Het panettone-project

zuurdesemstarter onder controle

Daniel Fantin (Australië)

Dit Arduino-project is geboren uit de frustratie van vele, vele mislukte panettones. Voor wie het niet weet, panettone is een traditioneel Italiaans zoet brood dat met Pasen en Kerstmis wordt gebakken. Het heeft een 'onmogelijk' lichte en luchtige textuur, waarvoor een zeer actieve gist van een zuurdesemstarter nodig is...

Figuur 1. Een fraaie panettone, nietwaar?

Kijk eens naar **figuur 1**. Zo ziet een geslaagde panettone eruit – licht, luchtig, boordevol boter en fruit. Als het water je in de mond loopt, vind je in dit artikel alles (tenminste technisch) wat je nodig hebt om er zelf zoiets perfect te bakken. Buon appetito!

Wat is een zuurdesemstarter?

Eenvoudig gezegd is het een mengsel van bacteriën en gist die, gevoed met vers meel en water, gassen vrijmaakt die zorgen voor een luchtig, gerezen brood. De gist moet sterk genoeg zijn om het gewicht van het brood of het panettone-deeg tegen te werken. Voor brood is dat niet zo'n probleem, maar bij panettone moeten we het opnemen tegen enorme hoeveelheden boter, fruit en suiker – een zware opgave voor elke gist.

Wanneer is een zuurdesemstarter sterk genoeg om te gebruiken in panettone? Simpel: wanneer het in vier uur driemaal zo hoog wordt bij een temperatuur van 27 °C. Dat klinkt eenvoudig, maar het blijkt in de praktijk tamelijk moeilijk. Niemand wil zijn huis verwarmen tot 27 °C alleen maar voor zijn starter, laat staan dat hij dat wekenlang volhoudt terwijl de sterkte van de starter toeneemt. En hoe weet je of de starter in vier uur tijd verdrievoudigd is? Wil je dat voortdurend controleren? Wekkers zetten en een meetlint in de keuken bewaren? Te lastig, te moeilijk – gewoon te veel. Dat kun je beter aan een robot overlaten.

Wat gebeurt er als de starter niet sterk genoeg is? De resulterende panettone is niet luchtig, moeilijk te eten, en ziet er mislukt uit. En als het wel sterk genoeg is? Werp nogmaals een blik op figuur 1. Je weet gewoon dat het delicaat, luchtig, fluweelachtig, zoet, knapperig, boterig is... alles wat je wilt dat het is.

Het is mei op het moment van schrijven, en ik heb nog even voordat mijn volgende portie panettone klaar moet zijn (december). Toen ons in de cursus aan de Deakin University werd gevraagd een probleem uit de 'echte wereld' op te lossen, was Pasen net voorbij, en was een portie panettone grandioos mislukt. Dus het leek me een echt probleem om op te lossen.

Eerst had ik overwogen een gistingsmachine of iets dergelijks te kopen, maar dat zijn dure dingen, vooral als je bedenkt dat wat ik wilde bouwen veel meer mogelijkheden moest hebben (en over het geheel genomen goedkoper moest zijn).

Project-criteria

Wat willen we met dit project bereiken?

- De temperatuur van de zuurdesemstarter controleren.
- Een melding zodra het sterk genoeg is voor panettone.

Daarvoor heb ik nodig:

- iets om de hoogte van de starter in de pot te volgen (dit vertelt me of de starter in vier uur tijd in hoogte is verdrievoudigd);
- iets om de temperatuur van de starter in de pot te controleren;
- iets om de pot te verwarmen of te koelen zodat hij altijd rond de 27 °C blijft;
- iets om me op de hoogte te brengen van belangrijke zaken (bijvoorbeeld die vier uur);
- een GUI zodat ik de zuurdesemstarter kan resetten na een nieuwe vulling;

- een GUI om de belangrijkste variabelen (temperatuur, groei) op afstand te volgen.

Ik wilde gebruik maken van modulaire ontwerpprincipes, zodat ik problemen altijd tot één gebied kon beperken. Hoewel dit project gemakkelijk op één board kon worden gerealiseerd, wilde ik iets maken dat de interfacing tussen verschillende systemen testte, zodat ik de kennis had om in de toekomst ingewikkeldere projecten uit te voeren. Daarom koos ik voor:

- **sensoren:** een Raspberry Pi 3B+ om de DHT en (lasergebaseerde) afstandssensoren aan te sturen en die gegevens draadloos (naar het Particle board, zie hieronder) en via USB serieel (naar de Arduino) te versturen;
- **verwarming/koeling:** een Arduino Uno om de verwarmingselementen en ventilator in te schakelen wanneer dat nodig is, om de starter op temperatuur te houden;
- **GUI/meldingen/rijzen**: een Particle Argon-board [1] om de gebruiker een externe GUI, een LCD-GUI te geven, en waarschuwingen te geven over belangrijke events.

Figuur 2 toont blokschematisch de systeemarchitectuur.

