

# Physical computing vanuit Scratch met gebruik van scratchClient – **Beginners**

*Bestuur servo's, LED's en meer vanuit Scratch  
met gebruik van Raspberry Pi, Arduino en  
scratchClient*

Hans de Jong - Coderdojo Eindhoven - 23 september 2017

# Deel 1: Introductie

# Als de presentatie een puinhoop is ...

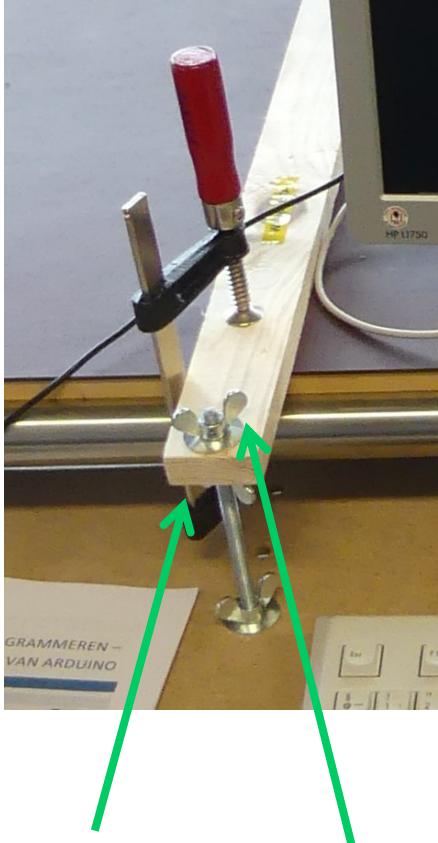
- Je moet het Calibri font op je Raspberry Pi hebben (is standard op Windows maar niet op Raspberry Pi).
- Of zorg dat font te krijgen en installer het, of ...
- ... kijk naar de PDF versie van de presentatie
  - Maar daaruit kun je niet kopieren en plakken.
  - Daarom is er een bestand *ForCopyPaste.txt* in de map die alles bevat wat je in deze les moet kopieren en plakken in deze les.

# Als je na de les naar de presentatie kijkt ...

- Kijk dan ook naar eventuele sprekers notities die op sommige pagina's aanwezig zijn.

# Ergonomics

- At the Weekendschool we teach children 5 golden rules how to sit when using computers:
  - Rest your arms on the table in a natural fashion.
  - If you do not need the keyboard, move it up and give room to the mouse.
  - Have the monitor at arms length.
  - Change position (lean forward, backward etc.).
  - Frequently stop, walk around and exercise.



To change the height, use the two upper wingnuts.



# Hoe is het vandaag georganiseerd?

- Welkom en introductiepresentatie (10 min).
- Daarna werkt iedereen in zijn of haar eigen tempo.
- Af en toe nemen presenteren we 5 minuten om een volgend concept uit te leggen.
- Taal: Nederlands, maar sommige delen van het materiaal is in het Engels.
- Aan het eind mag je als je dat wilt het materiaal dat je hebt gemaakt kopiëren naar je eigen USB stick
- Opruimen, of voorbereiding voor de les voor gevorderden.

- De grote stappen

1. Krijg de hardware werkend en maak een werkende scratchClient configuratiebestand met scratchClient config.
2. Zet de elektrische componenten op het bord en test wat je gemaakt hebt.
3. Schrijf code in Scratch en test uit of het werkt.
4. Voeg meer componenten toe en werk het configuratiebestand bij.  
Herhaal dit.
5. Als er tijd over is, schrijf een programma in Scratch.

# Het doel van vandaag

- Aan het eind van vandaag zou je in staat moeten zijn om:
  - Het opzetten van de apparatuur thuis te doen (als je tenminste de hardware hebt ☺ )
  - De volgende dingen begrijpen
    - Digitale output (b.v. een LED laten branden)
    - Digitale input (b.v. een knop uitlezen)
    - Analoge input (b.v. van een potentiometer)
    - Puls Breedte Modulatie (Pulse Width Modulation - PWM)
      - Om LED's te dimmen
      - Om servomotoren te besturen
      - Om een zoemer te laten klinken
  - Begrijpen waar alle weerstanden voor dienen
  - scratchClient te configureren en te laten lopen
  - Scratch programma's te maken om de fysieke input en output te besturen
  - De input en output waarden te kunnen laten zien
- **Plezier te hebben!**

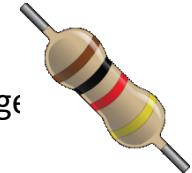
# Geen doel is ...

- Het is niet een doel om een compleet zinvol spel of ander programma te maken.
  - Dat kun je thuis doen met je eigen creativiteit nu dat je weet hoe je verschillende stukken hardware uit Scratch moet aansturen via scratchClient.

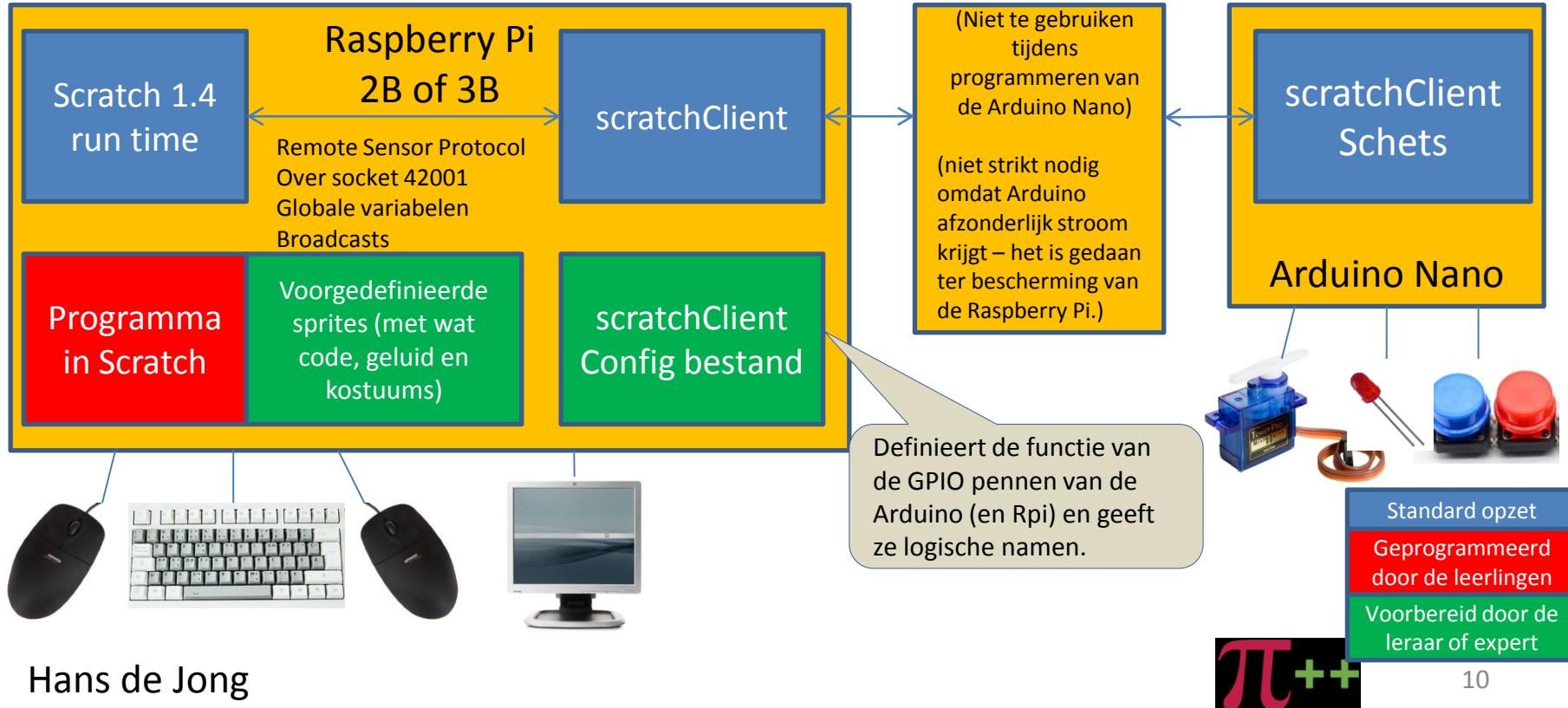
# Slechts een paar regels vandaag ..



- Als Raspberry Pi opnieuw gestart moet worden, dan moet de voeding uit en weer aan gedaan worden. Ga niet aan kabels trekken, maar gebruik de stekkerschakelaar waar de voeding in zit (nadat je de Raspberry Pi gestopt hebt).
- Zet altijd weerstanden in serie met de componenten als dat aangegeven is.
  - Als je denkt dat dit niet nodig is, laat het ons dan weten en dan zullen we de reden uitleggen.
- Als je de bedrading verandert:
  1. Koppel de USB kabel af van de Arduino Nano 
  2. Schakel de 9V af 
  3. Check, dubbel check and check nogmaals of de bedrading correct is.  
Je kunt componenten opblazen als de bedrading fout gedaan is.
  4. Zorg dat jullie beiden (4 ogen principe) overtuigd zijn dat de bedrading in orde is voordat je de spanning weer aansluit.
  5. Na het wijzigen van een config bestand: start scratchClient opnieuw.
- Als iets kapot gaat of beschadigd raakt: we hebben reserve materiaal
  - Stop **svp niets** dat kapot is terug in de doos.



# De opzet



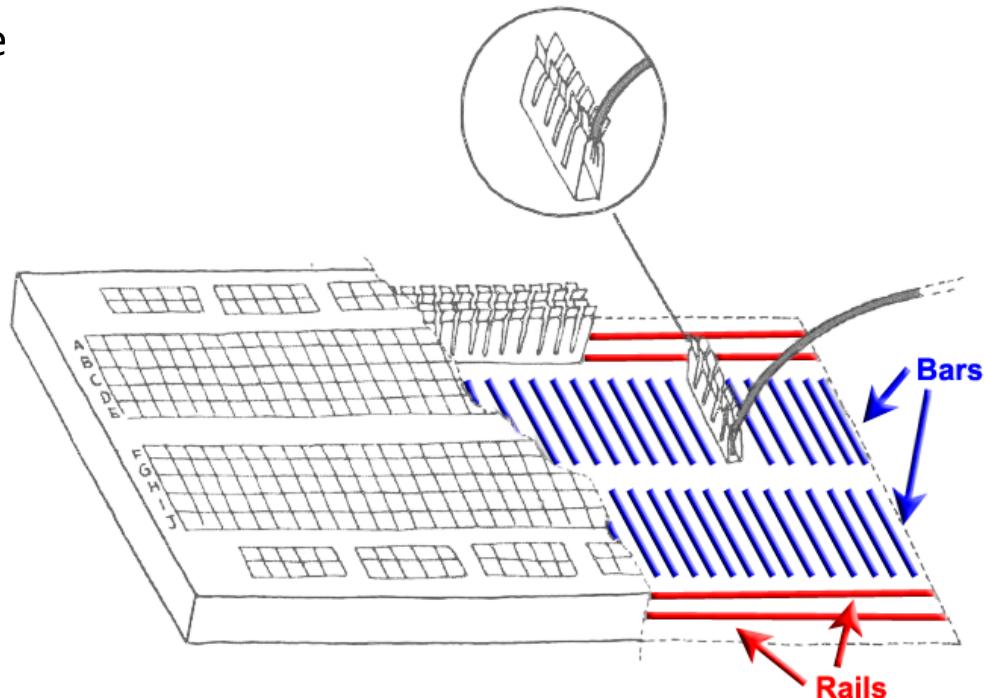
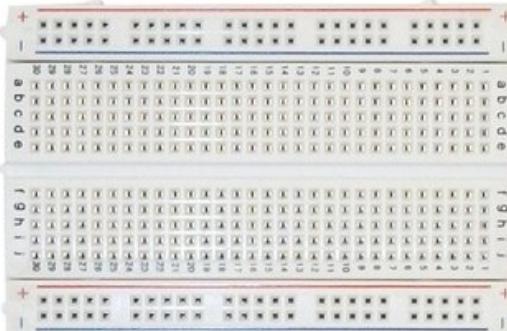
# Waar is de presentatie?

- De presentatie = de instructie op je bureaublad in de map PiAndMore
- Als je alles van de presentatie al weet en je gaat vervelen, dan mag je zelf verder gaan
  - In stiltesvp ... ☺

# Deel 2: Leer de componenten kennen

# Breadboard (broodplank)

- Wordt gebruikt om snel elektronische schakelingen te maken.
- Let op de 2 rails voor + (VCC) en – (GND = aarde)
- Let op de 2 rijen met elk 5 doorverbonden gaten.



# Bekijk het Arduino Nano uitbreidingsbord

Per GPIO signal 3 pennen:

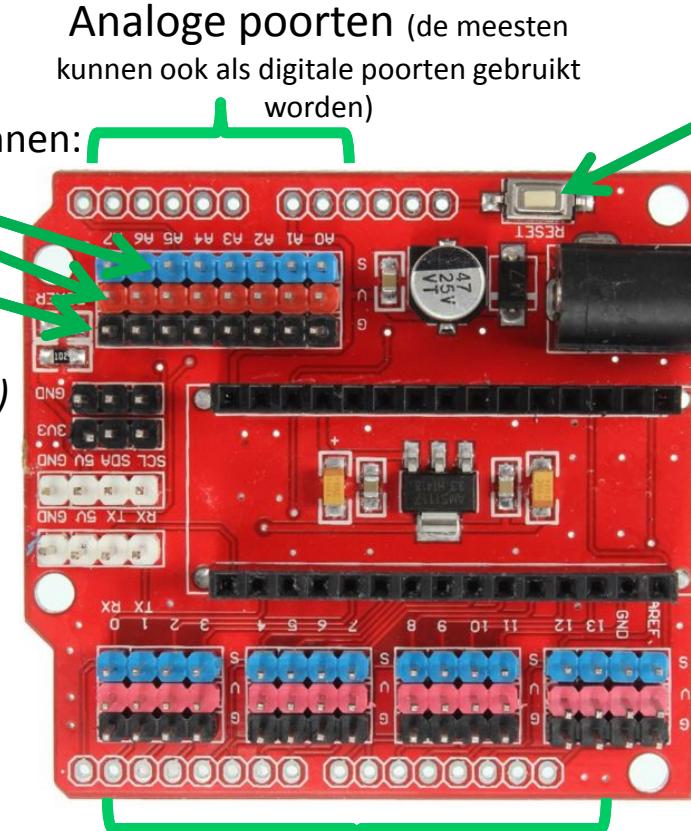
S (blauw = signaal)

V (rood = VCC = +)

G (zwart = GND = -)

(heel handig om b.v.  
servo's aan te sluiten)

*GPIO = General  
Purpose Input /  
Output = Algemeen  
bruikbare Input /  
Output*



Analoge poorten (de meesten kunnen ook als digitale poorten gebruikt worden)

