Physical computing: control servos, LEDs and more from Scratch using RPi, Arduino, scratchClient

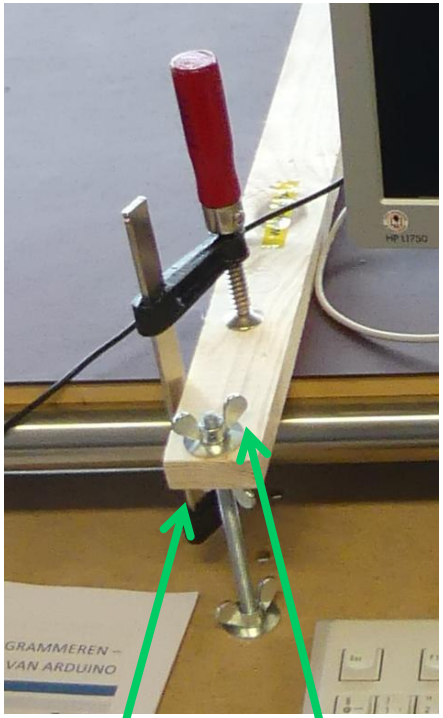
Hans de Jong

Pi And More 9½

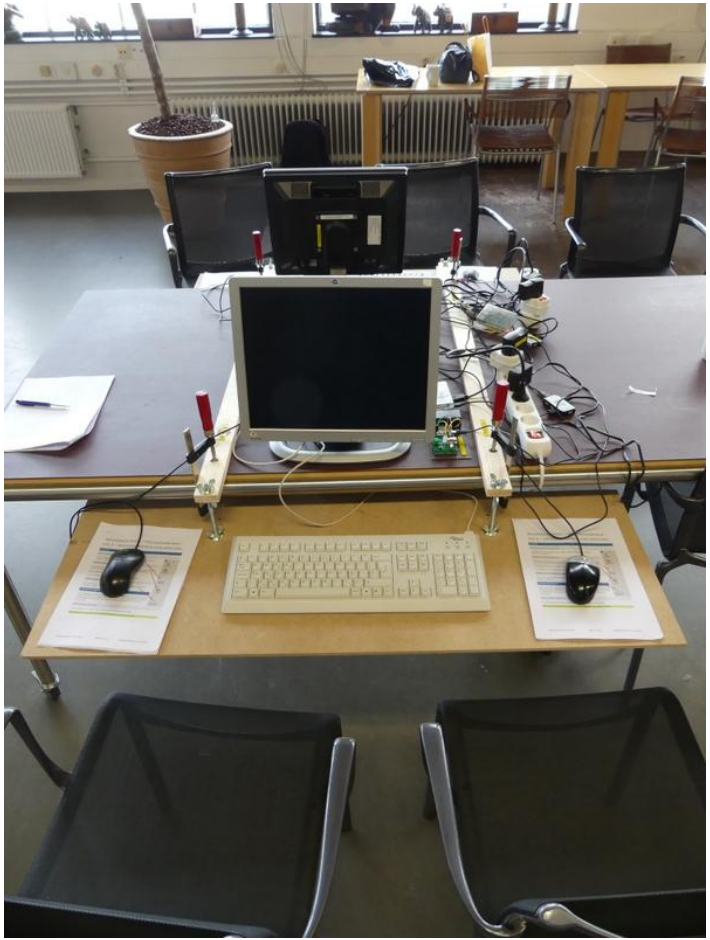
Krefeld – 14 January 2017

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.



Hans de Jong

π++

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 available.
- At the end copy the desktop to your USB stick (if you want)
- Cleanup.

Hans de Jong

- The major steps:
  1. Get a working hardware and scratchClient config using a Scratch test program.
  2. Update scratchClient config (moving to other GPIO pins)
  3. Write your own Scratch program to access the board.
  4. Add more hardware (buzzer, tri-color LED, button, joy stick) and adapt scratchClient config and adapt the Scratch program.
  5. If you want: have a look at the game used at the Weekendschool and program a bit yourself (but it is still in Dutch ☺)

  Start latest 30 minutes before the end of the workshop.

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
  - Understand what all the resistors are for
  - Be able to configure and run scratchClient
  - Program Scratch to control the physical input and output
- If time permits / as you desire: look at a game using servos, buttons, LEDs etc. as used on the Weekendschools in The Netherlands.
- **Have fun!**

Hans de Jong

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.

Hans de Jong

Lessons at the Weekendschool

- I personally teach
  - Mathematics (since 12 years)
  - Physics (since some 6 years)
  - Programming (started in 2016)
  - Electro (started in 2016)
- Programming: 3 lessons on 3 Sundays with Scratch, Raspberry Pi and Arduino
- Lesson 2: Physical computing: https://youtu.be/Qo1gnXNzhqE

