

WEEKENDSCHOOL – PROGRAMMEREN –

LES 3 – PROGRAMMEREN VAN ARDUINO

STARTEN

GEEF JE NAAM EN EMAILADRES

Als je het programma toegestuurd wilt krijgen, vergeet dan niet om je naam en emailadres op te geven.

OPENEN VAN ARDUINO IDE

We beginnen met het openen van Les_3b op het bureaublad. Dubbelklik deze map. Klik dan met de rechter muisknop op het bestand Les_3b.ino en kies Arduino IDE.

Je ziet dat de Arduino IDE geopend wordt.

En er opent een blanco blad. Hier gaan we het programma schrijven.

OPENEN VAN DE BESCHRIJVING

Ga nu weer naar het bureaublad en dubbelklik de beschrijving zodat je straks kunt knippen en plakken.

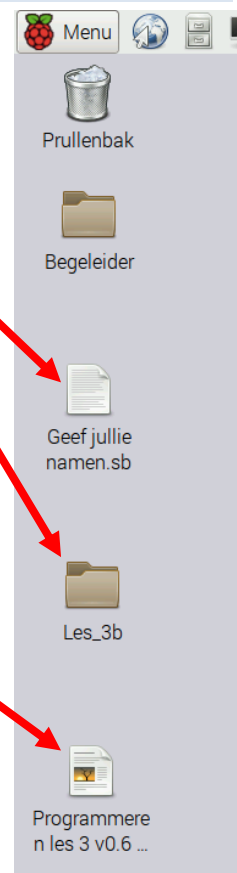
HEBBEN WE HAAST?

Werk zorgvuldig en werk gewoon goed door. Het is veel werk, maar als je goed doorwerkt moet je het klaar kunnen krijgen. Maar begrijpen wat je doet is belangrijker dan snelheid. En experimenteer niet te veel. Experimenteren mag je aan het eind als je klaar bent zoveel je wilt.

WAT JE MOET SNAPPEN

Er staat uitleg die je niet perse hoeft te weten, maar het geeft verdere uitleg.

Wat je wel echt moet weten staat op een groene achtergrond, net zoals deze regel.



BEGINNEN MET PROGRAMMEREN

Elke regel in het programma bevat een opdracht die wordt uitgevoerd. Er zijn 3 grote delen van het programma waar zulke regels in staan:

1. De globale declaraties. Hier staan definities die voor het hele programma gelden. Bijvoorbeeld op welke pennen de knoppen en LED's zijn aangesloten.
2. `setup()`: Het stukje programma dat eenmalig wordt uitgevoerd als de Arduino Nano start. Het stukje programma staat tussen accolades `{ }`.
3. `loop()`: Het stukje programma dat na `setup()` wordt uitgevoerd. Ook dit staat tussen accolades.

En daarna kunnen nog meer stukjes programma komen die we kunnen gebruiken in het hoofdprogramma. Maar dat gebruiken we vandaag niet.

GLOBALE DECLARATIES

Zie het gele blok hieronder.

Alles na `//` tot het einde van de regel is commentaar. Dat dient alleen om duidelijk te maken wat er gebeurt. Het is heel verstandig om als je aan het programmeren bent hier te noteren wat er wordt gedaan en waarom, want als je na een paar maanden naar het programma kijkt dan snap je er vaak niets meer van. En dat geldt ook voor iemand anders die er naar kijkt.

`#define` betekent dat overal waar het woord voorkomt dit wordt vervangen door wat er achter staat, tot het einde van de regel of het begin van het commentaar.

Dus, bijvoorbeeld overal waar in het programma staat `KnopLinks` zal dan 3 worden ingevuld.

Hieronder staat het eerste stel regels van het programma. Programmaregels staan in deze instructie altijd op een gele ondergrond. Je kunt dit blok tekst kopiëren en plakken. Dat gaat zo:

- Klik helemaal vooraan het gele blok.
- Houd de hoofdlettertoets (Shift) ingedrukt.
- Klik achter de laatste tekst in het gele blok.
- Kopieer dan met Ctrl-C.
- Ga dan naar het venster van de Arduino IDE. Klik het icoon bovenaan de balk of gebruik ALT-Tab
- Tik Ctrl-V.

Je kunt voor kopiëren en plakken ook de rechter knop van de muis gebruiken.

Wat je **NIET** kunt doen is een blok tekst selecteren en dan slepen en laten vallen. De tekst verdwijnt uit de handleiding maar komt niet in de Arduino IDE terecht (wat trouwens eigenlijk wel zou moeten).

Als je ooit tekst per ongeluk kwijt zou raken uit de handleiding, gebruik dan Ctrl-Z.

➔ Plak dit blok in het Arduino venster.

```
// ==== De globale declaraties (gelden voor het hele programma) =====

// Pennen van de Arduino. We geven ze een naam zodat we in het programma een naam kunnen
// gebruiken (dat is makkelijker) en omdat we dan, als we een wijziging moeten maken
// (bijvoorbeeld een knop op een andere pin aansluiten), slechts op 1 plek een wijziging
// hoeven te maken.

// Knoppen
#define KnopLinks 3           // De linker knop zit op pen 3
#define KnopRechts 12        // De rechter knop zit op pen 12
#define KnopKlein 4          // De kleine knop linksonder zit op pen 4

// LED's
#define LEDLinks 2           // De linker LED is aangesloten op pen 2
#define LEDRechts 11         // De rechter LED is aangesloten op pen 11

// In het huisje van de middelste LED zitten eigenlijk 3 LED's: een rode, een groene en
// een blauwe. Dus deze LED is met 3 pennen aangesloten.
#define LEDMiddenRood 5       // De rode LED in de middelste (grote) LED
// zit op pen 5
#define LEDMiddenGroen 6      // De groene LED in de middelste (grote) LED
// zit op pen 6
#define LEDMiddenBlauw 9      // De blauwe LED in de middelste (grote) LED
// zit op pen 9

#define LEDNano 13            // De kleine LED op de Arduino Nano zelf

// Analoge ingangen
#define PotMeter A0           // De potentiometer (potmeter) is een regelbare
// weerstand. Die is aangesloten op pen A0

// De 4 cijfers worden bestuurd door een chip TM1637 die aan de onderkant zit
// (je kunt het transparante plaatje omdraaien en dan zie je de chip met het nummer.
// Je ziet dan ook dat er 4 pennen zijn: GND, VCC, CLK en DIO.
// VCC en GND zijn de draden voor de voeding.
// Die gaan naar resp. +5 volt en GND (= 0 volt) op de Arduino zodat de chip en het
// blok cijfers spanning krijgen.
// DIO staat voor Digitale Input / Output. Over deze draad gaat de data die vertelt welke
// cijfers moeten oplichten.
// CLK staat voor klok (clock in het Engels). Die vertelt wanneer de TM1637 chip op de
// DIO draad de data moet bekijken.
#define CLK 7                 // Dit is de klok
#define DIO 8                 // Dit is de digitale input / output
```

SETUP()

Dit stuk van het programma wordt (eenmalig) uitgevoerd als de Arduino Nano wordt gestart.

