Lichtschakelaar DeLux

uiterst nauwkeurig lichtgestuurd schakelen

Clemens Valens (Elektor)

Lichtgestuurde schakelaars zijn in overvloed te koop voor een tientje of zo, maar de meeste schakelen ‘ergens in de schemering’. Soms vereisen toepassingen meer nauwkeurigheid en controle dan deze goedkope schakelaars mogelijk maken. Wilt u verlichting tot op de lux nauwkeurig regelen? Dan is dit project iets voor u.

Er bestaan veel ontwerpen voor lichtgestuurde schakelaars en de meeste voldoen uitstekend in de toepassing waarvoor ze zijn ontworpen. Daarbij gaat het dan meestal om het inschakelen van een lamp wanneer het gaat schemeren, en weer uitschakelen wanneer het lichter wordt. Soms wordt er ook nog een timer aan toegevoegd.
U zou kunnen denken dat dit voor alle mogelijke toepassingen volstaat; maar dat is niet waar. De reden is dat het al die ontwerpen ontbreekt aan nauwkeurigheid. Op basis van een LDR of fototransistor schakelen ze ‘ergens’ in de schemering. Lichtniveaus vertonen echter een veel sterkere variatie.

Helderheid is subjectief

Voor mensen is de intensiteit van het daglicht (de helderheid) redelijk constant. Natuurlijk zien we variaties door bewolking en de zon, maar we zijn daar niet erg gevoelig voor. De reden voor is de logaritmische helderheidsgevoeligheid van het oog. Op een bewolkte dag kan de helderheid variëren tussen 5000 en 10000 lux, maar wij zien nauwelijks verschil. Zonlicht kan resulteren in een niveau van meer dan 25000 lux, wat we natuurlijk wel merken, maar we ervaren het niet als drie of meer keer zo helder.
Planten daarentegen zijn veel gevoeliger voor lichtintensiteit dan mensen. Boeren weten dit en ze laten zelfs overdag kunstlicht op een deel van hun gewas schijnen om de opbrengst te verbeteren. Op zonnige dagen is dit meestal niet nodig, maar op bewolkte dagen kan het helpen. Om dit op een economische manier te doen, hebben ze lichtgestuurde schakelaars nodig die helderheidsverschillen nauwkeuriger detecteren dan een LDR of fototransistor.
Tegenwoordig bestaan er lichtsensoren die de helderheid direct omzetten in een luxwaarde, met een resolutie tot 16 bit. Sommige van deze sensoren meten niet alleen luxwaarden, maar ook de intensiteit van UV en wit licht. Met zo’n sensor is het vrij eenvoudig om een uiterst nauwkeurige lichtgestuurde schakelaar te bouwen.

Nauwkeurige omgevingslichtsensor

Een populaire lichtsensor is de VEML7700 van Vishay. Dit is een uiterst nauwkeurige sensor voor omgevingslicht met I²C-interface; voor een paar euro is die op een kleine module verkrijgbaar. Uit dezelfde familie stamt ook de VEML6075, een UVA- en UVB-lichtsensor met eveneens met I²C-interface.

Dankzij de digitale I²C-interface heeft de sensor geen separate A/D-omzetter nodig en kan hij rechtstreeks op de meeste microcontrollers worden aangesloten. De uitvoer is beschikbaar in twee 16-bits registers: een voor omgevingslicht (ook wel ALS genoemd) en een voor wit licht. Wit licht bestrijkt een breed spectrum van 250 nm tot 950 nm. Het ALS-spectrum is veel smaller, van ongeveer 450 nm tot 650 nm, omdat het correspondeert met de gevoeligheid van ons oog. Planten zijn geen mensen, dus de te gebruiken uitvoer hangt af van de toepassing.

De sensorgevoeligheid is ook belangrijk. De VEML7700 heeft een resolutie van 0,005 lx per LSB en een maximaal detectieniveau van 167.000 lx (minimaal 0,01 lx). Zo'n groot bereik zou 25-bit waarden vereisen, maar de sensor is slechts 16 bit breed; daarom kan een gevoeligheidswaarde worden gespecificeerd (soms 'gain' genoemd) om metingen binnen bereik te brengen. Door de grote gevoeligheid kan de sensor worden gebruikt achter oppervlakken die slechts weinig licht doorlaten en toch bruikbare resultaten opleveren.

