

Leren programmeren, meten  
en sturen met de Arduino



# Leren programmeren, meten en sturen met de Arduino

Jacco de Jong

## Leeswijzer

### ACTIE

### SKETCH

### FUNCTIES EN WAARDEN

### OEFENING

### OEFENING\*

stapsgewijs opzetten van hard- en software  
programmaregels

uitleg van programmaonderdelen

creatief aan de slag

oefeningen gemerkt met een \* zijn voor differentiatie;

overleg met je docent of je die moet maken

ISBN 978 90 5752 361 8 / NUR 124

Omslagontwerp: Hans Geel

Illustraties en fotografie: J.J. de Jong

Redactie: C. Casander / Henk Pel

Opmaak: Henk Pel

© 2017 Brinkman Uitgeverij, Amsterdam

Gehele of gedeeltelijke overneming of reproductie van de inhoud van deze uitgave, op welke wijze dan ook, zonder voorafgaande schriftelijke toestemming van de auteursrechthebbende is verboden, behoudens de beperkingen bij de wet gesteld. Het verbod betreft ook gehele of gedeeltelijke bewerking. De uitgever is met uitsluiting van ieder ander gerechtigd de door derden verschuldigde vergoedingen voor kopiëren, als bedoeld in artikel 17 Auteurswet 1912 en in het KB van 20 juni 1974 (Stb. 351, 1974) ex artikel 16b Auteurswet 1912, te innen en/of daartoe in en buiten rechte op te treden.

Correspondentie inzake overneming of reproductie richten aan:

Brinkman Uitgeverij, Postbus 59686, 1040 LD Amsterdam

[www.brinkman-uitgeverij.nl](http://www.brinkman-uitgeverij.nl)

tel. 020-4120970, fax 020-4120972

e-mail: [info@brinkman-uitgeverij.nl](mailto:info@brinkman-uitgeverij.nl)

# Inhoud

## Inleiding—1

Leerdoelen 1

## 1 Aftrap—3

- 1.1 Aanschaf 3
- 1.2 Arduino UNO R3 of Arduino MEGA 2560? 4
- 1.3 Poorten en voeding 5
- 1.4 Systeemeisen 6
- 1.5 Software 6
- 1.6 Arduino met pc verbinden 10
- 1.7 Software instellen 11

## 2 Eerste project—13

- 2.1 Knipperende LED 13
- 2.2 Sketch 15
- 2.3 Functies en waarden 17
- 2.4 Verifieer versus Upload 18
- 2.5 Oefeningen 19
  - 2.5.1 *Oefening – Knippersnelheid aanpassen* 19
  - 2.5.2 *Oefening – Sketch opslaan en openen* 19
  - 2.5.3 *Oefening – Zonder USB-kabel* 19

## 3 Werken met componenten—21

- 3.1 Breadboard 21
- 3.2 LED 22
- 3.3 Halfgeleider 23
- 3.4 Weerstand 24
- 3.5 LED en weerstand aansluiten 25
- 3.6 Oefeningen 26
  - 3.6.1 *Oefening – Twee LEDs knipperen* 27
  - 3.6.2 *Oefening – Versnelde LED* 29
  - 3.6.3 *Oefening – Externe LEDs zelfstandig laten knipperen* 29
  - 3.6.4 *Oefening – Knight Rider* 30
  - 3.6.5 *Oefening\* – Morsecode* 30

\* Oefeningen gemerkt met een asterisk zijn voor differentiatie.

## **4 Digitale input en output—31**

- 4.1 Drukknopschakelaar 31
- 4.2 HIGH en LOW 32
- 4.3 In- en uitgangen (digitaal) 33
- 4.4 Pull-down-weerstand 34
- 4.5 Pull-up-weerstand 35
- 4.6 LED in- en uitschakelen 35
- 4.7 Oefeningen 38
  - 4.7.1 *Oefening – 30 seconden uitschakelvertraging* 38
  - 4.7.2 *Oefening – Looplicht van acht LEDs op schakelaar* 39
  - 4.7.3 *Oefening\* – Veranderende functie van de schakelaar* 39
  - 4.7.4 *Oefening\* – Toggle-schakelaar* 39

## **5 Analoge sensors – deel 1—40**

- 5.1 Spanningsdeler 40
- 5.2 Potentiometer 41
- 5.3 Regel knippersnelheid met potentiometer 42
- 5.4 Lichtsensor LDR 43
- 5.5 LDR-spanningsdeler 44
- 5.6 Oefeningen 47
  - 5.6.1 *Oefening – Automatische verlichting* 47
  - 5.6.2 *Oefening\* – Volkomen rood kruispunt* 48

## **6 Rekenen op Seriële monitor—49**

- 6.1 Vermenigvuldigen 49
- 6.2 Optellen 52
- 6.3 Stopwatch 53
- 6.4 Oefeningen 54
  - 6.4.1 *Oefening – Aftellende klok* 54

## **7 Analoge sensors – deel 2—55**

- 7.1 Temperatuursensor LM35 55
- 7.2 Temperatuurmeting 57
- 7.3 Druksensor 60
- 7.4 Drukmeting 62
- 7.5 Gassensor MQ-x 65
- 7.6 Gasmeting (mg/l en ppm) 67
- 7.7 Oefeningen 67
  - 7.7.1 *Oefening – Lumenmeter* 67
  - 7.7.2 *Oefening – Tellen* 67
  - 7.7.3 *Oefening\* – Alcoholslot* 67

## **8 Digitale sensors—68**

- 8.1 DHT11 temperatuur- en relatieve-luchtvochtigheidssensor 68
- 8.2 Meten met de DHT11 68

## **9 Library—70**

- 9.1 Wat is een library? 70
- 9.2 Library installeren (standaard) 71
- 9.3 Library toevoegen – include 72
- 9.4 Library troubleshooting 73
- 9.5 Library installeren (custom) 74
- 9.6 Library gebruiken – vochtmeting met DHT11 76
- 9.7 Samenvatting – werken met libraries 81
- 9.8 Oefeningen 81
  - 9.8.1 Oefening – DHT11 tot op twee decimalen nauwkeurig 81
  - 9.8.2 Oefening\* – Thermostaat en hygrometer 81

## **10 Displays—82**

- 10.1 LCD-displays met 16 pinnen 82
- 10.2 LCD-display 1602 84
  - 10.2.1 Wat is LCD-display 1602? 84
  - 10.2.2 Tekst weergeven op LCD-display 1602 84
- 10.3 LCD-display 2004 86
  - 10.3.1 Wat is LCD-display 2004? 87
  - 10.3.2 Kamertemperatuur en relatieve vochtigheid op LCD-display 2004 87
- 10.4 I<sup>2</sup>C bussysteem 90
  - 10.4.1 Wat is het I<sup>2</sup>C-bussysteem? 90
  - 10.4.2 Tekst op display tonen via I<sup>2</sup>C-interface 91
- 10.5 Oefeningen 94
  - 10.5.1 Oefening – Backlight 94
  - 10.5.2 Oefening – Vergelijk sensoren 94
  - 10.5.3 Oefening\* – Verschil 94

## **11 Actuatoren—95**

- 11.1 Relais 95
- 11.2 Transistor 96
- 11.3 Relais schakelen met transistor 100
- 11.4 FET 101
- 11.5 PWM 102
  - 11.5.1 Frequentie en duty cycle 103
  - 11.5.2 Duty cycle van PWM regelen met potentiometer 104
  - 11.5.3 PWM op de oscilloscoop 105

|           |  |            |
|-----------|--|------------|
| 11.6      | Oefeningen                                       | 106        |
| 11.6.1    | Oefening – LED-dimmer                            | 106        |
| 11.6.2    | Oefening – Lichtdimmer                           | 106        |
| 11.6.3    | Oefening* – Lichtdimmer in balans                | 106        |
| <b>12</b> | <b>Elektromotor—</b>                             | <b>107</b> |
| 12.1      | Elektromotor schakelen met relais                | 107        |
| 12.2      | Elektromotor schakelen en regelen met FET        | 108        |
| 12.3      | Servomotor                                       | 111        |
| 12.3.1    | Werking servomotor                               | 112        |
| 12.3.2    | Servomotor aansturen met Arduino                 | 114        |
| 12.4      | Stappenmotor                                     | 115        |
| 12.4.1    | Werking van de stappenmotor                      | 115        |
| 12.4.2    | Soorten stappenmotoren                           | 117        |
| 12.4.3    | Stappenmotor aansturen met Arduino               | 120        |
| 12.4.4    | Stappenmotoraansturing zichtbaar gemaakt         | 124        |
| 12.5      | Oefeningen                                       | 125        |
| 12.5.1    | Oefening – Ventilator regelen met een DHT11      | 125        |
| 12.5.2    | Oefening – Airco sweeper                         | 126        |
| 12.5.3    | Oefening – Stuurinrichting met stappenmotor      | 126        |
| <b>13</b> | <b>Data loggen—</b>                              | <b>127</b> |
| 13.1      | Data loggen op SD-kaart                          | 127        |
| 13.2      | Oefeningen                                       | 133        |
| 13.2.1    | Oefening – Relatie temperatuur en licht          | 133        |
| <b>14</b> | <b>Telemetrie—</b>                               | <b>134</b> |
| 14.1      | Telemetrie in de praktijk                        | 134        |
| 14.2      | Datatransmissie via 433 MHz                      | 135        |
| 14.2.1    | Zenden op 433 MHz                                | 136        |
| 14.2.2    | Ontvangen op 433 MHz                             | 139        |
| 14.2.3    | Oefening – Professioneel weerstation via 433 MHz | 141        |
| 14.3      | Datatransmissie via het internet                 | 142        |
| 14.3.1    | Ethernet Shield W5100                            | 142        |
| 14.3.2    | Sensorwaarden monitoren op webpagina             | 146        |
| 14.4      | Bluetooth  | 150        |
| 14.4.1    | Bluetooth-module HC-06                           | 150        |
| 14.4.2    | Licht schakelen met smartphone                   | 150        |
| 14.4.3    | Verkeerslicht schakelen met smartphone           | 153        |



|           |   |     |
|-----------|---|-----|
| 14.5      | Oefeningen  | 156 |
| 14.5.1    | <i>Oefening – E-Health</i>  | 156 |
| 14.5.2    | <i>Oefening – Analoge regeling</i>  | 156 |
| 14.5.3    | <i>Oefening* – Arduino Webserver met beeld</i>                            | 157 |
| <b>15</b> | <b>Overige sensoren—158</b>   |     |
| 15.1      | Ultrasonische afstandsmeter   | 158 |
| 15.2      | Geluidssensor   | 159 |
| 15.3      | Vochtgehaltesensor  | 159 |
| 15.4      | Regensensor   | 160 |
| 15.5      | BMP180 barometersensor  | 160 |
| 15.6      | Windsnelheidsmeter (anemometer)   | 160 |
| 15.7      | Shield  | 161 |
| 15.8      | Oefeningen  | 163 |
| 15.8.1    | <i>Oefening – Afstandsmeting en alarm met ultrasonische afstandsmeter</i> | 163 |
| 15.8.2    | <i>Oefening – VU-meter</i>  | 163 |
| 15.8.3    | <i>Oefening* – Weerstation voltooien</i>                                  | 164 |
| <b>16</b> | <b>Arduino-programmeertaal—165</b>  |     |
| 16.1      | Structuur   | 165 |
| 16.1.1    | <i>Structuur – control</i>  | 166 |
| 16.1.2    | <i>Structuur – divers</i>   | 169 |
| 16.1.3    | <i>Structuur – rekenkundige bewerkingen</i>                               | 171 |
| 16.2      | Waarden – variabelen en constanten  | 172 |
| 16.2.1    | <i>Variabelen</i>   | 172 |
| 16.2.2    | <i>Typen variabelen</i>   | 173 |
| 16.2.3    | <i>Constanten</i>   | 175 |
| 16.3      | Functies  | 176 |
| 16.3.1    | <i>Digitale input- en outputfuncties</i>                                  | 176 |
| 16.3.2    | <i>Analoge input- en outputfuncties</i>                                   | 178 |
| 16.3.3    | <i>Tijdfuncties</i>   | 179 |
| 16.3.4    | <i>Wiskundige en goniometrische functies</i>                              | 180 |
| 16.3.5    | <i>Random waarden</i>   | 181 |
| 16.3.6    | <i>Datacommunicatie</i>   | 182 |

## **Eindoefening—184**

**Appendix 1 Troubleshooting—185**

- A1.1 Hardware – algemeen 185
- A1.2 Hardware – voedingsspanning 185
- A1.3 Hardware – spanning op analoge/digitale input 186
- A1.4 Hardware – GND 186
- A1.5 Hardware – MQ-x-gassensoren 186
- A1.6 Hardware – Ethernet Shield W5100 187
- A1.7 Software – Installatiefouten 188
- A1.8 Software – Foutmeldingen 188
- A1.9 Software – Aanhalingstekens 190
- A1.10 Software – Onrealistische sensorwaarden 191

**Appendix 2 Datasheets—192**

- A2.1 Datasheet weerstanden 192
- A2.2 Datasheet NPN-Transistor BC546..560 192
- A2.3 Datasheet N-Channel FET BUZ11 194
- A2.4 Datasheet LDR T9-serie 194
- A2.5 Datasheet DHT11 Sensor 195
- A2.6 Datasheet LM35 Sensor 195
- A2.7 MQ-3 Gassensor 196
- A2.8 Drukknopschakelaar 199

**Appendix 3 ASCII-tabel—200****Appendix 4 Onderdelenlijst—201****Register—203**

# Inleiding

Voor je ligt een boek waarin je op een praktische manier leert hoe je de Arduino moet programmeren en kunt toepassen in de praktijk. Het is de bedoeling dat je de grenzen van deze minicomputer opzoekt en vooral, dat je die grenzen passeert. Daar hoeft je echt niet veel moeite voor te doen. Want uit ervaring weet ik dat het leren snel overgaat in doen. Je zult merken dat er zoveel inspirerende en innoverende ideeën komen opborrelen, dat het werken met de Arduino bijna verslavend wordt. Laat bijvoorbeeld een verlicht reclamebord dansen op de maat van de muziek, maak een robot die anderen doet verbazen, laat Knight Rider herleven met lopende LEDs, bedenk het maar. Met de Arduino is echt alles mogelijk, dus overstijg vooral jezelf! Succes!

