

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

Trier – 24 June 2017

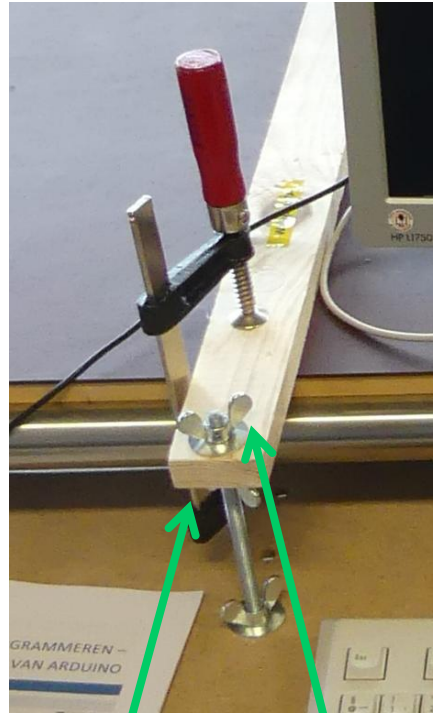
Part 1: Introduction

If the presentation is messed up ...

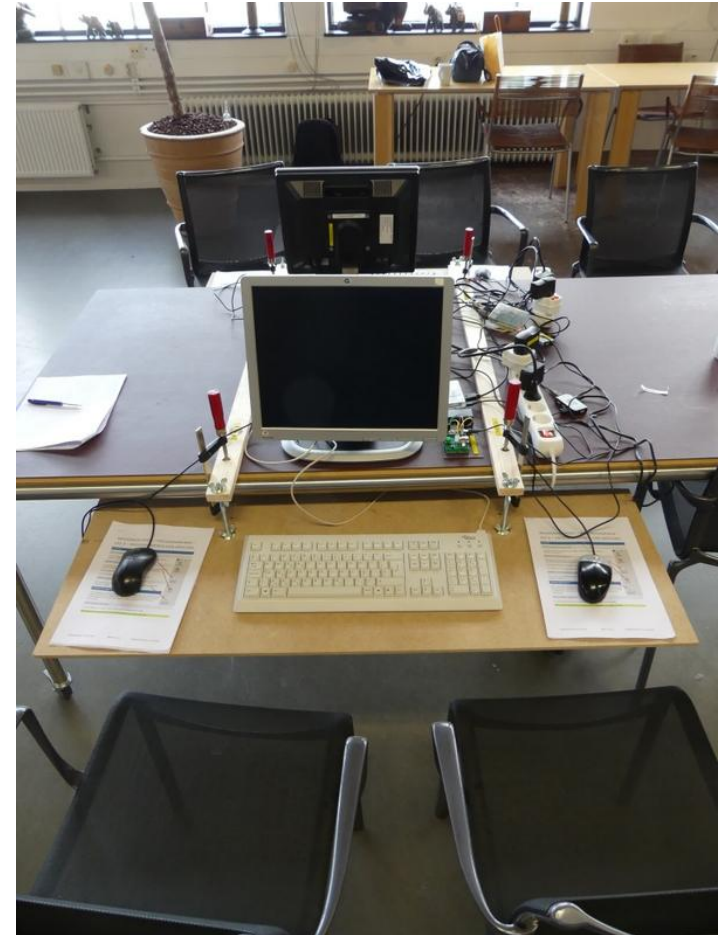
- You need the Calibri font on your Raspberry Pi (standard on Windows but not on Raspberry Pi).
- Either obtain and install that font, or ...
- ... look at the PDF version of the presentation
 - But in that version you cannot copy/paste.
 - Therefore there is a file called ForCopyPaste.txt in the directory holding whatever you would need to copy paste in this workshop.

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.



Workshop organisation

- Welcome and introduction presentation (10 min).
 - After that everyone will work at his or her own pace.
 - At times we will present for 5 minutes to explain a next concept.
 - Language: English, but help in German is available.
 - At the end copy the material you created to your USB stick (if you want)
 - Cleanup, or prepare for the Advanced workshop.
- The major steps:
 1. Get a working hardware and scratchClient config using a scratchClient config tool (new).
 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.
And repeat this.
 5. When time left: write a Scratch program.

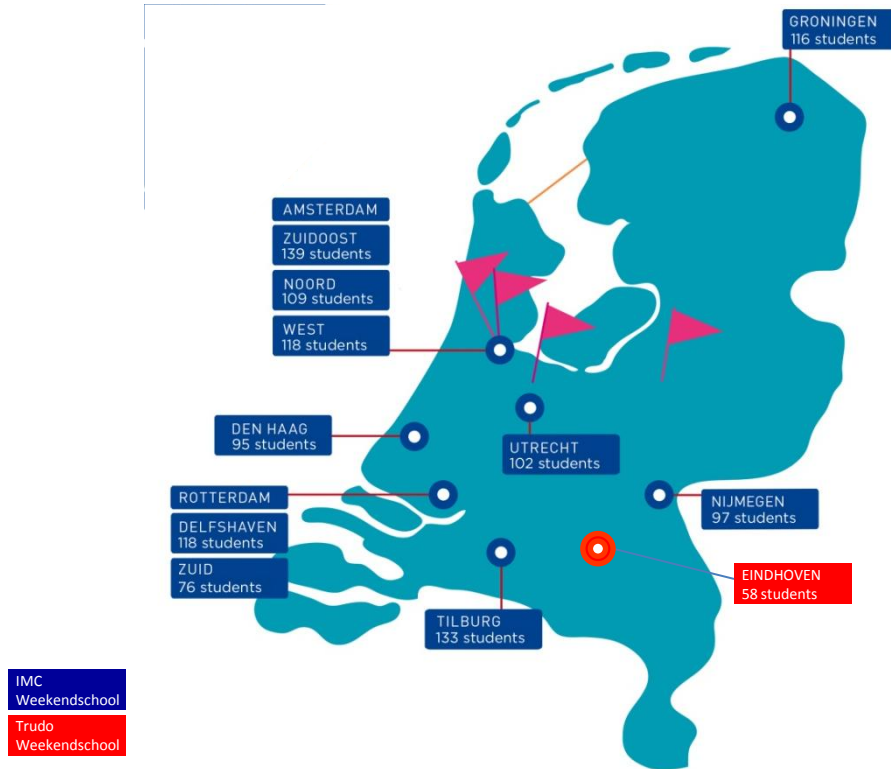
Objectives

- At the end of today you should be able to:
 - Reproduce the setup at home (provided you have the hardware 😊)
 - Understand
 - 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`.

Weekendschool





- Generally parents tell their children: “Learn for later”. But what is later?
- “I want to become an engineer. Do you maybe know an engineer? I would very much like to meet one”
- Professionals teach children on Sundays about their job. Large variety of topics.
- > 1000 students between 11 and 14 years from underprivileged areas.
- Curriculum lasts for 2.5 to 3 years
- 3500+ volunteers.
- Funded by 110+ sponsors (companies, individuals, foundations).
- Started in 1998.
- It works! Data shows: alumni have better professional prospects, are more self-aware, and feel more connected with society.

Lessons at the Weekendschool

- I personally teach
 - Mathematics (since 2004)
 - Physics (since 2011)
 - Programming (started in 2016)
 - Electro (started in 2016)
- Programming: 3 (2) lessons on 5 (3) Sundays with Scratch, Raspberry Pi and Arduino
 - Last lesson comprises 2 days but is optional and can be done in a later year.
- Lesson 2: Physical computing:
<https://www.youtube.com/watch?v=Qo1gnXNzhqE>

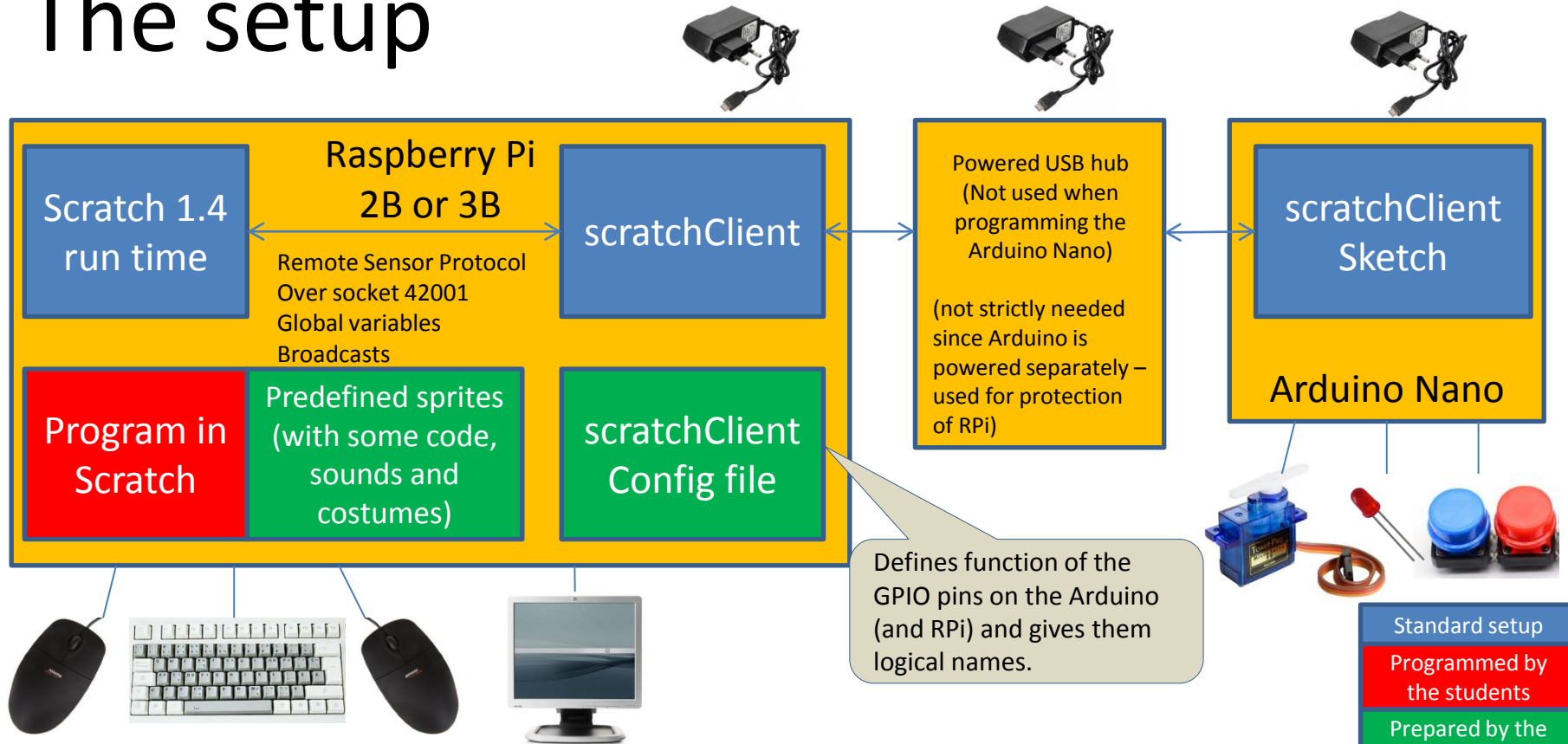
Only a few rules today



- If Raspberry Pi needs to be rebooted, it needs a powercycle. Please do not pull plugs but use the switch in the outlet after you have done the shutdown.
- 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 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



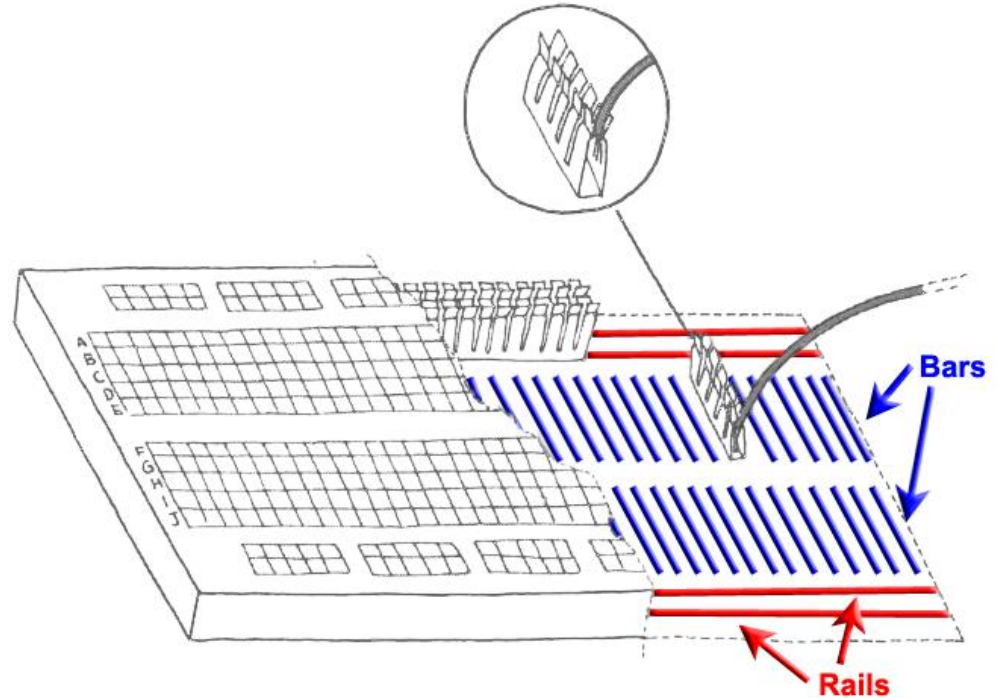
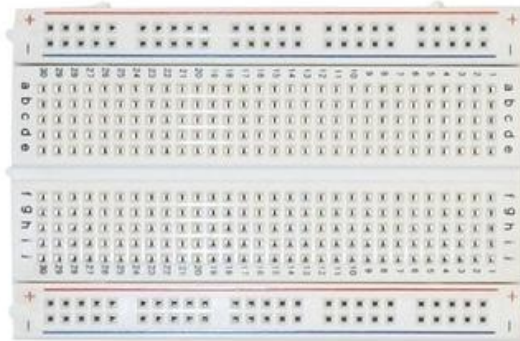
Where is the presentation?