Wanneer er weinig licht is, kan de sensor gedurende maximaal 800 ms integreren. En ten slotte kunnen lage en hoge drempelwaarden worden ingesteld die interrupts kunnen triggeren, wat het makkelijk maakt om een alarm of een automatische schakelfunctie te creëren.

Prik 'm op een ESP32

De VEML7700-module die ik voor mijn experimenten heb gebruikt, is gekocht bij Adafruit. Een goed platform om het mee te gebruiken is de ESP32-Connected Thermostat [1] (ook bekend als Automator, zie [2]). De module heeft vijf pinnen, maar omdat hij twee voedingsopties heeft, hebben we er maar vier nodig. Deze vier pinnen hebben dezelfde volgorde als de connector voor het OLED-display op de thermostaat, zodat we de module op K9 prikken. Zorg ervoor dat de VIN-pin niet is aangesloten en dat de module naar boven wijst (in tegenstelling de stand van het OLED-display, zie **figuur 1**).

Software met ESPHome

Nadat de sensor op de ESP32 is aangesloten, moeten we wat software produceren om het allemaal te laten werken. Omdat het doel een soort lichtgestuurde schakelaar is, ligt het voor de hand om een domotica-platform te gebruiken; mijn favoriet is ESPHome. Elektor heeft verschillende projecten gepubliceerd die ESPHome [3][4] gebruikten, dus u weet misschien al hoe u het kunt gebruiken en configureren, maar dit project introduceert een concept dat ik nog niet eerder heb behandeld: een aangepaste sensor. Als ESPHome nieuw voor u is, raad ik u aan eerst [3] en [4] te lezen en te bekijken.

We hebben een aangepaste sensor nodig

ESPHome ondersteunt veel sensoren, maar (op het moment van schrijven) niet de VEML7700. Het kent andere lux-sensoren, maar deze niet. Dat is echter geen probleem, aangezien ESPHome een methode biedt om uw eigen sensor toe te voegen. Om dit te doen, moet u wat C++ code schrijven, dus het maakt de zaken wel wat ingewikkelder.

Zoals eerder (zie [3] en [4]) moeten we een sensorsectie declareren in het ESPHome-configuratiebestand voor dit apparaat. Het *platform* van de sensor moet nu *custom* worden gemaakt. Dan weet ESPHome dat u voor alle details zorgt.

Dan volgt een *lambda*-sectie die bestaat uit een paar regels C++-code om ESPHome te vertellen een door ons ontworpen sensor te registreren (die we *veml7700* hebben genoemd). We moeten ook de datastroom/stromen specificeren die onze sensor produceert. In dit geval heeft hij drie uitgangen: ALS, lux en wit licht. De volgorde is belangrijk en moet worden gerespecteerd wanneer er elders in het YAML-bestand naar wordt verwezen.

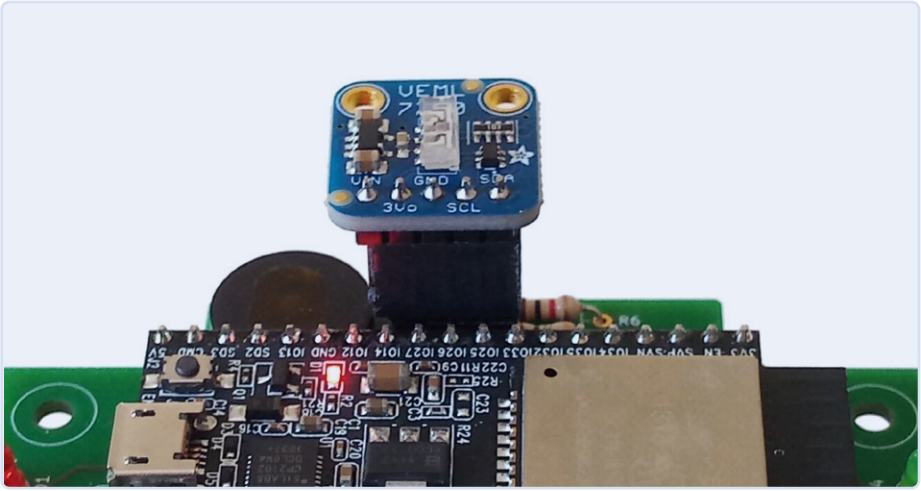
We gaan verder met normale YAML-statements om de sensoruitgangen nader te specificeren. Dat doen we met een sectie *sensors* (meervoud!) waar we voor elke datastroom de naam, de eenheden en het aantal decimalen kunnen specificeren. De volgorde is hetzelfde als in de return statement in de lambda-sectie.

