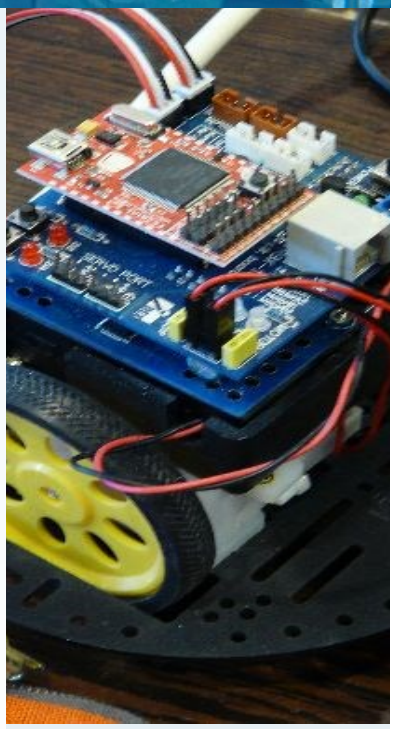
HCC ROBOTICA

Juni 2021
Jaargang 24
Nummer 2

ROBOBITS-92



VAN DE BESTUURSTAFEL

Beste lezer,

Het gaat weer de goede kant op. Echter zijn we nog niet van het virus en zijn mutaties af. Als je deel gaat nemen aan bijeenkomsten hou je dan nog steeds aan de regels!

Vooruitlopend op de HCC die vanaf 1 september weer start met fysieke bijeenkomsten, is er al een robotica bijeenkomst geweest in juni en er komen er meer. Zaterdag a.s. kun je na aanmelding vooraf weer naar de Dissel.

Verder heeft de Dissel aangegeven dat ze ook in augustus open zijn. Voor elke bijeenkomst geldt nu nog een maximum aantal van 30 deelnemers. Met de huidige regels gaat dat nog niet lukken in de Dissel, dus toegang op volgorde van aanmelden; je krijgt een bevestiging.

Of we ook al beginnen met presentaties na 11 uur hangt af van of er onderwerpen en sprekers beschikbaar komen. We gaan dan ook testen of we deze live kunnen streamen via wij.hcc.nl/

De HCC!kennisdag wordt georganiseerd in samenwerking met de Modelspoorbeurs op 17 juli. Hiervoor moet je je van te voren aanmelden met de keuze voor een tijdslot om het aantal bezoekers te reguleren. Als Robotica zullen we daar niet aanwezig zijn.

Het ziet er op dit moment dus ook gunstig uit voor de RoboRama 2021, maar dat is nog even afwachten.

Als je een beetje kunt schrijven of een leuk idee hebt om een artikel te maken voor in de Robobits, laat het ons weten! De Robobits ontleent zijn bestaansrecht aan artikelen voor en door leden. De volgende Robobits verschijnt eind september 2021.

Met vriendelijke groet,

Wim de Boer.

IN DIT NUMMER

Spot meet stralingsniveau

Spot, de beroemde robothond van Boston Dynamic, heeft een nieuwe baan: het inspecteren van de kernramp van Tsjernobyl om het stralingsniveau te meten.
Onderzoekers van de Universiteit van Bristol gaan de gegevens die Spot verzamelt gebruiken om 3D-warmtekaarten van de straling te maken.

Werken met servo's (deel 2)

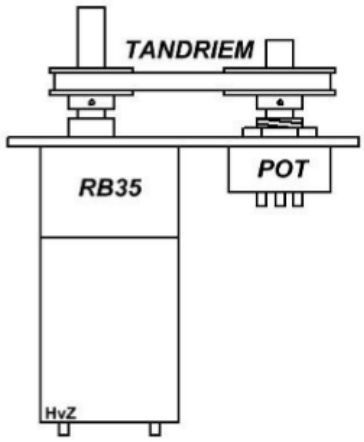
In deze Robobits gaan we verder met deel 2 van het artikel 'Werken met servo's en de besturing ervan'. In verband met de grootte van het artikel van Harry van Z. is het artikel verdeeld over drie Robobits: Robobits91, Robobits92 en Robobits93. Wil je alvast vooruitkijken dan kun je het artikel ook vinden op onze website.

Als je de servo nu aansluit op bijvoorbeeld één van de drie voorbeelden die ik gegeven heb, (de besturing met de NE555, de besturing met de controller, of de besturing met de window comparator) dan moet de servo stilstaan als de potmeter van de besturing in de middenstand staat. Als je nu de potmeter iets verdraait zal de servo langzaam gaan draaien, bijvoorbeeld rechtsom. Als je de potmeter wat verder draait gaat de servo sneller draaien. Draai je de potmeter nu terug naar de middenstand, dan zal de servo weer stil staan. Draai je de potmeter nu de andere kant op, dan loopt de servo linksom. Dus nu kan je de servo zowel rechtsom als linksom sturen en in toerental regelen.

En hoe komt het nu dat de servo zich anders gedraagt na de aanpassing? Dat komt omdat de interne potmeter zijn werk niet meer doet. De servo is zijn positie terugkoppeling kwijt. De servo wordt wel in toeren geregeld en een bepaalde kant op gestuurd, maar hij weet niet meer waar hij moet stoppen.

Maar zoals ik al eerder zei, kun je er op deze manier toch leuke dingen mee doen. Je zou er bijvoorbeeld een lineaire actuator mee kunnen maken, met bijvoorbeeld een spindel, een tandriem, tandheugel, enzovoort. Je zou bijvoorbeeld een nep hydraulische cilinder kunnen maken met een stuk draadeind en een kleine aangepaste servo, leuk om een hydraulische kraan mee te maken. En zo is er genoeg te verzinnen.

