RGB-stroboscoop met Arduino

kleurige toepassing van een nuttig instrument

Roel Arits (Nederland)

In vroeger dagen, voordat moderne microcontroller-elektronica zijn intrede onder de motorkap deed, gebruikte men een stroboscoop om de ontsteking van de motor af te stellen. Nu is dat zelden meer nodig – bij een niet al te antieke auto wordt de ontsteking door de ECU (Electronic Control Unit) geregeld onder gebruikmaking van tal van sensoren.

Vroeger was alles anders – niet noodzakelijkerwijs beter, maar toch. In een nog niet eens zo grijs verleden hadden we onder de motorkap van een auto ruimte genoeg om daar zelf te kunnen knutselen, repareren en afstellen. Dankzij moderne computerondersteunde ontwerptechnieken is het mogelijk alle componenten onder de motorkap zo compact en dicht op elkaar te monteren dat zelfs het vervangen van een ordinaire lamp een hels 'genoegen' is. En natuurlijk heeft de computer zelf (in de vorm van een groot aantal onderling verbonden microcontrollers) ook een plaatsje onder diezelfde motorkap gevonden.

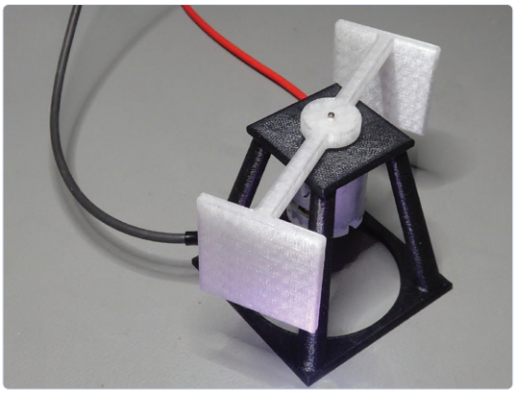
Maar terug naar – laten we zeggen – de jaren '70 of '80 van de vorige eeuw. Voor het goede functioneren van een verbrandingsmotor (benzinestoker) moeten de bougies van elke cilinder op precies het juiste moment een vonk produceren. Daarvoor zorgde de verdeler (een mechanische contraptie die bij bepaalde merken akelig vochtgevoelig was), die dan natuurlijk precies met de motor gesynchroniseerd moest worden. En daar komt dan eindelijk onze stroboscoop om de hoek kijken – of afstellamp zoals die toen ook wel genoemd werd.

In zo'n verdeler bevonden zich evenveel contacten (de onderbrekercontacten) als de motor cilinders rijk was; telkens wanneer zo'n onderbrekercontact geopend werd (vandaar de naam) ontlaadde de in de bobine (een spoel) opgeslagen elektrische energie zich via de bougie in een vonk die het benzine/lucht-mengsel deed ontbranden. En dan ging het vooruit...

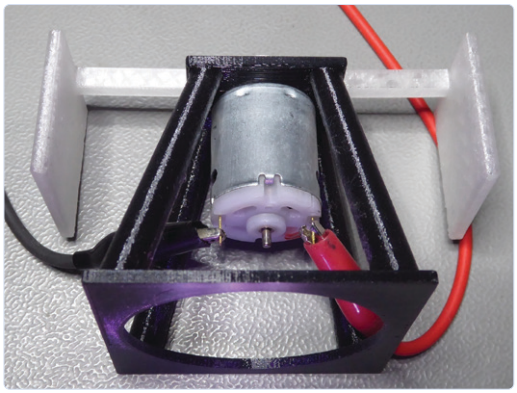
De stroboscoop die voor dit afstellen gebruikt werd, had een xenon- of neon-flitsbuis en een triggerkabel. Die werd op de referentie-bougiekabel aangesloten (met een inductieve opnemer); telkens als de referentie-bougie vonkte, werd de flitsbuis ontstoken en leverde deze een heel korte lichtflits. Op de krukasriemschijf was een wit merkteken (streepje) aangebracht; en op een soort hoekijzer daarboven was een tweede (vast) merkteken aangebracht. Wanneer de stroboscoop door de referentiebougie werd getriggerd moesten beide merktekens ten gevolge van het stroboscoopeffect precies tegenover elkaar stil lijken te staan. Als de streep op de bewegende krukasschijf niet tegenover maar links of rechts van de vaste streep stond, was dat een teken dat de verdeler bijgesteld moest worden. In de boven beschreven toepassing werd de stroboscoop gesynchroniseerd met de rotatie die we willen meten door een triggersignaal. Maar het kan ook zonder zo'n triggersignaal!

