PAPER BASED VISCOSITYMETER

LABO: MICRO- & NANOTECHNOLOGIE

Datum labo: [05/01/2019]

Student 1: **Jordi Jaspers** Groep: **EA – ICT 1**

Student 2: **Simon Aerts** Docent: **Jilles Oudenbrouckx**

Contents

[1. Inleiding 2](#_Toc534489126)

[**Doel van de labozitting:** 2](#_Toc534489127)

[**Benodigde componenten:** 2](#_Toc534489128)

[2. proefOpstellingen & Resultaten 3](#_Toc534489129)

[**Proef 1: Viscositeitsmeter** 3](#_Toc534489130)

[**Opstelling:** 3](#_Toc534489131)

[**Uitvoering:** 4](#_Toc534489132)

[**Metingen:** 4](#_Toc534489133)

[3. Besluit 5](#_Toc534489134)

[4. BIjlagen 6](#_Toc534489135)

[**Bijlage 1: Omzetten van de CSV naar een plotgrafiek.** 6](#_Toc534489136)

[**Bijlage 2: Inlezen van de waarde.** 12](#_Toc534489137)

[**Bijlage 3: Arduino file.** 13](#_Toc534489138)

# Inleiding

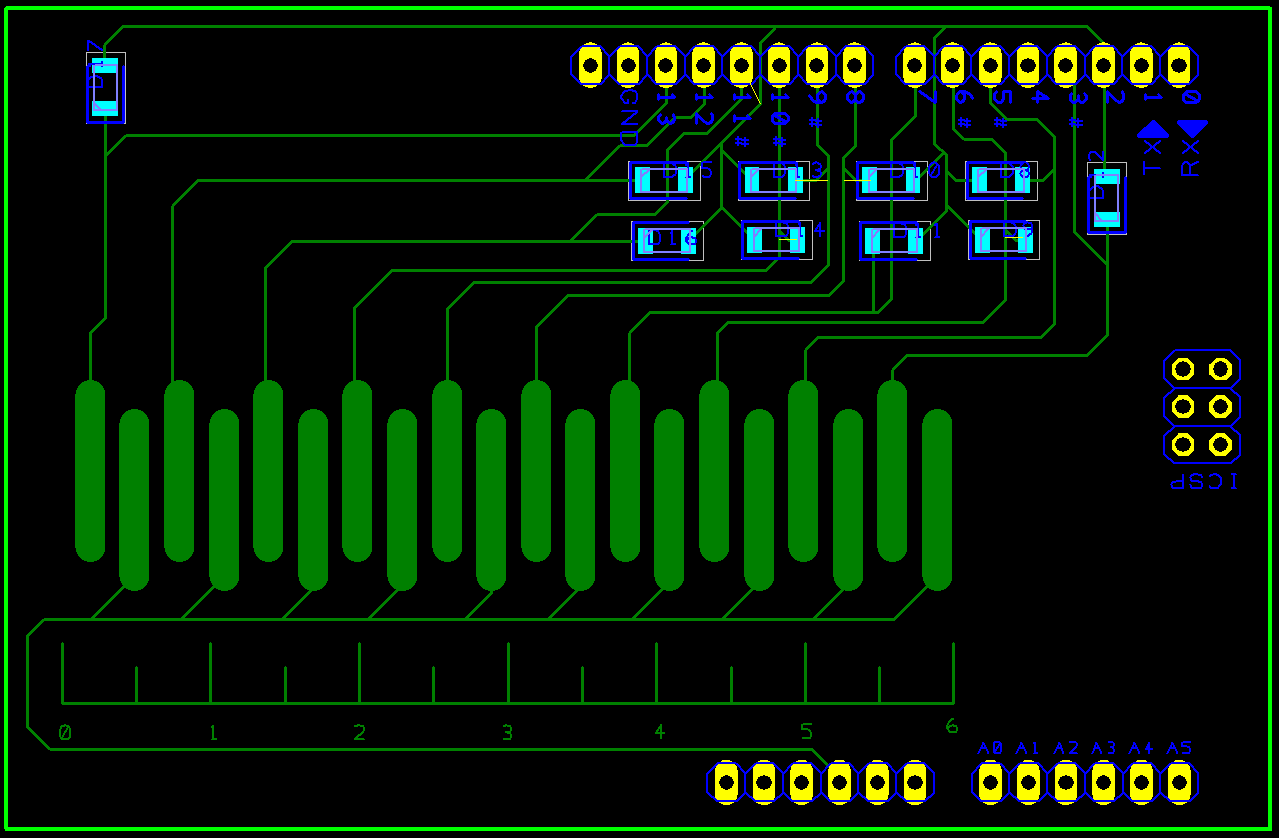
## **Doel van de labozitting:**

Doel van dit labo is om een papier gebaseerde viscositeitsmeter te maken met behulp van een Arduino. De viscositeitsmeter kan dan de stroperigheid of vloeibaarheid van verschillende vloeistoffen meten. Met bepaalde snelheidsberekeningen en eigenschappen van vloeistoffen kunnen we dit makkelijk bepalen. Met een paar sensoren, een Arduino voor de data te kunnen inlezen en een klein python programmatje die dan de gegevens makkelijk kan berekenen en omzetten tot een duidelijke grafiek.

## **Benodigde componenten:**

|  |  |
| --- | --- |
| * Arduino Uno * PCB (Figuur 1.1) * Weerstanden * Dubbelzijdige folie * Pipet | * Breadbord * Python Script * Arduino Script * Kapton tape * Vloeistoffen |

**Figuur 1.1: PCB-ontwerp**



# proefOpstellingen & Resultaten

## **Proef 1: Viscositeitsmeter**

## **Opstelling:**

Om te beginnen met het project hebben we eerst een PCB-design ontworpen die ervoor zou kunnen zorgen dat we de snelheid van de viscositeit kunnen opmeten. Dit doen we door een cap-sense