sortie vidéo sur les microcontrôleurs (1)

vidéo composite

Mathias Claussen (Elektor Lab)

Le thème de la sortie vidéo sur les microcontrôleurs remonte aux débuts de ces petites puces polyvalentes. Les microcontrôleurs d'aujourd'hui ont une puissance de calcul bien supérieure à celle de l'ordinateur familial Sinclair ZX81 d'il y a presque 42 ans, mais même les microcontrôleurs les plus récents sont loin de rivaliser en taille mémoire avec les cartes graphiques modernes où elle se mesure en gigaoctets. Cela n'empêche pas les développeurs de produire d'étonnantes images animées avec un ATmega, un ESP32 ou un RP2040. Dans la première partie de cette série, nous aborderons la sortie de vidéo composite. Ensuite, nous poursuivrons avec le VGA et même le DVI. Dans tous les cas, quelques astuces et un timing précis sont nécessaires. Il ne s'agit donc pas seulement de théorie, mais aussi d'exemples pratiques comme point de départ pour vos propres expériences.

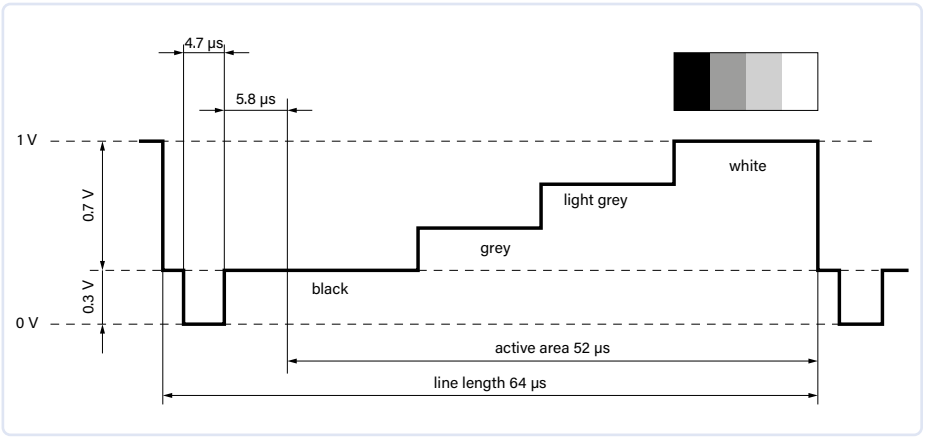

Figure 1. Signal VBS avec timing PAL. (Source : Wikipedia)

L'histoire des formats vidéo remonte aux débuts de la télévision. Comme pour la radio, on a élaboré des standards et des normes pour la télévision. À l'époque de la télévision couleur analogique, les normes les plus courantes étaient le National Television Systems Committee (NTSC - Amérique du Nord et du Sud, Japon), l'Alternance de Phase par Ligne (PAL - Europe, Amérique du Sud, Afrique et Asie) et le Séquentiel Couleur à Mémoire (SECAM - France, Afrique et URSS).

La procédure de base pour la transmission analogique des signaux vidéo a suivi la méthode VBS (Video Blanking and Sync) pour ces normes. La résolution et la fréquence d'images pour la VBS dépendent de la norme de télévision sous-jacente. Pour la norme NTSC, elle est de 480 lignes visibles avec 640 pixels visibles à 59,94 trames par seconde. PAL et SECAM ont 576 lignes visibles avec 720 pixels visibles et 50 trames par seconde (576i). En outre, NTSC, PAL et SECAM diffèrent également au niveau de la modulation et de la manière dont les informations de couleur sont ajoutées. À l'ère de la transmission numérique des images, ces trois méthodes ont largement perdu de leur importance. Cependant, elles subsistent sous la forme de formats d'images numériques pour les DVD ou la définition standard pour la télévision (SDTV).

Vidéo composite avec CVBS

Les premiers signaux de télévision normalisés étaient conçus pour la transmission d'images monochromes. Différents formats d'image ont été développés aux Etats-Unis et en Europe. La **figure 1** illustre le timing d'un signal VBS au format PAL, à titre d'exemple de ces normes.

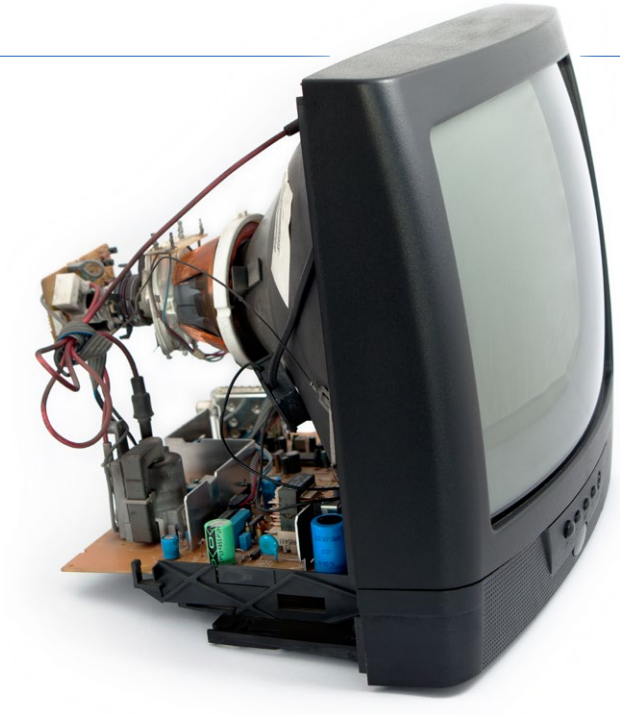

Figure 2. Téléviseur ouvert avec tube cathodique. (Source : Shutterstock/ Sergio Sergo)