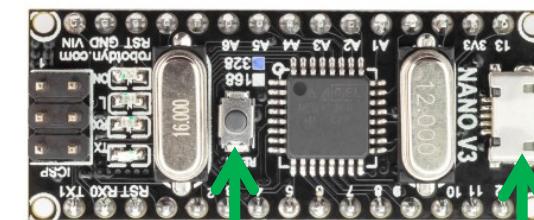
Digitale poorten

Hans de Jong

Reset knop

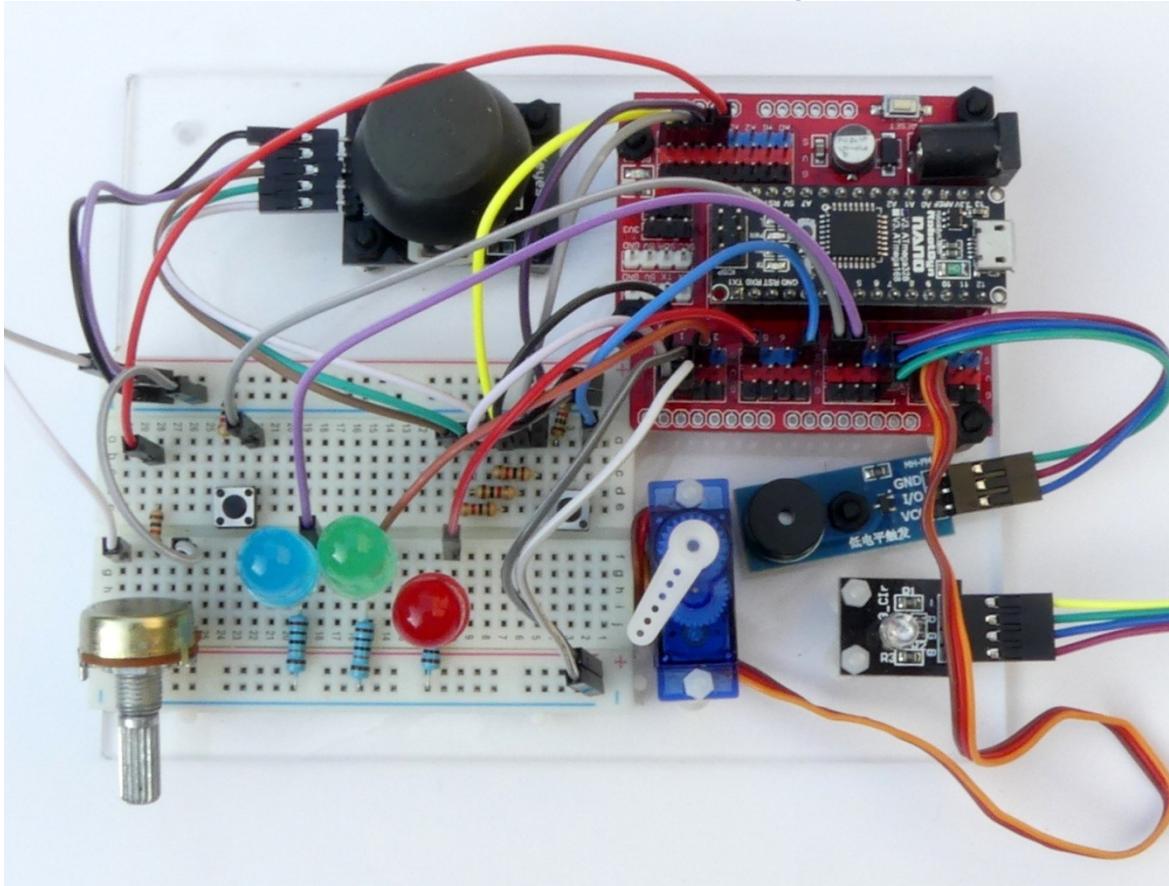
9V aansluiting

Arduino Nano met een 328P processor



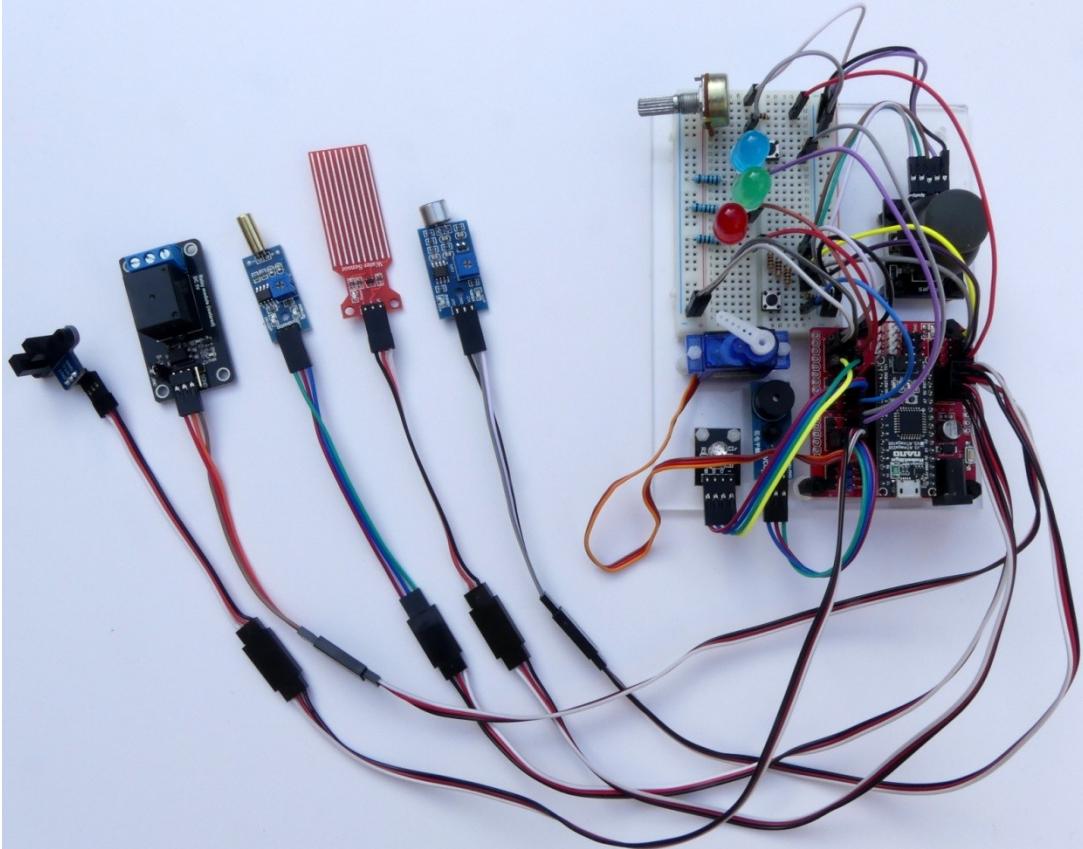
Micro USB poort  
Reset knop

# Hoe het uiteindelijke bord er uit ziet ...



... aan het eind  
van de  
beginners les.

... en als je verder gaat met de les voor gevorderden



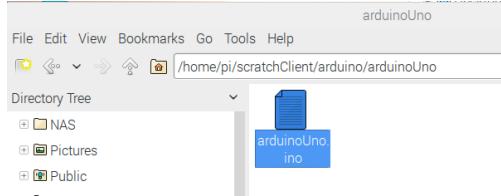
# Deel 3: Het laden van de schets in de Arduino

# Voorbereiden van het programmeren van de Arduino Nano

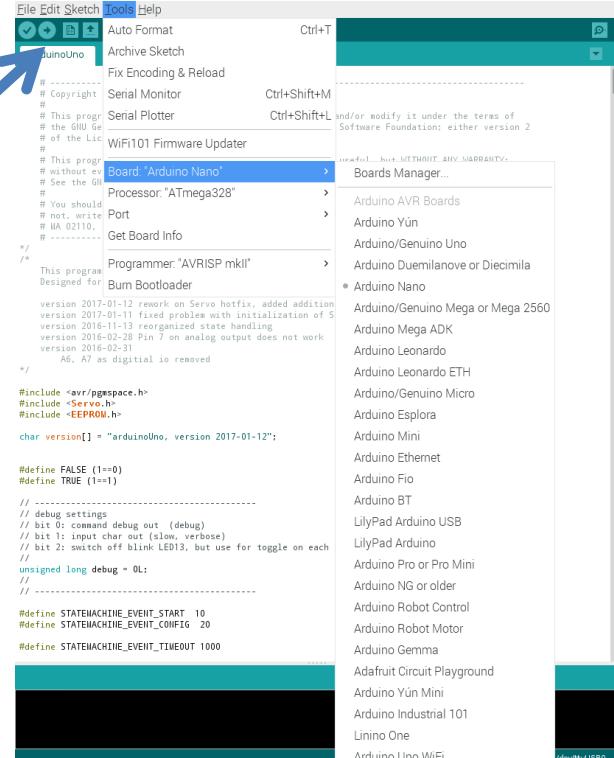
- De Arduino moet een programma draaien zo dat het kan communiceren met de Raspberry Pi en de bericht van scratchClient begrijpt.
  - We moeten beginnen dat erin te laden.
- Sluit de Arduino Nano aan
  - Zorg dat het een directe aansluiting is
    - Niet via de USB hub
    - Om merkwaardige redenen werkt het niet via (deze) USB hub, ondanks dat het wel werkte in een vorige release van Raspian (en dat zou het nog steeds moeten).

# Uploaden van scratchClient naar de Arduino

- Zorg ervoor dat de Arduino rechtstreeks aan de Raspberry Pi is aangesloten (*niet* via de USB hub)
- Open de scratchClient schets voor Arduino Uno in `/home/pi/scratchClient/arduino/arduinoUno`



- Dubbel klik om de Arduino IDE te openen.
- Klik *Tools* en zorg dat dit is gezet:
  - Bord: Aruino Nano
  - Processor Atmega328
  - Poort: de poort waar de Arduino Nano aangesloten is (in het algemeen /dev/ttyUSB0)
- Klik op de Upload knop.
- Wacht tot gerapporteerd is dat de Upload gedaan is.



# Deel 4: Het definiëren van de configuratie

Geef namen aan de pennen en definieer hoe elke pen gebruikt zal worden

# Het starten van de config tool

- Dit creëert config bestanden waarbij logische namen naar Arduino pennen worden vertaald.
- Navigeer naar `/home/pi/scratchClient/tools`
- Dubbelklik op `scratchClientConfig.sh` and kies *Uitvoeren*.
  - Als het bestand opent in plaats van een keuze te vertonen dan moet je eerste de rechten goed zetten.
  - Om de rechten te zetten: klik rechts op het icoon, kies *eigenschappen*, dan *rechten* en zet dan de *uitvoeren* rechten op tenminste *eigenaar*.

# Het maken van de eerste config file

The screenshot shows the ScratchClient configuration interface. On the left, there's a table for pin configuration:

name	arduino	direction	function	scratchName
D0	void			
D1	void			
D2	void			
D3	void			
<b>D4</b>	<b>out</b>	<b>output</b>	<b>Big Red LED</b>	
D5	void			
D6	void			
<b>D7</b>	<b>in</b>	<b>input_pullup</b>	<b>Button</b>	
D8	void			
D9	void			
D10	void			
D11	void			
D12	void			
D13	void			
A0	void			
A1	void			
A2	void			
A3	void			
A4	void			
A5	void			
A6	void			
<b>A7</b>	<b>void</b>			

Below the table are three configuration parameters:

- Parameter serial.device: /dev/ttyUSB0
- Parameter ident.check:
- Parameter ident.pattern: (empty)

On the right, the XML configuration file generated by the tool is displayed:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<config version="1.0">
<description>Generated configuration from ScratchCl</description>
<adapter class="adapter.arduino.UNO_Adapter" name="">
  <!-- id = 'D4' direction = 'out' function = 'output' value name="inputD4">
  <variable name="Big Red LED"/>
  </input_value>

  <!-- id = 'D7' direction = 'in' function = 'input' output value name="outputD7">
  <sensor name="Button"/>
  </output_value>

  <extension>
    <io dir="out" id="D4"/>
    <io dir="in" id="D7" pullup="on"/>
  </extension>

  <parameter name="serial.device" value="/dev/ttyUSB0"/>
  <parameter name="serial.baud" value="115200"/>

