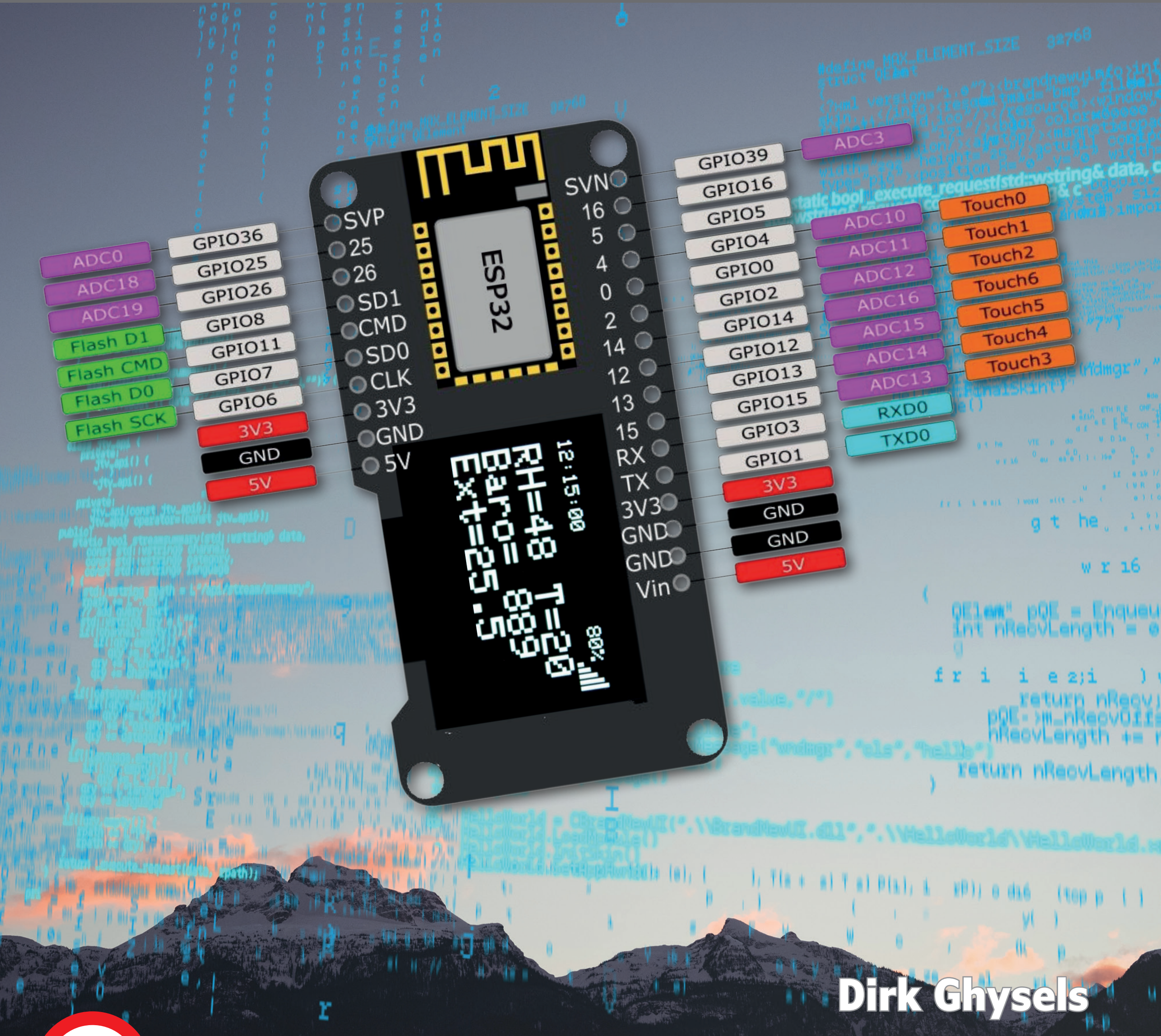
MET ARDUINO SOFTWARE

ESP32 PROGRAMMEREN VOOR BEGINNERS



Dirk Ghysels

elektor

LEARN DESIGN SHARE

Inleiding

Microcontrollers zijn geïntegreerde schakelingen die gebruikt worden om elektronische apparatuur aan te sturen. Een microcontroller bestaat minimaal uit een processor, geheugen en I/O. Bijna alle moderne apparaten bevatten een microcontroller, denk aan Tv-toestellen, wasmachines, telefoons, wagens, robotgrasmaaiers, afstandsbedieningen etc.

Arduino is een de-facto standaard geworden voor hobbyisten die microcontrollers willen (leren) gebruiken en toepassingen er willen voor ontwikkelen. Arduino is volledig opensource: de software is gratis te gebruiken en de hardware is betaalbaar. Compatibele hardware is goedkoop tot zeer goedkoop.

Figuur 1: Arduno UNO (foto: Arduino.cc)

De Arduino software vormt een geïntegreerde *programmeeromgeving*. Deze werkt op de PC en heeft een *editor* om programma's in te voeren, een *compiler* om de programma's te vertalen naar een vorm die de controller kan verwerken en een uploader die het gecompileerde programma overbrengt naar het inwendige geheugen van de controller.

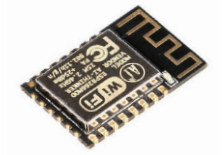
De Arduino hardware bestaat uit controller-bordjes (*boards*) en uitbreidingsbordjes (*shields*). Op een board zit de controller, een aansluiting naar de PC, USB of serieel, voor het uploaden van software en connectoren voor het aansluiten van in- en output. Arduino levert boards met 8-bit en 32-bit controllers van Atmel/Microchip.

De ESP8266 en ESP32 zijn producten van Chinese chipmaker Espressif. Op die boards zit o.a. een 32-bit controller (double-core voor de ESP32), RAM, flash-ROM, seriële in/uitgangen zoals I^2C, SPI, CAN, …, timers, een RTC, WiFi en bluetooth (alleen ESP32). Er bestaan heel wat bordjes met een ESP-chip, en Espressif heeft er met behulp van de open-source gemeenschap voor gezorgd dat deze samenwerken met de Arduino-software.

Dit boek beschrijft de Arduino programmeeromgeving en hoe deze werkt met goedkope ESP-ontwikkelbordjes.

Hoofdstuk 1 • ESP8266 en ESP32

ESP8266



Figuur 2: ESP8266

De ESP8266 is een WiFi-chip en bevat een volledige TCP/IP-stack en dus geschikt voor internetcommunicatie. De chip kan gebruikt worden als een standalone microcontroller systeem, of als een uitbreidingsmodule die WiFi-toegang verleend aan andere systemen. De chip wordt meestal geleverd als een module met ingebouwde antenne. Er bestaan verschillende varianten, de voornaamste kenmerken zijn:

- 32-bit CPU: Tensilica Xtensa LX106
- Kloksnelheid: 80 MHz
- Werkspanning: 3.3V (tussen 2.5V en 3.6V)
- 64 kB programmageheugen, 96 KB datageheugen
- Extern flashgeheugen, max. 16 MB
- WiFi IEEE 802.11 b/g/n
- WEP en WPA/WPA2 authenticatie
- Volledige TCP/IP stack
- 1 ADC-convertor, 10 bit
- 17 GPIO's (digitale input/output pinnen)
- 2xSPI, 1xI^2C, 2xI^2S, 2xUART

ESP32

De ESP32 is een vernieuwde en uitgebreide versie van de ESP-chips. De voornaamste verschillen zijn:

