interface audio **USB-S/PDIF**

Sortie audio numérique pour ordinateur, ordinateur portable, tablette etc.

Stephan Lück (Allemagne)

La plupart des PC, ordinateurs portables, tablettes et téléphones tactiles n'ont pas à proprement parler de sortie audio ; c'est au mieux un connecteur USB polyvalent et une sortie casque. Comment connecter un tel appareil à un ampli de qualité ou à un récepteur AV ? Voici une réponse sérieuse à cette question : une interface USB-S/PDIF de qualité.

INFOS SUR LE PROJET

Mots-clés
audio numérique, PC, ordinateur portable, tablette, téléphone tactile

Niveau
débutant – connaisseur – expert

Durée
environ 4 h

Outils
outils de soudure pour CMS et traversants, outils mécaniques

Coût
35 € à 40 €

CARACTÉRISTIQUES

- tension : +5 V
- courant : 36 à 43 mA
- µC : PIC32MX27F256B-I
- 3 fréquences d'échantillonnage : 44,1, 48 et 96 kHz
- format audio : 16, 20 et 24 bits
- sorties S/PDIF optiques et électriques
- télécommande IR (et RC5)
- compatible avec Windows 7/10, Linux, Android et Raspbian

S/PDIF ou S/P-DIF pour *Sony/Philips Digital Interface Format* est une interface de données audio numériques de courte distance entre modules voisins, dans un salon de cinéma ou de hi-fi. Cette interface unidirectionnelle sérielle est dépourvue d'horloge séparée ; l'horloge est extraite du signal lui-même. L'idée est de conserver les données audio dans le domaine numérique aussi longtemps que possible et de ne les convertir en analogique qu'au dernier moment.

S/PDIF [1] est basé sur la norme professionnelle AES3 ; le protocole des deux normes est compatible, mais leurs caractéristiques électriques divergent. Une interface AES3 professionnelle typique utilise un connecteur XLR à 3 voies et un câble blindé à paires torsadées symétriques d'une impédance de 110 Ω. Le niveau du signal varie de 3 à 10 $V_{càc}$ (AES/EBU 2 à 7 $V_{càc}$). Moins courants sont les connecteurs BNC avec un câble coaxial de 75 Ω. Les AES3 symétriques avec connecteurs XLR (paire torsadée blindée) peuvent être utilisés jusqu'à 1000 m de distance, 100 m pour la version coaxiale.

Deux types de connexion sont définis pour le S/PDIF. La plus simple (généralement utilisée dans les composants audio réputés «meilleurs») est un câble coaxial 75 Ω avec fiches RCA (généralement de couleur orange). Le niveau du signal est d'environ 0,5 $V_{càc}$.

La deuxième variante, sous le nom de Toslink (de *Toshiba Link*), est une fibre optique standardisée à LED émettrice rouge (659 nm). Normalement, c'est un simple câble POF (*Plastic Optical Fibre*). Elle peut être utilisée jusqu'à environ 10 m. Avec une fibre optique en verre de qualité, une distance maximale de 30 m est possible sans répéteur. Les signaux optiques ont la même logique que les signaux électriques, puisque le signal électrique ne fait qu'allumer et éteindre la LED.

Projet ambitieux

Le circuit décrit ici (**fig. 1**) est un convertisseur USB vers S/PDIF qui se connecte facilement à un PC et d'autres appareils par connecteur USB ; par sa sortie S/PDIF il se connecte à des récepteurs AV, des amplificateurs haut de gamme ou des DAC audio autonomes. La

