Een E-Ink WiFi kleuren-fotolijst

Jeroen Domburg (Espressif)

E-ink displays worden vaak in apparaten toegepast, maar dat zijn dan vooral zwart/wit-exemplaren; de weinige kleuren-displays zijn vrij duur en moeilijk in te passen in een DHZ-project. In dit artikel bespreken we het gebruik van een redelijk geprijsd zeven-kleuren-display dat een goede kleurechtheid biedt – dankzij dithering-technieken – en met de mogelijkheid om verbinding te maken met een WiFi-netwerk met behulp van een ESP32-C3-WROOM-02-module van Espressif.

Een tijdje geleden werd onze kleine geboren. Omdat haar (over)grootouders nogal verspreid over de globe wonen, zijn bezoekjes in levenden lijve niet vaak mogelijk. Natuurlijk is videobellen tegenwoordig een fluitje van een cent zodat we wel contact kunnen houden, maar ik wilde dat iedereen haar ook kon zien als we niet samen online waren. Dus dacht ik: waarom maken we geen fotolijstje met een kleurendisplay en een WiFi-chip? Vooral omdat ze zo snel groeit, zou het leuk zijn om elke dag een nieuwe actuele foto te sturen.

Je hebt vast wel eens een E-Ink display gezien: zo niet in een e-boek-lezer, dan toch zeker in een supermarkt of andere winkel waar ze de oude papieren prijskaartjes hebben vervangen. Ze zijn geweldig voor het weergeven van statische afbeeldingen, omdat ze het meest recente beeld vasthouden zonder dat dit stroom kost. De meeste van deze schermpjes zijn zwart/wit, met soms rood of geel als extra kleuroptie. Leuk, maar een zwart/wit-foto doet geen recht aan de levendige kleuren van een blije baby die met haar kleurrijke speelgoed aan de slag is.

Wat er bestaat

Op het moment van schrijven (begin 2023) beginnen er eindelijk e-boek-lezers met kleurenschermen te verschijnen. De kwaliteit lijkt behoorlijk, met kleuren die die van een krant benaderen. Helaas zijn ze duur, en de displays zelf lijken nog niet beschikbaar te zijn als reserveonderdelen, dus er is geen kans op eenvoudig hergebruik in een doe-het-zelf project.

Er is echter één type E-Ink kleurendisplay beschikbaar voor gebruik in DHZ-projecten. Het lijkt dat deze meestal verkocht worden door Waveshare, een Chinees bedrijf dat zich richt op de fabrikantenmarkt; het meest voorkomende model is een 5,65-inch 640×448-unit met zeven kleuren (**figuur 1**). Met slechts zeven kleuren en een verversingstijd van ongeveer een minuut lijkt het bedoeld als de grote broer van de varianten met een prijskaartje, waarbij statische informatie wordt weergegeven met de kleuren die worden gebruikt om bijvoorbeeld egale achtergrondkleuren te hebben. Ze zijn zeker niet bedoeld om fotorealistische afbeeldingen weer te geven. Dat is natuurlijk jammer: een E-Ink-display kan heel goed een "afgedrukte foto" simuleren, omdat het geen achtergrondverlichting nodig heeft en volledig statisch is. Ik ben niet de enige die er zo over denkt, en heel wat mensen ([1], [2], [3]) hebben al geprobeerd om met dithering de weergave zo op te peppen dat afbeeldingen een beetje natuurgetrouw worden weergegeven. Hier beschrijf ik mijn poging om dit voor elkaar te krijgen.

De hardware

In de eerste plaats moest ik de hardware bouwen. Ik begon mijn ontwerp met een aantal vereisten. Ik wilde dat het fotolijstje één keer per dag via WiFi verbinding zou maken met een server om nieuwe foto's op te halen. Een moest dan worden weergegeven op het E-Ink display. Als het lukte om nieuwe afbeeldingen te downloaden, moest een daarvan worden getoond, anders moest een eerdere afbeelding zichtbaar blijven. Daarna gaat de fotolijst weer 24 uur slapen. De keuze van een geschikte chip was niet moeilijk omdat ik toevallig voor Espressif werk, een bedrijf dat geweldige microcontrollers met WiFi-functionaliteit maakt die ook een deep-sleep modus hebben die behoorlijk goed werkt. Ik heb voor mijn project een ESP32C3 gekozen: die is lekker goedkoop en heeft alle functies die ik nodig heb.

