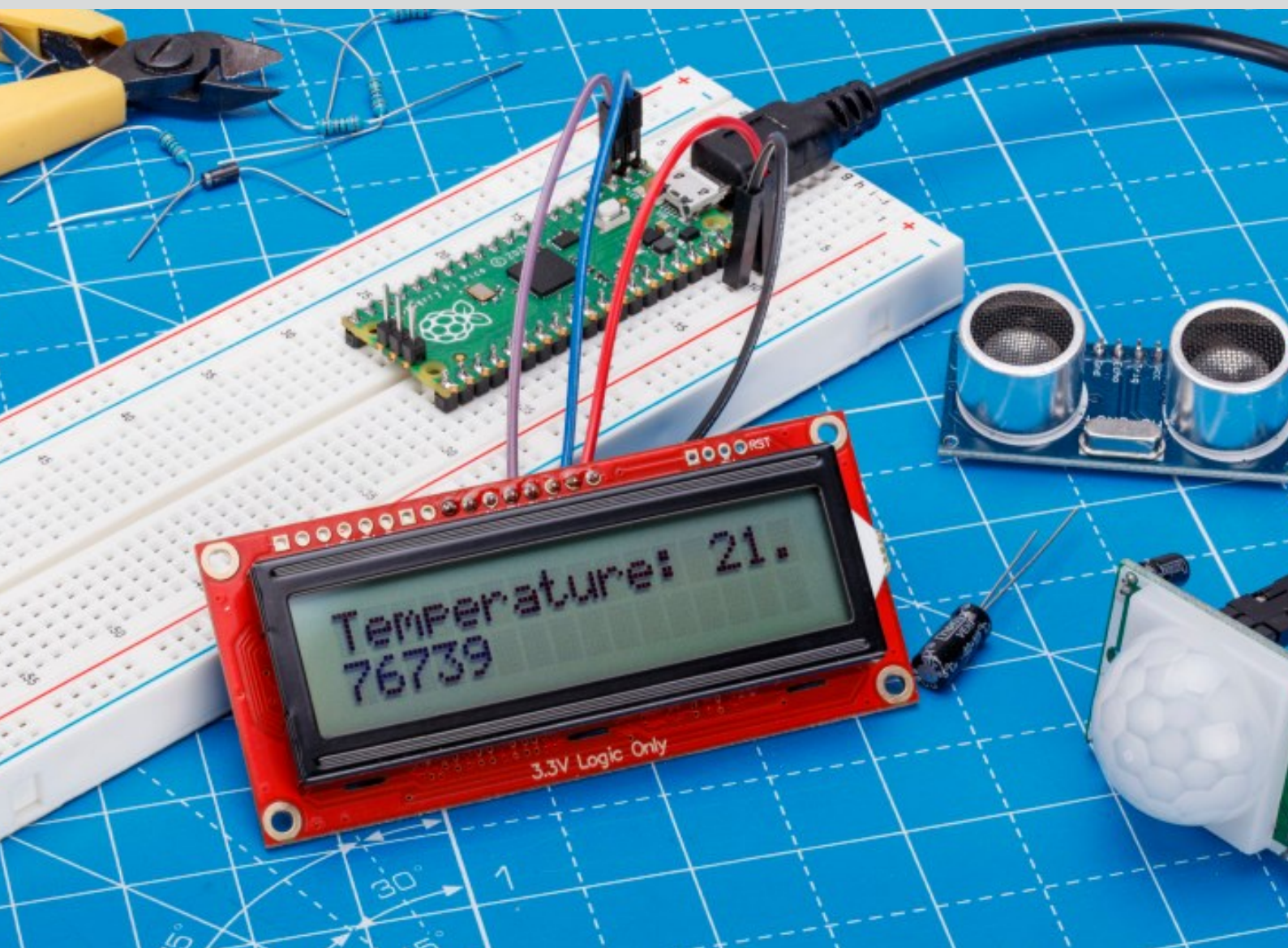


Bouw je eerste mini-satelliet

Metingen op een onbemande vlucht met
behulp van Raspberry Pi Pico
Lerarengids



- Bouw voor de eerste keer een elektronisch aangestuurd experiment ●●●●●
- Gebruik van een microcomputer van Raspberry Pi voor absolute beginners ●●●●
- Meten van luchtdruk, temperatuur en hoogte tijdens een onbemande vlucht ●●●●
- Opent de deur om deel te nemen aan uitdagende STEM projecten van ESERO ●●

OVER ESERO BELGIUM

ESERO is een scholenprogramma van de Europese Ruimtevaartorganisatie ESA. Het doel van dit programma is leraren van basisonderwijs en middelbaar onderwijs helpen om het populaire thema ruimtevaart in de klas te brengen, binnen hun lesopdracht. Dit doen we op drie manieren: **lesmateriaal** (online), **lerarenavormingen**, en **STEM projecten voor scholen**. Het aanbod is volledig gratis voor leraren in beroep en leraren in opleiding, en is afgestemd op de eindtermen in het onderwijs. Hedendaagse en inspirerende ruimtevaartmissies vormen de context diverse schoolvakken.

WWW.ESERO.BE

Nationale coördinator






Vlaamse coördinator




UGENT
VOLKSSTERRENWACHT
ARMAND PIEN

Frans- en Duitstalige coördinator

ULB
La Scientothèque

ESA Education beheert en coördineert alle ESERO's in Europa. Elke ESERO bestaat dankzij een cofinanciering van ESA en nationale partners. Het federaal wetenschapsbeleid (BELSPO) is de cofinancierende partner voor ESERO Belgium.



Mijn eerste mini-satelliet

Metingen tijdens een vlucht – voor absolute Pico beginners

Cursus kenmerken

Leeftijd doelgroep 14-18 jaar

Type Informatieve cursus met oefeningen en instructies

Vereist gebruik van

- Arduino Pico microcontroller en accessoires
- Luchtdruksensor en temperatuursensor
- Soldeerstation
- LED's en weerstanden
- Micro SD kaart bord

Eindtermen Diverse STEM eindtermen
(ideaal voor projectonderwijs, vrije ruimte, ...)

Samenvatting Hoe kan je metingen uitvoeren op een onbemande vlucht (drones, miniatuurvliegtuigen, weerballon, raket, ..)? Hiervoor heb je basis elektronica nodig. In deze cursus leren absolute beginners om de luchtdruk en de temperatuur te meten via een Raspberry Pi Pico microcomputer. Je leert eenvoudige basisvaardigheden zoals solderen, bedienen van een LED, communiceren met de microcontroller (via laptop), aanpassen van de programmeercode, en opslaan van gemeten data. Deze vaardigheden vormen een basis om deel te nemen aan STEM projecten zoals bijvoorbeeld [CanSat](#) van [ESA](#) en ASGARD Balloons for science. Hierin worden leerlingen uitgedaagd om een zelfbedacht experiment te lanceren.

Colofon

Eerste uitgave Oktober 2017 (met Arduino microcontroller)

Updates 2022, 2023, 2024

Gebruik en beschikbaarheid

- Dit cursusmateriaal mag gratis gebruikt worden voor niet-commerciële, educatieve doeleinden. Wie fragmenten eruit overneemt, dient de bron te vermelden.
- Lesmateriaal download op www.esero.be > Nederlandstalig > lesmateriaal.

AUTEURS

ESERO Belgium

- Inhoud van de originele Arduino cursus en layout
- Updates
- Vormingen voor leraren: organisatie
- Medewerker: Pieter Mestdagh

De Creatieve STEM

- Aanpassing naar Raspberry Pi Pico, 2023
- Aanpassingen inhoud en layout
- Lesgever van de vormingen 2023 en 2024

Ingegno (Drongen)

- Raspberry Pi Pico kits voor lesgevers en leerlingen

Uw mening is belangrijk

Cursussen van ESERO Belgium worden online aangeboden in dynamische vorm. Dat betekent dat elke zinvolle feedback van gebruikers leidt tot de publicatie van een aangepaste uitgave op www.esero.be (Nederlandstalig). Help toekomstige gebruikers door uw opmerkingen of aanvullingen per email op te sturen (www.esero.be > NL > contact). Gebruikers die aanzienlijke nieuwe onderdelen toevoegen aan de cursus, worden hierboven in de auteurslijst vermeld.

Inhoud

CURSUS KENMERKEN	1
COLOFON	2
INHOUD	3
① BOUW JE EERSTE SATELLIET	3
1A A SPACE MISSION AT SCHOOL	3
1B ONDERDELEN VAN EEN SATELLIET (PAYLOAD, BUS)	8
② PAYLOAD	9
2a Perifere onderdelen van de payload	9
<i>TEMPERATUUR SENSOR</i>	9
De NTC thermistor	10
De TMP36 sensor	10
DS18B20 Temperatuur Sensor	12
<i>COMMUNICATIE MET SENSOREN VIA I2C OF SPI</i>	13
I2C bus	14
SPI bus	15
<i>DRUKSENSOR</i>	16
BMP280: Inleiding	16
BMP280: Beschikbare pins	16
Werken met de BMP280	18
BME280 druksensor	19
Als je zelf een druksensor kiest	19
<i>μPROCESSOR OF μCONTROLLER</i>	20
<i>BATTERIJ/POWER BANK</i>	20
Energie voor satellieten	20
Welke batterij gebruiken we?	21
<i>WEERSTANDEN</i>	22
Weerstand?	22
Gebruik	22
Kleurcodes	23
<i>KABELS</i>	23
USB verbinding voor Raspberry Pi Pico	23
Gekleurde kabelset of jumper kabels	24
<i>LEDs</i>	24
Wat is een LED?	24
Een LED in het circuit opnemen	24
<i>POTENTIOMETER</i>	25
<i>SCHAKELAAR</i>	25
2B RASPBERRY PI PICO	26
<i>TERMINOLOGIE</i>	26
<i>HARDWARE Raspberry Pi Pico</i>	28
3 analoge input pinnen	30

26 digitale pinnen	30
3,3 V en 5 V input en output pinnen	30
Ground pinnen	31
Sinken en sourcen	31
<i>EXTENSIE BORDEN – SD CARD BORD</i>	32
SD Card shield	32
LiPo Batterij bord	32
RTC-bord	32
<i>STROOM CONTROLEREN</i>	33
De stroomsterkte begrenzen	33
Hoe bereken je de voorschakelweerstand?	34
Weerstand combineren	37
Pull up weerstand	37
Pull down weerstand	38
3 Software	39
3a Thonny	39
Download	39
De interface	39
Code Editor en Shell	39
Menubar	39
Toolbar	39
Code venster	39
Python Cheat Sheet	41
De Python Programmeertaal	41
<i>INSTALLLEER MICROPYTHON OP DE PICO</i>	43
Pico verbinding stoppen - starten	43
Seriële output - REPL interface (Shell)	43
3b Programmeren van de Pico	44
<i>EENVOUDIGE OEFENINGEN : Interne LED laten knipperen</i>	45
Opdracht	45
Hardware	45
Code 1: LED aansturen	45
Code opslaan en uitvoeren op de Pico	47
<i>EENVOUDIGE OEFENINGEN : Een LED laten knipperen</i>	47
Opdracht	47
Hardware	48
Python Script Oplossing 1: programma onderbreken	48
Upload	48
Oefening: Verander het knipperpatroon.	48
Python Script Oplossing 2: een Timer	48
Upload	49
<i>EENVOUDIGE OEFENINGEN : Interne temperatuur meten</i>	49
Opdracht	49
Hardware	50
Python Script Oplossing	51

Upload	51
Oefening	52
<i>EENVOUDIGE OEFENINGEN : Verkeerslicht</i>	52
Opdracht 1	52
Hardware	52
Circuit	52
Python Script	52
Opdracht 2	52
Hardware	52
Python script	52
<i>TEMPERATUUR METEN</i>	52
Opdracht	52
Hardware	52
Circuit en script	53
Meerdere temperatuursensoren	55
<i>LUCHTDRIJK METEN</i>	55
Opdracht	55
Hardware	56
Circuit	56
Library bmp280	57
Script	57
Hoogte meten	58
Oefening	62
<i>DATA OPSLAAN OP SD KAART</i>	62
SD-kaart module	62
Circuit	62
sdcard bibliotheek	62
Script: Opslaan van BMP280 meetgegevens op sd-kaart	64
Sensordata analyseren	68
Opdrachten	68
<i>DATA VERZENDEN/ONTVANGEN MET RADIO-COMMUNICATIE: Beperkte inleiding</i>	68
Space link	69
Radiogolven en frequenties	69
Hoe kan een elektromagnetische golf informatie bevatten?	70
Modulatie	73
Metadata	74
Radiocommunicatie met de Pico	74
4 De satelliet klaarmaken	74
<i>SOLDEREN</i>	75
Waarom Solderen?	75
Veiligheid	75
Soldeermetaal	76
Soldeerstation	76
Accessoires	77
Vorbereiding	77

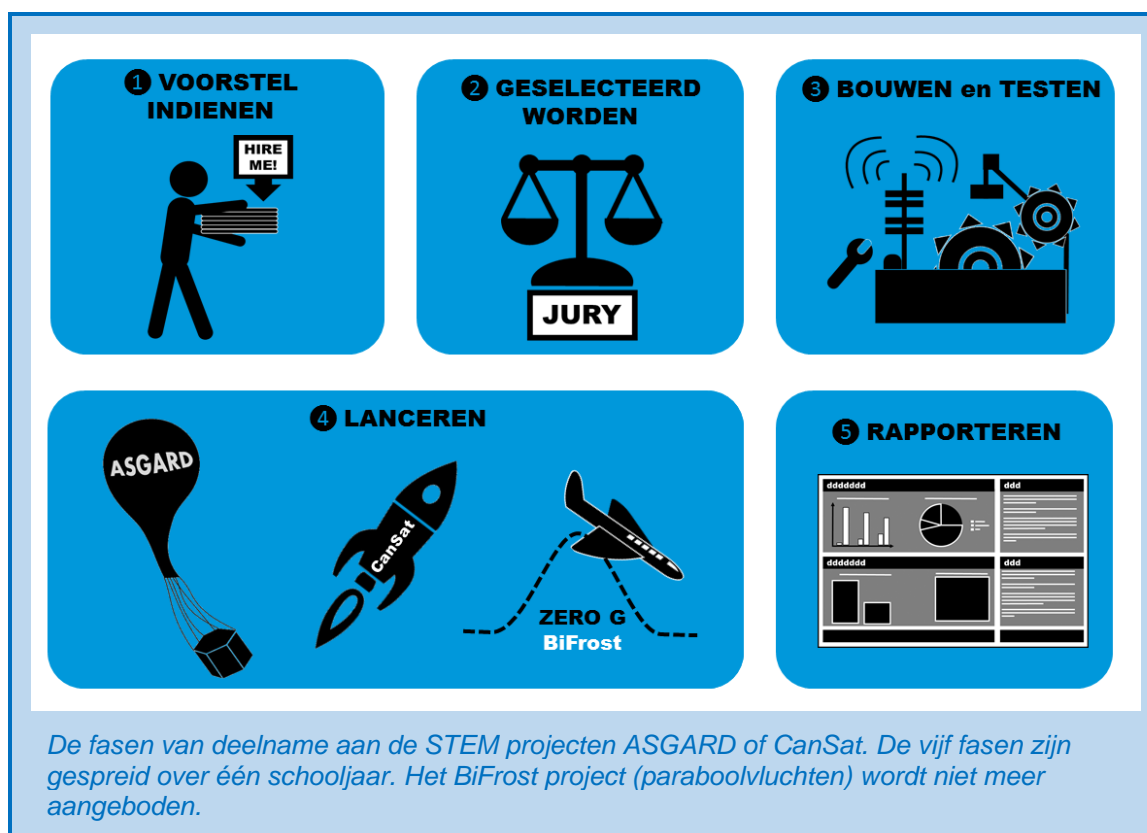
Solderen: Instructies	78
5 Lancering	79
6 Data verwerken en presenteren	79
<i>KALIBRATIE EN VERWERKING: INLEIDING</i>	79
<i>TEMPERATUUR</i>	80
Kalibratie van thermistor output	80
<i>LUCHTDRIUK EN HOOGTE</i>	81
De hoogte berekenen op basis van luchtdrukwaarden	81
De gegevens verwerken van de BM280	81
Bronnen	82
Bijlagen	82
Data types	82

1 Bouw je eerste satelliet

1a A space mission at school

Elk jaar kunnen Belgische secundaire scholen deelnemen aan STEM projecten waarbij leerlingen hun eigen experiment lanceren in een educatieve “ruimtemissie”. De oefening in deze cursus is in het bijzonder geschikt als introductie voor de projecten ASGARD en nCanSat. Het patroon is bij beide projecten hetzelfde:

- 1) **Inschrijving:** De leerlingen bedenken zelf een experiment, en beschrijven hun idee in een dossiertje ('proposal'). Een leraar is begeleider van dit leerlingenteam.
- 2) **Selectie:** een jury van deskundigen selecteert zoveel mogelijk proposals om deel te nemen. De jury is niet noodzakelijk op zoek naar de 'beste' inschrijvingen. Vooral de leerervaring is belangrijk.
- 3) **Het experiment bouwen:** Eens geselecteerd gaan de leerlingen hun zelfbedacht experiment in elkaar knutselen en testen. Een moeilijke, maar leerrijke fase.
- 4) **Lancering:** Afhankelijk van het project wordt het experiment gelanceerd met een stratosfeerballon van KMI of CanSatraket.
- 5) **Rapportering:** Zowel tijdens als na het projectverloop moeten de teams rapporteren over hun werk, los van het al dan niet bekomen van zinvol resultaat.



Drie verschillende “space missions”

De tabel hieronder geeft een overzicht van de STEM projecten die aangeboden worden aan Belgische scholen en waarvoor je de vaardigheden uit deze cursus kan gebruiken. Voor meer informatie: ga naar www.esero.be (projecten).

Project	Beschrijving	Wanneer	Teams
ASGARD	Je experiment bereikt met een stratosfeerballon een hoogte van +/- 30 km. Dat is boven de Ozonlaag, de omstandigheden zijn vergelijkbaar met het oppervlak van Mars. Niet competitief.	Elk schooljaar – bekijk de projectkalender op esero.be <ul style="list-style-type: none"> ▪ inschrijven voor 11 nov ▪ Lancering rond eind april 	<ul style="list-style-type: none"> ● Max 5 leerlingen + 1 of 2 leraren ● 2^{de}-3^{de} graad secundair (+1 klas uit derde graad lager onderwijs) ● Open voor scholen uit alle landen
CanSat Belgium	Je lanceert je satelliet ter grootte van een 33 cl blikje met een kleine raket naar een hoogte van 1 km. Je recupereert de satelliet via een veilig landingssysteem, en houdt radiocontact tijdens de vlucht. Competitief.	Elk schooljaar– bekijk de projectkalender op esero.be <ul style="list-style-type: none"> ▪ Inschrijven ergens in oktober ▪ Lancering rond eind april/begin mei 	<ul style="list-style-type: none"> ● 24 teams eindigen met raket-lancerings ● 4 tot 6 leerlingen + 1 leraar ● Derde graad secundair ● Voor Belgische scholen
Astro Pi	Je codeert (Python) een apparaat ‘Astro Pi’ dat aanwezig is in het ISS via een online simulatietool. De Astro Pi wordt bestuurd door een Raspberry Pi. Een Europese astronaut speelt alle leerlingencodes af in het ruimtestation. Niet competitief.	Elk schooljaar – bekijk de projectkalender op esero.be	<ul style="list-style-type: none"> ● Elk team maakt 1 code. ● Aantal leerlingen naar keuze ● Je kiest voor beginners-oefening of gevorderd.

Elektronische experimentjes voor absolute beginners

Hoewel er in het project ASGARD voorbeelden zijn van niet-elektronische experimenten, gebruiken de meeste leerlingenteams een microcontroller/microcomputer om hun experiment aan te sturen.

ESERO Belgium biedt aan Belgische leraren vormingen aan waarin men een eerste ervaring opdoet met het gebruik van de microcomputer Raspberry Pi Pico.

Maak je eerste mini-satelliet

In deze cursus gaat men onder begeleiding een eerste ‘satelliet’ maken die de luchtdruk en de temperatuur kan meten. Op het einde van de vorming maakt de satelliet van elk deelnemend team een vlucht met een drone of een val langs een koord. Vervolgens kan men de temperatuur, luchtdruk en hoogte van de vlucht uitzetten op een grafiek. De vorming haalt absolute beginners over de drempel om een elektronisch aangestuurd experiment te bouwen met een leerlingenteam, en aldus in te schrijven voor de STEM projecten ASGARD, of CanSat. Daarbovenop deelnemen aan het project Astro Pi is erg laagdrempelig en kan je inleiding in microcomputers en python code mooi versterken.

1b Onderdelen van een satelliet (payload, bus)

Je maakt een satelliet

Strikt genomen maakt u in deze vorming geen satelliet. Een satelliet is immers een kleiner object dat in een baan om een groter hemellichaam cirkelt. We onderscheiden natuurlijke satellieten (zoals een maan) en kunstmatige satellieten. Deze laatste zijn door mensen gemaakte apparaten die in een baan rond de aarde of een ander lichaam bewegen. We noemen het apparaat dat u zal vliegen toch een satelliet omwille van volgende gelijkenissen:

- Wat u maakt is de “**payload**”, en de drone waarmee het vliegt is de “**bus**”. Deze termen worden verder uitgelegd.
- Uw **metingen** dienen volledig **autonoom** uitgevoerd te worden tijdens de vlucht van de payload. Alles wordt vooraf op de grond geprogrammeerd en voorbereid.
- De satelliet werkt autonoom, maar wordt bijgestaan door de “**ground control**”. In dit geval is dat de besturing van de drone (door de drone piloot) en de dataverwerking van de metingen op uw laptop. Indien u de data live tijdens de vlucht via radiocommunicatie ontvangt, dan is ook de grondantenne deel van de “ground control”.

Payload

De payload is het gedeelte van de satelliet waarmee metingen of experimenten uitgevoerd worden. De payload is als het ware de nuttige lading. Dit laatste is trouwens de Nederlandse vertaling van het Engelse woord payload.

In deze vorming bestaat onze payload uit volgende onderdelen:

- de microcontroller (Raspberry Pi Pico)
- sensoren voor druk en temperatuur
- de geheugenkaart of de data-transmitter
- verbindingkabels
- weerstanden en LEDs
- batterij

Ze worden één voor één besproken in onderstaande hoofdstukken.

Bus

Een **bus** is in de elektronica een medium waarop verschillende elektronische signalen worden uitgewisseld. Het is dus een netwerk waarop meerdere componenten of apparaten met elkaar communiceren.

Bij een satelliet is de **bus** het geheel van subsystemen die de satelliet correct laten functioneren en in zijn baan laat vliegen. In onderstaande lijst worden algemene subsystemen van satellieten vergeleken met de subsystemen van onze drone-satelliet.

Algemene satelliet bus onderdelen	Onze mini-satelliet via drone gelanceerd
De structuur van het ruimtetuig	De drone en het harnas waarin de payload wordt vastgezet
Communicatiemechanismen om de satelliet te laten communiceren met de grondcontrole	Communicatiemechanismen die de drone laten communiceren met de piloot interface
Navigatiesystemen en telemetrie	Hoogtemeter en gps van de drone
Motoren en brandstofcellen	Motoren en batterijen van de drone
Temperatuur controlesystemen	Niet van toepassing
Energievoorziening (via brandstof of zonnepanelen)	Batterij (power bank) die u meestuurt met de payload

2 Payload

Alle componenten die u nodig heeft om de payload voor deze vorming te maken worden hieronder uitgebreid beschreven. Neem dit gedeelte eerst door, zodat u de componenten goed kent, en begrijpt hoe ze werken. In een volgende stap kunnen ze dan ingeschakeld worden in een circuit met de Raspberry Pi Pico.

Lijst van componenten die u meekrijgt wanneer u deze ESERO vorming volgt:

Categorie	Onderdelen
Microcontroller	<ul style="list-style-type: none"> ● Raspberry Pi Pico
Sensoren	<ul style="list-style-type: none"> ● Optioneel: DS18B20: temperatuursensor ● BMP280: sensor voor druk, temperatuur en hoogte
Hulpstukken	<ul style="list-style-type: none"> ● USB kabel ● Gekleurde Jumper kabeltjes ● Weerstanden 4.7 kOhm en 200 of 220 Ohm ● LEDs groen, oranje/geel, rood ● Breadboard medium afmeting
Data opslag	<ul style="list-style-type: none"> ● SD Card bord ● Micro SD kaart 2 - 16 GB

Stroomvoorziening

Tijdens de vorming verbindt u de microcontroller met uw laptop. Op dat moment wordt de microcontroller van stroom voorzien via de USB verbinding.

Tijdens de vlucht zorgt ESERO BE voor een externe batterij waarop u de microcontroller kan aansluiten.

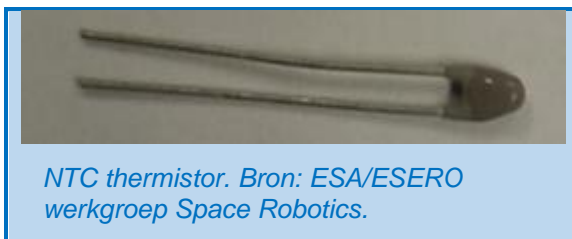
2a Perifere onderdelen van de payload

TEMPERATUUR SENSOR

Hoewel de vorming één bepaalde temperatuursensor gebruikt, is het interessant om ook een andere veel voorkomende soort temperatuursensor te leren kennen: de NTC

De NTC thermistor

NTC staat voor "negative temperature coefficient". De naam is logisch: wanneer de temperatuur stijgt, dan zal de elektrische weerstand van de thermistor dalen, en omgekeerd. Het gevolg is dat de Arduino een veranderde spanning zal registreren over deze sensor indien deze deel uitmaakt van een spanningsdeler. Deze veranderde spanning kan omgerekend worden naar veranderde weerstand, en vervolgens naar veranderde temperatuur. De NTC thermistor is erg eenvoudig, maakt geen interne bewerkingen of berekeningen, en heeft maar twee 'pootjes', net als een gewone weerstand.



Het woord **thermistor** is een samenvoeging van de Engelse woorden 'thermo' en 'resistor' (weerstand). De NTC is namelijk een elektrische weerstand waarvan de waarde afhankelijk is van de temperatuur.

De TMP36 sensor

Introductie

Tegenwoordig bevatten de meeste sensoren meer dan alleen maar de passieve component die verantwoordelijk is voor de variabele output (zoals de weerstand van de NTC). Met behulp van een set transistoren voert de sensor reeds bewerkingen uit, zodat de uitgangsspanning beter overeenkomt met de gemeten waarde.

Dit is ook het geval met de TMP temperatuursensor.



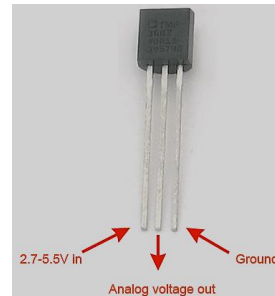
Merk op dat het output bereik (spanning) alleen maar positieve waarden heeft, wat de verwerking iets gemakkelijker maakt.

Tip:

Om te werken met een bepaalde sensor, kijk altijd naar de **datasheet** van de fabrikant, en ga niet gewoon op internet zoeken naar allerlei info over deze sensor. Iedereen kan om het even wat op internet zetten, en er is geen controle op juistheid.

Hoe gebruiken?

- Verbind de linkse pin met een spanningsbron (2,7 - 5,5 V).
- Verbind de rechtse pin met de grond (aarding, nul).
- Verbind de middelste pin met de analoge pin die de meetwaarde moet ontvangen.