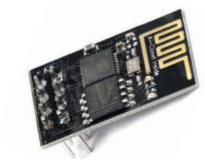
- Double-core 32-bit CPU LX6
- Kloksnelheid: 80 of 160 MHz
- 36 GPIO's (digitale input/output pinnen)
- 2x SPI, 2x I^2C, 2x I^2S, 2x UART, 1x CAN
- 10 Touch sensoren, temperatuur sensor
- Bluetooth 4.2
- 18ADC- kanalen, 12 bit

Ontwikkelbordjes met ESP-controller[1]

De ESP-01 is het oudste bordje voor de ESP-8266. Het bevat de ESP-chip, flash-geheugen, een LED, een antenne en een 2x4 DIL-connector voor aansluiting van voeding (3.3V),

1 Foto's: www.aliexpress.com

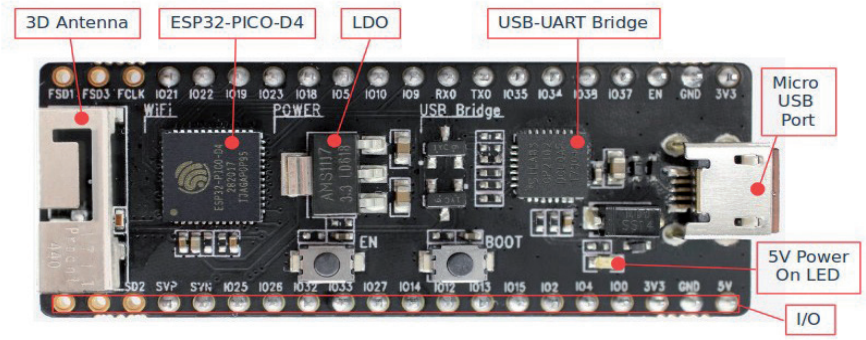
UART, reset en GPIO's[2]. Het bordje wordt geprogrammeerd met een PC via de USB poort en een UART/USB omzetter. Een UART/USB omzetter is de verbinding tussen de USB-poort van de PC en de seriële poort (of UART poort) van de controller.



Figuur 3: ESP-01

Uitgebreidere bordjes hebben een USB-aansluiting aan boord, een USB/UART omzetter, een voedingsmodule en meer aansluitpunten voor input en output. Hierna behandelen we een aantal bekende ESP-producten en hun bijzonderheden.

De **ESP32-PICO-KIT v4** is een ontwikkelbordje van Espressif en heeft een 40 MHz klok, 4 MB ram en een USB-aansluiting. Op de pinheaders aan de zijkanten zitten de in- en output-aansluitingen en de voedingsspanningen van 5V en 3,3V. Het bordje kan gevoed worden via de USB-poort of via de voedingsaansluitingen op de pinheaders. Let op: **Sluit nooit twee voedingen gelijk aan, dit kan het bordje beschadigen.** Op het bordje zit een LED die aangeeft of er een voedingsspanning aanwezig is, een resetknop (knop EN) en een Boot-knop. De Boot-knop moet je volgens Espressif indrukken tijdens het programmeren, het werkt ook zonder. Dit bordje is verkrijgbaar bij Elektor. De aansluitgegevens vind je in de bijlage.



Figuur 4: ESP32-PICO-KIT

Zoals gezegd worden ESP-chips in China geproduceerd, op Chinese webshops zijn dan ook veel ontwikkelbordjes te verkrijgen. De documentatie bij die bordjes is soms schaars, maar ze zijn zeer goedkoop (2,50 tot 5 € voor een ESP8266 bordje en 5 tot 10 € voor een ESP32) en de kwaliteit is in orde. Enkele voorbeelden:

Op het **Wemos Lolin ESP32 OLED** bordje zit een ESP32 met 80 MHz klok, een USB-poort voor voeding en programmering, 4 MB flash en een OLED schermpje Een LED op het bordje

2 General Purpose Input/Output poort

Hoofdstuk 2 • De Arduino programmeeromgeving

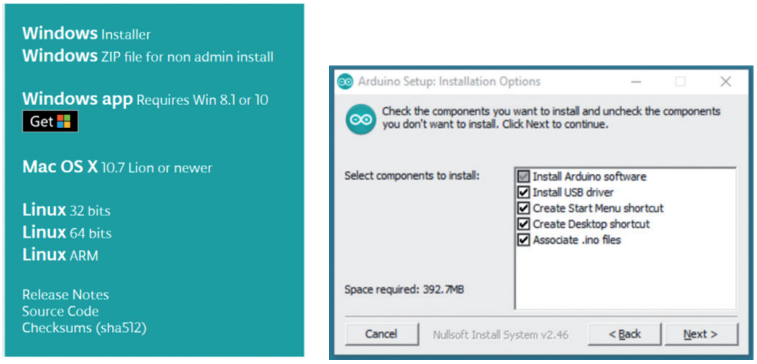
Arduino is een computerplatform dat het werken met microcontrollers eenvoudig moet maken. Het platform is bedoeld voor hobbyisten, artiesten en iedereen die slimme en creatieve objecten wil maken met microcontrollers. Het is ontwikkeld door een Italiaanse groep o.l.v. Massimo Banzi. Arduino bestaat uit hard- en software. Alles is open-source, iedereen mag vrij de software gebruiken, fabrikanten kunnen de software aanpassen aan hun systemen en er bestaat enorm veel compatibele hardware voor Arduino.

Arduino hardware

De hardware bestaat uit bordjes met een controller, de *boards*, en uitbreidingsbordjes, *shields*. De originele boards hadden een 8-bits controller uit de reeks ATMEGA van Atmel-Microchip. Arduino bouwt nu ook bordjes met een 32-bit controller van ARM. Andere fabrikanten leveren compatibele bordjes met nog andere controllers, hiervoor zijn uitbreidingen voor de Arduino-software gemaakt.

De programmeeromgeving installeren

Er bestaat een *online* versie van de Arduino IDE. Je gebruikt die zonder iets te installeren. Alle software die je maakt wordt bewaard in de *cloud* en je beschikt steeds over de allerlaatste versies van de software. Deze versie ondersteunt de ESP-chips (nog) niet. We gebruiken deze dan ook niet hier.



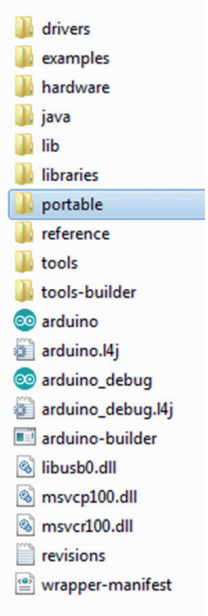
Figuur 9:Installatie Arduino

De *Desktop* versie bestaat voor Windows, Linux en MAC OS X. In dit boek wordt de Windows versie besproken.

Op de downloadpagina van www.arduino.cc vind je na enkele doorverwijzingen de link naar de Windows Installer. Download en start de Installer, bij de installatieopties installeer je de USB driver, de snelkoppelingen en de koppeling van de .ino-bestanden. De opties in de volgende vensters laat je op hun standaard waarde staan. Na de installatie kan je de IDE opstarten. De gemaakte sketches worden bewaard in de documentenfolder van jouw PC.

Arduino portable

De portable versie van de IDE installeert alle bestanden en folders op één plaats zonder andere folders van de computer te beïnvloedden. De portable IDE kan gebruikt worden door studenten die niets kunnen installeren op de schoolcomputers: installeer alles op een externe schijf, de gemaakte sketches komen ook op die schijf terecht. Dit systeem kan je op verschillende computers gebruiken.