Triggeren? Nergens voor nodig...

Wanneer we met een niet-gesynchroniseerde stroboscoop een toerental willen meten, moeten we de frequentie van de lichtflitsen zodanig bijregelen dat het merkteken op de schijf (nagenoeg) stil lijkt te staan, en er maar één merkje zichtbaar is. Wanneer er meerdere (nagenoeg) stilstaande merkjes te zien zijn, dan wil dat zeggen dat de flitsfrequentie een veelvoud is van het toerental of dat het toerental een veelvoud is van de flitsfrequentie.



Figuur 1. Het frame met de propeller die als reflector dienst doet.



Figuur 2. Close-up van de motor.

Als het wiel rechtsom draait en het merkje langzaam rechtsom lijkt te bewegen, dan is de flitsfrequentie iets te laag. De flits komt te laat zodat het merkje in de draairichting lijkt te bewegen. En als het wiel rechtsom draait en het merkje langzaam linksom lijkt te bewegen, dan is de flitsfrequentie iets te hoog: elke flits komt iets te vroeg.
Op deze manier kan het toerental worden gemeten door de flitsfrequentie van de stroboscoop zodanig bij te regelen dat er één stilstaand merktekentje duidelijk zichtbaar is. De duur van elke flits moet kort genoeg zijn om een duidelijke reflectie te verkrijgen (dus een duidelijk zichtbaar merkteken); als de flitsduur te lang is ten opzichte van de flitsfrequentie wordt de reflectie 'uitgesmeerd.'

Nu wordt het leuk

Wat gebeurt er wanneer we meerdere flitsen met dezelfde frequentie genereren, maar dan met verschillende kleuren en met een faseverschuiving tussen de afzonderlijke flitsen? En als we dan ook nog een roterend wit object gebruiken en frequentie, faseverschuiving en breedte van de verschillend gekleurde flitsen variëren?
Zoiets is niet moeilijk te realiseren met behulp van een microcontroller, dus niets weerhoudt ons om aan de slag te gaan. Voor de verschillend gekleurde flitsen kunnen we een RGB-LED gebruiken – beter nog meteen drie stuks om helderder flitsen te verkrijgen. Voor het regelen van frequentie, faseverschuiving en flitsduur gebruiken we een Arduino Pro Mini omdat die genoeg I/O heeft om mee te spelen, en voor ons doel meer dan snel genoeg is. Die LED's worden – hoe kan het ook anders – met pulsbreedte-gemoduleerde signalen (PWM) aangestuurd. We hadden natuurlijk drie speciale PWM-modules van de Arduino Pro Mini kunnen gebruiken, maar deze modules gebruiken drie verschillende timers. Dat maakt het moeilijker om ze te synchroniseren of faseverschuivingen te programmeren. Bovendien hebben we relatief laagfrequente PWM-signalen nodig en een resolutie van 16 bit is overdreven veel. Voor ons doel zijn softwarematig gegenereerde PWM-signalen (softPWM) meer dan goed genoeg. Met softPWM worden de signalen op normale digitale uitgangen uitgevoerd; we setten en resetten deze aan de hand van een teller; telkens wanneer deze een bepaalde stand bereikt wordt een interrupt gegenereerd. Op deze manier wordt een interval ingesteld dat kort genoeg is om de pulsbreedte of faseverschuiving van het PWM-signaal met voldoende nauwkeurigheid bij te regelen.

