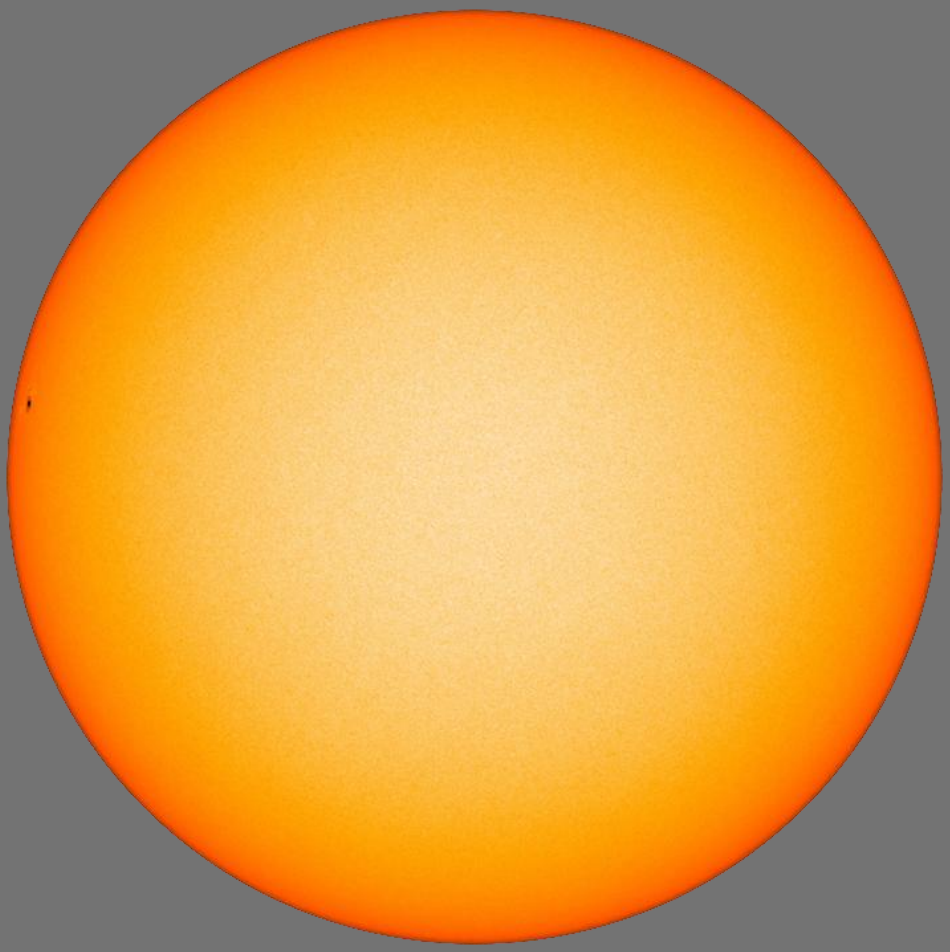
Zonlicht meten

Deel 4



Henk Siewert
SWT Observatory

Colofon

Versie 001, 23 September 2019

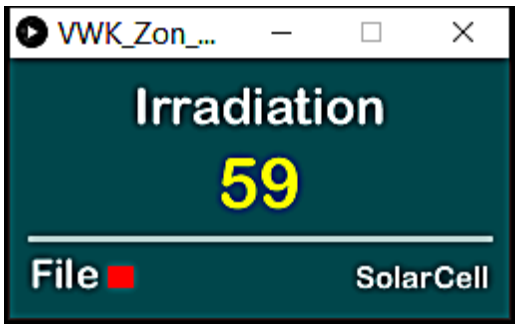
In deze publicatie beschrijf ik mijn hobby activiteiten.
Dat alles zonder enige pretentie van correctheid en volledigheid.
Hoewel ik mijn best doe om alles zo goed mogelijk te beschrijven
is dit alles hobbywerk en u kunt er dus ook geen enkel recht op baseren.
U bent zelf verantwoordelijk voor het gebruik, en de gevolgen daarvan,
van wat hier is beschreven.
Gebruik uw verstand en neem geen onnodige risico's.
Dan houden we het leuk.

