Sensor-ABC: de DS18B20 temperatuursensor

aansluiting op de 1-Wire bus

Mathias Claussen (Duitsland)

De Dallas Semiconductor DS18B20-sensor zit in veel beginner-kits en biedt een gemakkelijke introductie tot het thema temperatuurmeting. Voor je erin duikt, is het handig om enige kennis te hebben van de 1-Wire bus en de aansluiting van de sensor. In dit artikel maken we een korte excursie naar de basis van het gebruik van de DS18B20, zodat je vol vertrouwen temperaturen kunt gaan meten!

Als je temperaturen wilt registreren met een microcontroller, kun je kiezen uit verschillende sensoren en bussystemen. Een populaire keuze is de Dallas Semiconductor DS18B20. Deze heeft een bereik van –5 °C tot +125 °C (–67 °F tot +275 °F) met een indrukwekkende nauwkeurigheid van ±0,5 °C. Door zijn veelzijdigheid is hij niet alleen geschikt voor het meten van omgevingstemperaturen, maar ook voor het bewaken van vriezers en koelruimten. In dit artikel bekijken we hoe hij samenwerkt met een microcontroller, inclusief voorbeelden met broncode en schema's om te laten zien hoe je hem in je eigen projecten kunt integreren.

De DS18B20

De DS18B20-sensor maakt gebruik van het 1-Wire bussysteem, gepatenteerd door Dallas Semiconductor in 1989, en wat de aansluiting van meerdere sensoren mogelijk maakt. Met een voedingsbereik van 3,0 V tot 5,5 V kan de DS18B20 rechtstreeks worden aangesloten op GPIO's van een reeks apparaten – van de Arduino UNO tot de Raspberry Pi Pico. De 1-Wire bus biedt ook een parasitaire voedingsmodus, waarbij energie wordt onttrokken aan de datalijn om de sensor van stroom te voorzien. Dit betekent dat de 1-Wire bus slechts één pin van een microcontroller (MCU) nodig heeft voor de aansluiting van meerdere sensoren, waardoor het een populaire keuze is, vooral bij kleinere MCU-boards met een beperkt aantal GPIO's. Hoewel er andere 1-Wire sensoren bestaan, zullen we ons hier beperken tot de Dallas-component.

Figuur 1. De DS18B20 in TO-92 behuizing.

Figuur 2. Waterdichte versie van de DS18B20, met kabel.

Figuur 3. Pinning van de DS18B20.

De DS18B20-sensor is verkrijgbaar in verschillende uitvoeringen, zoals te zien in **figuur 1** en **figuur 2**. Dankzij de 1-Wire bus is het bijzonder eenvoudig om meerdere sensoren aan te sluiten. Hoewel het 1-Wire protocol zelf geen grens stelt aan het aantal sensoren dat op de bus kan worden aangesloten, worden die grenzen bepaald door de elektrische eigenschappen van de bus.

De 1-Wire bus

Voor de aansluiting van sensoren op de 1-Wire bus zijn drie draden nodig: massa (GND), data (DQ) en voeding (VCC). De 1-Wire bus is bidirectioneel en werkt volgens een controller/target-concept (master en slave), waarbij controller en target gegevens uitwisselen via de datalijn. De pinning van de DS18B20-sensor is geschetst in **figuur 3**.