Only a few rules

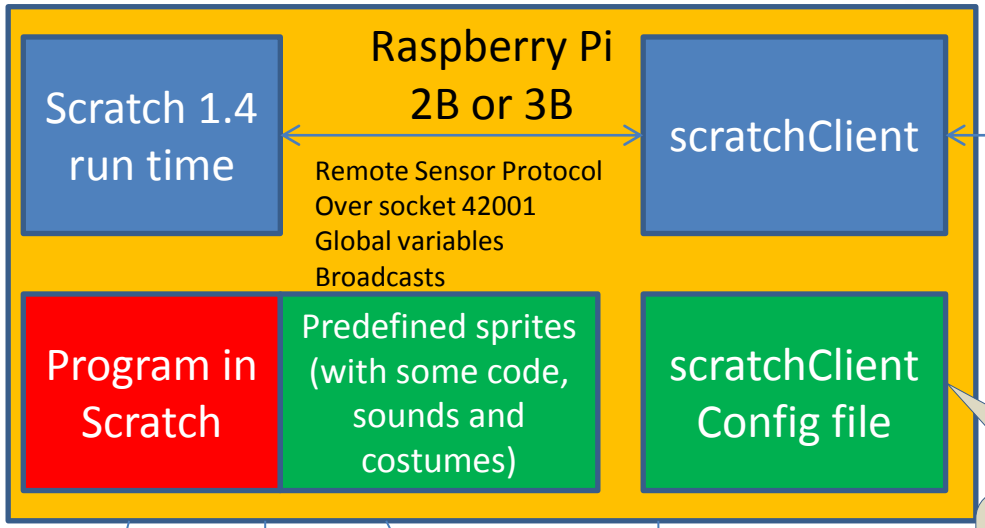
- If Raspberry Pi needs to be rebooted, it needs a powercycle. Please do not pull plugs but use the switch in the outlet.
- Always put a resistor in series with the components
  - If you think there is no need then please tell us and we will explain what the reason is (with one exception).

- When changing the wiring
  1. Stop scratchClient (just close the window)
  2. Detach the USB cable from the Arduino Nano
  3. Switch off the 9V power
  4. Check, double check and check again whether the wiring is correct. You may blow up components when wiring wrongly!
  5. Make sure that you ***both*** (4 eyes principle) are convinced the wiring is OK before turning on power again.

- If something breaks down or gets damaged: we have some spare material
  - Do ***not*** put anything that is broken back into the box please.

Hans de Jong

The setup



Powered USB hub

(not strictly needed since Arduino is powered separately – used for protection of RPi)

scratchClient Sketch

Arduino Nano

Defines function of the GPIO pins on the Arduino (and RPi) and gives them logical names.

Standard setup

Programmed by the students

Prepared by the teacher or expert

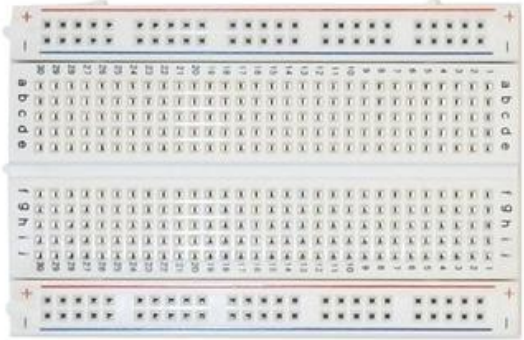
Hans de Jong

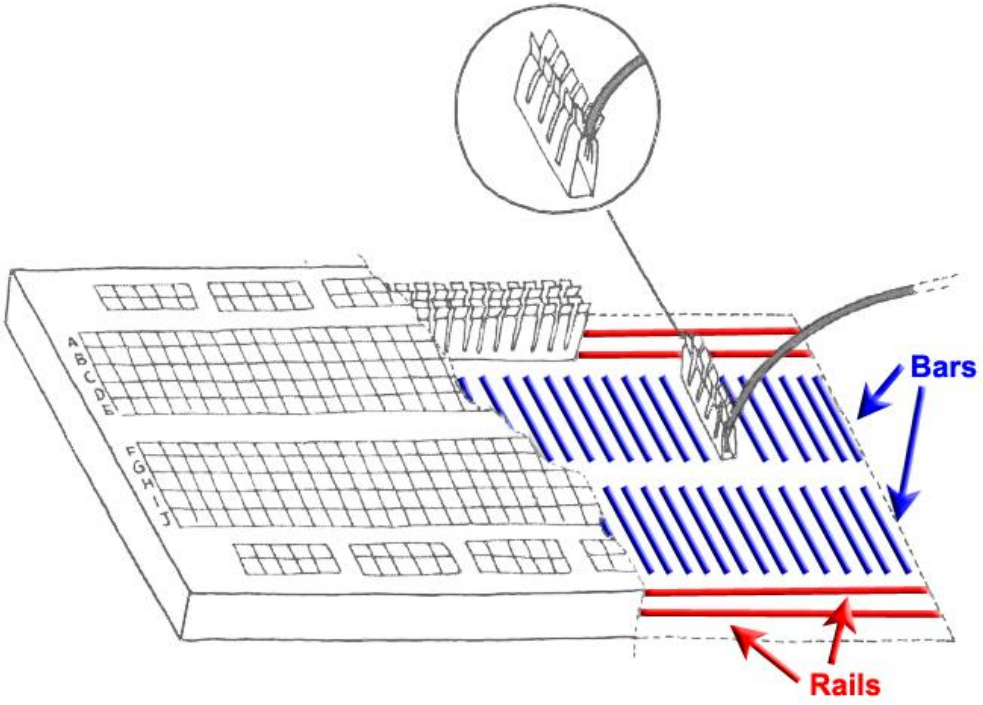
Where is the presentation?