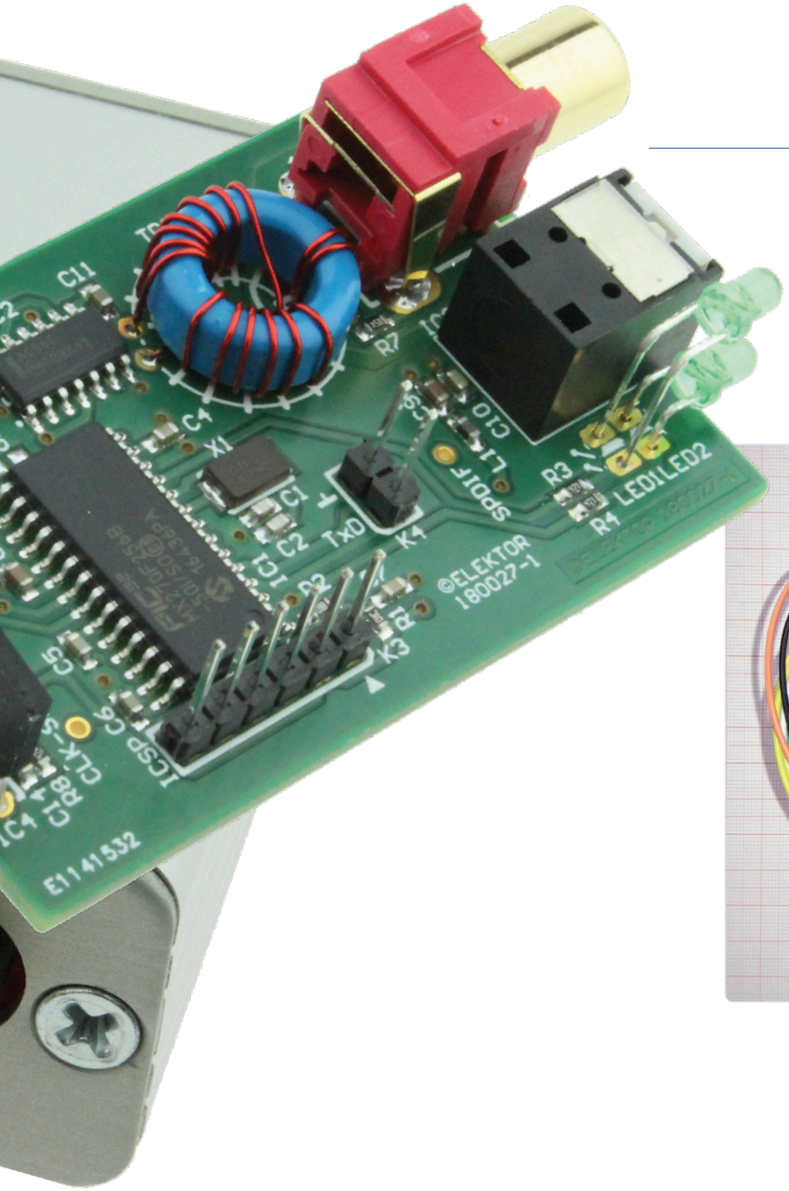

Figure 1. Interface USB-S/PDIF, élégante et compacte.

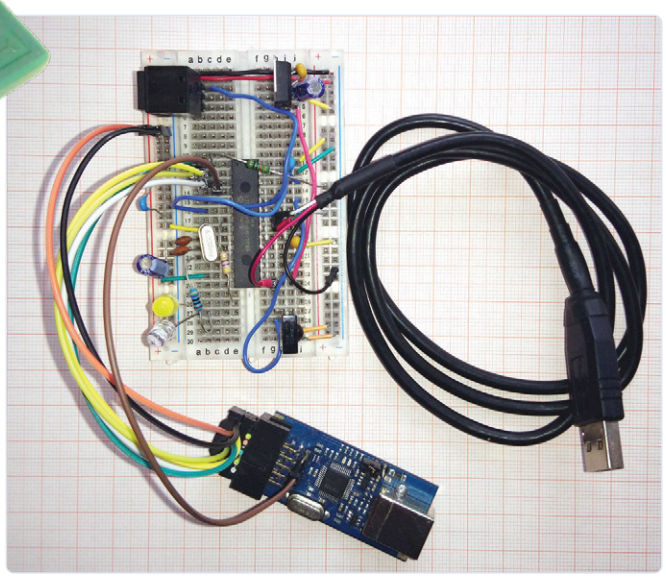

Figure 2. Prototype sur plaque d'expérimentation.

sortie S/PDIF existe sous forme électrique et optique ; la télécommande est possible grâce à un récepteur IR.

Le flux binaire S/PDIF est produit par le logiciel du µC chargé également de la communication USB. Cette solution à puce unique limite le matériel requis. Par rapport aux circuits intégrés audio USB spéciaux, cette solution est flexible et ouverte. Sa réalisation n'est pas trop critique, il peut être assemblé sur une plaque d'expérimentation (**fig. 2**).

Pour ce projet, nous avons choisi un µC PIC32MX270 doté des périphériques pour les applications audio USB, et de suffisamment de RAM pour stocker les trames S/PDIF codées. Il est disponible dans des variantes avec relativement peu de broches, ce qui simplifie le tracé du circuit imprimé.

La petite carte double face conçue pour ce projet contient le µC, l'alimentation et les sorties S/PDIF optiques et électriques, ainsi que le récepteur de télécommande IR, plus deux LED (qui indiquent la fréquence d'échantillonnage et l'activité de sortie). Tout cela tient dans un petit boîtier Hammond.

Section audio

La section audio (**fig. 3**) se compose d'une interface USB conforme à la classe audio USB [2][3], de l'implémentation logicielle du codeur S/PDIF et de la sortie S/PDIF, qui utilise la sortie SPI du µC. Un canal DMA copie en continu les trames S/PDIF du tampon circulaire vers le SPI.

Comme cette interface USB-S/PDIF utilise la spécification audio USB standard, l'installation de pilotes spéciaux sur l'hôte est inutile. La spécification de la *classe de l'appareil* est assez complexe. C'est pourquoi l'USB-IF (*USB Implementers Forum*) a publié une *USB Audio Device Class Specification for Basic Audio Devices* [4], qui contient un petit sous-ensemble de la spécification de *classe de dispositif audio* originale. L'interface audio USB de notre projet ressemble beaucoup à l'application pour casque d'écoute [4]. Nous avons ajouté quelques fréquences d'échantillonnage, omis la commande de volume, et le descripteur USB a été modifié de manière à décrire une sortie S/PDIF au lieu d'un haut-parleur. Notre interface audio USB présente les caractéristiques suivantes :

- trois fréquences d'échantillonnage (44,1 kHz, 48 kHz et 96 kHz)
- format audio à deux canaux avec 3 octets (24 bits) par canal (S24_3LE)
- possibilité de couper le signal audio

Ces caractéristiques sont spécifiées dans le descripteur de configuration USB (fichier `usb_descriptors.c` dans les archives du logiciel — informations complémentaires [3], [5] et [6].)