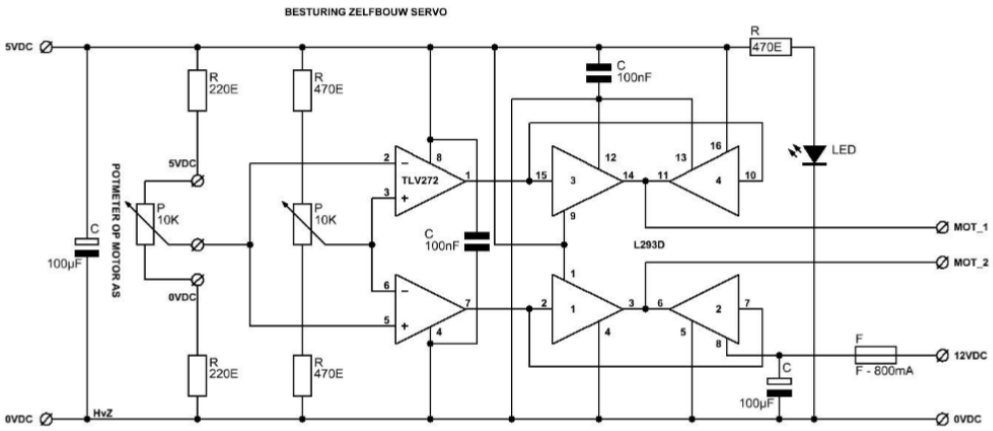
We hebben het over de in de handel zijnde servo's gehad, maar je kan er ook een zelf maken natuurlijk. Het voordeel is dan dat je een servo kan maken die enorm veel koppel kan leveren en het is tevens een leuk experimenteerproject. Zie hieronder een voorbeeld hoe je zoiets kan maken.



In het voorbeeld heb ik een tandriem gebruikt, maar het kan ook met bijvoorbeeld tandwielen. Of je zet de potmeter met een koppeling direct op de motor-as, alleen kan je dan slechter iets op de motor-as monteren. De motor kan je dan bijvoorbeeld aansturen met een L293D, dat is een H_brug in IC vorm. De brug kan je dan weer aansturen met bijvoorbeeld een opamp die als window comparator staat geschakeld , de potmeter (die op de motor zit) wordt dan teruggekoppeld naar de opamp.
Op deze manier kan je zelf een servo maken. Let er wel op dat de potmeter goed aangesloten moet worden, anders blijft de motor doorlopen en gaat de potmeter stuk.

EVEN VOOR DE GOEDE ORDE, DEZE ZELF GEMAAKTE SERVO'S KAN JE NIET MET EEN MODELBOUWZENDER- / ONTVANGER BEDIENEN.

Hieronder een schema van de besturing. Voor de hysterese kan je nog wat experimenteren met de 220 Ohm weerstanden of je zet -in plaats van één weerstand van 220 Ohm- een potmeter van 5K.

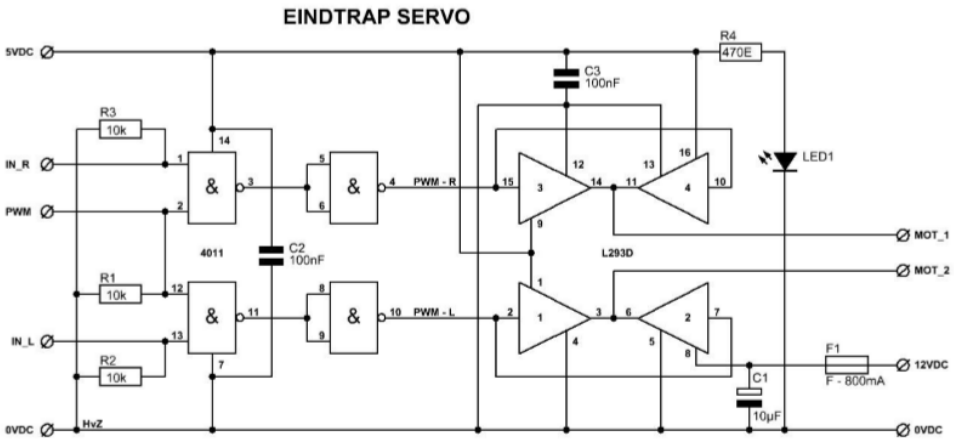


Deze besturing kunnen we ook met een microcontroller maken, we gebruiken hier een 16F887 voor maar dat mag ook een ander type zijn natuurlijk.
Op de volgende pagina is het programma hiervoor afgebeeld.

```
Device 16F887                       ; Processor type

Xtal 10                             ; Cristal 10Mhz

Asm                                 ; Config settings
CONFIG_REQ
__CONFIG _CONFIG1, HS_OSC & WDT_OFF & DEBUG_OFF & FCMEN_OFF & LVP_OFF &
IESO_OFF & BOR_OFF & CPD_OFF & CP_OFF & MCLRE_OFF & PWRTE_ON
__CONFIG _CONFIG2, WRT_OFF & BOR40V
EndAsm

All_Digital true                    ; Alle poorten digitaal

Declare Adin_Res = 8                ; resolutie 8 bits
Declare Adin_Tad = frc              ; set RC osc
Declare Adin_Stime = 50             ; sample tijd 5

Declare LCD_RSPin PORTD.2           ; Reset display poort D.2
Declare LCD_ENPin PORTD.3           ; Enable display poort D.3
Declare LCD_DTPin PORTD.4           ; Data display poort D.4 t/m D.7

Symbol UIT1 = PORTB.0               ; Naar ingang H_brug
Symbol UIT2 = PORTB.1               ; Naar ingang H_brug

Dim WAARDE_1 As Byte                ; Variabele in programma
Dim WAARDE_2 As Byte                ; Variabele van potmeter

Cls                                 ; Wis display

DelayMS 500                         ; Pauze 0.5 sec

Clear                               ; Wis geheugen

        ;543210                     ; Hulpregel poort A
PORTA  = %000000                    ; Maak poort A laag
TRISA  = %111111                    ; Poort_A I/O

        ;543210                     ; Hulpregel poort B
PORTB  = %000000                    ; Maak poort B laag
TRISB  = %000000                    ; Poort_B I/O

        ;76543210                   ; Hulpregel poort C
PORTC  = %00000000                  ; Maak poort C laag
TRISC  = %00000000                  ; Poort_C I/O

        ;76543210                   ; Hulpregel poort D
PORTD  = %00000000                  ; Maak poort D laag
TRISD  = %00000000                  ; Poort_D I/O

        ;210                        ; Hulpregel poort E
PORTE  = %000                       ; Maak poort E laag
TRISE  = %111                       ; Poort_E I/O

        ;76543210                   ; Hulpregel analoog
ADCON0 = %00000001                  ; ADCON0 register analoog

        ;543210                     ; Hulpregel analoog poort_B
ANSELH = %000000                    ; ANSEL register analoog poort_B
```

```
;-----------------------------
; PROGRAMMA ZELFBOUW SERVO.
;-----------------------------

positie_0:
    WAARDE_1 = 0

    WAARDE_2 = ADIn 0

    Print At 1,1,"GA NAAR POS 0"
    Print At 2,1,"POS"
    Print At 2,5,Dec ADIn 0, " "

    If WAARDE_1 > WAARDE_2 + 1 Then
     UIT1 = 1
      Else UIT1 = 0
       If WAARDE_1 < WAARDE_2 Then
        UIT2 = 1
         Else UIT2 = 0
       EndIf
    EndIf

    If WAARDE_2 = 0 Then
     GoTo wacht_0
    EndIf
GoTo positie_0

wacht_0:
    DelayMS 5000
    Cls
GoTo positie_1

positie_1:
    WAARDE_1 = 200

    WAARDE_2 = ADIn 0

    Print At 1,1,"GA NAAR POS 200"
    Print At 2,1,"POS"
    Print At 2,5,Dec ADIn 0, " "

    If WAARDE_1 > WAARDE_2 + 1 Then
     UIT1 = 1
      Else UIT1 = 0
       If WAARDE_1 < WAARDE_2 Then
        UIT2 = 1
         Else UIT2 = 0
       EndIf
    EndIf

    If WAARDE_2 = 200 Then
     GoTo wacht_1
    EndIf
GoTo positie_1

wacht_1:
    DelayMS 5000
    Cls
GoTo positie_2


positie_2:
    WAARDE_1 = 75

    WAARDE_2 = ADIn 0

    Print At 1,1,"GA NAAR POS 75"
    Print At 2,1,"POS"
    Print At 2,5,Dec ADIn 0, " "

    If WAARDE_1 > WAARDE_2 + 1 Then
     UIT1 = 1
      Else UIT1 = 0
       If WAARDE_1 < WAARDE_2 Then
        UIT2 = 1
         Else UIT2 = 0
       EndIf
    EndIf

    If WAARDE_2 = 75 Then
     GoTo wacht_2
    EndIf
GoTo positie_2

wacht_2:
    DelayMS 5000
    Cls
GoTo positie_3

positie_3:
    WAARDE_1 = 150

    WAARDE_2 = ADIn 0

    Print At 1,1,"GA NAAR POS 150"
    Print At 2,1,"POS"
    Print At 2,5,Dec ADIn 0, " "

    If WAARDE_1 > WAARDE_2 + 1 Then
     UIT1 = 1
      Else UIT1 = 0
       If WAARDE_1 < WAARDE_2 Then
        UIT2 = 1
         Else UIT2 = 0
       EndIf
    EndIf

    If WAARDE_2 = 150 Then
     GoTo wacht_3
    EndIf
GoTo positie_3

wacht_3:
    DelayMS 5000
    Cls
GoTo positie_4
```