- You will find this presentation = the instruction at your desktop in *PiAndMore/Part-1--Breadboard*
- If you already know the stuff and get bored then feel free to work on your own
  - Silently please … ☺

Breadboard

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





Hans de Jong

π++

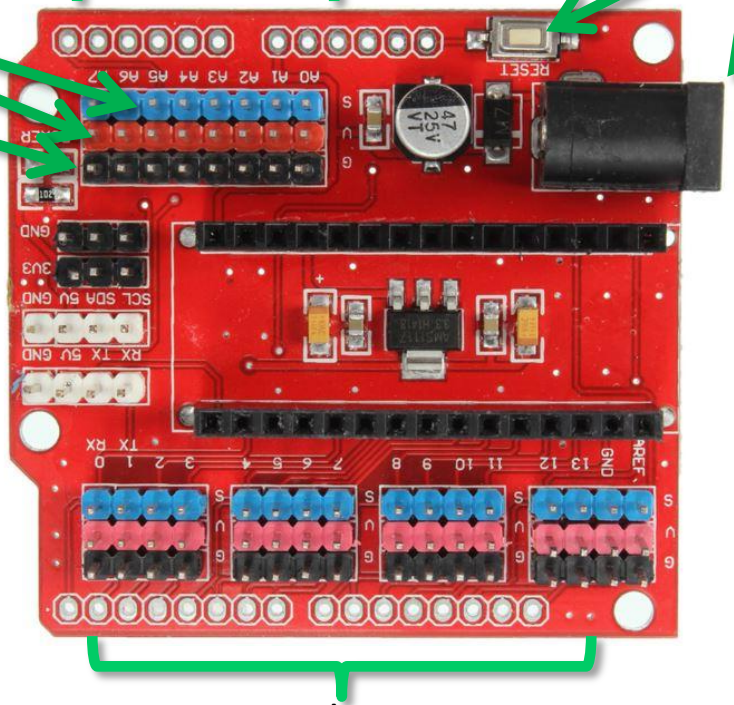
Looking at the Arduino Nano extension board

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

Reset button

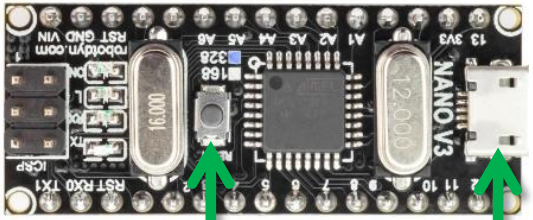
9V power socket

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

Hans de Jong

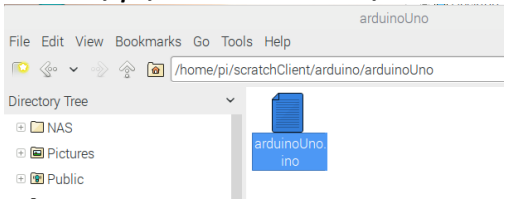
π++

Preparing for programming the Arduino Nano

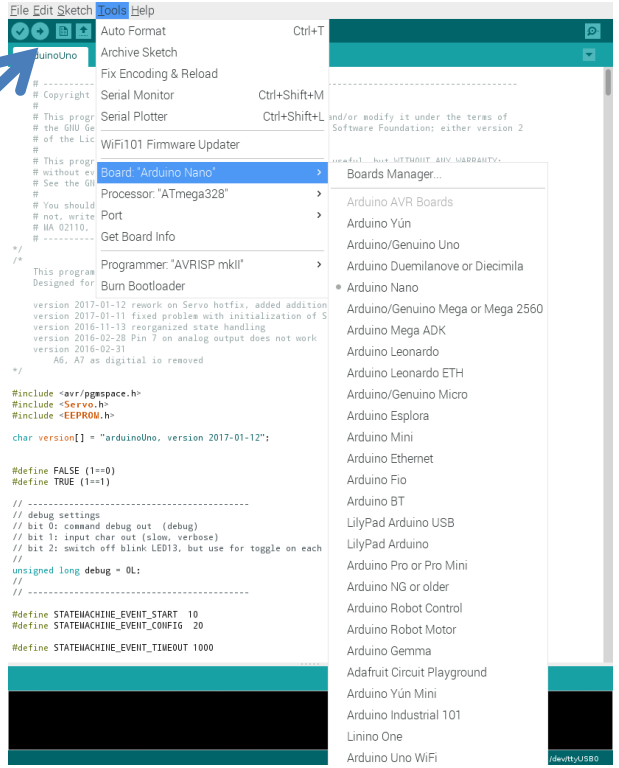
- Connect the Arduino Nano
  - Make sure to connect directly
    - not via the USB hub
    - for strange reasons it will not work although it used to in previous releases and still should.

