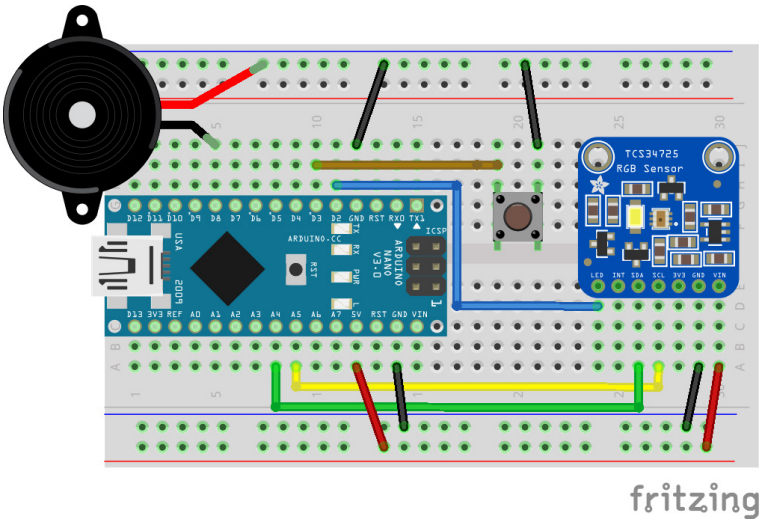
Kleur naar geluid

een kleursensor via I²C uitlezen

Annika Schlechter (Duitsland) en
Volker Ziemann (Zweden)

Elektronische systemen kunnen visueel gehandicapten helpen, zoals wordt gedemonstreerd door dit educatieve project. Voor iemand die geen kleuren kan zien, kan een kleursensor een uitkomst zijn; een zoemer levert akoestische terugkoppeling over kleur en helderheid. Hier beschrijven we twee versies van dit project, op basis van goedkope Arduino Nano's.



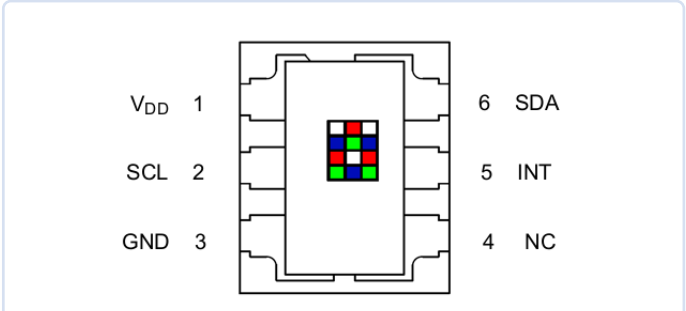
Figuur 1. Schema van het prototype. Een piëzo-buzzer en een TCS34725-kleursensor, gemonteerd op een klein break-out-board, zijn aangesloten op een Arduino Nano.

Hoe kan een visueel gehandicapte persoon 's morgens kleding kiezen met bij elkaar passende kleuren? Het idee achter dit project is om kleurinformatie om te zetten in een geluidssignaal. De kleur wordt vertaald in de toonhoogte van een geluid en de helderheid in de duur. In **figuur 1** ziet u het schema van een prototype met een piëzo-buzzer en een TCS34725-kleursensor op een klein break-out-board [1]. Beide zijn aangesloten op een Arduino Nano. Dit prototype is ontwikkeld als onderdeel van de cursus "From Sensor to Report" aan de Uppsala University [2], een cursus over data-acquisitie en het interfacen van sensoren en microcontrollers met de buitenwereld. Daarom gebruiken we geen kant-en-klare bibliotheken voor de sensor-interface, maar coderen we de basisfuncties voor het uitlezen van de sensoren zelf. Overigens zat de logica voor het toewijzen van klanken aan de kleuren in het begin nog in een Python-script op een laptop, en niet op de microcontroller. De PC en de microcontroller communiceren via een seriële interface. We hebben een communicatieprotocol ontwikkeld waarbij sensordata van de Nano naar de PC gaan en commando's voor het activeren van de buzzer in de andere richting. Later hebben we de functionaliteit uitgebreid, zodat de Nano standalone kan werken. Toch is dit nog geen productierijp systeem, maar een prototype dat de basisfunctionaliteit laat zien en de lezer uitnodigt om verder te experimenteren.