Webhooks/IFTTT

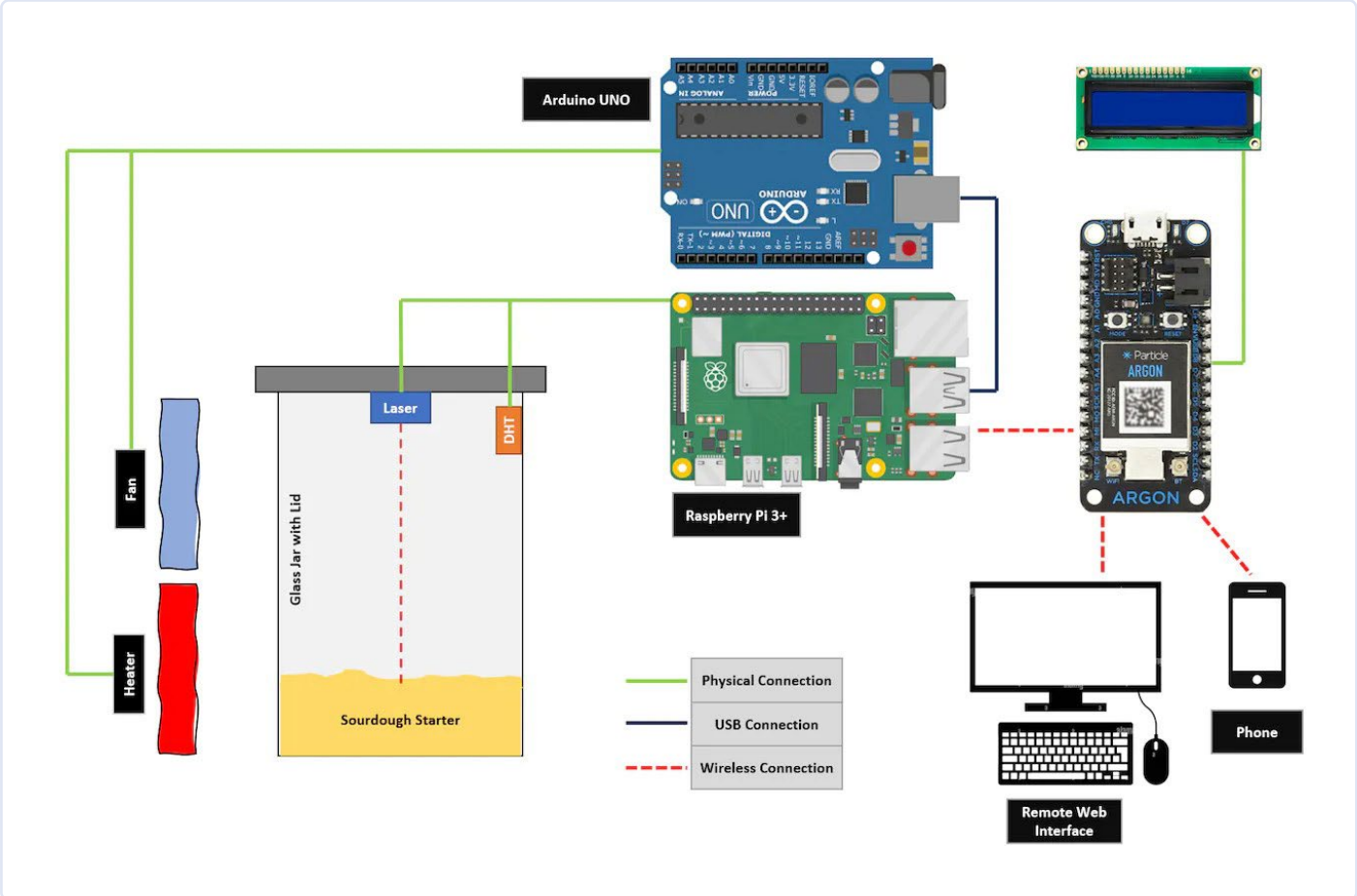
Alles moet met alles kunnen communiceren. Ik had geen noemenswaardige ervaring op dit gebied, en ik vond dat het IFTTT-platform [2] de eenvoudigste en gemakkelijkste manier was om deze communicatie tot stand te brengen. Dus zette ik een IFTTT-account op en maakte vervolgens een applet met een webhook als 'IF.' Het komt erop neer dat **IF** de webhook een webverzoek ontvangt met een JSON-payload (**J**ava**S**cript **O**bject **N**otation), dat wil zeggen een verzoek van de Raspberry Pi, **THEN** activeert het een Particle-functie. Het Particle-board kan dan de JSON-gegevens gebruiken op welke manier dan ook. De website PiMyLifeUp [3] heeft me enorm geholpen. Het belangrijkste is dat de webhook je een adres geeft in de vorm van:

```
https://maker.ifttt.com/trigger/
    tempsent/json/with/key/XXXXXX
```

waarin `XXXXXX` je persoonlijke sleutel is. Alles wat we dan moeten doen is die URL gebruiken in onze code om dit applet te activeren door op het juiste moment te posten.
De tweede IFTTT is een verbinding tussen Particle en de telefoon – dit is vrij eenvoudig. De IF wacht tot Particle een gebeurtenis met een specifieke naam publiceert, en stuurt vervolgens een melding naar de IFTTT-app op de telefoon als en wanneer deze wordt ontvangen.

