

"""main.py | Robin Forestier | 2021 / 2022

Goal : Create a user interface to control the light of the ELO's workshop.

Explanation : I use Flask for create a Web server.

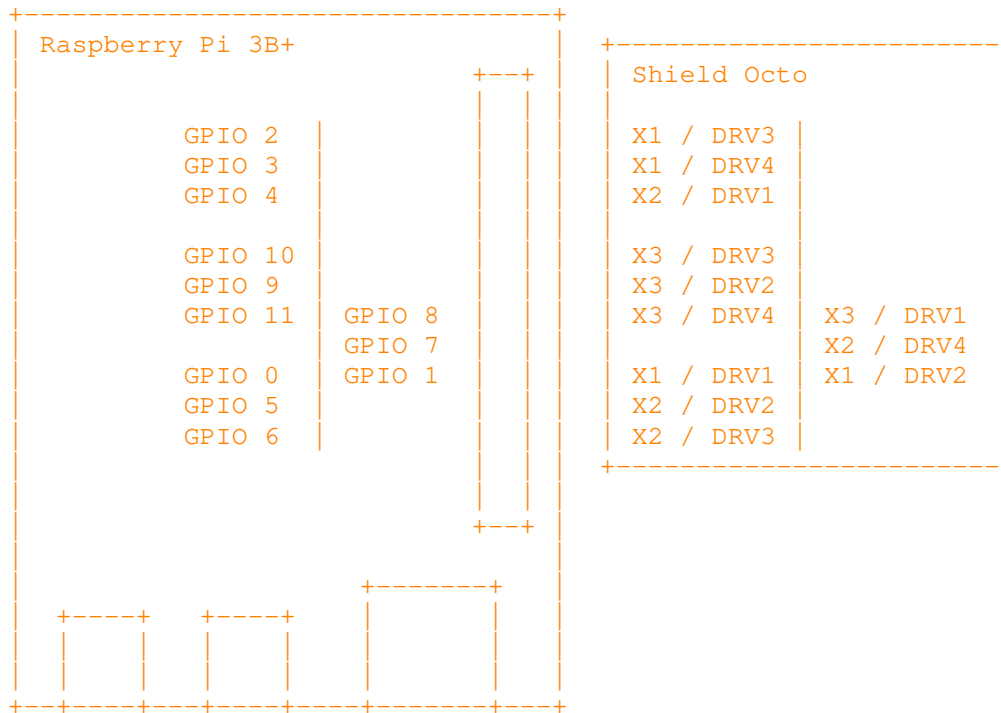
A shield is plug on the Raspberry Pi (server) with opto-isolator to ABB module with the GPIO.

TPI : For my IPT, I have to create a settings page controlling the time range where the cameras have the right

to request a modification of the lights.

I have to create a manual mode, to modify the date and time. I also need to see the status of the cameras and be able to change the passwords.

Raspberry Pi pinout usage :



GitLab : <http://172.16.32.230/Forestier/contrôle-des-lumières-knx>

http://172.16.32.230/Forestier/tpi_forestier_gestion_lumiere_knx

XWiki :

<https://xwiki.serverelo.org/xwiki/bin/view/Centre%20de%20Formation%20ELO/Projets/Centrale%20des%20lumières%20KNX/>

<https://xwiki.serverelo.org/xwiki/bin/view/Centre%20de%20Formation%20ELO/Travaux-diplome/>

[tpi_forestier_gestion_lumiere_knx/Documentation/](https://xwiki.serverelo.org/xwiki/bin/view/Centre%20de%20Formation%20ELO/Travaux-diplome/tpi_forestier_gestion_lumiere_knx/Documentation/)

"""

Importing the necessary modules to run the code.

import logging

import os

import pickle

import subprocess

import threading

import time

from datetime import datetime, timedelta

import numpy as np

Import flask

from flask import Flask, render_template, request, session, url_for, redirect, flash, abort

from flask_session import Session

from flask_sqlalchemy import SQLAlchemy

from werkzeug.exceptions import HTTPException

Creating a class called Table.