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*

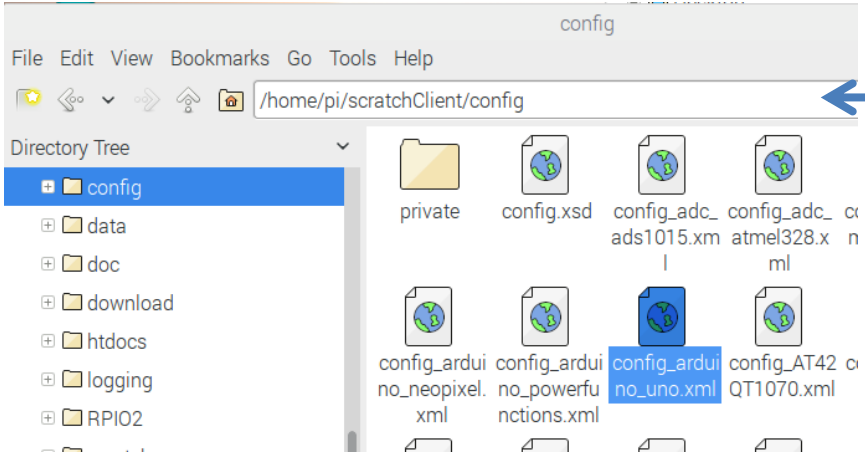


- Double click to open the Arduino IDE
- Click *Tools* and make sure that these are set:
  - Board: Aruino Nano
  - Processor Atmega328
  - Port: the port where the Arduino Nano is connected
- Click on the Upload button.
- Wait till the completion of the Upload is reported.



Hans de Jong

Open the scratchClient config file



/home/pi/scratchClient/config

Note: it is called config_arduino_***uno***.xml, however it also is for the Nano.

Hans de Jong

Looking at the scratchClient config file

```
<!-- digital input values are inputs for the adapter (but outputs for arduino) -->

<input_value name='inputD4'>
    <variable name='redLED' />
</input_value>

<input_value name='inputD7'>
    <variable name='buttonLED' />
</input_value>

<input_value name='pwmD5'>
    <variable name='greenLED_PWM' />
</input_value>

<input_value name='servoD6'>
    <variable name='servoD6' />
</input_value>
<input_value name='servoD11'>
    <variable name='servoD11' />
</input_value>


<input_value name='command'>
    <variable name='command' />
</input_value>

<output_value name='outputD3'>
    <sensor name='button_3' />
</output_value>
```

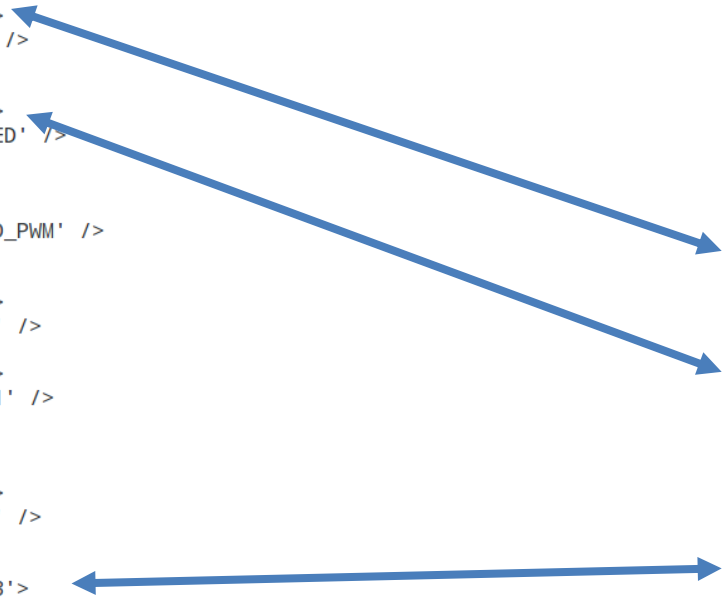
```
-- ================================================= -->
    <!-- this is the setup for an UNO arduino -->

-- digital io lines -->
-- @dir = void, pwm, in, out, servo -->
-- @pullup= on -->

-- do not use D0, D1 (serial  lines)  -->

    <io id='D0' dir='void' />
    <io id='D1' dir='void' />

    <!--  3,5,6,10,11 may be pwm -->
    <io id='D2' dir='out' />
    <io id='D3' dir='in' pullup='on' />

    <io id='D4' dir='out' />
    <io id='D5' dir='pwm' />

    <io id='D6' dir='servo' />
    <io id='D7' dir='out' />

    <io id='D8' dir='in' pullup='on' />
    <io id='D9' dir='in' pullup='on' />
    <io id='D10' dir='in' pullup='on' />

    <io id='D11' dir='servo'  />

    <io id='D12' dir='in' pullup='on' />

<!--  do not use D13 (onboard LED)  -->

    <io id='D13' dir='void' />
```

Note that every definition on the left has a line on the right. Note that the directions are counter intuitive.

