Sensors & Interfacing: labo

Tuur Vanhoutte

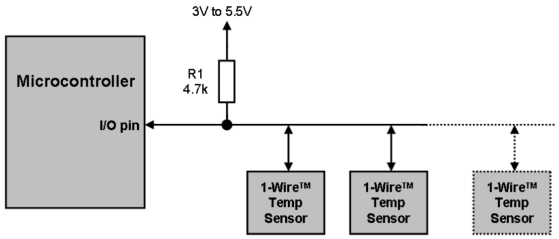
19 juni 2020

Inhoudsopgave

1 One-wire bus

- Ontworpen om over lange afstanden kleine hoeveelheden data te sturen.
- Kan makkelijk in kleine, goedkope sensoren ingebouwd worden
- Voorzien van 3 pootjes (3 draden): 2 voor voeding (+ en -), 1 voor data
- 1 master en 1 of meerdere slaves, elke slave heeft een uniek 64bit adres of slave ID

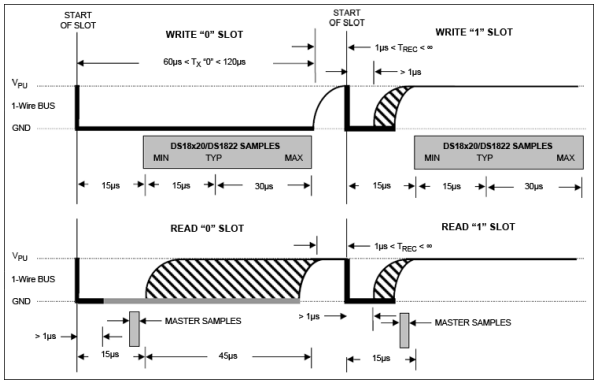
1.1 Schema



Figuur 1: Schema 1-wire bus met meerdere slaves

1.2 Timingdiagram

Omdat er maar 1 draad beschikbaar is voor communicatie, moet het protocol vertrouwen op erg strikte timing. Raspbian voorziet een driver die daarvoor zorgt.

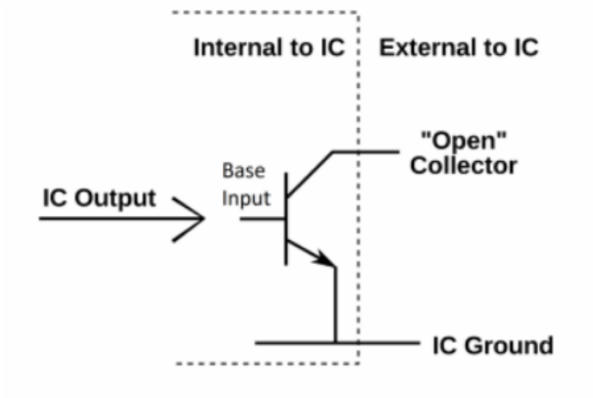


Figuur 2: Timingdiagram

1.3 Werking

1.3.1 Open-collector outputs

Het is niet mogelijk om zomaar een digitale 0 voor te stellen als 0V, en een digitale 1 als 3.3V. Dat kan zorgen voor kortsluiting. Daarom gebruiken we een open-collector uitgang:



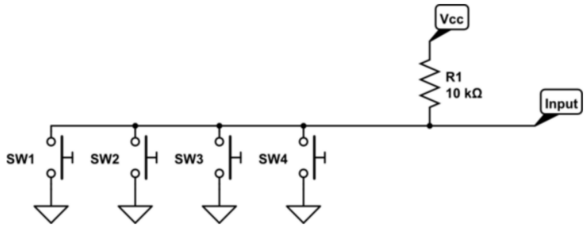
Figuur 3: Interne schakeling van een open-collector uitgang

In plaats van slaves toe te laten zelf een spanning op de bus te leggen, brengen we hem standaard op een hoog signaalniveau d.m.v. een **externe pull-up weerstand**. De uitgangen van ICs die op de bus communiceren worden dan voorzien van een **extra transistor** waarvan de emitter (of source bij een FET) verbonden is met massa. De basis (of gate) is verbonden met de uitgang van de logica en de collector (of drain) met de uitgangspin en dus de bus.

1.3.2 Data verzenden

De bus wordt via een externe weerstand verbonden met de voedingsspanning. Aangesloten apparaten in rust kunnen we beschouwen als een open circuit (de uitgangstransistor geleidt niet). De bus blijft dus standaard op een hoog signaalniveau en er vloeit (in theorie) geen stroom.