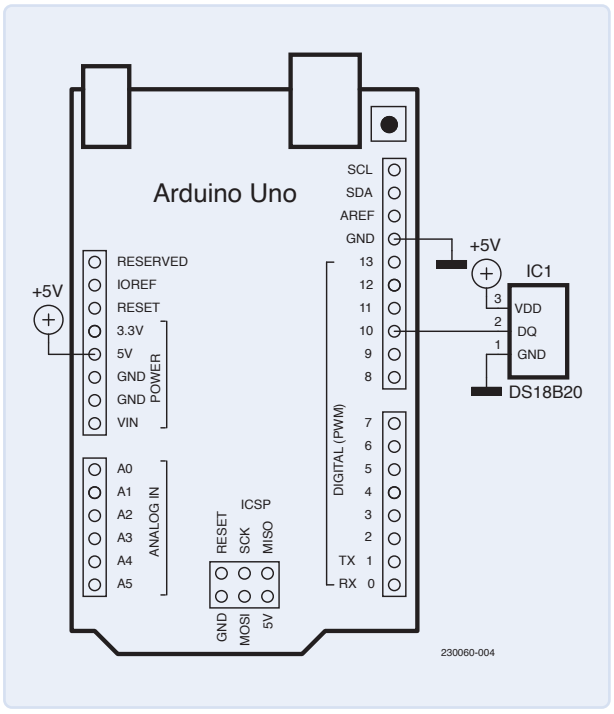
Alle knooppunten op de enkele datalijn gebruiken een open-collector driver; wanneer onverhoopt gegevens botsen (als twee knooppunten per ongeluk tegelijkertijd data proberen te versturen) heeft dat slechts corrupte data tot gevolg. Er kan nooit een hardwareconflict ontstaan waarbij een knooppunt probeert de datalijn hoog te trekken terwijl een ander knooppunt probeert deze laag te trekken. Er is een pull-up weerstand nodig omdat geen van de knooppunten de bus actief naar VCC kan trekken. **Figuur 4** toont de aansluiting van een 1-draads sensor op een Arduino UNO.

Naast de elektrische verbinding is een protocol nodig voor het gebruik van 1-Wire componenten. Gelukkig bevatten de meeste microcontrollers een UART die voor dit doel kan worden gebruikt, zonder dat daarvoor speciale hardware nodig is. Een geschikte schakeling is te zien in **figuur 5**.

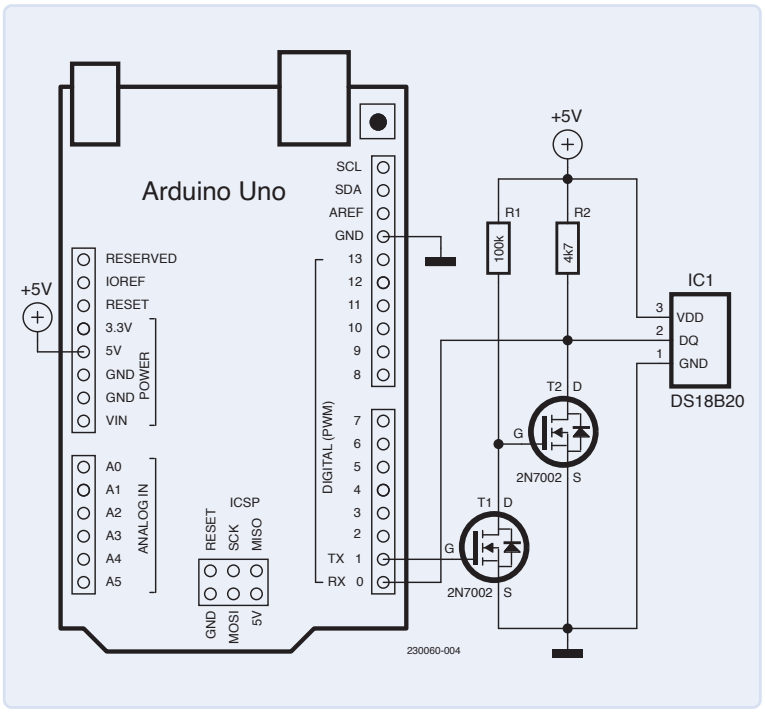
Voor de aansturing via UART heeft Analog Devices een nuttig artikel in de aanbieding [1]. Hoewel de UART kan helpen de CPU te ontlasten, is hij niet essentieel bij de huidige MCU's. Zelfs een kleine ATtiny heeft de middelen om via de 1-Wire bus aangesloten sensoren aan te spreken met een puur softwarematige bit-banging oplossing, waarvoor slechts één I/O-pin nodig is. De meeste bibliotheken die met de DS18B20 werken, gebruiken slechts één GPIO, waardoor het gebruik van een UART steeds minder voorkomt.