Kopieer het hele blok en plak het in het venster van de Arduino IDE. Doe dat achter het blok met declaraties dat er al staat.

Dit stuk programma vertelt hoe de pennen moeten werken. Dus of het voor invoer of uitvoer is, en hoe precies.

```
// ==== setup =====
// Wat tussen de accolades achter setup staat wordt (eenmaal) uitgevoerd nadat de
// Arduino is gestart.

void setup(){
  pinMode(KnopLinks, INPUT_PULLUP); // Dit is openingsaccolade die bij setup hoort.
                                     // De pen waaraan de linker knop zit is voor invoer
                                     // (input).
                                     // En er moet een optrekweerstand (pullup) worden
                                     // geconfigureerd om te zorgen dat
                                     // er een 5 volt signaal (HIGH) op de pen staat als de
                                     // knop NIET wordt ingedrukt.
                                     // Als er op de knop wordt gedrukt komt er 0 volt (LOW)
                                     // op.
  pinMode(KnopRechts, INPUT_PULLUP); // Idem voor de rechter knop
  pinMode(KnopKlein, INPUT_PULLUP);  // Idem voor de kleine knop links onder

  pinMode(LEDLinks, OUTPUT);          // De pen waaraan de linker LED zit is een
                                     // uitvoer (output).
  pinMode(LEDRechts, OUTPUT);         // Idem voor de rechter LED.
  pinMode(LEDMiddenRood, OUTPUT);     // Idem voor het rode deel van de middelste LED.
  pinMode(LEDMiddenGroen, OUTPUT);    // Idem voor het groene deel van de middelste LED.
  pinMode(LEDMiddenBlauw, OUTPUT);    // Idem voor het blauwe deel van de middelste LED.

  Serial.begin(115200);               // Deze opdracht is nodig om straks te kunnen zien op
                                     // de Raspberry Pi wat de Arduino aan het doen is.
                                     // 115200 wil zeggen dat de communicatie zal gaan met
                                     // 115200 bit/seconde.
                                     // Hoe je op de Raspberry Pi kunt zien wat de Arduino
                                     // doet komt later.
}                                     // Dit is de sluitaccolade die bij setup() hoort.
                                     // Dit sluit de rij opdrachten af die bij
                                     // het starten van de Arduino worden uitgevoerd.
```

LOOP()

Tussen de accolades na loop() komt het programma dat wordt uitgevoerd na de setup().

```
// ==== loop =====
// Wat tussen de accolades staat wordt uitgevoerd na setup(). En dat wordt eindeloos herhaald.

void loop(){
  digitalWrite(LEDLinks, HIGH);      // Laat de linker LED branden
  delay(1000);                       // Wacht 1000 milliseconde (ms) = 1 seconde

  digitalWrite(LEDLinks, LOW);       // Doe de linker LED weer uit
  delay(1000);                       // Wacht 1000 milliseconde (ms) = 1 seconde
}
```

En als alle opdrachtregels zijn uitgevoerd, dan start het gewoon weer van voren af aan bij de eerste regel in loop().

➔ Plak nu dit stukje programma aan het eind, maar wel **vóór** de sluitaccolade.

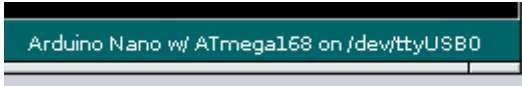
Wat je hier ziet zijn deze opdrachten:

<code>digitalWrite</code>	De eerste keer wordt de pen HIGH (hoog) gemaakt, dat is een “hoge” spanning (5 volt). Daardoor gaat de LED branden. De tweede keer maken we de pen LOW (laag), dus een lage spanning (0 volt). Dan gaat de LED weer uit.
<code>delay</code>	Delay betekent in het algemeen: <i>vertraag</i> . In dit geval betekent het <i>wacht</i> : wacht het aantal milliseconde dat tussen haakjes staat. Milli is 1/1000. Dus met <code>delay(1000)</code> wacht het programma 1 seconde tot de volgende opdrachtregel wordt uitgevoerd.

UITTESTEN

We gaan nu het programma laden in de Arduino. Maar eerst moet het programma worden vertaald in een code die de Arduino begrijpt. Dat heet **compileren**.

Controleer dat er rechtsonder op het scherm dit staat:

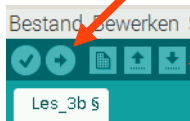


Arduino Nano w/ ATmega168 on /dev/ttyUSB0

Als er iets anders staat, vraag dan even je begeleider.

In Arduino heet een programma een *sketch*. Sketch is het Engelse woord voor *schets*.

➔ Druk nu hier om het programma te compileren en naar de Arduino Nano te *uploaden*.



OPSLAAN

Zorg dat je je werk vaak opslaat. Dat kan met deze knop.

WERKT HET?

Als het goed is gegaan, dan knippert de linker LED. Werkt het niet? Kijk dan even goed. Werkt het nog steeds niet? Vraag dan je begeleider.

SNELLER KNIPPEREN

Het LEDje is nu een seconde aan en dan een seconde uit.

- ➔ Verander dit zodat het LEDje 0,1 seconde aan is en dan 0,2 seconde uit.
- ➔ Natuurlijk moet je ook het commentaar aanpassen, anders snap je straks niet meer wat de bedoeling was.
- ➔ Upload het programma naar de Arduino.

OM NIET TE VERGETEN ...

Kijk naar het programma. Merk op dat alle opdrachten worden afgesloten met een puntkomma (;).

Daarna mag er commentaar staan. Commentaar begint met // en loopt door tot het eind van de regel.

Een puntkomma staat niet achter:

- Regels met #define
- Regels eindigend in een accolade { en }

NU BREIDEN WE UIT

Laat nu de middelste LED rood knipperen:

```
digitalWrite(LEDMiddleRood, HIGH); // Laat de middelste LED rood branden
delay(500)                          // Wacht 500 milliseconde (ms) = 0,5 seconde

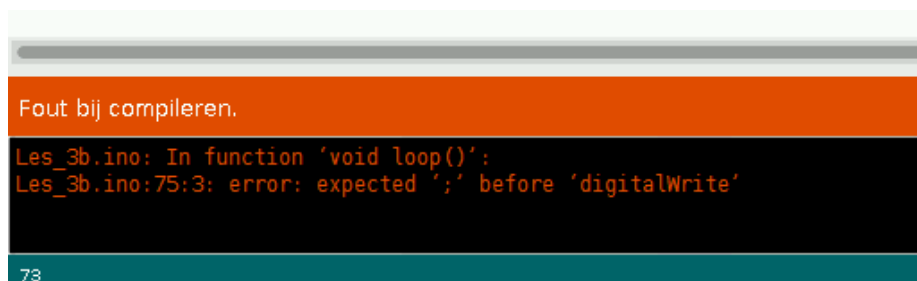
digitalWrite(LEDMiddleRood, LOW);   // Doe de LED weer uit
delay(100);                        // Wacht 100 milliseconde (ms) = 0,1 seconde
```

- ➔ Plak dit stukje programma onder wat we al hadden. Maar wel **voor** de laatste accolade }.
- ➔ Upload het programma naar de Arduino.

