micro:bit kaarten

Maker

Handleiding voor docenten

Inhoud

Vooraf

Hartstikke leuk dat je de micro:bit kaarten gebruikt. Op dit moment zijn er drie soorten kaarten:

- Explorer
- Programmer
- Maker

Deze handleiding beschrijft het gebruik van Maker kaarten, voor de Explorer/Programmer kaarten is er een aparte docentenhandleiding.
Als je opmerkingen en aanvullingen hebt dan horen we die graag!

Voortbouwend op de ontwikkeling van de Explorer en Programmer kaarten hebben DevLab Academy en CodeKids gewerkt aan de totstandkoming van de Maker kaarten.

Stichting DevLab Academy

CodeKids.nl

Als onderdeel van een Europese samenwerking, mogelijk gemaakt door Erasmus+, zijn de kaarten op diverse scholen getest.

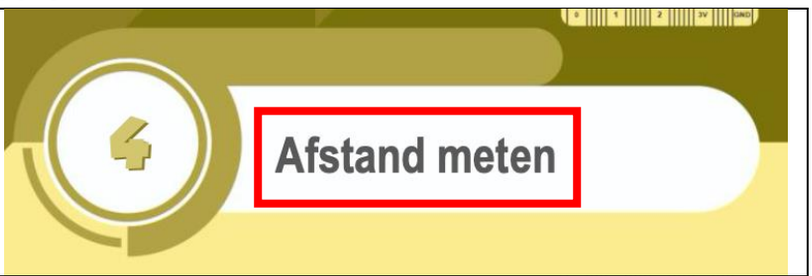
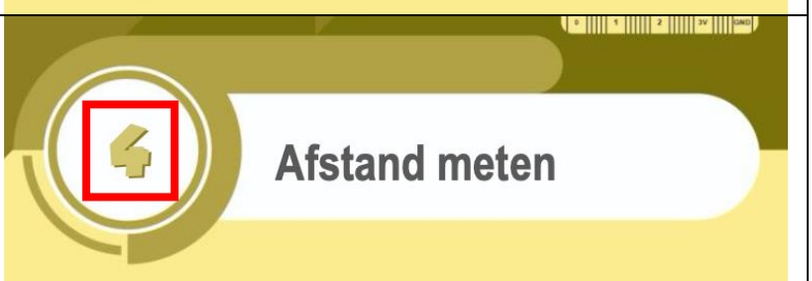
Erasmus+

Correspondentie over de kaarten kan gericht worden aan:

- Lex van Gijsel, Stichting DevLab Academy, lex.van.gijsel@devlab.nl
- Chris Dorna, CodeKids.nl, chris@codekids.nl

Tip	In Bijlage 1 staat hoe je de kaarten zelf het beste kan afdrukken en afwerken. De kaarten zijn ook te downloaden en bestellen via de webshop van de micro:bit NL community: www.micro-bit.nl.

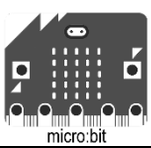
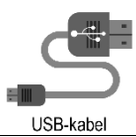
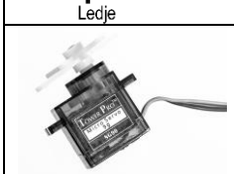
Onderdelen van iedere kaart

Titel van de kaart	
Nummer van de kaart	
Titel van de opdracht	**Maak het contact op de fiets** Op de foto zie je een voorbeeld van het contact. Dit bestaat uit een stuk ijzerdraad dat je om de spaken buigt en een kroonsteentje, draad en twee bindbandjes
Bouw het programma	**Het programma (digitaal)** Met dit programma krijgt de led iedere seconde een andere kleur als je op knop A drukt. 

Ik snap het	**Ik snap het** Het dendereffect van het contact lossen we op door telkens 200 ms te wachten. Als je dit niet zou doen, geeft de afstandsmeter dan meer of Dit zijn vragen/opdrachten die de leerling aan zichzelf kan stellen om te bepalen of hij/zij het snapt. In deze handleiding vind je de antwoorden op de vragen.

Dit heb je nodig

Voor de Maker opdrachten heb je het volgende:

	micro:bit		USB-snoertje
	Laptop (met internetverbinding)		micro:bit aansluitstrip met schroeven en moeren.
	LED		RGB LED
	servo SG90		piezo luidspreker met draadjes
	kroonsteentjes		draad (div. kleuren)
Zie ook de Maker kit, waarin de meeste materialen zitten. <<<verwijzing naar de Maker kit>>>			
Naast bovengenoemde materialen gebruiken we alledaagse materialen zoals o.a. papier, hout, karton, ijzerdraad, aluminiumfolie, tie-wraps en plakband.			
Voor het bouwen heb gangbare gereedschappen nodig zoals een kleine kruiskopschroevendraaier (o.a. t.b.v. de schroefjes), een kleine platte schroevendraaier (o.a. voor het kroonsteentje), schaar, zaag, kniptangetje, buigtangetje, striptang (voor het ontbloten (strippen) van geïsoleerd draad).			

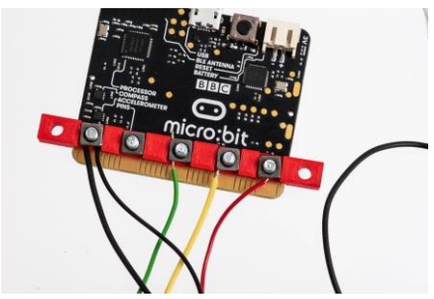
Externe componenten aansluiten

Voor het op de micro:bit aansluiten van externe componenten zoals een RGB led, een luidspreker of een servo zijn meerdere mogelijkheden. Er wordt veel gewerkt met krokodillenklemmen omdat hier geen gereedschap voor nodig is. In de praktijk voldoet dit echter niet altijd omdat er al gauw kortsluiting gemaakt wordt met andere contactvlakjes

op de micro:bit, en andere draden of krokodillenklemmen. Bovendien is het, afhankelijk van de kwaliteit van de krokodillenklem niet altijd makkelijk om de klem open te knijpen als gevolg van de isolatie die eromheen zit en bij het inknijpen wegschuift.

In de MAKER kaarten (zie foto's) gebruiken we de aansluitstrip.





Hiermee is het mogelijk om losse draden (aan het einde gestript = ontdaan van de isolatie) aan te sluiten door deze met een schroef en moertje in de strip te klemmen. Hiervoor is dan wel een kleine kruiskopschroevendraaier nodig. Doordat de moertjes vierkant zijn kunnen deze niet meedraaien als ze 'gevangen' zitten in de aansluitstrip, dus daar is geen gereedschap voor nodig. Uiteraard is ook een striptang nodig om het einde van de draden te ontdoen van de isolatie (1 cm volstaat meestal).

In de kaart van de pratende plant wordt uitgelegd hoe de servo ontdaan kan worden van de zwarte connector, zodat ook de servo met van de aansluitstrip kan worden aangesloten.

Werken met de micro:bit kaarten

Hieronder een paar overwegingen die wij hebben gemaakt bij het schrijven van de kaarten en wat praktische tips voor jouw micro:bit lessen.

Voor welke leeftijden?

De kaarten zijn voor kinderen en leerlingen in de bovenbouw PO en onderbouw VO.

Taal

In de eerdere serie kaarten werden op de Programmer kaarten Engelstalige blokjes gebruikt. Naar aanleiding van de ervaringen en recent gepubliceerd onderzoek hebben we besloten om de Nederlandse blokjes te gebruiken op alle kaarten, dus zowel Explorer, Programmer als Maker.

JUNE 26, 2017 BY BENJAMIN MAKO HILL

Learning to Code in One's Own Language

I recently published a paper with Sayamindu Dasgupta that provides evidence in support of the idea that kids can learn to code more quickly when they are programming in their own language.

https://mako.cc/copyrighteous/scratch-localization-and-learning

Opstelling in de klas

Op veel scholen zullen de MAKER opdrachten uitgevoerd worden in een techniekokaal, of in ieder geval een lokaal waar de nodige voorzieningen zijn om dingen te 'maken'. De opstelling in de klas zal hierdoor meestal al bepaald zijn. Voor zover er nog vrijheid is om de opstelling aan te passen, kan rekening gehouden worden met de volgende tips:

- Door het lange formaat van de kaarten kunnen ze ook op relatief kleine schooltafels naast het werkstuk, de laptop of het toetsenbord worden gelegd.
- Als je werkt met extra laptops zorg dan dat niemand over de aansluitsnoeren kan struikelen.
- Wij hebben gemerkt dat je als begeleider het prettigst werkt als de tafels in een U-vorm staan en je dus niet om de tafels hoeft te lopen om mee te kunnen kijken op het scherm van de leerlingen.

Meer leerlingen achter één computer

Het is goed mogelijk om twee leerlingen tegelijk met één laptop en één micro:bit te laten werken. Je hebt niet alleen minder apparatuur nodig, maar leerlingen leren ook van elkaar. Een paar tips om het samenwerken nog beter te laten verlopen:

- Koppel leerlingen van gelijk niveau.
- Maak afspraken over het afwisselen van het bedienen van computer. Bijvoorbeeld na iedere micro:bit kaart wisselen.
    - Bij de weekendschool wordt ook gewerkt in koppels. Hierbij zijn op de computers twee muizen aangesloten en dit blijkt verrassend goed te werken.
- Laat leerlingen met elkaar overleggen over de vragen die op de kaarten staan.
- Zorg dat er ruimte genoeg is om te werken en er voldoende ruimte is tussen de verschillende koppels.

Laat leerlingen zo zelfstandig mogelijk werken

- Laat de leerlingen zelf de muis en het toetsenbord hanteren. Vertel wat ze moeten doen in plaats van het voor ze te doen. Ook als leerlingen elkaar helpen is dit een goede gewoonte.

    Programmeerles geef je met de handen in de zakken of met je handen op de rug.
- Leren programmeren is vooral een kwestie van ontdekken en uitproberen. Het is geen enkel probleem (en zelfs de bedoeling) dat leerlingen afwijken van de opdrachten op de kaarten.
- Je zal merken dat sommige leerlingen heel erg snel gaan of met onverwachte oplossingen komen. Laat ze aan andere leerlingen hun ideeën en oplossingen vertellen.

Hoe lang duurt een les?

Dat is moeilijk te zeggen. Leerlingen kunnen de kaarten in hun eigen tempo doorwerken. Wij hebben gemerkt dat je rekening moet houden met een concentratieboog van 50 – 60 minuten, waarbij het goed is om iedere 20 minuten een breekpunt in te bouwen (bijvoorbeeld tussentijdse uitleg of elkaar laten vertellen hoe ver je bent).
Om ergonomische redenen is het belangrijk dat kinderen zich na ongeveer 45-60 minuten even kunnen bewegen.

Nummering van de kaarten en de opdrachten

De volgorde van de kaarten is redelijk willekeurig en we adviseren om vooraf na te denken over de beste volgorde, afhankelijk van de voorkennis van de leerlingen, de kennis van de docent en de beschikbare materialen voor het bouwen van de opdrachten.

Werken met de micro:bit

Welke browser?

De kaarten zijn getest op een Windows computer met Google Chrome als browser. De editor werkt ook in andere browsers en op andere computersystemen.

Als je werkt op een Apple computer dan ziet het downloaden en het kopiëren van bestanden er anders uit. Het pairen van een micro:bit met de browser Safari kan problemen opleveren.
Als je werkt met een Chromebook of een tablet (iPad) dan zie je op de kaarten Explorer 1d en 1e hoe je een programma overzet naar de micro:bit.
Op https://makecode.microbit.org/browsers vind je de eisen die aan het system worden gesteld.

Microsoft Makecode for micro:bit

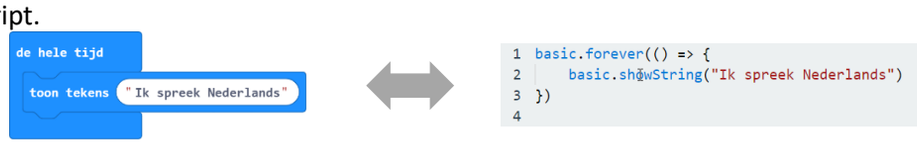
De micro:bit kan met verschillende editors worden geprogrammeerd. De Explorer- en Programmerkaarten zijn gebaseerd op Microsoft MakeCode for micro:bit (versie 1): https://makecode.microbit.org/.

Microsoft?

MakeCode is volledig open source en de broncode wordt gedeeld via het Github platform. De Make Code ondersteunt naast de micro:bit ook verschillende andere systemen en programma's waaronder Minecraft en Lego® Mindstorms®.

Blokken / { } JavaScript

Je kan in de Editor eenvoudig omschakelen van het programmeren in blokken of in JavaScript.



Op de kaarten maken we gebruik van de blokeditor. Op een enkele Programmer kaart maken de leerlingen voorzichtig kennis met de JavaScript editor.

USB-kabel

De standaard USB-kabel van de micro:bit is maar 15 cm lang. Zeker als je (voor de klas) iets wilt demonstreren is een langere kabel handig. Je kan hiervoor iedere micro usb-kabel gebruiken.

Let op: USB-kabels die worden meegeleverd met opladers of powerbanks zijn vaak alleen maar geschikt voor de voeding (stroom) en hebben niet de draadjes die nodig zijn om data te versturen.

