Arduino Elektronica, uitgangsniveau -0 Deel 3

Handleiding van Helpmij.nl

Auteur: drejansen

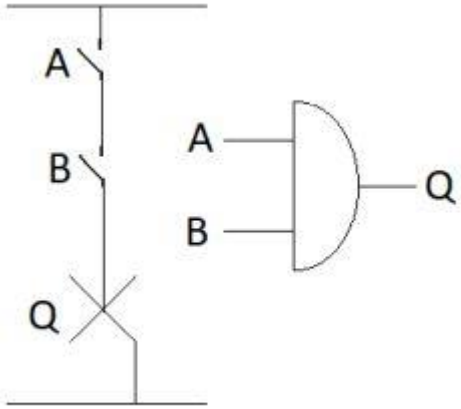
oktober 2021

Digitaal

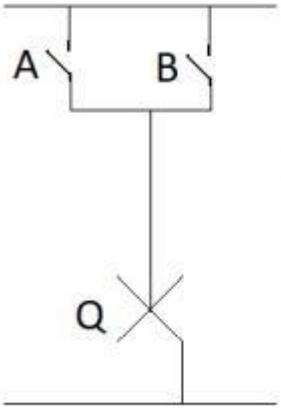
In de digitale techniek kennen we twee waarden -1- en -0-
Vermoedelijk wist u dat al.
Het gaat hier om schakelaars, een schakelaar kan 'open' (= uit) of 'gesloten' (= aan) zijn. Vaak wordt aan de 'gesloten' stand de waarde -1- toegekend, er staat dan een positieve spanning op de ingang.
maar let op, kan ook andersom! Dan heeft de open stand de waarde -1-. Daarover verderop in dit verhaal meer.

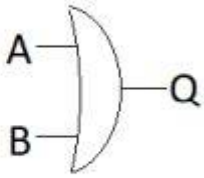
Er zijn diverse soorten poorten, maar laten we die eens met gewone schakelaars vergelijken.

Twee schakelaars in serie en een lamp.

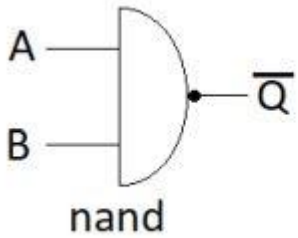


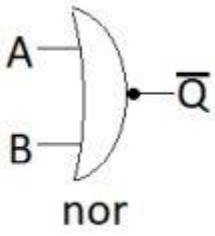
Twee schakelaars parallel en een lamp.

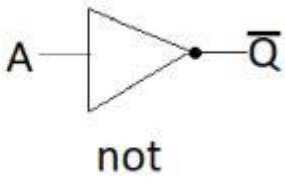




In de linker tekening staan twee schakelaars in serie, beiden schakelaars A en B moeten aan staan, om de lamp Q te laten branden. Deze eigenschap noemt men een AND- of EN poort.
Rechts staan ook twee schakelaar, daar hoeft slechts één schakelaar aan te zijn om de lamp te laten branden. Dus OF schakelaar A, OF schakelaar B moet aan, beiden mag natuurlijk ook, om de lamp aan te zetten. U raad het al, dit is een OR- ook wel OF poort. Nu kan je het ook andersom laten werken, in de linker situatie is de lamp aan ZOLANG beide schakelaars uit zijn…. Dan spreekt men van een NOT AND of korter: NAND poort.
Bij de OR poort kan je ook zo iets bedenken, dan heet dat een NOR poort.







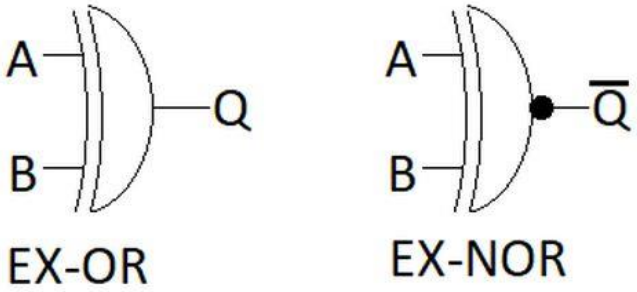
De geïnverteerde uitgang is herkenbaar aan het bolletje op de uitgang. De inverter komt ook als onafhankelijk component voor, dat heet dan een NOT poort. Bij de not-poort komt ingang A er als NIET-A uit. Hierboven staan al 3 verschillende poorten, er zijn er nog veel meer, maar uiteindelijk bestaat er maar één poort: de NAND door nand's te combineren kan je elke andere poort vormen.