Reacties zijn welkom.
Kijk daarvoor op de SWT Observatory website:
http://www.swtobservatory.nl

Zonlicht meten - Deel 4

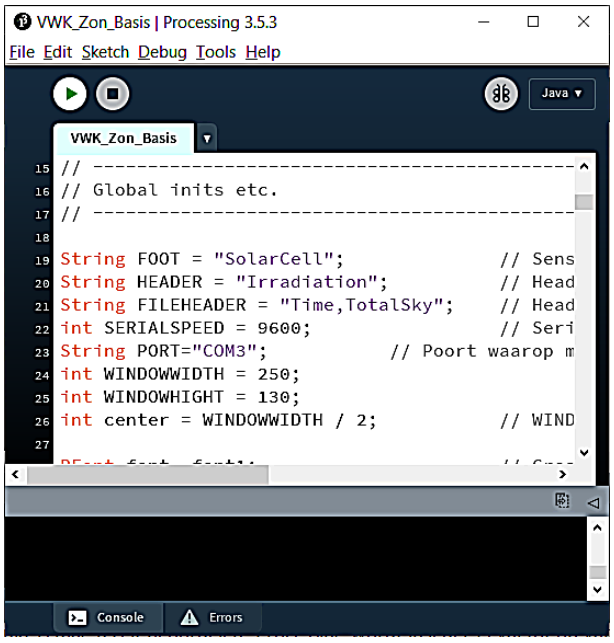
Henk Siewert

Het oog wil ook wat



Deze keer beginnen we met een afbeelding.
Dat maakt gelijk duidelijk waar we het in deze aflevering van onze 'continuing story' over gaan hebben.
Tot nu toe werden de waarden van onze metingen weergegeven in de seriële monitor van het programmeer omgeving (IDE) of in bijvoorbeeld TeraTerm.
Erg mooi ziet zo iets er nou niet echt uit.
En, zoals een 'oud gezegsweis' zegt: "Het oog wil ook wat". Ja toch, niet dan?
Om aan die wens te voldoen gaan we gebruikmaken van Processing. Processing is een grafisch georiënteerde taal gebaseerd op Java. Oeps, nou niet meteen in een kramp schieten. U hoeft helemaal geen moeilijke en soms zeer vreemd lijkende constructies te leren en / of te onthouden.
Als u een LaunchPad of een Arduino kunt programmeren, kunt u ook een Processing programma maken. Want weet u, zonder Processing was er geen LaunchPad- of Arduino IDE geweest.
Wat vertel ik u nou? Ja, echt waar hoor. Om dat te bewijzen gaan we even terug in de tijd.