Hans de Jong

Let’s wire the board …

```
<!-- digital input values are inputs for the adapter (but outputs for arduino) -->

<input_value name='inputD4'>
    <variable name='redLED' />
</input_value>

<input_value name='inputD7'>
    <variable name='buttonLED' />
</input_value>

<input_value name='pwmD5'>
    <variable name='greenLED_PWM' />
</input_value>

<input_value name='servoD6'>
    <variable name='servoD6' />
</input_value>
<input_value name='servoD11'>
    <variable name='servoD11' />
</input_value>


<input_value name='command'>
    <variable name='command' />
</input_value>

<output_value name='outputD3'>
    <sensor name='button_3' />
</output_value>
```

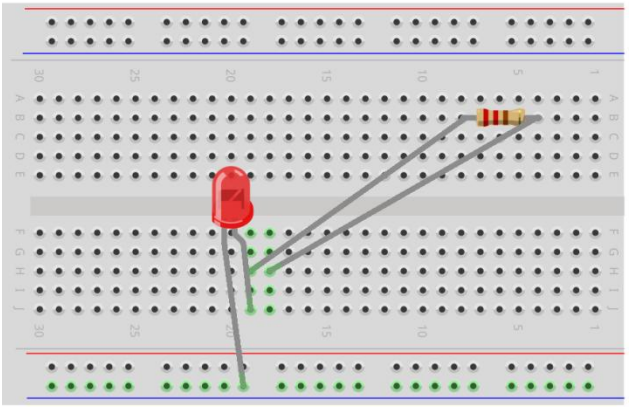


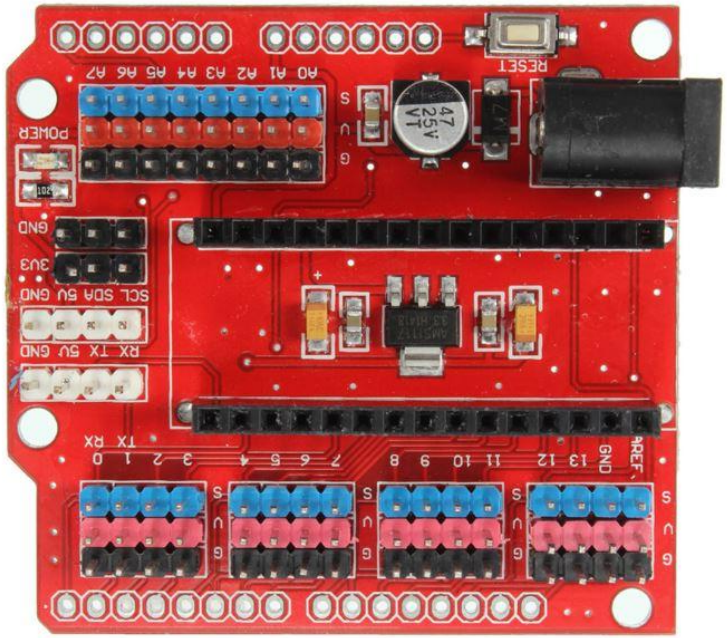
See diagram on the next page

- Wire +5V to the + rail at the bottom
- Wire GND to the – rail at the bottom
- Wire a red LED to pin 4.
  - Pin 4 → 200 Ohm resistor → GND (minus)
- Wire another LED (we use blue) to pin 7.
  - Pin 7 → 200 Ohm resistor → - (minus) rail
- Wire a button to pin 3.
  - Pin 3 → 1 kOhm resistor → GND (minus)