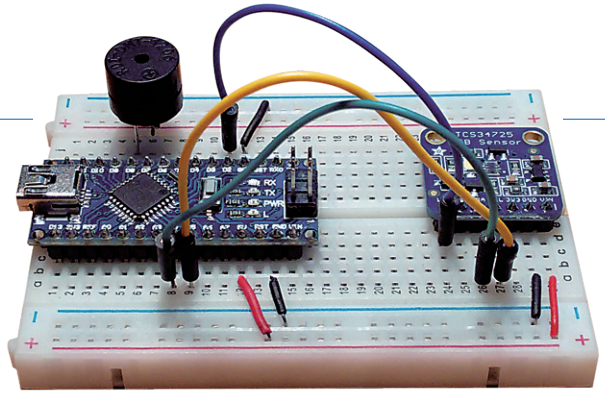
Voordat we de schakeling gaan opbouwen, moeten we eerst het Arduino-ontwikkelsysteem (IDE) voor het ontwikkelen van programma's voor de Nano installeren van [3]. De IDE is beschikbaar voor alle gebruikelijke besturingssystemen. Volg de aanwijzingen voor uw systeem. Als u nog nooit met Arduino's hebt gewerkt, werp dan een blik op [4] om vertrouwd te raken met de Arduino-IDE. Als tweede voorwaarde hebben we Python [5] nodig, ook dat is beschikbaar voor alle systemen. Uitvoerbare bestanden voor Windows en MAC OS zijn op [5] te vinden en vrijwel alle Linux-distributies hebben packages in hun repository. Volg ook hier de aanwijzingen voor uw systeem. Tot zo ver de voorbereidingen – we gaan nu aan de slag!

De kleursensor

Het hart van dit project is de TCS34725-kleursensor [6], die bestaat uit een 3x4-matrix van fotodiodes (**figuur 2**). Kleurfilters voor de diodes maken ze gevoelig naar rood, groen of blauw licht. Verder is er een diode zonder filter die informatie levert over de helderheid. Het analoge signaal van de diodes wordt gedigitaliseerd door A/D-converters in de sensor. De kleurinformatie is beschikbaar via een I²C-interface.



Figuur 2. De TCS34725-kleursensor bevat een 3x4-matrix van fotodiodes. Kleurgevoelige filters voor de diodes maken ze gevoelig naar rood, groen of blauw licht (bron: datasheet/ams [6]).



Figuur 3. De hardware conform figuur 1 wordt opgebouwd op een breadboard (hier zonder drukknop).

GERELATEERDE PRODUCTEN

- **Arduino Nano (SKU 17002)**
  www.elektor.nl/17002
- **Breadboard (830 Tie Points) (SKU 17092)**
  www.elektor.nl/17092

Die maakt de verbinding met een microcontroller via niet meer dan twee communicatielijnen, één voor het kloksignaal (SCL) en één voor data (SDA). De kleursensor wordt dan bestuurd door registers in de sensor te lezen en te schrijven; we komen daar later op terug bij de beschrijving van de software.
De hardware wordt (zonder solderen) opgebouwd op een breadboard (**figuur 3**), min of meer zoals in figuur 1. De Nano zit links en de TCS34725-kleursensor rechts. De voeding voor de sensor komt via de zwarte en rode draden naar massa en +5 V onderaan. Ook de Nano wordt op die manier gevoed. De groene en gele draden verbinden de data- (SDA) en klokpinnen (SCL) met de corresponderende aansluitingen van de Nano (respectievelijk A4 en A5). De blauwe draad verbindt de digitale uitgang D2 met de pen *LED* op de kleursensor. Hiermee wordt een witte LED aangestuurd die ook op het break-out-board zit. De buzzer is verbonden met pin D8 van de Nano. De drukknop die pin D3 met massa verbindt (de bruine draad in figuur 1) start een meting als de Nano in standalone modus werkt. We komen daar nog op terug.
De schakeling wordt tot leven gebracht door een Arduino-sketch die gebruik maakt van de *wire.h*-bibliotheek voor de communicatie via de I²C-lijnen. Die functies worden gebruikt in twee zelfgeschreven functies `I2Cread()` en `I2Cwrite()`, die alle communicatie met de sensor verzorgen.
`I2Cread()` wordt gebruikt om een sensorwaarde of parameter uit te lezen uit een register. De functie krijgt het adres van de sensor op de I²C bus (hier `0x29`) en het registeradres als parameters. Het duurde even voordat we ontdekten dat altijd 0x80 bij dat adres moet worden opgeteld. De functie retourneert de inhoud van het register.
Met de functie `I2Cwrite()` zenden we commando's van de controller naar de sensor, bijvoorbeeld om configuratiewaarden in te stellen. Deze functie krijgt als parameter, naast het adres van de sensor en het register, de nieuwe waarde die naar dat register moet worden geschreven. Deze functie retourneert geen waarde terug (`void`).