from table.table import Table

```

65 # Creating a class called Table2 that inherits from Table.
66 from table.table import Table2
67
68 # The above code is importing the RPi.GPIO module and setting it to use the BCM pin
  numbering scheme.
69 try:
70     import RPi.GPIO as GPIO
71
72     GPIO.setmode(GPIO.BCM)
73     GPIO.setwarnings(False)
74
75     # GPIO 0 to 11 in output mode at 0
76     for i in range(12):
77         GPIO.setup(i, GPIO.OUT, initial=GPIO.LOW)
78
79     # it's a raspberry pi !
80     rpi = True
81 except ImportError:
82     # it's not a raspberry pi !
83     rpi = False
84
85 # The above code is creating a log file called log.log and setting the format of the
  log file to the time, the level of
86 # the log message and the message itself.
87 logging.basicConfig(filename="log.log", format='% (asctime)s - %(levelname)s -
  %(message)s', level=logging.INFO)
88
89 # This is creating a new Flask object.
90 app = Flask(__name__)
91
92 # This is the configuration for the application.
93 app.config.update(
94     SECRET_KEY='somesecretkeythatonlyishouldknow',
95     SQLALCHEMY_DATABASE_URI="sqlite:///db.sqlite3",
96     SQLALCHEMY_TRACK_MODIFICATIONS=False,
97     SESSION_TYPE='sqlalchemy',
98     SESSION_COOKIE_NAME='MyBeautifulCookies',
99     SESSION_COOKIE_SAMESITE='Strict',
100 )
101
102 # This is creating a database object that will be used to access the database.
103 db = SQLAlchemy(app)
104 # This is telling Flask to use SQLAlchemy as the database.
105 app.config['SESSION_SQLALCHEMY'] = db
106 # This is creating a session object that will be used to store and retrieve session
  data from the user's browser.
107 sess = Session(app)
108 db.create_all()
109
110 # Ip of the server (for the local user) use to skip authentication.
111 # Checking if the IP address of the machine that is running the script is in the
  list of IP addresses that are
112 # automatically login.
113 authorize_ip = ["localhost", "127.0.0.1", "172.16.32.133"]
114
115 # Value of the buttons and the colors for the map light, each element is a string
  "off".
116 buttonSts_p1 = ["off"] * 8
117 buttonSts_p2 = ["off"] * 8
118 color = ["#333333"] * 8
119 # possible warning (like temp alert).
120 warning = ""
121
122
123 class User:
124     """User : create different user for the login"""
125
126     def __init__(self, id, username, password):
127         """
128         The __init__ function is a constructor method that is called when an object
  is created
129

```

```

130         :param id: The id of the user
131         :type id: int
132         :param username: The username of the user
133         :type username: str
134         :param password: The password of the user
135         :type password: str
136         """
137
138         self.id = id
139         self.username = username
140         self.password = password
141
142
143     # For the page: login 2
144     mot_de_pass = []
145
146     mode_manuel = 0
147     """mode_manuel (int) is set if a user activate it on the settings page. (0 ->
disable, 1 -> enable)"""
148
149     mode_heure_manuel = [0, 0, 0]
150     """mode_heure_manuel (list) is use to store:
151     1. (boolean) set if mode heure manuel activated.
152     2. (int) current day. 0 to 6 -> 0 is monday.
153     3. () delta time.
154     """
155
156     cam_connect = [[False, "", "b8:27:eb:26:5a:95", 0], [False, "", "b8:27:eb:ce:4f:78",
0]]
157     """cam_connect (list) of the authorized camera.
158     To check the authorisation, I used the mac adresse.
159     1. (str) True if connected.
160     2. (str) Ip adresse.
161     3. (str) Mac adresse.
162     """
163
164     # This is a timedelta object. It is a duration expressing the difference between two
dates.
165     TURNOFFTIME = timedelta(minutes=1)
166     """TURNOFFTIME (timedelta) is the time duration before the light turn off."""
167
168     cam_send_turn_off = []
169
170     number_of_people = 0
171     """number_of_people (int) is the number of people in the room."""
172
173     # Try to open password.pickle. this file contains the 3 passwords.
174     try:
175         # This code is opening a pickle file and reading it.
176         with open('password.pickle', 'rb') as f:
177             # This code is loading the pickle file.
178             password = pickle.load(f)
179     except OSError:
180         # This code is creating a pickle file that contains a list of passwords.
181         with open('password.pickle', 'wb') as f:
182             password = ["elo", "admin", "1234"]
183             pickle.dump(password, f, pickle.HIGHEST_PROTOCOL)
184
185     users = []
186     """
187     user (list) is a list of User objects.
188     elo - basic user / can only modifie light
189     admin - all access (settings access)
190     local - only used for the raspberry why the touchscreen (server) / can access to
settings with login page 2
191             / don't have to login to control the light.
192     """
193     # Create the users.
194     users.append(User(id=1, username='elo', password=password[0]))
195     users.append(User(id=2, username='admin', password=password[1]))
196     users.append(User(id=3, username='local', password=password[2]))
197