Om data te verzenden, kan een deelnemer de transistor in geleiding brengen en zo de bus verbinden met massa. Op dit moment zal er een stroom vloeien, door de pull-up weerstand en uitgangstransistor naar massa. De spanning op de bus bedraagt op dat moment (bijna) 0V, gegeven door de spanningsdeler tussen de externe pull-up weerstand en de restweerstand van de transistor in geleiding.



Figuur 4: Meerdere open-collector outputs (hier voorgesteld door schakelaars), op dezelfde bus

Wanneer nu een andere deelnemer op hetzelfde moment gaat communiceren, kan ook die enkel een verbinding van de bus naar massa maken. Elektrisch vormt dit geen enkel probleem, maar de communicatie van beide deelnemers zal uiteraard nog steeds verstoord zijn.

1.4 Voordelen en nadalen

1.4.1 Voordelen

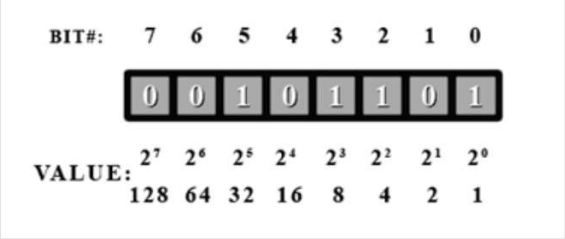
- **Geen elektrische conflicten**: Meerdere deelnemers kunnen enkel meerdere verbindingen naar massa maken.
- **Flexibele spanning**: Omdat enkel de uitgangstransistor elektrisch met de bus verbonden is, is het makkelijk om chips te maken die een verschillende spanning op de bus t.o.v. hun voedingsspanning kunnen tolereren.

1.4.2 Nadelen

- **Hoog stroomverbruik**: Een laag signaalniveau betekent een constante stroom door de geleider, en ook bij hoog signaalniveau zal er nog een kleine lekstroom door de transistors vloeien.

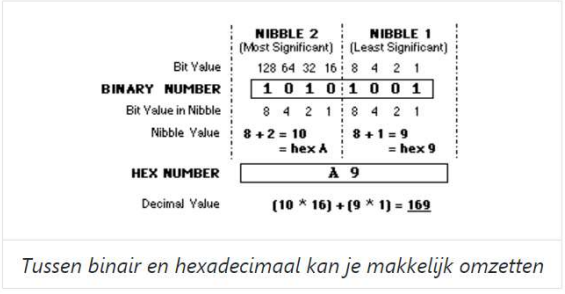
2 Bitoperaties events

2.1 Bits & bytes



Figuur 5: 1 byte. Least Significant Bit (LSB) = rechts, Most Significant Bit (MSB) = links

2.1.1 Hexadecimale notatie



Figuur 6: Omzetting van binair naar hexadecimaal. 1 nibble = 4 bits

2.2 Python

- bin(x) om waarden om te zetten naar binair

- hex(x) om waarden om te zetten naar hexadecimaal
- int(x) om waarden om te zetten naar decimaal

2.3 Bitwise Operators

- And: als beide inputs == 1, output = 1, anders 0
- Or: als 1 van de inputs == 1, output 1, anders 0
- Not: draai input om: 1 wordt 0, 0 wordt 1
- Xor: als precies 1 van de inputs == 1, output = 1, anders 0

Bitwise operator	Symbol in python
and	&
or	|
not	~
xor	^

```
a = 0011 1100

b = 0000 1101

-----------------

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a  = 1100 0011
```

Figuur 7: Voorbeelden bitoperaties

2.3.1 Set bits

De OR-operatie kan je gebruiken om bits aan te zetten, ongeacht hun huidige waarde:

```
>>> have = 0b11001010 # het register nu
>>> mask = 0b00000100 # dus maak ik een getal waar ENKEL bit 2 "aan"
    is
>>> xxxx = 0b11001110 # wat ik wil nu wil uitkomen

# en ik gebruik dat met de |-operator om de bit "aan" te zetten
>>> result = have | mask
>>> xxxx == result
True
```

Hetzelfde principe kan je gebruiken met de &-operator om bits op 0 te zetten, of bits te filteren (stel dat je alleen bit 2 nodig hebt, dan is je masker 0b00000100)

2.3.2 Toggle bits

Togglen van een bit kan je doen met de XOR-operatie (^)

2.3.3 Bitwise NOT

Het resultaat van de NOT-operatie op een binair getal is niet het omgekeerde van het getal, maar het negatieve getal. Om dat op te lossen moet je het negatieve getal AND-en met 0xFF.