ER IS EEN FOUT ...

Oeps ... Nu krijgen we dit. Een foutmelding. Helaas is die in het Engels. De tweede regel zegt:

Fout: verwacht ; voor digitalWrite



Waar zit de fout? Er staat 75.3. Dat betekent dat de compiler, dat is het programma dat het compileren uitvoert, is gestruikeld op het derde teken op regel 75.

(Als je meer of minder blanco regels hebt toegevoegd kan het regelnummer bij jou anders zijn.)

Maar waar is regel 75? Links onderaan zie je het regelnummer waar de aanwijzer (de cursor) staat. Ga met de pijltjes toetsen of de muis (en dan ook klikken om de regel te selecteren) naar regel 75, of wat er ook staat als regelnummer in jouw programma. Het is even proberen, maar dan vind je de regel. En daar staat

```
digitalWrite(LEDMiddleRood, LOW);
```

De regel begint met twee spaties en het derde teken is de *d* van *digital*. Maar wat kan daar nu fout aan zijn?

Hint: vaak zit de fout in de vorige (niet blanco) regel.

- ➔ Zoek nu de fout en herstel hem.
- ➔ Upload het programma naar de Arduino. En kijk of de linker en nu ook de middelste LED rood knipperen.

JULLIE EERSTE EIGEN PROGRAMMAREGELS

- ➔ Nadat de middelste LED rood geworden is, doe hem dan 0,1 seconde uit (dat staat er al) en laat hem dan blauw branden. 0,5 seconde aan en 0,1 seconde uit.

Hint: Bedenk dat je, net zoals bij Scratch, een aantal regels kunt kopiëren en dan een beetje aanpassen. Dat scheelt tikwerk.

- ➔ Upload het naar Arduino en laat het resultaat zien aan een begeleider.

DAN GAAN WE VERDER MET ANDERE KLEUREN ...

- ➔ Als de middelste LED 0,1 seconde uit is, en laat hem dan 0,5 seconde groen branden.
- ➔ Wacht dan 0,1 seconde en laat dan de middelste LED 0,5 seconde wit branden. Weet je nog uit de les Natuurkunde hoe je wit licht maakt?
- ➔ Nadat de middelste LED is uitgedaan, wacht 0,1 seconde
- ➔ Upload naar Arduino en laat het resultaat zien aan een begeleider.

DAN NOG DE RECHTER LED

- ➔ Vul het programma nu zo aan dat nadat de middelste LED is uitgegaan, de rechter LED aan gaat gedurende 0,1 seconde en dan uit. Wacht daarna 0,2 seconde.

Hint: Als je de regels van LEDLinks kopieert dan hoef je het minste aan te passen.

- ➔ Upload het naar de Arduino en laat het resultaat zien aan een begeleider.

HOE WORDT DE PUNTENTELLING VERTOOND?

Op het plankje zit een blok van 4 cijfers. Net zoals bijvoorbeeld op een digitale wekker.

Om daar de goede cijfers op te krijgen is heel wat werk, want er moeten heel wat signalen gestuurd worden naar de chip die aan de achterkant van het plaatje met het blok cijfers zit.

Maar gelukkig. Iemand heeft dat werk al voor ons gedaan. Op internet konden we gewoon een *bibliotheek* downloaden met daarin het programma dat we nodig hebben. We hoeven ook niet te snappen hoe het werkt, we kunnen het gewoon gebruiken. Alleen opletten dat er geen Trojaanse paarden of andere valse software in zit, zoals je dat altijd moet controleren als je iets van internet gaat downloaden. Hier is dat al voor je gedaan.

Om het te kunnen gebruiken moeten we vertellen dat we de bibliotheek in het programma willen gebruiken. Dat doen we zo:

Zet helemaal bovenaan het programma de volgende regel. Het wordt dus de **allereerste regel** in het programma.

```
#include <TM1637Display.h>
```

En dan net **boven** het programmadeel met `setup()` zet je

```
TM1637Display VierCijfers(CLK, DIO);
```

Dit zorgt dat er een object gemaakt wordt van het type *TM1637Display* met de naam *VierCijfers* die gebruik maakt van de pennen met de namen *CLK* en *DIO*. *CLK* en *DIO* hebben we bovenaan al een waarde gegeven.

Nu we dit gedaan hebben, kunnen we cijfers laten zien met de opdracht

```
VierCijfers.showNumberDec (Getal, Voorloophnullen, AantalPlekken, PlekVanEersteCijfer)
```

- **Getal** Het getal dat je wilt laten zien
- **Voorloophnullen** Of voorloophnullen vertoond moeten worden. Voorloophnullen zijn nullen voor het getal. Als je het getal 1 wilt laten zien op 4 posities met voorloophnullen dan krijg je 0001.
false = laat nullen aan de voorkant weg.
true = laat nullen aan de voorkant wel zien.
- **AantalPlekken** Hoeveel plekken wil je voor het getal gebruiken
De cijfers op de andere plekken blijven onveranderd
- **PlekVanEersteCijfer** Op welke plek komt het linker cijfer
0 = meest linkse plek
3 = meest rechtse plek

Laten we maar gauw iets gaan proberen.