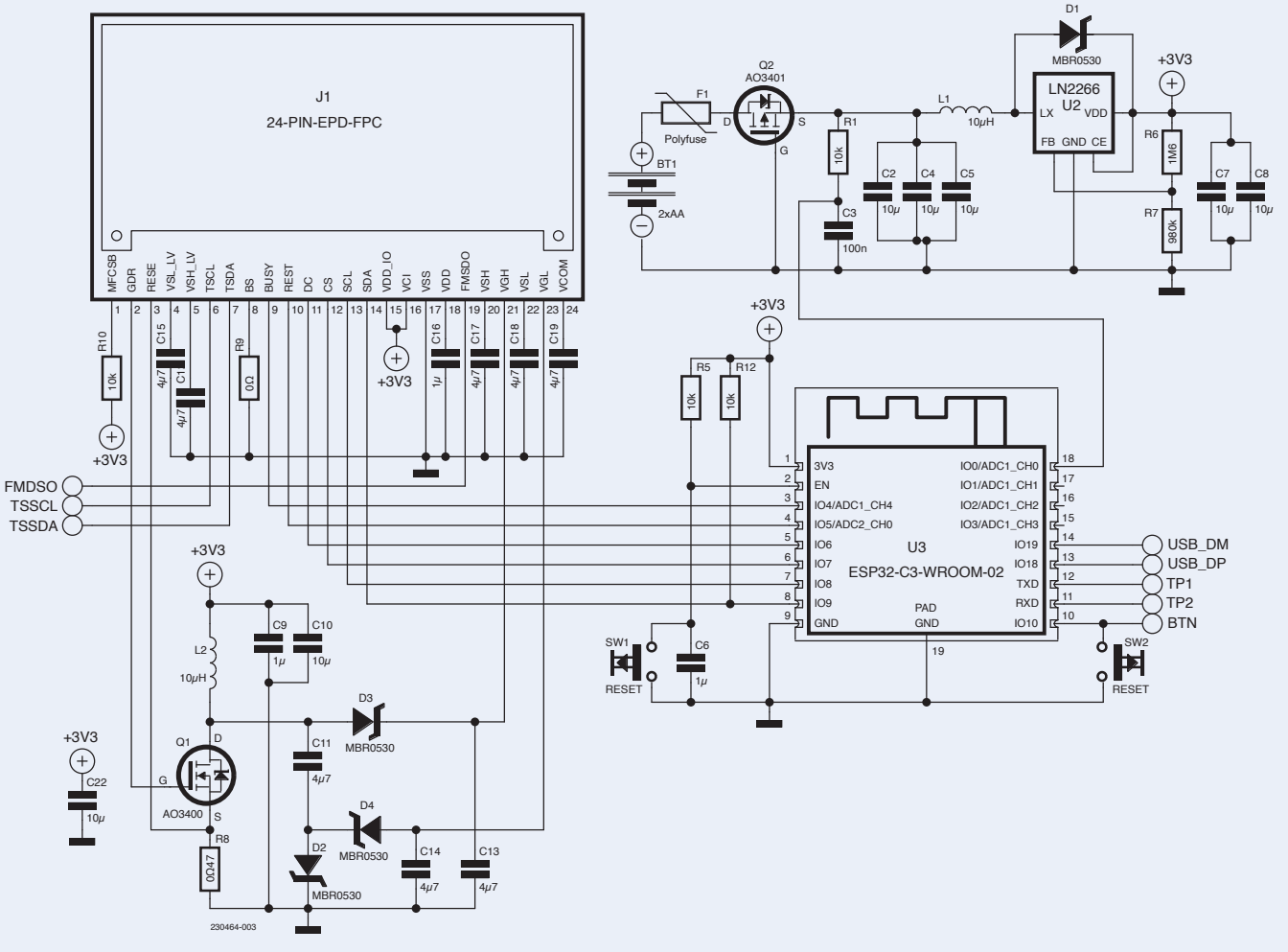
Verder had ik ook een aantal eisen voor de voeding. Ik wilde het display zoveel mogelijk op een normaal fotolijstje laten lijken (afgezien van het feit dat het 's nachts van foto wisselt), dus een externe voeding was uit den boze; de voeding moest een aan boord zijn. Ik wilde het ook internationaal verzenden, dus het zou het beste zijn als het zonder batterij verzonden kon worden. Die eis sloot elke soort oplaadbare Li-ion cel uit. Zoals je kunt zien in het schema van **figuur 2**, heb ik besloten om 2 AA-batterijen te gebruiken; die zijn overal verkrijgbaar en zelfs de grootouders kunnen die moeiteloos vervangen als dat nodig is.