meting uit te voeren op de PCB zelf. De grote oppervlakte die te zien zijn in figuur 1.1 zijn de platen voor condensator. Voor ieder paar oppervlakte is dan een weerstand verbonden om een kortsluiting tegen te gaan. De bedoeling van deze condensator is dat de vloeistof die erover gaat werkt als het medium tussen de condensators in. Zodat we een programma erna kunnen schrijven die deze verandering opvangt en kunnen noteren dat de vloeistof erover gegaan is. Dit principe werkt volgens de volgende formule. Naarmate dat het medium veranderd, verandert ook de impedantie waarde van deze condensator. Een verandering van impedantie betekent dan ook een verandering in spanning. Deze spanningsverandering moeten we hebben en gaan we dan opmeten met de arduino. Dit is ongeveer hetzelfde principe als capacitieve touchscreens in gsm’s.

**Figuur 1.2: Impedantie bepaling van een condensator**

|  |  |
| --- | --- |
| Afbeeldingsresultaat voor impedance capacitor | https://www.global.tdk.com/techmag/electronics_primer/img/img_electronics_files_primer_vol1_1.gif |

Hierbij is dan een klein Arduinoscript geschreven voor alle spanningsveranderingen op te vangen die ingelezen worden door de poorten van de digitale poorten van de arduino als input te programmeren. Natuurlijk is er ook een juiste Baud-rate en een pauze ingesteld voor een goede datatransfer te kunnen verkrijgen door de databus van de computer. Zonder deze factoren zou er een data-overflow zijn. Hiernaast is er een Python code gelinkt aan de arduino code die dan alle inkomende data opvangt en wegschrijft in een CSV-file **(=Comma Separated Values).** Deze file is makkelijk voor het omzetten van data naar zichtbare informatie. Dit hebben we dan ook gedaan met nog een tweede python-script die e csv file dan omzet naar een juiste grafiek voor een gebruikersvriendelijke weergave. Alle code wordt in de bijlagen meegegeven.

