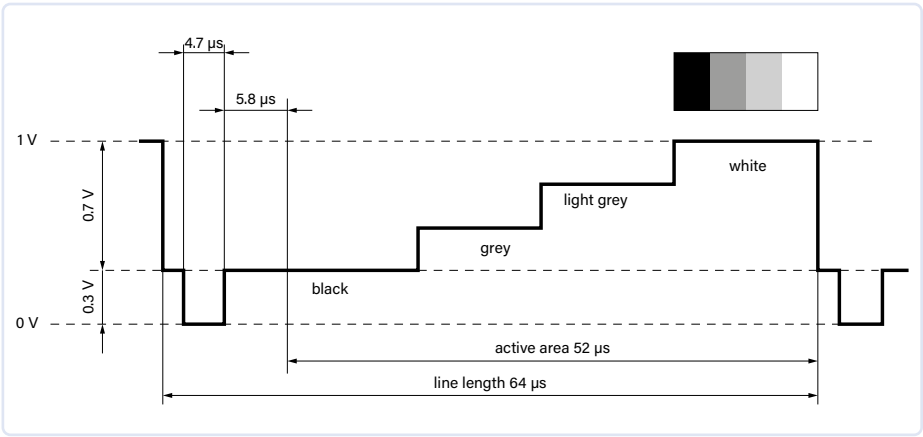
Video-output met microcontrollers (1)

composiet video

Mathias Claussen (Elektor Lab)

Het onderwerp video-output met microcontrollers gaat terug tot het begin van deze kleine allround chips. De microcontrollers van tegenwoordig hebben aanzienlijk meer rekenkracht dan bijvoorbeeld de bijna 42 jaar oude Sinclair ZX81 thuiscomputer, maar zelfs de huidige microcontrollers zijn nog ver verwijderd van de geheugengrootte van moderne grafische kaarten gemeten in gigabytes. Niettemin slagen ontwikkelaars er nog steeds in om verbazingwekkende bewegende beelden te produceren met een ATmega, een ESP32 of een RP2040. In het eerste deel van deze serie behandelen we de uitvoer van composiet video. In het volgende deel gaan we verder met VGA en zelfs DVI. In elk geval zijn een paar trucs en exacte timing nodig. Daarom gaat het hier niet alleen om theorie, maar ook om praktische voorbeelden als uitgangspunt voor je eigen experimenten.



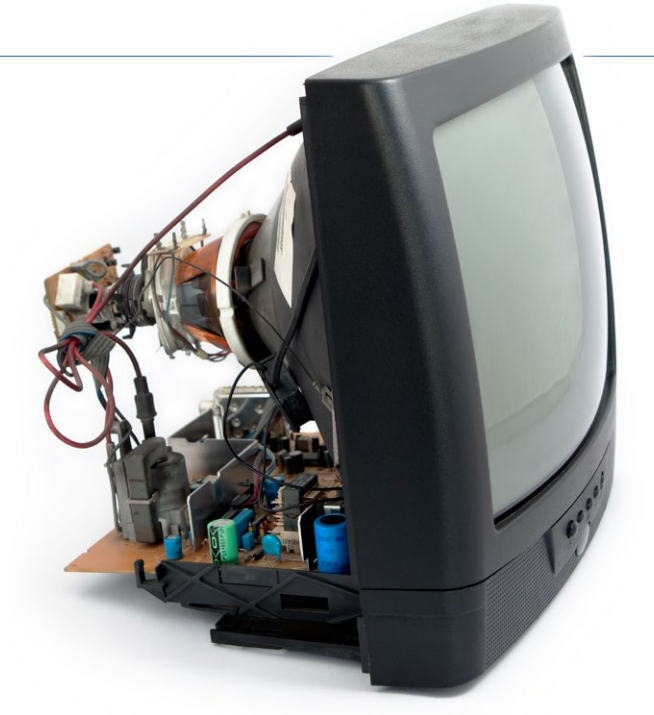
Figuur 1: VBS-signaal met PAL-timing. (Bron: Wikipedia)

De geschiedenis van videoformaten gaat ver terug tot het begin van de televisie. Net als bij radio-uitzendingen werd ook voor televisie gestreefd naar standaardisatie en normen. In het tijdperk van de analoge kleurentelevisie waren de meest gangbare normen National Television Systems Committee (NTSC - Noord- en Zuid-Amerika, Japan), Phase Alternating Line (PAL - Europa, Zuid-Amerika, Afrika en Azië), en Séquentiel Couleur à Mémoire (SECAM - Frankrijk, Afrika en USSR).

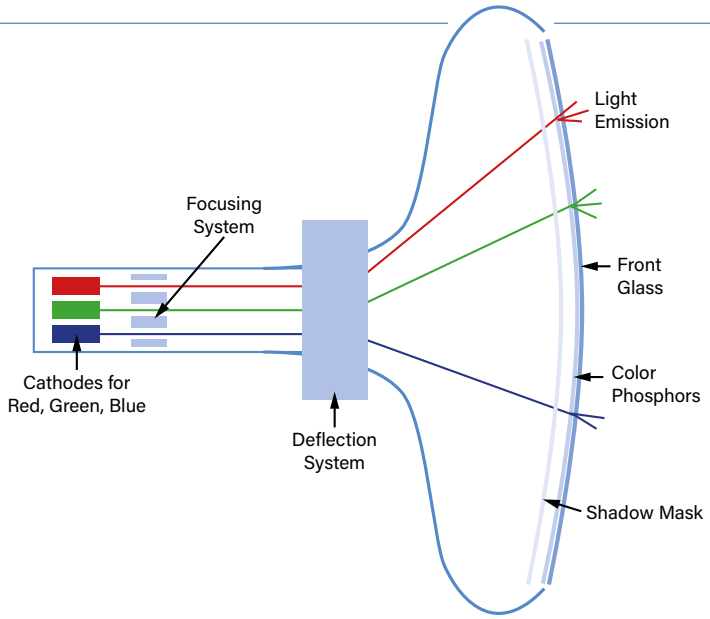
De basisprocedure voor analoge transmissie van videosignalen werd voor deze normen uitgevoerd volgens Video Blanking en Sync (VBS). De resolutie en beeldsnelheid voor VBS hangen af van de onderliggende televisiestandaard. Voor NTSC is dat 480 zichtbare lijnen met 640 zichtbare pixels bij 59,94 velden per seconde. PAL en SECAM hebben 576 zichtbare lijnen met 720 zichtbare pixels en 50 velden per seconde (576i). Daarnaast verschillen NTSC, PAL en SECAM ook in modulatie en de manier waarop kleurinformatie wordt toegevoegd. In het tijdperk van digitale beeldoverdracht hebben deze drie transmissiemethoden hun betekenis grotendeels verloren. Ze zijn echter bij ons gebleven in de vorm van digitale beeldformaten voor DVD-video of Standard Definition Television (SDTV).