De Raspberry Pi

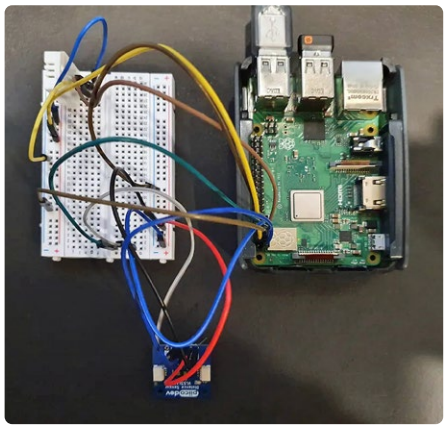
Laten we nu alle sensoren aan de praat krijgen en de RPi met alles communiceren.



Figuur 2. Blokschema van de systeemarchitectuur.

De aansluitingen:

- **DHT:** verbind de voedingspin van de DHT met de 5 V van de RPi, de tweede pin met RPi GPIO4, en de vierde pin met RPi GND. De derde pin van de DHT-sensor wordt niet gebruikt. Opmerking: het zou beter zijn geweest een DHT22 te gebruiken (vanwege de nauwkeurigheid) maar de mijne was defect en ik heb hem vervangen door een DHT11 voor het eindproduct.
- **VL53L1XLaser:** sluit Vin aan op de 3,3 V van de RPI, GND op RPi GND, SDA op RPi GPIO2 en SCL op RPi GPIO3. Dit is voor de seriële communicatie tussen de Raspberry Pi en de laser.



Figuur 3. Testopstelling voor de laser- en temperatuurmodules.

Dit zijn de redenen om een laser in plaats van een ultrasone sensor te gebruiken om de afstand te meten:

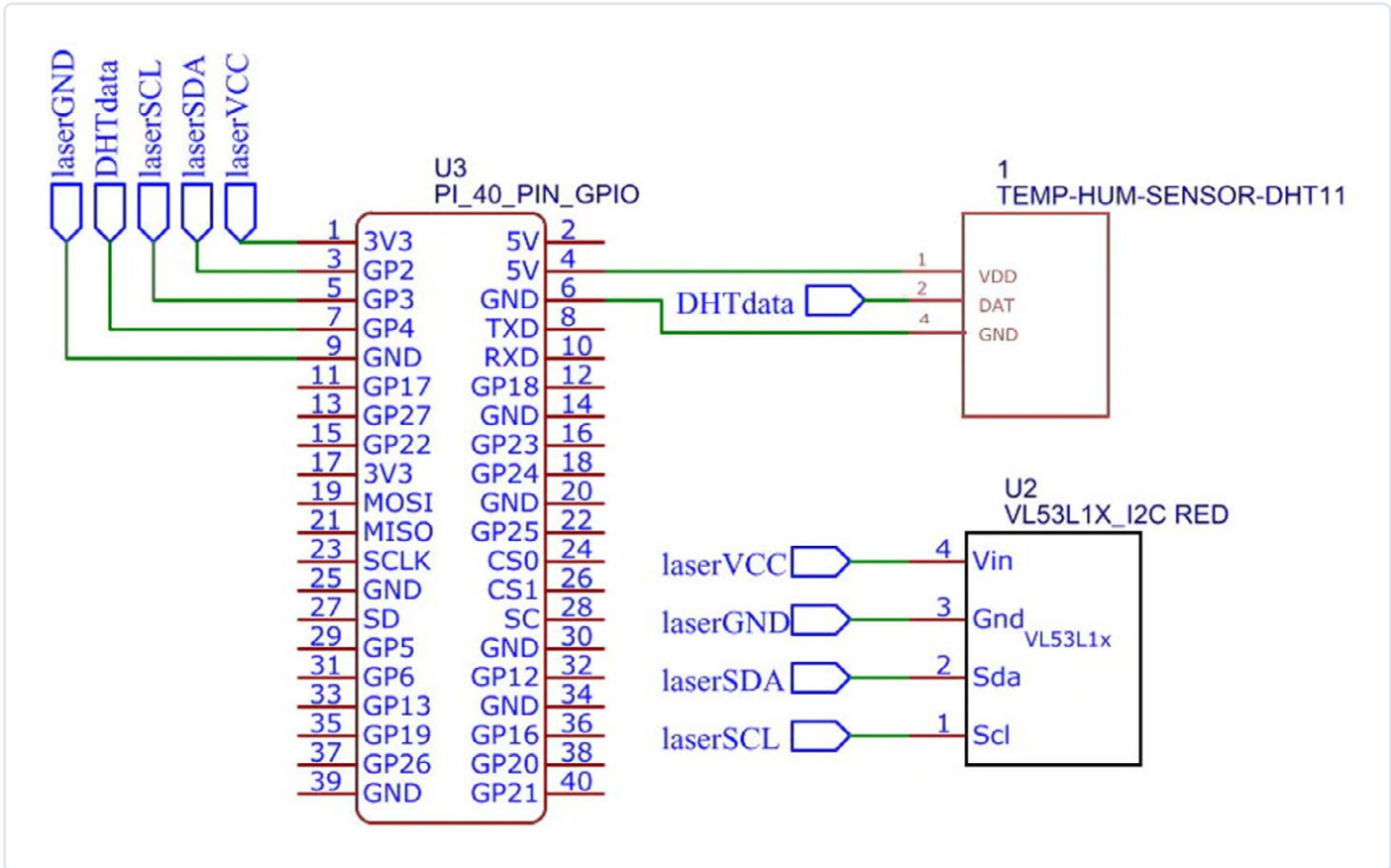
- nauwkeurigheid – lasermetingen zijn veel nauwkeuriger;
- een laser is betrouwbaarder, vooral omdat het deegoppervlak zacht is;
- ik wil mijn katten niet de hele dag lastigvallen met constante ultrasone geluiden.