Comme nous ne visons pas une conformité formelle à l'USB, nous avons retenu des identifiants de fournisseur et de produit utilisables sans risque de conflit avec les produits USB officiels.

Codeur S/PDIF et sortie sérielle

Pour transférer des échantillons audio vers S/PDIF, l'interface USB utilise le point terminal USB isochrone 1. Cela signifie que chaque trame USB de 1 ms transfère un paquet de données USB. Chaque fois que le matériel USB du µC a reçu un nouveau paquet isochrone, la routine d'interruption est appelée et convertit les trames PCM individuelles du

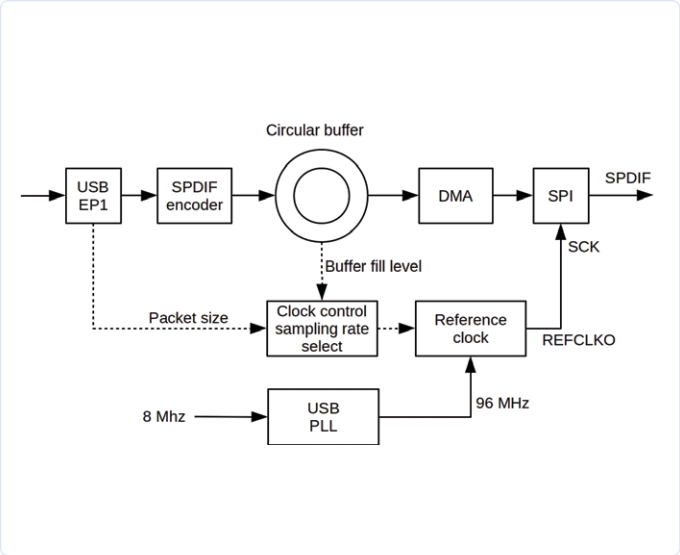

Figure 3. Schéma fonctionnel de l'interface.

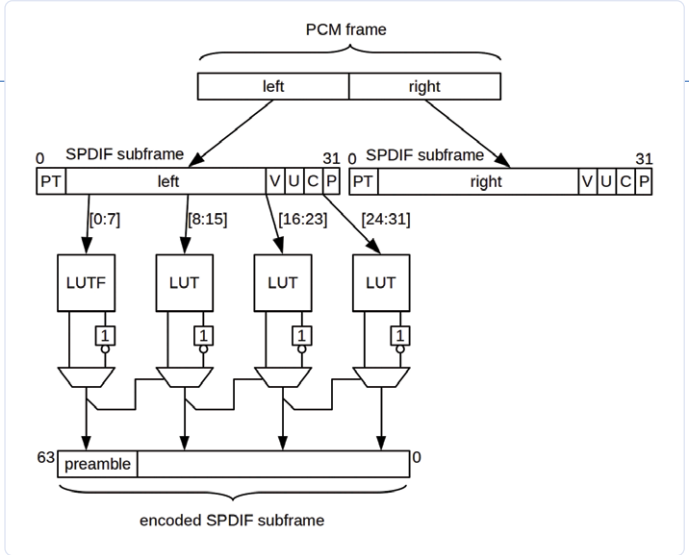

Figure 4. Principe du codeur S/PDIF logiciel.

paquet de données USB en trames S/PDIF correspondantes. Pour plus d'informations sur S/PDIF, consultez [7] et [8].

La **fig. 4** montre schématiquement comment fonctionne le logiciel de conversion S/PDIF. En haut, nous voyons la trame PCM de 48 bits ; chaque trame contient deux échantillons, l'un pour le canal gauche et l'autre pour le canal droit. Ces échantillons sont convertis en une représentation intermédiaire de 32 bits des sous-cadres S/PDIF en ajoutant une balise de préambule de 4 bits, qui spécifie le type de préambule S/PDIF (X, Y ou Z) qui doit être utilisé pour le codage final du sous-cadre S/PDIF. De plus, les quatre bits V, U C et P sont ajoutés à la fin. Les bits V et U correspondent respectivement aux bits de validité et de données utilisateur. Ici ces bits sont toujours à 0. Les bits C contiennent l'état du canal et les bits P sont les bits de parité pour les sous-cadres individuels. Ces bits de parité sont calculés pour chaque sous-trame à l'aide d'un algorithme optimisé [9].

Deux sous-trames S/PDIF successives forment une seule trame S/PDIF. 192 trames S/PDIF successives forment ensemble un bloc S/PDIF. L'état du canal est obtenu en mettant en chaîne tous les bits C d'un bloc l'un après l'autre - le bloc d'état du canal a donc un format de 192 bits (24 octets). Nous utilisons ici les bits d'état de canal pour indiquer au récepteur connecté à la sortie S/PDIF la fréquence d'échantillonnage actuelle.

Le S/PDIF utilise un codage de marque biphasé (également appelé codage *Manchester* différentiel) pour coder les données audio et les bits de préambule. Comme deux bits de code correspondent à un bit de données, la sous-trame S/PDIF codée comprend 64 bits. En utilisant la mise en œuvre la plus simple du codage de marque biphasée, chaque bit d'une sous-trame S/PDIF devrait être codé individuellement, ce qui, bien sûr, consommerait un temps de processeur précieux. C'est pourquoi nous utilisons deux tables de recherche différentes (LUT = *look-up table*) pour coder un octet à la fois. La table de recherche LUTF est utilisée pour coder le premier octet d'une sous-trame S/PDIF, tandis que la table de recherche LUT est utilisée pour coder les trois autres octets de la sous-trame. La table spéciale LUTF contient les modèles de bits du préambule.