Jacco de Jong

## Leerdoelen

- installeren van de Arduino-software
- kennis van programmeertaal
- bouwen en aanpassen van Arduino-sketch
- hergebruik van programmacode
- kennis van analoge en digitale elektronica
- kennis van actieve en passieve elektronische componenten
- kennis van mechatronische componenten
- kennis van seriële communicatie en LCD-displays
- toepassen van hardware en software in projecten
- data loggen
- datatransmissie en telemetrie

# Extra materiaal

De bestanden van de sketches staan als gezipd bestand bij de titelinformatie op [shop.brinkman-uitgeverij.nl](http://shop.brinkman-uitgeverij.nl).

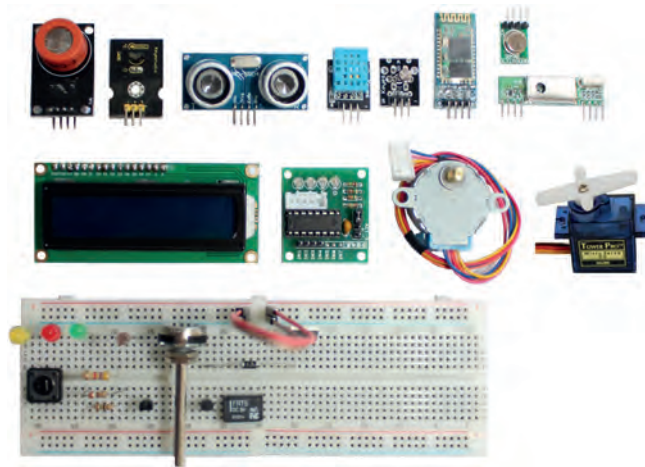
Voor de docenten is extra materiaal verkrijgbaar via [www.brinkman-uitgeverij.nl](http://www.brinkman-uitgeverij.nl).

# 1 Afttrap

De Arduino is ooit ontworpen voor ontwerpers en kunstenaars in de Italiaanse mode-industrie: de kleine computer moest LED-verlichting in kleding aansturen. Inmiddels is de Arduino een breed toegepaste microcontroller met een behoorlijke ‘crashbestendigheid’. Hij is gemakkelijk te doorgronden en ideaal als leerobject. Sterker nog, voor velen is de Arduino verslavend: hij nodigt uit tot meer en tot het verleggen van grenzen. De toepassingsmogelijkheden van de Arduino zijn dan ook bijna grenzeloos! Bovendien is het programmeren van een Arduino niet moeilijk. Je hebt er weinig of geen programmeerervaring voor nodig.

## 1.1 Aanschaf

Om met dit boek te kunnen werken heb je een Arduino nodig. Je kunt er een aanschaffen in een winkel of online. Prijzen variëren van 10 tot 25 euro. Het is aan te raden om een complete starterkit aan te schaffen met daarin alle onderdelen uit de Onderdelenlijst in Appendix 3. Een starterkit inclusief een Arduino UNO R3 kost rond de 50 euro. Een starterkit met de Arduino MEGA 2560 is iets duurder.



Afbeelding 1.1 – Starterkit met sensoren en actuatoren.

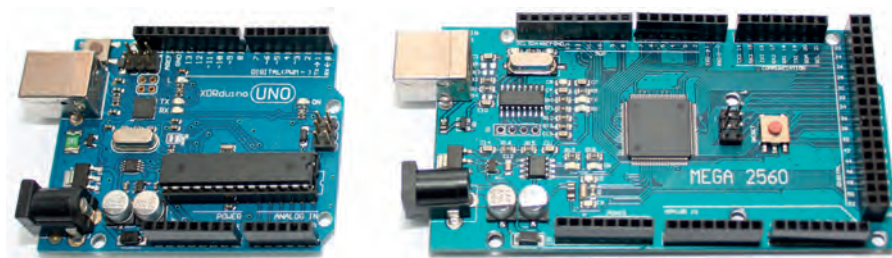
In de starterkit vind je diverse sensoren en actuatoren. In vrijwel elk Arduino-project kom je die tegen. Vrij vertaald zijn het ‘voelers’ en ‘doeners’. Een temperatuursensor voelt bijvoorbeeld hoe heet of hoe koud het is. Afhankelijk van de gemeten temperatuur stuurt de Arduino een actuator aan. Die actuator schakelt dan bijvoorbeeld de verwarming aan of uit.

Een sensor is een elektrisch ding dat een natuurkundige grootheid *meet*, zoals temperatuur, relatieve vochtigheid, druk, lichtintensiteit, gasconcentratie, straling enzovoort. Een actuator is een elektrisch ding dat iets *doet*. Zoals een LED (geeft licht), een motor (draait rond), een elektromagnetische klep (kan open of dicht), een relais (zet verwarming aan of uit) enzovoort. Afhankelijk van je project bepaal je welke sensor(s) en actuator(en) je nodig hebt.

## 1.2 Arduino UNO R3 of Arduino MEGA 2560?

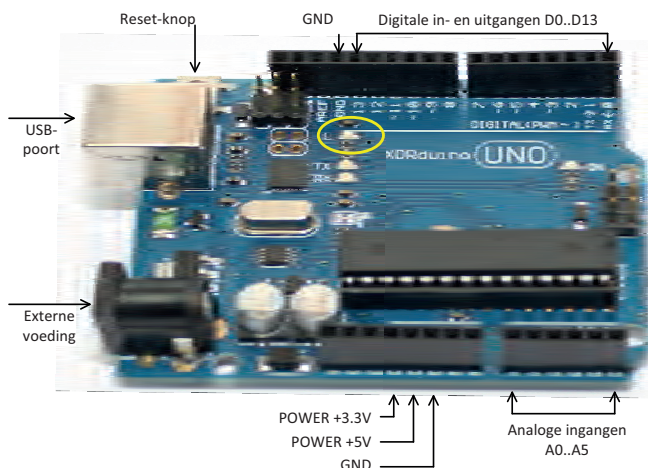
De Arduino wordt in meerdere uitvoeringen geleverd. De bekendste zijn Arduino UNO R3 en de duurdere Arduino MEGA 2560. Beide worden geleverd als print of 'board' met daarop een paar onderdelen. In een voltooid project wordt de compacte print gewoonlijk ingebouwd in een kastje of in een apparaat, maar in dit boek werk je met het 'kale' board.

Alle hier beschreven voorbeelden zijn gebaseerd op de Arduino UNO R3, maar werken ook op de Arduino MEGA 2560. De Arduino MEGA 2560 kan hetzelfde als de UNO R3, maar heeft meer in- en uitgangen voor sensoren en actuatoren. In latere projecten kunnen deze extra in- en uitgangen goed van pas komen, voor dit boek maakt het niet uit welke van de twee Arduino's je kiest. Waar nodig worden in dit boek de verschillen tussen de beide boards toegelicht; dat is onder andere het geval bij het instellen van de software.



Afbeelding 1.2 -Twee Arduino-boards: links de Arduino UNO R3 en rechts de Arduino Mega 2560.

## 1.3 Poorten en voeding



Afbeelding 1.3 – Aansluitingen en poorten van de Arduino UNO R3. Led L is geel omcirkeld.

Heb je je keuze gemaakt, maak jezelf dan eerst vertrouwd met het board. Afbeelding 1.3 toont het board van de Arduino UNO R3. Rechts naast de USB-poort zie je de (vierkante) microprocessor. Verder zie je vooral poorten, zoals ingangen (voor het aansluiten van sensoren) en uitgangen (voor het aansluiten van actuatoren). Om te kunnen functioneren heeft de Arduino voeding nodig. Die voeding krijgt hij van je laptop of pc via de USB-poort; de poort 'Externe voeding' hoef je dan niet aan te gebruiken (zie kader).

| Aanduiding poort | Omschrijving                           |
|------------------|--|
| USB              | USB-poort voor computeraansluiting     |
| Externe voeding  | Batterij of netvoeding (6 tot 20 volt) |
| POWER +3.3V      | 3,3 volt uitgang voor externe sensoren |
| POWER +5V        | 5 volt uitgang voor externe sensoren   |
| DIGITAL Do..D13  | Digitale in- en uitgangen (14 pinnen)  |
| ANALOG IN Ao..A5 | Analoge ingangen (6 pinnen)            |
| GND              | 0 volt of massa (ground)               |

Tabel 1.1 – Belangrijkste poorten van de Arduino UNO R3

---

### Externe voeding

Om los van de laptop of pc te kunnen functioneren kan de Arduino via de poort ‘Externe voeding’ worden gevoed met een 9 volt blokbatterij of een netadapter van 5 tot 12 volt. Volt wordt vaak afgekort tot de letter V of VDC. Ook een kleine 12 volt accu is geschikt; gebruik in dat geval altijd een zekering van circa 1 ampère. Als je een externe voeding gebruikt, wordt de voeding via USB automatisch uitgeschakeld. Je kunt de USB-kabel dus gewoon laten zitten.

---

## 1.4 Systeemeisen

De Arduino moet geprogrammeerd worden. Dat doe je op je pc of laptop met de Arduino-software. De Arduino-software draait onder Windows, Mac OS X 10.7 Lion of later, Linux 32-bits, Linux 64-bits en Linux ARM. In dit boek wordt uitgegaan van Windows 10, maar Arduino-software draait op elke x86- en x64-versie van Windows vanaf Windows XP.

## 1.5 Software

De Arduino-software is open source en gratis te downloaden van:

*<https://www.arduino.cc/en/Main/Software>*

In een paar stappen zet je de Arduino-software op je laptop of pc. Als voorbeeld is gekozen voor een systeem met Windows 10, omdat dit het meest gebruikte besturingssysteem is.

**NB** Computers op scholen, in openbare ruimtes en bij bedrijven zijn vaak zo geconfigureerd, dat alleen een beheerder nieuwe software mag installeren. Mogelijk dat het je wel lukt om software op zo’n computer te installeren, maar het kan zijn dat de computer – na het opnieuw opstarten – weer terugvalt in zijn basisconfiguratie. Dat wil zeggen dat de geïnstalleerde software en bijbehorende projecten verdwenen zijn. Informeer hiernaar voordat je de software installeert.

### ACTIE

- Start je browser en ga naar:

*<https://www.arduino.cc/en/Main/Software>*

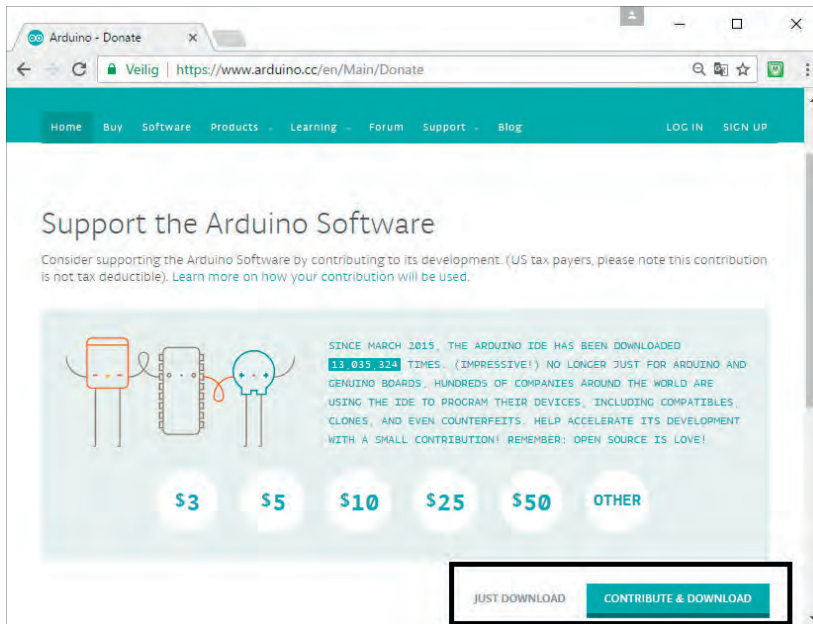
Selecteer het te installeren besturingssysteem. In dit geval *Windows Installer* (omkaderd in afbeelding 1.4).





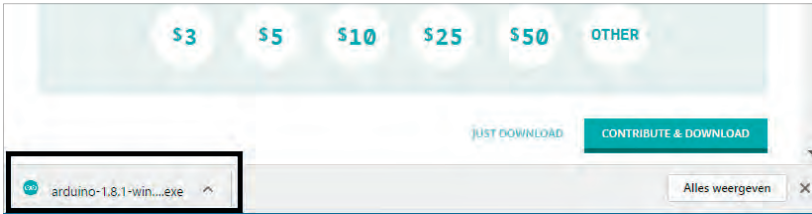
Afbeelding 1.4 – Keuze besturingssysteem.