Hans de Jong

Put the board in front of you in this way



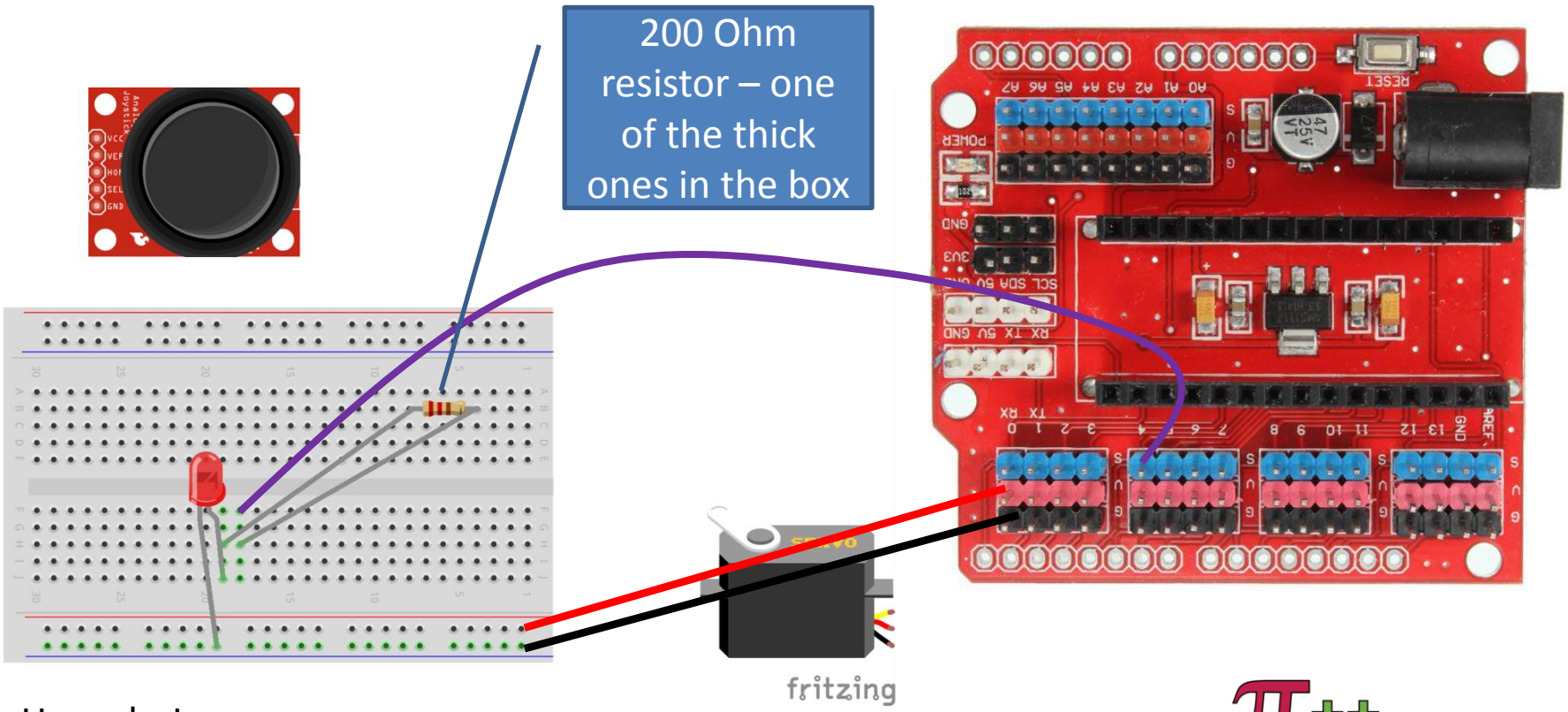


fritzing

Hans de Jong

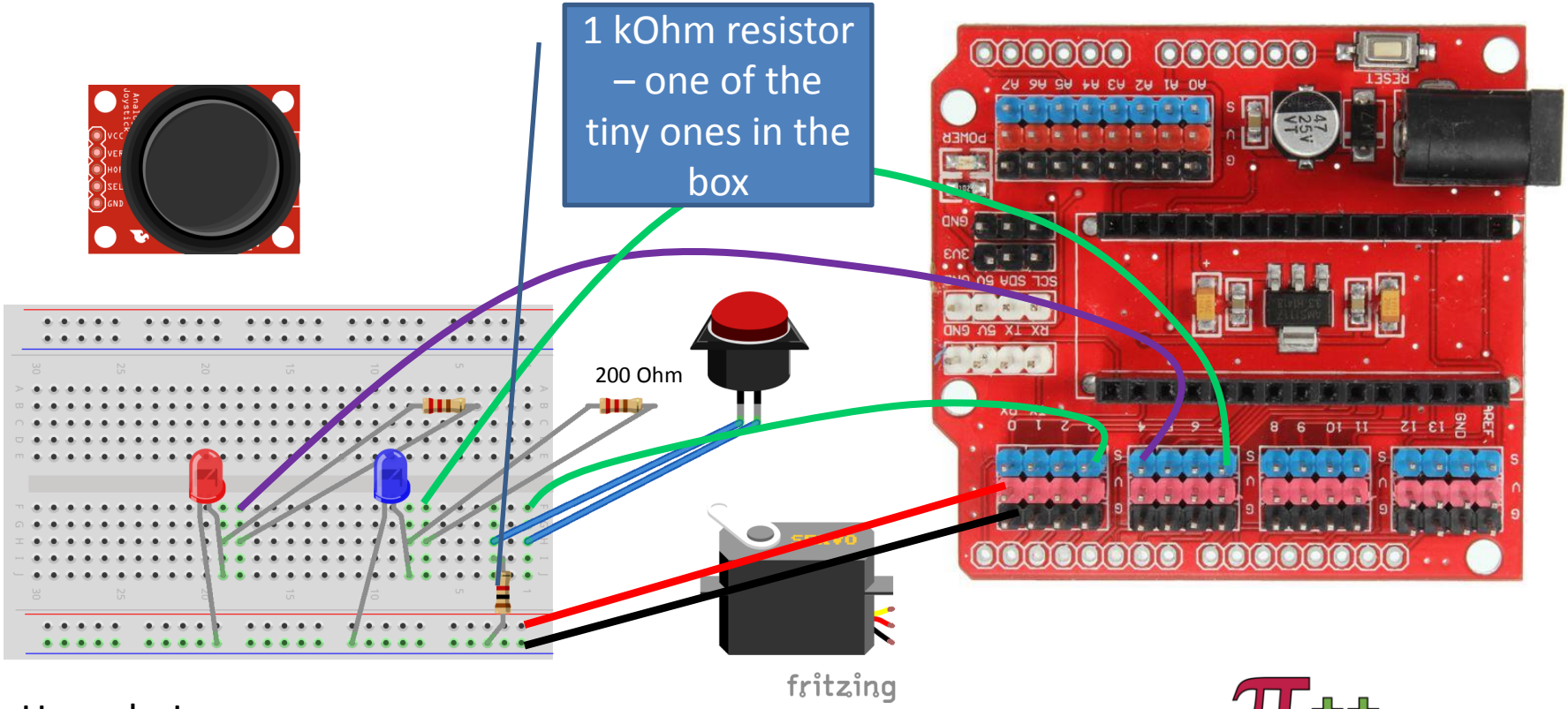
π++

First step: red LED and + and -



Hans de Jong

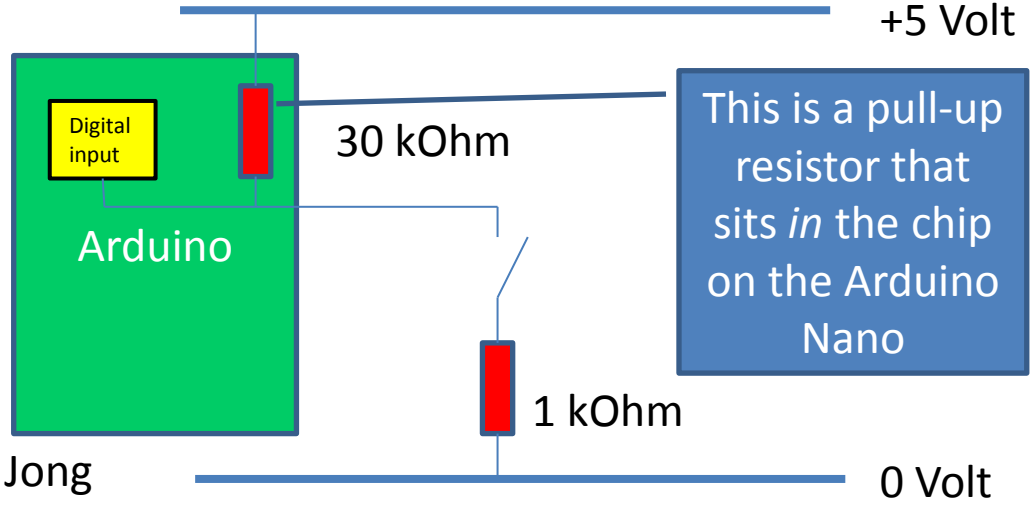
Second step: add button and blue LED



Hans de Jong

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 there 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 that the wiring is correct.

Make a script file to start scratchClient

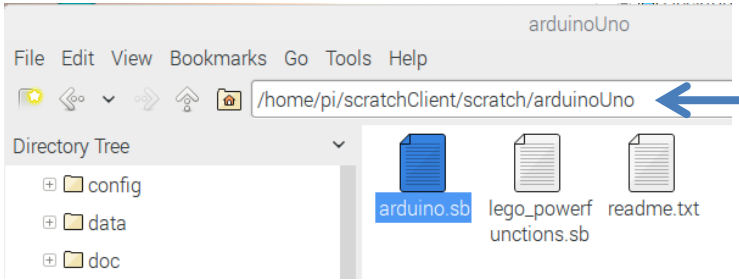
- Copy the config file to the desktop.
  - We will later modify it and want to keep the original.
- Then make a new file on the desktop
  - Call it `StartSC_First_Step.bash`
- Put this in:

```
!# /bin/bash
sudo python ~/scratchClient/src/scratchClient.py -c
  ~/Desktop/config_PiAndMore_ttyUSB0.xml
```

- Make the file executable.
- Do you understand what the file does?
  - If not, please ask

Bringing things together

- Connect the 9 Volt connector to the board
- Connect the USB connector to the Arduino
- Start scratchClient with the script you just made (e.g. double click it and when asked choose *Execute in Terminal*.
- Start the Arduino Scratch test. This is in:



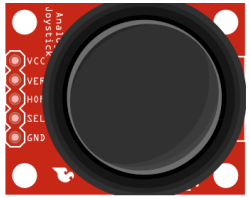
/home/pi/scratchClient/scratch/arduinoUno

- Press the green flag to start the Scratch program.

