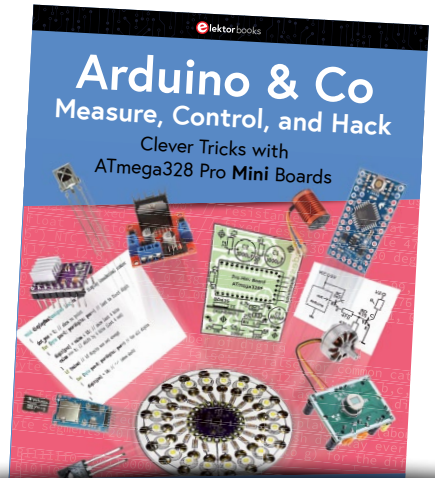
Infrageluid-recorder met de Arduino Pro Mini

een voorbeeldproject uit de Elektor-uitgave “Arduino & Co”

Robert Sontheimer (Duitsland)

Laten we iets geks doen: we gaan een Arduino Pro Mini (of compatibel board) combineren met een BMP180 luchtdruk- en temperatuursensor en een SD-kaartmodule en onderzoeken hoe we infrageluid gedurende een lange periode kunnen opnemen.



Noot van de redactie: *dit artikel is een gedeelte uit het 332 pagina's tellende Elektor-boek* Arduino & Co – Measure, Control, and Hack. *Het is opgemaakt en enigszins bewerkt zodat het past bij de conventies en paginaopmaak van Elektor Magazine. Omdat het een gedeelte is uit een grotere publicatie, kan dit artikel verwijzen naar passages elders in het boek. Auteur en redacteur hebben hun best gedaan om dat te voorkomen, en helpen graag bij vragen. De contactgegevens staan in het kader* ***Vragen of opmerkingen****.*

De BMP180 luchtdruksensor levert maximaal ongeveer 100 temperatuur- en luchtdrukwaarden per seconde. We kunnen dus een stereo-audiobestand genereren (in PCM-formaat met de extensie *.wav*) op basis van deze data, gedurende minuten, uren of dagen, waarbij we de luchtdruk opnemen op het linkerkanaal en de temperatuur op het rechterkanaal. We stellen de standaard afspeelfrequentie (sampling rate) bijvoorbeeld in op 44.100 Hz. Dit is een veelgebruikte waarde die ook wordt gebruikt voor audio-CD's en andere zaken.

Het bestand wordt dan later afgespeeld (bijvoorbeeld op een laptop of met een audiospeler) op 441 keer de normale snelheid; zo kunnen we infrageluid hoorbaar maken! Met de oorspronkelijke bemonsteringsfrequentie van 100 Hz bij opname kunnen we alle frequenties onder 50 Hz opnemen. Een infrasoon geluid van 10 Hz wordt 4,41 kHz als het wordt afgespeeld. 1 Hz wordt 441 Hz en zo vervolgens. Theoretisch zijn daaronder geen grenzen.

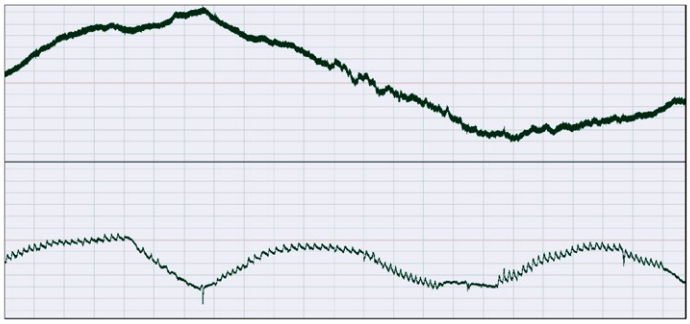
OK, toegegeven, de BMP180 is niet echt geschikt als zeer gevoelige microfoon. Hoewel hij al drukafwijkingen detecteert bij geringe hoogteveranderingen, moet infrageluid heel wat luider zijn om het te onderscheiden van de ruis bij het afspelen.

Het is echter veel interessanter om een audiobewerkingsprogramma te gebruiken om later naar de luchtdruk- en temperatuurgolfvormen te kijken – vooral als de opname meerdere dagen duurde. Het linkerkanaal toont ons dan de grafiek van de luchtdruk en het rechterkanaal die van de temperatuur.