2.3.4 Shift left/right

We kunnen bits opschuiven met $<<$ en $>>$. Dit is hetzelfde als vermenigvuldigen met 2 en delen met 2.

```
>>> 0b00010100 >> 4 == 0b00000001
True
>>> 0b11001010 << 4 == 0b110010100000
True
>>> 0b11001010 << 8 == 0b1100101000000000
True
```

2.3.5 In-place operators

```
>>> x += 2        # x = x + 2
>>> x -= 2        # x = x - 2
>>> x |= 4        # x = x | 0x04 --> bit 2 is nu 1
>>> x &= 253      # x = x & 0xfd --> bit 1 is nu 0
>>> x ^= 0xf0     # x = x ^ 0xf0 --> 4 MSB worden getoggled
>>> x <<= 2       # x = x << 2 --> alle bits 2 plaatsen naar links
>>> x >>= 3       # x = x >> 3 --> alle bits 3 plaatsen naar rechts
```

2.3.6 Volgorde van bewerkingen

1. not $\sim$
2. shift $<<$, $>>$
3. and &
4. xor ^
5. or |

Gebruik haakjes!

2.4 Binary Coded Decimals (BCD)

Bij BCD worden 4 bits gebruikt om 1 decimaal cijfer weer te geven

Voor het getal 91 worden dus groepjes van 4 bits gebruikt om elk getal weer te geven:

```
Decimal:      9     1
Binary:    1001  0001
```

Opgelet: je mag niet zomaar wiskundige operaties toepassen op een BCD code. De decimale waarde van BCD-getal 91 is 128+32+1 = 161

2.5 Interrupts & Events

2.5.1 Interrupts

Een interrupt is een voorziening om de processor op de hoogte te stellen van een bepaalde gebeurtenis. De CPU zal daarop zijn huidige werk onderbreken en een Interrupt Service Routine (ISR), ook wel Interrupt Handler genoemd, uitvoeren.

Na afloop gaat hij weer verder met de taak waar hij voordien aan werkte. Op deze manier hoeft er geen CPU-power verspild te worden met het wachten op gebeurtenissen.

2.5.2 Events

= signaalovergang

```
GPIO.add_event_detect(channel, GPIO.RISING) # add rising edge
    detection

do_something_else() # thread is bezig met andere dingen

if GPIO.event_detected(channel):
    print('Button released')
    # dit werkt nog altijd
```

2.5.3 Callbacks

```
def my_callback(channel):
    print('This is a edge event callback function!')
    print('Edge detected on channel %s'%channel)
    print('This is run in a different thread to your main program')

GPIO.add_event_detect(channel, GPIO.RISING, callback=my_callback)
```

Je kan meerdere callbacks op eenzelfde pin registeren door gewoon meerdere keren de methode GPIO.add_event_callback(...) op te roepen.

Bouncetime

Bijkomende parameter bij add_event_detect(): aantal milliseconden dat dient om de knop te ontdenderen (debounce).

3 Seriele communcatie

Bv: communicatie tussen Arduino en Raspberry Pi.

Opgelet: de Pi werkt op 3.3V en de meeste arduino's op 5V. We moeten een levelshifter gebruiken.

3.1 Serieel protocol

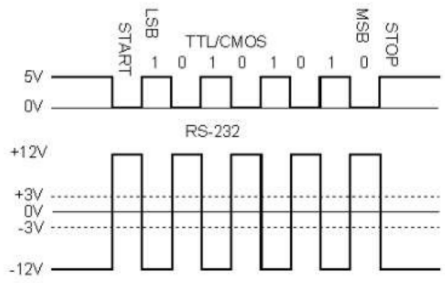
Zeer eenvoudig om full-duplex te werken:

- een draad om data te versturen
- een draad om data te ontvangen
- gemeenschappelijke massa (GND)
- Het is een point-to-point verbinding: 2 deelnemers, dus geen adressering nodig

De chip die zo'n seriele interface aanstuurt noemt men een UART (Universal Asynchronous Receiver / Transmitter). De termen UART, RS-232 en seriele poort worden vaak door elkaar gebruikt. RS-232 of Recommended Standard 232 omschrijft connectoren, elektrische eigenschappen ...

3.2 Hardware (Layer 1)

- Oorspronkelijk:
  - 0 = 3 tot 15V
  - 1 = -3 tot -15V
- Tegenwoordig:
  - Veel devices werken op -5V tot +5V
  - Hetzelfde protocol maar op TTL-niveau (0-5V)
- De bus is standaard hoog, wat door de UART zelf geregeld wordt. We moeten dus geen pull-up voorzien.