- Voer de meting uit, je krijgt een reeks uitgangswaarden V_{out} in mV (milliVolt).
- Pas onderstaande formule toe op de output waarden. De resultaten drukken de temperatuur uit in °C.

$$\text{Temperatuur} = (V_{out} - 500) / 10$$

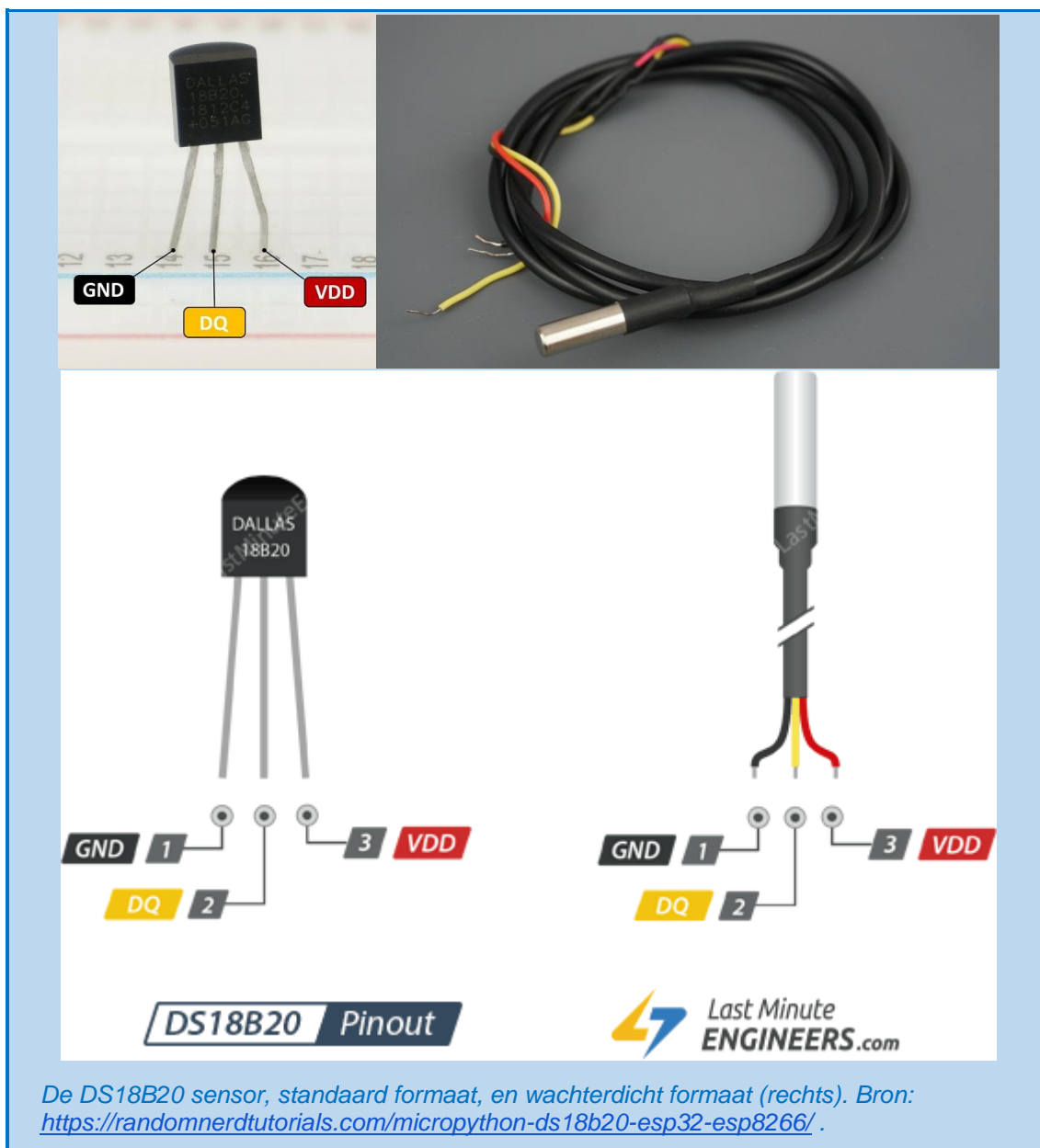
Opmerking

Wanneer meerdere sensoren aangesloten zijn op het systeem, gebeurt het soms dat de temperatuurmetingen onrealistische waarden opleveren. In dat geval is er interactie opgetreden met foute resultaten als gevolg. Je kan dit vermijden door in de code na elke meting de sensor te deactiveren, en een delay (even wachten via code) tussen beide metingen in te bouwen. Je doet er goed aan in code eventuele “onmogelijke” waarden al uit te filteren

DS18B20 Temperatuur Sensor

Een goed alternatief voor bovenstaande temperatuursensoren, is de digitale temperatuur sensor DS18B20, welke verkrijgbaar is als losse component of geïntegreerd in een waterbestendige behuizing voor gebruik in vochtige omgeving.





Elke DS18B20-temperatuursensor heeft een unieke 64-bits seriële code. Hierdoor kunt u meerdere sensoren op dezelfde datakabel aansluiten. U kunt dus de temperatuur van meerdere sensoren krijgen met slechts één GPIO.

Hier is een samenvatting van de meest relevante specificaties van de DS18B20 temperatuursensor:

Stat Limieten

Communicatie	via eendraads bus - 2 modes: normaal en parasitair
Temperatuur bereik	-55°C tot +125°C
Foutmarge	+/-0,5 °C (tussen -10°C tot 85°C)
Prijs	€ 0,90 - € 2,95 (waterdicht)
Voeding	3,0 V tot 5,5 V

Voor meer informatie raadpleeg het [DS18B20 datasheet](#).

We zullen in deze handleiding verder deze digitale sensor gebruiken omwille van zijn gebruiksgemak, en de mogelijkheid er meerdere parallel te schakelen. De eerdere sensoren werken evenwel net zo goed, je hebt enkel een andere schakeling, en andere code, nodig om ze uit te lezen.

COMMUNICATIE MET SENSOREN VIA I2C OF SPI

Alvorens de volgende sensor te bespreken, dienen we iets te weten over de communicatie die plaatsvindt tussen een microprocessor en de gekoppelde onderdelen. Er bestaan meerdere types communicatiesystemen (of communicatie interfaces). In deze vorming gebruiken we de volgende twee:

	I2C bus	SPI bus
Hoofdkenmerken	1 master 1 slave 2 verbindingen nodig	1 master 1 of meer slaves (CS) 4 verbindingen nodig
Voordelen Nadelen	Eenvoudig (2 pins) Traag bij veel data Goedkoop Hoger energie verbruik	Iets complexer (4 pins) Sneller Geen limiet woordgrootte Laag energie verbruik
Gebruik in deze vorming	Temp sensor (TMP36) Druk sensor (BMP280)	SD card

De communicatie systemen verschillen in de manier waarop uitgewisselde databits erkend worden als start en stop bits en als data bits. Met andere woorden: er zijn regels binnen de I2C bus en SPI bus waardoor de processor 'weet' wanneer 0 en 1 stroompjes moet geïnterpreteerd worden als het begin en einde van uit te lezen data die van de sensoren komen.

Beide systemen zitten zo in elkaar dat een tijdsignaal (SCL = Serial CLock, blokkgolfsignaal met vaste frequentie) als achtergrond dient om datasignalen uit te lezen.

Beide systemen maken het ook mogelijk dat een 'master' (de microcontroller) en één of meerdere 'slaves' (de sensoren of andere perifere apparaten) met elkaar communiceren. Zo kan de master als het ware bevel geven aan een slave om te beginnen met data leveren, alsook om ermee te stoppen.

Het is niet het opzet van deze cursus om de systemen in detail uit te leggen. Hieronder is een beperkte uitleg voorzien die nuttig is voor de oefeningen van de vorming zelf. Een minimaal inzicht is immers vereist om de sensoren correct aan te sluiten op de microprocessor.

I2C bus

I²C = **Inter-Integrated Circuit** (uitspraak: i kwadraat c)

Wat?

De microprocessor gebruikt een bepaalde combinatie van SCL-sigitaal/data-sigitaal als de start van te registreren data. Vanaf dan begint het bezorgen en uitlezen van de sensordata. Een andere combinatie van SCL-sigitaal/data-sigitaal staat voor een 'stop'. Dan stopt de microprocessor met het uitlezen van data van de sensor, en de sensor stopt met ze door te geven.

Het uitwisselen van start-stop-bevelen en van meetdata tussen de microcontroller en de sensor gebeurt over 1 verbinding (SDA in beide richtingen over 1 draad). De data kunnen niet gelijktijdig van master naar slave EN omgekeerd gebeuren.

Er zijn dus slechts 2 pins in gebruik voor data uitwisseling, en beiden kunnen in beide richtingen bits doorgeven:

- Het klok signaal **SCL** (Serial CLock)
- Het data signaal **SDA** (Serial DAta)

Uiteraard zijn er daarnaast ook 2 geleiders nodig voor de stroomvoorziening van de sensor:

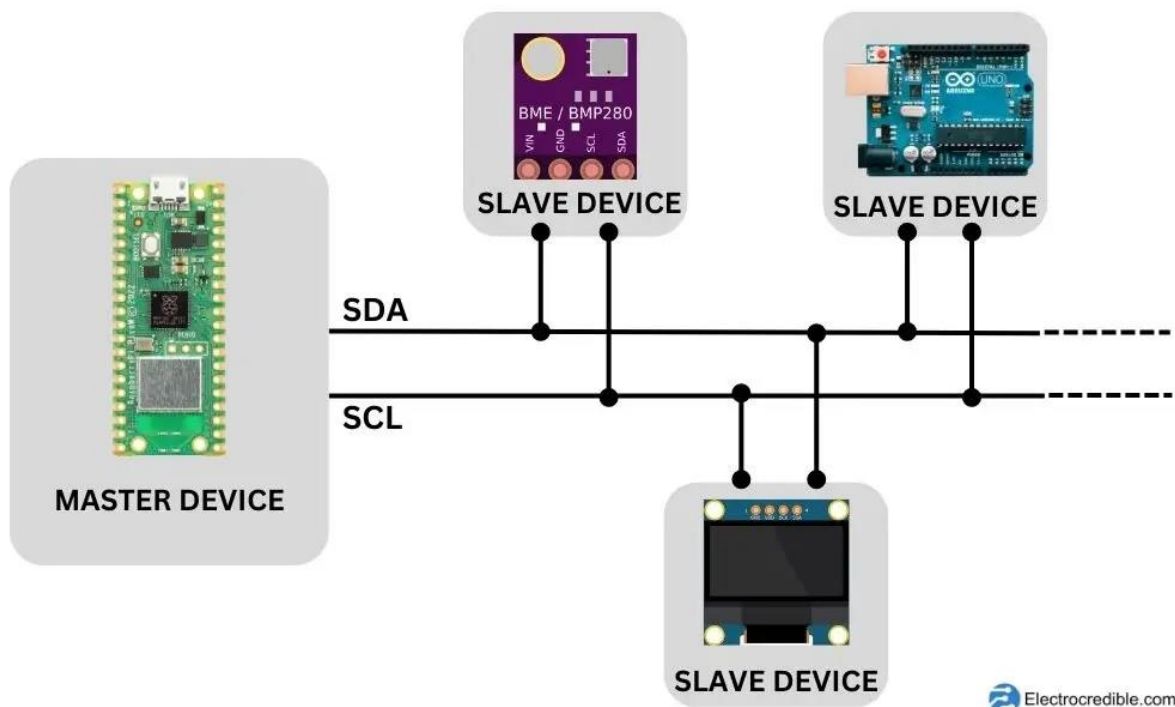
- **3.3 V of 5 V in** (spanningsbron)
- **GND** (aarding, nul)

Master

De microprocessor wordt de 'master' genoemd, want hier wordt controle uitgeoefend op de I2C bus. De processor geeft het start- en stopsignaal en zendt het SCL signaal uit.

Slave

Elke sensor die we verbinden met de microprocessor wordt een 'slave' genoemd. De sensor luistert als het ware naar de bevelen van de processor (master), en geeft zijn data door aan die processor.



SPI bus

SPI = **S**erial **P**eripheral **I**nterface

Wat?

In dit communicatiesysteem is er gelijktijdig informatieoverdracht van de master naar de slave en omgekeerd. Er zijn 4 lijnen vereist voor data-uitwisseling.

De communicatie start met het definiëren van een kloksnelheid, via het signaal op de SCL lijn. Vervolgens wordt een andere lijn (CS) gebruikt door de microprocessor (master) om één van de aangesloten sensoren (slave) te activeren, en te beginnen met data uitlezen. Dit gaat door totdat de master (terug via de CS lijn) het signaal geeft om te stoppen met data uitlezen.

Er zijn 4 pinnen nodig voor data-uitwisseling:

- **SCL** of **SCK** : seriële clock
- **MOSI** : Master output slave input: op deze lijn wordt de data verzonden die de master uitzendt en die bij de slave ontvangen wordt. Niet in omgekeerde richting.
- **MISO**: Master input slave output: op deze lijn wordt de data verzonden die de slave uitzendt en die de master ontvangt. Niet in omgekeerde richting.
- **CS**: Chip select (of SS Slave Select): Deze lijn dient om één van de slaves te activeren. Wanneer de CS lijn laag is (nul Volt), dan start de activering. Wanneer de CS hoog is (3.3 of 5 Volt), dan wordt deze slave stilgelegd (stop). Elke slave moet een unieke pin hebben voor de CS lijn.

Merk op dat het geven van start-stop bevelen door de master en het uitzenden van meetdata door de slave gelijktijdig kan gebeuren.

Uiteraard zijn er daarnaast ook 2 geleiders nodig voor de stroomvoorziening van de sensor:

- **3.3 V of 5V in** (spanningsbron)
- **GND** (aarding, nul)

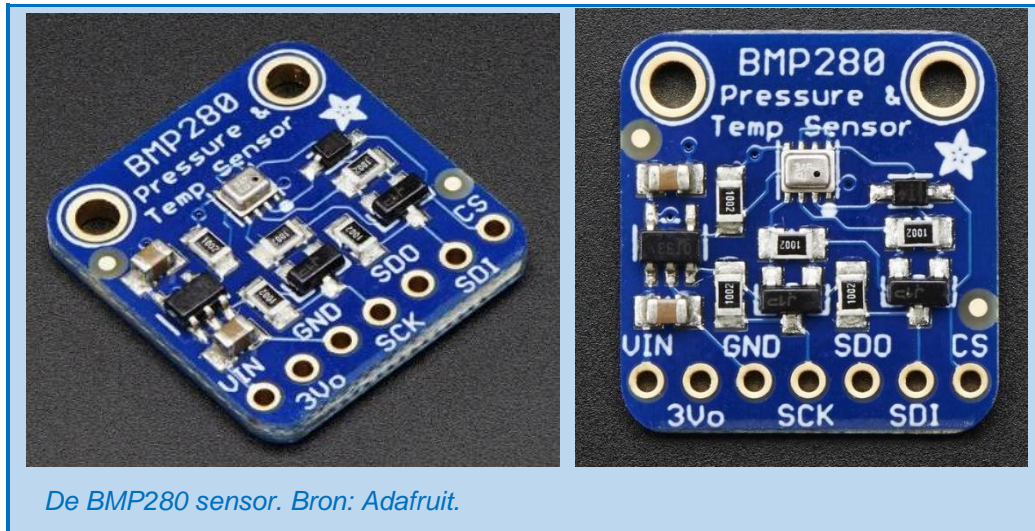
DRUKSENSOR

BMP280: Inleiding

Hierboven lieten we je kennis maken met een eenvoudige sensor met 1 functie. In de elektronica is het echter gebruikelijk om meerdere sensoren te integreren in dezelfde behuizing.

De **BMP reeks** van Bosch zijn sensoren waarmee **tegelijk** de luchtdruk en temperatuur gemeten kan worden, en waar rechtstreeks de hoogte kan uit afgeleid worden. De precisie van de luchtdrukmeting is ongeveer 1 hPa, en die van de temperatuur ongeveer 1°C. Dit maakt de sensor geschikt als barometer en om de hoogte te bepalen met ongeveer 1 meter foutmarge.





BMP280: Beschikbare pins

Er zijn 7 pins aan de onderzijde van de sensor. De eerste 3 zijn power pins:

- **Vin**
Hier komt de spanning binnen. De sensor werkt op 3,3 Volt. Door een spanningsstabilisator op de print te integreren kan de sensor ook met 5 Volt voeding gebruikt worden.
- **3Vo**
Hier kan de stroom buitengaan, met een spanning van 3.3V. Je kan deze pin dus gebruiken als stroombron voor iets anders, met een maximum van 100 mA. Zoniet, dan wordt deze pin niet aangesloten
- **GND**

De volgende 4 pinnen worden '**logic pins**' genoemd. Gebruik makend van de communicatie-interface I2C, hebben we er slechts 2 nodig. Bij communicatie via SPI, dat sneller data transfereert en minder stroom vraagt, hebben we 4 pins nodig.

Afhankelijk van de communicatie-interface (I2C of SPI, zie hoger) worden de pins van de BMP280 sensor anders gebruikt:

Bij I2C communicatie:

- **SCK** = Serial Clock
Deze moet bij onze microprocessor verbonden worden met een SCL pin. SCL staat voor Synchronize CLock. De pulsen van dit signaal vormen de referentietijd voor het hele systeem (gebruikt als referentie voor data, maar ook voor het uitvoeren van de code zelf), en worden door de microcontroller gegenereert.
- **SDI** = Serial Data In/Out
Deze pin stuurt data van de microcontroller naar de sensor EN omgekeerd. Deze moet bij onze microcontroller verbonden worden met de SDA pin (de voorlaatste pin langs de kant van de digitale, dit is de I2C data pin). De data die uitgewisseld worden op dit kanaal zijn de meetgegevens zelf, maar ook de commando's van de microcontroller, en de signalen nodig om één bepaald 'slave' apparaat te activeren of deactiveren.

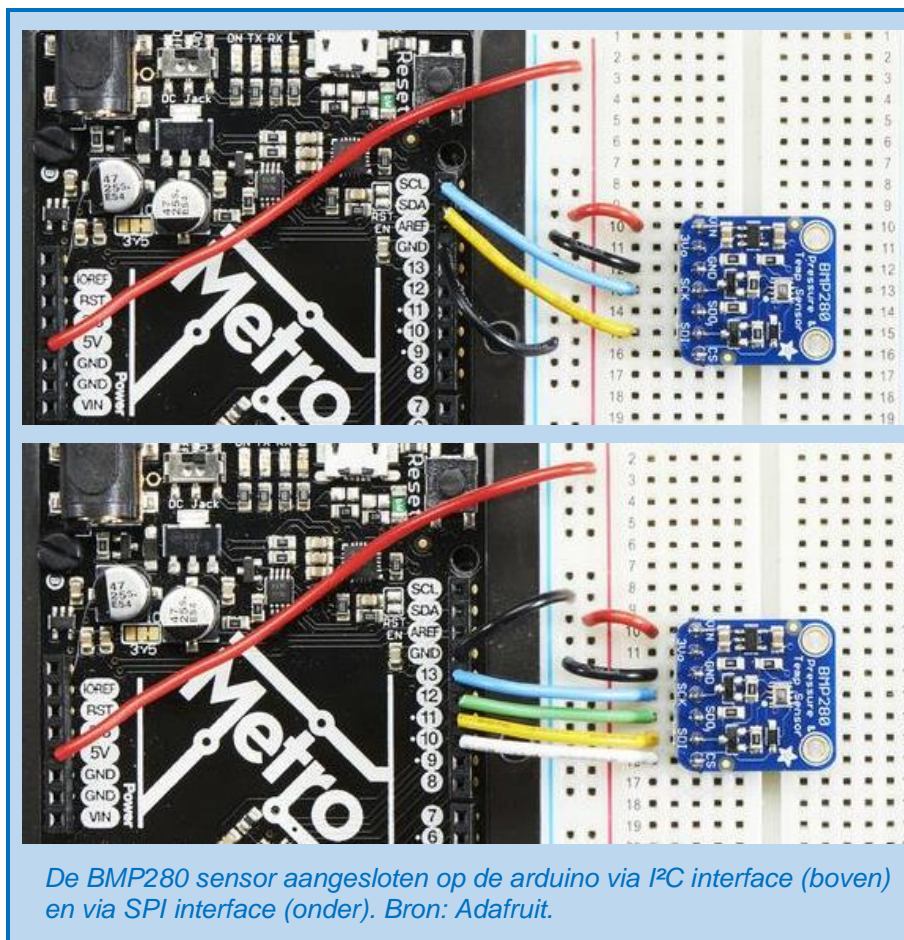
De andere twee pinnen blijven ongebruikt (niet verbonden).

Bij SPI communicatie:

- **SCK** = Serial Clock
Deze pin stuurt tijdsgebonden pulsen naar de microcontroller, idem als bij de I²C interface.
- **SDO** = Serial Data Out
Deze stuurt data van de sensor naar de microcontroller (MISO = Master In Slave Out pin).
- **SDI** = Serial Data In
Deze stuurt data van de microcontroller naar de sensor (MOSI = Master Out Slave In pin).
- **CS** = Chip Select
Je hebt per apparaat dat je aansluit een unieke CS pin nodig. Ze wordt gebruikt om te bepalen welke sensor op een bepaald moment geactiveerd moet worden.

Alle 4 deze pinnen moeten elk verbonden worden met een unieke digitale pin (nrs 2-13) die samen SPI ondersteunen.

Wanneer je meerdere sensoren aansluit met SPI interface, dan kunnen alle SDO, SDI en SCK pinnen telkens gedeeld worden op 1 pin. De CS pin daarentegen moet voor elke sensor een unieke pin krijgen.



Werken met de BMP280

Op de figuur hierboven is de BMP sensor aangesloten via een breadboard. Dit is een ideale manier om je satelliet in elkaar te knutselen en te testen, alvorens alles definitief aan elkaar te solderen. Om de sensor gemakkelijk te verbinden met de gaatjes van de breadboard, zit er een 'header' bij: een reeks aan elkaar hangende pinnetjes met dezelfde tussenafstand als de gaatjes van de sensorprint en het breadboard.

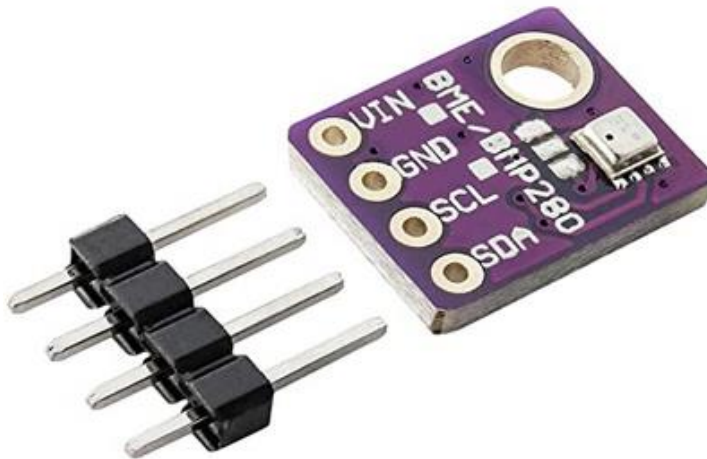


Wanneer de pinnen aangesloten zijn zoals hierboven, dan gebeurt alle verdere werk via de code. Je zal in de code de correcte functies van de verschillende aangesloten pins moeten definiëren, zodat die overeenkomen met de juiste BMP280 functies. Vervolgens kan je de

code schrijven om de gewenste meetresultaten te verkrijgen. Dit leren we verder in deze cursus. Meer weten? Lees dan [het datasheet van de BMP280](#).

BME280 druksensor

De BME280 is een versie van de BMP280 met enkel het I2C protocol beschikbaar. Het voordeel hiervan is dat deze sensor daardoor kleiner is.



Op deze foto zie je op de druksensor een gaatje. Dit gaatje moet verbonden zijn met de buitenlucht om correct de druk te kunnen meten.

Als je zelf een druksensor kiest

In deze vorming werd de druksensor voor jou uitgekozen. Als je echter voor een STEM project op school zelf op zoek gaat naar een druksensor, dan raden we aan om te letten op de volgende belangrijke eigenschappen:

- **Gevoeligheid (Sensitivity)**
Wat is de minimale drukwijziging die kan gemeten worden door deze sensor?
- **Reactietijd (Response Time)**
Hoe snel is de sensor?
- **Lineariteit (Linearity)**
Hebben de output data van de sensor een lineair verband met de luchtdruk (voor het beoogde meetbereik)?
- **Uiterste waarden (Range)**
Wat zijn de minimale en maximale drukwaarden die door de sensor kunnen gemeten worden?
- **Hysteresis**
Bij een dalende luchtdruk kan een sensor soms een lichtjes andere absolute meting geven dan bij een stijgende luchtdruk. Dit lichte verschil heet hysteresis, en is een maat voor een zekere traagheid waarmee een sensor de exacte waarde van de omgeving overneemt.

μPROCESSOR OF μCONTROLLER

De μ controller is het meest essentiële onderdeel dat in het vormingspakket zit. Deze component heeft een intern geheugen; het is hierop dat de code terecht komt die je zal schrijven voor je satelliet.

De Raspberry Pi Pico wordt verder besproken onder hoofdstuk 2b.

BATTERIJ/POWER BANK

Energie voor satellieten

Echte satellieten worden in de meeste gevallen voorzien van energie met zonnepanelen met fotonvoltaïsche cellen die in serie geplaatst zijn. Fotonvoltaïsche cellen zijn eigenlijk omgekeerde LEDs. Als er licht op invalt, wordt een kleine stroom gecreëerd. Gewoonlijk zullen de zonnepanelen eerst een batterij opladen, die vervolgens de nodige stroom voorziet voor de instrumenten van de payload. Dit is des te meer nodig wanneer een satelliet in een lage baan om de aarde cirkelt, aangezien hij bij elke omwenteling een tijdje in de schaduw van de aarde zit.

Het energieverbruik van de satelliet moet in detail berekend worden, zodat de stroomvoorziening er precies kan op afgesteld zijn.

In deze vorming zullen we enkel een batterij gebruiken, die vooraf ruim voldoende opgeladen wordt.

Welke batterij gebruiken we?

Als stroombron voor onze payload gaan we kiezen voor een power bank.



Volgende kenmerken kunnen van belang zijn voor een **power bank**:

- Soms elektronisch geregeld. Het kan zijn dat de powerbank geen stroom geeft, wanneer de gebruiker die eraan hangt (onze payload) te weinig stroom vraagt. Dit is ingebouwd

in sommige power banks om te voorkomen dat een klein lek de energie laat wegvloeien wanneer een aangesloten apparaat niet aanstaat. Bij aankoop moet men hierop letten.

- Bestaat in lichte varianten, zelfs lichter dan een 9V batterij.
- Heeft een grote energiedichtheid (Lithium batterij).
- Kan steeds herladen worden via USB voeding.

WEERSTANDEN

Weerstanden?

Deze component is naar zijn functie genoemd. Een elektrische weerstand **beperkt de doorgang van elektrische stroom** en veroorzaakt ter plekke een gewenste vermindering van het geleidingsvermogen. Hoe hoger de weerstandswaarde hoe sterker de stroomdoorgang wordt beperkt. Geleiders zoals koper en aluminium hebben een heel kleine weerstand voor stroom. Isolators zoals pvc en glas geleiden vrijwel geen stroom en hebben dus een heel hoge weerstand. Weerstanden in elektronische componenten hebben een vooraf **bepaalde waarde** die daar ergens tussen in zit. De afkorting voor weerstand zoals gebruikt worden op stuklijsten en schema's is de: **R (van Resistance)**.

Weerstand R wordt ook wel met het **Ω (ohm)** teken aangegeven, bijv. $R = 512 \Omega$. Dit is eigenlijk de **eenheid** van weerstand.

De "gewone" weerstanden (met uitlopers of pootjes) zijn normaal gesproken van een kleurcode voorzien (gekleurde ringen) en zijn niet ESD-gevoelig (ESD = electrostatic discharge). Daarom moet je ze niet te bewaren in gesloten plastic zakjes, en kan je ze aanraken zonder schade te veroorzaken.

Gebruik

Weerstanden hebben we nodig om de stroom doorheen bepaalde (gevoelige) onderdelen of doorheen het hele systeem te begrenzen. Dit is bij een microcontroller-gebaseerde satelliet van groot belang, want wanneer de maximale toegelaten stroom overschreden wordt, dan is onze microcontroller rijp voor de vuilbak! We leren verder in de cursus hoe we dit vermijden.

Kleurcodes

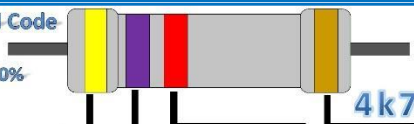
Weerstanden met vier kleurringen

- De eerste twee ringen bepalen het getal. Bijvoorbeeld: eerste band geel, tweede band paars: geeft basisgetal 47.
- De derde ring bepaalt de vermenigvuldigingsfactor, of hoeveel nullen komen er achter het getal. Bijvoorbeeld: derde band rood betekent vermenigvuldigen met 10^2 (of x 100).
- De vierde ring is om de tolerantie of foutmarge van de weerstand aan te geven. Bijvoorbeeld: vierde band goud betekent dat de reële waarde van de weerstand (in Ohm) tot 5% kan afwijken tov de opgegeven waarde.
- De waarde van de weerstand in ons voorbeeld is dus $4700 (+ \text{ of } - 235 \Omega)$. Of anders geschreven: $4k7 \Omega + \text{ of } - 5\%$.

De eerste ring zit het dichtst bij de draaduitloper en de laatste ring is breder gemaakt.