➔ Zet dit stukje aan het eind, voor de laatste accolade

```
VierCijfers.setBrightness(15); // Zet de helderheid op 15. 8 = zwak,
                                // 15 = fel
VierCijfers.showNumberDec (1234, false, 4, 0); // Laat het getal 1234 zien op 4 plekken
                                                // met het eerste cijfer
                                                // helemaal links (0)

delay(1000);
VierCijfers.showNumberDec (56, false, 2, 0); // Laat het getal 56 zien met 2 cijfers,
                                                // eerste cijfer helemaal links

delay(1000);
VierCijfers.showNumberDec (78, false, 2, 2); // Laat het getal 78 zien met 2 cijfers,
                                                // start vanaf het derde cijfer (plek 2,
                                                // want het eerste cijfer is plek 0)

delay(1000);
const uint8 t TweeCijfersLeeg[] = {0, 0}; // declareer een rij van 2 cijfers en
                                                // maak ze allemaal 0
VierCijfers.setSegments(TweeCijfersLeeg,2,0); // Wis de linker 2 cijfers
VierCijfers.setSegments(TweeCijfersLeeg,2,2); // Wis de rechter 2 cijfers
```

Hier gebeurt dit:

- Hier zetten we eerst de helderheid van de cijfers. Dat hoeft maar een keer te gebeuren en zou je dus ook in `setup()` kunnen zetten, maar zo gaat het ook.
 - Laat het getal 1234 zien met alle 4 cijfers en vanaf de meest linkse plek (plek 0).
 - Wacht een seconde en laat dan het getal 56 zien met 2 cijfers vanaf de meest linkse plek. De twee laatste cijfers blijven wat ze waren (34).
 - Wacht een seconde en laat dan het getal 78 zien met 2 cijfers vanaf plek 2, dat is dus het derde cijfer. De eerste twee cijfers blijven wat ze waren (56).
 - En dan wissen we de twee linker cijfers en daarna de twee rechter cijfers. Als je wilt weten hoe dat werkt dan moet je het even aan een begeleider vragen. Je hebt het niet nodig in deze les. De regel die begint met `const` moet eigenlijk bovenaan voor de `setup()` staan, maar dit is nu even handiger.
- ➔ Probeer het maar uit. Werkt het? Begrijp je wat er gebeurt? Laat het maar even aan een begeleider zien en leg het uit.

DE KNOPPEN EN VARIABLEN

We weten nu hoe de LED's en de cijfers worden bestuurd. Tijd om te kijken wat we met de knoppen doen. Maar eerst gaan we het over *variabelen* hebben.

In Scratch hebben we ook variabelen gebruikt. Die moesten we eerst aanmaken en dan konden we ze gebruiken. Dat moet ook in deze taal. Maar in Scratch kan een variabele een getal of tekst bevatten. Hier moeten we van tevoren weten wat we er mee gaan doen en ook hoe groot b.v. het getal is dat er in zal komen. We noemen de soort variabelen die er zijn *data typen*. Het meest gebruikte data type is *int*. Dat is een afkorting van het Engelse woord integer. Integer betekent: geheel getal. Er kan een getal in tussen tussen -32768 en 32767. En alleen maar een geheel getal. Dus geen cijfers achter de komma. 525 kan erin. Maar 3,2 kan niet. En 543210 kan ook niet.

We willen de toestand van het kleine knopje in een variabele `StatusKnopKleinNu` hebben.

→ Zet deze regel net onder de regel `void loop() {`

```
int StatusKnopKleinNu;
```

Daarmee is de variabele gedeclareerd. Dit noemen we een *lokale declaratie*.

We hebben eerder globale declaraties gezien die voor het hele programma gelden. Deze lokale declaratie betekent dat de variabele `StatusKnopKleinNu` alleen in `loop()` bekend is.

En zo kunnen we de status van het kleine knopje uitlezen.

→ Zet deze regels aan het eind van het programma, net voor de laatste accolade.

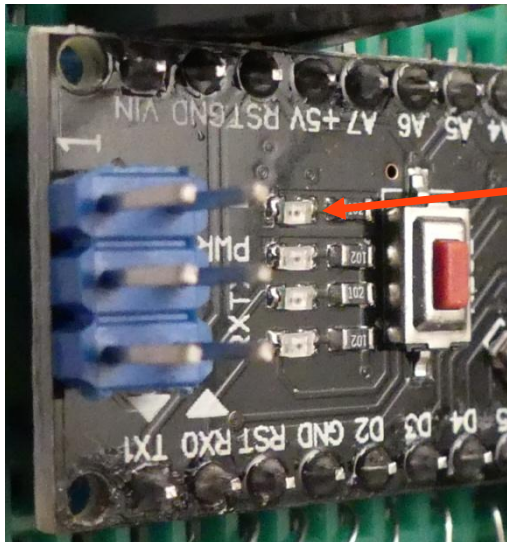
```
StatusKnopKleinNu = digitalRead(KnopKlein);

if (StatusKnopKleinNu == LOW) {
    digitalWrite(LEDNano, HIGH);
} else {
    digitalWrite(LEDNano, LOW);
}
```

Hier zie je dat `digitalRead` de manier is om de waarde van een digitale pin in te lezen. De waarde die terugkomt is `LOW` als de knop ingedrukt is en `HIGH` als hij losgelaten wordt.

Maar hoe kan dat nu? Er kon alleen maar een geheel getal in de variabele `StatusKnopKleinNu`? Dat komt omdat `LOW` hetzelfde is als 0. En `HIGH` is 1. Net zoals `false` 0 is en `true` 1. Die kunnen dus ook in deze variabele.

Als we nu de knop indrukken willen we dat het kleine LEDje op de Arduino aangaat. En weer uit als we hem loslaten. Dat deden we in Scratch met een als-dan-anders constructie. Die heet in deze taal *if-then-else*.



Dit is het LEDje op de Arduino Nano

Als je iets test, let dan op dat je == gebruikt en niet gewoon =, want dan zou je StatusKnopKleinNu niet testen of die LOW is, maar LOW *maken*.

Snap je wat er gebeurt?

- ➔ Leg het uit aan een begeleider.
- ➔ Upload het programma.

Werkt het? Hint: je moet de knop wel een tijdje vasthouden! Waarom is dat zo?

Trouwens, het had ook met een enkele opdrachtregel gekund:

```
digitalWrite(LEDNano,!digitalRead(KnopKlein));
```

Als je wilt weten hoe dat zit, vraag dan maar even aan een begeleider.

HET SPEL

EERST VOOR DE LINKER SPELER

We hebben nu alle onderdelen voor het spel.

Dit moeten we doen:

- Het spel start als je op het kleine knopje drukt.
- We beginnen om de cijfers op nul te zetten.
- Dan gaan we elke keer dat de linkerknop wordt ingedrukt een variabele ophogen en de waarde daarvan laten zien in de twee linker cijfers.
- Hetzelfde doen we voor de rechterknop. Maar dat komt pas als we de linker knop goed hebben.
- Als de tijd om is (bijvoorbeeld 15 seconde) of als een van de spelers 99 punten heeft dan kijken we niet meer naar de knoppen, maar kijken wie gewonnen heeft en dan laten we de cijfers van de winnaar knipperen.
- En als we klaar zijn dan wachten we weer totdat het kleine knopje wordt ingedrukt en dan kan de volgende ronde van start gaan.

We gaan eerst de linkerkant doen.

We hebben eerst een aantal variabelen nodig.