Figuur 8: TTL (boven) vs RS-232 (onder)

3.2.1 Crossed cable of null-modemkabel

De TX-pin van de zender moet verbonden worden met de RX-pin van de ontvanger en vice versa. In de context van de oude seriele poort noemt men zo'n crossed cable een null-modemkabel.

Seriële verbindingen werden veel gebruikt voor het verbinden van een modem met je PC, en met zo'n null-modemkabel kon je dan 2 computers rechtstreeks verbinden zonder de modems ertussen.

3.3 Protocol (Layer 2)

Alle data bits bevinden zich tussen een start- en stopsignaal. De start bit is één 0-bit, de volgende zeven of acht bits vormen een databyte. Eventueel kan er als controle nog een parity bit worden toegevoegd. Het teken wordt telkens afgesloten door één of twee stop bits.

Aan de kant van de ontvanger wordt de data aan het start- en stopsignaal herkend. Het aantal start-, data- en stopbits kan worden ingesteld en moet uiteraard langs beide kanten overeen komen.

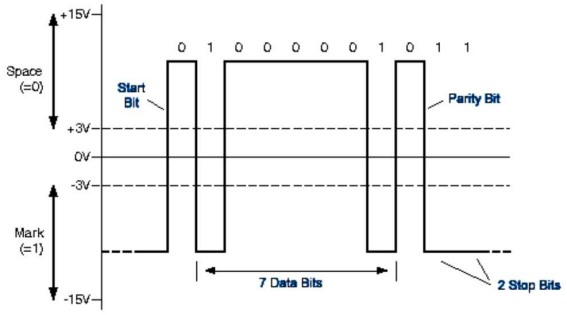
3.3.1 Baudrate

= De snelheid = symbolen per seconde

- Zender en ontvanger moeten dezelfde baudrate hebben
- Hogere rate $\rightarrow$ hogere eisen qua elektrische verbinding en timingnauwkeurigheid
- Lagere rate $\rightarrow$ minder snelle communicatie, maar grotere maximale kabellengte
- Positieve spanning: lage bit, negatieve spanning: hoge bit

Veelgebruikte baudrates:

- 9600 baud (=veelvoud van 300, vandaag standaard maar traag, wordt door vrijwel elke UART ondersteund)
- 19200, 57600, 115200 baud (= veelvouden van 9600)
- 64k baud en veelvouden
- 1M baud en veelvouden



Figuur 9: Timingdiagram seriele communicatie

3.4 Flow control

Mechanisme waarmee zender en ontvanger elkaar kunnen verwittigen wanneer ze klaar zijn voor datatransmissie. Dat kan zowel in hardware (dus met extra signaaldraden), of in software met speciale karakters.

Hardware:

- RTS/CTS: Request To Send/Clear To Send pins worden gebruikt om aan te geven dat het apparaat klaar is om data te sturen/ontvangen
- DSR/DTR: Data Set Ready/Data Terminal Ready-lijnen die eigenlijk aangeven dat het apparaat online en klaar is, worden soms misbruikt voor flow control

Software:

- XON/XOFF: Zijn speciale ASCII-controletekens die als data van ontvanger naar verzender worden gestuurd om transmissie te starten/stoppen

Flow control was van groot belang om een modem aan te sluiten maar wordt vandaag niet meer veel gebruikt.

3.5 Verschillende opties seriele communicatie

1. USB kabel
2. Gebruik van seriele pins op de GPIO connector

3.6 Byte serieel versturen

Het serieel protocol heeft maar 1 draad om een byte (char) te versturen of ontvangen. Daarom moeten we elke byte ontleden en bit per bit versturen. De meeste protocollen starten met de MSB (meest linkse bit).

3.6.1 In python: PySerial

= een library om makkelijk een seriele poort te openen:

```
import serial
ser = serial.Serial('/dev/ttyS0') # open serial port
print(ser.name)                   # check which port was really used
ser.write(b'hello')               # write a string
ser.close()                       # close port
```

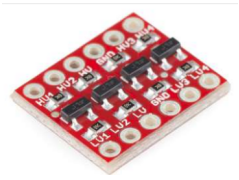
```
from serial import Serial, PARITY_NONE

# simple echo server
with Serial('/dev/ttyS0', 115200, bytesize=8, parity=PARITY_NONE,
    stopbits=1) as port:
     while True:
         line = port.readline()
         port.write(line)
```

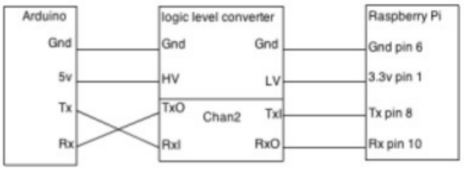
3.7 Level shifter

= een module die we kunnen gebruiken om componenten die op een verschillende spanning werken digitaal met elkaar te verbinden.