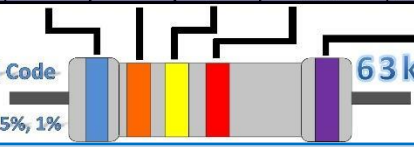
4 – Band Code
2%, 5%, 10%



4k7Ω ±5%

Color	1 st Band	2 nd Band	3 rd Band	Multiplier	Tolerance
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1%
Red	2	2	2	100Ω	± 2%
Orange	3	3	3	1kΩ	
Yellow	4	4	4	10kΩ	
Green	5	5	5	100kΩ	± 0.5%
Blue	6	6	6	1MΩ	± 0.25%
Violet	7	7	7	10 MΩ	± 0.1%
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5%
Silver				0.01Ω	± 10%

5 – Band Code
0.1%, 0.25%, 0.5%, 1%



63k4 0.1%

Betekenis van de kleurcodes bij weerstanden met 4 en met 5 ringen.

Bij **weerstanden met 5 ringen** zijn het niet de eerste twee ringen, maar wel de eerste drie ringen die het getal bepalen. De vierde band is dan de vermenigvuldiger, en de vijfde ring de tolerantie.

KABELS

Het vormingspakket bevat 2 soorten kabels die essentieel zijn voor elk microcontroller project:

USB verbinding voor Raspberry Pi Pico

Het type kabel: USB 2.0 A naar micro (male/male). Dit is het type kabel waarmee de Pico verbonden wordt met de laptop of desktop computer. Vaak heb je zo'n kabel in huis voor andere digitale toestellen (vroegere smartphones, batterij laders, ...).

Wanneer deze kabel aangesloten wordt op de computer en op de Pico wordt de microcontroller van stroom voorzien is. Er is hier geen indicatie van in de vorm van een oplichtende LED, zodat stroom niet verspild wordt.

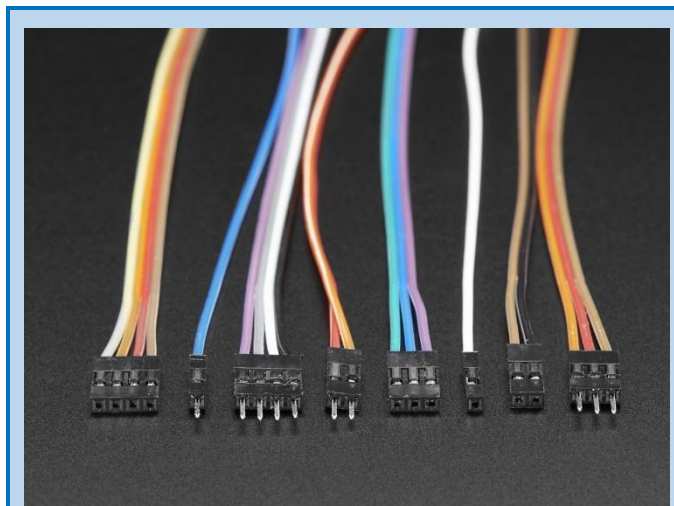
Behalve stroom worden uiteraard ook alle data (code, input, output data) uitgewisseld via deze kabel.



Gekleurde kabelset of jumper kabels

De serie gekleurde fijne kabels (in dit geval male/male) worden gebruikt om allerlei pinnen te verbinden. Ze worden ook soms jumper kabels genoemd.

Wanneer je nadenkt over het elektronisch circuit dat je in elkaar knutselt, zal je dat gewoonlijk eerst schematisch tekenen. De vele kleurtjes van de jumper kabels laten toe dat je het circuit vervolgens correct nabouwt met echte verbindingen op de breadboard.



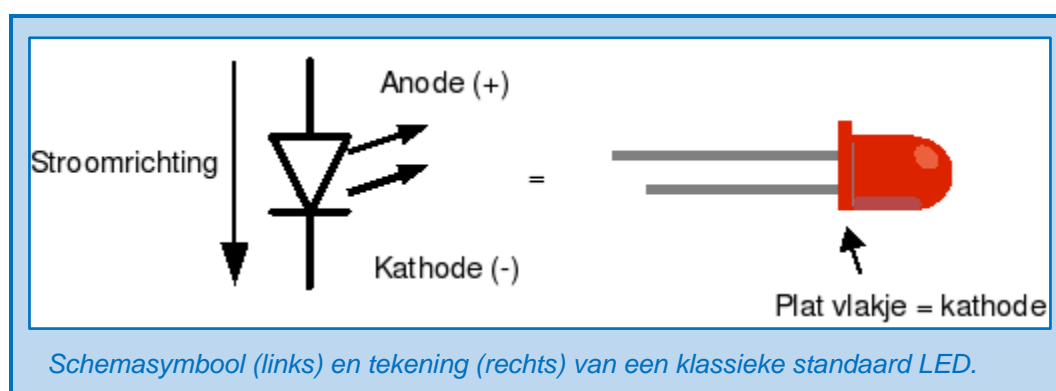
Platte jumper kabels in met mannelijke (male) en vrouwelijke (female) uiteinden. Bron: Adafruit.

LEDs

Wat is een LED?

LED is de afkorting van **Light Emitting Diode**, wat zoveel betekent als lichtgevende diode. En inderdaad vertoont een led ook een belangrijke eigenschap van een diode, namelijk dat de **stroom** slechts **in één richting** wordt doorgelaten; er kan alleen stroom lopen van de anode (positieve pool) naar de kathode (negatieve pool). Dit is een belangrijk verschil tussen een led en een gloeilamp, waarbij de polariteit niet uitmaakt.

Een ander verschil met de gloeilamp, is dat een diode de elektrische stroom niet afremt (geen weerstand). Daarom moeten we bij het gebruik van een LED zelf de **stroom afremmen**, door er een **weerstand** in serie te plaatsen.



Schemasympool (links) en tekening (rechts) van een klassieke standaard LED.

Bij de hier weergegeven “gewone” LED met twee aansluitdraden zijn de anode en de kathode als volgt te herkennen:

- De **anode** (de positieve aansluiting) is de langste draad.
- De **kathode** (de negatieve aansluiting) is de kortste draad. Verder is de kathode bij ronde leds vanaf 5 millimeter ook aangegeven door middel van een plat vlakje bij een van de draden.

Een LED in het circuit opnemen

Houd er rekening mee dat er LEDs in talloze kleuren, vormen, maten en varianten verkrijgbaar zijn, zodat niet altijd direct duidelijk is wat de anode en de kathode is. Meerkleurige LEDs hebben bijvoorbeeld meer dan 2 pootjes.

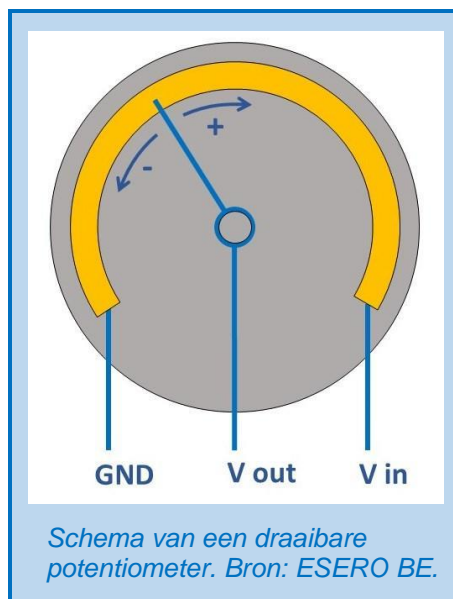
LEDs moeten in het circuit worden vergezeld van een weerstand (in serie geplaatst), om te vermijden dat er teveel stroom doorheen gaat. Het is aan de gebruiker om zelf de juiste weerstand te berekenen.

Er zijn dan ook twee waarden belangrijk bij het gebruik van een LED; je vindt ze in de bijhorende datasheet:

- U_{LED} is de **minimale spanning** die nodig is voor het verkrijgen van de halfgeleiderwerking. Bij een lagere spanning is er geen licht.
- De **maximale stroom** die door de LED mag gaan zonder dat hij kapot gaat.

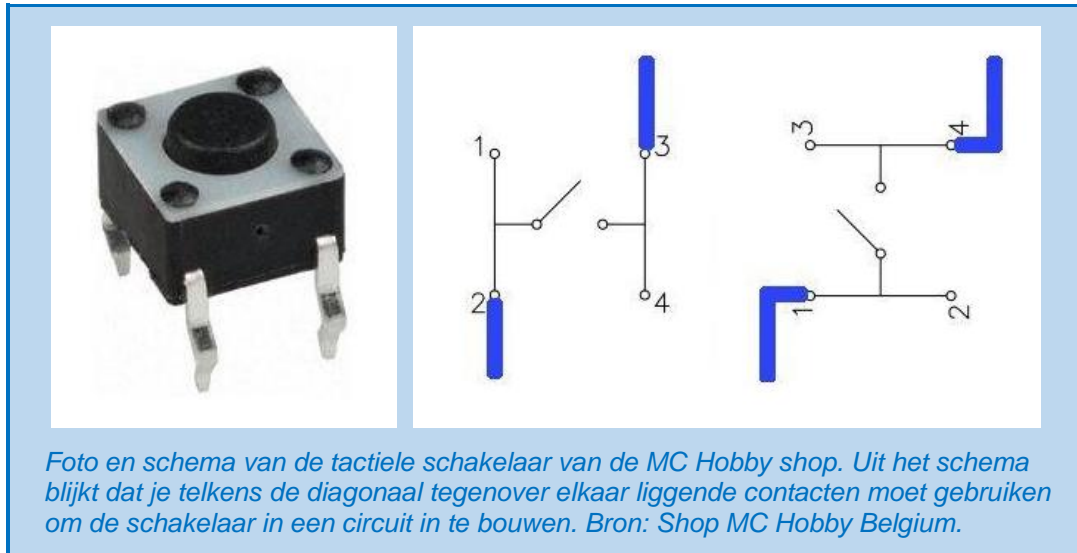
POTENTIOMETER

Een potentiometer of potmeter is een spanningsdeler, waarbij de spanning aan de V_{out} kan aangepast worden door een draai- of schuifbeweging te maken. Ze worden gebruikt om een spanning gradueel te verkleinen naar keuze.



SCHAKELAAR

Een **tactiele schakelaar** (eentje die je aan/uit doet door er 1 keer op te duwen) van 6 mm groot en 4 pootjes van 2,5 millimeter wordt in deze vorming gebruikt om de spanning naar de microcontroller aan en uit te zetten. We gebruiken hem samen met bepaalde weerstanden om onze Pico te beschermen tegen blijvende schade door te hoge stroom (zie verder).



De 4 pootjes passen precies in de breadboard.

2b Raspberry Pi Pico

TERMINOLOGIE

Bij het gebruik van de Pico (en andere microcontrollers) hoort een uitgebreide terminologie of woordenschat. We zetten de belangrijkste termen hier op een rijtje:

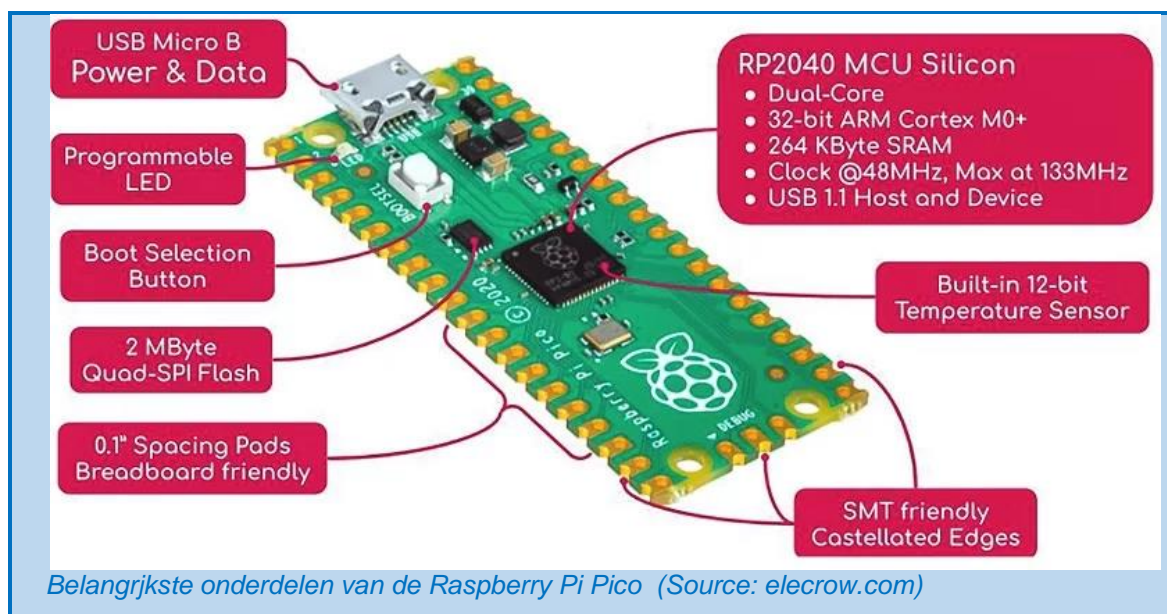
Term	Uitleg
Master/Slave	Gewoonlijk wordt de microcontroller beschouwd als de 'Master': het centrale apparaat dat de commando's geeft aan de perifere apparaten. Deze laatste ontvangen de commando's, voeren ze uit, en geven data (het 'werk') terug.
Code	Dit is een volledig programma(tje) geschreven in het code-venster van de software, om vervolgens op te laden naar de Pico controller. De code-taal die wij zullen gebruiken is python.
Bibliotheek (Library)	Dit is een stukje code met een zekere functie die je online vindt. Je verwijst naar een bibliotheek in je code helemaal in het begin van de code, via het commando " <code>import</code> ". Bibliotheeken worden gebruikt om te vermijden dat je voor bepaalde (courant gebruikte) functies telkens weer 'het wiel moet uitvinden', of om het gebruik van bepaalde onderdelen gemakkelijker te maken. Bibliotheeken krijgen gewoonlijk een naam die verradt over welke functies het gaat. De bibliotheek staat bewaard ergens op je laptop, je code geeft een verwijzing er naartoe.
Shield	Dit is een kunststof plaatje (Printed Circuit Board of PCB) met vooraf ingebouwde verbindingen en componenten/apparaten of door uzelf gemonteerde componenten/apparaten, die in zijn geheel aan de Pico gekoppeld wordt. Met dergelijke shields kan met in 'verdiepingen' werken om de Pico satelliet uit te breiden met meerdere functies.
compiling	Dit is het proces waarbij de code zoals wij die ingeven (een 'human readable' versie van het programma) omgezet wordt in een code die door de chip begrepen wordt, en dus kan uitgevoerd worden. Dit proces is het werk van de software. In het geval van python is het iets complexer: men spreekt van een <i>interpreter</i> , de code die je schrijft wordt ingelezen en geïnterpreteerd door python.
Syntax	Een taal heeft grammatica en leestekens om communicatie correct en duidelijk te laten verlopen. Dat is ook het geval bij de programmeertaal python. Hier wordt dat syntax genoemd. Het is de set regels om correct in de programmeertaal te schrijven.
Hex code	De software van de Pico communiceert met jou via een python-code. Dit is de programmeertaal waarin je code en de bibliotheken in geschreven zijn. Dit wordt ook wel een "human readable" medium genoemd. Maar uiteindelijk zal de software je programma omzetten in een totaal andere code alvorens die naar de Pico chip gestuurd wordt. Deze finale code – niet leesbaar voor 'humans', wel voor microcontrollers – wordt de hex code genoemd.

HARDWARE Raspberry Pi Pico

De Raspberry producten zijn 'open source', maar de chips zelf niet. In theorie kan om het even wie klonen en varianten maken van deze boards, en deze verkopen onder een andere naam, maar gezien de prijs al heel scherp is, is daar nog geen markt van, zoals die wel bestaat voor de Arduino.

Welke **onderdelen** vinden we terug op een Raspberry Pi Pico board? Hieronder worden de belangrijkste overlopen. De Pico heeft:

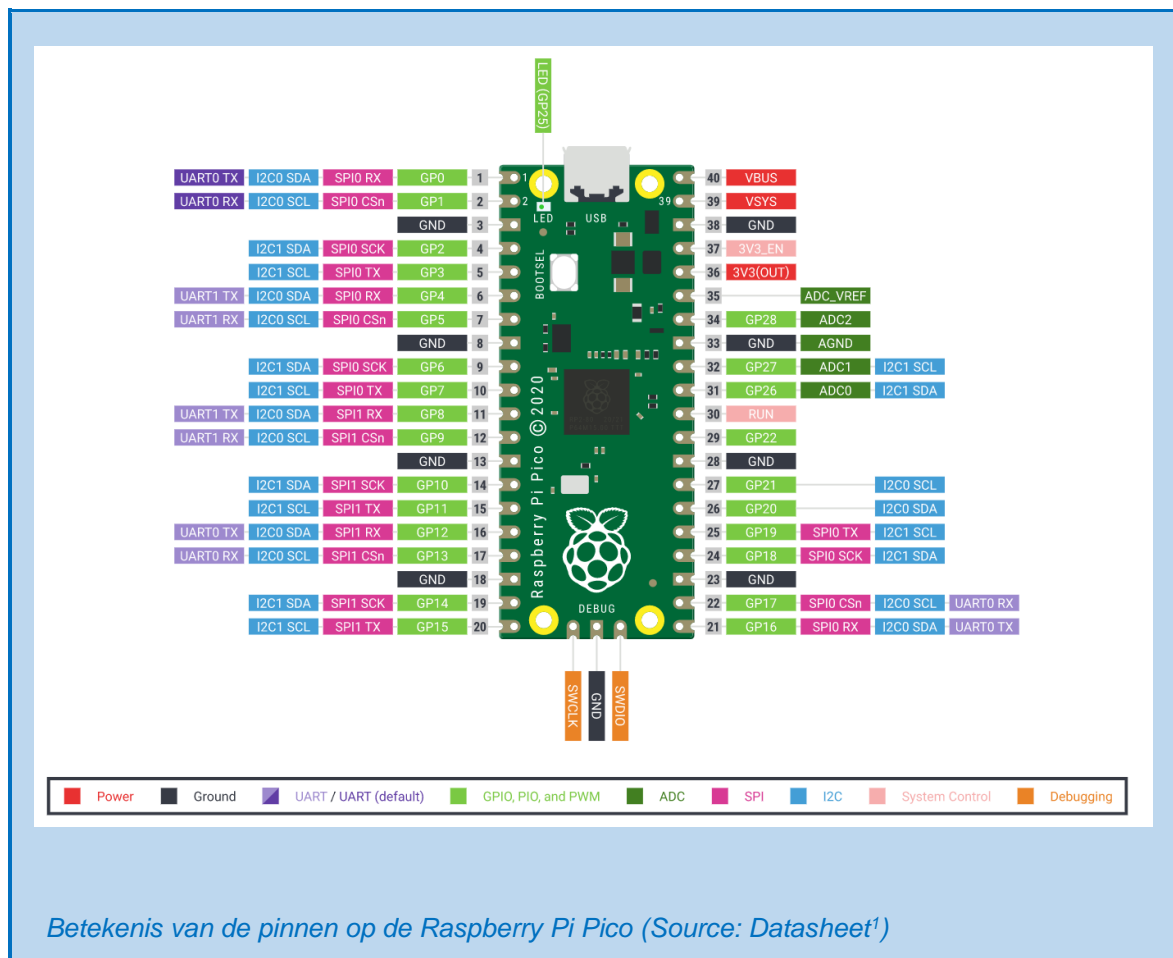
- Een 133MHz processor met 264 kilobytes aan RAM geheugen
- 2 megabytes aan bestandsopslag
- Een BOOTSEL knop die gebruikt wordt om MycroPython op de Pico te installeren
- Een groene LED
- Een USB connector om stroom te voorzien aan de Pico en software of data te versturen.
- 2 x 20 pins die gebruikt kunnen worden om de Pico van stroom te voorzien, alsook om externe elektronische componenten te controleren of data van te ontvangen.



Elk van de 40 pinnen heeft zijn eigen functie. Als u de pincodes voor een Raspberry Pi Pico wilt weten, kunt u het volgende diagram of [deze interactieve website](#) raadplegen.

Tijdens het werken met de Pico hoeft u alleen te werken met:

- Rode en zwarte pinnen = pinnen gerelateerd aan stroom- en aardverbinding
- Paarse pinnen = pinnen gerelateerd aan de UART-communicatie
- Roze-pinnen = pinnen gerelateerd aan het SPI-communicatieprotocol
- Blauwe pinnen = pinnen gerelateerd aan het I2C-communicatieprotocol
- Groene pinnen = pinnen voor input en output
- Donkergroene pinnen = pinnen voor ADC (Analoog naar Digitaal Conversie)



3 analoge input pinnen

Hier komen analoge data binnen van aangesloten analoge sensoren (max 3V). We noemen deze de ADC pinnen

26 digitale pinnen

3 van deze pinnen kunnen geconfigureerd worden als de ADC pinnen hierboven gegeven. Deze pins kunnen gegevens ontvangen van digitale aangesloten apparaten, en ook gegevens versturen naar aangesloten apparaten. In beide gevallen gaat het over spanningsvariaties tot maximaal 3V (bits: spanning nul of max).

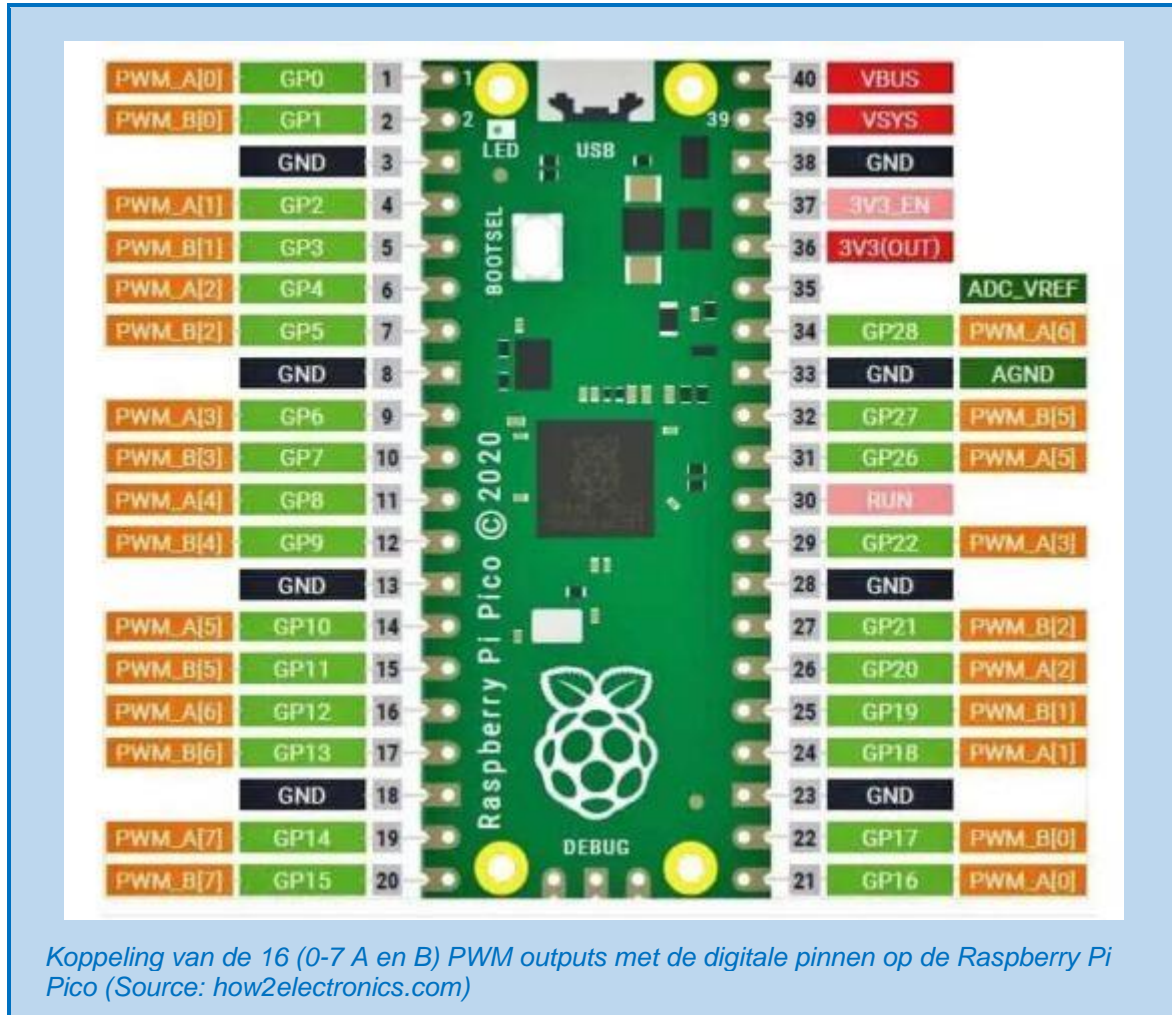
Digitale pins met tussenwaarden: PWM

De digitale pins kunnen gebruikt worden voor lagere spanningswaarden dan 3V. Maar hoe geeft een pin een ander voltage? De Pico kan op digitale pinnen immers alleen maar 3 of 0 volt geven? Eigenlijk wisselt hij duizenden keren per seconde tussen 0 en 3 volt. Door de intervallen van 0 V (of de intervallen van 3 V) korter of langer te maken, zal het gemiddeld

¹ <https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf>

uitgangssignaal een waarde zijn tussen 0 en 3V, bijvoorbeeld 1.5 V wanneer beide intervallen exact gelijk zijn.

De Pico heeft 16 onafhankelijke PWM generatoren, welke gekoppeld kunnen worden aan de digitale pinnen, sommige zijn dus dubbel! De koppeling is zoals in onderstaande pin layout.



TX en RX pinnen

T staat voor Transmit, R staat voor Receive. Ze kunnen gebruikt worden voor seriële communicatie tussen de Pico en een ander apparaat.

Standaard worden de digitale pins als input pins beschouwd. Indien je ze als output wilt gebruiken, dan moet dit zo bepaald worden in de code.

3,3 V en 5 V input en output pinnen

Deze pinnen dienen als voeding voor aan te sluiten apparaten, respectievelijk met een spanning van 3,3 V en van 5 V.

Op de **GND** pinnen moet de grond aangesloten worden. Is er geen USB kabel die stroom levert, dan dient de batterij GND via pin 38 te gebeuren.

Op de **VBUS** pin zie je de spanning van de USB verbinding. Deze is normaal 5 V, en zo kun je de 5 V ook verder gebruiken voor randapparatuur. We noemen dit de peripheral mode. Indien je vanuit de Pico zelf USB stroom wil leveren, de zogenaamde Host mode, voor bijvoorbeeld een externe connectie, dien je op VBUS zelf 5 V aan te sluiten.

De **VSYS** pin heeft als waarde deze van de **VBUS** indien de USB kabel aangesloten is aan je PC of battery pack. Wil je zelf stroom via een batterij gebruiken om je project voeding te geven, dan dien je ook de **VSYS** te gebruiken om je Pico los van de USB te voeden. Je kan batterijvoeding van **3,5 V tot 5,5 V gebruiken**.

De **3V3** pin laat toe om externe componenten van 3,3 V spanning te voorzien. Zorg er wel voor dat je niet meer dan 300 mA stroom gebruikt.

Ground pinnen

De GND pinnen geven toegang tot de laagste spanning van het hele systeem.