Hier hoort weer een ander schema bij, er wordt wel weer een L293D gebruikt. Hieronder het schema.

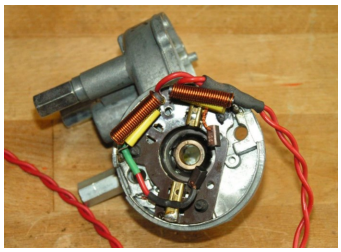


Poort B.0 en poort B.1 van de controller komen aan de klemmen IN_R en IN_L van de eindtrap. De PWM klem komt aan de 5VDC te zitten, omdat deze niet gebruikt wordt in het programma. Als de PWM functie in het programma erbij gezet wordt (dan moet het programma aangepast worden) dan kan de PWM klem van de eindtrap op de HPWM pin (poort C.1 of poort C.2) van de controller aangesloten worden. Op deze manier kan dan ook het toerental van de servo geregeld worden. Let er wel op dat de PWM frequentie niet groter dan 5KHz mag zijn voor de L293D.

Op de klemmen MOT_1 en MOT_2 wordt de motor aangesloten, op de klemmen 5VDC en 0VDC komt de 5VDC voeding voor de elektronica en op de klemmen 12VDC en 0VDC komt de 12VDC voeding voor de motor. De externe potmeter (de potmeter die op de motor-as zit) komt aan de analoge ingang A.0 (ADIN 0). Denk er ook hier weer aan dat de potmeter goed aangesloten moet worden.

Het programma loopt automatisch vier posities af, maar dat kan naar eigen wens aangepast worden natuurlijk. Let er ook op dat de voorbeelden met de L293D alleen geschikt is voor kleine motoren, zoals de RB35, dus motoren die meer dan 1.5A trekken gaan niet met deze schakeling.

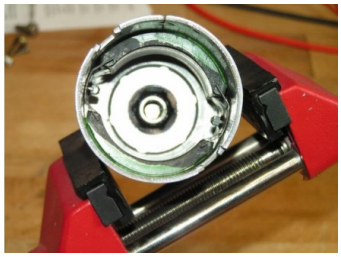
We gaan nog een servo maken van een ruitenwissermotor. Maar we moeten dan eerst de motor helemaal aanpassen. Ik ga met behulp van wat foto's uitleggen wat er moet gebeuren.



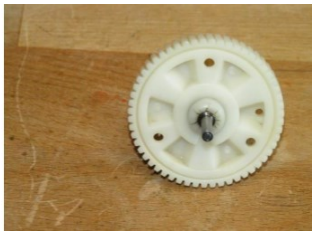
Ik had deze motor al een keer aangepast, dus heb hem voor wat foto's maar even uit elkaar gehaald. Op de foto kan je de borstelbrug zien, deze moet aangepast worden. Als je onder het linker spoeltje kijkt (één van de twee ontstoringsfilters) dan zie je daar twee sleufjes in de borstelbrug zitten, daar heeft nog een koolborstelhouder gezeten. Die zit er standaard op bij de meeste ruitenwissermotoren en is voor het hoge toerental. Die heb je niet nodig, dus die haal ik dan ook altijd weg. Zoals je kan zien op de foto heb je nu twee borstels tegenover elkaar zitten.
De spoeltjes staan in serie met de borstels en de nieuwe aansluitdraden.

De condensatoren (de gele blokjes) zitten aan de ene kant aan de spoel en aan de andere kant aan de behuizing, maar daar is in principe niets aan veranderd. Dus als je zo'n motor aan wil passen wijst alles eigenlijk voor zich. Hieronder de foto's van het anker en van het huis met de permanentmagneten. Zoals je kan zien op de foto's is zowel het anker als het huis nog in prima staat.

Aan het anker hoeft dus niets te gebeuren, tenzij de collector slecht is. Die kan je eventueel iets opschuren (op bijvoorbeeld de draaibank) of iets afdraaien. Hierbij is wel de nodige voorzichtigheid geboden, want als je niet oppast maak je het alleen maar erger. Ook moet je de lamellen (dat zijn de stukjes tussen de koper vlakken van de collector) iets wegfrezen, dus als je de spullen daar niet voor hebt kan je er beter niet aan beginnen.