- You will find this presentation = the instruction at your *desktop* in the folder *PiAndMore*.
- If you already know the stuff in the presentation and get bored then feel free to work on your own
 - Silently please ... 😊

Part 2: Getting to know the components

Breadboard

- Used to quickly build electronic circuits
- Note the 2 rails for + (VCC) and – (GND)
- Note 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

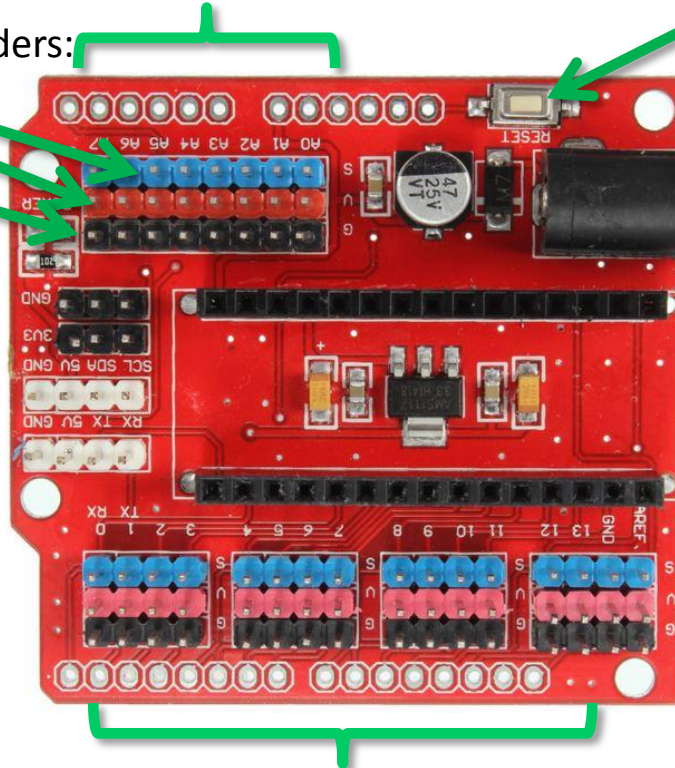
Per GPIO signal 3 headers:

S (blue = signal)

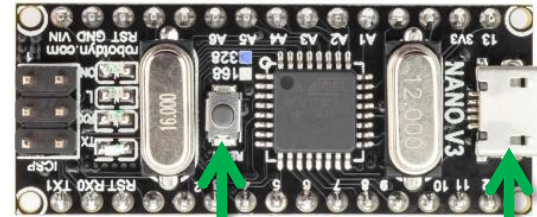
V (red = VCC = +)

G (black = GND = -)

(very handy to e.g. connect servos)



Arduino Nano with 328P processor

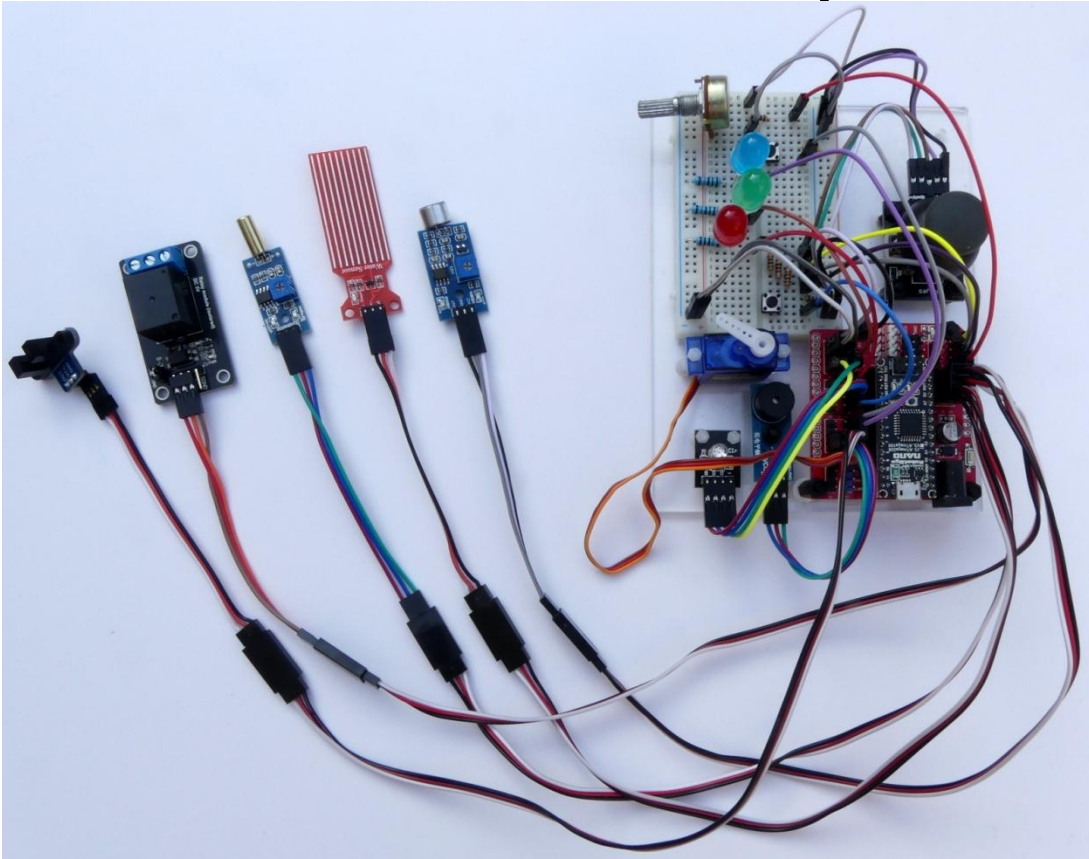


Micro USB port

Reset button

π_{++}

... and if you continue with the
advanced workshop



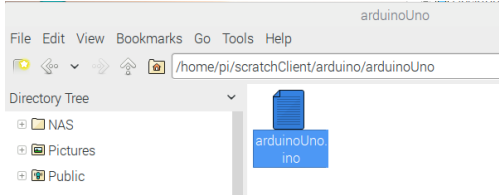
Part 3: Loading the sketch in the Arduino

Preparing for programming the Arduino Nano

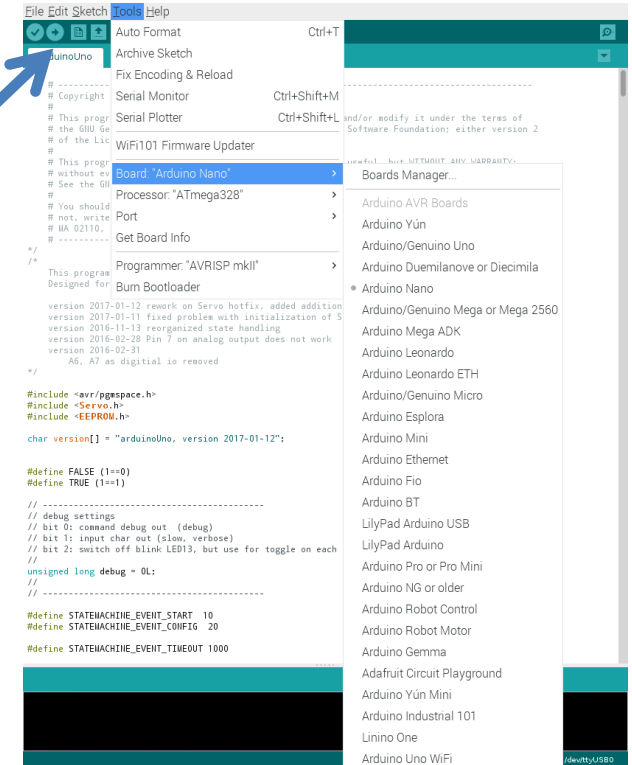
- The Arduino needs to run a program so that it can communicate with the Raspberry Pi and understand the messages from scratchClient.
 - We need to start putting that in.
- Connect the Arduino Nano
 - Make sure to connect directly
 - not via the USB hub
 - for strange reasons it will not work via a (this) USB hub, although it used to in previous releases of Raspian (and still should).

Uploading scratchClient to the Arduino

- Make sure that the Arduino is directly connected to the Raspberry Pi (not via the USB hub)
- Open 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.



Part 4: Defining the configuration

Give names to pins and define the purpose of the pin

Starting the config tool

- This creates config files to map between Arduino pins and logical names.
- Navigate to */home/pi/scratchClient/tools*
- Doubleclick *scratchClientConfig.sh* and chose *Execute*.
 - If the file opens rather than presenting the choice, then you must set execute permissions first.
 - To set permissions: right click on the icon, choose *properties*, then *permissions* and set execute rights to at least *owner*.

Define the first config file

The screenshot shows the Pi++ configuration tool interface. On the left, a table lists pins from D0 to A7. Pins D4 and D7 are highlighted with red circles. Below the table, the 'Parameter serial.device' is set to '/dev/ttyUSB0' and 'Parameter ident.check' is checked, both also highlighted with red circles. On the right, the XML configuration code is displayed, showing the configuration for pins D4 and D7. A red arrow points from the 'Parameter ident.check' checkbox to the XML code.

name	arduino	direction	function	scratchName
D0	void			
D1	void			
D2	void			
D3	void			
D4	out	output		Big Red LED
D5	void			
D6	void			
D7	in	input_pullup		Button
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			
A7	void			

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

    <!-- id = 'D7' direction = 'in' function = 'i
    <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/tty
    <parameter name="serial.baud" value="115200"/>