## **Uitvoering:**

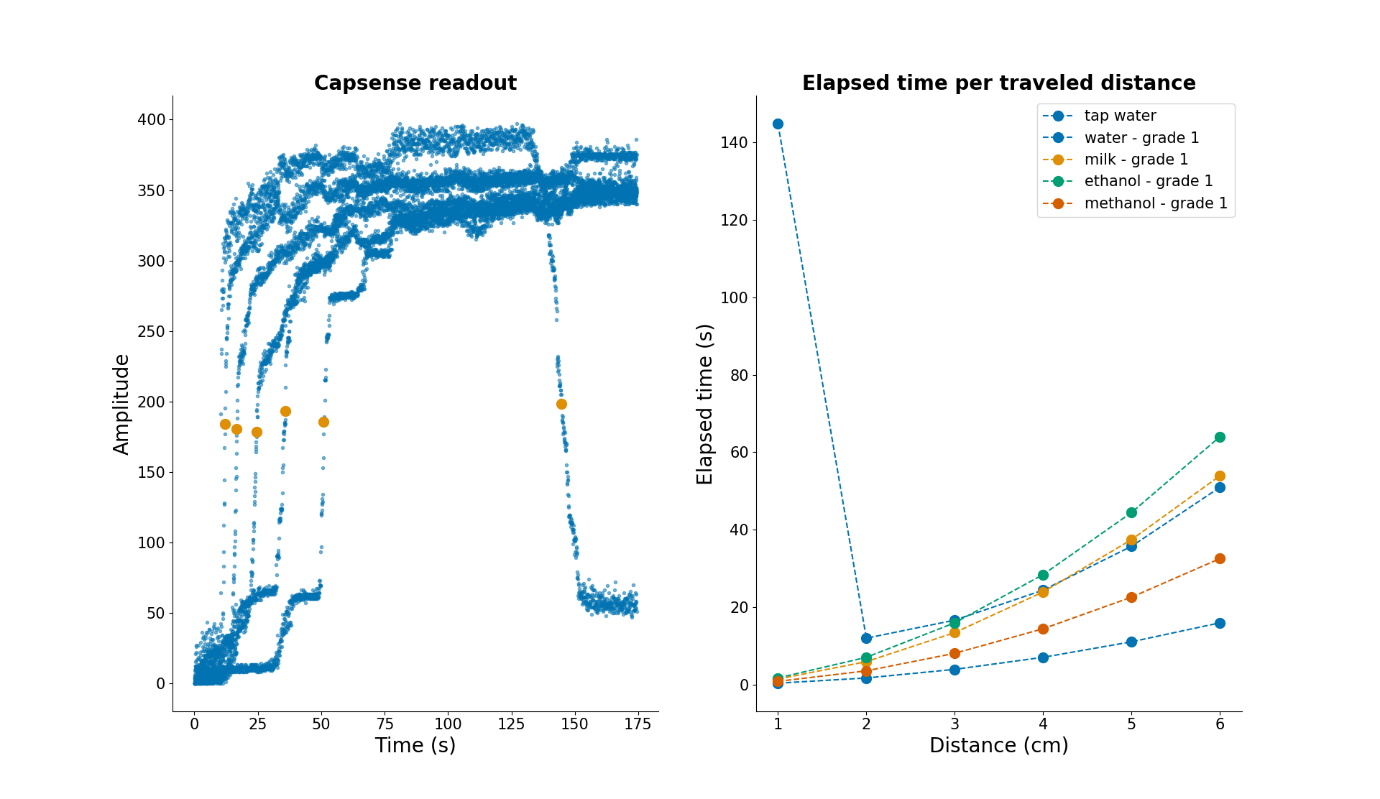
De PCB Krijgt eerst een laag met dubbelzijdige folie zodat er geen korsluiting komt tussen kort bijliggende aansluitingen. Op deze folie doen we een laag Kapton-tape. Dit is een speciaal soort papier met een hoge absorptie voor vloeistoffen. De tape hebben we 2 cm breed genomen en Rond deze tape wordt dan nog een laag folie gedaan zodat er geen druppels vloeistof van het papier afgaan en op de pcb terechtkomen.

Erna monteren we de PCB op de arduino met het ingeladen arduino-script. Deze opstelling sluiten we dan aan via een USB aan de computer die het pythonscript voor het inlezen van de waarde al uitvoert. Nu zien we allemaal verschillende waarde verschijnen op de terminal die vrij laag zijn. Dit betekent dat de opstelling werkt. Met een pipet nemen we een afgemeten hoeveelheid vloeistof **(=2ml**) en laten een druppel vallen op het beginpunt van de Kapton-tape. Nu is het wachten tot de vloeistof tot het einde is gekomen. Dit kan vrij lang duren voor andere vloeistoffen dan water omdat de meeste andere vloeistoffen een grotere viscositeit hebben dan water. Dit is goed want we gaan een grafiek uitplotten van tijd in functie van afgelopen afstand van de vloeistof. Dus hoe langzamer de vloeistof, hoe steiler de grafiek. Als we dit onderscheidt kunnen zien op de grafiek, dan weten we dat de proef geslaagd is.