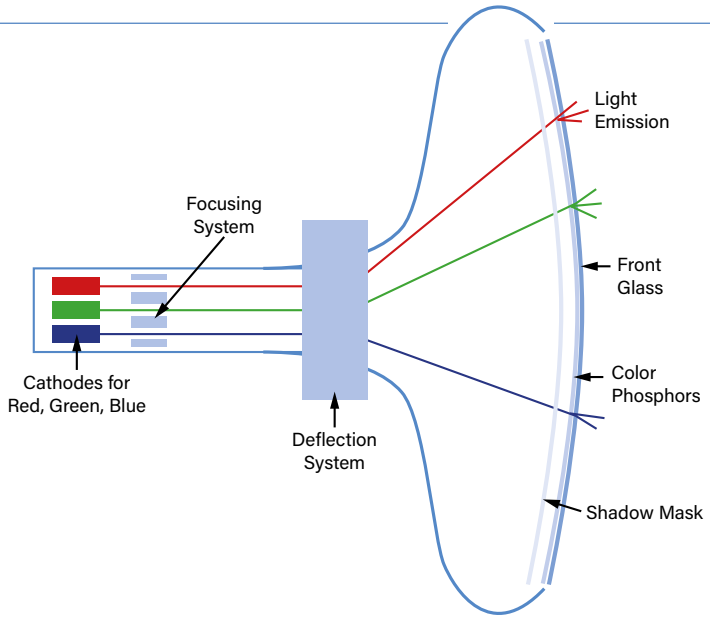

Figure 3. Structure schématique d'un tube image couleur. (Source : ITWissen.info)

Cette norme était particulièrement adaptée à la reproduction d'images avec des tubes cathodiques (**figure 2**). Le principe de fonctionnement de ces tubes diffère considérablement de celui des écrans LCD ou OLED. Dans ces tubes, un faisceau d'électrons modulé, dévié horizontalement et verticalement, est dirigé sur une couche phosphorescente. Au point d'impact, cette couche s'illumine proportionnellement à l'intensité du faisceau d'électrons (**figure 3**). Le balayage horizontal et vertical produit une image bidimensionnelle.

Du point de vue de l'observateur, une image (en noir et blanc) se compose de lignes balayées de gauche à droite et de haut en bas (**figure 4**). Le premier pixel d'une image est donc situé en haut à gauche. Cette convention est largement conservée dans les écrans numériques modernes.

Quelques connaissances historiques sur la génération d'images au moyen d'un faisceau d'électrons en mouvement facilitent la compréhension des signaux vidéo modernes. Le signal de la **figure 1** contient la structure d'une ligne d'image. Au début (**figure 5**) du signal, on distingue l'extinction (*blanking*) horizontale. Au début de cette extinction, le faisceau d'électrons se trouve sur le bord droit du tube image et doit donc retourner à l'extrême gauche pour la ligne suivante. Pendant ce saut de droite à gauche, le faisceau d'électrons est éteint en réglant le signal de luminosité sur le noir ou même sur « l'infra-noir ».

L'extinction horizontale prend 12 µs pour le PAL et 10,9 µs pour le NTSC. Elle se compose de trois sections appelées « palier avant » (*front porch*), « top de synchro » (*sync tip*) et « palier arrière » (*back porch*). Le niveau de luminosité du palier avant (PAL = 1,65 µs/NTSC = 1,4 µs) est égal ou légèrement inférieur à la valeur du noir à 0,3 V, tandis que le faisceau d'électrons se déplace encore plus vers la droite, hors de l'image visible. Le niveau du top de synchro (PAL et NTSC = 4,7 µs) est de 0 V,

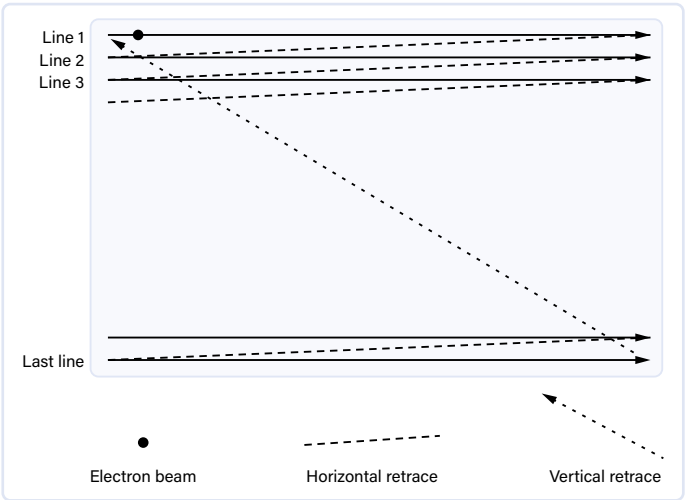

Figure 4. Composition d'image basée sur les lignes.

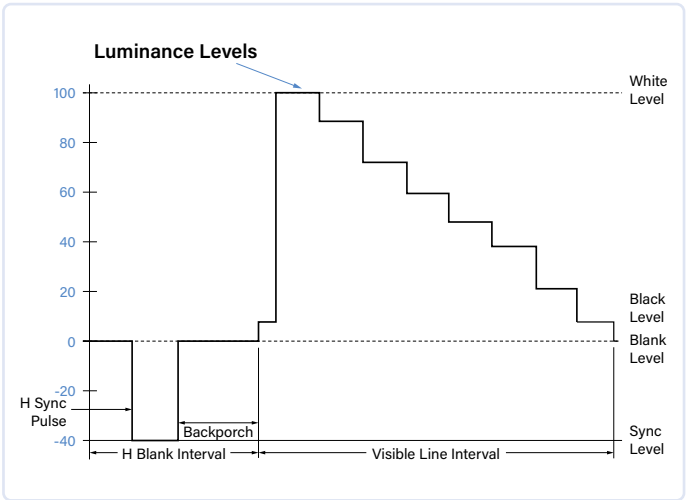

Figure 5. Extinction horizontale. (Source : www.edn.com)

nettement inférieur à la valeur du noir. Le top de synchro ramène le faisceau d'électrons de droite à gauche. Le palier arrière (PAL et NTSC = 1,6 µs) fournit le niveau de référence du noir (0,3 V) pour la ligne. Le contenu de l'image à afficher commence à la fin du palier arrière. On dispose alors de 52 µs pour la partie visible de la ligne. Le nombre de pixels affichés pendant ces 52 µs dépend du format vidéo. En PAL, il y a généralement 720 pixels visibles. Lorsque toutes les lignes sont tracées et que le faisceau d'électrons a atteint le bas à droite, il doit être renvoyé au point de départ de l'image (en haut à gauche). Ce processus sera décrit dans une section séparée.