Figuur 10: Folder portable

Installatie

- download de ZIP-versie van de downloadpagina;
- extraheer alles naar een gekozen plaats;
- maak folder ***portable*** in de gemaakte folder, hierin worden de sketches bewaard.
- start de IDE op met ***Arduino.exe***.

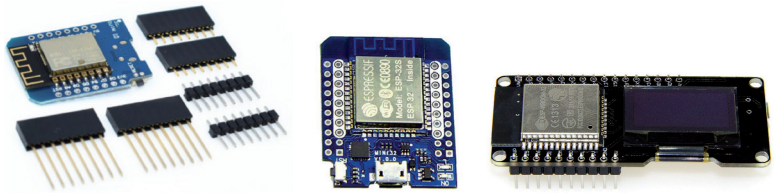
De IDE gebruiken

- in het *sketchvenster* maak je de sketch;
- onderaan in het *statusvenster* vind je meldingen van het systeem;
- Met *examples* uit menu *File* kan je **voorbeelden** oproepen;
- met *preferences* uit menu *File* stel je de IDE in naar je persoonlijke voorkeuren, o.a. de **gebruikerstaal**;
- met menu *Sketch* kan je de sketch **compileren** (omzetten naar machinetaal) en **uploaden** naar de controller;
- menu *Tools*:
  - kies het juiste controllerbordje;

Hoofdstuk 3 • Experimenteren met de ESP8266/ESP32

Voor de voorbeelden en toepassingen in volgende hoofdstukken gebruiken we vier soorten bordjes:

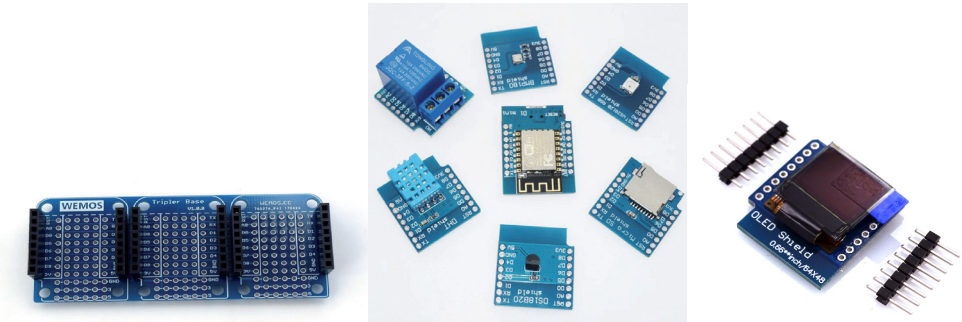
- Een bordje met ESP8266 in D1 mini formaat
- Een bordje met ESP32, ook in D1 mini formaat
- Een bordje met ESP32 met geïntegreerd OLED schermpje
- De ESP-PICO-KIT



Figuur 17: ESP bordjes voor de experimenten

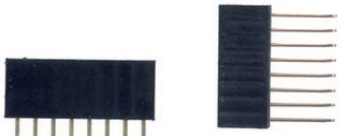
Van de eerste drie zijn er bij Aliexpress meerdere varianten voor een luttel bedrag te koop. De ESP-PICO-KIT is verkrijgbaar bij Elektor. We hebben minstens één ESP8266 bordje en één ESP32 nodig. De leveringstermijnen bij Aliexpress bedragen enkele weken, bestel zo vlug mogelijk.

Voor het D1 mini systeem gebruiken we een basisplaatje voor drie bordjes, een OLED schermpje in D1 mini formaat en enkele sensorbordjes in de vorm van een shield. De sensorbordjes zijn meestal te koop in een set waar ook een controllerbordje bij geleverd wordt.



Figuur 18: D1 mini, boards en shields

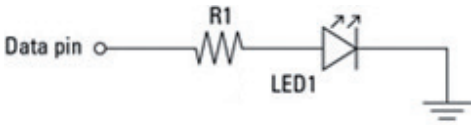
De aansluitingen van de bordjes worden voorzien van **pinheaders** of **female connectoren**. Pinheaders passen in een breadboard, in female connectoren plaats je jumpwires of pinheaders. Met lange pinheaders kan je bordjes op elkaar plaatsen.



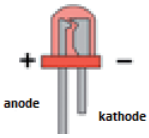
Figuur 19: Female connectoren

Figuur 20: Pinheaders

Bij experimenten zijn enkele **LED's** onmisbaar. Hiermee kan je signaalniveaus zichtbaar maken. Gebruik nooit een LED zonder **voorschakelweerstand**, de weerstand, 330 Ω tot 560 Ω, beperkt de stroom door de LED en vermijdt beschadiging van de controller.

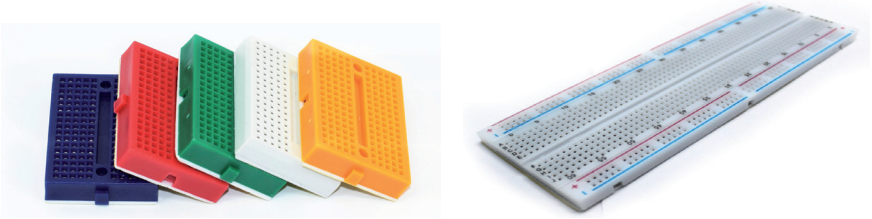


Figuur 21: LED met voorschakelweerstand



Figuur 22: Aansluitingen LED

Proefopstellingen maak je op **breadboards**. Die bestaan in verschillende vormen en maten.



Figuur 23: Verschillende breadboards

Een breadboard heeft rijen gaatjes waarin elektronische componenten geprikt kunnen worden. Intern zijn de gaatjes met elkaar verbonden, zie de tekening hieronder rechts.

Hoofdstuk 4 • De Arduino programmeertaal - Variabelen

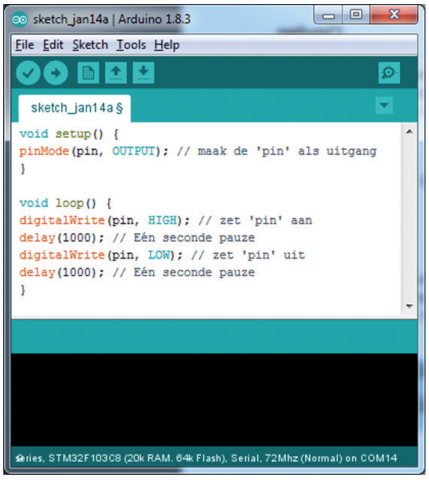
De Arduino programmeertaal is afgeleid van C. Arduino-instructies en C-instructies volgen dezelfde syntax:

- Elke programmaregel eindigt op puntkomma
- Arduino is hoofdlettergevoelig

Structuur van een sketch
Een Arduino programma, ook sketch genoemd, bestaat uit twee blokken met instructies (functies), **setup()** en **loop()**.

- **setup** wordt één maal, in het begin uitgevoerd.
- **loop** wordt steeds herhaald

```
void setup()
{
  instructie 1;
  instructie 2;
}
void loop()
{
  instructie a;
  instructie b;
}
```



Figuur 28: Structuur van een sketch

Deze sketch zet de spanning op de pin voortdurend aan en uit. As je een LED aansluit aan de pin, dan gaat die flikkeren: 1 seconde aan, 1 seconde uit, 1 seconde aan, …

Figuur 29: Voorbeeld van commentaar

Commentaar

Alle tekst achter // tot het einde van de regel en alle tekst tussen /* en */ is commentaar. Commentaar wordt genegeerd bij het compileren en het uitvoeren van het programma. Commentaar verduidelijkt de werking van een programma.

```
#define naam waarde
```

De compiler zal in heel de sketch de gedefinieerde *naam* vervangen door de gegeven *waarde*. Voorbeeld:

```
#define ledPin 3
// de compiler vervangt in heel de sketch ledPin door 3
```

Opgelet:

- Geen gelijkteken (=) tussen naam en waarde;
- Geen puntkomma na de #define-instructie.

