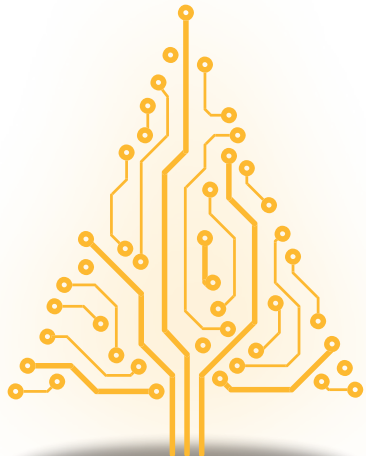
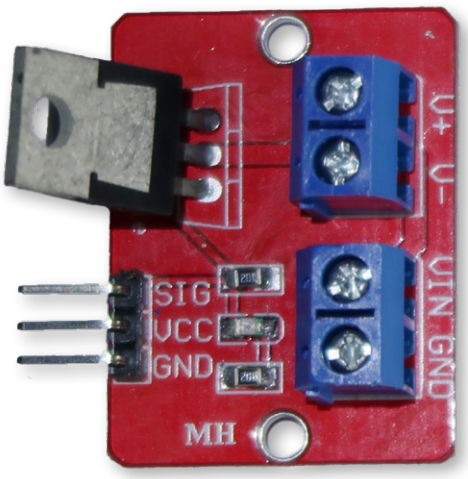


LED-slingers met ESP32 en FreeRTOS

knipperen dat het een lieve lust is

Serge Sussel

Voor de feestdagen kunnen 24-VDC LED-slingers die knipperen en variëren in helderheid een lust voor het oog zijn. Je hebt niet meer dan een ESP32 nodig om een heel systeem met 13 verschillende parameters aan te sturen. Met FreeRTOS kunnen meerdere programmataken tegelijk worden uitgevoerd.



Figuur 1. FET-module.

Voor de bekende feestelijkheden aan het eind van het jaar wilde ik de kerstboom versieren en verlichten. Ik had nog een heel oude kerstslinger liggen, met gloeilampjes in serie die elk in kleine gekleurde plastic lantaarntjes waren gemonteerd. En om deze slinger tot leven te brengen, was een primitieve thermische schakelaar tussen de in serie geschakelde lampjes en het stopcontact opgenomen. Bovendien was het snoer vrij kort, zodat de versiermogelijkheden tamelijk beperkt waren.

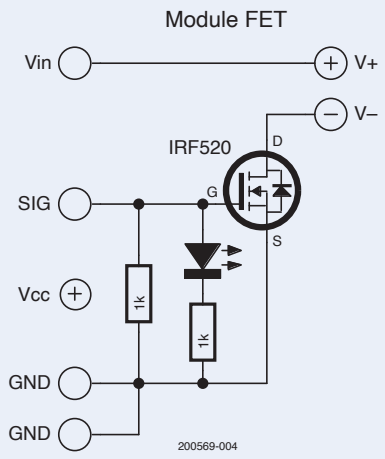
Dus zocht ik op het web naar slingers met LED's, die minder stroom verbruiken en niet zo warm worden. Ik vond er een (Lumitronix) en bestelde twee veel langere gekleurde LED-slingers met passende transformator. Het is een 24V-systeem met een gelijkrichter dat meerdere LED-strings van stroom kan voorzien. De gekleurde slingers blijven echter de hele tijd ingeschakeld zonder enig lichteffect – voor een kerstslinger een beetje triest.

Ik heb ook mijn oude slinger gemoderniseerd door de gloeilampen te verwijderen en te vervangen door LED's met dezelfde kleur als het plastic lantaarntje, en de bedrading tussen de LED's verlengd; de voedingsspanning is nu 24 V, waarbij de LED's een serieweerstand voor de stroombegrenzing in de connector hebben. Ik had nu drie kerstslingers ter beschikking.