Composiet video met CVBS

De eerste gestandaardiseerde televisiesignalen werden ontworpen voor de transmissie van monochrome beelden. In de VS en Europa werden verschillende beeldformaten ontwikkeld. De timing van een VBS signaal in



Figuur 2: Geopend tv-toestel met kathodestraalbuis. (Bron: Shutterstock / Sergio Sergo)



Figuur 3: Schematische opbouw van een kleurenbeeldbuis. (Bron: ITWissen.info)

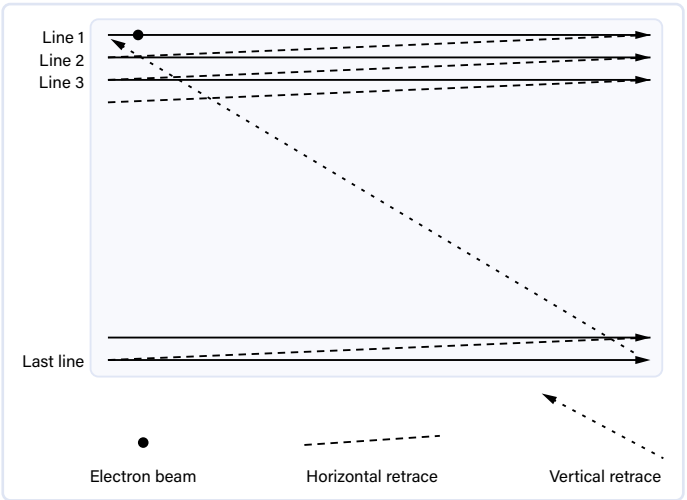
het PAL formaat is weergegeven in **Figuur 1** als voorbeeld van deze standaarden.

Deze norm was vooral geschikt voor beeldweergave met beeldbuizen (**Figuur 2**). Het werkingsprincipe van zulke beeldbuizen verschilt enorm van LCD's of OLED-schermen. In een beeldbuis wordt een gemoduleerde, horizontaal en verticaal afgebogen elektronenbundel op een fosforlaag gericht. Op het trefpunt gloeit de fosfor dan in verhouding tot de intensiteit van de elektronenbundel (**Figuur 3**). Tijdgesynchroniseerde horizontale en verticale afbuiging levert een tweedimensionaal beeld op.

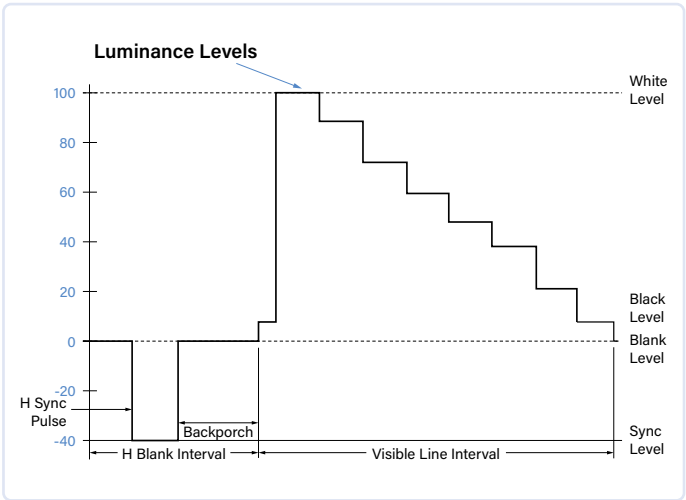
Vanuit het perspectief van de kijker is een beeld (zwart-wit) opgebouwd in lijnen van links naar rechts en de lijnen vervolgens van boven naar beneden (**Figuur 4**). De eerste "pixel" per beeld bevindt zich dus linksboven. Deze logica is in moderne digitale beeldschermen grotendeels gehandhaafd.

De historische kennis over beeldgeneratie door middel van een bewegende elektronenbundel maakt het gemakkelijker om moderne videosignalen te begrijpen. Het signaal in Figuur 1 bevat de structuur van een beeldlijn. In het voorste gedeelte (**Figuur 5**) van het signaal is de "horizontale blanking" te zien. Aan het begin van de "horizontale blanking" bevindt de elektronenbundel zich aan de rechterrand van de beeldbuis en moet daarom voor de volgende lijn terug naar uiterst links. Tijdens deze sprong van rechts naar links wordt de elektronenbundel onderdrukt door het helderheidssignaal op zwart of zelfs "zwarter dan zwart" te zetten.

Horizontale blanking duurt 12 µs voor PAL en 10,9 µs voor NTSC. Het bestaat uit drie delen, het "voorportaal", de "sync dip" en het "achterportaal". Het helderheidsniveau van het voorportaal (PAL = 1,65 µs / NTSC = 1,4 µs) ligt op of iets onder de waarde van zwart bij 0,3 V,



Figuur 4: Op lijnen gebaseerde beeldopbouw.

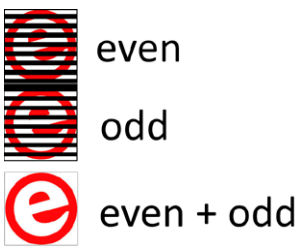


Figuur 5: Horizontale blanking. (Bron: www.edn.com)

terwijl de elektronenbundel nog verder naar rechts buiten het zichtbare beeld komt. De spanning bij de sync-dip (PAL en NTSC = 4,7 µs) is 0 V en ligt dus aanzienlijk lager dan het zwartniveau. De sync-dip zorgt ervoor dat de elektronenbundel van rechts naar links springt. Het achterportaal (PAL en NTSC = 4,7 µs) levert de referentie voor de zwartwaarde van 0,3 V voor de lijn. De te tekenen inhoud, d.w.z. de beeldinformatie, begint aan het eind van het achterportaal. Er is dan 52 µs beschikbaar voor het zichtbare deel van de lijn. Het aantal pixels of sample punten in deze 52 µs hangt af van het videoformaat. In PAL zijn er meestal 720 zichtbare pixels. Als alle lijnen getrokken zijn en de elektronenbundel rechtsonder heeft bereikt, moet een sprong naar het beginpunt van het beeld (linksboven) worden uitgevoerd. Dit proces wordt in een apart hoofdstuk beschreven.