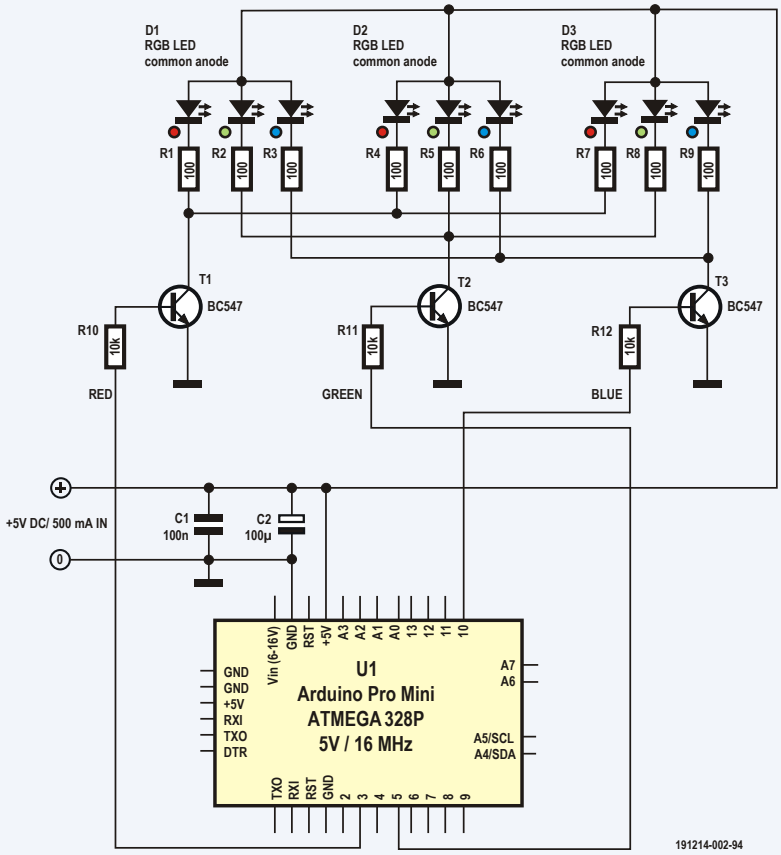
De praktijk

Om het bovenstaande te realiseren, heeft de auteur een 12V-gelijkstroommotortje gebruikt (Velleman MOT3N) en daarvoor een klein frame ontworpen. Dat motortje brengt een 'propeller' met twee verticale vlakken in beweging; die vlakken dienen als reflector. Beide delen zijn met een 3D-printer vervaardigd (de betreffende bestanden maken deel uit van de download bij dit artikel). In **figuur 1** en **figuur 2** ziet u wat de bedoeling is.
Let op: het door de auteur gebruikte motortje draait (onbelast) met zo'n 11500 rpm; dat is veel te veel voor de 3D-geprinte propeller. Bij een eerste experiment vlogen de auteur letterlijk de stukken om de oren. Dat is niet geheel ongevaarlijk; het is daarom geen gek idee om een veiligheidsbril te dragen. U bent gewaarschuwd!
De motor wordt daarom aangestuurd met een gelijkspanning van ongeveer 3 V; deze spanning is regelbaar om de motorsnelheid met de flitsfrequentie te kunnen synchroniseren. Met 3 V bedraagt het toerental ongeveer 2100 rpm, corresponderend met 35 Hz. Omdat de propeller symmetrisch is geconstrueerd met twee reflecterende vlakken (dat maakt het uitbalanceren wat eenvoudiger), bedraagt de flitsfrequentie het dubbele, dus 70 Hz (de periodeduur bedraagt dan ongeveer 14 ms).
Maar nu eerst de elektronica: het schema is getekend in **figuur 3** en is eigenlijk de eenvoud zelve. De drie RGB-LED's worden via bescheiden drivertransistoren op commando van de Arduino Pro Mini in- en uitgeschakeld. De voedingsspanning voor het geheel bedraagt 5 V_{DC}, bij een stroomopname van ongeveer 500 mA.
Voor de schakeling is geen print ontworpen, op breadboard zit de boel in een vloek en een zucht in elkaar (**figuur 4** toont de opstelling van de auteur).