Batterijen

De micro:bit kan zijn voeding halen uit de USB-aansluiting of uit een batterijhouder. Nu gaat het aansluiten van de batterijhouder niet echt makkelijk en heeft de houder die standaard wordt meegeleverd geen aan/uit schakelaar. Bij een aantal opdrachten gaan de leerlingen met een geprogrammeerde micro:bit op pad. In dit geval gebruiken ze de (onhandige) batterijhouder of een USB-powerbank.

Let op: Sommige (grotere) powerbanks hebben een beveiliging die ervoor zorgt dat de stroomtoevoer wordt uitgeschakeld als de afgenomen stroom laag wordt (als de mobiele telefoon is opgeladen). Deze powerbanks zijn niet te gebruiken omdat de micro:bit heel weinig stroom gebruikt.

Legenda in deze handleiding

Ter verduidelijking van de uitleg worden in deze handleiding de volgende notaties gebruikt:
naam van een variabele
waarde van een variabele
naam van een blokje

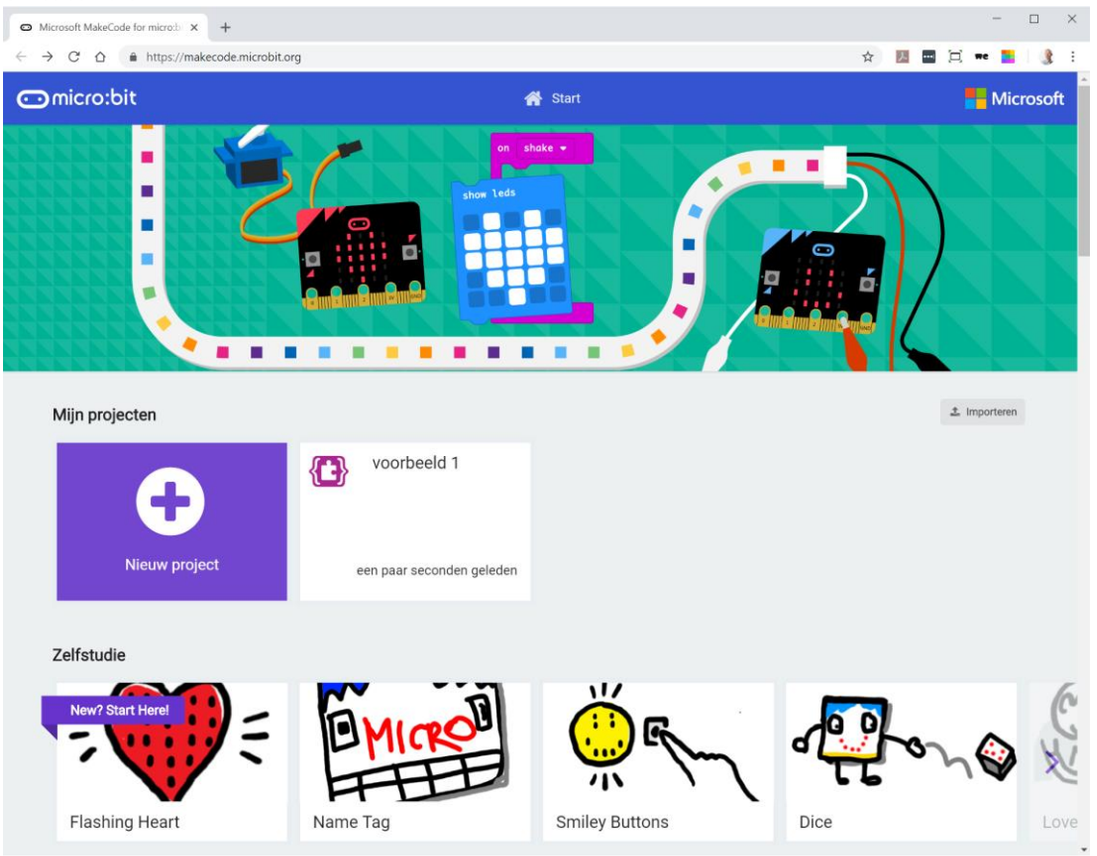
Meer weten?

Voorbeelden

Via de knop Start heb je toegang tot tientallen projecten en voorbeelden.

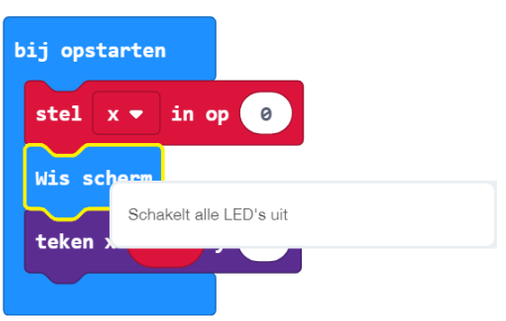
Startpagina

Op de startpagina staan de projecten die je zelf hebt gemaakt en verschillende voorbeelden en tutorials.



Helpfunctie

Als je met de muis even boven een blokje blijft hangen verschijnt er een korte uitleg over het blokje.



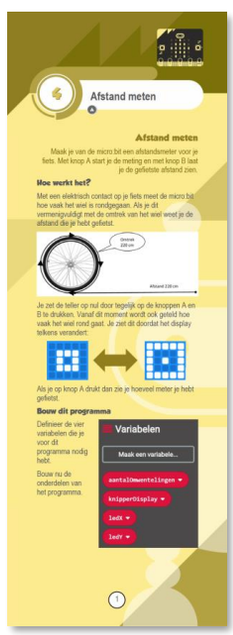
Als je meer wilt weten over de werking van een blokje klik er dan met je rechtermuisknop op en kies *Help*.

Maker kaarten

Bij de Maker kaarten gaat het om het realiseren van een werkend project.

Bij elk project wordt iets gemaakt van hout, papier, karton, elastiekjes etc. en wordt de micro:bit gebruikt voor het meten van een verschijnsel of het besturen van een servo, luidsprekertje of led, maar steeds ten dienste van het eindresultaat.

Er moet uiteraard ook geprogrammeerd worden, maar dit staat in tegenstelling tot de Explorer en Programmer kaarten niet meer op de voorgrond.

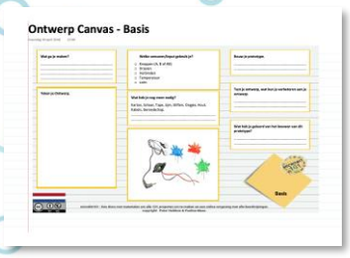


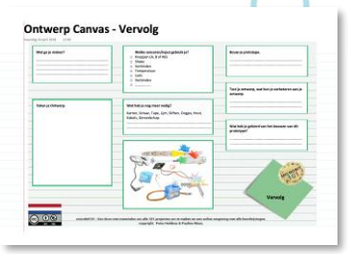
Verband Computational Thinking / leerlijn Programmeren in het PO en de MAKER kaarten

Per kaart wordt aangegeven welke Computational Thinking begrippen geraakt worden in die kaart, met daarbij een korte toelichting.

Ontwerpcanvassen

Aangeraden wordt om na elke Maker les, of in ieder geval op regelmatige basis, de leerlingen de kans te geven om met hetgeen ze geleerd hebben ook zelf een idee uit te werken. Als er na het doorlopen van een Maker les iets geleerd wordt over lussen, voorwaarden of bijvoorbeeld variabelen dan kan dat aanleiding geven tot de gedachte: "oh, maar als je dat kunt met een micro:bit dan zou je ook .... kunnen".





bron: De canvassen zijn ontwikkeld door Pauline Maas (4PIP) en Peter Heldens (Microsoft-Nederland).

De ontwerpcanvassen zijn hulpmiddelen om het ontwerpproces van leerlingen te stroomlijnen. Op die manier kan op een structurele manier gewerkt worden aan idee-creatie, de uitwerking daarvan en uiteindelijk de realisatie. Meestal worden deze ontwerpopdrachten uitgevoerd in groepjes van 2 of 3 leerlingen.

De canvassen zijn er in 3 verschillende moeilijkheidsgraden, basis, vervolg en expert. Het is aan de leerkracht/docent om te bepalen welk van de canvassen het meest past bij de betreffende klas/leerling(en).

De canvassen zijn als bijlagen te vinden in deze handleiding. Ze zijn echter ook te downloaden via de "micro:bit community NL" website: www.micro-bit.nl/lesmateriaal zodat de canvassen in het gewenste aantal geprint kan worden voor gebruik in de klas.

Een suggestie voor een lesaanpak met behulp van een ontwerpcanvas ziet er als volgt uit:

00:00-00:15	Terugblik op de Maker les • Wat hebben we gemaakt? • Welke programmeercodes hebben we gebruikt? • Vertel iets over de gebruikte blokjes
0:15-00:30	Docent legt het canvas uit en vertelt hoe de leerlingen dit kunnen gebruiken • Groepjes formeren Materiaal uitdelen
00:30-01:20	De leerlingen bedenken en ontwerpen met behulp van het ontwerp canvas een eigen prototype van een micro:bit toepassing, gebruik makend van de programmeerprincipes die in de Maker lees geleerd zijn
01:20-01:30	De leerlingen presenteren wat ze gemaakt hebben
01:30-02:00	Afsluiten en inleveren van het materiaal. Neem ook het ontwerp canvas en schrijf hier feedback op, zodat de leerlingen in een vervolgles nog beter met het ontwerp canvas kunnen omgaan en tot een prototype kunnen komen.

Over DevLab en CodeKids

De Stichting DevLab Academy en CodeKids hebben als doel om techniek, en in dit geval programmeren en micro:bit op een duurzame manier te integreren in het onderwijs. Dit omdat we er in geloven dat deze manier van kennismaking met elektronica en programmeren van toegevoegde waarde is voor het onderwijs van vandaag en morgen. In de toekomst zijn vaardigheden op het gebied van computational thinking niet meer weg te denken. DevLab en CodeKids vormen hiervoor samenwerkingsverbanden tussen bedrijven en instellingen die eenzelfde doel nastreven.

Maker 1: micro:beat

In deze les wordt een muziekinstrument gemaakt met behulp van de micro:bit. Het idee is gedeeltelijk gebaseerd op de Theremin, een instrument waarmee je toonhoogtes en sterkte regelt met hand en armgebaren. De micro:beat maakt pulserende tonen waarvan de toonhoogte (pitch) geregeld kan worden door de oriëntatie/helling van de micro:bit te veranderen.

De snelheid waarmee de pulsen gegeven worden, kan worden bestuurd door contact te maken tussen een met aluminiumfolie omwikkelde staaf en ijzerdraadjes op een plankje. De draadjes zijn zodanig aangebracht dat een binaire code wordt doorgegeven aan de micro:bit. Deze code bepaalt het tempo van de geluidspulsen.

Het is daarna aan de gebruiker van het instrument om hiermee muziek te maken. Iedereen krijgt er geluid uit, maar voor een mooi muzikaal resultaat moet je micro:beat leren bespelen.

Computational Thinking

In deze les worden de volgende aspecten geraakt uit het begrippenkader **Computational Thinking** en de leerlijn programmeren:

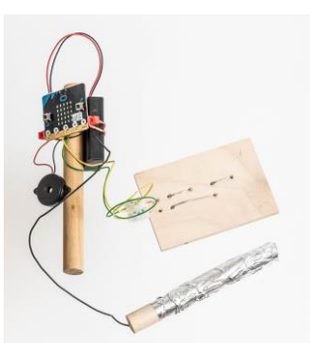
- ***Gegevens verzamelen***: er wordt op 2 poorten gemeten welke poorten aangeraakt worden om een binaire code af te lezen.
- ***Gegevens analyseren***: de binaire waarde wordt bereikt door de gegevens van 2 input poorten te combineren tot een binaire waarde en getal.
- ***Gegevens visualiseren***: het aanraken van ***P1*** en ***P0*** wordt zichtbaar gemaakt d.m.v. de ledjes van het matrix display.
- ***Probleem decompositie***: het algoritme is onderverdeeld in twee metingen, een totaalwaarde-bepaling, een omzetting naar een pauze-waarde en het daadwerkelijk afspelen van een toon met een pauze.
- ***Abstractie***: de totaalwaarde is een abstracte weergave van de uiteindelijk echte pauze in milliseconden die je tussen de afzonderlijk geluidstonen wilt wachten.
- ***Parallellisatie***: de processen die het indrukken van ***knop A*** of ***knop B*** detecteren lopen onafhankelijk (parallel) aan het proces om de muziek te maken..

Kaart 1 pag 1

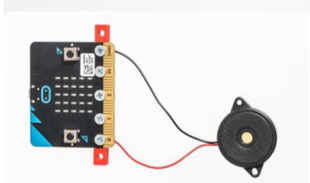
Bouw de micro:beat

Op de eerste foto zie je het eindresultaat van een gebouwde micro:beat. Het instrument bestaat uit 3 gedeelten:

1. Een plankje met 2 ijzerdraden die verbonden zijn met ***P1*** en ***P2*** van de micro:bit.
2. Een met aluminiumfolie omwikkelde staaf, elektrisch verbonden met ***GND*** van de micro:bit.
3. Een micro:bit waarop een luidsprekertje is aangesloten op ***P0*** en ***GND***, gemonteerd op en tweede staaf zodat je die in je hand kunt houden om zodanig de oriëntatie van het geheel makkelijk te kunnen aanpassen.