- De software is in principe gratis, maar een vrijwillige donatie wordt op prijs gesteld (zie afbeelding 1.5). Met een donatie draag je bij aan het verder ontwikkelen van de software. Wil je niet doneren, klik dan rechts onderin op *Just download*.



Afbeelding 1.5 – Vrijwillige bijdrage.

- Linksonder kun je het downloadproces volgen (zie omkadering in afbeelding 1.6). Zodra de download is voltooid klik je op de bestandsnaam om het installatieproces te starten.

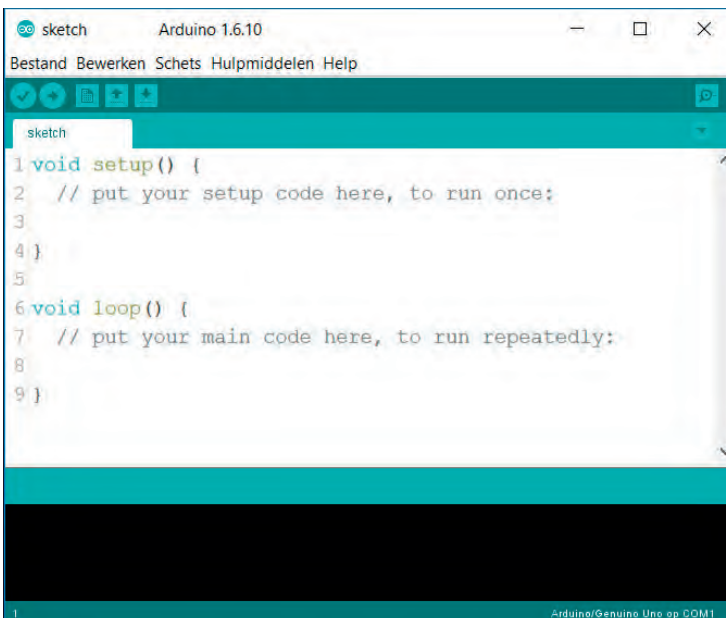


Afbeelding 1.6 – Het te installeren bestand is gedownload.

- Na de installatie zal het programma automatisch starten en zichzelf configureren (zie afbeeldingen 1.7 en 1.8). Mogelijk wordt ook gevraagd of hulpprogramma's (zoals USB-stuurprogramma's) geïnstalleerd mogen worden. Beantwoord die vragen met *Ja*. Na het configureren sluit het programma automatisch.



Afbeelding 1.7 – Starten van Arduino-software.



Afbeelding 1.8 – Openingsscherm van Arduino-software.

- Start de Arduino-software nu zelf. Dat kan op verschillende manieren, bijvoorbeeld (zie afbeelding 1.9):
  - Klik op de Windows *Start*-knop en klik in de lijst op *Arduino*.
  - Klik op de icoon of tegel op het bureaublad.



Afbeelding 1.9 – Opstarten via Windows Start.

Telkens na het starten zal de Arduino-software op zoek gaan naar software-updates. Daartoe maakt de software verbinding met het internet. Mogelijk krijg je dan een melding van de Windows Firewall. Klik in dat geval op *Toegang toestaan*.

- Je moet nu nog twee dingen doen: hardware aansluiten en software instellen, zie volgende paragrafen.

## 1.6 Arduino met pc verbinden

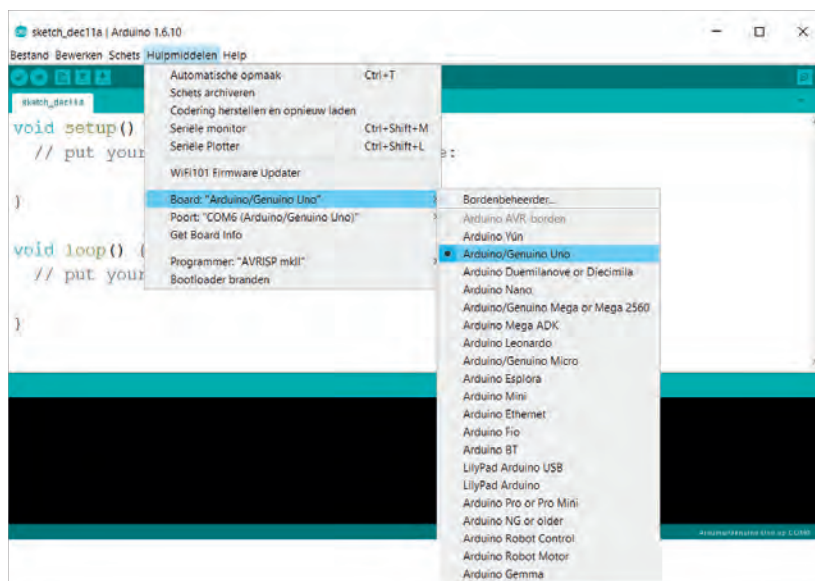


Afbeelding 1.10 – USB-verbinding tussen Arduino UNO en laptop.

Om de Arduino vanaf je pc of laptop te programmeren, moeten ze met elkaar kunnen communiceren. Daartoe verbind je de Arduino via een USB-kabel met je pc of laptop. Omdat de Arduino via die kabel ook zijn voeding krijgt, licht de ON-LED op het Arduino-board op. Enkele seconden later is ook de verbinding met de computer tot stand gekomen. Mogelijk wordt tijdens de eerste keer dat je de software gebruikt een USB-stuurprogramma gedownload en geïnstalleerd, dit kan enkele minuten duren.

**NB** Twee andere LEDs die mogelijk oplichten zijn TX en RX en hebben te maken met datacommunicatie via de USB-poort of een andere communicatiepoort.

## 1.7 Software instellen



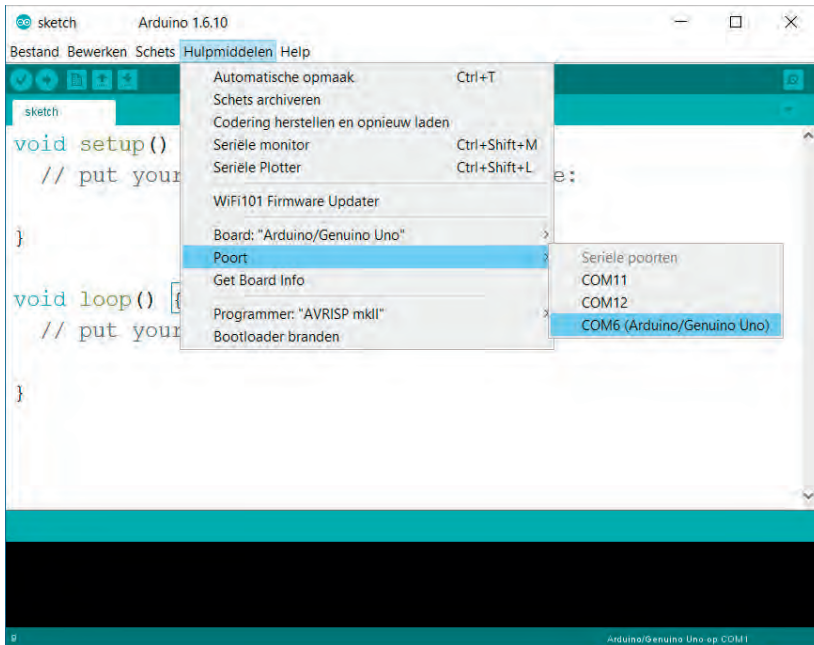
Afbeelding 1.11 – Instellen van de juiste Arduino.

Na het installeren van de Arduino-software en het aansluiten van de hardware moeten drie instellingen in de software worden gemaakt:

- instellen type Arduino;
- instellen COM-poort om met de Arduino te communiceren;
- inschakelen regelnummering.

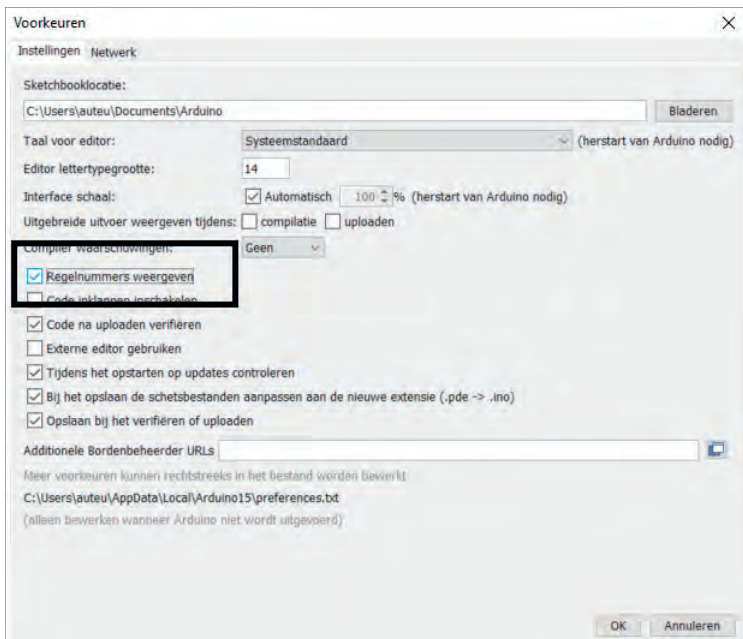
Start de Arduino-software, open het menu *Hulpmiddelen* en kies *Board*. Je ziet nu een lijst van verschillende Arduino's (zie afbeelding 1.11). Selecteer *Arduino/Genuino UNO*. Heb je een Arduino MEGA 2560, selecteer dan *Arduino/Genuino Mega or Mega 2560*.

Klik vervolgens in het menu *Hulpmiddelen* op *Poort* (zie afbeelding 1.12). Doorgaans is de juiste poort al ingesteld. In het afgebeelde voorbeeld is dat COM6. Dit betekent dat de USB-poort zich gedraagt als 'COM'municatie-poort nummer 6. Deze instelling is vooral bedoeld voor wie twee of meer Arduino's op de computer heeft aangesloten: elk heeft dan zijn eigen COM-poort en in dit menu kies je welke Arduino je met de software wilt aansturen.



Afbeelding 1.12 – Instelling van de juiste COM-poort.

De derde instelling betreft het inschakelen van de regelnummering in de sketches. Regelnummers zijn handig, bijvoorbeeld bij het lokaliseren van fouten. Je zult dat later zien. Open het menu *Bestand* en klik op *Voorkeuren* (zie afbeelding 1.13). Zet een vinkje bij de optie *Regelnummers weergeven* en sluit het venster.



Afbeelding 1.13 – Voorkeuren instellen.



## 2 Eerste project

In dit hoofdstuk maak je kennis met de Arduino-programmeertaal. Met de programmeertaal maak je een programma of 'schets', waarin je de Arduino vertelt wat hij moet doen. Je maakt de schets op je pc en 'stuurt' hem daarna naar de Arduino. In plaats van de Nederlandse termen 'schets' en 'sturen' zul je vaker de Engelse termen tegenkomen: 'sketch' en 'upload'. In dit hoofdstuk maak en upload je een sketch die een LED laat knipperen.

### 2.1 Knipperende LED

Op het Arduino-board zitten vier LEDs. Wat een LED precies is komt later ter sprake. Een van de vier LEDs brandt al: deze LED is gemarkeerd *ON*. Een andere LED is aangesloten op de digitale uitgang D13 en is gemarkeerd *L*. In afbeelding 1.3 is deze LED geel omcirkeld. Met een simpele sketch kun je deze LED laten knipperen of (in het Engels) 'blink'. De sketch die je hiervoor nodig hebt heet **Blink** en is al als voorbeeldsketch op je pc aanwezig.