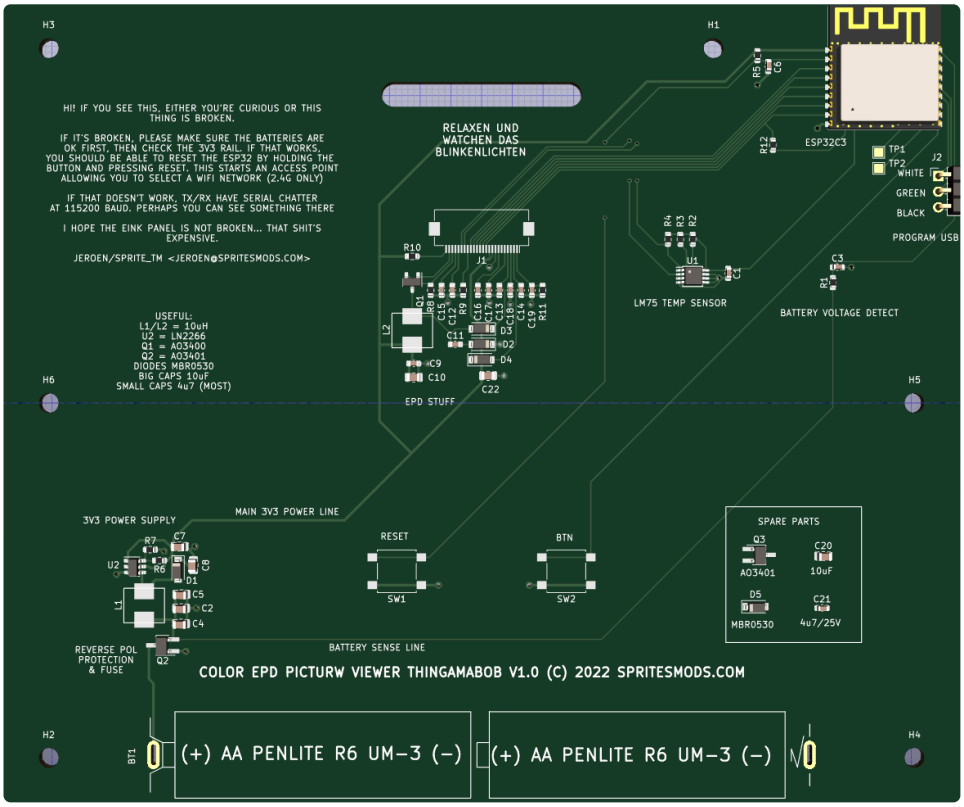
Figuur 1. Het 5,65-inch, zeven kleuren, 640×448-display van Waveshare (bron: Waveshare).

◀

Figuur 2. Het volledige schema van dit project, inclusief de 3,3 V DC/DC-converter, de ESP32-C3-WROOM-02-module en de voeding voor het display.

▼





Figuur 3. Onderdelenopstelling (componentzijde) van de ruim bemeten print voor het project.

▶

Omdat 2 AA-batterijen een totale spanning leveren die begint bij ongeveer 3,2 V en die afneemt tijdens de levensduur, had ik een boostschakeling nodig om die spanning op te voeren tot de 3,3 V die het E-Ink display en de ESP32C3 nodig hadden. Als ik naar de ontlaadgrafieken [4] kijk, zou het boostcircuit moeten werken met een ingangsspanning van ongeveer 2,0 V om bij alkaline-batterijen het onderste uit de kan te halen. Bovendien moet de ruststroom laag genoeg zijn, omdat de ESP32C3 zelfs in deep sleep-modus van stroom moet worden voorzien. Met dit 'boodschappenlijstje' ging ik op zoek naar een bruikbare boost-converter.