Sinken en sourcen

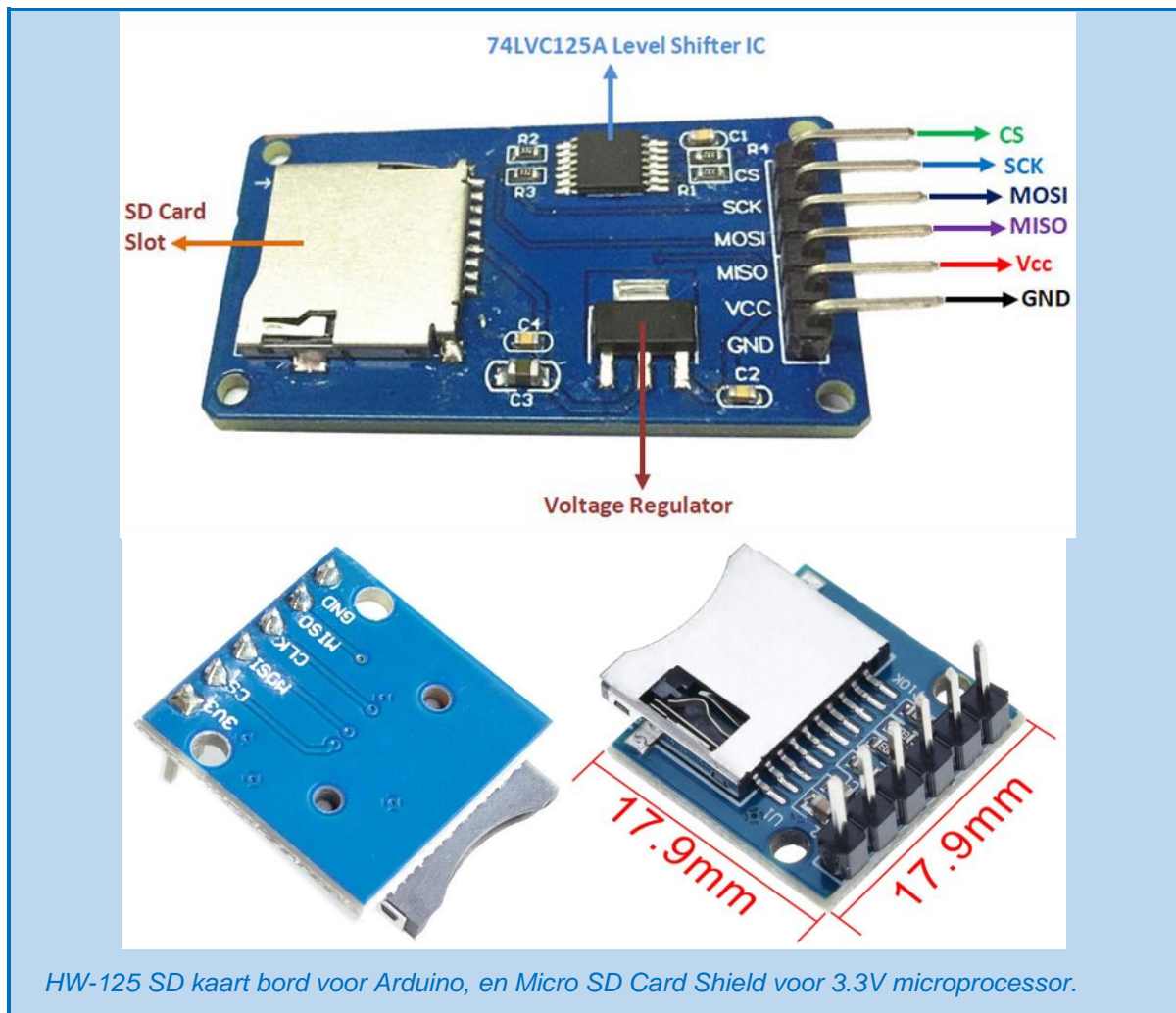
Wanneer in een circuit de Pico zelf een bron van stroom is voor een perifeer onderdeel, dan spreekt men over 'sourcen'. De totale stroom die je kan sourcen met 1 Pico is **50 mA** (max 12 mA per pinnetje, maar let dus op dat totaal niet boven 50 mA komt).

Wanneer de stroombron een extern apparaat is, dan dient de Pico als eindpunt van de stroom via de GND pinnetjes. Dit is 'sinken'. Er zijn echter 8 GND pinnetjes die hiervoor kunnen dienen. Bij sinken kunnen we dan ook maar een totaal van **50 mA** (max 12 mA per pinnetje) stroom ontvangen.

EXTENSIE BORDEN – SD CARD BORD

SD Card shield

In deze vorming beschik je over een extensie bord waarin een SD kaartje past. Dit hebben we nodig om meetdata op te slaan. De Pico heeft een intern geheugen, maar daar valt niet zo gemakkelijk mee te werken, en wordt al gebruikt om je code op te slaan. Veel geheugen is ook niet persistent, dat betekent dat de data verdwijnt bij een power cycle, en je ze dus makkelijk kwijtraakt. Daarom is een SD kaart noodzakelijk.



Meer uitleg over het gebruik van dit bord vind je verderop bij de oefeningen.

LiPo Batterij bord

Optioneel kun je ook een bord voorzien dat toelaat een LiPo batterij te gebruiken. Dit bord laat dat toe via usb de LiPo terug te laden. Dit type borden is ideaal als je een gesloten geheel wil maken van je satelliet.

RTC-bord

Een beperking van de Pico is dat we niet de exacte datum kennen, terwijl bij meetgegevens de juiste datum en uur kennen een grote meerwaarde is. Indien je juiste tijd wil kennen, kun je een Real Time Clock extensiebord voorzien, bijvoorbeeld de DS3231. Dankzij een coin cell batterij zal deze, eenmaal juist geconfigureerd, de tijd correct blijven opslaan, en kun je de tijd mee opslaan bij je meetgegevens.

STROOM CONTROLEREN

De stroomsterkte begrenzen

De datasheet van de Pico vermeldt enkele belangrijke **beperkingen** in verband met **stroomsterkte** in de bijhorende datasheet:

Absolute Maximum Ratings - the point where damage will start to happen

- DC Current per I/O Pin 12.0 mA
- DC Current VCC and GND Pins..... 50.0 mA voor alle pinnen samen

Deze maximale stroomsterktes moeten gerespecteerd worden, anders zal de microcontroller beschadigd worden.

Om zeker te zijn dat de maximale waarden niet overschreden worden, moet er gewerkt worden met weerstanden in het circuit, in serie met de verbruiker (LED, Sensor, ...). Een dergelijke weerstand heet een **voorschakelweerstand**.

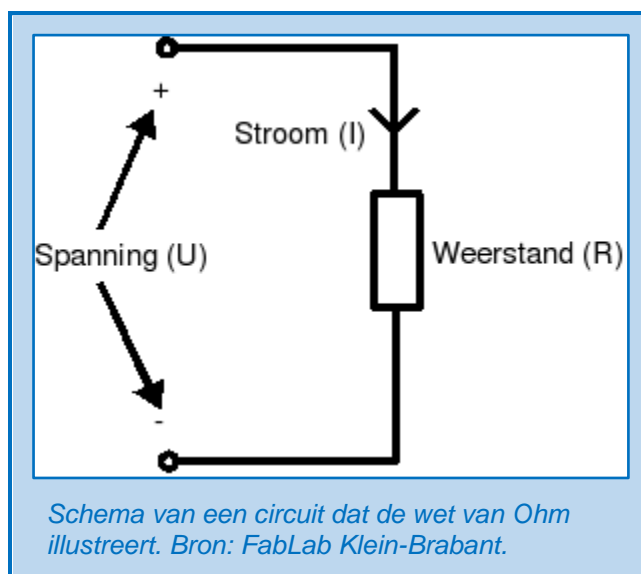
Hoe bereken je de voorschakelweerstand?

Uitgangspunt bij de berekeningen is de **wet van Ohm**:

$$U = I \times R$$

U is de spanning in volt,
I is de stroom in ampère,
R is de weerstand in ohm.

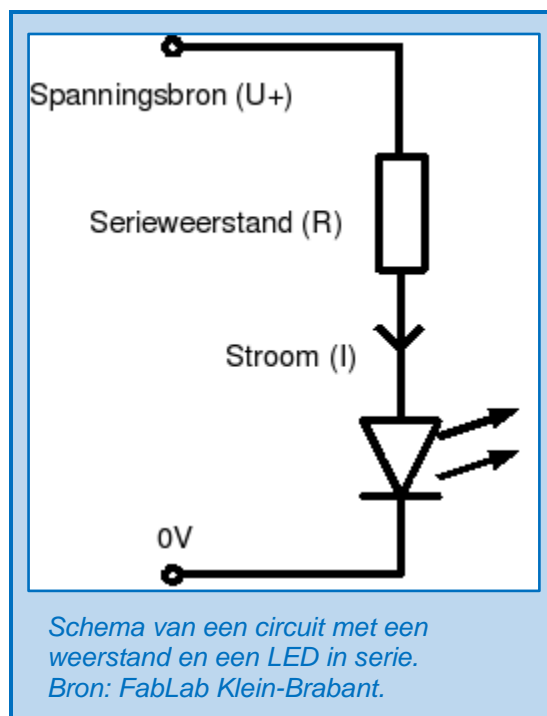
Deze formule vertelt je de spanning over een bekende weerstand als er een bekende stroom doorheen wordt gestuurd.



Maar wat te doen als je niet de spanning wilt weten, maar de weerstand, of de stroom? Geen probleem, met een beetje brugklaswiskunde kan de wet van Ohm worden omgerekend tot de volgende formules:

$$I = U / R \quad \text{en} \quad R = U / I$$

Het vorige schema geeft de situatie aan voor de wet van Ohm, maar er ontbreekt nog een stroomverbruiker: **de LED**. Deze laatste wordt in serie geschakeld met de weerstand.



Hierbij is de **volgorde** van led en weerstand niet van belang, zolang de led maar in de juiste polariteit is aangesloten. Dat laatste betekent dat de +pool van de led moet aangesloten worden op de +pool van de spanningsbron, en idem met de -pool.

Verder moet de **spanningsbron** een **hogere spanning** leveren dan de nominale spanningsval van de LED, anders kan er al helemaal geen stroom lopen. Met “nominale spanning” bedoelen we de effectieve spanning die continu op de LED staat om de LED te laten functioneren.

In dit **voorbeeld** gaan we uit van een rode LED met een nominale spanningsval van 1,9 volt. Als we dus een voedingsspanning U_+ van 3,3 volt nemen, zou het goed moeten gaan, want 3,3 V is groter dan 1,9 V. Als gewenste stroom I kiezen we 12 mA. Nu moeten we alleen nog de weerstand berekenen.

Nu zou je misschien denken dat je gewoon de waarden voor spanning (3,3 V) en stroom (0,012 A) in de wet van Ohm kunt invullen, maar dat is niet helemaal juist. Je moet nog rekening houden met de **nominale spanningsval over de LED**, die we U_{led} noemen. Deze laatste spanningsval moet van de voedingsspanning U_+ worden afgetrokken om de spanning over de weerstand R te krijgen. In formulevorm ziet dit er zo uit:

$$R = (U - U_{led}) / I$$

Nu kun je alle getallen invullen:

$$R = (3,3 - 1,9) / 0,012 = 1,4 / 0,012 = 167 \text{ ohm}$$

Je neemt dus gemakshalve een weerstand van **180 ohm** (een algemeen verkrijgbare waarde) als **voorschakelweerstand**. Deze weerstand is hoger dan de berekende 167 ohm, dus de stroom zal naar verwachting wat lager zijn. Maar dit is geen probleem, want de lichtopbrengst van een LED hoeft vrijwel nooit een erg nauwkeurige waarde te hebben.

We willen zoveel mogelijk controle over stroomsterktes. Dus, voor de aardigheid kunnen we nog even **uitrekenen** wat die **stroom** dan eigenlijk is bij 180 ohm in plaats van 167 ohm. Ook bij deze berekening moeten we weer de nominale spanningsval van de LED aftrekken van de voedingsspanning:

$$I = (U - U_{\text{led}}) / R = (3,3 - 1,9) / 180 = 7,78 \text{ mA}$$

Je ziet, de stroom is wat lager dan de gewenste 12 milli-ampère, maar dat is niet erg. Het verschil in lichtopbrengst zal nauwelijks zichtbaar zijn.

LET OP!

Goed, je kunt nu de benodigde voorschakelweerstand voor een LED berekenen, maar er zit een addertje onder het gras! Als de spanningsval over de weerstand (**$U - U_{\text{led}}$**) **erg klein** wordt ten opzichte van de voedingsspanning, kan een kleine afwijking in de voedingsspanning of de nominale spanningsval over de LED een grote afwijking in de stroom veroorzaken.

Stel, je hebt een blauwe LED (met U_{led} gelijk aan 3,2 volt), en een voedingsspanning van 3,3 volt. Als je een stroom van 12 mA wilt hebben, bereken je de voorschakelweerstand dus als volgt:

$$R = (3,3 - 3,2) / 0,012 = 8,3 \text{ ohm}$$

Je neemt dus een weerstand van 10 ohm, waarbij de stroom $(3,3 - 3,2) / 10 = 10 \text{ mA}$ wordt. Prima ... of toch niet?

Laten we eens kijken wat er gebeurt er als de spanning **een paar tiende volt afwijkt**. Bij een spanning van 3,2 volt (dus 0,1 volt onder de verwachte voedingsspanning) zit je al op de nominale spanningsval van de LED. Het ding zal dan **amper oplichten** door de weerstand die er ook nog is. Nog erger wordt het als de spanning 0,2 volt hoger wordt; de stroom wordt dan $(3,5 - 3,2) / 10 = 30 \text{ mA}$.

Oeps, dit is meer goed voor de meeste LEDS EN **boven de maximale stroom** die de Pico op een IO pin kan leveren!

Weerstanden combineren

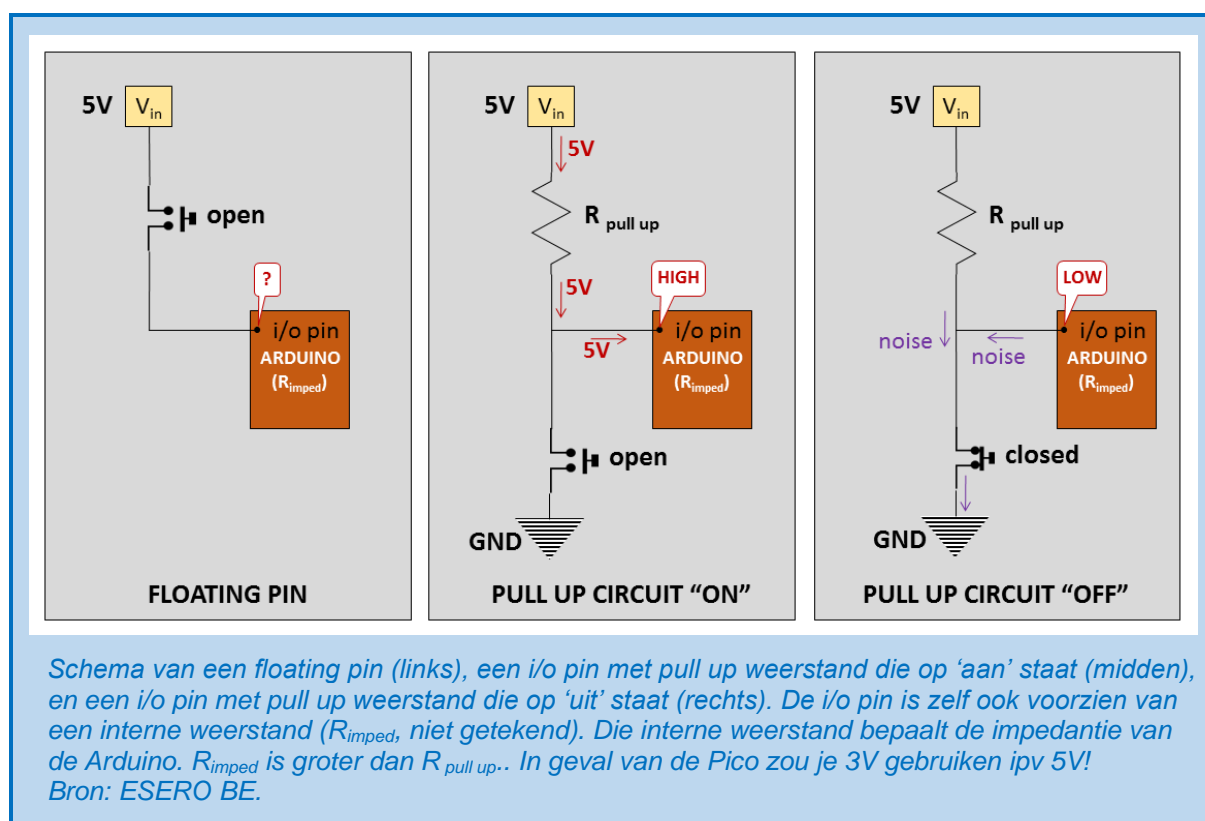
Als een weerstand met de gewenste waarde niet voorhanden is, dan kan je weerstanden in serie of parallel schakelen. De totale weerstand bereken je dan als volgt:

- Weerstanden in serie: $R_{\text{totaal}} = R_1 + R_2 + R_3 \dots$
- Weerstanden in parallel: $1/R_{\text{totaal}} = 1/R_1 + 1/R_2 + 1/R_3 + \dots$

Pull up weerstand

In normale omstandigheden zijn we omgeven door elektrische velden en statische elektriciteit. Deze komen van talrijke apparaten rondom ons (mobiele telefoons bijvoorbeeld). Deze kleine stroompjes vinden hun weg in de i/o pins van de microcontroller. Het zijn allemaal **stoorstroompjes**, die ervoor zorgen dat er dikwijls signalen zijn in de pins wanneer de batterij of andere stroombron uitgeschakeld is (noise of ruis). Wanneer onze hoofdschakelaar uit staat, zijn er dus toch i/o pins die een signaal doorgeven aan de microchip. Deze pins noemen we floating pins.

Om deze bron van ruis te vermijden, gaan we tussen de batterij en de i/o pin een **ziesprong naar 5V** voorzien waar ongewenste stroom kan wegvloeien. Uiteraard moet dan wel een weerstand voorzien worden tussen de batterij en de GND, om te verhinderen dat de batterij leegloopt. Het concept wordt hieronder uitgelegd.



- Wanneer de schakelaar simpelweg tussen de batterij (3 V) en de i/o pin zit, dan veroorzaken stoorstroompjes een onzekere toestand op de i/o pin. De input spanningswaarde op deze pin varieert daardoor ('floating'), en de digitale waarde kan daarom afwisselend 'HIGH' of 'LOW' zijn. Vandaar het vraagteken bij de **floating pin** hierboven.

We verbinden nu de batterij met de GND en de i/o pin. Op deze verbinding zetten we een drukknop en een weerstand.

- Als we deze drukknop niet indrukken, dan kan de stroom enkel naar de i/o pin lopen. Dan is de waarde van de **i/o pin "HIGH"** of 3 Volt.

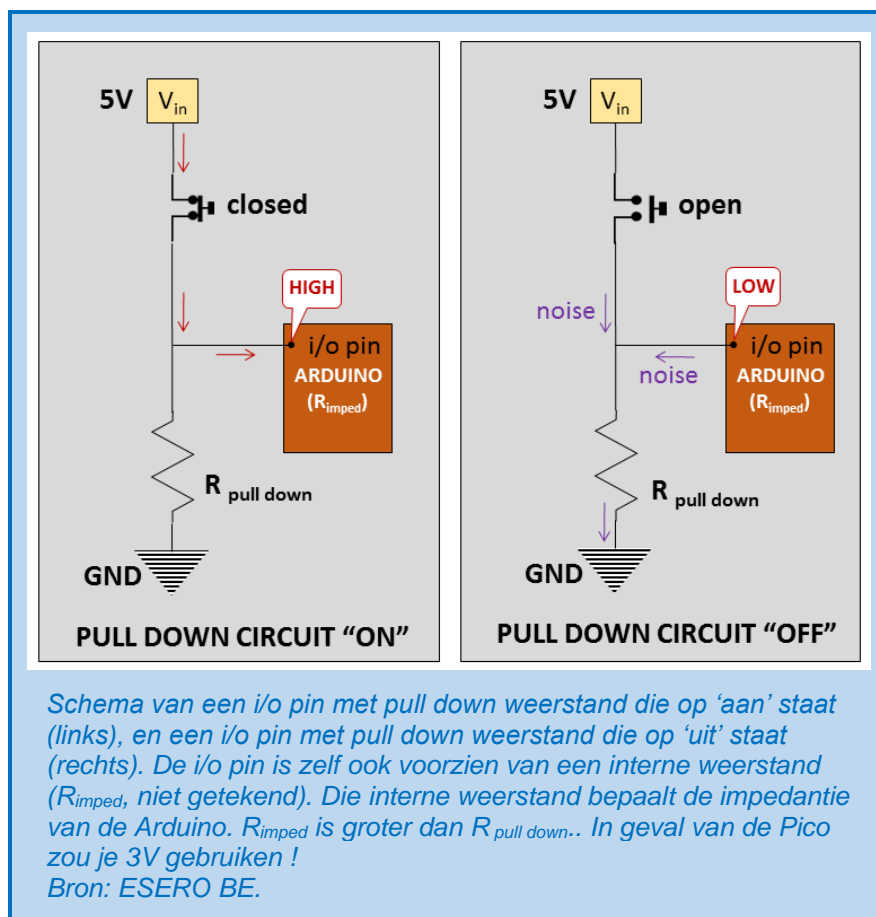
- Als de schakelaar dicht staat, dan loopt de stroom van de batterij naar de aarde GND. De weerstand $R_{\text{pull up}}$ zorgt ervoor dat de batterij niet leegloopt. Een interne weerstand in de microchip (hoogohmige R_{imped}) zorgt ervoor dat de stroom wel degelijk naar GND loopt, en niet naar de Pico. De logische waarde van de **i/o pin** is dan **“LOW”** of 0 Volt.

In deze opstelling moeten we dus de schakelaar uitzetten om de Pico van stroom te voorzien. De weerstand die we hier gebruiken is de ‘pull up’ weerstand. Hij zorgt ervoor dat de i/o pin de hoogste waarde heeft (HIGH, 3 V) in plaats van een ‘floating’ waarde. De waarde wordt als het ware opgetrokken (pull up).

Tegenwoordig zijn **pull up weerstanden ingebouwd** in de Pico pins. Je hoeft ze dus zelf niet te solderen in het circuit. Met de code in de software kan je de ingebouwde pull up weerstanden inschakelen of uitschakelen.

Pull down weerstand

Je kan het bovenstaande probleem ook anders oplossen. We zetten nu een weerstand tussen de i/o pin en de GND, dit noemt een pull down.



Als de drukknop open staat bij deze opstelling, dan zal er op de i/o pin zeker 0 Volt staan. De stoorstroompjes worden immers via de $R_{\text{pull down}}$ naar de GND geleid. Deze weerstand zal de de i/o pin dus **omlaag brengen (pull down)** naar de logische staat **“LOW”**.

Zowel een pull up als een pull down weerstand hebben als functie Pico pinnen op voorspelbare waarden LOW (0V) of HIGH (3V) te houden, en ‘floating’ te vermijden.

3 Software

3a Thonny

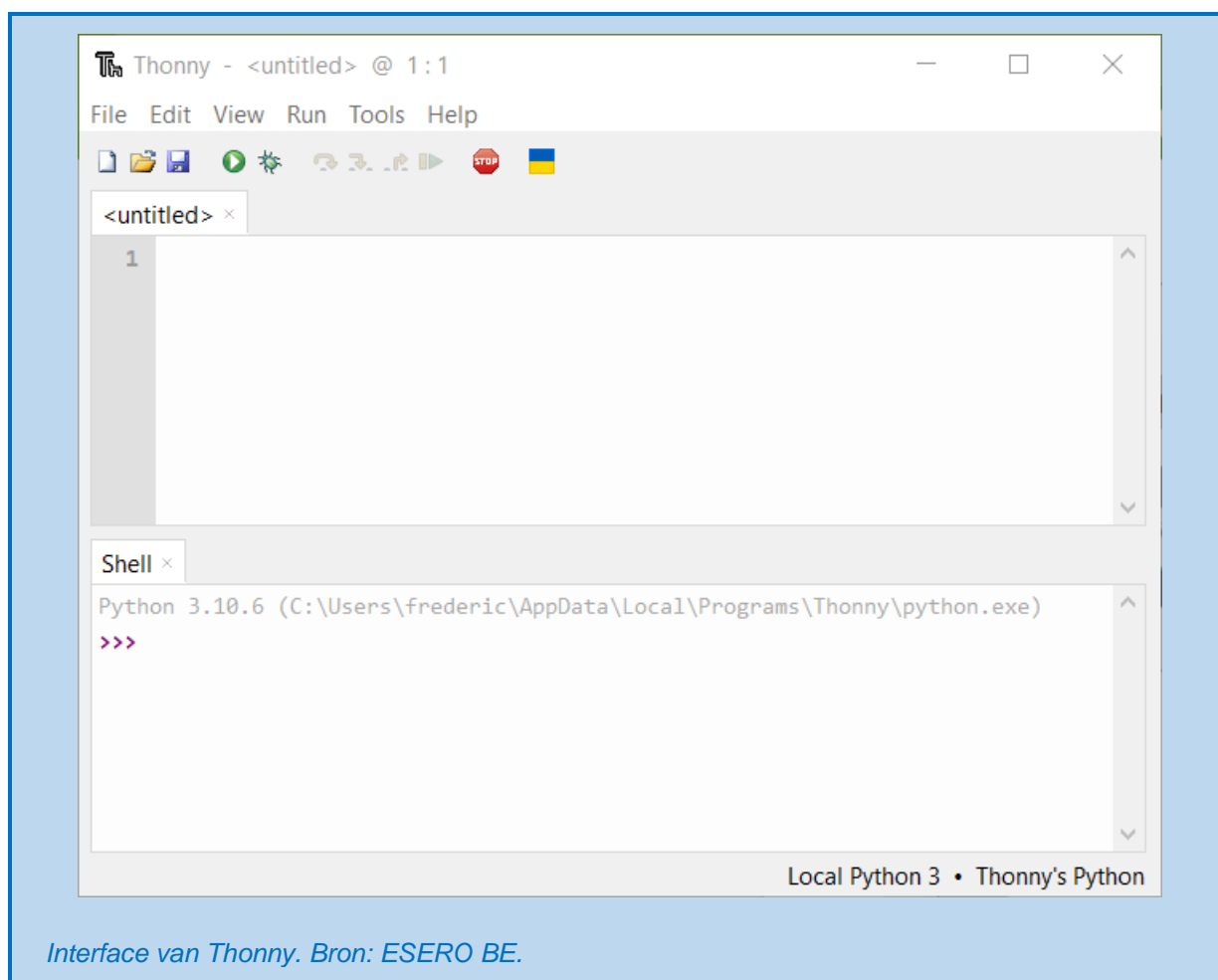
Download

Python is een taal voor algemeen gebruik die wordt gebruikt in een grote verscheidenheid aan toepassingen (datawetenschappen, kunstmatige intelligentie, statistiek, ...), terwijl MicroPython specifiek is ontworpen voor microcontrollers zoals de Raspberry Pi Pico die in ons project wordt gebruikt.

Om onze code in MicroPython-taal te bewerken, uit te voeren en te debuggen, installeren we een Integrated Development Environment (IDE) genaamd Thonny, beschikbaar op thonny.org.

De interface

Open Thonny vanuit uw applicatiestarter. Het zou er ongeveer moeten uitzien zoals in volgende figuur. Indien rechtsonderaan geen Python versie aangeduid is, dien je deze eerst te selecteren door te klikken rechtsonder.

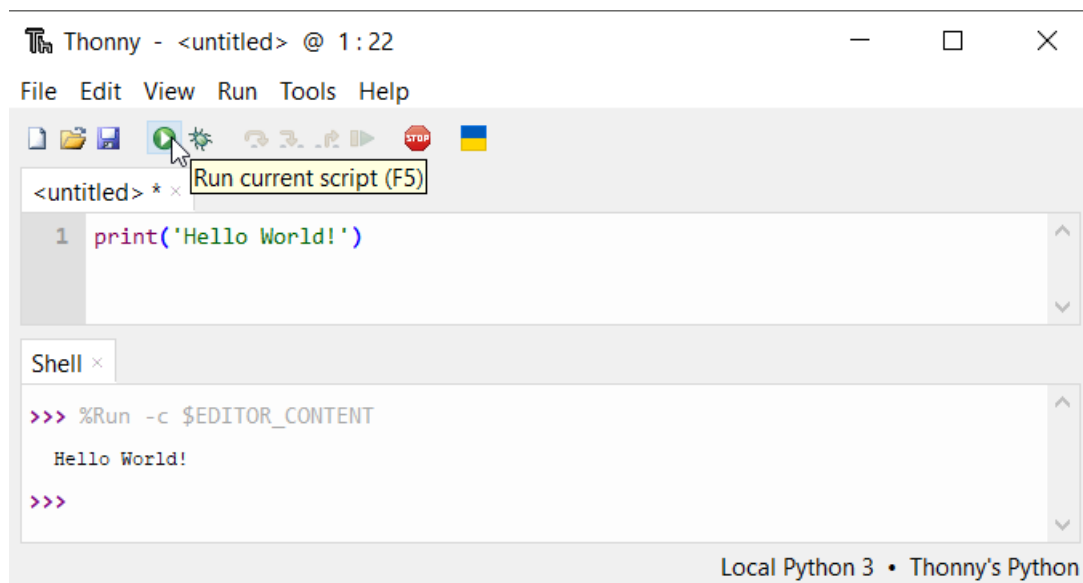


Code Editor en Shell

Je kan Thonny gebruiken om standaard Python-code te schrijven. Typ het volgende in het bovenste code editor venster en klik vervolgens op de knop Uitvoeren.

```
print("Hello, World!")
```

Na het drukken van de Uitvoeren knop zie je het resultaat in het "Shell" scherm

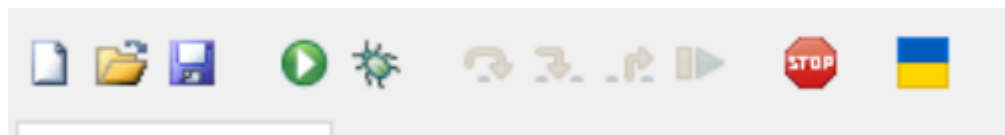


Menubar

Onder de openklapmenu's van de menubalk kan je elke toepassing terugvinden die de software aanbiedt. De meest gebruikte zijn echter gemakkelijker te gebruiken via een knop op de toolbar.