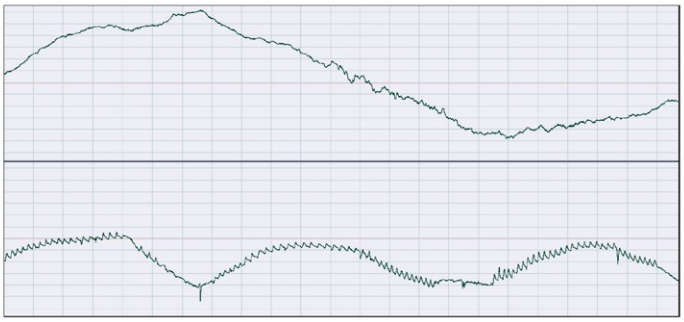
Als voorbeeld heb ik een opname bijna drie dagen laten duren, het resulterende .wav-bestand geopend met een audiobewerkingsprogramma en vervolgens beide kanalen afzonderlijk genormaliseerd en zo goed mogelijk geschaald naar het volledige bereik. De bovenste grafiek in **figuur 1** met de drukmetingen ziet er nog steeds wat vet en onregelmatig uit doordat elke pixel op de X-as uit tienduizenden metingen bestaat, en deze bevatten ook wat ruis en fluctueren dus in zekere mate.

Figuur 2 toont de resultaten na verwerking. De opname werd opnieuw bemonsterd en 4000 waarden werden gemiddeld tot één enkele waarde. De ruis is verdwenen en we kunnen de druk- en temperatuurcurven nu veel duidelijker zien.

Uit de grafieken kunnen we nu een paar dingen opmaken: bij de drukgrafiek zien we dat het iets winderiger werd gedurende de tweede helft van de meting, omdat daar meer kleinere fluctuaties optreden. Bij een werkelijke storm zouden de fluctuaties nog extremer zijn.



Figuur 1. Luchtdruk (boven) en temperatuur (onder) als funcetie van de tijd, vrijwel onbewerkt.



Figuur 2. Verloop van luchtdruk (boven) en temperatuur (onder) na bewerking.

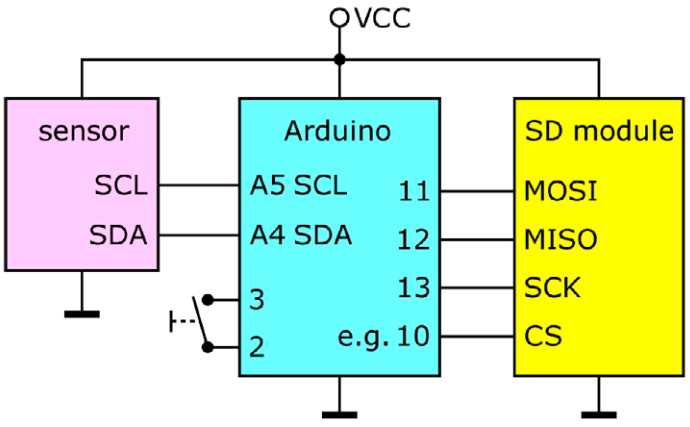
De temperatuurcurve onderaan toont duidelijk het dag/nacht-ritme. De bijna steeds optredende kleine piekjes zijn de schakelcycli van de thermostaat voor de verwarming van de kamer. Je zou kunnen tellen hoe vaak de thermostaat heeft in- en uitgeschakeld tijdens deze drie dagen. Verder zijn er twee neerwaartse spikes te zien toen het raam kort werd geopend.

Constructie

Om het project uit te voeren, hebben we nodig:

- 1 Arduino-board (16 MHz ATmega328)
- 1 BMP180 sensormodule
- 1 (micro)SD-kaartmodule
- 1 drukknop
- (jumper)draden
- 1 (micro)SD-kaart, maximaal 32 GB

De opbouw is vrij eenvoudig. Naast de BMP180-sensor moeten we de SD-kaartmodule en de drukknop voor het starten en stoppen van de opname aansluiten. Volg gewoon het aansluitschema van **figuur 3**. Gesoldeerde verbindingen zijn altijd het beste, maar we kunnen ook jumperdraden gebruiken. We hebben twee VCC-verbindingen nodig, maar de Pro Mini heeft maar één VCC-pin. Er bestaat echter een heel eenvoudige oplossing: omdat de BMP180 nauwelijks stroom verbruikt, kunnen we hem ook voeden vanuit een uitgang (bijvoorbeeld uit pin 9 in plaats van VCC). Dan hoeven we alleen de pin in `setup()` als uitgang te definiëren en deze hoog te maken.



