Knipperen met twee LED's

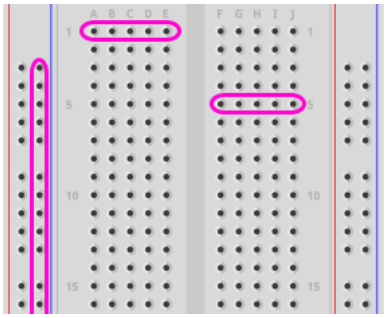
Arduino-experiment	130110-EN	*Trefwoorden*	Constanten, variabelen, expressions, for loop	
Versie	2018-06-26 / HS	*Niveau*	Basiscursus, module 2	*p.* 1/5

Wat je leert:

- Maak het programma duidelijker door gebruik te maken van constanten
- Sla een getal op in een variabele
- Herhaal een blok met instructies in een for loop

- Bouw een klein circuit op een breadboard
- Gebruik een LED met een serieweerstand

1 – Sluit een LED aan op een breadboard



Breadboard

Kleine schakelingen zijn gemakkelijk te bouwen zonder te solderen op een *breadboard*. De componenten worden simpelweg in de gaten gestoken die met elkaar verbonden zijn zoals op de tekening is aangegeven:

- In elk van de lange, buitenste rijen aan beide zijden zijn alle gaten met elkaar verbonden.
- De 5 gaten in elk van de genummerde rijen zijn met elkaar verbonden.
  Er is geen verbinding over het midden van de breadboard.

De lange rijen worden meestal gebruikt voor 5 V en 0 V. In deze module gebruik je een rij voor 0 V.

LED's en weerstanden

Bij gebruik van de ingebouwde LED op pin 13 is het waarschijnlijk niet opgevallen dat deze niet direct tussen pin 13 en 0 V is aangesloten – maar wel via een serieweerstand. Zo niet, dan zou de stroom zo groot zijn dat de LED of de Arduino beschadigd zou kunnen worden.

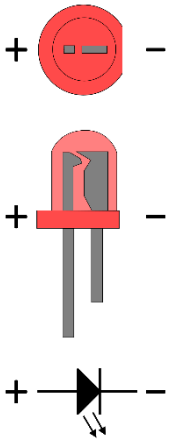
- Een LED moet altijd in serie worden geschakeld met een weerstand.
  Een waarde van 1 kΩ werkt meestal prima. Kleurcode: bruin, zwart, rood.

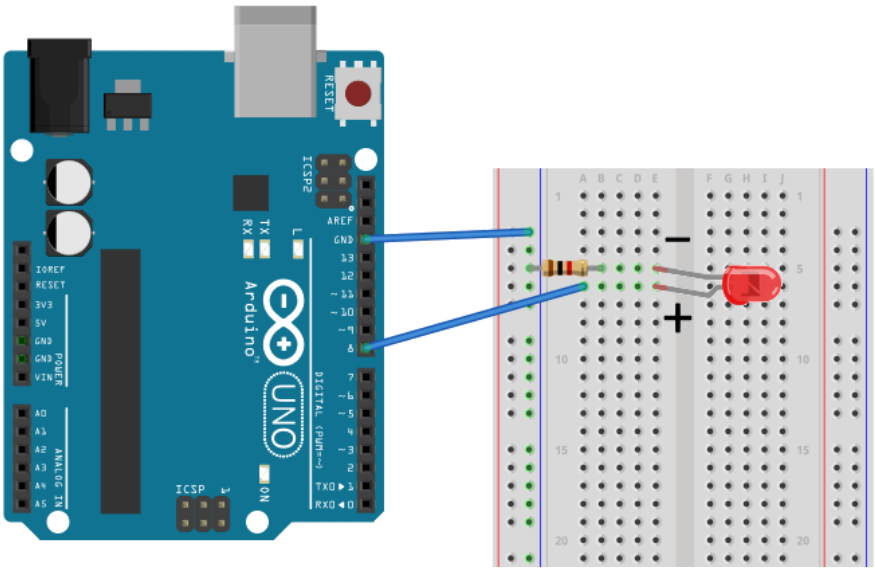
Hieronder is een opstelling te zien waarbij een LED in serie met een weerstand is aangesloten op pin 8 van de Arduino. Een van de lange rijen is verbonden met 0V: De GND-pin op de Arduino (afkorting van Ground).

De LED werkt alleen met de juiste polariteit: het langste loodje gaat naar “plus” (de Arduino-pin), het kortste naar “minus” (0 V). De minzijde wordt ook gekenmerkt door een afgeplatte rand.

- De LED moet de juiste polariteit krijgen om licht uit te stralen.

Het programma wordt uitgelegd op pagina 2.