Synchronisation et entrelacement

La bande passante analogique pour la transmission des images de télévision était suffisante pour seulement 25 images par seconde pour le PAL et 29,97 images par seconde pour le NTSC. Pour l'œil humain, cela se traduit par une image en mouvement, mais elle n'est pas très fluide et crée un effet de scintillement très désagréable. Avec une fréquence d'images plus rapide de 50 ou même 59,94 images par seconde, le scintillement est sensiblement réduit. La plupart des gens perçoivent alors cette image comme une impression visuelle lisse. La bande passante radio disponible étant limitée, l'amélioration est venue de la technique du balayage entrelacé. Au lieu de transmettre toutes les lignes d'une image à une fréquence plus élevée sous forme de trames complètes, les lignes dont le numéro est pair ou impair sont transmises alternativement à deux fois la fréquence, c'est-à-dire 50 ou 59,94 Hz, respectivement (**figure 6**).

Ainsi, on transmet une première trame avec les lignes de numéros impairs, puis une deuxième trame avec les lignes de numéros pairs. Pour que les lignes d'une trame puissent s'insérer dans les espaces de l'autre trame, un signal de détection est émis au début de chaque trame.

Les lignes d'une image complète étant numérotées en fonction de la séquence de transmission, la première trame en PAL comporte les lignes 1 à 313 et la numérotation de la deuxième trame continue jusqu'à la ligne 625. La synchronisation verticale pour la fin d'une trame et le début de la suivante est un peu délicate : les lignes 311 à 317 à la transition entre les trames impaire et paire ainsi que les lignes 623 à 625 à la transition entre les trames paire et impaire de l'image complète suivante ne sont pas visibles. Ces lignes comprennent des impulsions de synchronisation plus courtes avec des niveaux compris entre 0,3 V et 0 V. La **figure 7** montre la séquence pour une trame PAL complète. Grâce à ces impulsions, l'électronique peut reconnaître la parité des numéros de ligne de la trame.

Vidéo monochrome avec les contrôleurs AVR

Vous trouverez des informations de base sur les signaux composites tels que CVBS ou FBAS sur Wikipedia [1]. Pour plus de détails et des astuces de circuits, consultez le livre Analogue Video [2] d'Angelo La Spina.

Si le signal vidéo est transmis en tant que signal VBS (uniquement la luminance, sans

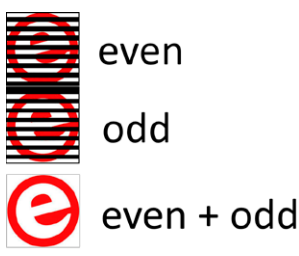

Figure 6. Composition de l'image à partir de deux trames.

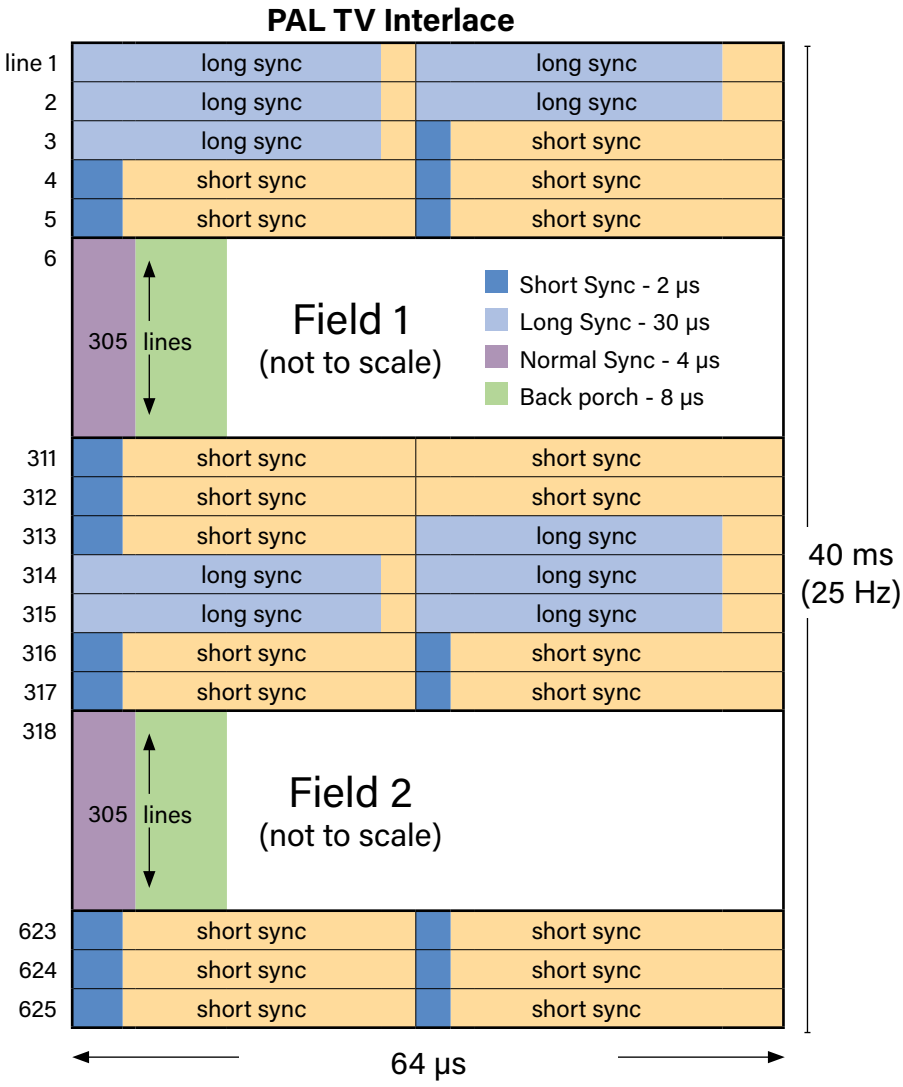

Figure 7. Composition d'une image PAL complète. (Source : martin.hinner.info)

la chrominance) à l'aide d'un câble vidéo composite approprié (**figure 8**), la génération des informations de chrominance peut être assez simple. Si vous n'utilisez que le blanc et le noir au lieu de l'échelle de gris, il vous suffit d'émettre deux tensions appropriées plus la synchronisation. Outre la simplification du circuit, cela permet également de réduire la mémoire nécessaire à la génération de l'image.