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

  </adapter>
</config>
```

Parameter serial.device: /dev/ttyUSB0

Parameter ident.check: ☒

Parameter ident.pattern:

type id message

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

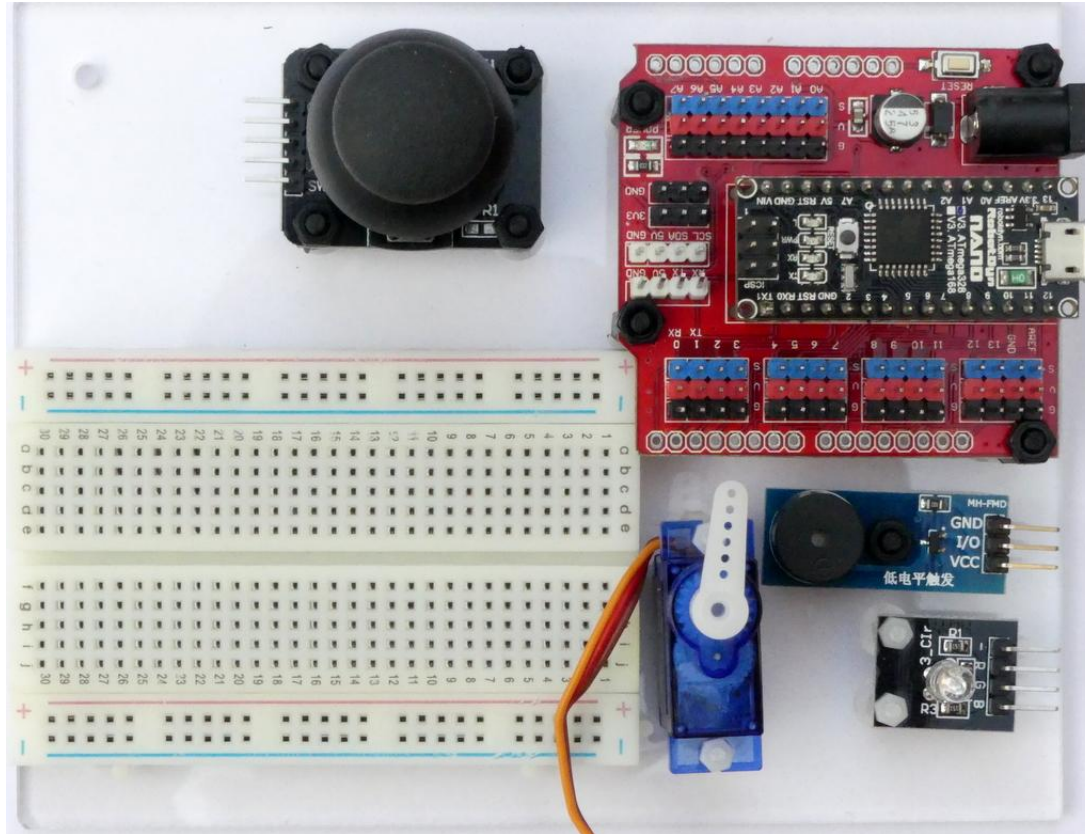
- Double click a cell to get a drop down menu
 - First for *direction* then for *function*.
- Make sure to give all pins a name if you choose something else than void
 - So make sure not to save if you still have red borders around cells. scratchClient will fail to start with such a config file.
- The 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 in the folder *PiAndMore* on the desktop
- Call it *PiAndMore.xml*
- **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

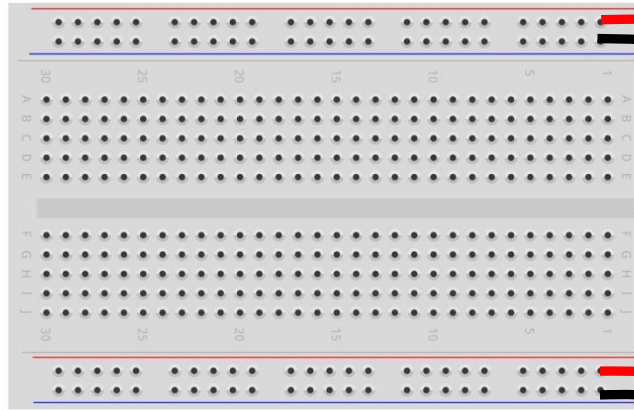
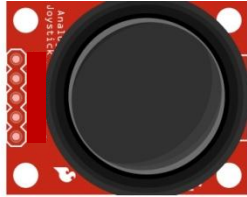


Use short wires and use the indicated holes

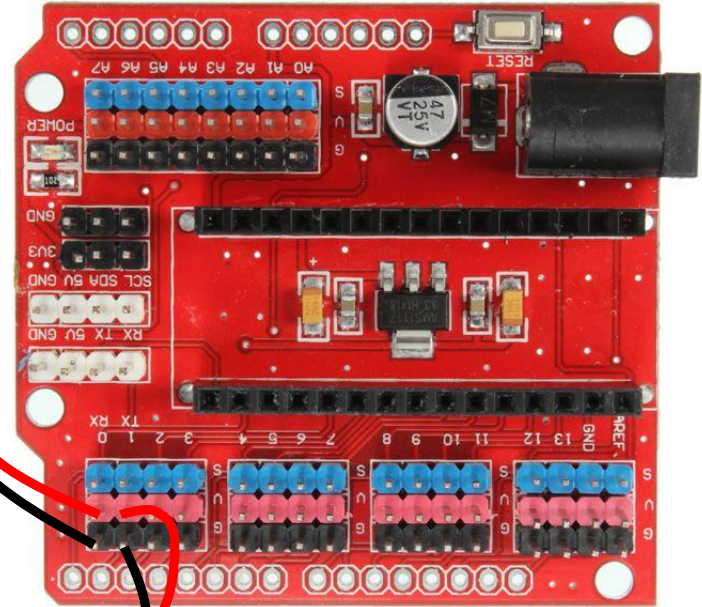
- There are some short wires (10 cm) and some longer (15 cm)
- Use the shortest that you can
 - Get a less messy setup
 - You may run out of long wires otherwise
- Ignore wire colors.
 - Unfortunately there are not enough wires of the correct color to get that right.
- You can in principle build up at 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

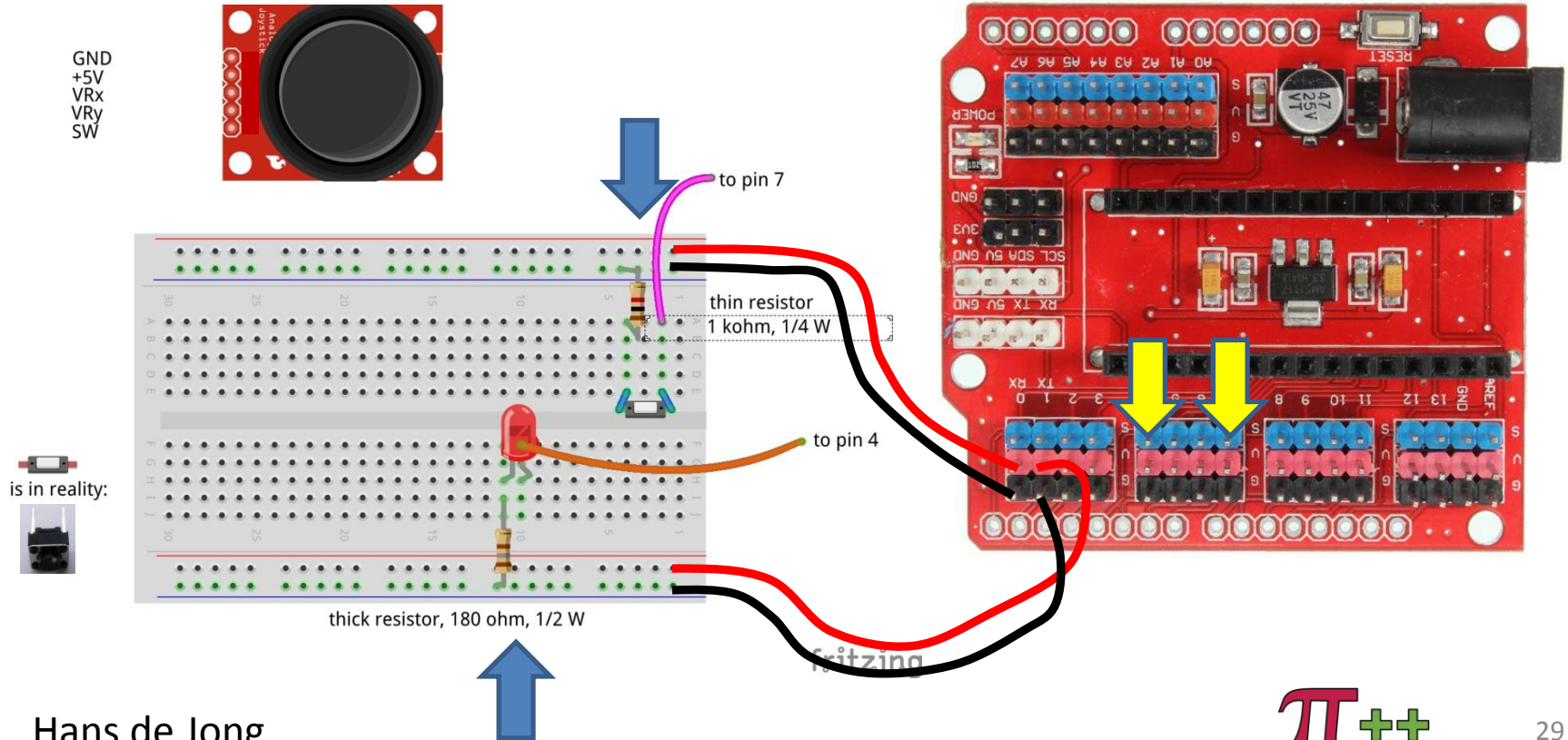
GND
+5V
VRx
VRy
SW



fritzing

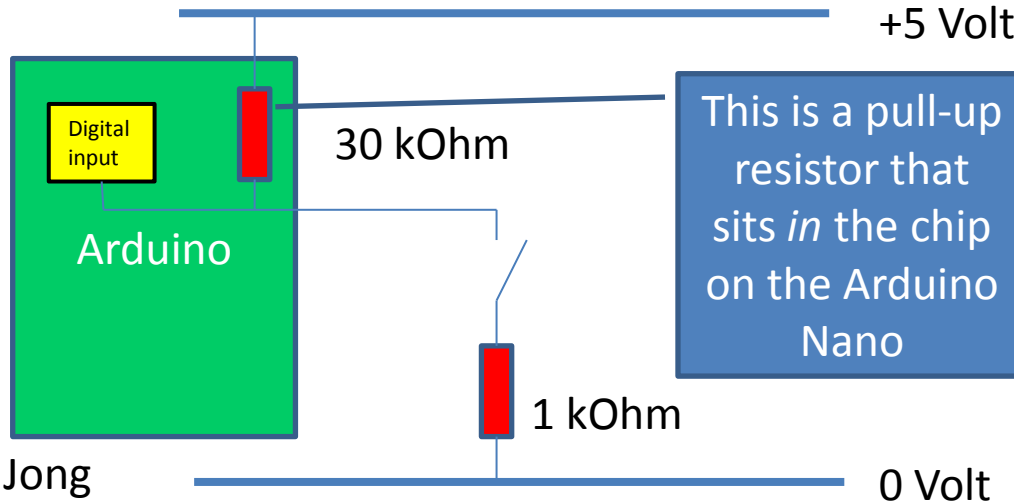


Insert the red LED and button



Why are these resistors needed?

- In series with a LED: to limit the current through the LED
 - You may blow up the LED and/or the Arduino otherwise.
- In series with buttons, potentiometers etc: to avoid blowing up the Arduino in case of config errors



Consider what would happen if the 1 kOhm resistor were not present and the port would be used as output and the button closed.

Then the Arduino would output 5 Volt, and the switch is directly leading it to the 0 Volt line → short circuit!

Check / double check

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

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"
```

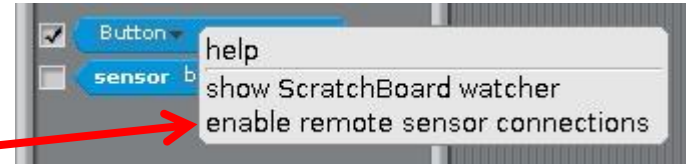
- Make the file executable (file properties, permissions)
- Do you understand what the file does?
 - If not, please ask

Bringing things together

- Connect the 9 Volt connector to the board and switch on power
- Connect the USB connector to the Arduino, this time go via the USB hub
- Start scratchClient with the script you just made (e.g. double click it and when asked choose *Execute in Terminal*).
- It will complain that it has no connection to Scratch
 - Which is logical because Scratch was not started yet.

Create the Scratch program

- Start Scratch  → Programming → Scratch
- Enable remote connections (right click on *sensor value*)
- Create the variable *Big Red LED*, available to all sprites
- Make the variable visible (tick the box in front)
- Make the *Button* sensor visible
- Save the file in the *PiAndMore* folder on the desktop
 - Name does not matter, e.g. *PiAndMore.sb*



Does it work? (see next slide for help)

- If the LED on the Arduino is blinking slowly **only then** the config is downloaded and scratchClient works.
- It may take 5 seconds before this happens.
- Change the value of *Big Red LED* to 1.
 - You can use the slider on the variable. You see no slider? Double click it a few times.
 - Does the LED lite?
- Press the button on the breadboard
 - Do you see the value of the sensor *Button* change from 1 to 0?

What if it does not work?

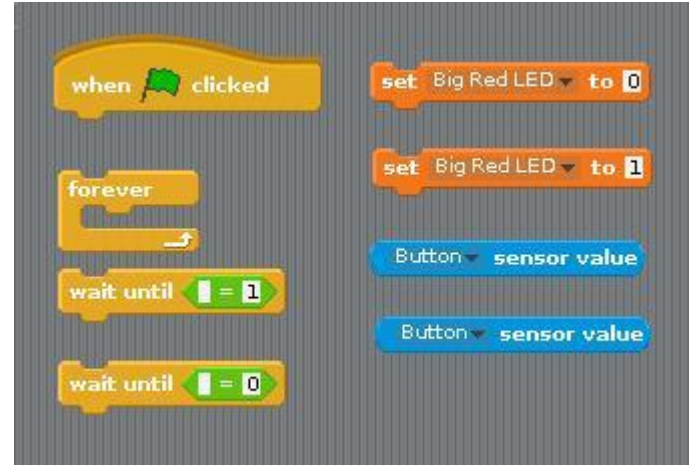
- Check whether you have multiple Scratch instances open
 - scratchClient can only work with one Scratch at a time.
- Check the output of the scratchClient terminal window
 - Did it find the board?
- Sometimes, especially after reboot, if everything seems fine, disconnecting and reconnecting the board may help.
- Try monitoring the variables (see next slide).
- If everything fails, scratchClient can be started with tracing on.
- But of course we are here to help you at any step.

Monitor the variables

- Open the browser
- Type *localhost:8080*
- Click *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).

Program in Scratch

- Make a program in scratch which will let the Red LED lite up when the button is pressed.
- You need these elements.



One reminder ...

