

9 Seriële communicatie

9.1 Einddoel

Je gaat een Raspberry Pi Pico en een Arduino-Uno met elkaar laten praten via seriële communicatie.

Als je klaar bent met de opdracht lever je in It's learning in:

- De definitieve python code waar de verschillende opdrachten in verschillende functies staan.
- Een foto van de opstelling waarop duidelijk te zien is hoe alles in aangesloten.
- Een mp4-filmpje met een demo van je programma.

Als dit gedaan is laat je de opdracht zien aan de docent zodat deze de opdracht kan goedkeuren.

9.2 Kennis

Voor deze opdracht heb je basis python en microPython kennis nodig. Verder de kennis hoe je een breadboard gebruikt.

9.3 Benodigheden

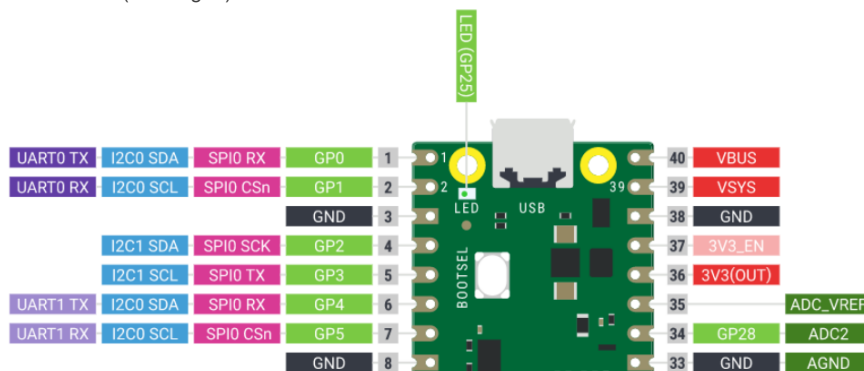
Voor deze opdracht heb je de onderstaande materialen nodig. Controleer aan het begin of al deze spullen aanwezig zijn. Bij het opruimen dien je weer te controleren of alles aanwezig is. Indien er iets defect is geraakt moet je de docent op de hoogte brengen.

- Arduino Uno
- Raspberry Pi Pico – H met breakout board
- UBS-usb mini kabel en USB-A naar USB-B kabel
- Level shifter TXS0108E
- 5 drukknoppen
- 4 leds met weerstanden 100 Ohm

9.4 Opdracht

We gaan gebruik maken van de ingebouwde UART (Universal Asynchronous Receiver/Transmitter) protocol van de Raspberry pi pico. Met dit protocol kunnen we gemakkelijk met andere microcontrollers en ic's communiceren.

Zoals je bij andere opdrachten hebt gezien is i2c ook een serieel communicatieprotocol. Die waren synchronisch (synchronous) zodat ze een kloksignaal nodig hadden (de SCL pin). We zien deze pinnen in het paars in onderstaande tekening. Ze hebben de naam Tx dat staat voor transfer(verzenden) en Rx voor Receive (ontvangen)



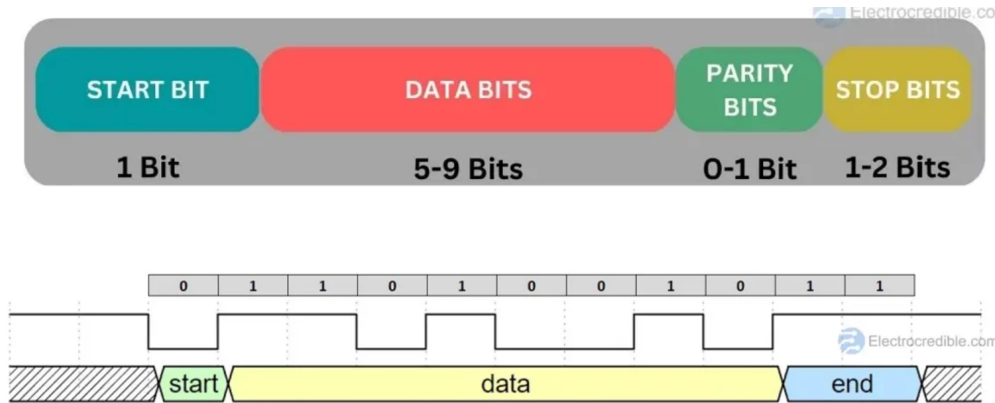
9.4.1 Wat is Seriële communicatie?