- ➔ Zet dit blok variabelen direct onder de plek waar we al eerder de declaratie van `StatusKnopKleinNu` hebben gedaan. Dat was de regel `int StatusKnopKleinNu;`

```
int StatusKnopLinksNu;
int StatusKnopLinksLaatst;
int PuntenLinks;
unsigned long StartTijd;           // De tijd (in milli seconde vanaf de
                                   // de start van de Arduino kan heel
                                   // groot worden. Dat past niet in een
                                   // int. Daar is de maximum waarde 32767.
                                   // Daarom nemen we een long.
                                   // En het getal zal nooit negatief worden,
                                   // daarom nemen we unsigned (zonder +/- teken,
                                   // dus alleen maar positieve getallen.
                                   // de speeltijd is 15 seconde
unsigned long SpeelTijd = 15;
```

➔ Zet het onderstaande blok in het programma. Het komt **direct voor** de laatste accolade in het programma.

```
// ==== Hier begint het spel =====
while (true) {                                // Herhaal dit eindeloos
    if (digitalRead(KnopKlein) == LOW) {        // Als het kleine knopje ingedrukt wordt
                                                // dan gaan we van start

        // ==== PLAATS 3 begin ==== Na deze regel komt straks het instellen van de speeltijd ==
        // ==== PLAATS 3 eind ===== Voor deze regel komt straks het instellen van de speeltijd =

        // ==== Voorbereidingen voor het spel =====

        StartTijd = millis();                  // Onthoud hoe laat het nu is
        PuntenLinks = 0;                      // Zet de punten van de linker speler op
                                                // nul
        VierCijfers.showNumberDec(PuntenLinks, false, 2, 0);
                                                // Vertoon het aantal punten van de linker
                                                // speler in de twee linker cijfers
        StatusKnopLinksNu = digitalRead(KnopLinks); // Lees de status van de linker knop
        StatusKnopLinksLaatst = StatusKnopLinksNu; // En dat is dan ook de beginwaarde

        while ((millis() < StartTijd + SpeelTijd * 1000) && (PuntenLinks < 99) ) {
                                                    // Als de huidige tijd (millis()) kleiner
                                                    // is dan de starttijd plus de speeltijd
                                                    // (die we met 1000 vermenigvuldigen om
                                                    // ook milliseconde te krijgen), dan
                                                    // kunnen we doorgaan: de speeltijd is
                                                    // nog niet om.
                                                    // Maar als de speler al 99 punten heeft
                                                    // dan moeten we toch stoppen, want
                                                    // meer dan 99 punten kunnen we niet
                                                    // laten zien.

            // ==== PLAATS 1 begin ==== Hieronder komen straks regels voor de knoppen =====
            // ==== PLAATS 1 eind ===== Voor deze regel komen de opdrachten voor de knoppen =====

        }

        // ==== PLAATS 2 begin ==== Hieronder komen straks de regels voor de winnaar =====
        // ==== PLAATS 2 eind ===== Hierboven komen straks de regels voor de winnaar =====

    }
}
```

Dit zijn de voorbereidingen, plus de programmalus waarin we straks de knoppen gaan bekijken. Die programmalus eindigt als de speeltijd over is of als er 99 punten zijn bereikt.

➔ Bestudeer het stukje programma goed en leg dan aan een begeleider uit wat er gebeurt.

Nu gaan we naar de linker knop kijken en geven de linker speler een punt erbij als de knop wordt ingedrukt.

```
StatusKnopLinksNu = digitalRead(KnopLinks);
if (StatusKnopLinksNu == LOW && StatusKnopLinksLaatst == HIGH) {
    // Als de knop nu LOW is en hij was HIGH
    // in de vorige keer dat we door de lus
    // kwamen, dan is de knop nu net ingedrukt
    PuntenLinks = PuntenLinks + 1; // De linker speler krijgt er een punt bij
    VierCijfers.showNumberDec(PuntenLinks, false, 2, 0);
    // Vertoon het aantal punten van de linker
    // speler in de twee linker cijfers
}
StatusKnopLinksLaatst = StatusKnopLinksNu; // Onthoud de waarde voor de volgende keer
// dat we door de lus komen
```

➔ Bestudeer het stukje programma goed en leg dan aan een begeleider uit wat er gebeurt.

➔ Zet dan het programma op de juiste plaats. Dat is plaats die is gemarkeerd met **PLAATS 1**.

➔ Test uit of het werkt.

SLA JE WERK OP

Als je dat nog niet gedaan hebt, sla je werk dan nu nogmaals op. Hierna gaan we iets doen wat wel eens fout zou kunnen gaan. Het kan ook verstandig zijn om het onder een andere naam op te slaan zodat als later iets fout gaat je in ieder geval deze versie nog hebt. Voeg een cijfer toe, bijvoorbeeld 02, toe aan het eind van de bestandsnaam.

NU VOOR DE RECHTER SPELER

Nu moeten we het ook voor de rechter speler in orde maken. Dat mag je nu zelf doen. Denk aan het volgende:

- Bedenk dat je stukken programma die je voor links hebt kunt kopiëren en plakken en dan Links door Rechts vervangen.
- Je moet nu ook variabelen maken voor rechts die je voor links had.
- Je moet de dingen die je als voorbereiding voor links deed (bijvoorbeeld punten op nul zetten) ook voor rechts doen.
- Je moet de dingen die je in de lus doet voor links nu ook voor rechts doen.
- Je moet ook stoppen als het aantal punten van de rechter speler op 99 staat.

➔ Pas het programma zo aan dat het ook werkt voor de rechter speler en test het uit.

WIE HEEFT ER GEWONNEN?

Tot slot willen we laten zien wie er gewonnen heeft. Dat doen we door de cijfers van de winnaar te laten knipperen. Wat als het gelijk spel is? Dan laten we ze gewoon beide knipperen!

Knipperen gaat door eerst de cijfers van de winnaar weg te halen (of van beiden bij gelijk spel), dan een pauze te houden en tot slot de cijfers weer te laten zien. Die geven ook opdracht om de cijfers van de verliezer te laten zien. Die stonden er al, maar dat maakt het programma simpeler.

Dit laten we doorgaan totdat we de volgende ronde doen, dus totdat op het kleine knopje wordt gedrukt.

- ➔ Bestudeer goed dit stukje programma, leg een begeleider uit wat er gebeurt en plak het dan op het juiste punt in het programma. Dat is de plaats die is gemarkeerd met **PLAATS 2**.

```
// Deze ronde is afgelopen. Laat nu de punten van de winnaar, of van beide bij gelijk
// spel knipperen. Dat doen we door eerst de cijfers van de winnaar (of beiden) weg te
// halen en dan beide weer te laten zien.
while (digitalRead(KnopKlein) == HIGH) {
    // Als de kleine knop wordt
    // ingedrukt dan stoppen we met
    // te laten zien wie heeft
    // gewonnen.

    if (PuntenLinks >= PuntenRechts) {
        // Als Links heeft gewonnen of als
        // het gelijk is ...
        VierCijfers.setSegments(TweeCijfersLeeg,2,0); // Wis de linker 2 cijfers
    }

    if (PuntenRechts >= PuntenLinks) {
        VierCijfers.setSegments(TweeCijfersLeeg,2,2); // Wis de rechter 2 cijfers
    }

    delay (100);
    VierCijfers.showNumberDec(PuntenLinks, false, 2, 0);
    VierCijfers.showNumberDec(PuntenRechts, false, 2, 2);
    delay (100);
}
```

- ➔ Upload het programma.
- ➔ Probeer het spel te spelen.