De opties onder de verschillende menu's spreken meestal voor zich. Ze worden enkel besproken in deze cursus in de mate dat we ze nodig hebben voor bepaalde oefeningen (zie verder).

Toolbar



Code venster

Hierin schrijf je de volledige code.

De software verandert automatisch de tekstkleur wanneer bepaalde codes herkend worden als 'geldige' commando's of functies.

Je kan ook waar je maar wilt commentaar erbij schrijven in gewone schrijftaal, meestal over wat de code precies doet. Er zijn namelijk symbolen die de software duidelijk maken dat je uitleg wilt bijschrijven die zelf geen code is. Deze symbolen vertellen de software: “wat ik nu ga schrijven is geen code, en moet dus niet naar de compiler gestuurd worden.” Met compiler bedoelen we: het vertaalprogramma die je code omzet in commando's die de Pico begrijpt na uploaden.

Als je **1 regel commentaar** wilt toevoegen, dan open je deze commentaar met het hashtag symbool (#), en je eindigt door op enter te duwen (nieuwe regel):

```
# Dit is commentaar
```

Als je dit gebruikt, zal je zien dat de software je bijgevoegde commentaar (inclusief de hashtag zelf) een andere kleur krijgen. Dit toont aan dat de software het herkend heeft als commentaar.

Als je **meerdere regels commentaar** wilt toevoegen, dan gebruik je 3 aanhaalttekens (""") om het commentaar te openen, en opnieuw (""") om te sluiten:

```
Code
"""
Dit is commentaar
geschreven over
meerdere lijnen
"""
Nog meer code
```

In Python-syntax voeg je eigenlijk een tekstblok toe over meerdere regels, zonder dit toe te kennen aan een variabele. Dit tekstblok doet dus niets, en wordt als commentaar beschouwd.


Voor alle andere weetjes die je nodig hebt om een code te schrijven, verwijzen we naar een de oefeningen verder in deze cursus (“eenvoudige codes”).

Als je programmeert, dan moet je de regels volgen van de gebruikte programmeertaal. Deze set van regels heet “**syntax**”. Wanneer je de regels niet correct toepast, dan krijg je van de software een foutmelding op de plaats waar het fout ging.

Er zijn veel syntaxregels, maar in dit gedeelte overlopen we er slechts enkele die voor elk programma van toepassing zijn. Je leert deze natuurlijk vooral gebruiken door zelf eenvoudige programmeer-oefeningen te maken (zie verder). Voor alle andere regels en mogelijkheden verwijzen we naar de oefeningen verder in de cursus.

Python Cheat Sheet

Hier geven we een handig overzicht van de python programmeertaal.



core
ELECTRONICS

Python Quick Reference

Maths

Addition 1+3
Exponent 2**3
Subtraction 1-3
Modulus 2%3

Multiplication 1*3
Integer Division 2//3
Division 1/3

Logic

Return True True
Return False False
not False not True
(True | False) # or (False | False) # or
(True & True) #and (True & False) #and

Compare

Return True if comparison holds
x > y # Greater than
x < y # Less
x <= y # Less or equal
x >= y # Greater or equal
x == y # Equal
x != y # Not equal

If..else

if(x > y):
 print("x is larger ")
elif(x < y):
 print("y is larger ")
else:
 print("Equality! ")

Functions

define a function
def threeTimes(xIn):
 return xIn*3
Call with input=5
threeTimes(5)
Define Lambda function
t = 6
tMult= lambda a : a*t
Call with a = 5 t=6
tMult(5)
t = 1
Call with a = 5 t=1
tMult(5)

Variable

Assign an number
x = 5
Assign String variable
maker = "I am a maker "
Make a number into a string
str(x)
Combine strings
newString = maker + " More string"

Strings

Output a string
print("Hello World ")
output a number
print(x)
output string and number
print("hello "+str(x)+" world")
#input
myVariable = input("message")
#to input number eval output
myVariable = eval(myVariable)

Lists

Make a list
myList = ["item0 ", "item1 ", "item2 "]
Add item to list
myList.append("item3")
Select first item in a list
myList[0]
Select last item in the list
myList[-1]

Import

import time
Call function from time
time.sleep(1) # Wait for 1 second
Import single function
from math import sqrt
Call
sqrt(2)

Loops

While less than
while(x < y):
 x += 1
 print(x)
While true with break
while(True):
 x += 1
 print(x)
 if(x>y):
 break

For loop with default count
for i in range(1,10):
 print(2**i)
For loop with custom count
for i in range(10,-10,-1):
 print(2**i)
For loop over list elements
for i in myList:
 print(i)

Classes

define Class
class myClass:
 # Initialisation
 def __init__(self,in1,in2):
 self.input1 = in1
 self.input2 = in2
 def add(self): # Class method
 return self.input1+self.input2
Run the program
myObject = myClass(1,2)
myObject.add ()

forum.core-electronics.com.au

Let op: de programmeertaal is **hoofdlettergevoelig**.

De Python Programmeertaal

Python is een populaire programmeertaal. Ze is heel krachtig, maar terzelfdertijd een ideale programmeertaal voor de beginner dankzij de heldere structuur. Het is niet onze bedoeling in deze cursus in detail aan te brengen. Hiervoor verwijzen we je graag naar online boeken, zoals bijvoorbeeld nl.wikibooks.org/wiki/Programmeren_in_Python.

Belangrijk om weten is dat Python niet werkt met een leesteken zoals { } om een codeblok aan te duiden. Python gebruikt indentatie voor code blokken. Je ziet dit in de cheat sheet bij de while loop en andere structuren zoals de for loop:

```
j = 0
for i in range(10):
    j += 2
    print(j**i)
```

Het code blok in de for loop heeft hier een indentatie van 4 spaties. Alle code lijnen in een blok **moeten** dezelfde indentatie hebben, zonet krijg je een syntaxfout. Het is daarom belangrijk een python editor te gebruiken als je python code schrijft, waarbij een tab automatisch omgezet wordt in 4 spaties, en tabs en spaties dus niet voor verwarring kunnen zorgen.

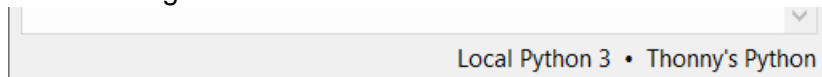
Als je compactere code wil schrijven kun je 2 spaties gebruiken om van code blok te veranderen, maar wees ervan bewust dat je dan al moeilijker de structuur van je code zult zien.

INSTALLEER MICROPYTHON OP DE PICO

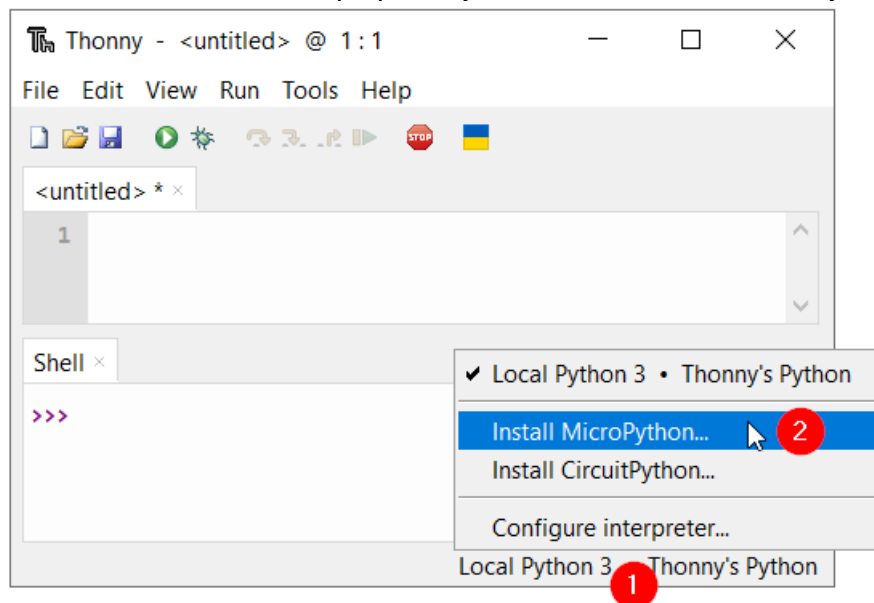
Uw nieuwe Raspberry Pi Pico heeft MicroPython nodig om uw software uit te voeren. Begin met het loskoppelen van de micro-USB-kabel van uw computer, maar laat deze aangesloten op uw Pico.

Druk op de BOOTSEL-knop en houd deze ingedrukt terwijl u het andere uiteinde van de micro-USB-kabel op uw computer aansluit. Eenmaal herkend door je PC, kun je de BOOTSEL loslaten

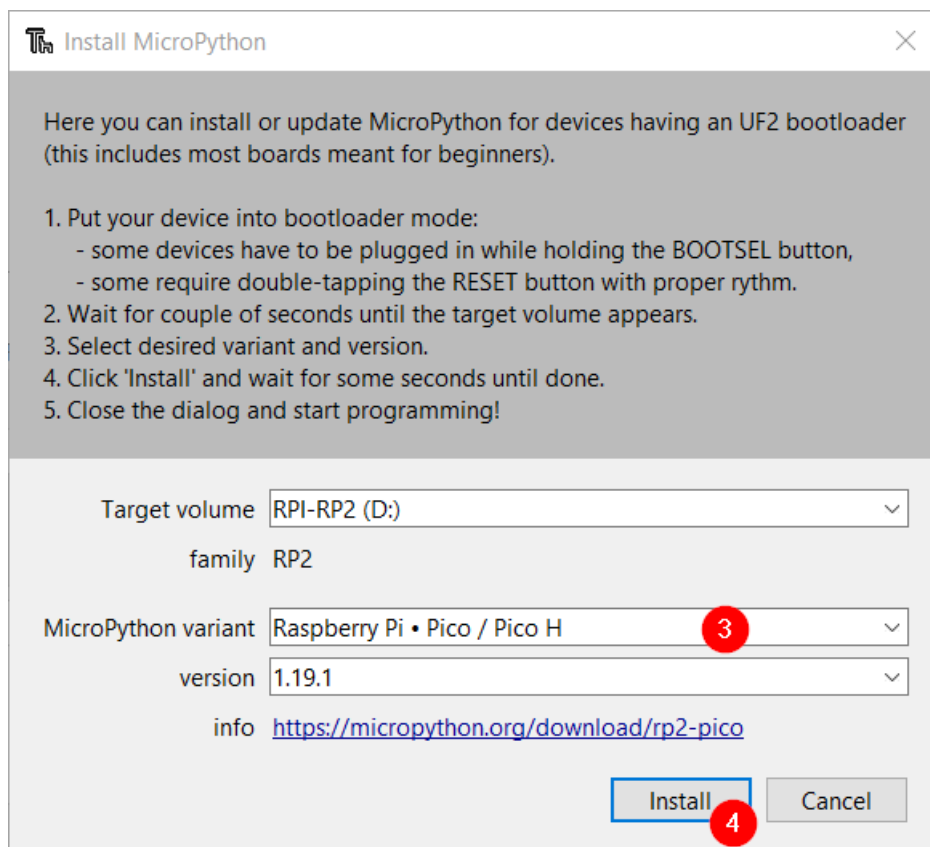
In de rechterbenedenhoek van het Thonny-venster ziet u de versie van Python die u momenteel gebruikt.



Klik met de linkermuisknop op de Python-versie en kies 'MicroPython installeren...'

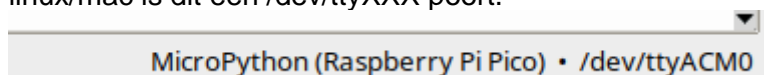


Er verschijnt een dialoogvenster om de MicroPython-firmware op uw Pico te installeren. Selecteer de juiste MycroPython-variant en klik op de knop Installeren. Wij gebruiken in de les de Pico H, dus zorg dat je deze selecteert als variant!



Wacht tot de installatie is voltooid en klik op de knop Sluiten.

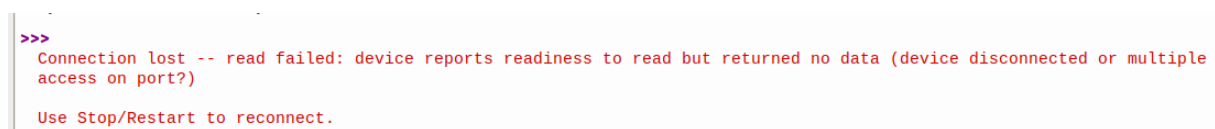
Als je verbonden bent met de pico via Thonny, zie je dit rechts onderaan, samen met de poort waarmee de PICO in verbinding staat met je PC. Op windows is dat bv COM7, op linux/mac is dit een /dev/ttyXXX poort.



U hoeft de firmware niet elke keer dat u uw Pico gebruikt bij te werken. De volgende keer kunt u hem gewoon op uw computer aansluiten zonder op de BOOTSEL-knop te drukken.

Pico verbinding stoppen - starten

Je kan de usb kabel verwijderen om de verbinding te verbreken. Je krijgt dan in de Shell een foutmelding te zien



Als je de verbinding wil herstarten, connecteer de Pico weer met je PC, en druk op de stop of start knop in de toolbar. In de Shell verschijnt of de verbinding gelukt is. Je ziet je versie van MicroPython, en het type microprocessor.

```
Shell X  
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040  
Type "help()" for more information.  
>>>
```

Seriële output - REPL interface (Shell)

Wie een alternatieve microcontroller zoals de Arduino kent, weet dat daar vaak via serial gecommuniceerd wordt tussen de microcontroller en je PC. Dit is **niet** mogelijk met de Pico op de manier waarbij wij gebruik maken van de Pico. Er is een seriële verbinding tussen je PC en de Pico, maar die wordt gebruikt door Thonny om code op te laden, en het debuggen mogelijk te maken. Er dient dus op een andere manier gewerkt te worden als je je code wil debuggen.

Dit kan via de REPL interface (Shell). REPL staat voor Read Evaluate Print Loop. Als Thonny verbonden is met de Pico, kun je code laten lopen op de Pico via de shell, en de antwoorden bekomen.

Doe volgende code in de shell:

```
print("Hallo, Wereld!")
```

en druk op Enter. De code **wordt op de Pico uitgevoerd**, en je krijgt het resultaat:

```
Shell X  
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040  
Type "help()" for more information.  
>>> print("Hallo, Wereld!")  
Hallo, Wereld!  
>>>
```

3b Programmeren van de Pico

Zorg ervoor dat uw Raspberry Pi Pico nog steeds op uw computer is aangesloten. Selecteer de MicroPython (Raspberry Pi Pico) tolk rechtsonder in Thonny indien dit niet langer het geval is.

De Pico is een open-source ontwikkelbord, waarmee het mogelijk (en relatief gemakkelijk) wordt om informatica te koppelen aan fysieke objecten. Het is dus een apparaat om fysieke informatica makkelijk en toegankelijk te maken. De gebruiker moet echter leren om python code te lezen en te schrijven.

Om de basis daarvan aan te leren, doen we hieronder enkele [eenvoudige oefeningen](#).

EENVOUDIGE OEFENINGEN : Interne LED laten knipperen

Opdracht

We gaan de interne Pico LED laten aan en uit knipperen. We gebruiken dit om te trainen met timing en niet blokkerende code

Hardware

Voor deze oefening heb je enkel de Pico nodig.

Code 1: LED aansturen

MicroPython voegt hardware-specifieke modules toe, zoals een `machine`, die u kunt gebruiken om uw Raspberry Pi Pico te programmeren. Laten we een `machine.Pin` object maken om te spelen met de ingebouwde LED, die verbonden is via GPIO-pin 25. Als u de waarde van de LED instelt op 1, gaat de ingebouwde LED branden.

Voer de volgende code in voor de tekstverwerker, zorg ervoor dat u na elke regel op Enter tikt.

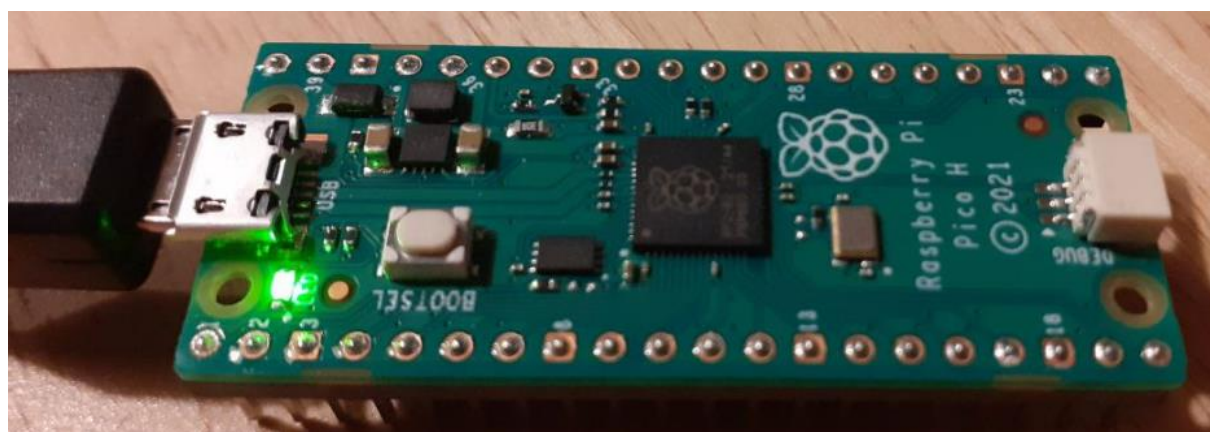
```
from machine import Pin
led = Pin(25, Pin.OUT)
led.value(1)
```

De eerste lijn is het importeren van een Object uit de module `machine`, welke manieren bevat om de Pico aan te sturen. We laden het object `Pin`, welke toelaat een pin van de Pico aan te sturen.

Op de tweede lijn maken we een object `Pin`, gelinkt aan GPIO pin 25, en definiëren we dat deze een digitale output pin dient te zijn.

We zetten de waarde van de pin op aan = 1 = 3.3 V (HIGH). Hierdoor zal de interne LED aangaan.

Druk in de toolbar op de “Run” knop, en verifieer dat de Pico LED brandt.



Je ziet dat we via de “Run” toets code uitvoeren in de MicroPython geïnstalleerd op de Pico. Dit is **NIET** code die opgeslagen wordt op de Pico, wel code die we uitvoeren op de Pico. Je kan via de Shell verder interageren met de Pico, waar nu in de MicoPython omgeving het object `led` gekend is. Tik in de **shell**:

```
led.value(0)
```

Je stelt vast dat deze code werkt, en de LED uitgaat. Zet nu de LED weer aan via de Run knop.

Oefening: Voer in de shell volgend commando uit, en probeer te begrijpen wat het commando doet:

```
led.toggle()
```

Oefening:

Trek nu de USB kabel uit. Na 5 seconden plug je de USB weer in. Je stelt vast dat de LED uitblijft. Herconnecteer Thonny met de Pico door de stop of start toets in de toolbar te klikken. Doe dan in de shell:

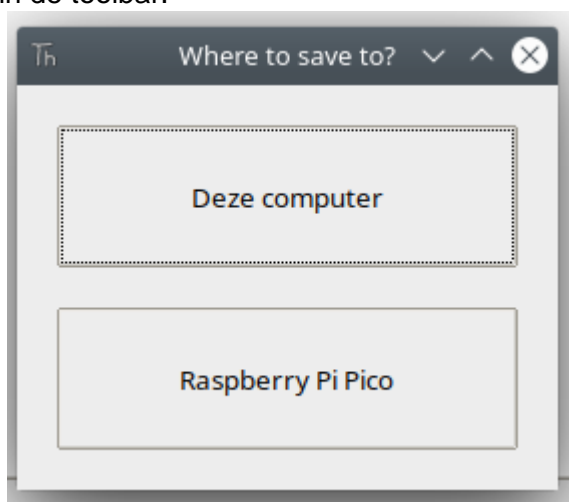
```
led.value(1)
```

Je stelt vast dat deze code faalt met **een fout**. Waarom?

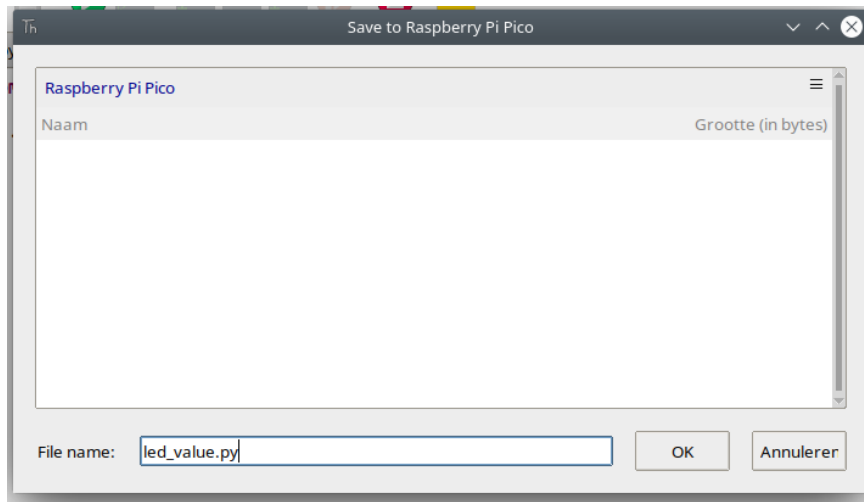
Oplossing: Het object led bestaat nog niet op de Pico, want je code werd nooit uitgevoerd! Voer de code opnieuw uit, zodat de LED weer gaat branden. Nu zal het commando in de shell wel weer werken.

Code opslaan en uitvoeren op de Pico

Als we code schrijven willen we deze opslaan. Er zijn **3 manieren** waarop je kan opslaan als een Pico verbonden is met Thonny, waaruit je moet kiezen bij het drukken op de save knop in de toolbar.



1. **Opslaan op je PC.** Doe dit altijd. Zo zorg je dat al je oefeningen opgeslagen zijn op een manier waarop je ze later opnieuw kunt openen en verder verfijnen. Sla deze oefening op als led_value.py
2. **Opslaan op de Pico.** Doe je CTRL+SHIFT+S, kun je opnieuw opslaan. Kies nu voor opslaan op de Pico.



Je kan meerdere python scripts opslaan op de Pico. Als je evenwel de Pico loskoppelt, en weer connecteert, zul je zien dat de LED niet brandt, dus deze code wordt niet uitgevoerd.

3. **Opslaan op de Pico als main.py.** Save de code nu nog een derde keer, maar sla deze nu op op de Pico met als naam **main.py**. Het python script met deze naam wordt automatisch uitgevoerd bij opstarten van de Pico. Als je nu de Pico loskoppelt, en weer verbindt, zal de LED branden.

EENVOUDIGE OEFENINGEN : Een LED laten knipperen

Opdracht

We gaan een LED laten aan en uit knipperen.

Je zou deze oefening kunnen doen met de ingebouwde LED van de Pico.

Hardware

We hebben nodig:

- De Pico

Python Script Oplossing 1: programma onderbreken

Het volgende script is een mogelijke oplossing. We maken gebruik van de `toggle()` die we in de vorige oefening gezien hebben. Uitleg bij de code staat als commentaar bij de code

```
# bibliotheken importeren: we gebruiken een Pin en de sleep functie
from machine import Pin
from time import sleep
# led variabele aanmaken die output is op pin 25
led = Pin(25, Pin.OUT)

# Opgelet, 4 spaties niet vergeten voor de while lus
# Oneindige lus: ons programma stopt NOOIT !
while True:
    # Wijzig led status van led.value(0) naar led.value(1) en omgekeerd
```

```
led.toggle()  
# Laat het programma een halve seconden stoppen  
sleep(0.5)
```

Upload

Klik op Uitvoeren en uw programma zal de LED doen knipperen totdat u op de knop Stop klikt.

Oefening: Verander het knipperpatroon.

Nu je de code in de blink sketch volledig begrijpt, kan je gaan spelen met het aantal seconden in de sleep functie. Probeer maar. Kun je de LED zo vlug knipperen dat je het niet langer kan zien?

Python Script Oplossing 2: een Timer

Het programma werkt, maar heeft een groot nadeel: ons programma wordt in een slaap gestoken, wat betekent dat we gedurende die tijd niet langer sensoren kunnen uitlezen of reageren op bijvoorbeeld iemand die op een knop drukt.

Een betere manier van werken is dus de LED laten knipperen op een manier die toelaat om ook andere zaken te doen in je programma. De Timer helpt ons om dit te doen.

```
# bibliotheken importeren: we gebruiken een Pin en de Timer  
from machine import Pin, Timer  
  
# led variabele aanmaken die output is op pin 25  
led = Pin(25, Pin.OUT)  
# timer variabele maken  
timer = Timer()  
  
# Een functie definiëren met de naam "blink". Deze toggled de LED  
def blink(timer):  
    # Wijzig led status van led.value(0) naar led.value(1) en omgekeerd  
    led.toggle()  
  
# Configureer de timer om onze blink functie op te roepen om de  
# 500 milliseconden. Dit moet steeds opnieuw herhaald worden  
timer.init(period=500, mode=Timer.PERIODIC, callback=blink)
```

Upload

Test je code. De volledige "Timer" bibliotheekdocumentatie is te vinden in [de officiële MicroPython-documentatie](#).

In onze code werkten we met een periode van 500 ms, dit komt overeen met een frequentie van 2 Hz. Hz is immers Hertz, of ook wel aantal keer per seconde, en een periode van 500 ms is 2 keer per seconde. Oefening: Wijzig de code om in plaats van `period=500` de optie `freq` te gebruiken van de timer.

EENVOUDIGE OEFENINGEN : Interne temperatuur meten

Opdracht

We gaan de temperatuur van de Pico meten. Dit kan met de interne thermometer, maar let op, deze dient om de temperatuur van de microchip op te volgen, en is dus niet bruikbaar om de omgevingstemperatuur te meten.

Hardware

We hebben nodig:

- De Pico

U vraagt u misschien af op welke pin deze ingebouwde temperatuursensor is aangesloten? Deze ingebouwde temperatuursensor wordt aangesloten op een van de ADC's of analoog-naar-digitaal-omzetter. De temperatuursensor heeft geen fysieke pin op het bord, maar is toegankelijk als ADC4. Deze ingebouwde temperatuursensor werkt door een spanning aan de ADC4-pin te leveren die evenredig is met de temperatuur. Als je de datasheet van de Raspberry Pi Pico bestudeert, zul je zien dat een temperatuur van 27 graden Celsius een spanning van 0,706 volt levert.

In het RP2040 Pico-bord ondersteunen de ADC-pinnen 12-bits, wat betekent dat de waarde van 0 tot 4095 kan gaan. Maar de MicroPython-code kan de ADC-waarden schalen naar een bereik van 16 bits. Dus we krijgen effectief het bereik van 0 tot 65535. De microcontroller werkt op 3,3 V, wat betekent dat een ADC-pin een waarde van 65535 retourneert wanneer er 3,3 V op wordt toegepast of 0 wanneer er geen spanning is. We kunnen alle tussenliggende waarden verkrijgen als de spanning die op de pin wordt toegepast tussen 0 en 3,3 V ligt. Laten we de code eens bekijken en u krijgt het idee.

Python Script Oplossing

```
# bibliotheken importeren: we gebruiken een ADC Pin en de Timer
from machine import ADC
from time import sleep
# Maak variabele voor de Analoog naar digital conversie
# De temperatuur is via ADC 4.
TempSensor = ADC(4)
# Conversie factor van ADC meting naar 0-3.3V
conversion_factor = 3.3 / 65535
while True:
    #Doe meting en converteer naar gemeten volt
    data = TempSensor.read_u16() * conversion_factor
    #formule om de temperatuur te bekomen
```

```
temperature = 27-(data-0.706)/0.001721
#toon waarde temperatuur
print(round(temperature,1), 'C')
sleep(1)
```

Upload

Test je code. Je zal de temperatuur zien. Die ziet er wat te laag uit? Dit is het gevolg van het feit dat de spanning waarmee we werken niet perfect 3.3 V is. Je kan met een multimeter de 3.3 V pin meten, en dit al wat corrigeren naar de “echte” waarde.

Wijzig in de code de 3.3 in 3.27, en je zal waarden krijgen die al 4 graden hoger liggen. Deze grote afhankelijkheid van de referentiespanning maakt deze temperatuursensor niet echt geschikt voor wetenschappelijke metingen bij eenvoudig gebruik, maar het kan wel een snelle indicatie geven van de temperatuur rondom de Pico, zonder een aparte temperatuursensor te voorzien!