+
−
+
−
+
−


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TXD▶1
RXD◀0
TX
RX
L
Arduino
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
DIGITAL (PWM~)
A B C D E
F G H I J
1
5
10
15
20
−
+

2 – Programmering met externe LED's

Je kunt er eventueel voor kiezen om het programma **blink-LED**, dat je in experiment 130110 hebt opgeslagen, te bewerken (als je dit programma niet hebt opgeslagen, voer dan een nieuw programma in zoals hieronder wordt weergegeven).

Open om het programma te vinden eerst de map met de naam blink-LED. Daar staat het Arduino bestand met dezelfde naam – open deze.

Telkens wanneer je op het punt staat een reeds getest programma te wijzigen, is het een goed idee om *het op te slaan onder een nieuwe naam*, zodat je kunt terugkeren naar de vorige versie: selecteer het menu-item *File / Save as*. Gebruik de naam **Blink-2**.

We gebruiken niet meer de ingebouwde LED, maar een externe, aangesloten op pin 8. Bewerk het programma om het er zo uit te laten zien:

```
const byte pinRed = 8;

void setup() {
  pinMode(pinRed,OUTPUT);
}

void loop() {
  digitalWrite(pinRed,HIGH);
  delay(200);
  digitalWrite(pinRed,LOW);
  delay(800);
}
```

In het eerste programma hebben we het pinnummer (13) direct in het programma ingevoerd.

Hier definiëren we in plaats daarvan een *constante* genaamd `pinRed` die vervolgens wordt gebruikt in de rest van het programma.

Toolbox: Constanten

Voorbeeld:

```
const byte pinRed = 8;
```

Elke keer dat je `pinRed` schrijft, wordt de waarde 8 ingevoegd.

Uitleg:

`const` ***A B*** `=` ***C***`;`

A Het **type** van de constante, hier gebruiken we het type `byte`. Een `byte` kan integers tussen 0 en 255 bevatten (incl.)

B De **naam** van de constante, hier `pinRed`

C De **naam** van de constante, hier 8

Naarmate projecten groter worden, kan het een hels karwei zijn om de numerieke waarden die hier en daar door het hele programma worden uitgestrooid te veranderen - het is veel veiliger om constanten te gebruiken die op een enkele plaats kunnen worden bewerkt.

De nummers worden in het Arduino geheugen bewaard op verschillende soorten locaties. Deze typen worden gebruikt in de definitie van constanten en variabelen (volgende paragraaf).

Het type `byte` neemt (- VERRASSING! -) 1 byte in beslag. Het biedt plaats aan integers van 0 tot max. 255.

Het type `int` neemt 2 bytes in beslag en kan integers tussen -32768 en 32767 bevatten.

Het type `word` neemt ook 2 bytes in beslag, maar het getalinterval gaat van 0 tot 65535.

Er zijn nog meer types dan alleen deze.

Uitdaging 1

Sluit een andere LED (bijv. een groene) aan via een serieweerstand. Gebruik pin 7 op de Arduino.

Voeg een andere constante toe aan het programma – bijvoorbeeld met de naam `pinGreen`.

Bewerk `setup()`, zodat pin 7 ook een output wordt – gebruik de constante.

Herschrijf `loop()` om de rode LED een kwart seconde te laten branden terwijl de groene LED uit is – vervolgens moet de groene LED een kwart seconde branden terwijl de rode LED uit is. Daarna moeten beide LED's een seconde uit zijn. (En dit moet oneindig worden herhaald.)

Er knipperen nogal wat LED's als mensen beginnen met het leren programmeren van een microcontroller. Dit is altijd bedoeld als een *voorbeeld* van het besturen van fysieke entiteiten met een softwareprogramma. Als je in plaats daarvan een schijnwerper van 1500 W wilt aanzetten is dat slechts een kwestie van wat extra hardware. (Maar dat zou niet bepaald lang kunnen duren als het doel is om snel te knipperen zoals hier ☺.)

3 – Herhalingen (loops)

Herhalende loops worden vaak gebruikt in programma's. De functie `loop()` zorgt voor de voortdurende herhaling van een opeenvolging van instructies. Maar hoe zit het met het herhalen van een kleinere reeks, bijvoorbeeld 10 keer, en dan doorgaan naar iets anders?

Je kunt de betreffende regels natuurlijk tien keer kopiëren! Maar dit kan snel rommelig worden – en als het nummer niet vaststaat, wordt het vrijwel onmogelijk.

Voordat we uitleggen hoe je loops kunt maken, introduceren we een zeer belangrijk concept: *een variabele*.

Variabelen

Zie de toolbox rechts.

Hier wordt een variabele genaamd `count` aangemaakt met een waarde van 10 – die precies lijkt op wat we een minuut geleden een constante noemden.

Het verschil is dat je de waarde van een variabele later in het programma kunt wijzigen:

```
count = 7;
```

Toolbox: Variabelen

Voorbeeld:

```
byte count = 10;
```

Reserveer hier ruimte voor in het geheugen en geef het een naam (en eventueel een beginwaarde).