## **Metingen:**

De resultaten van de opstelling zijn weergegeven in figuur 1.3. Dit is de grafiek die aangeeft hoe snel de gemeten vloeistoffen tot het einde zijn geraakt. Volgens de Washburn vergelijking zal melk trager geabsorbeerd worden door papier dan water. Voor de volledige absorptie van een strook Whatman Grade 1 papier van 6 cm lang zal de benodigde absorptie tijd voor melk 20 seconden hoger liggen dan die van water.

**Figuur 1.3: Vloeistofkarakteristiek**



# Besluit

Op de grafiek is duidelijk een onderscheid te zien tussen de vloeistoffen. Dit geeft aan dat de proef een succes was en dat er een onderscheidt gemaakt kan worden tussen verschillende viscositeiten. Het is heel belangrijk niet teveel van een vloeistof te nemen omdat het belletje dat in het begin ontstaat kan overlopen en een kortsluiting in het circuit kan geven. Een tweede belangrijk punt is dat de strook niet te dik genomen mocht worden. Dit hebben we zelf ondervonden. In het begin nam ik een strook van 4cm breed. Maar omdat deze strook een stuk breder is gaat de proef ook meer tijd in beslag nemen omdat de vloeistof opgenomen moet worden. Het kan dan ook zijn dat een druppel van een vloeistof niet genoeg is om over de hele strook te laten gaan. En dit kan irregulariteiten geven in de meting en dat is niet goed.

Ten laatste zijn de metingen zeker reproduceerbaar. Het enige dat gebeurt moet worden is een nieuw tape met dubbelzijdige folie te monteren op de PCB.

# BIjlagen

## **Bijlage 1: Omzetten van de CSV naar een plotgrafiek.**

**import** os

**import** glob

**from** natsort **import** natsorted

**import** tkinter **as** tk

**from** tkinter **import** filedialog

**from** pandas **import** **\***

**import** matplotlib**.**pyplot **as** plt

**import** numpy **as** np

color\_palette **=** **[**"#0173B2"**,** "#DE8F05"**,** "#029E73"**,** "#D55E00"**,** "#CC78BC"**,**"#CA9161"**,** "#FBAFE4"**,** "#949494"**,** "#ECE133"**,** "#56B4E9"**,**"#0173B2"**,** "#DE8F05"**,** "#029E73"**,** "#D55E00"**,** "#CC78BC"**,**"#CA9161"**,** "#FBAFE4"**,** "#949494"**,** "#ECE133"**,** "#56B4E9"**,**"#0173B2"**,** "#DE8F05"**,** "#029E73"**,** "#D55E00"**,** "#CC78BC"**,**"#CA9161"**,** "#FBAFE4"**,** "#949494"**,** "#ECE133"**,** "#56B4E9"**,**"#0173B2"**,** "#DE8F05"**,** "#029E73"**,** "#D55E00"**,** "#CC78BC"**,**"#CA9161"**,** "#FBAFE4"**,** "#949494"**,** "#ECE133"**,** "#56B4E9"**,**"#0173B2"**,** "#DE8F05"**,** "#029E73"**,** "#D55E00"**,** "#CC78BC"**,**"#CA9161"**,** "#FBAFE4"**,** "#949494"**,** "#ECE133"**,** "#56B4E9"**]**

SMALL\_SIZE **=** 15

MEDIUM\_SIZE **=** 20

BIGGER\_SIZE **=** 20

plt**.**rc**(**'font'**,** size**=**SMALL\_SIZE**)** # controls default text sizes

plt**.**rc**(**'axes'**,** titlesize**=**BIGGER\_SIZE**)** # fontsize of the axes title

plt**.**rc**(**'axes'**,** titleweight**=**'bold'**)** # fontsize of the axes title

plt**.**rc**(**'axes'**,** labelsize**=**MEDIUM\_SIZE**)** # fontsize of the x and y labels

plt**.**rc**(**'xtick'**,** labelsize**=**SMALL\_SIZE**)** # fontsize of the tick labels

plt**.**rc**(**'ytick'**,** labelsize**=**SMALL\_SIZE**)** # fontsize of the tick labels

plt**.**rc**(**'legend'**,** fontsize**=**SMALL\_SIZE**)** # legend fontsize

plt**.**rc**(**'figure'**,** titlesize**=**BIGGER\_SIZE**)** # fontsize of the figure title