Figuur 3 toont mijn testopstelling voor de laser-afstandsmeter en de temperatuurmodules; de aansluitingen moeten worden gemaakt zoals in **figuur 4**. Netpoorten zijn in het schema weergegeven om de bedrading duidelijker te maken. Het schema toont niet de USB-verbinding tussen de Arduino en de Raspberry Pi, hoewel dat een noodzakelijk onderdeel voor de communicatie is.
Nu wordt het tijd om te programmeren. Ik maakte een nieuw Python-bestand aan op de Raspberry Pi en ging aan de slag. Ik gebruikte de Python IDE, Thonny [3], omdat ik die heel gemakkelijk in het gebruik vind. Laten we beginnen met het importeren:

- Adafruit DHT [5] om de DHT-sensor te gebruiken.
- PiicoDev-VL53L1X van Core Electronics [6] voor de laser
- Time / Sleep
- Requests om de webhooks te gebruiken en draadloos te communiceren
- Serial voor communicatie via USB met de Arduino

Een paar setup-items

Ik gebruik vlaggen om Particle te signaleren dat de ventilator of de verwarming aan staan.



Figuur 4. De verbindingen van figuur 3.

De 'airmessage' wordt als integer serieel doorgegeven aan de Arduino om 'iets te doen'. De seriële poort werd ingesteld op het juiste adres door het te controleren in de terminal:

```
ls /dev/tty*
```

Voor de robuustheid heb ik een 'gemiddelde metingen'-methode gebruikt – deze neemt vijf opeenvolgende metingen en middelt ze voor ze worden geretourneerd naar webhooks/serieel enzovoort. Dit zorgde ervoor dat een enkele foutieve meting niet ten onrechte gebeurtenissen triggert. Het bevatte ook een controleroutine die de gebruiker een melding stuurt als er 100 foutieve metingen zijn opgetreden. Er staat veel commentaar in de hoofdlus, voornamelijk gebruikt voor het debuggen. Deze lus ontvangt het gemiddelde en beslist dan wat er moet gebeuren:

- schakel de verwarming in als de temperatuur lager is dan 24 °C;
- voer de temperatuur op tot 27 °C en zet dan de verwarming uit;
- als de temperatuur > 30 °C, zet dan de ventilator aan;
- laat de temperatuur dalen tot 27 °C en zet dan de ventilator uit;
- zolang de temperatuur in een bereik van 24...30 °C ligt (de optimale zone voor de starter), is geen device actief.

Dit leidt tot het Python-script *PiCodepy*, een onderdeel van de software dat gratis kan worden gedownload van de website bij dit artikel [7].

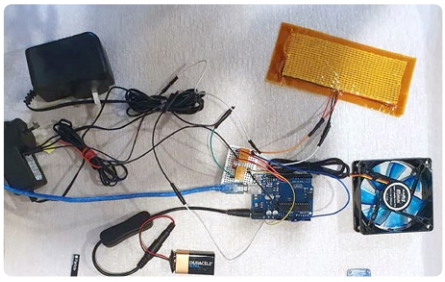
Arduino

De Arduino-opstelling is een grote uitdaging wat betreft de stroomvoorziening, maar gelukkig is mijn vader een elektronica-expert. De ventilator heeft 12 V nodig, wat veel meer is dan de Arduino kan leveren. Verder verlangen de verwarmingselementen 5 V bij een weerstand van 6 Ω. De benodigde stroom bedraagt dus bijna 1 A – veel meer dan wat een Arduino-pin kan leveren. Daarom levert een externe voedingsbron de benodigde stroom voor de ventilator/verwarming wanneer deze door de Arduino wordt geactiveerd.

