Weekendschool – Programmeren – Les 3b – Programmeren van Arduino

# 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 *geintegreerde 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.

# Starten

## Y:\2016-10-15-205610_177x599_scrot.pngGeef je naam en emailadres

Als je het programma toegestuurd wilt krijgen, vergeet dan niet om je naam en emailadres op te geven. Ook al heb je dat vandaag al eerder gedaan, we gebruiken nu een andere Raspberry Pi en daarom moet het nog een keer.

## 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. Maar veel belangrijker dan hoever je komt is dat je snapt wat je doet. Dus als er iets is wat je niet snapt, overleg dan met elkaar. En je kunt natuurlijk ook je begeleider vragen om het uit te leggen.

Verder: dit is de eerste keer dat we deze les doen, dus het is wat experimenteren hoe snel jullie zullen gaan.

## 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 kommentaar.

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.

* 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 LEDGroenLinks 2 // De groene LED links is aangesloten op pen 2

#define LEDGroenRechts 11 // De rechter groene LED is aangesloten op pen 11

#define LEDRood 5 // De rode LED zit op pen 5

#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 8 // Dit is de klok

#define DIO 9 // 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(){ // Dit is openingsaccolade die bij setup hoort.

pinMode(KnopLinks, INPUT\_PULLUP); // 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(LEDGroenLinks, OUTPUT); // De pen waaraan de linker groene LED zit is een

// uitvoer (output).

pinMode(LEDGroenRechts, OUTPUT); // Idem voor de rechter groene LED.

pinMode(LEDRood, OUTPUT); // Idem voor de rode 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(LEDGroenLinks, HIGH); // Laat de linker LED branden

delay(1000); // Wacht 1000 milliseconde (ms) = 1 seconde

digitalWrite(LEDGroenLinks, 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.

Wat je hier ziet zijn deze opdrachten:

digitalWrite 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.

delay Delay betekent in het algemeen: *vertraag*. In dit geval betekent het *wacht*: wacht het aantal milliseconde dat tussen haakjes staat. Milli is 1/1000. Dus met delay(1000) wacht het programma 1 seconde tot de volgende opdrachtregel wordt uitgevoerd.

## Uittesten

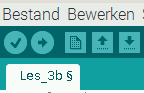
We gaan nu het programma laden in de Arduino. Maar eerst moet het programma worden vertaald in een code die de Arduino begrijpt. Dat heet ***compileren***.

Controleer dat er rechtsonder op het scherm dit staat:  


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*.



## Werkt het?

Als het goed is gegaan, dan knippert de linker groene 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 0,5 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 punt-komma (;).

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

Een punt-komma staat niet achter:

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

# Nu breiden we uit

Laat nu ook de rode LED knipperen:

digitalWrite(LEDRood, HIGH); // Laat de rode LED branden

delay(500) // Wacht 500 milliseconde (ms) = 0,5 seconde

digitalWrite(LEDRood, LOW); // Doe de rode LED weer uit

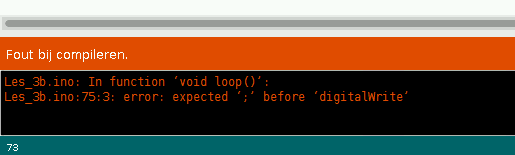
delay(100); // Wacht 100 milliseconde (ms) = 0,1 seconde

* 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 blanko 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(LEDRood, 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 groene en nu ook de rode LED knipperen.

# Jullie eerste eigen programmaregels

* Verander het programma nu zo dat nadat de rode LED is uitgegaan, de rechter groene LED aan gaat gedurende 0,5 seconde en dan uit gedurende 0,2 seconde.

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

* 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.

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 regel:

#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 ziet met de opdracht

VierCijfers.showNumberDec(Getal, Voorloopnullen, AantalPlekken, PlekVanEersteCijfer)

* Getal Het getal dat je wilt laten zien
* Voorloopnullen Of voorloopnullen vertoond moeten worden. Voorloopnullen zijn nullen voor het getal. Als je het getal 1 wilt laten zien op 4 posities met voorloopnullen 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 dan 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 variabelen

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 variable 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.

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 pen 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 variable 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 de LED 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-else.

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 maar even uit aan een begeleider.
* Upload het programma.

Werkt het? Hint: je moet de knop wel een tijdje vasthouden!

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.

unsigned long SpeelTijd = 15; // de speeltijd is 15 seconde

* 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 ==== Op deze plek 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 ==== Hier komen straks de opdrachten die kijken naar de knoppen.

}

// ==== PLAATS 2 ==== Hier komen straks de opdrachten om de winnaar te laten zien.

}

}

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.

## 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 dat je voor links hebt kunt kopieren 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 beiden 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. De geven ook opdracht om de cijfers van de verliezer te laten zien. Die stonden er al, mar 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.

## 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 brukkbare

// waarde voor de speeltijd

VierCijfers.showNumberDec(SpeelTijd, false,2, 1); // laat de speeltijd zien

}

### Werkt de potmeter logisch?

Normaal ben je gewend als je aan een knop naar rechts draait dat de het meer wordt: bij voorbeeld 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. 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?

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.

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

Daar zijn twee mogelijkheden voor

* Het programma aanpassen
* Elektronica toevoegen

Je gaat nu eerst zelf zo'n plankje maken. En als we dan tijd over hebben passen we het programma aan. Hebben we dat niet, dan zetten we er wat elektronica bij.