Alleen de lux-datastroom heeft eenheden ('lx'), en het heeft geen zin daarvoor decimalen te gebruiken, dus zetten we die op nul.

Als laatste moet u aangeven waar ESPHome de driver voor de aangepaste sensor kan vinden. We doen dit in de *esphome*-sectie aan het begin van het configuratiebestand waar we een bestand (*veml7700.h*) toevoegen aan de *includes*-subsectie. Op het systeem waar ESPHome op draait, moet dit bestand zich in dezelfde map bevinden als het YAML-bestand van het apparaat.

Het moeilijke deel

Klaar? Niet helemaal. Nu komt het moeilijke deel, namelijk de driver voor de aangepaste sensor. Deze driver moet voldoen aan de ESPHome-normen voor componenten (een sensor is een component), wat betekent dat hij bepaalde functies moet bieden die ESPHome van zijn componenten verwacht, en moet communiceren met de sensor zelf.



Figuur 1. De 5-pins VEML7700-module wordt op de 4-polige connector K9 geprikt. De VIN-pin is niet aangesloten, ook al lijkt dat anders.

Voor het tweede deel kunnen we terugvallen op Adafruit, die niet alleen de VEML7700-module produceert maar er ook een Arduino-bibliotheek voor heeft gemaakt [5]. Onze driver moet zorgen voor de interface tussen ESPHome en die Arduino-bibliotheek.

Een bibliotheek van derden installeren

ESPHome biedt een mechanisme voor het toevoegen van bibliotheken van de open-source community: voeg eenvoudig de exacte (!) naam van de bibliotheek toe aan de *libraries*-subsectie in de *esphome*-sectie. In ons geval is dit "Adafruit VEML7700 Library". Wanneer ESPHome dan het configuratiebestand verwerkt, zal het eerst de bibliotheek installeren (als dat nog niet is gebeurd, zie **figuur 2**) voordat het alles compileert. In uw aangepaste driver kunt u de bibliotheek gewoon opnemen zoals elke andere bibliotheek.

Ik ben hier niet al te diep op ingegaan, dus ik weet niet aan welke criteria een bibliotheek van derden moet voldoen om op deze manier te worden gebruikt. Wellicht moet u de platform.io-documentatie raadplegen voor meer details, aangezien dit de toolchain is die door ESPHome wordt gebruikt.

Een ingekapselde Arduino-sketch

Ons stuurprogramma is een C++ klasse die ten minste een constructor en een functie met de naam *update* moet bieden. We hebben ook een *setup*-functie nodig om de Adafruit-driver te initialiseren. De functies *setup* en *update* van onze klasse doen wat normaal zou worden gedaan in de *setup*- en *loop*-functies van een typische Arduino-sketch met behulp van de Adafruit-driver. Kortom, onze klas kapselt een Arduino-sketch in inclusief globale variabelen en voegt er enkele ESPHome-specifieke zaken aan toe. Voor ESPHome moeten we een aanroep van *publish_state* voor elke datastroom (ALS, lux en wit licht) in de functie *update* invoegen. Hierdoor worden de gegevens beschikbaar voor de rest van de wereld. De aanroep-volgorde moet dezelfde zijn als in het return-statement in de *lambda*-subsectie van de *sensor*-sectie (zie hierboven).

Omdat onze klasse erft van de *PollingComponent*-klasse, die op zijn beurt erft van de *Component*-klasse, zijn er andere functies beschikbaar die u misschien wilt gebruiken. Een polling-component is een component die periodiek wordt aangeroepen, en hoe vaak dat gebeurt kan worden gespecificeerd (bijvoorbeeld in onze constructor). Raadpleeg [6] voor meer details.