Deze instructie kan leiden tot ongewenste effecten en moeilijk terug te vinden fouten. Als de gedefinieerde naam voorkomt als deel van een variabele naam, dan wordt dat stuk van de variabele naam ook vervangen door de waarde.

Waarden

Arduino werkt met waarden: getallen, stukken tekst, …

- Numerieke waarden zijn getallen, geheel of niet-geheel;
- Gehele waarden worden decimaal, binair of hexadecimaal voorgesteld,
  - 44: decimale voorstelling
  - 0x2C: hexadecimale voorstelling van 35
  - 0b101100: binaire voorstelling van 35
- Bij niet-gehele getallen gebruiken we punt als decimaal teken en/of E als exponent
  - -3.456E7 = -3.456 x 10^{7} = -3456000
  - 23.5E-2 = 23.5 x 10^{-2} = 0.235.235

Hoofdstuk 5 • Rekenen met variabelen en constanten

Rekenkundige operatoren: +, -, *, /, %

- Optellen, aftrekken, vermenigvuldigen en delen doe je met **+, -, *, /.**
- % geeft de rest van een deling van twee gehele getallen

Een bewerking van twee gehele getallen is geheel. Is één van de twee float, dan is het resultaat float

```
11 / 3 = 3
11 % 3 = 2
11.0 / 3 = 3.666666666  // 11.0 is float
```

Vergelijkingsoperatoren
Vergelijk variabelen en/of constanten met elkaar. Het resultaat is **true** of **false**

```
x == y   // x gelijk aan y ?
```

Let op: dubbel gelijkheidsteken! Een enkel gelijkheidsteken geeft foute resultaten bij de uitvoering van jouw programma.

```
x != y   // x niet gelijk aan y ?
x < y    // x kleiner dan y ?
x > y    // x groter dan y ?
x <= y   // x kleiner dan of gelijk aan y ?
x >= y   // x groter dan of gelijk aan y ?
```

Logische operatoren
Logische berekeningen met logische waarden geven resultaat **true** of **false**:

```
A && B   // logische AND - true als A en B true zijn
A || B   // logische OR - true als A en/of B true is
!A       // logische NOT - true als A false is, false als A true is
```

Let op de volgorde van de bewerkingen:
- eerst * en / en % - dan + en -
- dan vergelijken
- dan logische berekeningen Bewerkingen met dezelfde prioriteit: van links naar rechts

```
1 + 2 * 3                  // = 7 (eerst *, dan +)
1+2*3 == 7 && !(7 > 3)  // = false
```

Samengestelde opdrachten
Samengestelde opdrachten zijn verkorte vormen van sommige Arduino- en C- instructies.

Hoofdstuk 6 • Tekstvariabelen: String en char

Een variabele met type **char** heeft als waarde één karakter, een letter, een cijfer, een leesteken of een speciaal teken. Een char waarde wordt voorgesteld met enkele aanhalingstekens:

```
char X = 'v';
```

De char variabele wordt bewaard in een byte. Gehele variabelen en char-variabelen kunnen aan elkaar gelijk gesteld worden, de ASCII-waarde wordt doorgegeven.

```
int a, b;
char X, Y;
a = X;   // a wordt de ASCII-waarde van X
Y = b    // Y wordt ASCII-teken nr. b
```

Op www.arduino.cc staan enkele char-functies, voorbeeld:

> **isDigit(xx)** gaat na of xx (char-variabele, -constante) een cijfer voorstelt. Resultaat is TRUE of FALSE

Een **string** is een rij tekens en eindigt met het null-karakter, voorgesteld door '\0'. Strings worden voorgesteld met dubbele aanhalingstekens.

> "dit is een string" en "#@gG56;}" zijn strings

Een string kan gedeclareerd worden als type String of als een array van type char. Hieronder zie je vier maal de declaratie van string "arduino":

> **String str1** = "arduino"; //variabele type String, zie hierboven
> **char str2[8]** = {'a','r','d','u','i','n','o'}; //array type char met maximaal 8 tekens
> **char str3[8**] = {'a','r','d','u','i','n','o','\0'}; //het nulkarakter wordt expliciet vermeld
> **char str4[**] = {'a','r','d','u','i','n','o'}; //array, aantal tekens is onbepaald

functie String()

- String (waarde) maakt van waarde een string. Waarde kan numeriek, string of char zijn.
- String (waarde, BASE) maakt van een gehele waarde een string, BASE is HEX, BIN of DEC.
- String (waarde, decimalen) maakt van een niet-gehele waarde een string met aantal cijfers na de komma gelijk aan decimalen.

Hoofdstuk 7 • Programmastructuren

Voorwaardelijke instructies

Afhankelijk van een voorwaarde worden instructies al dan niet uitgevoerd. Arduino heeft drie soorten: **if**, **if ... then** en **switch ... case**

if

```
if (logische expressie)
{
  instructie 1;
  instructie 2;
  …
```

Voer instructies 1, 2, .. uit als de logische expressie **true** is.

if … else

```
if (logische expressie)
{
  instructie 1;
  instructie 2;
  …
}
else
{
  instructie a;
  instructie b;
  …
}
```

Voer instructies 1, 2, .. uit als de logische expressie **true** is.
Voer instructies a, b, .. uit als de logische expressie **false** is.

switch … case

```
switch (expr)
{
  case a:
    statements A;
    break;
  case b:
    statements B;
    break;
  default
    statements C;
    break;
}
```

Voer instructies A uit als expr == a.
Voer instructies B uit als expr == b.
Voer instructies C uit als expr ≠a en ≠b.

Break zorgt er voor dat de volgende cases niet worden uitgevoerd.

Hoofdstuk 8 • Functies en libraries

Ee functie is een blok code met naam. Deze code wordt uitgevoerd bij het oproepen van de functie en berekent eventueel een resultaat.

Ingebouwde functies

Veel functies zijn gedefinieerd binnen Arduino:

- Wiskundige functies:
  - Y = sin(x); // x in radialen
  - Y = log(10); // = 1
  - Y = ln(x); // natuurlijke logaritme
  - Y = max(2, 5.7); // = 5,7
  - Y = pow (2, 3); / = 2^3 = 8
  - Y = sqrt (1024); // = 32 (vierkantswortel)
  - …
- I/O functies: digitalWrite(), pinMode(), …
  Zie volgend hoofdstuk.
- Tijdfuncties:
  - delay(2000); // wacht 1000 milliseconden (1 seconde)
  - y = millis(); // geeft het aantal milliseconden sinds de start van de sketch
- String functies:
  - Str1 = “arduino”;
    Str1.setCharAt(0,’A’); // verandert karakter nr. 0 in ‘A’
  - toInt: Maakt een integer van een string
    Y = toInt(23); // =23
    Y = toInt(56RTY); // = 56
    Y = toInt(RT45YJU); / = 0
    …
    Zie ook vorige hoofdstukken.

De volledige lijst vind je op de Arduino website.

