1

TRAVAIL D'INITIATIVE PERSONNELLE ENCADRÉ

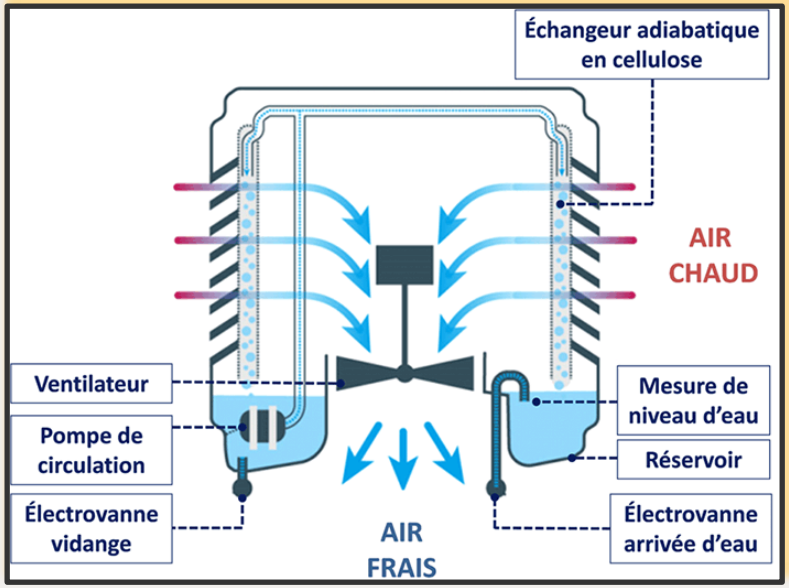
CLIMATISATION ADIABATIQUE

YVAN JACOB

Nb SCEI : 42634

CLIMATISATION ADIABATIQUE

Usage industriel



Sc : Oxycom, Xpair

CLIMATISATION ADIABATIQUE

Ancrage au thème

Transformation	Evaporation de l'eau
Transition	Une solution à la climatisation
Conversion	Energie électrique en énergie de refroidissement

Problématique

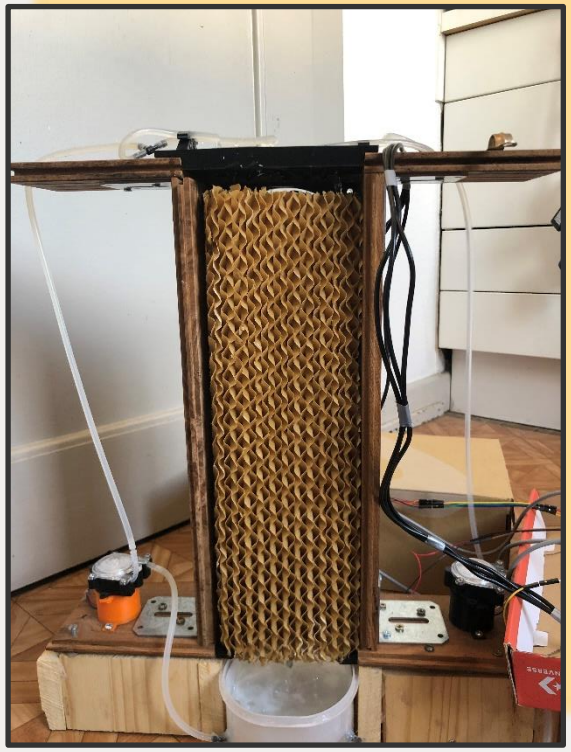
Passer d’un système millénaire à un système moderne : la climatisation adiabatique, mise en place, asservissement et rendement.

5

Passer d’un système millénaire à un système moderne : la climatisation adiabatique, mise en place, asservissement et rendement.