De I²C-bus toevoegen

Nu hoeven we alleen nog maar de sensor in de software aan te sluiten op de I²C-bus (dus een logische verbinding tussen beide maken, aangezien we al een fysieke verbinding hebben). De Adafruit-bibliotheek gebruikt hiervoor de Wire-bibliotheek van Arduino. In het ESPHome YAML-bestand voegen we daarom een *i2c*-sectie toe zodat ESPHome weet dat het nodig is.

De ESP32 heeft twee I²C-bussen met SDA- en SCL-signalen die op bijna elke pin van de chip kunnen worden aangesloten. ESPHome ondersteunt daarom meerdere I²C-bussen met vrij toewijsbare pinnen, zodat we slechts moeten specificeren wat waar naartoe gaat. Maar de Arduino Wire-bibliotheek ondersteunt slechts één I²C-bus, dus hoe vertel je hem om de gewenste bus te gebruiken? Welnu, dat kan niet, omdat hij altijd de standaardbus zal gebruiken. In ESPHome is de standaard I²C-bus de eerste bus die is gespecificeerd in de *i2c*-sectie. Het zou leuk zijn geweest om een I²C-bus met zijn ID te kunnen specificeren, en ESPHome zou dit waarschijnlijk graag ondersteunen, maar de onderliggende Arduino Wire-bibliotheek staat dit niet toe.

Klaar

Hier eindigt dit artikel. Als de Automator is geprogrammeerd door ESPHome met de hierboven beschreven code, is het resultaat een apparaat dat kan worden bediend vanaf een domotica-controller zoals Home Assistant (of zichzelf). Er staan geen automatiseringsregels in het ESPHome YAML-bestand, dus zonder controller meet het alleen de intensiteit van het omgevingslicht. Automatiseringsregels kunnen worden toegevoegd aan het YAML-bestand zelf, of ze kunnen worden gemaakt in bijvoorbeeld Home Assistant. Raadpleeg [3] en [4] voor een beschrijving van de rest van de YAML-code – dat is niets bijzonders.

Het YAML-configuratiebestand en de C++ code kunnen worden gedownload van de projectpagina bij dit artikel [7].

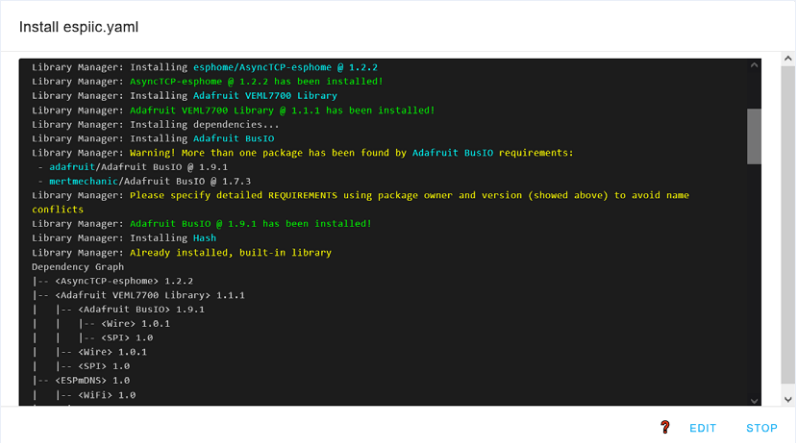
210190-03

Een bijdrage van

Ontwerp, tekst en foto's: **Clemens Valens**
Redactie: **Jens Nickel, C.J. Abate**
Vertaling: **Eric Bogers**
Layout: **Harmen Heida**

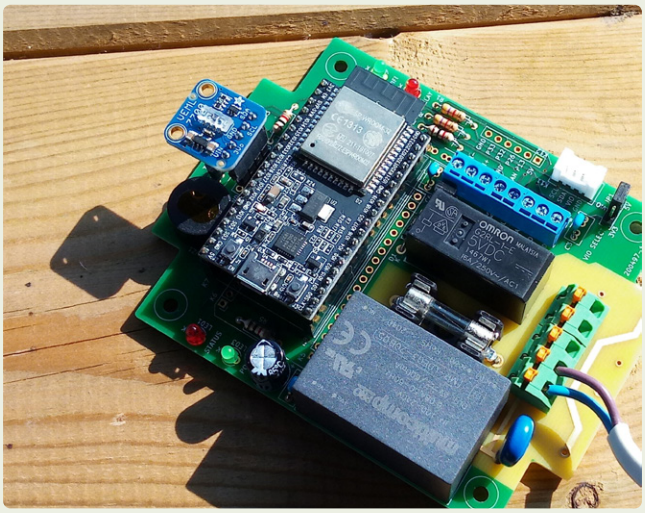
Vragen of opmerkingen?