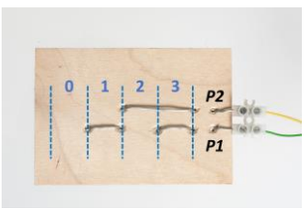


Op de tweede foto is te zien hoe de luidspreker met een aansluitstrip en boutjes/moertjes aangesloten is op de poorten ***P0*** en ***GND***.



Kaart 1 pag 2

Op de eerste foto wordt duidelijk gemaakt hoe de twee ijzerdraadjes door een plankje met gaatjes wordt ‘geregen’. Samen vormen ze een zogenaamde binaire code die kan worden omgezet in een decimaal getal:



binair	***P2***	0	0	1	1
	P1	0	1	0	1
decimaal		**0**	**1**	**2**	**3**

De twee foto’s van de staaf geven aan hoe de elektrische geleidende draad en het aluminiumfolie aangebracht kan worden zodat e.e.a. goed contact maakt met elkaar.

Kaart 1 pag 3

Op de foto zie je het geheel op elkaar aangesloten, maar nog niet op een handvat voor een makkelijke beweging van de micro:bit.

Bouw het programma

Er moeten een aantal variabelen worden gemaakt, van een verschillend type.

Typen variabelen

De MakeCode omgeving van micro:bit geeft de mogelijkheid om met 3 verschillende typen variabelen te werken:

- een **Number**: een numerieke waarde (getal, positief of negatief)
- een **String**: een regel tekst (een letter is een ook tekst van slechts één letter)
- een **Boolean**: een variabele die alleen de waarden ‘waar’ of ‘onwaar’ aan kan nemen

De eerste keer dat een variabele een waarde krijgt, in dit geval in het *bij opstarten* blok, wordt bepaald welk type variabele hier bedoeld wordt. De variabele *beatAan* krijgt als waarde ***onwaar***. Hiermee is bepaald dat het om een **Boolean** gaat. Vanaf dat moment is *beatAan* in het gehele programma een **Boolean**. Het toewijzen van een getal of een letter of tekst is daarna niet meer mogelijk.
De variabele *pauze* krijgt de waarde ***0*** waarmee is bepaald dat *pauze* (voor het hele programma) een **Number** is.

De variabelen worden in het programma steeds aangepast en hebben de volgende functie:

Naam	Type	Functie
beatAan	**Boolean**	Geeft aan of het programma al dan niet de beat moet laten horen, hiermee kun je het geluid daarna in- of uitschakelen.
pauze	**Number**	Bepaalt de lengte van de pauze tussen twee openvolgende geluidspulsen, hoe korter de pauze hoe hoger het tempo van de geluidspulsen.

Naam	Type	Functie
waarde1	**Number**	De waarde die ***P1*** vertegenwoordigd in het decimale getal (in dit geval 1 als ***P1*** hoog is en 0 als ***P1*** laag is).
waarde2	**Number**	De waarde die ***P2*** vertegenwoordigd in het decimale getal (in dit geval 2 als ***P2*** hoog is en 0 als ***P2*** laag is).
totaal	**Number**	De optelling van waarde1 en waarde2, is dus de decimale waarde van de binaire code die op ***P1*** en ***P2*** af te lezen is.

Kaart 1 pag 4

Door op ***knop A*** te drukken wordt *beatAan* op ***waar*** gezet. Met ***knop B*** op ***onwaar***. De waarde van *beatAan* wordt in het programma in het blokje *de hele tijd* bepaald of er geluid klinkt. De code in de lus *terwijl* wordt dus alleen uitgevoerd als *beatAan* ***waar*** is.

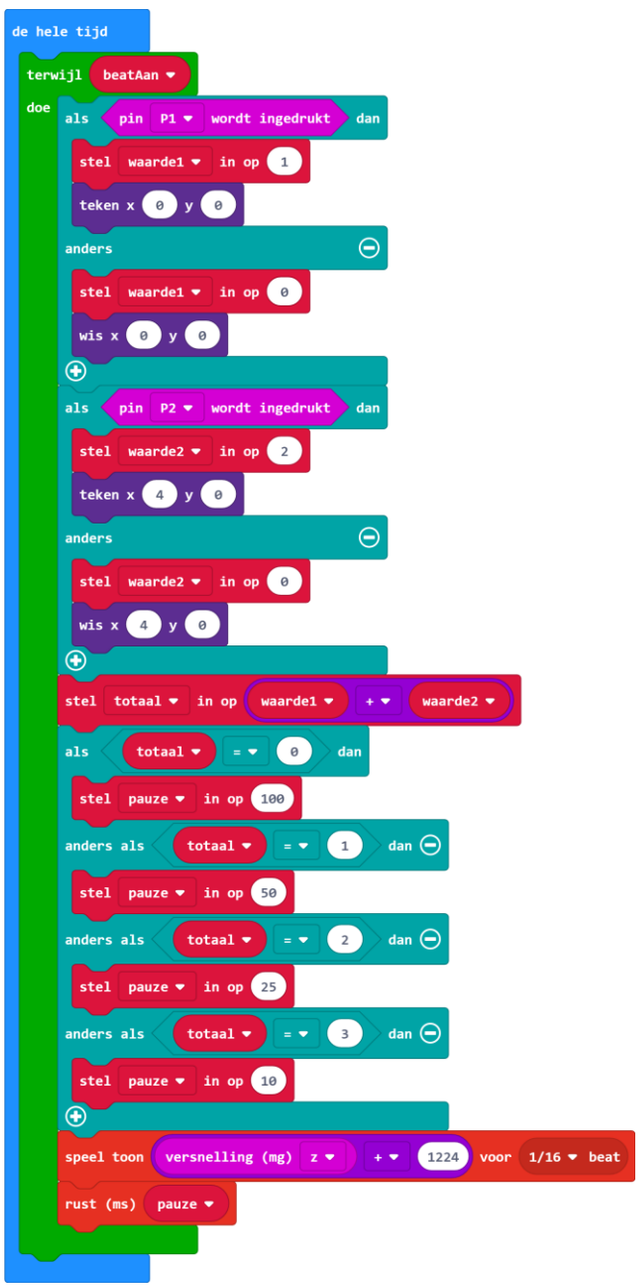
Met de blokken *als pin P1 wordt ingedrukt* en *als pin P2 wordt ingedrukt* wordt gemeten met welke ijzerdraadjes de staaf met aluminiumfolie contact maakt.

Met *teken x y* en *wis x y* wordt een ledje aan of uitgezet zodat je makkelijk kunt zien of de staaf goed contact maakt met de ijzerdraadjes. Op kaart 4 (Afstand meten) wordt meer uitgelegd over de coördinaten x en y.

De variabele *totaal* krijgt zijn waarde (***0***, ***1***, ***2*** of ***3***) op basis van de inputs ***P1*** (*waarde1*) en ***P2*** (*waarde2*). Hiervoor worden *waarde1* en *waarde2* bij elkaar opgeteld.

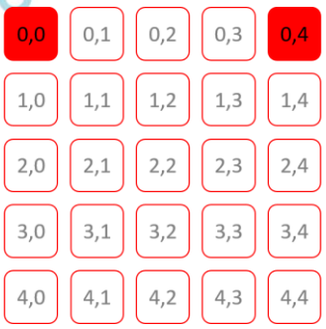
In het laatste *als dan* blok wordt de waarde van *totaal* omgezet in een pauze in milliseconden. Voor de vertaaltabel zie pagina 5 van de kaart.

De laatste twee blokken laten de toon horen met een pauze. De toonhoogte wordt bepaald door de *versnelling z* welke een maat is voor de helling waarmee je de micro:bit vasthoudt.



Kaart 1 pag 5

Op deze pagina wordt uitgelegd met welke coördinaten x en y je een bepaalde led op de led-matrix van de micro:bit kan in- of uitschakelen.



Ook wordt nogmaals weergegeven wat de relatie is tussen de metingen op ***P1*** en ***P2***, hoe deze als binaire code de decimale waarde van *totaal* bepalen, en hoe de waarde wordt om gezet naar de lengte van de *pauze*.

P2	***P1***	binair	*waarde2 + waarde1*	*totaal*	*Pauze*
Nee	Nee	00	0+0	0	100
Nee	Ja	01	0+1	1	50
Ja	Nee	10	2+0	2	25
Ja	Ja	11	2+1	3	10

Kaart 1 pag 6

De toonhoogte van de geluidspuls wordt bepaald door de oriëntatie cq. helling van de micro:bit, dus hoe scheef je hem houdt. De waarde van de versnelling is een getal tussen de -1024 en +1023 en levert zodoende niet het juiste getal op voor een frequentie/toonhoogte. Door hier 1224 op te tellen gaan de waarden variëren tussen de 200 en 2247, en dat zijn ook de toonhoogtes in Hertz die te gehore gebracht worden.

Maker 2: Dobbelsteen

Over de kaarten

In deze les wordt een dobbelsteen gemaakt die het gegooide aantal ogen niet weergeeft met ledjes, maar met piepjes. Dit is nodig als je de dobbelsteen geschikt wilt maken voor blinden of slechtzienden, maar je kunt op deze manier ook een dobbelsteen maken van een kubus waarmee gegooid kan worden, zoals je dat van een echte dobbelsteen ook gewend bent.

De micro:bit wordt dan in de kubus gebouwd en ligt na elke worp in een andere positie, dat kan dus ook onderop zijn, waardoor het display niet meer af te lezen is. De micro:bit meet de stand/oriëntatie van de kubus waarin deze blijft liggen. Afhankelijk van de oriëntatie laat de micro:bit het juiste aantal piepjes horen. Zo is het niet meer nodig dat het display van de micro:bit moet worden afgelezen.

Computational Thinking

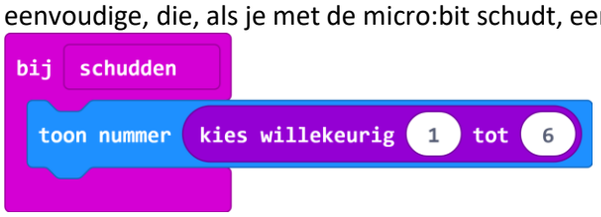
In deze les worden de volgende aspecten geraakt uit het begrippenkader **Computational Thinking** en de leerlijn programmeren:

- ***Gegevens verzamelen***: uitlezen van de sensor met de verschillende mogelijkheden van het *als schudden* blok. Het gebruik van de variabele *ogen* voor het opslaan van de worp.
- ***Gegevens analyseren***: relatie leggen tussen de oriëntatie en het aantal gegooide ogen van de dobbelsteen.
- ***Gegevens visualiseren***: het aantal ogen omzetten in een aantal piepjes (auditief visualiseren)
- ***Probleem decompositie***: stapsgewijze opbouw van de complexiteit, door eerst een dobbelsteen te maken met een getal in het scherm, daarna met oriëntatie en als laatste stap een algoritme voor alle oriëntaties
- ***Algoritmes***: de herhalings-lus voor het aantal piepjes voor elk mogelijke oriëntatie van de dobbelsteen, gebruik makend van de variabele *ogen*.
- ***Automatisering***: de oriëntatie van de micro:bit als trigger voor het bepalen van het gegooide aantal ogen.
- ***Modellering***: de kubus als model voor de dobbelsteen en de relatie met de micro:bit oriëntatie.
- ***Parallellisatie***: de metingen van de oriëntatie lopen allen parallel aan elkaar (voortdurende metingen) en lopen tevens parallel aan de lus *de hele tijd* waarin het aantal gegooide ogen ten gehore wordt gebracht.

Kaart 2 pag 1

Bouw het eerste programma

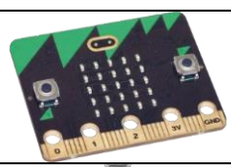
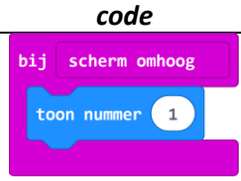
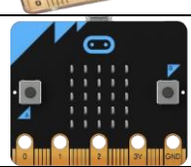
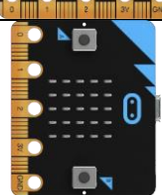
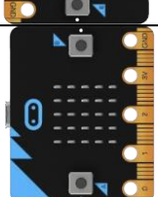
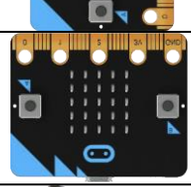
De micro:bit wordt in deze les gebruikt om een dobbelsteen te maken. Allereerst een heel eenvoudige, die, als je met de micro:bit schudt, een willekeurig getal laat zien tussen 1 en 6.