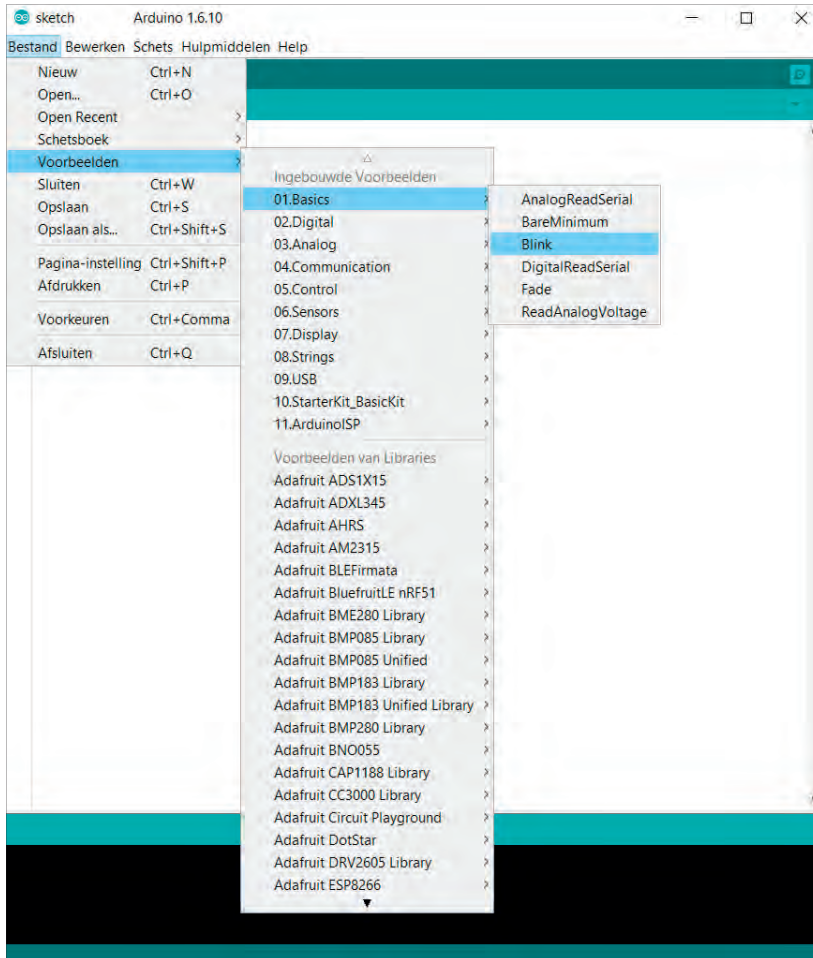
#### ACTIE

- Zorg dat er geen losse onderdelen rond de Arduino liggen. Als die per ongeluk onder het Arduino-board terechtkomen kunnen ze kortsluiting veroorzaken en de elektronica beschadigen.
- Verbind de Arduino via de USB-kabel met je pc.
- Start de Arduino-software.
- Kies de sketch: open het menu *Bestand > Voorbeelden > 01.Basics > Blink* (zie afbeelding 2.1).
- Vervolgens opent een nieuw venster met daarin de sketch (zie afbeelding 2.2). De afgebeelde sketch bevat Nederlandstalige uitleg, op het scherm is de uitleg in het Engels.
- In de groene menubalk van het venster zie je een paar knoppen. Beweeg er met je muis overheen om de betekenis van de knoppen te lezen. Klik op de knop *Upload*.
- Een groene voortgangsbalk geeft het compileren en uploaden van de sketch naar de Arduino weer. Compileren is het vertalen van de sketch naar de machinetaal van de microprocessor op het Arduino-board.
- Tijdens het uploaden knipperen op het Arduino-board de communicatie-LEDs TX en RX. Wacht tot de LEDs TX en RX niet meer knipperen.
- De sketch draait nu: de LED op D13 knippert 1 seconde aan, 1 seconde uit en zo voort.

Tijdens het uploaden wordt de sketch in het geheugen van de Arduino geplaatst. Dit geheugen is flash-geheugen dus de sketch blijft bewaard, ook als de voeding wegvalt. Probeer dit uit:

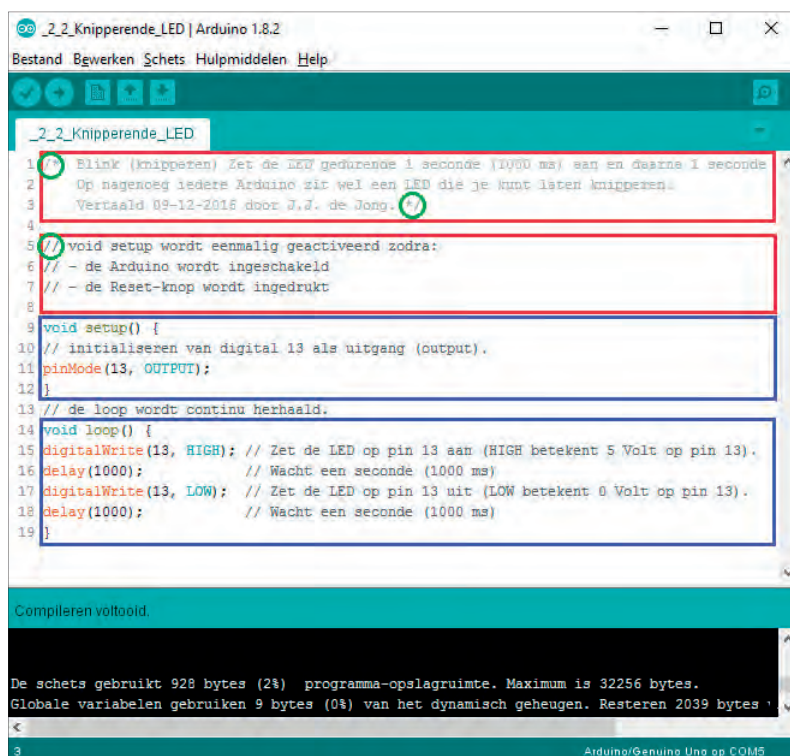
## ACTIE

- Trek de USB-kabel uit de Arduino, de LED dooft.
- Wacht een paar seconden en steek de USB-stekker er weer in: de sketch start opnieuw en de LED knippert weer.



Afbeelding 2.1 – Open de sketch **Blink**.





Afbeelding 2.2 – Sketch (programma) voor knipperende LED.

## 2.2 Sketch

In dit hoofdstuk wordt de structuur van een sketch uitgelegd. Elke sketch bestaat uit minimaal twee functies: *void setup* en *void loop*. In afbeelding 2.2 zijn die blauw omkaderd. Tussen de accolades staan statements. Een statement is een regel met code en aan het eind een puntkomma. Een statement bevat onder meer variabelen en andere functies. In de *void setup* zie je bijvoorbeeld een statement met de functie *pinMode()*:

```
pinMode(13, OUTPUT);
```

In *void loop* zie je onder andere een statement met de functie *delay()*:

```
delay(1000);
```

De Arduino-programmeertaal is hoofdlettergevoelig. Let dus goed op de juiste notatie van de hoofdletter in bijvoorbeeld *pinMode*. Spaties daarentegen hebben geen betekenis voor de programmeertaal. Je kunt ze gebruiken om je sketch visueel en overzichtelijk te ordenen. Dat kan op verschillende manieren, zoals de sketches in dit boek laten zien.

Als geheugensteun voor jezelf en om andere programmeurs uit te leggen hoe de sketch werkt, maak je gebruik van commentaarregels. Daar zijn twee manieren voor:

- Een commentaar van meerdere regels zet je tussen de tekens `/*` en `*/`. Zie bijvoorbeeld de eerste drie rood omkaderde regels in afbeelding 2.2. De tekens zijn groen omcirkeld. Tekstregels zijn alleen voor de programmeur zichtbaar en worden niet geüpload naar de Arduino. Ze zullen dus geen geheugenruimte van de Arduino in beslag nemen.
- Gaat het om één enkele commentaarregel (zoals regel 5) begin dat commentaar dan met de tekens `//` (groen omcirkeld). Er zijn geen eindtekens, want het einde van de regel is automatisch het eindteken. Ook deze commentaarregels worden niet naar de Arduino geüpload.

Alles wat tussen de accoladetekens van de functie *void setup* staat wordt één keer uitgevoerd: telkens als het programma opstart of nadat op de Reset-knop op het Arduino-board is gedrukt. In de *void setup* bepaal je onder andere welke pinnen je als in- en uitgangen gaat gebruiken. In het voorbeeld wordt pin D13 met de functie *pinMode* als *OUTPUT* (uitgang) ingesteld. Je zult zien dat de Arduino-programmeertaal op 'vreemde' plaatsen hoofdletters gebruikt, zoals de M in *pinMode*. De bedoeling daarvan is om duidelijk onderscheid te maken tussen twee woorden: pin en Mode. Om dezelfde reden zie je in afbeelding 2.2 dat tekst, commando's en waarden in verschillende kleuren worden weergegeven.

```
void setup() {
// Initialiseer digitale poort D13 als uitgang (OUTPUT):
pinMode(13, OUTPUT);
}
```

Alles wat tussen de accoladetekens van de functie *void loop* staat (tweede blauwe omkadering in afbeelding 2.2) wordt continu herhaald. Loop betekent hier 'herhaallus'. In elke herhaallus worden door de Arduino-processor statements uitgevoerd en waarden gelezen. In het voorbeeld is dat het sturen van een HIGH of LOW naar uitgang 13 en daarna 1 seconde wachten. HIGH betekent hier 5 volt op pin 13, LOW betekent 0 volt op pin D13. De precieze betekenis van de functies en waarden lees je in het volgende hoofdstuk.

```
// de loop wordt continu herhaald.
void loop() {
digitalWrite(13, HIGH);    // Zet de LED op pin 13 AAN
delay(1000);               // Wacht een seconde (1000 ms)
digitalWrite(13, LOW);     // Zet de LED op pin 13 UIT
delay(1000);               // Wacht een seconde (1000 ms)
}
```

Samenvattend kun je de sketch **Blink** als volgt lezen: de sketch begint met het definiëren van uitgang 13 als digitale output (regel 9 in afbeelding 2.2). In regel 13 begint de herhaallus. Met het commando 'zet uitgang 13 HOOG' wordt de LED op pin D13 AAN gezet (regel 15). Daarna begint de functie *delay(1000)*, ofwel 'wacht

1000 milliseconden' (regel 16). Tijdwaarden worden in de functie *delay* uitgedrukt in milliseconden (1000 milliseconden is 1 seconde). Na de pauze stuurt de microprocessor het commando 'zet uitgang 13 LAAG' (regel 17). De LED is nu uit. Daarna wordt het volgende statement *delay(1000)* gelezen (regel 18): 'wacht 1000 milliseconden'. Na de pauze wordt de loop herhaald. De microprocessor stuurt weer het commando 'zet uitgang 13 HOOG' (regel 15), enzovoort.

## 2.3 Functies en waarden

Functies en waarden in de sketch bepalen wat de Arduino moet doen. De Arduino-programmeertaal kent tientallen functies en waarden, waarvan je de meeste in dit boek zult leren kennen. In de voorbeeldsketch zijn de volgende functies en waarden (constanten en variabelen) gedefinieerd en gebruikt:

### ► *pinMode* (poort, OUTPUT / INPUT)

Elk van de digitale poorten D1..D13 kan fungeren als ingang of als uitgang. Het is zaak dat je voor elke digitale poort die je in je sketch gebruikt, aangeeft hoe die poort moet werken. Dat doe je met de functie *pinMode*. In de functie noteer je het nummer van de poort en de modus waarin die poort moet werken: als INPUT of als OUTPUT. INPUT en OUTPUT zijn constanten, ze zijn dus niet variabel. In de modus 'ingang' kun je er sensoren op aansluiten, zoals een schakelaar of temperatuursensor. In de modus 'uitgang' kun je er actuatoren op aansluiten, zoals een LED, een motor, een elektromagnetische klep enzovoort.

```
pinMode(13, OUTPUT);
```

### ► *digitalWrite* (poort, HIGH / LOW)

Deze functie stuurt een digitale poort aan. Tussen haakjes staan het nummer van de poort en de waarde die de bijbehorende pin krijgt: HIGH of LOW. HIGH en LOW zijn constanten, ze zijn dus niet variabel. LOW betekent digitaal 0 (nul), HIGH betekent digitaal 1. Zo wordt met *digitalWrite(13, HIGH)* pin D13 HOOG gezet, waardoor de daarop aangesloten LED oplicht.

```
digitalWrite(13, HIGH);  
digitalWrite(13, LOW);
```

### ► *delay(x)*

De functie *delay(x)* last een pauze *x* in. Tijdwaarden worden in de functie uitgedrukt in milliseconden en vermeld tussen haakjes: 1000 milliseconden is 1 seconde. Zodra de ingestelde pauze om is, gaat de Arduino verder naar het volgende statement.

```
delay(1000);
```

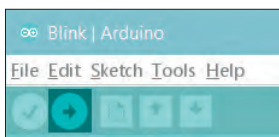
### ► parameters

De meeste Arduino-functies worden gevolgd door twee haakjes. Tussen die haakjes staan zogenaamde ‘parameters’. Parameters bepalen mede de uitkomst en werking van de functie. Zo zorgt de parameter ‘1000’ in de *delay*-functie voor een pauze van 1000 milliseconden. Sommige functies hebben één parameter, andere hebben er twee of meer. Afhankelijk van de functie kan een parameter een variabele zijn, een constante of tekst.

## 2.4 Verifieer versus Upload

Naast de *Upload*-knop zie je helemaal links in de menubalk ook een *Verifieer*-knop (knop met vinkje). De betekenis van beide knoppen is:

- *Verifieer*-knop – Hiermee kun je de sketch controleren, vóórdat je hem naar de Arduino uploadt; de verificatie geeft onder andere de grootte van de sketch aan, zodat je weet of hij wel of niet in het flash-geheugen van de Arduino zal passen. Eventuele fouten in de sketch worden onderin beeld gemeld. Corrigeer ze en herhaal de verificatie totdat de sketch foutloos is.
- *Upload*-knop – Hiermee stuur je de sketch naar de Arduino. Een eventueel reeds aanwezige sketch wordt automatisch uit het geheugen gewist. Na het uploaden start de sketch automatisch.



Afbeelding 2.3 – *Verifieer*-knop en *Upload*-knop.

Wat de harde schijf is voor je pc, is het ingebouwde flash-geheugen voor de Arduino. Het flash-geheugen van een Arduino UNO R3 is met 32 kilobytes tamelijk beperkt, temeer omdat hiervan slechts 28 kB beschikbaar is voor de sketches. De oefeningen in dit boek passen daar wel in, maar in de praktijk komen grote en complexe sketches voor, die te groot zijn voor een UNO R3. Met de *Verifieer*-knop kom je daar snel genoeg achter. De Arduino MEGA 2560 biedt dan de oplossing, want die heeft met 256 kB aanzienlijk meer ruimte.