- De ene kant is low voltage (LV), de andere high voltage (HV)



Figuur 10: Een levelshifter met 4 bits



Figuur 11: Verbinding tussen Arduino en Pi. Let op de crossed cabling tussen Arduino en levelshifter

4 Shiftregister

Een shiftregister ontvangt een aantal bits in serie die parallel naar buiten gebracht worden. Op die manier kunnen we via één data pin van onze raspberry pi meerdere uitgangen creëren.

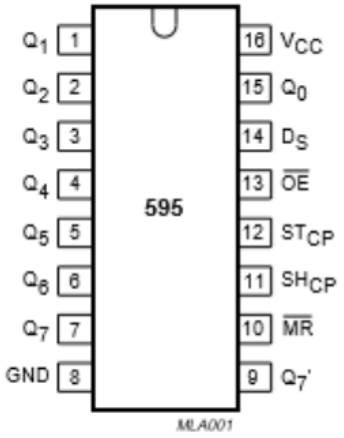
Een shiftregister is opgebouwd uit een aantal elektronische componenten (flip flops of latches) die in serie geschakeld worden met elkaar en dezelfde klok delen. De uitgang van elke flip flop wordt gebruikt als ingang van de volgende. Zo'n schakelingen wordt ook een cascade (waterval) schakeling genoemd. Op die manier kan je ook meerdere shift registers met elkaar verbinden.

4.1 Shiftregister 74HC595

De 74HC595 is een high speed 8-bit 'serieel in parallel out' shiftregister. We kunnen dus 8 bits na elkaar serieel versturen naar het register, die bits worden vervolgens naar 8 parallelle uitgangen gebracht.

Dit shiftregister heeft ook een storage register dat de data bij kan houden. Voor het 7-segment display is dit handig omdat we de vorige waarde kunnen blijven tonen.

4.2 Pinnummering



Figuur 12: Pin configuratie 74HC595

Deze pins moeten we verbinden met een GPIO-pin van de RPi:

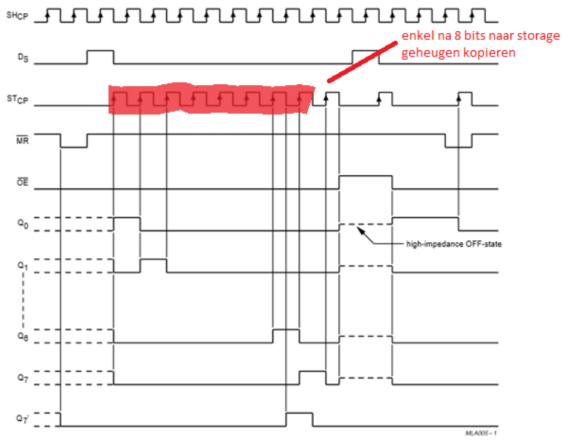
- Master Reset (MR)
- De twee klokken (SHCP en STCP)

- De Output Enable (OE)
- De Serial Data (SD of D_S)

Q0 t.e.m. Q7 sluiten we aan ons component (hier: 7-segment display)

Door gebruik te maken van 5 GPIO pinnen beschikken we nu over 8 digitale uitgangen. Door een shiftregister toe te voegen beschikken we over 16 uitgangen met onze 5 GPIO-pinnen.

4.3 Protocol



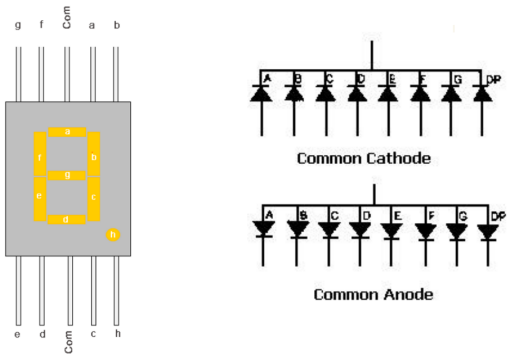
Figuur 13: Timingdiagram shiftregister 74HC595

4.4 7-segment display met shiftregister

= display dat bestaat uit 8 segmenten (7 voor het cijfer + 1 voor het punt)

4.4.1 Twee soorten

1. Common Cathode
2. Common Anode



Figuur 14: 7-segment display: twee soorten

4.5 Shift input

Sommige shiftregisters kunnen ook als input gebruikt worden → meerdere digitale inputs met slechts 5 pinnen.