  <!-- optional parameters for IDENT check -->
  <parameter name="ident.check" value="yes"/>
  <parameter name="ident.pattern" value="" />
</adapter>
</config>
```

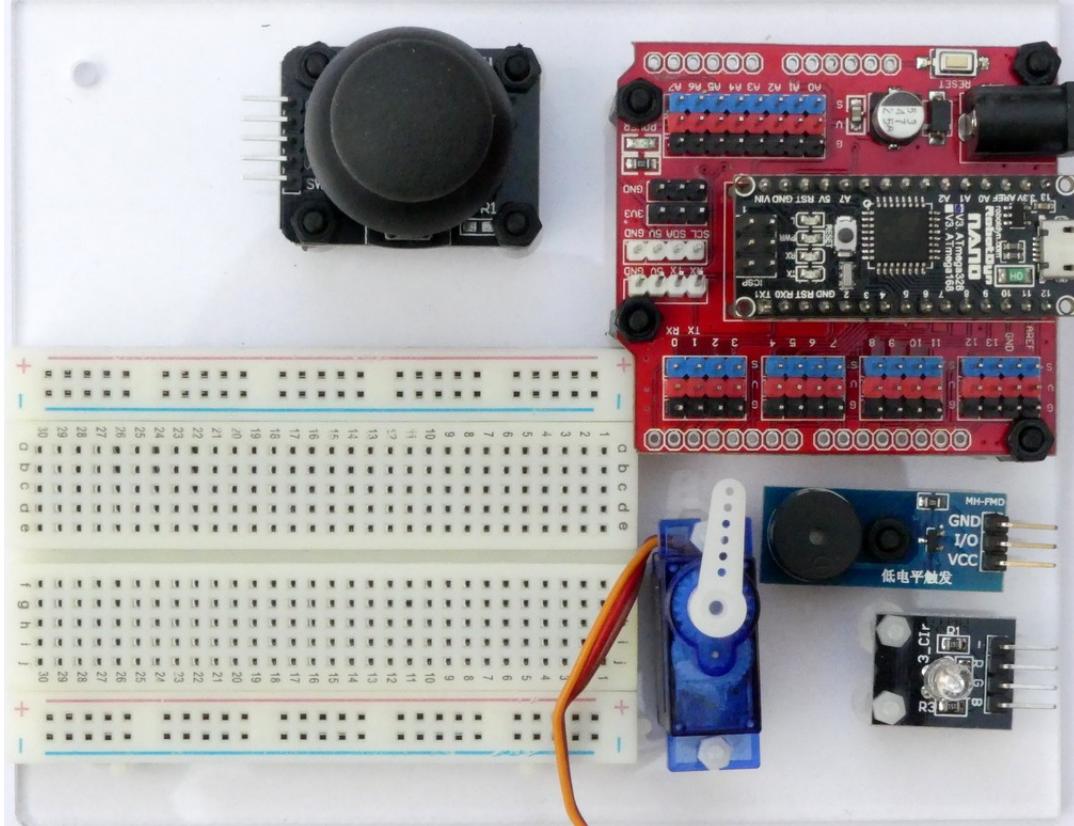
- Dubbelklik op een cel zodat je een menu krijgt
  - Eerst voor *direction* dan voor *function*.
- Zorg ervoor dat je alle pennen namen geeft als je iets anders kiest dan void
  - Dus niet opslaan als je nog steeds rode randen om een cel heen hebt.  
scratchClient zal niet willen opstarten met zo'n config bestand.
- Het tool controleert op verkeerde configuraties. Bijvoorbeeld:
  - Pennen 0, 1 en 13 kunnen niet gebruikt worden.
  - *Analoog in* alleen beschikbaar op A0 t/m A7.
  - Pennen A6 en A7 kunnen alleen voor *analoog in* gebruikt worden.
  - Pennen 9 en 10 kunnen niet voor *PWM* gebruikt worden als enige andere pen is geconfigureerd voor een *servo* (zie later).

# Sla het config bestand op

- Sla het config bestand op in de map *PiAndMore* op het bureaublad
- Noem het *PiAndMore.xml*
- **Laat het tool open** voor de volgende opgaven.

Deel 5: Bedraden van het bord en  
test met het eerste config bestand

# Leg het bord op deze manier voor je

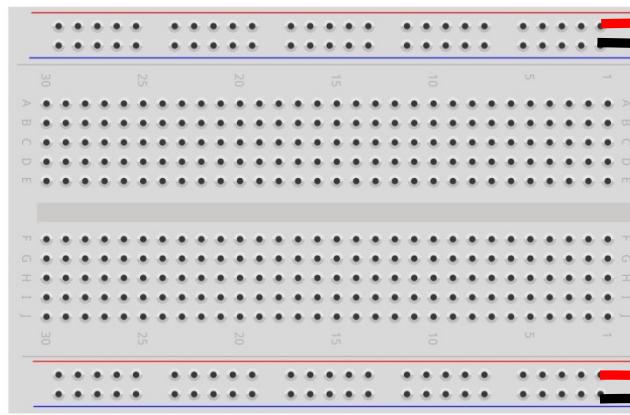
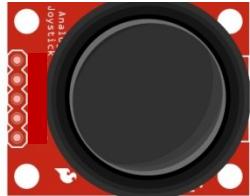


# Gebruik korte draden en gebruik de aangegeven gaten

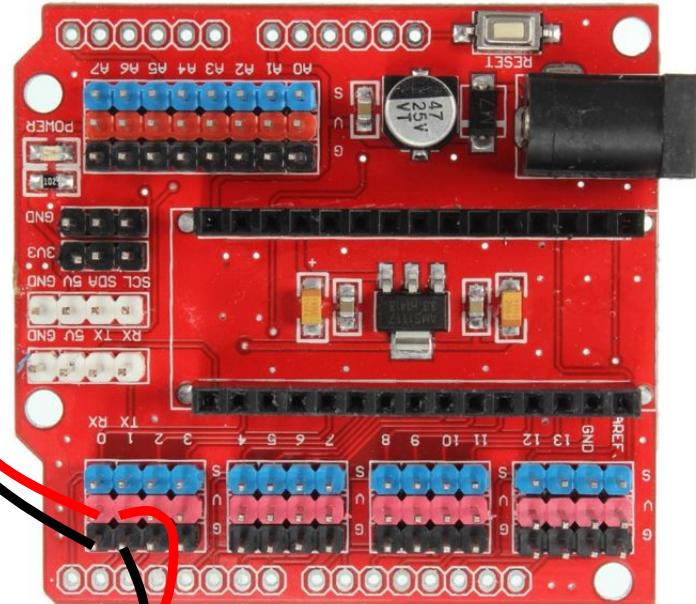
- Er is een stel korte draden (10 cm) en een stel lange (15 cm)
- Gebruik de kortste die passen
  - Dat geeft minder een oerwoud aan draden
  - En anders zou je wel eens te weinig lange draden kunnen hebben
- Let niet op kleuren van de draden.
  - Helaas zijn er onvoldoende draden van juiste kleur om dat goed te kunnen krijgen.
- Je kunt in principe bouwen op verschillende plaatsen op het breadboard
  - Echter, gebruik svp de aangegeven kolommen om te vermijden dat je te weinig ruimte hebt op het breadboard later in de les.

# Sluit de voedingsdraden aan

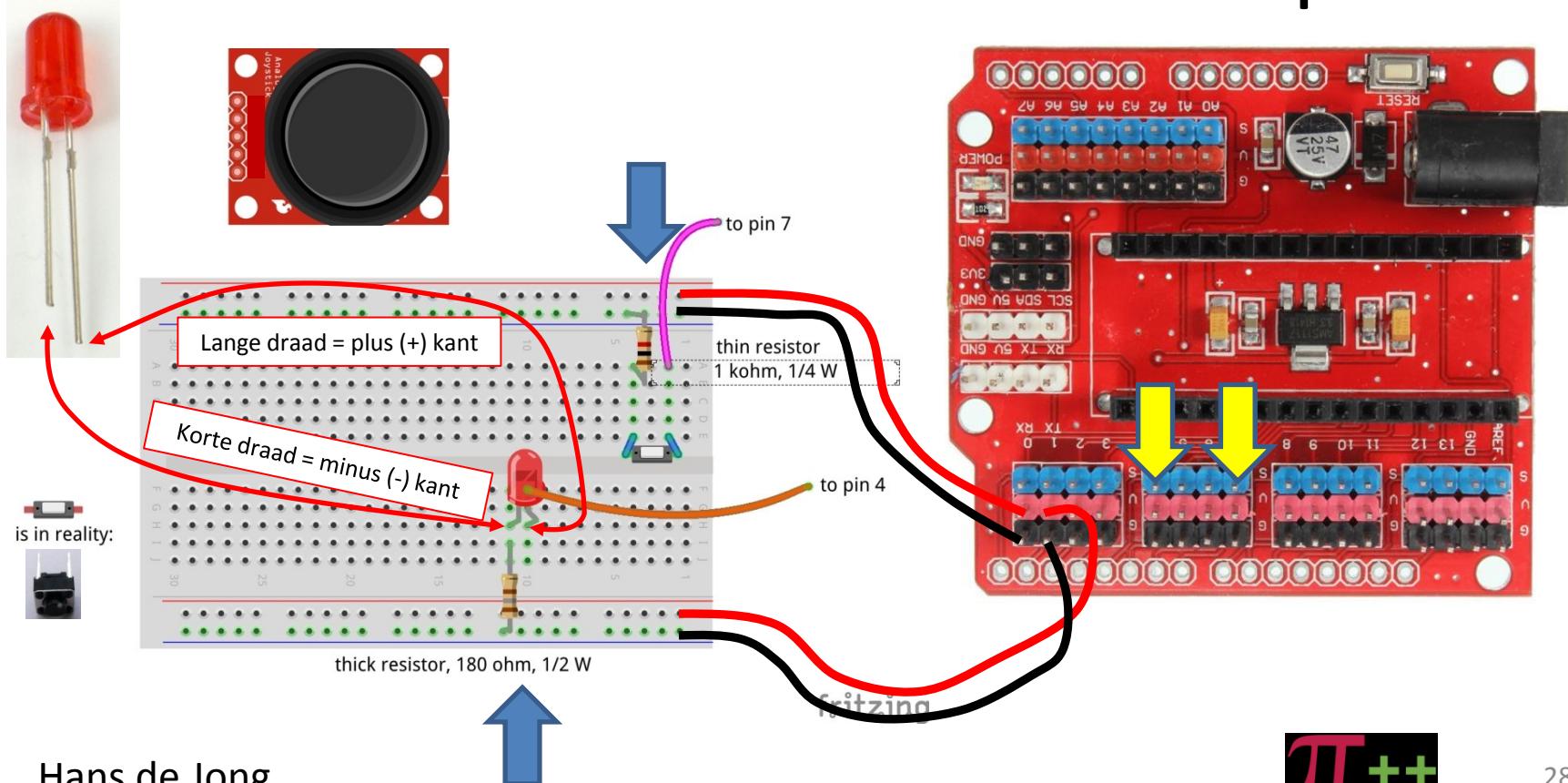
GND  
+5V  
VRx  
VRy  
SW



fritzing



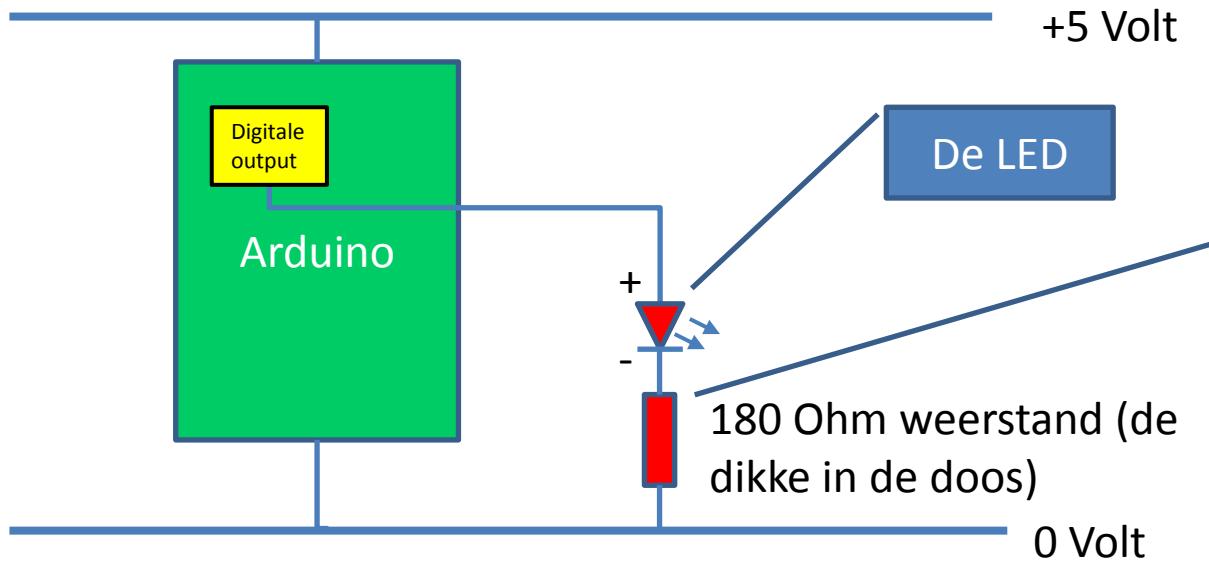
# Sluit de rode LED en de drukknop aan



Hans de Jong



# Waarom een weerstand in serie met de LED?

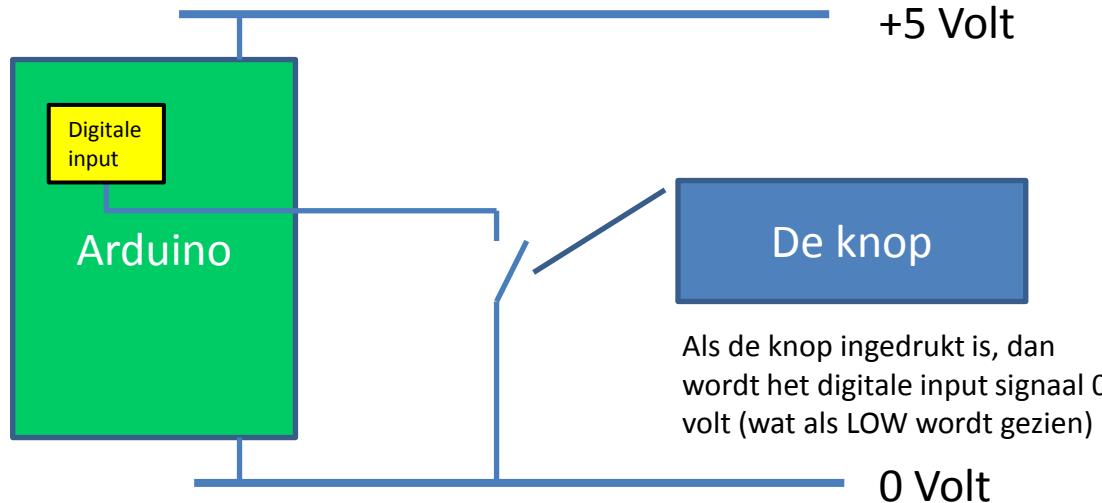


Deze weerstand is om de stroom door de diode (de LED) te beperken (anders blaas je de LED en/of de Arduino op).

# Wat moet Arduino krijgen op een Digitale Input pen?

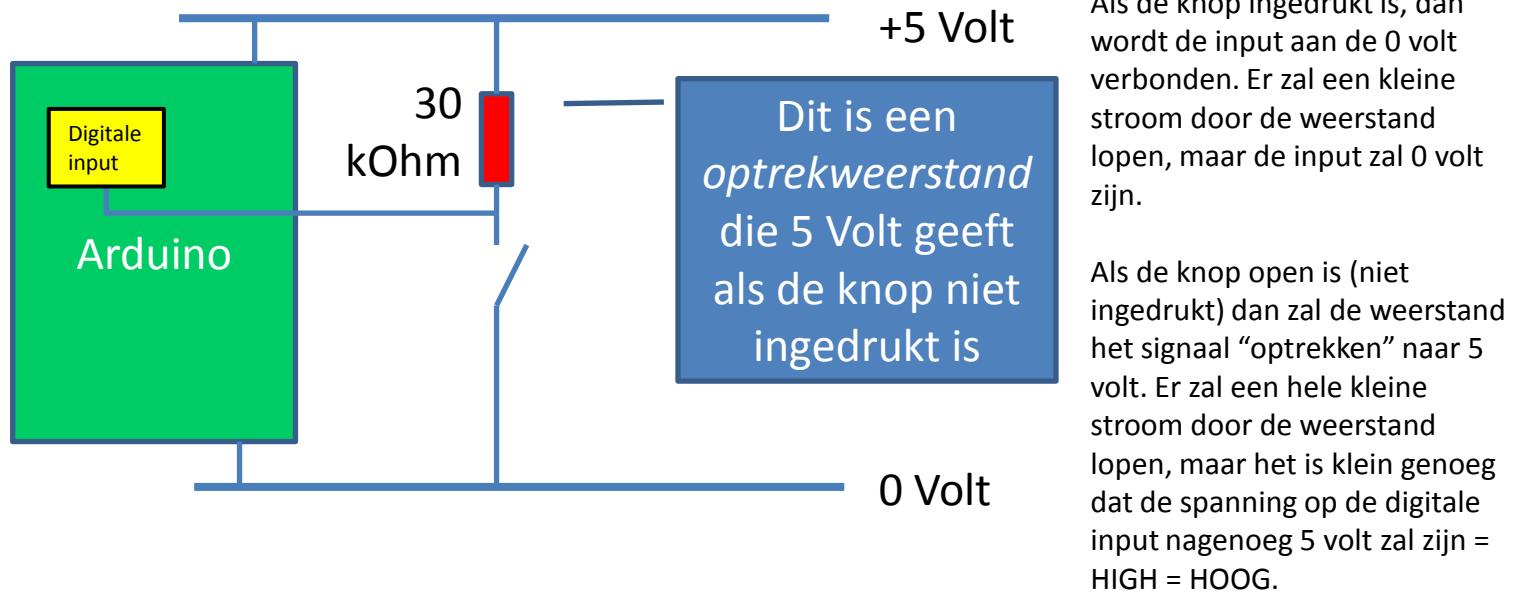
- Een digitale input pin van Arduino moet een van deze input waarden krijgen:
  - 0 volt input (om precies te zijn: 0 tot 1.5 volt wordt gezien als een LOW (= LAAG) input signaal)
  - 5 volt input (om precies te zijn: 2.5 volt tot 5 volt wordt gezien als een HIGH (=HOOG) input signaal)
- Als de Arduino iets krijgt tussen 1.5 volt en 2.5 volt is de interpretatie van het signaal niet stabiel (kan LOW of HIGH zijn).
- Als de Arduino geen enkel signaal krijgt dan is het signaal niet stabiel (kan LOW of HIGH zijn).

# Het maken van een LOW (= laag) signaal

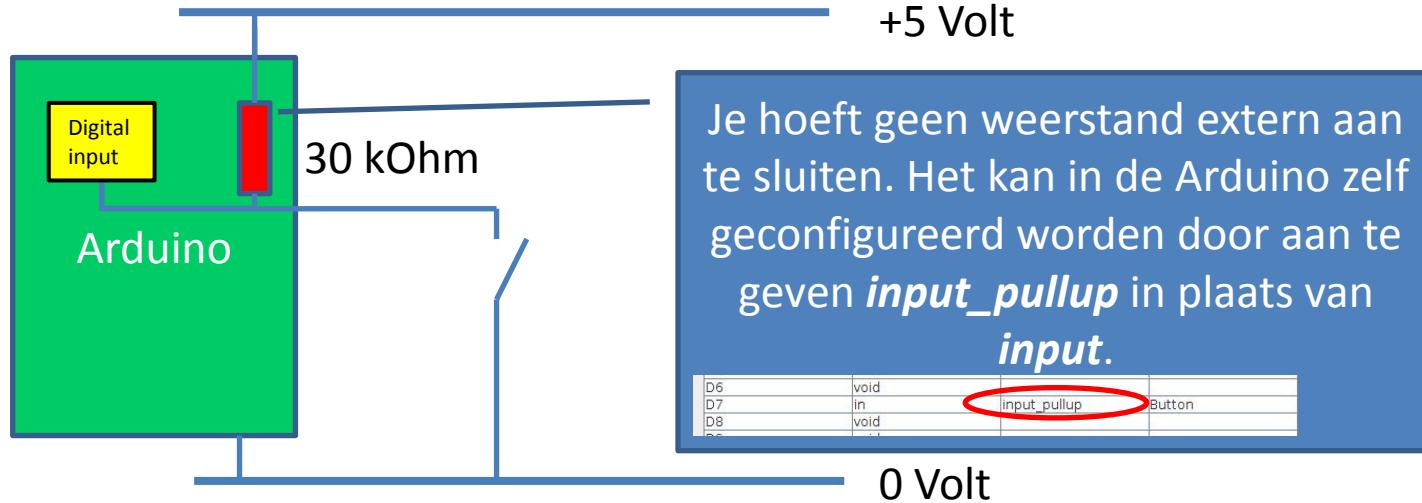


Maar wat ziet de Arduino als de knop **niet** ingedrukt is?

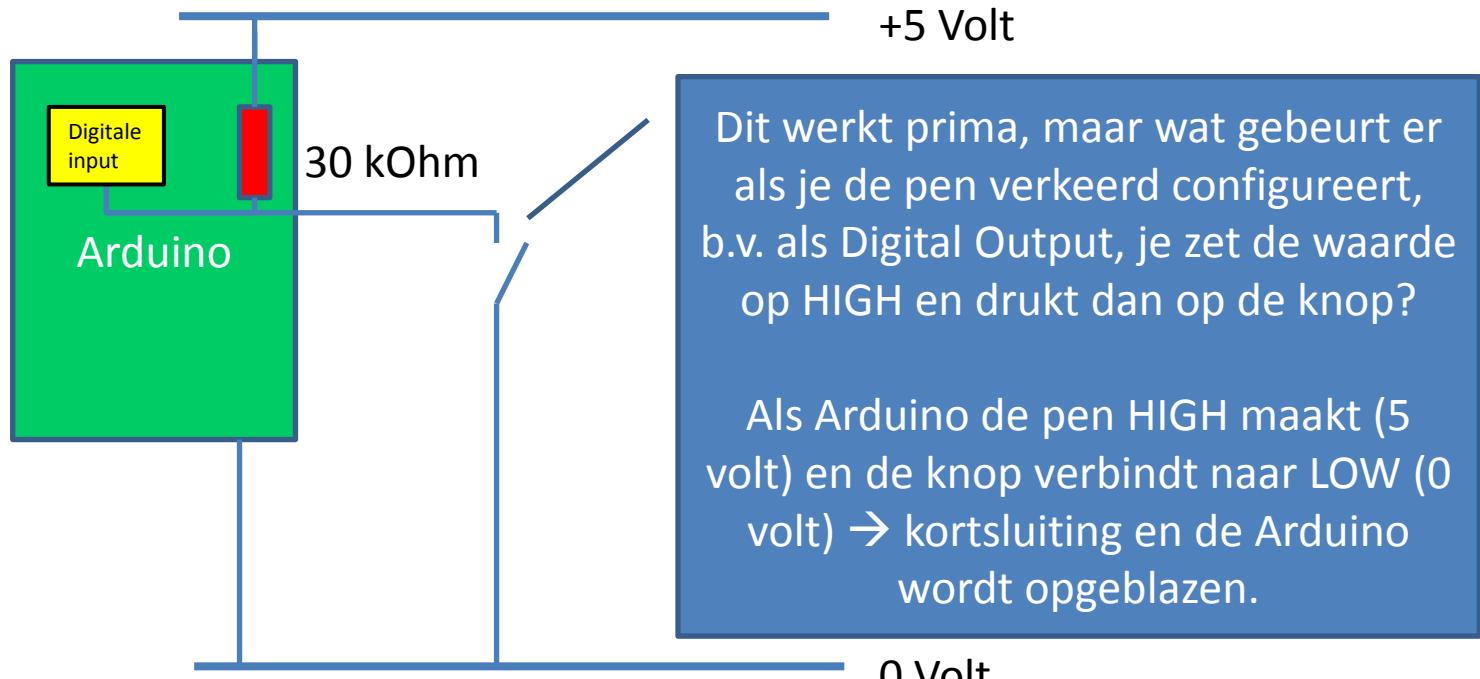
# Het maken van een HIGH (= hoog) signaal



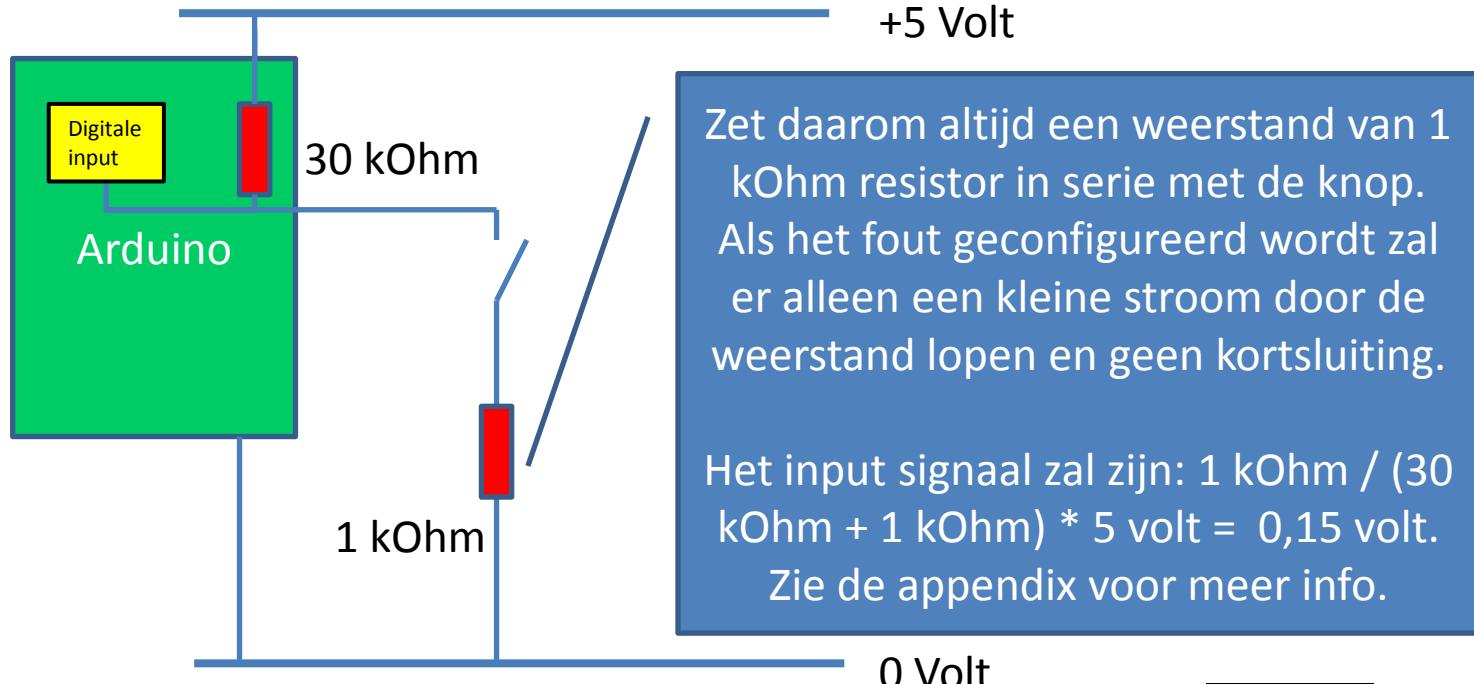
# Maak de input HIGH zonder externe weerstand



# Gevaar als je verkeerd configureert



# Weerstand in serie om schade te voorkomen bij verkeerde configuratie



# Check / dubbel check

- Controleer nu **beiden** dat de bedrading correct is.

# Maak een script bestand om scratchClient te starten

- Maak een nieuw bestand in de map PiAndMore op het bureaublad
  - Noem die StartSC.bash
- Zet dit in het bestand (kopieer en plak uit deze presentatie):