La table de recherche LUT apparaît trois fois dans la **fig. 4**, car elle est utilisée trois fois pour coder une sous-trame S/PDIF. En réalité, il n'existe en mémoire qu'une seule instance de cette table. Les tables de consultation sont produites par une fonction d'initialisation exécutée une fois après réinitialisation du µC. Dans le code de marquage biphasé, il doit y avoir une transition (flanc) entre toute paire de bits de code qui représente un bit de données. Pour assurer cette transition, les mots de code de 16 bits des tables de recherche sont inversés ou non en fonction du dernier bit du mot de code précédent.

Les tables de recherche remplissent une deuxième fonction : l'inversion de l'ordre des bits. Dans la norme S/PDIF, les sous-trames sont transmises en commençant par le bit de poids le plus faible (LSB), mais le SPI le fait dans l'ordre inverse (MSB d'abord). C'est pourquoi les LUT fournissent les bits dans l'ordre inverse, afin qu'ils soient correctement transmis par le matériel SPI.

Les mots de code S/PDIF de 64 bits ainsi obtenus sont placés dans une mémoire tampon circulaire (dans la mémoire SRAM du µC) et envoyés par le port SPI en utilisant le DMA.

Horloge de bits S/PDIF

Un diviseur de fréquence fractionnaire programmable intégré au µC dérive l'horloge binaire S/PDIF d'un signal d'horloge interne de 96 MHz. La fréquence d'horloge S/PDIF requise s'élève à

$$f_{SPDIF} = 128 * f_S$$

où f_S est la fréquence d'échantillonnage audio. La spécification audio USB ne permet pas d'indiquer explicitement la fréquence d'échantillonnage f_S utilisée par l'hôte du périphérique USB. Cela signifie que le logiciel du µC la détecte d'après la base du nombre de trames PCM dans les paquets USB isochrones. Notez que le descripteur USB contient trois fréquences d'échantillonnage autorisées. L'hôte n'est pas autorisé à utiliser d'autres fréquences d'échantillonnage que ces 44,1 kHz, 48 kHz ou 96 kHz. L'avantage est que le nombre de trames MIC dans un paquet USB permet au µC de deviner la fréquence d'échantillonnage choisie par l'hôte.

Périls de l'horloge

Conformément à la spécification *USB Basic Audio Devices* [4], c'est le type de synchronisation «synchrone» qui a été sélectionné pour le point terminal audio isochrone. Autrement dit, la vitesse d'échantillonnage du flux audio dépend du domaine de l'horloge USB, c'est-à-dire de la fréquence des jetons SOF exécutés par l'hôte. L'horloge de bits S/PDIF est en revanche dérivée de l'oscillateur local. Pour éviter tout dépassement ou la vacance du tampon circulaire, le logiciel modifie fréquemment la partie fractionnaire du diviseur de fréquence, en fonction du niveau moyen de remplissage du tampon circulaire.

