

# Physical computing from Scratch using scratchClient – **Beginners**

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

Hans de Jong & Gerhard Hepp

Pi And More 10 1/2

Stuttgart – 24 February 2018

# Part 1: Introduction

# When you look at this at home ...

- If the presentation is messed up ...
  - You need the Calibri font on your Raspberry Pi (standard on Windows but not on Raspberry Pi).
  - Hence look at the .pdf version which does not need the font to be installed.

# Workshop organisation

- Welcome and introduction presentation (5 min).
  - After that everyone will work at his or her own pace.
  - Choose with your “workplace partner” which topics you want to do.
    - There is more material than what can be covered in 110 minutes.
  - Language: Both German and English
  - At the end copy the material you created to your USB stick (if you want)
  - Break down & clean up (5 min)
- The major steps:
    1. Get a working hardware and scratchClient config using a scratchClient config tool.
    2. Put components on the board and test out your setup.
    3. Write some code in Scratch
    4. Add more hardware and update the config file.
    5. And repeat this.

# Objectives

- At the end of today you should be able to do a few of these items:
  - Reproduce the setup at home (provided you have the hardware 😊 )
    - You can get all files via <https://github.com/hansdejongehv/scratchClient-Tutorials>
    - Or go to [www.github.com](https://www.github.com) and search for *scratchClient*
  - Understand (depending on how far you get and how deep you dive in)
    - Digital output (e.g. lighting a LED)
    - Digital input (e.g. sensing a button)
    - Analog input (e.g. from a potentiometer)
    - Pulse Width Modulation (PWM)
      - For dimming LEDs
      - For controlling servos
      - For sounding a buzzer
  - Understand what all the resistors are for
  - Be able to configure and run scratchClient
  - Program Scratch to control the physical input and output
  - Monitoring the inputs and outputs
- **Have fun!**

# Non-objective

- It is ***not*** an objective to create a complete useful game or other program.
  - You can do that with your own creativity at home now that you know how to control several pieces of hardware from Scratch using scratchClient.

# Example of what can be created with Scratch and scratchClient

- <https://www.youtube.com/watch?v=Qo1gnXNzhqE>

# Versions of Scratch

- scratchClient can work with
  - Scratch 1.4
  - The (for Raspberry Pi) new Scratch 2
- This workshop is written for Scratch 2
  - Scratch 2 on Raspberry Pi has some bugs, but we will work around it.
  - If you want to see how to do it for Scratch 1.4, see the end of the presentation in Appendix A.

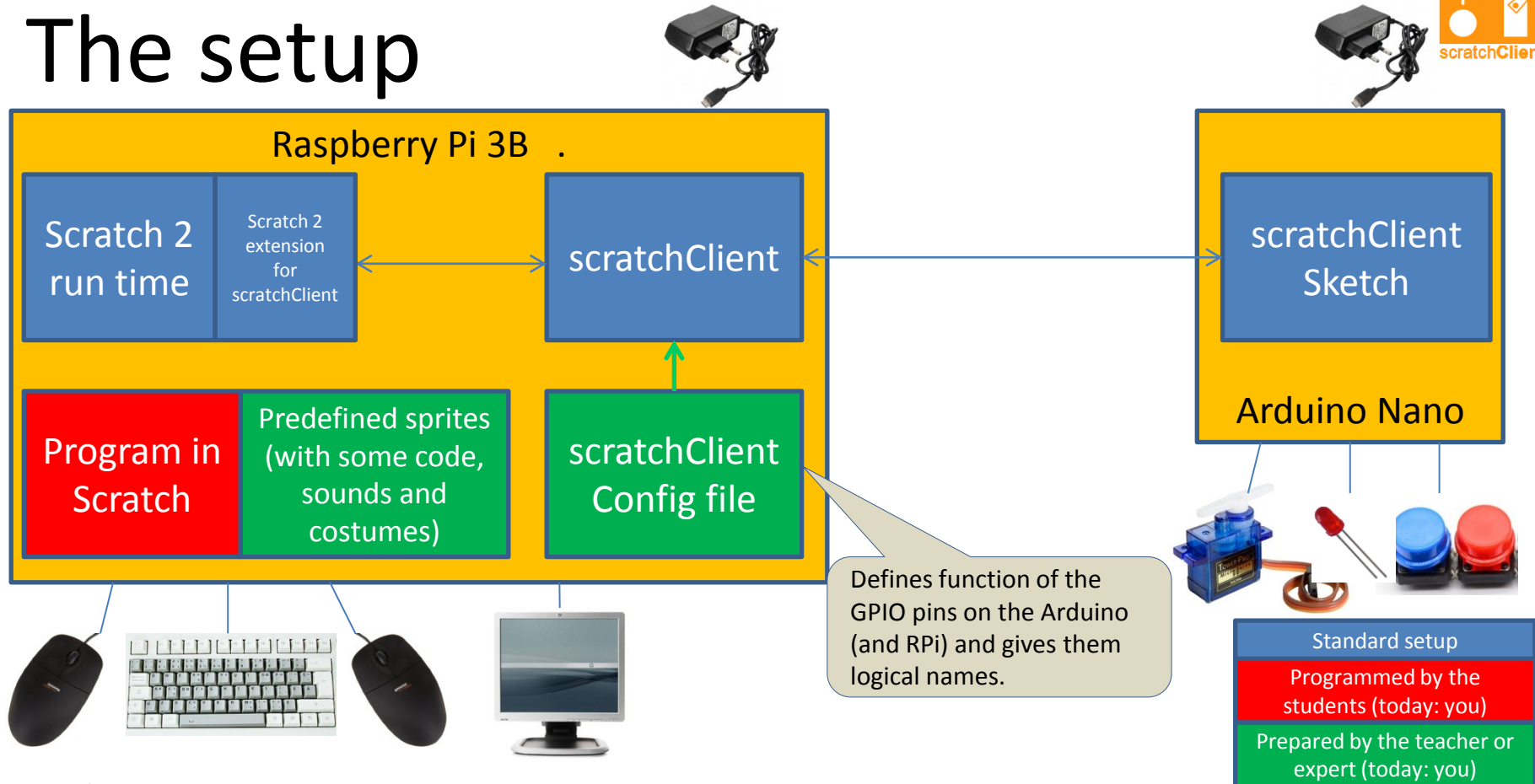


# Only a few rules today

- Always put a resistor in series with the components when indicated.
  - If you think there is no need then please tell us and we will explain what the reason is.
- When changing the wiring
  1. Detach the USB cable from the Arduino Nano
  2. Switch off the 9V power
  3. Check, double check and check again whether the wiring is correct. You may blow up components when wiring wrongly!
  4. Make sure that you **both** (4 eyes principle) are convinced the wiring is OK before reconnecting and turning on power again.
  5. After changing a config file: restart scratchClient
- If something breaks down or gets damaged: we have some spare material
  - Do **not** put anything that is broken back into the box please.



# The setup



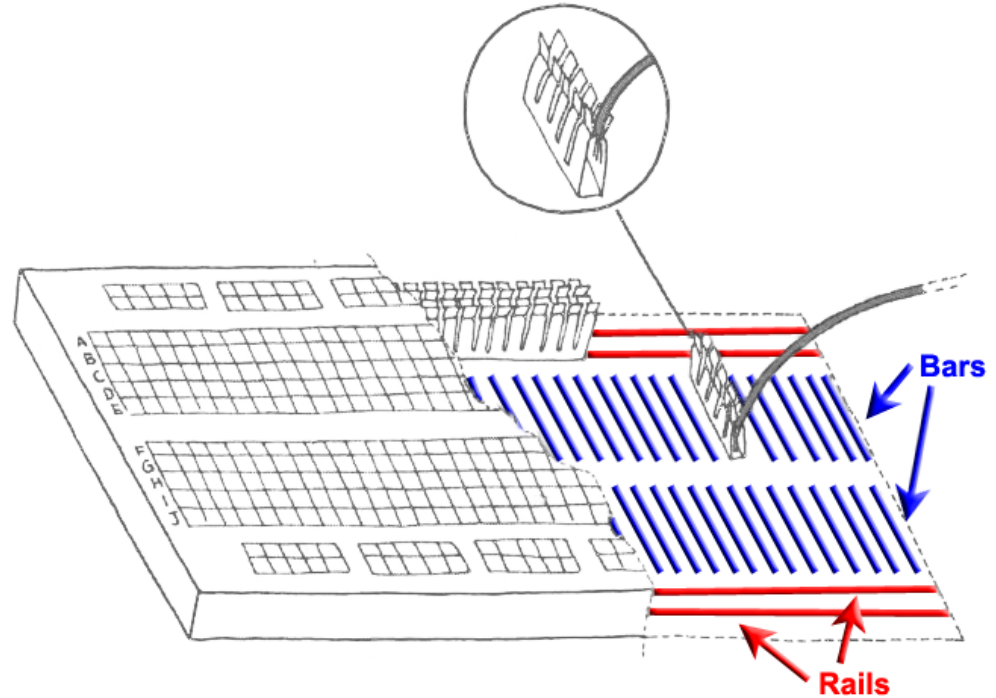
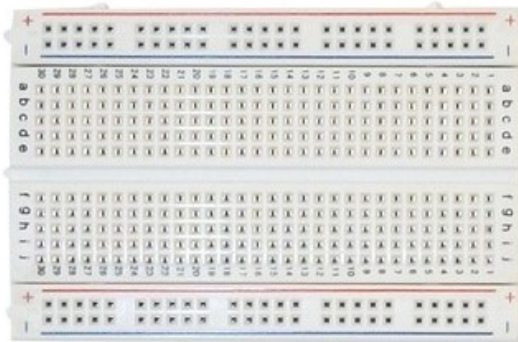
# Choose what you want to do