Wat het werkgeheugen is voor je pc, is het SRAM-geheugen voor de Arduino. Dit geheugen wordt gewist zodra je de Arduino uitschakelt. Het SRAM-geheugen is 2 kB voor een UNO R3 en 8 kB voor een MEGA 2560.



Afbeelding 2.4 – Knipperen in de praktijk.

## 2.5 Oefeningen

In de volgende oefeningen breng je wijzigingen aan in de sketch en laat je de Arduino werken zonder USB-verbinding.

### 2.5.1 Oefening – Knippersnelheid aanpassen

Door de sketch aan te passen en opnieuw te uploaden kun je de Arduino andere dingen laten doen. Zo kun je bijvoorbeeld de LED sneller laten knipperen.

- Verander de knippersnelheid in de sketch van 1000 ms in 200 ms.
- Verifieer de aangepaste sketch met de *Verifieer*-knop.
- Upload de aangepaste sketch naar de Arduino en bekijk het resultaat.

### 2.5.2 Oefening – Sketch opslaan en openen

Er zijn tal van redenen om een sketch op te slaan: back-up, verspreiden via het internet, kopiëren naar andere Arduino's enzovoort. In deze oefening laat je afwisselend de sketch **Blink** en **Blink2** werken:

- Bewaar de gewijzigde sketch onder de naam **Blink2** via *Bestand > Opslaan als*. De naamextensie van een sketch-bestand is *.ino*.
- Sluit de Arduino-software via *Bestand > Afsluiten*.
- Open de oorspronkelijke sketch **Blink** via *Bestand > Open Recent* en upload de sketch.
- Open de gewijzigde sketch **Blink2** via *Bestand > Open Recent* en upload de sketch.

### 2.5.3 Oefening – Zonder USB-kabel

Zodra een Arduino via *Upload* geprogrammeerd is en de sketch in zijn flash-geheugen staat, is de computerverbinding in principe niet meer nodig. Met een externe voeding kun je de Arduino zelfstandig verder laten werken, zonder hulp van de computer.

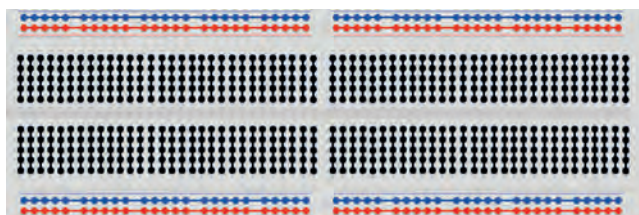
- Trek de USB-stekker uit de Arduino. Alle LEDs op het board doven nu, want de voedingsspanning is weggevalen.

- Sluit een 9 volt blokbatterij aan op de poort 'Externe voeding' (zie afbeelding 1.3).
- De sketch wordt opnieuw opgestart en de LED knippert weer.

## 3 Werken met componenten

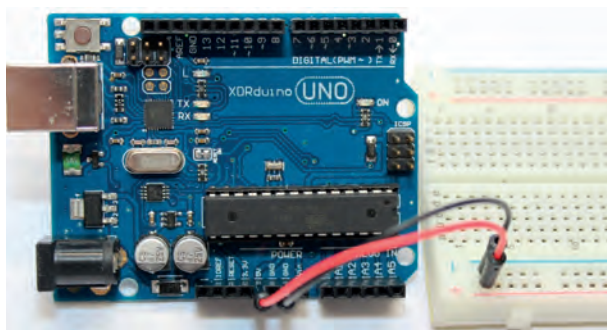
Dit hoofdstuk besteedt kort aandacht aan twee componenten (of onderdelen) uit de elektronica: de LED en de weerstand. Deze twee pas je veelvuldig toe in Arduino-projecten, samen met andere componenten. Om die componenten gemakkelijk en snel te verbinden gebruik je een breadboard.

### 3.1 Breadboard



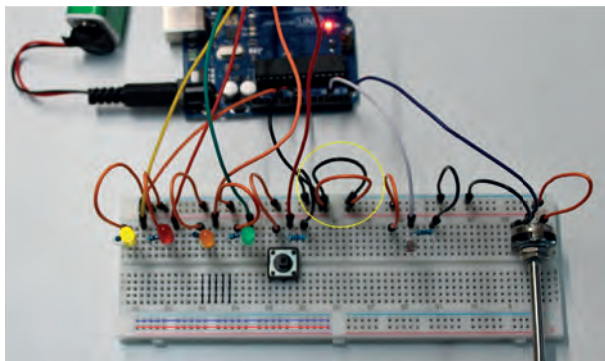
Afbeelding 3.1 – Breadboard: de zwart gemarkeerde gaatjes zijn in groepen van vijf doorverbonden, de rood (5 volt) en blauw (GND) gemarkeerde gaatjes zijn voor de voeding.

Een breadboard is een kunststof bordje met gaatjes, dat gebruikt wordt om snel elektronische proefschakelingen op te bouwen, bijvoorbeeld voor testdoeleinden. Het grote voordeel is dat er niet gesoldeerd hoeft te worden, wat veel tijd scheelt. De componenten worden in de gaatjes geprikt en automatisch vastgeklemd door middel van verende metaalstripjes onder de gaatjes. Via deze metaalstripjes zijn de gaatjes onderling doorverbonden in groepjes van vijf. Aan de lange zijde zitten gaatjes voor de voeding (+) en GND (-). Rode en blauwe lijnen op het breadboard geven aan hoe deze gaatjes zijn doorverbonden. Op sommige breadboards zijn deze lijnen halverwege onderbroken; die moet je overbruggen (zie gele omcirkeling in afbeelding 3.3).



Afbeelding 3.2 – Breadboard betreft zijn voeding van de Arduino.

Zoals in afbeelding 3.2 is te zien betreft het breadboard zijn voeding van de Arduino. De POWER +5V aansluiting is via de rode draad verbonden met de + (PLUS) van het breadboard, de POWER GND aansluiting is via de zwarte draad verbonden met de – (MIN) van het breadboard. In de elektronica worden de kleuren rood en zwart (of blauw) vaak gebruikt voor plus respectievelijk min. Door deze kleuren consequent toe te passen houd je de schakeling overzichtelijk.



Afbeelding 3.3 – Voorbeeld van schakeling opgebouwd op het breadboard.

Een van de minpunten van een breadboard is dat de klemverbindingen niet altijd even betrouwbaar zijn waardoor een breadboard ongeschikt is voor een permanente opstelling. Voor een permanente opstelling moet je de componenten solderen of shields gebruiken (zie later).

### 3.2 LED

LEDs zijn onmisbaar als indicator. Op vrijwel elk apparaat zitten wel een of meer LEDs om aan te geven welke functie(s) ingeschakeld zijn, zoals de LED op uitgang D13 van de Arduino. Een voordeel van LEDs is de lange levensduur, het lage stroomverbruik en de diversiteit. Ze zijn er in allerlei vormen, maten en kleuren. Veelgebruikt zijn de kleuren rood, geel, groen, oranje, wit, infrarood en blauw. De LED op D13 is een zogenaamde 'SMD'-LED. Om ook de status van andere uitgangen inzichtelijk te maken gebruik je externe LEDs. SMD-LEDs zijn daarvoor te klein, aanzienlijk hanteerbaarder zijn LEDs in de grotere 3-mm- of 5-mm-behuizing (zie afbeelding 3.4).

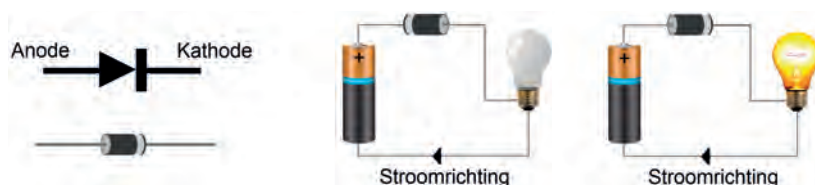


Afbeelding 3.4 – LEDs in een behuizing van 5 mm diameter.



### 3.3 Halfgeleider

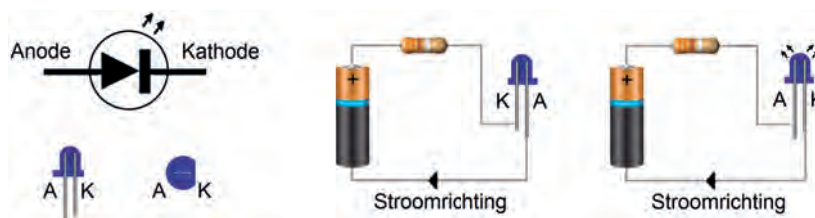
De letters LED staan voor Light Emitting Diode. Letterlijk vertaald: een diode die licht uitstraalt. In de basis is de LED een diode. Een diode is een component met twee aansluitingen: hij laat elektrische stroom maar in één richting door, in de richting van de pijl in het symbool (zie afbeelding 3.5). Een diode kun je vergelijken met een fietsventiel: de lucht gaat wel van buiten naar binnen, maar niet van binnen naar buiten. Evenzo laat een diode (of LED) de stroom wel door van plus naar min, maar ‘spert’ hij de stroom van min naar plus. Het bijzondere van een LED is bovendien dat hij – als hij elektrische stroom doorlaat – ook nog licht uitstraalt. LED en diode zijn halfgeleiders. Het tegenovergestelde van een halfgeleider is een geleider. Een stuk koperdraad bijvoorbeeld is een geleider: die laat elektrische stroom altijd volledig en in beide richtingen door.



Afbeelding 3.5 – Diode: linksboven het symbool.

In het geval van diodes en LEDs spreken we niet van plus en min, maar van ‘anode’ en ‘kathode’. Bij de diode is de kathode gemerkt met een streepje, bij een LED is het lange draadje de anode, het korte draadje de kathode. Als iemand de draadjes op gelijke lengte heeft afgeknipt, dan kun je de kathode nog op een andere manier herkennen: de ronde behuizing is aan de kant van de kathode iets afgevlakt (zie afbeelding 3.6).

Elektrische stroom gaat altijd van plus naar min. Als je dus de PLUS van een batterij aansluit op de anode, en de MIN op de kathode, dan zal de LED stroom doorlaten. Echter, die stroom is dermate groot, dat de LED meteen sneuvelt. Daarom moet een LED altijd worden gecombineerd met een stroombegrenzer. In de praktijk is dat bijna altijd een weerstand. Deze zogenaamde ‘voorschakelweerstand’ begrenst de stroom.



Afbeelding 3.6 – Linksboven: LED-symbool. Linksonder: vooraanzicht en onderaanzicht van een LED.

3.4 Weerstand



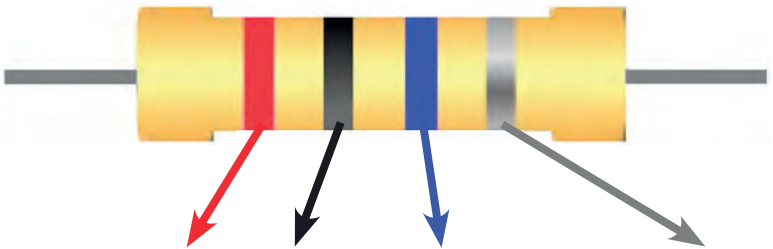
Afbeelding 3.7 – Links: symbool van een weerstand. Kleurringen geven de weerstandswaarde weer.

Een weerstand is een elektrische component die de doorgang van elektrische stroom bemoeilijkt. Vergelijk dit met een prop in de stofzuigerslang: door de prop ondervindt de luchtstroom weerstand en stroomt er minder lucht door de slang. De eenheid van weerstand is ohm en wordt aangeduid met de Griekse letter omega  $\Omega$ . Een kleine weerstand is bijvoorbeeld  $47\ \Omega$ . Grotere weerstandswaarden zoals  $1.000\ \Omega$  worden geschreven met de k van kilo, dus  $1.000\ \Omega$  schrijf je als 1 k of  $1\text{ k}\Omega$ . In schema's en documentatie wordt de spatie meestal weggelaten, dus 1k of  $1\text{k}\Omega$ . Een weerstand van  $4.700\ \Omega$  schrijf je als 4k7. Een weerstand van  $1.000.000\ \Omega$  of 1 megaohm wordt geschreven als 1M en een van  $1.600.000\ \Omega$  als 1M6. Zie de bijbehorende vermenigvuldigingsfactoren in tabel 3.1.

| Vermenigvuldigingsfactoren |       |         |            |                     |
|----------------------------|-------|---------|------------|---------------------|
| Factor                     | Naam  | Symbool | Betekenis  | Decimaal equivalent |
| $10^{-12}$                 | pico  | p       | biljoenste | 0,000.000.000.001   |
| $10^{-9}$                  | nano  | n       | miljardste | 0,000.000.001       |
| $10^{-6}$                  | micro | $\mu$   | miljoenste | 0,000.001           |
| $10^{-3}$                  | milli | m       | duizendste | 0,001               |
| $10^{-2}$                  | centi | c       | honderdste | 0,01                |
| $10^{-1}$                  | deci  | d       | tiende     | 0,1                 |
| $10^1$                     | deca  | da      | tien       | 10                  |
| $10^2$                     | hecto | h       | honderd    | 100                 |
| $10^3$                     | kilo  | k       | duizend    | 1.000               |
| $10^6$                     | mega  | M       | miljoen    | 1.000.000           |
| $10^9$                     | giga  | G       | miljard    | 1.000.000.000       |

Tabel 3.1 – Vermenigvuldigingsfactoren.