Waarheidstabel

Dat klinkt wat moeilijker dan het is, bij een waarheidstabel wordt er op de ingangen spanning gezet en op de uitgang gemeten. De situaties of mogelijkheden, legt men in een tabel vast.

AND			NAND			OR			NOR			EX-OR			EX-NOR		
in	in	uit	in	in	uit	in	in	uit	in	in	uit	in	in	uit	in	in	uit
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1
1	0	0	1	0	1	1	0	1	1	0	1	1	0	1	1	0	0
0	1	0	0	1	1	0	1	1	0	1	1	0	1	1	0	1	0
1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	1	1	1

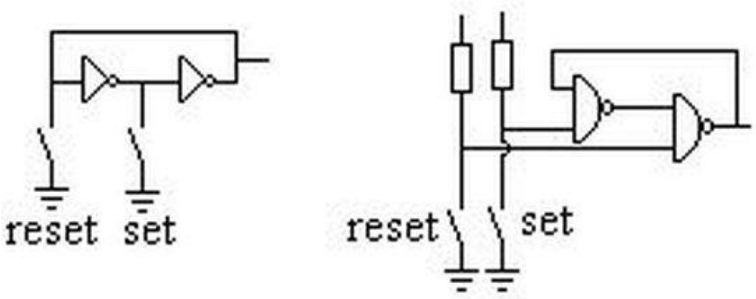
Wat is nou een EX-OR? Welnu, uit de tabel kan je aflezen wat dit ding doet. Dit is een OR, maar met de bijzonderheid, dat wanneer beide pootjes hoog of laag zijn, de uitgang altijd LAAG is.
Dit in tegenstelling tot een 'normale' OR of NOR. (EX staat voor 'exclusief'.) Jawel, ook daarvoor is een symbool, een (N)OR met een extra boogje ervoor.



Bekijk de waarheidstabellen nog eens goed! Vergelijk de OR en de AND eens met elkaar. Wanneer je de nullen van de AND vergelijkt met de enen van de OR, dan gedragen de nullen zich als een OR. Dus als je NOT's vóór de ingangen van de AND schakelt, dan heb je een NOR, sluit je op de Q óók een NOT aan, dan heb je een OR gemaakt van een AND. Laat dat even rustig inwerken, want dit is toch wel een belangrijk gegeven.
Dat nullen verhaal wordt 'negatieve logica' genoemd. Het wordt niet vaak gebruikt, maar let op, het komt voor! Een streepje b̄ōv̄ēn̄ een letter betekent **'niet'**.

Schakelen

Toch wel een belangrijke schakeling, die ondanks alles, vaak wordt toegepast: de RS flipflop.
Bij de Arduino worden sensoren en actuatoren gebruikt. Een veel gebruikte 'sensor' is een gewoon schakelaartje. Bij het schakelen lijkt het wel dat je éénmalig contact maakt, maar dat is niet zo. De schakelcontacten stuiteren meerdere keren op elkaar. Wij zien dat niet, maar de Arduino ziet dat alsof je meerdere malen de schakelaar: aan/uit/aan/uit/aan/uit/aan, hebt gezet. Bij het uitschakelen gebeurd dit ook, maar dan stuitert het minder vaak. Binnen Arduino is een kreet die dit opvangt: "debounce" maar dat vertraagd het hele programma (schets) een beetje. Daarom gebruiken we liever de onderstaande schakeling:



Dat kan met twee NOTs, maar mooier en netter is het om hier twee NANDs voor te gebruiken. Een 7400 of 74ls00 is een goedkoop chipje waarin vier two-input-nands zitten. Waarmee je twee RS flip-

flops kan maken. Bij de NOTs sluit je de uitgang even kort, dat vind hij niet leuk, maar hij gaat er niet meteen kapot van. Een chip met 6 not's is de 7404 of 7414 (mooier).
Hier zijn twee losse schakelaartjes gebruikt, maar één (terugverend) wisselschakelaartje is beter. Dit is dan een hardware anti-dendering. Beredeneer de werking van deze schakeling aan de hand van de waarheidstabel.