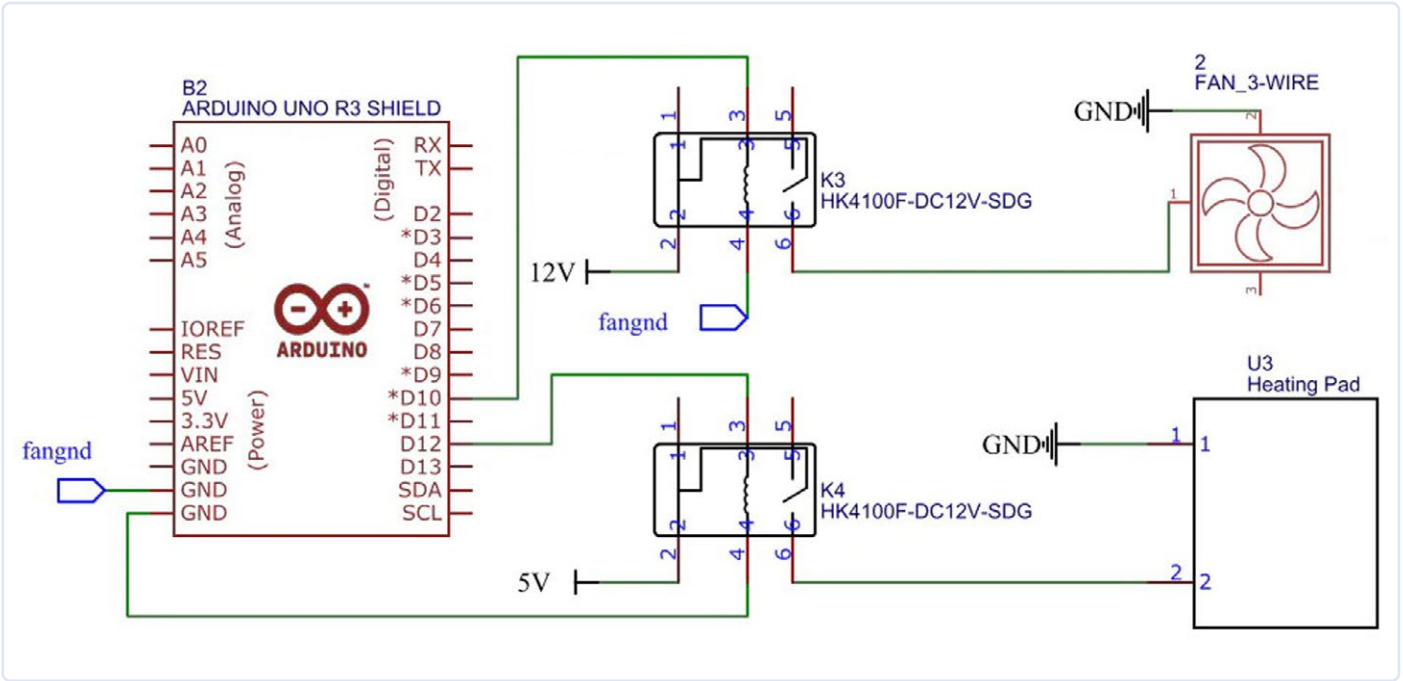
Setup

- **Verwarming:** sluit pin 12 van de Arduino aan op de ene kant van de relaisspoel, en de andere kant op Arduino GND. Sluit +5 V van de voeding aan op de common rail van het relais. Verbind de positieve pool van de verwarmingselementen met de 'altijd uit'-kant van het relais.
- **Ventilator:** sluit pin 10 van de Arduino aan op de ene kant van de relaisspoel, en de andere kant op Arduino GND. Sluit de +12 V van de voeding aan op de common-aansluiting van het relais. Sluit de positieve pool van de ventilator aan op de 'altijd uit'-kant van het relais.
- **GND:** sluit de GND van de verwarmingselementen en de ventilatoren aan op de GND van de voeding.
- **Arduino:** sluit een 9 V batterij (of voeding) aan op de Arduino. Alleen de Arduino wordt gevoed door de batterij, NIET de Raspberry Pi.

Figuur 5 toont de experimentele opstelling en **figuur 6** de schakeling van het systeem rond verwarming en ventilator. De Arduino krijgt gegevens van de RPi via USB (serieel). De berichten zijn gecodeerd in UTF-8. Het enige wat we moeten doen is de seriële code ophalen en afhankelijk van de inhoud de relevante actie uitvoeren. Dit gebeurt



Figuur 5. Provisorische opstelling van het systeem.



Figuur 6. Aansluitingen rond het Arduino-board.

via een `if`-statement in de hoofdlus van de code. De website Automatic Addison [8] was hierbij zeer behulpzaam. De Arduino-code is als bestand te vinden op [7].