Bouw het tweede programma

In de tweede stap maakt de leerling kennis met de ander mogelijkheden van de 3d accelerometer, de versnellingssensor. Deze is niet alleen in staat om trillingen waar te

nemen, zoals wanneer je *bij schudden* gebruikt, maar ook kan deze sensor waarnemen in welke oriëntatie de micro:bit zich bevindt.
Wanneer de micro:bit dus in een kubus zit die als dobbelsteen gebruikt wordt, kan de sensor detecteren in welke stand de dobbelsteen ligt. Het vereist ruimtelijk inzicht om de oriëntatie van de micro:bit te koppelen aan de gegooide worp van de dobbelsteen. De oplossing/koppeling die in deze les gebruikt wordt, ziet er als volgt uit:

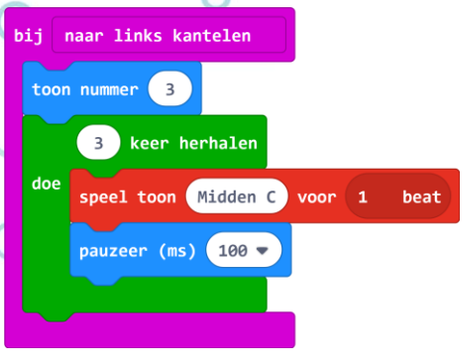
oriëntatie	***ogen***		***code***
Scherm omhoog	1		
Logo omhoog	2		
Naar links gekanteld	3		
Naar rechts gekanteld	4		
Logo naar beneden	5		
Scherm naar beneden	6		

Geluid toevoegen

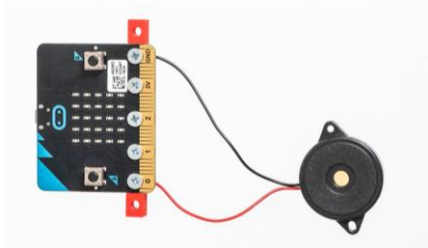
De oriëntatie wordt gemeten met de sensor en de gegooide worp wordt omgezet naar een aantal piepjes. Zo is de dobbelsteen geschikt gemaakt voor blinden en slechtzienden. Het aantal piepjes geeft aan hoeveel ogen er gegooid zijn. De pauze van 100ms tussen de piepjes geeft een korte stilte tussen de piepjes zodat deze makkelijker te tellen zijn.

Kaart 2 pag 2

In het programmavoorbeeld wordt getoond hoe de code er uit ziet als een 3 gegooid wordt. Voor elke mogelijk worp van 1 t/m 6 moet deze code toegevoegd worden met daarbij de juiste oriëntatie en het juiste aantal piepjes.



De luidspreker wordt aangesloten op pin ***P0*** en ***GND*** door middel van de aansluitstrip.



Handiger programmeren

De leerlingen gaan het programma iets verfijnen door het gegooide aantal ogen als getal in een variabele op te slaan.

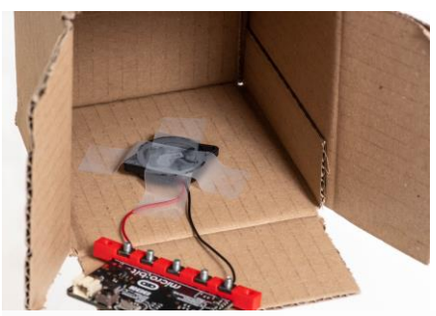
In een aparte lus *de hele tijd* wordt het aantal piepjes ten gehore gebracht dat in de variabele is opgeslagen. Op deze manier hoeft er maar 1 stukje code worden geprogrammeerd voor alle 6 de mogelijke worpen.

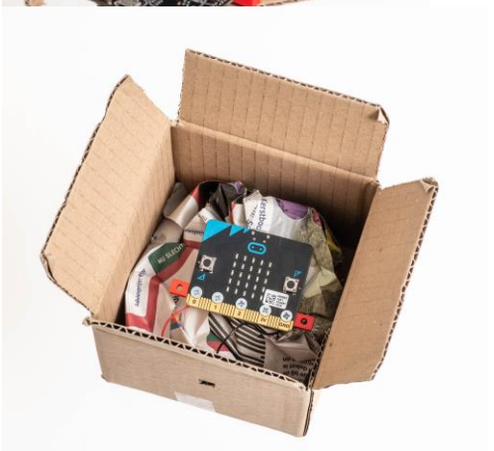
Kaart 2 pag 3 en 4

De code laat zien hoe je aan de hand van de variabele *ogen* het juiste aantal piepjes laat horen. Dit aantal wordt iedere 2,5 sec (= 2500 ms) herhaald. Maak je deze tijd langer, dan moet je gemiddeld langer wachten voordat de piepjes klinken, maak je de tijd korter, dan wordt de worp vaker ten gehore gebracht.

Dobbelsteen bouwen

De tekst op de kaarten aangevuld met de foto's spreken voor zich. Elke andere bouwwijze om hetzelfde te bereiken is natuurlijk ook goed.





Maker 3: Bibberspiraal

In deze les maak je een zogenaamde bibberspiraal, een welbekend spelletje. De bedoeling is om een ring om een spiraal te bewegen zonder de spiraal te raken. Als je de spiraal raakt, ben je af.
Een tweede variant geeft je de kans om een aantal missers te maken, degene die dan het minst aantal missers heeft, heeft gewonnen.
De derde en laatste variant zet ook nog eens tijdsdruk op het spelletje door een maximale tijd te geven om van het begin naar het einde te bewegen. Haal je het niet binnen die tijd, dan ben je ook af, ook al heb je de spiraal nog niet geraakt.

Computational Thinking

In deze les worden de volgende aspecten geraakt uit het begrippenkader **Computational Thinking** en de leerlijn programmeren:

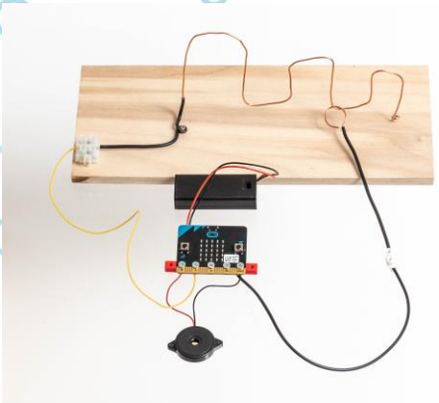
- ***Gegevens verzamelen***: er wordt tijdens het spel gemeten of de spiraal wordt geraakt met de ring, en ook of het eindcontact wordt geraakt.
- ***Gegevens visualiseren***: missers worden aan de gebruiker visueel gemaakt door middel van symbolen op het display.
- ***Probleem decompositie***: het spel wordt verdeeld in het meten van het aanraken, het bijhouden van een puntentelling en het bijhouden van de tijd.
- ***Parallellisatie***: de code in het blokje *wanneer pin P1 wordt aangeraakt*, en *wanneer P2 wordt aangeraakt*, draaien parallel aan de *de hele tijd* lus.

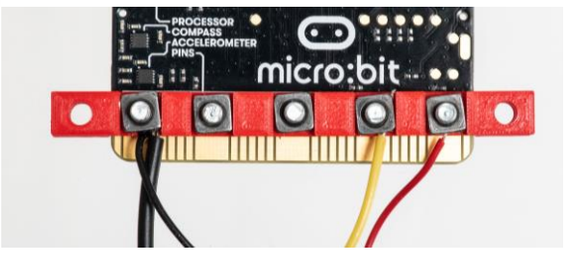
Kaart 3 pag 1

De eerst pagina laat de bouw van de ring en de spiraal zien. Een stuk ijzerdraad kan in de juiste vorm gebogen worden en met een kroonsteentje verbonden worden met een draad. Dit geldt voor zowel de ring als de spiraal. De draad wordt vervolgens op de micro:bit aangesloten.
De spiraal wordt op een plankje gemonteerd om een stabiele verticale opstelling van de spiraal te krijgen.
NB: elke andere manier om een opstelling te creëren die hetzelfde doet is natuurlijk ook goed.

Kaart 3 pag 2

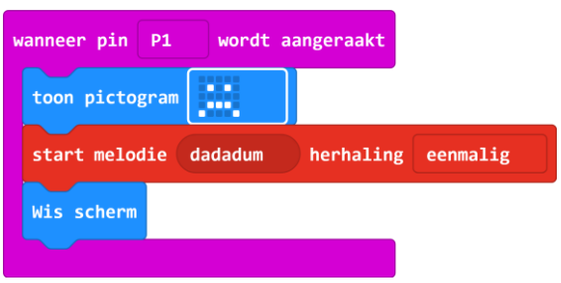
Hier is in de eerste figuur de aansluiting op de micro:bit te zien van de spiraal en de ring, en het luidsprekertje waarmee melodietjes ten gehore worden gebracht, hetzij voor een misser hetzij voor de overwinning. Om de draden en de luidspreker op de micro:bit aan te sluiten wordt gebruik gemaakt van de aansluitstrip met schroefjes en moertjes. De luidspreker wordt aangesloten op ***P0*** en ***GND***. Op ***P1*** wordt gemeten of de ring de spiraal raakt die verbonden is aan ***GND***.





Het basisprogramma

Dit is de eerste variant van het spel waarin alleen maar gedetecteerd wordt of de bibberspiraal geraakt wordt. Als ***P1*** verbonden wordt met ***GND*** doordat de ring de spiraal raakt wordt een 'mislukt' melodietje afgespeeld en een 'sad face' getoond zolang als het melodietje klinkt.

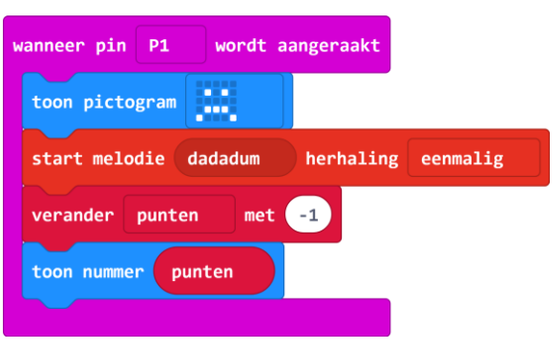


Eenvoudige puntentelling

In de tweede variant van het spel krijg je eerst 10 punten in het blokje *opstarten*. De variabele *punten* wordt op ***10*** gezet.

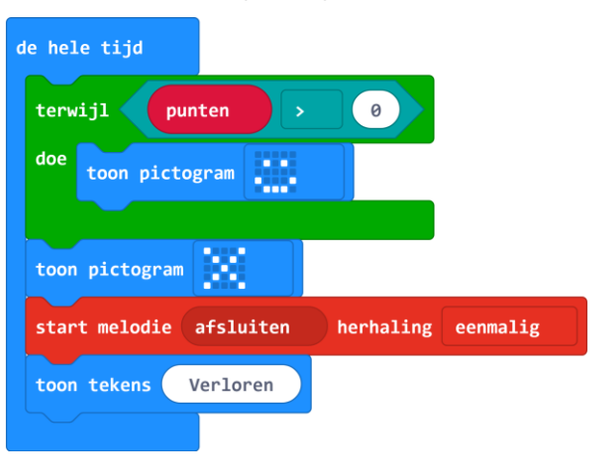
Kaart 3 pag 3

Elke keer dat je de spiraal raakt met de ring gaat er een punt af. Tevens wordt een 'sad face' getoond en een 'mislukt' melodietje afgespeeld.



Uitbreiding 1: Puntentelling

Door in een *de hele tijd* lus een smiley te tonen, zolang het aantal punten groter is dan 0 blijft het programma herhalen in de *terwijl* lus, laat je de speler zien dat hij/zij nog in de race is. Gebruik je alle punten en staat de teller *punten* dus op ***0***, dan springt het programma uit de *terwijl* lus en wordt met een icoontje, geluid en met tekst duidelijk gemaakt dat je verloren hebt.
Als tijdens het spelen de spiraal wordt geraakt, wordt het stukje programma uitgevoerd dat in het *wanneer pin P1 wordt aangeraakt* blok staat, dus 'sad face' etc., en als dat klaar is springt het programma weer terug naar de *de hele tijd* lus met daarin de *terwijl* lus.



Uitbreiding 2: Maximale speeltijd

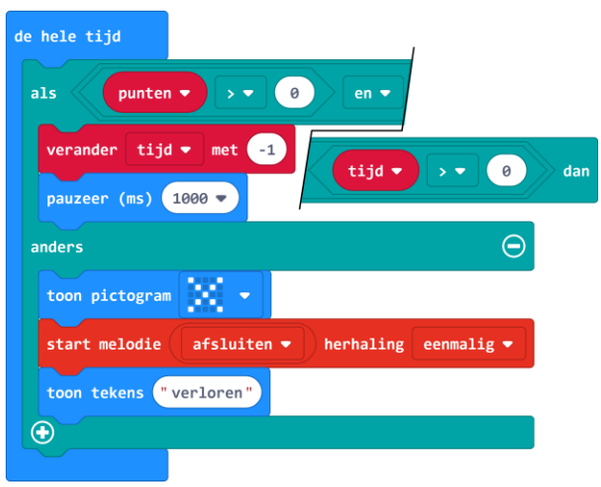
Om het nu nog spannender te maken krijg je een maximum tijd waarin welke het spel moet worden voltooid.

Kaart 3 pag 4

Naast de variabele *punten* (het maximaal aantal missers dat je mag maken) wordt ook een variabele *tijd* gedefinieerd (het maximaal aantal seconden waarin je het spel moet voltooien).