Identificatie van de sensor

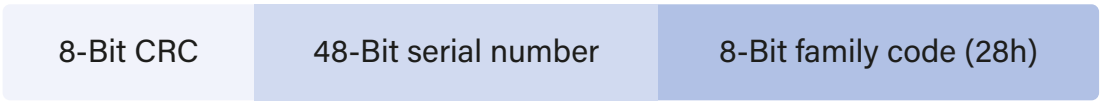
Wanneer meer dan één sensor op een gemeenschappelijke busdraad is aangesloten, moet elke sensor afzonderlijk kunnen worden geadresseerd, zodat slechts één sensor tegelijk naar de bus kan schrijven en zijn gegevens niet door een andere sensor kunnen worden verstoord of gecorrumpeerd. Daartoe heeft elke



Figuur 4. Schema van de Arduino UNO met een DS18B20.



Figuur 5. Schema van de DS18B20 1-Wire bus, aangesloten op de UART.



Figuur 6. Formaat van het DS18B20 UUID-bericht.

deelnemer op de 1-Wire bus een unieke, 64bit-identificatiecode (UUID). Deze bestaat uit een 8bit-code die de sensorfamilie aangeeft, een 48bit-serienummer en een 8bit-checksum (CRC) (**figuur 6**). Het serienummer van de sensor staat echter nergens op de sensor, maar is intern in ROM opgeslagen in de vorm van 8 bytes. Het kan softwarematig worden achterhaald met behulp van een zoekfunctie op de 1-Wire bus. Het zoekalgoritme is te vinden op de pagina van Analog Devices [2].

Voorbeeldcode voor Arduino

Op basis van het schema in figuur 4 is de opzet zeer eenvoudig gehouden. De sensor gebruikt een 5V-voeding zoals de Arduino UNO. De Arduino UNO GPIO, aangesloten op de data- of DQ-lijn, heeft een 4,7 kΩ pull-up weerstand nodig. De bibliotheek voor de 1-Wire bus die we hier gebruiken is de OneWire-bibliotheek van Paul Stoffregen [3]. De voorbeeldcode van de bibliotheek omvat ook het lezen van gegevens van de DS18B20-sensoren (**listing 1**), die we nu wat meer in detail kunnen bekijken.
Om de bibliotheek te gebruiken, moet deze eerst worden opgenomen met `#include <OneWire.h>` aan het begin van de sketch. Vervolgens wordt met `OneWire ds(10)` de 1-Wire bus-datalijn toegewezen voor gebruik met pin 10. Met `ds.search(addr)` wordt de volgende sensor op de bus gezocht. Wanneer alle sensoren zijn geïdentificeerd, wordt het zoeken beëindigd. De identificatie van elke gevonden sensor wordt opgeslagen in `addr`.
Nu de sensor is geïdentificeerd, kan ermee worden gewerkt. De eerste byte, `addr[0]`, bevat de code die de familie aangeeft. Dit kan worden gebruikt om te identificeren of een DS18B20, DS18S20, of DS1822 temperatuursensor is aangesloten, of dat het een ander type 1-Wire buscomponent is.
In regel 69 en volgende zien we:

```
ds.reset();
ds.select(addr);
ds.write(0x44, 1);
```