Le microcontrôleur Atmega328P d'un Arduino Uno est donc capable de générer une image monochrome de 128x96 pixels. Tous les pixels peuvent être stockés dans la mémoire interne de l'Arduino Uno, car seuls 1536 octets sont nécessaires. Le câblage d'un Arduino Uno avec sortie composite est illustré à la **figure 9.**

Comme vous pouvez le constater, deux résistances sont suffisantes pour produire un signal adéquat. Pour produire une image, vous pouvez utiliser la bibliothèque *TVOut* [3] qui prend en charge la synchronisation pour NTSC et PAL. La sortie vidéo qu'elle permet de réaliser va du simple texte à vos propres jeux. Hackvision [4] est une plateforme de console de jeu. Il s'agit d'un matériel ouvert pour lequel, dans les cas extrêmes, une carte d'expérimentation et quelques composants suffisent. Un Arduino peut également être converti en plateforme Hackvision (**figure 10**) et ainsi utiliser la bibliothèque de jeux associée.

La sortie vidéo avec un microcontrôleur AVR nécessite un de ses timers. La synchronisation horizontale et verticale est assurée par le Timer1 et sa broche de sortie PB1 (OC1A). Sous le contrôle du timer, les pixels apparaissent ligne par ligne sur une broche dédiée (PD7).

Les valeurs des résistances à PB1 et PD7 peuvent être déterminées assez facilement. Une entrée composite sur les moniteurs ou les téléviseurs a une impédance de 75 Ω. Le niveau pour la synchronisation est compris entre 0 V et 0,3 V. La tension pour les valeurs de luminosité des pixels est comprise entre 0,3 V pour le noir et 1 V pour le blanc, respectivement. Dans ce qui suit, il est supposé que le microcontrôleur est alimenté en 5 V.

La **figure 11** montre le diviseur de tension pour la sortie vidéo à l'impédance typique de 75 Ω. La broche 9 fournit les signaux de synchronisation. Mathématiquement, une

Figure 8. Câble pour vidéo composite avec fiche RCA. (Source : Shutterstock/Woodpond)

valeur de 1175 Ω serait nécessaire pour R1, de sorte que la chute de tension sur R2 soit de 0,3 V avec un niveau haut sur la broche 9. 1 kΩ fournit un maximum de 0,34 V sur R2, valeur suffisamment précise. Les valeurs de luminosité apparaissent sur la broche 7. Sur R2, il faut maintenant des valeurs comprises entre 0,34 V (noir) et 1 V (blanc). Pour obtenir 1 V sur R2, une valeur de 375 Ω est nécessaire pour R1 et R3 en parallèle, ce qui donnerait 600 Ω pour R3. La valeur immédiatement supérieure de 470 Ω de la série E-12 garantit que la tension à R2 ne pourra jamais dépasser la valeur de 1 V.

Le fait que cela fonctionne avec un Arduino Uno ou un ATmega328P prouve qu'une faible puissance de calcul et peu de mémoire suffisent pour afficher des graphiques sur un écran. Comme l'image complète peut être conservée dans la mémoire interne, le dessin d'une nouvelle image n'est pas critique en termes de temps, seule la génération des signaux vidéo à l'aide de Timer1 est critique.

Est-il également possible de produire des niveaux de gris, c'est-à-dire plusieurs valeurs de luminosité avec l'Arduino ? Oui, mais la mémoire interne est le facteur limitant. Avec 16 nuances de gris (niveaux de gris 4 bits) et des résistances adéquates, il faut 6144 octets pour la mémoire vidéo si l'on veut conserver la résolution de 128x96 pixels et qu'une image complète puisse tenir dans la mémoire.

Raspberry Pi Pico et Composite

Un Raspberry Pi Pico peut également émettre un signal composite, et il supporte aisément plus de 50 nuances de gris. Pour cela, une échelle de résistances R-2R [5] est utilisée comme convertisseur numérique/analogique. La **figure 12** montre un circuit adéquat composé de résistances de 180, 320 et 360 Ω. Comme on peut le voir sur la page GitHub

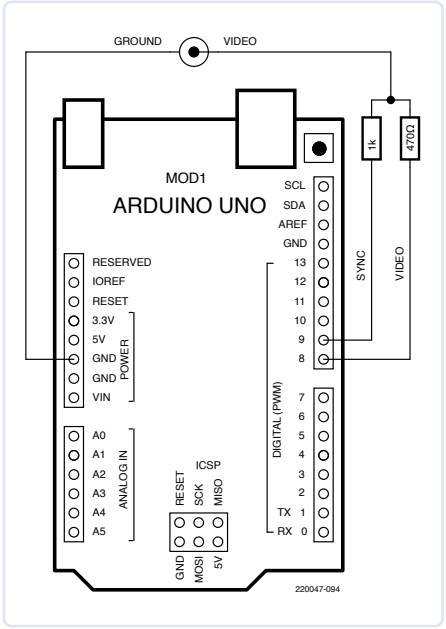

Figure 9. Sortie vidéo N/B avec deux résistances sur un Arduino Uno.

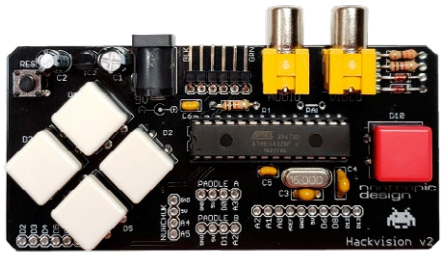

Figure 10. Matériel Hackvision. (Source : nootropic design)

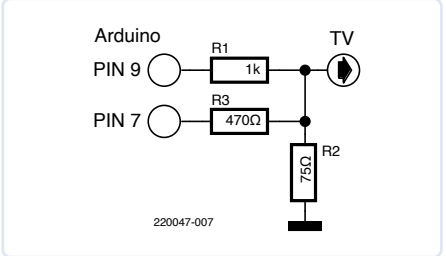

Figure 11. Le circuit de sortie vidéo de la figure 9.