In tegenstelling tot parallelle communicatie waar over verschillende draadjes tegelijk de data verstuurd wordt (de enen en nullen) wordt bij seriële communicatie maar 1 draadje gebruikt waar de enen en nullen achter elkaar verstuurd worden. 4 bits communicatie heeft dan 4 draadjes bij parallel nodig, en maar 1 bij serieel. 8 bits heeft 8 draadjes nodig, en bij serieel nog steeds maar 1.

Voorbeelden van seriële communicatiemethoden zijn i2c, SPI en USB. De datalijnen voor seriële communicatie worden ook wel de seriële bus genoemd.



Om het verzenden goed te laten verlopen volstaat het niet om alleen enen en nullen over de lijn te sturen. We willen ook zeker weten dat wat we versturen ook op de correcte manier aan de andere kant aankomt. Hiervoor sturen we ook parity bits mee om dit te controleren. Als de data niet goed is aangekomen dan wordt er door de ontvanger gevraagd om het nogmaals te sturen. Wij hoeven ons hier helemaal niet druk om te maken, dit wordt allemaal voor ons gedaan door het protocol.



Zoals al verteld is wordt in dit protocol geen kloksignaal meegestuurd. Van tevoren spreken verzender en ontvanger de kloksnelheid (baud rate) af hoe snel de data verstuurd wordt. Als we dus twee apparaten met elkaar willen laten communiceren via seriële communicatie dan moet de Rx aangesloten zijn op de Tx van het andere apparaat, en andersom.

9.4.1.1 Code op de Raspberry pi pico

De onderstaande code zorgt ervoor dat op regel 8 de Raspberry pi pico een 't' wordt verstuurd van de Tx poort naar de Rx poort van de Arduino Uno.

De pico gaat dan kijken of er een bericht voor hem aanwezig is. Op regel 9. Als dat bericht er is wordt deze op regel 10 uitgelezen. Als dit een 'm' is dan gaat de led aan (of uit als deze al aan was).

```
1 from machine import Pin,UART
2 import time
3 uart = UART(1, baudrate=9600, tx=Pin(4), rx=Pin(5))
4 uart.init(bits=8, parity=None, stop=2)
5 led = Pin("LED", Pin.OUT)
6
7 while True:
8     uart.write('t')
9     if uart.any():
10        data = uart.read()
11        if data== b'm':
12            led.toggle()
13        time.sleep(1)
14
```

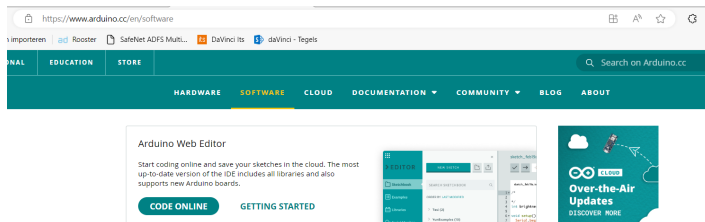
Wat is nieuw in deze code. Op regel 11 zien we een b voor de string staan. De data die we binnen hebben gekregen op regel 10 is nog binair, dus moeten we dit vergelijken met de binaire variant van de letter 'm'.

Op regel 3 en 4 wordt de UART seriële communicatie ingesteld. We gebruiken de pinnen 4 en 5 om respectievelijk te verzenden en te ontvangen. We communiceren met 9600 baud.

Op regel 4 zien we dat we 8 bits versturen, geen parity (check bits) gebruiken en 2 stop bits.


9.4.2 Code op de Arduino Uno

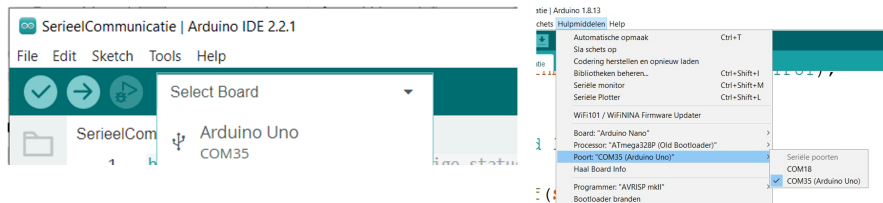
Download en installeer de Arduino IDE software op je computer.



Downloads



Na het installeren kan je de Arduino aansluiten op je computer op de USB-poort, en selecteer de board in het bovenste menu. Vervolgens type je de code over en druk je op  om de code te controleren.