PS: er zijn verschillende geavanceerde mogelijkheden om betere ADC metingen met de Pico te bekomen, maar deze hebben allemaal diverse nadelen en zijn niet simpel in implementatie. Ze komen erop neer dat de referentiespanning stabiel gehouden wordt, een noodzaak voor stabiel gebruik van de ADC. Gebruik de temperatuursensor alvast enkel om benaderde waarden te bekomen.

Oefening

Wijzig de code naar een timer. Toon de temperatuur elke seconde in de shell, en doe ondertussen met een ander ritme de interne LED flikkeren.

EENVOUDIGE OEFENINGEN : Verkeerslicht

Opdracht 1

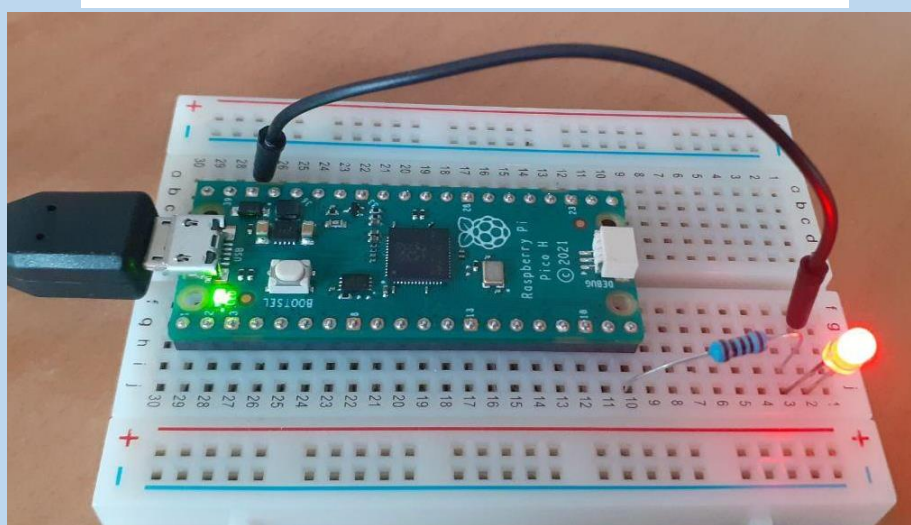
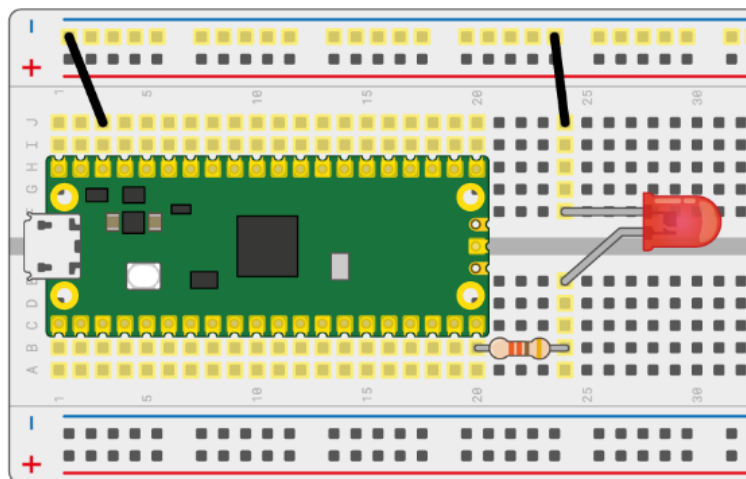
We maken een eerste fysische schakeling met een breadboard: maak een enkele rode externe LED die je aanstuurt GPIO 15.

Hardware

- Breadboard
- Rode LED
- 1 weerstand van 220 Ω
- de Pico
- Jumper kabeltjes

Circuit

Maak volgend circuit:



Circuit voor de oefening "Verkeerslicht - 1 LED". Bron: ESERO en Ingegno.

Python Script

Hergebruik 1 van je vorige scripts, maar wijzig de GPIO pin die je aanstuurt

```
led = Pin(15, Pin.OUT)
```

Problemen? Misschien werkt de jumper kabel niet goed? Om u wat hoofdpijn en tijd te besparen, raden we u ten eerste aan om al uw jumpers te testen met deze oefening door de 2 jumpers te vervangen door de andere jumpers die bij de kit zijn geleverd. Of gebruik een multimeter en controleer dat er geen weerstand is (goede jumper = 0 Ohm).

Nog problemen? Heb je de LED juist geconnecteerd? Lange beentje aan de zijde van de GPIO pin.

Opdracht 2

We gaan een verkeerslicht programmeren. Het verkeerslicht moet telkens groen en rood aangeven, en de overgang wordt aangekondigd met oranje. Zoals bij een echt verkeerslicht blijft deze cyclus zich eendeloos herhalen.

Hardware

- Breadboard
- Rode LED, oranje LED, groene LED
- 3 weerstanden van 220 Ω
- De Pico
- Jumper kabeltjes

Python script

Probeer dit via de vorige voorbeelden. Eenvoudigst is via sleep te werken, gezien de tijden van oranje verschillend zijn van rood en groen. Maar er zijn ook alternatieve mogelijkheden, bv via een timer die niet periodisch is, en telkens wijzigt in welke functie opgeroepen wordt:

```
timer.init(mode=Timer.ONE_SHOT, period=2000, callback=mijn_functie)
```

TEMPERATUUR METEN

Opdracht

Meet de temperatuur elke halve seconde met de meegeleverde DS18B20 sensor. Laat de gemeten waarden verschijnen in $^{\circ}\text{C}$ in de shell. Warm de sensor op met je hand (en/of koel hem af met ijs) om te testen.

Hardware

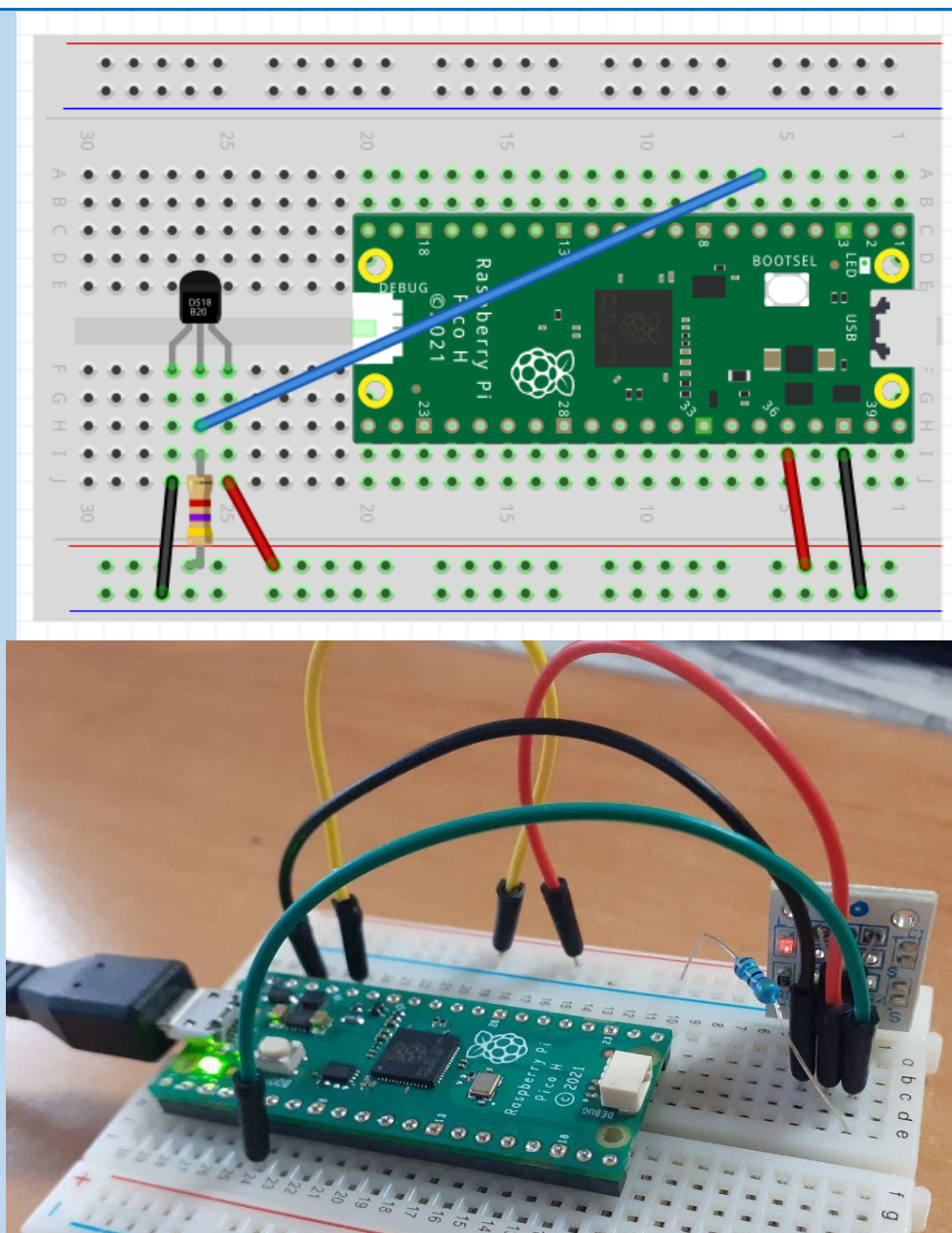
We hebben nodig:

- 1 pico
- 1 breadboard
- 1 temperatuursensor DS18B20
- 1 weerstand 4.7 k Ω (geel, paars, rood, goud)
- 5 jumper kabeltjes

Circuit en script

De pinout voor de DS18B20 werd eerder gegeven, en gebruiken we in onderstaand circuit. Let op indien de DS18B20 gesoldeerd is op een PCB, je moet dan verifiëren waar de GND en VCC connecties zijn met een multimeter. Bij de Velleman IO borden bijvoorbeeld, is de GND aangeduid met een -, en de onewire datalijn met een S.

Maak volgend circuit.



Circuit voor de oefening “De temperatuur meten”. Bron: Ingegno.

In deze sketch komen we enkele nieuwe commando's tegen:

- **Bibliotheken**

Dit zijn vooraf geschreven programma's die je downloadt/installeert, en waarvan je de commando's gebruikt. Normaal is er per sensor een bibliotheek beschikbaar om deze uit te lezen. Voor deze oefening zijn de bibliotheken die we nodig hebben reeds beschikbaar in micropython. We gebruiken:

- **onewire** : bibliotheek om communicatie te doen over een enkele draad, zoals gebruikt door onze sensor
- **ds18x20** : bibliotheek om de DS18B20 uit te lezen, met functies die de temperatuur kunnen bekomen.
- Deze bibliotheken zijn niet voor alle microprocessors beschikbaar, maar wel voor de Pico, zoals weergegeven in de [RP2 documentatie](#).

- **for loop**

Met dit commando bekom je een lus over alle waarden in een lijst. Zie de python cheat sheet voor overzicht van commando's. In python is de lijst een heel belangrijke programmaconstructie. Een lijst wordt aangeduid met vierkante haken: [...], met elk element gescheiden door een komma. Bij een for loop over een lijst zul je per doorlopen van de lus werken met 1 element van de lijst.

De lijsten zelf kunnen op vele manieren bekomen worden, bijvoorbeeld via de range functie. Test volgende code in de shell:

```
for getal in range(1, 11, 1):
    print(getal)
```

```
for getal in range(1, 11, 2):
    print(getal)
```

```
for getal in range(11):
    print(getal)
```

- **lijst inclusie**

In python kun je lopen over een lijst en deze omvormen naar een nieuwe lijst via lijst inclusie:

```
[functie(x) for x in lijst]
```

Probeer maar eens:

```
[x.upper() for x in 'help']
```

```
[x*2 for x in range(10)]
```

- **string join**

Je kan lijsten van tekst (strings) samenvoegen via de join functie

```
' '.join([x.upper() for x in 'help'])
```

```
'+'.join([x.upper() for x in 'help'])
```

Zorg dat je de 4.7 kOhm weerstand JUIST hebt aangesloten. Volgende script zal de temperatuur uitlezen, commentaar in de code maakt duidelijk wat er gebeurt:

```
# importeer de nodige bibliotheken
import machine, onewire, ds18x20, time

# data pin verbonden met GPIO4
ds_pin = machine.Pin(4)
# sensor object aanmaken via doorgeven van de juiste communicatielijn
(onewire over GPIO4)
```

```

ds_sensor = ds18x20.DS18X20(onewire.OneWire(ds_pin))

# bekom alle temp sensoren op de datalijn. Normaal maar 1 bij ons
roms = ds_sensor.scan()
# print hoeveel sensoren gevonden. len() bekomt lengte van een lijst
print('Found DS devices: ', len(roms))

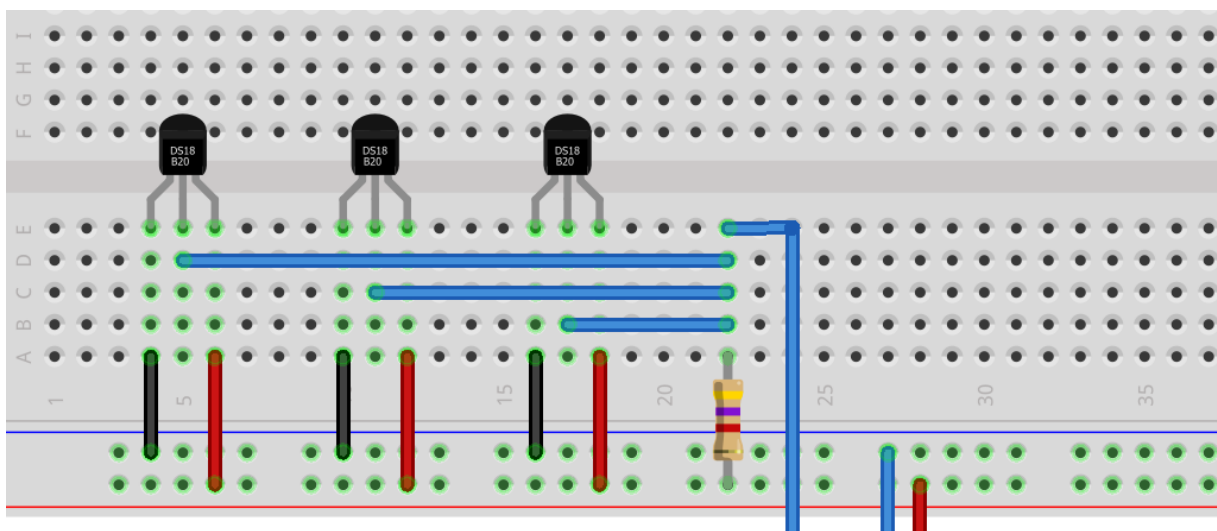
# oneindige lus voor programma dat nooit stopt
while True:
    # lees de temperatuur uit
    ds_sensor.convert_temp()
    # wacht 750 ms om tijd te geven om sensors uit te lezen
    time.sleep_ms(750)
    # print in de shell de temperatuur van elke sensor
    for rom in roms:
        # print de 64bit code van de sensor als hex string
        print('Device', ' '.join([hex(x) for x in rom]))
        # print de gemeten temperatuur
        print(ds_sensor.read_temp(rom))

    # wacht 5 seconden voor volgende meting te doen.
    time.sleep(5)

```

Meerdere temperatuursensoren

Wil je meerdere sensoren uitlezen, dan moet de data pin op dezelfde GPIO pin aangesloten worden. Het circuit ziet er als volgt uit dan



LUCHTDRUK METEN

Opdracht

Meet de luchtdruk met de meegeleverde BMP sensor. Laat de gemeten waarden verschijnen in Pa in de shell. Creëer een overdruk of onderdruk rond de sensor om te testen.

Hardware

We hebben nodig:

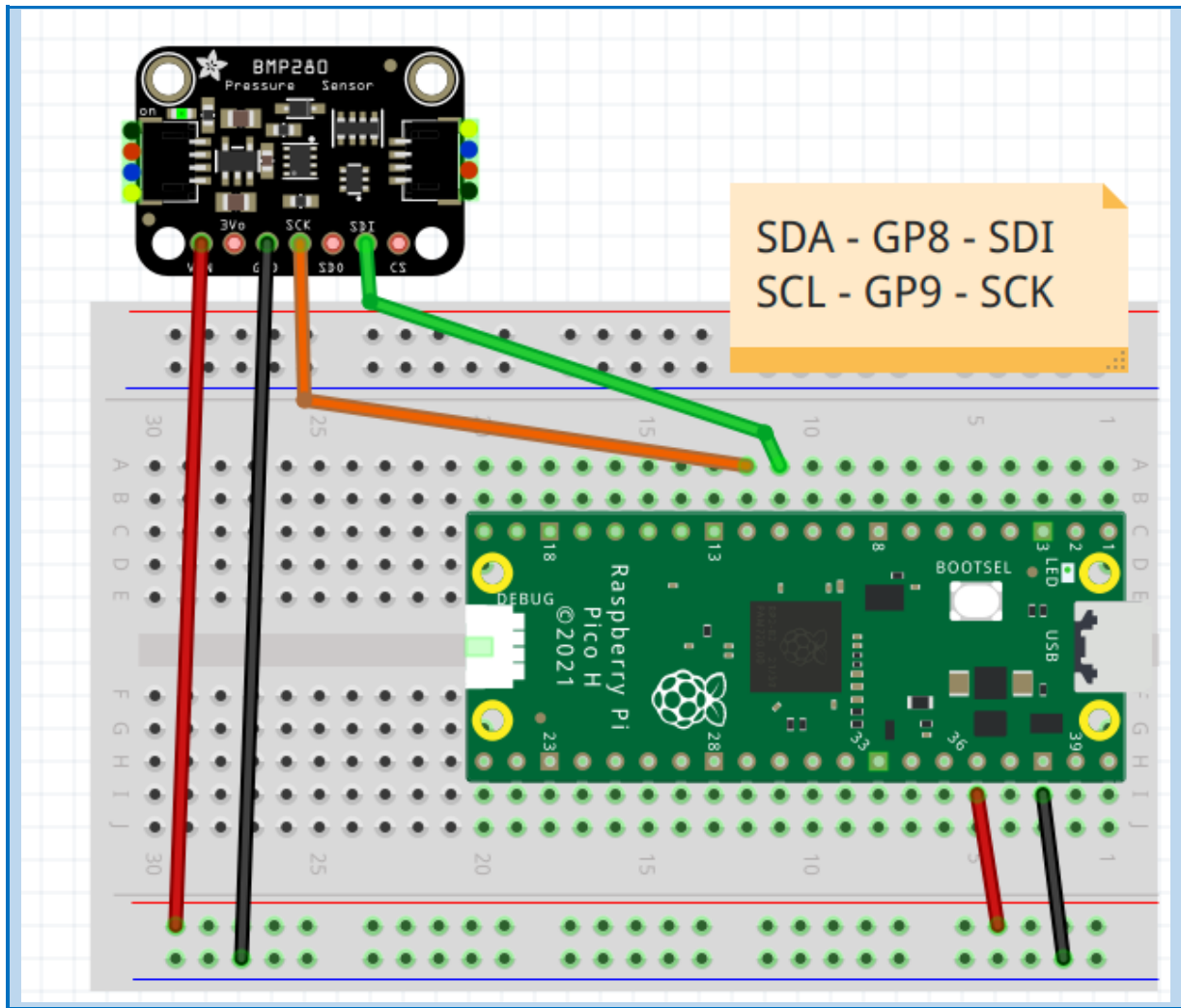
- 1 Pico
- 1 breadboard
- 1 sensor BMP280
- 4 jumper kabeltjes

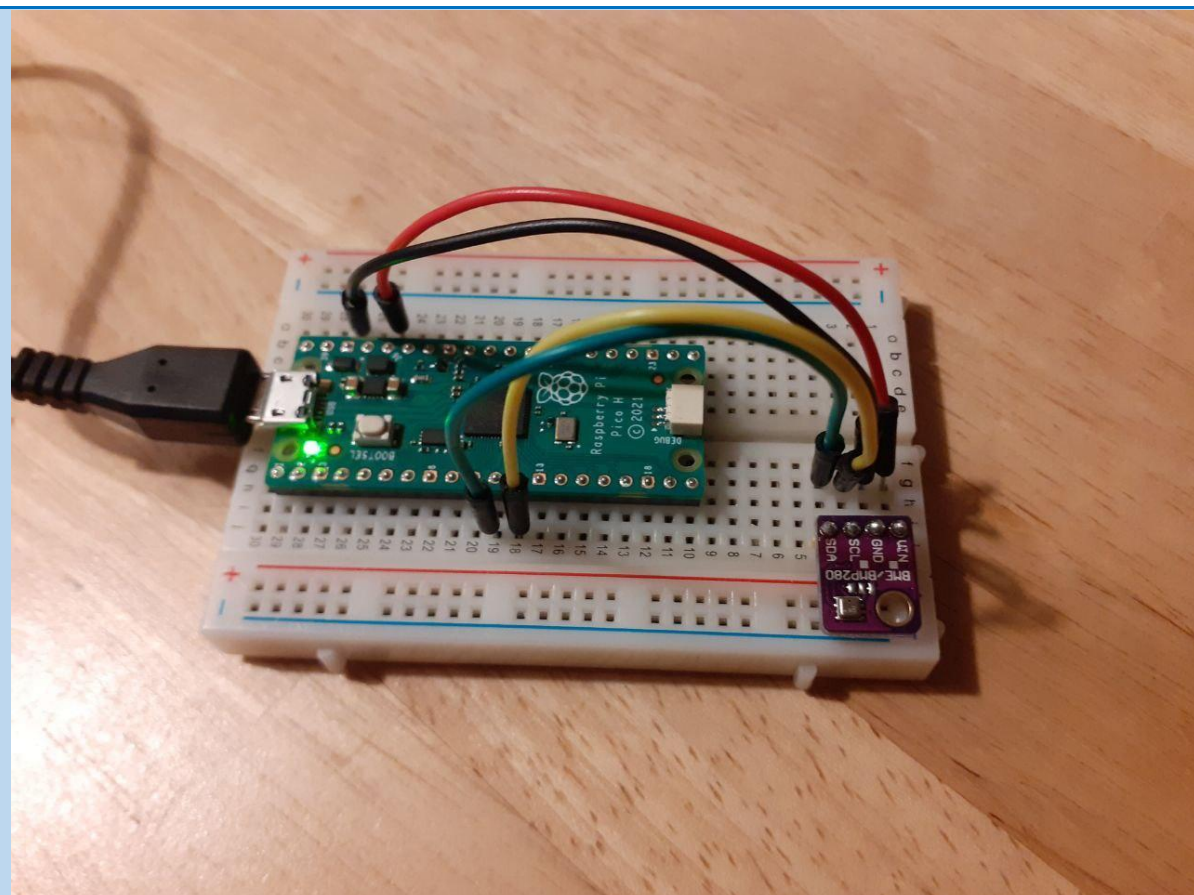
Circuit

We sluiten de sensor aan volgens de I2C communicatie-interface. De Pi Pico heeft twee fysieke I2C-interfaces die kunnen worden geconfigureerd om verschillende paren pinnen te gebruiken om te passen bij een PCB-ontwerp of behoeften voor andere interfaces.

Voorlopig gebruiken we I2C op pinnen GP8 en GP9;

- Sensor Vin op de 3V3 pin
- Sensor GND pin op een van de GND pinnen
- Sensor SCK pin op de GP9 (SCL)
- Sensor SDI pin op de GP8 (SDA)

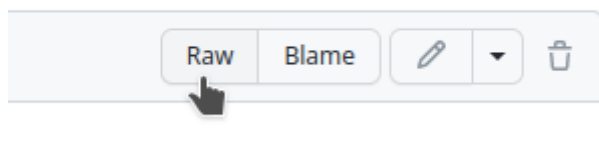




Circuit voor de oefening "Luchtdruk meten". - Bron: Ingegno

Library bmp280

We hebben een bibliotheek nodig die via I2C kan communiceren met de BMP280. Er zijn er verschillende beschikbaar, wij gebruiken deze: [micropython-bmp280](https://github.com/dafvid/micropython-bmp280)². Klik op het bmp280.py file, klik dan rechts op de Raw knop



Je bekomt hiermee het effectieve python script, welke je via CTRL+S kunt opslaan in de bestandsmap waar je je code opslaat als `bmp280.py`. Je kunt na dit gedaan te hebben, de functies geprogrammeerd in deze bibliotheek gebruiken vanuit een python file in dezelfde bestandsmap via het commando:

```
import bmp280
```

Om deze bibliotheek te gebruiken voor code die uitgevoerd wordt op de Pico, zul je dus via Thonny dit bestand `bmp280.py`, moeten opslaan op de Pico!

² <https://github.com/dafvid/micropython-bmp280>

Script

Volgend script kan gebruikt worden om de temperatuur en de druk te meten:

```
from machine import Pin, I2C
import bmp280
import time

# zet I2C connectie op via pin 8 en 9.
bus = I2C(0, scl=Pin(9), sda=Pin(8), freq=200000)
# maak een bmp object over de I2C connectie
bmp = bmp280.BMP280(bus)

# de bmp kan in low power en high power werken. We willen snel
# wijzigingen detecteren, dus kiezen we high power mode INDOOR.
# Kies je geen use case, dan wordt BMP280_CASE_HANDHELD_DYN
# geselecteerd, welke een lagere resolutie heeft, maar nog goed
# zou werken
bmp.use_case(bmp280.BMP280_CASE_INDOOR)

while True:
    # lees druk uit
    pressure = bmp.pressure
    p_bar = pressure/100000
    p_mmHg = pressure/133.3224
    # lees temperatuur uit
    temperature = bmp.temperature
    print("Temperature: {}°C".format(temperature))
    print("Pressure: {} Pa, {} bar, {} mmHg".format(pressure, p_bar,
        p_mmHg))
    # wacht 1 seconde
    time.sleep(1)
```

We komen hier enkele nieuwe commando's tegen die meer uitleg vergen.

Hoogte meten

Indien we weten wat de druk op zeeniveau is, en de temperatuur van de luchtlaag, kunnen we de **hoogte** bepalen uit een drukmeting. Het spreekt voor zich dat dit maar een beperkte nauwkeurigheid heeft, je dient de metingen buiten te doen met buitenlucht (en dus temperatuur buiten ook), waarna je lokaal hoogte kunt bekomen.

We schrijven een functie om de hoogte te bepalen, en gebruiken die dan in onze code.

Het script is als volgt:

```

from machine import Pin, I2C
import bmp280
import time

# zet I2C connectie op via pin 8 en 9.
bus = I2C(0, scl=Pin(9), sda=Pin(8), freq=200000)
# maak een bmp object over de I2C connectie
bmp = bmp280.BMP280(bus)

# de bmp kan in low power en high power werken. We willen snel
# wijzigingen detecteren, dus kiezen we high power mode INDOOR.
# Kies je geen use case, dan wordt BMP280_CASE_HANDHELD_DYN
# geselecteerd, welke een lagere resolutie heeft, maar nog goed
# zou werken
bmp.use_case(bmp280.BMP280_CASE_INDOOR)

def get_altitude(pressure, temperature,
sea_level_pressure_hPa=1013.25):
    # de druk op het punt waar je de hoogte van wil bepalen
    # temperature = de buitenlucht temperatuur in graden Celcius.
    # sea_level_pressure_hPa = druk op zeeniveau, een getal in hPa

    # volgende formule berekent de hoogte.
    altitude = ((pow((sea_level_pressure_hPa / pressure),
(1.0 / 5.257)) - 1)
                * (temperature + 273.15)) / 0.0065
    # onze functie geeft de hoogte terug aan de oproeper.
    return altitude

# Als de Pico aanschakelt meten we de druk, we slaan die op als druk
# op zeeniveau
sea_level_pressure_hPa = bmp.pressure/100
time.sleep(1)

while True:
    # lees druk uit
    pressure = bmp.pressure
    p_bar = pressure/100000
    p_mmHg = pressure/133.3224
    # lees temperatuur uit
    temperature=bmp.temperature
  
```

```
print("Temperature: {}°C".format(temperature))
print("Pressure: {} Pa, {} bar, {} mmHg".format(pressure, p_bar,
                                                p_mmHg))

# we berekenen nu de hoogte in meter
hoogte = get_altitude(pressure/100, temperature,
                      sea_level_pressure_hPa)
print("Hoogte: {} m".format(hoogte))
# wacht 1 seconde    time.sleep(1)
```

Probeer deze code. Ga met je laptop naar buiten, en laat de Pico op temperatuur komen met de buitenlucht, zodat juiste temperatuur waarden gemeten worden. Herstart dan de code zodat de juiste druk op “zeeniveau” bekomen wordt. Ga nu snel trappen op, de hoogte zou moeten weergegeven worden als 4 m op een 1e verdiep.

Oefening

Wijzig de code zodat je met een timer werkt. Is de temperatuur meting even goed als de DS18B20 meting? Je kan code schrijven om beide sensoren te gebruiken.