Sc : Build-Green

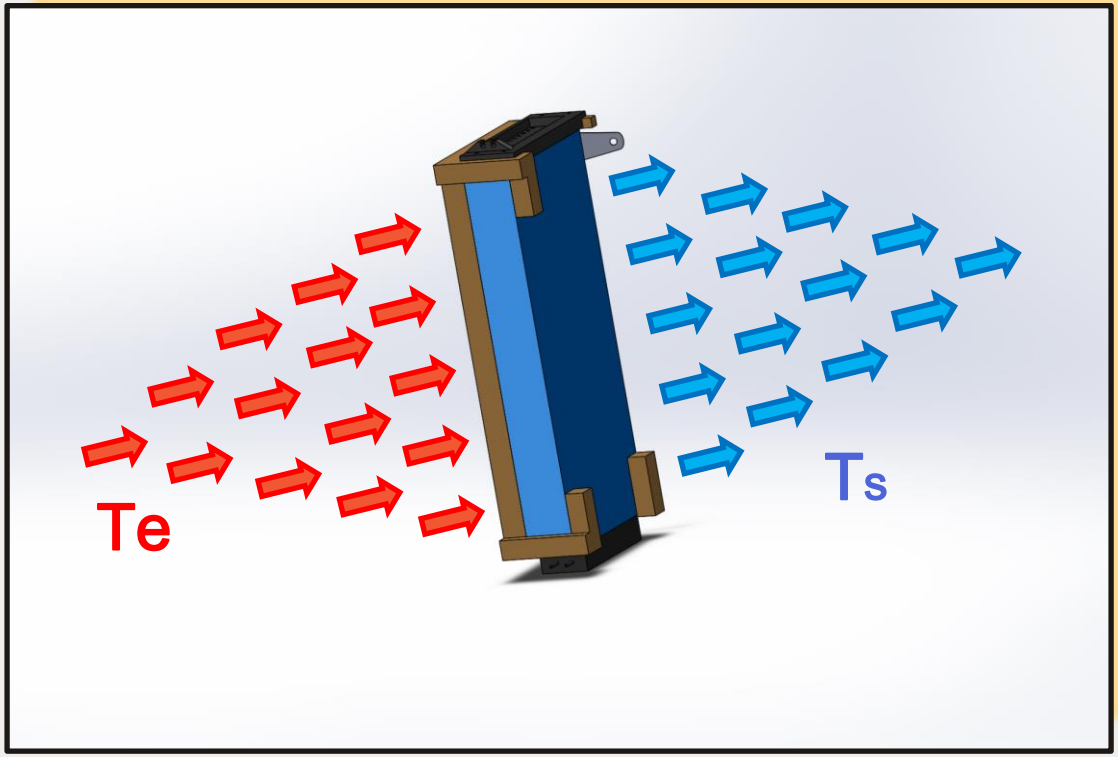
Electricité



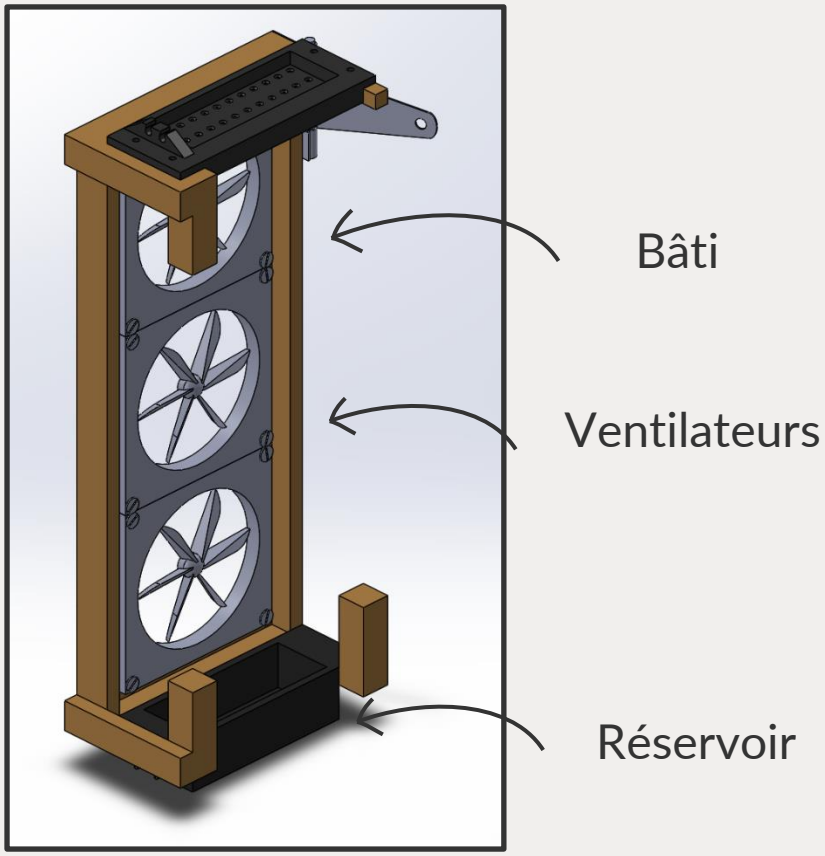
SOMMAIRE

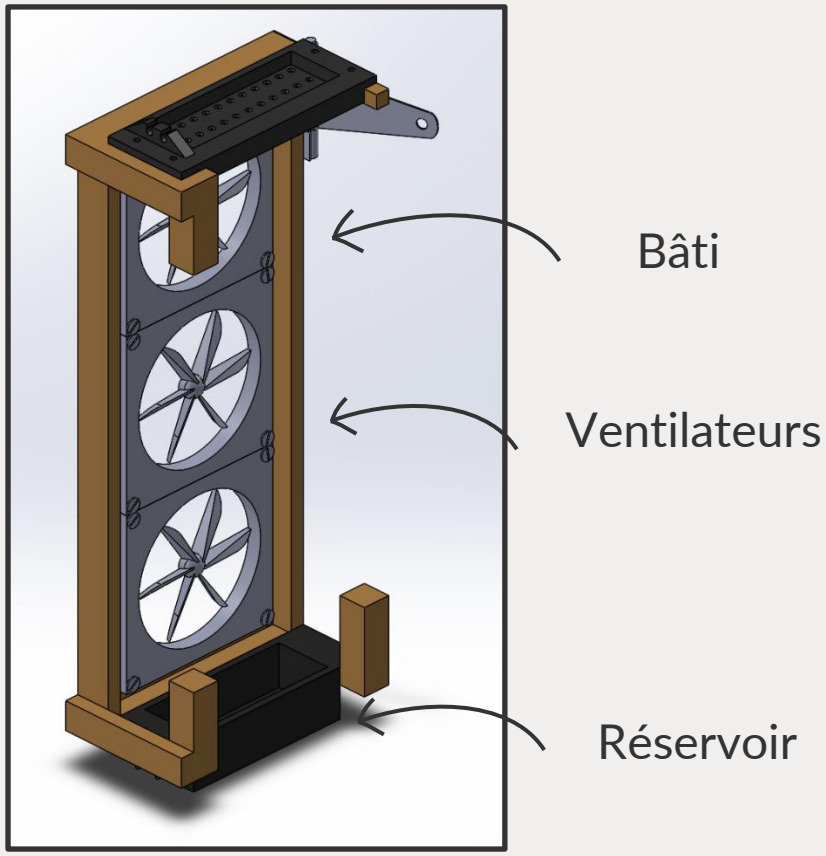
01

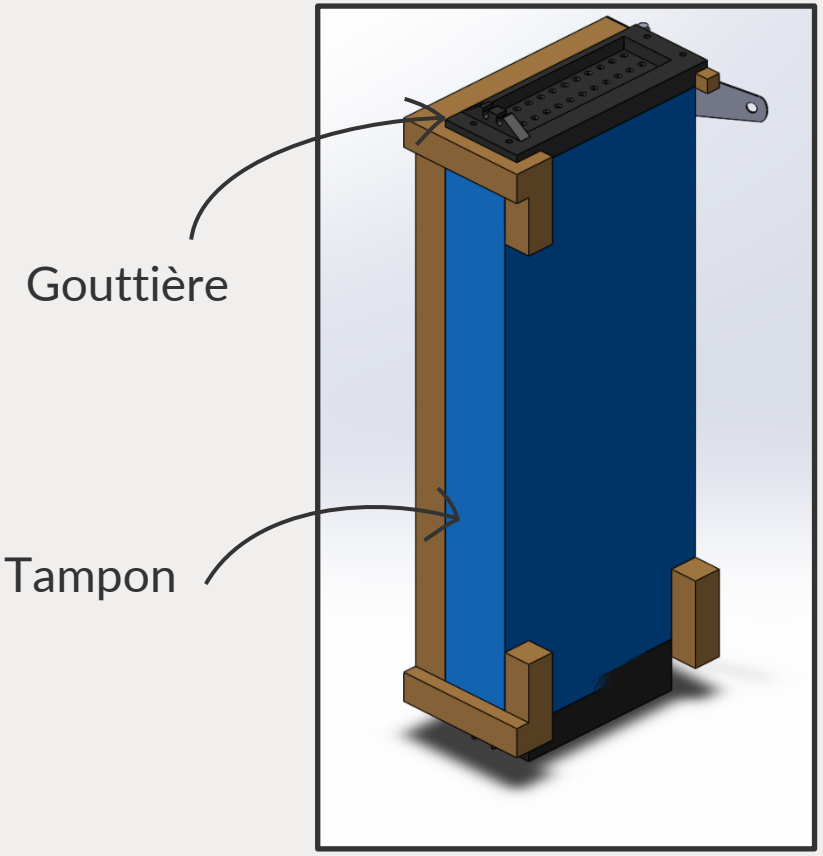
BUT

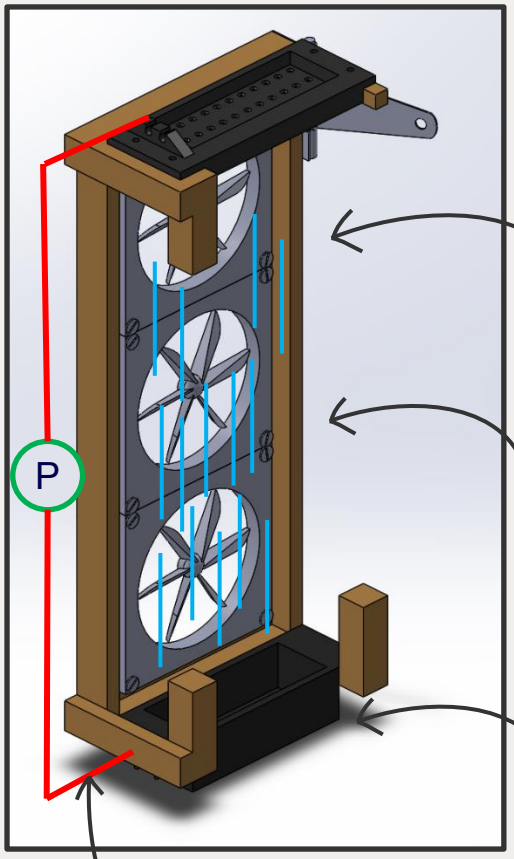


Création d’un système de refroidissement: climatisation adiabatique directe


Bâti
Ventilateurs
Réservoir


Bâti
Ventilateurs
Réservoir


Gouttière
Tampon



Bâti

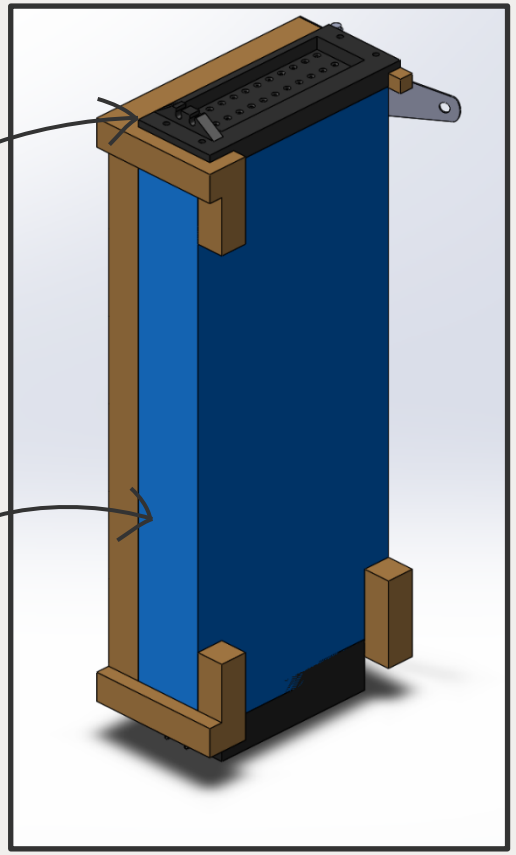
Ventilateurs

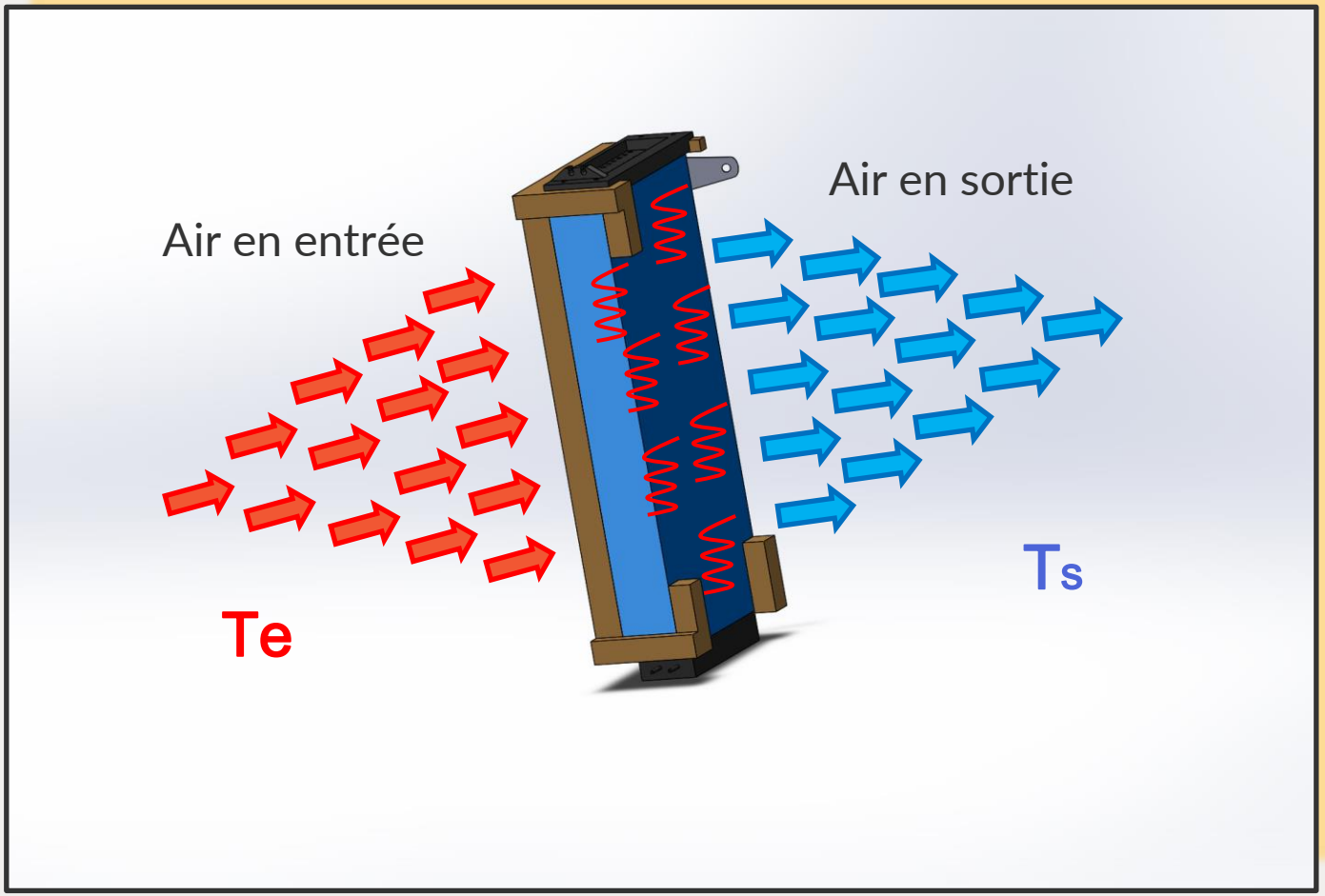
Réservoir

Pompe

Gouttière

Tampon



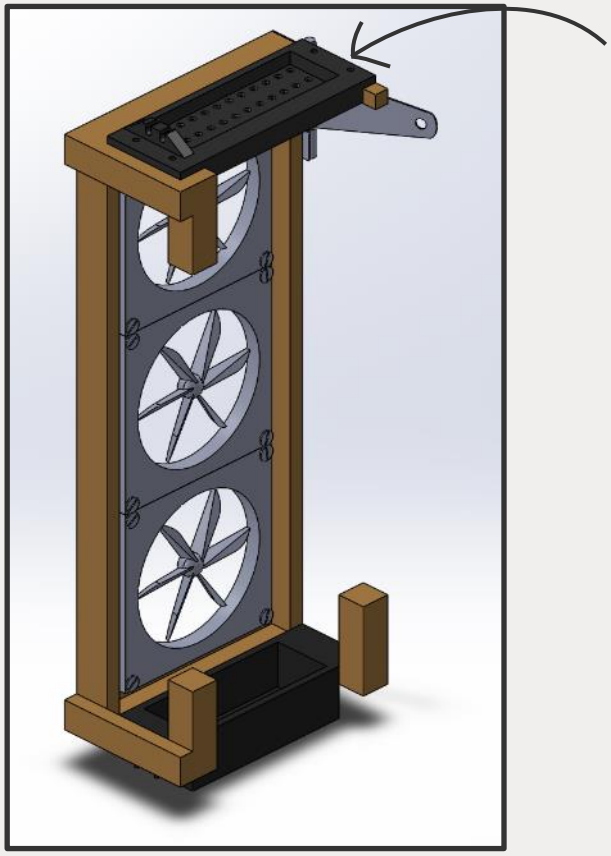

Air en entrée
Air en sortie
Te
Ts

02

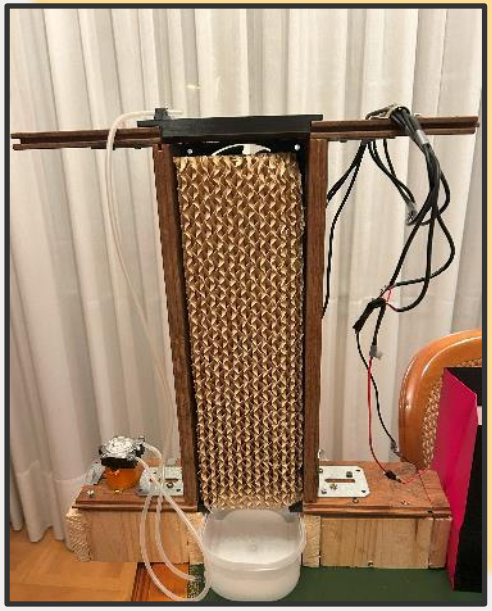
Construction

Réalisation de la climatisation

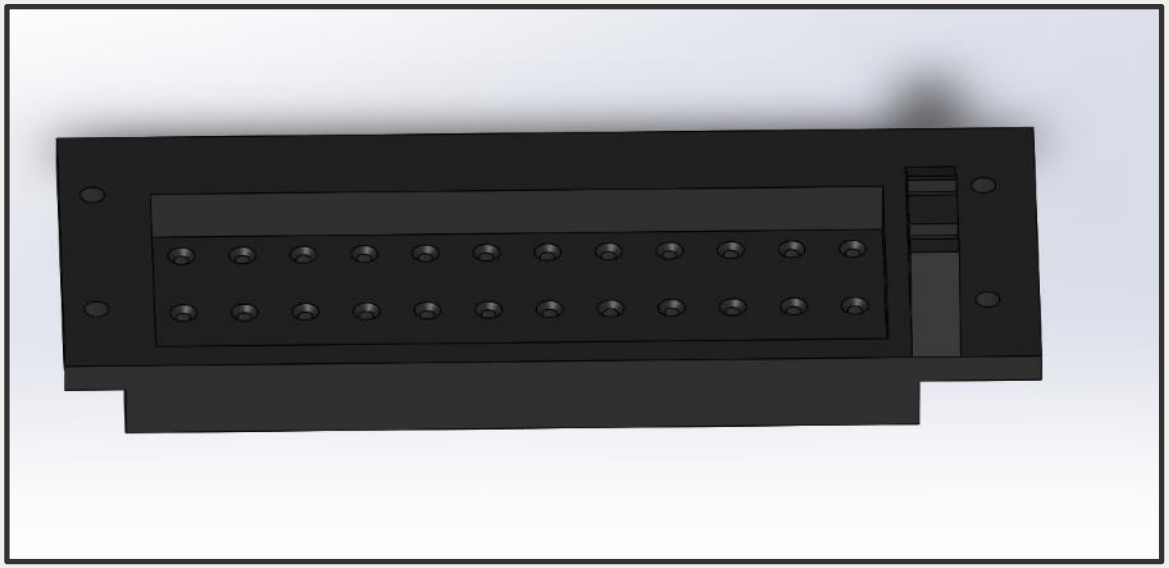
Mon prototype

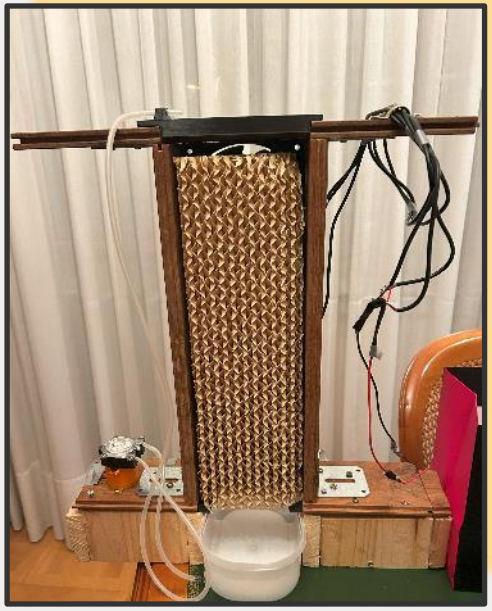


Gouttière de distribution d’eau



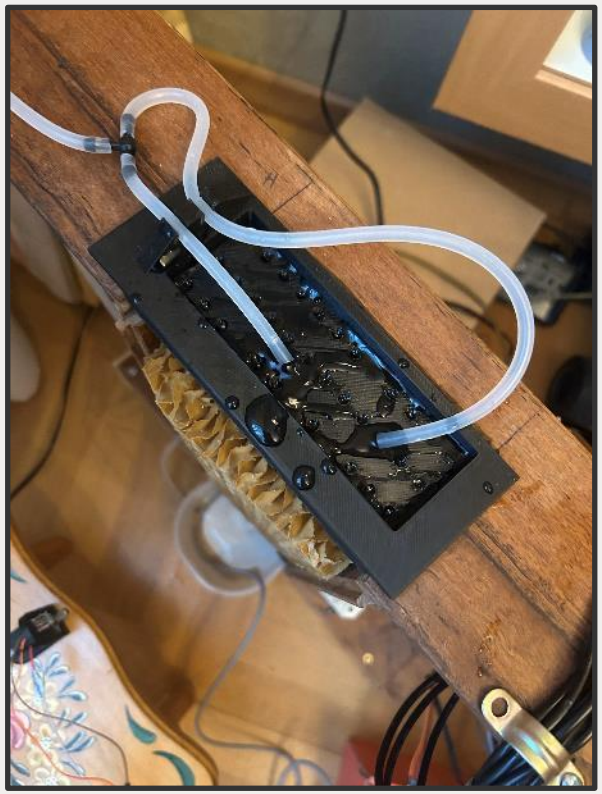
Mon prototype

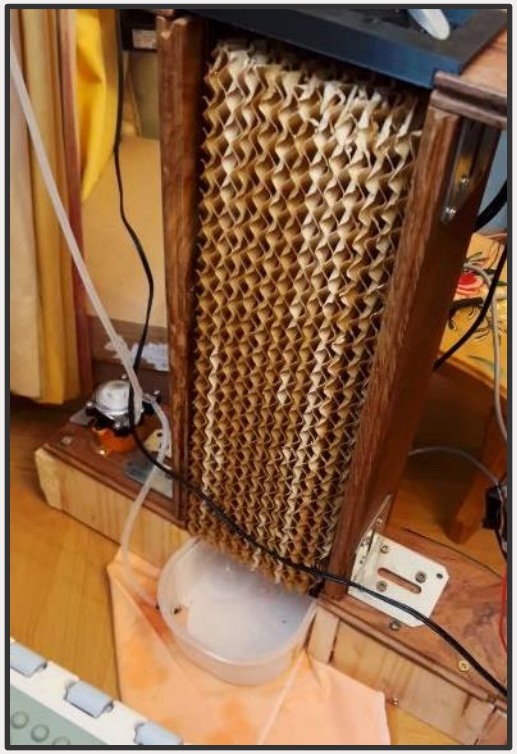




- Modélisation 3D avec SolidWorks

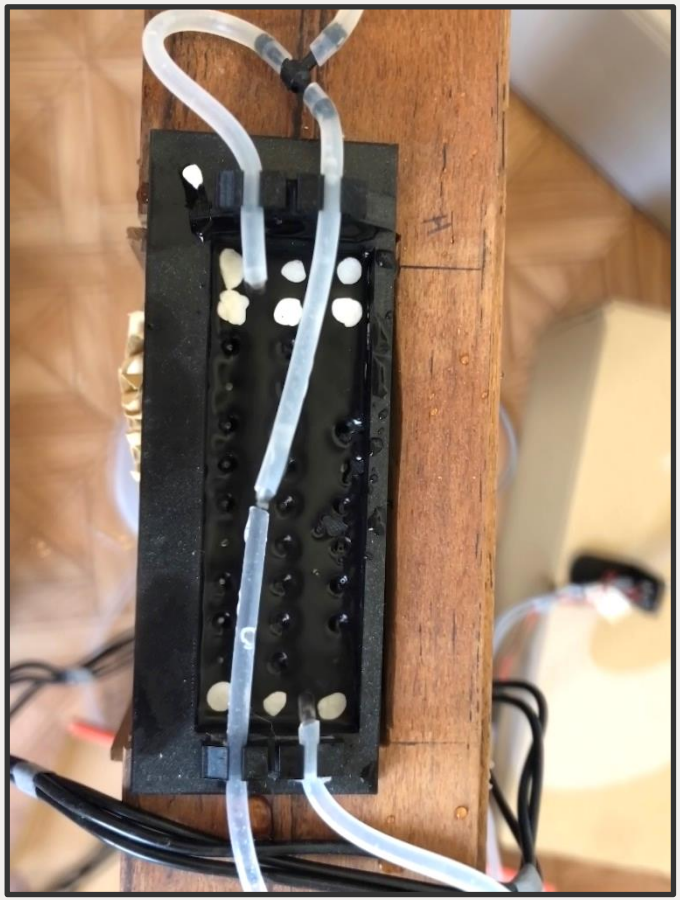
Mon prototype: problèmes

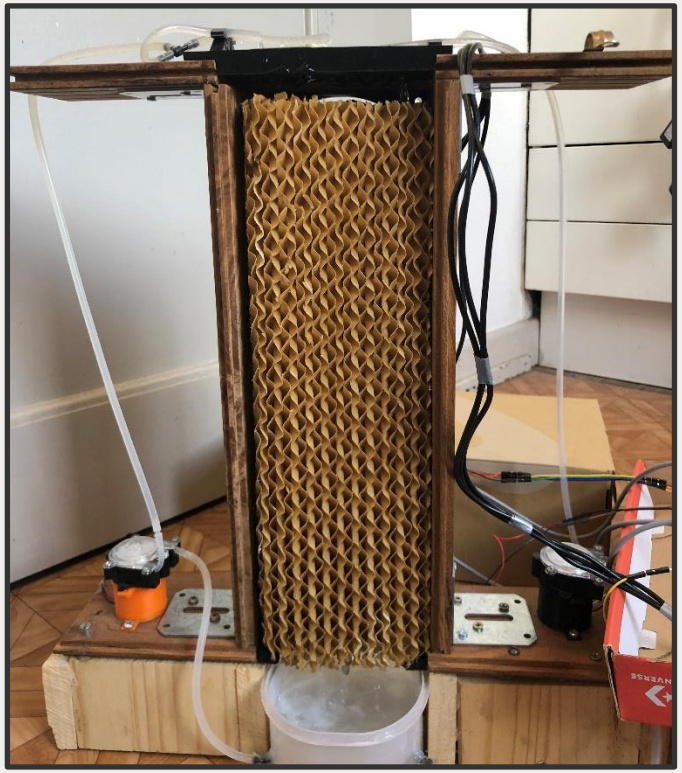




- Répartition eau
- Humidification tampon

Mon prototype

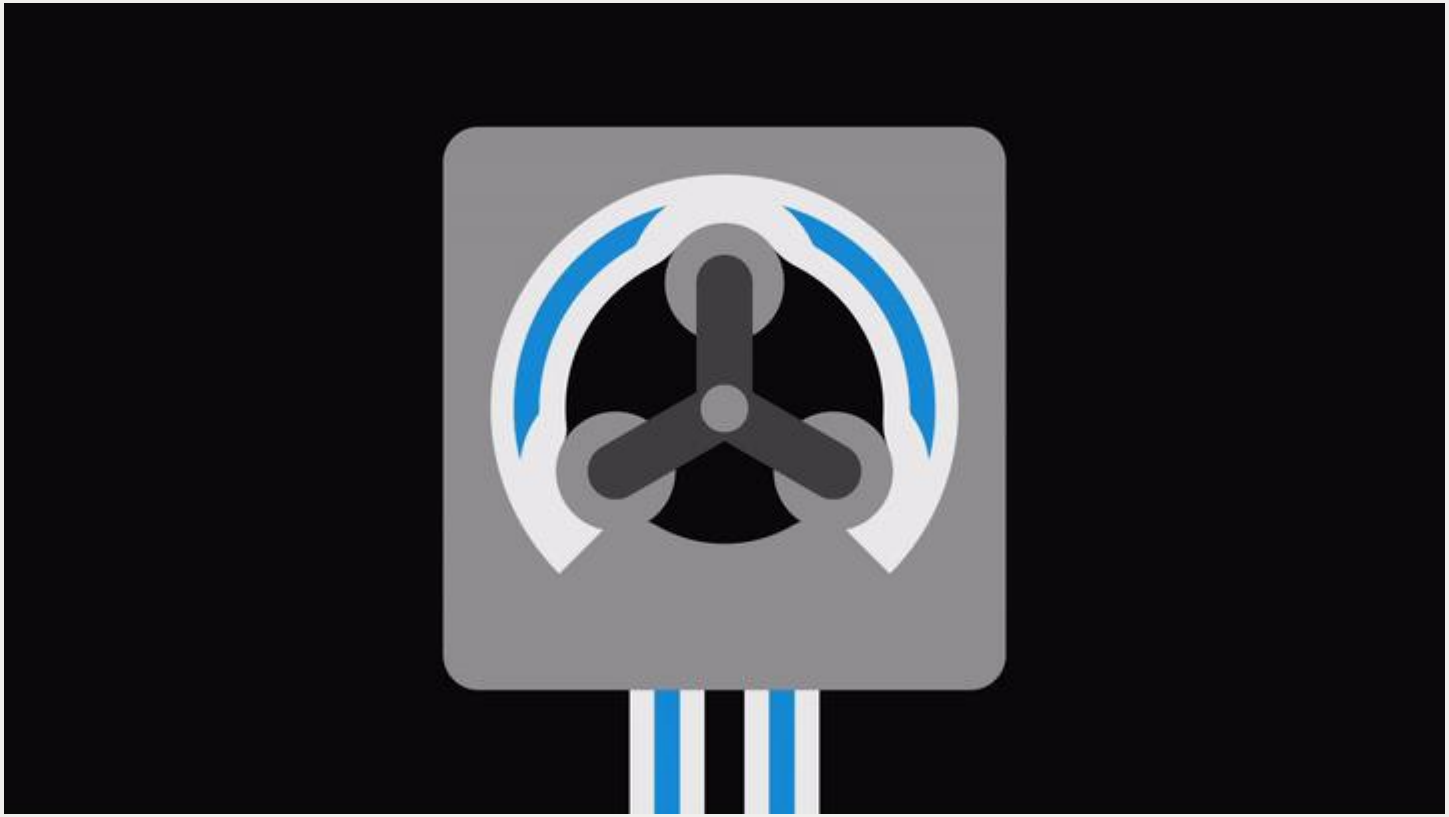




- Répartition eau uniforme

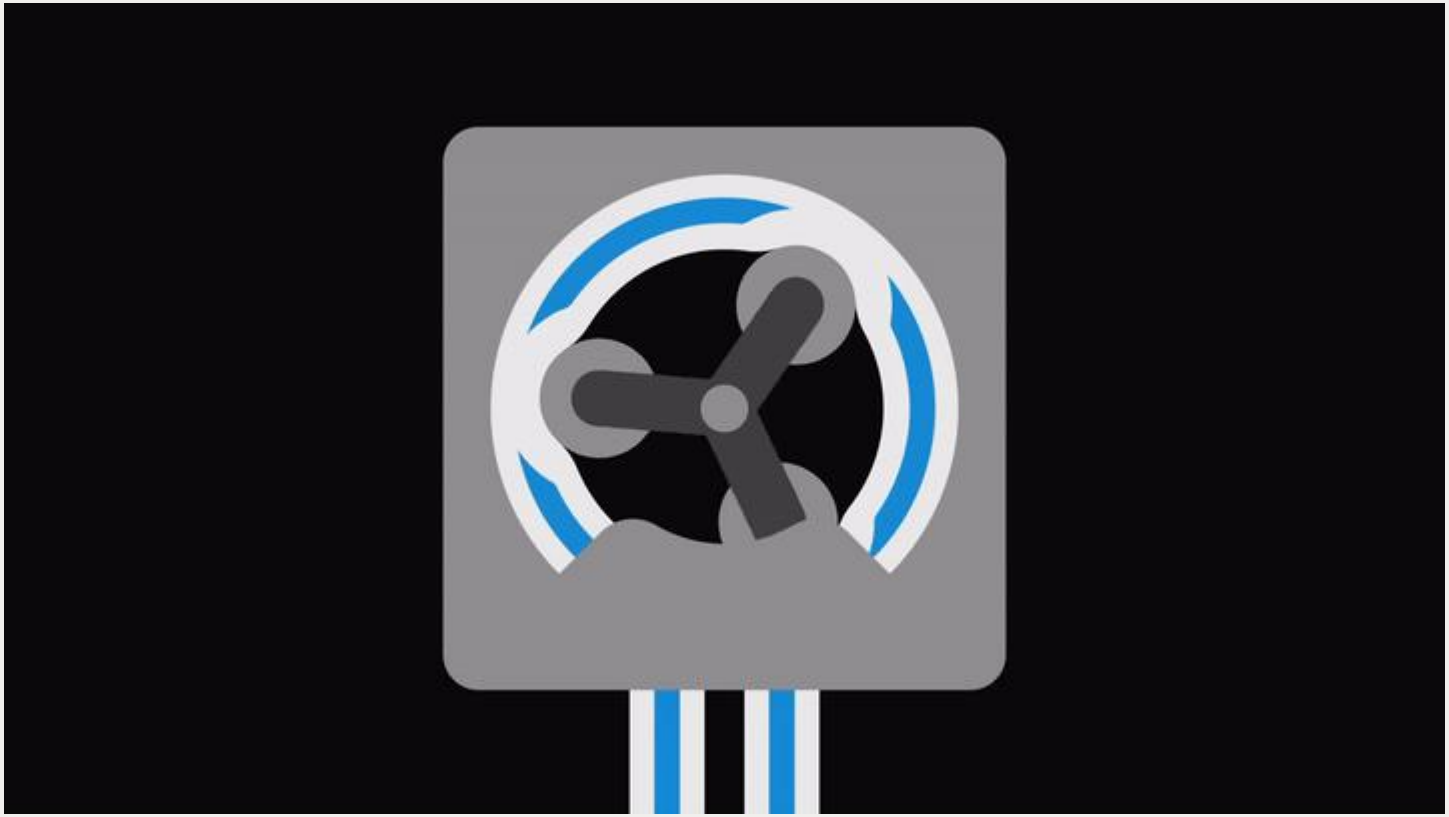
- Tampon entièrement humidifié

Pompe péristaltique



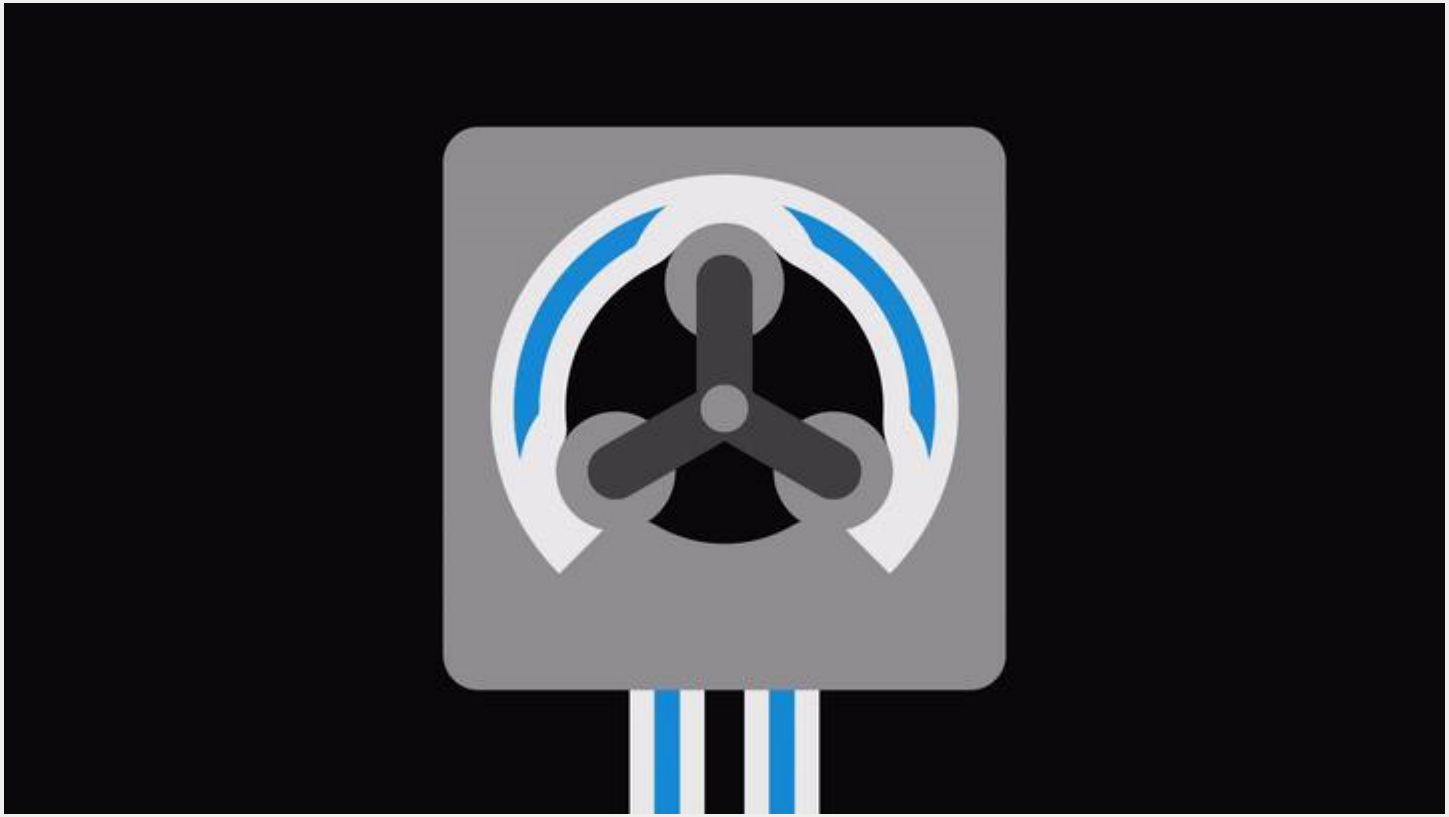
- Pompe péristaltique

Pompe péristaltique



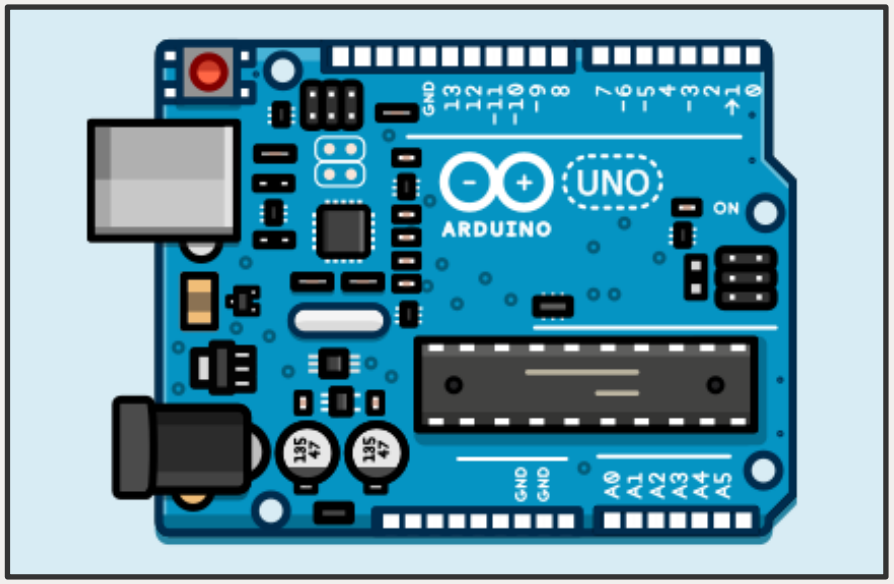
- Pompe péristaltique

Pompe péristaltique



- Pompe péristaltique

Commander la climatisation

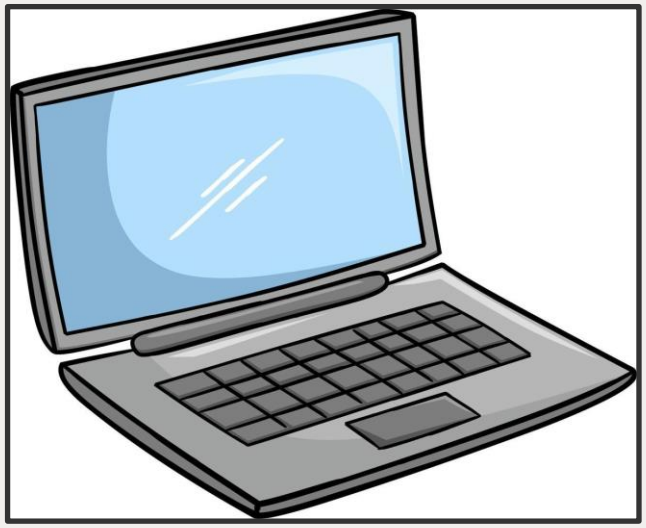


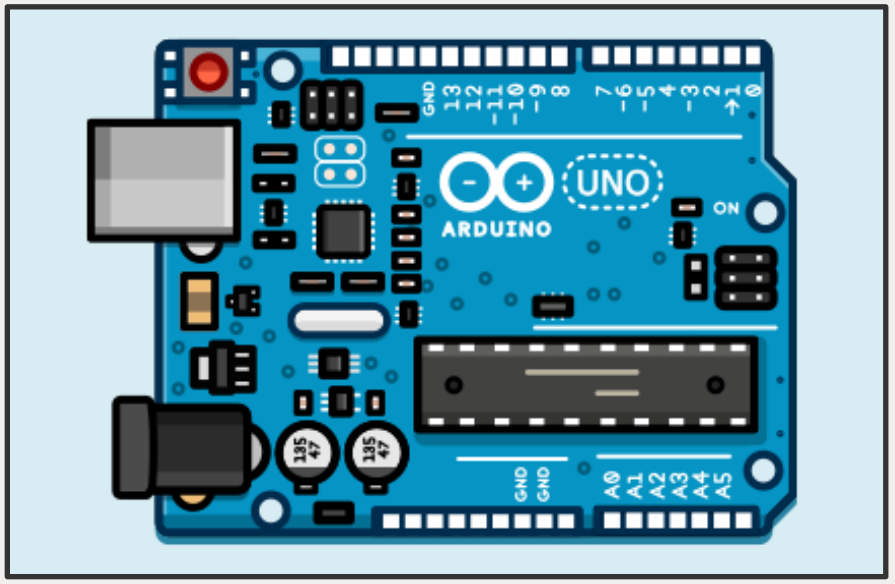
- Commander un automate
- Faire des mesures

Dialoguer avec un utilisateur

Commander la climatisation

Réaliser des mesures

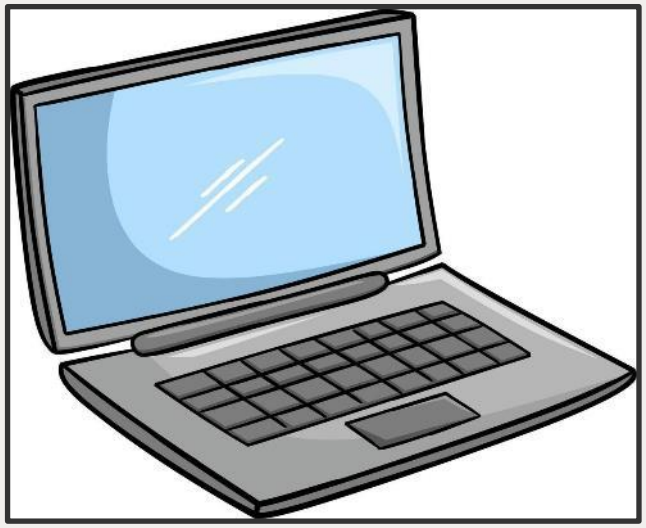


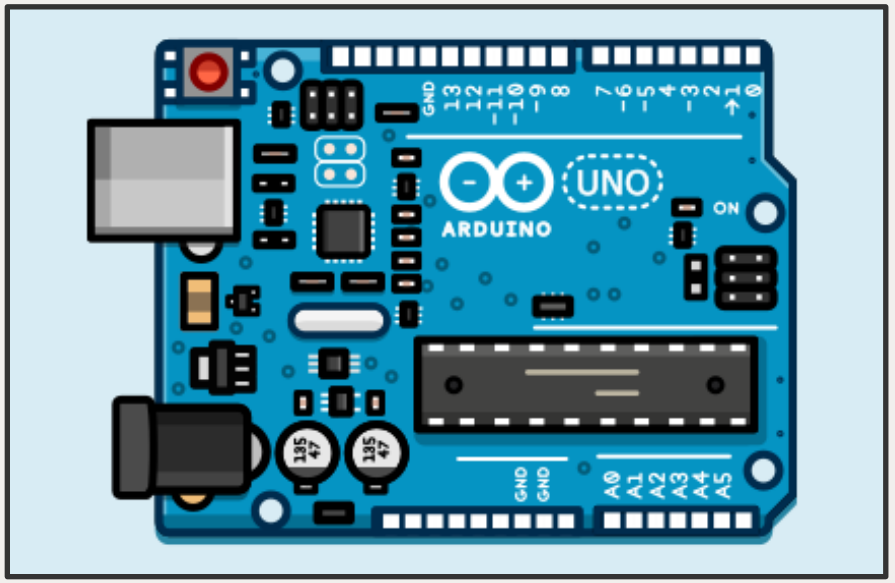


Commander la climatisation

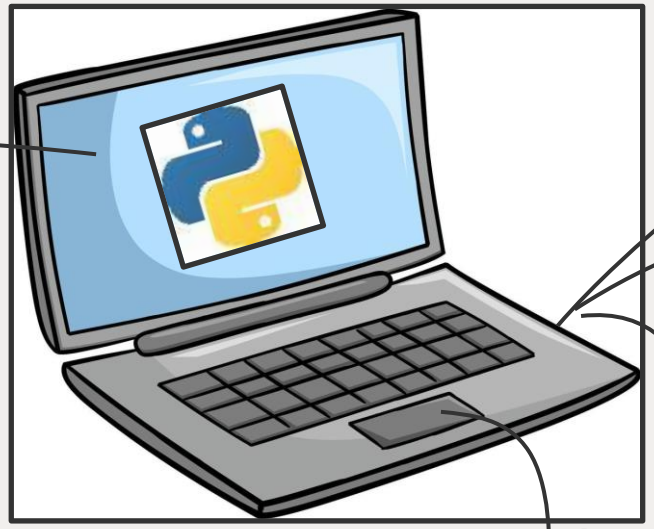
Réaliser des mesures

Allumer ventilateurs et pompes





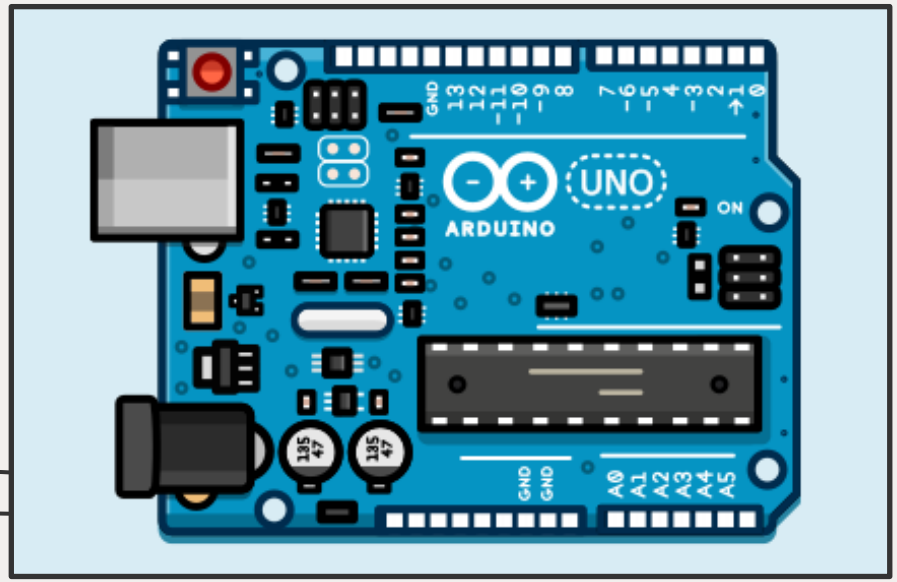
Dialogue



Allumer ventilateurs et pompes

Asservissement

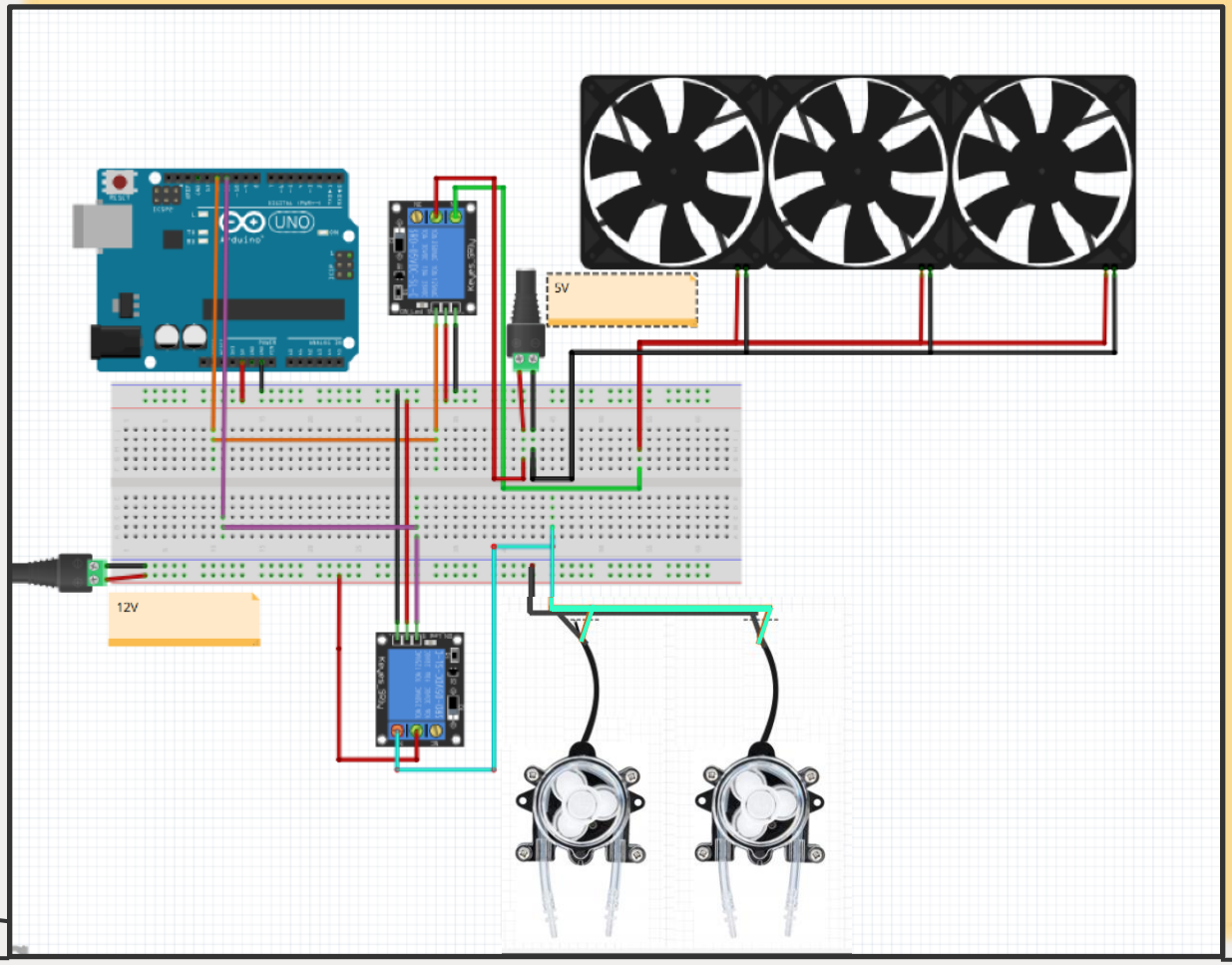
Lancer le programme



Réaliser des mesures

.