In de lus *de hele tijd* wordt de variabele *tijd* elke seconde met 1 verlaagd door 1000 milliseconden te wachten en dan te verlagen. In het *als dan* blok zie je dat nu twee voorwaarden zijn opgenomen in een **en**-constructie, zodat die allebei ***waar*** moeten zijn:

- *punten* groter dan 0
  en
- *tijd* groter dan 0



*Als de twee voorwaarden in een **of**-combinatie zouden zitten, zou maar één van de voorwaarden **waar** hoeven te zijn.*

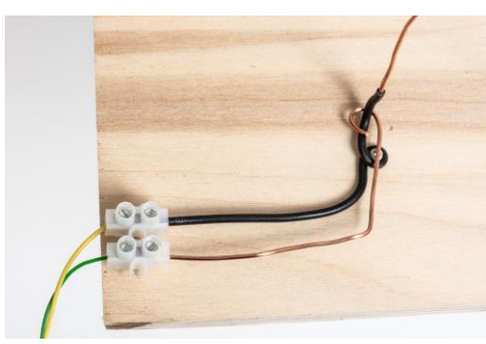
Zolang je nog punten hebt én tijd mag je door, anders ben je af.

De code voor het stukje programma dat wordt uitgevoerd als je de spiraal raakt blijft hetzelfde als in de vorige variant van het spel.

Finish gehaald

Om te kunnen meten of je het spel binnen de tijd hebt gespeeld is er een eindcontact nodig dat registreert of je klaar bent met het spel.

Kaart 3 pag 5



Op de foto zie je dat er een eindcontact is toegevoegd. Deze wordt via een kroonsteentje met een draad verbonden met pin ***P2*** van de micro:bit. Tevens wordt de variabele geïntroduceerd die bijhoudt of de tijdsmeting loopt of niet. We definiëren de **Boolean** variabele *tijdmeting*, die de waarde ***waar*** of ***onwaar*** kan aannemen. In het *bij opstarten* blok wordt *tijdmeting* op ***waar*** gezet zodat vanaf dat moment de tijdregistratie loopt.

De spiraal is aangesloten op ***P1***, net zoals in de vorige variant van het spel. Ook de code voor wanneer de spiraal wordt geraakt is hetzelfde.
Er komt een nieuw stukje code bij om de tijdmeting stop te zetten zodra het eindcontact wordt geraakt door de ring. Dit wordt gemeten op ***P2***, dus de code komt in het blok *wanneer pin P2 wordt aangeraakt*. De variabele *tijdmeting* wordt op ***onwaar*** gezet.


wanneer pin P2 wordt aangeraakt
stel tijdmeting in op onwaar
toon pictogram
stel punten in op 3
stel tijd in op 20
start melodie opstarten herhaling eenmalig

Maker 4: Afstand meten

Over deze kaarten

Deze kaarten laten je een afstandsmeter bouwen met een micro:bit die je op je fiets kan gebruiken. Het aantal omwentelingen van het (fiets-)wiel wordt geteld door de micro:bit, en als je dat vermenigvuldigt met de omtrek van het wiel, dan weet je de afgelegde afstand vanaf het punt waar je begon te tellen.
Tijdens het tellen verschijnt een wisselend figuur op het display van de micro:bit om duidelijk te maken dat het tellen goed gaat. Druk je op ***knop A*** dan zie je de afgelegde afstand, vanaf de plek waar je de micro:bit aanzette of toen je de teller op 0 gezet hebt.
Met ***knop A+B*** zet je de teller op 0.

Computational Thinking

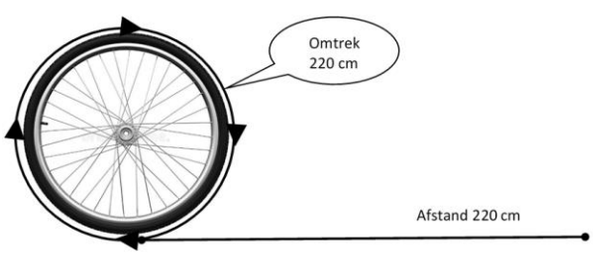
In deze les worden de volgende aspecten geraakt uit het begrippenkader **Computational Thinking** en de leerlijn programmeren:

- ***Gegevens verzamelen***: hardware input, meting van een puls-contact voor het tellen van de omwentelingen, ook het meten of uitrekenen van de omtrek van een wiel
- ***Gegevens analyseren***: verschillende typen variabelen zoals **Number** voor de teller en de coördinaten van het led display en **Boolean** voor het aan en uitzetten van het knipperende display
- ***Gegevens visualiseren***: het door het beeld laten scrollen van de afstand in tekst, ook het laten knipperen van een symbool als indicatie dat de puls-meting goed gaat
- ***Algoritmes***: de lus voor de puls-meting, de geneste lussen voor het knipperen van het display en het berekenen van de afstand
- ***Modellering***: de omtrek van het wiel als maat voor de afstand
- ***Parallellisatie***: terwijl het programma onder ***knop A*** loopt gaat de lus *terwijl waar* door met tellen, deze programmadelen lopen dus parallel, door middel van een semafoor (de **Boolean** variabele *knipperDisplay*) wordt tijdelijk het knipperende display uitgeschakeld

Kaart 4 pag 1

Hoe werkt het

Hier wordt uitgelegd dat je de omtrek van het wiel moet weten om uit te kunnen rekenen wat de afgelegde afstand is.



Tip: Je kan hier eventueel het getal pi introduceren:

$$omtrek = \pi \times diameter$$
$$omtrek = 2 \times \pi \times straal$$
$$\pi = 3{,}14159265358979323846264338327950288419716939937510 \ldots.$$

Bouw dit programma

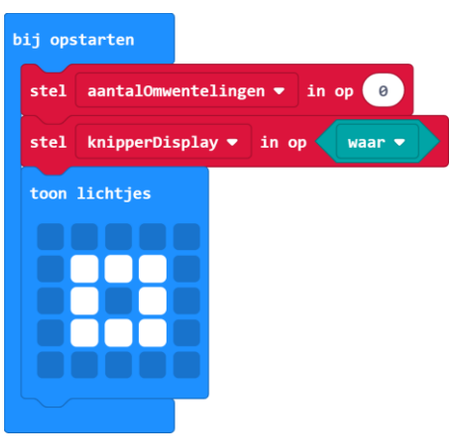
Je hebt een aantal variabelen nodig om e.e.a. te onthouden terwijl het programma uitgevoerd wordt:

Naam	Type	Functie
aantalOmwentelingen	**Number**	Het aantal omwentelingen van het wiel.
knipperDisplay	**Boolean**	Deze variabele kan alleen maar de waarden ***waar*** of ***onwaar*** aannemen en wordt gebruikt om aan te geven of het display met een afwisselende afbeelding moet aangeven dat er geteld wordt. Op het moment dat de afgelegde afstand op het display getoond wordt, wil je namelijk dat het knipper-symbool even niet wordt afgebeeld, anders wordt de getoonde tekst onleesbaar.
ledX	**Number**	Het x-coördinaat van de led die je aan of uit wilt zetten.
ledY	**Number**	Het y-coördinaat van de led die je aan of uit wilt zetten, zie voor het gebruik van deze x en y coördinaten pagina 2 van de kaart.

-

Kaart 4 pag 2

Bij opstarten



Om te starten is het nodig dat we variabelen *aantalOmwentelingen* en *knipperDisplay* hun beginwaarden krijgen. Het *aantalOmwentelingen* begint bij ***0***, en *knipperDisplay* wordt op ***waar*** gezet. Tevens wordt het eerste symbool op het display gezet.

Typen variabelen

De MakeCode omgeving van micro:bit geeft de mogelijkheid om met 3 verschillende typen variabelen te werken:

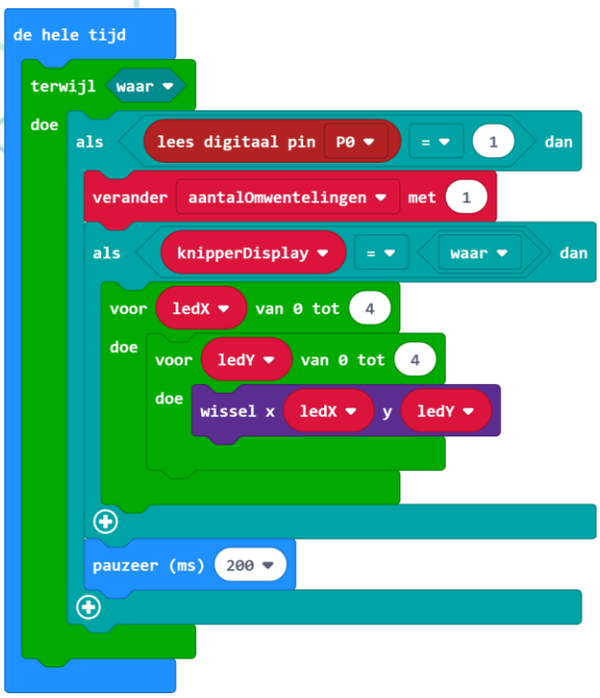
- een **Number**: een numerieke waarde (getal, positief of negatief)
- een **String**: tekst
- een **Boolean**: een variabele die alleen de waarden ***waar*** of ***onwaar*** kan aannemen

Zodra de variabele *aantalOmwentelingen* voor de allereerste keer een waarde krijgt, in dit geval in het *bij opstarten* blok, wordt bepaald welk type variabele hier bedoeld wordt. De variabele *aantalOmwentelingen* krijgt een getal als waarde. Vanaf dat moment weet de micro:bit dat deze variabele een numerieke waarde bevat en dus een **Number** is. Voor het gehele programma zal micro:bit deze variabele interpreteren als een **Number**. Het is vanaf dan niet meer mogelijk om de waarde 'waar' of een tekst aan deze variabele toe te wijzen. Voor de variabele *knipperDisplay* is dat anders. De eerste keer dat deze variabele een waarde krijgt, in dit geval ook in het *bij opstarten* blok, krijgt deze de waarde ***waar***. Vanaf dat moment interpreteert de micro:bit deze variabele als een **Boolean**. Het toewijzen van een getal of een letter of tekst is daarna niet meer mogelijk.

De programmalus

Je gaat in het programma het aantal keren tellen dat er een contact is geweest tussen pin ***P0*** en ***3V***. We noemen deze contactmomenten "pulsen". De puls wordt gemaakt door ***3V***

en ***P0*** met elkaar contact te laten maken zoals getoond wordt op pagina 4 en het wiel rond te draaien. Bij elke omwenteling wordt één puls geproduceerd.



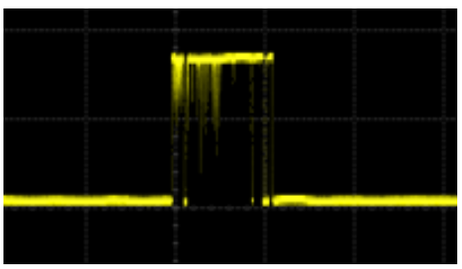
Het valt meteen op dat het programma een lus in een lus gebruikt. Het blok *de hele tijd* is een oneindig voortdurende lus, maar het blok *terwijl waar* ook. In die lus wordt o.a. het aantal omwentelingen geteld.

Als je alleen de lus *de hele tijd* gebruikt, zonder de *terwijl waar* lus erin, dan gaat de meting niet goed. De lus *de hele tijd* wordt door de micro:bit namelijk te langzaam uitgevoerd waardoor je pulsen zult missen en dus niet alle omwentelingen van het wiel zult tellen. De gemeten afstand zal dan kleiner zijn dan de werkelijke afstand.

De lus *terwijl waar* wordt veel sneller uitgevoerd door de micro:bit en zal dus geen pulsen/omwentelingen missen waardoor de meting overeenkomt met de werkelijk afgelegde afstand.
Bij elke puls (*lees digitaal pin P0*) wordt de variabele *aantalOmwentlingen* verhoogd met ‘1’.
Als de **Boolean** variabele *knipperDisplay* ***waar*** is, dan wordt bij elke omwenteling gewisseld van symbool op het display. Zou je voor het symbool het blok *toon lichtjes* of *toon pictogram* gebruiken dan staat dat pictogram ongeveer 0,6 seconden in beeld voordat het programma verder gaat. Je mist dan pulsen, dus omwentelingen. We doen het dus anders door alle 25 leds van waarde te laten veranderen. Dit is een actie die zeer snel zonder tijdvertraging door de micro:bit kan worden uitgevoerd. Met 2 lussen in elkaar wordt elke led die aan is uitgezet, en elke led die uit was wordt aangezet met het blok *wissel x y*. De 2 in elkaar gebouwde lussen zorgen ervoor dat dit voor elk coördinaat in de led matrix gebeurt, x van 0 tot en met 4, en ook y van 0 tot en met 4.
Als de **Boolean** variabele *knipperDisplay* ***onwaar*** is, zal er niet geknipperd worden met het symbool zodat het display beschikbaar is om iets anders af te beelden.

De lus *de hele tijd* is nodig om de lus *terwijl waar* op te starten en ook weer door te laten gaan wanneer de micro:bit even iets anders gedaan heeft, bijvoorbeeld de afstand op het display laten zien na het indrukken van ***knop A***.

Dendereffect



De pauze van 200 ms is ingebouwd om te voorkomen dat er te veel pulsen geteld worden. Het contact met de draadjes is niet ideaal en zal denderen, d.w.z. dat in een hele korte tijd het contact een aantal keren gemaakt en verbroken wordt. Zie de figuur. Dit is een normaal verschijnsel. Nu tellen we als de eerst puls gemeten wordt en wordt vervolgens 200 ms gewacht voordat er weer een puls gemeten kan worden. Met deze pauze kun je tot 5 pulsen per seconde meten. Dat komt overeen met een snelheid van meer dan 40 km/uur als je op een fiets rijdt met een standaard wielmaat van 28".