Figuur 3. Bedrading van de BMP180-sensor, SD-kaartmodule, schakelaar en Arduino.

Sketch voor de infrageluid-recorder

Een groot deel van de sketch met de naam *13.9.ino* die voor dit project is geschreven, wordt gepresenteerd in de vorm van de 'partiele' listings **1a** tot en met **1h**, samen met de bespreking hieronder. Het volledige programma is veel langer dan de som van de hier gegeven listings. Het programma en de subprogramma's die hieronder worden genoemd, staan in het softwarebestand dat voor het boek is uitgebracht en dat gratis kan worden gedownload van de Elektor Books support-website [1]. Ga op de webpagina naar *Downloads*. Om dit artikel te kunnen volgen, moet je *13.9.ino* ter naslag bij de hand hebben.

Listing 1a: omdat we de luchtdruksensor combineren met de SD-kaartmodule, moeten we drie bibliotheken installeren: de I²C-interface voor de luchtdruksensor, de SPI-interface voor de SD-kaartmodule en de speciale SD-kaartfuncties.
Vervolgens definiëren we twee pinnen waarop we een drukknop kunnen aansluiten om de opname te starten en te stoppen. Pin 3 wordt laag gemaakt en dient als massa voor de knop. Dit is erg praktisch, omdat de twee pinnen van de knop dan vlak naast elkaar zitten. We kunnen natuurlijk ook de echte massa (GND) gebruiken in plaats van pin 3.
De andere definities en variabelen, evenals alle programma-onderdelen van de luchtdruksensor en de SD-kaart sketches die in het boek worden besproken, worden hier niet afgedrukt. We zijn nu alleen geïnteresseerd in recorder-specifieke zaken.

Listing 1a

```
#include <Wire.h>     // library for I2C interface
#include <SPI.h>      // library for SPI interface
#include <SD.h>       // library for SD card
#define record_pin 2 // switch pin to ground for
                      //   recording
#define low_pin 3     // Set output pin to LOW
                      //   (used as GND for button)
…
```

Listing 1b: hier hebben we de instellingen van de recorder. De variabele `oversampling` bepaalt ook het aantal drukmetingen. Om 100 waarden per seconde op te nemen, moet deze instelling 0 zijn. `audio_rate` daarentegen bepaalt hoeveel waarden per seconde worden geleverd als we de opname later afspelen als een audiobestand. `max_file_size` bepaalt de maximale opnamelengte. "10 minutes" verwijst echter naar de afspeeltijd – de corresponderende opnametijd is enkele dagen. We kunnen deze waarde echter

bijna willekeurig aanpassen. Als de maximale bestandsgrootte is bereikt, wordt het bestand gesloten en wordt een nieuwe opname gestart.
De variabele `duration` bepaalt de vertraging tussen de metingen. De tijden zijn hier echter 500 µs langer omdat we hier niet met delays werken, maar met een vaste klokfrequentie gedurende welke ook de Wire-instructies voor de temperatuur- en drukmeting worden uitgevoerd.

Listing 1b

```
// 0 -> 1x, 1 -> 2x, 2 -> 4x, 3->8x
char oversampling = 0;
// sampling rate of the output audio file
long audio_rate = 44100;
// 10 minutes' playback time (at 44,100 Hz)
unsigned long max_file_size = 105840044;
// time (µs) according to oversampling
int duration[4] = ;
// time for temperature measurement
int t_duration = 5000;
…
```

Listing 1c: na de speciale variabelen voor de temperatuur- en drukberekening, die we hier weer hebben overgeslagen, zijn deze variabelen later nodig voor de recorder.

Listing 1c

```
// system time at which the data are available
unsigned int next_time;

long zero_value;  // pressure zero line
long record_value;// pressure value to save
long zero_temp;   // temperature zero line
long temp_value;  // temperature value to save

// indicates whether recording is activated
boolean must_record = false;
// indicates if recording is currently running
boolean is_recording = false;
// indicates whether pushbutton is pressed
boolean pressed = false;
// actual file size
unsigned long file_size;
// actual file number
unsigned int file_number = 0;
// counts how long button is not pressed anymore
byte state_counter = 0;
File wave_file;   // recording file
```

Listing 1d: `setup()` begint met de definitie van de pinnen voor de drukknop. Dankzij de geactiveerde interne pull up-weerstand hebben we geen extra onderdelen nodig.