Met `ds.reset()` worden alle busdeelnemers éénmalig gereset Daarna wordt met `ds.select(addr)` de ontdekte sensor aangesproken. Om een temperatuurmeting van de sensor op te vragen is het noodzakelijk om eerst een meting te starten. Het betreffende commando is `ds.write(0x44, 1)`, waarbij `0x44` het in de sensor herkende commando is. Met de tweede parameter (hier `1`) wordt de I/O-pin actief hoog gemaakt om sensoren zoals de DS18S20 van parasitaire voeding te voorzien. Het resultaat van de temperatuurmeting is beschikbaar na 750 tot 1000 ms. In dit voorbeeld wordt een vertraging gebruikt om te wachten tot de verwerking voltooid is. Zodra de temperatuurmeting is voltooid en het resultaat klaarstaat, kan het uit de sensor worden opgehaald. De sensor wordt opnieuw aangesproken met `ds.select(addr)` en geïnstrueerd dat nu de in zijn scratchpad-geheugen opgeslagen waarde moet worden uitgelezen (`ds.write(0xBE)`). De negen bytes die hier zijn opgeslagen, inclusief de temperatuurmeting, worden nu uitgevoerd.
Met de DS18**B**20 is de temperatuur na uitlezing beschikbaar als een 16bit-waarde, bestaande uit twee registerbytes. Indien een sensor van het type DS18**S**20 wordt gebruikt in plaats van een DS18B20, worden de registerwaarden anders gewogen zodat hun positie in het register moet worden aangepast. Het codevoorbeeld houdt hiermee rekening door de ruwe waarde driemaal naat links te schuiven.
De DS18B20 voert de temperatuur uit met 12bit resolutie en heeft 750 ms nodig voor de conversie. Hij kan indien gewenst 11bit-, 10bit- of 9bit-waarden uitvoeren, wat de conversie navenant versnelt. Bij 9bit-resolutie is de waarde van de lage bits niet gedefinieerd, zodat deze op nul kunnen worden gezet.
De temperatuurmeting van de sensor is altijd in graden celsius; de omzetting naar graden fahrenheit moet je in software doen als dat nodig is.

DS18B20 – The Clone Wars

De DS18B20 is een populaire sensor die vaak deel uitmaakt van diverse sensorkits en microcontroller-starterkits. Deze sensoren zijn ook erg goedkoop; je kunt er al eentje kopen voor slechts 30 cent op bepaalde online-marktplaatsen. Distributeurs zoals Mouser of Farnell vragen echter meer dan 4 euro voor een enkele DS18B20.
De populariteit van deze sensor heeft het voor sommige malafide chipsbakkers de moeite waard gemaakt om hun eigen versies te ontwikkelen, die lijken op en zich schijnen te gedragen als de originele sensor, maar bij nadere beschouwing sterk kunnen afwijken van het origineel. Erkende distributeurs betrekken hun componenten van gecertificeerde leveranciers.
Om er zeker van te zijn dat je een authentieke sensor hebt, biedt de GitHub-pagina van Chris Petrich [4] Arduino-sketches om je sensor te testen, evenals meer informatie over de verschillende klonen en hun afwijkingen van het oorspronkelijke ontwerp.

Samenvattend

Een temperatuursensor kan eenvoudig op een microcontroller worden aangesloten met behulp van de 1-Wire DS18B20. Zoals we zagen, zijn er voor het Arduino-platform een heleboel bibliotheken en broncode-voorbeelden beschikbaar om het proces te stroomlijnen.

Met deze opzet is slechts één pin van de MCU nodig om met de sensor te communiceren, en kunnen meerdere sensoren worden aangesloten, zodat een kleine MCU met één vrije GPIO meerdere sensoren tegelijk kan ondersteunen.

230060-03

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.

Gerelateerde producten

- **Elektor 37-in-1 Sensor Kit (SKU 16843)**
  www.elektor.nl/16843
- **Cytron Maker UNO (SKU 18634)**
  www.elektor.nl/18634
- **MakePython ESP32 Development Kit (SKU 20137)**
  www.elektor.nl/20137

WEBLINKS

[1] Using a UART to implement a 1-wire-bus-master, Analog Devices: https://analog.com/en/technical-articles/using-a-uart-to-implement-a-1wire-bus-master.html
[2] 1-Wire zoekalgoritme, Analog Devices: https://analog.com/en/app-notes/1wire-search-algorithm.html
[3] OneWireLibrary van Paul Stoffregen op GitHub: https://github.com/PaulStoffregen/OneWire
[4] Chris Petrich, "Your DS18B20 temperature sensor is likely a fake, counterfeit, clone...," GitHub: https://github.com/cpetrich/counterfeit_DS18B20