- Workshop is for everyone, from beginners to experts.
- Not enough time to do all, so choose what you want to do.
- **Yellow slides** have background information and you can skip them if you want or look at it later
- Recommended
  - All: learn to configure scratchClient by doing LEDs (digital out) and Button (digital in)
  - All: try it out in Scratch
  - After that: select further topics from separate files
    - Intermediate level
    - Advanced level
    - Expert level

## Part 2: Getting to know the components

# Breadboard

- Used to quickly build electronic circuits
- Note the 2 rails for + (VCC) and – (GND)
- Note on every column (vertical) the 2 bars with 5 interconnected holes each



# Looking at the Arduino Nano extension board

Analog ports (most can be used as digital ports as well)

Reset button

9V power socket

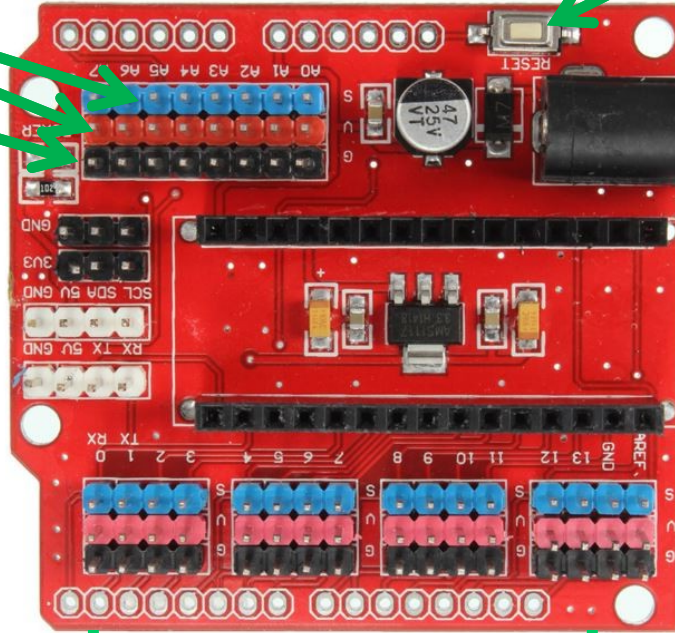
Per GPIO signal 3 headers:

S (blue = signal)

V (red = VCC = +)

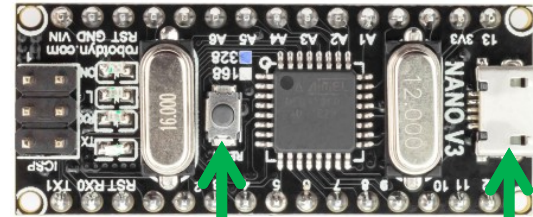
G (black = GND = -)

(very handy to e.g. connect servos)