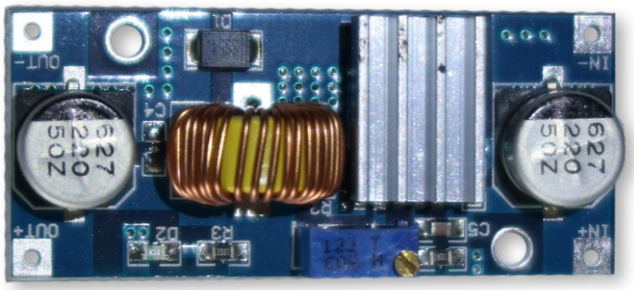
Om het geheel wat leven in te blazen, wilde ik een microcontroller programmeren en via een MOSFET-interface de slingers zo aansturen dat ze zouden knipperen of de helderheid variëren. Daartoe heb ik boeken tover C/C++ en over de Arduino gelezen – want het is altijd spannend iets nieuws te leren.

Als eerste vroeg ik me af hoe ik meerdere PWM-uitgangen tegelijk zou kunnen gebruiken, maar dan wel asynchroon, om zo voor elke slinger een andere effect te hebben. Ik heb overwogen om hiervoor een finite-state machine te ontwikkelen, maar dat vond ik een beetje te ingewikkeld.

Daarom begon ik het project met een Arduino Nano en een eenvoudig programma dat een enkele PWM-uitgang aanstuurde waarop de MOSFET-interface was aangesloten. Daarna heb ik dit gedupliceerd om twee slingers aan te sturen.



Figuur 2. Schema van de FET-module.



Figuur 3. DC/DC-converter met 5V-uitgangsspanning.

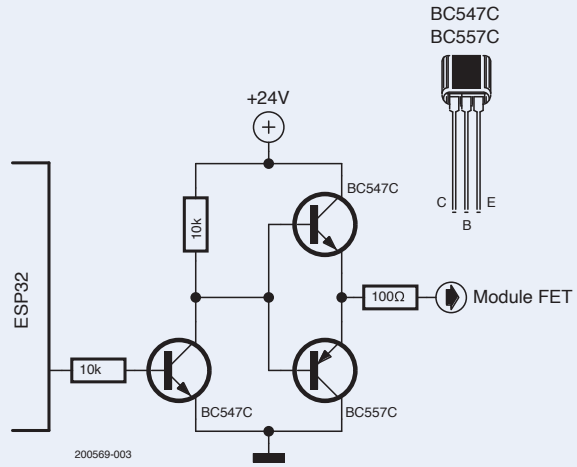
Omdat ik drie strings had, vertoonde de derde dezelfde effecten als een van de andere twee. Tijdens het testen van het project heb ik ook verschillende keren de animatie-instellingen van het programma aangepast om de visuele esthetiek te verbeteren.

Het ESP32-project

Tijdens het ontdekken en lezen van de artikelen van Warren Gay over FreeRTOS in Elektor [1], en ook na de aanschaf van zijn boek over dit onderwerp (zie kader **Gerelateerde producten**), probeerde ik FreeRTOS aan te passen aan de Arduino Nano, maar ik ontdekte al snel de beperkingen ervan. Na diverse pogingen kon ik niet meer dan één RTOS-taak tegelijk uitvoeren op de Nano. Dat moest beter kunnen.
Dus zocht ik mijn toevlucht bij de ESP32 om mijn project naar dit platform te porten, en om FreeRTOS en onafhankelijke asynchrone taken te implementeren. Ik begon met het schrijven van één taak , en ontdekte daarbij de verschillen tussen deze microcontroller (ESP32) en de Arduino Nano bij het programmeren in C/C++.
Na een aantal programmeer- en debugsessies had ik geen compilatiefouten meer, en de taak werkte. Met mijn oscilloscoop op de PWM-uitgang zag ik de signalen die ik wilde. Met FreeRTOS is elke taak onafhankelijk van andere taken, als we geen wacht-en-synchroniseer mechanisme gebruiken. En dat is precies wat ik wilde hebben. Elke slinger kan dus zijn eigen (willekeurige) effect hebben; twee identieke effecten op hetzelfde moment zal niet vaak voorkomen, en zelfs als dat gebeurt, zullen ze niet op hetzelfde moment beginnen door vertragingen in eerdere effecten.
Ik kon het vermogen van de ESP32 meten en na verschillende proeven en tests lukte het me om twee, dan drie, en tenslotte vier taken te schrijven die gelijktijdig draaiden. Dus gebruikte ik FreeRTOS op de ESP32 om vier uitgangen aan te sturen via vier taken. De ESP32 kan echter nog meer PWM-uitgangen aansturen.
Ik gebruik de PWM-uitgangen zodat ik de helderheid van de LED-slingers geleidelijk kan variëren, en ze ook in en uit kan schakelen – naast andere effecten. Ik heb mijn project op gaatjesprint opgebouwd en het geheel in een plastic doos ondergebracht. Ik heb dit project ook op Elektor Labs geplaatst [2].