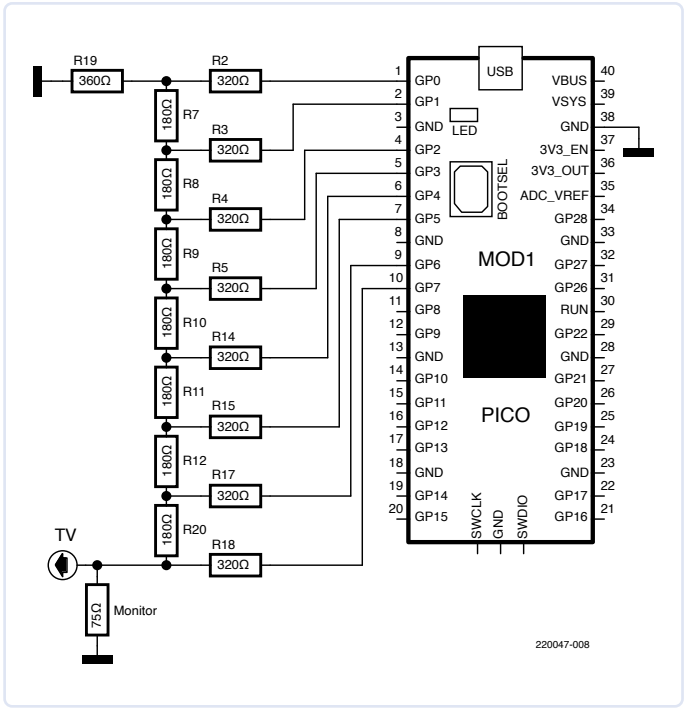

Figure 12. Un réseau R-2R comme CNA sur le Raspberry Pi Pico.

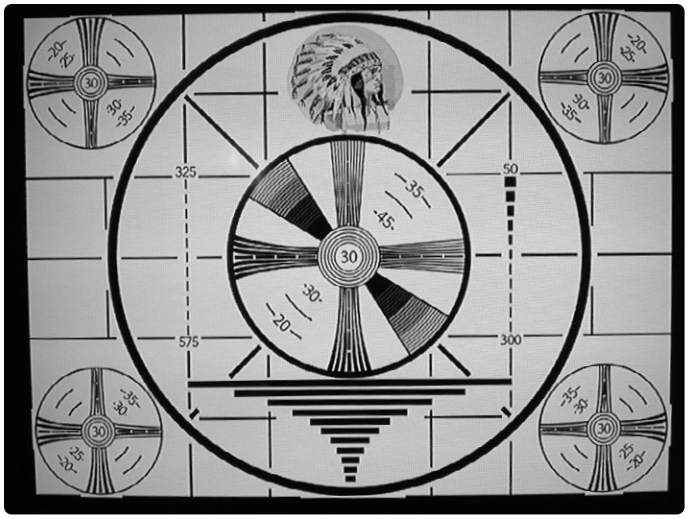

Figure 13. Image en niveaux de gris provenant d'un Raspberry Pi avec 512x384 pixels. (Source : tinyurl.com/2p8z27a2)

du projet « pico-composite8 » [6], la résistance interne des broches GPIO ne peut pas être négligée ici. Le développeur du projet l'a évaluée à 40 Ω environ. Le Pico émet un signal composite conforme à la norme NTSC. Il charge les images soit à partir de la RAM soit à partir de la Flash et peut fournir un maximum de 512x384 pixels, comme le montre la **figure 13**.

Toutefois, le projet n'est qu'une démonstration de faisabilité et n'inclut pas de bibliothèque générique prête à l'emploi. Le fait qu'un Raspberry Pi Pico puisse produire une vidéo en niveaux de gris de 512x384 pixels montre que ce n'est pas tant une question de puissance de calcul que de bon timing. Si une image complète est stockée dans la RAM du Raspberry Pi Pico, il ne reste disponibles qu'environ 64 ko des 264 ko pour vos propres applications. Mais si vous considérez que même un ATmega peut gérer des jeux comme Tetris ou Pong, y compris la sortie vidéo, cet espace devrait être plus que suffisant pour vos propres créations.

Couleur pour la vidéo composite

Plus de 50 nuances de gris, c'est bien, mais la couleur, c'est bien mieux. Lorsqu'il s'agit de vidéo composite et de couleur, il devient beaucoup plus difficile qu'auparavant de générer un signal analogique adéquat. En 2003, lors d'un événement Hackaday [7], Rickard Gunée a démontré la génération d'un signal composite (PAL ou NTSC) en utilisant un Scenix/Ubicom SX28 à environ 50 MHz.

Mais ajouter de la couleur à un signal vidéo, à quel point est-ce difficile ? La réponse dépend de la manière dont les informations de couleur sont ajoutées au signal composite.

Plein de couleurs grâce à PAL et NTSC

Lors du passage à la télévision en couleur, il y a quelques décennies, aucun signal entièrement nouveau n'a été introduit, car il fallait s'assurer que les téléviseurs en noir et blanc existants restaient compatibles. Ce problème a été résolu au niveau international de trois manières différentes, ce qui explique la coexistence des normes NTSC, PAL et SECAM. Leur point commun est que l'information couleur a été ajoutée au signal monochrome existant.

Alors que le PAL et le NTSC sont similaires dans leur principe (modulation en quadrature pour la couleur), le SECAM diffère par l'utilisation de la modulation de fréquence.

Les couleurs qui sont fournies dans leurs trois composantes de base (rouge, vert et bleu) doivent être converties dans un espace couleur YUV ou YCbCr [9] avant de pouvoir être émises par un signal vidéo composite. La conversion de RGB en YUV se calcule comme suit :

$$Y = 0{,}299 * R + 0{,}578 * G + 0{,}144 * B$$
$$U = 0{,}493 * (B - Y)$$
$$V = 0{,}877 * (R - Y)$$

Cela transforme les valeurs de R, G et B en un niveau compris entre 0 et 1. Les formules montrent qu'une mise en œuvre pour les microcontrôleurs nécessite soit une puissance de calcul, soit des compétences en programmation.