De applicatie

We gaan terug naar onze sketch. In de functie `setup()` initialiseren we de seriële communicatie, de I²C-functionaliteit en de sensor (via de functie `color_begin()`). In de functie `loop()`, die continu wordt herhaald, controleren we eerst of er een commando van de PC is aangekomen via de seriële verbinding. Als dat het geval is, lezen we de registerinhoud van de sensor voor alle drie de kleuren. Voor elke kleur moeten we twee 8-bits registers uitlezen en de ontvangen bytes samenvoegen tot een 16bit-woord volgens de instructies in de datasheet. Voor de rode kleur doen we dat als volgt:

```
b3=I2Cread(TCS34725,0x16);   // raw data, red
b4=I2Cread(TCS34725,0x17);
red=b4*256+b3;
```

Hier bevat `b3` het minst significante byte en `b4` het meest significante byte. We vermenigvuldigen dat met 256 en tellen `b3` er bij op om de 16-bits variabele `red` in te vullen. We behandelen de andere kleuren op dezelfde manier maar dan met de bijbehorende registeradressen. Verder lezen we de helderheid van de ongefilterde diode uit en slaan die op in de variabele `clea`. Die bepaalt de tijdsduur `d` van het geluid. Als de waarde van `clea` kleiner is dan 1000, maken we een toon van 0,5 s, als de waarde groter is, duurt de toon 1,5 s.
Dan lezen we het ontvangen commando en kopiëren het resultaat naar het character array `line`, zodat we de standaard C-functie `strstr()` kunnen gebruiken om te bepalen welk commando we hebben gekregen. Dat gaat met het volgende stukje code:

```
Serial.readStringUntil('\n').toCharArray(line,30);
if (strstr(line,"color?")==line) {
  Serial.print(clea); Serial.print(" ");
  Serial.print(red); Serial.print(" ");
  Serial.print(green); Serial.print(" ");
  Serial.println(blue);
} else if (strstr(line,"tone_red")==line) {
  tone(buzz, 262, d);
...
```

Bij het commando *color?* stuurt de Nano de waarden van de vier variabelen `clea`, `red`, `green` en `blue` terug via de seriële verbinding. Als we *tone_red* hebben ontvangen, gebruiken we het ingebouwde commando `tone()` om de piëzo-buzzer te activeren, in dit geval met de frequentie die is toegewezen aan de kleur rood. Daarna volgt een aantal andere `strstr()`-blokken met een vergelijkbare structuur die de bijbehorende acties starten.
We gebruiken dus een heel simpel protocol waarbij tekststrings over en weer worden gestuurd. Een verzoek van de laptop eindigt met een vraagteken, bijvoorbeeld *"color?"*, en de Nano reageert daarop met het terugsturen van de gevraagde gegevens. Een opdracht wordt gegeven door alleen het commando te zenden, bijvoorbeeld *"tone_red"*, wat ervoor zorgt dat de buzzer een geluid geeft van 262 Hz met een tijdsduur gespecificeerd door `d`. De communicatie lijkt een beetje op de SCPI-taal die moderne oscilloscopen en andere test- en meetapparaten ondersteunen. Dit maakt interfacen met externe programma's die commando's van de seriële lijn ondersteunen zeer eenvoudig; Python, Octave, Matlab en LabView ondersteunen dat allemaal.

Het PC-programma

We hebben op de laptop Python 3 gebruikt om te communiceren met de Nano. De uiterst eenvoudige programmacode is te zien in **listing 1**. Na het importeren van ondersteuning voor de seriële communicatie

en de timing-functionaliteit openen we de seriële poort waar de Nano op is aangesloten met de baudrate die op de Nano is ingesteld. We geven het besturingssysteem 3 seconden om de verbinding op te bouwen en dan sturen we het commando "*color?*".
Python 3 codeert strings als Unicode, terwijl de Nano normale ASCII-strings verwacht. Daarom moeten we de letter "b" voor het commando "*color?*" zetten. Als we dat doen, wordt de string als ASCII verzonden. Even later ontvangen we de respons van de Nano in de variabele `reply`. Die bevat dan dus de waarden van de vier kleuren, die we individueel ophalen met de methode `.split()` en toewijzen aan duidelijke namen, zoals `red`.
Nu we de waarden beschikbaar hebben, kunnen we de geluiden gaan toewijzen aan kleuren. We hebben flink moeten experimenteren met gekleurd papier om een goede indeling te vinden. We gebruikten papier met een lichte, middelmatige en donkere tint voor elk van de drie basiskleuren om de sensor te kalibreren. We hebben meermaals de verschillende kleurwaarden gemeten bij een vaste afstand en gebruikten die informatie om de juiste constanten te vinden voor de gegeven Python-code. Het commando *tone_red* wordt naar de Nano gestuurd als de rode component, genormaliseerd voor de helderheid, groter is dan 0,6. Dan testen we de groene intensiteit en sturen *tone_green* als de drempelwaarde wordt overschreden. Tenslotte testen we de waarde van de blauwe component. Als er ook geen blauwe component is, wordt "*no_color*" naar de Nano, gestuurd, waarop die het bijpassende geluid maakt. Het zal zeker nodig zijn om wat te experimenteren met de constanten om het systeem af te stemmen op uw garderobe.