Zelfgedefinieerde functies
Een functie moet gedeclareerd worden:

```
type functionName (type1 arg1, …)
{ statement1;
  …
  return value1
}
```

Het resultaat van de functieaanroep is *value1*.

voorbeeld

```
float discriminant (float a, b, c) {
 D = sqrt(b*b – 4*a*c) ;
 return D;
}

void setup() {
 …
}

void loop() {
 ..
 d = discriminant(a, b, c);
}
```

$$D = \sqrt{b^2 - 4ac}$$

Libraries
Met **libraries** kan de Arduino omgeving uitgebreid worden. Ze bieden extra functionaliteit die in sketches kunnen gebruikt worden. Enkele libraries zijn standaard geïnstalleerd in het Arduino-systeem maar je kan ook extra libraries downloaden of zelf schrijven. Een library bevat bestanden waarin functies gedefinieerd worden. Die kunnen in eigen toepassingen opgeroepen worden.

Library-bestanden kunnen gebruikt worden als ze opgenomen worden met een sketch. Dat doe je met:

```
#include “bestand.h”            // geen puntkomma!
```

of

```
#include <bestand.h>
```

Een gedownloade library wordt meestal opgeslagen als een .ZIP-bestand. Je integreert die in het systeem met menu *Sketch - Include Library - Add .ZIP Library.* Selecteer de **gezipte** library.

Hoofdstuk 9 • Input en Output

Input/output via de seriële poort
Uitvoer van een sketch kan direct doorgestuurd worden naar het monitorvenster van de arduino-IDE. Open de monitor met menu *Tools*. De verbinding wordt in de sketch opgestart met

```
Serial.begin(baudrate);
```

Baudrate is de transmissiesnelheid. ESP-bordjes werken snel, gebruik hiervoor een hoge waarde zoals *115200*. Uitvoer naar het scherm doe je met:

```
Serial.print (expressie);
```

of

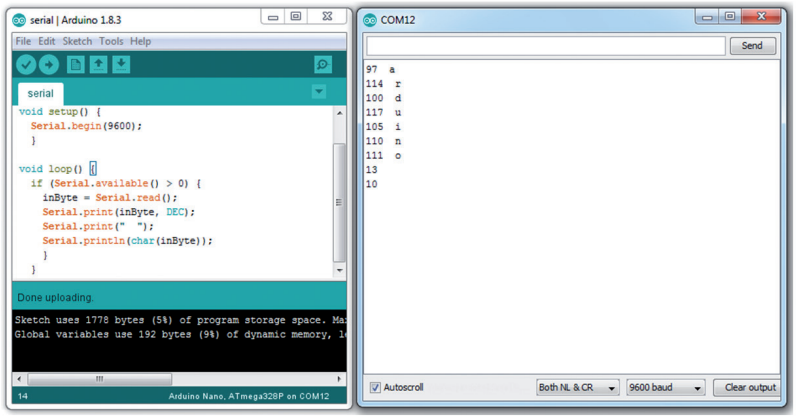
```
Serial.println (expressie);
```

De tweede vorm zal na de expressie ook de regel afsluiten, de volgende uitvoer gaat naar een volgende regel.

Als data toekomt in de seriële poort, worden die tijdelijk in een buffergeheugen opgeslagen. **Serial.available()** geeft het aantal beschikbare karakters in de buffer.

Serial.read() leest het eerste beschikbare karakter en geeft de bijhorende ASCII-waarde. Als er geen karakter beschikbaar is, is het resultaat -1.

Voorbeeld:



Figuur 32: Input en output via de seriële poort

Start de sketch. Typ in het venster bovenaan de monitor een woord, bijvoorbeeld arduino. Klik dan op *send*. Alle karakters worden ingelezen. Op het scherm verschijnen de karakters met hun ASCII-waarde. De laatste twee getallen zijn 13, *naar het begin van de regel,* en 10, *naar volgende regel*.

Digitale in- en output pinnen

De ESP-chips hebben een aantal input/output pinnen (GPIO's), de ESP8266 heeft er 17, de ESP32 heeft er 36. Op de bordjes zijn er altijd minder pinnen beschikbaar. Elke poort heeft een nummer, de nummers staan meestal vermeld op de bordjes, anders vind je ze terug in de documentatie van de fabrikant. Elk van die poorten kan als input- of outputpoort geconfigureerd worden.

Sluit nooit spanningen hoger dan 3.3V aan op een inputpin, de controller zal beschadigd worden.

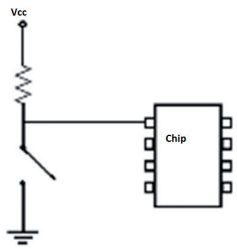
Lezen van een inputpoort

```
const int inputPin = 2
int inputStatus = 0;

void setup()
{
  pinMode(inputPin, INPUT);
}

void loop() {
  inputStatus = digitalRead(inputPin);
}
```

inputPin (pin2) is geconfigureerd als input
inputPin wordt gelezen, het resultaat staat in inputStatus

Uitlezen van een schakelaar

Een open inputpoort (niets aangesloten) heeft een onbepaald niveau, het lezen van die poort geeft niet te voorspellen resultaten. Een schakelaar met pullup-weerstand heeft altijd een duidelijk ingangsniveau: 0 V als de schakelaar gesloten is, 3,3 V als hij open is.



Figuur 33: Pullup-weerstand van 10 kΩ

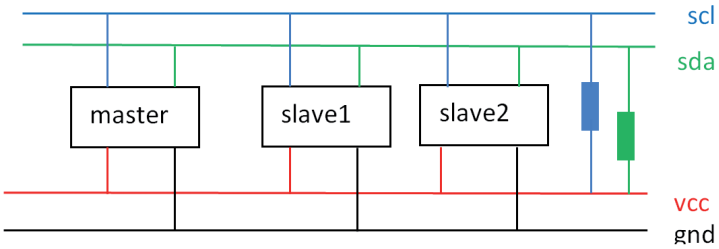
Alle poorten, behalve poort 0, hebben een ingebouwde pullup-weerstand. De externe weerstand kan je dan weglaten. Je activeert die bij de configuratie:

```
pinMode (inputPin, INPUT_PULLUP);
```

Hoofdstuk 10 • De I²C-bus

De *I²C-bus*, *Inter IC Bus*, is in 1979 ontwikkeld en gepatenteerd door Philips als goedkope verbindingsbus tussen processor en periferie in consumentenelektronica. *I²C* is gestandaardiseerd in 1990 en andere fabrikanten hebben toen deze technologie geïmplementeerd in hun chips. Ze noemen ze soms T*WI, Two Wire Interface*. De verbinding gebeurd met twee buslijnen: *SDA, Serial Data*, verstuurd data en *SCL, Serial CLock* het kloksignaal. De bus bestaat uit één **master** en minimaal één **slave**. De master, meestal de controller, stuurt de bus aan. Slaves communiceren alleen over de bus als de master het hen verzoekt. Slaves hebben een 7 bit adres op de bus, er kunnen maximaal 127 slaves de bus gebruiken. De oorspronkelijke I²C-bus werkt met een snelheid van 100 kbit per seconde, latere versies gebruiken 400 kb/s of 3.4 Mb/s en hebben een adresseringsruimte van 10 bit.

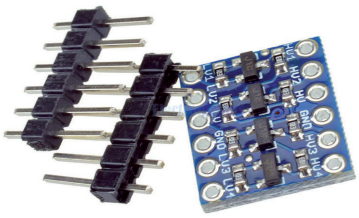
Figuur 41: I²C-logo



Figuur 42: I²C systeem, 1 master, 2 of meer slaves

SLA en SCL zijn met pull-up weerstanden van ongeveer 1 kΩ verbonden met de voedingslijn. Je kunt ze weglaten bij een systeem met één master en één slave.