Commander la climatisation



Réalisé avec Fritzing

03

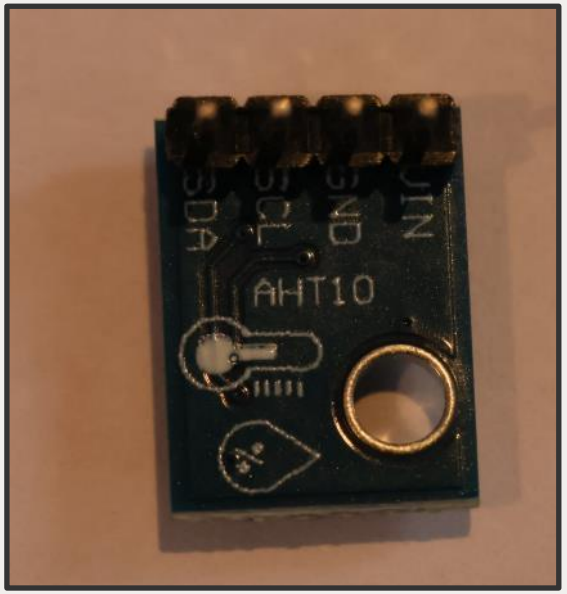
Mesures

Comment ont été faites les mesures, quels sont les résultats?

Température

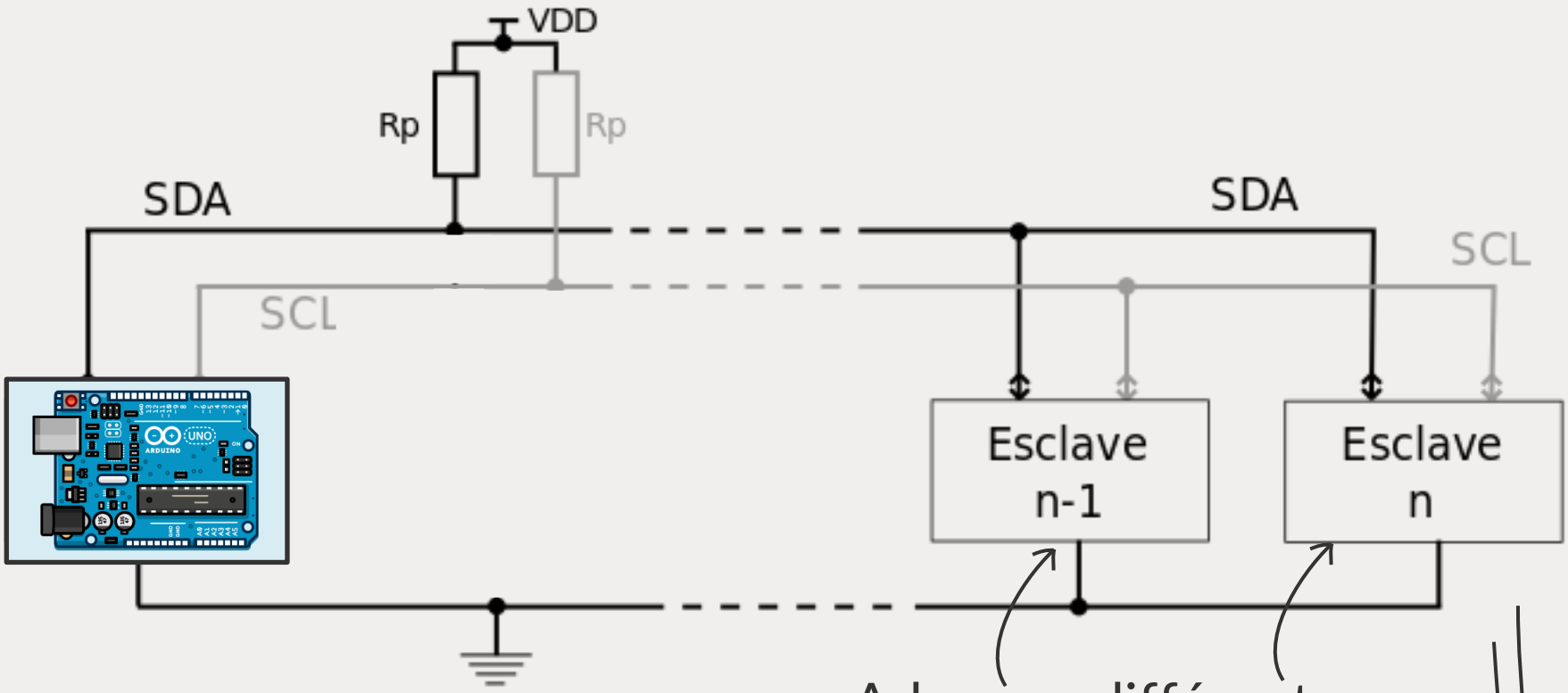
Ports bidirectionnels

Humidité



Ports I2C

Capteur d'humidité: connections

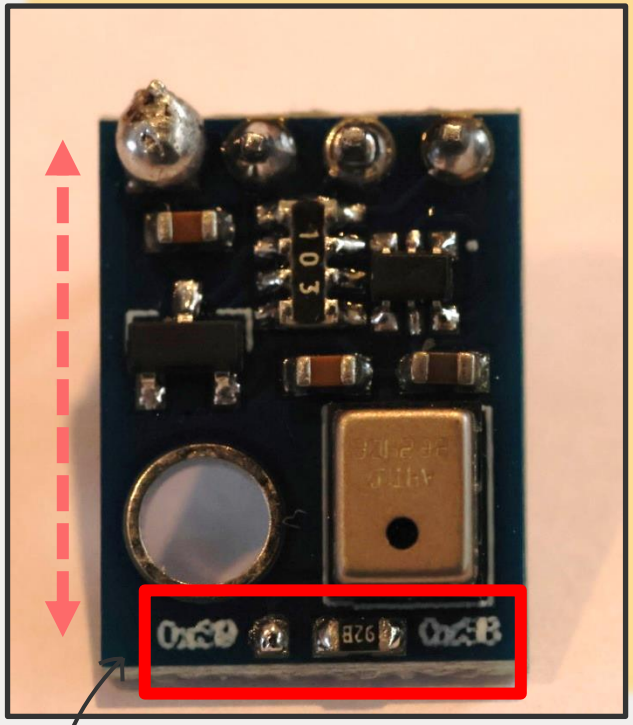


Adresses différentes

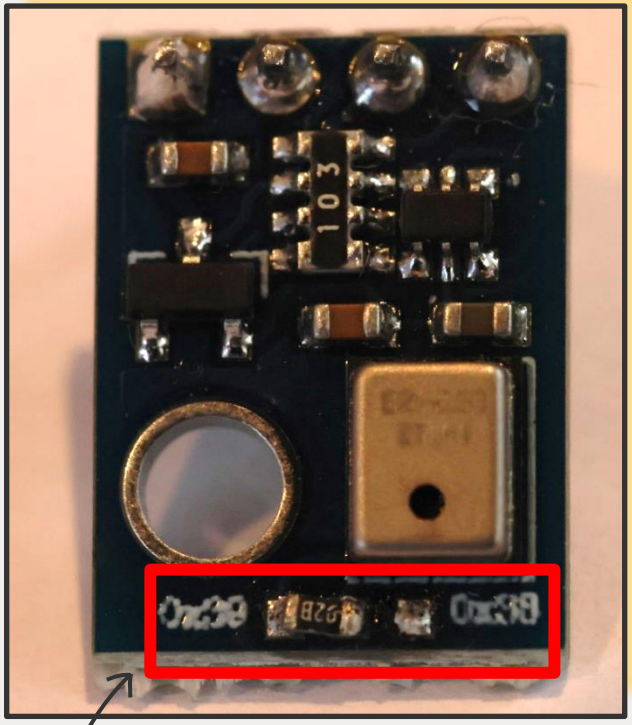
- Numériques
- Physiques

Capteurs d'humidités: adresses

~1cm

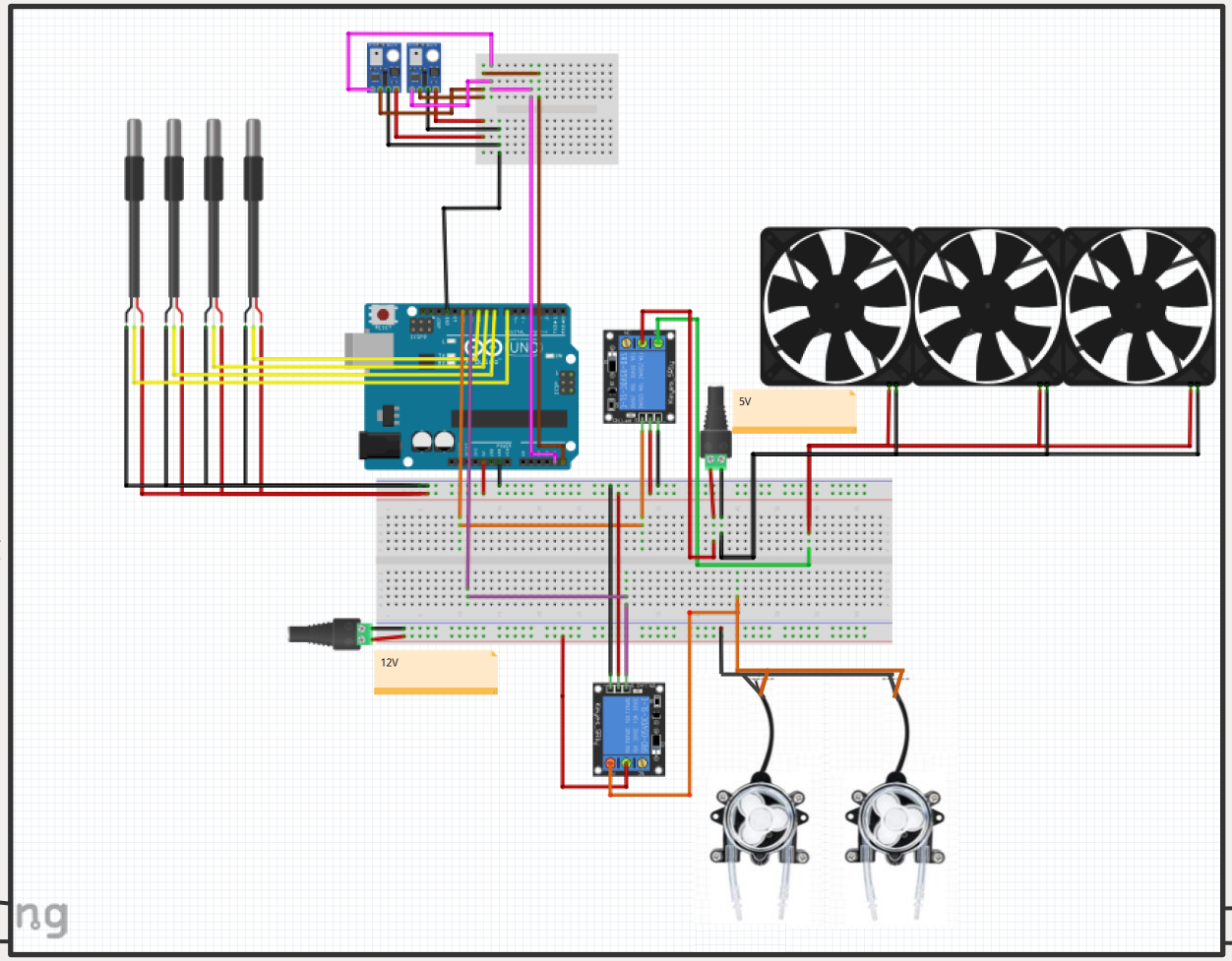


Ox39



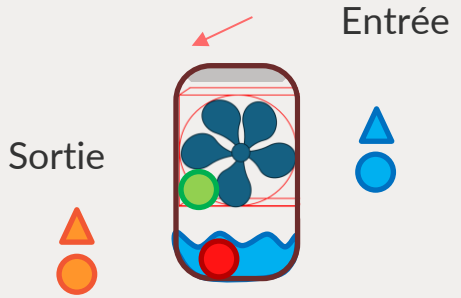
Ox38

Capteurs: connections



Réalisé avec Fritzing

Grandeurs mesurées, capteur

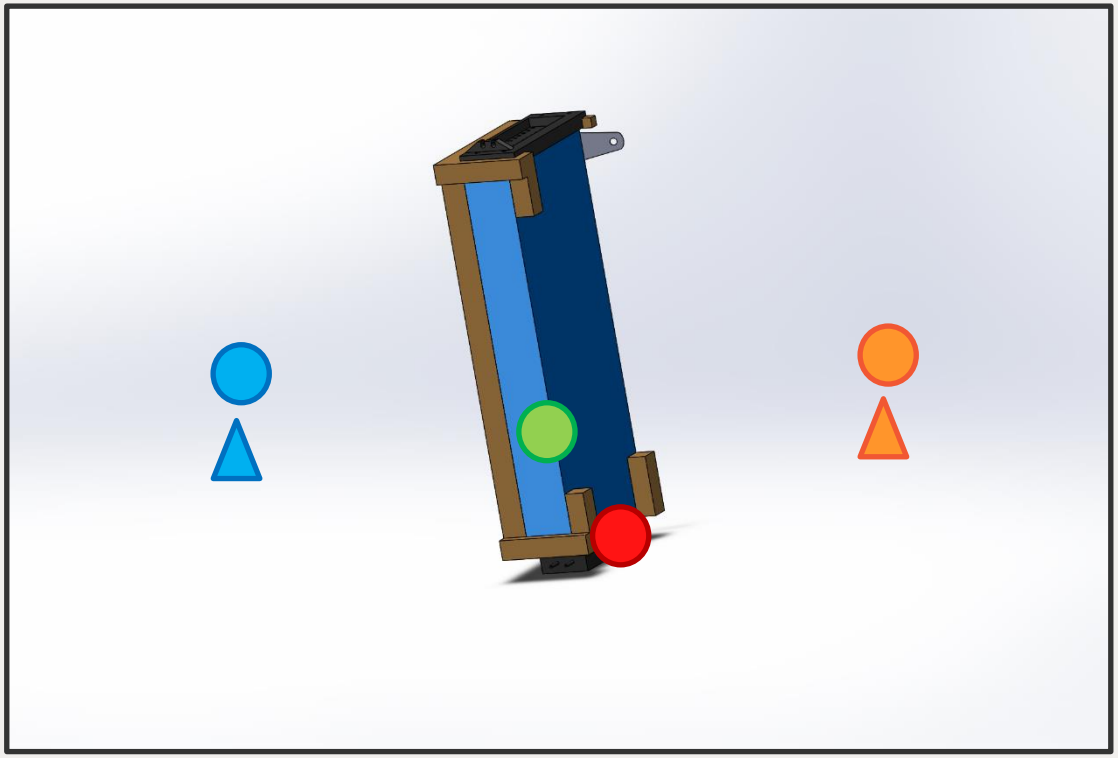


Humidité:

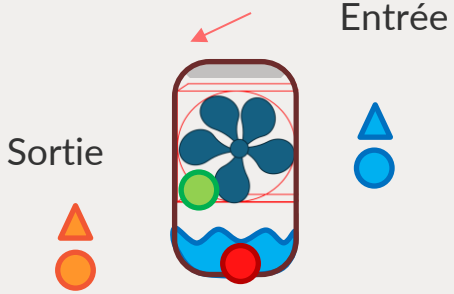
- Entrée
- Sortie

Température:

- Entrée
- Sortie
- Eau
- Tampon(IN)



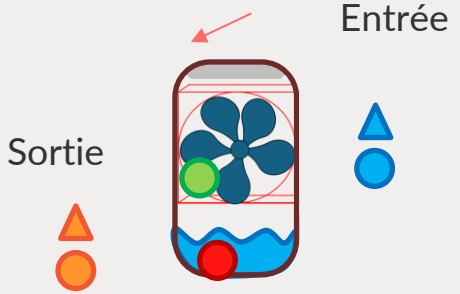
Dispositif expérimental



Humidité:
- Entrée
- Sortie

Température:
- Entrée
- Sortie
- Eau
- Tampon(IN)

Dispositif expérimental



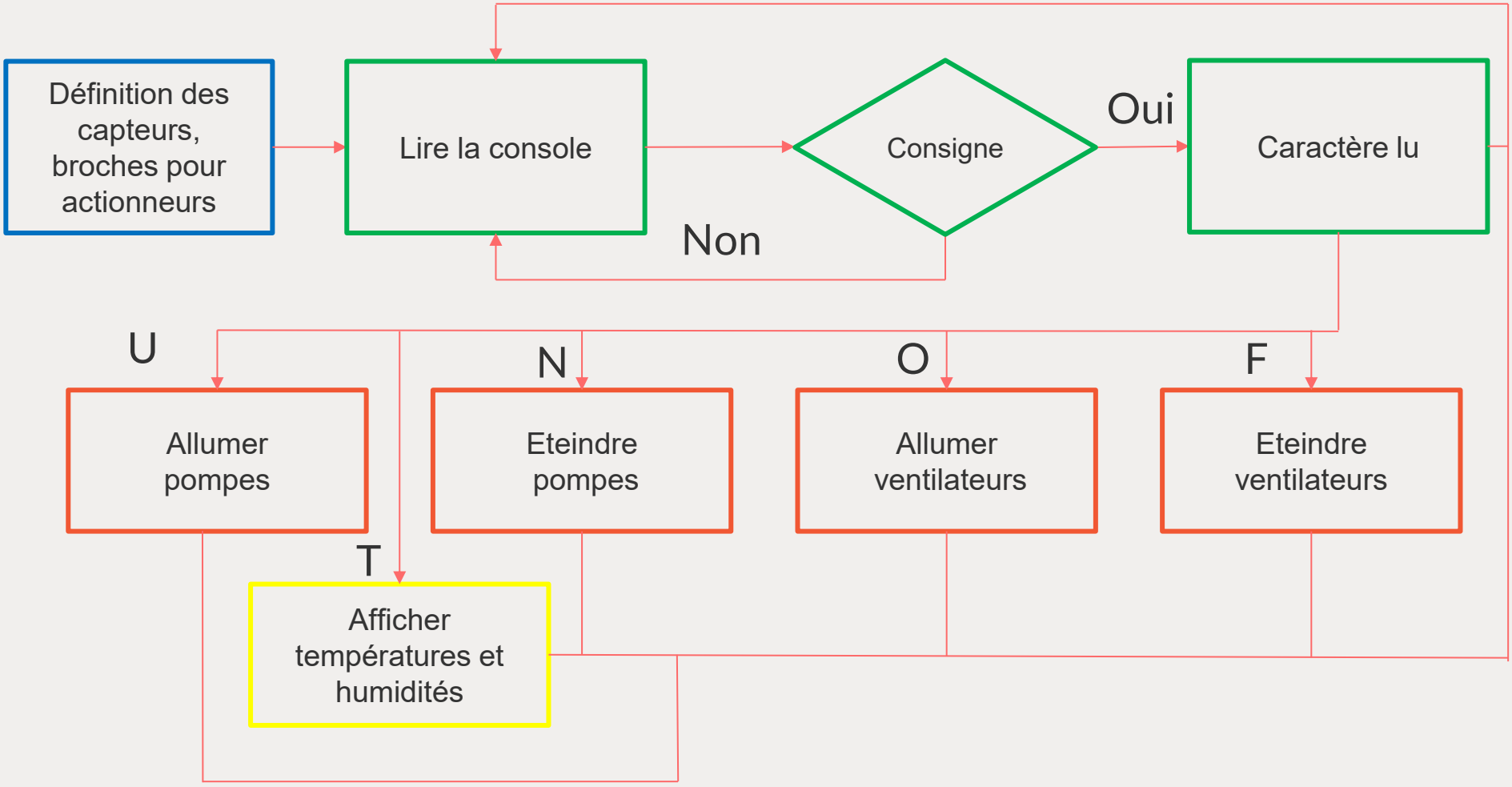
Humidité:

- Entrée
- Sortie

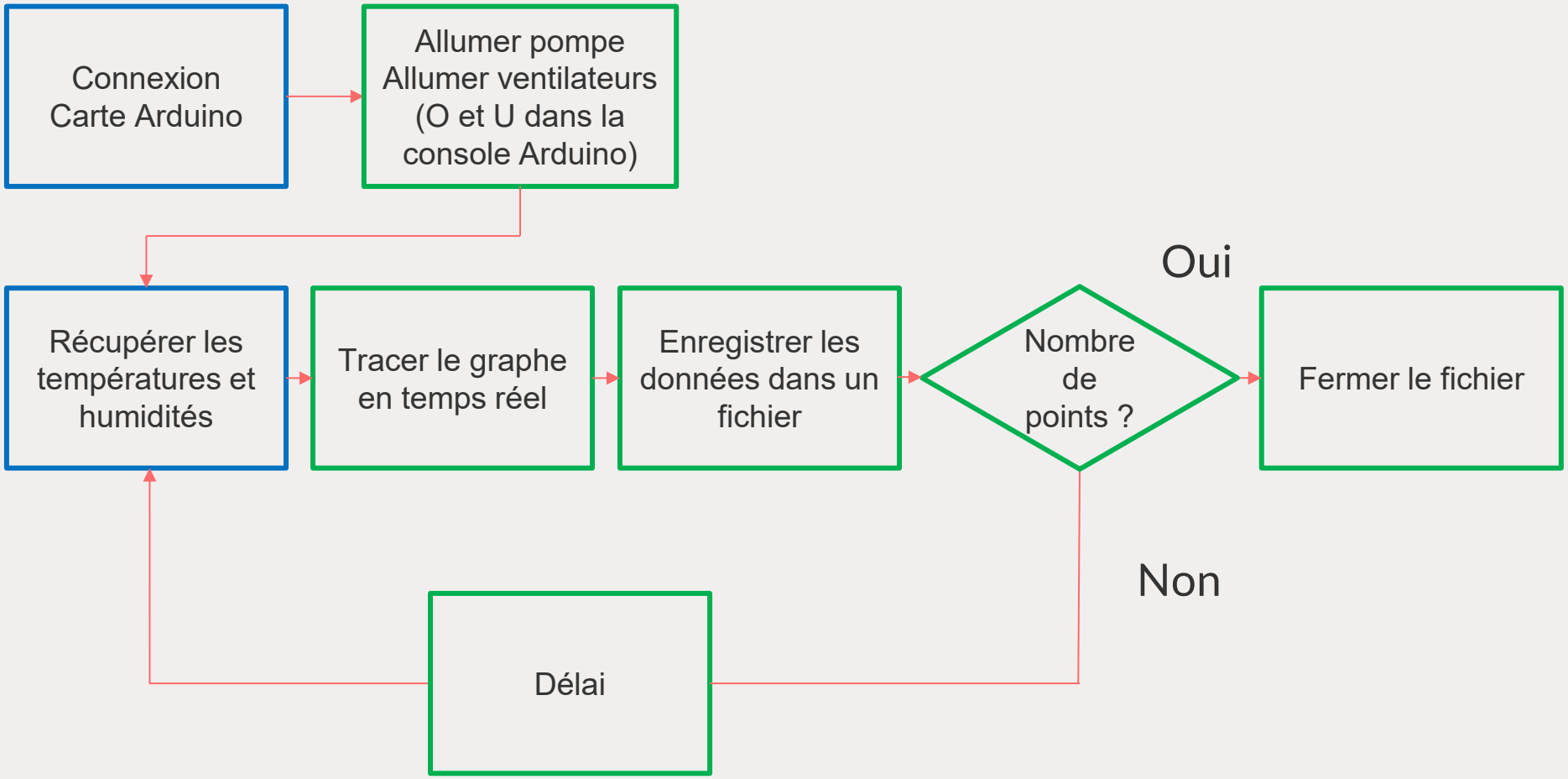
Température:

- Entrée
- Sortie
- Eau
- Tampon(IN)

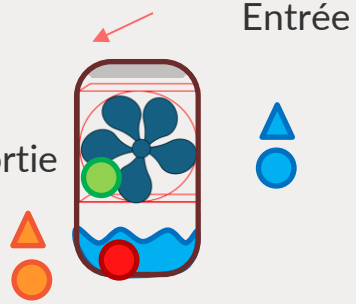
Logigramme Arduino (système pas automatisé)

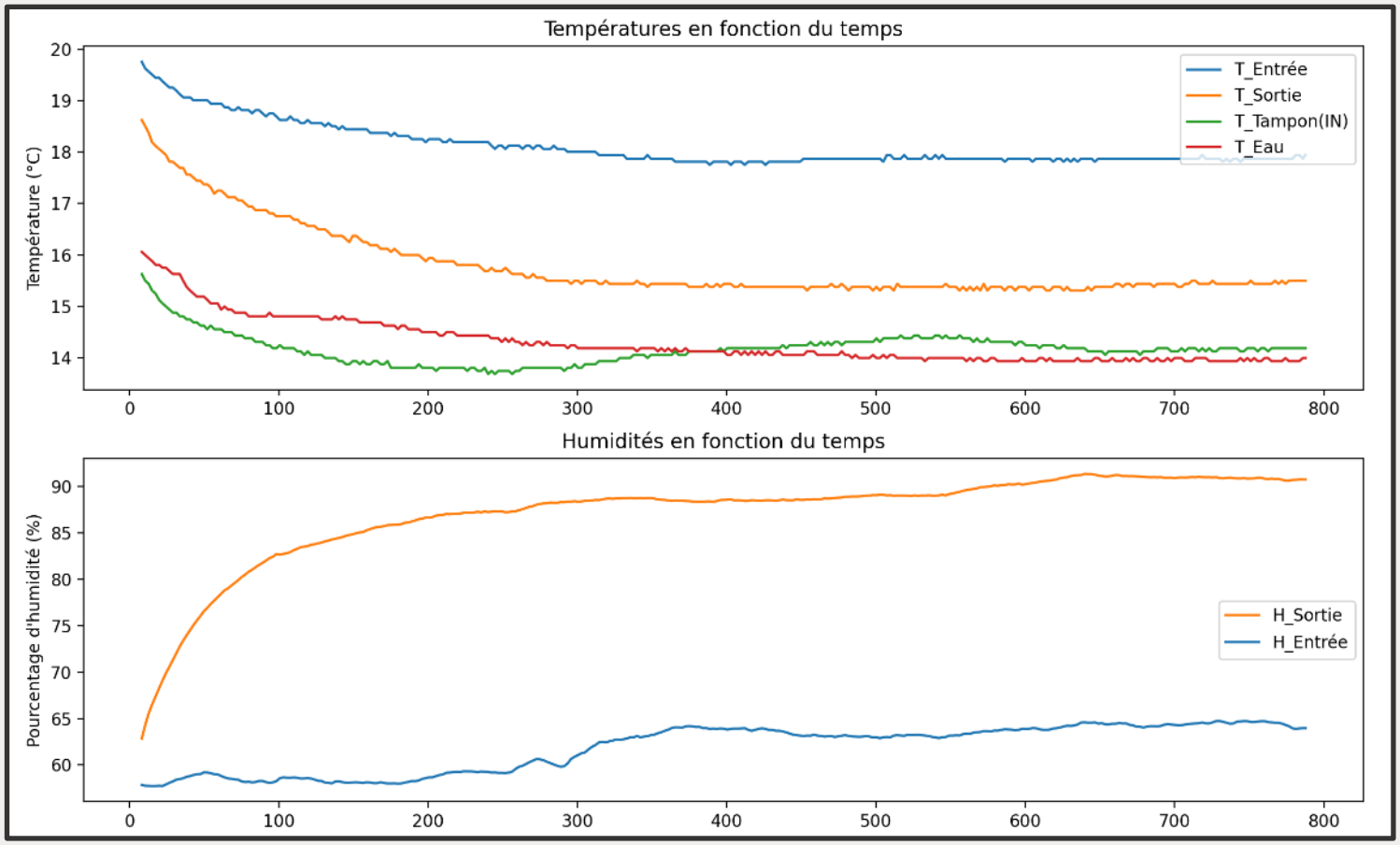


Logigramme Python: sans asservissement

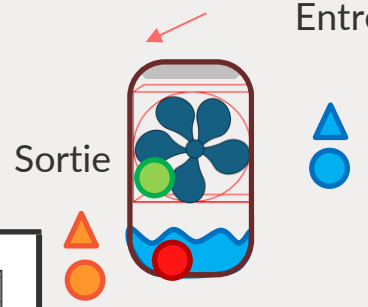


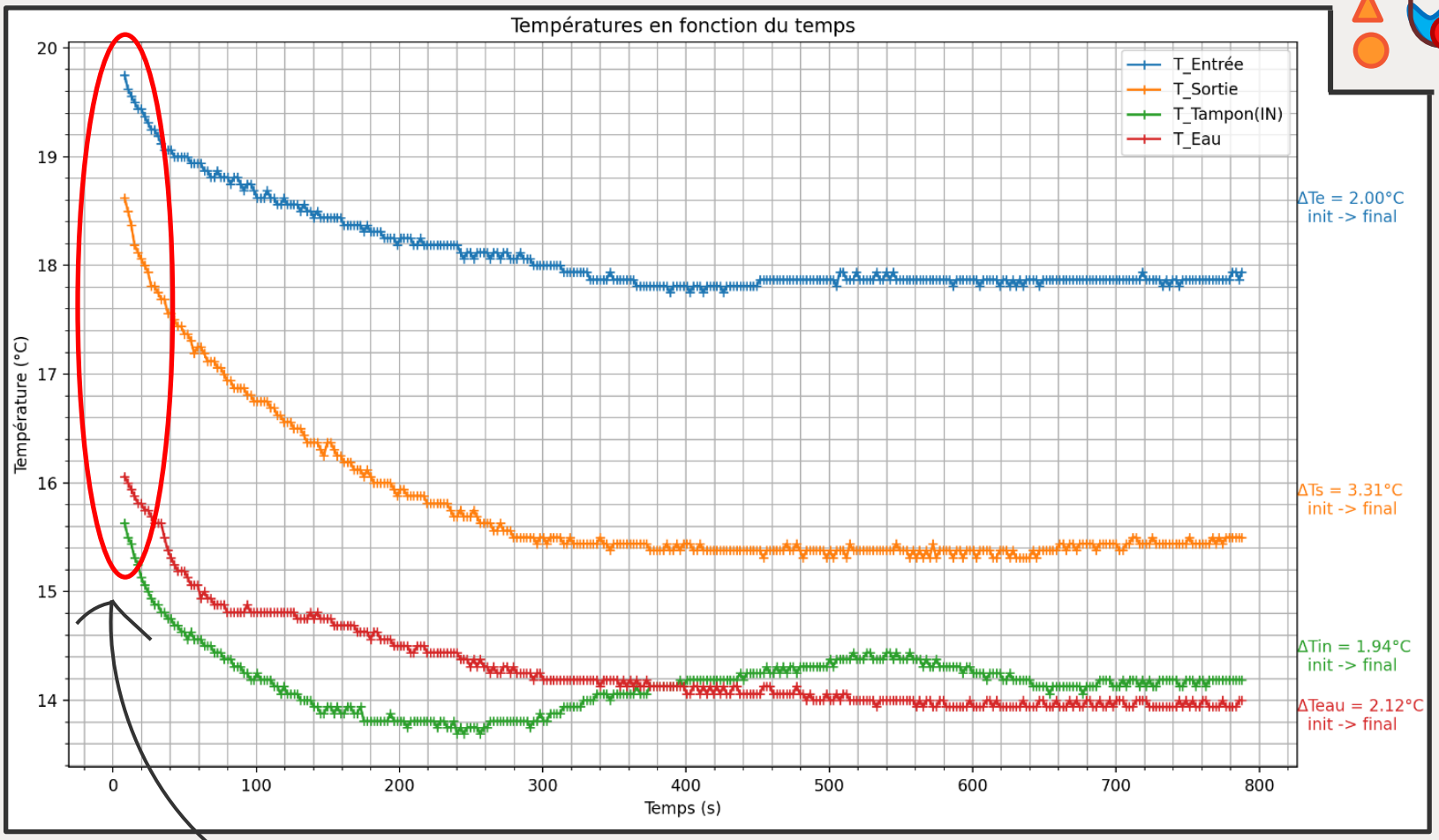
Un début...