**class** **Liquid:**

**def** \_\_init\_\_**(**self**,** name**,** n**,** y**):**

self**.**name **=** name

self**.**n **=** n # dynamic viscosity [Ns/m\*\*2] https://www.engineeringtoolbox.com/absolute-viscosity-liquids-d\_1259.html

self**.**y **=** y # surface tension [N/m] http://www.surface-tension.de/

liquid\_water **=** Liquid**(**name**=**'water'**,**

n **=** 0.00089**,**

y **=** 72.80e-3**)**

liquid\_milk **=** Liquid**(**name**=**'milk'**,**

n **=** 0.003**,**

y **=** 72.80e-3**)**

liquid\_olive\_oil **=** Liquid**(**name**=**'olive oil'**,**

n **=** 10e-1**,**

y **=** 29.75e-3**)** # http://www.surface-tension.org/news/61.html

liquid\_ethanol **=** Liquid**(**name**=**'ethanol'**,**

n **=** 0.001095**,**

y **=** 22.39e-3**)**

liquid\_methanol **=** Liquid**(**name**=**'methanol'**,**

n **=** 0.00056**,**

y **=** 22.50e-3**)**

**class** **Paper:**

**def** \_\_init\_\_**(**self**,** name **,** pore\_size**):**

self**.**name **=** name

self**.**pore\_size **=** pore\_size # [m]

paper\_what\_grad\_1 **=** Paper**(**name**=**'Whatman Grade 1'**,**

pore\_size**=**11e-6**)**

paper\_what\_grad\_2 **=** Paper**(**name**=**'Whatman Grade 2'**,**

pore\_size**=**8e-6**)**

paper\_what\_grad\_3 **=** Paper**(**name**=**'Whatman Grade 3'**,**

pore\_size**=**6e-6**)**

paper\_what\_grad\_4 **=** Paper**(**name**=**'Whatman Grade 4'**,**

pore\_size**=**22e-6**)**

paper\_what\_grad\_5 **=** Paper**(**name**=**'Whatman Grade 5'**,**

pore\_size**=**2.5e-6**)**

paper\_what\_grad\_6 **=** Paper**(**name**=**'Whatman Grade 6'**,**

pore\_size**=**3e-6**)**

paper\_1um **=** Paper**(**name**=**'Whatman Grade 6'**,**

pore\_size**=**1e-6**)**

**def** select\_csv\_file**():**

**print(**"Select .csv file:"**)**

csv\_file **=** filedialog**.**askopenfilename**()**

**print(**"Selected .csv file: {}"**.**format**(**csv\_file**))**

**print(**""**)**

**return** csv\_file

**def** set\_header\_row\_nr**():**

header\_row **=** input**(**"Set the header-row-number: "**)**

**print(**"Header row set to: {}"**.**format**(**header\_row**))**

**print(**""**)**

# do minus 1 because python starts counting from zero

header\_row **=** int**(**header\_row**)** **-** 1

**return** int**(**header\_row**)**

**def** get\_header\_values**(**file**,** header\_row\_nr**):**

'''

Sets header\_values

header\_values look like: [Time(s),I(A),U(V),...]

we find the header\_values by using the header\_row\_nr

'''

**print(**"set\_header\_values was executed"**)**

**with** open**(**file**,** "r"**)** **as** f**:**

**for** x**,** line **in** enumerate**(**f**):**

**if** x **==** header\_row\_nr**:**

# print(line)

header\_list **=** line**.**split**(**","**)**

**break**

**print(**header\_list**)**

**return** header\_list

**def** select\_column\_by\_name**(**header\_values**,** message**):**

index **=** 0

**for** header **in** header\_values**:**

**print(**"({}) - {}"**.**format**(**index**,** header**))**

index **=** index **+** 1

selected\_index **=** int**(**input**(**message**))**

selected\_header **=** header\_values**[**selected\_index**]**

**print(**"Selected column with header: {}"**.**format**(**selected\_header**))**

**print(**""**)**

**return** selected\_index

**def** read\_single\_column\_from\_csv\_file**(**csv\_file**,**header\_row\_nr**,**column\_number**):**

'''

returns the column in a numpy array

'''

col\_data **=** pandas**.**read\_csv**(**csv\_file**,** sep**=**','**,** header **=** header\_row\_nr**,** usecols**=[**column\_number**],**encoding **=** "ISO-8859-1"**)**

col\_data\_np\_array **=** col\_data**.**as\_matrix**()**

**return** col\_data\_np\_array

**def** read\_multiple\_columns\_from\_csv\_file**(**csv\_file**,**header\_row\_nr**,**use\_cols**):**

'''

returns the column in a numpy array

'''

col\_data **=** pandas**.**read\_csv**(**csv\_file**,** sep**=**','**,** header **=** header\_row\_nr**,** usecols**=**use\_cols**,**encoding **=** "ISO-8859-1"**)**

col\_data\_np\_array **=** col\_data**.**as\_matrix**()**

**return** col\_data\_np\_array

**def** get\_filename**(**full\_filepath**):**

'''