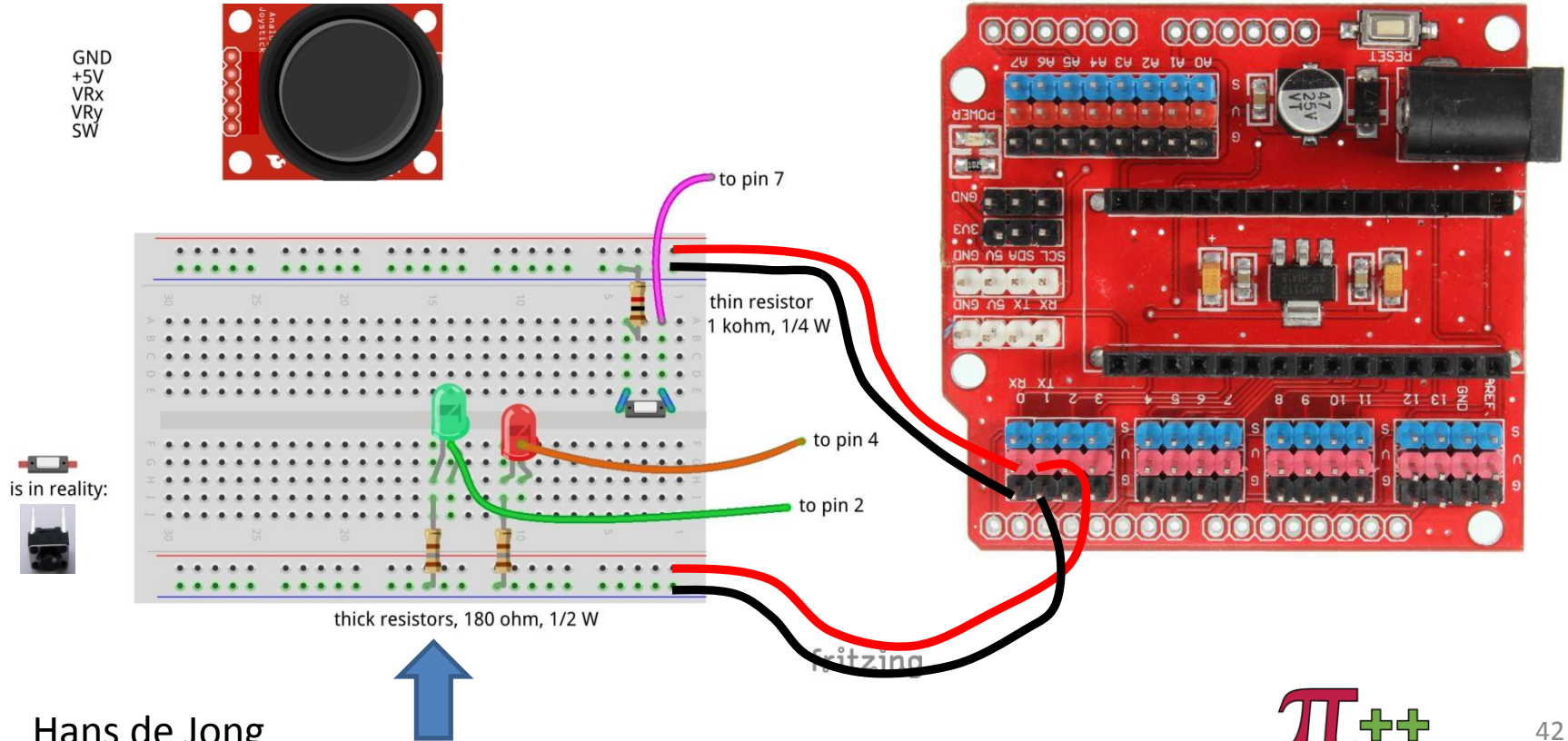
- Save your Scratch program regularly. Otherwise it will be lost on power out
 - Power out can easily happen since you are pulling cables and may impact the power connector in the Raspberry Pi.
- Every now and then copy the Scratch file to a backup file
 - Recommendation: give those copies a sequence number.

Part 6: Adding the Big Green LED

Changing the wiring on the board

- When changing the wiring on the board
 - First pull the USB cable out (from the board or from the hub)
 - Switch off the 9V supply

Adding the green LED



Check and reconnect

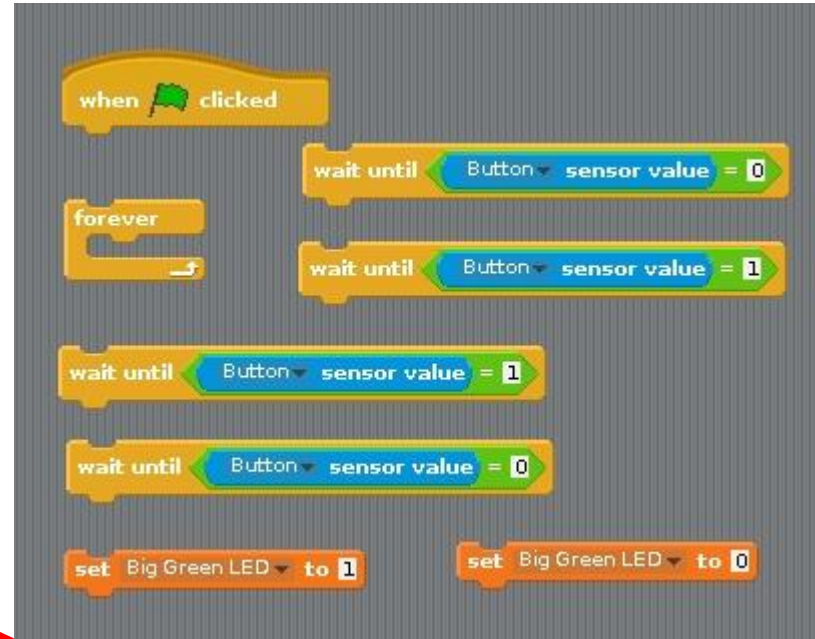
- Check the correct wiring
- Switch on the 9V power
- Connect the USB cable again

Update the config file and restart scratchClient

- Stop scratchClient (close the window)
- Use the config tool (which should still be open)
- Define an output (direction: out, function: output) on pin 2 and call it *Big Green LED*
- Save the config file (and leave the tool open)
- Restart scratchClient using the script that you created before

Update Scratch

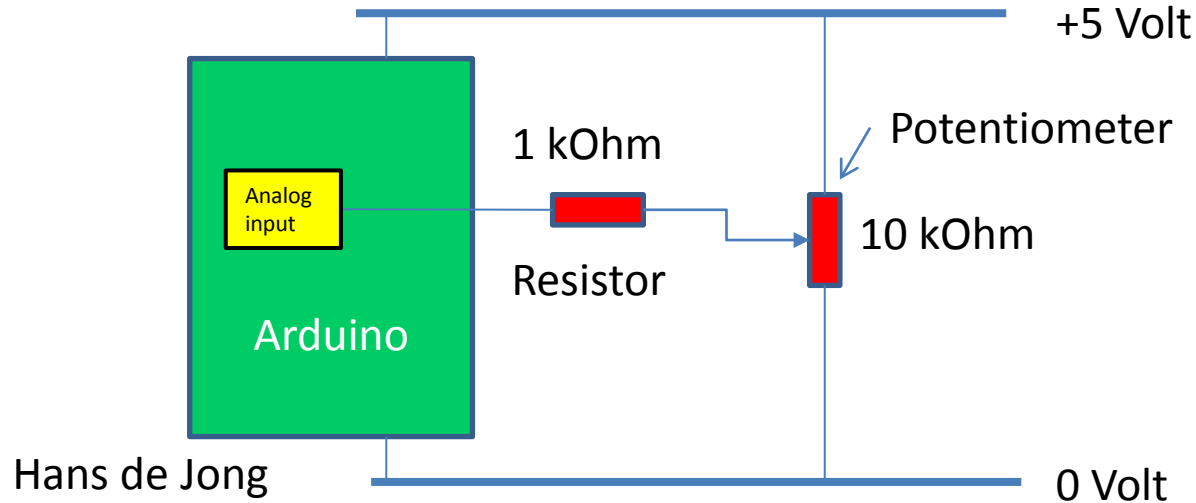
- Define the variable *Big Green LED*
- Create a sprite for the green LED
- Create in that sprite a program that implements a toggle switch:
 - Press once: LED goes on
 - Press once more: LED goes off
- You need these program elements.



Part 7: Adding analog input

The electronics of a potentiometer

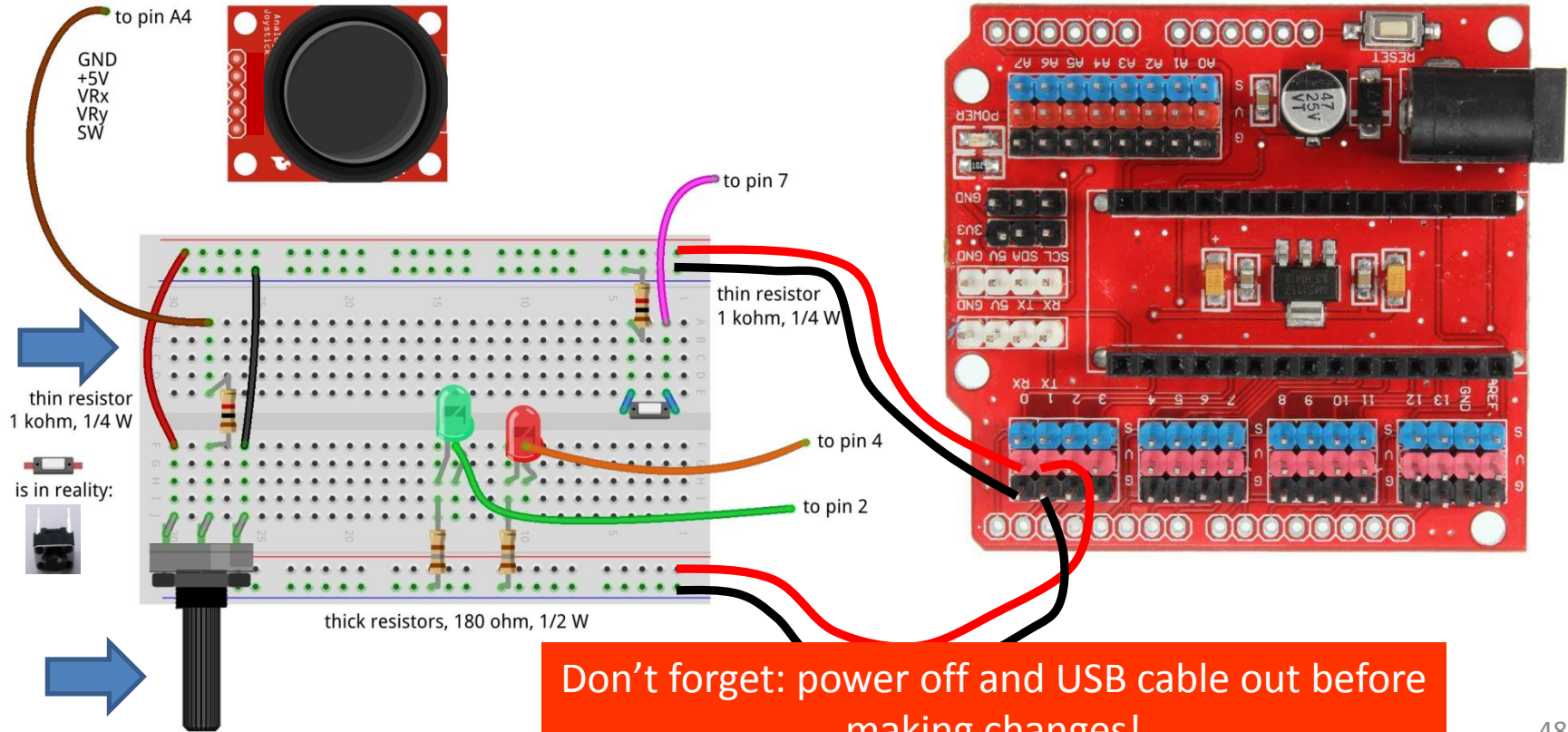
- The potmeter is a resistor that will be connected between 5 V and 0 V.
- Over the length of the resistor the voltage drops linearly.
 - E.g. in the middle it would be 2.5 Volt
 - So this is an analog signal. The value can be changed between 5 V and 0 V.
- There is a 1 kOhm resistor in series for the reasons discussed earlier



Consider what would happen if the 1 kOhm resistor were not there and the port would be used as output and potmeter positioned close to the 0 Volt side.

Then the Arduino would output 5 Volt, and the potmeter is directly leading it to the 0 Volt line → short circuit!

Adding analog input



Update the config file

- Update the config file again
 - Define *Potmeter* on pin A4 (direction: in, function: analog)
- Now that you are updating anyway, already define for the next steps:
 - Define *Servo 1* on pin 12 (direction: out, function: servo)
 - Define *Big Blue LED* on pin 5 (direction: out, function: PWM)
 - Define *Buzzer* on pin 11 (direction: out, function: PWM)
- Don't forget to save

Test whether it works

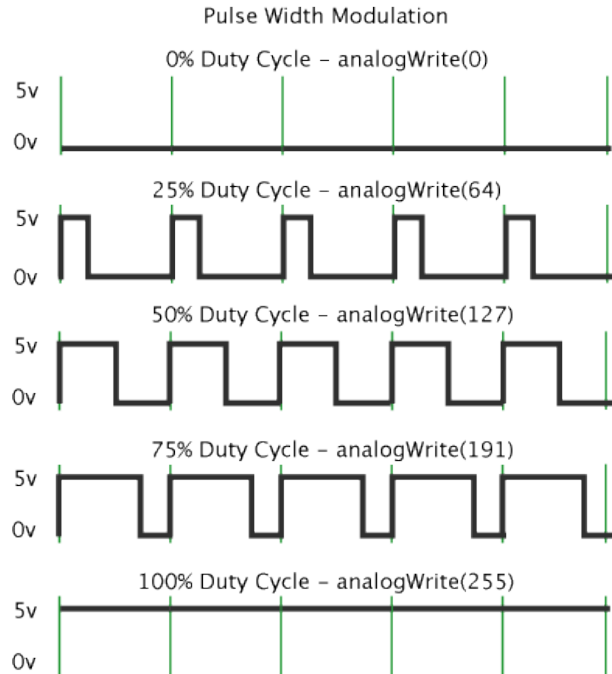
- Stop and restart scratchClient
- Reconnect and repower the board
- Make the sensor *Potmeter* visible
- Turn the knob and see the values of *Potmeter* change
 - Between about 0 and about 1024
- We will write some Scratch program to use it in the next step

Part 8: Pulse Width Modulation

Where we are ...