Entrée
Sortie


Températures en fonction du temps
Température (°C)
T_Entrée
T_Sortie
T_Tampon(IN)
T_Eau
Humidités en fonction du temps
Pourcentage d'humidité (%)
H_Sortie
H_Entrée

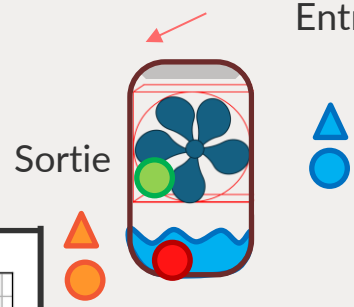
Réglage du biais de mesure

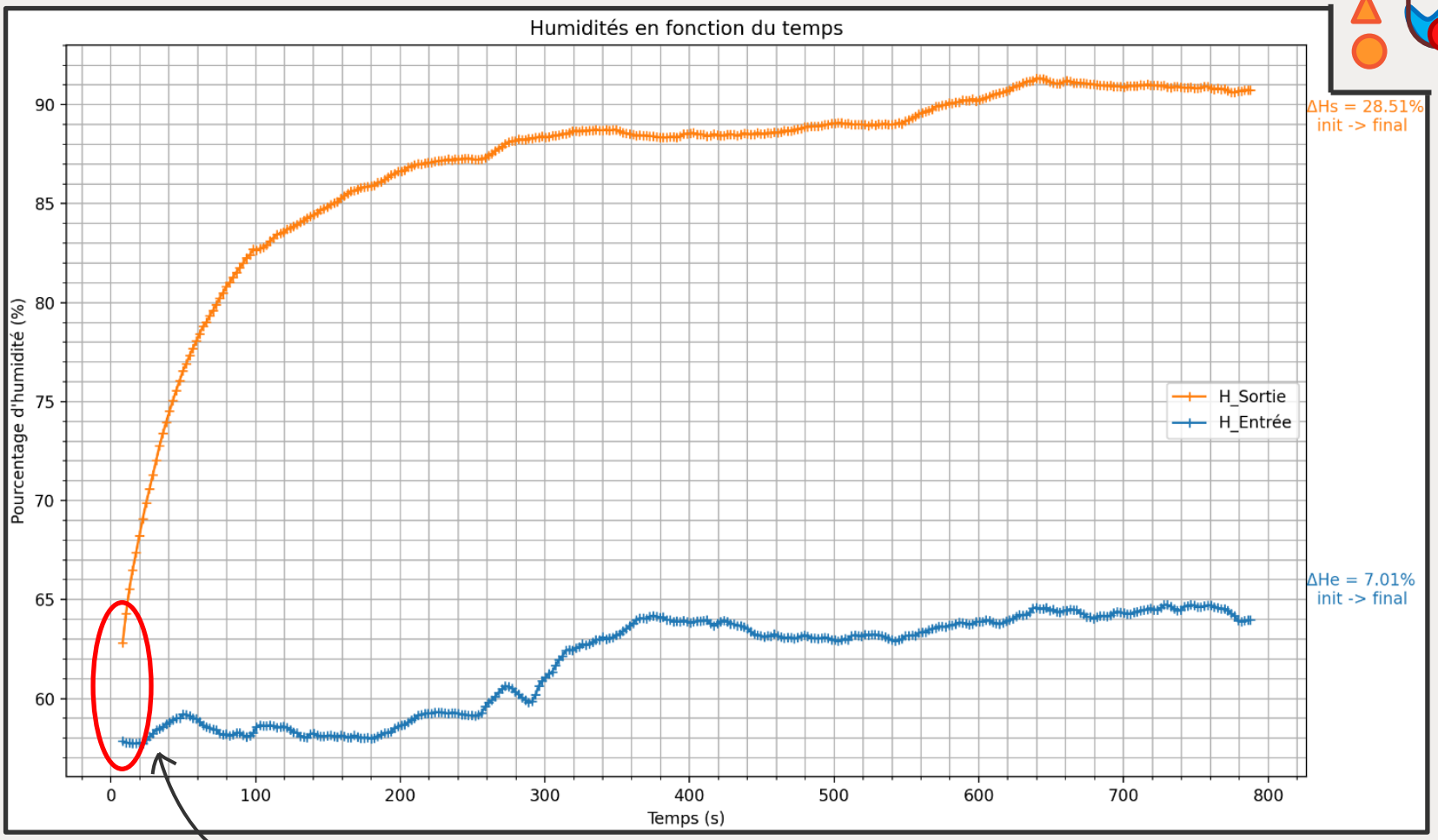




Non calibrés

Réglage du biais de mesure





Non calibrés

Capteurs:

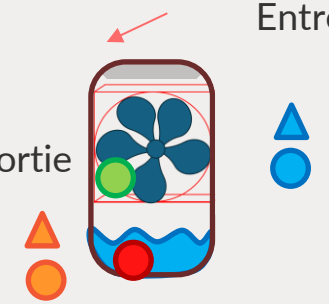
-Température ±0.1°C

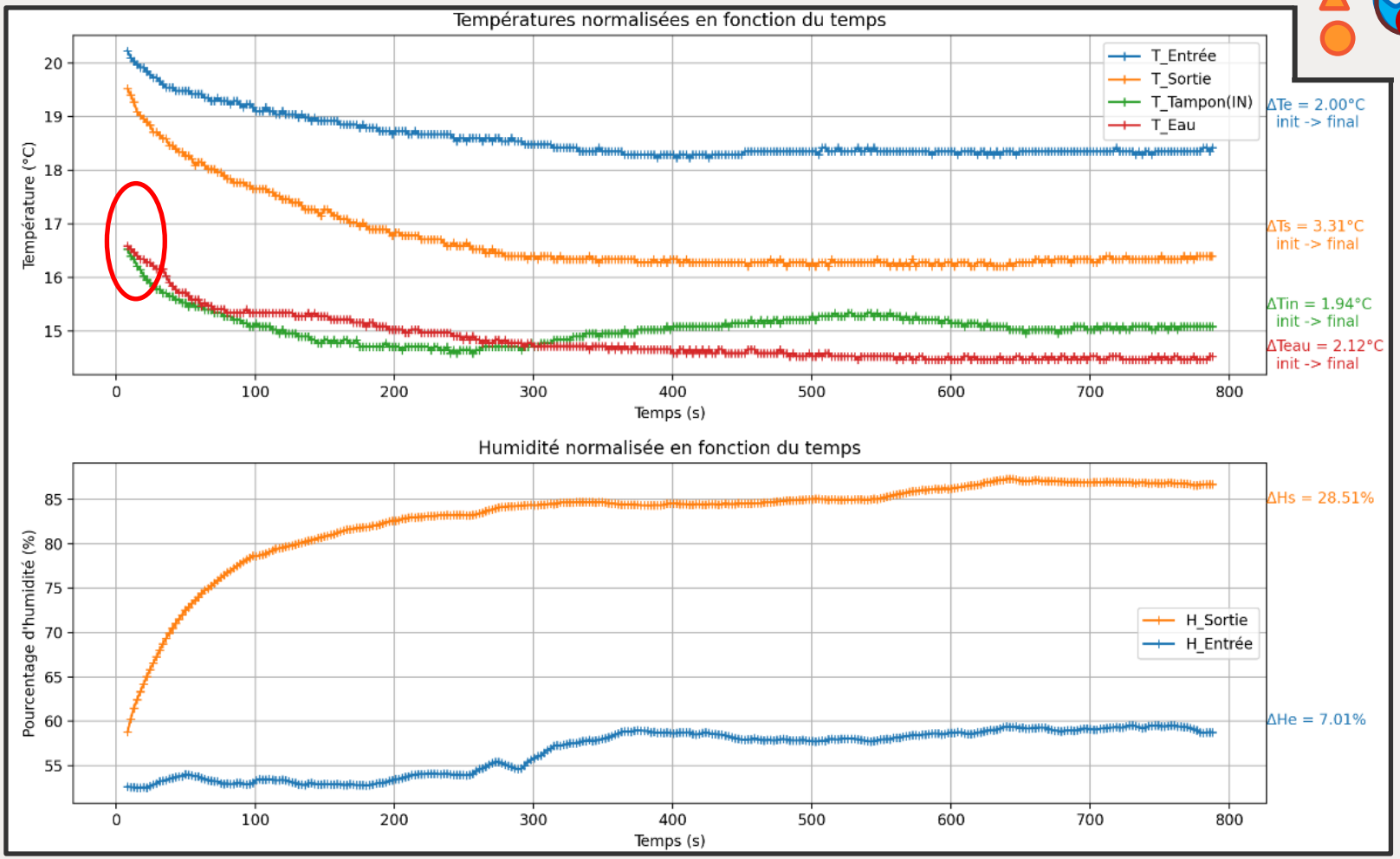
-Humidité ±2%

```
64  # Normalisation
65  T1_dec = 0.899999999999986
66  T2_dec = 0.529999999999976
67  T3_dec = 0.779999999999976-0.3
68  T4_dec = 0.899999999999986
69  T5_dec = 1.599999999999979-0.2
70  T7_dec = 1.419999999999982-0.2
71
72  H6_dec = -5.189999999999998
73  H8_dec = -4.039999999999999
74
```