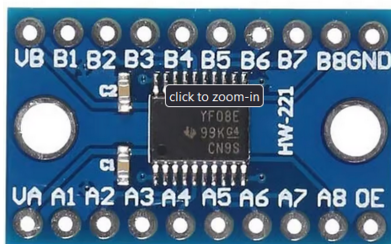


```
1 bool ledState = true; //De huidige status van de led
2
3 void setup()
4 {
5     Serial.begin(9600); // baud rate is 9600
6     pinMode(LED_BUILTIN, OUTPUT);
7 }
8
9 void loop()
10 {
11     if(Serial.read() == 't')
12     {
13         digitalWrite(LED_BUILTIN, ledState);
14         ledState = !ledState;
15         Serial.print('m'); //schrijf 'm' naar UART
16     }
17 }
18
```

Iedere keer als je de code gaat uploaden naar de Arduino dien je de Tx en Rx draden uit de Arduino te halen. Het uploaden van de code gaat via seriële communicatie en als de draadjes aangesloten zijn kan dat upload fouten als gevolg hebben.

9.4.3 Serieel data versturen tussen een Arduino UNO en een Raspberry Pi Pico

We gaan een aansluiting maken tussen de Arduino Uno en een Raspberry pi pico. Ten eerste moeten we in de gaten hebben dat de Arduino werkt met een spanning van 0 tot 5 volt, en de pico van 0 tot 3.3 volt. Om dit te corrigeren zodat ze goed kunnen praten met elkaar gebruiken we een power shifter (TXS0108E). Deze verhoogt razendsnel een waarde van 3.3 volt naar 5 volt en van 5 volt naar 3.3 volt. Als je dit componentje niet gebruikt dan zullen er veel meer fouten in de communicatie optreden met als gevolg dat de communicatie trager zal gaan. Er zullen dan namelijk meer data pakketjes opnieuw verstuurd moeten worden.

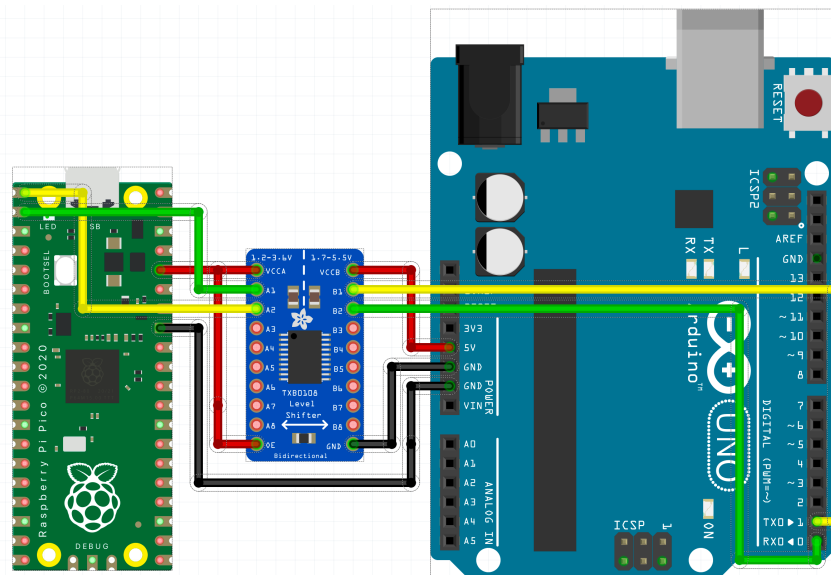


TXS0108E 8 Channel Bidirectional Logic Level Converter Specifications:

- **Two-Way Communication:** Facilitates seamless interaction between devices.
- **VCCA Voltage Support:** Ranges from 1.2V to 3.6V.
- **VCCB Voltage Support:** Ranges from 1.65V to 5.5V.
- **Versatile Level Conversion:** Offers common level conversions such as 3.3-5V and 1.8-3.3V.

9.4.4 Elektronische opstelling

We maken de onderstaande opstelling om de data te versturen. We zien nu dat ieder seconden de led op de Pico aangaat door de 't' dat de pico verzendt en de Arduino op reageert met een 'm'.



9.4.5 Opdracht 1: meer data versturen

We gaan nu de code uitbreiden. We sluiten 5 drukknoppen aan op de Arduino. Als je op de eerste drukknop drukt gaat er 1 led aan. Als je op de tweede knop drukt gaan er twee aan. En zo verder tot en met knop 4. Om alles uit te zetten druk je op knop 5.