Digital ports

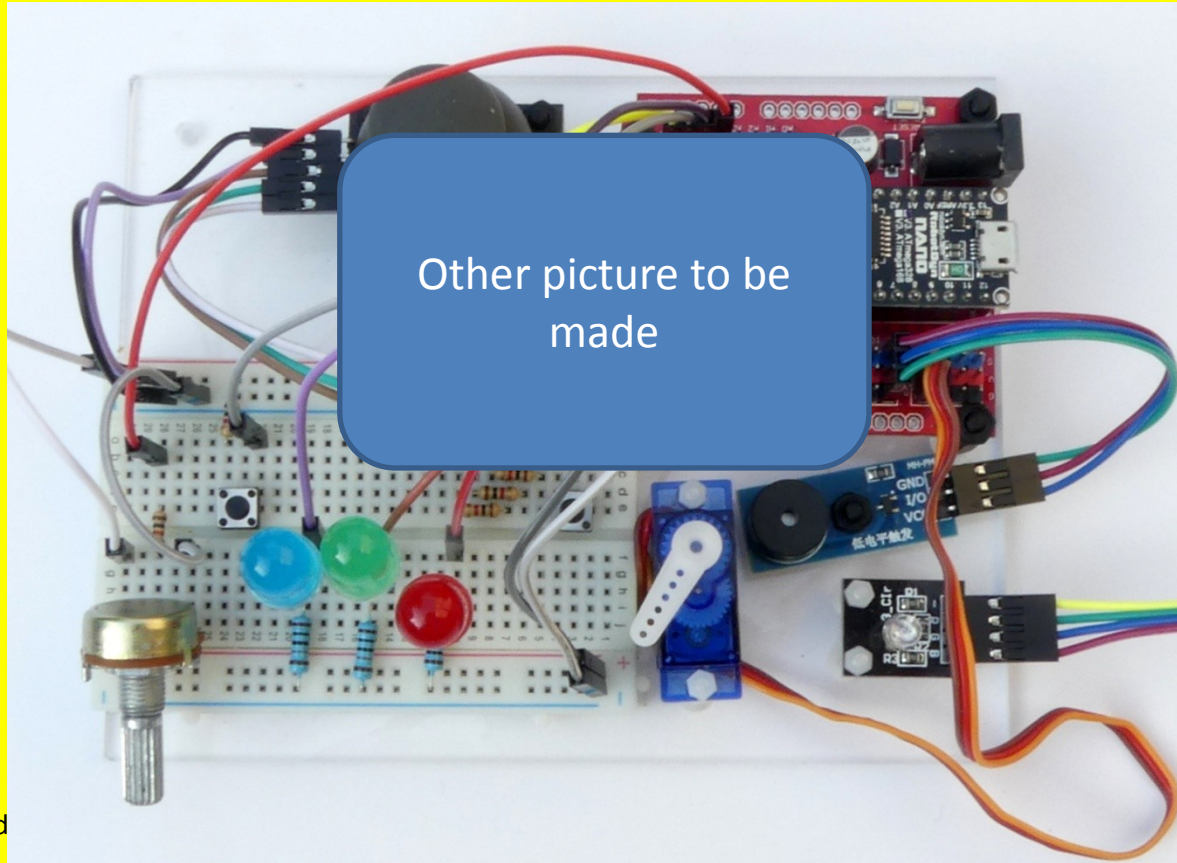
Arduino Nano with 328P processor



Micro USB port

Reset button

# What the final board looks like ...

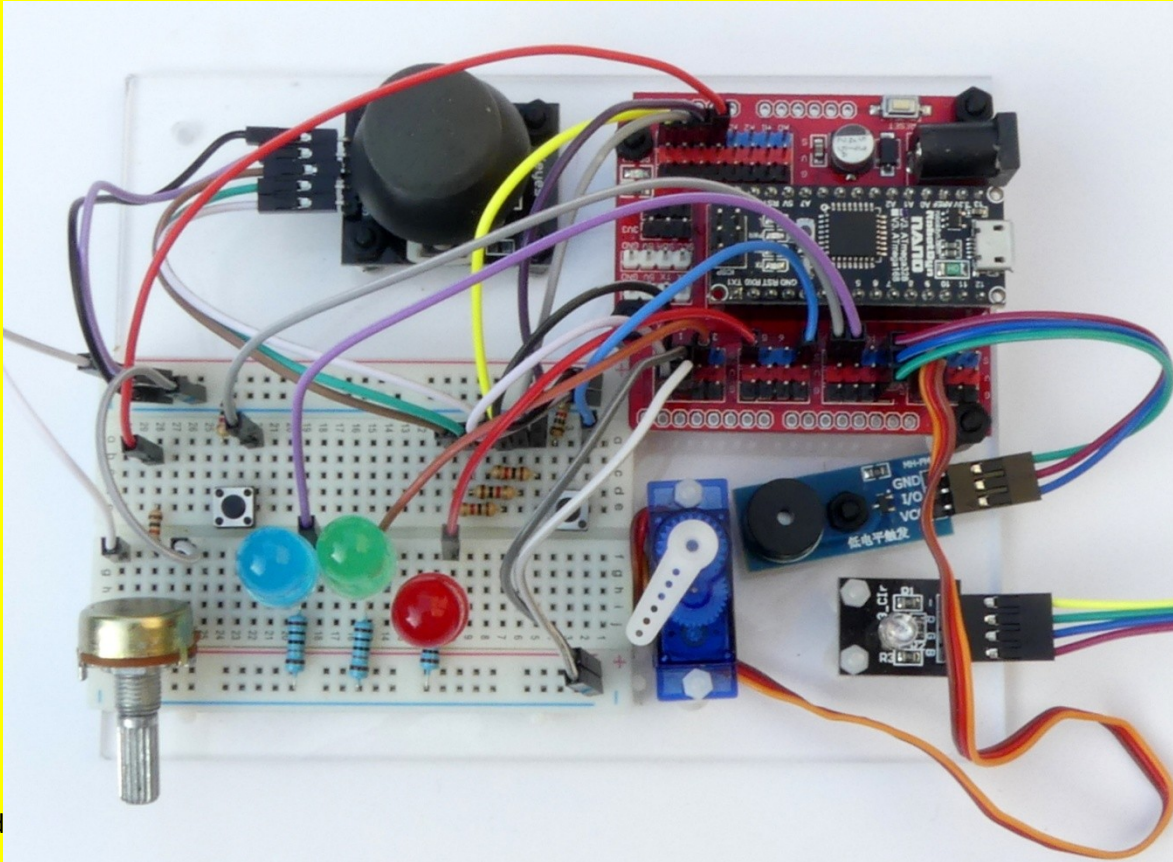


Other picture to be made

... at the end of the basic level

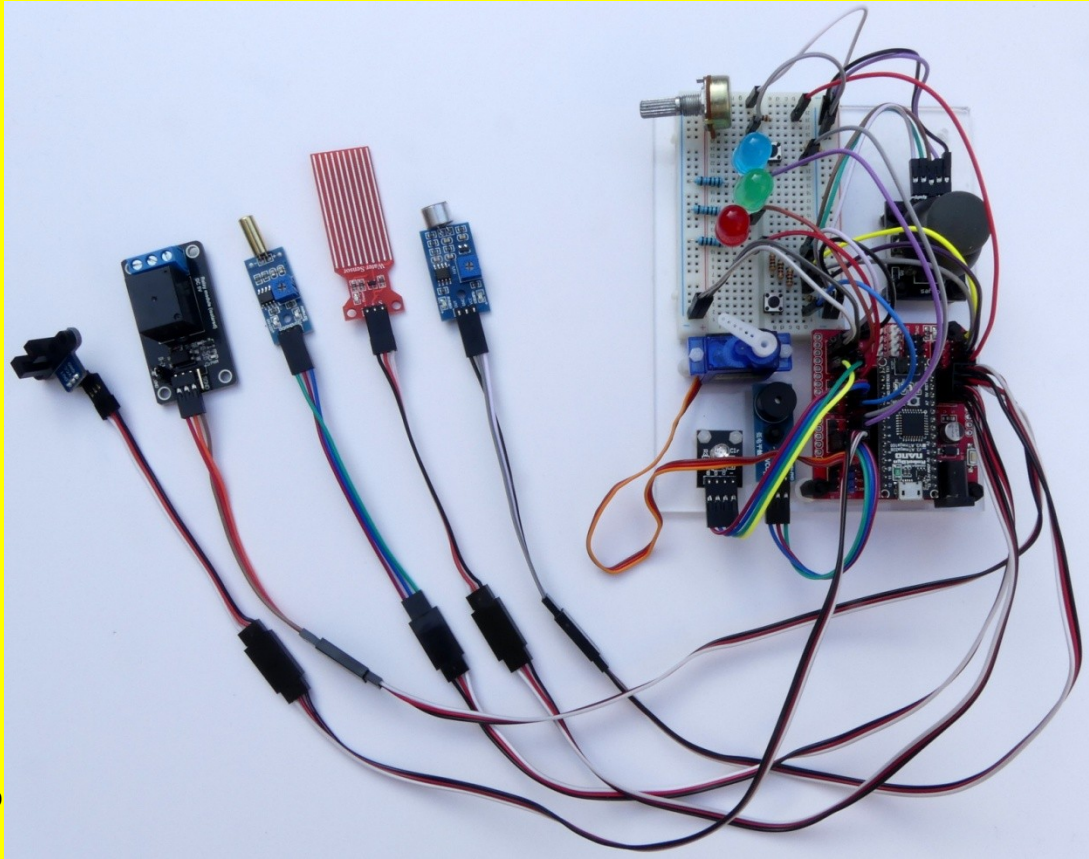


... at the end of the intermediate level ...





# ... and at the end of the advanced level



# Part 3: Loading the sketch in the Arduino (optional)

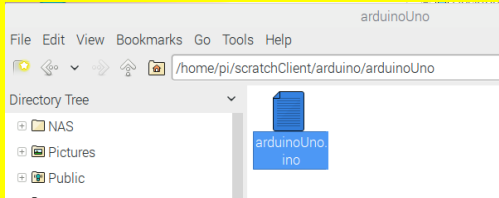
# Preparing for programming the Arduino Nano




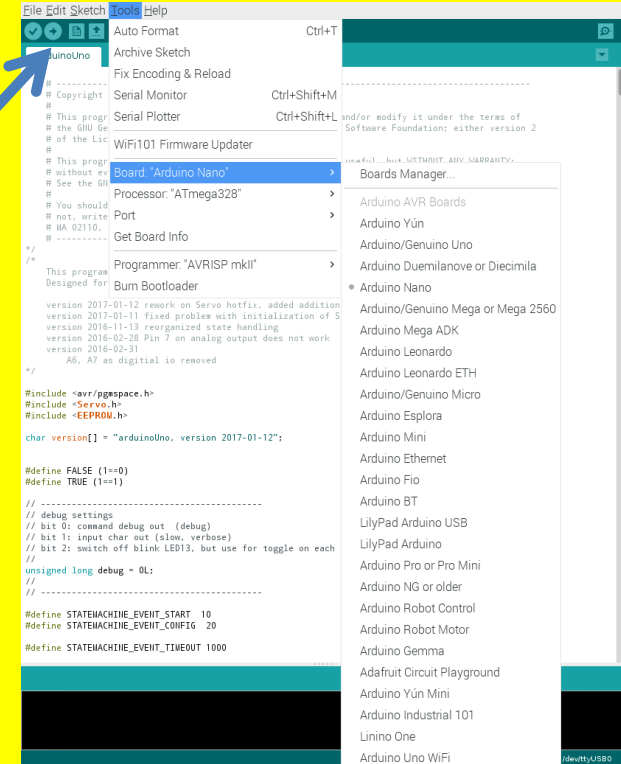
- The Arduino needs to run a program (in Arduino terms: sketch) so that it can communicate with the Raspberry Pi and understand the messages from scratchClient.
- We need to start putting that in.
  - However that has already been done, so you can skip the next slide, unless you want to try for yourself.

# Uploading scratchClient to the Arduino

- Navigate to the scratchClient sketch for Arduino Uno in `/home/pi/scratchClient/arduino/arduinoUno`



- Double click to open the Arduino IDE
- Click *Tools* and make sure that these are set:
  - Board: Arduino Nano
  - Processor Atmega328
  - Port: the port where the Arduino Nano is connected (normally `/dev/ttyUSB0`)
- Click on the Upload button. 
- Wait till the completion of the Upload is reported (without errors).

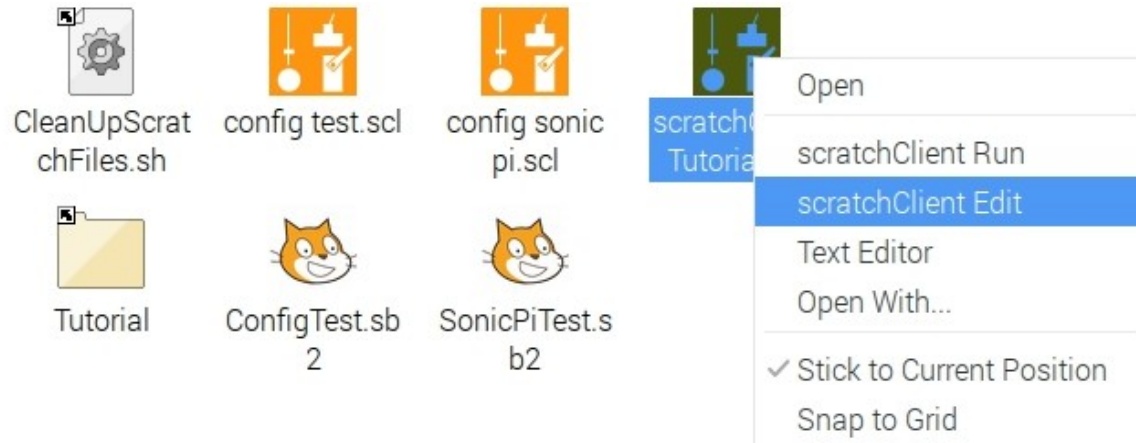


# Part 4: Defining the configuration

Give names to pins and define the purpose of the pin

# Starting the config tool

- We have put an empty config file on the desktop:
  - *scratchClient Tutorial.scl*
- Right click the file and choose *scratchClient Edit*