UITBREIDING: SPEELTIJD INSTELLEN

Nu is de speeltijd vast ingesteld op 15 seconde. Het zou leuk zijn als we die tijd zouden kunnen instellen. Op het plankje zit ook een regelbare weerstand. Die noemen we een potentiometer, kortweg potmeter.

Als je daaraan draait, dan verandert de spanning op pen A0. Als we die pen uitlezen zouden we de tijd kunnen regelen.

De waarden van de potmeter zijn 0 als de knop helemaal naar rechts is gedraaid en 1024 (ongeveer) als de knop helemaal naar links staat.

Het lezen van de stand van de potmeter doen we zo:

```
PotMeterStand = analogRead(PotMeter);
```

Maar we willen geen speeltijd hebben van 1024 seconde. Dat is veel te lang. En ook geen 0 seconde. Dat is te kort. Redelijk zou zijn als het tussen 5 en 60 seconde instelbaar is.

Maar hoe vertalen we een waarde tussen 0 en 1024 naar een waarde tussen 5 en 60?

Arduino heeft daar een hele mooie opdracht voor: *map*

```
SpeelTijd = map(PotMeterStand, 0, 1024, 5, 60);
```

Dit vertaalt de waarde van PotMeterStand die ligt tussen 0 en 1024 naar een waarde tussen 5 en 60.

➔ Zet dit stukje programma op de plaats die gemarkeerd is als **PLAATS 3**.

```
// ==== Stel nu eerst de speeltijd in
VierCijfers.setSegments(TweeCijfersLeeg, 2, 0); // Wis de linker 2 cijfers
VierCijfers.setSegments(TweeCijfersLeeg, 2, 2); // Wis de rechter 2 cijfers

int i;
int PotMeterStand;
for (i=1; i<1000; i++) { // doe het 1000 keer om de spelers
                        // een beetje tijd te geven
    PotMeterStand = analogRead(PotMeter); // lees de stand van de potmeter

    SpeelTijd = map(PotMeterStand, 0, 1024, 5, 60); // vertaal het naar een bruikbare
                                                    // waarde voor de speeltijd
    VierCijfers.showNumberDec(SpeelTijd, false, 2, 1); // laat de speeltijd zien

    // ==== PLAATS 4 begin ==== Hieronder komen straks de regels voor de toegift =====
    // ==== PLAATS 4 eind ===== Hierboven komen straks de regels voor de toegift =====
}
```

WERKT DE POTMETER LOGISCH?

Normaal ben je gewend als je aan een knop naar rechts draait dat de het meer wordt: bijvoorbeeld op een radio het geluid harder. Maar hier werkt het net andersom.

Echter, met een hele kleine aanpassing in deze regel kun je het wel laten werken zoals je gewend bent. Weet je hoe? Pas het maar aan. Hier moet je iets in aanpassen:

```
SpeelTijd = map(PotMeterStand, 0, 1024, 5, 60);
```

IS HET SPEL EERLIJK?

- ➔ Probeer eens langzaam te spelen. Kijk eens wat er gebeurt als je herhaaldelijk ferm op de linker knop drukt. Wel ferm op een knop drukken, maar niet te snel. Kijk goed wat er gebeurt. Krijg je elke keer dat je drukt een punt erbij of is het soms anders? En waarom is dat bij de rechterknop anders?

Je ziet hier het effect van *contactdender*. Als de schakelaar wordt ingedrukt komen in de schakelaar twee stukken metaal tegen elkaar. En het kan zijn dat het metaal terugveert en dan weer tegen het andere deel van het contact komt.

En dan kan het zijn dat de Arduino, die zo'n 15.000 keer per seconde naar de stand van de schakelaar kan kijken, ziet dat het contact een paar keer geopend en gesloten wordt. Terwijl jij maar één keer hebt gedrukt.

Dat effect kunnen we goed zien als we een paperclipschakelaar nemen. Twee rechtgebogen paperclips tikken we tegen elkaar aan en dit is de schakelaar.

- ➔ Sluit de paperclip schakelaars aan en kijk wat er gebeurt. Vraag maar aan een begeleider waar de paperclipschakelaar moet worden aangesloten.
- ➔ Waarom is dat bij de rechter knop anders?

Om het spel goed te maken moeten we daar iets aan doen.

Daar zijn twee mogelijkheden voor:

- Het programma aanpassen
- Elektronica toevoegen (op dit plankje is dat voor de rechter knop al gedaan)

Je gaat nu eerst zelf zo'n plankje maken. En daarbij voegen we wat elektronica toe om de contactdender te voorkomen. Want dat gaat nu het snelste.

ZELF EEN PLANKJE MAKEN – OF IETS ANDERS

Je mag nu zelf zo'n plankje gaan maken. Je krijgt daarvoor een aparte instructie. ***Doe dat alleen als je er ook iets mee wilt gaan doen***, bijvoorbeeld laten zien aan je familie, op school, het spel spelen met vriendjes en vriendinnetjes. Als je het later niet meer wilt hebben, breng het dan terug, want het is zonde als het bij het afval zou belanden.

Als je geen plankje wilt maken dan mag je doorgaan met de opdrachten hieronder of je mag zelf iets maken in Scratch.

Dus je moet nu kiezen:

1. Zelf een plankje maken (en verder programmeren als je klaar bent en er nog tijd is)
2. Verder programmeren met de Arduino
3. Zelf iets maken in Scratch

DE TOEGIFT: REGENBOOGKLEUREN

Weet je nog wat je bij de les Natuurkunde hebt geleerd? Hoe worden de kleuren licht gemaakt?

De matte LED kan blauw, rood en groen worden, maar ook alle combinaties. En alle combinaties van een beetje van een kleur met een hoeveelheid van een andere kleur.

We gaan nu toevoegen dat je met de potmeter de felheid van elke kleur kunt instellen. Welke kleur je instelt bedien je door op één van de drie knoppen te drukken.

- Rood stel je in door op het kleine knopje te drukken, ingedrukt te houden en aan de potmeter te draaien.
- Groen stel je in op dezelfde manier, maar dan met de linker knop.
- En blauw ook zo, maar dan met de rechter knop.