Vereiste onderdelen

Om de componenten voor mijn project te vinden, ging ik op het internet in het Verre Oosten op zoek en vond allerlei kleine modules met gemonteerde MOSFET's, weerstanden en LED's. De levertijden beliepen echter weken in plaats van dagen. Daarom heb ik enige reverse engineering toegepast (zie **figuur 1** en het schema van **figuur 2**).
Zo vond ik ook een volledig geassembleerde en regelbare voedingsmodule die 24 V DC op zijn ingang accepteert en die, na het instellen van de multiturn-potentiometer de juiste uitgangsspanning (5 V) leverde voor de microcontroller (**figuur 3**).
In een artikel in Elektor vond ik ook een interessante schakeling om de MOSFET-module aan te sturen, met drie transistoren (twee stuks



Figuur 4. Driver voor MOSFET.

Listing 1. 13 parametersets voor de effecten.

```
#define MAXPROGR 13   // Max value of LED
                      // programs (0 to MAXPROGR -1)

// struct for LED programs
 struct  pled_t    /* structure def */
 {
     int ledvar;    /* 0: flashing ; 1: light
                                        varying */
     int ledon;     /* time ON */
     int ledoff;    /* time OFF */
     int ledvaloff; /* value for OFF status */
     int ledxtime;  /* repeat this program x times */
 };
pled_t pled [MAXPROGR];  /* reserve memory for the
                  parameters for the LED programs */


...


// init for the LED programs' parameters

     pled[0].ledvar = 0;
     pled[0].ledon = 750;
     pled[0].ledoff = 900;
     pled[0].ledvaloff = 18;
     pled[0].ledxtime = 7;

     pled[1].ledvar = 0;
     pled[1].ledon = 1100;
     pled[1].ledoff = 800;
     pled[1].ledvaloff = 18;
     pled[1].ledxtime = 8;

...

     pled[5].ledvar = 1;
     pled[5].ledon = 1024;
     pled[5].ledoff = 1024;
     pled[5].ledvaloff = 0;
     pled[5].ledxtime = 6;

     pled[6].ledvar = 1;
     pled[6].ledon = 1280;
     pled[6].ledoff = 1280;
     pled[6].ledvaloff = 0;
     pled[6].ledxtime = 5;

...
     pled[12].ledvar = 1;
     pled[12].ledon = 1024;
     pled[12].ledoff = 512;
     pled[12].ledvaloff = 0;
     pled[12].ledxtime = 8;
```

Listing 2. Variabele modus voor langzaam aan en uit.

```
// values for each step in variable program for
// LED,
// 32 steps for OFF to ON and same for ON to OFF,
// half sine values
 const int varval[] = { 0, 10, 25, 35, 45, 60,
75, 90, 100, 112, 122, 134, 145, 155, 165, 175,
184, 192, 200, 208, 215, 220, 226, 232, 236, 239,
241, 244, 247, 250, 253, 255 };
```

Listing 3. Defines voor de hardware.

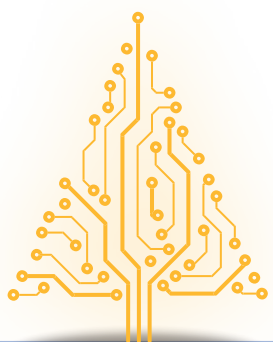
```
// Defining User Types
 // 4 garlands for this project
 #define GUIRL_A  16        // GPIO for garland A
                          // digital  PWM - Task1Led
 #define GUIRL_B  17        // GPIO for garland B
                          // digital  PWM - Task2Led
 #define GUIRL_C  18        // GPIO for garland C
                          // digital  PWM - Task3Led
 #define GUIRL_D  19        // GPIO for garland D
                         // digital  PWM - Task4Led


 ...


// Setting PWM ESP32 properties
 const int freqpwm = 5000;     // Freq Hz
 const int resolution = 8;     // 8 bits
 const int ledChannelA = 0;    // Channels for
                               // each garland
 const int ledChannelB = 1;    //
 const int ledChannelC = 2;    //
 const int ledChannelD = 3;    //
```

Listing 4. Een taak die zichzelf verwijdert.