Get the filename by using the full\_filepath

'''

filename **=** os**.**path**.**basename**(**full\_filepath**)**

**return** filename

**def** washburn\_eq**(**L**,** liquid**,** paper**):**

# L = distance to travel

# n = viscosity

# D = pore size

# y = surface tension

# This function returns the time it takes for a fluid to travel L

n **=** liquid**.**n

y **=** liquid**.**y

D **=** paper**.**pore\_size

t **=** **(**L**\*\***2 **\*** 4 **\*** n**)** **/** **(**y**\***D**)**

**return** t

data\_file **=** 'ArduinoData\_Jordi\_Jaspers.csv' # select\_csv\_file()

header\_row\_nr **=** 0 # set\_header\_row\_nr()

time\_col **=** read\_single\_column\_from\_csv\_file**(**data\_file**,**header\_row\_nr**,**0**)**

time\_col **=** np**.**cumsum**(**time\_col**)/**1000

# drop noisy data at the end

end\_time **=** 235

end\_index **=** min**(**range**(**len**(**time\_col**)),** key**=lambda** i**:** abs**(**time\_col**[**i**]-**end\_time**))**

time\_col **=** time\_col**[**0**:**end\_index**]**

data\_col **=** read\_multiple\_columns\_from\_csv\_file**(**data\_file**,**header\_row\_nr**,[**1**,**2**,**3**,**4**,**5**,**6**])**

data\_col **=** data\_col**[**0**:**end\_index**]**

**print(**type**(**data\_col**))**

**print(**data\_col**.**shape**)**

ax = plt.subplot(1,2,1)

ax.set\_title('Capsense readout') # Title

ax.set\_ylabel('Amplitude') # Y label

ax.set\_xlabel('Time (s)') # X label

ax.plot(time\_col, data\_col, '.', color=color\_palette[0], alpha=0.5)

# Hide the right and top spines

ax.spines['right'].set\_visible(False)

ax.spines['top'].set\_visible(False)

# Only show ticks on the left and bottom spines

ax.yaxis.set\_ticks\_position('left')

ax.xaxis.set\_ticks\_position('bottom')

# Bad idea

# Map all data between 1 and 100

# i = 0

# for column in data\_col.T:

# max\_value = np.amax(column)

# print(max\_value)

# plt.figure(1)

# plt.plot(time\_col, (column/max\_value)\*100, '.', color=color\_palette[1], alpha=0.5)

# Better idea

# Find center datapoint

intersect\_time\_array = np.array([])

i = 0

for column in data\_col.T:

max\_value = np.amax(column)

min\_value = np.amin(column)

center\_value = (min\_value + max\_value)/2

center\_value\_index = min(range(len(column)), key=lambda i: abs(column[i]-center\_value))

print('Center value = {}'.format(center\_value))

plt.figure(1)

time\_point = time\_col[center\_value\_index]

intersect\_time\_array = np.append(intersect\_time\_array, time\_point)

plt.plot(time\_point, center\_value, 'o', markersize='10', color=color\_palette[1], alpha=1)

distance\_array = np.array([1,2,3,4,5,6]) # in cm

ax2 = plt.subplot(1,2,2)

ax2.set\_title('Elapsed time per traveled distance') # Title

ax2.set\_ylabel('Elapsed time (s)') # Y label

ax2.set\_xlabel('Distance (cm)') # X label

# Hide the right and top spines

ax2.spines['right'].set\_visible(False)

ax2.spines['top'].set\_visible(False)

# Only show ticks on the left and bottom spines

ax2.yaxis.set\_ticks\_position('left')

ax2.xaxis.set\_ticks\_position('bottom')

plt.plot(distance\_array, intersect\_time\_array, '--o', markersize='10', color=color\_palette[0], alpha=1, label='tap water')