Listing 1d

```
void setup() {
  // switch pin on input with pull-up
  pinMode(record_pin, INPUT_PULLUP);
  // low pin on output ...
  pinMode(low_pin, OUTPUT);
  // ... and LOW [...]
  digitalWrite(low_pin, LOW);
  …
}
```

Listing 1e: in `loop()` hebben we nu alleen het afwisselende afroepen van de temperatuur- en drukmetingen, want om geen tijd te verliezen wachten we daar niet telkens met een delay op de gegevens, maar gebruiken we de wachttijd voor alle andere taken. Voor de temperatuurmeting wordt de functie `calculate()` aangeroepen terwijl we wachten op het resultaat. Voor de drukmeting roepen we de functie `recording()` aan.

Listing 1e

```
void loop() {
  get_t(); // read temperature
  get_p(); // read pressure
}
```

Listing 1f: nu de `calculate()`-functie. De werkelijke temperatuur en werkelijke druk worden eerst berekend uit de gemeten waarden, voor zover de druk al gemeten is. Ik heb dit nu weer afgekort met "`...`", want ik weet zeker dat niemand deze berekeningen (met de vele formuleregels uit de datasheet van de BMP180) nog eens wil lezen.
Vervolgens controleren we of de nul-assen voor druk en temperatuur al gedefinieerd zijn. Als dit niet het geval is, wordt de eerste meting nu gedefinieerd als de basiswaarde. Als er daarentegen nog geen meting is gedaan, verlaten we de functie volledig.
Nu worden de waarden voorbereid voor opname. Hiervoor trekken we de nul-waarde van de druk af van de gemeten waarde. Als de waarde wordt overschreven, wordt deze begrensd tot het toegestane bereik van –32.768 tot 32.767. Daarna doen we hetzelfde met de temperatuur. Beide waarden zijn nu klaar voor opslag.

Listing 1f

```
// calculate temperature (in tenth degree) and
// pressure (in pascals)
void calculate() {

  // if pressure has already been measured before
  if (p) {
    …
  }
```

```
  // if zero value has not been determined yet
  if (!zero_value) {
    // cancel if no measurement has been taken yet
    if (!pressure) return;
    // actual value as zero line
    zero_value = pressure;
    // actual value as zero line
    zero_temp = b5;
  }

  // pressure value for recording
  record_value = pressure - zero_value;

  // max value if clipping
  if (record_value > 32767)
    record_value = 32767;
  // negative clipping
  else if (record_value < -32768)
    record_value = -32768;
  // temperature value for recording
  temp_value = b5 - zero_temp;
  // max value if clipping
  if (temp_value > 32767) temp_value = 32767;
  // negative clipping
  else if (temp_value < -32768) temp_
  value = -32768;
}
```

Listing 1g: in de functie `recording()` worden eerst twee voorwaarden gecontroleerd: of de opname zou moeten lopen (omdat deze is ingeschakeld en ook de maximale bestandsgrootte nog niet is bereikt) en of de opname daadwerkelijk loopt. Deze twee verschillende situaties resulteren nu in vier mogelijkheden. In het eerste geval zou de opname moeten lopen, maar loopt deze nog niet.

Listing 1g

```
void recording()  { // recording function
  // if recording should run
  if (must_record && file_size < max_file_size) {
    // if recording is not running yet
    if (!is_recording) {
      …
    }
  }
}
```

Listing 1h: hier wordt een nieuwe opname gestart. Hiervoor wordt het bestandsnummer verhoogd in de `do-while` lus totdat dit (samen met de verdere tekst) resulteert in een bestandsnaam die nog niet bestaat. Er wordt dan een nieuw bestand geopend met de corresponderende bestandsnaam, en een bericht wordt serieel uitgevoerd. In het geval van een fout wordt er een tweede bericht uitgevoerd en wordt het proces gestopt met behulp van een eindeloze lus.