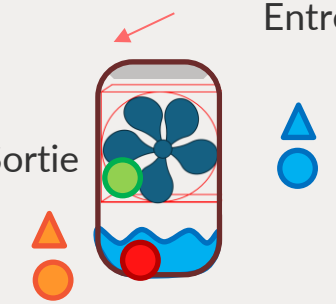
Capteurs pas étalonnés d'usine

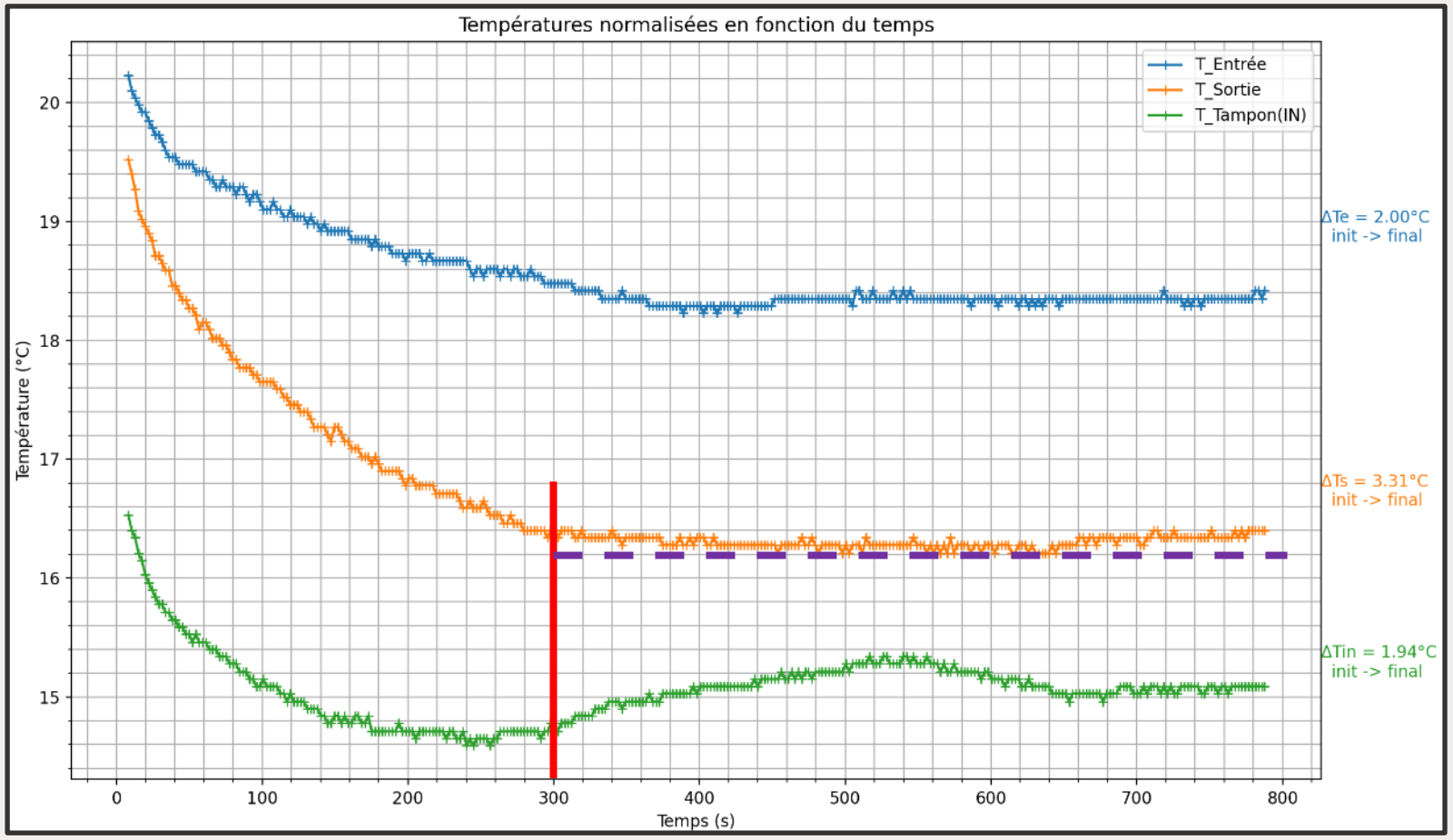
Mesures: normalisées



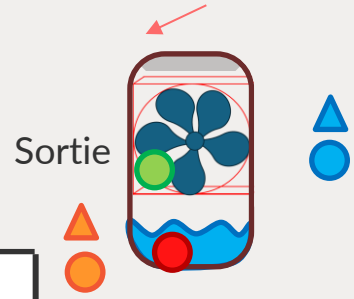


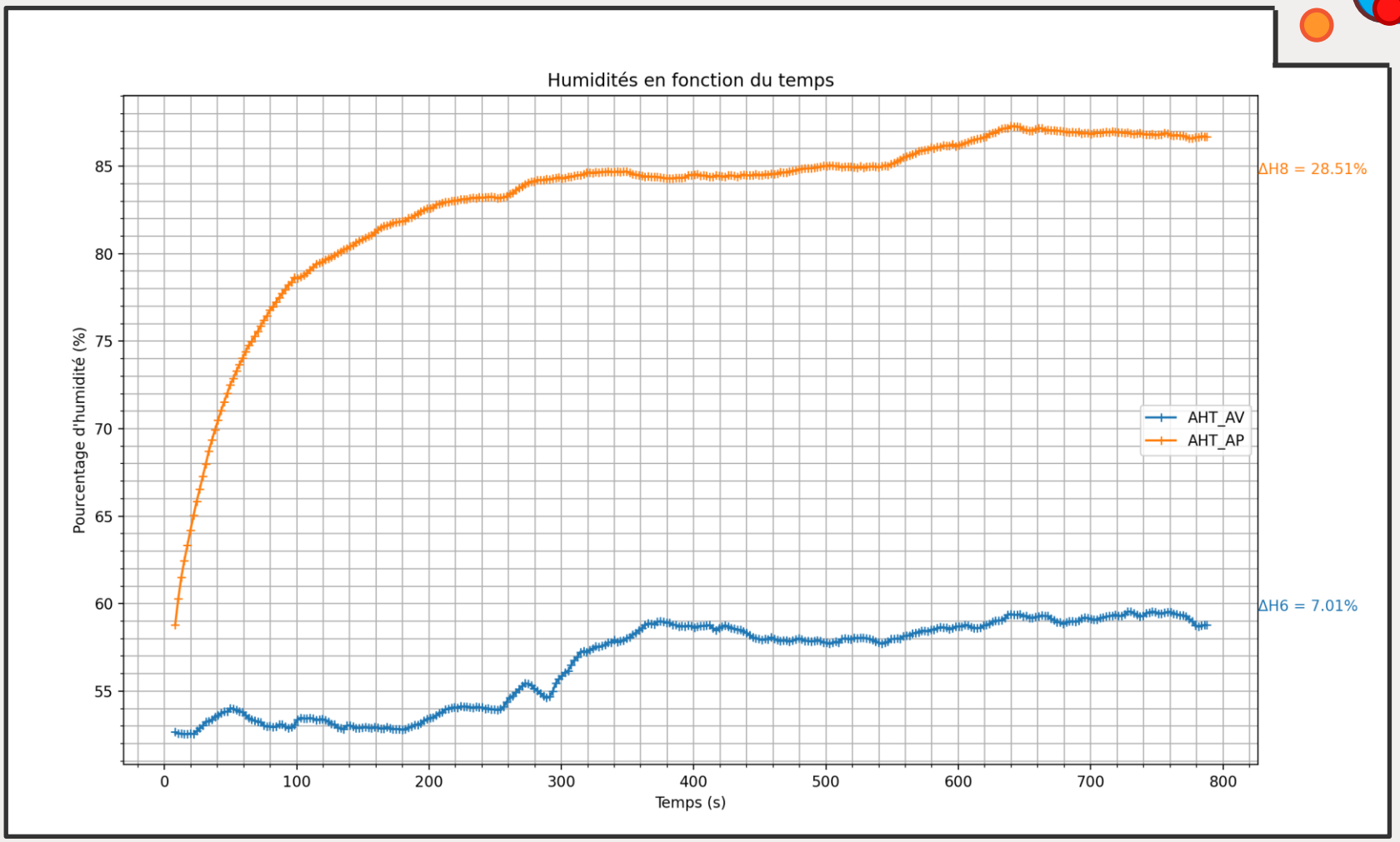
Températures normalisées



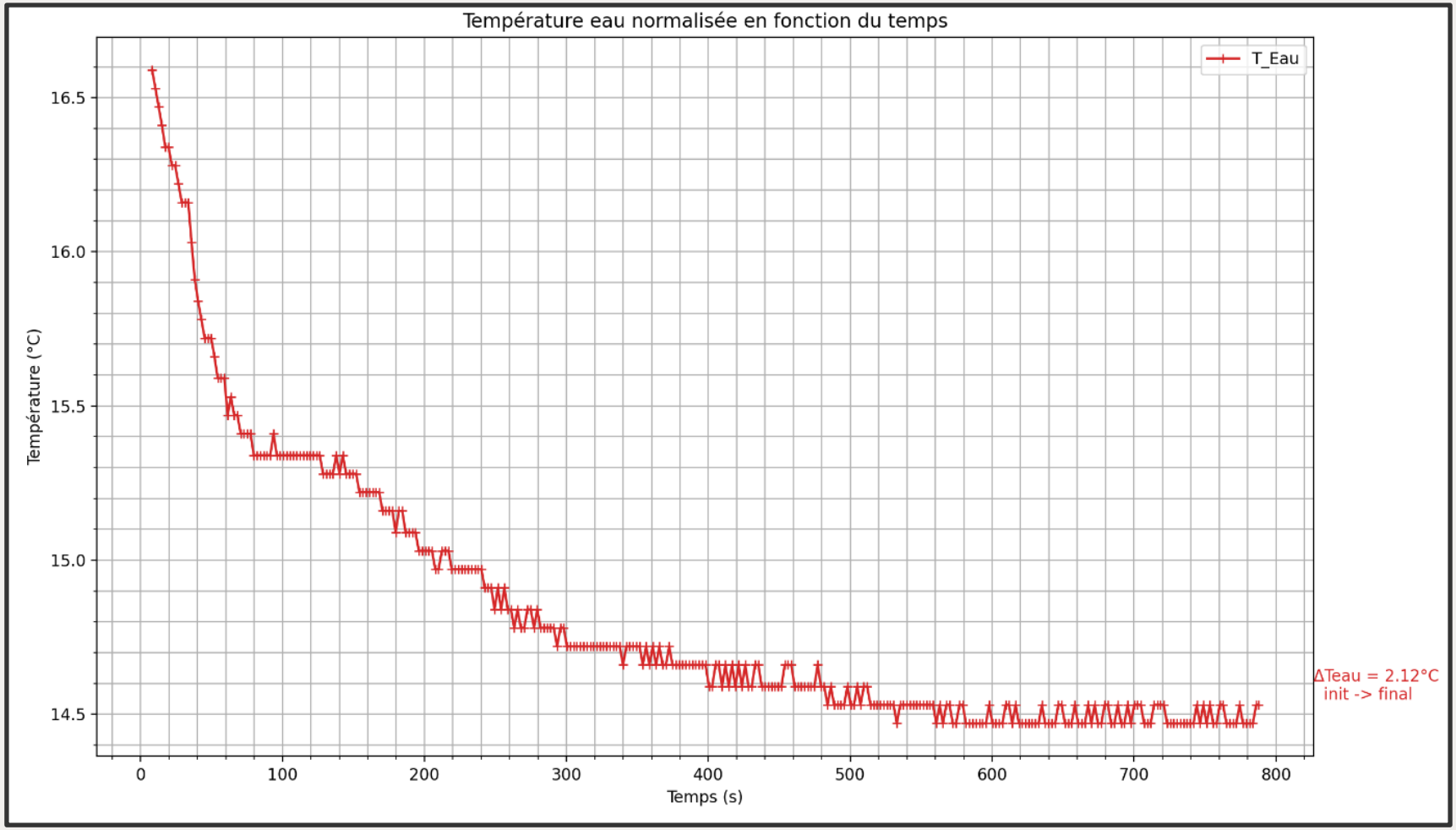


Humidités normalisées

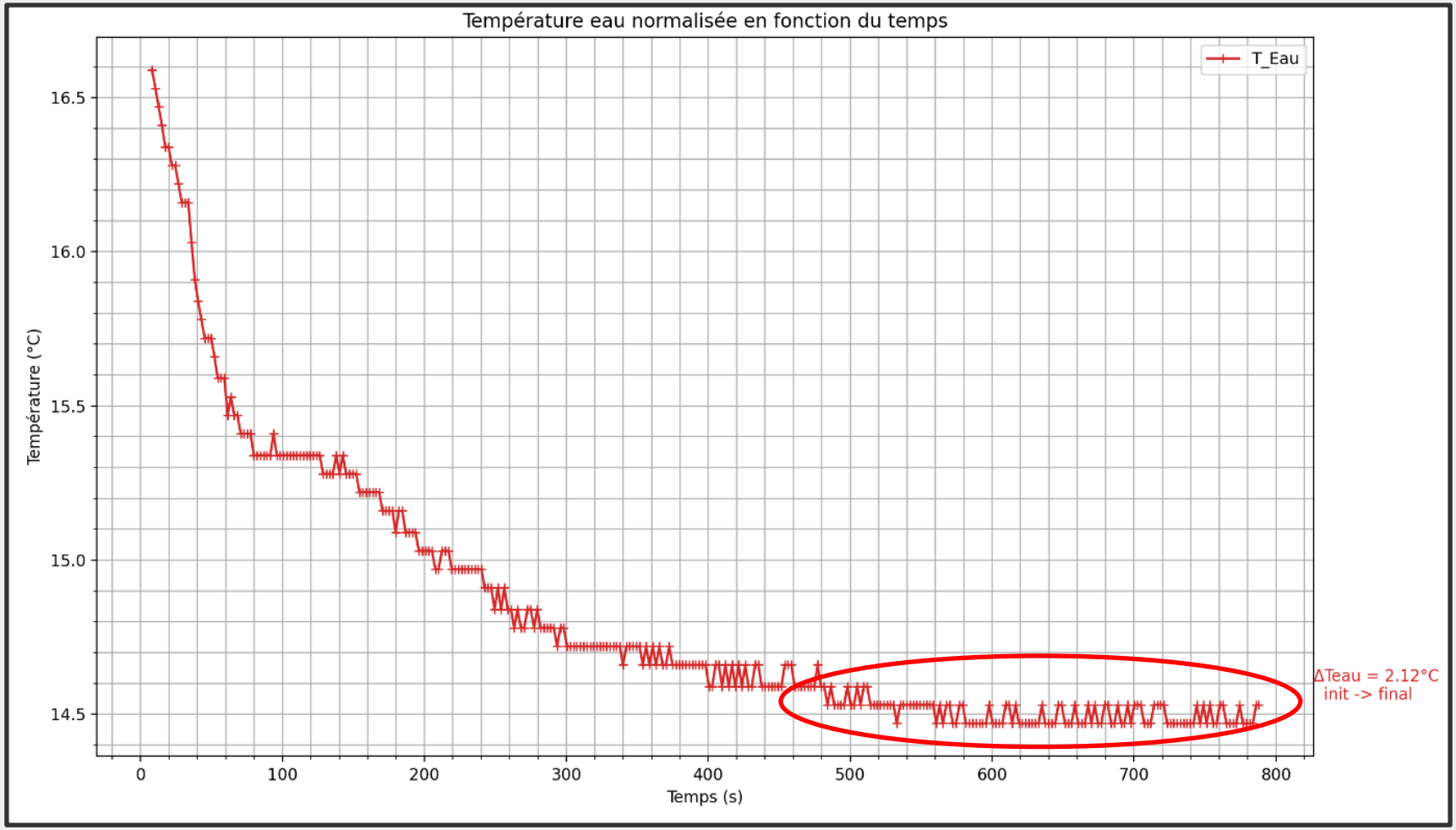




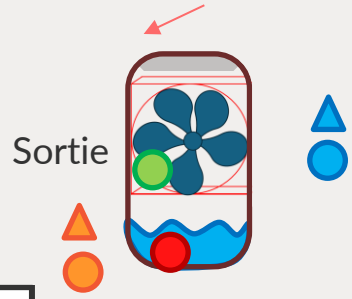
Température de l'eau

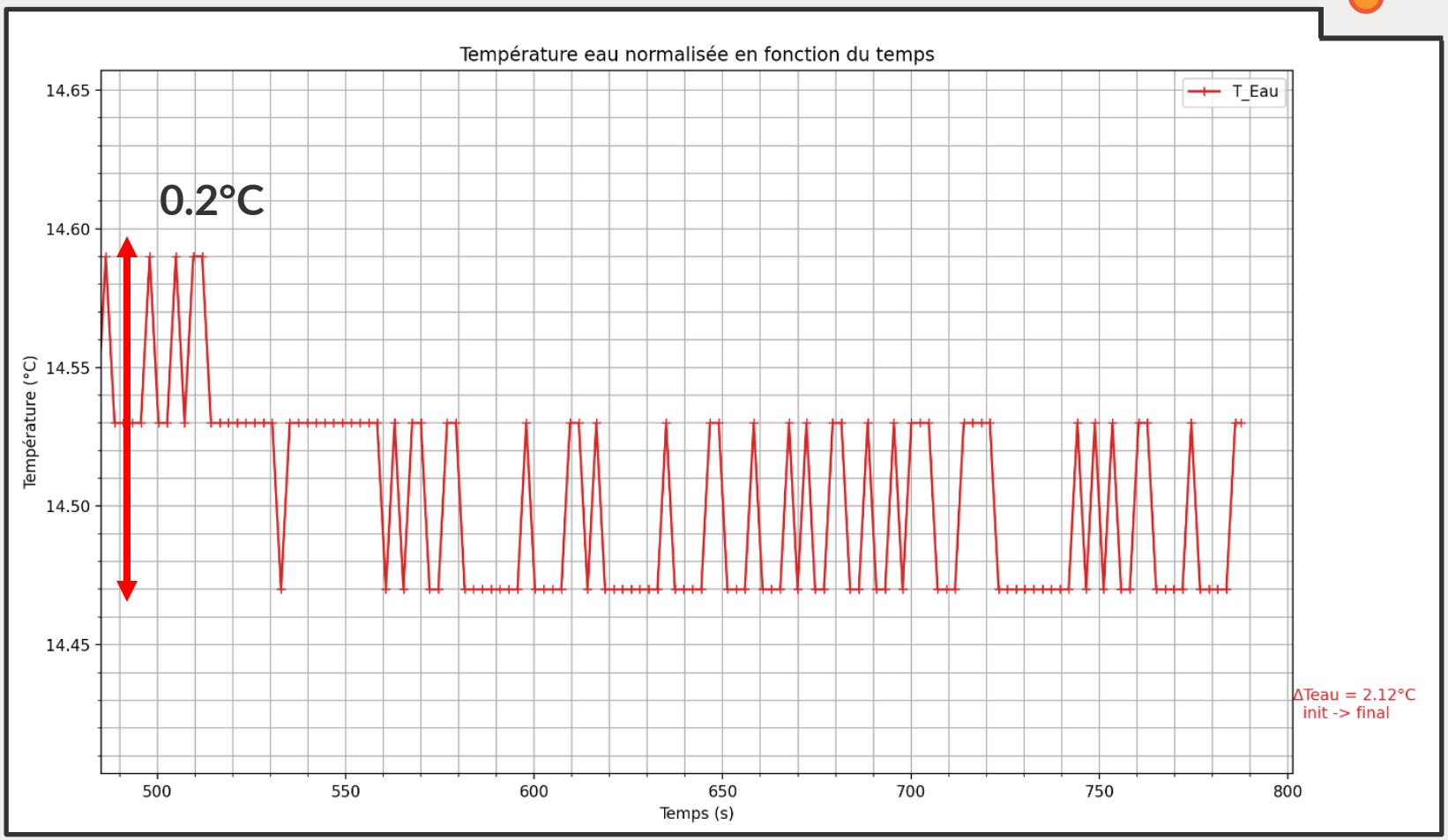


Température de l'eau

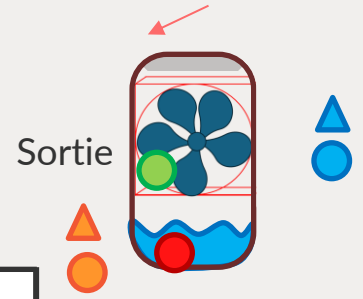


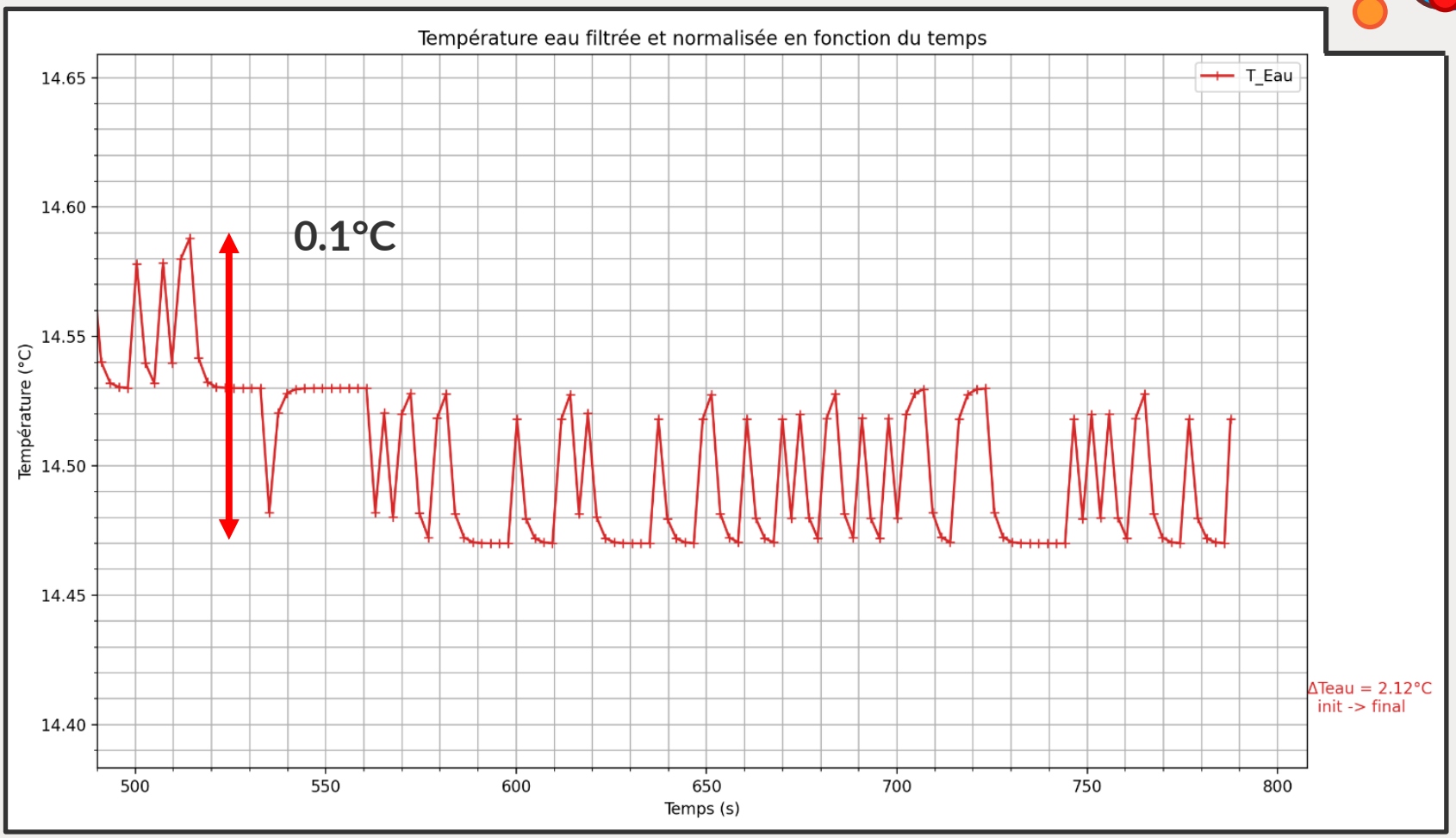
Température de l'eau





Résultats du filtre

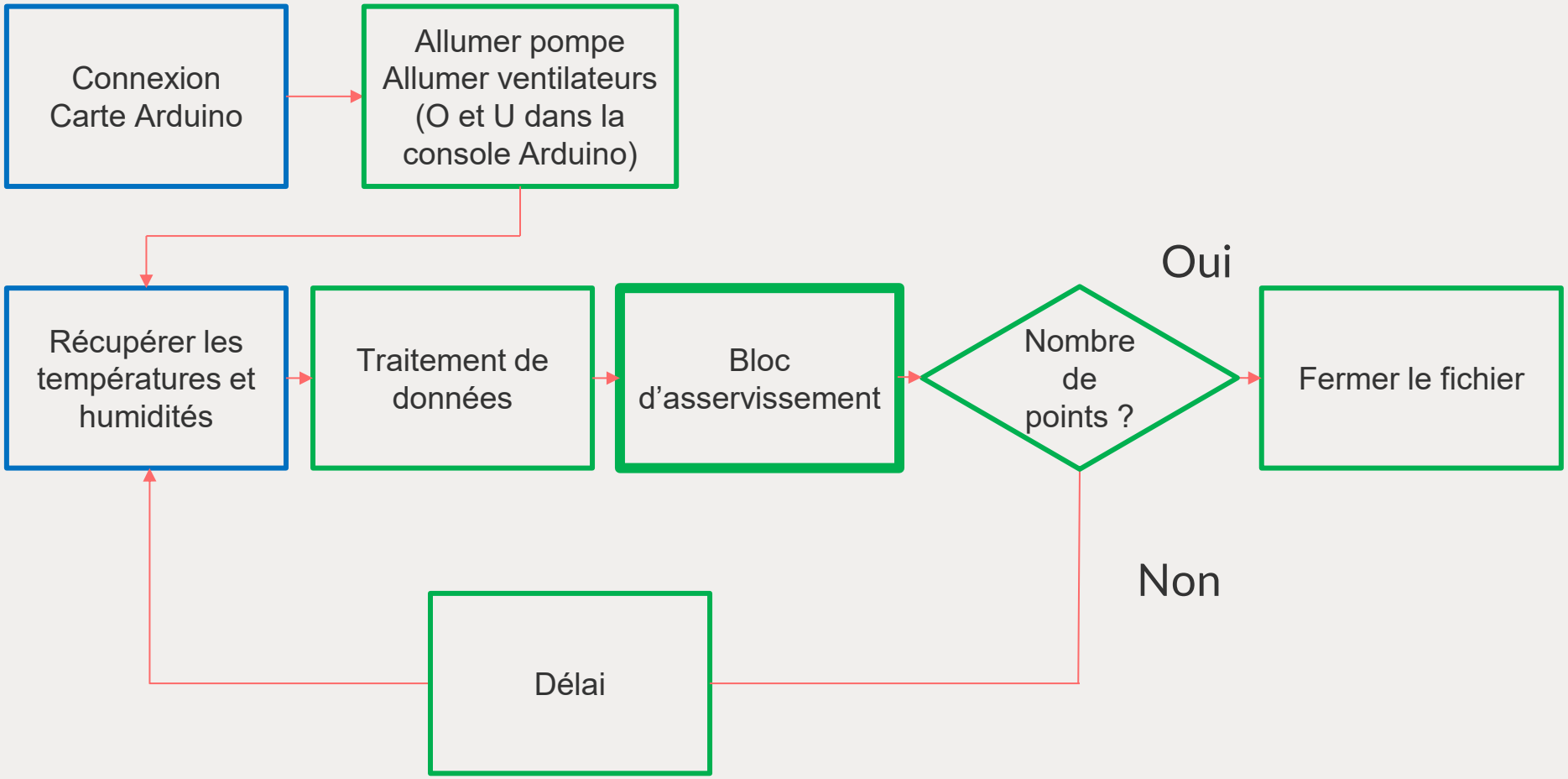




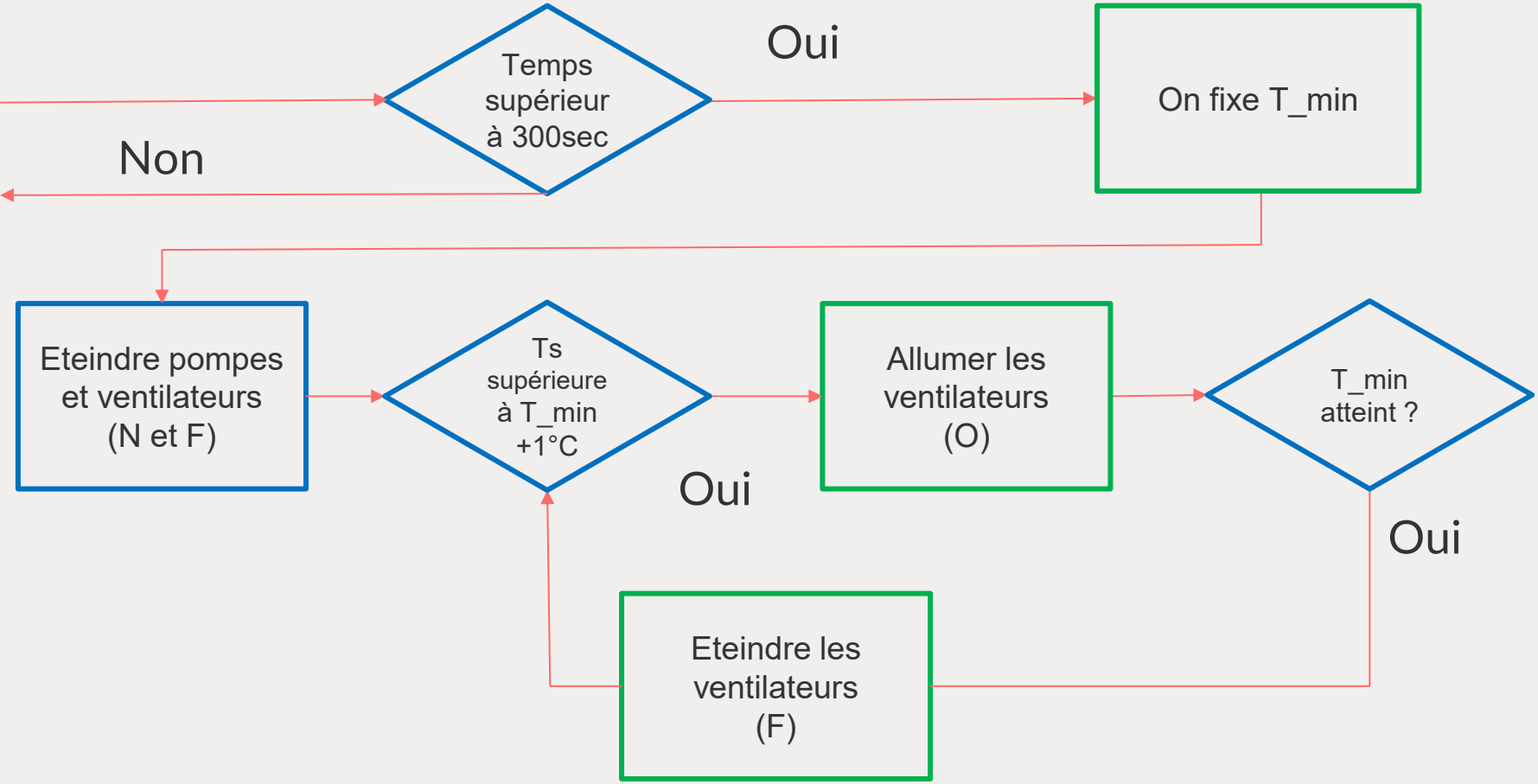
Résultats de la première mesure

Difficultés	Solution
Biais de mesure	Offset des capteurs
Précision	Moins de dispersion
Asservissement (PYTHON)	----------

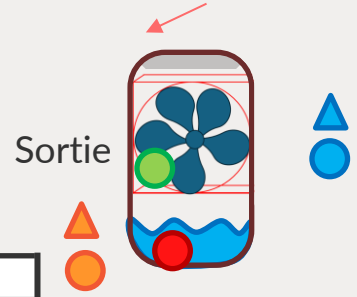
Logigramme Python: asservissement

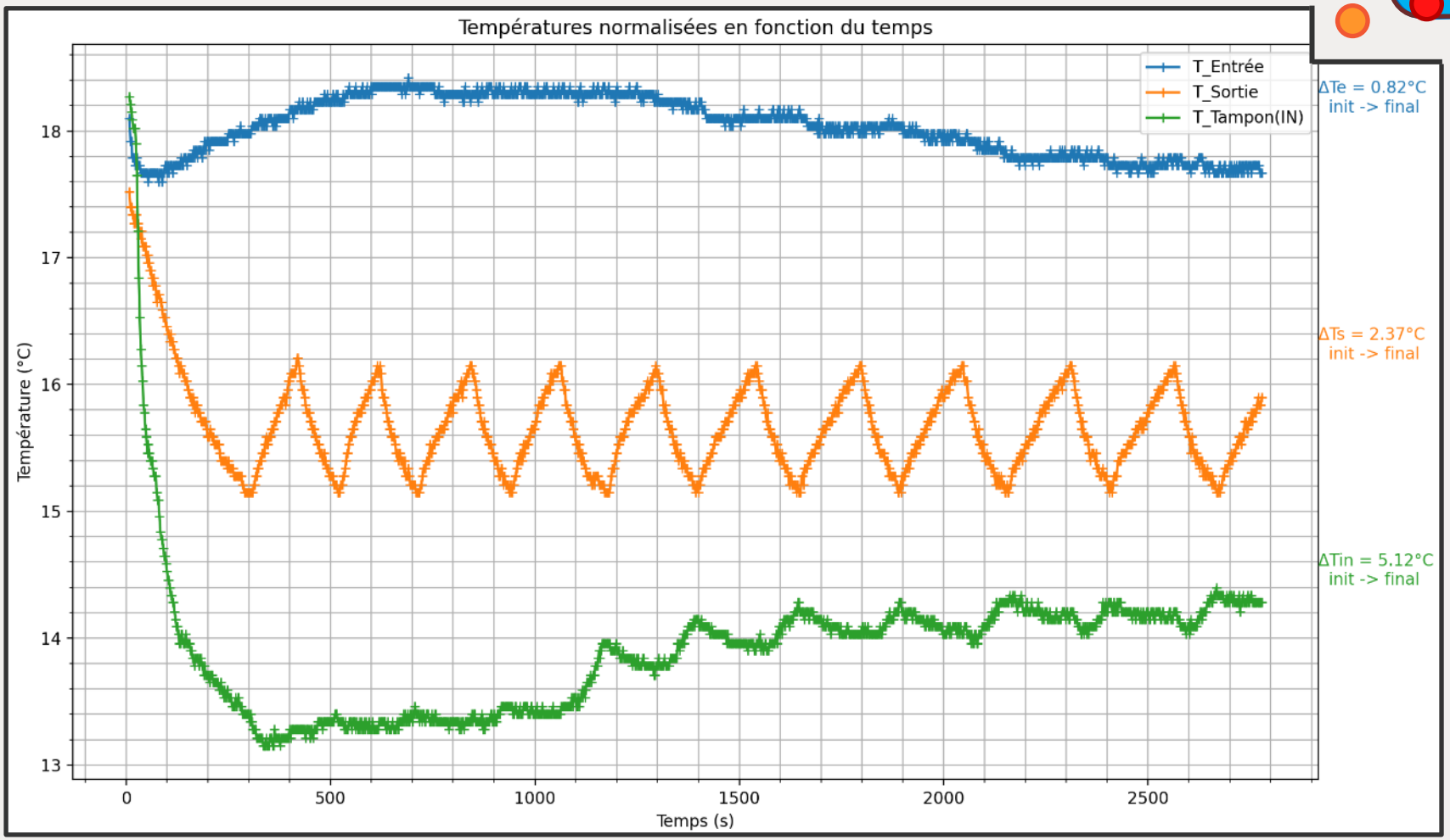


Algorithme Python: bloc d'asservissement

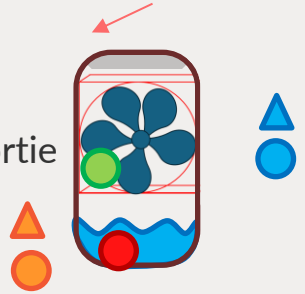


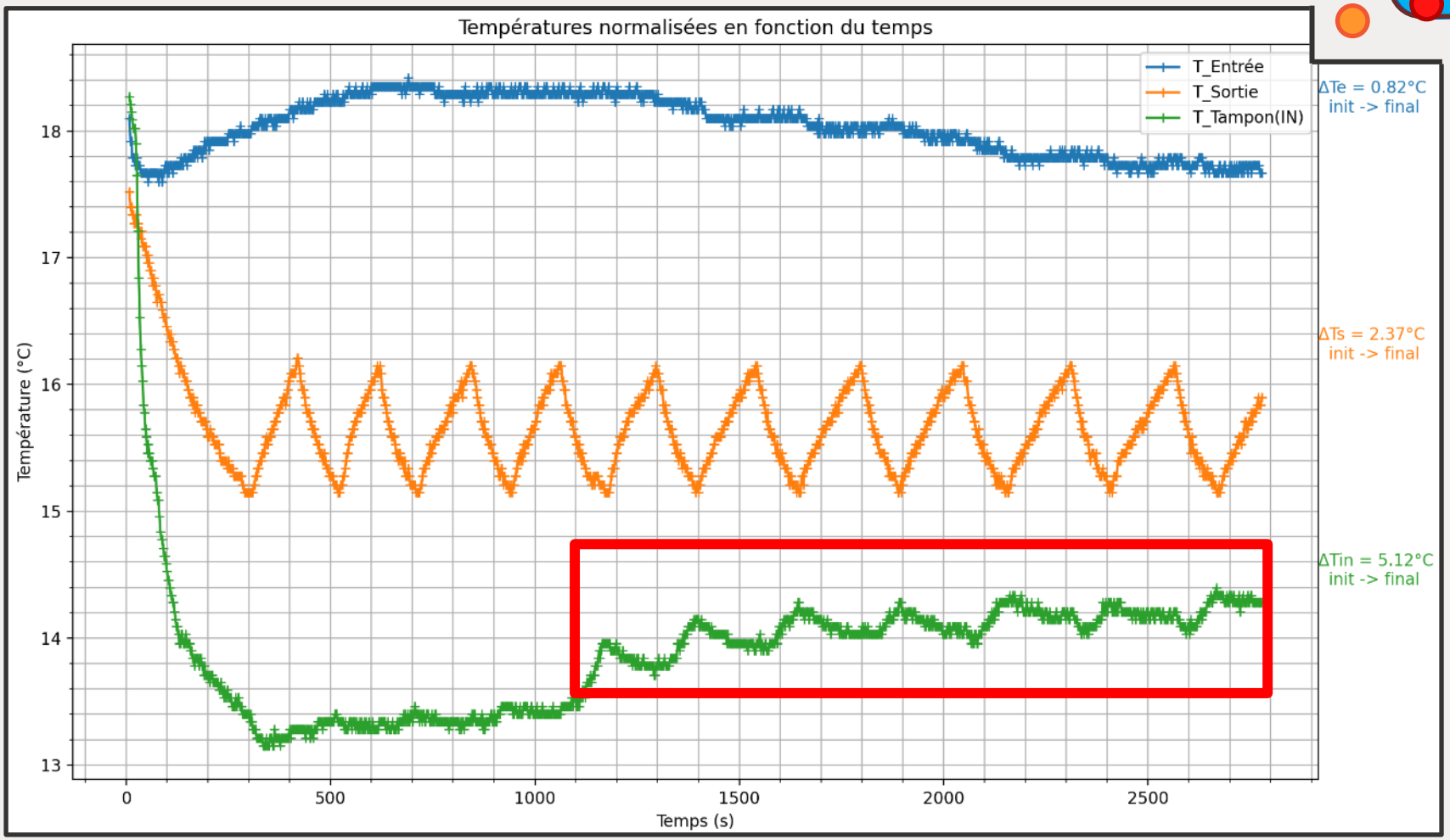
Températures





Températures: ajustement





Réarrangement de l'asservissement

Besoin de pouvoir
humidifier tampon

→

Contrôle des pompes

Réarrangement de l'asservissement

Besoin de pouvoir humidifier tampon → Contrôle des pompes

Modification automatique de la température de référence → Asservissement sur les itérations

Températures

Températures normalisées en fonction du temps

T_Entrée
T_Sortie
T_Tampon(IN)

ΔTe = 1.56°C
init -> final

ΔTs = 2.74°C
init -> final

ΔTin = 3.24°C
init -> final

Température (°C)

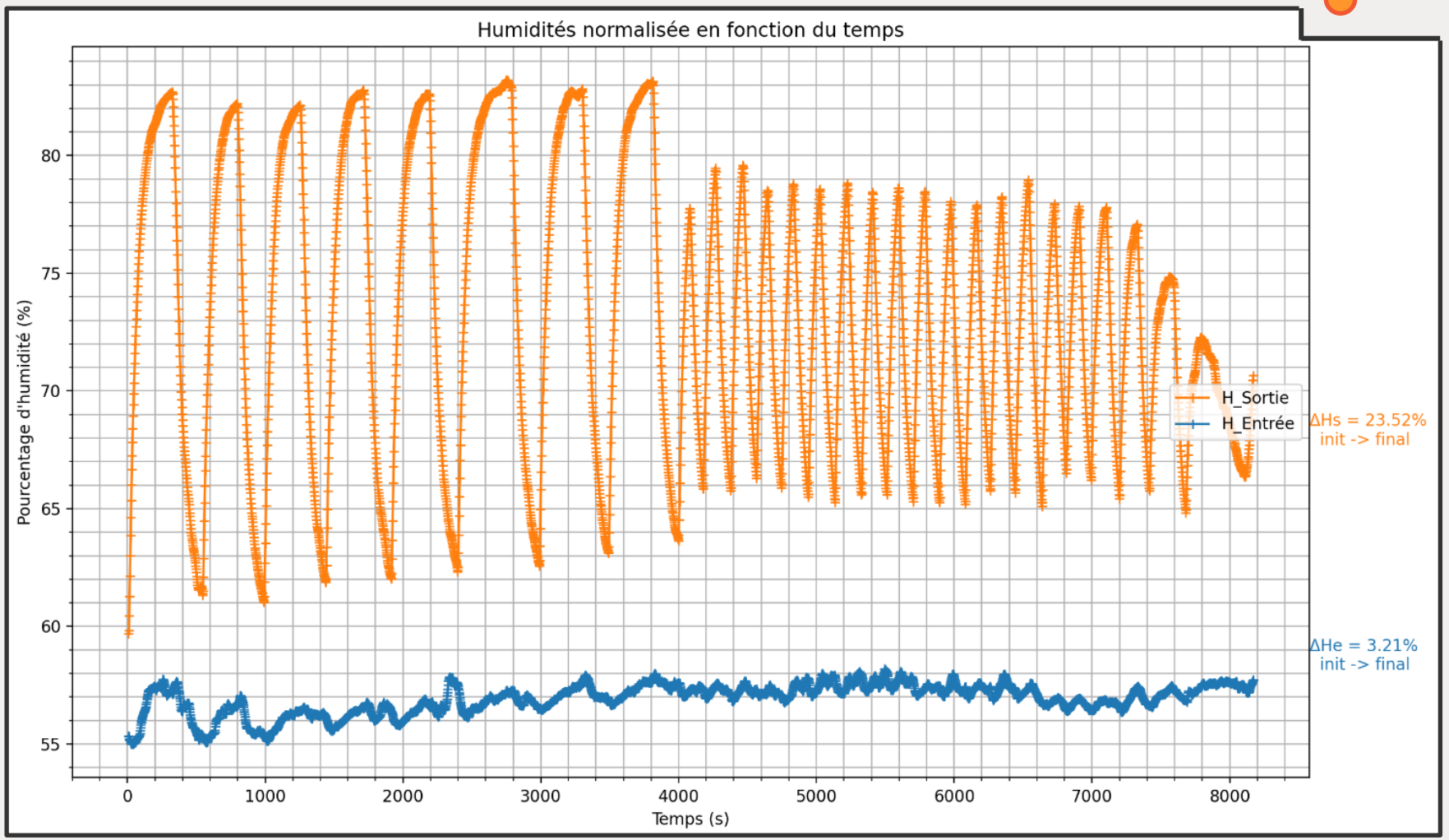
21
20
19
18
17

0
1000
2000
3000
4000
5000
6000
7000
8000

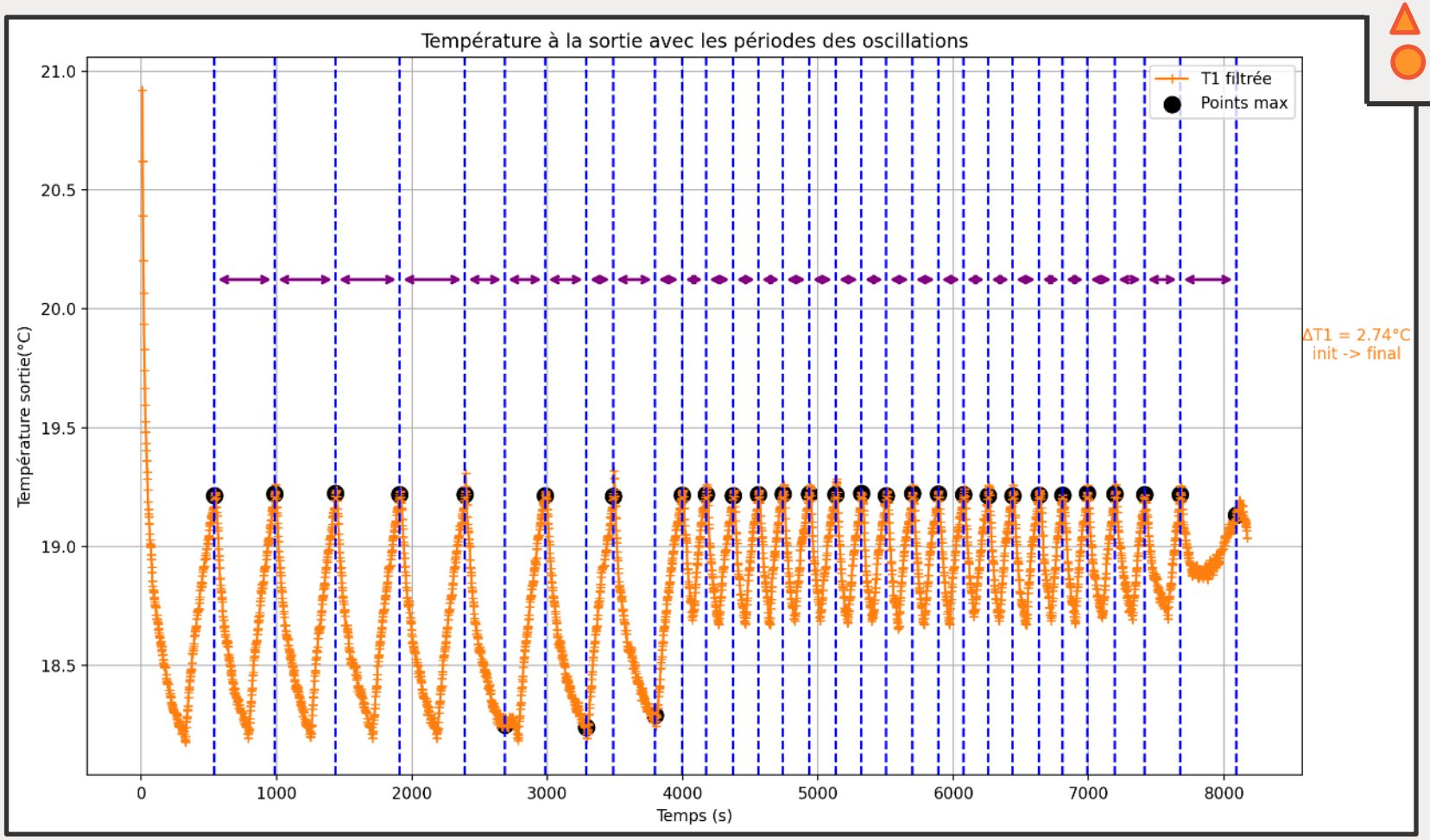
Temps (s)