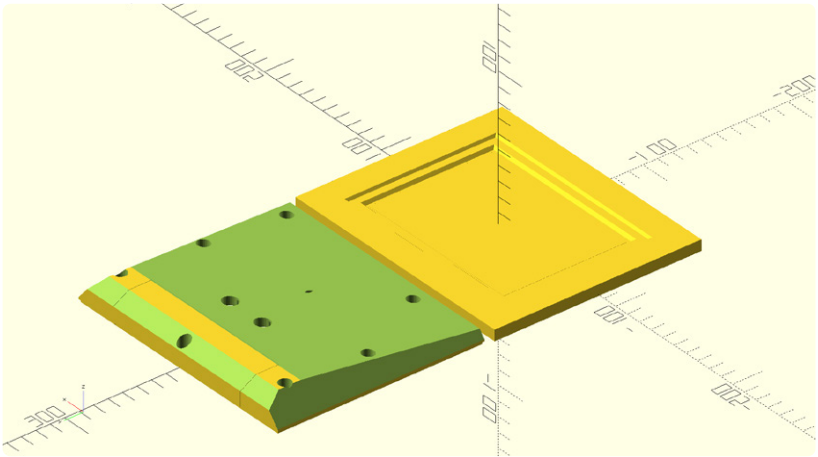
Ik heb gekozen voor de Natlinear LN2266. Deze kleine chip wordt gemaakt door een Chinese fabrikant, zodat hij waarschijnlijk iets minder gevoelig voor de huidige siliciumcrisis zal zijn. Hij werkt vanaf 2 V, heeft een nullaststroom van 56 µA en kan de ongeveer 500 mA aan WiFi-opstartstroom leveren die de ESP32C3 nodig heeft. Ik heb de schakeling zo ontworpen dat de LN2266 zijn stroom uit de twee batterijen haalt via een p-kanaal MOSFET die zo is aangesloten de batterijen te beschermen als ze verkeerdom worden geplaatst. De accuspanning gaat ook via een RC-filter naar een van de ADC-pinnen van de ESP32-C3, zodat deze een idee heeft van hoeveel stroom er nog over is. Aanvankelijk had ik ook een kleine polyfuse in het ontwerp opgenomen, maar die had voor de goede werking een te grote spanningsval nodig. Ik heb hem vervangen door een weerstand van nul ohm in mijn printen, en heb hem helemaal verwijderd uit de ontwerpbestanden.

De ESP32-C3 wordt geleverd in de vorm van een ESP32-C3-WROOM-02-module, rechts te zien in het schema van figuur 2. Deze module bevat de WiFi microprocessor plus 4 MB flash alsmede alle benodigde RF-componenten, inclusief een antenne in de vorm van een printspoor. Om hem te programmeren heb ik een interne header toegevoegd, waar ik een USB-kabel aan kan solderen. De interne USB-JTAG serieelconverter zorgt voor de rest. Ik heb een OTA firmware-update-functie toegevoegd aan de firmware (dus de lijst kan nieuwe firmware downloaden van mijn server); als ik dan units moet updaten die al gesloten en op locatie zijn, kan ik dat doen door de nieuwe firmware remote te pushen.

Dan komen we bij het E-Ink display. Het is verbonden met de print via een platte lintkabel en heeft een aantal dingen nodig om te werken, zoals te zien is linksonder in figuur 2: een MOSFET, een spoel en wat diodes en condensatoren om de benodigde spanningen te genereren, een paar ontkoppelcondensatorenen een weerstand of twee. Het E-Ink display heeft ook een aansluiting voor een externe LM75-temperatuursensor. Ik heb er 'voor het geval dat' een in het ontwerp opgenomen, maar de

Figuur 4. De behuizing, ontworpen met OpenSCAD.

▼



huidige firmware gebruikt hem niet, omdat het display net zo goed werkt zonder die sensor.
Dit alles is gemonteerd op een zeer ruim bemeten print, waarvan de componentenopdruk (componentenzijde) te zien is in **figuur 3**. Aangezien het kale E-Ink display van glas is en ietwat kwetsbaar, heb ik de print zo ontworpen dat hij als een steun fungeert om de fotolijst als geheel minder breekbaar te maken. De print is iets groter dan het display, omdat de montagegaten, de paar through-hole componenten (zoals de batterijcontacten) en de WiFi-antenne niet onder het E-Ink display mogen verdwijnen.