Dan hebben we het tandwiel waar de worm in loopt (het wormwiel) Zie foto's hieronder.



Daar heb ik een as van 4mm op gemaakt om bijvoorbeeld een encoder schijffje of een potmeter op te bevestigen. Ik heb op de draaibank een gat van 3,9mm in de as geboord en uitgeruimd naar 4mm. In het asje van 4mm heb ik met een centerpunt kleine putjes geslagen, zodat deze strak in het gat van 4mm past. Het asje is met Loctite 638 in het gat geperst en zit dus muurvast. Op deze manier kun je makkelijk een potmeter op de as monteren. Je kan de potmeter ook op dezelfde wijze monteren als in het voorbeeld met de RB35 motor natuurlijk, maar ik vind het op deze manier mooier. Je kunt op deze manier ook makkelijker iets op de hoofdas monteren.

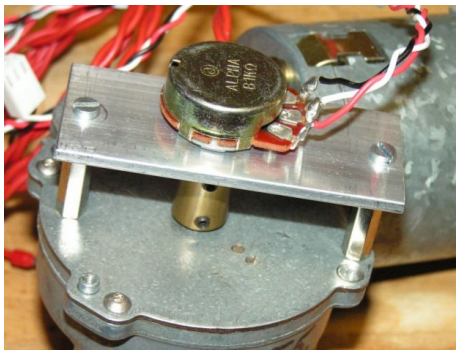
Hiernaast heb je een foto van het geheel weer gemonteerd.
Je ziet op de foto de afdekplaat zitten (dat is de plaat die over de worm en het wormwiel zit). Daar zie je naast de schroefjes putjes zitten. Daar zit de afdekplaat standaard mee vastgeklonken om de motor uit elkaar te halen moet je die dus wegboren. Om de plaat weer vast te zetten heb ik nieuwe gaatjes geboord en schroefdraad (M3) getapt. Op deze manier kan je de afdekplaat weer goed vast zetten.

Je ziet ook een sleuf gat in de afdekplaat zitten; daar hebben de sleepcontacten aangezeten. Die zijn in kunststof ingegoten en in de afdekplaat geperst. Die moet je dus ook verwijderen, want die worden niet meer gebruikt. Op het wormwiel zaten ook koperplaatjes, ook die mag je verwijderen.

Om het anker in de borstelplaat te plaatsen, plak ik één borstel even met een stukje tape vast. De andere borstel houd ik zelf even op zijn plek. Op deze manier kan je het anker makkelijk in de plaat monteren. Niet vergeten om het plakband weg te halen.

Bij het anker in het magneethuis plaatsen moet je wel even goed opletten. Je moet het anker goed vasthouden anders vliegt hij je magneethuis in, met het gevolg dat je borstels weer uit je borstelhouders schieten en dan kan je weer opnieuw beginnen. Zelf houd ik hem altijd bij de worm vast, dan heb je lekker grip op het geheel en kan er weinig gebeuren. Op deze manier kan je een ruitenwissermotor dus ombouwen. Niet alleen om er een servo mee te maken, maar ook voor andere toepassingen. Het zijn sterke motoren en ze kunnen tegen een stootje.

Hieronder een foto met de potmeter op de motor gemonteerd.



De potmeter is op een aluminium hoeklijntje gemonteerd, de verbinding tussen de potmeter en de as is met een messing koppeling gedaan. Deze koppelingen zijn zo te koop, dus die hoef je niet zelf te maken. De koppeling heeft een gat van 4 en 6mm. Het geheel staat op twee afstand houders en is daarmee op de motor gemonteerd. Let er wel op dat alles precies uitgelijnd is, anders komt er teveel zijdelingse kracht op de potmeter. Het geheel is nu klaar om gebruikt te worden.

Om een goed werkend geheel te krijgen moet er zo min mogelijk speling op de worm en wormwiel zitten. Vaak zit er op de kopse kant (in de behuizing boven de worm) van de motor een stelschroef, daar kan de axiale speling van het anker en worm mee afgesteld worden. Die speling moet minimaal zijn, anders gaat de motor staan oscilleren. Stel de schroef ook weer niet te strak af, anders loopt de motor te zwaar.

Voordat je de motor aansluit op de besturing kan je het beste één van de stelschroeven (waar de potmeter mee op de as zit) even losgezet worden. Dit is om te controleren of de potmeter goed aangesloten zit. Als je de bedienpotmeter verdraait moet de motor in een bepaalde richting gaan lopen. Als je nu de potmeter die op de as zit in dezelfde richting draait als de motor loopt, moet de motor op een bepaald punt stoppen. Doet de motor dat niet dan moet je de aansluitdraden van de motor omdraaien. Als alles wel goed werkt, maar je wil dat de motor de andere kant op moet lopen, dan moet je de aansluitdraden van de motor omdraaien en de plus en min draden van de potmeter.

(eventueel vooruit lezen via deze link of wachten tot volgende Robobits.)