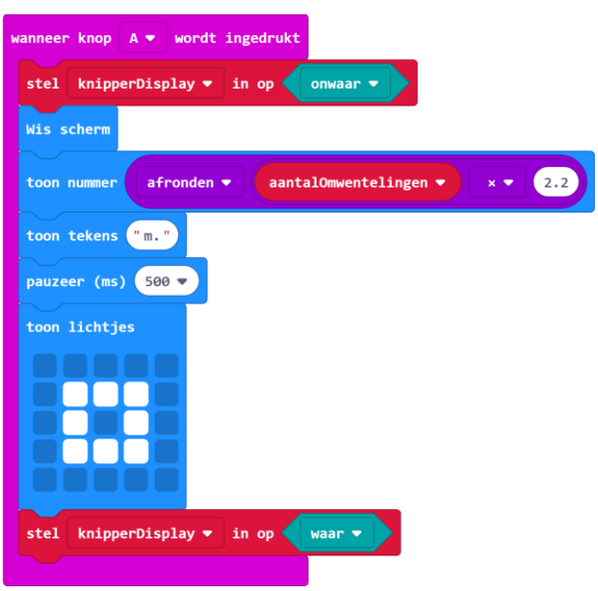
Kaart 4 pag 3

Ik snap het

De lus *terwijl waar* wordt behoorlijk snel uitgevoerd door de micro:bit, dus als het contact nog aan het 'denderen' is, nadat de lus 1 keer is uitgevoerd, dan wordt er nog een keer geteld en eventueel nog een keer etc. Zo kunnen er dus meerdere tellingen plaatsvinden bij een eenmalig contact van de spaak. Het lijkt daardoor of er een grotere afstand is afgelegd dan in de werkelijkheid. De pauze van 200ms voorkomt dit effect.

Knop A: laat de afstand zien

Als je ***knop A*** indrukt wordt de **Boolean** *knipperDisplay* op ***onwaar*** gezet. Hiermee stop je het wisselende symbool als indicatie voor het tellen. Het tellen loopt wel door.
Vervolgens wis je het scherm, en laat de afstand zien door het aantal omwentelingen te vermenigvuldigen met de omtrek van je wiel. Hier is die omtrek 2,2 meter. Als je een ander wiel hebt, kun je zelf een andere vermenigvuldigingsfactor invullen. Na een halve seconde wordt het beginsymbool weer getoond en wordt de variabele *knipperDisplay* weer op ***waar*** gezet zodat het symbool weer gewisseld wordt bij elke omwenteling.



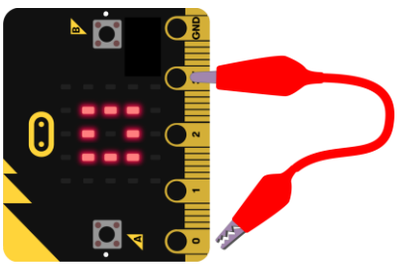
Knop A+B: zet de teller op 0

Door op ***knop A*** en ***knop B*** tegelijk te drukken wordt de teller *aantalOmwentlingen* op 0 gezet. Dus daarmee start je de afstandsmeting weer opnieuw vanaf de plek waar je op dat moment bent.

Er is bewust voor gekozen om dit niet te programmeren onder ***knop B***, omdat je dan te makkelijk per ongeluk de teller op 0 kan zetten. De knoppen A en B druk je niet makkelijk per ongeluk tegelijk in.

Testen

Door met een draadje pin ***P0*** te verbinden met ***3V*** kun je je programma testen nog voordat je de afstandmeter op je fiets hebt gemonteerd.



Kaart 4 pag 4

Maak het contact op de fiets

De foto's laten zien hoe je de afstandsmeter op je fiets monteert. Je maakt van een stukje ijzerdraad een kort spaakje parallel aan één van de spaken van je wiel. Draai het ijzerdraadje ook op naburige spaken heen zodat het ijzerdraadje niet rond de spaak kan ronddraaien.
Twee draadjes, die je bijvoorbeeld met een kroonsteentje en enkele bindbandjes kunt vastmaken aan je vork, maken contact met elkaar doordat ze bij elke omwenteling tegen het zelfgemaakte spaakje komen.

NB: Zorg ervoor dat de draadjes de gewone spaken niet kunnen raken.
Sluit de draadjes aan op pin ***P0*** en ***3V*** van de micro:bit. De micro:bit zelf kun je door middel van de aansluitstrip en 2 elastiekjes op je stuur binden, zodanig dat de batterijhouder ook klem zit op je stuur.

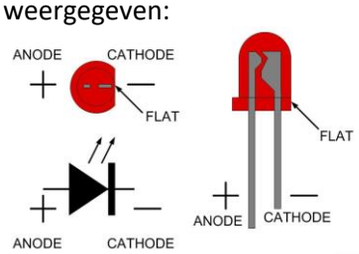
Maker 5: Spelen met licht

Over deze kaarten

Bij dit project ga je aan de slag met een RGB-led, een led die veel verschillende kleuren aan kan nemen. En je leert te programmeren hoe je die led de juiste kleur gaat geven.

Een led

Een led is een zogenaamde Light Emitting Diode, een licht uitstralende diode. Daarbij is een diode een elektronisch component die stroom maar in één richting doorlaat, een soort elektrisch ventiel. De stroom loopt van de anode (de plus) naar de kathode (de min), en een stroom in de andere richting wordt door de diode tegengehouden. Een led straalt licht uit als er een stroom loopt in de doorlaatrichting en wordt met het volgende symbool weergegeven:



Om aan te geven welke aansluiting de anode en welke de kathode is zijn 2 manieren bedacht, zie de afbeelding:

- Een afplatting van de behuizing aan de kathode kant
- Een langere aansluitdraad voor de anode

Een led kan worden aangestuurd door de kathode te verbinden met de ***GND*** pin en de anode aan te sluiten op een van de andere pinnen van de micro:bit. Je kan de led zowel digitaal als analoog aansturen:

- met *schrijf digitaal pin* kun je de aangesloten led aan (1) of uit (0) zetten
- met *schrijf analoog pin* kun je de led ook dimmen, m.a.w. de helderheid van de led bepalen door waarden in te vullen tussen 0 en 1023

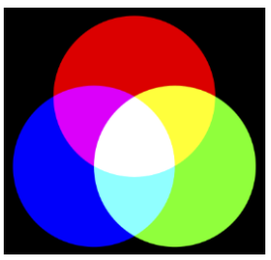
Ezelsbruggetjes

Er zijn twee bekende ezelsbruggetjes om te onthouden wat de positieve aansluiting (Anode) en wat de negatieve aansluiting (Kathode) van een led is

1. KNAP: **K**athode-**N**egatief, **K**athode-**K**ort, **A**node-**P**ositief
2. Stroom loopt van boven (lange draad) naar beneden (korte draad)

LET OP: Sluit een led nooit zomaar aan op een batterij of adapter want dan kan de led kapot gaan. In serie met de led zou dan een weerstand moeten worden opgenomen om de hoeveelheid stroom te beperken die door de led gaat lopen. Typisch wordt bij een voedingsspanning van 5 Volt een weerstand van ongeveer 220 Ohm gebruikt. Eigenlijk zou je een led ook niet rechtstreeks mogen aansluiten op een uitgangspin van de micro:bit (max. 3 Volt), maar omdat de pinnen van de micro:bit niet zo veel stroom kunnen leveren, gaat het in de praktijk (meestal) goed.

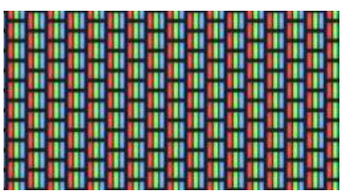
Een RGB led



Een RGB led bestaat uit 3 led's, één die rood licht geeft, één voor groen en één voor blauw licht. Door de verhouding van **R**ood, **G**roen en **B**lauw te regelen kun je verschillende kleuren maken. Zo levert een combinatie van rood en blauw een magenta resultaat. Combineer je rood met groen, dan levert dat geel op. Ook alle tussenliggende kleuren zijn mogelijk door een beetje meer of minder van een kleur aan te sturen.

LET OP: Dit is niet hetzelfde als het mengen van verf waarmee je ook mengkleuren kunt maken. Het verschilt omdat je met led's kleur maakt met licht en de kleuren bij elkaar optellen (additieve kleurmenging) terwijl je bij verf kleur produceert door kleur van het opvallende licht te absorberen (niet te weerkaatsen) en bij menging de kleuren van elkaar aftrekken (subtractieve kleurmenging. Voor meer informatie hierover zie WikiPedia.

Hier hebben we dus te maken met additieve menging. De afbeelding laat zien wat er gebeurt als je kleuren mengt.



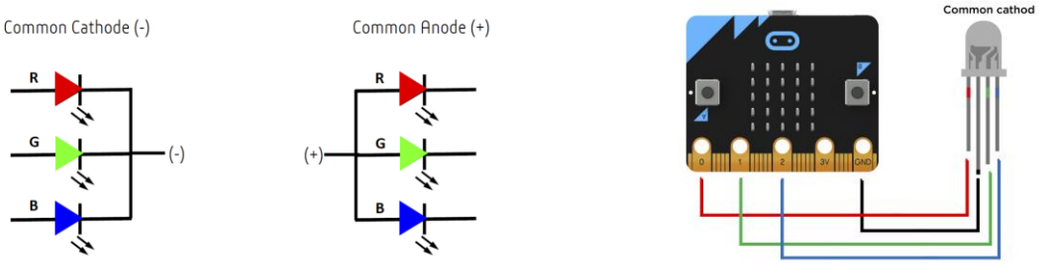
Ditzelfde principe wordt ook gebruikt bij een televisie. Als je met een vergrootglas kijkt naar het beeldscherm van een televisie, zie je allemaal pixels, elke pixel (punt op het scherm) is een RGB led.

Er zijn verschillende soorten RBG led's die we aan kunnen sluiten op de micro:bit:

- Gewone RGB-led met gemeenschappelijke kathode (min). Dit is de meest voorkomende.
- Gewone RGB-led met gemeenschappelijke anode (plus).
- Neopixels zijn speciale led's die achter ekaar worden geschakeld en worden aangestuurd via een speciale micro:bit-uitbreiding (extension) in MakeCode. *Dit valt buiten de scope van deze handleiding.*

Op deze kaart werken we met een common cathode led.

Common anode/cathode RGB led



In een common cathode led zijn 3 led's voor de verschillende kleuren gecombineerd in één behuizing, waarbij de 3 kathodes aan elkaar zijn verbonden. Zo'n led heeft dus 4 aansluitingen, zie de figuur. Als voorbeeld zie je in de figuur een aansluitschema voor het besturen van een common cathode led door een micro:bit. Dit is ook het type led waar we in dit project gebruik van maken.

Computational Thinking

In dit project worden de volgende aspecten geraakt uit het begrippenkader **Computational Thinking** en de leerlijn programmeren:

- ***Gegevens analyseren***: kleuren zien als opgebouwd uit 3 verschillende kleuren rood, groen en blauw waarbij elke kleur met een bepaalde intensiteit kan worden weergegeven.
- ***Gegevens visualiseren***: het mengen van de RGB waarden voor het verkrijgen van de juiste, gewenste kleur.
- ***Algoritmes***: de lus waarin per cyclus de intensiteit, helderheid van de led wordt aangepast en weergegeven, voorwaarden voor het gereguleerd optellen en aftrekken waarbij grenzen worden aangegeven: maximaal 1023 en minimaal 0
- ***Modellering***: het verschil tussen digitaal aansturen van een RGB mengkleur met 2^3=8 mogelijke kleuren, en het analoog aansturen van de RGB kleuren, met 1024^3=meer dan 1 miljard kleuren (theoretisch aantal: in de praktijk lijken veel van de mogelijke kleuren op elkaar waardoor je visueel minder mogelijkheden hebt).
- ***Parallellisatie***: het aansturen van 3 poorten (pin ***P0*** voor rood, ***P1*** voor groen en ***P2*** voor blauw) die gezamenlijk een resultaat (mengkleur) opleveren.

Kaart 4 pag 1

RGB-led

Korte uitleg van de RGB led.

RGB led aansluiten

De led heeft 4 aansluitdraden: de langste is de common cathode die aan ***GND*** van de micro:bit verbonden moet worden. Kijk in de figuur waar t.o.v. de cathode de 3 verschillende anodes zitten voor rood, groen en blauw en sluit aan volgens dit schema:.

RGB led	Kleur	micro:bit	Draad
GND		***GND***	Zwart
R	Rood	***P0***	Rood
G	Groen	***P1***	Groen
B	Blauw	***P2***	Blauw

Gebruik een kroonsteentje voor het verbinden van de led aan de micro:bit.

Uiteraard mag je ook andere kleuren draad gebruiken, maar het is het duidelijkst als de kleuren van de draden overeenkomen met de kleuren van de led. Het is een goed gebruik om zwart draad te gebruiken voor 0-draden (***GND***).

Kaart 4 pag 2

Het programma (digitaal)

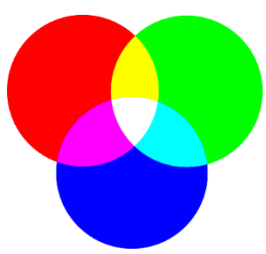
De code begint met het digitaal aansturen van één kleur; rood op ***P0***. Bij het digitaal aansturen met *schrijf digitaal pin* heb je alleen de mogelijkheid om de kleur aan (1) of uit (0) te zetten.