Standalone-versie

De standalone-versie van de code is beschikbaar als download va de projectpagina [7]. Deze lijkt sterk op het bovenstaande voorbeeld, behalve dat nu niet gewacht wordt op commando's via de seriële verbinding. In deze versie wacht de Nano nu tot er op de knop wordt gedrukt, leest dan de sensor uit en speelt het corresponderende geluid af. De kleurinformatie wordt dus rechtstreeks verwerkt op de Nano, zoals in het volgende codefragment:

```
if (red/clea > 0.6) {
  tone(buzz, 262, d);
} else if (green/clea > 0.3) {
...
```

Als u de kleding wilt verlichten met de ingebouwde LED op het breakout-board, kunnen we die doen oplichten met `digitalWrite(led,HIGH)` net voor het lezen van de registers van de TCS34725, en hem vervolgens weer uitschakelen.
Als derde stap kunnen we het systeem draagbaar maken door een kale ATmega328 te gebruiken en die te flashen met een Arduino-bootloader. U kunt het systeem dan voeden uit een accu of batterij zolang een knop is ingedrukt. Het systeem start dan op en converteert voortdurend kleur in geluid totdat de drukknop wordt losgelaten. Die software is ook beschikbaar op de projectpagina bij het artikel. De opbouw van het systeem is een leuke oefening voor enthousiaste lezers.
Bij de cursus "Sensor to Report" was de kleursensor erg populair. Een tweede student, die de kleur van de hemel over een langere tijd wilde monitoren om de

Over de auteurs

Volker Ziemann's interesse in elektronica begon met Elektor, rond de tijd van de 40-W Edwin-versterker (midden jaren '70), maar hij dwaalde af naar een studie natuurkunde en heeft sindsdien gewerkt met deeltjesversnellers bij SLAC in de Verenigde Staten, CERN in Genève en nu in Uppsala in Zweden. Omdat elektronica een belangrijke rol speelt bij het besturen en de data-acquisitie van deeltjesversnellers, heeft die vroege interesse hem tijdens zijn hele carrière geholpen. Hij doceert nu aan de Uppsala University en heeft drie boeken geschreven, één daarvan over data-acquisitie met Arduino-boards en de Raspberry Pi.
Annika Schlechter studeert natuurkunde aan de universiteit van Heidelberg in Duitsland. Op dit moment werkt ze aan haar kandidaatsscriptie bij het Duitse instituut voor kankeronderzoek in Heidelberg. Dit project begon tijdens een Erasmus-semester aan de University of Uppsala in Zweden, waar ze Volker's colleges over "Sensor to Report" volgde.

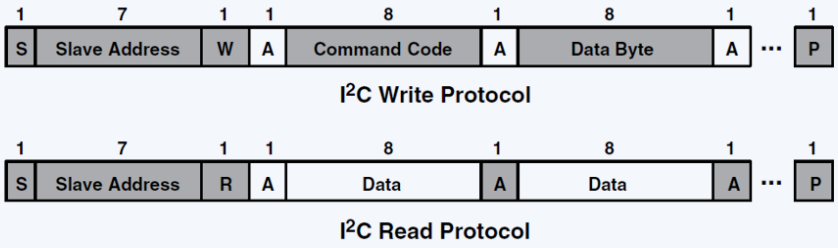
Listing 1. Via dit Python-script communiceert de laptop met de Arduino Nano.

```
# colortosound_arduino.py
import serial, time

ser=serial.Serial("/dev/ttyUSB0",9600,timeout=1)
time.sleep(3);

ser.write(b"color?\n")  # write command to get colors
time.sleep(2);

reply=ser.readlines()   # read answer

# process answer to get color values
colors = reply[0].split()
for i, entry in enumerate(colors):
    colors[i]= int(entry.decode('utf-8').replace('\r\n', ''))

intensity = colors [0]; red = colors [1]
green = colors [2]; blue = colors [3]

# write command to control buzzer according to color

if red/intensity > 0.6:
    ser.write(b"tone_red\n")
elif green/intensity > 0.3:
    ser.write(b"tone_green\n")
elif blue/intensity > 0.24:
    ser.write(b"tone_blue\n")
else:
    ser.write(b"no_color\n")

ser.close()
```