De ESP8266 en de ESP32 kunnen elk tweetal GPIO's als SDA en SCL gebruiken.



Figuur 43: Logical level convertor

Opgelet: Er zijn I²C onderdelen met signaalniveaus en voedingsspanning van 5 volt, en andere met niveaus van 3.3 volt. Als je die door elkaar gebruikt worden onderdelen

Hoofdstuk 11 • Timer interrupts

Met timer interrupts kan je taken uitvoeren op juist getimede intervallen, onafhankelijk van de rest van de code. Timer interrupts zijn nuttig voor o.a.:

- metingen op gelijke tijdsintervallen
- het berekenen van de tijd tussen gebeurtenissen
- het maken van signalen met een specifieke frequentie
- klokken, knipperlichten, …

De ESP8266 en ESP32 hebben 4 timers: timer0 tot timer 3. De timers van de ESP8266 kunnen het WiFi-systeem verstoren, het wordt afgeraden om ze in eigen programma's te gebruiken. De ESP32 heeft twee processorkernen met elk een eigen timerset. Eén CPU-kern wordt gebruikt voor interne processen van de ESP zoals WiFi en de andere kan vrij gebruikt worden voor eigen programma's.

Timers hebben een teller die met elke klokpulsen verhoogd of verlaagd wordt, onafhankelijk van de andere software. Wanneer de teller een vooraf bepaalde waarde bereikt, wordt ze nul en geneert ze een interrupt.

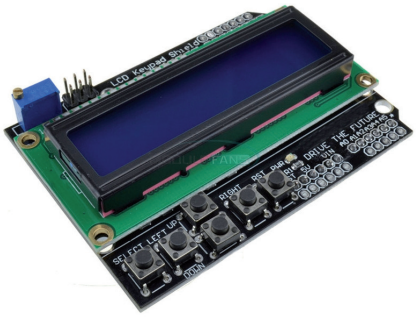
Voorbeeld: deze timer verandert elke seconde de status van de LED.

- **hw_timer_t * timer = NULL;**
  Variabele timer wordt gedeclareerd als hardware-timer**timer = timerBegin(0, 80, true);**
- gebruik timer **0**
- De klok van 80 MHz. gedeeld door **80** geeft 1MHz of 1000 000 pulsen per seconde
- **True** laat de timer optellen
- **timerAttachInterrupt(timer, &onTimer, true);**
  Functie onTimer wordt gekoppeld aan de timer.
- **timerAlarmWrite(timer, 1000000, true);**
  Als de timer eindwaarde **1000000** bereikt (na 1 seconde) wordt een interrupt gegenereerd die functie onTimer oproept. Deze verandert de status van de LED.
- **timerAlarmEnable(timer);**

Functie millis( )
Functie millis( ) geeft de tijd in milliseconden sinds de start van de controller. Na ongeveer 50 dagen wordt een overflow bereikt en wordt terug vanaf nul begonnen. Met millis kan soms het gebruik van timer-interrupts vermeden worden.

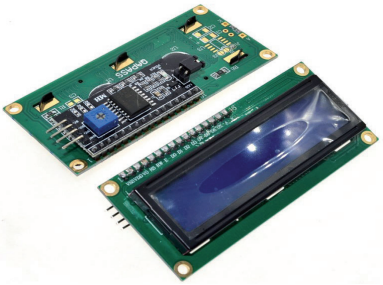
Hoofdstuk 12 • Displays

Voor klassieke Arduino-boards is er een ruime keuze aan displays beschikbaar, meestal in de vorm van een shield. De shields zijn meestal niet bruikbaar voor de ESP-bordjes die hier gebruikt worden, de pinnen van de bordjes volgen niet de Arduino-standaard.



Figuur 47: Arduino LCD shield

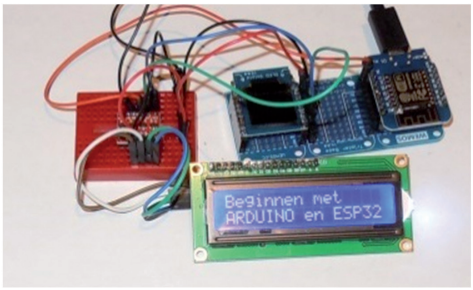
LCD-displays



Figuur 48: I²C LCD shield

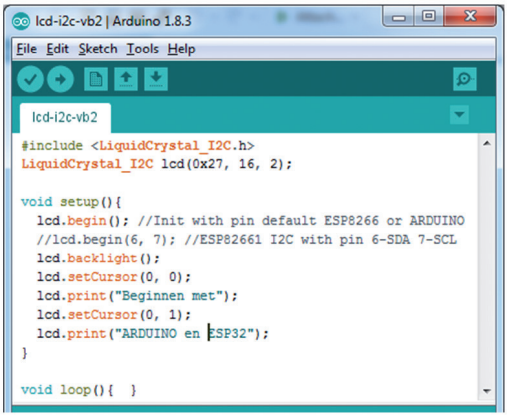
Losse LCD-displays worden ondersteund door Arduino maar hiervoor zijn minstens 7 GPIO's nodig. De meeste ESP-bordjes hebben er niet zo veel.

Er bestaan LCD's met I²C convertor. Hiermee kan het LCD aangesloten worden op een I²C-bus. De displays zijn alfanumeriek, ze hebben een vast aantal plaatsen voor tekens. Let op: het zijn 5V-bordjes, gebruik een level-converter als je ze op een ESP aansluit. Voor deze bordjes heb je een library nodig. Bij veel fabrikanten kan je die downloaden van hun website. Op github vind je library LiquidChristal_I2C (https://github.com/lucasmaziero/LiquidCrystal_I2C). Die is bruikbaar voor de klassieke Arduino bordjes en voor de ESP8266, helaas niet voor de ESP32.



Figuur 49: ESP8266 met Level-convertor en I²C-display

Het voorbeeld hieronder zet een stuk tekst op het LCD.



Figuur 50: LCD aan de ESP8266

- LiquidCrystal_I2C lcd(0x27, 16, 2) is de constructor. lcd wordt aangemaakt, waarden 0x27 (het I²C adres), 16 (het aantal kolommen van de display) en 2 (het aantal rijen) worden doorgegeven. Dit display heeft 32 (16 x 2) plaatsen voor karakters.
- lcd.begin() start het display. Gebruik deze regel als sda = 4 en scl = 5.
- Gebruik lcd.begin(sda, scl) als de bus niet de standaardwaarden 4 en 5 gebruikt.
- lcd.setCursor(0, 0) zet de cursor op kolom 0 en rij 0.
- lcd.print zet gegevens op het schermpje.

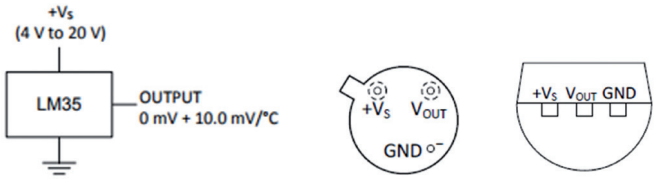
OLED-displays

Kleine monochrome OLED-displays zijn goedkoop, energiezuinig en hebben een zeer goede beeldkwaliteit. In Elektor van januari 2018 vind je een meer informatie over OLED. Ze hebben een I²C-interface en worden bijna allemaal aangestuurd door dezelfde chip, de SSD1306. Er bestaan universele libraries voor deze bordjes. Die zijn terug te vinden op de websites van fabrikanten en leveranciers. Bij Aliexpress vind je een heleboel van die displays, met prijzen vanaf 2,50 euro.