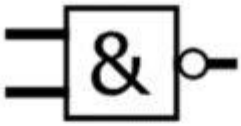
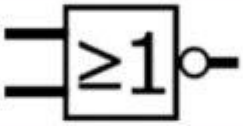
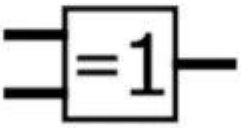
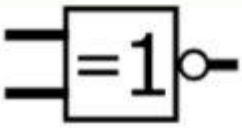
Rekenen

Computer betekend rekenaar naar het Engelse werkwoord: to compute. Dus gaan we rekenen. Maar niet te moeilijk hoor, want we rekenen met 0 en 1.

0 x 0 = 0 1 x 0 = 0 zo ook 0 x 1 = 0 en 1 x 1 = 1 dit is de waarheidstabel van een AND.

0 + 0 = 0 1 + 0 = 1 zo ook 0 + 1 = 1 maar ook: 1 + 1 = 1 ja, ik zij het al, -2- bestaat niet.

Een lampje is aan of uit, 'aaner dan aan' kan een lampje niet zijn. Vandaar: 1 + 1 = 1.
Hoe een computer toch aan grotere getallen komt, is een verhaal apart. Zoals ik in deel -1- al vertelde, ik wil het niet moeilijker maken dan het al is. Laat dat moeilijke maar aan de Arduino zelf over, dan komt het wel goed. Wat is dan de reken formule symbool van een ex-or? Hier introduceren we een nieuw rekensymbool: een omcirkeld **+** teken. $\oplus$. Deze tekstverwerker kent helaas geen overschrijf commando, dus dat is lastig schrijven, maar bekijk de waarheidstabel. De EX-OR is bijna gelijk aan de OR, met als enige uitzondering: 1 $\oplus$ 1 = 0. Bij de ex-nor is dat natuurlijk geïnverteerd: 1. Met die poorten zijn leuke schakelingen te bedenken, maar dat doen wij niet, wij gebruiken daarvoor natuurlijk een Arduino! (Behalve bij het ontdenderen van een schakelaartje.)
Poorten kunnen ook anders worden getekend: dit plaatje vond ik op internet.

	IEC-symbool	ANSI-symbool	Uitgang 0	Uitgang 1	Vergelijking
NAND			Als alle ingangen **1** zijn	Zodra één of meerdere ingangen **0** zijn	$\overline{A.B}$
NOR			Zodra één of meer ingangen **1** zijn	Als alle ingangen **0** zijn	$\overline{A+B}$
XOR			Als één van de ingangen **1** is	Als één ingang **1** is	...
XNOR			Als één ingang **1** is	Als één van de ingangen 1 is	...

Tot slot nog iets over tellen, want we hebben het tot op heden gehad over één bit.
Maar meestal wordt er over **byte** gesproken. Een byte is een groepje van 8 bits.
Soms kom je het woord **nibble** tegen, dat is een groepje van 4 bits. Het benoemen van de afzonderlijke bitjes van een byte of nibble is lastig, daarom benoemen we die in de hexadecimale vorm.

0 1 2 3 4 5 6 7 8 9 A B C D E F
0000 - 0001 - 0010 - 0011 - 0100 - 0101 - 0110 - 0111 - 1000 - 1001 - 1010 - 1011 - 1100 - 1101 -

1110 - 1111
Waarom A-B-C-D-E-F ? welnu, A=10 hebben we nu een 1 en een 0, of 10 ?
Deze telling heet hexa-decimaal, dat wordt heel veel gebruikt! Leer dat maar uit je hoofd. (Dit zijn dus nibbles.)
Een andere reeks waarvan het toch wel handig is om die uit je hoofd te leren is de volgende: 0-1-2-4-8-16-32-64-128-256-512-1024-etc. Dat kan verder, maar deze reeks is vaak voldoende.

Volgende maand komt de Arduino zelf aan de beurt.

Groeten, Dré