# Define the first config file

File Help

name: scratchCI Tutorial

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

D4  
Direction: out  
Function: output  
scratchName: BigRedLED

D7  
Direction: in  
Function: input\_pullup  
scratchName: Button

Parameter serial.device: /dev/ttyUSB0

Parameter serial.device: /dev/ttyUSB0

Parameter ident.check: ☒

Parameter ident.pattern:

```

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

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

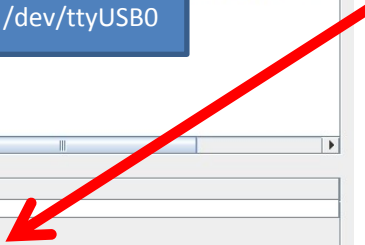
<parameter name="serial.device" value="/dev/ttyUSB0"/>
<parameter name="serial.baud" value="115200"/>

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

type id message

INFO empty ident.pattern connects only to arduino with empty ident

- **Double** click a cell to get a drop down menu
  - First for *direction* then for *function*.
- Make sure to give all pins a name if you choose something else than *void* in *direction*.
  - So make sure not to save if you still have red borders around cells. scratchClient will fail to start with such a config file.
- The tool checks wrong configurations. Examples:
  - Pins 0, 1 and 13 cannot be used at all
  - Analog In only available on A0 to A7
  - Pins A6 and A7 can only be used for Analog In
  - Pins 9 and 10 cannot be used for PWM if any pin is configured for servo (see later)



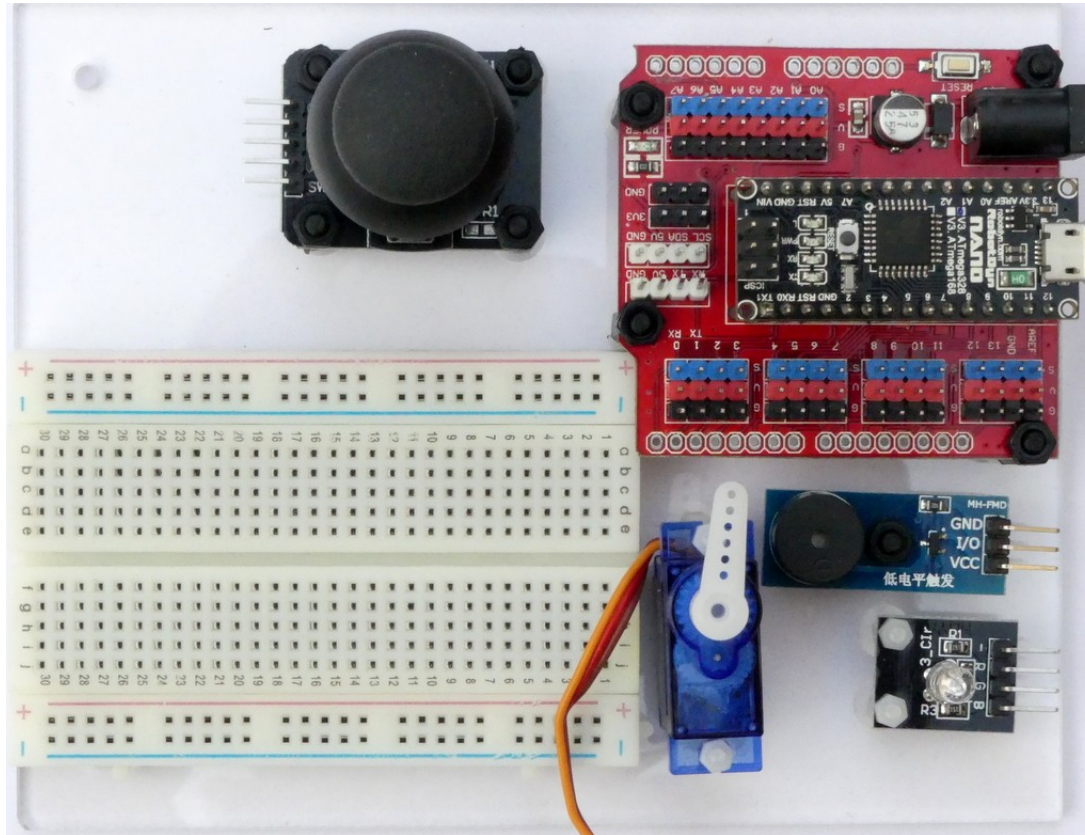
# Save the config file

- Save the file (ctrl-S or File → Save).
- **Warning:** closing the file without saving will loose all changes.
- **Leave the tool open** for the next exercises.



# Part 5: Wiring the board and run the first setup

# Put the board in front of you in this way

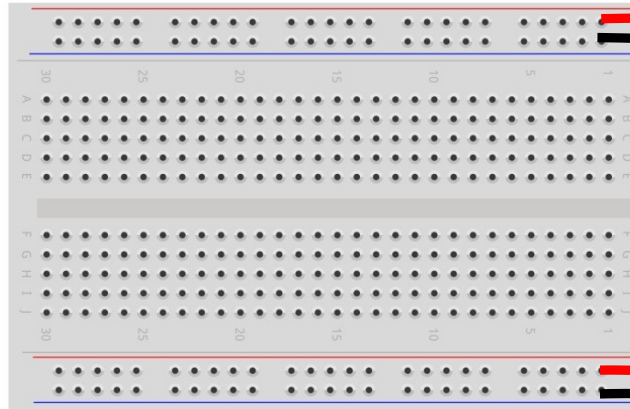
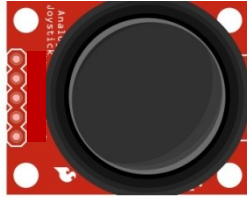


# Use *short* wires and use the indicated holes

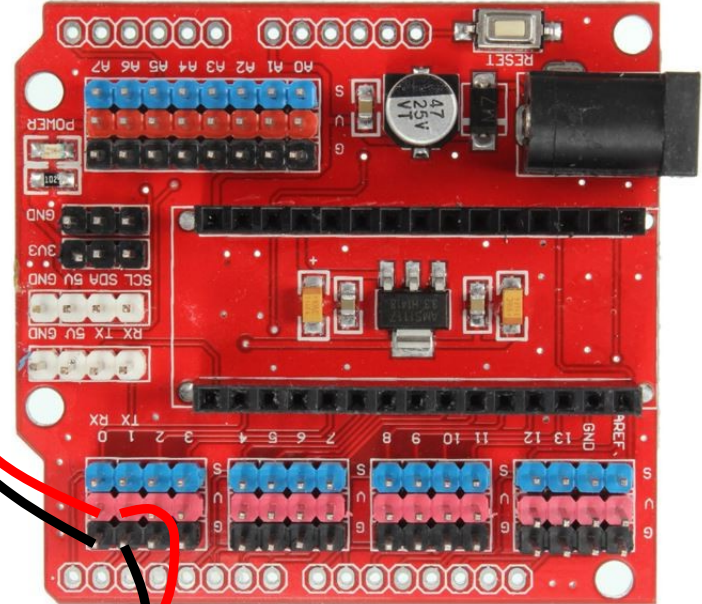
- There are some short wires (10 cm) and some longer (15 cm)
- Use the shortest that you can
  - Get a less messy setup
  - You may otherwise run out of long wires later
- Ignore wire colors.
- You can in principle build up at different places on the breadboard, however ...
  - ... please use the indicated columns to avoid running out of space on the breadboard in the later part of the workshop.

# Connect the power wires

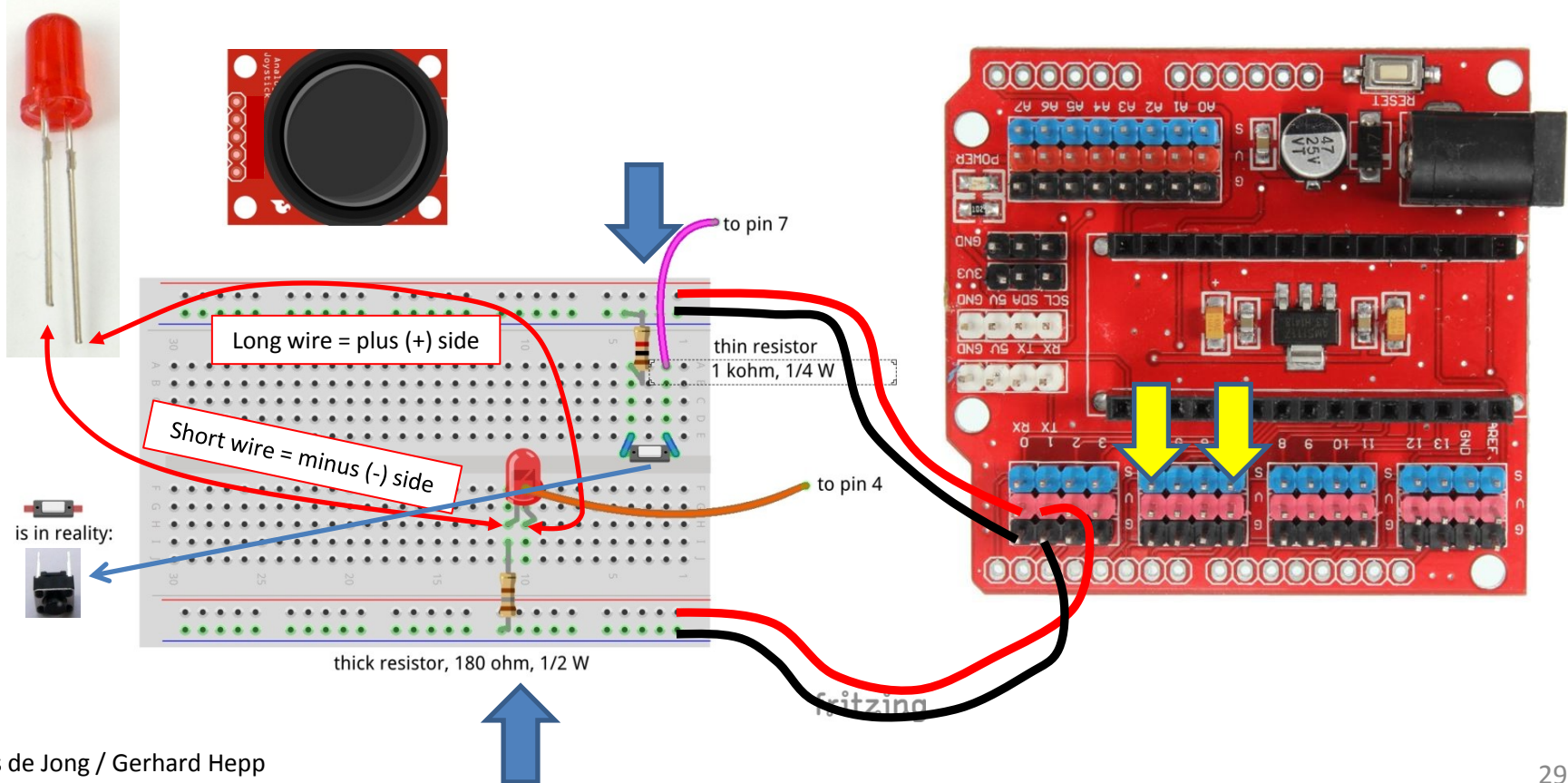
GND  
+5V  
VRx  
VRy  
SW



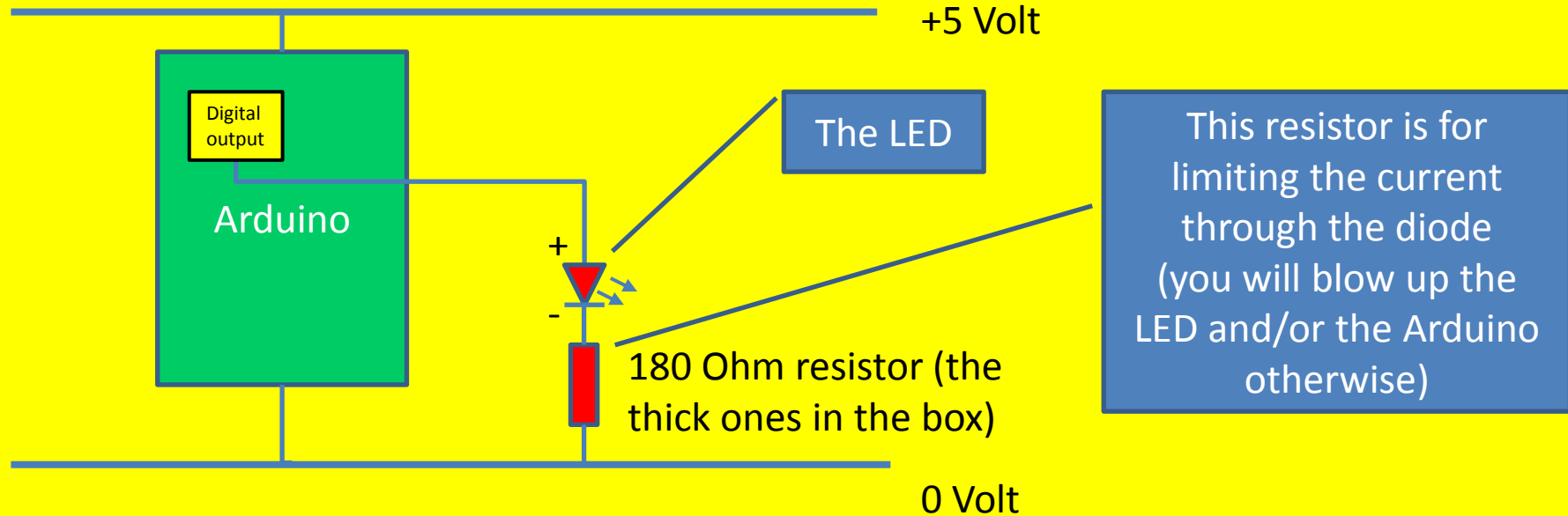
fritzing



# Insert the red LED and button



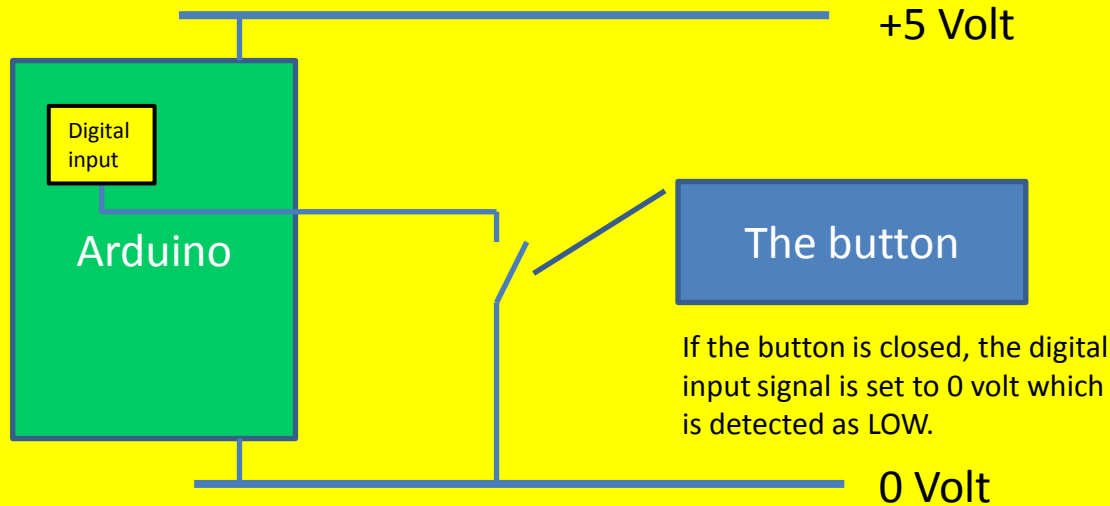
# Why putting a resistor in series with the LED?



# What does Arduino need on a Digital Input pin?

- A digital input pin of Arduino needs to get either