Behuizing

Tot slot is er de behuizing, die ik heb ontworpen met OpenSCAD en die is afgebeeld in **figuur 4**. Met een paar kunstgrepen ziet die er niet zo zwaarlijvig uit – de AA-batterijen moeten er immers in passen, en ik wilde geen grote bult aan de onderkant, dus heb ik wat grote afschuiningen en een schuine achterzijde gebruikt. Die zie je niet als je de fotolijst van voren bekijkt, dus het geheel ziet er een stuk slanker uit. Het fotolijstje leek ook tamelijk fors in vergelijking met het zichtbare deel van het E-Ink display. Om dat te verhelpen, heb ik een kader toegevoegd (ook wel 'passe-partout' geheten). Omdat dit kader wit is en mooi contrasteert met de zwarte behuizing, breekt het het teveel aan zwart en ziet alles er proportioneel uit.
Aan de achterkant zit het batterijcompartiment. Dit wordt geopend door één schroef los te draaien (ik wilde niet vertrouwen op fragiele 3D-geprinte clips om het op zijn plaats te houden) en omdat de opdruk van de print de juiste stand van de batterijen toont, zou het vervangen van batterijen eenvoudig moeten zijn.

Software/firmware en WiFi-verbinding

Aangezien de ESP32C3 wordt gevoed door batterijen, heb ik geprobeerd om hem zo weinig mogelijk te laten doen. Hij moet verbinding maken met WiFi, met mijn server communiceren, kijken of er nieuwe foto's zijn die hij nog niet heeft gedownload en, als dat zo is die ophalen, uitzoeken wat de beste afbeelding is om te tonen (de nieuwste of de minst bekeken foto), die weergeven en afsluiten. Uiteraard zijn er nog andere dingen die gedaan moeten worden, zoals het afhandelen van fouten en de omgang met lege batterijen.
Om verbinding te maken met WiFi heeft het apparaat een SSID en een wachtwoord nodig. Ik wilde dit niet hard coderen, voor het geval het veranderd moest worden. Daarom wordt de firmware geleverd met een kopie van ESP32-WiFi-Manager [5], aangepast om enkele bugs op te lossen. Als je op een van de knoppen op de achterkant van het fotolijstje drukt terwijl je het lijstje reset met de andere knop, wordt er een access point gestart waarmee je verbinding kunt maken. Een embedded webserver levert vervolgens een gebruikersinterface om een nieuwe SSID te kiezen en het wachtwoord ervoor in te stellen.
Nadat de fotolijst verbinding heeft gemaakt met WiFi, probeert hij gegevens op te halen van een vast gecodeerde URL om te zien wat gedaan moet worden. De URL codeert ook enkele statusgegevens, zoals de MAC van het apparaat, de batterijspanning en de huidige firmware-ID. De server stuurt de ID van de meest recente firmware voor dat specifieke apparaat terug, evenals een index van de tien meest recent opgehaalde afbeeldingen. Er worden ook enkele voorkeuren verstuurd, zoals de tijdzone en het tijdstip waarop de fotolijst moet proberen wakker te worden en een update uit te voeren.
De fotolijst heeft eigenlijk plaats voor tien foto's in zijn flashgeheugen; hijn vervangt de oudste door nieuw gedownloade afbeeldingen als de server foto's in de aanbieding heeft die nog niet in het lokale geheugen zitten. Op deze manier is er altijd een voorraad relatief recente foto's, wat handig is als om wat voor reden dan ook de verbinding wegvalt. Het is zelfs mogelijk om de fotolijst naar een andere locatie te verplaatsen: hoewel hij zonder WiFi-vrbinding oude foto's zal hergebruiken, wordt er nog steeds elke dag iets anders weergegeven.
Het grootste deel van de server-side software is niet zo ingewikkeld: er is een eenvoudige front-end webpagina gemaakt rond Cropper.js [6], die je kunt openen op je telefoon of PC. Hiermee kun je een foto selecteren en het gedeelte dat je op het fotolijstje wilt weergeven uitsnijden. Er is een stukje JavaScript dat de afbeelding vervolgens bijsnijdt en opschaalt aan de clientzijde en de resulterende gegevens naar de server stuurt. De server neemt deze over, converteert ze naar ruwe data voor het E-Ink display en slaat ze op in een MariaDB-database.
Wanneer een fotolijst verbinding maakt, slaat de server de door de lijst verzonden gegevens op, zodat ik een logboek van accuspanningen heb en ik kan zien of een firmware-update echt 'gelukt is.' De server controleert vervolgens de MariaDB-database voor de laatste tien afbeeldingen en andere informatie, zoals de laatste firmware-versie, en codeert dat in JSON en retourneert die. Dat is allemaal eigenlijk triviaal.