```
#!/bin/bash
python ~/scratchClient/src/scratchClient.py -c
~/Desktop/PiAndMore/PiAndMore.xml
pause -p "Druk op Enter om verder te gaan"
```

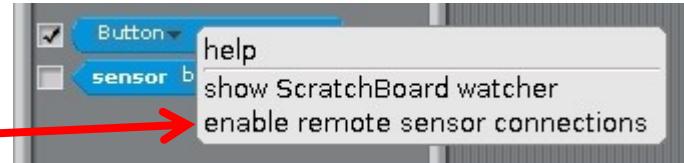
- Maak het bestand uitvoerbaar (bestand eigenschappen, rechten)
- Begrijp je wat het bestand doet?
  - Zo niet, dan moet je het vragen

# Breng alles samen

- Sluit de 9 Volt steker aan op het bord en schakel de stroom in
- Sluit de USB connector aan op de Arduino, nu wel via de USB Hub
- Start scratchClient met het script dat je net gemaakt hebt (bijv. dubbelklikken en als je gevraagd wordt kies dan *Uitvoeren in terminal*)
- Er zal geklaagd worden dat er geen verbinding is met Scratch
  - Hetgeen logisch is, omdat Scratch nog niet gestart is.

# Maak het Scratch programma

- Start Scratch  Programmeren →
- *Sensor verbinding met elders inschakelen* (klik rechts op sensor waarde)
- Maak de variabele *Big Red LED*, beschikbaar voor alle sprites
- Maak de variabele zichtbaar (zet een vinkje voor de variabele)
- Maak de *Button* sensor zichtbaar
- Sla het bestand op in de map *PiAndMore* op het bureaublad
  - De naam maakt niet uit, b.v. *PiAndMore.sb*



# Werkt het? (zie de volgende pagina voor hulp)

- Als de LED op de Arduino langzaam knippert, dan pas is het config bestand gedownload en werkt scratchClient.
- Dit kan 5 seconde duren.
- Zet de waarde van variabele *Big Red LED* op 1.
  - Je kunt de schuif op de variabele gebruiken. Zie je geen schuif?  
Dubbeklik dan een paar keer op de variabele.
  - Brandt de LED?
- Druk op de knop op het breadboard.
  - Zie je de waarde van de sensor *Button* veranderen van 1 naar 0?

# Wat als het niet werkt?

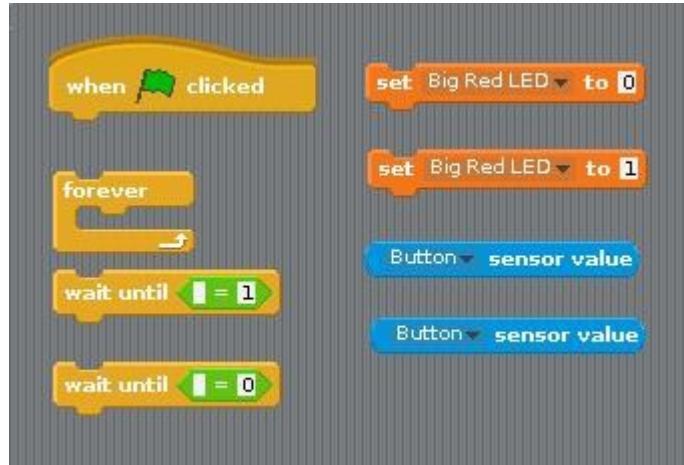
- Controleer of je meerdere instanties van Scratch open hebt
  - scratchClient kan slechts werken met één Scratch tegelijk.
- Controleer de output van scratchClient in het terminal venster
  - Heeft scratchClient het bord gevonden?
- Soms, vooral na het opstarten, als alles verder goed lijkt, kan het helpen om de USB stekker uit het bord te trekken en er weer in te stoppen.
- Probeer de variabelen to monitoren (zie volgende pagina).
- Mocht alles falen, dan kan scratchClient gestart worden met tracing aan.
- Maar we zijn er natuurlijk om je te helpen bij elke stap.

# Het monitoren van de variabelen

- Open de browser
- Tik: *localhost:8080*
- Klik op *adapters*
- Let op de input en output richtingen:
  - Output van een adapter is een input voor Scratch.
  - Output van scratch is een input voor de adapters.
  - Daarom lijkt het dat de namen input en output omgekeerd zijn van wat in het config bestand staat.
  - Daarom is het het handigste om naar de namen van de variabelen te kijken.
- Je zult zien dat de waarden alleen worden weergegeven nadat de variabele een andere waarde gekregen heeft (anders staat er een vraagteken (?) ).

# Programma in Scratch

- Maak een programma in Scratch dat de rode LED laat branden als je de knop indrukt.
- Dit heb je nodig



# Denk je nog aan ...

- Sla je Scratch programma regelmatig op. Anders is het verloren als de spanning wegvalt.
  - Het wegvallen van de spanning kan makkelijk gebeuren, want je bent bezig om kabels erin en eruit te halen en dan kan de voedingsteker van de Raspberry Pi los raken.
- Om de zoveel tijd het Scratch bestand naar een backup bestand te kopiëren?
  - Aanbeveling: geef die kopieen een volgnummer.

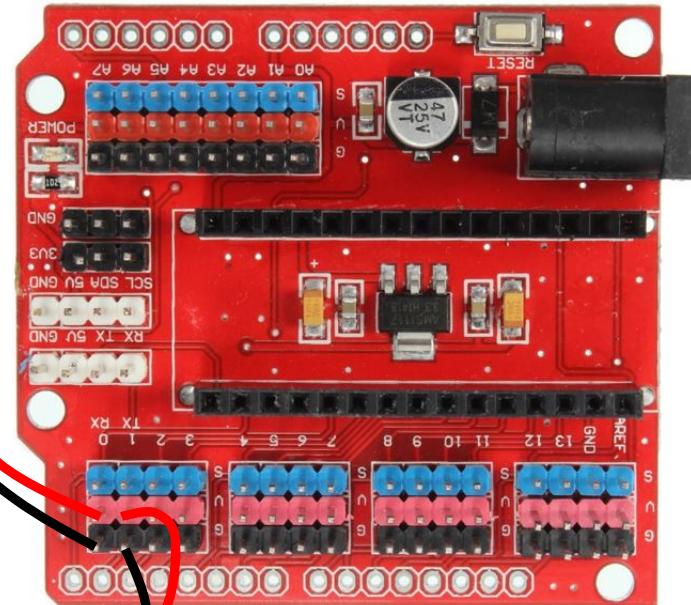
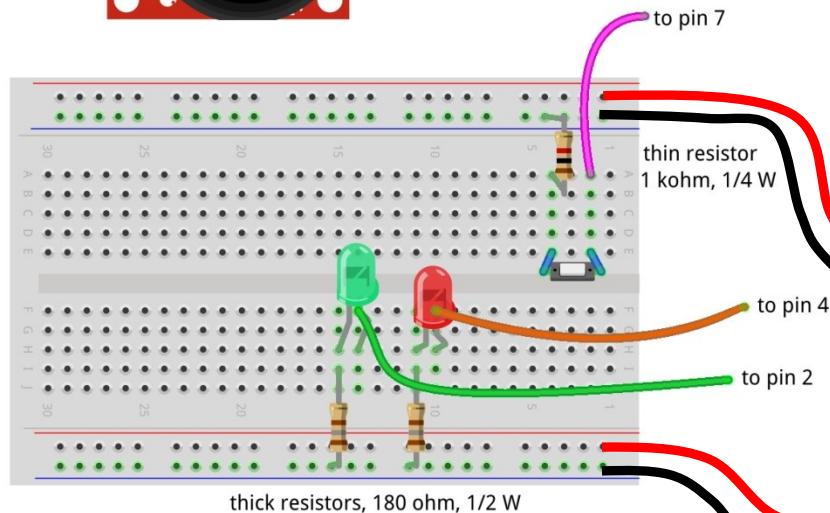
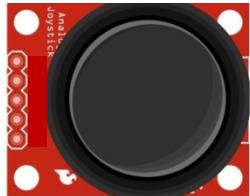
# Deel 6: Voeg de grote groene LED toe

# Verander de bedrading op het bord

- Als je de bedrading op het bord verandert ...
  - Trek dan eerst de USB kabel eruit (uit het bord of uit de hub)
  - Schakel de 9V voeding uit

# Het toevoegen van de groene LED

GND  
+5V  
VRx  
VRy  
SW



# Check en sluit opnieuw aan

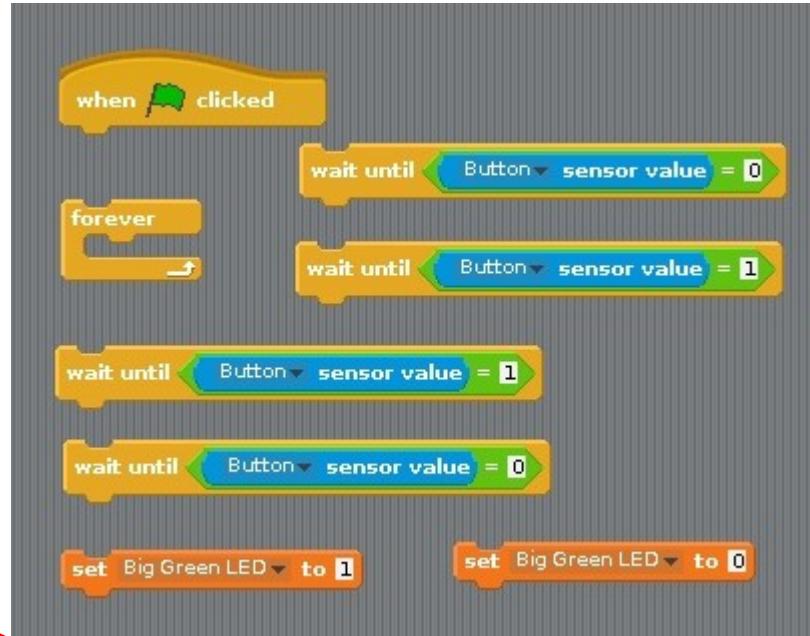
- Check dat de bedrading correct is
- Schakel de 9 volt voeding weer in
- Sluit de USB kabel weer aan

# Werk het config bestand bij en start scratchClient opnieuw

- Stop scratchClient (sluit het venster)
- Gebruik het config tool (die nog steeds open zou moeten zijn)
- Definieer een output (direction: out, function: output) op pen 2 en noem die *Big Green LED*
- Sla het config bestand op (en laat het venster open)
- Herstart scratchClient met het script dat je eerder gemaakt hebt

# Werk het Scratch programma bij

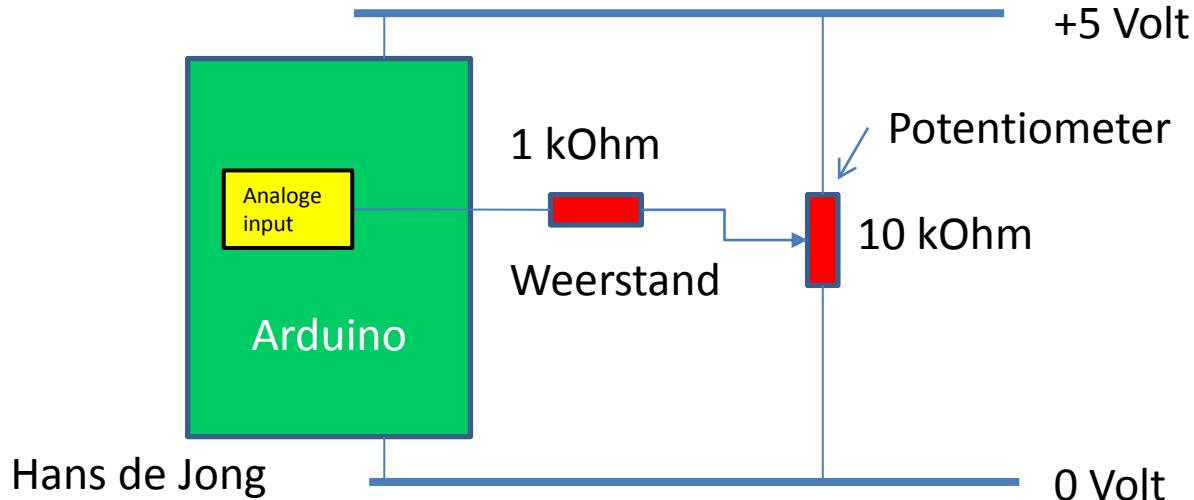
- Definieer de variabele *Big Green LED*
- Maak een sprite voor de groene LED
- Maak in die sprite een programma dat dit doet:
  - Druk één keer: LED gaat aan
  - Druk nogmaals: LED gaat uit
- Je hebt deze programma onderdelen nodig.



# Deel 7: Voeg analoge input toe

# De elektronica van een potentiometer

- De potmeter is een weerstand die wordt aangesloten tussen 5 V en 0 V.
- De spanning verandert gelijkmatig over de lengte van de weerstand.
  - Bijvoorbeeld in het midden zal het 2.5 Volt zijn
  - Dus dit is een analoog signaal. De waarde kan veranderen tussen 5 V en 0 V.
- Er zit een 1 kOhm weerstand in serie om de reden die we eerder hebben besproken.

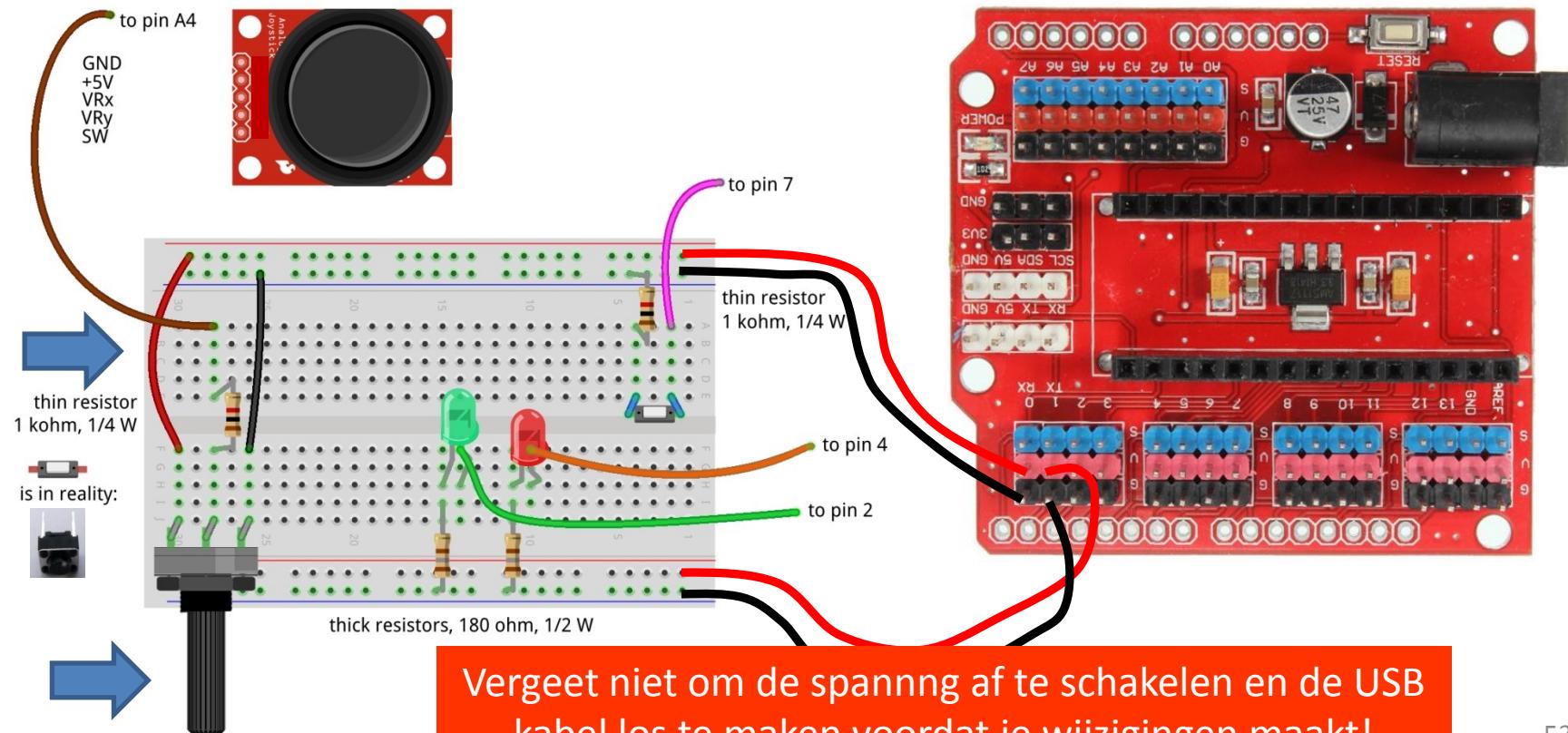


Bedenk wat er gebeurt als de 1 kOhm weerstand er niet zou zijn en de pen zou gebruikt worden als output en de potmeter zo dicht staan bij de 0 Volt kant.

Dan zou de Arduino 5 Volt kunnen afgeven en de potmeter zou dit direct doorverbinden met de 0 Volt → kortsluiting.



# Voeg analoge input toe



# Werk het config bestand bij

- Werk nogmaals het config bestand bij
  - Maak *Potmeter* op pen A4 (direction: in, function: analog)
- Nu dat je toch bezig bent met het config bestand, maak alvast voor de volgende stappen:
  - Maak *Servo 1* op pen 12 (direction: out, function: servo)
  - Maak *Big Blue LED* op pen 5 (direction: out, function: PWM)
  - Maak *Buzzer* op pen 11 (direction: out, function: PWM)
- Vergeet niet om het op te slaan

# Test of het werkt

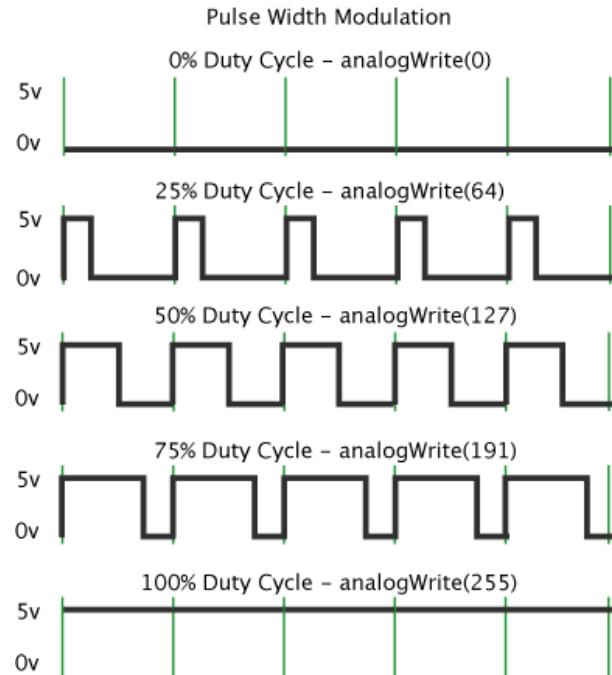
- Stop and herstart scratchClient
- Sluit de USB connector weer aan en ook de voeding
- Maak de sensor *Potmeter* zichtbaar
- Draai aan de knop en zie de waarden van *Potmeter* veranderen
  - Tussen ongeveer 0 and ongeveer 1024
- We gaan in de volgende stap een Scratch programma maken om het te gebruiken

# Deel 8: Pulsbreedte Modulatie (Pulse Width Modulation = PWM)

# Waar we zijn ...

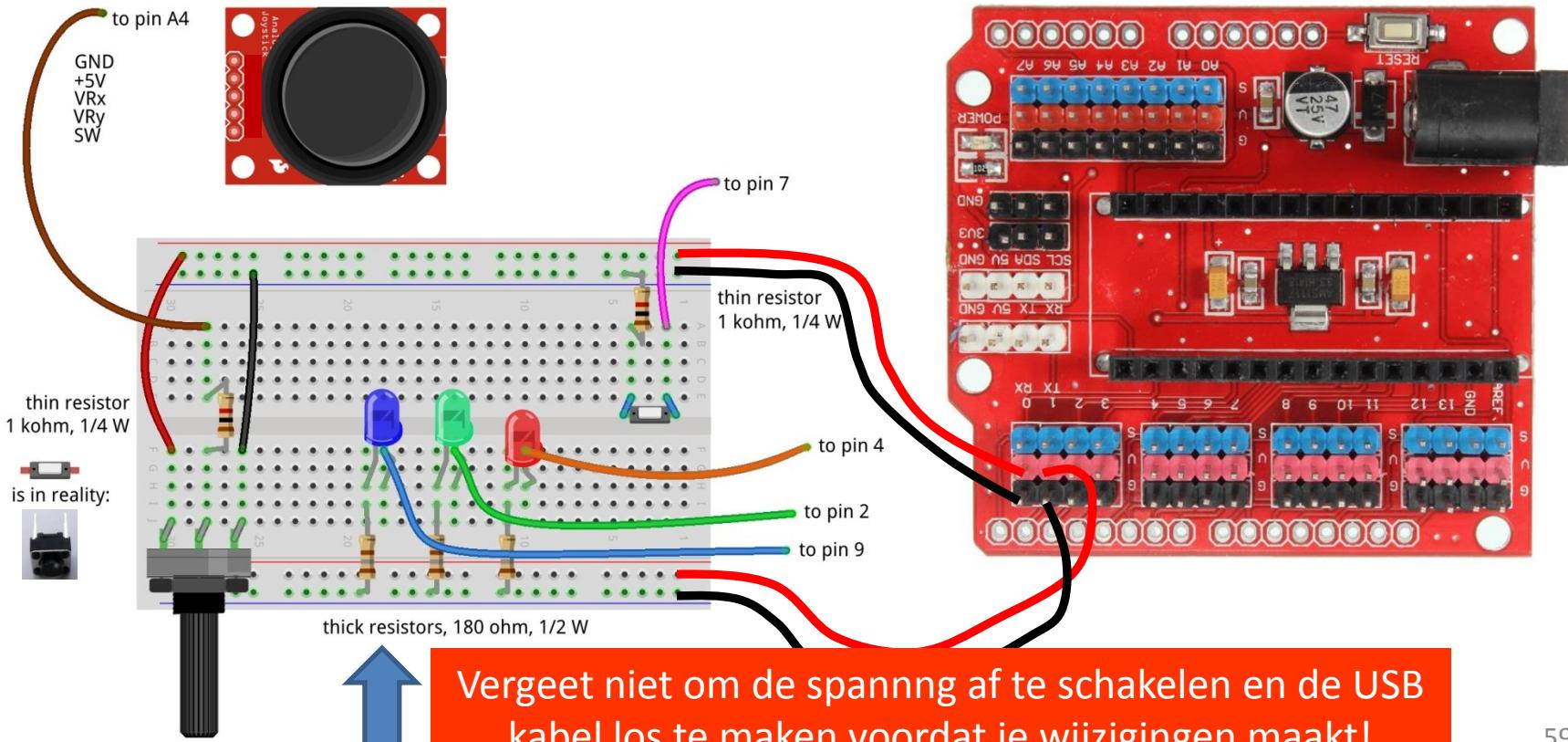
- Wat we gezien hebben:
  - Digitale Input (de drukknop)
  - Digitale Output (de LEDs)
  - Analoge Input (de potentiometer)
- Dus wat zou de meest logische volgende stap zijn?
  - Analoge Output? → maar dat bestaat niet!
  - Echter, er is een goed alternatief: Pulse Breedte Modulatie (PWM – Pulse Width Modulation)

# Pulsbreedte Modulatie (PWM)



- Door het moduleren (veranderen) van de pulsbreedte verandert de hoeveelheid energie die aan b.v. een LED wordt gegeven. Daardoor verandert de intensiteit waarmee je de LED ziet branden.
- De LED knippert in de praktijk met 800 flitsen per seconde.
- Echter, het menselijk oog kan niet meer zien dan 100 flitsen / seconde. Daarom lijkt het constant te branden met minder intensiteit.

# Het dimmen van een LED met PWM



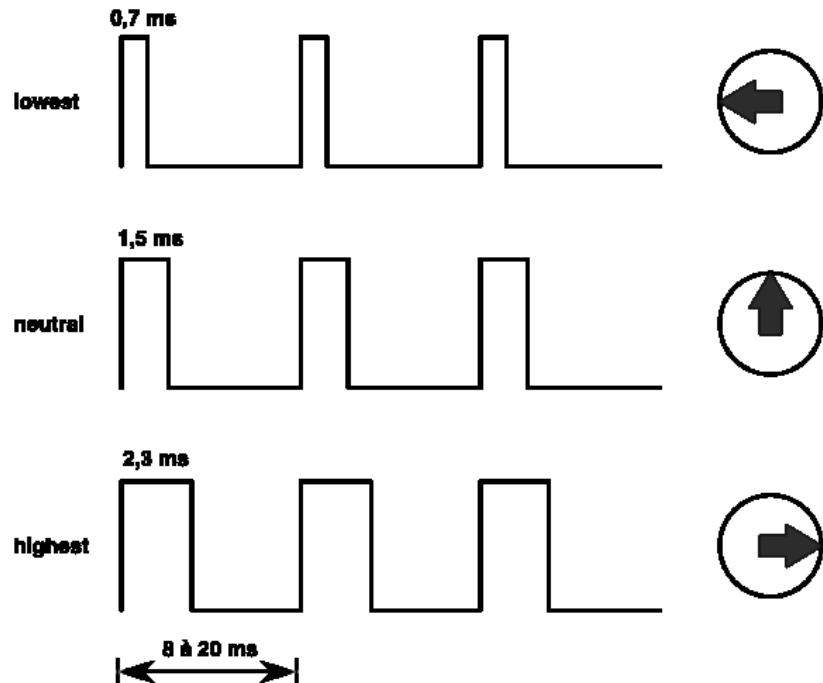
# Test

- Schakel de voeding in en sluit de USB kabel opnieuw aan
- Je zult zien dat scratchClient het bord weer vindt
- Je hebt het config bestand al bijgewerkt in de vorige stap, dus scratchClient hoeft niet opnieuw gestart te worden
- Definieer in Scratch een variable *Big Blue LED*
- Maak de variabele zichtbaar
- Geef *Big Blue LED* waarden tussen 0 en 255
  - Je kunt de schuif op de variabele gebruiken voor waarden tussen 0 en 100
  - Voor waarden boven 100 gebruiken we een Scratch programma, zie de volgende pagina
- Verandert de helderheid van de LED?

# Bestuur de grote blauwe LED met de Potmeter (via Scratch)

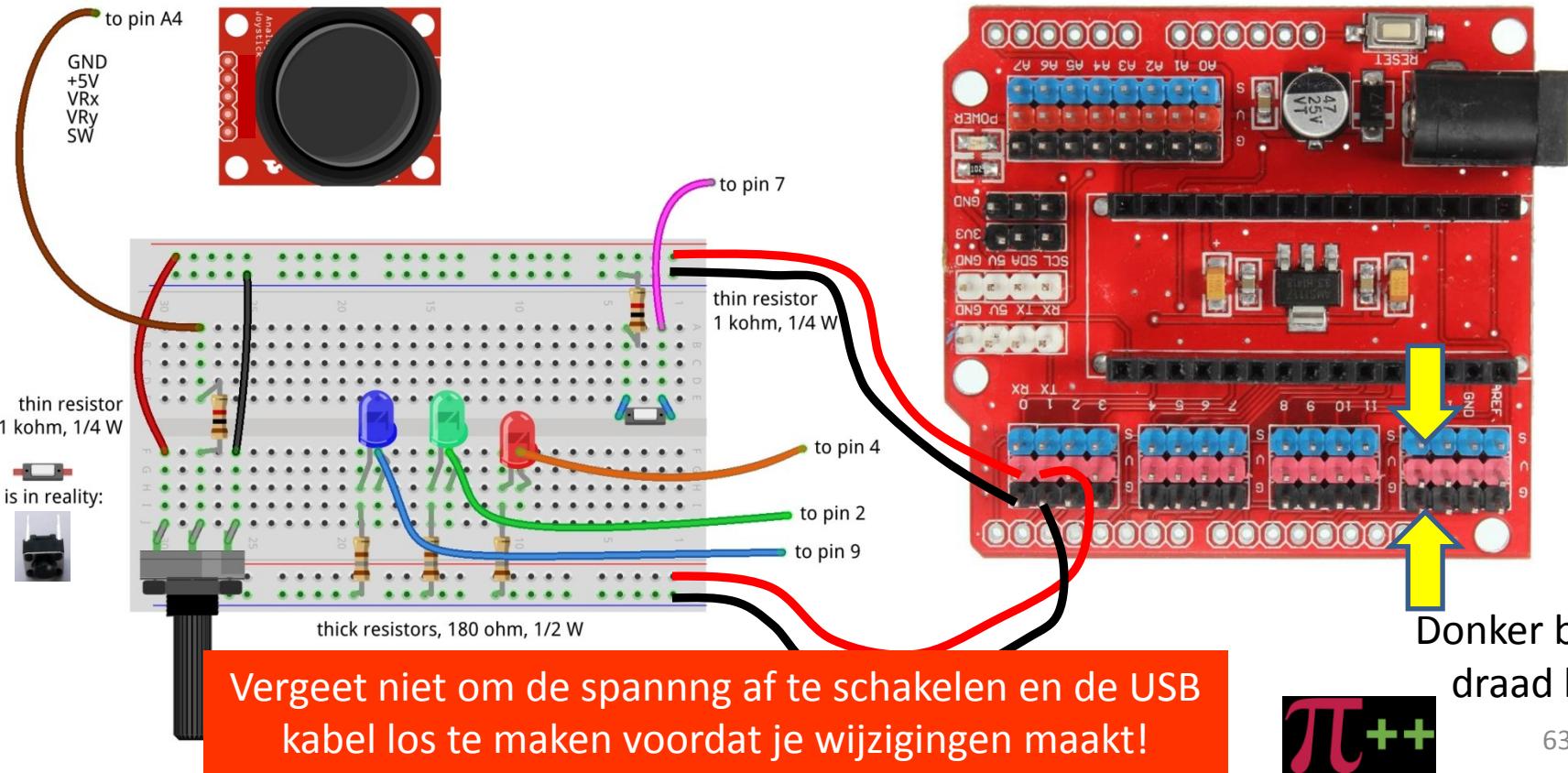
- Maak een stukje programma dat de waarde van de potentiometer (tussen 0 en 1024) omzet naar 0 tot 255 (dus deel door 4) en gebruik dat om de waarde van *Big Blue LED* te zetten.
- Probeer uit of het draaien aan de potmeter de intensiteit van de LED verandert.