Hoofdstuk 13 • Temperatuur en luchtvochtigheid

LM35, LM36, LM37: temperatuursensoren

De LM35, 36 en 37 zijn temperatuursensoren met analoge uitgang. Ze zijn compatibel met de TMP35/36/37. De uitgang kan eenvoudig worden ingelezen door een analoge ingang van de controller.



Figuur 56: LM35, aansluitgegevens

	LM35/TMP35	LM36/TMP36	LM37/TMP37
Output	10 mV/°C	500 mV + 10 mV/°C	20 mV/°C
Temperatuur	10°C - 125°C	-40°C - 125°C	5°C - 125°C
Nauwkeurigheid bij 25 °C	± 1°C		
Nauwkeurigheid over volledig bereik	± 2°C		
Voeding	2.7 V – 5.5 V		

Een voorbeeldsketch vind je in hoofdstuk *Input- Output*

DHT11 en DHT22: temperatuur en luchtvochtigheid

De DHT11 en DHT22 zijn goedkope sensoren voor temperatuur en lucht-vochtigheid. Ze hebben een chip aan boord die de analoge metingen omzetten in een digitaal signaal. Dat signaal wordt met 1 draad doorgestuurd naar een controller. De tabel toont enkele kenmerken van DHT11 en DHT22:

	DHT11	DHT22
Voeding, signaalniveau	3V tot 5V	
Temperatuur	0°C - 50°C + 2°C	-40°C – 80°C + 0,5°C
Vochtigheid	20 - 80 %rH + 5 %	0 – 100 %rH + 5 %
# metingen / sec.	1	2

Hoofdstuk 14 • Netwerken met de ESP32

Netwerk standaarden

Figuur 60: WiFi-logo

Computers kunnen met elkaar communiceren als ze dat op dezelfde manier doen. Maar ze moeten voldoen aan een aantal afspraken om elkaar te verstaan. De afspraken zijn vastgelegd in een aantal netwerk standaarden.

WiFi

Er bestaan meerder manieren om computers fysiek met elkaar te verbinden: met koperen kabels, met glasvezel of draadloos. WiFi is de meest gebruikte standaard voor draadloze lokale netwerken. De ESP-chips hebben hard- en software aan boord voor draadloze WiFi toegang. De ESP's ondersteunen WiFi 802.11 b/g/n.

IP: Internet Protocol

IP is de standaard voor de adressering van computers. Elke computer die met andere computers communiceert, moet een uniek adres hebben. Sommige toestellen hebben een vast ingesteld adres, andere gebruiken tijdelijke adressen. Die worden toegekend door een DHCP-server[4].

Een **IPv4** adres bestaat uit 32 bits of 4 bytes, er zijn dus 2^{32} = 4294967296 adressen beschikbaar voor Internet. Het eerste deel van het adres is het **netwerkadres**, de rest geeft het toestel aan binnen het **lokale netwerk**. Het **subnetmask** geeft aan welk deel netwerkadres is. Het **gateway-adres** geeft aan op welke manier toestellen buiten het lokale netwerk bereikbaar zijn.

Voorbeeld:

IP = 192.168.1.33
Subnetmask: 255.255.255.0
Gateway: 192.168.1.1

*Het **netwerkadres** is 192.168.1: alle toestellen met een adres dat begint met deze getallen behoren tot hetzelfde **lokale LAN**. Een verbinding met een toestel buiten dit LAN gebeurt via de **gateway**.*

IPv6 adressen bestaan uit 16 bytes. Dit levert een quasi onbeperkte hoeveelheid adressen. Door de snelle groei van Internet raken de IPv4 adressen op. Internet schakelt geleidelijk over van IPv4 naar IPv6.

4 Een serverprogramma dat een IP-adres toekent aan een PC in het netwerk

De IP-gegevens worden meestal automatisch toegekend door een DHCP-server. Die is ingebouwd in de WiFi-router van je lokaal netwerk.

TCP en UDP
Informatie wordt over het netwerk verstuurd in pakjes. **TCP** en **UDP** zorgen voor de verdeling in pakjes aan de zenderkant en voor het terug aan elkaar zetten van de pakjes, in de juiste volgorde aan de ontvangerkant.

TCP zorgt ook voor foutendetectie en correctie. Pakjes met fouten worden een tweede keer door het netwerk gestuurd. TCP wordt daarom gebruikt als informatie zonder fouten moet toekomen. Mailsystemen werken over **TCP**.

UDP werkt zonder foutencorrectie en werkt dan ook sneller. **UDP** wordt gebruikt als de timing van een signaal belangrijk is dan het wegwerken van eventuele foutjes. Telefonie en videoconferentie gebruiken **UDP**

Toepassingsprotocollen
Netwerktoepassingen werken met TCP/IP of UDP/IP netwerklagen. Elke toepassing volgt zijn eigen standaard: Webservices werken volgens de http-standaard, mailsystemen gebruiken er weer andere. Elke toepassing heeft een eigen **poortnummer**, een code die de client bij elke datapakket doorstuurt naar de server en waarmee de server dan weet naar welke toepassing het pakket moet gestuurd worden. Een webserver heeft poort 80. De client stuurt ook een eigen poort-nummer mee, de antwoordpakketten van de server worden hiermee op naar de juiste clienttoepassing gestuurd.

Protocollen en de ESP32
WiFi, IP, TCP en UDP zijn ingebakken in de ESP chip. Toepassingen worden in de sketches geprogrammeerd.

Library WiFi
Deze library hoort bij het ESP32 systeem, hiermee maak je een server of een client van de ESP. Bij de library horen enkele klassen:

- **WiFi** maakt een verbinding met WiFi. De voornaamste functies zijn:
  - **WiFi.begin(*ssid, password*)** maakt de verbinding
  - **WiFi.status()** geeft aan of er verbinding is, geeft waarde 1 .. 6 terug,
  - **WiFi.localIP()** geeft het IP addres van de ESP32
- **WiFiClient** maakt een TCP/IP client. De voornaamste functies zijn:
  - **connect (*host, port*)** maakt een verbinding met server met IP-adres ***host*** op poort ***port***
  - **print(*data[]*)** stuurt ***data*** naar de server
  - **stop()** stopt de verbinding
  - **available()** gaat na of er gegevens beschikbaar zijn om in te lezen
  - **read()** leest een byte
  - **read(*uint8_t *buf, size_t size*)** leest gegevens met lengte ***size*** en zet deze in ***buf***

Hoofdstuk 15 • Het weer

In dit hoofdstuk gaan we het weer opvragen aan een Weerserver. Dat is een server die van gelijk welk dorp op de wereld de weerparameters kan doorsturen: temperatuur, luchtdruk, windsnelheid, algemene toestand, … .

We gebruiken weerserver Wunderground, www.wunderground.com, die geeft op aanvraag gegevens van vandaag en weersvoorspellingen voor de volgende 10 dagen. Op de website vind je alle mogelijke informatie over het weer op alle mogelijke plaatsen. Als je weerinformatie live wil opvragen in je eigen programma, moe je je eerst registreren en abonneren op de weerservice.

Wunderground biedt verschillende abonnementsformules aan, inclusief een gratis versie. Die is beperkt tot 10 opvragingen per minuut en maximaal 500 per dag. Dat is voldoende voor onze toepassing. Ga op de startpagina naar **Home** en klik op **Weather API for developers**. Als heel de procedure doorlopen is, krijg je een sleutel, een code, die je in je programma's moet opnemen.