PicoPi Robot: Lijnvolgen met een camera

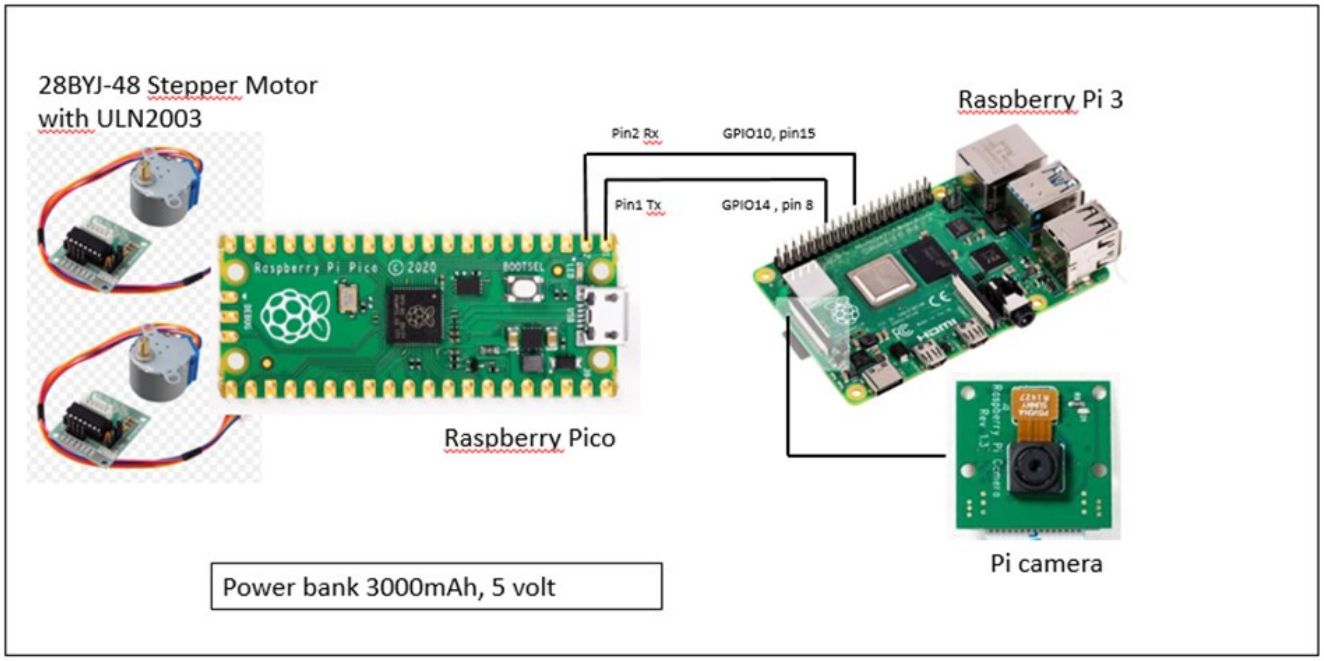
Ik ben al een aantal jaar met veel plezier betrokken bij de HCC Robotica Club. De club is een inspiratiebron en biedt mij kennis en ideeën voor de uitvoering van mijn hobbyprojecten.
Zo moest ik constateren dat ik nog nooit zelf een robot heb gebouwd en heb deelgenomen aan de jaarlijkse RoboRama wedstrijd. Misschien komt daar dan toch verandering in?

In dit artikel beschrijf ik de bouw van een robot die in staat is om met een camera een lijn(baan) te volgen. Het project is nog niet af maar resulteert tot nu toe in een werkend prototype.

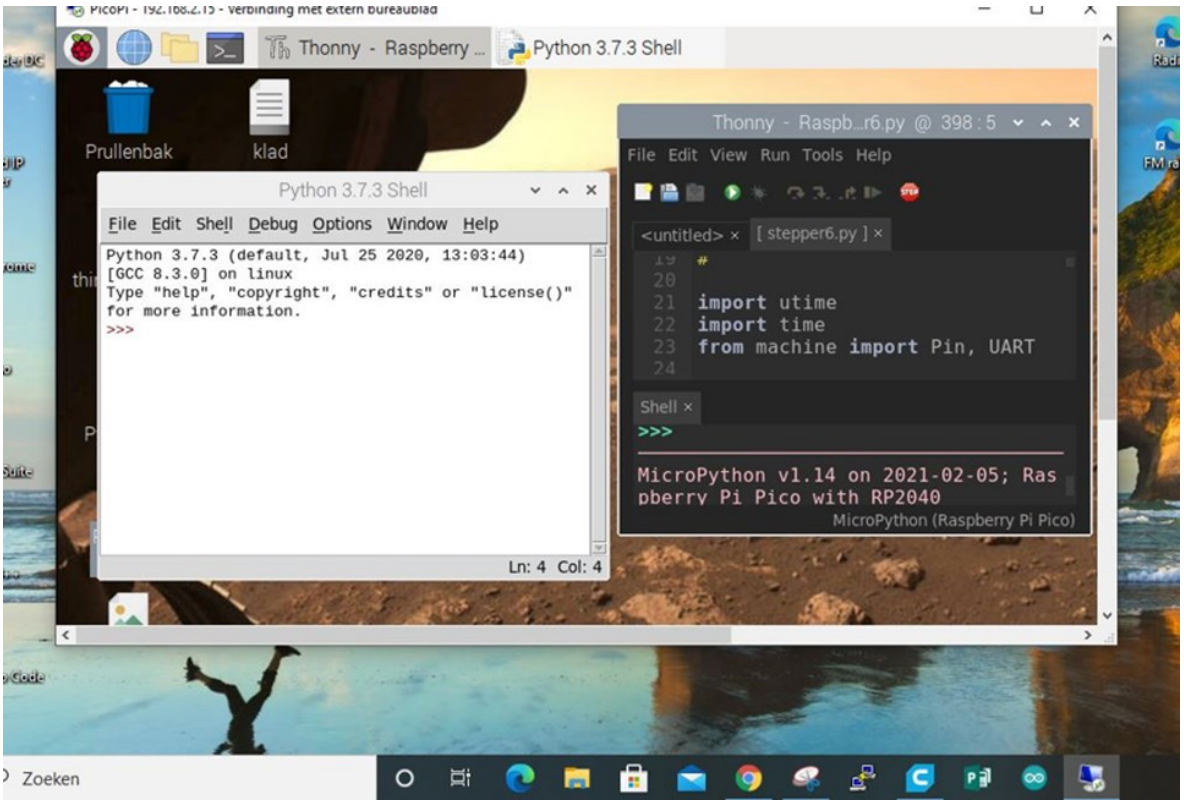
Raspberry Pi + Pico = PicoPi

In een eerste kennismaking met de beeldanalyse software 'OpenCV' op een Raspberry Pi3 met een camera bij een ander project was ik verbaasd over de performance van de microcomputer tijdens het analyseren van beelden. Met gemak kon de Pi3 met een framerate van 32 fr/s de beeldjes met formaat van 1024x768 pixels bewerken met nauwelijks haperingen of vertragingen.
Ook het feit dat de moedertaal van de Pi3 Python is en de Pi3 eenvoudig headless is te gebruiken (bv. via extern bu-

De aandrijving van mijn robot bestaat uit twee stappenmotoren. De Pi3 zou die kunnen aansturen. Echter uit interesse in vernieuwing en uitdaging heb ik gekozen om de stappenmotoren aan te sturen met een nieuwe Raspberry Pico microcontroller, gebaseerd op de RP2040 chip. Via een seriële verbinding tussen de Pi3 en de Pico worden stuurcommando's voor de stappenmotoren naar de Pico gestuurd. De stuurcommando's worden door de Pi3 'bedacht'.
De voeding van 5 volt wordt geleverd door een telefoon power bank van 3000mAh. Deze voedt de Pi3 en de Pico en levert voldoende om de robot meerdere uren te laten rijden. Schematisch ziet het er ongeveer zo:



*De Pi3 wordt ingelogd op het (thuis) Wifi netwerk. Via het commando **install apt-get xrdp** op de Pi3 wordt het (linux) bureaublad ingesteld. Nu kan vanaf een (windows) computer of laptop in hetzelfde Wifi netwerk via een 'bureaublad sessie' de Pi3 headless over worden genomen. Het programmeren, testen en uiteindelijk het lijn volgen kan volledig draadloos vanaf de laptop worden beheerd. Ook de videobeelden zijn via deze connectie goed op het bureaublad te volgen. Een ideale ontwikkel– en testomgeving.*
Onderstaande foto geeft een indruk over hoe ik op mijn windows 10 laptop te werk ga, met op de voorgrond de bureaublad sessie van de Pi3 met daarin de geopende Python IDE en de Thonny programmeeromgeving van de Pico.