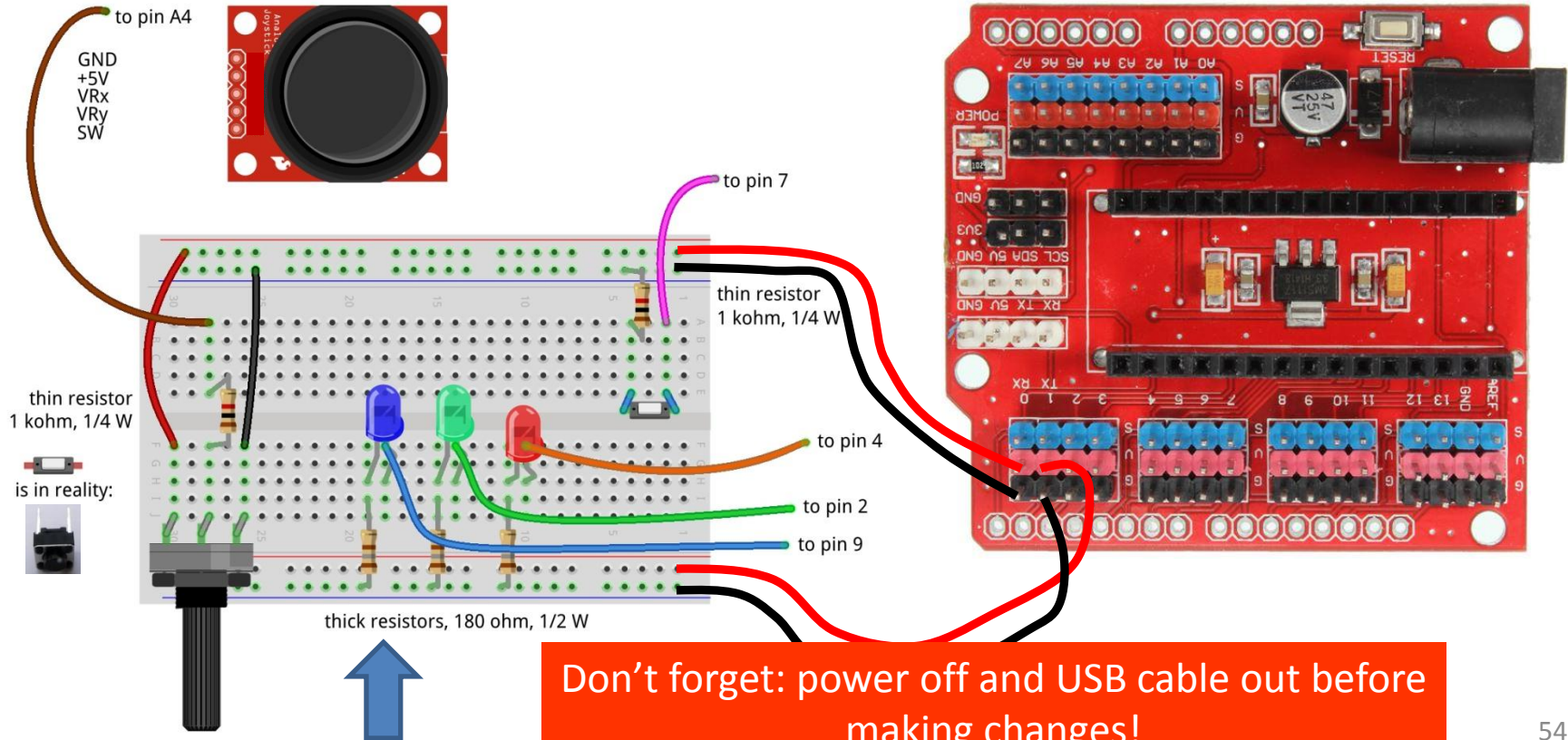
- We have seen:
 - Digital Input (the button)
 - Digital Output (the LEDs)
 - Analog Input (the potentiometer)
- So what would be most logical next?
 - Analog Output? → but it does not exist!
 - However there is a good alternative: Pulse Width Modulation

Pulse Width Modulation (PWM)



- By modulating (changing) the pulse width, the amount of energy fed to the e.g. LED is changed, and hence the intensity with which you see it lighting.
- Note that in practice the LED is blinking some 800 blinks / second.
- However, no human eye can see more than 100 blinks / second.

Dimming a LED with PWM



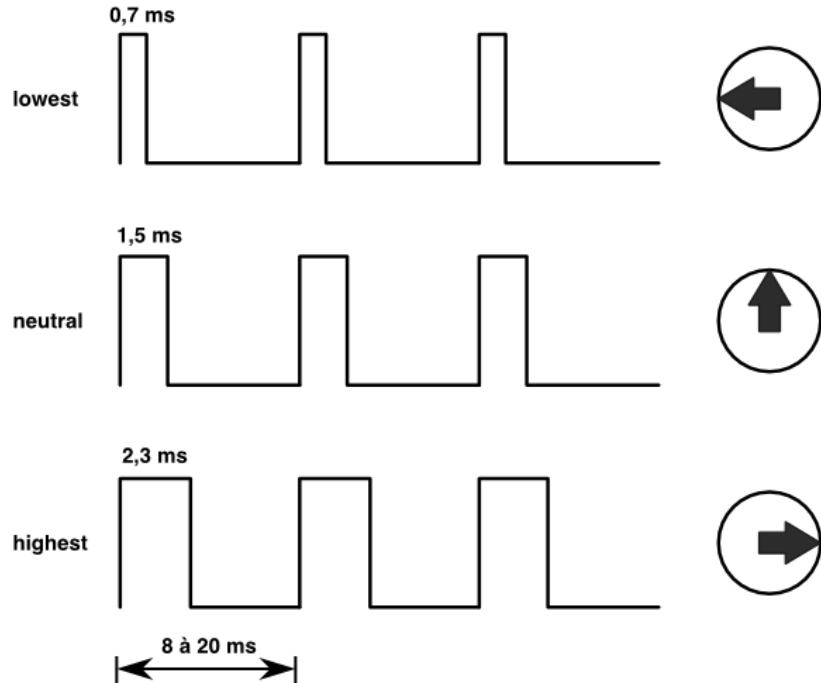
Test

- Switch the power on again and reconnect
- You will see that scratchClient finds the board again
- You already updated the config file in the previous step, so no need to restart scratchClient
- Define in Scratch a variable *Big Blue LED*
- Make the variable visible
- Give *Big Blue LED* values between 0 and 255
 - You can use the slider on the displayed variable for values between 0 and 100
 - For values above 100 we will use a program, see next slide
- Does the LED brightness change?

Connect Potmeter and Big Blue LED (via Scratch)

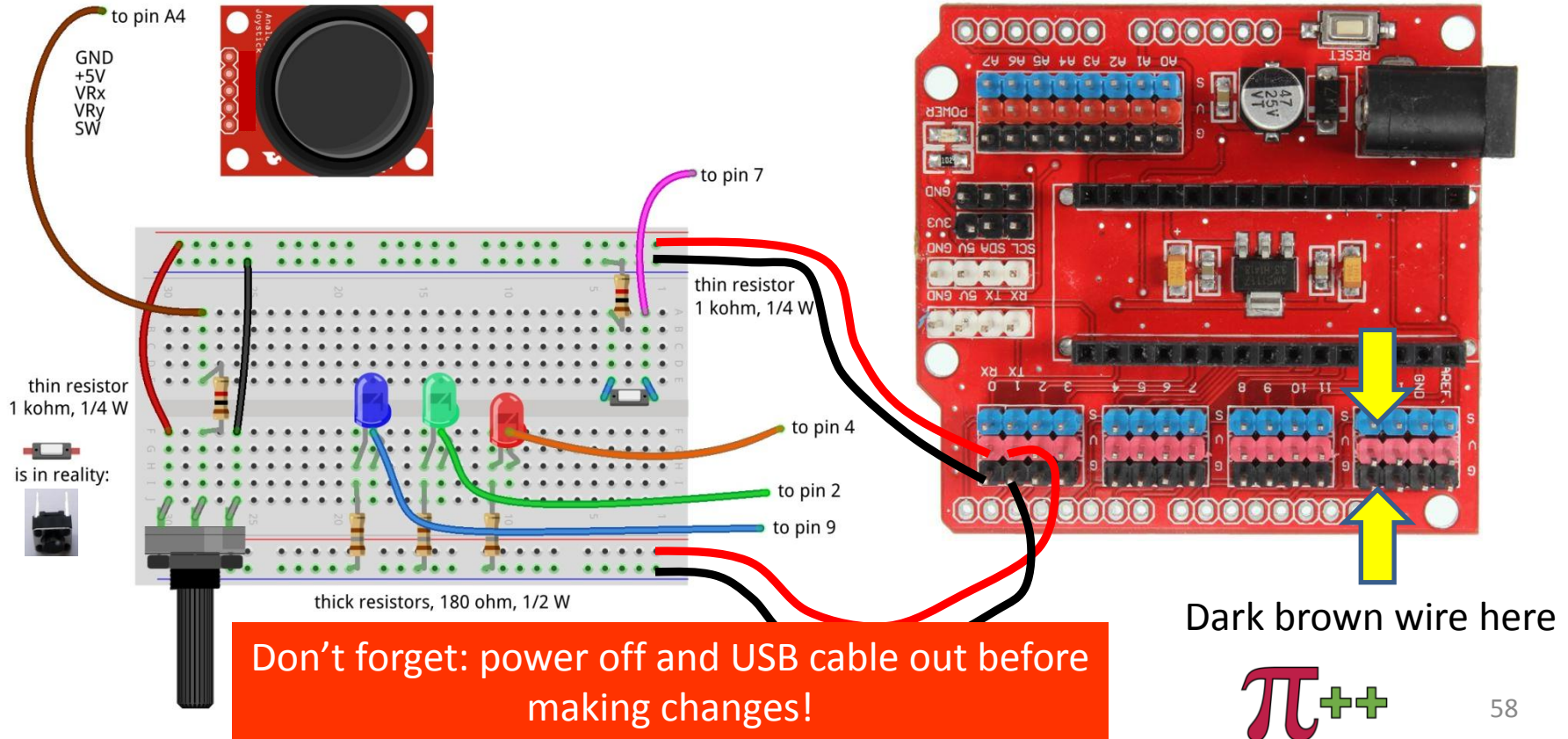
- Make some code that takes the potentiometer reading (between 0 and 1024) and transforms it into the range 0 to 255 (so divide by 4) and set the value of *Big Blue LED*.
- Try out whether turning the potmeter changes the light intensity.
- Notice that when turning to the right, the intensity goes down.
 - You would expect it to go up ...
 - How can you very simply change this by interchanging two wires?

Controlling a servo with PWM



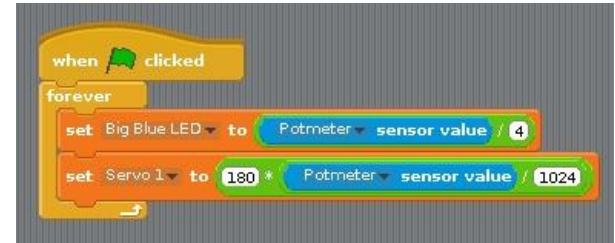
- The position of the servo is changed by sending pulses of different width.
- The servo looks at the pulsewidth and turns as desired.
- The servo gets power separately.
- With a servo, the pulse width modulation is not controlling the amount of energy fed to the servo
- With a servo, pulse width modulation is rather a communication protocol.

Adding the servo to pin 12



Testing the servo

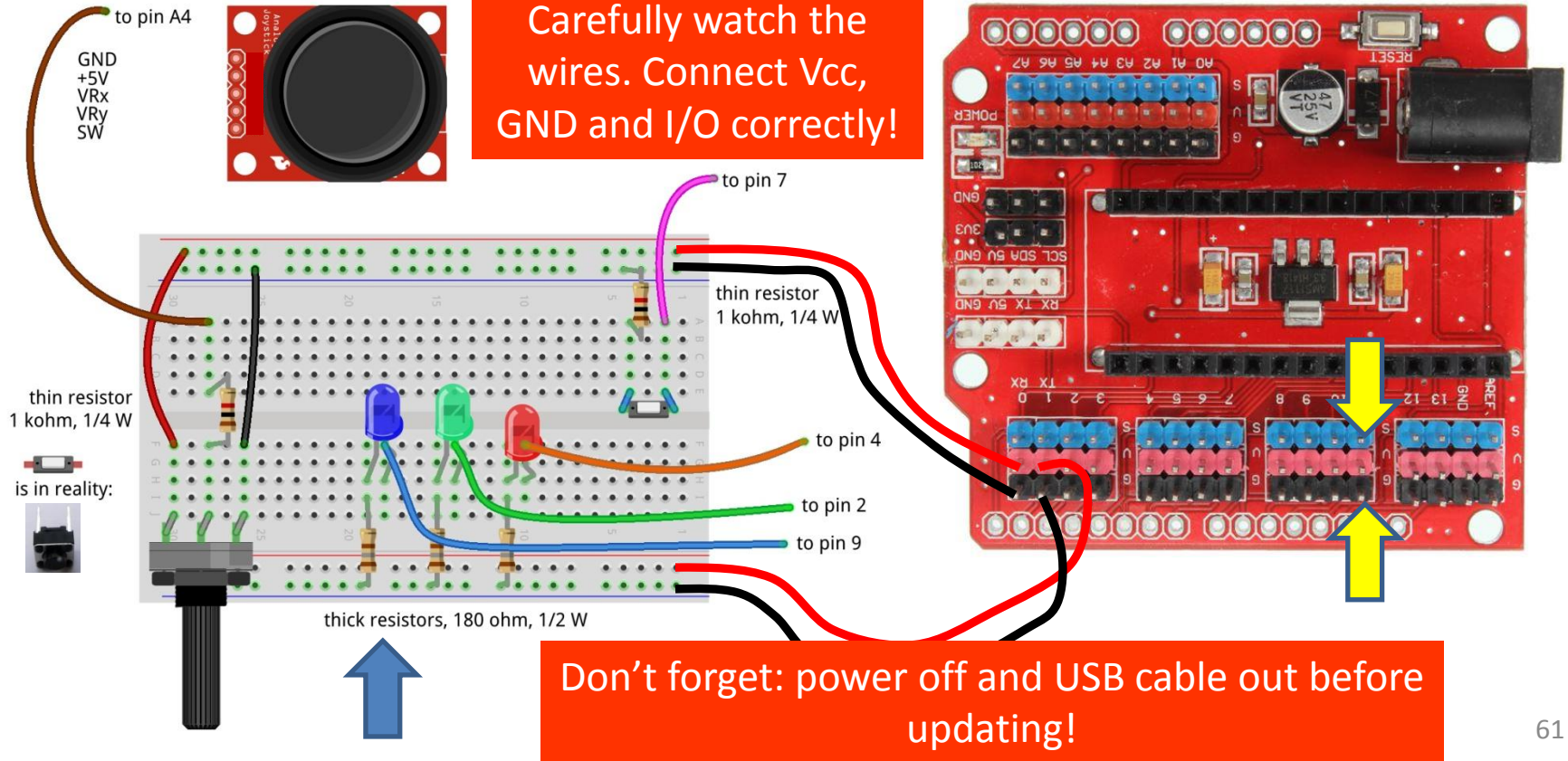
- Reconnect and repower the board
- In Scratch, create a variable *Servo 1* and make it visible.
- Try out giving it values between 0 and 100 via the slider on the variable.
 - The servo can handle values between 0 and 180
 - Does it move?
- Update the loop where you set the *Big Blue LED* value dependent on the potmeter reading to now also set *Servo 1*.
 - But watch out: values have to be between 0 and 180.
 - To avoid overflow: first divide by 1024, then multiply by 180.
- You may observe jitter.
 - Because the values of the potmeter will drift a bit and there will be power fluctuations, the reading of the potmeter will not be constant.
 - We will in the advanced workshop show how to deal with jitter