De sketch is een Webclient die een aanvraag doorstuurt naar de server. In de aanvraag zitten o.a. de sleutel en de plaats waarvan we het weer wensen. Het resultaat is een JSON-bestand.


```
{„coord“:{„lon“:-0.13,“lat“:51.51},“weather“:[{„id“:300,“main“:“Drizzle“,“-
description“:“light intensity drizzle“,“icon“:“09d“}],“base“:“stations“,“-
main“:{„temp“:280.32,“pressure“:1012,“humidity“:81,“temp_min“:279.15,“temp_
max“:281.15},“visibility“:10000,“wind“:{„speed“:4.1,“deg“:80},“clouds“:{„al-
l“:90},“dt“:1485789600,“sys“:{„type“:1,“id“:5091,“message“:0.0103,“cou-
ntry“:“GB“,“sunrise“:1485762037,“sunset“:1485794875},“id“:2643743,“name“:“Lon-
don“,“cod“:200}
```


Json-bestand voor het weer in Londen, 14/03/2018, in verkorte vorm

Json is opgebouwd uit objecten, arrays en waarden. Dingen die bij elkaar horen, worden met accolades bij elkaar gehouden. Het Json-bestand met weergegevens is te groot voor het geheugen van een ESP. We hebben een parser nodig die de gegevens analyseert terwijl ze voorbij komen en die er direct de nodige gegevens uithaalt.

Een goed programma voor de weerclient is esp-wunderground.ino, en is te vinden op github (https://gist.github.com/bbx10/149bba466b1e2cd887bf). Dit programma heeft library ArduinoJson-master nodig, die is ook te vinden op github (https://github.com/bblanchon/ArduinoJson). De sketch vraagt éénmaal per vijf minuten nieuwe weergegevens. Dat valt nog binnen de beperkingen van een gratis abonnement.

De sketch moet nog aangepast worden met:

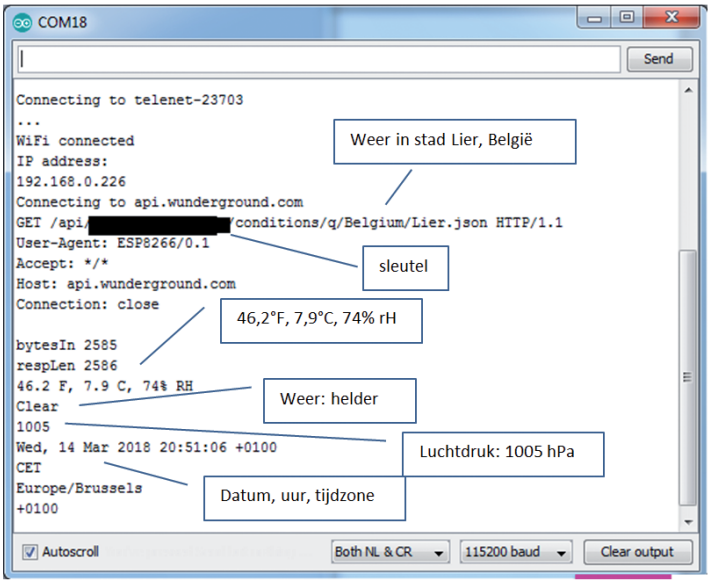
- jouw netwerkgegevens
  const char SSID[] = “*******”;
  const char PASSWORD[] = “*******”;
- jouw sleutel bij Wunderground
  #define WU_API_KEY “**********”
- de plaats waarvan je de weergegevens opvraagt
  #define WU_LOCATION “Belgium/Lier”

Pas de sketch aan zodat de gegevens op een display getoond worden!

Figuur 68: Ook voor ESP32

Deze sketch werkt ook op een **ESP32** als je in het begin **#include <ESP8266WiFi.h>** vervangt door **#include <WiFi.h>**

Uitvoer van het programma:



Inhoud van de download-folder

Op de website van Elektor, www.elektor.nl, vind je bij de beschrijving van het boek bestand **Arduino-ESP32-download.zip**. Dit bestand bevat de voorbeeldsketches en de libraries uit het boek. Elke sketch zit in een folder met dezelfde naam als de sketch, op die manier kan Arduino die sketches direct openen. De sketches zijn:

hoofdstuk 10

- morse-sos: de sketch die morsesignalen genereert

hoofdstuk 11

- serial: input/output via de seriële poort
- temp-lm35: analoge input
- pwm-esp32: analoge output voor de ESP32
- touch: de touch-ingangen van de ESP32

hoofdstuk 12

- eeprom_24c16: de 24c16: een I^2C-eeprom

hoofdstuk 13

- timer-esp32: timer interrupts
- knipperled-millis: functie millis()

hoofdstuk 14

- OLED-ESP32-lolin: OLED-displays
- klok1: project KLOK
- klok2: project KLOK

hoofdstuk 15

- dht11: DHT11 en DHT22: temperatuur en luchtvochtigheid
- DS18B20: DS18B20: temperatuursensor

hoofdstuk 16

- wificlient1: De ESP als webclient
- webserver1: De ESP32 als webserver
- webserver2: Webserver, temperatuur en luchtvochtigheid
- SoftAP3: Access-point webserver
- NTP-klok_esp8266: UDP: NTP-klok

hoofdstuk 16

- esp-wunderground: het weer

De libraries staan in .ZIP-vorm in het document. Zo kunnen ze in Arduino ingevoerd worden (zie hoofdstuk 10). De libraries zijn:

hoofdstuk 10

- morse.zip

hoofdstuk 14

- LiquidCrystal_I2C-master
- esp8266-oled-ssd1306-master
- hoofdstuk 15
- dht-sensor-library

hoofdstuk 16

- ArduinoJson-master

Index

MET ARDUINO SOFTWARE

ESP32

PROGRAMMEREN VOOR BEGINNERS

Dirk Ghysels

Dirk Ghysels (1954) studeerde wiskunde aan de KU Leuven en vervolgens informatica aan de Universiteit Antwerpen. Hij werkte als docent, informaticus en preventieadviseur.

In ESP32 programmeren voor beginners bekijken we de ESP32 en zijn voorganger, de ESP8266. Deze kleine, maar zeer krachtige Chinese chips bestaan uit een processor, geheugen, input- en outputaansluitingen en hebben draadloze netwerktoegang via WiFi.

Met dit boek tonen we hoe de programmeeromgeving Arduino werkt en hoe die kan gebruikt worden met ESP-hardware. Aan de hand van voorbeelden en projecten verduidelijken we hoe de ESP-hardware kan worden toegepast.

- We beschrijven de opbouw van ESP-experimenten.
- We tonen hoe displays worden aangesloten.
- Eeprom_24c16 demonstreert hoe een externe geheugenchip wordt gebruikt om getallen en tekst op te slaan en weer uit te lezen.
- Project Klok is de eenvoudige implementatie van een digitale klok.
- We geven een handleiding voor het gebruik van temperatuur- en luchtvochtigheidsensoren.
- Webclient laat zien hoe de ESP32 gegevens van het internet kan halen.
- Met Webserver kunnen we op afstand met een pc of smartphone gegevens halen uit een ESP32, sensoren uitlezen en toestellen aan- en uitschakelen.
- Een NTP-klok haalt tijdinformatie van Tijdserver. Die geeft zowel de tijd als het aantal seconden sinds 1 januari 1900. Het boek beschrijft hoe hieruit de tijd kan worden berekend.
- In het laatste hoofdstuk gebruiken we het internet om nauwkeurige weerinformatie en voorspellingen op te vragen.

De voorbeeldprogramma's kunnen gedownload worden van de website van Elektor.

ISBN 978-90-5381-296-9

9 789053 812969

NUR950
www.elektor.nl
Elektor International Media B.V.