▲

Figuur 5. De foto die voor testdoeleinden is gebruikt.

Figuur 6. Eerste conversiepoging, met 100% zwart en 100% wit.

Figuur 7. Zwart-wit conversie met error diffusion.

Figuur 8. De foto van figuur 7 met toevoeging van RGB-waarden (zie tekst).

Figuur 9. De voorbeeldfoto met de toepassing van de CIEDE2000-standaard.

Dithering

Alleen het omzetten van de RGB-afbeelding in de zeven vrij specifieke kleuren die het E-Ink diaplay kan weergeven, is werkelijk gecompliceerd. Neem bijvoorbeeld de afbeelding van **figuur 5**.

Als we dit in zwart en wit willen omzetten, kunnen we gewoon de luminantie (lichtheid) voor elke pixel bepalen en, als deze dichter bij zwart ligt, de pixel zwart maken; als deze dichter bij wit ligt, maken we hem wit. Met andere woorden, we nemen de dichtstbijzijnde 'kleur (waarbij we de 'kleuren' beperken tot 100% zwart en 100% wit) en veranderen de afbeelding in het resultaat van **figuur 6**. Dat lijkt in de verste verte niet op het origineel.. Zelfs met zwart/wit kunnen we het beter doen door iets te gebruiken dat we *error diffusion* noemen. In feite nemen we, elke keer dat we een grijze pixel zwart of wit maken, het verschil tussen de luminantie van de pixel in de originele foto en de luminantie van de pixel die we daadwerkelijk op het E-Ink scherm tonen (de 'error' in 'error diffusion') en voegen dit gedeeltelijk toe aan de omliggende pixels (de 'diffusion'). Dit proces kan op verschillende manieren worden uitgevoerd, waarvan Floyd-Steinberg de meest voorkomende is, en het geeft een vrij goed zwart/wit geditherede afbeelding, zoals geïllustreerd in **figuur 7**.

We kunnen deze methode ook gebruiken voor ons display met zeven kleuren. Het probleem is dat de definitie van 'dichtstbijzijnde kleur' een stuk ingewikkelder wordt, net als de definitie van 'toevoegen.' Zelfs de definitie van 'kleur' wordt lastig, omdat een E-Ink display geen achtergrondverlichting heeft en de waargenomen kleuren dus verschillen afhankelijk van de lichtbron: als je het verlicht met een kaars, zie je andere kleuren dan wanneer je het display in fel zonlicht bekijkt.

Om de kleuren te krijgen, nam ik een lamp met regelbare kleurtemperatuur, stelde die in op 4800 K (de gemiddelde kleurtemperatuur waarop ik denk dat het scherm zal worden bekeken), gaf de zeven kleuren weer als egale rechthoeken op het display en nam er een foto van. Ik importeerde deze in mijn computer en paste de kleuren handmatig aan totdat de kleuren op de monitor zo dicht mogelijk in de buurt kwamen van de E-Ink kleuren. Vervolgens nam ik de gemiddelde RGB-waarden van de zeven kleuren en voerde die in mijn programma in.

Voor deze zeven kleuren hebben we een manier nodig om twee kleuren te vergelijken, en omdat we echte kleuren gebruiken en niet alleen zwart en wit, kunnen we er niet

langer mee wegkomen om gewoon de helderheid te gebruiken. Een snelle manier om twee kleuren te vergelijken is door de (gelineariseerde) RGB-ruimte te zien als een driedimensionale ruimte en de rechtlijnige afstand te gebruiken als maat voor hoe dicht twee kleuren bij elkaar liggen. Met dit model kunnen kleuren worden toegevoegd door simpelweg de RGB-waarden bij elkaar op te tellen. Als we de Floyd-Steinberg dithering aanpassen om deze methode voor het kiezen van de dichtstbijzijnde kleur toe te passen, krijgen we een redelijk acceptabele afbeelding (**figuur 8**).