- Merk op dat als je naar rechts draait, de intensiteit afneemt.
  - Je zou vast verwachten dat het andersom is (want dat ben je gewend) ...
  - Hoe zou je dat heel simpel kunnen realiseren door twee draden om te wisselen?

# Besturen van een servo met PWM



- De positie van de servo wordt veranderd door pulsen te zenden van verschillende breedte.
- De servo kijkt naar de pulsbreedte en draait zoals gewenst.
- De servo krijgt apart voeding.
- Bij een servo, regelt de pulsbreedte modulatie niet de hoeveelheid energie die aan de servo wordt toegevoerd.
- Bij een servo is pulsbreedte modulatie daarom eigenlijk meer een communicatie protocol.

# Voeg een servo toe op pin 12



# Testen van de servo

- Sluit USB en de voeding opnieuw aan.
- Maak in Scratch een variabele *Servo 1* en maak die zichtbaar.
- Probeer het waarden te geven tussen 0 en 100 via de schuif op de variabele.
  - De servo kan waarden aan tussen 0 en 180.
  - Beweegt de servo?
- Verander de lus waar je de waarde van *Big Blue LED* zet zo dat je ook afhankelijk van de waarde van de potmeter ook *Servo 1* verandert.
  - Maar pas op: de waarden moeten tussen 0 en 180 liggen.
  - Om te grote getallen te vermijden: deel eerst door 1024, en vermenigvuldig dan met 180.
- Je kunt soms jitter zien (de servo blijft niet goed op dezelfde plaats).
  - Omdat de waarden van de potmeter een beetje zullen veranderen en omdat er de voedingsspanning wat kan veranderen is de waarde die die potmeter afgeeft niet constant.
  - In de les voor gevorderden laten we zien hoe we met jitter om gaan.



# Stuur een zoemer aan met PWM

- De zoemer laat een toon horen als die een signaal krijgt met een hoorbare frequentie.
- Een PWM signaal op Arduino geeft een signaal van ca. 800 Hz.
  - Verschillend voor de verschillende pennen
- Je zult zien dat er niet veel invloed is van het veranderen van de pulsbreedte.

# Sluit de zoemer aan op pin 11

Kijk goed naar de draden. Sluit Vcc, GND and I/O juist aan!

thin resistor 1 kohm, 1/4 W

is in reality:

to pin A4

GND  
+5V  
VRx  
VRy  
SW

to pin 7

thin resistor 1 kohm, 1/4 W

to pin 4

to pin 2

to pin 9

thick resistors, 180 ohm, 1/2 W

Vergeet niet om de spanning af te schakelen en de USB kabel los te maken voordat je wijzigingen maakt!

$\pi^{++}$

66

# Het testen van de zoemer

- Het is niet nodig om scratchClient opnieuw te starten, omdat het config bestand niet veranderd is.
- Maak een variable *Buzzer* in Scratch
- Geef het waarden tussen 0 en 100
  - Gebruik de schuif op de variabele
  - Heeft de waarde veel effect?
- Voeg de het zetten van de waarde van de zoemer toe aan de lus waar je al de waarde zet van de servo en van Big Blue LED.
  - Zoals met de LED moeten de waarden tussen 0 en 255.
- Maar test het snel (om de oren van je buren te sparen ☺).

# PWM beperkingen van Arduino (Nano and Uno)

- In het algemeen is PWM beschikbaar op pennen 3, 5, 6, 9, 10 en 11.
- De servo kan worden geconfigureerd op deze pennen, maar ook op 2, 4, 7, 8 en 12
- Als er een servo zit op enige pen, dan kan op pennen 9 en 10 geen PWM meer worden gedaan.
  - Het config tool waarschuwt je als je het fout doet.

# Deel 9: Maak een Scratch programma

Extra materiaal voor het geval dat je tijd over hebt

# Integrate with sprites

- Load the following sprite
  - Big Red LED
  - Sits in the *PiAndMore* folder on the desktop in a folder *Sprites*.
- Take a look at the code of the sprite.
- Update the code that you wrote earlier today for the *Big Red LED* so that it sends messages to the sprite to turn the LED on and off

# Add the Big Green LED sprite

- Same that you did with the Big Red LED sprite, but now for the Big Green LED sprite.

# Add the Big Blue LED sprite

- Load the Big Blue LED sprite
- Analyse the code in that sprite, which is different
  - The Big Blue LED sprite will move over the screen as the intensity changes
  - You can also move it with your mouse and see the intensity change.
- Update your code that you wrote earlier to make use of the sprite.

# Part 10: Take your work home

# Do you want to take your work home?

- If you brought your own USB stick, then connect it and copy the *PiAndMore* folder on the desktop
- The rest of the material you can download from  
[www.github.com](http://www.github.com)
- Take the flyer with you to remember where to find the material on github.

# Part 11: Summary & take aways

# Take aways of the beginners workshop

- With scratchClient you define:
  - Function of each pin
  - Symbolic name for each configured pin
- scratchClient config is the tool to setup the configuration
- Restart scratchClient after you changed the configuration
- Put a resistor in series with LEDs
- Put a resistor in series with switches
- Put a resistor in series with the middle contact of a potentiometer.
- Configure a pull up resistor if the input signal goes between 0 Volt and being open rather than between 0 Volt and 3 to 5 Volt.
- Output signals are controlled from Scratch by giving a variable a value
- Input signals are monitored by the sensor programming elements
- You can monitor the value of all pins from the browser
- **scratchClient can do much more...**
- Functions that a pin on Arduino can have:
  - Digital In
  - Digital Out
  - Analog In
  - *No Analog Out*
  - Pulse Width Modulation as alternative
    - For modulating the brightness of a LED
    - For controlling a servo
    - For controlling a buzzer
  - There are a few more, see the advanced workshop
  - You can configure pull up resistors on Digital In

# If you want to know more ...

- Join the advanced workshop
- Download the material and read the extensive documentation

# Part 12: Clean up / teardown

# If you continue to the Advanced Workshop ...

- You can continue to use your setup, with one exception:
- You need to move the Big Blue LED to pin 9
  - Direction: out, function: output (so no PWM)
- In the mean time you should know how to do this, otherwise please ask.

# If your scratchClient day ends here ...

- Unplug the board from USB and power off the device
- Please remove all components and wires from the **breadboard**
- Remove all wires from the **Arduino board**.
- **Leave the wires on the 3 color LED** (you did not use that)
- **Leave the wires on the buzzer**
- If something is broken, please
  - Throw it away or hand it in (if it is unclear)
  - Put a note in the box that it is missing
  - Do not put anything that is broken back in the box
- Leave the Arduino running
- Let us know what you thought about this workshop, now orally or later by email
  - [hans.piam@hanselma.nl](mailto:hans.piam@hanselma.nl)
  - [heppg@web.de](mailto:heppg@web.de)

# More information

- All workshop material
  - [www.github.com](https://www.github.com) and search for *Weekendschool* or for *PiAndMore*
- scratchClient
  - [http://heppg.de/ikg/wordpress/?page\\_id=6](http://heppg.de/ikg/wordpress/?page_id=6)
- Scratch
  - <https://scratch.mit.edu/>
- Scratch on Raspberry Pi
  - <https://www.raspberrypi.org/forums/viewforum.php?f=77>
- Raspberry Pi
  - <https://www.raspberrypi.org/>
- Arduino
  - <https://www.arduino.cc/>

End of the beginners  
workshop

# Physical computing from Scratch using scratchClient – Advanced

*Control servos, LEDs and more from Scratch  
using RPi, Arduino, scratchClient*

Hans de Jong & Gerhard Hepp

Pi And More 10

Trier – 24 June 2017

# Before we start

# If you do not roll over from the beginners workshop ...

- If you are familiar with scratchClient, but not with the new config tool ...
  - ... you may want to look at some slides of the beginners workshop
- Other than that, you do **not** have to build up the complete setup what the people of the beginners workshop did.
  - Only those things on the next slide.
- You will however see the components of the beginners workshop in the diagrams.
  - Just ignore those.

# Make a script file to start scratchClient

- Then make a new file in the PiAndMore folder on the Desktop
  - Call it StartSC.bash

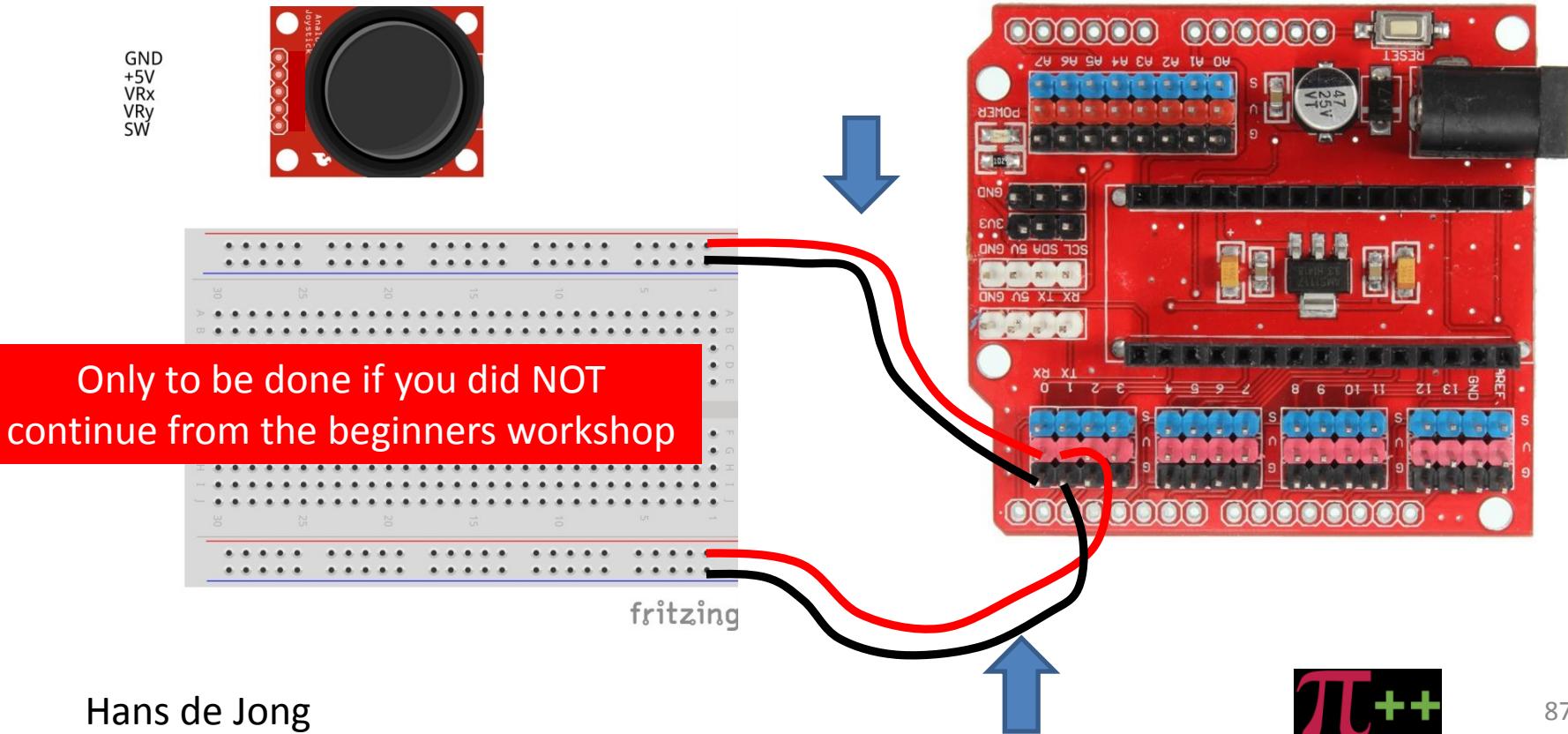
- Put this into the file (copy/paste from this presentation):