Bien que la note d'application AN1422 [10] de

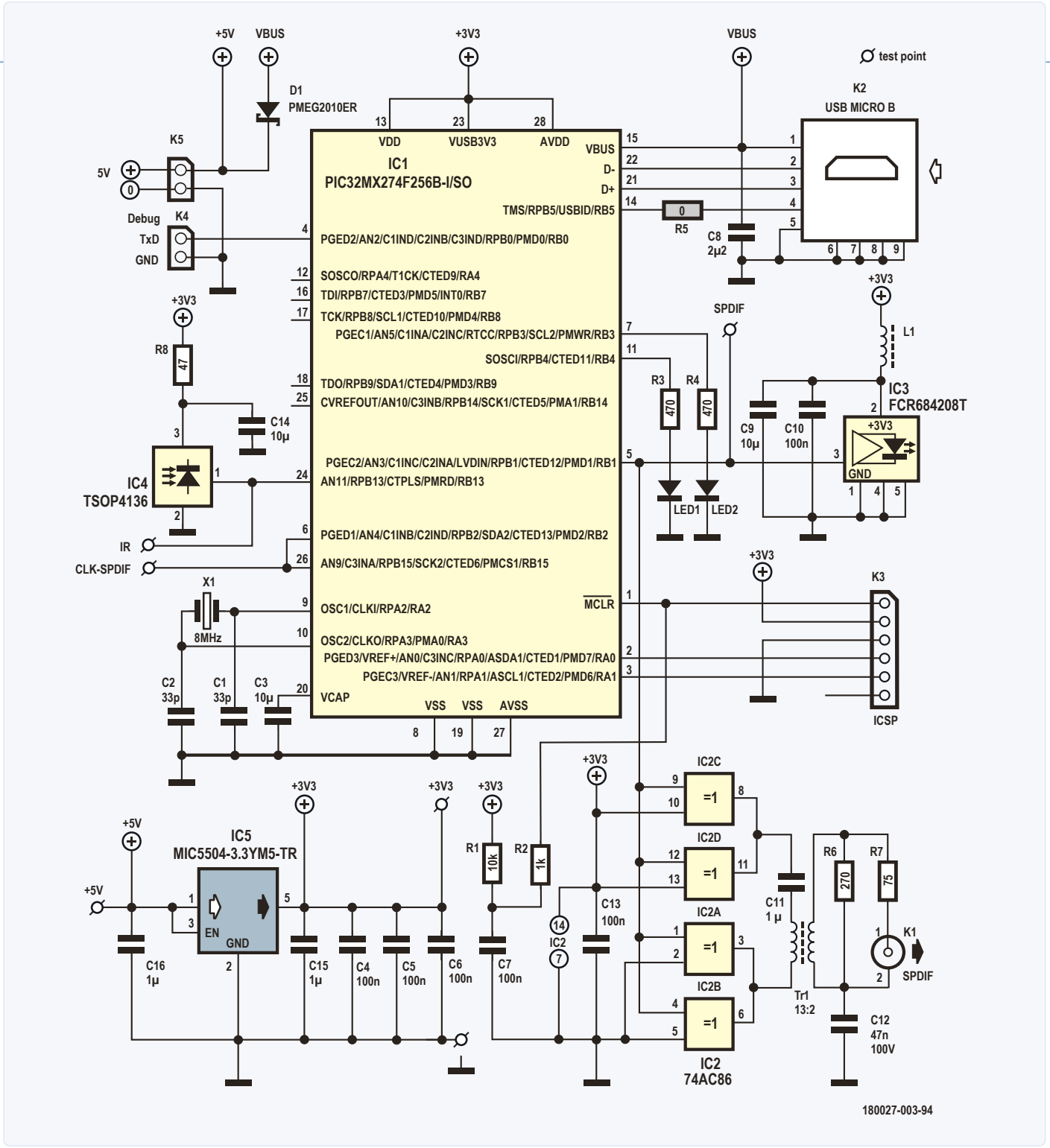

Figure 5. Schéma complet de l'interface USB-S/PDIF.

Microchip indique explicitement que l'horloge de référence peut être ajustée en cours de fonctionnement, nous avons constaté des problèmes lors du changement du rapport de division lorsque ce dernier était faible (c'est-à-dire avec la fréquence d'échantillonnage de 96 kHz). Chaque modification du rapport de division se traduisait par une courte interruption du signal d'horloge et des artefacts audibles.

Pour résoudre ce problème, nous relié la sortie de l'oscillateur d'horloge de référence REFCLKO (broche 6) à l'entrée d'horloge SPI SCK (broche 26), de façon à contourner le générateur de débit en bauds du SPI.

Télécommande IR

Le récepteur de la télécommande IR est indépendant de la section audio du circuit. Tout ce que le récepteur reçoit est transmis tel quel à l'hôte sans influencer la section audio. Pour décoder le signal IR de la télécommande, on utilise la bibliothèque libre IRMP [11] qui permet de décoder de nombreux signaux IR différents. Le logiciel du µC comprend une implémentation USB HID [12] qui permet d'envoyer des codes comme le spécifie le document *HID Usage Tables* [13].

La connexion entre l'IRMP et l'implémentation HID est obtenue à l'aide d'une table de

LISTE DES COMPOSANTS

Résistances (CMS 0603, 1%, 0,1 W)
R1 = 10 kΩ
R2 = 1 kΩ
R3,R4 = 470 Ω
R5 = 0 Ω **omise**
R6 = 270 Ω
R7 = 75 Ω
R8 = 47 Ω

Condensateurs (CMS 0603)
C1,C2 = 33 pF, 50 V, 5%, C0G/NP0
C3,C9,C14 = 10 µF, 16 V, 20%, X5R
C4...C7,C10,C13 = 100 nF, 50 V, 10%, X7R
C8 = 2,2 µF, 10V, 10%, X7R
C11,C15,C16 = 1 µF, 50 V, 10%, X5R
C12 = 47 nF, 100 V, 10%, X7R

Inductances
L1 = 600 Ω @ 100 MHz, 0,15 Ω, 1,3 A, CMS 0603 (Murata, BLM18KG601SN1D)
TR1 = noyau torique, 12,5 x 5 mm, matériau T38, Epcos B64290L0044X038

Semi-conducteurs
LED1,LED2 = LED verte, 3 mm, traversante
D1 = PMEG2010ER, CMS SOD-123
IC1 = PIC32MX274F256B-I/SO, CMS SO-28
IC2 = 74AC86, CMS SO-14
IC3 = FCR684208T, Toslink (Cliff Electronic Components)
IC4 = TSOP4136
IC5 = MIC5504-3.3YM5-TR, CMS SOT-23-5

Divers
K1 = prise RCA encartable, traversante (Pro Signal, PSG01545)
K2 = micro-USB type B, femelle, CMS (Molex, 47346-0001)

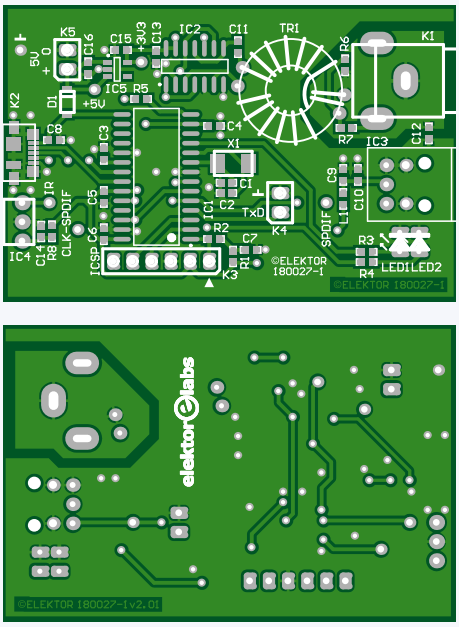

Figure 6. Dessin du circuit imprimé compact conçu pour l'interface.