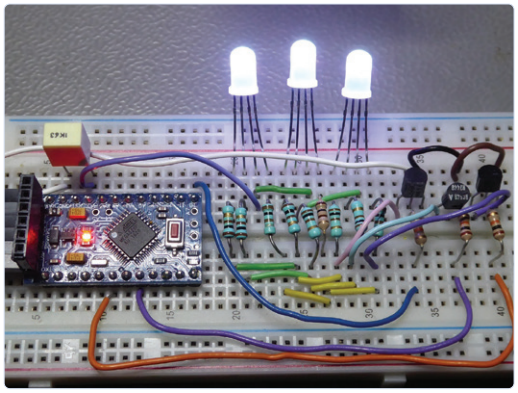
De timing van de stroboscoop wordt geregeld op basis van een timer-interrupt die elke 100 µs wordt aangeroepen. Daarmee is dit interval van 0,1 ms de resolutie waarmee de flitsfrequentie kan worden gevarieerd; dit biedt voldoende regelmogelijkheden om het 'stilstaande' beeld van de propeller te verschuiven.

```
...
// Setup 16bit timer1 in normal operation with
   interrupt at 100us
// 16MHz/1 = 6.25ns. So to get 100us we need to let
   the timer count 1600 ticks.
TCCR1A = 0;
TCCR1B = _BV(CS10); //prescaler divide by 1
TCNT1 = 0xFFFF - 1600; // overflow is at 65535 =
   0xFFFF
TIMSK1 = _BV(TOIE1); // overflow interrupt
TCNT1 = 0;
sei();
}
ISR (TIMER1_OVF_vect)
{
TCNT1 = 0xFFFF - 1600; //100us interrupt
...
```

De flitsperiode is 'hard' in de software gecodeerd en is ingesteld op 144 – dat is het aantal timer-interrupts tussen twee opeenvolgende flitsen; deze periode bedraagt dus 144 · 100 µs = 14,4 ms. De flitsduur voor elke LED is vast ingesteld op 8, overeenkomend met



Figuur 3. Het schema is niet bijzonder ingewikkeld.



Figuur 4. De opbouw is niet kritisch; voor eerste experimenten is een breadboard prima geschikt.

8 · 100 µs = 800 µs. De duty cycle (puls/pauze-verhouding) bedraagt zodoende 800 µs / 14,4 ms = 5,5%.

```
#define STROBE_PERIOD 144

TSoftPwm Pwm[] = {TSoftPwm(ID_RED, 0, 8,
   STROBE_PERIOD),
TSoftPwm(ID_GREEN, 0, 8, STROBE_PERIOD),
TSoftPwm(ID_BLUE, 0, 8, STROBE_PERIOD)};
```

Met deze duty cycle krijgen we (zodra het toerental van de motor optimaal is ingesteld) een mooi 'strak' beeld. De schakeling plus software (die gratis kan worden gedownload van de projectpagina bij dit artikel [4]) is geen 'afgewerkt' project. De auteur bedoelde het veeleer als 'proof of concept' en heeft die daarom niet voorzien van een fraaie gebruikersinterface en dergelijke. De software die beschikbaar is, demonstreert de mogelijkheden door met de parameters flitsfrequentie, flitsduur en faseverschil te spelen. Omdat de software van veel en duidelijk commentaar is voorzien, gaan we er hier niet verder op in.
Natuurlijk is een niet-gesyncroniseerde stroboscoop niet echt optimaal. In principe zou het niet zo moeilijk moeten zijn om bij de propeller een lichtsluisje te monteren waarvan het signaal als trigger voor de stroboscoop kan dienen. Dit laten we echter met een gerust hart aan de geïnteresseerde lezer over.
De auteur heeft twee video's op Youtube gezet waar u de werking van de stroboscoop kunt bewonderen [1] en de onderlinge relatie van de LED-stuursignalen op het scherm van een oscilloscoop kunt bekijken [2]. Het project is ook op Elektor Labs te vinden [3].