Processing

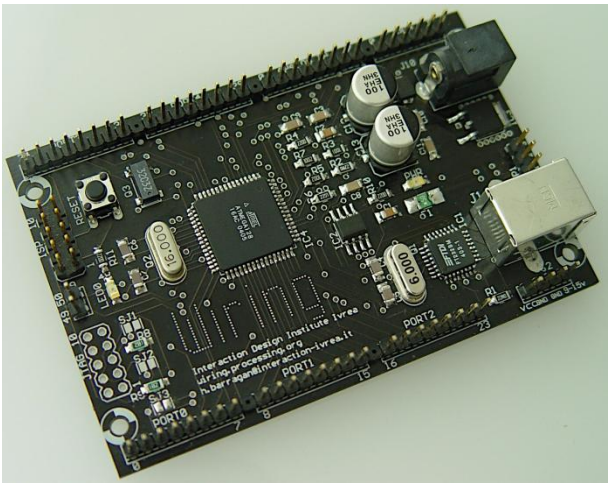


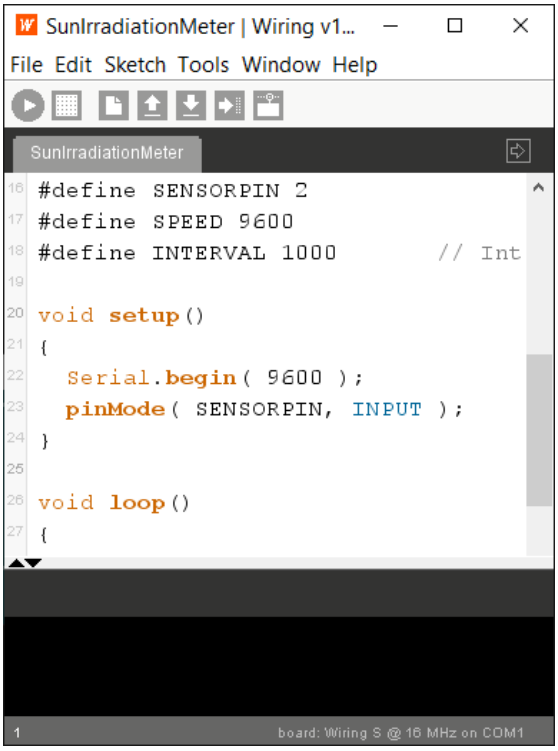
In 2001 begonnen Casey Reas en Ben Fry aan het MIT[i] met een project om het voor kunstenaars een stuk makkelijker te maken om hun ideeën in soft- en hardware om te zetten.
Het belangrijkste aan het project was het maken van een IDE waar je geen uitgebreide cursus voor hoefde te volgen om het te kunnen gebruiken. En het vereenvoudigen van de gebruikte programmeertaal, in dit geval Java, en de daarbij behorende 'tool chain' met compilers, linkers enz., zodat ook niet technische artiesten er mee konden werken. Of dat gelukt is mag u zelf beoordelen. Processing wordt in ieder geval tot op de dag van vandaag veel gebruikt[ii].

Wiring

In 2003 realiseerde Hernando Barragán, toen studerende aan het Interaction Design Institute Ivrea[iii] , dat het handig zou zijn om niet alleen software te gebruiken voor kunstuitingen maar dat hardware daar een belangrijke rol bij kon spelen. Maar dan moest

die hardware net zo makkelijk te programmeren zijn als software gemaakt met Processing. Op basis van de Processing IDE ontwierp hij een programmeeromgeving, taal (gebaseerd op Java), en een tool chain die uiteindelijk resulteerde in Wiring. Wiring bestond uit bij elkaar behorende software en hardware die weer gebruik konden maken van de mogelijkheden van Processing. Zoals u





kunt zien lijkt het IDE als twee druppels water op het IDE van Processing.

Het eerste, publiekelijk beschikbare, Wiring bordje was gebaseerd op de AVR ATMega128, de voorloper van de ATMega328 die nu in de Arduino Uno zit.
Het Wiring IDE en bordje hebben model gestaan voor de Arduino IDE en bordjes.
De laatste versie van Wiring kan nog steeds gedownload worden en is prima te gebruiken om bepaalde Arduino bordjes te programmeren[iv].

Java

In tegenstelling tot de programmeertaal van Wiring en Arduino, allebei een C++ dialect, is de taal van Processing gebaseerd op Java[v] . Processing heeft het gebruik van Java op dezelfde manier vereenvoudigd als Wiring en Arduino met C++ hebben gedaan. Of eigenlijk is het andersom. Wiring en Arduino hebben Processing nagedaan. Maar dat maakt het voor ons een stuk eenvoudiger om in Processing te programmeren. Dus laten we er ons voordeel er mee doen.

Processing structuur

De opbouw van een Processing programma, het zal u niet vebrazen, heeft in principe de zelfde structuur als een Arduino programma.

```
/* Hallo World */

PFont font;                                   // Font object
String data = "Hallo World";                  // Text to be printed

void setup()
{
 size(400, 250);                              // Window size
```

```
  font = createFont( "Arial Rounded MT Bold", 48 );          // Create font
}

void draw()
{
  background( 0, 0, 0 );          // Window background color
  fill( 255, 255, 255 );          // Text color
  textFont( font, 36 );           // Text size
  text( data, 100, 100 );         // Print text
}
```

Zo, gelijk maar even het obligate “Hallo World” programma gemaakt.
Zo als u kunt zien bestaat het programma uit twee delen. Namelijk de functie setup() en draw().
setup() is het zelfde als bij een Arduino-programma en heeft ook dezelfde functie.
Waar in een Arduinoprogramma loop() staat, staat bij een Processingprogramma de functie draw().
draw() is net als loop() een functie die alles wat er in staat steeds weer herhaalt.
OK, laten we de code maar eens stap voor stap doornemen.

```
/* Hallo World */

PFont font;                                                  // Font object
String data = "Hallo World";                                 // Text to be printed
```