```
#!/bin/bash
sudo python ~/scratchClient/src/scratchClient.py -c
~/Desktop/PiAndMore/PiAndMore.xml
pause -p "Press Enter to continue"
```

- Make the file executable (file properties, permissions)
- Do you understand what the file does?
  - If not, please ask

Only to be done if you did NOT  
continue from the beginners workshop

# Connect the power wires



# Make the config file for all steps

- In order not to loose time, it is recommended to add all setups in the config file now.
  - If you continued from the beginners workshop then leave in what you have, only add.
    - D3: out, pwm, 3Color Red LED
    - D5: out, pwm, 3Color Blue LED
    - D6: out, pwm, 3Color Green LED
    - D8: in, counter\_pullup, Counter button
    - D10: in, counter, IR sensor
    - A0: in, analog, Rain Sensor
    - A1: in, input\_pullup, Tilt Sensor
    - A2: out, output, Relay
    - A3: in, analog, Sound Sensor
    - A5: in, input\_pullup, Joystick Button
    - A6: in, analog, JoyStickY
    - A7: in, analog, JoyStickX

# Make variables

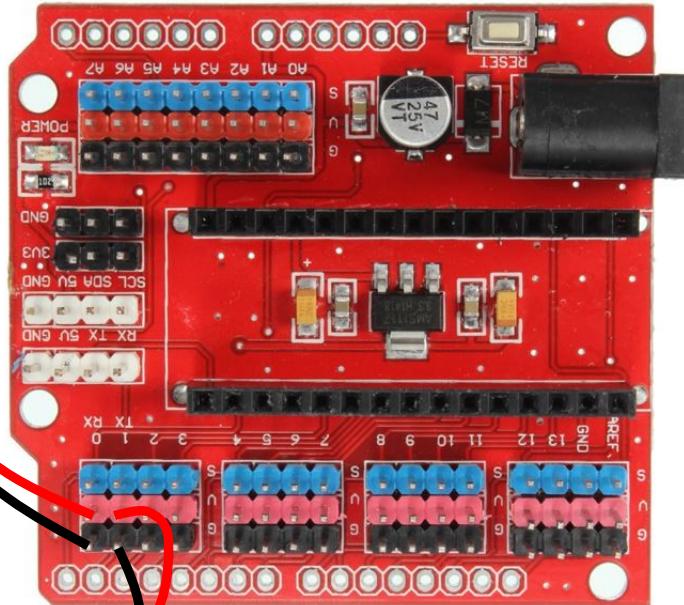
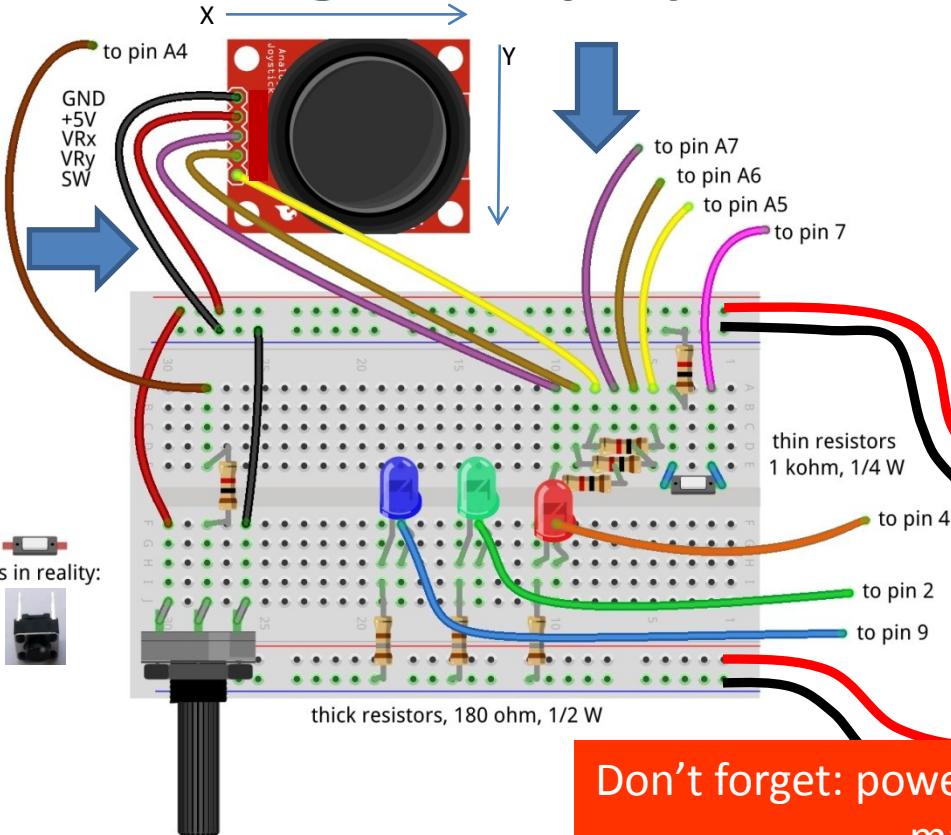
- Save the config file
- (Re)start scratchClient
- In Scratch, create a variable for each of the *out* parameters.
  - Note: variable names are case sensitive.
- In Scratch, make all of the sensors visible, so all *in* parameters.
  - Will only work if scratchClient successfully communicates to the board.

# Choose the topics you want to give priority

- The advanced workshop may be too short to do all activities.
  - So pick the order of the topics from the list
- The **red topics** teach you more about scratchClient.
- The **green topics** teach you more about electronics, sensors and engineering.
- If you rather would do pieces of the beginners workshop then that is fine as well.
- Joystick
  - Control position or control speed
  - Take care of calibration and drift
- 3-color LED
  - To make any color
- Counter function
  - With button or IR slotted sensor
- Rain sensor
- Tilt sensor
- Sound sensor
- Strongly link the config file to the Arduino
  - For safety
  - For connection errors
- Control multiple Arduinos concurrently
- Power On Self Test
- Controlling a relay
- Controlling 220 Volt appliances (only info)
- Controlling a camera (only info)

# Joy stick

# Adding the joy stick



Don't forget: power off and USB cable out before making changes!

# Try it out

- A joy stick has two small potentiometers and a button
  - When the joy stick is released (neutral position), the potentiometers are in the middle of the value range (middle between 0 and 1024)
  - There is one potentiometer for X and one for Y
  - The button is operated when pressing the button.
- Reconnect and repower
- Wait till the Arduino LED blinks slowly.
- Now try moving and pressing the joy stick. Look at the 3 joy stick values to change
  - JoystickX and JoystickY between about 0 and about 1024.
  - Joystick Button between 0 and 1

# Uses of Joysticks

- You can directly use the value of the joystick
  - E.g. to control the position of a servo
  - E.g. to control the intensity of a LED
  - When releasing the knob, it will move back to the middle value.
  - E.g.: LED = JoystickX
- You can alternatively use the joystick to determine the speed of the change
  - E.g. move a servo fast or slow. Let the servo stop at the latest position when you release the knob.
    - E.g.:  $\text{LED} = \text{LED} + (\text{JoystickX} - 512) / 200$  (512 = neutral position value, 200 = sensitivity)
  - You can also use the Scratch pen function draw on the screen.
    - And e.g. use the potentiometer to change pen width or color.
- Take care of drift and jitter (see next slide)

# Take care of drift and jitter

- Not all joysticks will give the same value if they are in the middle position
- Influenced by temperature, the value produced in the middle position can drift over time.
- Therefore build in some threshold around the middle position
  - If the middle position is 512, then do only react if the value changes by at least ca. 15, so e.g. > 525 or < 500.
  - Whether 15 is enough as threshold you will learn over time. Increase the value if it drifts more than that.
  - If you have a general program that works with several joysticks then you may have to use a larger threshold, or you need to calibrate (adapt the program for each particular servo).

# Control a 3-color LED

To make all colors

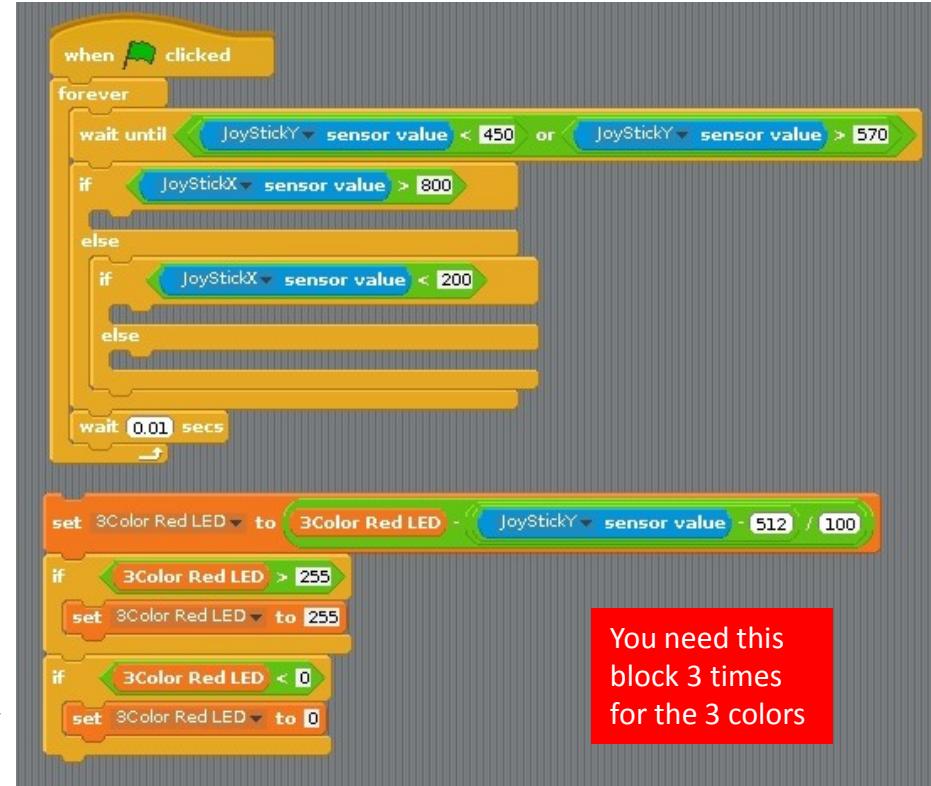
# Control a 3-color LED

- A 3-color LED has 3 LEDs in one package
  - Green
  - Red
  - Blue
- Use PWM to change the intensity of each color
  - In that way you can create the entire spectrum of light
  - Including white light
- Connect the 3-color LED carefully to the pins as defined in the config file.
- Also connect the minus (–) pin to GND
- Reconnect and repower
- No need to restart scratchClient (since no update to the config file was made)
- Give the corresponding variables in Scratch values between 0 and 100 with the slider on the variable.
  - Does it work?
- We will see in the next slide how to use values between 0 and 255

Don't forget: power off and USB cable out before making changes!

# Connecting the 3-color LED to the joy stick (via Scratch)

- Write a program in Scratch that:
  - Uses the value of the X direction of the joy stick to determine red, green or blue
  - Uses the value of the Y direction to change the intensity of the respective LED
  - When releasing the joy stick, the colors should stay
  - You will need these elements



# The new counting function

# Why a special counting function?

- You can count in Scratch, however you can only reliably count a few pulses per second.
- With this new counting function, you can do it much faster
- If you need to detect small pulses that scratchClient may not see reliable, you can use a counter function to detect whether a (short) pulse did arrive.

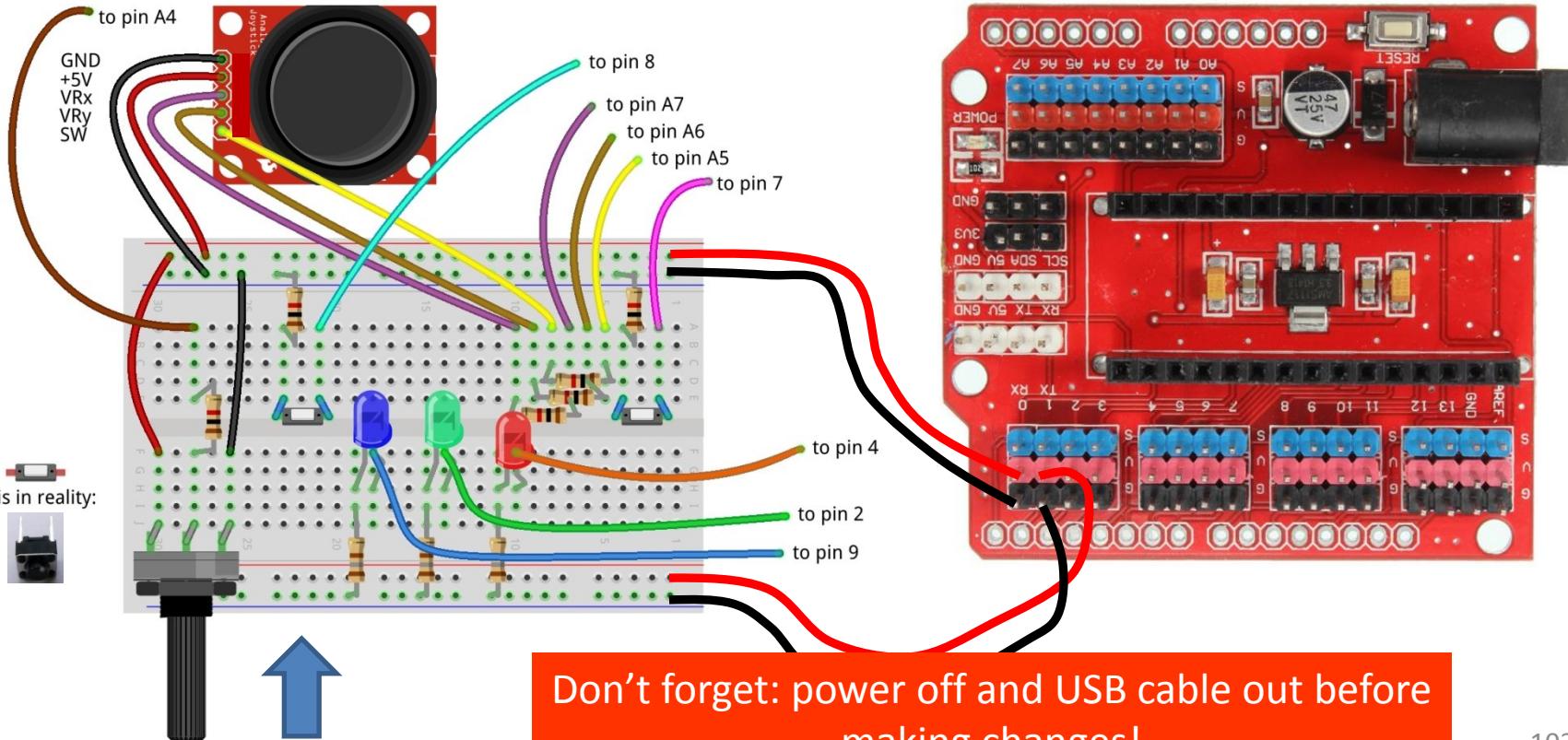
# Counter function using a button

- It will count up. It will wrap round at a very large value that will be reached after days of counting.
- Max. ca. 80 counts per second = 4800 per minute
- There is a 4 ms debouncing delay