I²C-communicatie

De communicatie tussen twee geïntegreerde schakelingen (vandaar I²C) is gebaseerd op een synchroon serieel protocol dat via twee lijnen verloopt. Het volgt de eenvoudige regel dat het spanningsniveau op een van de lijnen (SDA) wordt gedetecteerd nadat de spanning op de andere lijn (SCL) is veranderd. Dat levert een enkele bit op en acht opeenvolgende bits vormen een byte. Een van de deelnemers op de I²C-bus, meestal een microcontroller, orkestreert de communicatie en levert altijd het kloksignaal. De andere deelnemers hebben een kans om te gaan communiceren als ze worden aangesproken en gemachtigd worden om hun informatie op SDA te zetten. Bovendien heeft de sensor na elke byte één klokcyclus beschikbaar om de ontvangst van de vorige byte te bevestigen. Om de communicatie met een sensor te starten, stuurt de microcontroller één byte. De eerste zeven bits van die byte bevatten het adres en het laatste bit geeft aan of de controller of de sensor de SDA ter beschikking krijgt om het volgende byte te verzenden. Als de controller wil doorgaan met zenden, bijvoorbeeld om een register van de sensor te wijzigen, stuurt hij twee extra bytes, het registeradres en de nieuwe waarde. Als de controller daarentegen een register van de sensor wil lezen, verzendt hij een bit om aan te geven dat de sensor nu de baas is over SDA, en leest hij SDA synchroon met het kloksignaal dat hijzelf blijft genereren.

Het bovenste diagram in de figuur illustreert het schrijven naar de sensor. De gearceerde bits worden bestuurd door de microcontroller en de niet-gearceerde bits door de sensor. Om een transactie te starten, creëert de microcontroller een startvoorwaarde, aangeduid met 'S', gevolgd door de zeven bits van het adres van de sensor en het achtste bit dat aangeeft dat de controller wil doorgaan met verzenden. De sensor verstuurt het bit 'A' om aan te geven dat hij het heeft begrepen. Het tweede byte bevat het adres van het register van de sensor, gevolgd door een byte met de waarde die in het register moet worden geschreven. Let op het bit met het label 'A' waarmee de sensor elk ontvangen byte moet bevestigen. Het lezen van de sensor, weergegeven door het onderste diagram in de figuur, werkt ongeveer hetzelfde, alleen het achtste bit van het eerste byte geeft nu aan dat de sensor het recht krijgt om te 'spreken'; SDA wordt dan door de sensor aangestuurd terwijl de microcontroller elk ontvangen byte moet bevestigen.



Bron: TCS34725 datasheet/ams [6].

gegevens later te analyseren, gebruikte in essentie dezelfde opbouw waarbij de Nano die communiceert met een host-computer via een seriële verbinding. Ze gebruikte een Python-script op een Raspberry Pi om de data op te slaan in een MySQL-database, die ook op de Pi draaide, en gebruikte later Octave voor het maken van diagrammen van de kleuren die in de loop van een week waren geregistreerd. De gegevens werden kort voor Kerstmis geregistreerd en maakte ons pijnlijk duidelijk hoe kort de dagen kunnen zijn in Uppsala. Meer data-acquisitieprojecten, met een vergelijkbare opzet, worden in [8] beschreven.

210051-03

Vragen of opmerkingen?

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

Een bijdrage van

Ontwerp en tekst: **Annika Schlechter, Volker Ziemann**
redactie: **Jens Nickel**
Vertaling: **Evelien Snel**
Layout: **Sylvia Sopamena**

WEBLINKS

[1] TCS34725 breakout-board: https://www.adafruit.com/product/1334
[2] "Sensor to Report"-cursus: https://ziemann.web.cern.ch/ziemann/teaching/s2r19/
[3] Arduino-website: https://www.arduino.cc/
[4] Arduino-tutorials: https://www.arduino.cc/en/Tutorial/HomePage
[5] Python-website: https://www.python.org/
[6] Datasheet TCS34725-kleursensor: https://ams.com/tcs34725#tab/documents
[7] Projectpagina bij dit artikel: http://www.elektormagazine.nl/210051-03
[8] V. Ziemann, A Hands-on Course on Sensors using the Arduino and Raspberry Pi, CRC Press, 2018.