Synchronisatie en interlacing

De analoge bandbreedte voor het uitzenden van televisiebeelden was voldoende voor slechts 25 beelden per seconde voor PAL en 29,97 beelden per seconde voor NTSC. Voor het menselijk oog resulteert dit in een bewegend beeld, maar het is niet erg vloeiend en geeft een zeer onaangenaam flikkerend effect. Met een snellere beeldsnelheid van 50 of zelfs 59,94 beelden per seconde wordt het flikkeren merkbaar verminderd. De meeste mensen ervaren dit dan als een vloeiende visuele indruk. Omdat de beschikbare radiobandbreedte beperkt was, werd de interlaced scan technologie geïntroduceerd: in plaats van alle beeldlijnen op een hogere frequentie als volledige frames uit te zenden, worden de lijnen met even of oneven nummers afwisselend op tweemaal de frequentie uitgezonden, dus respectievelijk 50 of 59,94 Hz (**Figuur 6**).



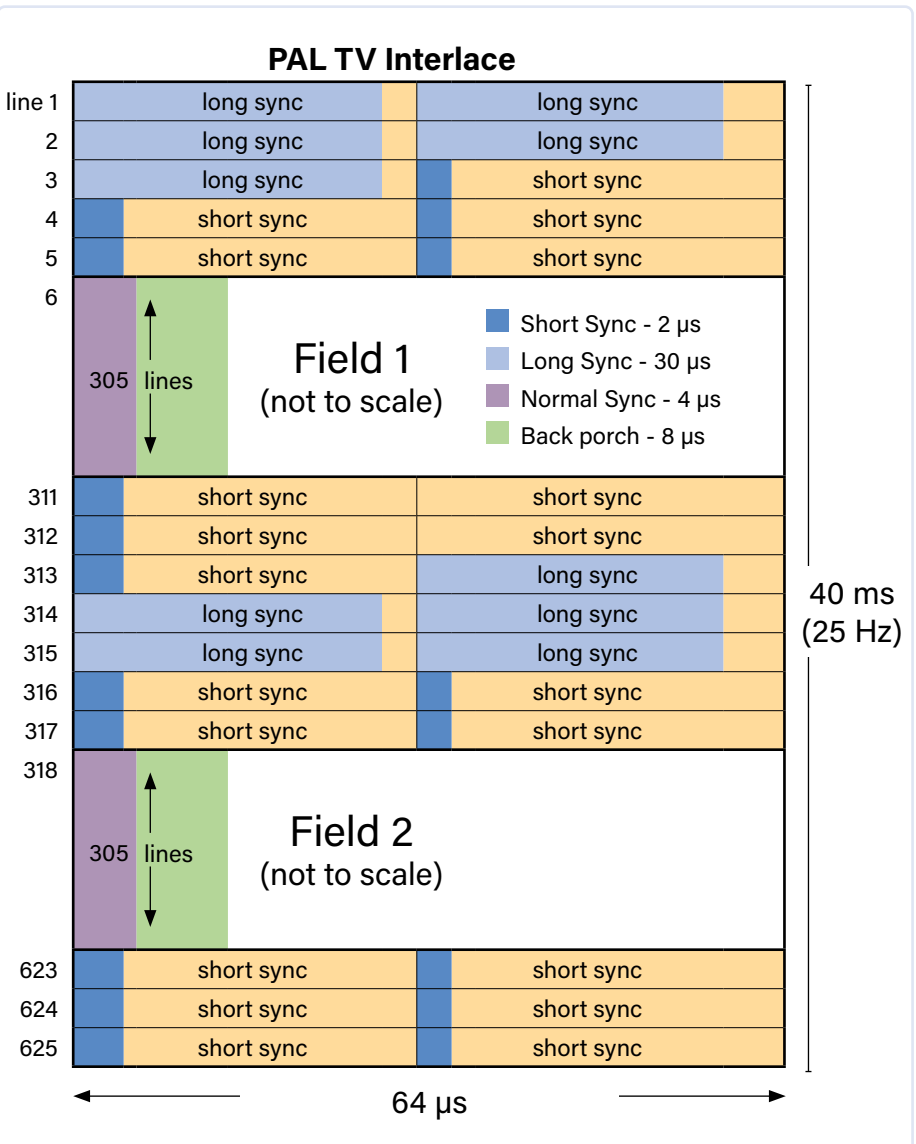
Figuur 6: Frame-opbouw van twee velden.

Dus wordt eerst een half frame ("veld") met alle oneven lijnnummers uitgezonden, gevolgd door het tweede veld met de even lijnnummers. Om de lijnen van de twee velden in de openingen van het respectieve andere veld te laten vallen, is er een detectiesignaal aan het begin van elk veld.

Omdat de lijnen van een compleet frame genummerd zijn volgens de zendvolgorde, heeft het eerste veld in PAL de lijnen 1 tot en met 313 en het tweede veld telt dan door tot en met lijn 625. De verticale synchronisatie voor het einde van een veld en het begin van het volgende is wat lastig: lijnen 311 tot 317 bij de overgang van oneven naar even velden en lijnen 623 tot 5 bij de overgang van het even naar het oneven veld van het volgende complete frame zijn niet zichtbaar. In deze lijnen worden snellere synchronisatiepulsen met niveaus tussen 0,3 V en 0 V uitgezonden. **Figuur 7** toont de volgorde voor een compleet PAL-frame. Aan de hand van deze synchronisatiepulsen kan de elektronica herkennen of het om het veld met even of oneven lijnnummers gaat.

Monochrome video met AVR-controllers

Basisinformatie over composiet signalen zoals CVBS of FBAS kun je vinden op Wikipedia [1]. Als je meer details en vaardigheidjes zoekt, kun je die zeker vinden in het boek *Analogue Video* [2] van Angelo La Spina.