  - 0 volt input (actually, 0 to 1.5 volt is considered a LOW input signal)
  - 5 volt input (actually, 2.5 volt to 5 volt is considered a HIGH input signal)
- If it gets something between 1.5 volt and 2.5 volt, the interpretation will not be stable (could be LOW or HIGH).
- If it does not get any signal in, then the interpretation of the signal is not stable (could be LOW or HIGH).

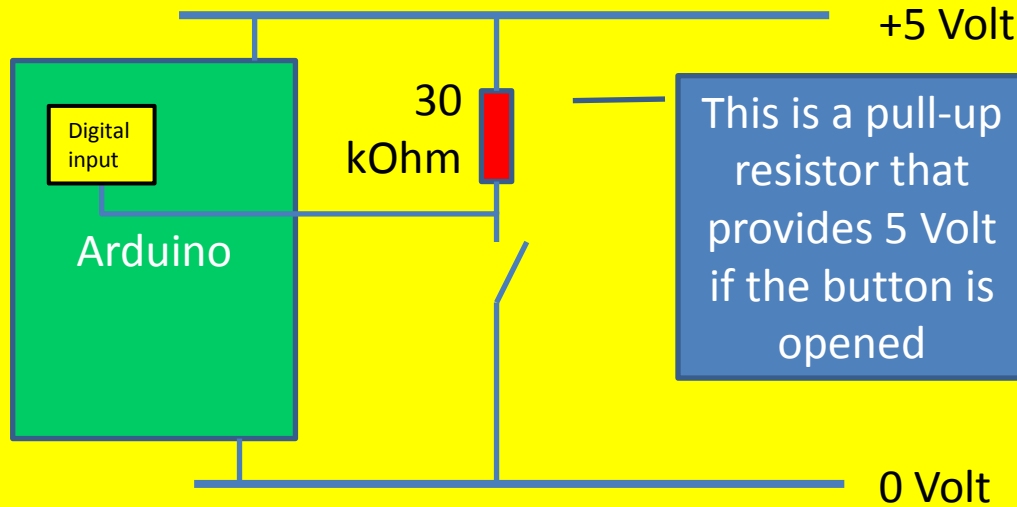
# Producing a LOW value



But what will the Arduino detect if the button is ***open***?



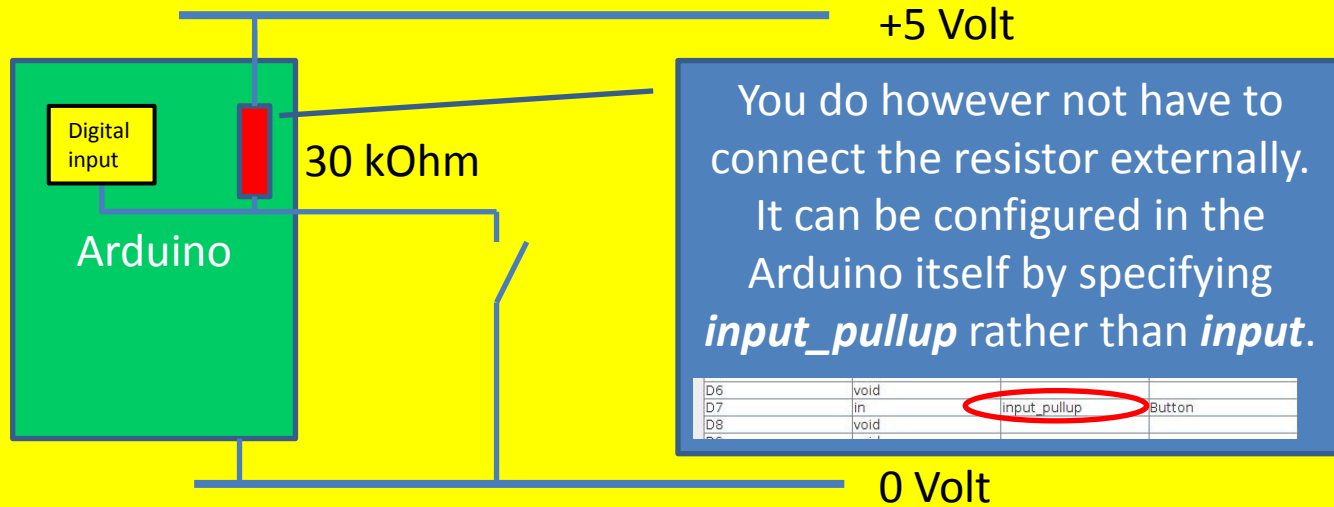
# Producing a HIGH value



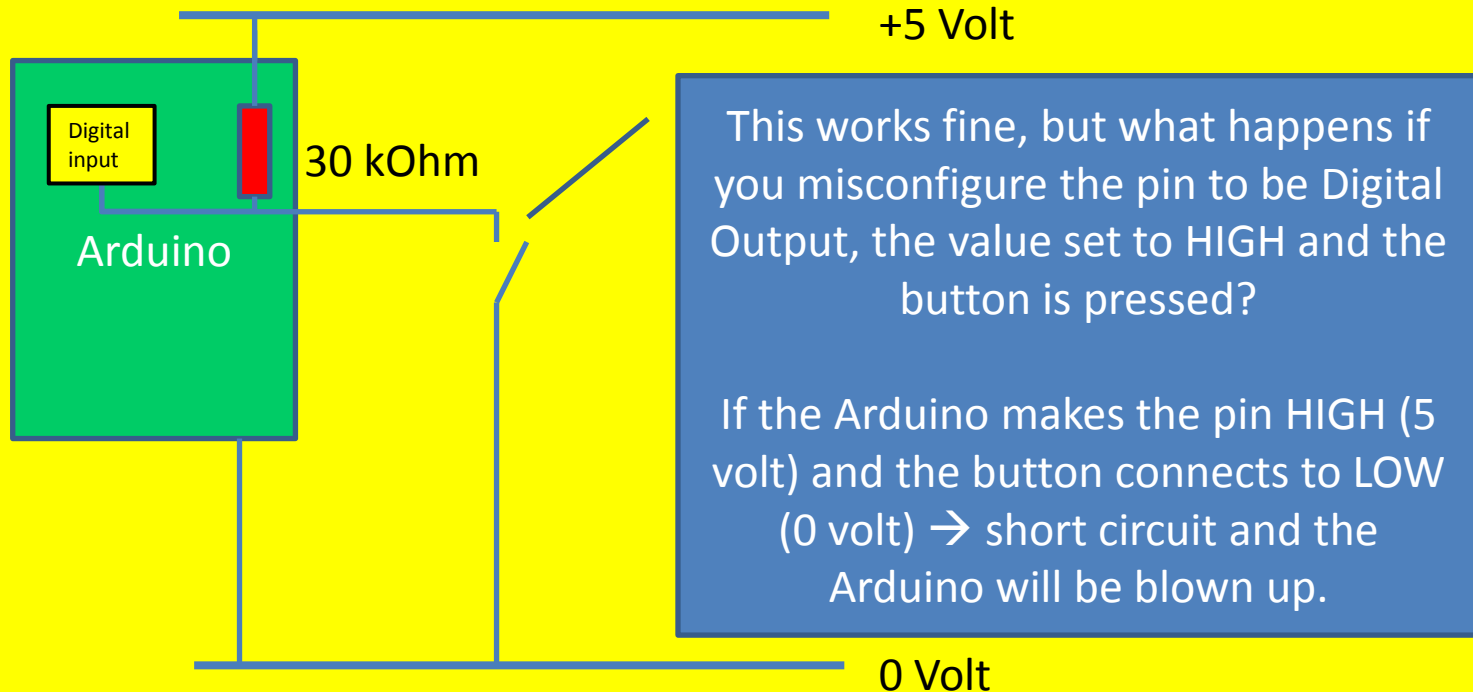
If the button is pressed, then it connects the input of the Arduino to 0 volt. There will run a small current through the resistor, but the input will be 0 volt.

If the button is open, the resistor will pull up the input to 5 volt. A very small current will flow through the resistor, but is small enough that the voltage on the digital input will be very close to 5 volt = HIGH.

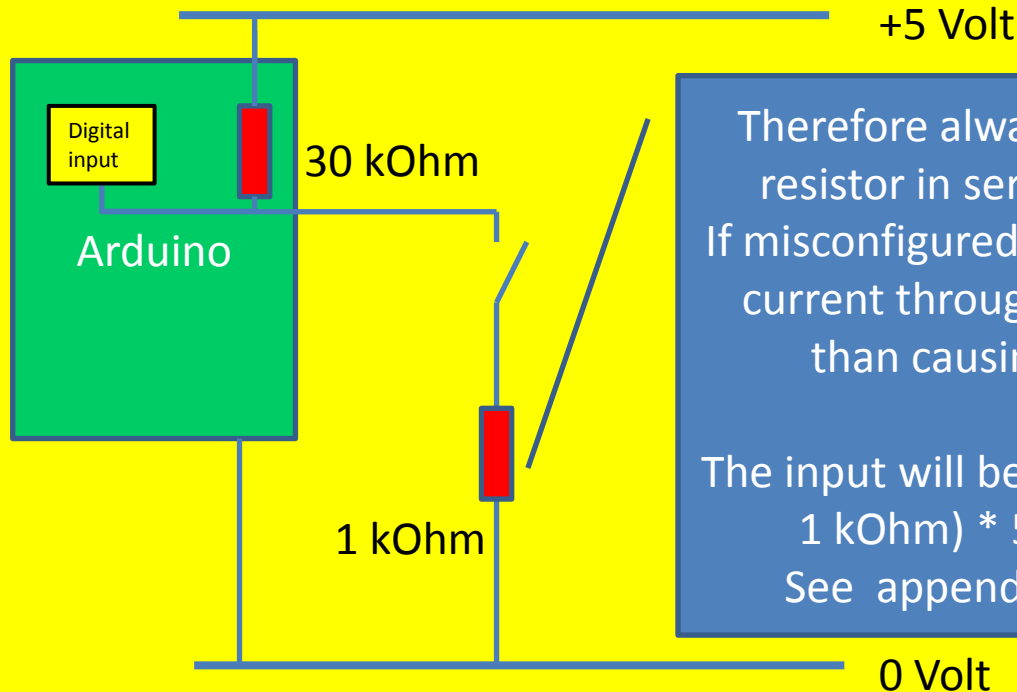
# Producing a HIGH value without external resistor



# Danger when misconfiguring



# Resistor in series to prevent damage in case of misconfiguration



Therefore always include a 1 kOhm resistor in series with the switch. If misconfigured, there will run a small current through the resistor rather than causing a short circuit.

The input will be  $1 \text{ kOhm} / (30 \text{ kOhm} + 1 \text{ kOhm}) * 5 \text{ volt} = 0,15 \text{ volt}$ . See appendix B for more info.

# Check / double check

- Please now check **both of you** that the wiring is correct.

# Bringing things together

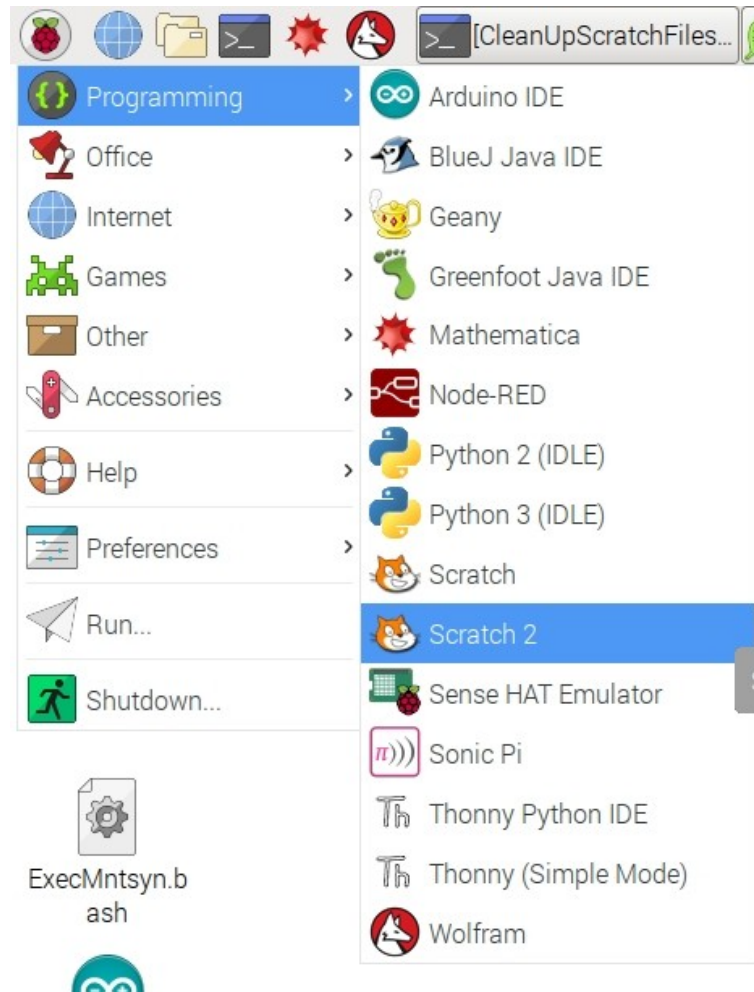
- Connect the 9 Volt connector to the board and switch power on
- Connect the USB connector to the Arduino
- Doubleclick *scratchClient Tutorial.scl* on the desktop to start scratchClient with the config file you have just updated



scratchClient  
Tutorial.scl

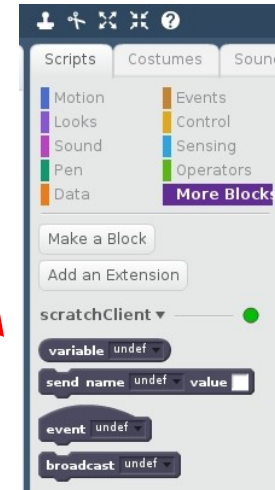
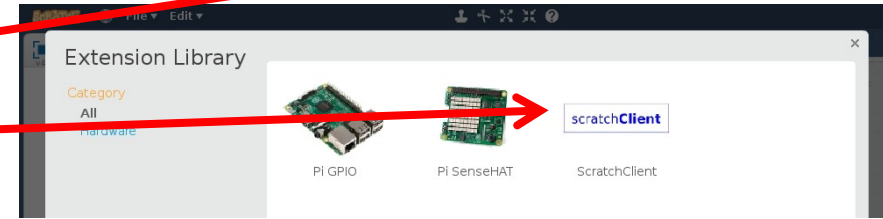
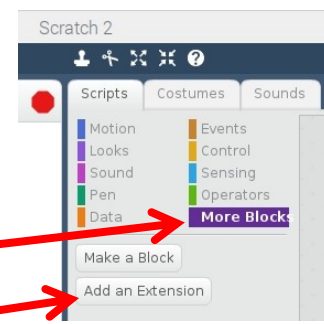
- It will also open a browser window where you can see variables
  - Explained later in the presentation
- In the terminal window you will see complaints that scratchClient has no connection to Scratch
  - Which is logical because Scratch was not started yet.

# Start Scratch 2



# Start Scratch 2 and get scratchClient blocks

- Click on *More Blocks*
- Click on *Add an Extension*
- Choose *scratchClient*
- Now the extra blocks of scratchClient get included.
- Wait 10 seconds and look whether the blue LED on the Arduino Nano blinks slowly instead of fast. If so, the connection is established.