Met `digitalWrite` kun je de LED helemaal aan of helemaal uit doen. Met `analogWrite` kun je de LED een beetje aan doen. De waarden die je kunt geven is tussen 0 (helemaal uit) en 255 (helemaal aan).

Tijdens het instellen laten we op het display zien (met een getal tussen 0 en 255) hoe de LED is ingesteld.

➔ Zet dit stukje programma op **PLAATS 4**.

```
if (digitalRead(KnopKlein) == LOW) { // Als het kleine knopje ingedrukt is
// dan gaan we het rood van de LED
// veranderen
LEDWaardeBegin = LEDMiddenWaardeRood; // onthoud de beginwaarde van de LED
PotMeterStandBegin = analogRead(PotMeter); // lees de stand van de potmeter
while (digitalRead(KnopKlein) == LOW) { // Doe dit zolang de knop is ingedrukt
PotMeterStand = analogRead(PotMeter); // lees de nieuwe stand van de
// potmeter

LEDWaardeNieuw = LEDWaardeBegin
+ map(PotMeterStand-PotMeterStandBegin,-1024, 1024, 260, -260);
// verander de waarde van de LED.
// Zie de Bijlage C in de handleiding
// voor de uitleg

LEDWaardeNieuw = constrain(LEDWaardeNieuw, 0, 255);
// Begrens de waarde tot een getal
// tussen 0 en 255

analogWrite(LEDMiddenRood, LEDWaardeNieuw); // Zet de nieuwe waarde voor de LED

VierCijfers.showNumberDec(LEDWaardeNieuw, false,4, 0);
// Laat de waarde van de LED zien
}

// We hebben het display gebruikt om de waarde van de LED aan te geven
// Dat moeten we nu weer uitpoetsen, want na het loslaten van de knop gaan we
// weer verder met het instellen van de speeltijd.
VierCijfers.setSegments(TweeCijfersLeeg,2,0); // Wis de linker 2 cijfers
VierCijfers.setSegments(TweeCijfersLeeg,2,2); // Wis de rechter 2 cijfers

LEDMiddenWaardeRood = LEDWaardeNieuw; // Onthoud de nieuwe waarde van de LED
// voor een volgende ronde

i = i - 200; // Geef een beetje extra tijd
// door de teller terug te zetten
}
```

- ➔ Upload het programma naar de Arduino.
- ➔ Oops, er zijn nog een paar variabelen die moeten worden gedeclareerd met `int`. Doe dat en probeer opnieuw.
- ➔ Kijk of je nu de helderheid van de middelste LED kunt instellen en laat het aan een begeleider zien.

DE POTMETER KAN NIET VERDER ...

Je zult zien dat soms de potmeter niet verder kan terwijl je toch niet op 0 of 255 bent aangekomen. Hoe kan dat nu?

Als je erover nadenkt is het niet zo vreemd. Want als de potmeter bijvoorbeeld al aan het uiteinde staat en je drukt dan de knop in en de waarde staat op bijvoorbeeld 100, dan kun je maar een richting op veranderen, want de potmeter staat al aan het einde.

Wat is de oplossing?

- Laat de knop los.
- Draai de potmeter weg van het uiteinde waar die nu staat.
- Druk weer op de knop.
- Draai aan de potmeter. Je zult nu zien dat je verder kunt.

Vergelijk het met je muis. Als de muis aan de tafelrand is en je wilt verder, dan pak je de muis op en verplaatst hem. Dan kun je weer verder.

NU OOK VOOR GROEN EN BLAUW

- ➔ Als dit werkt, doe het dan ook voor de helderheid van groen en blauw.
- ➔ Sla het programma op, want je bent klaar met de opdrachten die we hadden bedacht.

WAT NU?

Als je alles gedaan hebt en nog tijd over hebt dan ben je heel erg snel. Je mag dan zelf verzinnen wat je verder gaat doen. Overleg met je begeleider. En sla je programma op onder een andere naam, zodat je dit wat je nu hebt veilig stelt.

BIJLAGE A: INTRODUCTIE

Dit is de introductie van de les. Dit staat in de bijlage, want we hebben dit mondeling besproken.

WAT GAAN WE DOEN?

In de vorige les heb je gezien dat je met programmeren ook kunt werken met knopjes en LEDjes. En je hebt gezien dat op het plankje met de eend ook een heel klein computertje zat: de Arduino Nano.

In deze les gaan we zo'n Arduino Nano programmeren. En wel zo, dat het plankje kan worden afgekoppeld en zonder de Raspberry Pi computer kan werken.

WAAROM GEEN SCRATCH?

In de vorige lessen hebben we met Scratch gewerkt als programmeertaal. Dat is een taal voor het onderwijs. Als je grote programma's wilt maken dan is het niet zo handig. Echte programmeurs gebruiken daarom meestal andere programmeertalen. Heel populair is de taal C en C++. We gebruiken vandaag die talen, echter met wat beperkingen.

MOETEN WE VEEL INTIKKEN?

In de vorige lessen hebben we het meeste met de muis gedaan. Het blijkt dat het voor programmeren toch meestal het snelste is om te tikken op het toetsenbord. Daarom gebruiken veel programmeurs programmeertalen waarbij het programma wordt ingetikt.

Echter, vandaag gebruik je veel voorbeelden en die kun je uit deze instructie met kopiëren en plakken gewoon overzetten. Soms moet je – net als een echte programmeur – wel dingen intikken.

WAAROM ARDUINO NANO?

Wat is het leuke aan de Arduino Nano? Hij is zo goedkoop dat jullie allemaal in het laatste deel van de les zelf zo'n plankje in elkaar mogen knutselen om mee naar huis te nemen en dan het spel kunnen spelen dat je gaat programmeren. En je krijgt er naar keuze een batterij of een netvoeding bij zodat je het thuis kunt spelen zonder dat je verder iets nodig hebt! Maar alleen als je het leuk vindt om dat te doen en er thuis iets mee gaat doen. Want ook al is zo'n plankje niet zo duur, het is zonde als het ding bij het huisvuil belandt.

EN WAT MAKEN WE DAN?

We beginnen een spel te maken dat kijkt wie het vaakste op een knop kan drukken. We tellen het aantal keer van elke partij en laten dat zien.

Als er tijd is maken we de speeltijd instelbaar met een draaiknop.

Dat lijkt heel veel, en dat is het ook. Maar je hoeft niet alles te maken om toch een leuk spel te hebben.

DE ARDUINO IDE

Bij Scratch konden we programmeren en het programma laten lopen op dezelfde computer: de Raspberry Pi. De Arduino Nano is zo klein en kan zo weinig dat je het programma daar niet kunt programmeren. Er kan geen beeldscherm en geen toetsenbord aan.

Daarom doen we het anders. We gebruiken de Raspberry Pi om te programmeren. Dat doen we met een programma dat heet: Arduino IDE. IDE staat voor Integrated Development Environment. Dat betekent *geïntegreerde ontwikkelomgeving*.