Figuur 7: Samenstelling van een compleet PAL-beeld. (Bron: martin.hinner.info)

Als het videosignaal als VBS signaal (geen kleurinformatie, alleen helderheid) wordt doorgegeven met een geschikte composiet videokabel (**Figuur 8**), kan het genereren van de helderheidsinformatie vrij eenvoudig zijn. Als je alleen wit en zwart gebruikt in plaats van grijstinten, hoef je slechts twee geschikte spanningen plus synchronisatie uit te zenden. Behalve dat dit de schakeling vereenvoudigt, vermindert het ook het geheugen dat nodig is voor de beeldgeneratie.

De Atmega328P microcontroller van een Arduino Uno kan dus een monochroom beeld genereren met 128x96 pixels. Alle pixels kunnen worden opgeslagen in het interne geheugen van de Arduino Uno, want er zijn slechts 1536 bytes nodig. De bedrading van een Arduino Uno met composietuitgang is weergegeven in **Figuur 9**.

Zoals je ziet zijn twee weerstanden voldoende om een geschikt signaal uit te voeren. Voor de beelduitvoer hoef je het wiel niet opnieuw uit te vinden met je eigen code, maar kun je de TVOut bibliotheek [3] gebruiken. Deze bibliotheek ondersteunt timing voor NTSC en PAL. De video-uitvoer die ermee gedaan kan worden varieert van eenvoudige tekst tot je eigen spelletjes. Hackvision [4] is een platform voor een spelcomputer. Dit is open hardware waarvoor in extreme gevallen alleen een breadboard en een paar onderdelen nodig zijn. Een Arduino kan ook omgezet worden in een Hackvision platform (**Figuur 10**) en zo de bijbehorende bibliotheek van spelletjes gebruiken.

Voor video-output met een AVR microcontroller is één van zijn timers nodig. De horizontale en verticale synchronisatie wordt verzorgd door Timer1 en zijn uitgangspen PB1 (OC1A). Op basis van de timer worden de pixels lijn voor lijn uitgevoerd via een speciale pin (PD7).

De waarden voor de weerstanden bij PB1 en PD7 kunnen vrij eenvoudig worden bepaald. Een composiet ingang op monitoren of TV-toestellen heeft een ingangsimpedantie van 75 Ω. Het niveau voor synchronisatie ligt tussen 0 V en 0,3 V. De spanning voor de helderheidswaarden van de pixels ligt respectievelijk tussen 0,3 V voor zwart en 1 V voor wit. In het volgende wordt aangenomen dat de microcontroller gevoed wordt met 5 V.

Figuur 8: Kabel voor composiet video met RCA-connector. (Bron: Shutterstock / Woodpond)

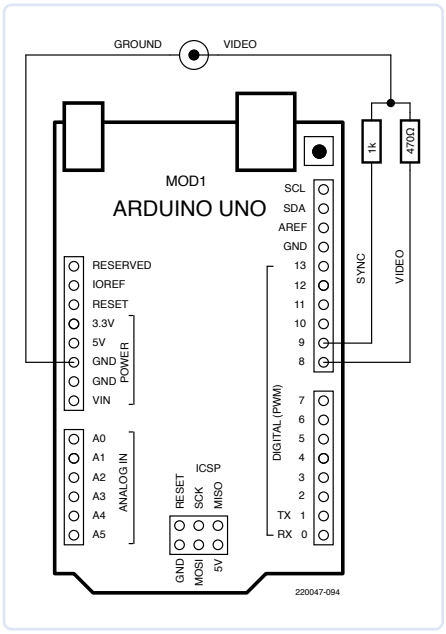
Figuur 11 toont de spanningsdeler voor de video-uitgang bij de typical impedantie van 75 Ω. Pin 9 levert de synchronisatiesignalen. Wiskundig gezien zou een waarde van 1175 Ω nodig zijn voor R1, zodat de spanningsval op R2 0,3 V is met een hoog niveau op pen 9. 1 kΩ levert maximaal 0,34 V op R2, wat nauwkeurig genoeg is voor dit doel. Pin 7 voert de helderheidswaarden uit. Waarden tussen 0,34 V (zwart) en 1 V (wit) zijn nu nodig op R2. Om 1 V bij R2 te bereiken is een waarde van 375 Ω nodig voor de parallelle verbinding van R1 en R3, wat zou resulteren in 600 Ω voor R3. De volgende grotere waarde van 470 Ω uit de E-12 reeks zorgt ervoor dat de spanning op R2 nooit hoger kan worden dan de waarde van 1 V.

Het feit dat dit werkt met een Arduino Uno of een ATmega328P bewijst dat weinig rekenkracht en weinig geheugen volstaat om afbeeldingen op een scherm weer te geven. Omdat het complete beeld in het interne geheugen kan worden bewaard, is het tekenen van een nieuw beeld niet tijdkritisch, alleen het genereren van de videosignalen met behulp van Timer1.

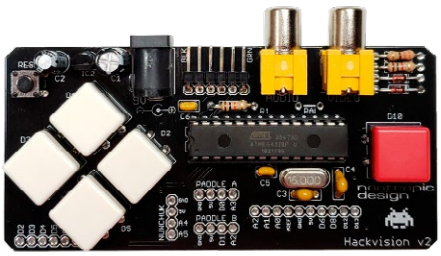
Is het ook mogelijk om met de Arduino grijstinten = meerdere helderheidswaarden uit te voeren? Dat is mogelijk, maar het interne geheugen is de beperkende factor. Met 16 grijstinten (4-bits grijstinten) en geschikte weerstanden zijn 6144 bytes nodig voor het videogeheugen als je de resolutie van 128x96 pixels wilt aanhouden en een compleet beeld in het geheugen moet passen.