Controlling a buzzer with PWM

- The buzzer will sound if it gets a signal in an audible frequency range.
- A PWM signal on Arduino gives ca. 800 Hz.
 - Different for different pins
- You will see that there is not much influence by changing the duty cycle.

Connect the buzzer to pin 11



Testing the buzzer

- No need to restart scratchClient, because the config file was not changed.
- Define a variable *Buzzer* in Scratch
- Give it values between 0 and 100
 - Using the slider on the variable
 - Does the value have much impact?
- Add the setting of the buzzer to the loop where you already set the servo and the Big Blue LED.
 - Like with the LED, the values must be between 0 and 255.
- Only test a short time (to save the ears of your neighbors 😊).

PWM limitations of Arduino (Nano and Uno)

- General PWM is available on pins 3, 5, 6, 9, 10, 11
- Servo can be configured on those pins, but also on 2, 4, 7, 8, 12
- If a servo is configured on any pin, pins 9 and 10 cannot be configured as PWM anymore.
 - The config tool will warn you if you do it wrongly.

Part 9: Make a Scratch program

Extra material in case you have time left

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
- 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 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
 - heppg@web.de

More information

- All workshop material
 - www.github.com and search for *Weekendschool* or for *PiAndMore*
- scratchClient
 - 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):

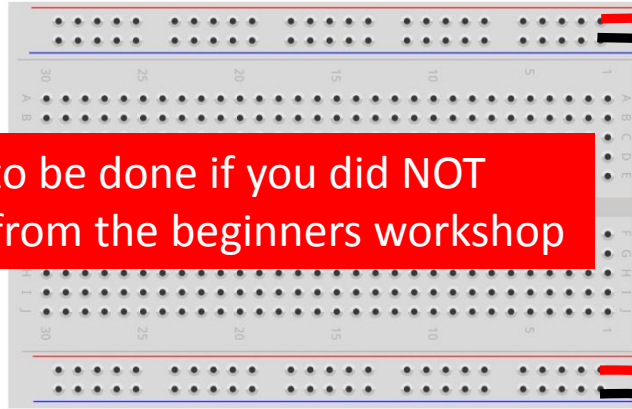
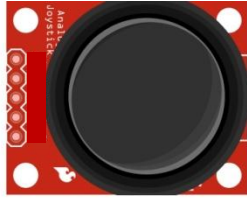
```
#!/bin/bash  
sudo python ~/scratchClient/src/scratchClient.py -c  
~/Desktop/PiAndMore/PiAndMore.xml  
pause -p "Press Enter to continue"
```

- 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

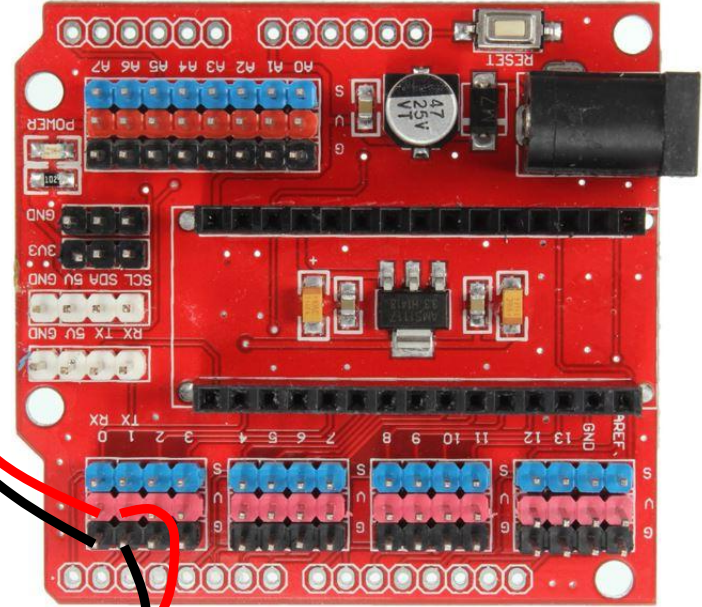
Connect the power wires

GND
+5V
VRx
VRy
SW



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

fritzing



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

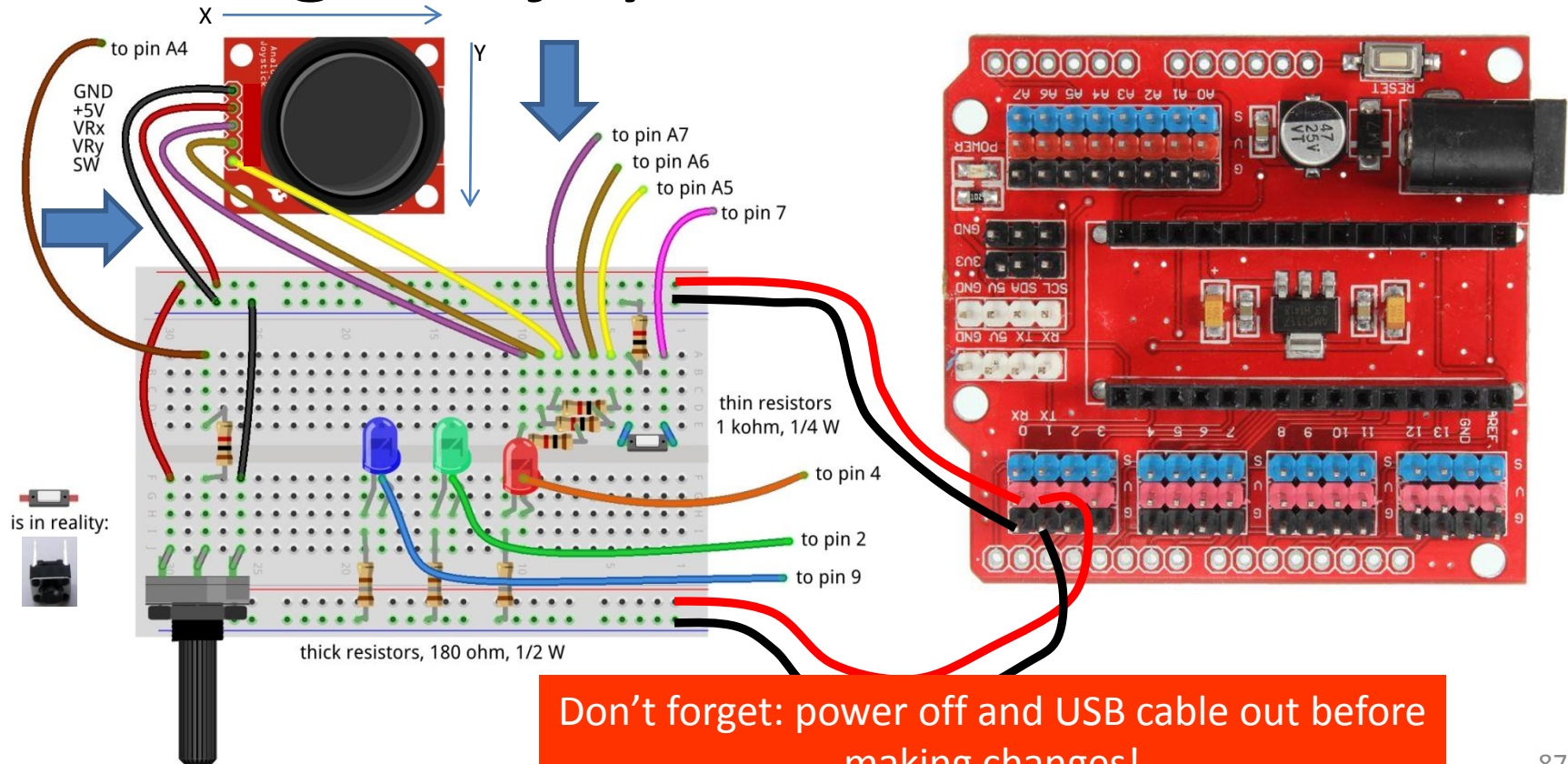
- Save the config file
- (Re)start scratchClient
- In Scratch, create a variable for each of the *out* parameters.
 - Note: variable names are case sensitive.
- In Scratch, make all of the sensors visible, so all *in* parameters.
 - Will only work if scratchClient successfully communicates to the board.

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.: $LED = LED + (JoystickX - 512) / 200$ (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

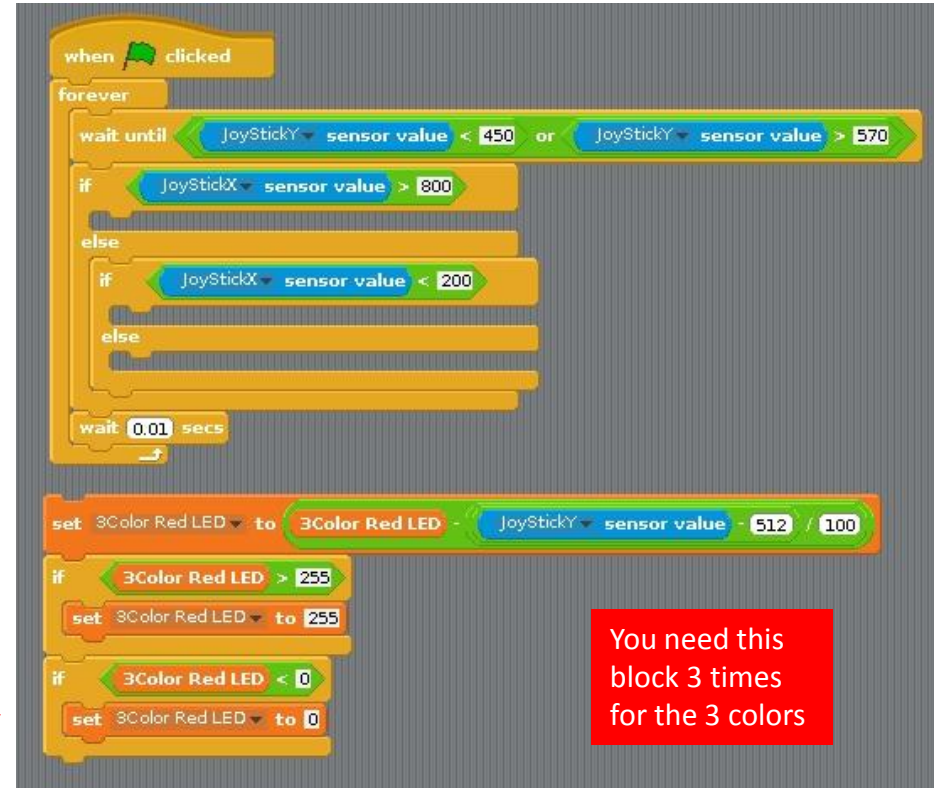
Control a 3-color LED

- 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.