Voici une explication de base de l'incorporation de la couleur basée sur les normes PAL et NTSC. La **figure 1** montre un signal PAL monochrome auquel aucune information de couleur n'a encore été ajoutée. Pour les informations de couleur, PAL utilise une sous-porteuse à 4,433 618 75 MHz, qui sert de signal de référence. Cette référence est envoyée

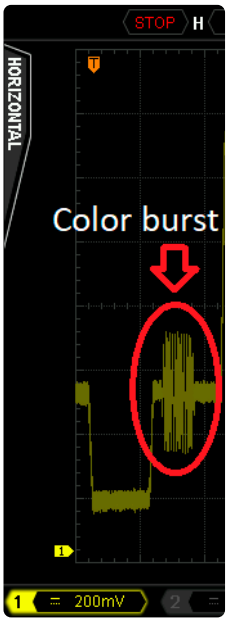

Figure 14. Oscillogramme de la salve de couleurs d'un signal PAL.

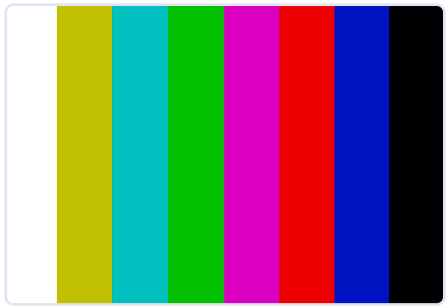

Figure 15. Mire avec barres de couleur.

dans chaque ligne sous forme de « salve de couleur » (**figure 14**) pendant la synchronisation horizontale. La mire (**figure 15**) sert à expliquer comment sont codées les différentes couleurs de l'image test. L'image de test correspond aux barres de couleur de l'UER [10] ; un Raspberry Pi Zero sert de générateur de signal d'image.

Le signal d'une ligne d'image est représenté sur la **figure 16**. La luminosité (ligne blanche comme valeur moyenne du signal) et l'amplitude de l'information de couleur sont facilement visibles. Il existe une troisième information dans le signal, qui est cachée dans la phase du signal. La **figure 17** montre (entouré en rouge) le changement de phase lors de la transition vers une autre couleur.

Ainsi, les informations relatives à la luminosité et à l'amplitude du signal couleur et au déphasage sont contenues dans le signal. S'il y a une erreur dans la phase ou son évaluation pendant la transmission ou la réception du signal, la couleur de l'image change, ce qu'on pouvait compenser manuellement sur les récepteurs NTSC au moyen d'un bouton de « réglage de la teinte » [11] (plus tard, il y eut aussi des solutions électroniques). Le PAL évite ce problème en décalant l'information de phase de 180° une ligne sur deux, ce qui compense l'erreur de phase entre deux lignes consécutives.

Vidéo composite couleur avec ESP32

Un ESP32 peut facilement produire un signal composite monochrome avec quelques petites astuces, et même un Arduino est capable de le faire. Mais la génération d'un signal composite couleur impose des exigences nettement plus élevées en matière de modulation.

En 2018, bitluni a démontré sa méthode de génération d'un signal composite couleur en utilisant un ESP32 [12]. Il a montré que les 13,33 MSa/s des CNA de l'ESP32 sont suffisants pour ajouter une porteuse et des informations de couleur au signal. Comme pour le

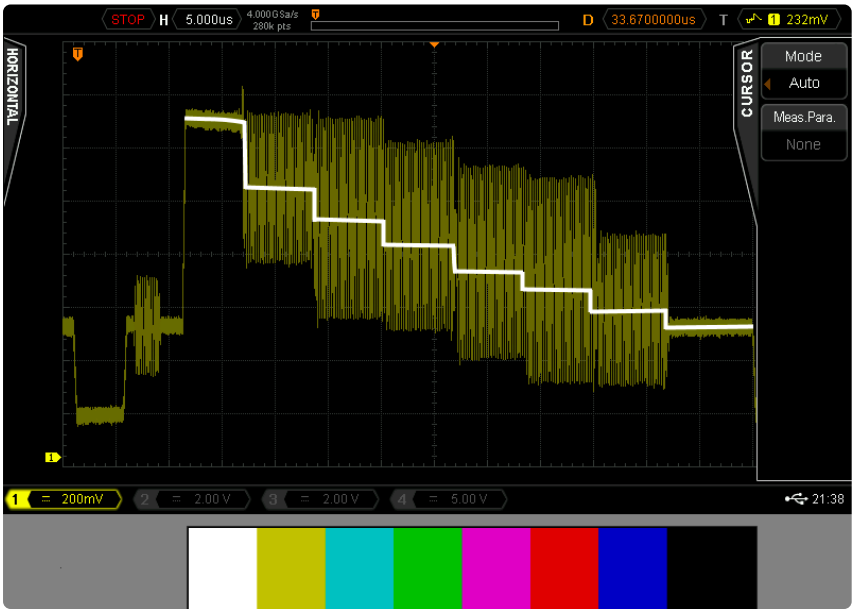

Figure 16. Oscillogramme d'un signal couleur PAL.

Figure 17. Changements de phase pendant le changement de couleur.

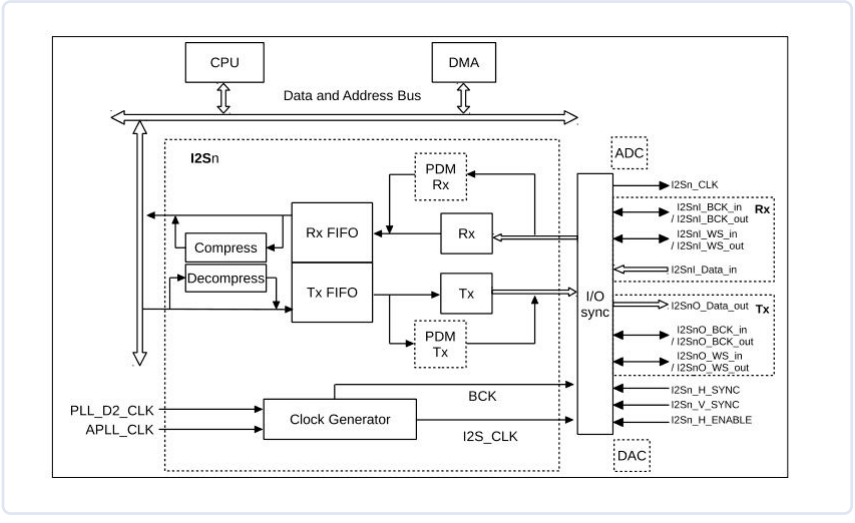

Figure 18. Le bloc I2S de l'ESP32. (Source : Espressif/tinyurl.com/yrrbnjak)