Raspberry Pi Pico en Composiet

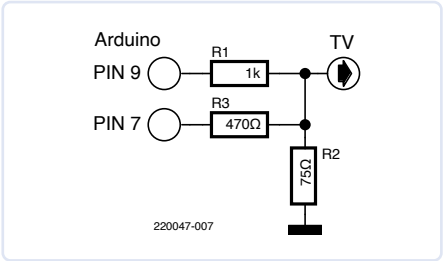
Een Raspberry Pi Pico kan ook een composiet signaal uitvoeren, en ondersteunt gemakkelijk meer dan 50 grijstinten. Hiervoor wordt een R-2R weerstandsladder [5] gebruikt



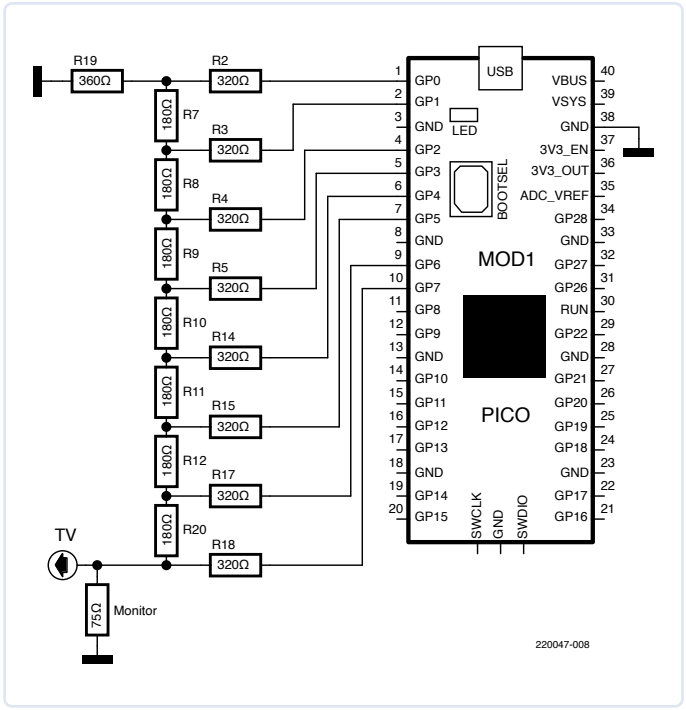
Figuur 9: Z/W video-uitgang met twee weerstanden op een Arduino Uno.



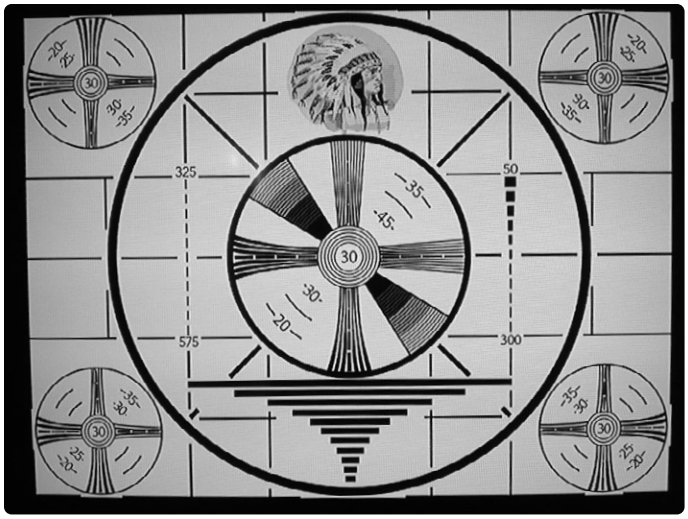
Figuur 10: Hackvision hardware. (Bron: nootropic design)



Figuur 11: De schakeling van de video-uitgang van Figuur 9.



Figuur 12: Een R-2R netwerk als DAC op de Raspberry Pi Pico.



Figuur 13: Grijswaardenbeeld van een Raspberry Pi met 512x384 pixels. (Bron: tinyurl.com/2p8z27a2)

als digitaal/analoog converter. **Figuur 12** toont een geschikte schakeling bestaande uit weerstanden van 180, 320 en 360 Ω. Zoals te zien is op de GitHub pagina van het project "pico-composite8" [6], kan de interne weerstand van de GPIO-pinnen hier niet verwaarloosd worden. De ontwikkelaar van het project heeft hiervoor waarden bepaald van ongeveer 40 Ω. De Pico voert hier een composiet signaal uit volgens NTSC. Hij laadt de beelden ofwel uit RAM of Flash en kan maximaal 512x384 pixels uitvoeren. Het resultaat is te bewonderen in **Figuur 13**.

Het project is echter slechts een haalbaarheidsdemonstratie en biedt geen kant-en-klare generieke bibliotheek. Het feit dat een Raspberry Pi Pico grijswaardenvideo met 512x384 pixels kan weergeven laat zien dat het niet zozeer een kwestie is van rekenkracht maar van de juiste timing. Als een compleet beeld in het RAM van de Raspberry Pi Pico wordt bewaard, blijft er slechts ongeveer 64 KB van de 264 KB beschikbaar voor je eigen toepassingen. Maar als je bedenkt dat zelfs een ATmega spellen als Tetris of Pong aankan, inclusief video-uitvoer, zou dit meer dan genoeg ruimte moeten bieden voor je eigen creaties.

Kleur voor Composiet Video

Meer dan 50 tinten grijs is goed, maar kleur is gewoon beter. Als het gaat om composiet video en kleur, wordt het veel moeilijker dan voorheen om een geschikt analoog signaal te genereren. In 2003 demonstreerde Rickard Gunée het genereren van een composiet signaal (PAL of NTSC) op een Hackaday evenement [7] met behulp van een Scenix / Ubicom SX28 op ongeveer 50 MHz.

Maar hoe moeilijk is het om kleur toe te voegen aan een videosignaal? Het antwoord hangt af van de manier waarop kleurinformatie wordt toegevoegd aan het composiet signaal.

Kleuren dankzij PAL en NTSC

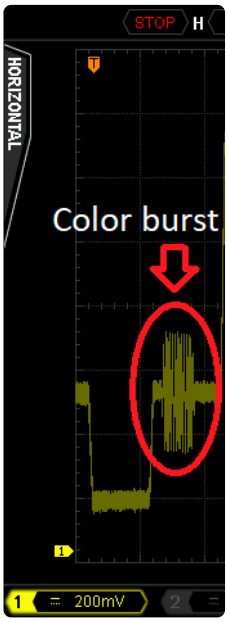
Bij de overgang van zwart-wit naar kleurentelevisie enkele decennia geleden werd geen geheel nieuw signaal gedefinieerd, omdat ervoor gezorgd moest worden dat bestaande zwart-wit televisies het beeld nog steeds konden weergeven. Dit probleem werd internationaal op drie verschillende manieren opgelost, waardoor de normen NTSC, PAL en SECAM naast elkaar kwamen te bestaan. Wat ze gemeen hebben is dat de kleureninformatie werd toegevoegd aan het bestaande monochrome signaal.