Meer informatie en voorbeeld van een scherm gebruiken: [zie hier](#).

DATA OPSLAAN OP SD KAART

SD-kaart module

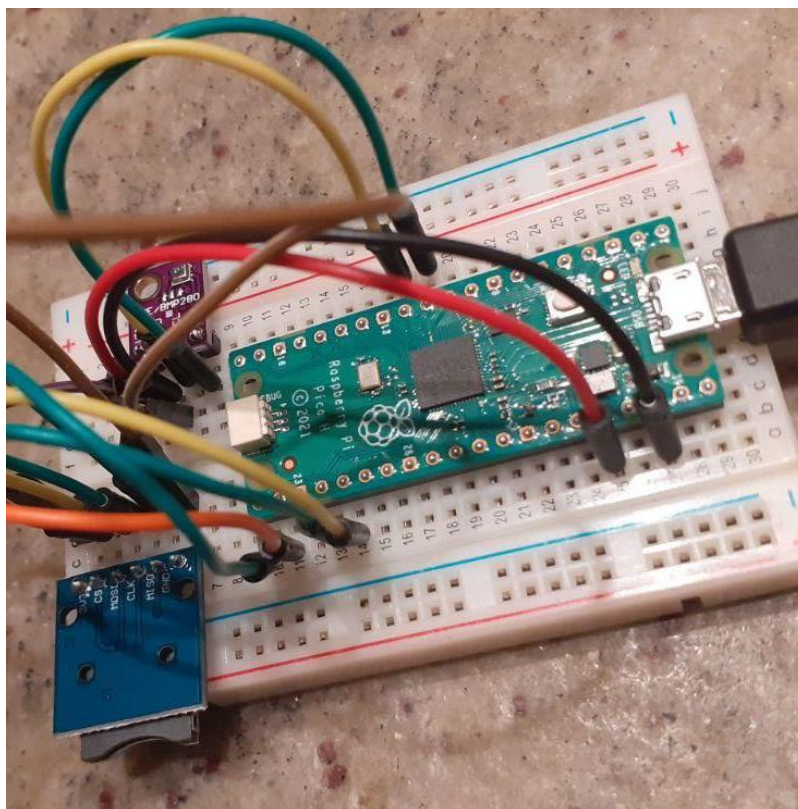
Ook als we kunnen communiceren met een satelliet, moeten we toch voor de veiligheid ook alle meetgegevens opslaan, de verbinding kan immers steeds verbroken worden. De eenvoudigste manier om dit te doen is via een SD-Card module. Voor lage datasnelheid, zoals het loggen van meetgegevens, kunnen we de SPI interface gebruiken om gegevens naar de kaart top te slaan. We gebruiken een bord geschikt voor 3.3V voeding, wat betekent een bord zonder stroomconversie zoals de HW-125 voor Arduino, maar wel een bord specifiek gemaakt voor de Pico of ESP32..

Circuit

Combineer de GND en VCC van de BMP280 en het SD-kaart Shield. Je heb dan nog de SPI connecties nodig: CS, SCK, MOSI en MISO. De Pico heeft SPI0 en SPI1. Wij gebruiken SPI0, namelijk: CS naar pin 22 (GP17), SCK of CLK naar pin 24 (GP24), MOSI naar pin 25 (TX, GP19), en MISO naar pin 21 (RX, GP16).



Op een breadboard ziet dit er dan als volgt uit:

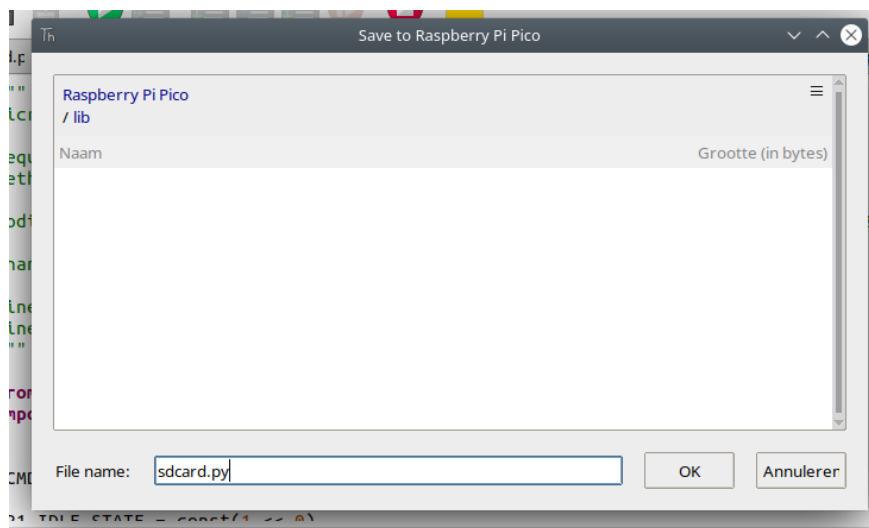


sdcard bibliotheek

Om gegevens te kunnen opslaan op de sd-kaart, dienen we een bibliotheek te installeren, op dezelfde manier zoals we dit deden voor de BMP280. Om de code overzichtelijk te houden, is het een goed idee om om de bibliotheek op te slaan op de pico in een map **lib**, in die map kun je alle hulpcode verzamelen. In versie 1.19 van Micropython is de SDCard bibliotheek voor de Pico nog niet standaard inbegrepen, maar verschillende personen hebben online wel al zo'n bibliotheek beschikbaar. We gebruiken deze van CoreElectronics:

<https://github.com/CoreElectronics/CE-Makerverse-R2R-DAC-MicroPython-Module/blob/main/sdcard.py>

Ga, net zoals bij de bmp280, naar de **raw**, versie, download deze, open ze via Thonny, en sla op op de Pico in map **lib** met naam **sdcard.py**. Alle bestanden opgeslagen in de map lib zullen als eerste gevonden worden door python via het import commando. Je kan dus ook de bmp280.py code in deze map plaatsen.



De SD card die gebruikte moet FAT32 zijn. De kaarten in de kit zijn reeds geformatteerd en kunnen direct gebruikt worden. Indien je zelf nog een kaart wil formatteren, dan kan dat in Linux met het programma **gparted**, en in Windows via volgende stappen:

1. Connecteer de kaart met je Windows PC
2. Open Explorer (Bestanden)
3. Rechts klik op de kaart drive letter in Explorer
4. Selecteer "Formateer..."
5. Zorg dat je FAT32 hebt geselecteerd onder het "Bestandsysteem" drop-down drop-down menu
6. Klik "Start" om het formatteren te starten.

Script: Opslaan van BMP280 meetgegevens op sd-kaart

De meetgegevens worden dus best weggeschreven naar een geheugenkaart. We doen dit in de vorm van een **tekstbestand**. Dat bestand kan je nadien opslaan als .csv (staat voor 'comma seperated values'). Je doet dit simpelweg door in Windows Verkenner (of Finder van Mac) de extensie .txt te overschrijven met .csv. Een csv bestand kan onder meer gelezen worden door MS Excel, Google Spreadsheet,

We hebben geen juiste tijdsbepaling op de Pico, dus slaan we telkens de meetgegevens op in een nieuw bestand, waarbij we een teller (counter) gebruiken om te weten wat de naam van het databestand moet zijn. Deze teller slaan we ook op in een tekstbestand op de SD kaart. Je dient dan zelf met je horloge op te nemen wanneer je de Pico van stroom voorziet, de tijd in het databestand zelf zal het aantal seconden zijn na het opstarten van de Pico.

De code is als volgt:

```
from machine import Pin, I2C, SPI
import bmp280
import sdcard
import time, uos
```

```

# zet I2C connectie op via pin 8 en 9.
bus = I2C(0, scl=Pin(9), sda=Pin(8), freq=200000)
# maak een bmp object over de I2C connectie voor drukmeting
bmp = bmp280.BMP280(bus)

# Maak een chip select (CS) pin (en start het HIGH)
cs = machine.Pin(17, machine.Pin.OUT)
# Intialiseer SPI peripheral 0 (start met 1 MHz)
# We doen dit met pinnen voor sck, mosi en miso:
spi = SPI(0,                                     # SPI0 of SPI1
          baudrate=1000000,
          polarity=0,
          phase=0,
          bits=8,
          firstbit=machine.SPI.MSB,
          sck=machine.Pin(18, machine.Pin.OUT),
          mosi=machine.Pin(19, machine.Pin.OUT), # De SPI0 TX
          miso=machine.Pin(16, machine.Pin.OUT)) # De SPI0 RX
# Creeer een SDCard object sd met de spi en cs:
sd = sdcard.SDCard(spi, cs)

# de bmp kan in low power en high power werken. We willen snel
# wijzigingen detecteren, dus kiezen we high power mode INDOOR.
# Kies je geen use case, dan wordt BMP280_CASE_HANDHELD_DYN
# geselecteerd, welke een lagere resolutie heeft, maar nog goed
# zou werken
bmp.use_case(bmp280.BMP280_CASE_INDOOR)

def get_altitude(pressure, temperature,
                 sea_level_pressure_hPa=1013.25):
    # de druk op het punt waar je de hoogte van wil bepalen
    # temperature = de buitenlucht temperatuur in graden Celcius.
    # sea_level_pressure_hPa = druk op zeeniveau, een getal in hPa

    # volgende formule berekent de hoogte.
    altitude = ((pow((sea_level_pressure_hPa / pressure), (1.0 / 5.257))
- 1)
                * (temperature + 273.15)) / 0.0065
    # onze functie geeft de hoogte terug aan de oproeper.
    return altitude

# Als de Pico aanschakelt meten we de druk, we slaan die op als druk op
zeeniveau

```

```

sea_level_pressure_hPa = bmp.pressure/100
time.sleep(1)

# We openen de SD kaart en koppelen aan directory /sd
vfs = uos.VfsFat(sd)
uos.mount(vfs, "/sd")
# we schrijven de directory van de SD kaart uit in REPL
files = uos.listdir('/sd')
print("SD-kaart bestanden:", files)
# we wijzigen uos naar directory /sd
uos.chdir('/sd')

# Controleer of counter.txt aanwezig is
counter = 1
if "counter.txt" in files:
    # open het en lees de teller uit (read = 'r')
    print ("counter found")
    with open('/sd/counter.txt', "r") as countfile:
        counter = countfile.readline()
        print(counter)
        counter = int(counter)
    # we tellen 1 bij de teller en slaan op voor volgende keer (write =
    'w')
    with open('/sd/counter.txt', "w") as countfile:
        countfile.write(str(counter + 1)) #schrijf teller
        countfile.write('\n')           # Een nieuwe lijn op einde
else:
    # counter.txt bestaat niet, dus maak het met inhoud 1
    with open('/sd/counter.txt', "w") as countfile:
        print("Maken counter.txt aan!")
        countfile.write(str(counter)) # de teller als string
        countfile.write('\n')       # een nieuwe lijn

# sla sensordata op in bTdat_XXX.txt met XXX de teller:
filenamedata = 'bTdat_' + str(counter) + '.txt'
# schrijf de header van de data uit (kolomheaders)
with open('/sd/' + filenamedata, "a") as datafile:
    datafile.write('time [s]') # Write time sample was taken in seconds
    datafile.write(' ; ') # Een separator
    datafile.write('temp [C]')
    datafile.write(' ; ') # Een separator
    datafile.write('druk [Pa]')
    datafile.write(' ; ') # Een separator

```



```

datafile.write('druk [bar]')
datafile.write(' ; ') # Een separator
datafile.write('druk [mmHg]')
datafile.write(' ; ') # Een separator
datafile.write('hoogte [m]')
datafile.write(' ; ') # Een separator
datafile.write('zeedruk [hPa]')
datafile.write('\n') # A nieuwe lijn

# voer nu metingen uit zolang er stroom is
while True:
    # lees druk uit
    pressure = bmp.pressure
    p_bar = pressure/100000
    p_mmHg = pressure/133.3224
    # lees temperatuur uit
    temperature=bmp.temperature
    print("Temperature: {}°C".format(temperature))
    print("Pressure: {} Pa, {} bar, {} mmHg".format(pressure, p_bar,
p_mmHg))
    # we berekenen nu de hoogte in meter
    hoogte = get_altitude(pressure/100, temperature,
sea_level_pressure_hPa)
    print("Hoogte: {} m".format(hoogte))
    # we slaan de sensordata op door achteraan in bestand te schrijven
    # toevoegen = append = "a"
    with open('/sd/' + filenameedata, "a") as datafile:
        # bereken aantal seconden dat de Pico aan staat:
        t = time.ticks_ms()/1000
        datafile.write(str(t)) # Schrijf tijd van meting in seconden
        datafile.write(' ; ') # Een separator
        datafile.write(str(temperature))
        datafile.write(' ; ') # Een separator
        datafile.write(str(pressure))
        datafile.write(' ; ') # Een separator
        datafile.write(str(p_bar))
        datafile.write(' ; ') # Een separator
        datafile.write(str(p_mmHg))
        datafile.write(' ; ') # Een separator
        datafile.write(str(hoogte))
        datafile.write(' ; ') # Een separator
        datafile.write(str(sea_level_pressure_hPa))
        datafile.write('\n') # Een nieuwe lijn

```

```
# wacht 1 seconde
time.sleep(1)
```

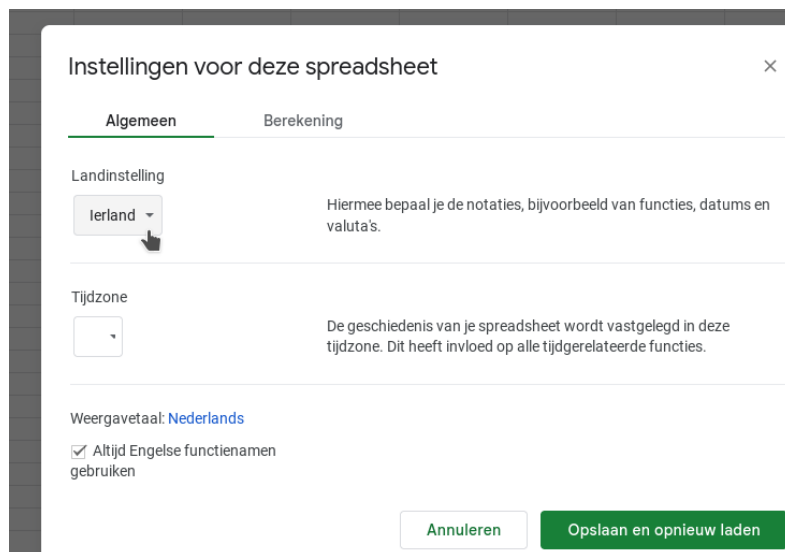
Voorbeeld van de data opgeslagen in een bTdat_XXX.txt betand:

```
1 time [s] ; temp [C] ; druk [Pa] ; druk [bar] ; druk [mmHg] ; hoogte [m] ; zeedruk [hPa]
2 7.894 ; 21.33 ; 102857.4 ; 1.028574 ; 771.4939 ; 0.2160292 ; 1028.6
3 8.922 ; 21.41 ; 102856.4 ; 1.028564 ; 771.4864 ; 0.3025231 ; 1028.6
4 9.95 ; 21.45 ; 102856.4 ; 1.028564 ; 771.4865 ; 0.3025642 ; 1028.6
5 10.977 ; 21.46 ; 102856.7 ; 1.028567 ; 771.4888 ; 0.2755589 ; 1028.6
6 12.005 ; 21.45 ; 102856.8 ; 1.028568 ; 771.489 ; 0.2701466 ; 1028.6
7 13.034 ; 21.44 ; 102856.9 ; 1.028569 ; 771.4897 ; 0.2647346 ; 1028.6
8 14.063 ; 21.45 ; 102856.1 ; 1.028561 ; 771.484 ; 0.3295788 ; 1028.6
9 15.096 ; 21.47 ; 102855.3 ; 1.028553 ; 771.4777 ; 0.3998441 ; 1028.6
10 16.124 ; 21.47 ; 102852.4 ; 1.028524 ; 771.4561 ; 0.6375892 ; 1028.6
11 17.151 ; 21.48 ; 102851.0 ; 1.02851 ; 771.446 ; 0.7510839 ; 1028.6
12 18.179 ; 21.48 ; 102851.6 ; 1.028516 ; 771.4502 ; 0.7024527 ; 1028.6
13 19.207 ; 21.48 ; 102853.2 ; 1.028532 ; 771.4622 ; 0.5727691 ; 1028.6
14 20.236 ; 21.48 ; 102852.8 ; 1.028528 ; 771.4596 ; 0.5997865 ; 1028.6
15 21.265 ; 21.46 ; 102853.9 ; 1.028539 ; 771.4676 ; 0.5132959 ; 1028.6
16 22.298 ; 21.46 ; 102854.5 ; 1.028545 ; 771.4718 ; 0.4646679 ; 1028.6
17 23.326 ; 21.47 ; 102854.0 ; 1.028539 ; 771.4678 ; 0.5079101 ; 1028.6
18 24.353 ; 21.47 ; 102854.8 ; 1.028548 ; 771.474 ; 0.4376672 ; 1028.6
19 25.381 ; 21.47 ; 102856.5 ; 1.028565 ; 771.4868 ; 0.2971814 ; 1028.6
20 26.409 ; 21.47 ; 102855.4 ; 1.028554 ; 771.4784 ; 0.3890375 ; 1028.6
21 27.438 ; 21.48 ; 102856.1 ; 1.028562 ; 771.4844 ; 0.3242089 ; 1028.6
22 28.466 ; 21.47 ; 102857.0 ; 1.02857 ; 771.491 ; 0.2485517 ; 1028.6
23 29.495 ; 21.48 ; 102857.4 ; 1.028574 ; 771.4939 ; 0.2161393 ; 1028.6
24 30.527 ; 21.48 ; 102857.6 ; 1.028576 ; 771.4955 ; 0.1999288 ; 1028.6
```

Sensordata analyseren

Je kan de sensordata eenvoudig analyseren in Google Spreadsheet. Open hiertoe het bestand in een ascii tekst verwerker zoals bv Notepad. Selecteer alle data met CTRL+A, en copieer deze via CTRL+C.

In Google Drive, maak een nieuw spreadsheet (rekenblad) aan. Wijzig nu de taal van het spreadsheet naar Ierland, zodat de internationale nummerschrijfwijze gebruikt wordt (decimaal punt). Je doet dit via menu Bestand → Instellingen



Instellingen voor deze spreadsheet

Algemeen Berekening

Landinstelling

Ierland

Hiermee bepaal je de notaties, bijvoorbeeld van functies, datums en valuta's.

Tijdzone

De geschiedenis van je spreadsheet wordt vastgelegd in deze tijdzone. Dit heeft invloed op alle tijdgerelateerde functies.

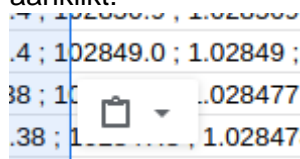
Weergavetaal: Nederlands

Altijd Engelse functienamen gebruiken

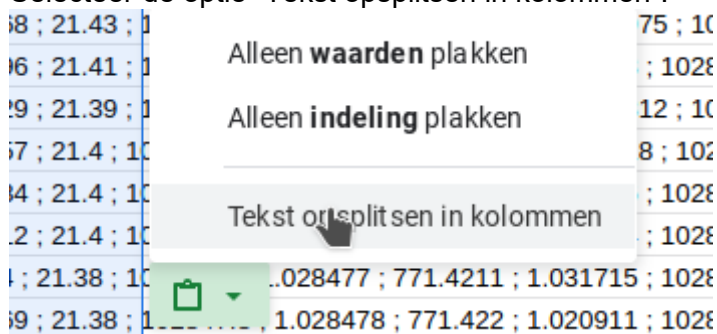
Annuleren Opslaan en opnieuw laden

Klik op *Opslaan en opnieuw laden*.

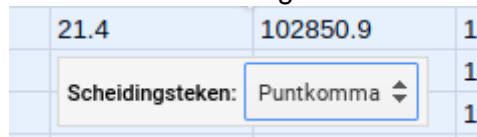
Plak de data in de eerste cel met CTRL+V. Er verschijnt dan onderaan een icoon, welke je aanklikt:



Selecteer de optie “Tekst opsplitsen in kolommen”:



Je dient het scheidingsteken te selecteren. Selecteer Puntkomma



Selecteer nu kolom A en kolom F terzelfdertijd door op A te klikken, en dan met CTRL ingedrukt ook op kolom F.

Klik dan in de toolbar op icoon voor “diagram toevoegen”, en wijzig diagram type in Spreidingsdiagram:

Diagrameditor ×**Instellen**

Aanpassen

Diagramtype

Spreidingsdiagram ▼

Gegevensbereik

A1:A999,F1:F999 ⊞

Bereiken combineren

Horizontaal ▼

X-as

123 time [s] ⋮ Verzamelen

Reeks

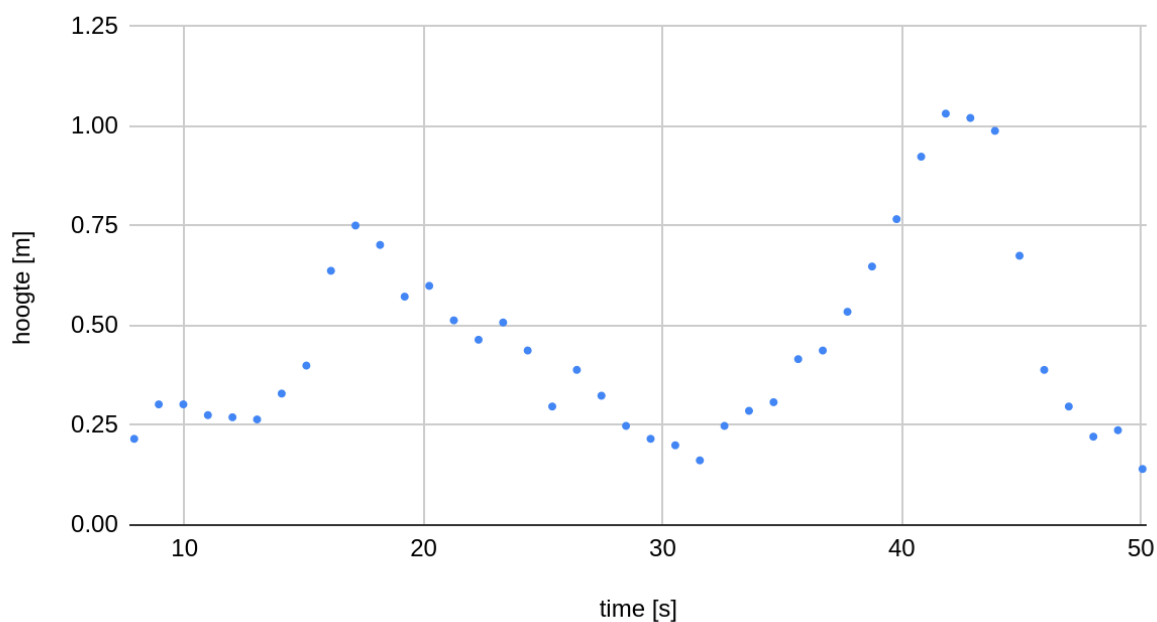
123 hoogte [m] ⋮

Reeks toevoegen

 Rijen/kolommen verwisselen Rij 1 als kop gebruiken Kolom A als labels gebruiken Labels als tekst behandelen

Je bekomt een mooie figuur van hoe de hoogte wijzigt met de tijd.

hoogte [m] versus time [s]

**Opdrachten**

1. Maak een diagram van de temperatuur versus de tijd
2. Kun je de snelheid van stijgen berekenen? Hoe?
3. De "zeedruk" is de druk bij opstart van de Pico. Maar mogelijks stond de Pico niet op de grond toen je startte? Je kan achteraf, of op bepaald moment na opstart Pico, een nulmeting op de grond doen (de vloer). Je dient dan de correcte hoogte te herberekenen in je rekenblad achteraf. Een idee hoe je dat doet?

DATA VERZENDEN/ONTVANGEN MET RADIO-COMMUNICATIE: Beperkte inleiding

Het valt buiten het opzet van deze vorming om te werken met radio-communicatie. Om u toch enigszins in te leiden geven we hieronder een klein beetje achtergrond informatie. Het installeren van een radio communicatie is echter geen deel van deze cursus, en zal u derhalve hieronder niet vinden. In een latere ESERO vorming zullen we dit wel toevoegen.

Space link

Een “space link” is het communicatiesysteem tussen de satelliet en het grondstation (of meerdere grondstations). Het bestaat essentieel uit deze twee delen:

- **Telemetrie (TM) downlink**
Meetgegevens van de payload worden door de satelliet uitgezonden. Deze data worden in het grondstation ontvangen.
- **Telecomand (TC) uplink**
Besturingscomando's worden in het grondstation uitgezonden, en door de satelliet ontvangen.

In deze vorming is de TC uplink niets anders dan de besturingsmodule van de drone, bestuurd door de piloot. De TM downlink moeten we zelf voorzien, tenzij we de data alleen lokaal opslaan in onze satelliet zelf op een SD kaartje (om na landing te recupereren).

Radiogolven en frequenties

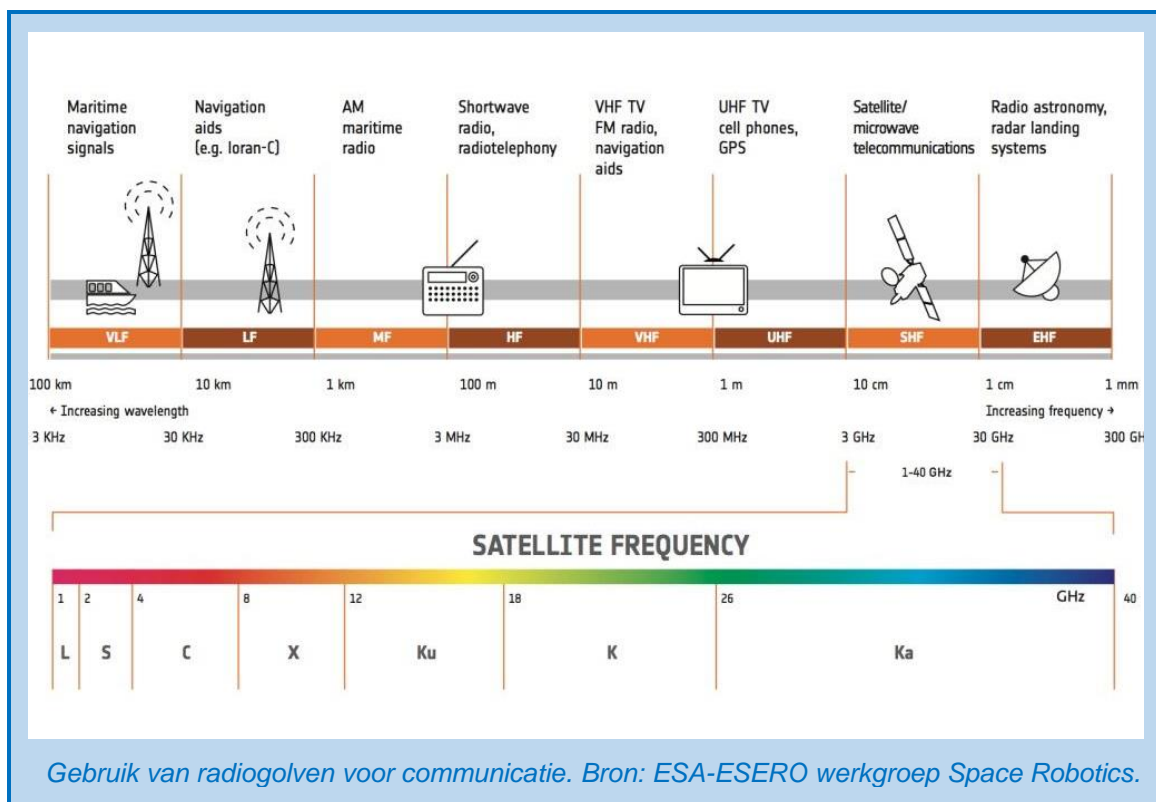
Radiogolven liggen aan het uiteinde van het elektromagnetisch spectrum. Alle elektromagnetische golven bewegen zich voort aan de lichtsnelheid c . In het luchtledige is c gelijk aan ongeveer 300.000 km/s (300.000.000 m/s).

Radiogolven zijn de langste golven. Hun golflengte varieert van 10 cm tot vele kilometers. Zoals bij alle elektromagnetische golven is hun frequentie eenvoudig te berekenen wanneer de golflengte λ bekend is:

$$\lambda f = c$$

λ = golflengte, f = frequentie, c = lichtsnelheid. Bijgevolg kunnen we afleiden dat de frequenties van radiogolven kleiner of gelijk zijn aan 30.000.000 Hz (30 MHz).