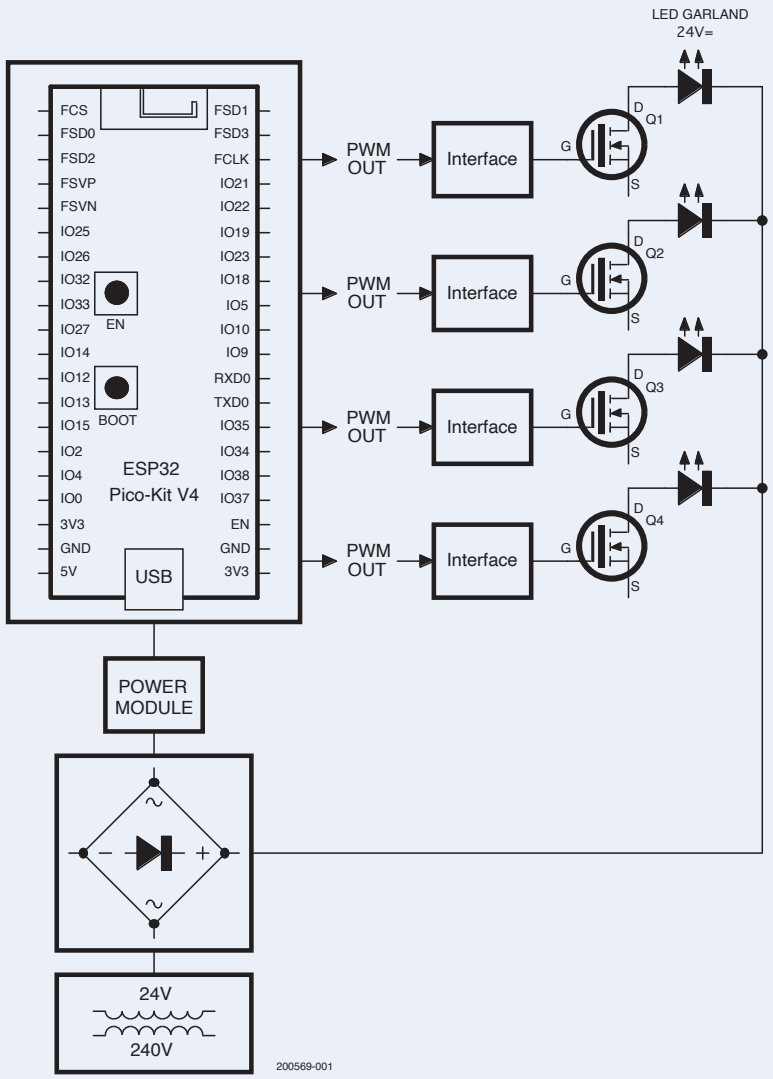
```
void loop()
{
 // Delete self, unused
  vTaskDelete(nullptr);
} // end loop
```



BC547 en een BC557) en enkele weerstanden (zie **figuur 4**) om de signaalvorm op de gate van de MOSFET te verbeteren. Hierdoor wordt het signaal echter geïnverteerd, dus daar moet bij het programmeren rekening mee worden gehouden. Weer een bestelling, en weer geduldig wachten op levering.

Toen ik alle onderdelen eindelijk in handen had, kon ik alles in elkaar zetten en de laatste tests uitvoeren. Gelukkig werden de MOSFET-modules geleverd als een set van vijf, want ik eindigde bij één exemplaar met slechts de behuizing van één MOSFET transistor in mijn vingers. Hij moet verschillende keren verbogen zijn geweest, waardoor de pennen verzwakt waren. Een beetje knutselen mijnerzijds voorkwam verdere gebroken aansluitpinnen.