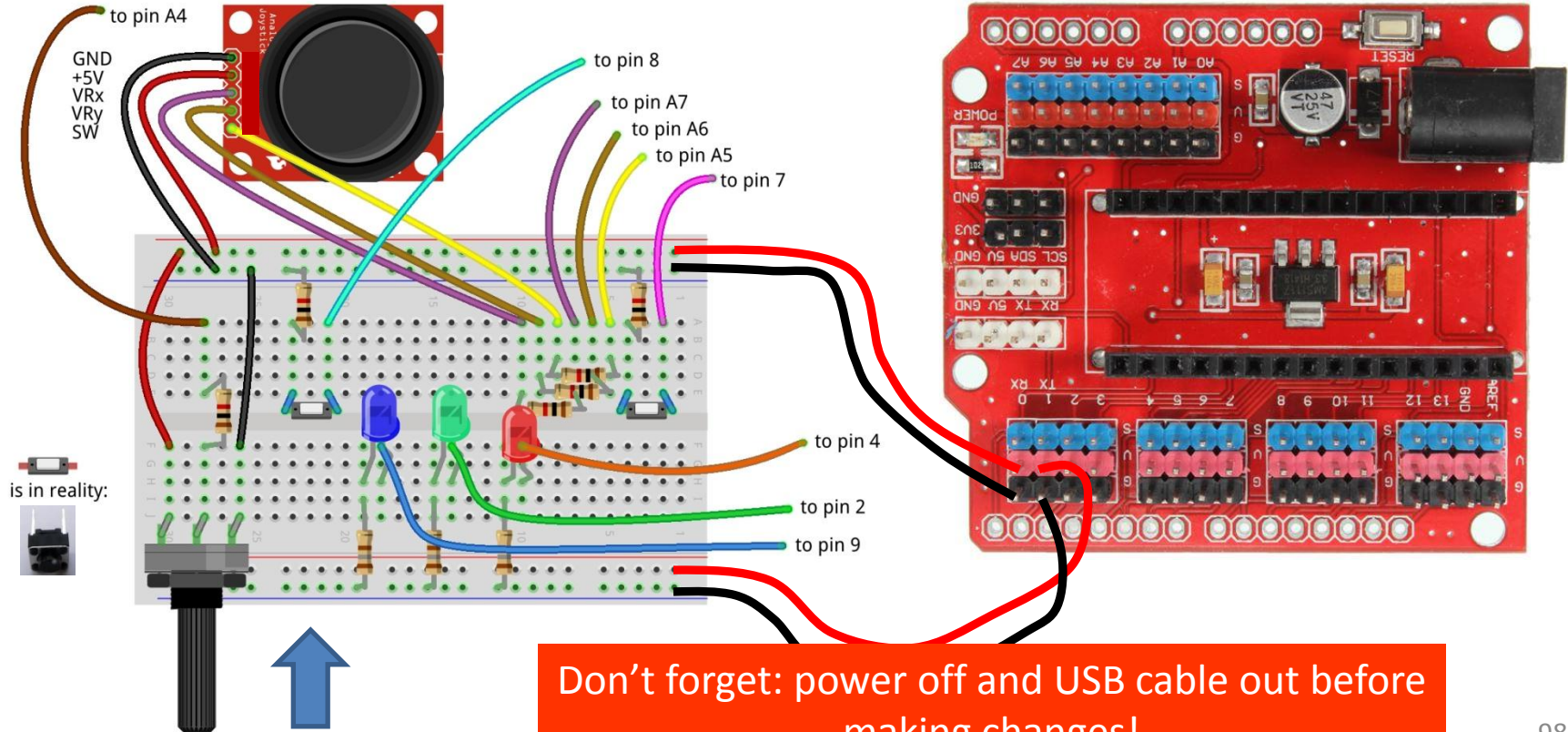
Counter function using a button

- 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 be processed as a single count
 - So no need for capacitors to do debouncing

Debouncing

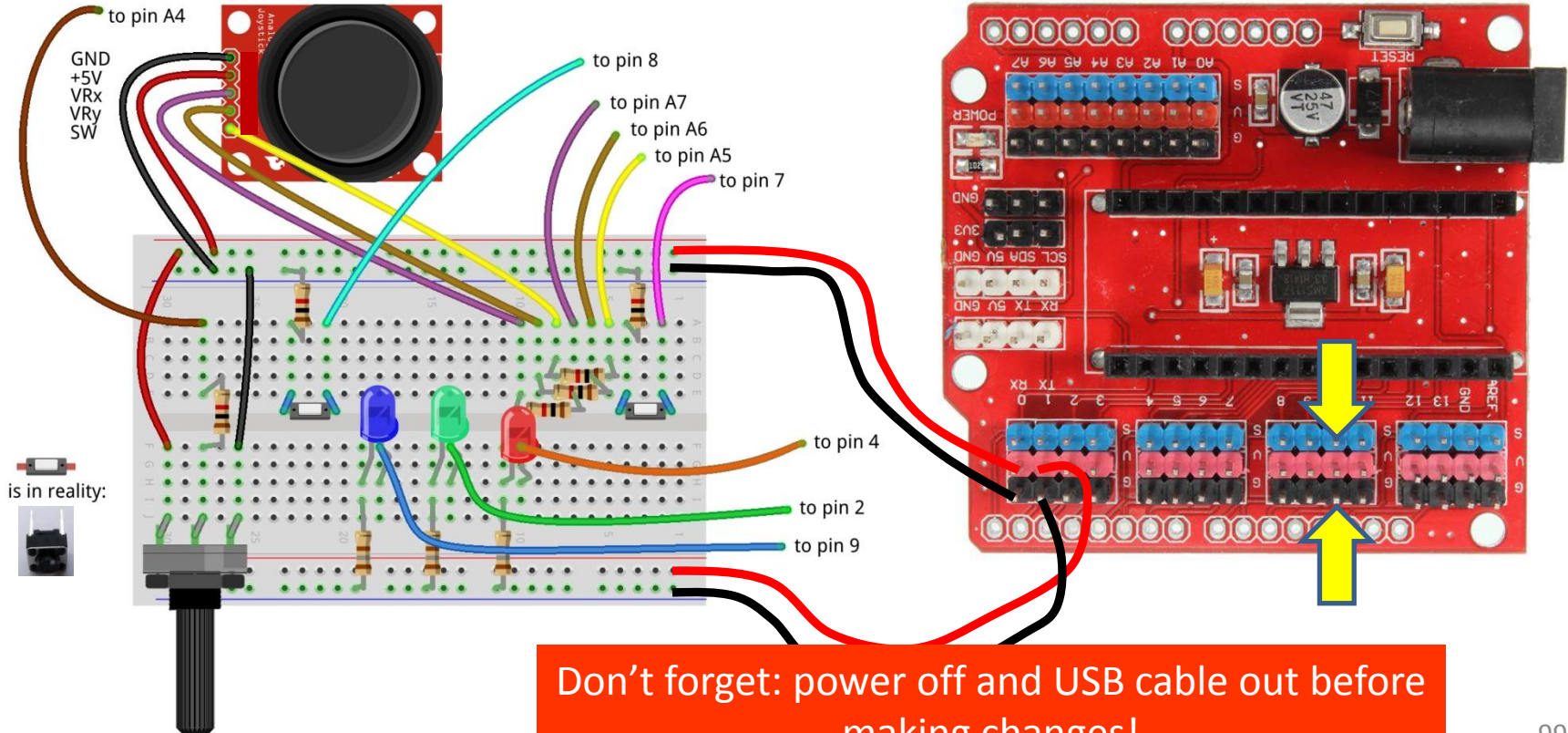
- If a mechanical switch is operated, it can generate a series of pulses rather than one.
- This is because one of the metal pieces bounces back a few times before it stays in position.
- You would normally need take care of this if you want to get a single pulse
 - E.g. add a capacitor.
- scratchClient 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!



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.

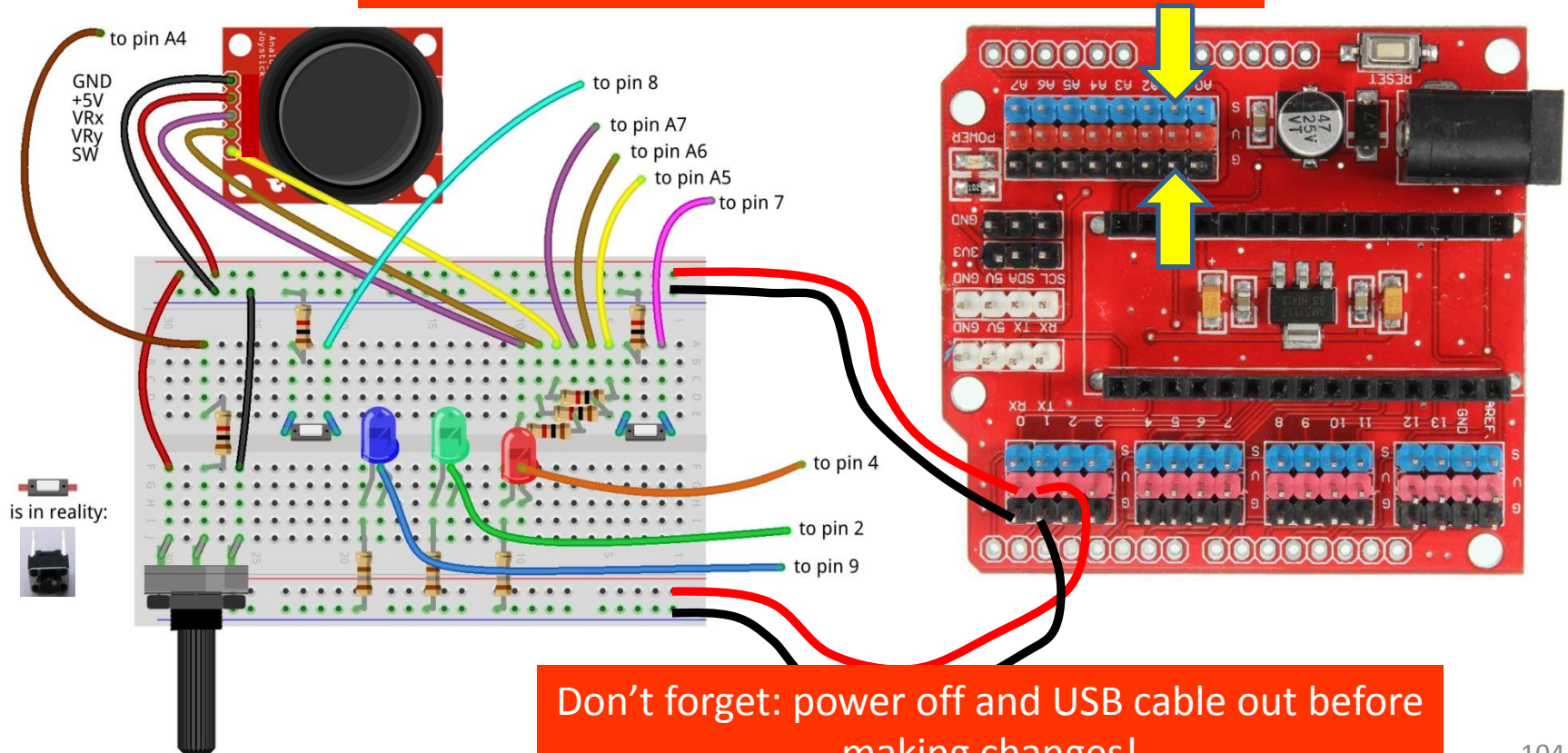
Tilt sensor

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!



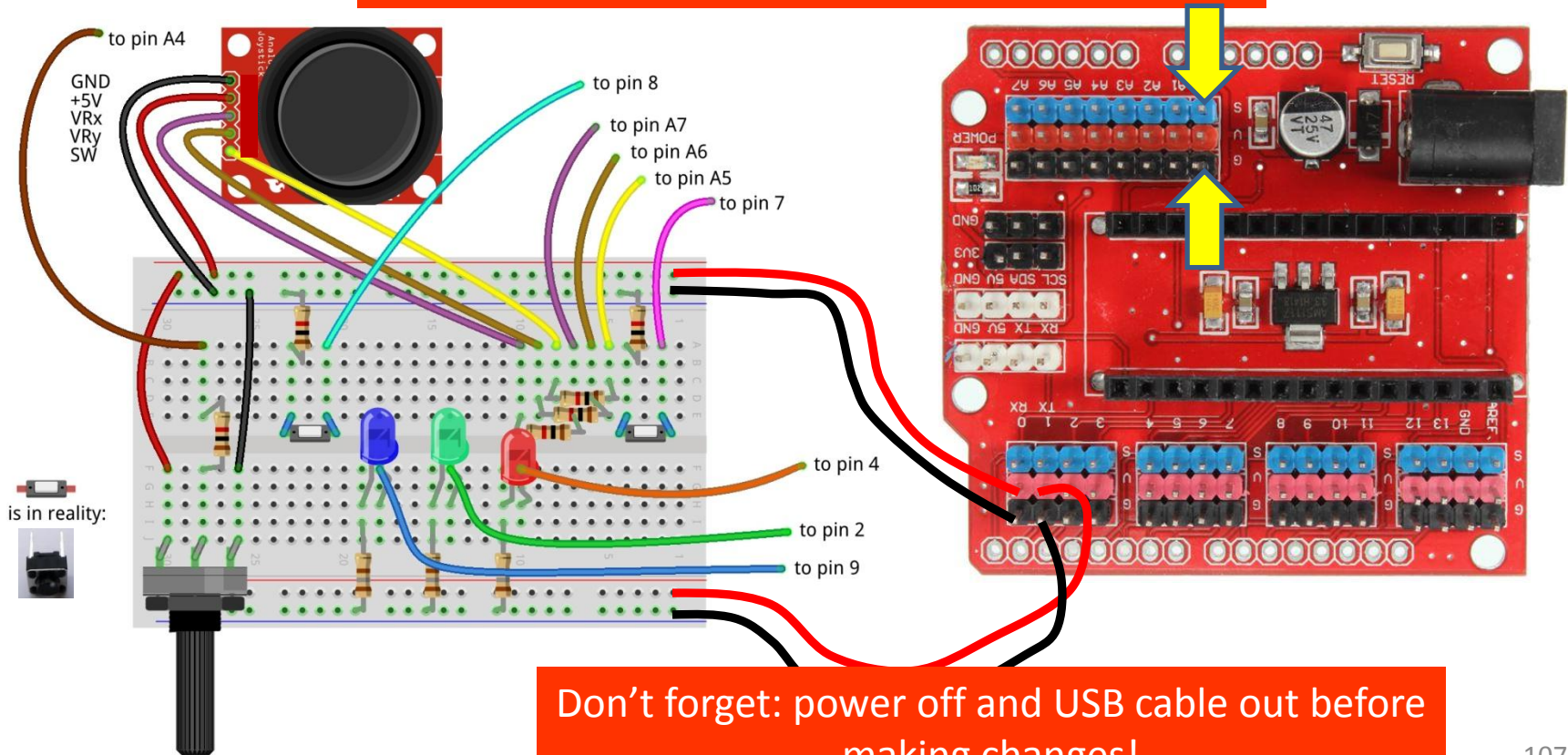
Rain Sensor

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!