Golven met een kleinere golflengte (en dus grotere frequentie) worden microgolven en radargolven genoemd. De gewone radiogolven ($\lambda \geq 10$ cm) worden op aarde gebruikt voor radiocommunicatie. Microgolven en radargolven worden gebruikt voor communicatie in de ruimtevaart, omdat ze zonder verstoring door de atmosfeer reizen. Ze worden niet gereflecteerd door de ionosfeer.



Hoe kan een elektromagnetische golf informatie bevatten?

Stel je een radiogolf voor als een sinusgolf. Het opvangen van zo'n golf in een elektronisch apparaat, zal een variërende spanning veroorzaken. Die variërende spanning kan op elk moment bepaald worden. De spanning die op 1 bepaald moment gemeten kan worden, wordt de 'displacement' genoemd. De displacement Y (in Volt) kan je berekenen:

$$Y = A \sin (2 \pi f t)$$

A is de maximale amplitude van de sinusgolf.

f is de frequentie van de golf.

t is het tijdstip van de meting.

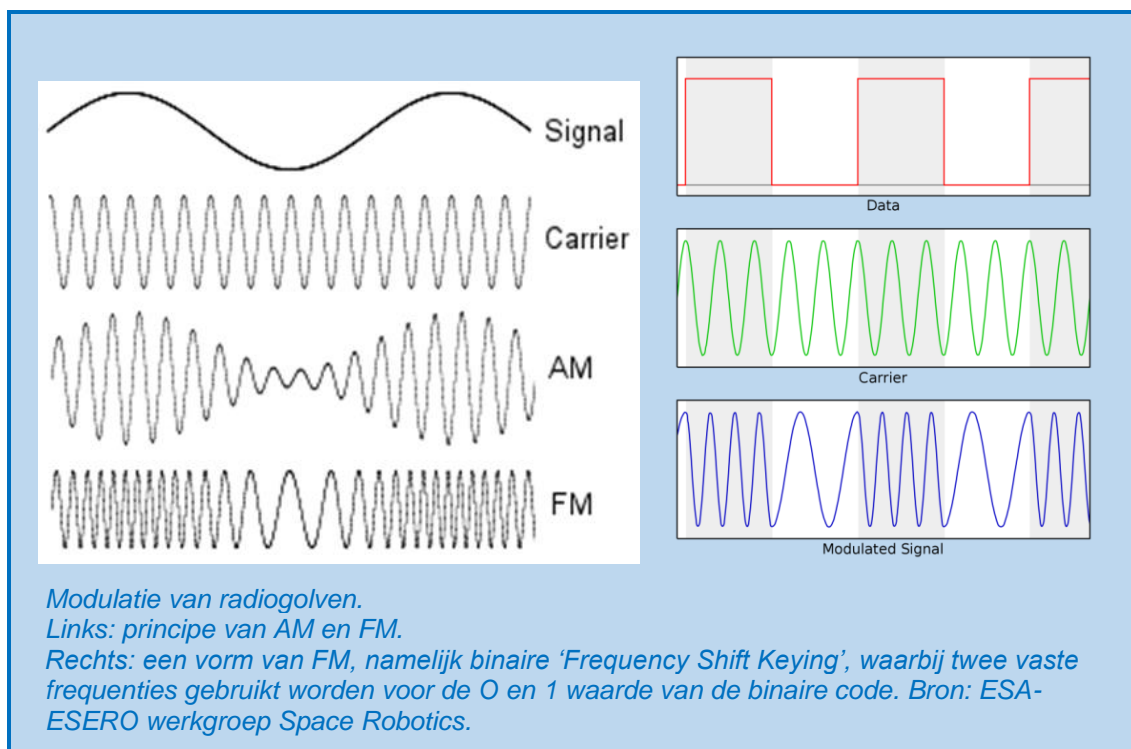
Nu is onze perfecte sinusgolf niet erg interessant om uit te lezen. De displacement zal immers een eentonig patroon vertonen: hoog-laag-hoog-laag-... met een regelmatige frequentie en vaste amplitude. Deze golf is geschikt als **draaggolf (carrier wave)** omdat we kunnen afstemmen op 1 bepaalde frequentie om hem te ontvangen. Maar er moet nog informatie aan toegevoegd worden.

Modulatie

Bij golfmodulatie gaan we een draaggolf combineren met een golf die de eigenlijk data of boodschap bevat, de **signaalgolf (signal wave)**. Het product van zulke combinatie van draaggolf en signaalgolf is een **gemoduleerde golf**.

De data worden overgebracht in bits (0 of 1, hoog of laag) in een bepaald patroon. De gemoduleerde golf brengt deze bits over met variaties in de amplitude of met variaties in de

frequentie. In het eerste geval spreekt men van *Amplitude Modulation* of AM. In het tweede geval gaat het over *Frequentie Modulation* of FM.



Voor radiocommunicatie met onze mini-satelliet is het een optie om gebruik te maken van **FM**. Deze methode van modulatie heeft voordelen:

- Minder ruis die mee versterkt wordt
- Minder power is nodig om dezelfde informatie te verzenden
- De bandbreedte is groter

Metadata

'identifier'

In principe zal de organisator van een STEM project een unieke frequentie bezorgen die je als team kan gebruiken voor het verzenden en ontvangen van je data. Maar wanneer meerdere teams data per radiocommunicatie versturen, kan het toch nog voorkomen dat er interferentie optreedt tussen meerdere satellietjes. Het is daarom belangrijk om je microcontroller ook een unieke code of uniek woord te laten meesturen met de meetgegevens. Zo herken je met zekerheid je eigen data.

Timestamp

Bovendien is het ook handig om samen met de meetgegevens de datum en het uur mee te verzenden. Zo geraken data niet gemakkelijk verward.

Radiocommunicatie met de Pico

De pico is relatief recent, dus zijn er nog niet zoveel goed uitgewerkte voorbeelden van radio communicatie. In volgende twee links vind je mogelijkheden:

- <https://github.com/aallan/pico-lorawan-circuitpython>
- <https://rpi-rfm69.readthedocs.io/en/latest/index.html>

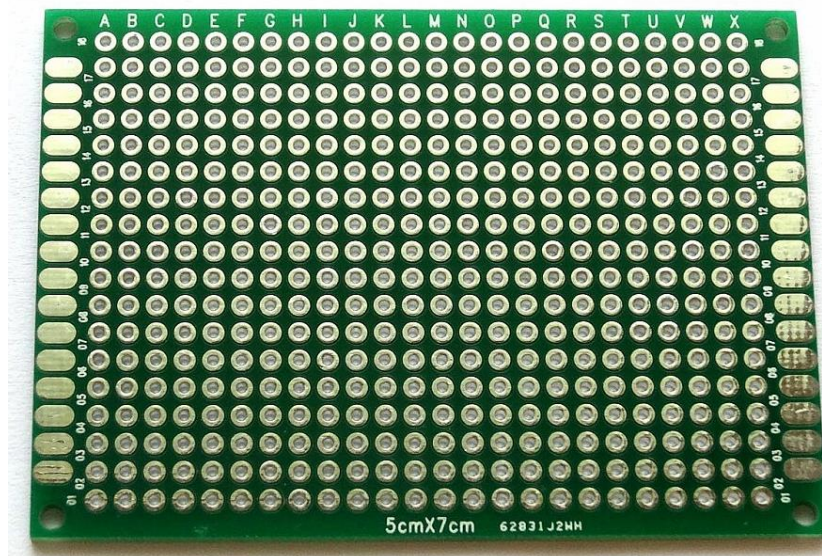
Een alternatief is de Pico H om te wisselen voor een Pico WH met wifi, en dan kun je via WiFi (100m max) communiceren indien je antenne van de receiver goed positioneert voor maximale ontvangst.

4 De satelliet klaarmaken

4a De afgewerkte satelliet: Hardware

Met een BMP280 en een SD-kaart bord op je breadboard, heb je de basisvorm van de satelliet af. Test deze. Je zal vaststellen dat op deze manier het geheel niet erg stevig is: draadjes komen gemakkelijk los, een breadboard is niet ideaal en heeft nogal veel storing, wat vooral problematisch is voor de SD-kaart. Niettemin, een eerste test is mogelijk.

Wil je beter doen? Dan dien je het geheel te solderen op een prototyping board.



Je soldeert dan headers waarin de Pico en de sensoren passen, en gebruikt vaste connecties voor de verbindingen ipv jumper kabels. Belangrijk is dat je dan correct soldeert.

SOLDEREN

Waarom Solderen?

Het voordeel van het vast solderen van satellietonderdelen is heel duidelijk: ze komen niet los tijdens de lancering of de vlucht. Een goed gesoldeerde component is super betrouwbaar. Is het solderen noodzakelijk voor elk elektronisch aangestuurd experiment? Hierop is maar 1 mogelijk antwoord: JA, het is absoluut noodzakelijk. Het risico dat al je werk verloren gaat door een loskomend contact is heel groot als ze niet gesoldeerd zijn.

Veiligheid

- Een soldeerbout heeft een operationele temperatuur van 300 à 400°C. Wees dus erg voorzichtig met de soldeertip om brandwonden te vermijden.
- Zet de soldeerbout altijd uit de buurt van andere voorwerpen.
- Schakel de soldeerbout uit zodra het werk gedaan is, en laat het voldoende afkoelen voordat je het opbergt in een kast.
- Draag een veiligheidsbril.

Soldeermetaal

Met soldeermetaal bedoelen we de metaaldraad die je laat smelten op de plek van de aan te maken verbinding.

Het traditionele soldeermetaal is een loodlegering. Dit is een legering die bij het smelten heel mooi en vlot in het gaatje vloeit. Het is daarom het gemakkelijkste soldeermetaal voor beginners. Maar vanwege de loodhoudende damp bij het smelten is het verboden op school. Gebruik dus een van de alternatieve niet-loodhoudende soldeermetalen.

Soldeerstation

De soldeerbout is het centrale gereedschap bij het solderen. Het heeft een metalen punt die opwarmt tot ongeveer 400°C. Gewoonlijk wordt het geleverd met een handige houder, die toelaat dat je de hete soldeerbout uit je handen zet zonder gevaar.

De kostprijs van een goedkoop soldeerstation ligt in de grootte-orde van 25 euro. Je kan er prima mee solderen. Het voordeel van duurdere toestellen is dat ze gewoonlijk veel langer intact blijven, zelfs al laat je ze dagelijks continu aanstaan.

Accessoires

Je houdt best bij de hand:

- Een kleine tang met puntig uiteinde
- Een kniptang om kabeltjes af te knippen
- Een striptang om kabeltjes van isolatie te ontdoen
- Eventueel: een vergrootglas op een voet (handenvrij)
- Goede verlichting

Voorbereiding

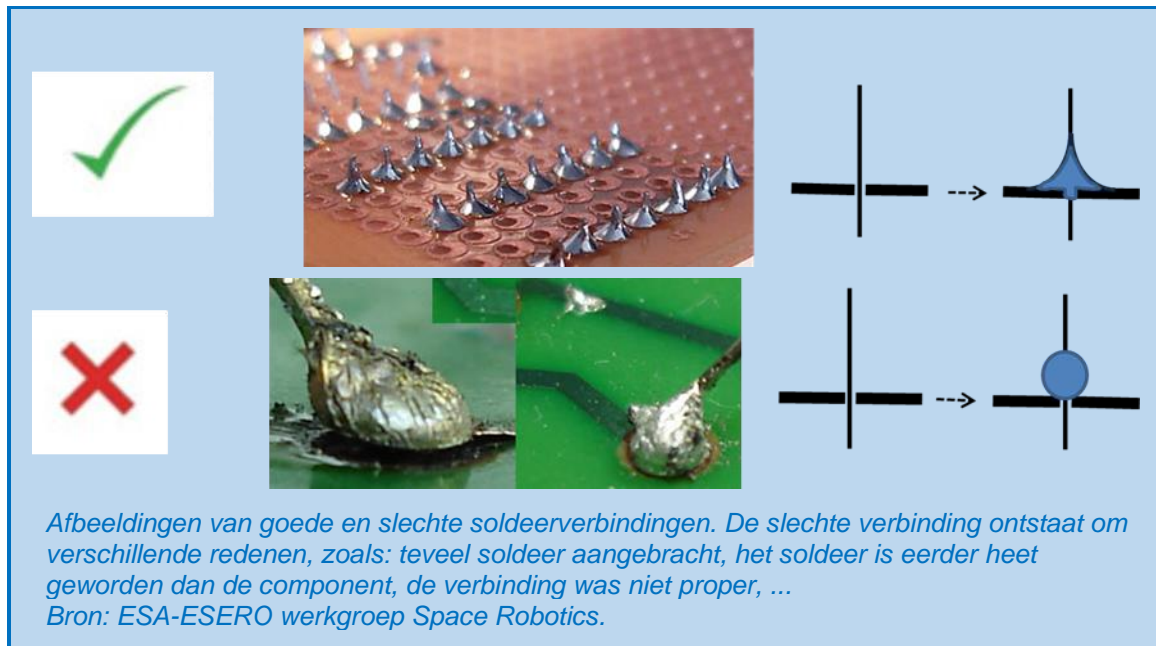
1. Plaats de soldeerbout in zijn standaard en stop de stekker in het stopcontact. De soldeerbout heeft een paar minuten nodig om tot de **juiste temperatuur** van ongeveer 350 - 400°C te komen.
2. Bevochtig de **spons** in de standaard. De spons moet vochtig zijn maar niet kletsnat. Knijp het teveel aan water eruit.
3. Wacht enkele minuten tot de soldeerbout **op werkte temperatuur** is. Je kan dit checken door een klein beetje soldeer tegen de punt te houden. Smelt het goed dan is de soldeerbout op de juiste temperatuur.
4. Nu de soldeerbout op temperatuur is veeg je **de punt schoon** op de vochtige spons
5. **Smelt een klein beetje soldeer** op de soldeertip. Dit wordt vertinnen genoemd en het helpt om de hitte van de punt te verplaatsen naar de te maken verbinding. Dit doe je alleen nadat je de punt hebt schoongemaakt op de spons.

Solderen: Instructies

Je bent nu klaar om te gaan solderen.

1. Hou de soldeerbout vast **als een pen**. Doe net alsof je je naam gaat schrijven. Kijk uit dat je het hete gedeelte niet aanraakt.

2. **Raak de plek** waar je de verbinding wil gaan maken aan **met het soldeerpunt**. Zorg ervoor dat het punt zowel het onderdeel als de printbaan of het andere onderdeel aanraakt. Hou het soldeerpunt op deze plek voor 3 seconden en dan....
3. **Plaats wat soldeer** op het te solderen punt. Als alles goed op temperatuur is zal het soldeer soepel vloeien over de printbaan en het component of componenten onderling. Het moet de vorm van een vulkaan aannemen. Voeg soldeer toe tegen de te maken verbinding en niet tegen de soldeerbout.
4. **Haal de soldeer weg** en daarna de soldeerbout. Laat de gemaakte verbinding even afkoelen en houdt de verbinding stil !
5. **Inspecteer** de verbinding goed! Het soldeer moet glimmen en de vorm hebben van een vulkaan. Als dit niet zo is dan moet je de verbinding opnieuw verwarmen en wat meer soldeer toevoegen. Zorg er dit keer voor dat zowel de printbaan als het onderdeel goed heet zijn.



Solderen kan in principe vanaf twee verschillende zijden:

- **Thru hole soldering**
Het component pootje/pinnetje wordt langs de ene kant van de printplaat in een gaatje gebracht, het soldeer wordt erin gesmolten vanaf de andere kant (onderkant).
- **Smt**
Alles langs dezelfde kant. Met een gewone soldeerbout is deze techniek wel iets moeilijker.

Een checklist voor een perfecte soldeerverbinding

1. Alle delen moeten goed schoon zijn en niet geoxideerd.
2. Zet de te solderen printplaat vast in een standaard zoals een *helping hand* of een andere manier.
3. Verwarm het heet soldeerpunt met een kleine hoeveelheid soldeer. Doe dit ook altijd met nieuwe soldeerpunten die voor het eerst worden gebruikt.
4. Maak het uiteinde van de hete soldeerpunt op een vochtige spons schoon.
5. Verwarm beide delen van de verbinding met de soldeerbout voor ongeveer een seconde.

6. Blijf verwarmen en voeg vervolgens voldoende soldeer toe om een adequate verbinding te vormen.
7. Verwijder de soldeerbout en zet deze terug in de soldeerstandaard.
8. Het vergt totaal slechts twee of drie seconden hoogstens, om de gemiddelde verbinding te solderen.
9. Beweeg geen onderdelen tot het soldeer is afgekoeld en de soldeerverbinding hard is.
10. Controleer of de verbinding goed is gemaakt.

Meer info:

<https://learn.sparkfun.com/tutorials/how-to-solder-through-hole-soldering>

5 Lancering

We lanceren de minisatelliet. We kunnen dit doen via een koord waaraan we de satelliet hangen, en die dan we van een hoog gebouw laten zakken terwijl we metingen doen.

Een alternatief is een drone gebruiken waarbij we de satelliet veilig bevestigen onderaan de drone. De drone moet wel voldoende gewicht kunnen meenemen, en moet bestuurd worden door een persoon met een drone brevet.

6 Data verwerken en presenteren

KALIBRATIE EN VERWERKING: INLEIDING

Je kan de microcontroller zo programmeren dat de ruwe data (meetgegevens) reeds in de payload gekalibreert en verwerkt worden. Op die manier kan je in het grondstation meteen de gewenste metingen aflezen zoals je ze wou hebben. Tijdens de ESERO vorming zullen we dit doen om tijd te winnen.

Maar dit wordt afgeraden. Het is beter de ruwe data eerst te verwerven, en ze later met de computer te verwerken. Zo is het risico dat er iets fout gaat veel lager. Bovendien zijn de ruwe data erg waardevol wanneer je vreemde resultaten gekregen hebt, en je wilt de verwerking opnieuw doen, of op een andere manier. Veel leerlingenteams krijgen bij hun eerste elektronisch aangestuurde experimentjes een set van (op het eerste zicht) zinloze of absurde data terug. In dat geval is er nog hoop wanneer de ruwe data opnieuw kunnen verwerkt worden.

TEMPERATUUR

Omdat we in de ESERO vorming gebruik maken van de BMP280 sensor, zullen we meteen de verwerkte temperatuur, luchtdruk en hoogte krijgen op de SD kaart. Desondanks wordt hieronder toch uitgelegd hoe de kalibratie van temperatuur kan gebeuren wanneer je een analoge sensor gebruikt. Hoewel dit in de ESERO vorming niet nodig is, kan het toch erg nuttig zijn wanneer u zelf een elektronische meting wil doen op school.

Kalibratie van thermistor output

De output data van een simpele thermistor zijn variabele spanningen, met Volt als eenheid. Je kan deze omzetten in corresponderende temperaturen in graden Celsius.

Voor de meest voorkomende temperaturen op Aarde, kunnen we ervan uitgaan dat de thermistor output (gemeten spanning) en de temperatuur een lineair verband hebben. Het volstaat daarom enkele testjes vooraf te doen waarbij de temperatuur bekend is (gemeten met thermometer), en de corresponderende spanning over de thermistor gemeten wordt. Wanneer je deze punten uitzet op een grafiek, dan kan je de best passende rechte vinden die de relatie tussen outputspanning en temperatuur vervolledigt.

Nu willen we natuurlijk een formule, liever dan een lijn op een grafiek, om de gemeten spanningen om te zetten in temperaturen. De algemene wiskundige formule van een lineair verband (een schuine rechte) is als volgt:

$$y = mx + c$$

De waarde m bepaalt de helling van de rechte.

De waarde c bepaalt waar de lijn zijn nulpunt heeft (de x -as doorsnijdt).

Toegepast op de lineaire verhouding tussen thermistor spanning en temperatuur kan je stellen:

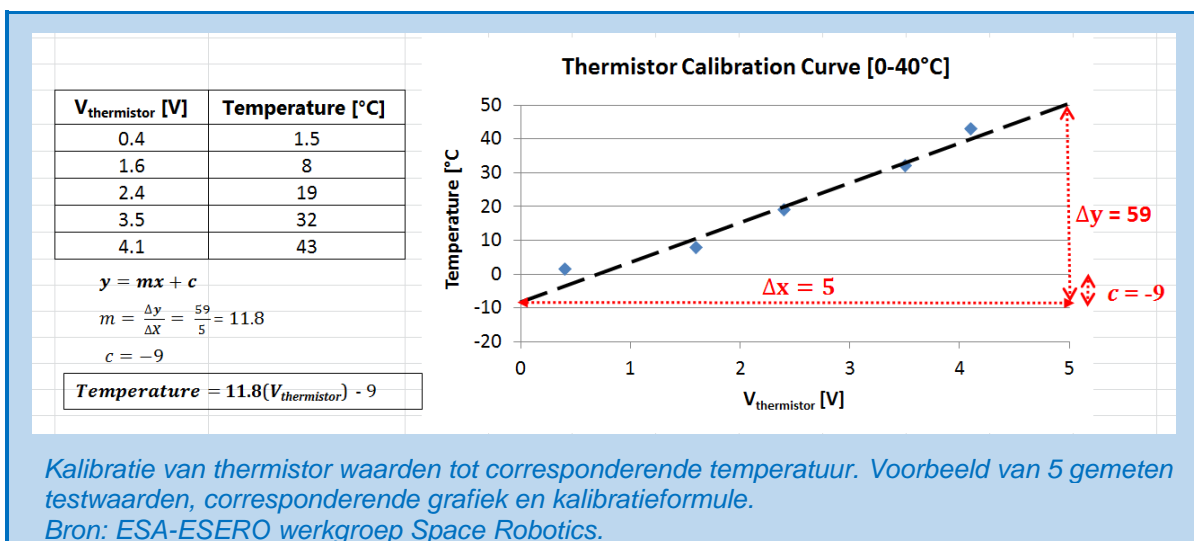
Y = temperatuur

X = thermistor spanning

In onze grafiek die gebaseerd is op enkele testmetingen, kunnen we de waarde m vinden als volgt:

$$m = \Delta y / \Delta x$$

Het volstaat nu om de Δy en Δx van de uiterste testwaarden in te vullen, zodat de helling m bekend is:



LUCHTDRIK EN HOOGTE

De hoogte berekenen op basis van luchtdrukwaarden

De relatie tussen atmosferische druk en temperatuur wordt gegeven door de formule:

$$P = 101325 (1 - 2.25577 \cdot 10^{-5} h)^{5.25588}$$

Hierbij is de luchtdruk P uitgedrukt in Pascal (Pa), en de hoogte h in meter boven zeeniveau (m). De waarde 101325 is een gemiddelde luchtdruk op zeeniveau. Je dient deze waarde aan te passen aan de luchtdruk op zeeniveau op het moment van uw vlucht. Deze waarde kan je vinden op de website van de meteorologische dienst van uw land.

Om de hoogte te vinden wanneer enkel de luchtdruk bekend is (we meten immers luchtdruk met onze sensor), wordt de formule als volgt herschreven:

$$h = \frac{10 \left(\frac{\log \log \left(\frac{P}{101325} \right)}{5.25588} \right) - 1}{-2.25577 \times 10^{-5}}$$

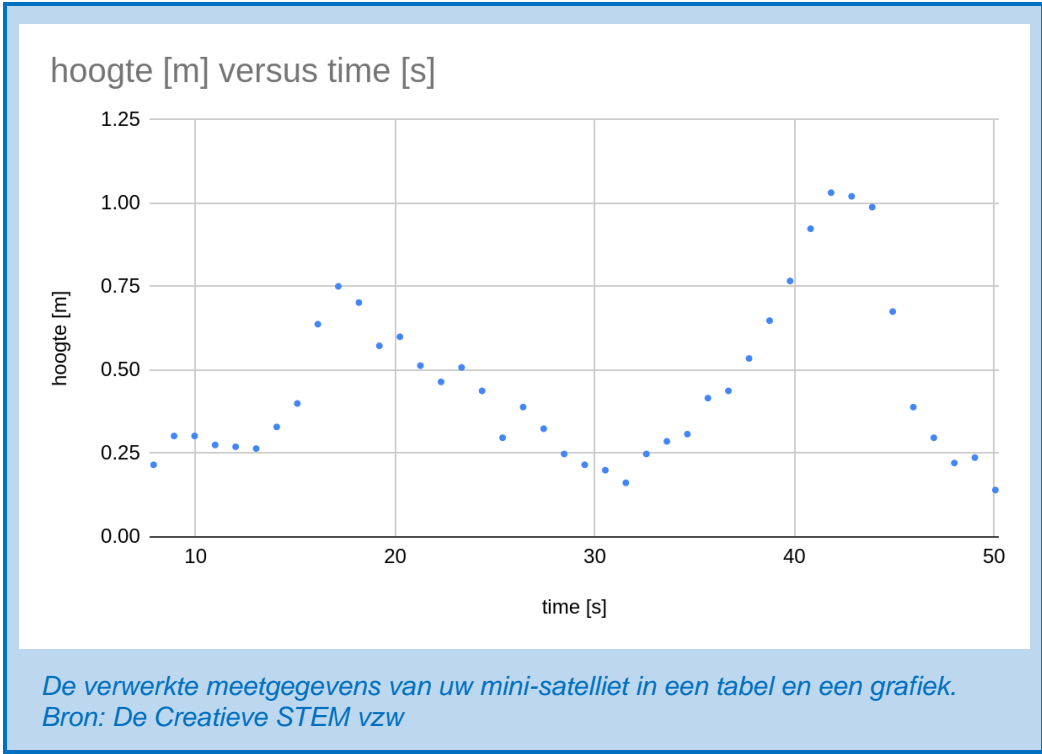
Zo kan je de hoogte van de vlucht per seconde reconstrueren. Alweer: vergeet niet de waarde 101325 aan te passen aan de luchtdruk op zeeniveau voor jou locatie en tijdstip. Een niet aangepaste waarde kan een fout in berekende hoogte opleveren van meer dan 10 meter.

De gegevens verwerken van de BM280

Let op: ook hier moet vooraf in de sketch de actuele luchtdruk op zeeniveau aangepast zijn aan uw locatie en tijdstip. In de sketch is verder alles voorzien zodat het output .txt bestand kan gelezen worden in excel. Het bevat per tijdstip (meerdere per seconde) metingen van de temperatuur en hoogtes (die laatste berekend op basis van de gemeten luchtdruk).

Op de SD kaart vinden we na de vlucht een txt bestand. Ga als volgt te werk:

- Sla het txt bestand op in uw computer
- Ga naar Windows Verkenner, en zoek het txt bestand
- Klik twee keer op het bestand: je krijgt de mogelijkheid om de naam van het bestand te veranderen
- Verander niet alleen de naam (naar keuze), maar verander ook de extensie .txt in .csv (comma separated values).
- Open dit bestand nu in MS Excel
- Nu heb je alle gegevens in een gemakkelijk te verwerken database. Pas eventueel de layout aan, zodat de data overzichtelijk weergegeven worden.
- Maak grafiek



Bronnen

- YouTube: Arduino voor beginners: Ohm my God
- Evans, Brian W. (2007). Arduino programming notebook.
<http://creativecommons.org/licenses/by-nc-sa/3.0/>
- Adafruit learning resources: <https://learn.adafruit.com>
- Okdo Getting started with Pico <https://www.okdo.com/getting-started/get-started-with-raspberry-pi-pico-gpio-micropython/>
- Digikey SD-card met Pico <https://www.digikey.be/en/maker/projects/raspberry-pi-pico-rp2040-sd-card-example-with-micropython-and-cc/e472c7f578734bfd96d437e68e670050>
- core-electronics pico guide <https://core-electronics.com.au/guides/raspberry-pi-pico/makerverse-micro-sd-adapter-micropython-guide/>

Bijlagen

Data types

Data type	RAM geheugen		Grenswaarden
boolean	1 byte		0 to 1 (True or False)
byte	1 byte = 8 bits	Numerieke waarden zonder decimalen na de komma	0 to 255
char	1 byte		-128 to 127
unsigned char	1 byte		0 to 255
int	2 bytes = 16 bits	Numerieke waarden zonder decimalen na de komma	-32,768 to 32,767
unsigned int	2 byte		0 to 65,535
word	2 byte		0 to 65,535
long	4 bytes = 32 bits	Numerieke waarden zonder decimalen na de komma	-2,147,483,648 to 2,147,483,647
unsigned long	4 byte		0 to 4,294,967,295
float	4 bytes = 32 bits	Numerieke waarden met decimalen na de komma	-3.4028235E+38 to 3.4028235E+38
double	4 byte		-3.4028235E+38 to 3.4028235E+38
string	1 byte + x		Arrays of chars
array	1 byte + x	Waarden die elk een index nummer dragen. Ze worden opgeroepen via dit indexnummers. De waarde voor elk index nummer moet vooraf bepaald worden door de gebruiker.	Collection of variables