Listing 1h

```
// search for the first not yet used
// recording number
do {
  file_number++; // next number (starting with 1)
} // repeat with next number while
   //file already exists

while (SD.exists("FILE_" + String(file_number)
       + ".wav"));
Serial.println("Recording no."
               + String(file_number)
               + " is started!");
wave_file = SD.open("FILE_"
                    + String(file_number)
                    + ".wav", FILE_WRITE);

if (!wave_file) { // if file could not be opened
   Serial.println
      ("Error: File could not be opened!");
// do not continue (infinite loop)
    while (true);
}
```

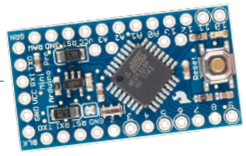
Daarmee is de ruimte voor de programmabespreking in dit artikel verbruikt. Gelukkig is de rest van het programma voor de infrageluid-recorder net zo goed gedocumenteerd als de hier getoonde listings 1a tot en met 1h.

Gebruik

Aan het begin van de sketch kunnen we drie variabelen instellen:

- `oversampling` moet op 0 staan om de hoogste opnamefrequentie van 100 Hz in te stellen;
- `audio_rate` specificeert de latere afspeelfrequentie. Hier kunnen we ook een waarde kleiner dan 44.100 gebruiken, om de tijdcompressie (44.100 / 100 = 441) niet zo hoog te laten worden. Gebruikelijke waarden zijn 8.000, 11.025, 16.000, 22.050, 32.000, 44.100 en 48.000. Niet elke speler kan andere frequenties spelen;
- met `max_file_size` definiëren we de maximale bestandsgrootte. 105.840.044 komt overeen met een afspeeltijd van 10 minuten bij 44.100 Hz, bij een opnametijd van meer dan drie dagen.

Vervolgens moeten we de voltooide opname 'normaliseren', dat wil zeggen het volume van het signaal opschalen zodat het volledige bereik wordt gebruikt, maar zonder het maximum te overschrijden. Zonder normalisatie zal het signaal niet luid genoeg zijn. We zouden natuurlijk een versterkingsfactor kunnen toevoegen tijdens de opname, maar we weten niet van tevoren hoeveel de temperatuur en luchtdruk zullen variëren tijdens de opname. Normalisatie kan met een paar kliks worden gedaan met audiobewerkingsprogramma's. Dit moet afzonderlijk worden gedaan voor beide kanalen. Met zo'n programma kunnen we de opname vervolgens



in grafische vorm bekijken. Het linkerkanaal toont het verloop van de luchtdruk tijdens de opname, terwijl het rechterkanaal de temperatuur toont.

Weerrecorder

Als we de oversampling-waarde veranderen in 3, zullen er slechts 32 (in plaats van 100) waarden per seconde worden opgenomen, wat nog steeds veel meer is dan we nodig hebben voor weer-opnamen. We zouden ook de hoogtemeter (uit paragraaf 13.8 van het boek) kunnen combineren met de SD-kaartfuncties, zodat we ook voor elke seriële uitgang een corresponderende tekstregel naar een bestand kunnen schrijven. Op die manier krijgen we de weeropname als een tekstbestand.

Je ziet dat er dus talloze mogelijkheden zijn voor wat je kunt doen, en de afzonderlijke hoofdstukken in het boek bieden – naast de concrete projecten – veel informatie en suggesties die je zouden moeten helpen om zoveel mogelijk van je eigen ideeën te realiseren. Helaas zijn we hiermee aan het einde gekomen van dit artikel; ik wens alle lezers veel plezier met meten, controleren en hacken!

vertaling: Willem den Hollander – 230393-03

WEBLINK

[1] Software-archief bij dit boek: http://www.elektor.nl/20243

Vragen of opmerkingen?

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

Over de auteur

Robert Sontheimer was er meteen bij toen de eerste thuiscomputers zo'n 40 jaar geleden onze huiskamers binnenkwamen, met de ZX81 en C64. Destijds bouwde hij een plotter om tot een scanner, naast andere eigenzinnige en originele ideeën. Tegenwoordig gebruikt hij Arduino Pro Mini's om complete CNC-lasermachines te besturen en heeft hij zelfs een bijpassend afzuigsysteem uitgevonden: zijn 'zelfvernieuwend wc-papier-filter.' In zijn kantoor heeft hij een magneet die al jaren zweeft – natuurlijk onder besturing van een Pro Mini.

Gerelateerde producten

- **Robert Sontheimer, Arduino & Co., Measure, Control, and Hack, Elektor 2022**
  boek: www.elektor.nl/20243
  e-boek: www.elektor.nl/20244