Humidités

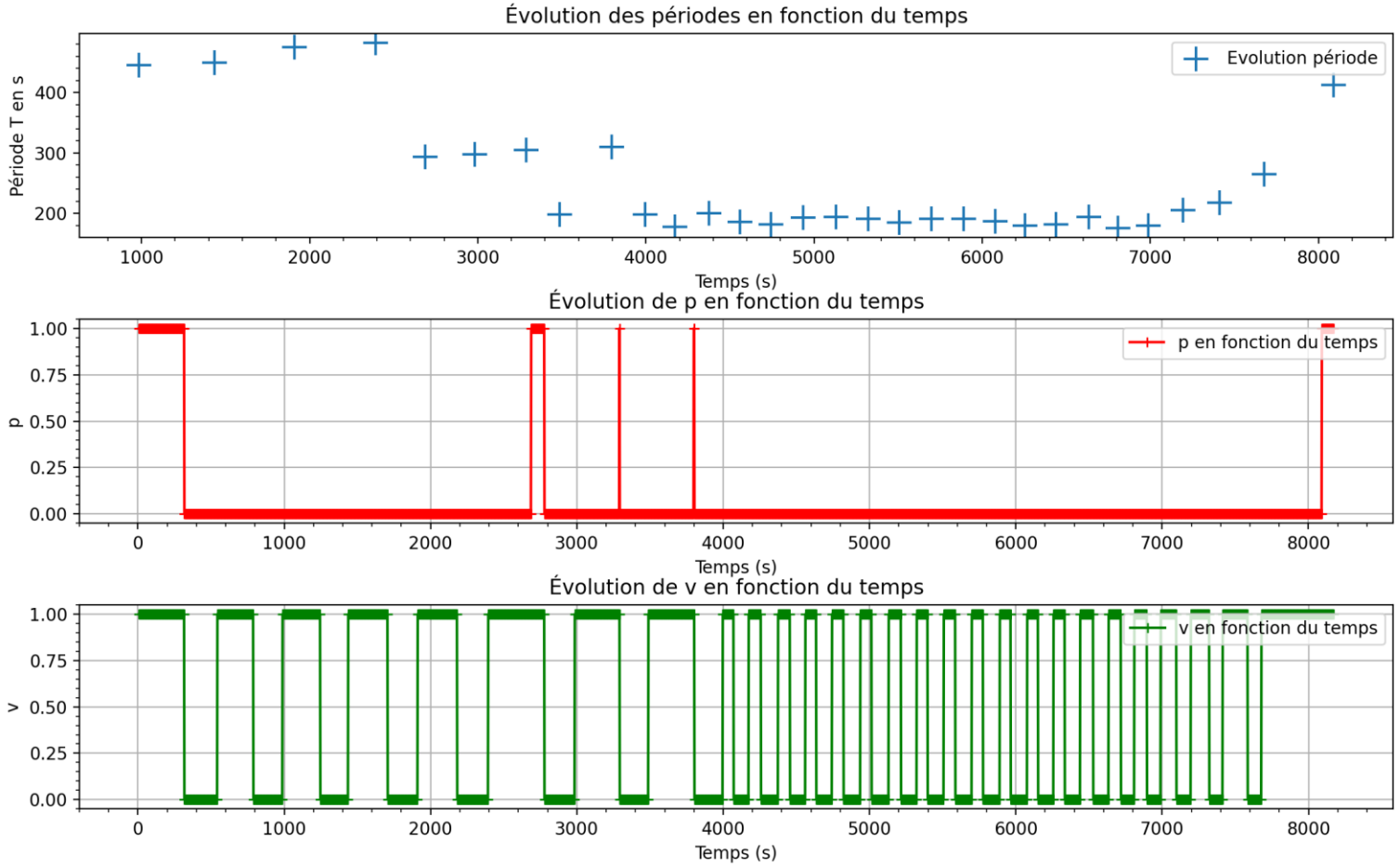

Sortie


Humidités normalisée en fonction du temps
Pourcentage d'humidité (%)
Temps (s)
H_Sortie
H_Entrée
ΔHs = 23.52%
init -> final
ΔHe = 3.21%
init -> final

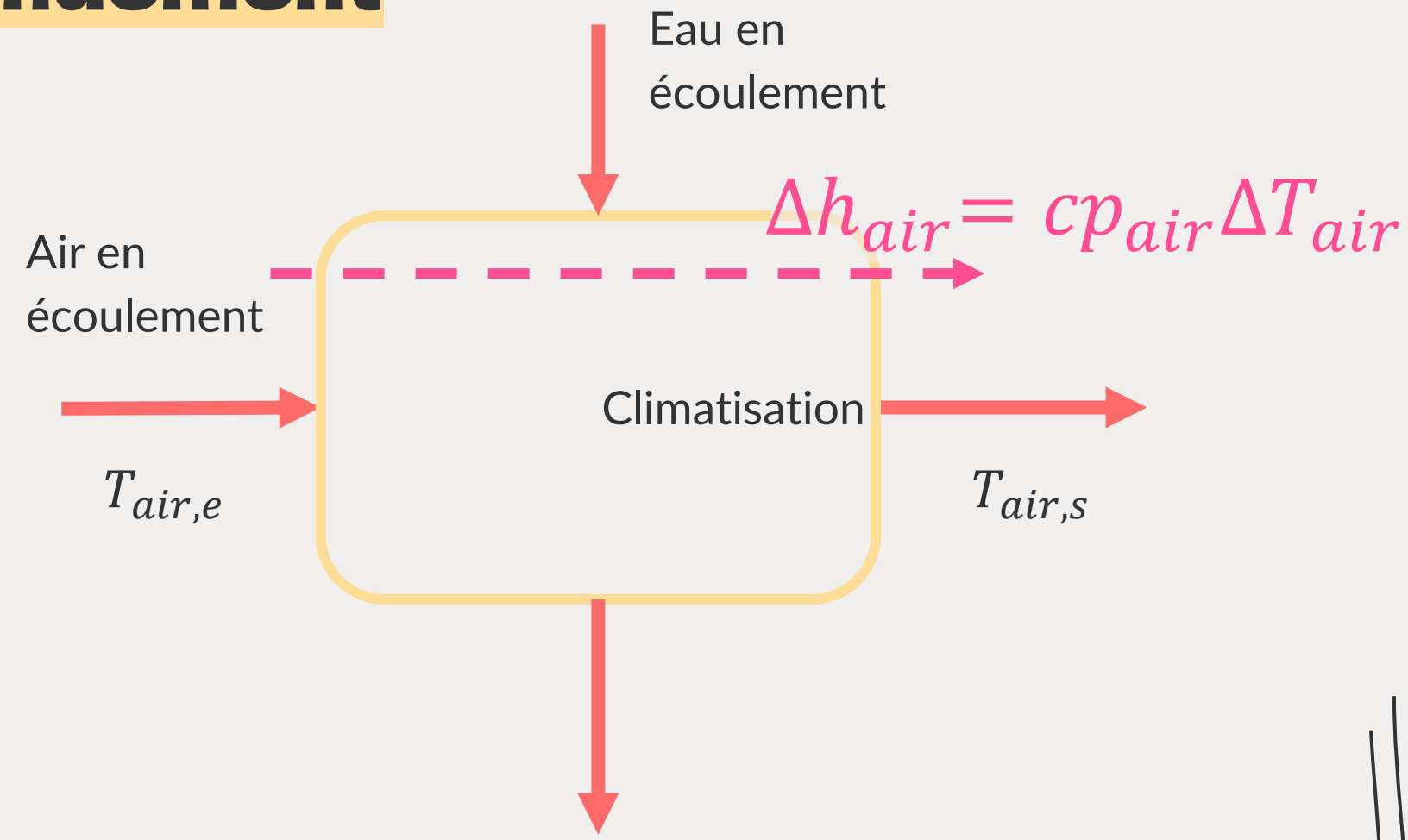
Asservissement sur les périodes



Asservissement sur les périodes



Rendement


Eau en
écoulement
$\Delta h_{air} = cp_{air} \Delta T_{air}$
Air en
écoulement
Climatisation
$T_{air,e}$
$T_{air,s}$

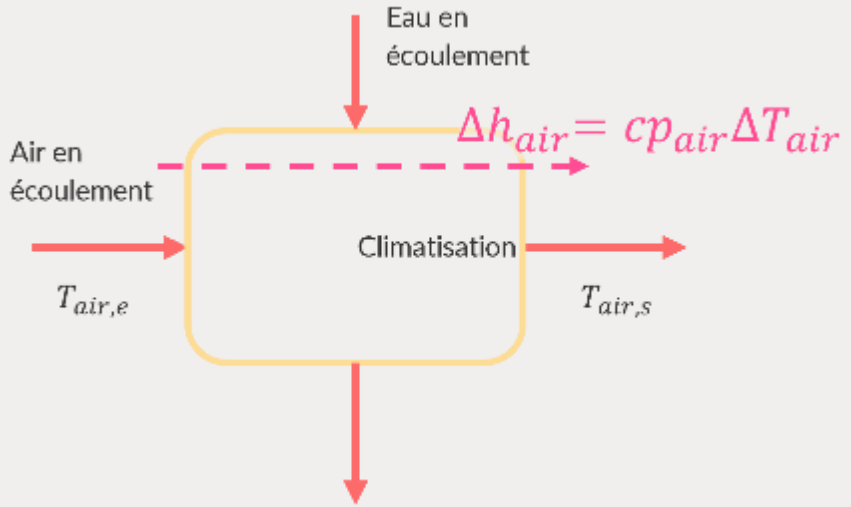
Rendement ~ efficacité ~ rapport

$$\eta = \frac{gain}{dépense}$$

Gain = Puissance thermique

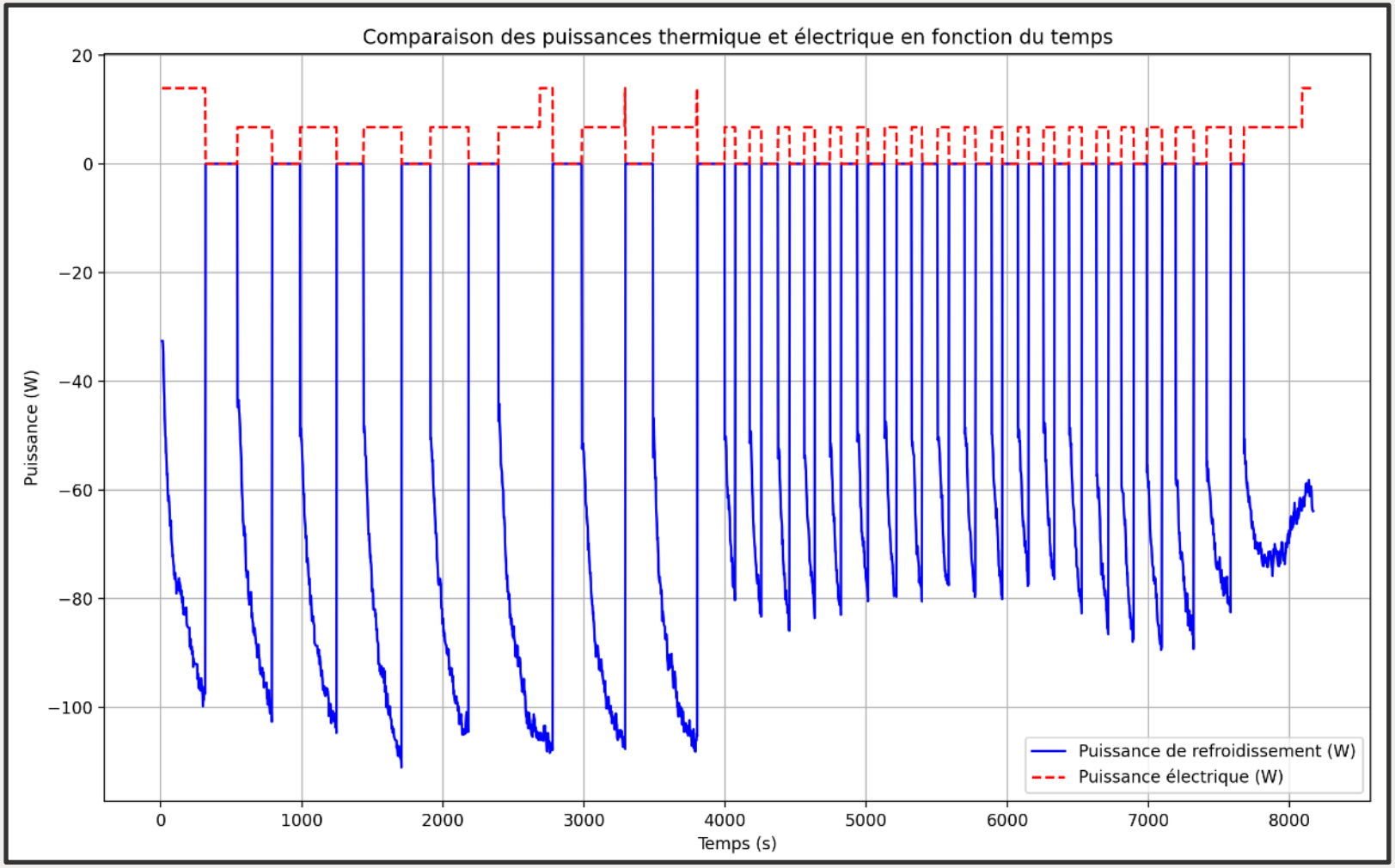
Dépense = Puissance électrique

$$P_{th} = D_{m,air} \times cp_{air} \times \Delta T$$



Puissance électrique avec les données constructeurs (sans facteur de puissance)

Rendement


Comparaison des puissances thermique et électrique en fonction du temps
Puissance (W)
Temps (s)
20
0
−20
−40
−60
−80
−100
0
1000
2000
3000
4000
5000
6000
7000
8000
Puissance de refroidissement (W)
Puissance électrique (W)

Rendement : comparaison

Comparaison des rendements et temps des fichiers

Fichier	Rendement (%)	Temps Total (s)
mesure_rendement_V2_fil.txt	**2538.36**	**2929.67**
eau_chaude_fil.txt	2055.46	3106.51
Asservissement_V4_ete_fil.txt	**2019.19**	**8231.74**
mesure_rendement_V1_fil.txt	1941.98	1442.47
Asserv_1_4_glacons_2000pts.txt	1404.95	4628.48
Fini_Asserv_V2_fil.txt	1071.60	9332.59
Fini_Asserv_V3_fil.txt	**1044.45**	**8163.44**
Mesure_all_on_first.txt	931.32	779.53

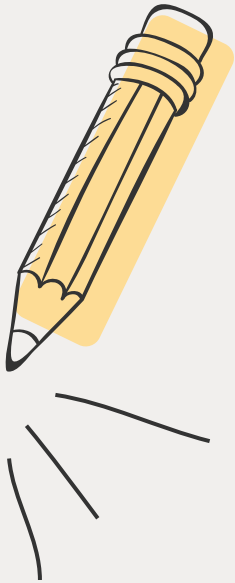
04

CONCLUSION

Climatisation adiabatique

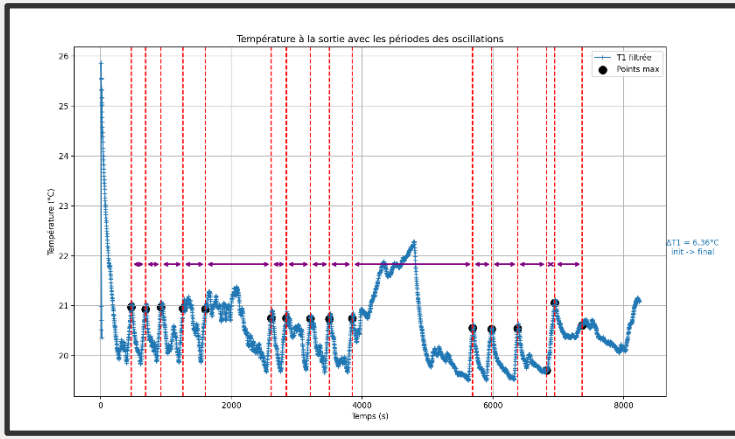
Rappel de Problématique

Passer d'un système millénaire à un système moderne : la climatisation adiabatique, mise en place, asservissement et rendement.



Humidité → Echangeur

Refroidissement → ΔT1 = 6.36°C init -> final



Climatisation locale → Refroidit correctement en face

Efficacité → ~10

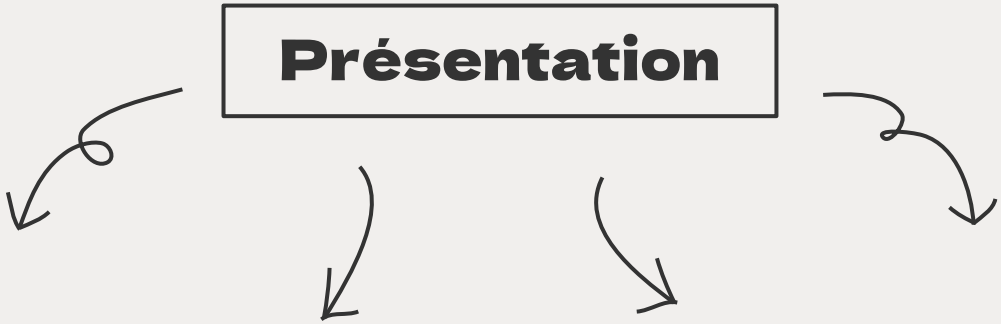
53

FIN

YVAN JACOB

Nb SCEI :42634

Retour sur la présentation



But	Construction	Mesures	Conclusion
PAGES : Présentation : 7 Principe : 8	PAGES : Problème: 12->14 Pompe : 15 Commander: 16>20	PAGES : Capteurs : 22->25 Logigramme : 26/27 Ajustement : 28->36 Asservissement: 37->45 Rendement : 46->49	PAGES : Conclusion: 52

05

ANNEXE

SOMMAIRE : ANNEXE

Code Python

Autres

Calculs rendement

Code Arduino

ANNEXE

01

Python

Discussion avec arduino

```
# Affichage des ports séries disponibles
print("Ports serial disponibles")
for COM, Descriptif, Infos in serial.tools.list_ports.comports():
    print(COM, "/", Descriptif)

# Initialisation du port série
ser = serial.Serial(port="COM3", baudrate=9600)
ser.timeout = 5
print(ser.portstr)
if ser.isOpen():
    print("Open")
time.sleep(4)
ser.flushOutput()

ser.flushOutput()  # On vide le tampon de sortie
TX = "O"
FX = "U"  # Chaîne simple
ser.flushInput()
ser.write(FX.encode())  # Écriture sur le port série
time.sleep(2)
ser.write(TX.encode())
```

Discussion avec arduino

```
for i in range(NbMesures):
    MaPetiteLigne = LireUneLigneSurCommande("T")
    data = MaPetiteLigne.split(";")
    # Lecture et ajustement des données avec les décalages
    t1.append(float(data[0]))
    T1.append(float(data[1]) + T1_dec)
    t2.append(float(data[2]))
    T2.append(float(data[3]) + T2_dec)
    t3.append(float(data[4]))
    T3.append(float(data[5]) + T3_dec)
    t4.append(float(data[6]))
    T4.append(float(data[7]) + T4_dec)
    t5.append(float(data[8]))
    T5.append(float(data[9]) + T5_dec)
    t6.append(float(data[10]))
    H6.append(float(data[11]) + H6_dec)
    t7.append(float(data[12]))
    T7.append(float(data[13]) + T7_dec)
    t8.append(float(data[14]))
    H8.append(float(data[15]) + H8_dec)
    t9.append(float(data[16]))
    v.append(float(data[17]))
    t10.append(float(data[18]))
    p.append(float(data[19]))
    T_lec.append(float(T_lec_f))

```

Filtre

```
124     # Application du filtre de Champesse sur les températures et humidités
125     # Pour les courbes de température (exemple avec T1, T2, T3, T4, etc.)
126     t1_filtered, T1_filtered = ChampesseFilter(t1, T1, f_tau)
127     t2_filtered, T2_filtered = ChampesseFilter(t2, T2, f_tau)
128     t3_filtered, T3_filtered = ChampesseFilter(t3, T3, f_tau)
129     t4_filtered, T4_filtered = ChampesseFilter(t4, T4, f_tau)
130     t5_filtered, T5_filtered = ChampesseFilter(t5, T5, f_tau)
131     t7_filtered, T7_filtered = ChampesseFilter(t7, T7, f_tau)
132
133     # Pour les courbes d'humidité (exemple avec H6, H8)
134     t6_filtered, H6_filtered = ChampesseFilter(t6, H6, f_tau)
135     t8_filtered, H8_filtered = ChampesseFilter(t8, H8, f_tau)
136
```

Filtre

```python
def ChampesseFilter(f_x, f_y, f_tau):
    f_t, f_sig = [f_x[0]], [f_y[0]]
    for i in range(len(f_y)-1):
        tmp = f_sig[-1] * f_tau + f_y[i] * (1 - f_tau)
        f_sig.append(tmp)
        f_t.append(f_x[i+1])
    return f_t, f_sig
```

Asservissement

```
    if not T_min_determiné and (t1_filtered[i]/1000) >= 300:
        T_min = min(T1_filtered)  # Calcul de T_min uniquement dans les 5 premières minutes
        #print(T1)
        print(f"T_min déterminé: {T_min}")
        T_min_determiné = True  # On bloque le recalcul pour les prochaines itérations
        T_pompe=T_min+2
        T_vent=T_min+1
        T_lec_f=T_min_determiné

    # Algorithme d'asservissement
    current_temp = T1_filtered[-1]
    #print(f"T_analysée : {T1}")
    if T_min_determiné:
        if current_temp <= T_min:
            if p[i]==1:
                ser.write("N".encode()) # Si pompe allumée, on éteint
                ser.write("F".encode()) # Mais aussi le ventilateur
                #print("Ventilateur et pompe éteints")
                etat_vent,etat_pompe = False, False
                t_delta=t1_filtered[-1]
            else:
                ser.write("F".encode())  # Eteindre que le ventilateur
                #print("Ventilateur éteint")
                etat_vent = False
                t_delta=t1_filtered[-1]
                t1_filtered[-1]

        elif (current_temp >= T_pompe or abs(t_delta-t1_filtered[-1])>=t_asserv_pompe) and not etat_pompe:
                    ser.write("O".encode())  # Allumer la pompe
                    ser.write("U".encode())  # Allumer le ventilateur
                    print("Ventilateur et pompe allumés")
                    etat_pompe = True
                    etat_vent = True
                    compte_pump+=1
                    t_pompe=t1_filtered[-1]

```

Asservissement

```
206         elif current_temp >= T_vent and not etat_vent:
207             ser.write("O".encode())  # Activer le ventilo
208             #print("Ventilateur allumé")
209             etat_vent = True
210             compte_vent+=1
211
212         elif t_pompe-t1_filtered[-1]>=delta_t_pompe or compte_pump>=3:
213             print("T_min +0.5")
214             T_min +=0.5
215             compte_pump=0
216             compte_vent=0
217             T_pompe=T_min+2
218             T_vent=T_min+1
219         elif compte_vent>=5:
220             print("T_min -0.2")
221             T_min-=0.2
222             compte_vent=0
223             T_pompe=T_min+2
224             T_vent=T_min+1
225
```

```
Ports serial disponibles
COM3 / USB-SERIAL CH340 (COM3)
COM3
Open
Minimum 0
T_min déterminé: 19.02768836257453
T_min -0.2
Ventilateur et pompe allumés
T_min -0.2
Ventilateur et pompe allumés
Ventilateur et pompe allumés
T_min +0.5
Ventilateur et pompe allumés
```

Stocker

```
97  for i in range(NbMesures):
98      MaPetiteLigne = LireUneLigneSurCommande("T")
99      data = MaPetiteLigne.split(";")
```

```
137     # Sauvegarde des données
138     MyTmpString = ";".join(data) + "\n"
139     FileSave(MyTmpString, "Enregistrement_Asservissement_V2.txt")
140
141     # Sauvegarde des données filtrées
142     MyTmpStringFiltré = f"{t1_filtered[-1]};{T1_filtered[-1]};{t2_filtered[-1]};
    {T2_filtered[-1]};{t3_filtered[-1]};{T3_filtered[-1]};{t4_filtered[-1]};
    {T4_filtered[-1]};{t5_filtered[-1]};{T5_filtered[-1]};{t6_filtered[-1]};
    {H6_filtered[-1]};{t7_filtered[-1]};{T7_filtered[-1]};{t8_filtered[-1]};
    {H8_filtered[-1]};{t9[-1]};{v[-1]};{t10[-1]};{p[-1]};{T_lec[-1]}\n"
143     FileSave(MyTmpStringFiltré, "Enregistrement_Asservissement_V2_fil.txt")
144
```

Lecture

```
MonFichierOuvert = open(Fichier, "r")
lLignesFichers = list(MonFichierOuvert)
MonFichierOuvert.close()

# Récupérer les couleurs
default_colors = plt.rcParams['axes.prop_cycle'].by_key()['color']
#print(default_colors)
# Consignes
Tconsigne = 16
Hconseille = 70

# Initialisation des listes pour stocker les données
t1, T1, t2, T2, t3, T3, t4, T4, t5, T5, t6, H6, t7, T7, t8,
H8,t9,v,t10,p,T_déterminé = [], [], [], [], [], [], [], [], [], [], [], [], [], [],
[], [], [], [], [], [], []

T_lec_f=15.15 #pour le premier essai

#for i in range(1, len(lLignesFichers)):
for i in range(1, len(lLignesFichers), 1):
    tmp = lLignesFichers[i].rstrip()
    data = tmp.split(";")
    t1.append(float(data[0]) / 1000)
    t2.append(float(data[2]) / 1000)
    t3.append(float(data[4]) / 1000)
    t4.append(float(data[6]) / 1000)
    t5.append(float(data[8]) / 1000)
    t6.append(float(data[10]) / 1000)
    t7.append(float(data[12]) / 1000)
    t8.append(float(data[14]) / 1000)
    t9.append(float(data[16]) / 1000)
    t10.append(float(data[18]) / 1000)
    T1.append(float(data[1]))
    T2.append(float(data[3]))
    T3.append(float(data[5]))
    T4.append(float(data[7]))
    T5.append(float(data[9]))
    H6.append(float(data[11]))
    T7.append(float(data[13]))
    H8.append(float(data[15]))
    v.append(float(data[17]))
    p.append(float(data[19]))
    T_déterminé.append(float(T_lec_f))
```

```python
# Normalisation
T1_dec = 0.8999999999999986
T2_dec = 0.5299999999999976
T3_dec = 0.7799999999999976-0.3
T4_dec = 0.8999999999999986
T5_dec = 1.5999999999999979-0.2
T7_dec = 1.4199999999999982-0.2

H6_dec = -5.189999999999998
H8_dec = -4.039999999999999

#Ajuster toutes les températures et humidités
T1 = [T + T1_dec for T in T1]
T2 = [T + T2_dec for T in T2]
T3 = [T + T3_dec for T in T3]
T4 = [T + T4_dec for T in T4]
T5 = [T + T5_dec for T in T5]
T7 = [T + T7_dec for T in T7]

H6 = [H + H6_dec for H in H6]
H8 = [H + H8_dec for H in H8]


deltas = {
    "ΔT1": max(T1) - min(T1),
    "ΔT2": max(T2) - min(T2),
    "ΔT3": max(T3) - min(T3),
    "ΔT4": max(T4) - min(T4),
    "ΔT5": max(T5) - min(T5),
    "ΔT7": max(T7) - min(T7),
    "ΔH6": max(H6) - min(H6),
    "ΔH8": max(H8) - min(H8),
}