Terwijl PAL en NTSC in principe vergelijkbaar zijn (quadratuur modulatie voor kleur), verschilt SECAM in het integreren van kleur door gebruik te maken van frequentie modulatie.

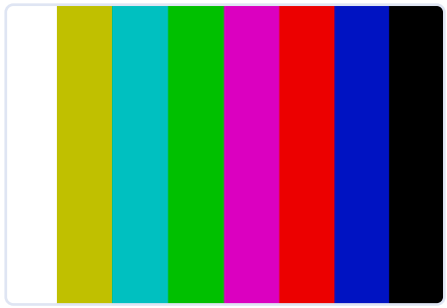
Kleuren die worden geleverd in hun drie basiscomponenten rood, groen en blauw moeten worden omgezet naar een YUV of YCbCr kleurruimte [9] voordat ze kunnen worden uitgevoerd via een composiet videosignaal. De conversie van RGB naar YUV wordt als volgt berekend:

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

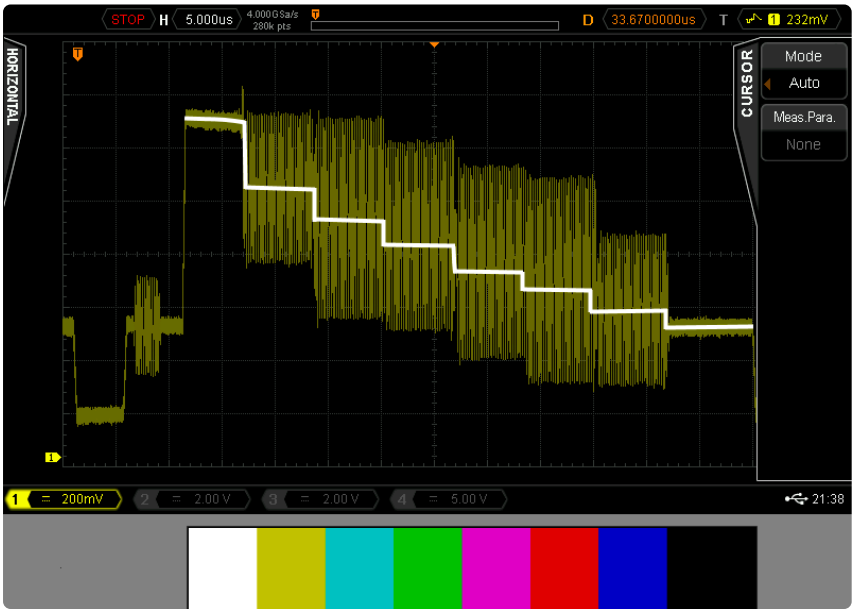
Dit transformeert de waarden voor R, G en B in een bereik tussen 0 en 1. De formules laten zien dat een implementatie voor microcontrollers ofwel rekenkracht ofwel enige programmeervaardigheid vereist.



Figuur 14: Kleurenburst oscillogram van een PAL-signaal.

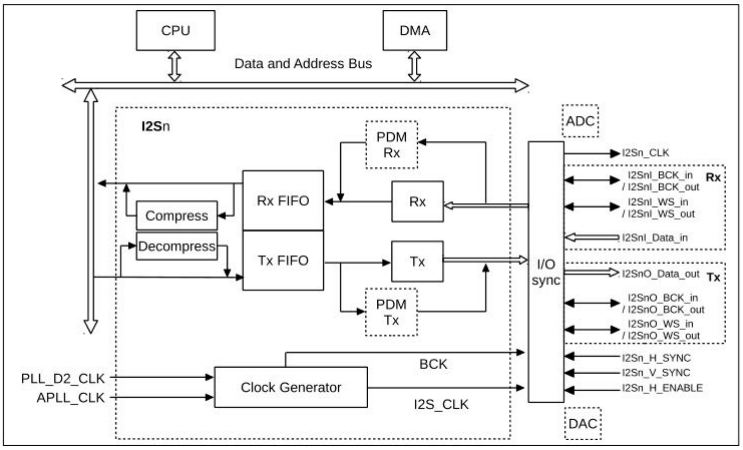


Figuur 15: Testpatroon met kleurenbalken.



Figuur 16: Oscillogram van een PAL-kleursignaal.

Figuur 17: Faseveranderingen tijdens de kleurverandering zijn gemarkeerd.



Figuur 18: Het I2S-blok van de ESP32. (Bron: Espressif / tinyurl.com/yrrbnjak)