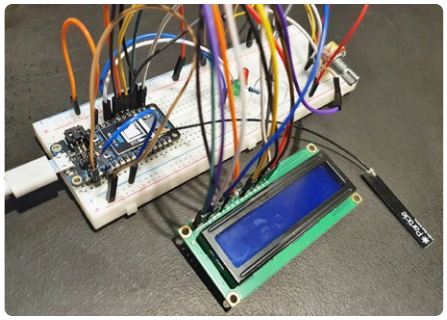
Particle Argon

De Particle Argon speelt een centrale rol in de communicatie tussen de gebruiker en de code en devices. Het LCD, de potentiometer en de LED's moeten op het board worden aangesloten. Het LCD wordt via I²C aangestuurd. Toch moest ik nog enkele extra verbindingen maken (**figuur 7**). De Hackster website [9] hielp me om het goed te krijgen.
Vervolgens heb ik twee LED's aangesloten: een groene voor 'starter is goed' en een rode voor 'starter is niet goed.' Deze werden aangesloten op de Argon-pinnen D6 en D7, en vervolgens via een 220Ω-weerstand aan massa gelegd. Na het toevoegen van de potentiometer was het Particle-subsysteem klaar (**figuur 8**). Het volledige schema staat in **figuur 9**.
De Particle-code [7], gegenereerd binnen de Particle IDE, is de langste en ingewikkeldste code, omdat deze alles met elkaar verbindt. Een overzicht:

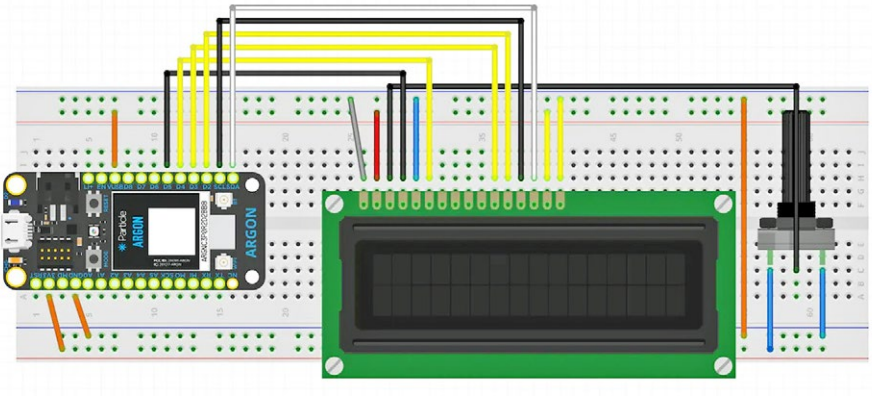
- Aanmaken van het `lcd`-object, lege strings, initialisatie indien nodig.
- Hard-coderen van de starthoogtes van de pot en van het deeg, omdat dit bekende en vaste waarden zijn.
- Aanmaken van de functie `resetbutton()`. Deze wordt op afstand via HTML geactiveerd om de timer te resetten zodra de vier uur zijn verstreken (en de resultaten duidelijk zijn).

Setup

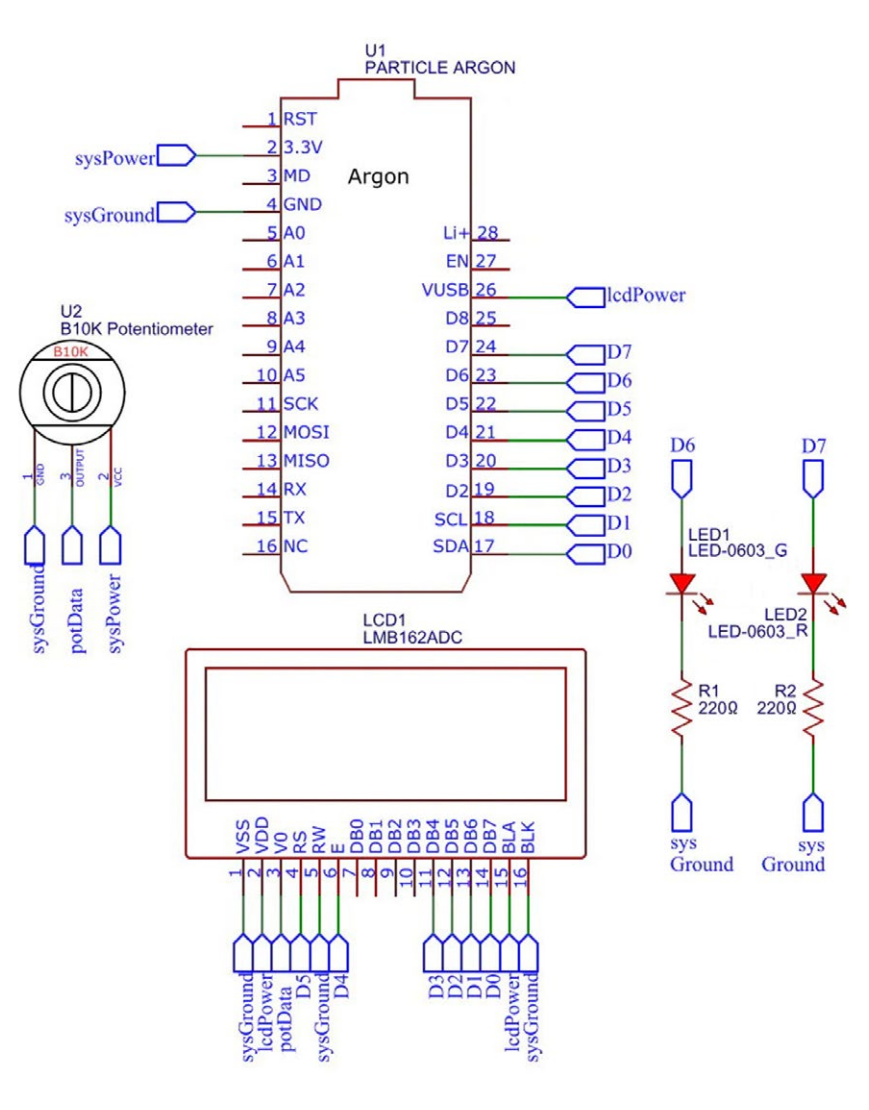
- Wis het LC-display en toon een introbericht.
- Initialisatie van de twee functies die samenwerken met IFTTT. `displayTemp` voert de