# Create the Scratch program

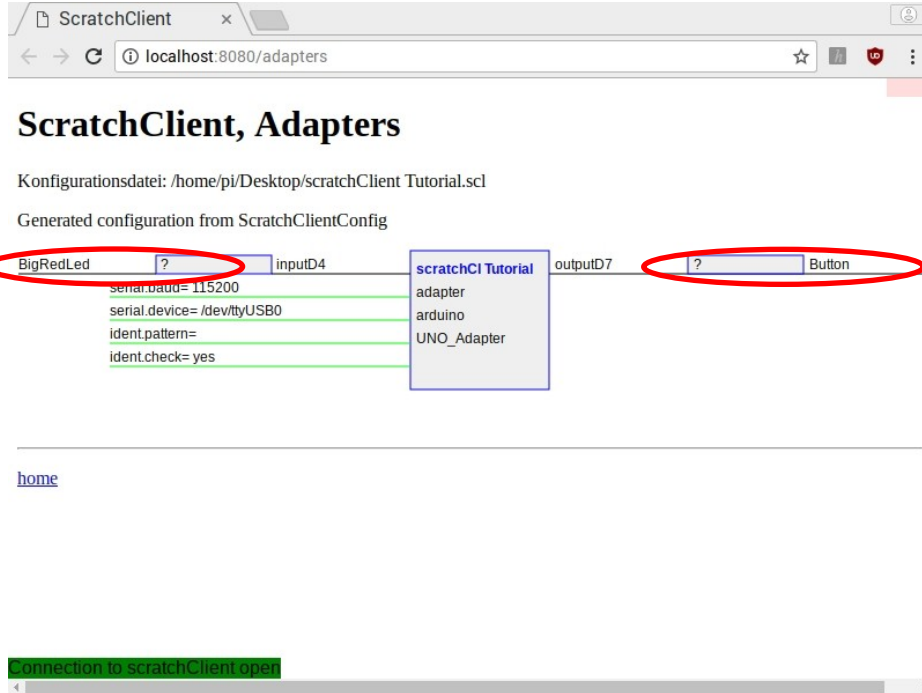
- Create this program in Scratch and try it out (click the set of blocks or click the green flag above the animation window).
- Digital in (Button):
  - 0 = pressed
  - 1 = not pressed
- Digital out (LED)
  - 0 = off
  - 1 = on
- Analyse how the program works.



# Does it work? (see next slide for help)

- If the LED on the Arduino is blinking slowly **only then** the config is downloaded and **only then** scratchClient works.
  - It may take 10 seconds after both scratchClient and the scratchClient extensions in Scratch 2 are loaded before this happens.

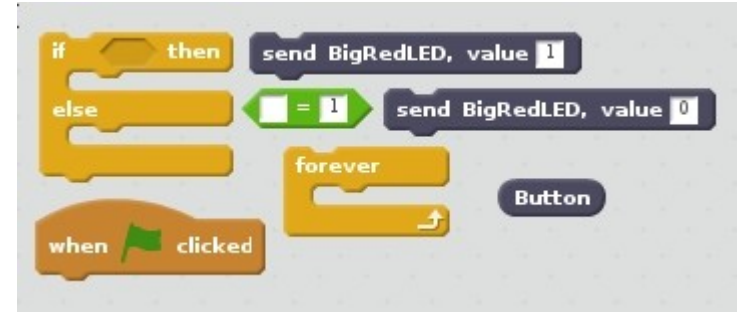
# You can monitor the values that are exchanged

A screenshot of a web browser window showing the ScratchClient interface. The address bar shows "localhost:8080/adapters". The page title is "ScratchClient". Below the title, it says "Konfigurationsdatei: /home/pi/Desktop/scratchClient Tutorial.scl" and "Generated configuration from ScratchClientConfig". The main content area shows a configuration for a "scratchClient Tutorial" adapter. On the left, there is a "BigRedLed" input field with a question mark, connected to "inputD4". Below it, there are several green lines representing data flow, with labels like "serial.baud= 115200", "serial.device= /dev/ttyUSB0", "ident.pattern=", and "ident.check= yes". In the center, there is a box labeled "scratchClient Tutorial" with "adapter", "arduino", and "UNO\_Adapter" listed below it. On the right, there is an "outputD7" connected to a "Button" output field with a question mark. At the bottom left, there is a green button labeled "connection to scratchClient open".

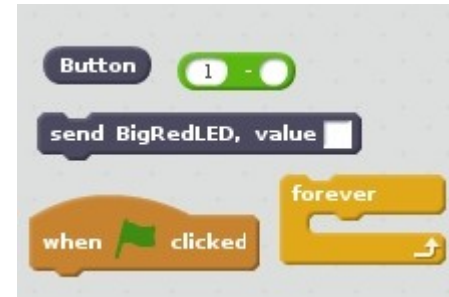
- Together with scratchClient, the browser is opened with URL *localhost:8080/adapters*
- Note the input & output directions:
  - Output of an adapter is an input for scratch
  - Output of scratch is an input for the adapters.
  - Hence the names input and output look to be reversed from what is in the config file.
  - Therefore best refer to the variable names.
- You will see that values are only displayed after they have changed (otherwise a question mark (?) is displayed).
- Not for now, but you can click the field and type a value which then will be sent in the appropriate direction.

# Change the program

- Make a change in the program so that the LED goes on when the button is pressed.
- There are (at least) two ways
- You need the blocks depicted here



Blocks for method 1



Blocks for method 2

# What if it does not work?

- Check whether you have multiple Scratch instances open