Kaart 4 pag 3

Kleuren mengen

Voor 3 kleuren geeft dit de volgende mogelijke mengkleuren:

R	G	B	Resulterende mengkleur
Rood	Groen	Blauw	
0	0	0	led is uit
1	0	0	Rood
0	1	0	Groen
1	1	0	Geel
0	0	1	Blauw
1	0	1	Magenta
0	1	1	Cyaan
1	1	1	Wit



Helderheid regelen (analoog programma)

Door de pinnen analoog aan te sturen heb je de mogelijkheid om de helderheid van een kleur te regelen. Elke afzonderlijke kleur (rood, groen of blauw) kan ingesteld worden op een waarde tussen de 0 en 1023. In totaal geeft dit theoretisch 1024^3=meer dan een miljard mogelijke mengkleuren. Omdat in de praktijk veel van die kleuren op elkaar lijken, zijn er visueel minder mogelijkheden, maar nog steeds heel veel.

Er wordt gewerkt met twee variabelen:

- *helderheid* geeft de waarde aan tussen 0 en 1023 waarmee de pin wordt aangestuurd
- *stapjes* geeft aan met welke sprongen je de helderheid laat toenemen en afnemen, het heeft namelijk niet zoveel zin om voor elke kleur alle

1024 mogelijkheden te laten zien, de kleurverandering gaat dan te langzaam, bij stapjes van 100 loop je in grofweg 10 sprongen door de helderheden heen. Je kunt naar eigen inzicht de stapgrootte aanpassen.

Kaart 4 pag 4

De voorwaarden *als helderheid > 1023* en *als helderheid < 0* zorgen ervoor dat de waarde van *helderheid* niet buiten zijn grenzen komt, en tevens dat de richting waarin door de helderheid gelopen wordt omdraait:

- wordt de *helderheid* groter dan 1023, dan gaan we vanaf nu met stappen naar beneden
- wordt de *helderheid* kleiner dan 0, dan gaan we vanaf nu met stappen omhoog

Maker 6: De quiz

Bij een quiz wil je graag weten wie het eerst op de knop heeft gedrukt. In dit project gaan we met micro:bit drukknoppen maken die meten welke als eerste is ingedrukt, welke als tweede etc. Het maakt niet uit hoeveel knoppen je wilt gebruiken. Elke knop is verbonden met zijn eigen micro:bit, dus wil je 7 knoppen voor 7 quiz-kandidaten, dan heb je 8 micro:bits nodig. Ook één voor de quiz-master namelijk.
We gebruiken in deze les de functionaliteiten uit het menu 'radio'. Dit heeft niets te maken met de echte radio, maar wel met draadloze communicatie tussen micro:bits onderling. Dit maakt het mogelijk om meerdere micro:bits te laten samenwerken in één systeem, zonder dat ze met draden aan elkaar hoeven worden aangesloten. Bovendien leren we zo de radiofunctionaliteit kennen waarmee nog heel veel andere gave projecten mogelijk zijn.

Computational Thinking

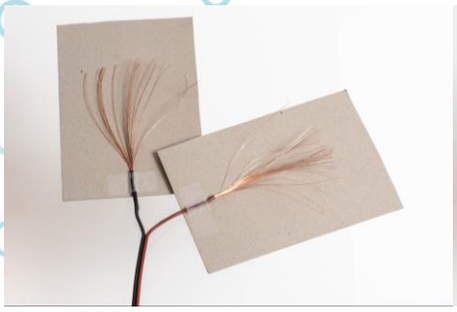
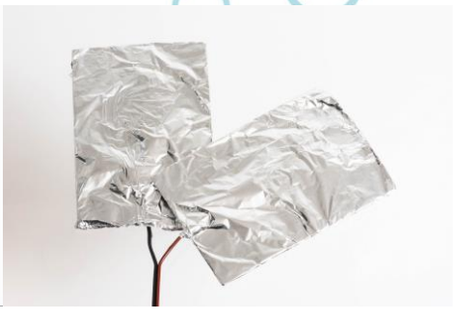
In deze les worden de volgende aspecten geraakt uit het begrippenkader **Computational Thinking** en de leerlijn programmeren:

- ***Gegevens verzamelen***: het meten van het indrukken van de knop en het bijhouden van een 'gemeenschappelijk' getal *mijnPlaats*, zolang je zelf nog niet gedrukt hebt.
- ***Gegevens analyseren***: bijhouden en controleren of je zelf al gedrukt hebt.
- ***Gegevens visualiseren***: als hoeveelste je gedrukt hebt, wordt in het display weergegeven.
- ***Probleem decompositie***: het programma bevat delen voor het zelf bijhouden van het drukken, het bijhouden van andere micro:bits waarop gedrukt wordt, en het resetten door de quiz-master.
- ***Abstractie***: Meerdere micro:bits vormen bij elkaar en samenwerkend systeem, waarbij het proces in elke micro:bit synchroon loopt, totdat er op de eigen micro:bit gedrukt wordt. Dan gaat he proces op de eigen mico:bit een andere weg, echter wordt in het systeem van meerdere micro:bits toch een gezamenlijk resultaat bereikt. Het verschil tussen systeemgedrag en systeemresultaat en de eigen rol van een individueel element vormen meerder abstractielagen.
- ***Automatisering***: acties op één micro:bit heeft automatische gevolgen bij (meerdere) andere micro:bits
- ***Parallellisatie***: de lus *de hele tijd* en gebeurtenis *wanneer de radio ontvangt* lopen parallel aan elkaar.

Kaart 6 pag 1

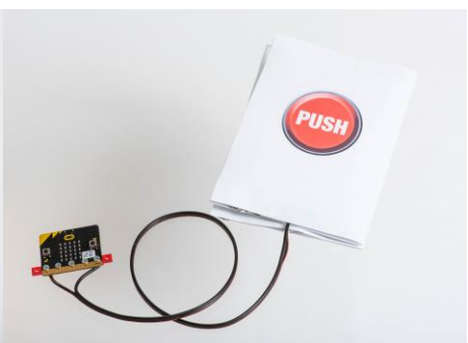
De knoppen

Allereerst hebben we knoppen nodig waarop we hard kunnen slaan, want in de hitte van de quiz-strijd zal dat zeker gebeuren. Je kunt natuurlijk officiële knoppen kopen en die op de micro:bit aansluiten, maar die zijn vaak kostbaar. Dus we gaan de knop zelf maken van goedkope materialen.
De foto's laten zien hoe je draad gebruikt dat goed contact moet maken met een stuk aluminiumfolie. Wikkel e.e.a. om een stuk karton, doe dit 2 maal en je hebt de basiselementen van een knop.

Kaart 6 pag 2

Door de knopdelen in te pakken in papier met daaruit geknipte gaten maak je het mogelijk om ze op elkaar te leggen zonder dat ze elektrisch contact maken. Wanneer je er dan op drukt zullen de aluminium folies op de plek van de uitgeknipte gaten elkaar raken en is er elektrisch contact dat door de micro:bit kan worden gemeten. Sluit één knopdeel aan op input ***P0*** en de andere op ***GND***. We gebruiken hiervoor een aansluitstrip met boutjes en moertjes.

Controleer of de knop werkt

Met dit programmaatje kan je de knop testen.

Kaart 6 pag 3

Programma bouwen

Om micro:bits via de radiofunctie met elkaar te laten communiceren moeten ze zijn ingesteld op dezelfde groep. Een micro:bit kan namelijk alleen informatie zenden en ontvangen in de groep waarin de micro:bit zich bevindt. Het groepsnummer (van 0 tot 255) wordt ingesteld met het blok *Radio instellen groep*. Het is hiermee mogelijk om meerdere onafhankelijke systemen te bouwen die elk bestaat uit

meerdere micro:bits, die wel kunnen communiceren binnen de eigen groep, maar geen last hebben van de communicatie in een andere groep. In het algemeen wordt de radio groep gedefinieerd in het *bij opstarten* blok, maar dat hoeft niet. Het is ook mogelijk om van groep te wisselen terwijl het programma wordt uitgevoerd. Dat doen we hier echter niet.

We definiëren in het *bij opstarten* blok ook twee variabelen *geblokkeerd* en *mijnPlaats* van verschillende typen.

Typen variabelen

De MakeCode omgeving van micro:bit geeft de mogelijkheid om met 3 verschillende typen variabelen te werken:

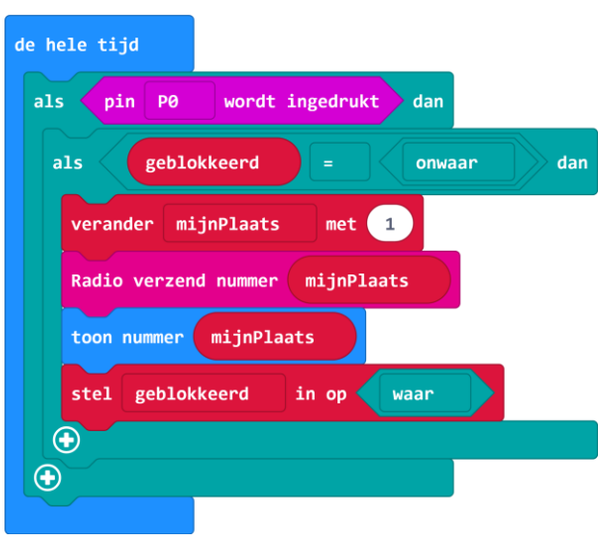
- een **Number**: een numerieke waarde (getal, positief of negatief)
- een **String**: een regel tekst (een letter is een ook tekst van slechts één letter)
- een **Boolean**: een variabele die alleen de waarden ***waar*** of ***onwaar*** aan kan nemen

De eerste keer dat een variabele een waarde krijgt, in dit geval in het *bij opstarten* blok, wordt bepaald welk type variabele hier bedoeld wordt. De variabele *geblokkeerd* krijgt als waarde ***onwaar***. Hiermee is bepaald dat het om een **Boolean** gaat. Vanaf dat moment is *geblokkeerd* in het gehele programma een **Boolean**. Het toewijzen van een getal of een letter of tekst is daarna niet meer mogelijk.
De variabele *mijnPlaats* krijgt de waarde ***0*** waarmee is bepaald dat *mijnPlaats* (voor het hele programma) een **Number** is.

De variabelen worden in het programma steeds aangepast en hebben de volgende functie:

Naam	Type	Functie
mijnPlaats	**Number**	Uiteindelijk bevat deze variabele een waarde die aangeeft als hoeveelste jij hebt gedrukt.
geblokkeerd	**Boolean**	Geeft aan of jouw knop al een keer is ingedrukt, want daarna moet het opnieuw drukken van anderen geen effect meer hebben op de verhoging van jouw variabele *mijnPlaats*.

In de *de hele tijd* lus wordt bij herhaling bepaald of de knop wordt ingedrukt door het ingangssignaal van pin ***P0*** te meten. Zodra dit het geval is wordt gecontroleerd of je al eerder gedrukt hebt. Dat is te zien aan de variabele *geblokkeerd*. Als deze ***onwaar*** is, had je nog niet eerder gedrukt. Als deze ***waar*** is, had je kennelijk al gedrukt en gebeurt er niets. Druk je voor het eerst dan verhoogt meteen de teller *mijnPlaats* die bijhoudt hoeveel mensen er al gedrukt hebben, en verzendt dit getal draadloos naar alle andere quiz-knoppen, zodat die ook weten dat er door iemand gedrukt is.



Dit gebeurt met *radio verzend nummer mijnPlaats*. Het nummer dat in *mijnPlaats* is opgeslagen wordt op het scherm getoond om aan te geven als hoeveelste jij hebt gedrukt. De **Boolean** *geblokkeerd* wordt op ***waar*** gezet, zodat vaker drukken geen effect meer zal hebben.

Voor sommigen is het lastig te begrijpen dat micro:bits hier met elkaar samenwerken. De variabele *mijnPlaats* houdt bij hoeveel mensen er gedrukt hebben. Die variabele wordt in elke micro:bit apart bijgehouden, maar is dus in die zin geen gemeenschappelijke variabele/teller die door elke micro:bit bekeken kan worden. Om alle tellers in de verschillende micro:bits toch gelijk te laten lopen, wordt de radio functie gebruikt om alle andere micro:bits te informeren dat er door iemand gedrukt is, en dat dus de teller *mijnPlaats* met 1 moet worden verhoogd.

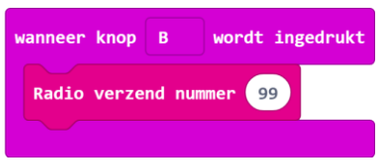
Als je zelf gedrukt hebt wordt de teller nog een laatste maal verhoogd, maar blijft verder hangen op die waarde omdat de variabele *geblokkeerd* op ***waar*** wordt gezet en daarmee voorkomt dat *mijnPlaats* bij de eigen micro:bit nog verder wordt verhoogd. Bij de andere micro:bits die zelf nog niet gedrukt hebben loopt het ophogen wel door. Zo krijgt iedere micro:bit uiteindelijk een eigen waarde van *mijnPlaats*.