Kleurringen op de weerstand geven de weerstandswaarde aan (zie tabel 3.2). Houd de zilveren of gouden ring rechts en lees de ringen van links naar rechts. De afgebeelde weerstand boven de tabel heeft de waarde  $20 \times 1000000 = 20$  megaohm. Heb je een weerstand met vijf ringen, raadpleeg dan de tabel in Appendix 2.1.



| Kleur        | 1e ring | 2e ring | 3e ring =<br>Vermenigvuldigingsfactor | 4e ring<br>Tolerantie |
|--------------|---------|---------|---------------------------------------|-----------------------|
| Zwart        | 0       | 0       | 1                                     | 1 Ω                   |
| Bruin        | 1       | 1       | 10                                    | 10 Ω                  |
| Rood         | 2       | 2       | 100                                   | 100 Ω                 |
| Oranje       | 3       | 3       | 1000                                  | 1 kΩ                  |
| Geel         | 4       | 4       | 10000                                 | 10 kΩ                 |
| Groen        | 5       | 5       | 100000                                | 100 kΩ                |
| Blauw        | 6       | 6       | 1000000                               | 1 MΩ                  |
| Paars/violet | 7       | 7       | 10000000                              | 10 MΩ                 |
| Grijs        | 8       | 8       | 100000000                             | 100 MΩ                |
| Wit          | 9       | 9       |                                       |                       |
| Goud         |         |         | 0,1                                   | Ω                     |
| Zilver       |         |         | 0,01                                  | Ω                     |

Tabel 3.2 – Kleurentabel voor weerstanden met vier kleurringen (zie ook Appendix 2.1).

| Kleurringen | 1e ring   | 2e ring    | 3e ring         | 4e ring<br>Tolerantie | Weerstands-<br>waarde | Korte<br>notatie |
|-------------|-----------|------------|-----------------|-----------------------|-----------------------|------------------|
|             | bruin = 1 | zwart = 0  | rood = *100     | zilver = 10%          | 1.000 Ω               | 1k               |
|             | geel= 4   | violet = 7 | rood = *100     | goud = 5%             | 4.700 Ω               | 4k7              |
|             | bruin = 1 | zwart = 0  | groen = *100000 | goud = 5%             | 1.000.000 Ω           | 1M               |
|             | bruin = 1 | grijs = 8  | groen = *100000 | goud = 5%             | 1.800.000 Ω           | 1M8              |

Tabel 3.3 – Voorbeelden van weerstanden en hun kleurringen.

### 3.5 LED en weerstand aansluiten

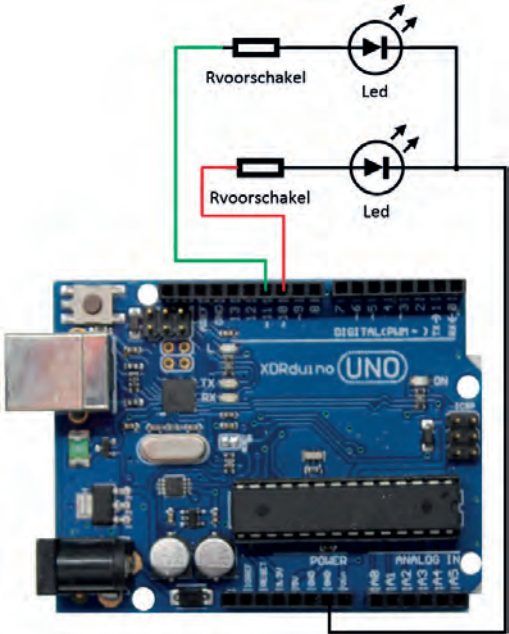
Je weet inmiddels dat je een voorschakelweerstand nodig hebt om een LED te laten oplichten. De waarde van deze weerstand wordt bepaald door twee factoren: de stroom door de LED (ampère) en de voedingsspanning (volt). Een rode LED geeft licht als er circa 20 mA (milliampère) ofwel 0,02 ampère doorheen stroomt. De

spanning op een uitgangspoort van de Arduino is maximaal 5 volt (HIGH). Volgens de Wet van Ohm (zie later in paragraaf 5.1) zou de benodigde weerstand gelijk zijn aan  $5/0,02 = 250\text{ ohm}$ . Door een spanningsval over de LED valt die echter 1,5 volt lager uit en daarmee ook de weerstand (zie tabel 3.4).

| Kleur LED  | Spanningsval | Waarde voorschakelweerstand |
|------------|--------------|-----------------------------|
| infrarood  | 1,1 volt     | ca. 195 ohm of 200 ohm      |
| rood       | 1,5 volt     | ca. 175 ohm of 180 ohm      |
| geel/groen | 2 volt       | 150 ohm                     |
| wit/blauw  | 3,5 volt     | 75 ohm                      |

Tabel 3.4 – Voorschakelweerstand voor LEDs bij een voedingsspanning van 5 volt.

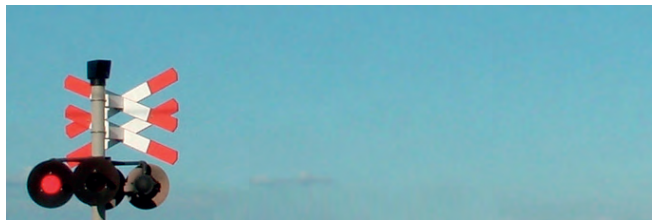
Gebruik voor gele en groene LEDs bij een voedingsspanning van 5 volt een voorschakelweerstand van 150 ohm (kleurringen Bruin – Groen – Bruin) en voor een rode LED een van 180 ohm (Bruin – Grijs – Bruin). Het maakt niet uit of je de voorschakelweerstand aan de anode-kant of kathode-kant van de LED plaatst. Gebruik voor elke LED een aparte voorschakelweerstand, zoals in afbeelding 3.8. Let wel, de LEDs in deze schakeling lichten pas op nadat je een bijpassende sketch hebt gemaakt en geüpload.



Afbeelding 3.8 – Schema van twee LEDs op D10 en D11.

## 3.6 Oefeningen

In de volgende oefeningen ga je aan de slag met externe LEDs. Je laat ze knipperen, heen en weer 'lopen', en gecodeerde berichten versturen.



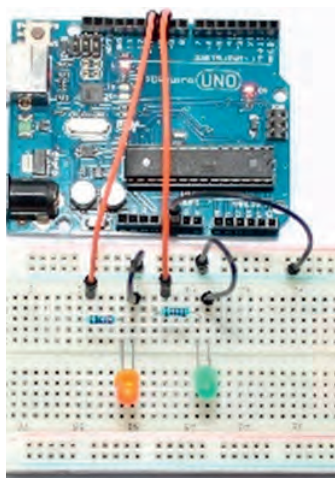
Afbeelding 3.9 - Praktijkvoorbeeld van om-en-om knipperen.

### 3.6.1 Oefening – Twee LEDs knipperen

In deze oefening breng je de schakeling van afbeelding 3.8 in praktijk en laat je de LEDs om-en-om knipperen.

#### ACTIE

- Sluit de Arduino weer aan op de USB-kabel van de computer.
- Sluit op pin D10 en pin D11 externe LEDs aan zoals in afbeeldingen 3.8 en 3.10.
- De LEDs zullen nog niet oplichten.
- Afbeelding 3.11 toont de benodigde sketch, voorafgegaan door een paar commentaarregels.
- Typ de sketch over en verifieer hem met de *Verifieer*-knop.
- Stuur de sketch naar de Arduino met de *Upload*-knop.



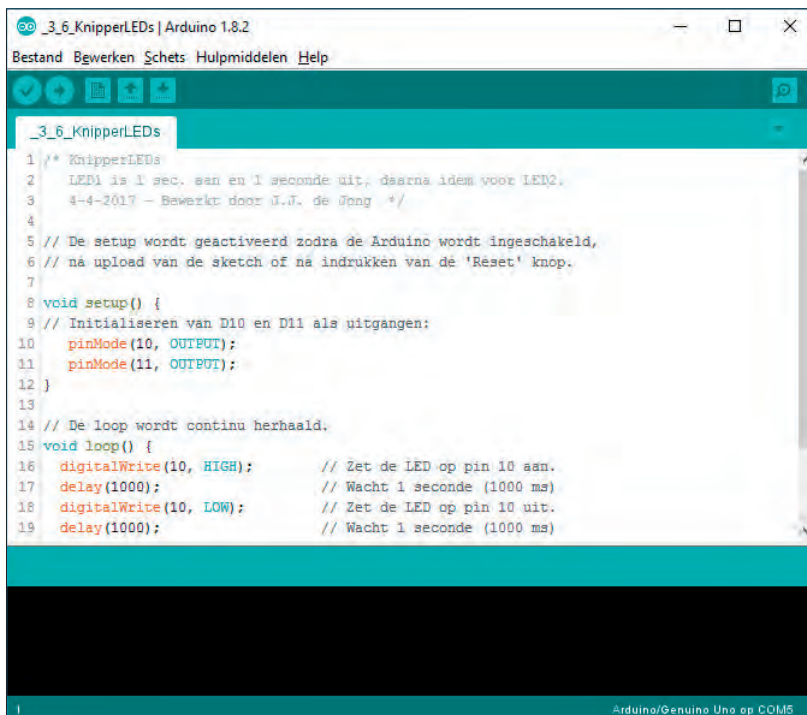
Afbeelding 3.10 – Twee externe LEDs met voorschakelweerstand aangesloten op pin D10 en D11.

## SKETCH

```

1  /* KnipperLEDs
2     LED1 is 1 sec. aan en 1 sec. uit, daarna idem voor LED2.
3  */
4
5  // De setup wordt geactiveerd zodra de Arduino wordt ingeschakeld,
6  // na upload van de sketch of na indrukken van de Reset-knop.
7
8  void setup() {
9    // Initialiseren van D10 en D11 als uitgangen:
10     pinMode(10, OUTPUT);
11     pinMode(11, OUTPUT);
12 }
13
14 // De loop wordt continu herhaald.
15 void loop() {
16     digitalWrite(10, HIGH);      // Zet de LED op pin 10 aan.
17     delay(1000);                 // Wacht 1 seconde (1000 ms).
18     digitalWrite(10, LOW);       // Zet de LED op pin 10 uit.
19     delay(1000);                 // Wacht 1 seconde (1000 ms).
20     digitalWrite(11, HIGH);      // Zet de LED op pin 11 aan.
21     delay(1000);                 // Wacht 1 seconde (1000 ms).
22     digitalWrite(11, LOW);       // Zet de LED op pin 11 uit.
23     delay(1000);                 // Wacht 1 seconde (1000 ms).
24 }

```



Afbeelding 3.11 – Screenshot van de sketch **KnipperLEDs**.

---

### Voorkom vaak gemaakte fouten

De regels die bij *void setup* en *void loop* horen moeten tussen accolades staan. Een statement eindigt altijd met een puntkomma. Dat wordt vaak vergeten. Of in plaats van een puntkomma wordt een dubbele punt getypt, zoals hieronder. Tijdens het verifiëren komen dit soort fouten aan het licht en krijg je een foutcode.

```
22 delay(1000):
```

---

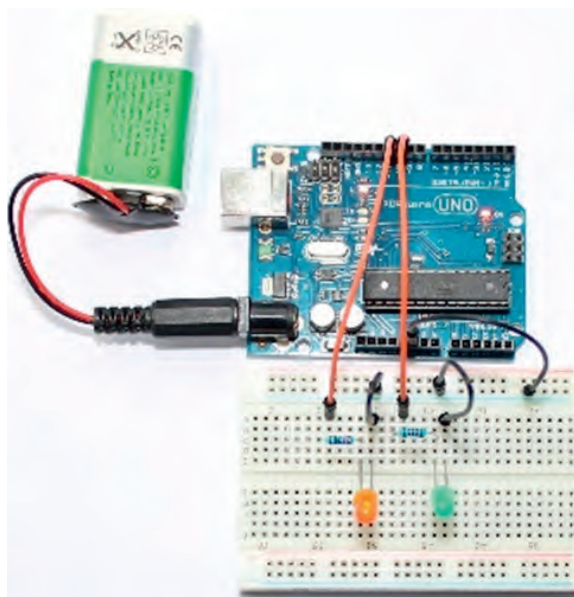
### 3.6.2 Oefening – Versnelde LED

Laat de beide LEDs vijf maal sneller knipperen. Verander daartoe beide delay-tijden van 1000 milliseconde in 200 milliseconde.

### 3.6.3 Oefening – Externe LEDs zelfstandig laten knipperen

Wijzig de sketch zo, dat de LED op D10 vijf keer kort knippert, waarna de LED op D11 ook vijf keer kort knippert, enzovoort. Ontkoppel de Arduino van de computer, zodanig dat het knipperen gewoon doorgaat. Daar is een externe batterijvoeding voor nodig. Doe dit als volgt:

- Zorg dat de gewijzigde sketch klopt en geüpload is.
- Check de werking van de knipperende LEDs.
- Sluit een batterij aan op de poort 'Externe voeding', zoals in afbeelding 3.12.
- Haal de USB-kabel uit de USB-poort.
- Check opnieuw de werking van de knipperende LEDs.