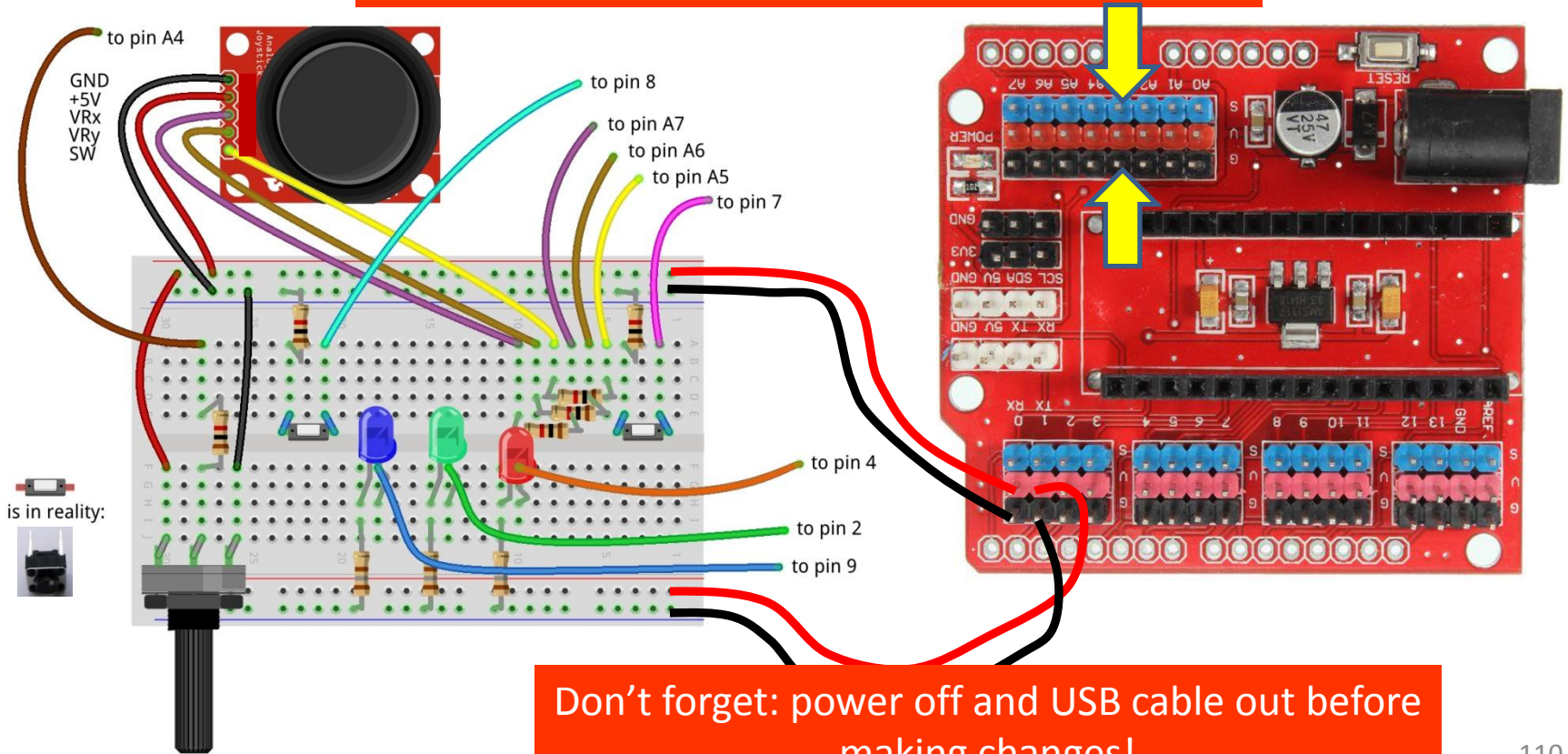
Sound sensor

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

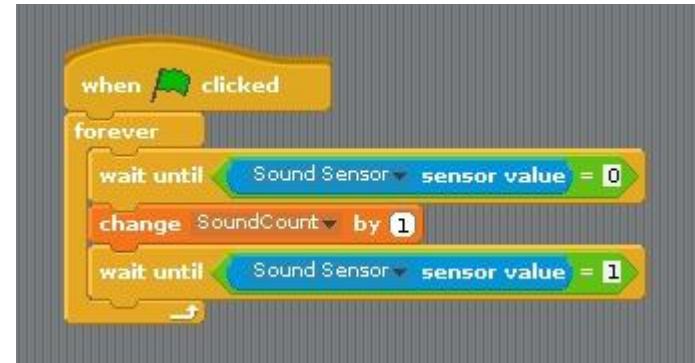
Carefully follow the wires to connect correctly!



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

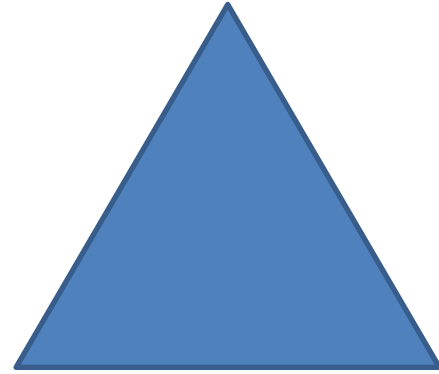
- Running out of pins on one Arduino? Connect more of them!
- The `/dev/ttyUSBnn` port in the config file will define which part of the configuration is used for which Arduino.
- Using the `ident` functionality (see previous topic) can be used to check that the correct Arduino was connected to the intended 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/ttyUSB**1**).

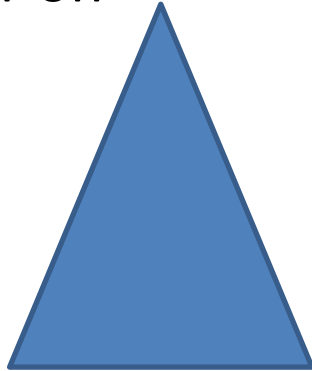
Trying it out – merge and config

- 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/ttyUSB1`.
- Disconnect both boards.
- Restart scratchClient with the merged file.
- Connect the workshop board first. It will become `/dev/ttyUSB0`.
- Connect the Weekendschool board second. It will become `/dev/ttyUSB1`.



Trying it out – Control from Scratch

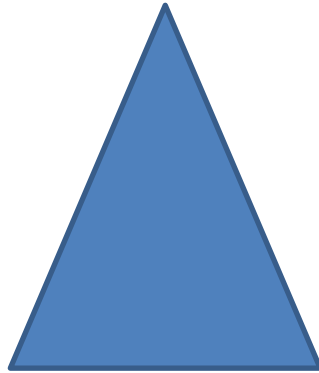
- In the Scratch program of the workshop, define these variables
 - Tiltservo
 - Panservo
- Try giving them values between 0 and 100 with the slider on the variable.
- Does the duck tilt and pan?



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

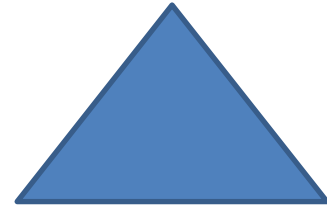


Power on self test (POST)

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?

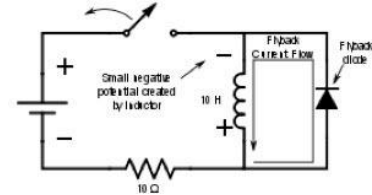
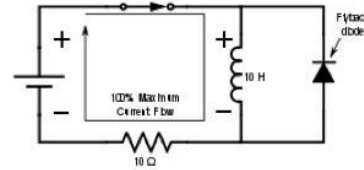
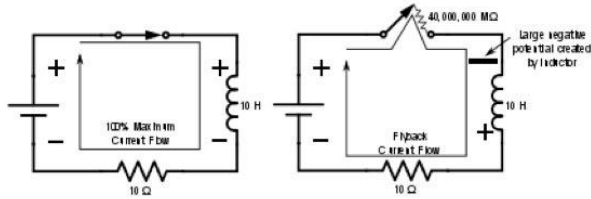


- You will need to address the pin numbers directly. You **cannot** make use of the config file of scratchClient.
- See the Arduino sketch of the Weekendschool board.
 - Between the xxxx and yyyy markers
- Now try on the PiAndMore board to lite 3-color LED
 - 0.5 seconds Red
 - 0.5 seconds Green
 - 0.5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



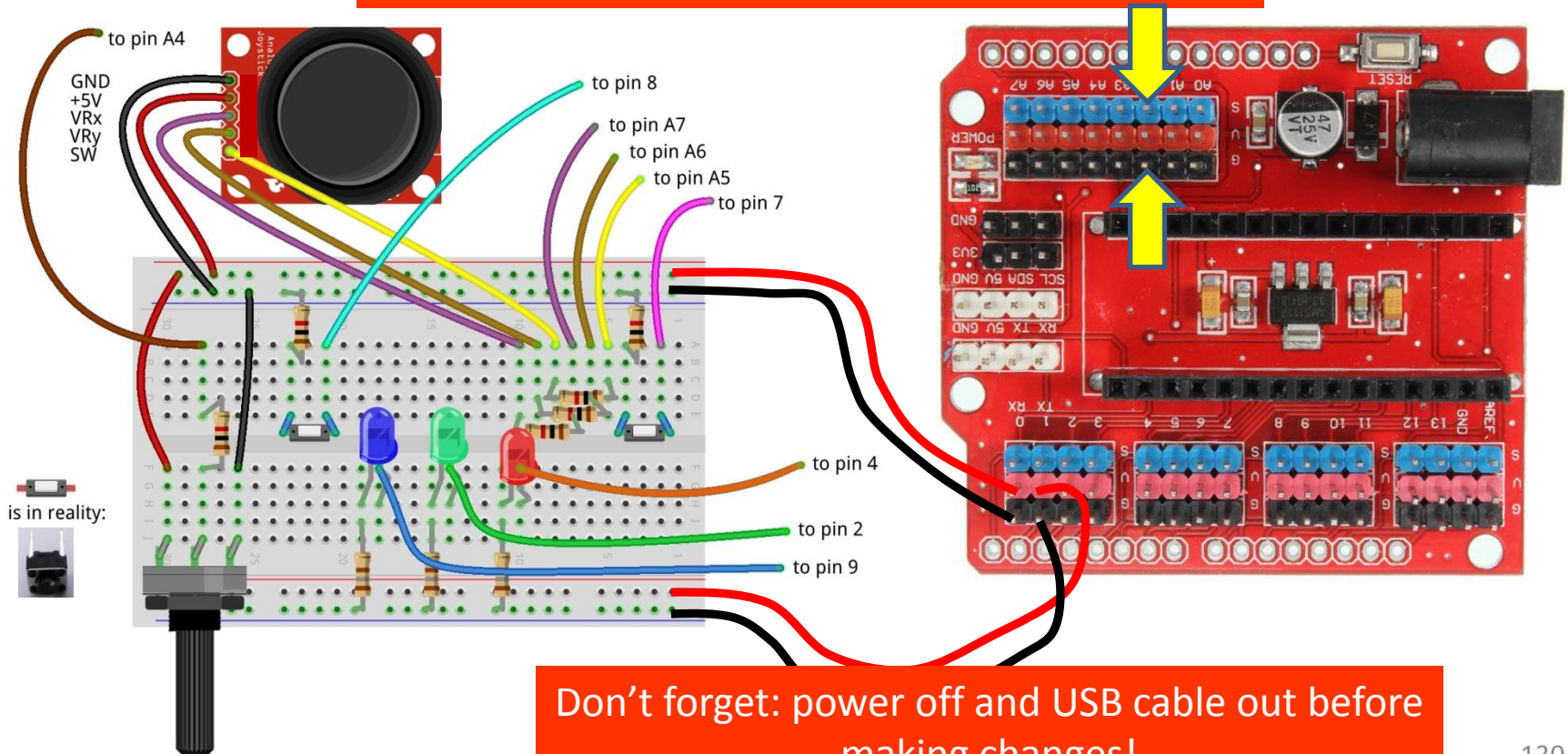
- At least a flyback diode should be used.

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
- 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
 - heppg@web.de

More information

- All workshop material
 - www.github.com and search for *Weekendschool* or for *PiAndMore*
- scratchClient
 - 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

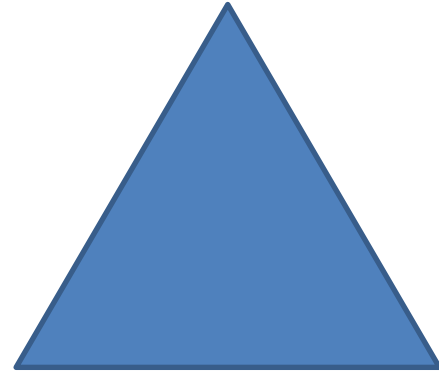
Maximum current per pin

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

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