K3 = barrette1x6, verticale, pas 2,54 mm, traversante
K4,K5 = barrette 1x2, verticale, pas 2,54 mm, traversante
X1 = quartz 8 MHz, 5 x 3,2 mm, CMS (Abracon, ABM3-8.000MHZ-D2Y-T)
TR1 = 0,3 m de fil de cuivre verni (CuL), Ø 0,5 mm
Boîtier Hammond 1455D601, 60 x 42,5 x 23 mm

circuit imprimé 180027-1 v2.01 (e-choppe Elektor)

correspondance de clés qui couple les codes détectés par l'IRMP aux ID d'utilisation [13]. Nous avons décidé d'ajouter les codes de la norme RC5, certes de moins en moins utilisée mais disponible sur de nombreuses télécommandes universelles. Elle fonctionnera bien dans de nombreuses situations, sans interférer avec d'autres appareils tels que les téléviseurs. Outre les codes RC5 énumérés [14], le tableau des codes contient également des codes pour télécommande Denon.
La carte des clés se trouve dans le fichier source `irhid.c` sous le nom `irhid_keymap`, et peut facilement être modifiée selon vos besoins en ajoutant ou en supprimant des lignes et en recompilant ensuite le logiciel. Pour ajouter une touche, vous devez trouver les codes IRMP pour le protocole concerné, l'adresse et la commande. Cela peut se faire en connectant un émulateur de terminal (115200 bits/s) à la sortie S/PDIF et en appuyant sur le bouton correspondant. Cela devrait suffire pour voir apparaître les codes correspondants dans l'émulateur de terminal. Si cela ne fonctionne pas, vous devrez peut-être activer le code de la télécommande appropriée en éditant le fichier `irmpconfig.h`. Vous avez également besoin de l'identifiant d'utilisation USB HID correspondant qui doit être couplé à la touche. Vous trouverez cet ID dans [13].

Assemblage

Après toutes les explications déjà données, le tour du schéma (**fig. 5**) sera vite fait. IC1, le microcontrôleur, peut être programmé en circuit via le connecteur ICSP K3 ; si vous préférez ne pas le faire vous-même, vous pouvez acheter le µC préprogrammé dans l'e-choppe d'Elektor. La source d'alimentation externe +5 V est connectée à K5.
Le signal USB entre par K2 et va directement au µC. La résistance de 0 Ω R5 ne doit pas être installée, car la version actuelle du logiciel n'utilise pas l'USBID.
IC4 est un récepteur IR standard ; IC3 s'occupe de la sortie optique (Toslink) ; le signal de sortie électrique arrive sur K1 via le quadruple tampon IC2. Nous arrivons ainsi au composant le moins banal de ce circuit : le transformateur TR1. Sa fonction n'est pas tant d'assurer une séparation galvanique que d'éviter les boucles de terre. Pour réduire le bruit RF, la sortie doit être découplée, ce que fait C12. Les opérateurs logiques EXOR d'IC2 qui forment un pont, présentent l'avantage d'augmenter la tension primaire et offrent un meilleur couplage à l'enroulement secondaire. Le rapport de transformation du transformateur est de 13:2 et cela donne une tension qui est pratiquement exactement de 0,5 $V_{càc}$ pour une charge de 75 Ω.
Un circuit imprimé (double face), **fig. 6**, a été dessiné pour ce circuit. La plupart des composants sont montés en surface, mais l'assemblage ne devrait pas poser de problème pour un électronicien expérimenté.
Le circuit imprimé tient dans un boîtier Hammond en alu de type 1455D601 (60 x 42,5 x 23 mm) où il peut simplement être glissé comme le montre la **fig. 7**. Selon les éventuelles tolérances de fabrication, il faudra peut-être passer une lime sur les bords de la carte.
Si vous utilisez le boîtier Hammond de la liste des composants, il est préférable de supprimer l'une des deux collerettes en plastique noir : à cause d'elle, le connecteur micro-USB n'affleure pas à la surface du boîtier, et le contact de la fiche mâle correspondante est mauvais. Les panneaux latéraux nécessitent bien sûr le perçage et la découpe d'ouvertures appropriées pour les connecteurs et les LED. Les broches des LED peuvent être pliées de telle manière qu'elles «regardent» vers l'extérieur, l'une au-dessus de l'autre (**fig. 7**).

Une remarque sur le transformateur : vous devrez le bobiner vous-même sur le noyau torique spécifié dans la liste des composants. La **fig. 8** montre que l'enroulement primaire (13 tours) est effectivement enroulé en deux moitiés sur le tore. Cela présente l'avantage que les connexions pour les enroulements primaire et secondaire sont opposées l'une à l'autre. Le secondaire n'a que deux tours, ce n'est pas vraiment pas sorcier, il suffit d'une trentaine de centimètres de fil de cuivre verni de 0,5 mm.