  - So multiple pulses within 4 ms will processed as a single count
  - So no need for capacitors to do debouncing

# Debouncing

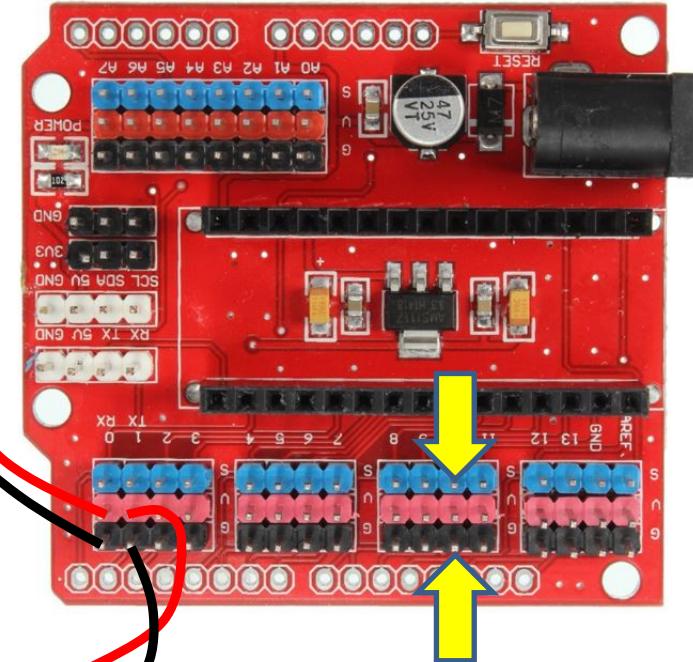
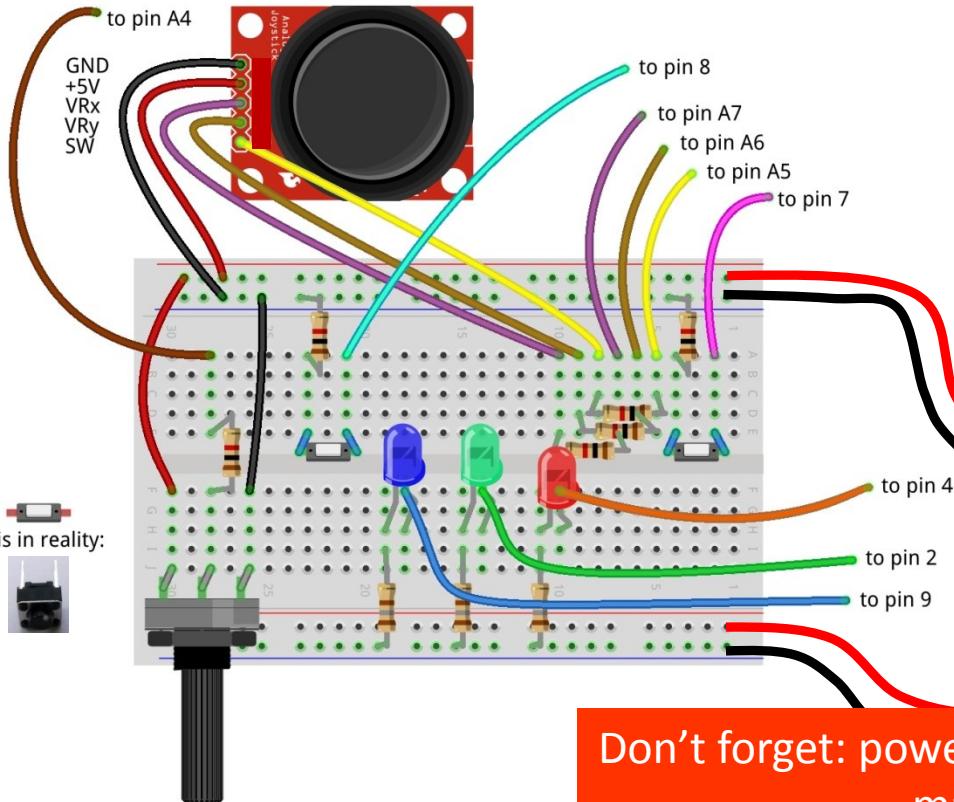
- If a mechanical switch is operated, it can generate a series of pulses rather than one.
- This is because one of the metal pieces bounces back a few times before it stays in position.
- You would normally need take care of this if you want to get a single pulse
  - E.g. add a capacitor.
- scratchClient will on the digital input pins only sample with 10 Hz, so it is unlikely that you will experience bounce problems.

# Adding the count button



# Adding an IR speed sensor that counts on pin 10

Carefully follow the wires to connect correctly!



Don't forget: power off and USB cable out before making changes!

# Differences between the counters

- The IR sensor gives an output of 0 Volt or 5 Volt.
  - Therefore it does not need a pull up resistor (see the config file)
- The button gives an output of 0 Volt or *open*.
  - Hence it needs a pull up resistor (as specified in the config file)

# Try out the setup

- The button you can just press.
- The IR sensor you need to have good opaque material to let it work.
  - You can check the sensing by the LED at the backside.

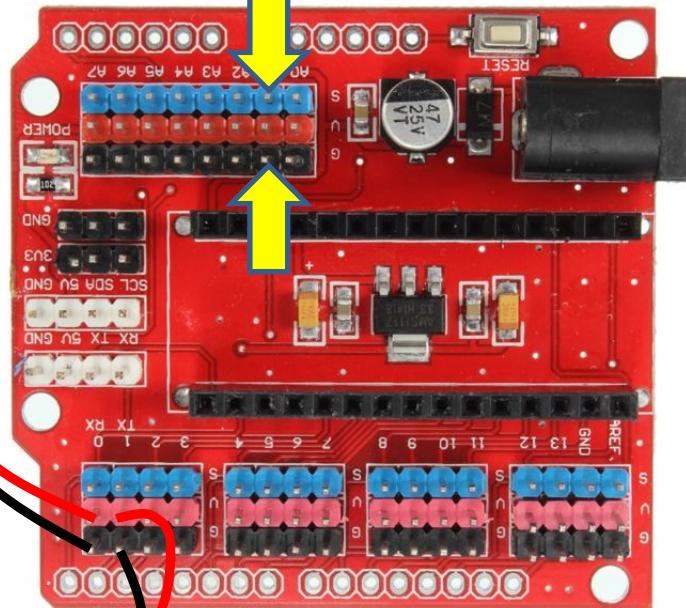
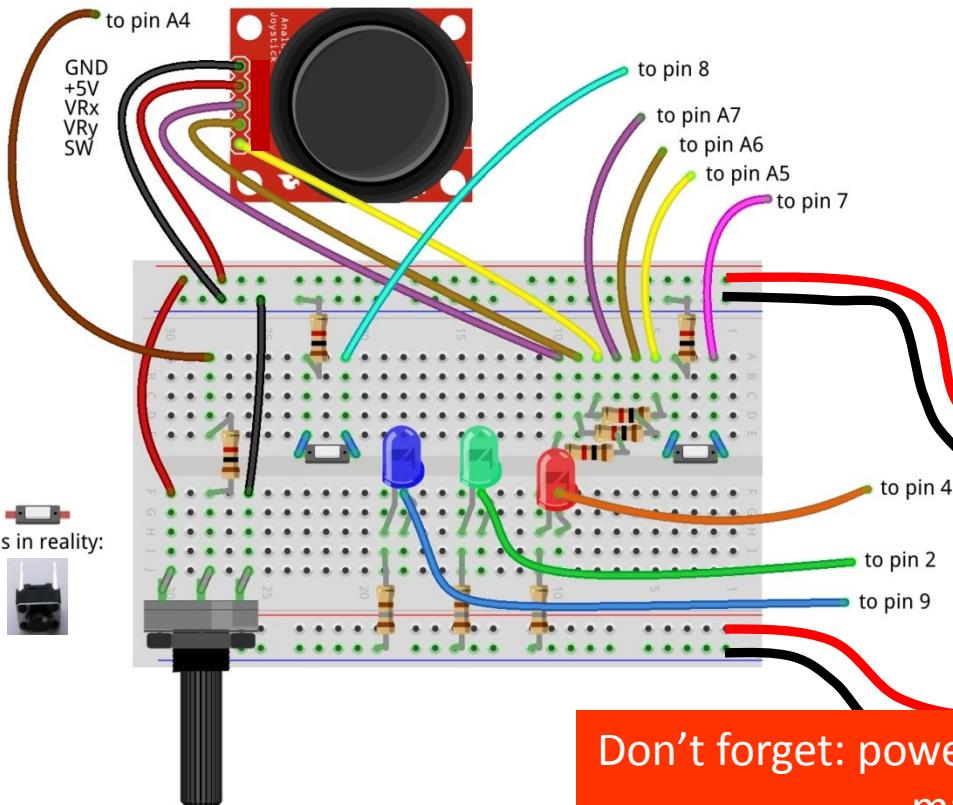
# Tilt sensor

# Tilt sensor

- The tilt sensor is a sensor containing a rolling ball between contacts.
  - You can hear it rattling if you shake it.
- It is not in the box. We only have a few, so you will need to share.

# Connect the tilt sensor to pin A1

Carefully follow the wires to connect correctly!



Don't forget: power off and USB cable out before making changes!

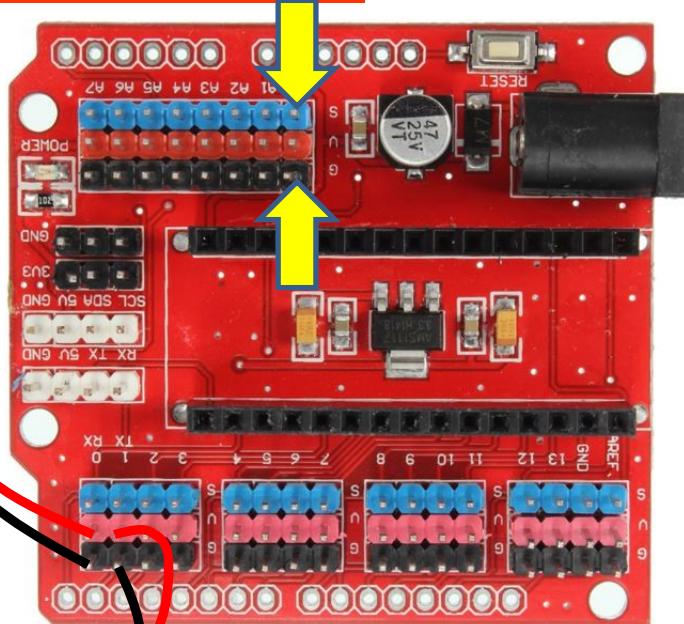
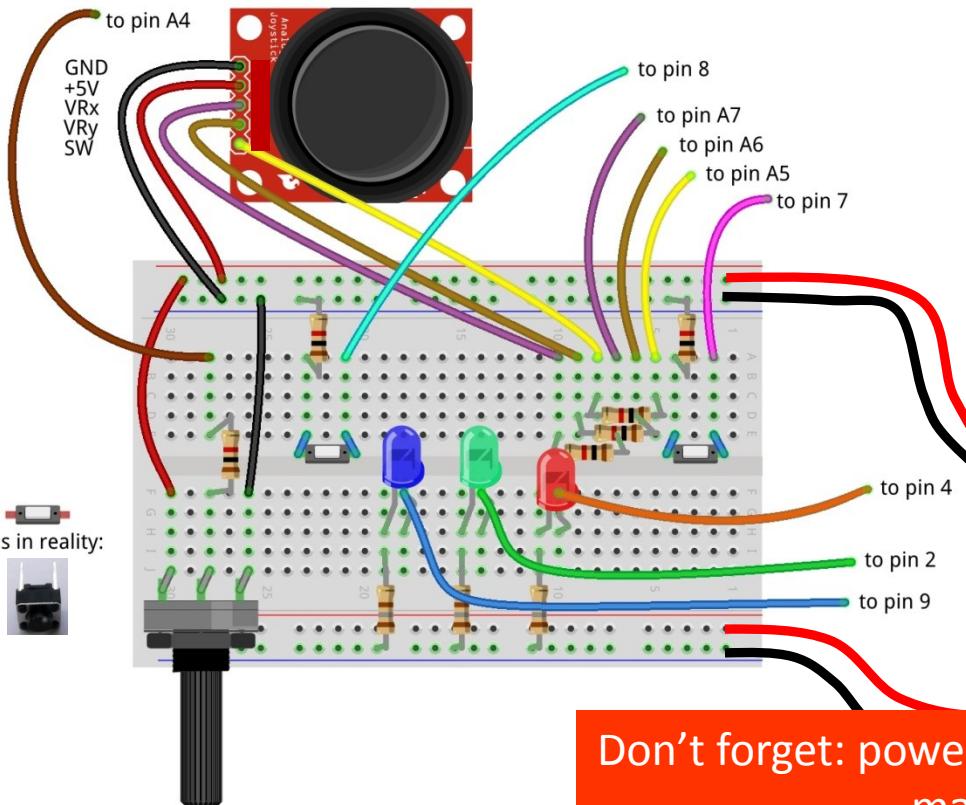
# Rain Sensor

# Rain sensor

- A rain sensor is an analog sensor that measures conductivity that is influenced by water.
- It is not in the box. We only have a few, so you will need to share.

# Connect the rain sensor to pin A0

Carefully follow the wires to connect correctly!



Don't forget: power off and USB cable out before making changes!

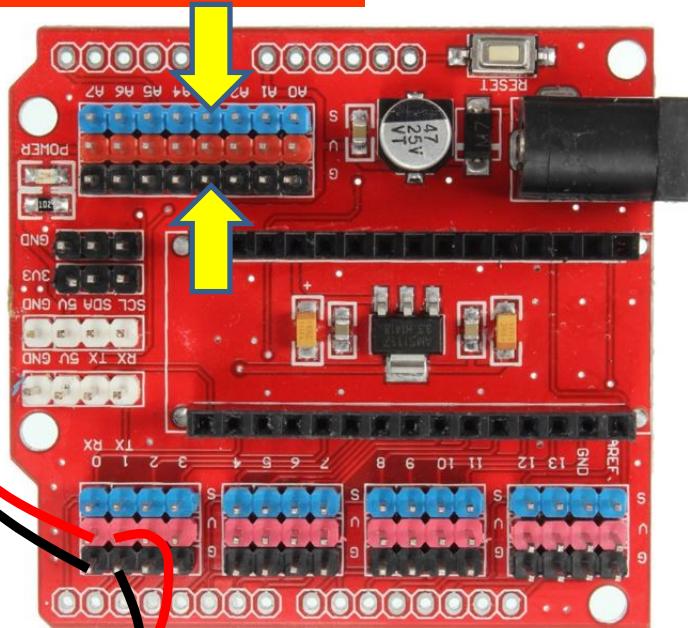
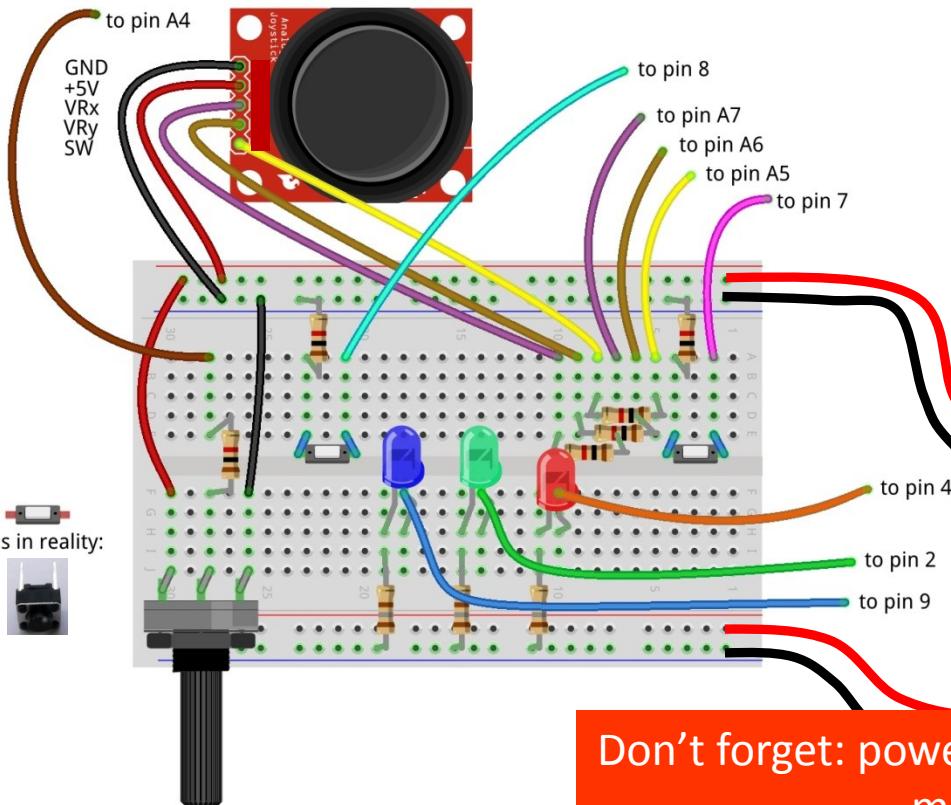
# Sound sensor

# Sound Sensor

- The sound sensor will detect sound.
  - It is a digital sensor that contains a microphone and will give 0 Volt if sound is detected, 5 Volt if no sound detected.
- It is not in the box. We only have a few, so you will need to share.

# Connect the sound sensor to pin A3

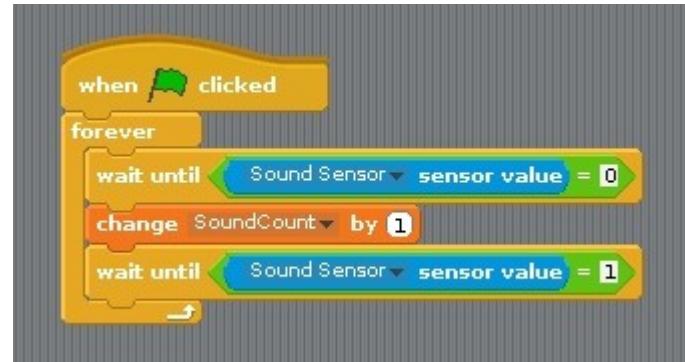
Carefully follow the wires to connect correctly!



Don't forget: power off and USB cable out before making changes!

# See the short pulses

- The changes of the sound sensor are sometimes difficult to see just from the displayed sensor value
- This code can be used to see that the sensor is actually working. You must create a variable SoundCount (or choose a different name) to hold the count.



Strongly link the config file to the  
Arduino

# The ident functionality

- You created a board, but maybe you have omitted some resistors and therefore damage would occur if it were used with the wrong config file.
- That is what the ident functionality is for
  - Configure an identity in the Arduino
  - Specify the identity in the config file

# Programming the ident in the Arduino

- Have the Arduino connected
- Start the Arduino IDE
- Go to monitor mode
- Set the speed to *115200*
- Set the line end to *newline*
- You can now type commands in the window.
- Type:
  - *cident:PiAndMore10*
  - *cident?* (you should get the string back)
- Put in the config tool the value *PiAndMore10* in the field *Parameter ident.pattern*

# Testing ident functionality

- Start scratchClient and see that it works
- Now change the value of the ident in the config file.
  - Hardly matters what
- Start scratchClient again and see that it will be rejected

# More about ident

- The ident can in the config file be specified as a regular expression.  
E.g.:
  - A dot (.) matches any character
  - A star (\*) repeats the previous character (0 or more times)
  - If you want to know more about regular expressions, see e.g.  
<http://www.rexegg.com/regex-quickstart.html>
- So you could specify
  - PiAndMore.\* and it would work when PiAndMore9, PiAndMore10 or PiAndMore100 is programmed in the ident of the Arduino
- Ident is also needed to identify a specific Arduino if multiple will be connected concurrently (see next slides).

# Controlling multiple Arduinos

# Control multiple Arduinos

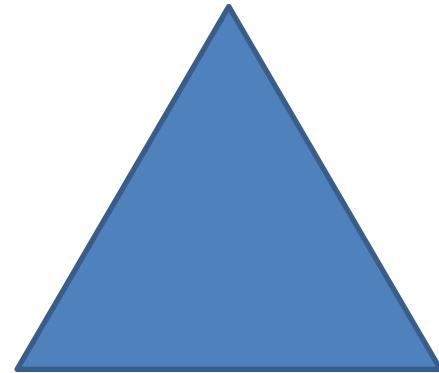
- Running out of pins on one Arduino? Connect more of them!
- The /dev/ttyUSBnn port in the config file will define which part of the configuration is used for which Arduino.
- Using the ident functionality (see previous topic) can be used to check that the correct Arduino was connected to the intended port.
  - You can get the Arduinos on the correct ports by connecting them always in the same sequence.

# Creating a config file for multiple Arduinos

- First create two separate config files for each Arduino and test them out.
  - The config tool can only work on a single adapter at a time.
- Now create a new .xml file, and
  - Copy in the first file completely
  - Take the adapter section (most of the file) of the second file and paste this after the first adapter
  - Adapt the USB port in one of the files (make the board that you connect last /dev/ttyUSB1).

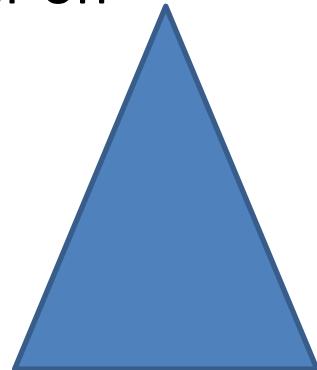
# Trying it out – merge and config

- Take the Weekendschool board with the Duck
- Test it out with the config file and the Scratch program
- Merge the two files in the way described.
  - Make the Weekendschool board /dev/ttyUSB**1**.
- Disconnect both boards.
- Restart scratchClient with the merged file.
- Connect the workshop board first. It will become /dev/ttyUSB**0**.
- Connect the Weekendschool board second. It will become /dev/ttyUSB**1**.



# Trying it out – Control from Scratch

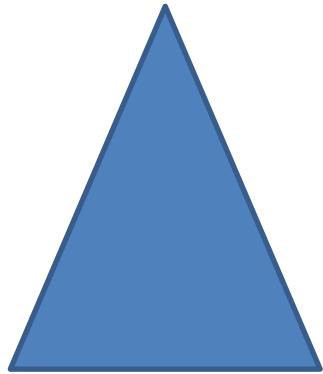
- In the Scratch program of the workshop, define these variables