Pico

De Raspberry Pi Pico is programmeerbaar in de talen C en MicroPython. Ik heb gekozen voor de laatste.
Wanneer je op de Pi3 de laatste versie van Thonny installeert (a Python IDE for beginners) dan is het eenvoudig om de Pico te programmeren via de USB poort van de Pi3. Ook is er een Windows versie van Thonny zodat je ook via windows 10 en een USB poort de Pico kunt programmeren.

*Via een tweede seriële verbinding communiceert de Pi3 met de Pico die stuurcommando's ontvangt. Zo heb ik commando's geformuleerd om de twee stappenmotoren een gedefinieerde beweging te laten maken. Bijvoorbeeld: '**MF100**' (move forward 100 steps) of '**TL100**' (turn left 100 steps).*

Met de volgende micro-python code op de Pico is de seriële verbinding eenvoudig gerealiseerd:

```
    run = True
    while run:
        if uart.any():
            received_data = uart.readline()
            s=received_data.decode()
            #print(s, len(s))
            if len(s) == 6 :
                run = False

    utime.sleep(0.01)
    return
```

Vervolgens worden de stappenmotoren aangestuurd met de volgende micro-python code:

```
#==========movement=================
def turnL(steps):
    for x in range(steps):
        step_motorL(1)
        step_motorR(-1)
    return

def turnR(steps):
    for x in range(steps):
        step_motorL(-1)
        step_motorR(1)
    return

def forward(steps):
```

```
def step_motorR(count):
    # count is het aantal stappen rechtsom
    # -count is het aantal stappen linksom
    global current_stepR
    while (count != 0):
        if (count < 0):
            high_motorR = stepRR[current_stepR]
        else:
            high_motorR = stepR[current_stepR]
        set_motorR_low(motorR)
        set_motorR_high(high_motorR)
        current_stepR += 1
        if current_stepR == len(stepR):
            current_stepR = 0
        time.sleep(speed)
        if (count < 0):
            count +=1
        else:
            count -=1
    return
```

Het enige wat nu nog moet gebeuren is het samenstellen van de juiste motor commando's om PicoPi een lijn te laten volgen!

Beeldanalyse

*Op de Pi3 draait het Python programma **PicoPy.py**. Ik zal in grote lijnen uit te leggen hoe ik ben gekomen tot dit resultaat.*

De camera is een Pi3 camera die zeer veel wordt toegepast bij Raspberry Pi projecten. De camera ingesteld op 320x240 pixels en framerate van 32 beeldjes per seconde. Voor een eenvoudige lijn detectie moet dit voldoende zijn. In de rawCapture variabele komt het te bewerken video frame te staan:

```
yy  = 240  # x-as
xx  = 320  # y-as
# initialize parameters
camera          = PiCamera()
camera.resolution = (xx, yy)
camera.framerate  = 32
rawCapture      = PiRGBArray(camera, size=(xx, yy))
```

De main loop van het programma is eenvoudig; bepaal van elk frame uit de rawCapture de positie van de te volgen lijn en blijf vervolgens op de weg!

```
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):

    image = frame.array   # maak beeld van de omgeving
    getLane(image,1)      # bepaal de lijn en toon beeld
    controlLane(setpunt,afwijking,param)  # blijf op de weg !
```

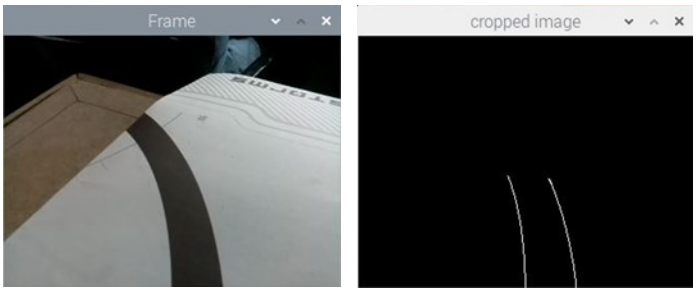
*Nu moet **OpenCV** aan het werk.*
In de getLane() procedure wordt er van elk frame een grijswaarde beeld gemaakt. Vervolgens wordt het plaatje 'geblurred' en de informatie uit het plaatje wordt omlijnd via de Canny functie.
Omdat we straks alleen in een lijn geïnteresseerd zijn en niet zozeer in de naaste omgeving kunnen we ons beeld beperken tot een deel van het frame, namelijk tot het region of interest. Dit beeld wordt opgeslagen in de cropped_image:

```
    gray_image    = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY )
    blur_image    = cv2.GaussianBlur(gray_image,(5,5),0)
    canny_image   = cv2.Canny(blur_image,100,200)
    cropped_image = region_of_interest(canny_image,
                    np.array([region_of_interest_vertices],np.int32),)
```

Het resultaat van de bewerkingen van de plaatjes met OpenCV is sterk afhankelijk van de hoeveelheid omgevingslicht. Ook zijn er veel parameters die bij de aanroep van de OpenCV procedures moeten worden opgegeven . Toch is al snel een resultaat te behalen zoals hieronder te zien is:

Dit resultaat toont een voorbeeld van een videoframe dat na de 'blur' en na de Canny functie de omlijningen laat zien van het frame. De witte driehoek is het gekozen gebied als 'region of interest'.