  - scratchClient can only work with one Scratch at a time (regardless of being Scratch 1.4 or Scratch 2).
- Check if the blue LED on the Arduino Nano is blinking *slowly*.
- Sometimes, especially after reboot, if everything seems fine, disconnecting and reconnecting the Arduino Nano may help.
- Try monitoring the variables, see before.

# One reminder ...

- Save your Scratch program regularly. Otherwise it will be lost on power out.
  - Power out can easily happen since you are pulling cables and may impact the power connector in the Raspberry Pi.
- Here come some current Scratch 2 anomalies on Raspberry Pi
  - You should **not** use spaces in filenames (if you do, they will be removed)
  - Scratch 2 forgets the folder where the file was opened from (Desktop). Just save it in */home/pi* as Scratch proposes
  - Scratch 2 will save it with a *.sbx* extension regardless what you specify.
- Hence once after power up
  - On the desktop, doubleclick *CleanUpScratchFiles.sh*
  - You can minimize the window, but let it run forever
  - This will permanently monitor the folder */home/pi*, and
    - Move all files with *.sbx* to the desktop as *.sb2* files
    - Any spaces in filenames are removed
    - If there is already such a file on the desktop, it will be moved to the folder ScratchProgramsBackup on the desktop



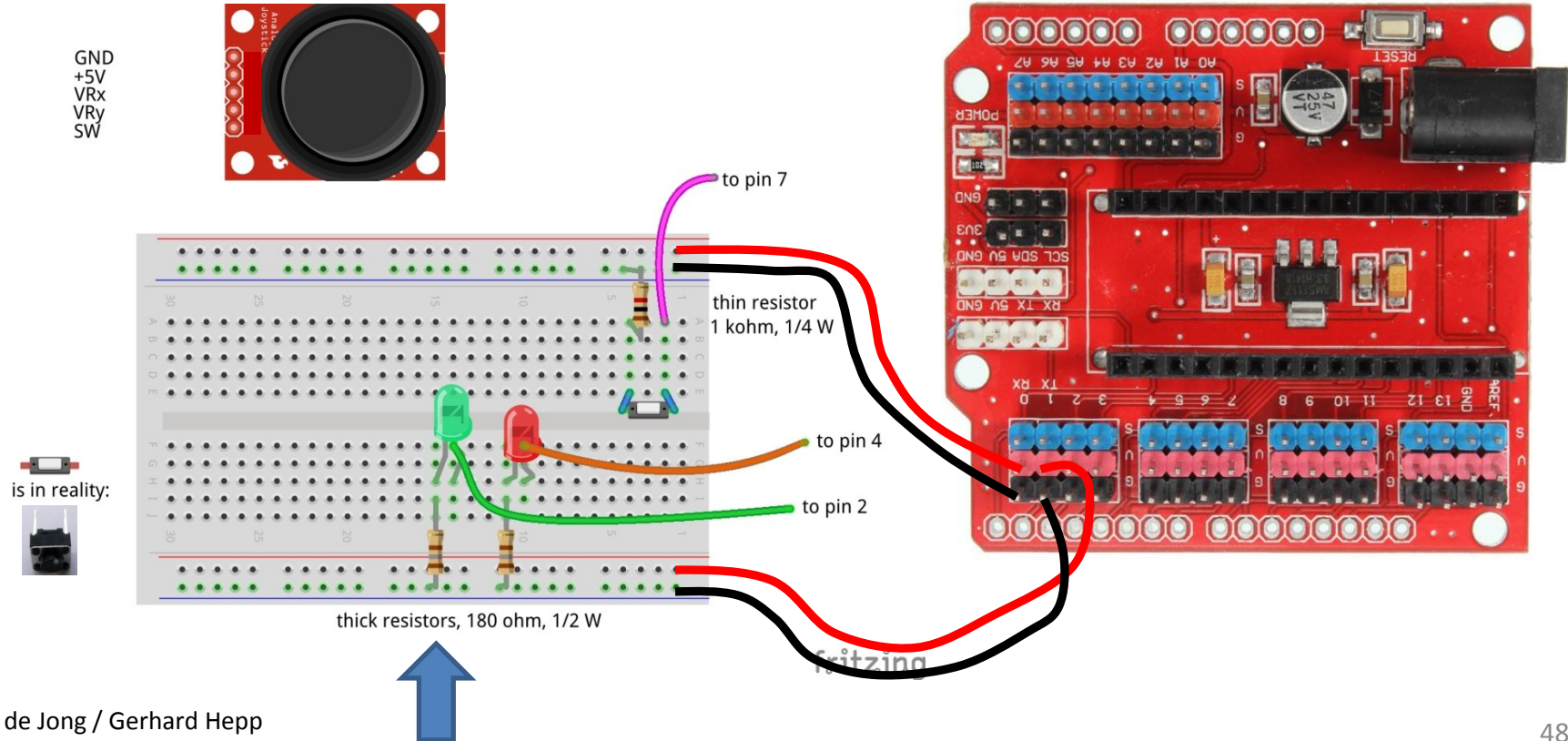
CleanUpScratchFiles.sh



ScratchProgramsBackup

# Part 6: Adding the Big Green LED

# Adding the green LED



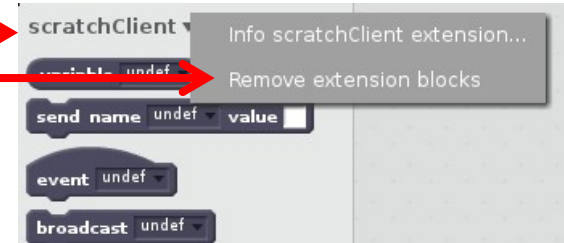


# Check and reconnect


- Check the correct wiring
- Switch on the 9V power
- Connect the USB cable again

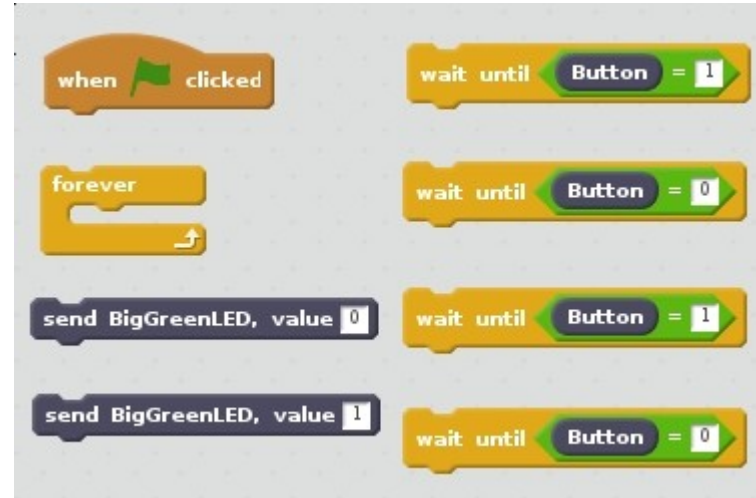
# Update the config file and restart scratchClient

- Use the config tool (which should still be open)
- Define an output (direction: out, function: output) on pin 2 and call it *BigGreenLED*
- Save the config file (and leave the tool still open)
- Doubleclick *scratchClient Tutorial.scl* on the desktop again
- This will stop the previous instance of scratchClient and restart with the updated configuration.
- Remove the extension block.
  - Right click here
  - Choose *Remove extension blocks*
- Add the extension block again.
  - You will see in the *More Blocks* section that you can now choose to send values to BigGreenLED



# Update Scratch

- Add code that does this:
  - Press button: LED goes on
  - Press once more: LED goes off
- You need these program elements. 



# This is the end of the beginners level

- You got *digital out* (LED) and *digital in* (a button) working
- You know how to work with scratchClient and Scratch 2
- You know how to configure scratchClient