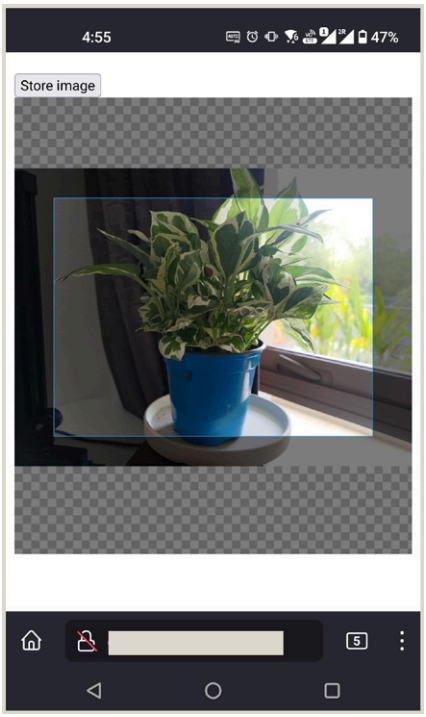
Dat is eigenlijk niet zo slecht! Er zijn echter een paar vreemde kleurafwijkingen. De meest in het oog springende is dat de bloempot niet dezelfde tint blauw heeft: dat komt omdat het E-Ink display simpelweg niet over de juiste kleuren beschikt om die tint te reproduceren, en geen algoritme kan dat compenseren. Maar er is meer vreemds: merkwaardige kleurbanden, bijvoorbeeld in de schaduw onder de linkerarm van de aap, en de buik van de aap is meer oranje dan in de originele foto.

Een andere aanpak: kleurruimte

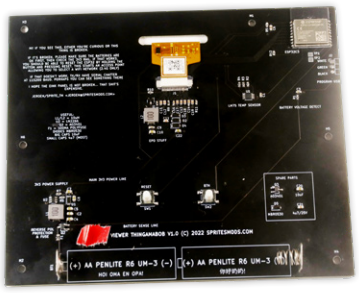
Het probleem met het berekenen van kleurverschillen in RGB-ruimte is dat je ogen eigenlijk niet in een lineaire RGB-ruimte werken. Het verschil tussen twee kleuren is eigenlijk veel moeilijker te definiëren en er zijn hiervoor verschillende benaderingen bedacht. Een van de eerste was om de RGB-kleuren te converteren naar de CIELAB-kleurruimte en het verschil te nemen. Deze aanpak wordt veel aanbevolen op internet, maar het blijkt dat het niet echt goede resultaten geeft voor kleuren die niet volledig verzadigd zijn. Voor mij bleek de beste aanpak het gebruik van de CIEDE2000-standaard [7]. Dit is een van de meer recente modellen, en hoewel het niet triviaal is om te berekenen, geeft het wel de beste resultaten, zoals te zien **figuur 9**. Het is maar goed dat ik al besloten had om dit aan de server-kant te doen, zodat ik de batterijen niet leeg hoefde te trekken met dit zware rekenwerk. Merk op dat de kleurband in de schaduwen weg is en dat de buik van de aap een meer accurate tint rood heeft. Er zijn nog steeds fouten in de afbeelding, zoals de kleur van de bloempot, maar zoals ik al eerder zei, komt dit omdat de E-Ink gewoon niet de kleuren bezit om die specifieke tint goed weer te geven.

Omwille van de snelheid is alle logica geïmplementeerd in een eenvoudig C-programma. Nadat de afbeelding is bijgesneden in JavaScript op de client (met Cropper.js), wordt deze naar de server gestuurd, waar het PHP-script dit C-script aanroept om de afbeelding om te zetten in E-Ink pixels, die worden opgeslagen in een MariaDB-tabel voor de fotolijstjes om op te halen wanneer ze verbinding maken.

De webpagina is ook geschikt voor mobiel gebruik, dus als ik of iemand anders met toegang tot de pagina een bijzonder mooie foto neemt, kunnen we deze meteen bijsnijden en in de wachtrij zetten voor distributie naar alle geïnstalleerde fotolijstjes (**figuur 10**).



Figuur 10. Quick-crop functie voor mobiel gebruik.



Figuur 11. De compleet geassembleerde print.

Figuur 12. De twee delen van de behuizing voor dit project.