Quelques remarques pratiques

Nous avons bien sûr testé à fond l'interface USB-S/PDIF dans notre labo et essayée avec différents systèmes d'exploitation ; cela a bien fonctionné avec Windows 7 et Windows 10, Linux (Lubuntu et Kubuntu), Android et même Raspbian sur une Raspberry Pi 3 B+ en utilisant *2019-09-26-raspbian-buster-full.img*.

Windows

Windows a automatiquement reconnu l'interface ; la fréquence d'échantillonnage peut être réglée dans le panneau de configuration sous l'option Son. Sélectionnez «Propriétés de l'interface SPDIF» ; sur la page d'onglet Avancé, vous pouvez sélectionner le format, comme indiqué sur la **figure 9**.

Lubuntu

Sous Linux (nous avons utilisé Lubuntu), les fréquences d'échantillonnage dépendent du fichier. En utilisant le simple GNOME MPlayer, un fichier de 44,1 kHz apparaît comme 44,1 kHz sur la sortie S/PDIF. Mais à chaque fois, nous avons dû sélectionner «USB_SPDIF Stereo (IEC958) (PulseAudio)» comme sortie audio – même si nous l'avions déjà fait auparavant. C'est ennuyeux. Vous pouvez éviter cela en éteignant tous les autres appareils audio (ici : Sound & Video - PulseAudio Volume Control - Configuration). Un fichier de 48 kHz et un fichier de 96 kHz sont tous deux lus en 48 kHz. Une autre possibilité est d'utiliser ALSA directement, sans PulseAudio, dans un terminal. Si vous n'entendez rien, démarrez d'abord *alsamixer* dans un terminal (Ctrl+Alt+T et entrez `alsamixer`) ; appuyez sur F6 pour sélectionner la carte son 'USB_SPDIF' (il n'y a qu'un petit élément de contrôle qui affiche 00 et PCM, car l'interface n'a pas de réglage de volume). Ensuite, fermez le mélangeur et entrez (`//.../` est le chemin d'accès réel au fichier audio) :

```
aplay -D plughw:CARD=USBSPDIF
   //.../
```

Vous ne pouvez l'utiliser que pour lire des fichiers wav non compressés. Pour lire des fichiers mp3 à partir de la ligne de commande, vous pouvez utiliser p. ex. mpg123, mais il existe de nombreuses autres possibilités.
Il existe des CN/A audio USB depuis de nombreuses années, cependant la méthode standard de traitement des fichiers audio sous Linux reste perfectible. Le réglage de la fréquence d'échantillonnage de sortie est également possible en éditant les fichiers de configuration pour PulseAudio (`/etc/pulse/daemon.conf`). Des informations à ce sujet peuvent être trouvées en ligne. Faites d'abord une sauvegarde du fichier original, il est plus facile de faire une erreur que de ne pas en faire...

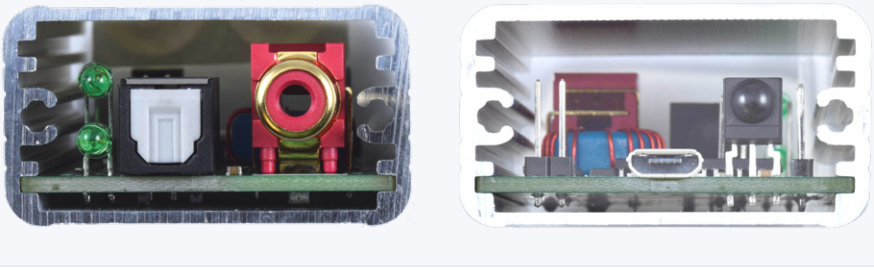

Figure 7. Circuit imprimé assemblé glissé dans le boîtier en alu.

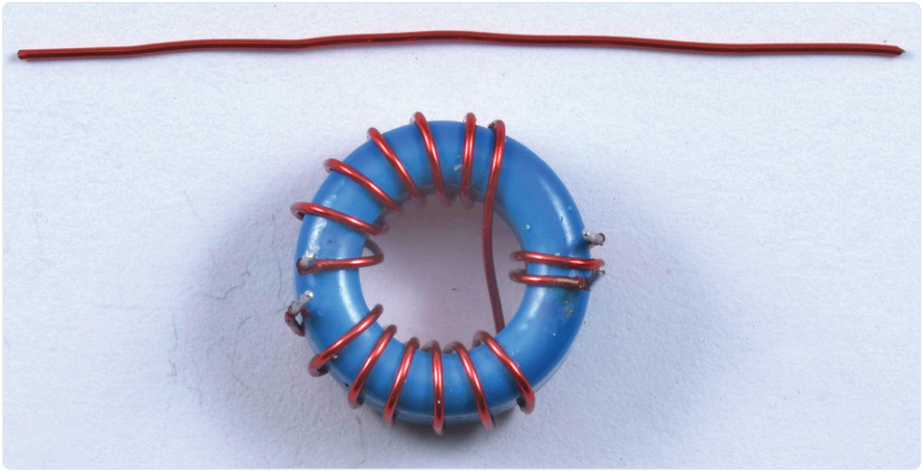

Figure 8. Bobinage du transfo torique. L'enroulement primaire est divisé en deux.