Aan de uitgang van de ESP32 heb ik dus voor elke LED-slinger een interfacemodule met BC547- en BC557-transistoren, die elk een MOSFET-module aansturen die op zijn beurt één of meer LED-slingers kan aansturen (zie **figuur 5**). Ik heb alles in een plastic behuizing ondergebracht en connectoren gemonteerd voor de DC-ingang en de uitgangen voor de lampjes.



Figuur 5. Schema met ESP32 en MOSFET's.

De ESP32 effect-code

Ik heb de Arduino IDE gebruikt om het programma [1] te ontwikkelen, te compileren en te uploaden naar de microcontroller. Nu komen we bij het hart van het project, namelijk, de LED-effecten. Ik heb hiertoe een tabel gedefinieerd met een set parameters in een `struct` die de effecten ('animaties') beschrijft. Er zijn 13 ingangen in deze tabel (zie **listing 1** voor de gebruikte `struct` en de waarden om deze te initaliseren). Deze tabel kan echter worden uitgebreid met meer ingangen.

Ik heb twee soorten effecten gedefinieerd. Het ene type is een ON/OFF-animatie met een extra optie voor zwak nalichten in de OFF-toestand. Elke uitgang heeft een ON-tijd, een OFF-tijd en het aantal iteraties dat moet worden uitgevoerd voor elke in- en uitschakelcyclus. Het tweede type animatie is de variabele modus. Hiervoor heb ik een tabel gemaakt met 32 elementen. Dit zijn sinusachtige waarden om de LED's geleidelijk aan en uit te laten gaan (zie **listing 2**). De duur van de cyclus wordt geregeld door de aan- en uit-tijden in de parametertabel.

Aan het einde van deze iteraties laadt het programma op basis van een toevalsgetal een andere parameterset uit de tabel. Als dit toevalsgetal hetzelfde is als het vorige, dan wordt een andere waarde gegenereerd om te voorkomen dat twee keer na elkaar dezelfde animatie wordt uitgevoerd voor dezelfde slinger. Op deze manier zien de effecten in de kerstboom er beter uit.

Codesecties in het kort

Aan het begin van het programma worden de gebruikte PWM-uitgangen en de constanten voor de ESP32 gedefinieerd (**listing 3**). Dan komt de functie die gebruikt wordt om een pseudo-willekeurig getal te genererenn (regels 61–67 in de code [1]), en dan de code voor de RTOS-taak (regels 72–145), die gedurende onbepaalde tijd zal worden uitgevoerd. Deze taak is herbruikbaar, omdat hij gedefinieerd is met een parameter (PWM-pinnummer) die er aan doorgegeven wordt.

Daarna komt de `setup()` functie, die eerst alle waarden aan de eerder genoemde effect-parametertabel toekent. Het gedeelte daarna behandelt de PWM-uitgangsparameters (regels 231–245), en het aanmaken van de FreeRTOS-taken, met hun parameters om ze te activeren (regels 251–315). In de ESP32 wordt de scheduler zelf opgestart, en begint met de taken zodra ze zijn gedeclareerd.

Tenslotte, in de `loop()` functie, die zelf een taak is, laten we het zichzelf

verwijderen, omdat het geen code uitvoert – een taak kan zichzelf verwijderen en resources vrijmaken. De betreffende regels zijn te zien in **listing 4**. En in **figuur 6** komt de feeërieke verlichting tot leven!

Maar dat is nog niet alles

Met zijn meerdere PWM-uitgangen kan de ESP32 andere slingers aansturen. We kunnen ons voorstellen dat we alle PWM-uitgangen gebruiken om slingers aan te sturen om zo een tuin of een gevel op te leuken. Een andere mogelijkheid zou zijn om gebruik te maken van de WiFi-functionaliteit (die nu eenmaal beschikbaar is) om de ESP32 vanaf een mobiele telefoon te besturen via een kleine interface (die dan wel moet worden ontwikkeld). Ik sta open voor suggesties.

Alle materialen voor dit project zijn te vinden op de Elektor Labs-pagina [1] bij dit project.