Resultaat

Als de print gereed is en de 3D-geprinte onderdelen klaarliggen, is het tijd om alles in elkaar te zetten (**figuur 11**). Alle elektronica is op de achterkant van de print gemonteerd. De print is eigenlijk ontworpen met het oog op repareerbaarheid: als het stuk gaat en ik ben niet in de buurt om het te repareren, ken ik misschien wel een paar mensen in de buurt die goed zijn met elektronica en er even naar kunnen kijken. Dat betekent dat er wat debug-instructies op de achterkant staan en zelfs wat reserveonderdelen voor het geval er iets kapot gaat. Verder is de print niet al te dicht bestukt: de afmetingen worden voornamelijk bepaald door het E-Ink display aan de andere kant. Gezien de beschikbare ruimte had ik grotere componenten kunnen gebruiken, maar ik heb veel rollen met 0603-onderdelen op voorraad, dus daar heb ik het bij gehouden voor het standaardgrut; met behulp van een goede soldeerbout en een binoculaire microscoop kan ik dat allemaal prima met de hand solderen.

Figuur 13. Het batterijvak met zijn eigen klepje.

Figuur 14. De uiteindelijke fotolijst naast het origineel.

De behuizing is te zien in **figuur 12** compleet met wit passe-partout. Die laatste heeft een uitsparing voor het E-Ink display: zelfs als de lijm die het op de print houdt op de een of andere manier zou losraken, wordt het hierdoor op zijn plaats gehouden. De behuizing wordt gesloten met schroeven die in messing bussen worden geschroefd. Die laatste worden op hun plaats gehouden door ze te verhitten met een soldeerbout (met een oude verroeste soldeerstift erin – ik wil geen goede stiften opofferen) en ze op hun plaats te smelten.

Alles zit aan elkaar met een stel M3-schroeven. Het batterijvak heeft een eigen klepje (figuur 13), dus je hoeft niet alles uit elkaar te halen om de batterijen te vervangen. De batterijen gaan volgens mijn berekeningen meer dan een jaar mee. Merk ook op dat er twee knoppen op de achterkant zitten. De ene is direct verbonden met de reset van de ESP32 en de andere met een GPIO voor algemeen gebruik. Je kunt de resetknop gebruiken om het apparaat een handmatige verbindings- en verversingscyclus te laten uitvoeren. Als je de andere knop ingedrukt houdt terwijl de fotolijst wordt gereset, wordt een toegangspunt gestart waarmee je verbinding kunt maken met de fotolijst en de WiFi-verbindingsgegevens opnieuw kunt configureren.

Tot slot kun je in **figuur 14** het eindresultaat bewonderen. Deze foto haalt niet de eerste prijs qua natuurgetrouwheid, maar het feit dat we de lijst elke dag een nieuwe foto kunnen sturen van waar ook tere wereld, maakt dat meer dan goed.

Zoals gebruikelijk voor mij is dit project open source: met een 3D-printer, toegang tot een server en wat handigheid kun je je eigen versie van dit fotolijstje maken. Alle bronnen, print-layouts enzovoort zijn te vinden op GitHub [8].

230464-03

Gerelateerde producten

- **Waveshare 5.65" ACeP 7-Color E-Paper E-Ink Display Module (600×448)**
  www.elektor.nl/19847
- **ESP32-DevKitC-32E**
  www.elektor.nl/20518
- **Dogan Ibrahim, The Complete ESP32 Projects Guide, Elektor 2019**
  www.elektor.nl/18860

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via jeroen@spritesmods.com of naar de redactie van Elektor via redactie@elektor.com.

Over de auteur

Jeroen Domburg is Senior Software en Technical Marketing Manager bij Espressif Systems. Met meer dan 20 jaar embedded ervaring is hij betrokken bij zowel het software- als het hardware-ontwerpproces van de SoC's van Espressif. In zijn vrije tijd knutselt hij ook graag met elektronica om dingen te maken die praktisch bruikbaar zijn, maar ook minder nuttige zaken.

WEBLINKS

[1] GitHub E-Paper zeven-kleuren-fotolijst: https://github.com/robertmoro/7ColorEPaperPhotoFrame
[2] Raspberry Pi E-Paper lijst: https://reddit.com/r/raspberry_pi/comments/10dbhnj/i_built_a_raspberry_pi_epaper_frame_that_shows_me
[3] E-Paper fotoproject op YouTube: https://youtu.be/YawP9RjPcJA
[4] Ontlaadgrafieken: https://powerstream.com/AA-tests.htm
[5] ESP32-WiFi-Manager: https://github.com/tonyp7/esp32-wifi-manager
[6] Cropper.js: https://fengyuanchen.github.io/cropperjs
[7] CIEDE2000-standaard: https://en.wikipedia.org/wiki/Color_difference#CIEDE2000
[8] GitHub-repository: https://github.com/Spritetm/picframe_colepd