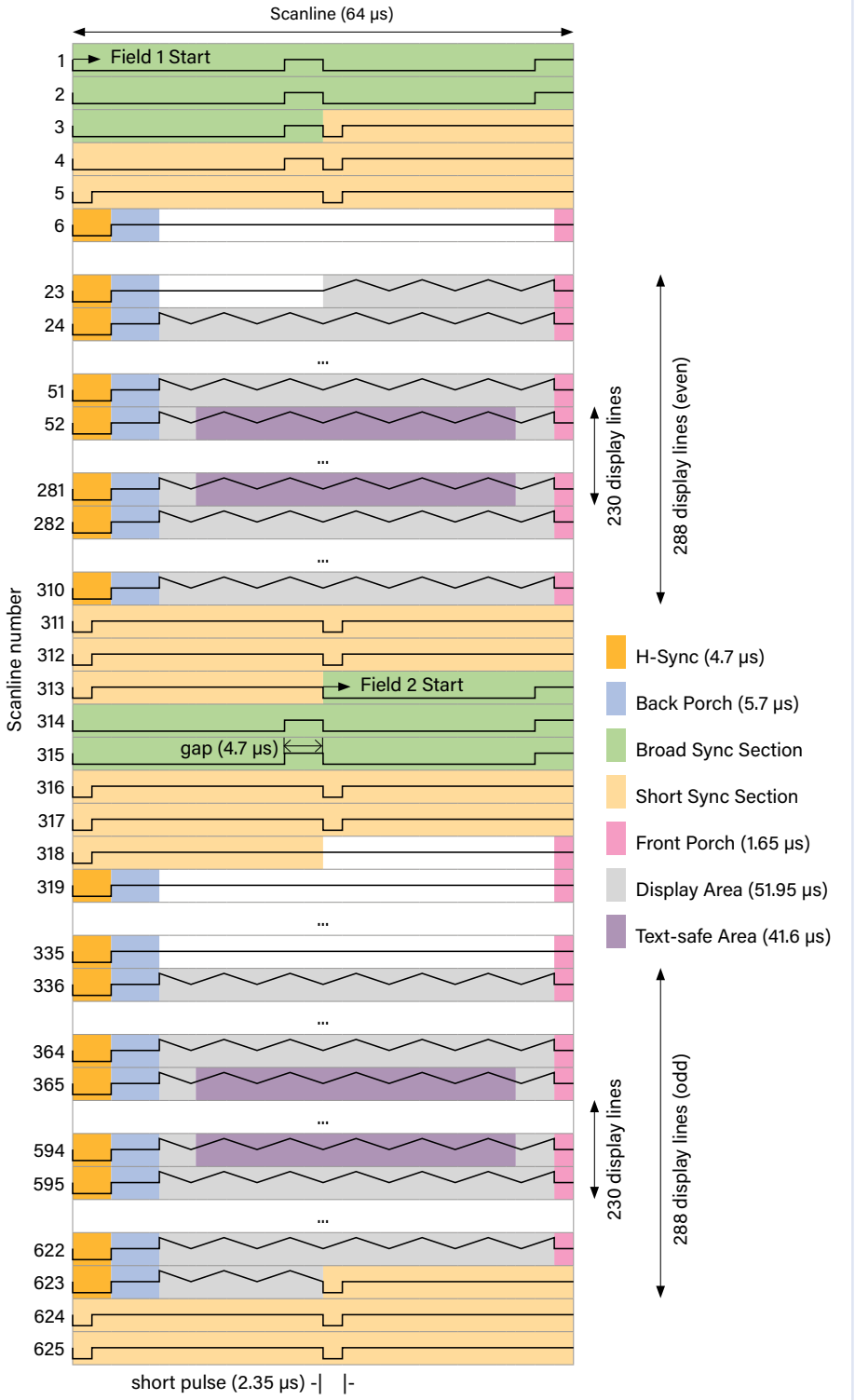

Figure 19. Signal de sortie d'image avec une demi-ligne de trame. (Source : batsocks.co.uk/https://tinyurl.com/4fychhmk)

signal monochrome, un cœur de processeur de l'ESP32 est utilisé pour générer le signal vidéo. La configuration utilise le bloc I2S de l'ESP32 pour envoyer les données au CNA (**figure 18**).

Dans les systèmes PAL et NTSC, l'image est entrelacée, c'est-à-dire qu'elle est composée de deux trames successives avec les lignes impaires et les lignes paires. Comme nous l'avons déjà mentionné, cela permet de doubler le taux de rafraîchissement perçu (pour une même largeur de bande), ce qui réduit considérablement le scintillement perçu. La synchronisation de ces deux trames présente quelques particularités, comme le fait qu'il n'y a qu'une demi-ligne à la fin de l'image dans l'une des deux trames (**figure 19**).

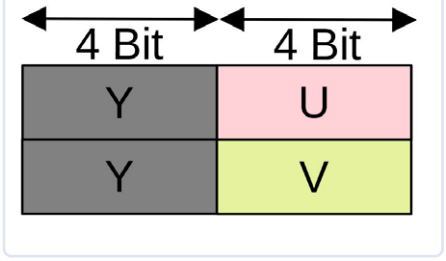

Figure 20. Stockage des informations sur les couleurs par le bitluni.

Comme la RAM d'un ESP32 ne suffit que pour produire une image avec la moitié de la résolution, les lignes doivent être dessinées deux fois : le contenu de la première ligne apparaît donc également sur la deuxième. L'entrelacement garantit donc que le même contenu est dessiné deux fois. On peut alors se demander s'il faut continuer à émettre les deux trames ou s'il suffit d'utiliser deux fois la trame paire, ce qui simplifie considérablement le code d'émission. C'est cette astuce qui a déjà été utilisée par la Super Nintendo Entertainment System (SNES). Le code de Bitluni l'utilise également pour simplifier le code pour la synchronisation. Ainsi, seules 288 lignes réelles à 50 Hz sont émises (288p). Pour NTSC, le mode équivalent serait de 240 lignes à 60 Hz (240p).

