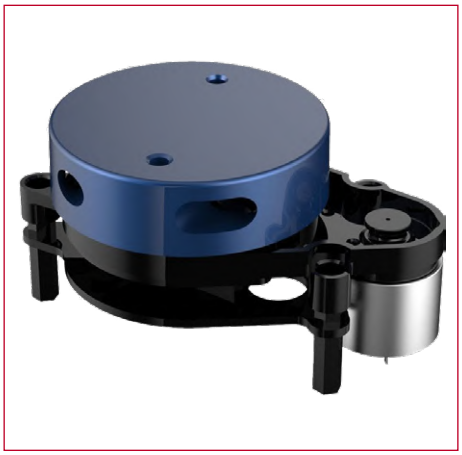
(livre en anglais)

L'objectif de ce livre en anglais est d'enseigner la programmation avec l'EDI Arduino et le langage MicroPython à l'aide de projets basés sur l'ESP32, en utilisant la très populaire carte de développement DevKitC ESP32. Les nombreux projets sont de niveau de difficulté croissant. Ils ont tous été testés et fonctionnent. Les explications sont complétées par des schémas de principe, les schémas des montages et les programmes.

Prix (membres) : 35,96 €

www.elektor.fr/esp32-projects-guide

YDLIDAR X2 – télémétrie au laser à 360°



YDLIDAR X2 est un système Lidar de télémétrie à 360°. Cet instrument est basé sur le principe de la triangulation. Ses composants optiques et électroniques ainsi que ses algorithmes lui permettent de réaliser des mesures de distance avec une grande précision. La mesure de distance est complétée par des informations sur le balayage à 360° puisque l'angle de rotation est continuellement indiqué

Prix (membres) : 71,96 €

www.elektor.fr/ydlidar-x2

Hexadoku casse-tête pour elektorniciens

Votre magazine se termine toujours et encore par une grille de sudoku particulière. Éteignez le fer à souder, oubliez les écrans de PC et d'oscilloscope, rangez les pinces et les fils. Faites travailler votre matière grise d'une autre façon : attrapez un crayon pour remplir la grille d'hexadoku.
N'oubliez pas d'envoyer votre réponse pour tenter de gagner un bon d'achat.

Une grille hexadoku est composée de chiffres du système hexadécimal, de 0 à F. Remplissez le diagramme de 16 x 16 cases de telle façon que **tous** les chiffres hexadécimaux de 0 à F (0 à 9 et A à F) n'apparaissent **qu'une seule et unique fois** dans chaque rangée, colonne et carré de 4 x 4 cases (délimités par un filet gras).

Certains chiffres, déjà placés dans la grille, en définissent la situation de départ.
Pour participer, inutile de nous envoyer toute la grille, il suffit de nous donner **la série de chiffres** sur fond grisé.

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

Où envoyer votre réponse ?
Envoyez votre réponse (les chiffres sur fond grisé) avec vos coordonnées par courriel, avant le **16 septembre 2019** à l'adresse **hexadoku@elektor.fr**

Les gagnants

La solution de la grille du numéro de juillet/août 2019 est **70DA8**.
Les cinq bons Elektor d'une valeur de **50 €** vont à : Michel **Jamin** (France), Peter **Maarse** (Pays-Bas), Vladimir **Saric** (Serbie), Ulrich **Schoor** (Allemagne), Peter **Wäckerle** (Suisse) .

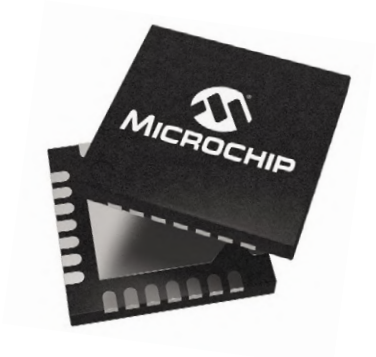
Bravo à tous les participants et félicitations aux gagnants !

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

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

Faîtes votre choix : n'importe quel coeur, n'importe quel niveau de performance, n'importe quel jeu de fonctions !

Performances évolutives selon les variations de vos besoins



Etes-vous (encore) confronté à des exigences de conception changeantes ? Laissez Microchip vous aider à mettre fin aux frustrations et aux pertes de temps qu'engendrent ces changements. Microchip est le seul fournisseur de semiconducteurs qui innove à la fois dans les microcontrôleurs 8, 16 et 32 bits, les contrôleurs de signaux numériques, et les microprocesseurs. Nos architectures à compatibilité ascendante préservent le temps et les ressources que vous avez investis dans le développement de code. En outre, nos outils de développement vous permettent de tirer parti d'un même écosystème sur plusieurs projets. L'évolution des exigences de conception ne doit pas nécessairement être douloureuse ! Découvrez comment Microchip peut vous aider à la vivre.

Simplifiez-vous la vie en allant sur www.microchip.com/Scalable

MICROCHIP

Forum de l'électronique

24 > 26 SEPT 2019

PARIS EXPO
PORTE DE VERSAILLES
HALL 4 / FRANCE

LE SALON DE L'INNOVATION ET DES SOLUTIONS ÉLECTRONIQUES

www.forum-electronique.com

Made by GL events

MÊME LIEU / MÊME DATE

Measurement world

19e CONGRÈS INTERNATIONAL DE MÉTROLOGIE CIM2019