Afbeelding 3.12 – Arduino en LEDs losgekoppeld van de computer, schakeling werkt nu op batterijvoeding.

3.6.4 Oefening – Knight Rider

*Knight Rider* is een film en een televisieserie, waarin alles draait om KITT. In de oorspronkelijke serie uit de jaren 1980 en in de film is KITT een zwarte Pontiac. Op YouTube kun je hem zien:  
<https://www.youtube.com/watch?v=iQwlrEdka6Q>

Sluit acht externe LEDs aan op de digitale poorten D1 tot en met D9. Verbind de kathode van elke LED met een voorschakelweerstand. Laat de LEDs heen en weer lopen, net zoals op de auto in de video.



Afbeelding 3.13 – Schaalmodel van KITT.

3.6.5 Oefening\* – Morsecode

Morsecode bestaat uit korte en lange signalen met daartussen een korte pauze. Elke letter en elk cijfer heeft zijn eigen combinatie (zie tabel 3.5). De code werd uitgevonden door Samuel Morse (1835) en is een soort voorloper van de latere digitale communicatie met enen en nullen. Je kunt de signalen op allerlei manieren doorgeven, bijvoorbeeld door op tafel te kloppen, tegen een tralie of verwarmingsbuis te tikken (wat in gevangenissen veel werd gedaan), met lampen (scheepvaart), met telegrafiesignalen enzovoort. Een bekend morsecodevoorbeeld uit de scheepvaart en luchtvaart is SOS.

| Letter | Morse   | Letter | Morse   | Letter | Morse   | Letter | Morse   | Letter | Morse   | Cijfer | Morse     |
|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|-----------|
| A      | · —     | G      | — — ·   | M      | — —     | S      | · · ·   | Y      | — · — — | 4      | · · · —   |
| B      | — · · · | H      | · · · · | N      | — ·     | T      | —       | Z      | — — · · | 5      | · · · · · |
| C      | — · — · | I      | · ·     | O      | — — —   | U      | · · —   | 0      | — — —   | 6      | — · · · · |
| D      | — · ·   | J      | · — — — | P      | · — · · | V      | · · — — | 1      | · — — — | 7      | — — · · · |
| E      | ·       | K      | — · ·   | Q      | — — · · | W      | · — —   | 2      | · · — — | 8      | — — — ·   |
| F      | · · — · | L      | · — · · | R      | · — ·   | X      | — · · — | 3      | · · · — | 9      | — — — —   |

Tabel 3.5 – In morsecode heeft elke letter en cijfer zijn eigen code.

Ga aan de slag met de Arduino en zorg dat beide externe LEDs allebei het SOS-signaal laten zien. Een kort lichtsignaal duurt 100 ms, een lang lichtsignaal duurt 300 ms, de pauze na elke letter is 500 ms en de pauze na elk woord is 1000 ms.



# 4 Digitale input en output

Bij veruit de meeste toepassingen van de Arduino draait het om twee dingen: je stopt er iets in (input) en je verwacht dat er iets uit komt (output). Je drukt op een knop (input) en er gaat een lamp branden (output). Een temperatuur komt boven een bepaald niveau (input) en een actuator sluit de brandstofleiding (output). Een lichtsensor wordt onderbroken (input) en de draairichting van een motor keert om (output). Met sensoren op de ingangen en actuatoren op de uitgangen kun je een Arduino van alles laten doen.

De Arduino heeft veertien digitale poorten, die elk als ingang of als uitgang kunnen fungeren. Deze poorten werken met digitale signalen. Een digitaal signaal is 1 of 0. Je kunt ook zeggen ‘aan’ of ‘uit’. In termen van de Arduino heb je het over HIGH of LOW. Maar wat is HIGH en LOW precies voor een Arduino? En wat is 1 en 0? Met een van de simpelste sensoren wordt dat duidelijk: de drukknopschakelaar.

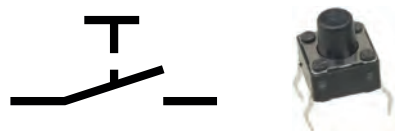
## 4.1 Drukknopschakelaar

Dagelijks druk je tientallen malen op een drukknopschakelaar of ‘pushbutton switch’. De knop van de deurbel, de zap-knop op de afstandsbediening van de tv, de lettertoetsen op je toetsenbord enzovoort. Een drukknopschakelaar heeft een terugverende knop en twee aansluitingen (zie de tabel). Een kenmerk van de drukknopschakelaar is dat hij maar twee toestanden kent, net als in de digitale techniek.

| Knop      | Stand  | Digitaal |
|-----------|--|----------|
| Indrukken | De drukknopschakelaar maakt contact, de schakelaar is ‘gesloten’ (aansluitingen zijn doorverbonden, er kan stroom lopen) | 1        |
| Loslaten  | Het contact is verbroken, de schakelaar staat ‘open’ (aansluitingen zijn niet verbonden, er kan geen stroom lopen)       | 0        |

Tabel 4.1 – Twee toestanden van een drukknopschakelaar.

Er zijn ook schakelaars die in een positie blijven staan: druk je eenmaal, dan maakt de schakelaar contact, ook als je de knop loslaat. Druk je nogmaals, dan wordt het contact verbroken, enzovoort. Ook al wordt deze schakelaar anders bediend, het resultaat is hetzelfde: 1 of 0.



Afbeelding 4.1 – Links het symbool van een terugverende drukknopschakelaar, rechts een afbeelding van een veelgebruikt type drukknopschakelaar voor montage op een print.

## 4.2 HIGH en LOW

Elke digitale uitgang van de Arduino is ofwel HIGH ofwel LOW. Eerder heb je al gelezen dat dit gelijk is aan 5 volt of 0 volt. Voor ingangen ligt het iets anders. De digitale ingang van de Arduino beschouwt elke spanning lager dan 2,4 volt als een digitale 0 (LOW) en elke spanning gelijk aan of hoger dan 2,4 volt als een digitale 1 (HIGH).

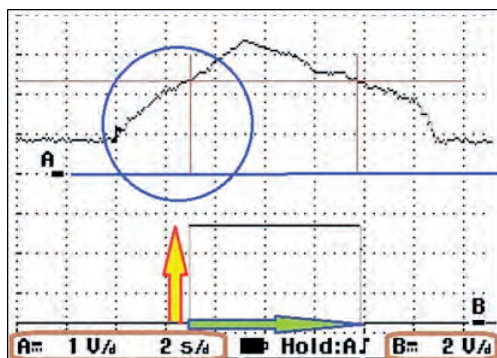
Het oscilloscoopbeeld van afbeelding 4.2 maakt dit duidelijk. Een oscilloscoop is een apparaat dat laat zien wat er in een bepaald tijdvak (horizontaal) met de spanning (verticaal) gebeurt. In dit boek maken we vaker gebruik van oscilloscoopbeelden, omdat je met dit instrument als het ware ‘in de elektronica’ kijkt.

Horizontaal zie je tien blokjes of ‘divisies’. Elk divisie staat voor twee seconden ( $2 \text{ S/d} = 2 \text{ seconden per divisie}$ ). De afbeelding laat dus zien wat er in  $10 \times 2 = 20$  seconden is gebeurd. De gebruikte oscilloscoop is een tweekanaals oscilloscoop, wat wil zeggen dat hij twee signalen tegelijk kan laten zien: A en B. Je moet het beeld daarom in tweeën delen:

- de bovenste vier rijen blokjes zijn voor signaal A; elk divisie staat hier voor 1 volt ( $1 \text{ V/d}$ )
- de onder vier rijen blokjes zijn voor signaal B; elk divisie staat hier voor 2 volt ( $2 \text{ V/d}$ )

De korte dikke streepjes bij de letters A en B geven voor elk signaal het ‘nul volt niveau’ aan. Signaal A is een spanning van 0,8 V die geleidelijk toeneemt tot 3,4 V en daarna weer afneemt tot 0,8 V. Signaal A wordt aangeboden aan digitale poort D2 van de Arduino; deze poort fungeert als ingang. Signaal B laat zien hoe de Arduino dit signaal heeft herkend door middel van een signaal op digitale poort D13; de poort fungeert als uitgang:

- De eerste 7 seconden gebeurt er niks op uitgang B, ook al neemt signaal A langzaam toe.
- Na 7 seconden bereikt signaal A de waarde 2,4 V (blauw omcirkeld). Prompt springt signaal B van LOW naar HIGH, zie de gele pijl. HIGH is gelijk aan:  $2,5 \text{ vakje} \times 2 \text{ volt} = 5 \text{ volt}$ .
- Van 7 tot 14 seconden (groene pijl) blijft signaal A boven 2,4V en blijft signaal B HIGH.
- Na 14 seconden zakt signaal A onder 2,4 V en zie je signaal B van HIGH naar LOW gaan.



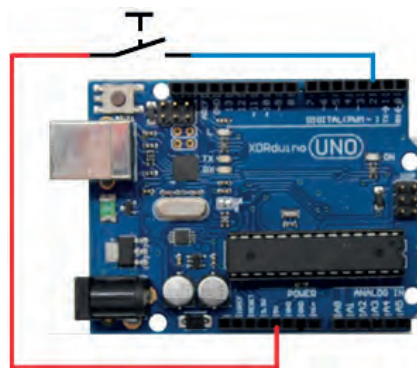
Afbeelding 4.2 – Oscilloscoopbeeld: signaal A is een spanning op poort D2 (ingang) en signaal B is de resulterende spanning op poort D13 (uitgang).

Conclusie: Als een digitale poort als ingang wordt gebruikt, dan wordt elke spanning groter dan of gelijk aan 2,4 volt gezien als HIGH. Elkeingangsspanning kleiner dan 2,4 volt wordt gezien als LOW.

### 4.3 In- en uitgangen (digitaal)

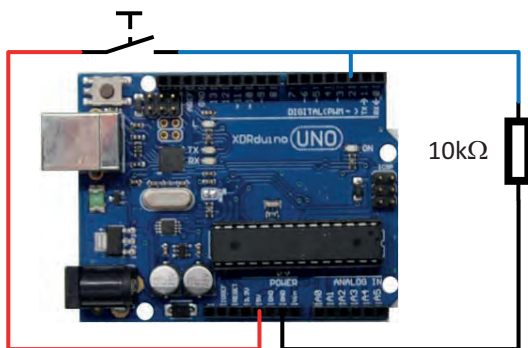
De digitale poorten van een Arduino zijn te gebruiken als ingangen en uitgangen voor verschillende functies. Gebruik je een digitale poort als uitgang, dan kun je er een LED op aansluiten, maar ook een motor of een andere actuator.

Gebruik je de digitale poort als ingang, dan kun je er een schakelaar of een digitale sensor op aansluiten. In de praktijk komt het vaak voor dat het signaal van zo'n sensor zeer kort is en dat er in één seconde duizenden van die korte signalen worden aangeboden. Om ook de geringste sensorsignalen te detecteren is de digitale poort zeer gevoelig. Aan die gevoeligheid kleef een nadeel. Stel dat je digitale poort D2 gebruikt als ingang door er een schakelaar op aan te sluiten. Zou de schakelaar open staan (dus geen contact maken), dan is D2 nergens mee verbonden. Hij 'zweeft' als het ware.



Afbeelding 4.3 – Als de drukknopschakelaar 'open' staat, is poort D2 nergens mee verbonden, hij 'zweeft'.

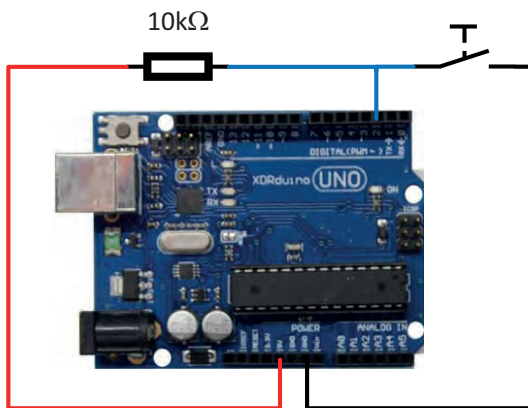
Als een ingang zweeft bestaat de mogelijkheid dat de ingang het signaal oppikt van een nabije mobiele telefoon, of dat hij reageert op eventuele statische elektriciteit van je vinger. Een zwevende ingang is instabiel, je weet nooit wanneer hij HIGH of LOW is. Om dit te voorkomen moet je een zogenaamde 'pull-up'- of 'pull-down'-weerstand toepassen.



Afbeelding 4.4 – De pull-down-weerstand zorgt dat ingang D2 met GND (o) verbonden is, als de schakelaar open staat.

#### 4.4 Pull-down-weerstand

In afbeelding 4.4 zie je dat digitale poort D2 via een weerstand van 10 kΩ is verbonden met GND. Als de drukknopschakelaar niet gesloten is, wordt hij via de pull-down-weerstand 'omlaag getrokken' naar de GND, vandaar ook de naam: pull-down. De spanning op de poort is nu 0 volt, dus LOW. Omdat voor het 'omlaag trekken' maar weinig stroom nodig is, mag de pull-down-weerstand een hoge waarde hebben. Een gebruikelijke waarde is 10 kΩ, maar in veel schakelingen kom je ook weerstanden van bijvoorbeeld 1 megaohm tegen. Als de drukknopschakelaar wordt gesloten heeft de weerstand geen invloed meer: een spanning van 5 volt zorgt voor een HIGH op de poort plus een verwaarloosbare stroom (0,05 milliam-père) door de weerstand.