Android

Connectez l'interface à l'aide d'un câble OTG. L'installation de l'application «USB Audio Player Pro» est probablement la façon la plus simple d'utiliser notre interface. Cela permet de contourner les limitations audio d'Android. Les trois fréquences d'échantillonnage de l'interface sont prises en charge. Une fois que l'application a démarré, l'interface est immédiatement reconnue.

Raspbian

Notre dernier test portera sur l'interface avec Raspbian. Tout comme Lubuntu, Raspbian est un système d'exploitation libre également basé sur Debian ; il n'est donc pas surprenant que le test ait suivi le même schéma. La lecture des fichiers wav non compressés se fait ainsi :

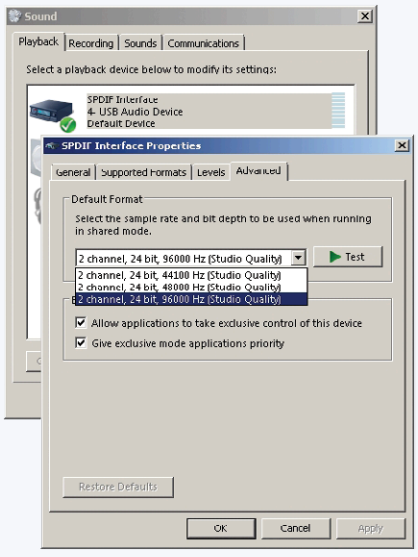

Figure 9. Paramètres de configuration pour Windows 7.

PRODUITS

- **Interface USB-S/PDIF, carte nue :** www.elektor.fr/180027-1
- **Interface USB-S/PDIF, microcontrôleur programmé :** www.elektor.fr/180027-41

```
aplay -D plughw:CARD=USBSPDIF //.../
```

Appuyez sur Ctrl+c pour arrêter la lecture.
Lors de la lecture de fichiers mp3 en ligne de commande, mpg123 ne fonctionnait (s'il n'est pas déjà installé : `sudo apt-get install mpg123`) avec notre nouvelle installation de Rasbian qu'après que PulseAudio et son contrôle de volume aient également été installés. Installez PulseAudio comme ceci :

```
sudo apt-get install pulseaudio
```

Installation du réglage de volume associé :

```
sudo apt-get install pavucontrol
   paprefs
```

Redémarrez le système après avoir tout installé.
Tout comme avec Lubuntu, vous devez désactiver l'audio embarqué dans la section *Sound & Video ➞ PulseAudio Volume Control ➞ Configuration*.
Le lecteur multimédia VLC préinstallé ne produisait aucun son avant l'installation de PulseAudio. Un fichier échantillonné à 32 kHz sera lu à 96 kHz.
Nous avons également mesuré les performances audio. Sur le site Elektor Labs [15], vous trouverez une description de ces mesures et de leurs résultats sur la page de ce projet.
Le logiciel mis à jour peut être téléchargé sur la page en ligne de cet article [16] ou sur GitHub [17].
Nous vous souhaitons beaucoup de plaisir lors de la réalisation de cette interface et bien sûr lors de son utilisation ! Vos observations et vos critiques sont les bienvenues.

180027-04

LIENS

[1] https://kompendium.infotip.de/spdif_toslink.html
[2] **spécification universelle des bus de série, révision 2.0** : www.usb.org/document-library/usb-20-specification
[3] **définition de la classe de dispositif de bus série universel pour les appareils audio, version 1.0** : www.usb.org/document-library/audio-device-document-10
[4] **spécification de la classe de dispositif audio du bus série universel pour les dispositifs audio de base, version 1.0** : www.usb.org/document-library/audio-device-class-spec-basic-audio-devices-v10-and-adopters-agreement
[5] **définition de la classe de dispositif de bus série universel pour les formats de données audio, version 1.0** : h www.usb.org/document-library/audio-data-formats-10
[6] **définition de la classe de dispositif de bus série universel pour les types de terminaux, version 1.0** : h www.usb.org/document-library/audio-terminal-types-10
[7] **AES3-2003** : Norme AES pour l'audio numérique - Interface entrée-sortie numérique - Format de transmission en série pour les données audio numériques à deux canaux représentés linéairement
[8] **Crystal Application None AN22: "Overview Of Digital Audio Interface Data Structures"** : https://statics.cirrus.com/pubs/appNote/an22.pdf
[9] **Sean Eron Anderson: "Bit Twiddling Hacks", section "Compute parity in parallel"** : http://graphics.stanford.edu/~seander/bithacks.html#ParityParallel
[10] **Microchip AN1422: "High-Quality Audio Applications Using the PIC32"** : http://ww1.microchip.com/downloads/en/AppNotes/01422A.pdf
[11] **Infrared Multi-Protocol decoder (IRMP)** : www.mikrocontroller.net/articles/IRMP
[12] **USB Device Class Definition for Human Interface Devices (HID), Version 1.11** : h www.usb.org/document-library/device-class-definition-hid-111
[13] **USB HID Usage Tables, Version 1.12** : www.usb.org/document-library/hid-usage-tables-112
[14] **protocoles de télécommande par IR :** www.elektormagazine.fr/magazine/elektor-200103/9101
[15] **page du projet sur Elektor Labs** : www.elektormagazine.com/labs/usb-spdif-interface-180027
[16] **page en ligne de cet article** : www.elektormagazine.fr/180027-04
[17] **logiciel sur GitHub** : https://github.com/kiffie/usb-spdif