Figuur 8. Configuratie van het Particle-board.



Figuur 7. Display- en Particle-board



Figuur 9. Schema van Particle en componenten.

functie `tempDisplay()` uit en `reset` voert de functie uit die de knop reset.
- Vastleggen van de starttijd.
- Doof de LED's.

Loop

- Er is een vertraging van tien seconden, zodat het LC-display niet telkens opnieuw wordt hertekend.
- Toon de huidige commando's (`cmd1` en `cmd2`). Deze worden ingesteld in de functie `tempDisplay()`.
- Berekening van de groei van de starter.
- Als de timer vier uur bereikt, verschijnt een nieuw bericht met het eindresultaat op het display.

De functie tempDisplay()
Hier worden de JSON-gegevens ontvangen van de eerdergenoemde webhook. Dit zijn alle kritieke afstandsgegevens van de Raspberry Pi, die draadloos naar Particle worden verzonden. Helaas kon ik JSON niet parsen om het gemakkelijk toegankelijk te maken. In plaats daarvan nam ik het gewoon als een string en deelde het in substrings, wat heel eenvoudig was. Ik zorgde ervoor dat de RPi-formateerregels heel streng waren, zodat dit altijd correct zou werken. De JSON bevatte de volgende gegevens: afstand, temperatuur en of de ventilator of verwarming aan staat. Deze gegevens werden gebruikt om zowel de HTML als het LC-display aan te sturen. Deze code is ook te vinden op [7].

HTML
HTML werd gebruikt om op afstand toegang te krijgen tot het systeem. Hierdoor kan ik de timer resetten wanneer de vier uur zijn verstreken, en de temperatuur en groei controleren overal waar ik toegang heb tot het internet. De code is te lang om hier af te drukken, maar we kunnen wel een paar functies laten zien. De meeste code komt uit de Particle-tutorial [10]. De belangrijkste functies van de HTML zijn:

- Toon de actuele temperatuur (door subscribing op een event-stream).
- Toon actuele groeigegevens (door subscribing op een event-stream).
- Reset de timer (door een functie aan te roepen wanneer deze knop wordt gedrukt).

De `particle.getEventStream()`-calls (**listing 1**) laten zien hoe gegevens uit de

Listing 1. De particle.getEventStream() calls.

```
particle.getEventStream({ name: 'tempEvent', auth: sessionStorage.
  particleToken }).then(
{
    function (stream) {
        console.log('starting event stream');
        stream.on('event', function (eventData)
        showTemp(eventData)
    });
});

particle.getEventStream({ name: 'growEvent',auth: sessionStorage.
  particleToken }).then( function (stream)
{
   {
    console.log('starting event stream');
    stream.on('event', function (eventData)
    showGrow(eventData)
   });
});

particle.getEventStream({ name: 'timeEvent', auth:sessionStorage.
  particleToken }).then(
    function (stream) {
    {
       console.log('starting event stream');
       stream.on('event', function (eventData)
       showTime(eventData)
    });
});

Listing resetControl
function resetControl(cmd) {
     //const deviceId = $('#deviceSelect').val();
    $('#statusSpan').text('');
    particle.callFunction({ deviceId: 'X', name: 'reset', argument:
  cmd,auth:sessionStorage.particleToken}).then(err);
    function (data) {
        $('#statusSpan').text('Reset completed'); },
    function (err) {
         $('#statusSpan').text('Error calling device: ' +
    } );
}
```

Listing 2. resetControl() functie.

```
function resetControl(cmd) {
 //const deviceId = $('#deviceSelect').val();
    $('#statusSpan').text('');
    particle.callFunction({ deviceId: 'X', name: 'reset', argument:
  cmd, auth:sessionStorage.particleToken}).then(err);
    function (data) {
        $('#statusSpan').text('Reset completed'); },
    function (err) {
         $('#statusSpan').text('Error calling device: ' +
} );
}
```

event-stream worden gehaald. De `resetControl()`-functie (**listing 2**) wordt getriggerd door het indrukken van een knop (Particle ID is verwijderd).