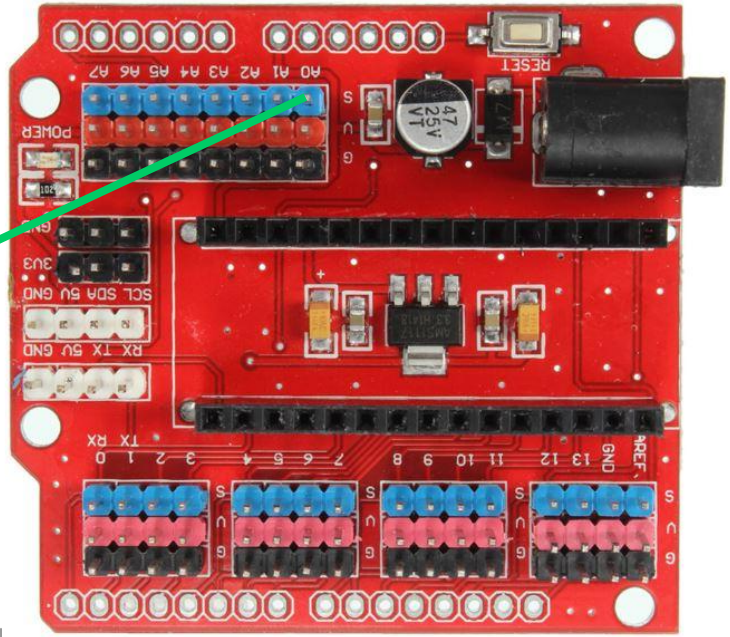
Hans de Jong

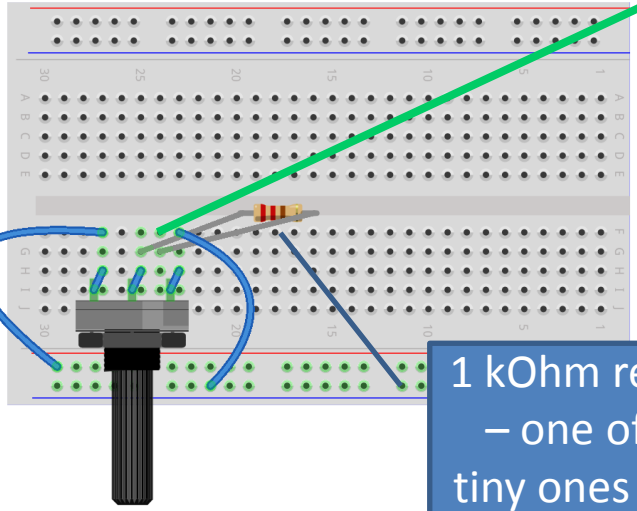
Does it work?

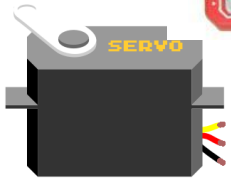
Analog input



Note: only the additional wires are shown.



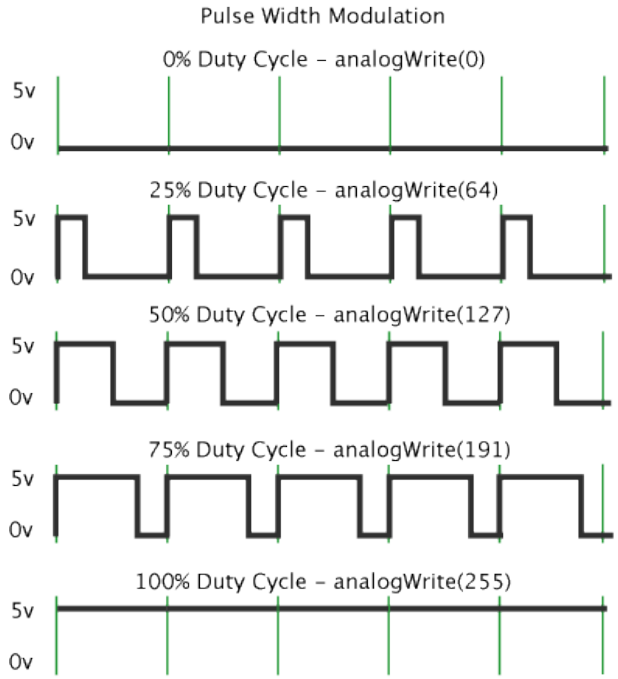




1 kOhm resistor – one of the tiny ones in the box

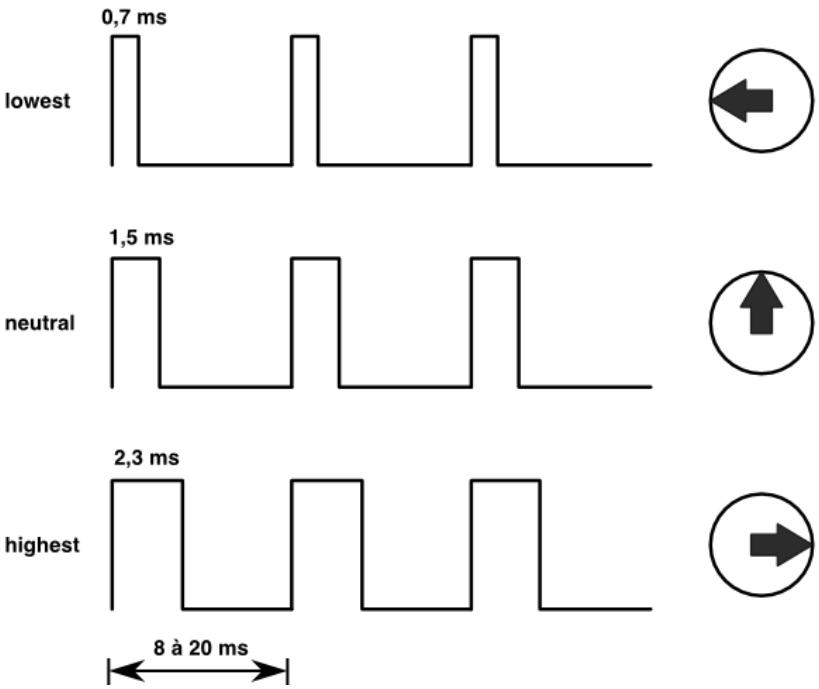
fritzing

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.

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.

Hans de Jong

Dimming a LED with PWM

- Put the green LED in the right place.

Hans de Jong

More information

- This presentation, source files of the game and 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/

Start of the advance material – not worked out.

Add a Joystick

- The joy stick consists of
  - Potentiometer for X
  - Potentiometer for Y
  - Button
- Wire these 3 signals via a 1 kOhm resistor on the breadboard to pins A1 and A2 for X and Y and the button to one of the digital porst
- Adapt the scratchClient config
- Once it runs, you can see values for the potmeter as sensors in Scratch.