Listing 1. DS18B20 Arduino-voorbeeld.

```
001        #include <OneWire.h>
002
003        // OneWire DS18S20, DS18B20, DS1822 Temperature Example
004        //
005        // http://www.pjrc.com/teensy/td_libs_OneWire.html
006        //
007        // The DallasTemperature library can do all this work for you!
008        // https://github.com/milesburton/Arduino-Temperature-Control-Library
009
010        OneWire ds(10);  // on pin 10 (a 4k7 resistor is necessary)
011
012        void setup(void) {
013          Serial.begin(9600);
014        }
015
016        void loop(void) {
017
018          byte i;
019          byte present = 0;
020          byte type_s;
021          byte data[9];
022          byte addr[8];
023          float celsius, fahrenheit;
024
025          if (!ds.search(addr)) {
026            Serial.println("No more addresses.");
027            Serial.println();
```

continued overleaf

```
028           ds.reset_search();
029           delay(250);
030           return;
031         }
032
033
034         Serial.print("ROM =");
035         for(i = 0; i < 8; i++) {
036           Serial.write(' ');
037           Serial.print(addr[i], HEX);
038         }
039
040         if (OneWire::crc8(addr, 7) != addr[7]) {
041             Serial.println("CRC is not valid!");
042             return;
043         }
044
045         Serial.println();
046         // the first ROM byte indicates which chip
047         switch (addr[0]) {
048
049           case 0x10:
050             Serial.println("  Chip = DS18S20");  // or old DS1820
051             type_s = 1;
052             break;
053
054           case 0x28:
055             Serial.println("  Chip = DS18B20");
056             type_s = 0;
057             break;
058
059           case 0x22:
060             Serial.println("  Chip = DS1822");
061             type_s = 0;
062             break;
063
064           default:
065             Serial.println("Device is not a DS18x20 family device.");
066             return;
067         }
068
069         ds.reset();
070         ds.select(addr);
071         ds.write(0x44, 1); // start conversion, with parasite power on at the end
072         delay(1000);       // maybe 750ms is enough, maybe not
073         // we might do a ds.depower() here, but the reset will take care of it.
074
075         present = ds.reset();
076         ds.select(addr);
077         ds.write(0xBE);         // Read Scratchpad
078
079         Serial.print("  Data = ");
080         Serial.print(present, HEX);
081         Serial.print(" ");
082         for (i = 0; i < 9; i++) {           // we need 9 bytes
083           data[i] = ds.read();
084           Serial.print(data[i], HEX);
085           Serial.print(" ");
086         }
087         Serial.print(" CRC=");
088         Serial.print(OneWire::crc8(data, 8), HEX);
```

```
089         Serial.println();
090         // Convert the data to actual temperature
091         // because the result is a 16-bit signed integer, it should
092         // be stored to an "int16_t" type, which is always 16 bits,
093         // even when compiled on a 32-bit processor.
094
095         int16_t raw = (data[1] << 8) | data[0];
096         if (type_s) {
097           raw = raw << 3; // 9 bit resolution default
098           if (data[7] == 0x10) {
099             // "count remain" gives full 12 bit resolution
100             raw = (raw & 0xFFF0) + 12 - data[6];
101           }
102         } else {
103           byte cfg = (data[4] & 0x60);
104           // at lower res, the low bits are undefined, so let's zero them
105           if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
106           else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
107           else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
108           // default is 12-bit resolution, 750 ms conversion time
109         }
110
111         celsius = (float)raw / 16.0;
112         fahrenheit = celsius * 1.8 + 32.0;
113         Serial.print("  Temperature = ");
114         Serial.print(celsius);
115         Serial.print(" Celsius, ");
116         Serial.print(fahrenheit);
117         Serial.println(" Fahrenheit");
118       }
```