191214-01

Related Products

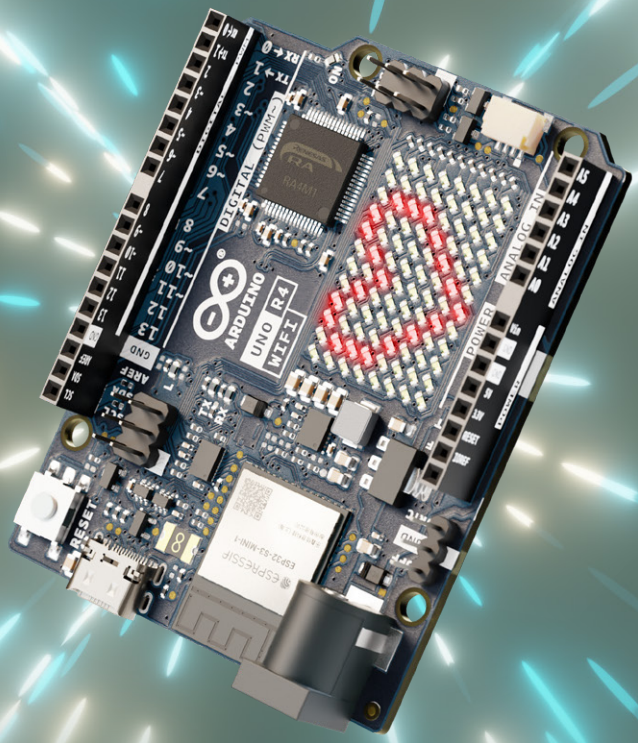
- **50 Mini microcontroller projecten met ATtiny en Arduino**
  www.elektor.nl/50-mini-microcontroller-projecten-met-attiny-en-arduino
- **Arduino – projecten voor gevorderden**
  www.elektor.nl/arduino-nl
- **Home Automation Projects with Arduino**
  www.elektor.nl/home-automation-projects-with-arduino

WEBLINKS

[1] Demo-video 1: https://youtu.be/WHv6DCWM82k
[2] Demo-video 2: https://youtu.be/3tYqUin4-vQ
[3] Projectpagina op Elektor Labs: www.elektormagazine.com/labs/arduino-rgb-color-stroboscope
[4] Projectpagina bij dit artikel: www.elektormagazine.nl/191214-01

ARDUINO

UNO R4



The new dimension of making

Take a quantum leap with the new UNO R4, the most advanced and powerful UNO ever!

Faster and larger memory - 48 MHz, 256 kB Flash, 32 kB RAM - perfect to perform more precise calculations and handle more complex projects.

New built-in peripherals - The RA4M1 (Arm® Cortex®-M4 Core with FPU) includes a DAC, CAN-BUS, HID support and OpAMP embedded allowing for more advanced projects.

Expand your projects with UNO shields (5 V compatible) and the huge catalog of *Qwiic nodes.

Connectivity *ESP32-S3 coprocessor - Wi-Fi® and Bluetooth® Low Energy connectivity, leaving the RA4M1 microcontroller free.

Robust by design - with a larger voltage range, 6-24 V, now you can use the UNO R4 in combination with motors, LED strips or other actuators from a single power supply.

UNO R4 WiFi

UNO R4 Minima

*Note - these features are only available on the UNO R4 WiFi model.