Alles dat tussen /* en */ staat wordt door het programma genegeerd. Dat geldt ook voor tekst achter //. Dit biedt ons de mogelijkheid om commentaar en aanwijzingen in de code op te nemen.
Met PFont font; // Font object wordt plaats ingeruimd voor het gebruik van een lettertype.
String data = "Hallo World";   // Text to be printed maakt en vult de variabele data met tekst.

```
void setup()
{
  size(400, 250);                                            // Window size
  font = createFont( "Arial Rounded MT Bold", 48 );          // Create font
}
```

size(400, 250); bepaalt de grootte van het venster waar we bepaalde gegevens naar toe zenden om weer te geven.
font = createFont("Arial Rounded MT Bold", 48); maakt in het geheugen een tekst font aan van 48 punten.

```
void draw()
{
  background( 0, 0, 0 );          // Window background color
  fill( 255, 255, 255 );          // Text color
  textFont( font, 36 );           // Text size
  text( data, 100, 100 );         // Print text
}
```

In draw() bepalen we eerst de achtergrondkleur van het venster in RGB. 0,0,0 is zwart.
Daarna bepalen we de kleur van de tekst, ook in RGB, 255,255,255 is wit.
Met textFont geven we de grote van de tekst aan in punten (points). Deze punten zijn Amerikaanse computer punten. Heeft dus niets te maken met de punten die vroeger in de grafische industrie werden gebruikt. U weet wel, 12 punten in een Augustijn (ook wel Cicero genoemd). En de Augustijn was dan weer afgeleid van de duim, die weer was afgeleid van de Franse koningsvoet. In die wereld was 1 punt 0,376 mm. En daar moesten we dus mee rekenen. In het duodicimale (12 tallige[vi]) stelsel. Misschien is het daarom dat veel zetters en drukkers zo goed konden overstappen naar de computer en overweg konden met hexadecimaal? Die afwijking van het decimale hadden ze al ingebakken.
Afijn, we gaan verder met het afdrukken (wat is dat toch met die drukkers en computers?) van de tekst in de variabele data. Dat doen we met de opdracht :
`text( data, 100, 100 );// Print text` 100, 100 geeft de plaats van de tekst in het venster aan. 100 pixels van links en 100 pixels van boven.

Als u vervolgens op de knop  met de tekst “Hallo World”. drukt zal, als alles goed gaat een venster verschijnen Missie geslaagd.

Heel leuk natuurlijk, zo’n tekstje. Als ik u was zou ik daar eens mee gaan experimenteren. Print uw eigen tekst in uw eigen kleur, in een ander font in een ander venster enz.
U kunt niets kapot maken. Het ziet er misschien alleen wat leuker (vreemder) uit.
Maar na deze spielerei gaan we natuurlijk aan het serieuze werk beginnen: gegevens via de seriële poort binnen halen en die weergeven in een venster.

OK, voor zover de basis. Allemaal leuk, maar wat we eigenlijk willen is de gegevens van onze meteo-apparatuur weergeven. Daarvoor moeten we contact maken tussen ons programma op de computer en de apparatuur waarvan we de meetgegevens op het scherm willen zien. In de meeste gevallen maken we daarvoor gebruik van een USB-aansluiting. In de programmeurwereld spreken we dan meestal van een seriële-aansluiting. Mocht u een ‘oude’ computer hebben met een ‘ouderwetse’ seriële (RS232C) aansluiting, dan verwijs ik u graag naar een artikel dat ik heb gepubliceerd in PC-Active. U kunt het artikel en de software downloaden van de website[vii].