Afbeelding 4.5 – De pull-up-weerstand zorgt dat ingang D2 met +5V (1) verbonden is, als de schakelaar open staat.

## 4.5 Pull-up-weerstand

Een pull-up-weerstand doet precies het tegenovergestelde van een pull-down-weerstand. Hij zorgt ervoor dat een ingang HIGH is als de erop aangesloten drukknopschakelaar open staat. In afbeelding 4.5 is een schakeling afgebeeld met een externe pull-up-weerstand. Als de drukknop niet is ingedrukt, is poort D2 via de weerstand van 10 k $\Omega$  verbonden met de +5 V pin. De ingang wordt als het ware ‘omhoog getrokken’ naar 5 volt, vandaar de naam: pull-up. Met de schakelaar in ruststand staat er dus 5 volt op de ingang en daardoor is de status HIGH. Wordt de knop ingedrukt, dan wordt de ingang verbonden met GND. Op dat moment staat er 0 volt op de ingang, en dat levert de status LOW. De externe weerstand speelt dan geen rol meer.

Een externe pull-up-weerstand is niet noodzakelijk. In elk van de digitale poorten van de Arduino, die als INPUT fungeren, is namelijk al een interne pull-up-weerstand aanwezig, maar deze is niet actief. Wil je de weerstand van een ingang activeren dan moet je dit in de functie *pinMode* aangeven met INPUT\_PULLUP. Voorbeeld:

```
pinMode(buttonPin, INPUT_PULLUP);
```

| Gewenste ingangstatus | Vereist   |
|-----------------------|---|
| LOW                   | externe pull-down-weerstand                     |
| HIGH                  | externe pull-up-weerstand<br>of<br>INPUT_PULLUP |

Tabel 4.2 – Keuze voor pull-up- of pull-down-weerstand.

## 4.6 LED in- en uitschakelen

In deze oefening bedien je met een drukknopschakelaar de interne LED van poort D13. De LED brandt zolang je de knop ingedrukt houdt.

### ACTIE

- Steek de drukknopschakelaar in het breadboard. Mochten de vier pinnen van de drukknopschakelaar een knikje vertonen, buig die dan eerst recht met een puntbektang, zodat de pinnen goed contact kunnen maken met de contacten in het breadboard.
- Verbind pin 1 of 2 van de drukknopschakelaar met +5V (zie datasheet in appendix 2.8).
- Verbind pin 3 of 4 van de drukknopschakelaar met poort D2. Maak gebruik van een breadboard.
- Sluit een pull-down-weerstand aan op poort D2 en GND.
- De schakeling ziet er nu uit zoals in afbeelding 4.4.

- Open de bijbehorende sketch in het Arduino-programma via *Bestand > Voorbeelden > Digital > Button*. Of open de Nederlandstalige bewerking **Sketch\_4\_6\_Button\_of\_drukknopschakelaar**.
- Upload de sketch naar de Arduino.
- Door op de knop te drukken laat je de interne LED op poort D13 branden.

## SKETCH

```

1  /* Button of drukknopschakelaar
2     LED op pin D13 gaat aan als op drukknopschakelaar wordt gedrukt
3  */
4
5  // Twee constanten vertegenwoordigen pinnen D2 en D13:
6  const int buttonPin = 2;    // Pinnummer drukknopschakelaar (D12)
7  const int LedPin = 13;     // Pinnummer LED (D13)
8
9  // Variabele vertegenwoordigt status van de drukknopschakelaar:
10 int buttonState = 0;
11
12 void setup() {
13 // Initialiseer digitale poort D13 als uitgang (OUTPUT):
14 pinMode(LedPin, OUTPUT);
15 // Initialiseer digitale poort D2 als ingang (INPUT):
16 pinMode(buttonPin, INPUT);
17 }
18
19 void loop() {
20 // Lees de status van de drukknopschakelaar:
21 buttonState = digitalRead(buttonPin);
22
23 // Controleer of de drukknopschakelaar wordt ingedrukt, dus
24 // check of de status van de drukknopschakelaar hoog (HIGH) is:
25 if (buttonState == HIGH)
26 {
27 // Zo ja, zet de LED aan:
28 digitalWrite(LedPin, HIGH);
29 } else {
30 // Zo nee (dus schakelaar is niet ingedrukt), zet LED uit:
31 digitalWrite(LedPin, LOW);
32 }
33 }

```

## FUNCTIES EN WAARDEN

In deze sketch zijn de volgende functies en waarden (constanten en variabelen) gedefinieerd en gebruikt:

### ► const

In regel 6 en 7 worden twee constanten gedefinieerd. De eigenschap van een constante is dat hij niet verandert gedurende de uitvoering van de sketch. Het is een vaste waarde die alleen gelezen kan worden. Zo heeft de constante 'buttonPin' de waarde 2, en de constante 'LedPin' de waarde 13.

```
const int buttonPin = 2;
const int LedPin = 13;
```

#### ► int

In regel 10 wordt een variabele gedefinieerd: 'buttonState' krijgt de waarde 0 toegewezen. Deze variabele is een 'int' of een zogenaamde integer. Dat wil zeggen dat zijn waarde altijd een geheel getal is, dus een getal zonder decimalen.

```
int buttonState = 0;
```

#### ► pinMode

In regel 14 wordt aangegeven dat LedPin een uitgang (OUTPUT) is. Omdat eerder aan LedPin de waarde 13 is toegekend, betekent dit dus dat digitale poort D13 als **UITGANG** fungeert. In regel 16 wordt constante buttonPin aangeduid als ingang (INPUT). Omdat eerder aan buttonPin de waarde 2 is toegekend, betekent dit dus dat digitale poort D2 als **INGANG** fungeert.

```
pinMode(LedPin, OUTPUT);
pinMode(buttonPin, INPUT);
```

#### ► void loop()

De lus is begonnen. Bij elke uitvoering van de lus wordt in regel 21 de huidige digitale waarde van 'buttonPin' gelezen. Deze waarde wordt als variabele 'buttonState' in het geheugen opgeslagen.

```
buttonState = digitalRead(buttonPin);
```

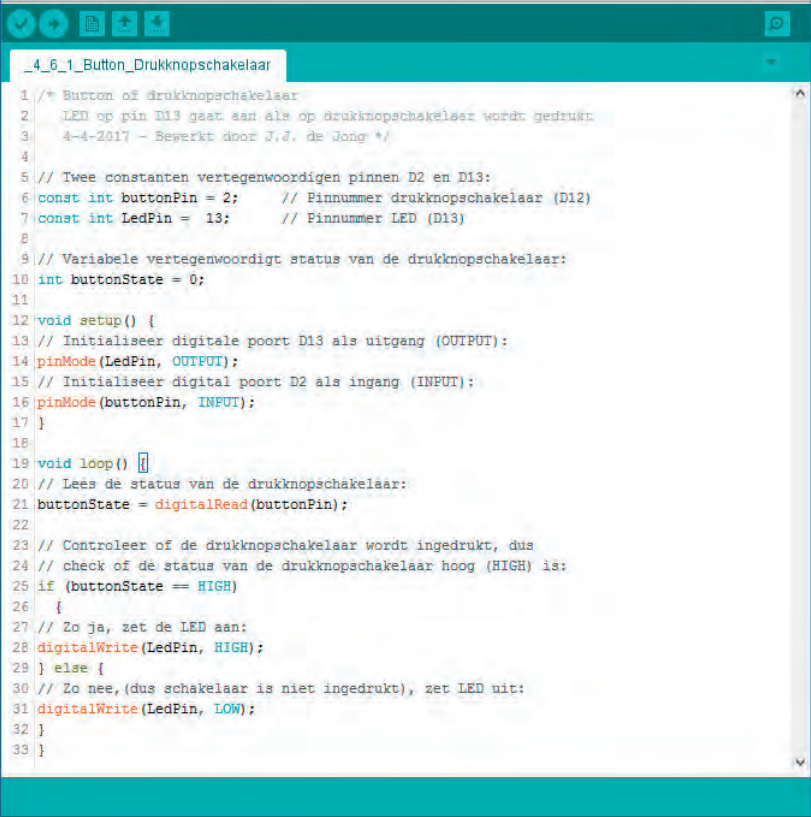
#### ► if ... else

In regel 25 t/m 32 van de lus wordt met een *if...else*-functie getest en besloten wat er moet gebeuren. De functie bestaat uit drie delen: vergelijking, {actie1}, {actie2}. Met het dubbele isgelijktteken (==) wordt een vergelijking gemaakt: in dit geval wordt de waarde van *buttonState* vergeleken met de waarde HIGH. Is de vergelijking waar, dan wordt actie1 uitgevoerd, dus alles tussen de twee accolades. Is de vergelijking niet waar, dan wordt actie2 uitgevoerd, dus alles tussen het tweede accoladepaar. In regel 23 t/m 29 staat dus: als (if) *buttonState* == HIGH, zet dan de LED aan, en anders (else) zet de LED uit. Behalve == zijn ook andere vergelijkingen mogelijk. In de tabel zie je hoe die genoteerd worden en wat hun betekenis is.

```
if (buttonState == HIGH) {
  digitalWrite(LedPin, HIGH);
} else {
  digitalWrite(LedPin, LOW);
}
```

| Vergelijking | Betekenis                        |
|--------------|----------------------------------|
| x == y       | x is gelijk aan y                |
| x != y       | x is niet gelijk aan y           |
| x < y        | x is kleiner dan y               |
| x > y        | x is groter dan y                |
| x <= y       | x is gelijk aan of kleiner dan y |
| x >= y       | x is gelijk aan of groter dan y  |

Tabel 4.3 – Met de if-functie kun je verschillende soorten vergelijkingen maken.



```
1 /* Button of drukknopschakelaar
2  LED op pin D13 gaat aan als op drukknopschakelaar wordt gedrukt.
3  4-4-2017 - Bewerkt door J.J. de Jong */
4
5 // Twee constanten vertegenwoordigen pinnen D2 en D13:
6 const int buttonPin = 2;    // Pinnummer drukknopschakelaar (D12)
7 const int LedPin = 13;      // Pinnummer LED (D13)
8
9 // Variabele vertegenwoordigt status van de drukknopschakelaar:
10 int buttonState = 0;
11
12 void setup() {
13 // Initialiseer digitale poort D13 als uitgang (OUTPUT):
14 pinMode(LedPin, OUTPUT);
15 // Initialiseer digital poort D2 als ingang (INPUT):
16 pinMode(buttonPin, INPUT);
17 }
18
19 void loop() {
20 // Lees de status van de drukknopschakelaar:
21 buttonState = digitalRead(buttonPin);
22
23 // Controleer of de drukknopschakelaar wordt ingedrukt, dus
24 // check of de status van de drukknopschakelaar hoog (HIGH) is:
25 if (buttonState == HIGH)
26 {
27 // Zo ja, zet de LED aan:
28 digitalWrite(LedPin, HIGH);
29 } else {
30 // Zo nee, (dus schakelaar is niet ingedrukt), zet LED uit:
31 digitalWrite(LedPin, LOW);
32 }
33 }
```

Afbeelding 4.6 – Sketch\_4\_6\_1\_Button\_Druknopschakelaar.

4.7 Oefeningen

In de volgende oefeningen ga je aan de slag met pull-up- en pull-down-weerstanden.

4.7.1 Oefening – 30 seconden uitschakelvertraging

Pas de sketch van paragraaf 4.6 aan, zodanig dat de LED na het loslaten van de drukknopschakelaar nog even blijft branden. Zorg dat de LED pas uitgaat 30 se-



conden nadat de knop is losgelaten. De uitschakelvertraging wordt onder meer toegepast bij trappenhuisverlichting in etagewoningen. Tip: voeg ergens aan het eind een statement toe, dat je ook in de eerste oefening van dit boek hebt gebruikt.

#### 4.7.2 Oefening – Looplicht van acht LEDs op schakelaar

Plaats acht LEDs op de digitale uitgangen 1 t/m 8 en laat deze vijfmaal heen en weer oplichten zoals bij de Knight Rider, zodra je de drukknopschakelaar indrukt.

#### 4.7.3 Oefening\* – Veranderende functie van de schakelaar

Als oefening 4.7.2, maar wijzig de sketch zodanig, dat de LEDs gaan lopen als je de drukknopschakelaar LOS laat.

#### 4.7.4 Oefening\* – Toggle-schakelaar

Via een toggle-functie kun je met dezelfde schakelaar een actuator zowel in- als uitschakelen. Voorbeeld: knop kort indrukken, LED aan, knop kort indrukken, LED uit enzovoort. Ga uit van **Sketch\_4\_6\_1\_Button\_Drukknopschakelaar** en pas deze aan. Twee tips:

**Tip 1:** Gebruik een teller om het aantal knopindrukken te tellen. Laat de LED alleen branden als de teller even is, en doven als de teller oneven is. Even tellerstanden zijn te herkennen als 'teller gedeeld door 2' geen rest oplevert:

```
if (teller % 2 == 0) {led aan} { led uit}
```

**Tip 2:** Las na elke knopindruk een pauze in om te voorkomen dat mechanische trillingen ('bouncing') van de schakelaarcontacten als aparte knopindrukken worden herkend. Een geschikte pauze is *delay{50}*.



Uitschakelvertraging wordt toegepast bij trappenhuisverlichting in etagewoningen en flats.