# Plot curves for different liquids

distance\_array\_in\_meters = distance\_array/100

times\_for\_water = washburn\_eq(distance\_array\_in\_meters, liquid\_water, paper\_what\_grad\_1)

times\_for\_milk = washburn\_eq(distance\_array\_in\_meters, liquid\_milk, paper\_what\_grad\_1)

times\_for\_ethanol = washburn\_eq(distance\_array\_in\_meters, liquid\_ethanol, paper\_what\_grad\_1)

times\_for\_methanol = washburn\_eq(distance\_array\_in\_meters, liquid\_methanol, paper\_what\_grad\_1)

# times\_for\_water2 = washburn\_eq(distance\_array\_in\_meters, liquid\_water, paper\_what\_grad\_5)

# times\_for\_milk2 = washburn\_eq(distance\_array\_in\_meters, liquid\_milk, paper\_what\_grad\_5)

# times\_for\_ethanol2 = washburn\_eq(distance\_array\_in\_meters, liquid\_ethanol, paper\_what\_grad\_5)

# times\_for\_methanol2 = washburn\_eq(distance\_array\_in\_meters, liquid\_methanol, paper\_what\_grad\_5)

plt.plot(distance\_array, times\_for\_water, '--o', markersize='10', color=color\_palette[0], alpha=1, label = liquid\_water.name + ' - grade 1' )

plt.plot(distance\_array, times\_for\_milk, '--o', markersize='10', color=color\_palette[1], alpha=1 ,label = liquid\_milk.name + ' - grade 1' )

plt.plot(distance\_array, times\_for\_ethanol, '--o', markersize='10', color=color\_palette[2], alpha=1, label = liquid\_ethanol.name + ' - grade 1' )

plt.plot(distance\_array, times\_for\_methanol, '--o', markersize='10', color=color\_palette[3], alpha=1, label = liquid\_methanol.name + ' - grade 1' )

# plt.plot(distance\_array, times\_for\_water2, '--^', markersize='10', color=color\_palette[0], alpha=1, label = liquid\_water.name + ' - grade 5' )

# plt.plot(distance\_array, times\_for\_milk2, '--^', markersize='10', color=color\_palette[1], alpha=1 ,label = liquid\_milk.name + ' - grade 5' )

# plt.plot(distance\_array, times\_for\_ethanol2, '--^', markersize='10', color=color\_palette[2], alpha=1, label = liquid\_ethanol.name + ' - grade 5' )

# plt.plot(distance\_array, times\_for\_methanol2, '--^', markersize='10', color=color\_palette[3], alpha=1, label = liquid\_methanol.name + ' - grade 5' )

ax2.legend()

plt.show()

## **Bijlage 2: Inlezen van de waarde.**

**import** serial

**import** csv

file **=** 'ArduinoData\_Jordi\_Jaspers.csv'

port **=** 'COM3'

baudRate **=** 9600

fileMode **=** 'w'

arduino **=** serial**.**Serial**(**port**,** baudRate**)**

arduino**.**flush**()**

arduino**.**reset\_input\_buffer**()**

**print(**"Arduino Connected @ COM3!"**)**

**with** open**(**file**,** fileMode**,** newline **=** ''**)** **as** WriteFile**:**

writer **=** csv**.**writer**(**WriteFile**)**

**print(**'file opened!'**)**

**while** **(True):**

**while** **(**arduino**.**inWaiting**()** **==** 0**):**

**pass**

**try:**

arduinoData **=** arduino**.**readline**()**

inputArray **=** arduinoData**.**decode**().**rstrip**().**split**(**','**)**

inputArray **=** list**(**map**(**int**,** inputArray**))**

arduino**.**reset\_input\_buffer**()**

**print(**inputArray**)**

**if(**len**(**inputArray**)** **>** 10**):**

writer**.**writerow**(**inputArray**)**

**else:**

**print(**'Need more data!'**)**

**except** **(**KeyboardInterrupt**,** SystemExit**,** IndexError**,** ValueError**):**

**pass**

WriteFile**.**close**()**

**break**

## **Bijlage 3: Arduino file.**

#include <CapacitiveSensor.h>

/\*

\* CapitiveSense Library Demo Sketch

\* Paul Badger 2008

\* Uses a high value resistor e.g. 10M between send pin and receive pin

\* Resistor effects sensitivity, experiment with values, 50K - 50M. Larger resistor values yield larger sensor values.