Uitleg (in twee versies):

1: ***A B;***
2: ***A B = C;***

A Het **type** van de variabele, hier wordt `byte` gebruikt.

B De **naam** van de variabele, hier `count` genoemd.

C De **beginwaarde** (hier 10).
Versie 1 hierboven gebruikt geen beginwaarde – dit betekent dat de inhoud van deze geheugenruimte *ongespecificeerd* is totdat de variabele later in het programma een waarde krijgt.

Vaak krijgt een variabele een waarde die wordt berekend op basis van andere variabelen. Andere keren wil je gewoon de waarde met 1 verhogen – wat zo vaak gebeurt dat er een speciale syntaxis voor deze taak is geïntroduceerd. De volgende twee regels zullen beide de waarde van `count` met 1 doen toenemen:

```
count = count + 1;
count++;
```

Het teken "=" moet niet worden gelezen als "staat gelijk aan" maar als "krijgt de waarde van"; eerst wordt de rechterkant berekend – vervolgens wordt het resultaat opgeslagen in de variabele.

Toepasbaarheid van de variabelen

Als je de definitie van een variabele bovenaan het programma plaatst, buiten alle functies, heb je een *globale* variabele aangemaakt. Deze is overal toegankelijk en heeft overal dezelfde waarde.

Als je de definitie van de variabel binnen accolades **{}** – b.v. binnen de `setup()` functie – schrijft, dan wordt de variabele "onzichtbaar" buiten dat programmablok. Dit wordt een *lokale* variabele genoemd. In grotere programma's is het gebruik van lokale variabelen een voordeel.

De `for` loop

Nu zijn we klaar om een `for` loop te maken – zie de toolbox rechts.

In het voorbeeld wordt een speciaal soort lokale variabele gebruikt. Deze wordt gedefinieerd door "`int i=0`" en is zichtbaar tot de laatste **}** van de loop.

Dit had gebruikt kunnen worden om de vertragingen te laten variëren. Als de laatste `delay(150)` wordt veranderd in

```
delay(150*i);
```

Toolbox: Loops met `for`

Voorbeeld:

```
for (int i=0; i<5; i++) {
  digitalWrite(pinRed,HIGH);
  delay(150);
  digitalWrite(pinRed,LOW);
  delay(150);
}
```

Herhaal 5 keer: Knipperen met de rode LED

Uitleg:

```
for (A; B; C) { D }
```

A Uitgevoerd **vóór** de loop. Hier krijgt een integer-variabele de naam `i` en beginwaarde 0.

B Voorwaarde voor uitvoeren van ***D***. Hier wordt gecontroleerd dat `i` is *kleiner is dan* 5. (Zo niet, dan gaat de uitvoering door na de `for` loop)

C Uitgevoerd **na** elke herhaling van ***D***. Hier wordt 1 toegevoegd aan `i`.

D Code te herhalen. In het voorbeeld knippert de LED één keer.

- dan is de eerste vertraging 0 ms, de volgende 150 ms, de derde 300 ms, enz.

(gaat verder…)

Uitdaging 2

Gebruik een `for` loop om de rode en de groene LED afwisselend 10 keer te laten knipperen (200 ms per flits).

Dit wordt gevolgd door een pauze van 1 seconde met beide LED's uit.

(En dit moet zich tot in het oneindige herhalen.)

Uitdaging 3

Herschrijf de `for` loop om de rode en groene LED's te laten knipperen met een variërende duur: in eerste instantie moet de rode LED kort knipperen en de groene LED lang. Geleidelijk aan moet de rode LED steeds langer gaan knipperen, terwijl de groene LED steeds korter gaat knipperen.

Dit wordt gevolgd door een pauze van 1 seconde met beide LED's uit.

(En dit moet zich tot in het oneindige herhalen.)

4 – Extra oefening

Tot nu toe hebben we constanten gebruikt om de Arduino pinnen aan te spreken (zoals `pinRed`, `pinGreen`). Dit kan net zo goed met variabelen. Het pinnummer is zelfs te vinden via een berekening:

```
byte pin = 2;

pinMode(pin,OUTPUT);
pinMode(pin+4,OUTPUT);
```

Het stukje code hierboven stelt pin 2 en pin 6 in als outputs.

Uitdaging 4 – "Night Rider" looplichten

Sluit 8 LED's via serieweerstanden aan op pin 2 tot en met 9 van de Arduino.

Gebruik een `for` loop in `setup()` om van al deze pinnen outputs te maken.

Gebruik een andere `for` loop om telkens één LED achter elkaar te laten knipperen. Op deze manier lijkt het alsof er een lichtje over de lijn van de LED's "loopt".

(En dit moet zich tot in het oneindige herhalen.)