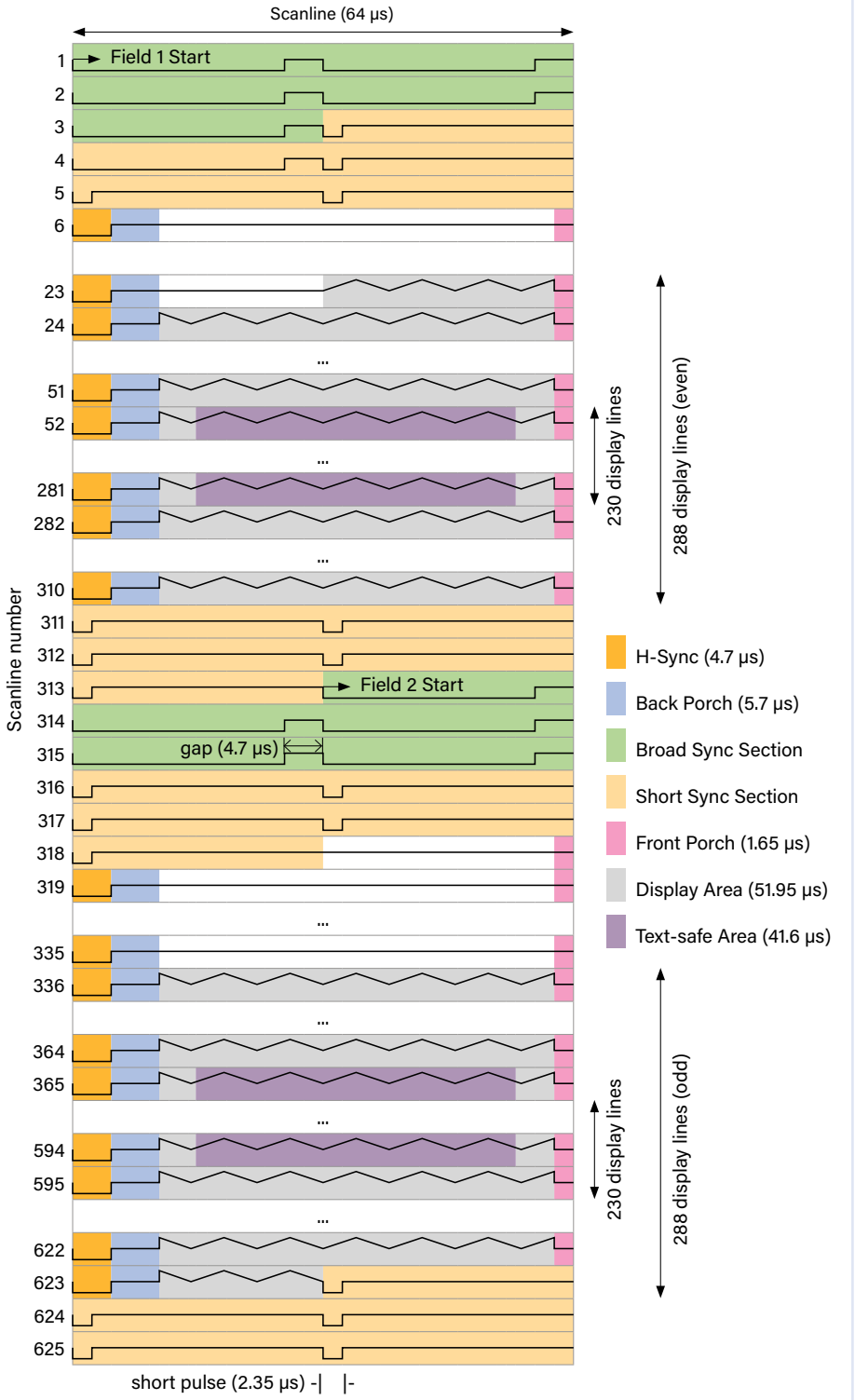
Hieronder volgt een basisuitleg van de implementatie van kleur op basis van PAL en NTSC. **Figuur 1** toont een monochroom PAL-signaal waaraan nog geen kleurinformatie is toegevoegd. Voor kleurinformatie gebruikt PAL een subcarrier op 4,43361875 MHz, die gebruikt wordt als referentiesignaal. Deze referentie wordt in elke lijn verzonden als een "color burst" (**Figuur 14**) tijdens de horizontale synchronisatie. Het testpatroon (**Figuur 15**) wordt gebruikt om uit te leggen hoe de verschillende kleuren van het testbeeld worden gecodeerd. Het testbeeld komt overeen met de EBU kleurenbalken [10]; een Raspberry Pi Zero dient als beeldsignaalgenerator.

Het signaal voor één beeldlijn is te zien in **Figuur 16**. De helderheid (witte lijn als gemiddelde waarde van het signaal) en de amplitude van de kleurinformatie zijn goed zichtbaar. Er is nog een derde stukje informatie in het signaal, dat verborgen is in de signaalfase. **Figuur 17** toont (in rood) de faseverandering bij de overgang naar een andere kleur.

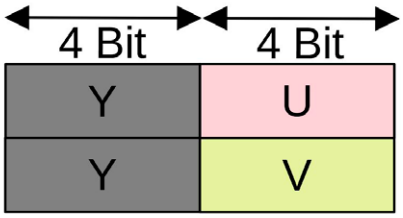
De informatie voor de helderheid en de amplitude van het kleursignaal en de faseverschuiving zit dus in het signaal. Als er tijdens transmissie of ontvangst van het signaal een fout optreedt in de fase of de evaluatie daarvan, verandert de kleurindruk van het beeld, wat NTSC ontvangers handmatig konden compenseren door middel van een "tintregeling" [11] via een draaiknop (later ook met elektronische oplossingen). Bij PAL werd dit probleem vermeden omdat de fase-informatie bij elke tweede lijn 180° wordt verschoven. Een fasefout wordt dus gecompenseerd tussen twee aangrenzende lijnen.

Kleuren Composiet Video met ESP32

Een ESP32 kan gemakkelijk een monochroom composiet signaal uitzenden met een paar kleine trucjes, en zelfs een Arduino is daartoe in staat. Maar het genereren van een kleuren-composiet signaal stelt aanzienlijk hogere eisen aan de modulatie.



Figuur 19: Signaal van beelduitvoer met halve beeldlijn. (Bron: batsocks.co.uk / https://tinyurl.com/4fychhmk)



Figuur 20: Opslag van kleurinformatie door bitluni.

In 2018 demonstreerde bitluni zijn aanpak voor het genereren van een kleurencomposiet signaal met behulp van een ESP32 [12]. Aangetoond werd dat de 13,33 MSa/s van de DAC's van de ESP32 voldoende zijn om een kleurendrager en kleurinformatie in het signaal te integreren. Net als bij het monochrome signaal wordt één processorkern van de ESP32 gebruikt om het videosignaal te genereren. De opstelling gebruikt het I2S-blok van de ESP32 om gegevens naar de DAC te sturen (**figuur 18**).

In PAL en NTSC wordt het beeld interlaced weergegeven, d.w.z. achtereenvolgens in velden met oneven lijnen en even lijnen. Zoals gezegd verdubbelt dit de waargenomen verversingssnelheid (bij dezelfde bandbreedte), wat de waargenomen flikkering aanzienlijk vermindert. De timing van deze twee velden heeft een paar eigenaardigheden, zoals slechts een halve kaderlijn aan het eind van het beeld in een van de twee velden (**Figuur 19**).