Om met onze apparatuur in Processing te kunnen communiceren moeten we via de USB-aansluiting, meestal een virtuele com-poort[viii], verbinding maken. In de meeste programmeertalen is het een heel gedoe om dat voor elkaar te krijgen. Maar omdat Processing is gemaakt voor mensen die meer geïnteresseerd zijn in beeld dan in de techniek van het programmeren, hebben ze het ons juist heel makkelijk gemaakt.
Om contact te maken met de USB-aansluiting vier regels voldoende:

```
import processing.serial.*;      // Seriële bibliotheek importeren
Serial serialPort;               // Naam voor seriële poort
```

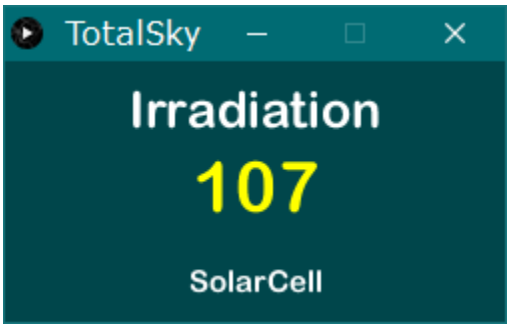
En in de setup() functie:

```
serialPort = new Serial( this, COMPOORT, SERIALSPEED );     // Open de serial port
serialPort.bufferUntil( BUFC );                             // Buffer until \nl
```

Wat dan nog overblijft is de gegevens binnenhalen en op het scherm plaatsen.
Als we er even van uitgaan dat er per meting 1 gegeven op de poort binnenkomt dat wordt afgesloten met een regeleindecode (\nl) wordt het binnenhalen gedaan met:

```
// ----------------------------------------------------------------------------------------------
// Starts when data on serial port - Is automatic interrupt
// ----------------------------------------------------------------------------------------------
void serialEvent( Serial port )
{
  data = port.readStringUntil( BUFC );  // Read data in serial buffer upto and including BUFC
  data = trim( data );                  // Clean-up string
  gvalue = true;                        // Flag data received
}
// ----------------------------------------------------------------------------------------------
```

Uiteraard komt er bij het maken van een compleet programma dan altijd nog de nodige code bij om het geheel goed te laten werken. Niet alleen moet de waarde worden binnengehaald maar ook moet de binnengehaalde waarde, meestal in de vorm van een ‘string’, worden opgeschoond en bewerkt. De totale listing is een beetje lang geworden om hier af te drukken. Maar de code is natuurlijk van de website te downloaden. Ik kan u wel het resultaat laten zien.



Als u de code download om te gebruiken wil ik wel nog even uw aandacht vestigen op de COMPOORT variabele. In de code staat die op “COM3” omdat dat de poort is waarop de sensor in mijn systeem is aangesloten. Maar ik weet natuurlijk niet welke com-poort u gebruikt. Dus moet u die zelf even aanpassen. Als sensor het ik de al besproken zonnecel gebruikt.

Als font heb ik "Arial Rounded MT Bold" gebruikt. Maar ik heb begrepen dat dit font op de nieuwere Windows-systemen niet meer wordt meegeleverd. Het font is van de website de downloaden zodat u het zelf kan installeren. Of u kunt een ander font kiezen dat wel op uw systeem beschikbaar is.
De serialEvent() functie zit in Processing ingebouwd. Het is een zogenaamde interrupt die tijdens de uitvoering van het programma constant kijkt of er gegevens op de com-poort worden aangeboden. Als dat zo is gaat hij die gegevens ophalen en geeft vervolgen het programma weer vrij om verder de gegevens te verwerken.

Verder heb ik zoveel mogelijk commentaar in de code opgenomen. Deels in het Engels en deels in het Nederlands. Het programma is een combinatie van al bestaande code en aanpassingen speciaal voor dit artikel. En ja, programmeren gaat nu eenmaal in het Engels. Dat betekent dat je gedachtetrein op een gegeven moment overschakelt op Engels. Maar als er vragen zijn kunt u altijd een mail sturen.
De code, afbeeldingen en andere informatie kunt u vinden op:
http://www.swtobservatory.nl

[i] https://www.mit.edu/
[ii] https://processing.org/
[iii] https://en.wikipedia.org/wiki/Interaction_Design_Institute_Ivrea
[iv] http://wiring.org.co/
[v] https://www.java.com/nl/
[vi] https://nl.wikipedia.org/wiki/Typografische_eenheid
[vii] http://swtobservatory.nl/pcactive/pca305/pca305.htm
[viii] http://janaxelson.com/usb_virtual_com_port.htm