  - Tilt servo
  - Pan servo
- Try giving them values between 0 and 100 with the slider on the variable.
- Does the duck tilt and pan?



# Making the /dev/ttyUSBnn not matter anymore

- In the way described before, the Arduinos must be connected in a specific sequence
- You can alternatively use a program that will
  - Look at the connected boards (using the ident string)
  - Give a selection option to start with config files for which all the boards are present.

# Explaining the startup tool

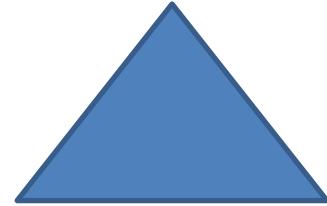


# Power on self test (POST)

# Power on self test (POST)

- See the Weekendschool board with the duck.
- When setting up the lesson, one wants to be able to quickly see whether the board works correctly.
- Also, when the boards have to be packed, the duck should bow deep to make sure it fits in the box.
  - Moving servos by hand is not preferred.
- The scratchClient Arduino sketch has a place where you can insert your own code to realise this functionality.

# How to do the POST?

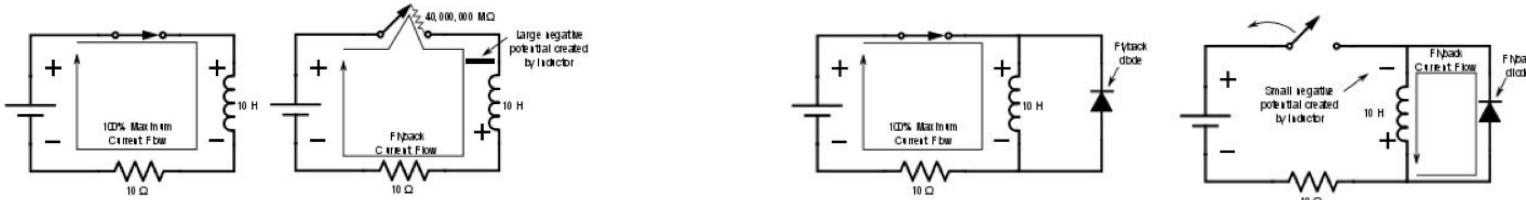


- You will need to address the pin numbers directly. You **cannot** make use of the config file of scratchClient.
- See the Arduino sketch of the Weekendschool board.
  - Between the xxxx and yyyy markers
- Now try on the PiAndMore board to lite 3-color LED
  - 0.5 seconds Red
  - 0.5 seconds Green
  - 0.5 seconds Blue
  - 0.5 seconds White
- Load the sketch in the board and look whether it works.

# Controlling a relay

# Controlling a relay

- Controlling a relay like controlling a LED? NOT a good idea!
- A relay has a coil that stores energy when powered.
  - When breaking the current flow, the voltage over the relay coil can get so high that it destroys the Arduino – and more.



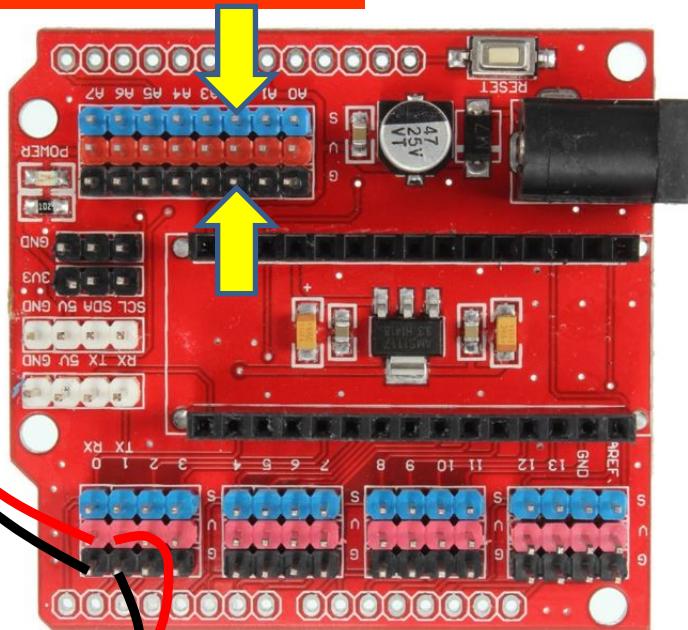
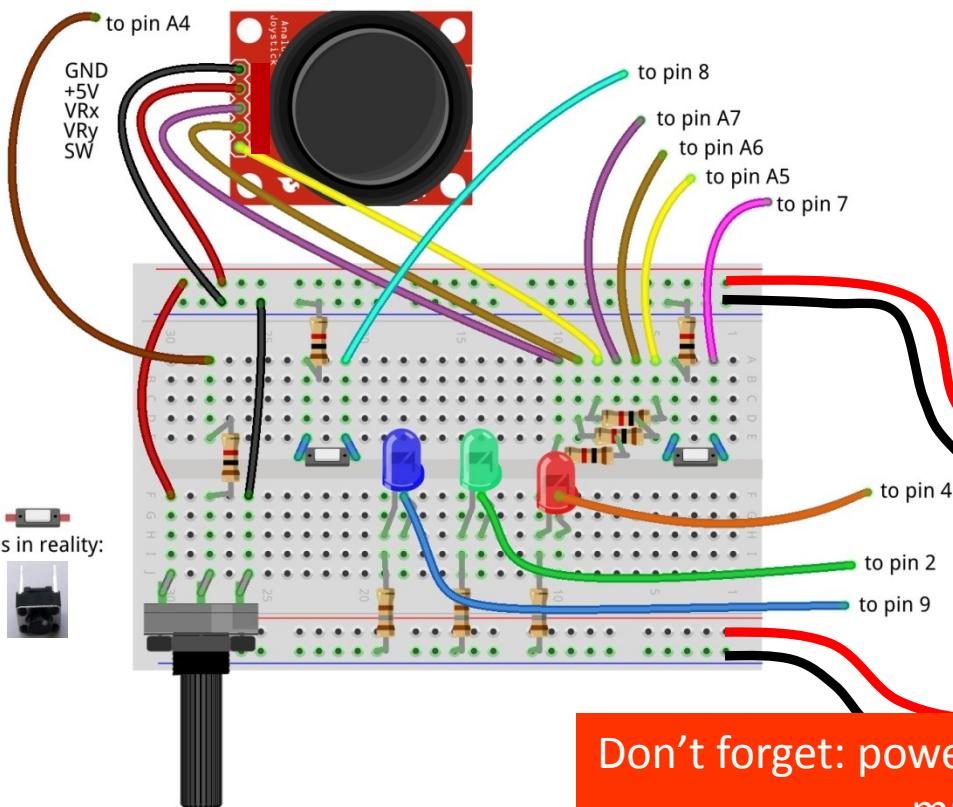
- At least a flyback diode should be used.

# Other issues – and the solution

- Often the current of the coil will be too high to drive directly from the Arduino
  - So a transistor will be needed.
- Solution
  - Use a relay module with an opto coupler
  - Cost is generally not an issue. The relay module we use here is < € 1.
  - We only have a few relays today, so you will need to share.

**Connect the relay module to pin A2**

**Carefully follow the wires to connect correctly!**



Don't forget: power off and USB cable out before  
making changes!

Use a servo to press buttons on  
appliances

# Safely controlling 220 Volt appliances

- You can in principle use a relay to control 220 Volt appliances.
- However, this is not something you want to do in a classroom situation
- Alternative: use a remote control and use servos to press the buttons.
- See also the next slide.

# Controlling a camera

- Use a servo to press the button of the camera
- Cannot do in the workshop
- See the example setup in the workshop

# What else can scratchClient do?

# Can scratchClient do more? Yes!

- It can also let you configure and use the GPIO pins on the Raspberry Pi itself.
  - However be aware:
    - There is no analog input on Raspberry Pi.
    - The pins are **NOT** 5 Volt tolerant (max 3.3 Volt).
    - The max current per pin on Raspberry Pi is lower than on Arduino.
    - If you blow up an Arduino (clone), the cost is limited to a few euro.
      - A Raspberry Pi is much more expensive.
- scratchClient can also control Sonic Pi for high quality audio.
- There are also Arduino sketches to control LED strips (Neopixel WS2812) and to control LEGO Power.
- scratchClient can work with Scratch 2.0 (soon).
- scratchClient can also run on Windows and then via Arduino control peripherals.
- scratchClient can work with multiple Raspberry Pi configurations in a network.

Take your work home

# Do you want to take your work home?

- If you brought your own USB stick, then connect it and copy the *PiAndMore* folder on the desktop
- The rest of the material you can download from  
[www.github.com](http://www.github.com)
- Take the flyer with you to remember where to find the material on github.

# Clean up / teardown

# Unplug and pack up

- Unplug the board from USB and power it off
- Please remove all components and wires from the **breadboard**
- Remove all wires from the **Arduino board**.
- ***Leave the wires on the 3 color LED***
- ***Leave the wires on the buzzer***
- If something is broken, please
  - Throw it away or hand it in (if it is unclear)
  - Put a note in the box that it is missing
  - Do not put anything that is broken back in the box
- Put everything in the box. The board goes on top
- Shutdown the Arduino
- Let us know what you thought about this workshop, now orally or later by email
  - [hans.piam@hanselma.nl](mailto:hans.piam@hanselma.nl)
  - [heppg@web.de](mailto:heppg@web.de)

# More information

- All workshop material
  - [www.github.com](https://www.github.com) and search for *Weekendschool* or for *PiAndMore*
- scratchClient
  - [http://heppg.de/ikg/wordpress/?page\\_id=6](http://heppg.de/ikg/wordpress/?page_id=6)
- Scratch
  - <https://scratch.mit.edu/>
- Scratch on Raspberry Pi
  - <https://www.raspberrypi.org/forums/viewforum.php?f=77>
- Raspberry Pi
  - <https://www.raspberrypi.org/>
- Arduino
  - <https://www.arduino.cc/>

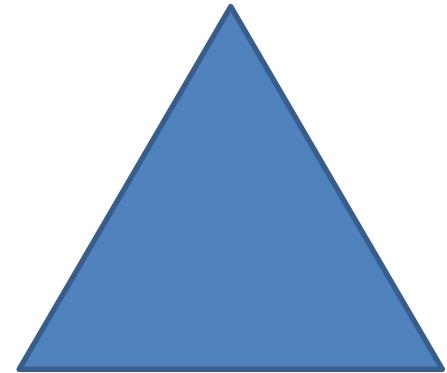
# End of the Advanced Workshop

# Appendix

# Maximum current per pin

- Max output current per pin = 20 mA
- Total current for the board: 1 A
- Note that also the potentiometer and the joystick take power

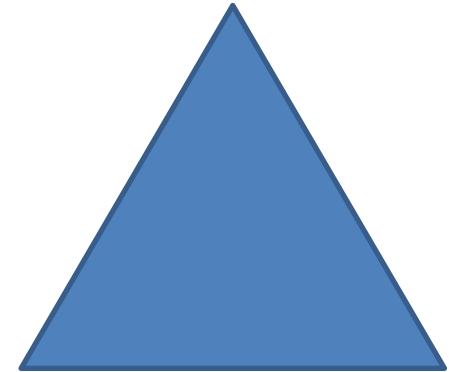
# Calculation of the resistors



# Cost of the setup (1 of 2)

- This is a list of the cost of the setup of the external board.
- Not included is the cost of the perspex board and the nuts and bolts, which are not an absolute minimum requirement
  - And for which there are alternatives, e.g. use cardboard or an MDF board
- All material comes from Aliexpress
  - Very cheap
  - Often no shipping costs to Europe
  - Takes between 2 weeks and 2 months
    - Or never arrives, but then money is promptly returned.
  - When ordering below 22 euro, no import duties and handling costs

# Cost of the setup (2 of 2)



# The End