Omdat het RAM van een ESP32 slechts voldoende is om een beeld met de halve resolutie uit te voeren, moeten lijnen tweemaal getekend worden: de inhoud van regel één verschijnt dus ook in regel twee. De interlacing zorgt er dus voor dat dezelfde inhoud tweemaal wordt getekend. Nu rijst de vraag of men werkelijk zowel het oneven veld als het even veld moet uitvoeren, of dat het ook voldoende zou zijn om het even veld tweemaal te gebruiken en zo de code voor de uitvoer aanzienlijk te vereenvoudigen. Precies deze truc werd al gebruikt door het Super Nintendo Entertainment System (SNES). En ook de code van bitluni gebruikt deze methode om de code voor synchronisatie te vereenvoudigen. Zo worden slechts 288 echte lijnen bij 50 Hz uitgevoerd (288p) - voor NTSC zou het equivalent 240 lijnen bij 60 Hz (240p) zijn.

In tegenstelling tot de monochrome beeldweergave moet nu ook de kleureninformatie in het RAM van de ESP32 worden opgeslagen. In het algemeen worden de kleuren weergegeven door hun componenten rood, groen en blauw. Voor het genereren van een standaard composiet signaal kan de kleurinformatie echter beter in het RAM worden opgeslagen als YUV waarden. Dit is de enige manier waarop de ESP32 en zijn DAC deze gegevens snel genoeg kunnen uitvoeren. Omdat de hoeveelheid RAM in de ESP32 beperkt is, zijn enkele trucs nodig bij het coderen van de gegevens. bitluni slaat de informatie in het RAM op als gecombineerde YU-, YV- en V-waarden met elk een 4-bits resolutie (**Figuur 20**).

Helaas is met de aanpak van bitluni alleen composiet video-uitgang mogelijk volgens de PAL-standaard, met NTSC werkt het niet. De verhouding tussen de 3,579545 MHz van de NTSC color burst en de sample rate van de DAC is zo ongunstig dat er geen bruikbare color burst kan worden uitgevoerd en ontvangers niet kunnen synchroniseren.

Figuur 21: Emulatoren voor 8-bit spelcomputers. (Bron: tinyurl.com/ykd9ezap)

Kleur met ESP32 in PAL en NTSC

Met zijn project "esp_8_bit" heeft de GitHub gebruiker rossumur bewezen dat het ook mogelijk is om kleur volgens NTSC te maken met een ESP32. Dit project is een verzameling op ESP32 gebaseerde emulaties van verschillende 8-bit consoles, van de Atari 400 tot het Nintendo Entertainment System en het Sega Master System (**Figuur 21**).

Het project gebruikt eenvoudigweg één pin van de ESP32 om een composiet signaal in NTSC of PAL uit te voeren. De truc zit verborgen in de audio PLL van de ESP32. Deze kan worden gebruikt om de DAC te laten werken met sample rates tot ongeveer 20 MHz. Vier maal de frequentie van de NTSC kleurendrager is 14,318182 MHz (voor PAL 17,734475 MHz). Met de APLL kunnen 14,318180 MHz en 17,734476 MHz worden gegenereerd, wat dicht genoeg ligt bij de benodigde frequenties voor PAL en NTSC. Op deze manier geeft de DAC nu een geheel veelvoud van de kleurensubdragerfrequentie uit, wat resulteert in verwerkbare videosignalen.

Met de APLL kan de DAC van de ESP32 niet alleen een videosignaal uitvoeren, deze aanpak is ook zeer interessant voor andere Directe Digitale Synthese (DDS) toepassingen. Het gebruik van de APLL heeft echter een nadeel bij hoge samplesnelheden. De DAC van de ESP32 heeft twee kanalen. Als de APLL één daarvan voorziet van gegevens voor video-uitgang en men probeert het tweede kanaal te voorzien van audiogegevens uit de I2S interface, dan krijgt de DAC last van dropouts. Deze storingen treden op beide kanalen op, zodat je je toevlucht moet nemen tot andere methoden voor audio-uitvoer.

Op basis van het "esp_8_bit" project is "espflix" ontwikkeld. Hiermee kan de ESP32 video's afspelen die zijn opgeslagen op een server van het espflix project. Dit vereist echter enkele concessies aan het videomateriaal. Details hierover zijn te vinden op de bijbehorende Github pagina [14].

Deel 2 vooruitblik

Het tweede deel van het artikel gaat over VGA, DVI en sprites. Vooral met een ESP32 of de RP2040 van de Raspberry Pi Pico zijn hier verbazingwekkende effecten mogelijk. Als je niet wilt wachten tot het tweede deel, kun je je inschrijven voor het webinar "Microcontroller als Pixel Artist" [15], waar de onderwerpen VGA, DVI en sprites ook aan bod komen.

220047-03

Webinar "Microcontroller as Pixel Artist"

Omdat bewegende beelden meer zeggen dan woorden, zeker over dit onderwerp, en uitgebreide opsommingen niet in het artikel pasten, kun je het Elektor webinar voor dit nummer bekijken. Je kunt je al inschrijven op [15]. Het zal gaan over graphics met Arduino, ESP32 en Raspberry Pi Pico.
www.elektormagazine.com/webinars

Vragen of opmerkingen?

Heb je technische vragen of opmerkingen over dit artikel? Stuur een e-mail naar de auteur via mathias.claussen@elektor.com of neem contact op met Elektor via redactie@elektor.com.

WEBLINKS

[1] Composite Video: https://en.wikipedia.org/wiki/Composite_video
[2] Angelo La Spina, "Analogue Video": https://www.elektor.com/analogue-video-e-book
[3] Arduino TVOut Library: https://github.com/Avamander/arduino-tvout
[4] Hackvision: https://nootropicdesign.com/hackvision/
[5] Resistor ladder: https://en.wikipedia.org/wiki/Resistor_ladder
[6] pico-composite8: https://github.com/obstruse/pico-composite8
[7] Hackaday event: https://hackaday.com/2022/08/17/chips-remembered-the-scenix-ubicom-parallax-sx
[8] Color with SX Chips: https://elinux.org/images/e/eb/Howtocolor.pdf
[9] YCbCr color space: https://en.wikipedia.org/wiki/YCbCr
[10] EBU color bars: https://en.wikipedia.org/wiki/YUV
[11] NTSC tint control: https://en.wikipedia.org/wiki/Tint_control
[12] bitluni, "ESP32 Composite Video": https://bitluni.net/esp32-composite-video
[13] Github user rossumur: https://github.com/rossumur
[14] espflix: https://github.com/rossumur/espflix
[15] Webinar "Microcontroller as Pixel Artist": https://www.elektormagazine.com/webinars