\* Receive pin is the sensor pin - try different amounts of foil/metal on this pin

\*/

CapacitiveSensor cs1 **=** CapacitiveSensor**(**2**,**3**);**

CapacitiveSensor cs2 **=** CapacitiveSensor**(**2**,**4**);**

CapacitiveSensor cs3 **=** CapacitiveSensor**(**2**,**5**);**

CapacitiveSensor cs4 **=** CapacitiveSensor**(**2**,**6**);**

CapacitiveSensor cs5 **=** CapacitiveSensor**(**2**,**7**);**

CapacitiveSensor cs6 **=** CapacitiveSensor**(**2**,**8**);**

CapacitiveSensor cs7 **=** CapacitiveSensor**(**2**,**9**);**

CapacitiveSensor cs8 **=** CapacitiveSensor**(**2**,**10**);**

CapacitiveSensor cs9 **=** CapacitiveSensor**(**2**,**11**);**

CapacitiveSensor cs10 **=** CapacitiveSensor**(**2**,**12**);**

void setup**()**

**{**

cs1**.**set\_CS\_AutocaL\_Millis**(**0xFFFFFFFF**);** // turn off autocalibrate on channel 1 - just as an example

cs2**.**set\_CS\_AutocaL\_Millis**(**0xFFFFFFFF**);** // turn off autocalibrate on channel 2 - just as an example

cs3**.**set\_CS\_AutocaL\_Millis**(**0xFFFFFFFF**);** // turn off autocalibrate on channel 3 - just as an example

cs4**.**set\_CS\_AutocaL\_Millis**(**0xFFFFFFFF**);** // turn off autocalibrate on channel 4 - just as an example

cs5**.**set\_CS\_AutocaL\_Millis**(**0xFFFFFFFF**);** // turn off autocalibrate on channel 5 - just as an example

cs6**.**set\_CS\_AutocaL\_Millis**(**0xFFFFFFFF**);** // turn off autocalibrate on channel 6 - just as an example

cs7**.**set\_CS\_AutocaL\_Millis**(**0xFFFFFFFF**);** // turn off autocalibrate on channel 1 - just as an example

cs8**.**set\_CS\_AutocaL\_Millis**(**0xFFFFFFFF**);** // turn off autocalibrate on channel 2 - just as an example

cs9**.**set\_CS\_AutocaL\_Millis**(**0xFFFFFFFF**);** // turn off autocalibrate on channel 3 - just as an example

cs10**.**set\_CS\_AutocaL\_Millis**(**0xFFFFFFFF**);** // turn off autocalibrate on channel 4 - just as an example

Serial**.**begin**(**9600**);**

**}**

void loop**()**

**{**

long start **=** millis**();**

long total1 **=** cs1**.**capacitiveSensor**(**30**);**

long total2 **=** cs2**.**capacitiveSensor**(**30**);**

long total3 **=** cs3**.**capacitiveSensor**(**30**);**

long total4 **=** cs4**.**capacitiveSensor**(**30**);**

long total5 **=** cs5**.**capacitiveSensor**(**30**);**

long total6 **=** cs6**.**capacitiveSensor**(**30**);**

long total7 **=** cs7**.**capacitiveSensor**(**30**);**

long total8 **=** cs8**.**capacitiveSensor**(**30**);**

long total9 **=** cs9**.**capacitiveSensor**(**30**);**

long total10 **=** cs10**.**capacitiveSensor**(**30**);**

Serial**.**print**(**millis**()** **-** start**);** // check on performance in milliseconds

Serial**.**print**(**','**);** // tab character for debug windown spacing

Serial**.**print**(**total1**);**

Serial**.**print**(**','**);**

Serial**.**print**(**total2**);**

Serial**.**print**(**','**);**

Serial**.**print**(**total3**);**

Serial**.**print**(**','**);**

Serial**.**print**(**total4**);**

Serial**.**print**(**','**);**

Serial**.**print**(**total5**);**

Serial**.**print**(**','**);**

Serial**.**print**(**total6**);**

Serial**.**print**(**','**);**

Serial**.**print**(**total7**);**

Serial**.**print**(**','**);**

Serial**.**print**(**total8**);**

Serial**.**print**(**','**);**

Serial**.**print**(**total9**);**

Serial**.**print**(**','**);**

Serial**.**println**(**total10**);**

delay**(**10**);** // arbitrary delay to limit data to serial port

**}**