```

Lecture

Lecture

```
## Première section : Tracer les températures normalisées
plt.subplot(2, 1, 1)  # 2 lignes, 1 colonne, 1ère courbe
plt.cla()
plt.plot(t1, T1, label='DS18X20_1_AP', color='#1f77b4')
plt.plot(t2, T2, label='DS18X20_2_EAU',color='#ff7f0e')
plt.plot(t3, T3, label='DS18X20_3_AV', color='#2ca02c')
plt.plot(t4, T4, label='DS18X20_4_IN', color='#d62728')
plt.legend(loc="upper right")
# plt.title('Températures normalisées en fonction du temps')
plt.title('Températures en fonction du temps')
plt.ylabel('Température (°C)')

# Tracer les humidités normalisées
plt.subplot(2, 1, 2)
plt.cla()
plt.plot(t6, H6, label='AHT_AV')
plt.plot(t8, H8, label='AHT_AP')
plt.legend(loc="center right")
plt.title('Humidités normalisées en fonction du temps')
# plt.title('Humidités en fonction du temps')
# plt.xlabel('Temps (s)')
plt.ylabel('Pourcentage d\'humidité (%)')
plt.show()
```

Lecture

```
125 ## Deuxième section : Températures sans eau
126 plt.figure(figsize=(10, 8))
127 plt.grid(True, which='both')
128 plt.minorticks_on()
129 y_min_T = min(min(T1), min(T3), min(T4))
130 y_max_T = max(max(T1), max(T3), max(T4))
131
132
133
134 y_position_T1 = (T1[-1]-y_min_T) / (y_max_T - y_min_T) + 0.05
135 y_position_T2 = (T2[-1]-y_min_T) / (y_max_T - y_min_T)
136 y_position_T3 = (T3[-1]-y_min_T) / (y_max_T - y_min_T) + 0.05
137 y_position_T4 = (T4[-1]-y_min_T) / (y_max_T - y_min_T) + 0.05
138
139
140
141 plt.plot(t1, T1, label='DS18X20_1_AP', color='#1f77b4', marker='+')
142 plt.plot(t4, T4, label='DS18X20_4_IN', color='#d62728',marker='+')
143 plt.plot(t3, T3, label='DS18X20_3_AV', color='#2ca02c',marker='+')
144 plt.text(1, y_position_T1, f"ΔT1 = {deltas['ΔT1']:.2f}°C\n  init -> final",
145          transform=plt.gca().transAxes, fontsize=10, color='#1f77b4')
146 plt.text(1, y_position_T3 , f"ΔT3 = {deltas['ΔT3']:.2f}°C\n  init -> final",
147          transform=plt.gca().transAxes, fontsize=10, color='#2ca02c')
148 plt.text(1, y_position_T4, f"ΔT4 = {deltas['ΔT4']:.2f}°C\n  init -> final",
149          transform=plt.gca().transAxes, fontsize=10, color='#d62728')
150 plt.legend(loc="upper right")
151 plt.xlabel('Temps (s)')
152 plt.title('Températures normalisées en fonction du temps')
153 # plt.title('Températures en fonction du temps')
154 plt.ylabel('Température (°C)')
155
156 plt.show()
```

```python
## Troisième section: humidité

y_min_H = min(min(H6), min(H8))
y_max_H = max(max(H6), max(H8))

y_position_H6 = (H6[-1]-y_min_H) / (y_max_H - y_min_H) + 0.05
y_position_H8 = (H8[-1]-y_min_H) / (y_max_H - y_min_H) + 0.05

plt.figure(figsize=(10, 8))
plt.grid(True, which='both')
plt.minorticks_on()
plt.plot(t6, H6, label='AHT_AV',marker='+')
plt.plot(t8, H8, label='AHT_AP',marker='+')
# plt.text(1, y_position_H6, f"ΔH6 = {deltas['ΔH6']:.2f}%\n  init -> final",
#          transform=plt.gca().transAxes, fontsize=10, color='#1f77b4')
# plt.text(1, y_position_H8, f"ΔH8 = {deltas['ΔH8']:.2f}%\n  init -> final",
#          transform=plt.gca().transAxes, fontsize=10, color='#ff7f0e')
plt.text(1, y_position_H6, f"ΔH6 = {deltas['ΔH6']:.2f}%",
         transform=plt.gca().transAxes, fontsize=10, color='#1f77b4')
plt.text(1, y_position_H8-0.15, f"ΔH8 = {deltas['ΔH8']:.2f}%",
         transform=plt.gca().transAxes, fontsize=10, color='#ff7f0e')

plt.legend(loc="center right")
# plt.title('Humidités normalisée en fonction du temps')
plt.title('Humidités en fonction du temps')
plt.xlabel('Temps (s)')
plt.ylabel('Pourcentage d\'humidité (%)')
plt.show()

# ## Intermédiaire : écart
# T5=[]
# for i in range (len(T1)):
#     T5.append(T1[i]-18.5)
# plt.figure(figsize=(10, 8))
# plt.grid(True, which='both')
# plt.minorticks_on()
# plt.plot(t6, T5, label='Différence entrée sortie(Ts-Te)',marker='+')
# plt.title('écart de température en fonction du temps')
# plt.xlabel('Temps (s)')
# plt.ylabel('Ecart ')
# plt.show()
```

Lecture

Lecture

```
03 ## Quatrième section : Température eau
04
05 y_min_T = min(min(T2), min(T4))
06 y_max_T = max(max(T2), max(T4))
07 y_position_T2 = (T2[-1]-y_min_T) / (y_max_T - y_min_T) + 0.05
08 y_position_T4 = (T4[-1]-y_min_T) / (y_max_T - y_min_T) + 0.05
09
10 plt.figure(figsize=(10, 8))
11 plt.grid(True, which='both')
12 plt.minorticks_on()
13
14 plt.plot(t2, T2, label='DS18X20_2_EAU',color='#ff7f0e',marker='+')
15 # plt.plot(t4[500:], T4[500:], label='DS18X20_4_IN', color='#d62728',marker='+')
16 plt.text(1, y_position_T2, f"ΔT2 = {deltas['ΔT2']:.2f}°C\n  init -> final",
17          transform=plt.gca().transAxes, fontsize=10, color='#ff7f0e')
18 # plt.text(1, y_position_T4, f"ΔT4 = {deltas['ΔT4']:.2f}°C\n  init -> final",
19 #          transform=plt.gca().transAxes, fontsize=10, color='#d62728')
20
21 plt.legend(loc="upper right")
22 plt.xlabel('Temps (s)')
23 plt.title('Températures normalisées en fonction du temps')
24 plt.ylabel('Température (°C)')
25
26 plt.show()
```

```
## Neuvième section: allumage des actionneurs et période
plt.figure(figsize=(10, 12))

# Premier graphique: Evolution des périodes en fonction du temps
plt.subplot(3, 1, 1)
plt.scatter(L_per[1:], delta_per, label="Evolution période", color='#1f77b4',
marker='+', s=200)
plt.legend(loc="upper right")
plt.xlabel('Temps (s)')
plt.ylabel("Période T en s")
plt.title("Évolution des périodes en fonction du temps")
plt.minorticks_on()

# Deuxième graphique: p en fonction de t10
plt.subplot(3, 1, 2)
plt.plot(t10, p, label='p en fonction du temps', color='red', marker='+')
plt.legend(loc="upper right")
plt.xlabel('Temps (s)')
plt.ylabel('p')
plt.title('Évolution de p en fonction du temps')
plt.grid()
plt.minorticks_on()

# Troisième graphique: v en fonction de t9
plt.subplot(3, 1, 3)
plt.plot(t9, v, label='v en fonction du temps', color='green', marker='+')
plt.legend(loc="upper right")
plt.xlabel('Temps (s)')
plt.ylabel('v')
plt.title('Évolution de v en fonction du temps')
plt.grid()
plt.minorticks_on()

# Ajustement de l'espacement entre les sous-graphiques
plt.subplots_adjust(hspace=0.4)

plt.show()
```

Lecture

```python
## Première section : rapport avec définiton de la puissance
# Air
debit_unite = 0.0152   # m³/s (+-10%) vitesse de vent entre 1.2 et 1.4 m/s
masse_volumique_air = 1.225  # kg/m³
cp_air = 1005  # J/kg/K

puissance_vent_elec = 5*0.45  # W
nombre_vents = 3
debit_air = debit_unite * nombre_vents # avec le datasheet
#calcul a la main, 0.4*0.11*1.2 + 0.05 * 0.01*1.4=0.0535
# ec_air = 0.1 delta_ec est négligeable

# Eau
puissance_pompe_elec = 0.3 * 12  # W
nombre_pompes = 2

puissance_refroidissement_inst = []
energie_utile = 0
A=[]
for i in range(len(T1) - 1):
    if v[i] == 1:  # vents on
        delta_T_r_air = T1[i] - T3[i]  # Différence de température entre l'entrée et la sortie
        puissance_air = debit_air * (masse_volumique_air *cp_air*delta_T_r_air)
        #print(delta_T_r_air)
        # Énergie totale extraite
        puissance_inst = puissance_air

        dt = t1[i+1] - t1[i]
        energie_utile += puissance_inst * dt
        puissance_refroidissement_inst.append(puissance_inst)
        A.append(puissance_air)
    else:
        puissance_refroidissement_inst.append(0)  # vents off
        A.append(0)

# Énergies électriques
temps_total_p = sum([(t10[i+1] - t10[i]) for i in range(len(p)-1) if p[i] == 1])
temps_total_v = sum([(t9[i+1] - t9[i]) for i in range(len(v)-1) if v[i] == 1])
```

Lecture

Lecture

```
 99  energie_vent = nombre_vents * puissance_vent_elec * temps_total_v
100  energie_pompe = puissance_pompe_elec * nombre_pompes * temps_total_p
101  energie_totale_elec = energie_vent + energie_pompe
102
103  # Rendement
104  rendement = (abs(energie_utile / energie_totale_elec)) * 100  # en %
105
106  print(f"Énergie utile : {energie_utile:.2f} J")
107  print(f"Énergie totale consommée : {energie_totale_elec:.2f} J")
108  print(f"Rendement : {rendement:.2f} %")
```

```
111 def afficher_tableau(energie_utile, energie_totale_elec, rendement, temps_total_p,
    temps_total_v, energie_vent, energie_pompe, temps_max):
112     fig, ax = plt.subplots(figsize=(12, 6))
113     ax.axis('off')
114 
115     fraction_temps_p = temps_total_p / temps_max if temps_max > 0 else 0
116     fraction_temps_v = temps_total_v / temps_max if temps_max > 0 else 0
117     fraction_energie_vent = energie_vent / energie_totale_elec if
    energie_totale_elec > 0 else 0
118     fraction_energie_pompe = energie_pompe / energie_totale_elec if
    energie_totale_elec > 0 else 0
119 
120     # Création des données pour la table
121     table_data = [
122         ["Paramètre", "Valeur", "Fraction"],
123         ["Énergie utile (J)", f"{energie_utile:.2f}", "-"],
124         ["Énergie totale consommée (J)", f"{energie_totale_elec:.2f}", "-"],
125         ["Rendement (%)", f"{rendement:.2f}", "-"],
126         ["Temps total pompe (s)", f"{temps_total_p:.2f}", f"{fraction_temps_p:.
    2%}"],
127         ["Temps total ventilateurs (s)", f"{temps_total_v:.2f}",
    f"{fraction_temps_v:.2%}"],
128         ["Énergie ventilateurs (J)", f"{energie_vent:.2f}",
    f"{fraction_energie_vent:.2%}"],
129         ["Énergie pompes (J)", f"{energie_pompe:.2f}", f"{fraction_energie_pompe:.
    2%}"]
130     ]
131 
132     table = ax.table(cellText=table_data, colLabels=None, loc='center',
    cellLoc='center', colColours=['#f2f2f2']*3)
133 
134     table.auto_set_font_size(False)
135     table.set_fontsize(16)
136     table.scale(1.2, 1.2)
137 
138     plt.title("Résultats de l'analyse thermique et énergétique", fontsize=22,
    fontweight='bold', pad=20)
139     plt.show()
```

Lecture

ANNEXE

02

Arduino

Constantes

```
#include <OneWire.h>
#include <Wire.h>
#include <Adafruit_AHTX0.h>
#include <SPI.h>

byte i;
byte present = 0;
byte data[12];
byte addr_DS1[8]; // code sur 8 bits
byte addr_DS2[8];
byte addr_DS3[8];
byte addr_DS4[8];
byte RX;

Adafruit_AHTX0 aht1;
Adafruit_AHTX0 aht2;

OneWire DS_Numero_1(2);    // Le capteur- qui répond au numéro 1, est sur la broche grove D2
OneWire DS_Numero_2(3);    //  Le capteur qui répond au numéro 2, est sur la broche grove D3
OneWire DS_Numero_3(4);    //  Le capteur qui répond au numéro 3, est sur la broche grove D4
OneWire DS_Numero_4(8);    //  Le capteur qui répond au numéro 4, est sur la broche grove D8

const int Relai1 =  5; // Pompe, bleu
const int Relai2 = 6; // Ventilo, blanc

int vent = 0;  // pour déclarer état vent
int pump = 0;  // pour déclarer état pompe
```

VOID SETUP

```
30
31  void setup(void) {
32    Serial.begin(9600);
33    delay(100);
34     //Initialisation du capteur AHT10
35    if (!aht1.begin(&Wire, 0, AHTX0_I2CADDR_DEFAULT)) {
36      Serial.println("Could not find AHT at 0x38");
37      while (1) delay(10);
38    }
39    if (!aht2.begin(&Wire, 0, AHTX0_I2CADDR_ALTERNATE)) {
40      Serial.println("Could not find AHT at 0x39");
41      while (1) delay(10);
42    }
43    //Serial.println("AHT10 trouvés at 0x38 and 0x39.");
44
45    // Initialisation des relais
46    pinMode(Relai1, OUTPUT);
47    pinMode(Relai2, OUTPUT);
48
49    // Initialisation des boutons
50
51
52        Find_DS_Numero_1();
53        Find_DS_Numero_2();
54         Find_DS_Numero_3();
55          Find_DS_Numero_4();
56
57  }
```

Récuperer valeurs

```
197  // Fonctions pour la gestion des capteurs DS18B20
198  void Find_DS_Numero_1() {
199    if (!DS_Numero_1.search(addr_DS1)) {
200      Serial.print("No more addresses.\n");
201      DS_Numero_1.reset_search();
202      delay(250);
203      return;
204    }
205    DS_Numero_1.search(addr_DS1);
206    Serial.print("DS_Numero_1=");
207    for (byte i = 0; i < 8; i++) {
208      Serial.print(addr_DS1[i], HEX);
209      Serial.print(" ");
210    }
211    Serial.println();
212    if (OneWire::crc8(addr_DS1, 7) != addr_DS1[7]) {
213      Serial.print("CRC is not valid!\n");
214    }
215  }
216
217  void DS1_DS18B20_Start() {
218    DS_Numero_1.reset();
219    DS_Numero_1.select(addr_DS1);
220    DS_Numero_1.write(0x44);
221  }
222
```

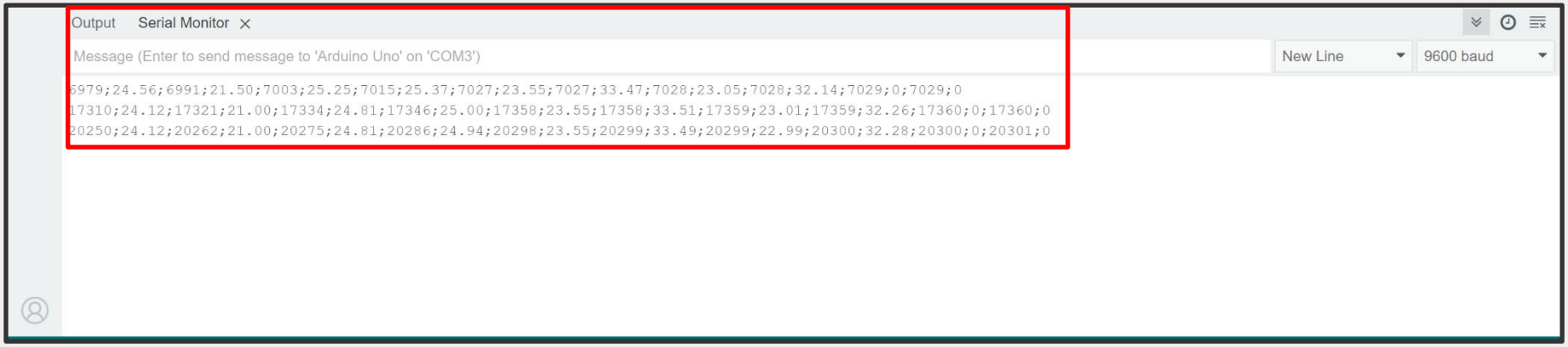
```
223  float DS1_DS18B20_Temp() {
224    DS_Numero_1.reset();
225    DS_Numero_1.select(addr_DS1);
226    DS_Numero_1.write(0xBE);
227    byte data[9];
228    for (byte i = 0; i < 9; i++) {
229      data[i] = DS_Numero_1.read();
230    }
231    int16_t raw = (data[1] << 8) | data[0];
232    byte cfg = (data[4] & 0x60);
233    if (cfg == 0x00) raw = raw & ~7;
234    else if (cfg == 0x20) raw = raw & ~3;
235    else if (cfg == 0x40) raw = raw & ~1;
236    float celsius = (float)raw / 16.0;
237    return celsius;
238  }
```

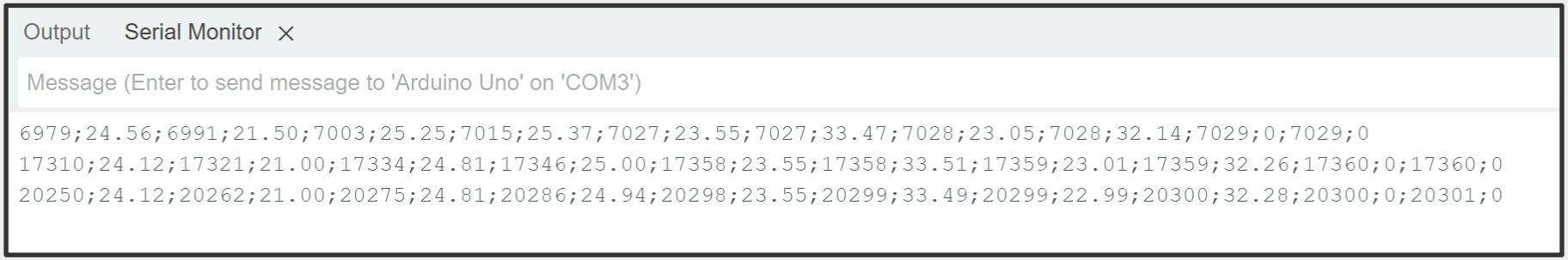
Capteurs

```
92    if (command == 'T')
93    {
94
95      DS1_DS18B20_Start();
96      DS2_DS18B20_Start();
97      DS3_DS18B20_Start();
98      DS4_DS18B20_Start();
99
100     sensors_event_t humidity1, temp1;
101     aht1.getEvent(&humidity1, &temp1);
102     sensors_event_t humidity2, temp2;
103     aht2.getEvent(&humidity2, &temp2);
104
105     delay(1000);
106     Serial.print(millis());
107     Serial.print(';');
108     Serial.print(DS1_DS18B20_Temp());
109     Serial.print(';');
110     Serial.print(millis());
111     Serial.print(';');
112     Serial.print(DS2_DS18B20_Temp());
113     Serial.print(';');
114     Serial.print(millis());
115     Serial.print(';');
116     Serial.print(DS3_DS18B20_Temp());
117     Serial.print(';');
118     Serial.print(millis());
119     Serial.print(';');
120     Serial.print(DS4_DS18B20_Temp());
121     Serial.print(';');
122     Serial.print(millis());
123     Serial.print(';');
124     Serial.print(temp1.temperature); // Premier capteur AHT10
125     Serial.print(';');
126     Serial.print(millis());
127     Serial.print(';');
128     Serial.print(humidity1.relative_humidity);
```

```
129     Serial.print(';');
130     Serial.print(millis());
131     Serial.print(';');
132     Serial.print(temp2.temperature); // Deuxième capteur AHT10
133     Serial.print(';');
134     Serial.print(millis());
135     Serial.print(';');
136     Serial.print(humidity2.relative_humidity);
137     Serial.print(';');
138     Serial.print(millis());
139     Serial.print(';');
140     Serial.print(vent); //
141     Serial.print(';');
142     Serial.print(millis());
143     Serial.print(';');
144     Serial.println(pump);
145
146   }
147 }
148 }
149
```

Capteurs : réponse





Actionneurs

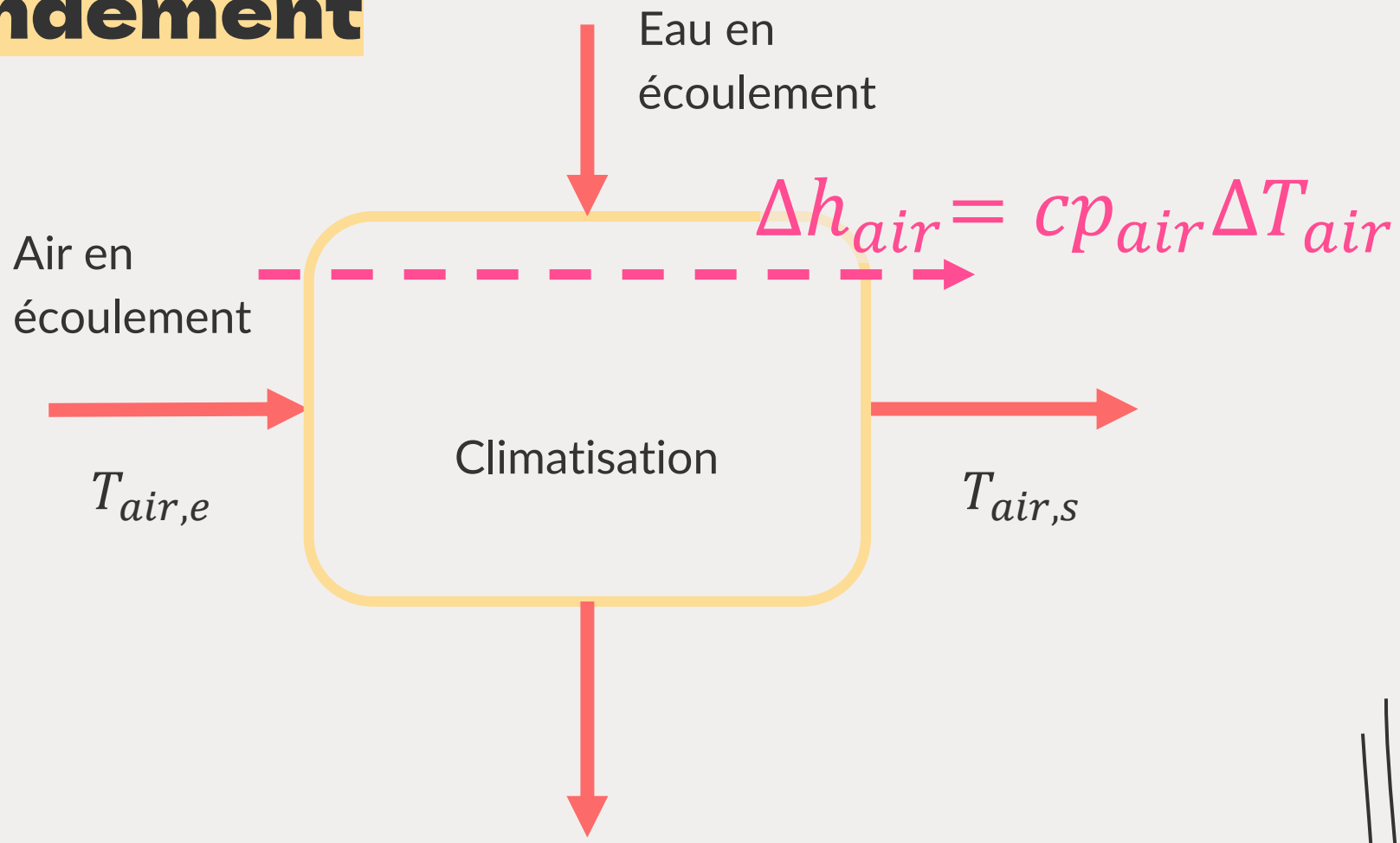
```
59  void loop() {
60    // Vérifie si des données sont disponibles dans la console série
61    if (Serial.available() > 0)
62    {
63      char command = Serial.read(); // Lit une commande envoyée via la console série
64
65      // Contrôle des relais en fonction de la commande
66      if (command == 'O') { // Ventilo
67        digitalWrite(Relai1, HIGH);  // Allume le relais 1
68        vent = 1;
69
70  //      Serial.println("Relais 1 allumé");
71      }
72      if (command == 'F') {
73        digitalWrite(Relai1, LOW);   // Éteint le relais 1
74        vent = 0;
75        //Serial.println("Relais 1 éteint");
76      }
77      if (command == 'U') { // Pompe
78        digitalWrite(Relai2, HIGH);  // Allume le relais 2
79        pump = 1;
80  //      Serial.println("Relais 2 allumé");
81      }
82      if (command == 'N') {
83        digitalWrite(Relai2, LOW);   // Éteint le relais 2
84        pump = 0;
85        //Serial.println("Relais 2 éteint");
86      }
87
```

ANNEXE

03

Rendements

Rendement


Eau en écoulement
$\Delta h_{air} = cp_{air} \Delta T_{air}$
Air en écoulement
Climatisation
$T_{air,e}$
$T_{air,s}$

Rendement

Puissance maximale: conditions 55% d'humidité et 18°C

- on fait couler 518,4g

-ventilateur à fond et on regarde l'eau récupérée dans le réservoir

-le reste a été évaporé (ici 459,9g dans le réservoir)

-m_{evap} = 102g et ce en 80min

$L_{vap}(H_2O)$=2454 kJ/kg

Rendement

Puissance crée par le débit d'air:

conditions 55% d'humidité et 18°C

- Débit d'air : 0,05 m^3/s (1.2 x 0.4 * 0.12)
- $\mu = 1{,}208$ kg/ m^3
- Dm = 1,208 x 0,05
- On récupère $\Delta T = 2{,}8°C$
- $P_{th,air} = Dm(cp \times \Delta T) = 1{,}208 \times 0{,}05 \times 1009 \times 2{,}8 = 181$ W

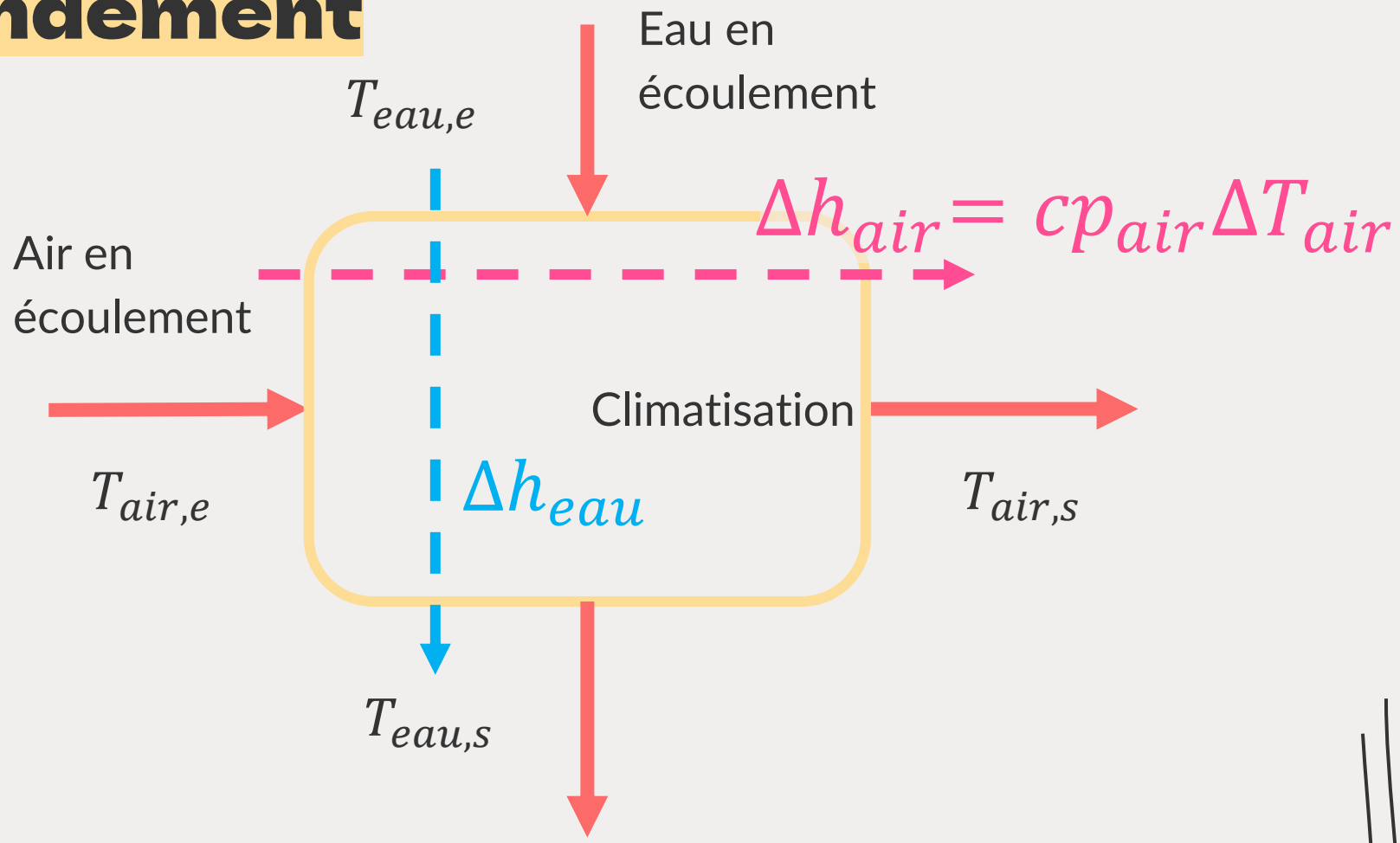
Rendement

- Pertes par réchauffement terrestre
- Pertes dans la détente
- Pertes dans le refroidissement de l’eau