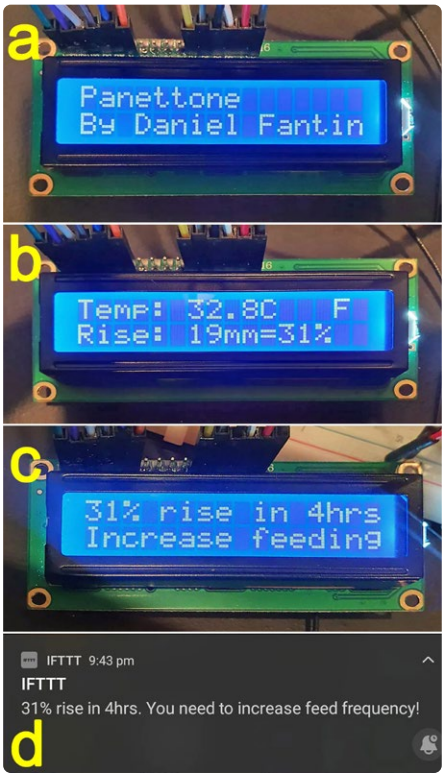
Actiefoto's

Figuur 10 toont enkele kiekjes van het LC-display op verschillende tijdstippen. De laatste, *d*, toont de melding van *c* op een smartphone. **Figuur 11** toont het complete robotsysteem (a), de doos met de elektronica erin (b) en de printen en draden bovenop de grote doos (c). En als je je afvraagt hoe de starter eruitziet, toont **figuur 12** de snelle groei van de starter tussen voor en na de vier uur durende sessie.

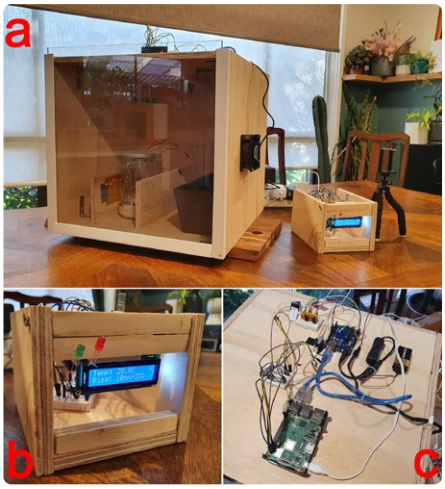
Conclusie

Dat is het. Alles werkt, en alles is gesynchroniseerd. Nu is het tijd om de eerste zuurdesemstarter automatisch te produceren met deze deegrobot. En, natuurlijk, wanneer de starter sterk genoeg is, te gaan bakken en genieten!

220416-03 (vertaling: Willem den Hollander)



Figuur 10. Verschillende inhoud op het LC-display (a, b en c), plus het smartphone-bericht na vier uur.



Figuur 11. De complete robot (a) met stuureenheid (b) en enkele boards en bedrading (c).

Figuur 12. Zuurdesemstarter: begintoestand (a) en na het bereiken van de juiste hoogte vier uur later (b).

Over de auteur

Daniel Fantin is tweedejaarsstudent informatica aan de Deakin University. Hij heeft een passie voor voedsel en technologie, dus de combinatie van Italiaanse keuken en robotica is een natuurlijke ontwikkeling.

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via Hackster [12] of naar de redactie van Elektor-redactie via redactie@elektor.com.

Gerelateerde producten

- **Arduino Uno**
  www.elektormagazine.nl/arduino-uno

WEBLINKS

[1] Particle Argon: https://docs.particle.io/argon/
[2] IFTTT: https://ifttt.com
[3] PiMyLifeUp: https://pimylifeup.com/using-ifttt-with-the-raspberry-pi/
[4] Thonny IDE: https://thonny.org
[5] Adafruit DHT: https://github.com/adafruit/Adafruit_Python_DHT
[6] PiicoDev_VL53L1X: https://elektor.link/piicodevgithub
[7] Project-software: www.elektormagazine.com/220416-01
[8] Automatic Addison: https://elektor.link/rpiarduino2way
[9] Hackster-website: https://elektor.link/particlehackster
[10] Particle-tutorial: https://elektor.link/particleapitut
[11] Project op hackster.io: https://elektor.link/hacksterpanettone
[12] Daniel Fantin op hackster.io: www.hackster.io/danielfantin