En als we een stuk programma klaar hebben dan gaan we het *uploaden* in de Arduino Nano. Zodra dat gedaan is wordt het programma in de Arduino Nano uitgevoerd.

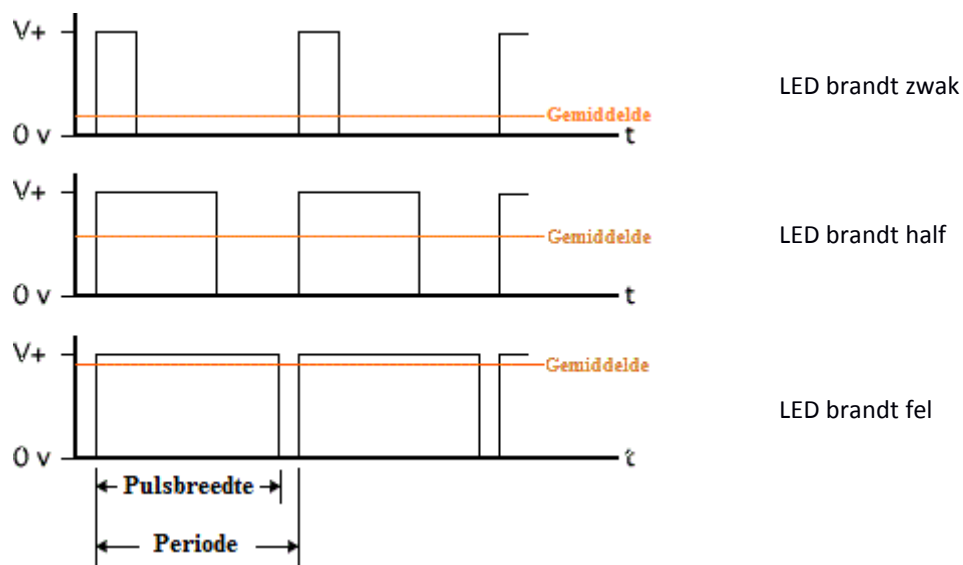
Dus elke keer als je een stuk programma wilt testen moet je het eerst uploaden. Dat kost gelukkig maar een paar seconde.

BIJLAGE B: PULSBREEDTE MODULATIE

De opdracht `analogWrite` gebruik je om het middelste LEDje te dimmen. Je zou kunnen denken omdat het *analog* heet dat de spanning op de LED die normaal 5 Volt is hierdoor wordt veranderd. Maar dat is niet zo.

Wat er gebeurt is dat LED heel snel aan en uit wordt gedaan. Dus eigenlijk knippert de LED heel snel. En dat gaat zo snel dat je het met het oog niet kunt zien. Jouw oog kan maximaal 90 knipperingen per seconde zien. Dat wil zeggen er zijn mensen die dat nog kunnen zien. Maar dat zijn er heel weinig. De meeste mensen zien bij minder knipperingen het al niet meer knipperen.

De `analogWrite` laat de LED ca. 500 keer per seconde knipperen. Dus jij ziet een LED die wat minder hard brandt.



Als de LEDje maar heel kort aan is (zie het bovenste plaatje), dan zie je de LED heel zwak branden. Als de LED bijna de hele tijd aan is dan zie je hem heel fel branden.

Als je nu de drie kleuren kunt dimmen, kun je de drie kleuren in de LED verschillend laten branden. En jouw oog ziet dan een nieuwe kleur. Zo kun je alle kleuren maken.

Deze techniek heet *pulsbreedte modulatie*. Dat betekent dat de breedte van de puls wordt veranderd (gemoduleerd).

De Engelse term is meer gebruikelijk: *Pulse Width Modulation* (PWM).

BIJLAGE C: DE OPDRACHT MET DE POTMETER EN VERANDERING VAN LED FELHEID

Bij het veranderen van de kleur door de potmeter hebben we deze opdrachten gebruikt. Dat verdient wel wat uitleg.

```
LEDWaardeNieuw = LEDWaardeBegin  
    + map(PotMeterStand-PotMeterStandBegin,-1024, 1024, 260, -260);  
LEDWaardeNieuw = constrain(LEDWaardeNieuw, 0, 255);
```

We willen zorgen dat als je de knop indrukt de helderheid van de LED niet verandert. Daarom trekken we de beginwaarde van de potmeter af van de huidige waarde. En met dat verschil gaan we de helderheid aanpassen.

De potmeter kan waarden hebben tussen 0 en 1024. De waarde voor de LED moet zijn tussen 0 en 255. Dus we moeten iets vertalen. Maar er is nog iets.

We gebruiken het verschil tussen de oude en de nieuwe stand van de potmeter. Dan zijn dit de uiterste waarden:

- Het ene uiterste: 1024
 - PotMeterStandBegin = 0
 - PotMeterStand = 1024
 - Dus PotMeterStand-PotMeterStandBegin = 1024
- Het andere uiterste -1024
 - PotMeterStandBegin = 1024
 - PotMeterStand = 0
 - Dus PotMeterStand-PotMeterStandBegin = -1024

In het ene uiterste willen we de waarde met 255 vermeerderen.

In het andere uiterste willen we de waarde met 255 verminderen.

Dat kunnen we doen door deze waarde op te tellen bij de huidige waarde:

```
map(PotMeterStand-PotMeterStandBegin,-1024, 1024, -255, 255)
```

Wat we ook willen is dat als de potmeter in de ene uiterste stand staat en de waarde van de LED is 0 of 255 we in een volledige slag van de potentiometer naar het andere uiterste kunnen, dus resp. 255 of 0.

Maar sommige potentiometers geven niet helemaal 0 of helemaal 1024. Daarom passen we de waarden een beetje aan en gebruiken 260 i.p.v. 255 als maximale waarde om bij de waarde van de LED op te tellen of ervan af te trekken. Waarom 260? Dat blijkt te werken. Dus dan krijgen we:

```
map(PotMeterStand-PotMeterStandBegin,-1024, 1024, -260, 260)
```

En dan hebben we al gezien bij het instellen van de speeltijd dat de potmeter van hoog naar laag gaat als van links naar rechts draait. Terwijl we meer gewend zijn aan het omgekeerde. Dat kunnen we simpel oplossen:

```
map(PotMeterStand-PotMeterStandBegin,-1024, 1024, 260, -260)
```

Maar als we dit optellen bij de oorspronkelijke waarde van de LED, dan kunnen we buiten de geldige waarden van 0 t/m 255 komen. Dat lossen we simpel op door deze opdracht, waarbij de waarden onder de 0 op 0 worden gezet en alles boven 255 wordt 255.

```
LEDWaardeNieuw = constrain(LEDWaardeNieuw, 0, 255);
```