Als een andere micro:bit op de knop drukt wordt dat dus het getal *mijnPlaats* draadloos uitgezonden. Als je zo'n getal ontvangt (en je had zelf nog niet gedrukt) dan moet je de waarde van de teller *mijnPlaats* van die andere micro:bit overnemen, om met zijn allen gelijk te lopen in de telling. Dit gebeurt met het blok *wanneer de radio ontvangt receivedNumber*.

Als je het nummer 99 ontvangt dan wordt de hele micro:bit gereset, als het ware opnieuw opgestart, zodat het hele proces weer van voren af aan begint. De quiz-master heeft een micro:bit waarmee je na elke quiz-vraag het getal 99 kunt verzenden, en reset zo alle deelnemers.

Kaart 6 pag 4

De quizmaster hoeft zelf niet te drukken om een antwoord te geven op een gestelde vraag, maar heeft een micro:bit met een ander programma. Dit heeft als enige doel het uitzenden van het getal 99 om alle deelnemende micro:bits weer te resetten om ze klaar te zetten voor de volgende vraag.



Maker 7: Pratende plant

De plant gaat je in dit project waarschuwen wanneer deze water nodig heeft. Met de micro:bit wordt bepaald hoe vochtig de grond is. En als deze te droog wordt dan zal de micro:bit dat aangeven door middel van een servo-motor die een zelfgemaakte pijl omhoog zal steken als een sein.
Dit project wordt in verschillende stappen geperfectioneerd daarbij rekening houdend met het energiegebruik van de metingen.

Computational Thinking

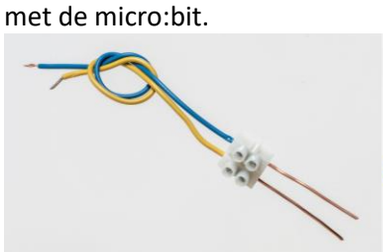
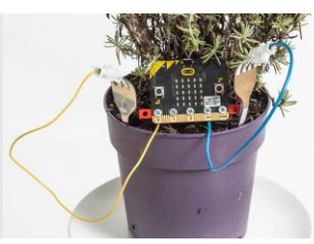
In deze les worden de volgende aspecten geraakt uit het begrippenkader **Computational Thinking** en de leerlijn programmeren:

- ***Gegevens verzamelen***: meting van de spanning op ***P0*** als maat voor de vochtigheid.
- ***Gegevens analyseren***: het proefondervindelijk bepalen van een geschikte drempelwaarde.
- ***Gegevens visualiseren***: het afbeelden van een waarde op een grafisch staafdiagram (bar graph) en het aangeven van een berekening door middel van een stand van de servo
- ***Probleem decompositie***: meting opdelen in aanbrengen van de voorwaarde voor de meting, meting doen, wegnemen van de voorwaarden ten behoeve van het energiegebruik, en het aangeven van de uitkomst van de meting.
- ***Abstractie***: het energiegebruik meenemen in de overwegingen om de meetstrategie te bepalen.
- ***Automatisering***: repetitie in de meting

Kaart 7 pag 1

De sensor

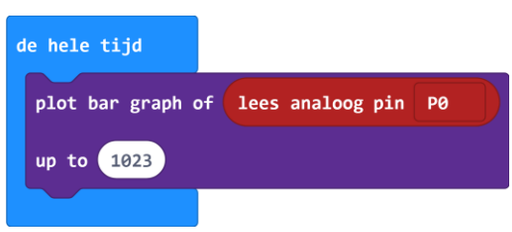
De vochtmeting gebeurt eigenlijk heel eenvoudig met behulp van twee geleiders die in de grond naast de plant gestoken worden. Het maakt daarbij niet uit of met 2 ijzerdraadjes gewerkt wordt, zoals in de eerste foto, of bijvoorbeeld met gewone vorken die met krokodillensnoertjes, of in dit project met kroonsteentjes en draden verbonden worden met de micro:bit.


De twee geleiders worden verbonden met de pinnen ***3V*** en ***P0*** van de micro:bit. In dit project wordt hiervoor de aansluitstrip met boutjes en moertjes gebruikt.

Kaart 7 pag 2

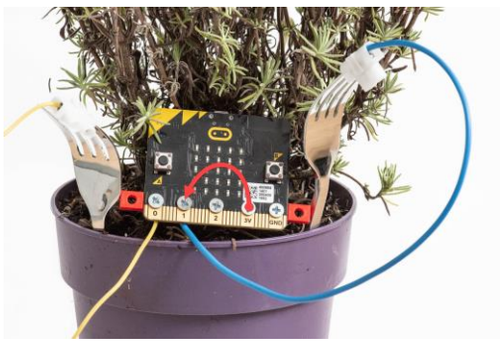
Het programma



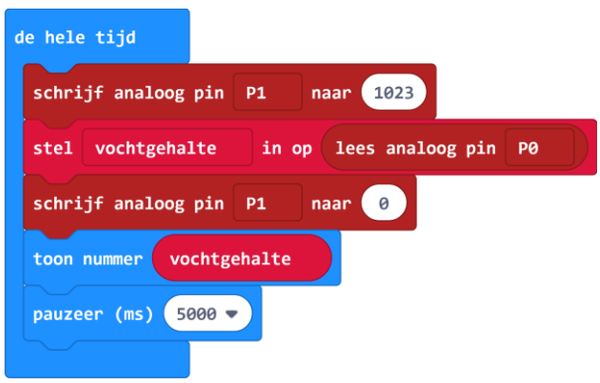
Het eerste programma is heel eenvoudig. De pin ***P0*** wordt analoog uitgelezen en geeft een waarde tussen de 0 en 1023 afhankelijk van de spanning die gemeten wordt op pin ***P0***. Deze spanning wordt veroorzaakt doordat de andere geleider is aangesloten op een spanning van 3 Volt. De hoogte van de gemeten spanning op ***P0*** is afhankelijk van de mate

van geleiding van de vochtige grond tussen de twee geleiders. Vochtige grond geleidt beter dan droge grond en zo heb je dus een maat voor de vochtigheid van de grond.
Het blokje *plot bar graph of* laat een grafische representatie zien van de waarde tot aan een maximale waarde van 1023.

Energiezuinig programma



Het bezwaar met het eenvoudige programma is dat er op één van de geleiders continu 3 Volt staat, waardoor er ook continu een stroom loopt ten behoeve van de meting. Maar het heeft in dit geval geen zin om continu te meten. De vochtigheid van de grond zal namelijk maar zeer langzaam minder worden, dus een meting hoeft eigenlijk maar af en toe te gebeuren. Het idee daarbij is dat de geleider, die normaal op ***3V*** aangesloten is, nu op een output ***P1*** wordt aangesloten zodat we met de micro:bit kunnen besturen wanneer en hoe lang er 3 Volt op de geleider wordt aangeboden. Dus poort ***P1*** naar 3 Volt, snel de meting doen van ***P0*** en dan meteen daarna ***P1*** weer naar 0 Volt. Als je dit proces elke 5 seconden herhaalt, weet je ook de vochtigheid, en gebruik je maar een fractie van de energie. In het geval dat deze opstelling met een batterij wordt gevoed, zul je zien dat de batterij heel veel langer mee gaat.

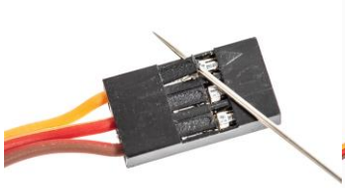


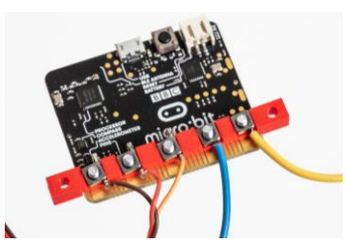
Het vochtgehalte wordt opgeslagen in de numerieke variabele *vochtgehalte*.

Kaart 7 pag 3

Toon bordje water

We gaan een servomotor gebruiken om met een bewegende pijl aan te geven of de plant water nodig heeft. Op de foto's is te zien hoe de geleiders uit de connector gehaald kunnen worden zodat ze via de aansluitstrip op de micro:bit kunnen worden aangesloten. De rode draad van de servo wordt aangesloten op ***3V*** en de bruine op ***GND***. Sluit de oranje draad aan op ***P2***. Daarmee is de servomotor te besturen.





Kaart 7 pag 4

Maak van papier een pijl of vlag aan de bewegende arm van de servo vast. Dit kan uiteraard naar eigen inzicht. In de foto is een pijl gebruikt, en is de servo vastgemaakt aan een houten prikker die ook in de aarde van de plant wordt gestoken. Als je hierover andere ideeën hebt is dat uiteraard ook goed.

De waarde van de variabele *vochtgehalte* wordt nu gebruikt om de servomotor op pin ***P2*** te laten draaien als de waarde boven een bepaalde drempel komt. In het voorbeeld is de grens gelegd op 1000, maar het is verstandig even in de praktijk te bepalen waar de drempelwaarde het best kan liggen. De gemeten waarde is van meerdere dingen afhankelijk, de soort aarde, de afstand tussen de geleiders, de mate van geleiding van de geleiders, en natuurlijk de vochtigheid die we willen meten. Stel naar eigen inzicht de meest zinnige drempelwaarde in.

Bijlage 1: Kaarten afdrukken

De kaarten hebben een langwerpige vorm. Hierdoor kunnen ze ook op de (relatief) smalle Nederlandse schooltafeltjes naast de laptop of toetsenbord worden gelegd.

1. Druk de pdf in kleur af op wit papier. Kies er bij de printeropties voor dat het document op ware grootte wordt afgedrukt.

2. Snijd rondom 5 mm af
3. Vouw het papier dubbel
4. Lamineer de kaart. Gebruik als het even kan geen glanzend, maar mat lamineerplastic.

Als je meerdere kaarten moet afdrukken dan kan je ze dubbelzijdig afdrukken en na het afsnijden van de 5 mm doormidden snijden.

Ontwerp Canvas - Basis

μ ασνδαγ 30 απριλ2018 22:06

Wat ga je maken?

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

Teken je Ontwerp.

Welke sensoren/ input gebruik je?

- □ Κνοππεν (Α, Β οφ ΑΒ)
- □ Δρααιεν
- □ ςερβινδεν
- □ Τεμ περατυυρ
- □ Λιχητ

Wat heb je nog meer nodig?

Καρτον, Σχηααρ, Τατε, Λιμ, Σπιδεν, Οογεσ, Ηουτ, Καβελσ, Γερεεδσχηαπ.
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□...

Bouw je prototype.

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

Test je ontwerp, wat kun je verbeteren aan je ontwerp.

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

Wat heb je geleerd van het bouwen van dit prototype?

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

Basis

Ontwerp Canvas - Vervolg

μ ααν δαγ 30 απριλ2018 22:06

Wat ga je maken?

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

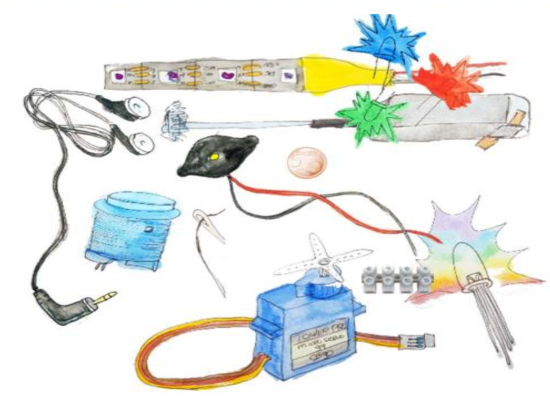
Teken je Ontwerp.

Welke sensoren/ input gebruik je?

- □ Κνοππεν (Α, Β οφ ΑΒ)
- □ Σηακε
- □ ςερβινδεν
- □ Τεμ περατυυρ
- □ Λιχητ
- □ ςερβινδεν
- □ □□□□□..

Wat heb je nog meer nodig?

Καρτον, Σχηααρ, Ταπε, Λιφι , Στιφτεν, Οογφεσ, Ηουτ, Καβελσ, Γερεεδσχηαπ.
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□...



Bouw je prototype.

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

Test je ontwerp, wat kun je verbeteren aan je ontwerp.

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

Wat heb je geleerd van het bouwen van dit prototype?

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

Ontwerp Canvas - Expert

μ ονδαy 30 απριλ2018 22:06

Wat ga je maken?

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.
□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

Teken je Ontwerp.

Welke sensoren/input gebruik je?

- □ Κνοππεν (Α, ΒοφΑΒ)
- □ Σηακε
- □ ςερβινδεν
- □ Τεμ περατυυρ
- □ Λιχητ
- □ ςερβινδεν
- □ □□□□□..

Wat heb je nog meer nodig?

Καρτον, Σχηααρ, Τοπε, Λιμ, Σπιθτεν, Οογαε, Ηουτ, Καβελα, Γερεεδσχηαπ.
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Welke programmeer taal?

- □ ΜακεΧοδε
- □ Σχρατχη
- □ Μιχρο:Πψτηον
- □ ϑαυαΣχριπτ

Test je ontwerp, wat kun je verbeteren aan je ontwerp.

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

Bouw je prototype.

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

Wat heb je geleerd van het bouwen van dit prototype?

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□

MICROBIT 101

Expert

lex.van.gijsel@devlab.nl

chris@codekids.nl