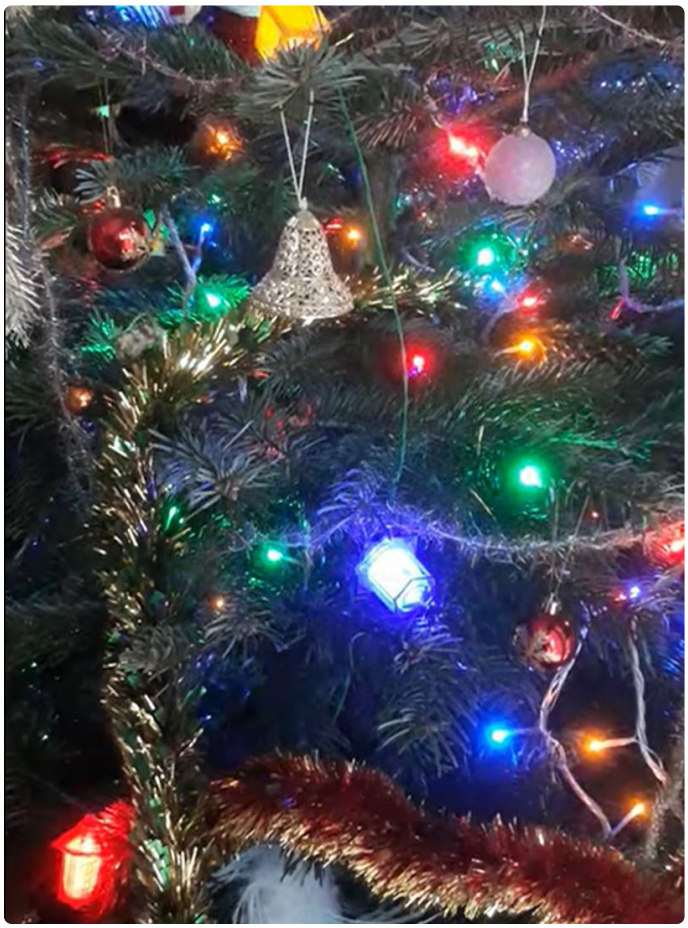
200569-03

Over de auteur

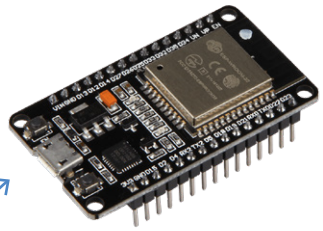
Serge Sussel ontdekte de elektronica medio jaren zestig door elektronica-tijdschriften te kopen zoals Elektor, Radio Plans, Haut-Parleur en Audiophile. Daarna heeft hij tijdens zijn studies onder meer zijn informaticakennis ontwikkeld. Hij werkte bij de Franse Nationale Centrale Bank (NCB) aan grote systemen op OS-niveau (Operating System) en het database-subsysteem, ook door het ontwikkelen van een preprocessor voor het compileren van de beveiligingselementen van transacties. Op persoonlijk vlak is hij van meet af aan (zodra hij de smaak te pakken had gekregen) geïnteresseerd geweest in analoge elektronica. Hij bouwde onder meer meetapparatuur met Heathkit-kits, hifi-voorversterkers en -eindversterkers, en een drieklaviersorgel dat in het begin van de jaren tachtig als bouwpakket werd verkocht. Inmiddels gepensioneerd houdt hij zich nog steeds bezig met analoge elektronica en met microcontroller-platforms, en hij restaureert zwaar beschadigde apparaten en apparaten die zijn afgedankt door mensen die niet beseffen dat die na vervanging van een paar componenten vaak nog jaren bruikbaar zijn.

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.



Figuur 6. De kerstboom met feestelijke verlichting.



GERELATEERDE PRODUCTEN

- **Joy-IT NodeMCU ESP32 Development Board (SKU 19973)**
  www.elektor.nl/19973
- **Warren Gay, *FreeRTOS for ESP32-Arduino* (Elektor 2020, SKU 19341)**
  www.elektor.nl/19341

WEBLINKS

[1] Mutiltasking met de ESP32 (5 afleveringen): http://www.elektormagazine.nl/magazine/elektor-144/57032
[2] Alle materialen voor dit project op Elektor Labs: http://www.elektormagazine.com/freertos-led