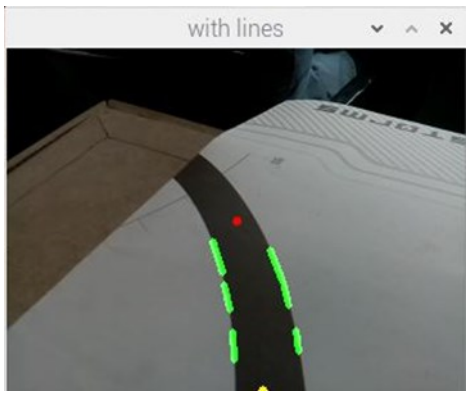
In geval we gaan lijn volgen en indien we de camera richten op een stukje 'lijn volg baan' dan is het volgende zichtbaar te maken met de bovengenoemde getLane() procedure:



Nu komt het belangrijkste onderdeel van het hele lijn volg programma. Het 'Hough filter'.
Dit rekenkundig filter is in staat om lijnen, cirkels en ellipsen af te leiden uit een bewerkt videoframe. Wij zijn geïnteresseerd in lijnen maar hetzelfde filter is te gebruiken om ook blikken waar te nemen tijdens een RoboRama wedstrijd.
Het aanroepen van het Hough filter met de cropped_image gaat als volgt:

```
228     # zoek lijnen
229     # (default 6,pi/60, 160,100,10 )
230     # werkend 6,pi/60, 160,15,10 )
231     lines = cv2.HoughLinesP(cropped_image,
232                             6,
233                             np.pi/60,
234                             threshold = 10,
235                             minLineLength=15,
236                             maxLineGap=10)
237
238     if lines is not None:
```

Zoals je ziet is het maar één aanroep maar met veel parameters. Ik heb veel tijd moeten besteden aan het (proefondervindelijk) vaststellen van enkele parameters om uiteindelijk een paar lijnen vast te bepalen uit het frame. In onderstaande plaatje zijn de gevonden lijnen door het Hough filter in het groen weergegeven in de region of interest in het originele frame:

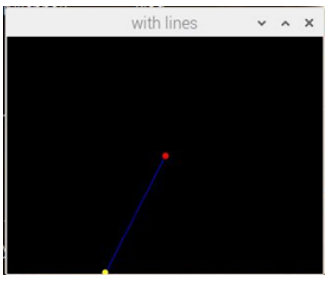


We willen graag de lijnen zoals die in het beeld worden getoond in een rekenkundige vorm hebben zoals y=ax+b.
Deze lijnen zijn eenvoudig af te leiden als je bedenkt dat het frame uit een x,y-coördinatenstelsel bestaat van 320x240 pixels .

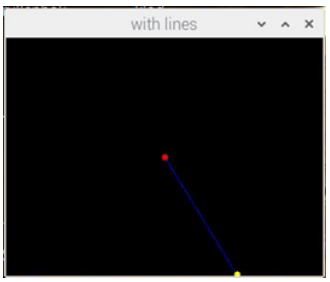
Het Hough filter vindt dus de linker en rechter kantlijnen van de zwarte te volgen lijn.

Ik bereken nu een gemiddelde lijn die tussen de groene lijnen in loopt (blauwe lijn).
De gemiddelde lijn loopt door het rode (middel) punt en snijdt de x-as in het gele punt. Het gele punt zal dus per beeldframe wijzigen en over de x-as heen schuiven, afhankelijk van de afwijking van de robot ten opzichte van de te volgen lijn.
Ik gebruik dit gele punt (x-waarde, y=0) als mijn setpunt voor de motorbesturing.

Afwijking = bijna 0: weinig regelactie; blijf rechtuit rijden.



Afwijking ten opzichte van het midden: stuur de robot naar rechts



Afwijking ten opzichte van het midden: stuur de robot naar links

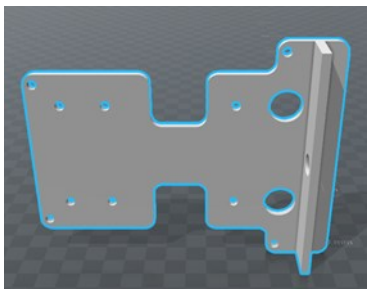
Deze manier van regelen betekent dus dat de robot aan het begin van de lijnvolg sessie precies in het midden van de te volgen lijn moet worden gepositioneerd.

De procedure controlLane() regelt dit proces met behulp van een PID regelaar. Het instellen van de PID parameters is een uitdaging apart. Uiteindelijk heb ik resultaat met vooral P- en D actie (en I-actie op nul). De output van de regelaar is dan een maat voor het aantal stappen en richting voor de stuurcommando's:

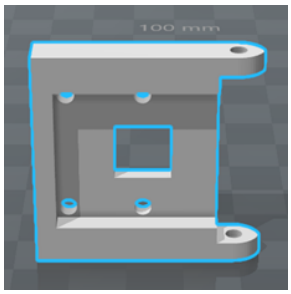
```
266  #---------------------------------------
267  def controlLane(setp, meting, parameters):
268      Kp, Ki, Kd, Pb = parameters
269      global prevT
270      global eintegral
271      global dedt
272      global eprev
273
274      currT = round(time.monotonic() * 1000)
275      deltaT = (currT - prevT) # ordegrootte = 300 ms
276      prevT = currT
277      error = meting - setp
278      dedt = (error - eprev) / deltaT
279      eintegral = eintegral + error * deltaT
280      eprev = error
281
282      out = Kp * error +  Ki * eintegral + Kd * dedt
283
```

De robot PicoPi

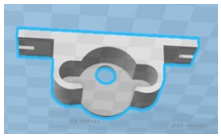
Nu moet alles bij elkaar komen. Enkele van de lezers zullen het frame van de robot misschien herkennen uit een eerder gezamenlijk HCC Robotica project (met dank aan Henny). Het frame wordt nu hergebruikt. Samen met wat onderdelen, geprint met een 3D printer, is alles opgebouwd en samengevoegd met als resultaat het volgende prototype 'pronkstuk' op de volgende bladzijde:

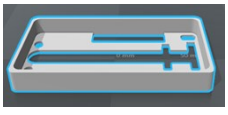


De bodemplaat waarop de Raspbery pi en de Pico wordt gemonteerd.