Rendement

Modélisation infinitésimale

Rendement


Eau en écoulement
$T_{eau,e}$
$\Delta h_{air} = cp_{air} \Delta T_{air}$
Air en écoulement
Climatisation
$T_{air,e}$
Δh_{eau}
$T_{air,s}$
$T_{eau,s}$

Rendement

$$\alpha = \frac{\delta m_b}{\delta m_h}$$

Haut = entrée de l'eau
Bas = sortie de l'eau

Pour l'eau, $dH_e = -(\delta m_h \times cp_{eau} \times \theta_h - \delta m_b \times cp_{eau} \times \theta_b)$
Pendant dt : $\Longrightarrow dH_e = cp_{eau} \times \delta m_h \times (\alpha \times \theta_b - \theta_h)$

La différence du débit d'air entre l'entrée et la sortie est la masse d'eau évaporée, (au max = 0.01 g/s)

Rendement

Pour l'air, $dH_{ah} = \delta m_{air,sec} \times cp_{air,sec} \times (\theta_2 - \theta_1)$
$+ \delta m_{eau,haut} \times cp_{eau} \times (\theta_h - \theta_b)$
$+ \delta m_{eau,haut} \times (1-\alpha) \times cp_{eau} \times \theta_b$
$+ \delta m_{eau,haut} \times (1-\alpha) \times lvap_{eau \text{ à } \theta_h}$

Car entrée : $\delta H_{ah,1} = \delta m_{air,sec} \times cp_{air,sec} \times \theta_1 + \delta m_{eau,bas} \times cp_{eau} \times \theta_1 + \delta m_{eau,bas} \times lvap_{eau \text{ à } \theta_1}$

$\delta H_{ah,2} = \delta m_{air,sec} \times cp_{air,sec} \times \theta_2 + \delta m_{eau,haut} \times cp_{eau,g} \times \theta_2 + \delta m_{eau,haut} \times lvap_{eau \text{ à } \theta_1}$

Normalement, $dH_{ah} = dH_e$

Rendement: incertitudes

Tableau des Incertitudes — fichier : Fini_Asserv_V3.txt

Capteur	Moyenne	Écart-type	Incertitude	Incertitude composée
T1	17.91	0.30	±0.01	±0.10
T2	15.93	0.33	±0.01	±0.10
T3	19.24	0.08	±0.00	±0.10
T4	17.16	0.68	±0.01	±0.10
H6	62.13	0.64	±0.01	±2.00
H8	77.23	6.12	±0.10	±2.00
T1 (filtré)	17.91	0.30	**±0.01**	**±0.10**
T2 (filtré)	15.93	0.33	**±0.01**	**±0.10**
T3 (filtré)	19.24	0.08	**±0.00**	**±0.10**
T4 (filtré)	17.16	0.68	**±0.01**	**±0.10**
H6 (filtré)	62.12	0.64	**±0.01**	**±2.00**
H8 (filtré)	77.22	6.12	**±0.10**	**±2.00**

Rendement: incertitudes

Tableau des Incertitudes — fichier : Asservissement_V4_ete_fil.txt

Capteur	Moyenne	Écart-type	Incertitude	Incertitude composée
T1	20.50	0.73	±0.01	±0.10
T2	16.83	0.91	±0.02	±0.10
T3	23.66	0.22	±0.00	±0.10
T4	20.15	1.98	±0.03	±0.11
H6	35.25	4.42	±0.07	±2.00
H8	57.02	12.00	±0.20	±2.01
T1 (filtré)	20.50	0.73	**±0.01**	**±0.10**
T2 (filtré)	16.83	0.91	**±0.02**	**±0.10**
T3 (filtré)	23.66	0.23	**±0.00**	**±0.10**
T4 (filtré)	20.15	1.98	**±0.03**	**±0.11**
H6 (filtré)	35.25	4.41	**±0.07**	**±2.00**
H8 (filtré)	57.01	12.00	**±0.20**	**±2.01**

Rendement: incertitudes

```python
def calcul_incertitude(mesures, type_capteur):
    n = len(mesures)
    moyenne = np.mean(mesures)
    ecart_type = np.std(mesures, ddof=1)
    incertitude_type = ecart_type / np.sqrt(n)
    incert_capteur = 0.1 if type_capteur == "T" else 2.0
    incert_composee = np.sqrt(incertitude_type**2 + incert_capteur**2)
    return moyenne, ecart_type, incertitude_type, incert_composee

def ChampesseFilter(f_x, f_y, f_tau):
    f_t, f_sig = [f_x[0]], [f_y[0]]
    for i in range(len(f_y) - 1):
        tmp = f_sig[-1] * f_tau + f_y[i] * (1 - f_tau)
        f_sig.append(tmp)
        f_t.append(f_x[i + 1])
    return f_t, f_sig

```

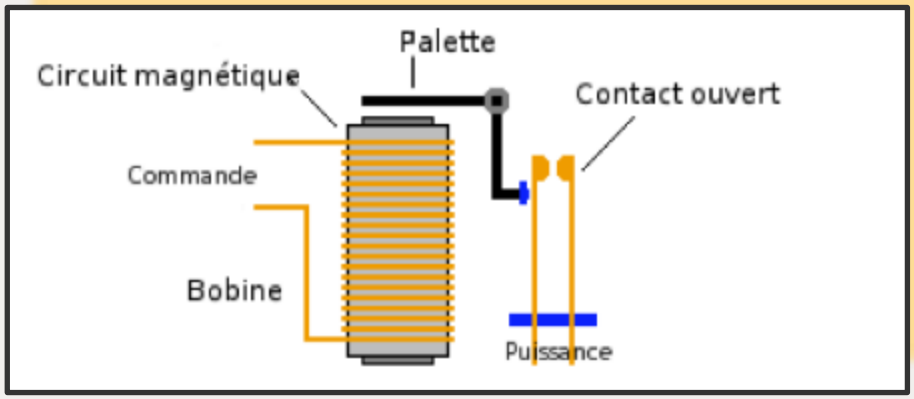
ANNEXE

04

Autres

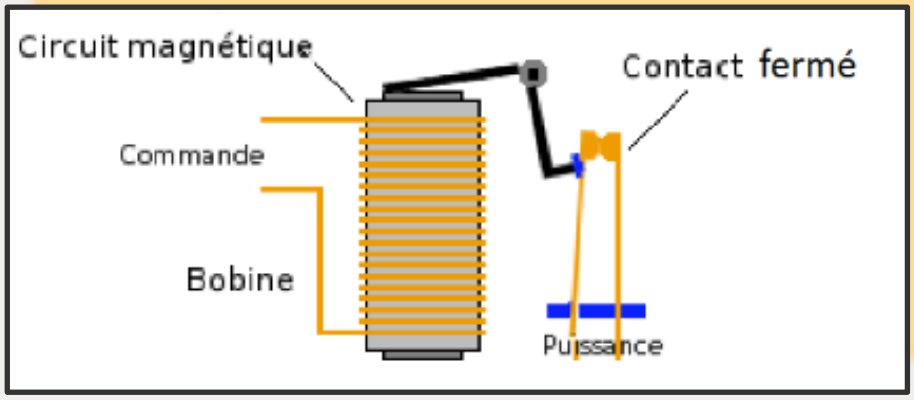
Relai

- Arduino permet le contact



Intensité

- Lorsqu'il y a contact, pompes et/ou ventilateurs sont allumés



Oxycom

- Datasheet IntrCool Plus

AIRFLOW WITH FILTERS		
Nominal airflow	14000 m³/h (3.889 m³/s)	8240 CFM (3889 L/s)
Nominal ESP	80 Pa	--
Maximum ESP - ISO ePM1 70% (F7)	160 Pa	--
Maximum ESP - ISO ePM10 >50% (M5)	155 Pa	--
Maximum ESP - ISO Coarse >60% (G4)	120 Pa	--
Face velocity at nominal airflow	1.8 m/s	

FINNED HEAT EXCHANGER	
Amount	4
Anti-corrosive coil coating	Yes

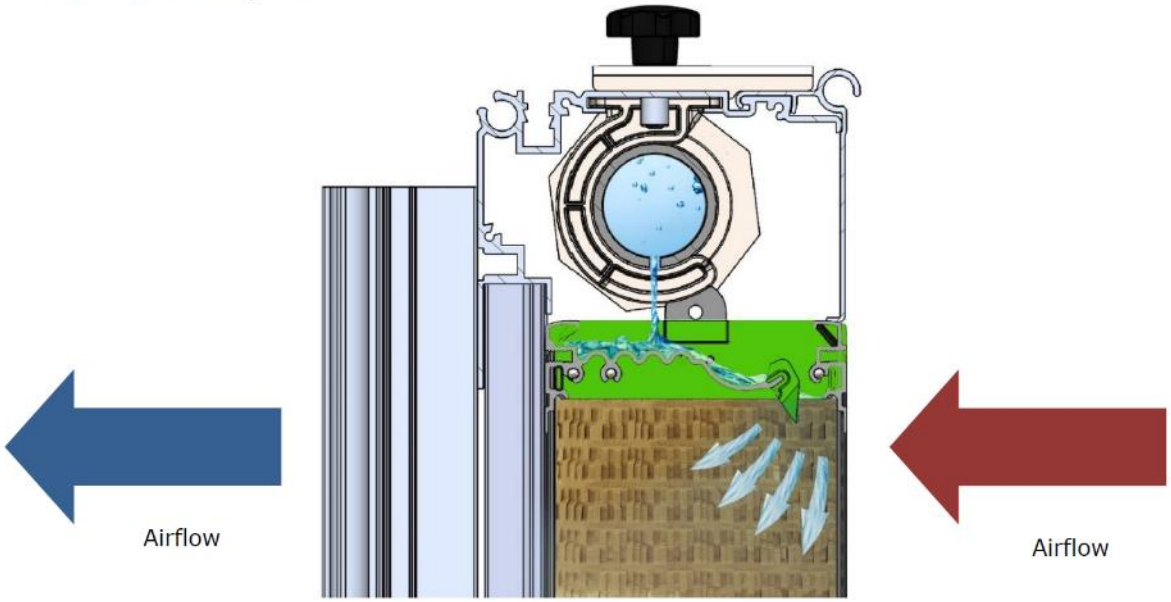
Oxycom

- Système de distribution d'eau

The Oxyvap® water distribution system is designed to operate with a water flow rate of 471 liters/hour/meter width. When exceeding the nominal flow, droplet lift-off may occur at higher velocities. Insufficient water flow may result in a decrease in efficiency and excessive mineral and/or dust build-up over time. With insufficient water flow the evaporative media's operation will be compromised and life-time will be shortened.

Distribution over the width of the Oxyvap® pads is realized by jetting water through small holes along a tube (Ø40 mm or Ø25 mm). The hole pitch is 50 mm, and the hole diameter is 3.5 mm. It is advised to use a threaded end cap at the other end of the tube, to assure easy cleaning.

Water distribution system



To ensure even wetting of the pad, both the Oxyvap® pad and the water distribution tube shall be placed level.

Oxycom

- Températures atteignable grâce à leur système en fonction de la température extérieure et de l'humidité extérieure

Outdoor air temperature	Supply temperature 1 m/s	1.5 m/s	2 m/s	Supply temperature 1 m/s	1.5 m/s	2 m/s	Supply temperature 1 m/s	1.5 m/s	2 m/s	Supply temperature 1 m/s	1.5 m/s	2 m/s	Supply temperature 1 m/s	1.5 m/s	2 m/s	Supply temperature 1 m/s	1.5 m/s	2 m/s	Supply temperature 1 m/s	1.5 m/s	2 m/s
50 °C	24.2	25.5	26.4	28.7	29.7	30.4	32.7	33.6	34.2												
48 °C	23.2	24.5	25.3	27.4	28.4	29.1	31.2	32.0	32.6												
46 °C	22.2	23.4	24.2	26.1	27.1	27.8	29.6	30.4	31.0												
44 °C	21.3	22.4	23.2	24.9	25.8	26.5	28.1	28.9	29.5	31.2	31.8	32.3									
42 °C	20.3	21.3	22.1	23.6	24.6	25.2	26.7	27.5	28.0	29.6	30.2	30.6									
40 °C	19.2	20.3	21.0	22.4	23.3	23.9	25.3	26.0	26.5	28.0	28.6	29.0	30.5	30.9	31.2						
38 °C	18.2	19.2	19.9	21.2	22.0	22.6	23.9	24.6	25.1	26.4	27.0	27.4	28.7	29.2	29.5						
36 °C	17.2	18.1	18.7	20.0	20.8	21.3	22.5	23.2	23.6	24.9	25.4	25.8	27.1	27.5	27.8	29.1	29.5	29.7			
34 °C	16.1	17.0	17.6	18.7	19.5	20.0	21.1	21.8	22.2	23.4	23.9	24.2	25.4	25.9	26.1	27.4	27.7	27.9	29.2	29.4	29.6
32 °C	15.0	15.8	16.4	17.5	18.2	18.7	19.8	20.4	20.8	21.8	22.3	22.7	23.8	24.2	24.5	25.6	26.0	26.2	27.4	27.6	27.8
30 °C	13.8	14.6	15.2	16.2	16.9	17.4	18.4	18.9	19.3	20.3	20.8	21.1	22.2	22.6	22.8	23.9	24.2	24.4	25.6	25.8	25.9
28 °C	12.7	13.4	14.0	14.9	15.6	16.0	17.0	17.5	17.9	18.8	19.3	19.6	20.6	20.9	21.2	22.2	22.5	22.7	23.8	24.0	24.1
26 °C	11.5	12.2	12.7	13.6	14.2	14.7	15.5	16.1	16.4	17.3	17.7	18.0	18.9	19.3	19.5	20.5	20.8	20.9	22.0	22.2	22.3
24 °C		10.9	11.4	12.3	12.9	13.3	14.1	14.6	14.9	15.8	16.2	16.4	17.3	17.6	17.9	18.8	19.0	19.2	20.2	20.4	20.5
22 °C				10.9	11.4	11.8	12.6	13.1	13.4	14.2	14.6	14.8	15.7	16.0	16.2	17.1	17.3	17.5	18.4	18.5	18.7
20 °C							11.1	11.6	11.9	12.6	13.0	13.2	14.0	14.3	14.5	15.3	15.6	15.7	16.6	16.7	16.9
	10%			20%			30%			40%			50%			60%			70%		
	Outdoor air relative humidity																				

Outdoor air temperature	Supply temperature 1 m/s	1.5 m/s	2 m/s
50 °C			
48 °C			
46 °C			
44 °C			
42 °C			
40 °C			
38 °C			
36 °C			
34 °C			
32 °C	29.0	29.2	29.3
30 °C	27.1	27.3	27.4
28 °C	25.2	25.4	25.5
26 °C	23.4	23.5	23.6
24 °C	21.5	21.6	21.7
22 °C	19.6	19.7	19.8
20 °C	17.8	17.9	17.9
	80%		
	Outdoor air relative humidity		

Oxycom

- Calcul pour déterminer le moment où il faut réinjecter de l'eau

Counting the amount of refills

When water level within the sump is maintained by two level sensors, counting the amount of refills in combination with the conductivity of the fresh water supply, the conductivity level in the sump can be calculated. The controller will drain the sump after a certain predetermined amount of refills.

Required data:

- Electrical conductivity of fresh water supply
- Tank volume (high level)
- Tank volume (low level)

$$\text{Maximum Amount of Refills} = \frac{\frac{\text{Maximum Conductivity } \left[\frac{\mu S}{cm}\right]}{\text{Fresh Water Conductivity } \left[\frac{\mu S}{cm}\right]} - 1}{1 - \frac{\text{Tank Volume Low Level } [L]}{\text{Tank Volume High Level } [L]}}$$

Capteurs

DS18B20

Capteur semi-conducteur à jonction PN

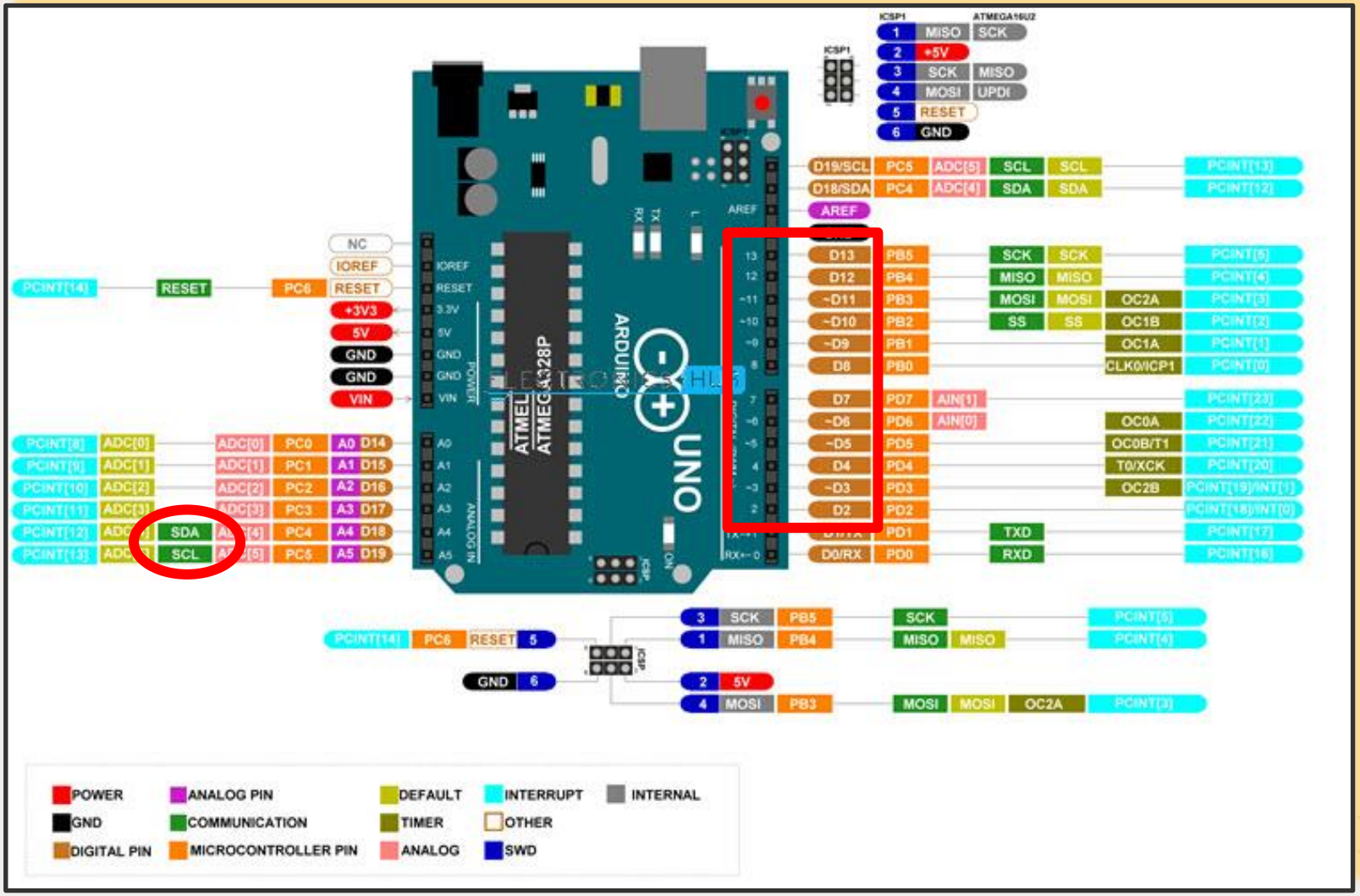
- une jonction PN produit une tension dépendante de la température
- Conversion grâce à un CAN

AHT10

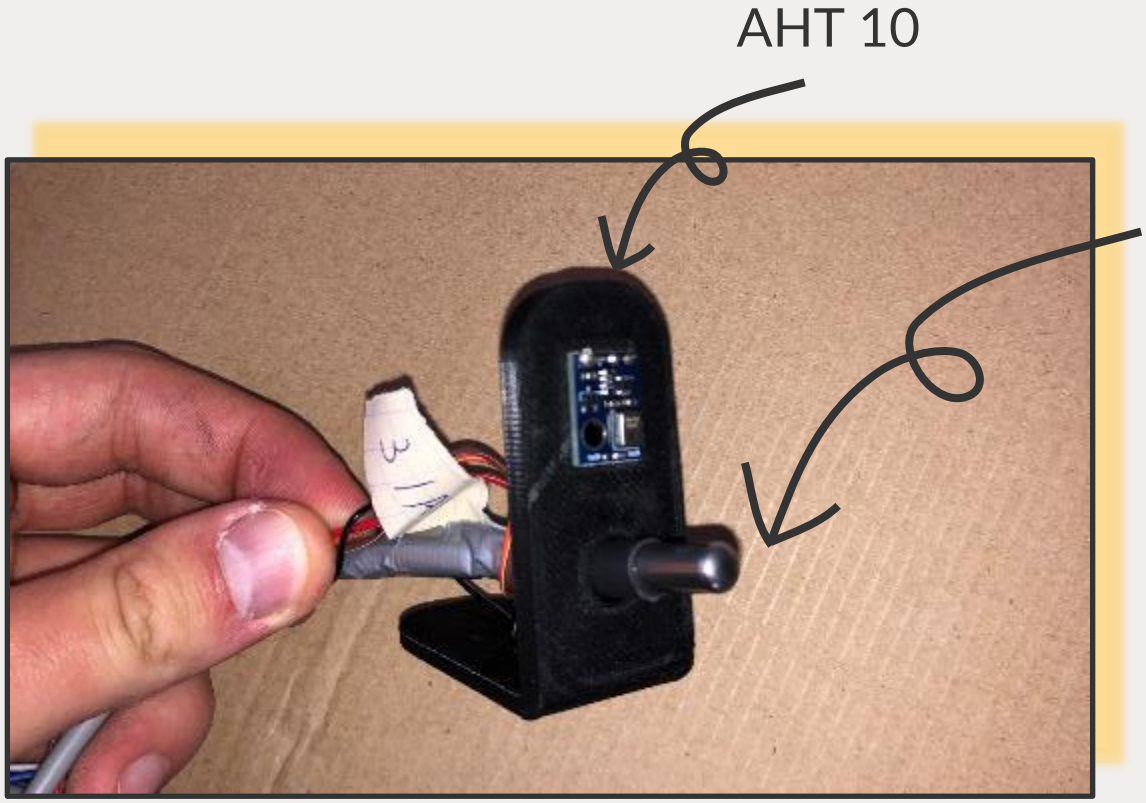
Capteur résistif

- Thermomètres à résistance basés sur la variation de la résistance d'un matériau métallique en fonction de la température.

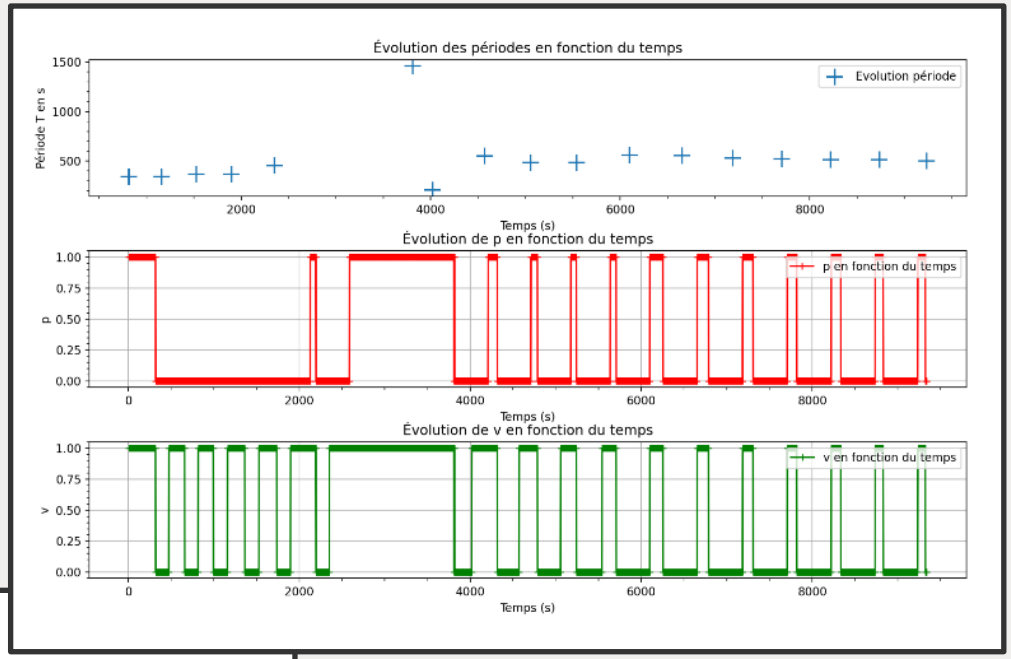
Capteurs: branchements

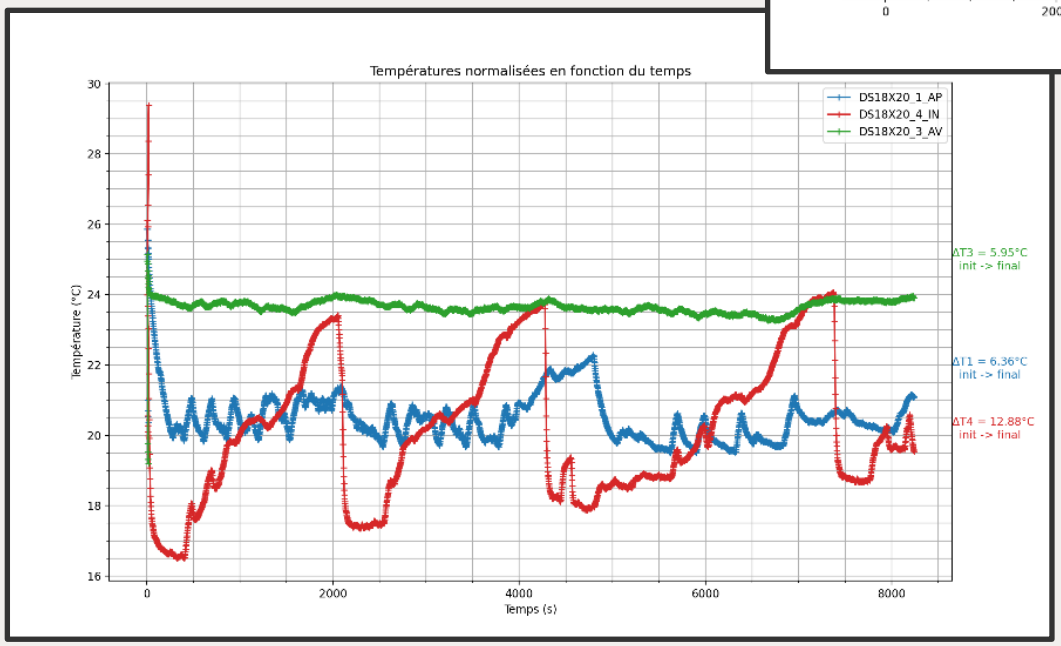


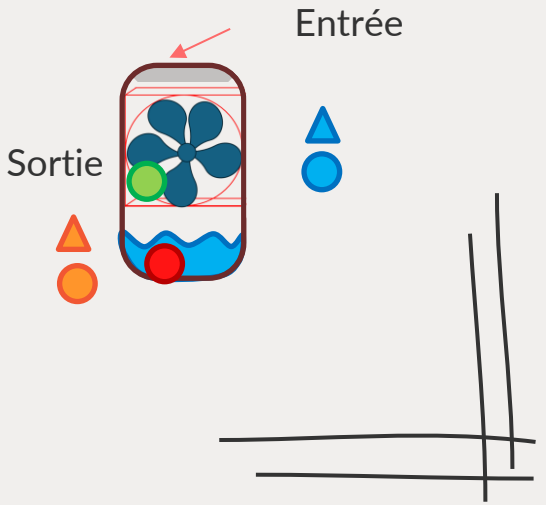
Capteurs: fixation



Test été.


Évolution des périodes en fonction du temps
Période T en s
Evolution période
Temps (s)
Évolution de p en fonction du temps
p
p en fonction du temps
Temps (s)
Évolution de v en fonction du temps
v
v en fonction du temps
Temps (s)


Températures normalisées en fonction du temps
DS18X20_1_AP
DS18X20_4_IN
DS18X20_3_AV
Température (°C)
ΔT3 = 5.95°C
init -> final
ΔT1 = 6.36°C
init -> final
ΔT4 = 12.88°C
init -> final
Temps (s)


Entrée
Sortie


FIN