- If you want and have time you can
  - Look at some of the yellow slides if you skipped them earlier
  - Continue with one of the next levels
    - These are in different files.
- Otherwise,
  - Copy the material to a USB stick and clean up. See further below.

# Overview of next levels

- Intermediate level
  - Analog In: Potentiometer
  - Pulse Width Modulation (in the absence of analog out)
    - Dimming a LED
    - Controlling a servo
    - Controlling a buzzer
- Advanced level
- Expert level

# 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 *scratchClient Tutorial.scl* file on the desktop, plus all .sb2 files that you created on the desktop
- The rest of the material you can download from [www.github.com](https://www.github.com), search for *scratchClient*
- Take the flyer with you to remember where to find the material on github.

# Part 11: Summary & take aways



# Takeaways 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.
- In Scratch 2, you use the extension block for scratchClient to get blocks that you can use to interact with scratchClient
- 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 a few more, see the advanced level tutorial
  - You can configure pull up resistors on Digital In

\* See the intermediate level tutorial

# Part 12: Clean up / teardown

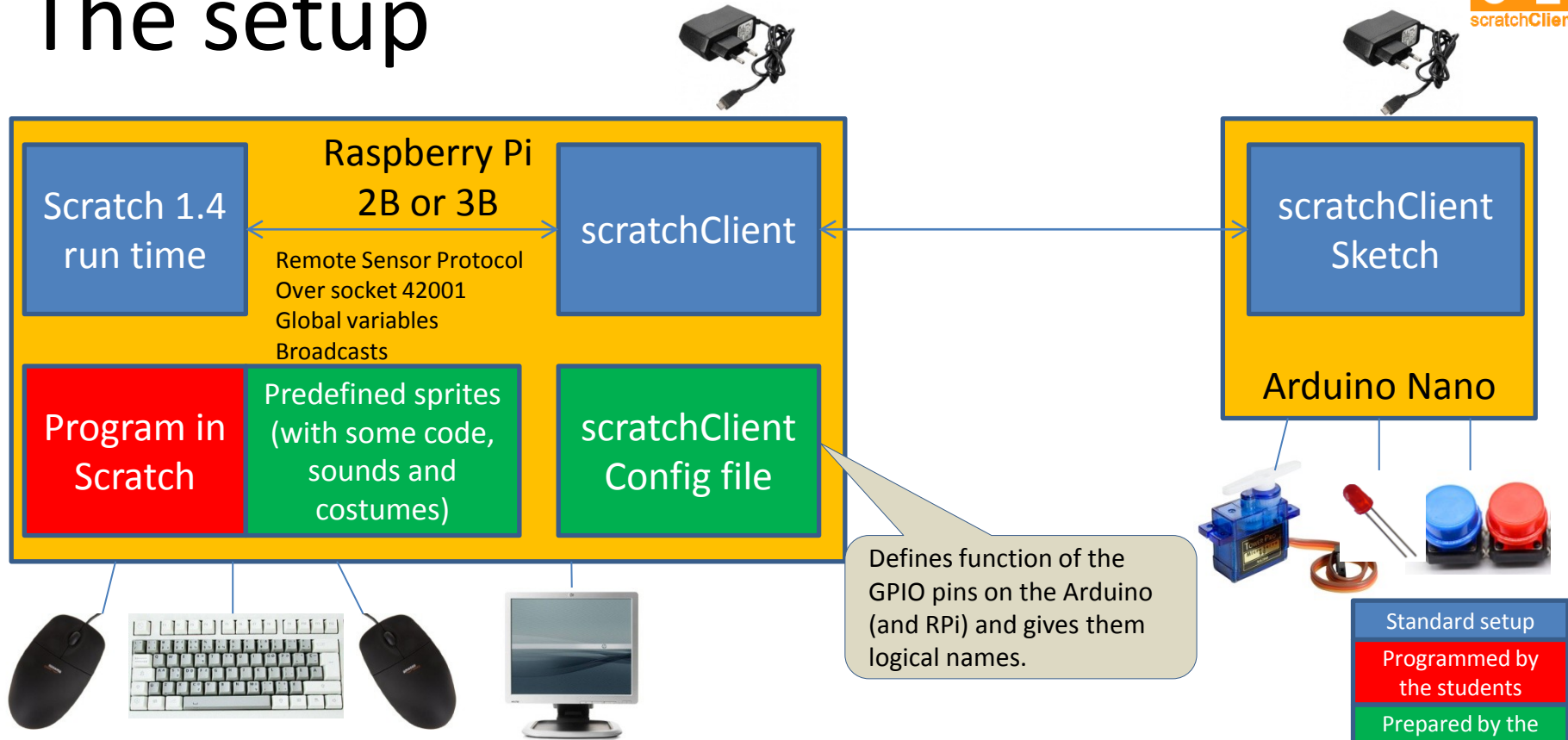
# If your scratchClient day ends here ...

- 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** (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
- Shutdown the Raspberry Pi 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)

# Appendix A

## When using Scratch 1.4

# The setup



Standard setup


Programmed by  
the students

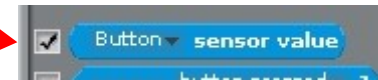
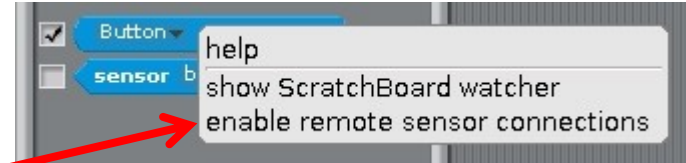
Prepared by the  
teacher or expert

# Defining the config file

- Nothing new, you can use the exact same config file for Scratch 1.4 as for Scratch 2.

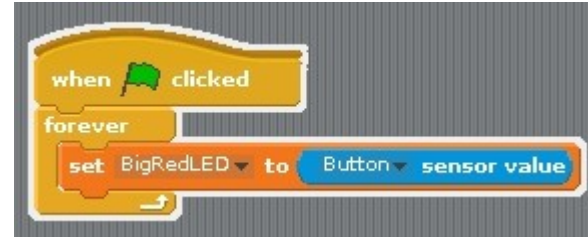
# Create the Scratch program

- Start Scratch  → Programming → Scratch
- Enable remote connections (right click on *sensor value*)
- Create the variable *BigRedLED*, available to all sprites
- Make the variable visible (tick the box in front)
- Make the *Button* sensor visible
- Save the file on the desktop.



# Program in Scratch 1.4

- Make this program in scratch which will let the Red LED lite up when the button is released.
- Test whether it works.





# Further ...

- You can now look at the Scratch 2 material earlier in the presentation.
- Try out adding the BigGreenLED and the same blocks to process it.

# Appendix B

## Basic Electronics

# To be added

# End of the **beginners** workshop