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

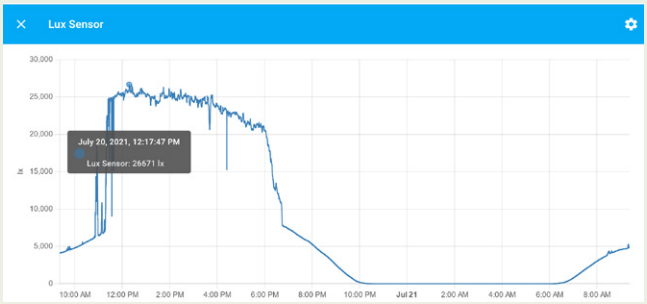


Figuur 2. ESPHome installeert de Adafruit-bibliotheek automatisch.

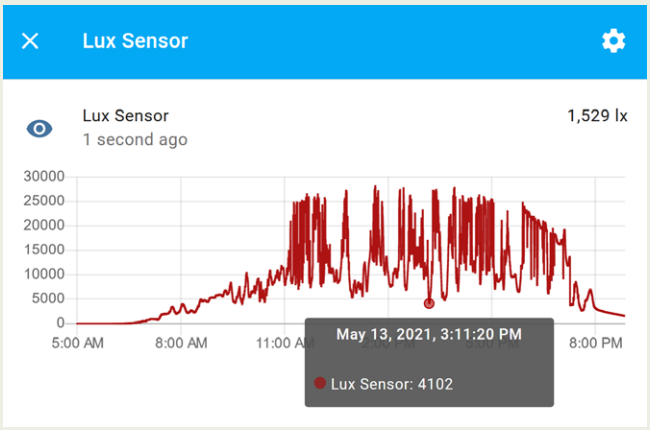
Een paar resultaten



De lichtsensor in fel zomerzonlicht. 's Middags, met de gevoeligheid ingesteld op 0,125 en een integratietijd van 100 ms, meldde de sensor waarden van ongeveer 48.000 voor ALS en iets meer dan 30.000 voor wit licht. Deze waarden komen overeen met een lichtintensiteit van ongeveer 22.000 lx. Om 10:30 uur onder dezelfde omstandigheden heb ik 16.800 lx gemeten, een verschil van 5.200 lx. In mijn ogen was het in beide situaties echter vrijwel even helder.



Verloop van de helderheid op een zonnige dag in juli. Het 'stuiteren' aan de linkerkant komt door de schaduw van een boom. Van 12:00 tot 18:00 uur neemt de helderheid langzaam af, maar dat merken we niet echt. De dips worden veroorzaakt door bewolking. De schaduw van het huis valt vanaf 18:00 uur op de sensor, wat de steile afname verklaart. Daarna neemt de lichtintensiteit continu af tot het donker wordt. Voor mensen is 5.000 lx al helder.



Op een licht bewolkte dag in mei kan de helderheid ergens tussen de 4.000 en 27.000 lx liggen.

Alle grafieken zijn gemaakt met behulp van Home Assistant op een breedtegraad van 47° N.

GERELATEERDE PRODUCTEN

- **ESP32 DevKitC (SKU 18701)**
  www.elektor.nl/18701
- **0,96" 128×64 I²C OLED display (SKU 18747)**
  www.elektor.nl/18747
- **Koen Vervloesem, *Getting started with ESPHome* (SKU 19738)**
  www.elektor.nl/19738

WEBLINKS

[1] Y. Bourdon, "Connected thermostaat met ESP32", Elektor Magazine september/oktober 2021: www.elektormagazine.nl/magazine/elektor-185/59914
[2] Elektor Automator: www.elektormagazine.com/labs/automator
[3] ESPHome begint hier: www.elektormagazine.com/labs/how-to-home-assistant-esphome
[4] C. Valens, "Domotica helemaal niet moeilijk", Elektor Magazine september/oktober 2020: www.elektormagazine.nl/magazine/elektor-157/59027
[5] Adafruit VEML7700-bibliotheek: https://github.com/adafruit/Adafruit_VEML7700