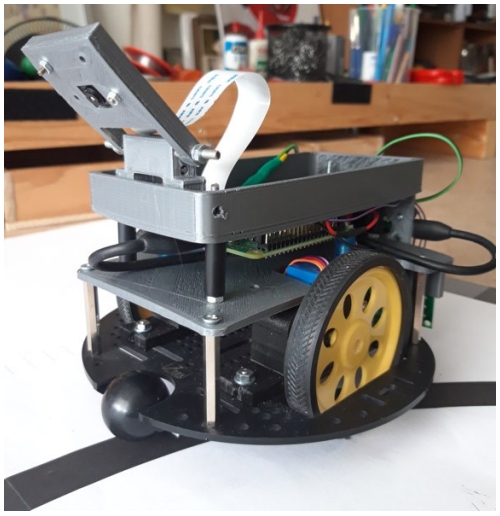


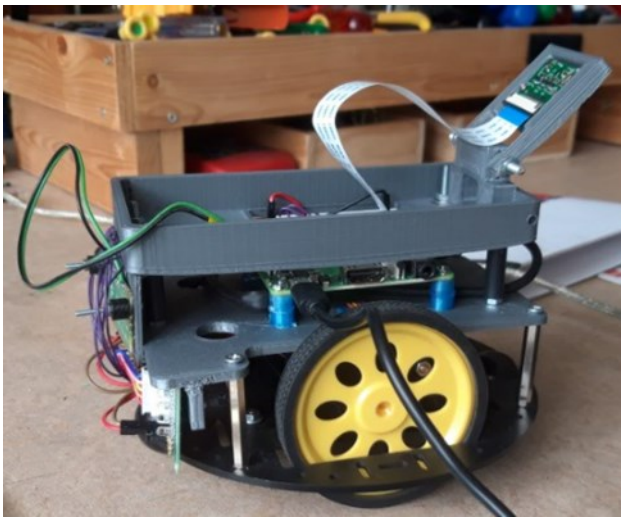
De camera houder.



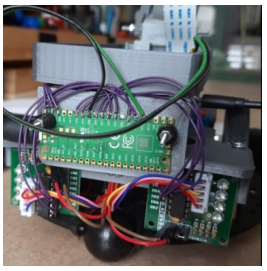


De motor behuizing en het bakje voor de telefoon power bank





Conclusie



Achterkant met de Pico

Het lijn volgen op basis van een camera en OpenCV is zeer goed mogelijk. Ook het opsporen van blikken zou op basis van OpenCV en een Raspberry Pi met camera mogelijk moeten zijn.

Seriële communicatie tussen een Raspberry en een Pico is eenvoudig te realiseren. Het programmeren van de uart moet echter beter kunnen wat performance betreft. De buffer kreeg ik niet goed geleegd (waarschijnlijk door beperkte programmeer ervaring). Nieuwe data wordt nu toegevoegd aan data die nog in de buffer zit. Daarom heb ik een beetje vreemde (hulp) constructies in de pico/micro-python software toegepast.

Bij gebruik van een Raspberry Pi4 (de krachtige opvolger van de Pi3) is het mogelijk om hogere resolutie plaatjes te verwerken. Of dit leidt tot betere resultaten zal moeten worden uitgezocht. Ook kan de 'region of interest' worden vergroot.

Het hier getoonde prototype werkt maar er zijn nog heel veel verbeteringen te bedenken:

- De gebruikte stappenmotoren zijn zwak en draaien erg langzaam. De robot is (veel) te traag.

- Is de keuze van het setpunt wel ideaal?

- Het printen van onderdelen voor de robot gaat uitstekend met een 3d printer maar besteed dan wel veel tijd aan het ontwerp en design van de onderdelen van de robot voordat je werkelijk begint te bouwen.
Mijn picoPi kwam tot stand via een 'organisch groeimodel' ; bouwen met voortschrijdend inzicht..
Dus als je goed kijkt zie je wel een extra geboord gat of iets wat niet helemaal recht zit!

Juni 2021. Z. Otten

HCC!Robotica ig

HCC-Robotica is een interessegroep die zich bezig houdt met het ontwikkelen, ontwerpen, programmeren en bouwen van elektronica en mechatronica, toegepast op robots. Deze meer of minder intelligente en autonome robots en machines met verschillende sensoren, actuatoren, processoren en bewegende onderdelen worden onder andere ingezet bij de jaarlijkse georganiseerde Roborama wedstrijden. Wij komen elke eerste zaterdag van de maand bijeen in dorpshuis de Dissel te Hooglanderveen. Kennis delen, kennis vergaren, presentaties en workshops bijwonen zijn terugkerende activiteiten tijdens deze bijeenkomsten.

U bent van harte welkom!

Agenda

Goed nieuws: de **HCC!kennisdag op 17 juli 2021** is definitief. Nadat eerdere Kennisdagen helaas om de bekende reden moesten worden afgezegd, kunnen we nu in Expo Houten weer een mooie dag organiseren voor alle HCC-leden en introducees.

Veel HCC!interessegroepen bereiden zich voor op de Kennisdag en er staat ons een indrukwekkend programma te wachten.

Samen met de Modelspoorbeurs
De HCC!kennisdag wordt georganiseerd in samenwerking met de Modelspoorbeurs, dus die twee evenementen kun je mooi combineren.

Binnenkort gaat de aanmeldprocedure van start. Houd daarvoor de HCC-nieuwsbrief en de website in de gaten van de hcc. Uiteraard houden we rekening met alle nog geldende coronamaatregelen.

Samen met Expo Houten plant HCC dit jaar twee keer een Kennisdag. Zet daarom ook **9 oktober** alvast in je agenda.

Discussiegroepen
HCCROBOTICA:

http://groups.google.nl/group/hcc_robotmc

Blogs
http://zotten.wordpress.com/

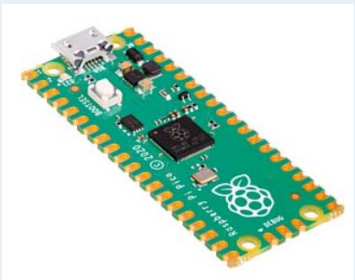
https://avretro.wordpress.com/
http://www.robotblog.nl/

Facebook

HCC!Robotica ook op Facebook.

Gewoon om te laten weten, dat wij ook op Facebook actief zijn.

Raspberry Pi Pico



De Raspberry Pico is een klein, snel en veelzijdig bord gebouwd op basis van een RP2040.
Ontworpen door Raspberry Pi, is de RP2040 voorzien van een dual-core Arm Cortex-M0 processor met 264 KB interne RAM en ondersteuning voor maximaal 16 MB off-chip Flash.
Ondersteuning van I2C, SPI en **micro-python**. Programmeren van een microcontroller is nog nooit zo eenvoudig geweest!

HCC!Robotica ig

Dagelijks bestuur:
Voorzitter : Wim de Boer
Secretaris : Rob van der Ouderaa
Penningmeester : Ed Buzzi

Het Kernledenbestand ziet er als volgt uit en zal het dagelijks bestuur ondersteunen:
Redacteur Robobits : Zeno Otten
Bibliotheek beheerder - webmaster : Bert Berrevoets
Wedstrijd coördinator RoboRama : Bert Ruben
Contactpersoon externe evenementen : Kees Kerling
Technisch advies : Joep Suijs, Edith van Putten