Contrairement à la sortie d'images monochromes, les informations sur les couleurs doivent désormais être conservées dans la mémoire vive de l'ESP32. En général, les couleurs sont représentées par leurs composantes rouge, verte et bleue. Cependant, pour générer un signal composite conforme aux normes, il est préférable de stocker les informations de couleur dans la RAM sous forme de valeurs YUV. C'est le seul moyen pour l'ESP32 et son CNA de sortir ces données assez rapidement. Comme la quantité de RAM de l'ESP32 est limitée, il faut recourir à quelques astuces pour coder les données. bitluni stocke les informations dans la RAM sous forme de valeurs YU, YV et V combinées avec une résolution de 4 bits chacune (**figure 20**).

Malheureusement, avec l'approche de bitluni, la sortie vidéo composite n'est possible que pour le standard PAL, avec le NTSC cela ne fonctionne pas. Le rapport entre les 3,579 545 MHz de la salve de couleur NTSC et la fréquence d'échantillonnage du CNA est si défavorable qu'aucune salve de couleur utilisable ne peut être émise et que les récepteurs ne peuvent pas se synchroniser.

Figure 21. Emulateurs pour consoles de jeux 8 bits. (Source : tinyurl.com/ykd9ezap)

Couleur avec ESP32 en PAL et NTSC

Avec son projet « esp_8_bit », l'utilisateur de GitHub rossumur a prouvé qu'il est également possible de créer de la couleur conforme au NTSC avec un ESP32. Ce projet est une collection d'émulations basées sur l'ESP32 de diverses consoles 8 bits, de l'Atari 400 à la Nintendo Entertainment System et à la Sega Master System (**figure 21**).

Ce projet utilise simplement une broche de l'ESP32 pour sortir un signal composite en NTSC ou PAL. L'astuce est cachée dans le PLL audio de l'ESP32. Elle peut être utilisée pour faire fonctionner le CNA avec des fréquences d'échantillonnage allant jusqu'à environ 20 MHz. Le quadruple de la fréquence de la porteuse couleur NTSC, est 14,318 182 MHz (pour PAL 17,734 475 MHz). Avec l'APLL, on peut générer 14,318 180 MHz et 17,734 476 MHz, ce qui est assez proche des fréquences nécessaires pour le PAL et le NTSC. De cette façon, le CNA émet un multiple entier de la fréquence de la sous-porteuse couleur, ce qui donne des signaux vidéo traitables.

Avec l'APLL, le CNA de l'ESP32 peut non seulement produire un signal vidéo, mais cette approche est également très intéressante pour d'autres applications de synthèse numérique directe (DDS). Cependant, l'utilisation de l'APLL présente un inconvénient pour les fréquences d'échantillonnage élevées. Le CNA de l'ESP32 possède deux canaux. Si l'APLL alimente l'un d'entre eux avec des données pour la sortie vidéo et que l'on tente d'alimenter le second avec des données audio provenant de l'interface I2S, le CNA souffrira d'interruptions apparaissant sur les deux canaux, ce qui oblige à recourir à d'autres méthodes pour la sortie audio.

« espflix » a été développé basé sur le projet « esp_8_bit ». Il permet à l'ESP32 de jouer des vidéos qui sont stockées sur un serveur du projet espflix. Toutefois, cela nécessite quelques concessions sur le matériel vidéo. Des détails à ce sujet peuvent être trouvés sur la page Github associée [14].

Partie 2 : perspectives d'avenir

La deuxième partie de l'article traite du VGA, du DVI et des sprites. En particulier avec un ESP32 ou le RP2040 du Raspberry Pi Pico, on peut obtenir des effets surprenants. Si vous ne voulez pas attendre la deuxième partie, vous pouvez vous inscrire au webinaire *« Microcontroller as Pixel Artist »* [15], où les sujets VGA, DVI et sprites seront également abordés.

220047-04 — VF : Helmut Müller

Webinaire « Le microcontrôleur comme artiste du pixel »

Étant donné que les images animées en disent plus que les mots, surtout sur ce sujet, et que cet article ne pouvait offrir la place pour une revue étendue, nous vous invitons à participer au webinaire Elektor consacré aux applications graphiques avec Arduino, ESP32 et Raspberry Pi Pico. Les inscriptions sont d'ores et déjà ouvertes sur [15].
www.elektormagazine.com/webinars

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (mathias.claussen@elektor.com) ou contactez Elektor (redaction@elektor.fr).

LIENS

[1] Vidéo composite : https://fr.wikipedia.org/wiki/Vid%C3%A9o_composite
[2] Livre en anglais « Vidéo analogique », Angelo La Spina : https://www.elektor.fr/analogue-video-e-book
[3] Bibliothèque Arduino TVOut : https://github.com/Avamander/arduino-tvout
[4] Hackvision : https://nootropicdesign.com/hackvision/
[5] Échelle de résistances : https://fr.wikipedia.org/wiki/%C3%89chelle_de_r%C3%A9sistances
[6] pico-composite8 : https://github.com/obstruse/pico-composite8
[7] Événement Hackaday : https://hackaday.com/2022/08/17/chips-remembered-the-scenix-ubicom-parallax-sx
[8] Couleur avec les puces SX : https://elinux.org/images/e/eb/Howtocolor.pdf
[9] Espace couleur YCbCr : https://fr.wikipedia.org/wiki/YCbCr
[10] Barres de couleur de l'UER : https://fr.wikipedia.org/wiki/YUV
[11] Contrôle de la teinte NTSC : https://en.wikipedia.org/wiki/Tint_control
[12] bitluni, « ESP32 Composite Video » : https://bitluni.net/esp32-composite-video
[13] Utilisateur Github rossumur : https://github.com/rossumur
[14] espflix : https://github.com/rossumur/espflix
[15] Webinaire « Le microcontrôleur comme artiste du pixel » : https://www.elektormagazine.com/webinars

Produits

- **Raspberry Pi Pico RP2040 (SKU 19562)**
  www.elektor.fr/19562
- **ESP32-DevKitC-32D (SKU 18701)**
  www.elektor.fr/18701
- **Arduino Uno Rev3 (SKU 15877)**
  www.elektor.fr/15877