```

```

198
199 def gpio_modif():
200     """gpio_modif : modif all gpio"""
201     if rpi:
202         for i in range(8):
203             if buttonSts_pl[i] == "off":
204                 # OFF
205                 GPIO.output(i, 0)
206             else:
207                 # ON
208                 GPIO.output(i, 1)
209                 # time.sleep(1)          #security time for fuses
210     else:
211         pass
212
213
214 def get_time():
215     """Get the current time and day
216
217     :return: A tuple containing the current time and the current day of the week.
218     :rtype: tuple
219     """
220
221     # The above code is creating a global variable called mode_heure_manuel.
222     global mode_heure_manuel
223
224     # This code is checking if the first element of the list mode_heure_manuel is
225     # equal to 1.
226     # If the first element is 1 the mode hour manuel is set.
227     # The current hour become current hour + delta time.
228     if mode_heure_manuel[0] == 1:
229         t = datetime.now() + mode_heure_manuel[2]
230         current_time = t.strftime("%H:%M")
231         current_day = mode_heure_manuel[1]
232     else:
233         # Creating a time object.
234         t = time.localtime()
235         # The above code is creating a string of the current time in hours and
236         # minutes (10:10).
237         current_time = time.strftime("%H:%M", t)
238         # This code is checking the current day of the week and then using that to
239         # determine
240         # which day of the week it is (0 is monday, 6 is sunday).
241         current_day = datetime.today().weekday()
242
243     return current_time, current_day
244
245
246 def myping(host):
247     """myping : check if a host is up
248
249     :param host: The host to check.
250     :type host: str
251     :return: True if the host is up, False if not.
252     :rtype: bool
253     """
254
255     # Checking if the host is empty. If it is empty, it will return False.
256     if host == "":
257         return False
258
259     if rpi:
260         # This code is using the subprocess module to run the ping command on a host.
261         response = str(subprocess.check_output(["ping", "-c", "1", host]))
262     else:
263         # This code is using the os.popen() function to ping the host.
264         response = os.popen("ping -n 1 " + host).read()
265
266     # The above code is checking to see if the ping was successful.
267     if "0 received" in response:
268         return False
269     else:

```

```

267         return True
268
269
270 def getmacadd(host):
271     """getmacadd : get the mac adresse of a host
272
273     :param host: The host to check.
274     :type host: str
275     :return: The mac adresse of the host.
276     :rtype: str
277     """
278
279     # This code is using the ping function to ping the host.
280     rep = myping(host)
281     if rep:
282         # The above code is using the subprocess module to run the arp command on
283         # the host machine. The arp command is
284         # used to display the ARP (Address Resolution Protocol) table. The arp table
285         # contains the hardware (MAC) address
286         # for each IP address on the local area network.
287         answer = str(subprocess.check_output(["arp", "-a", host]))
288
289         if "aucune" in answer:
290             return False
291         else:
292             # Splitting the answer into a list of words.
293             answer = answer.split(" ")
294             # Return only the MAC address
295             return answer[3]
296     else:
297         return False
298
299 @app.errorhandler(HTTPException)
300 def handle_exception(e):
301     """handle_exception : handle http error
302
303     :param e: The http error.
304     :type e: HTTPException
305     :return: The HTML error page.
306     :rtype: render_template
307     """
308
309     e = str(e) # 404 not ... : The request URL ...
310     code = e.split(":") # 404 Not Found
311
312     # This is a function that takes in an error code and returns the error page for
313     # that error.
314     return render_template('404.html', error=code[1], title=code[0]), code[0][0:3]
315
316 @app.route("/login", methods=['POST', 'GET'])
317 @app.route("/", methods=['POST', 'GET'])
318 def login():
319     """login : login page
320     Receive from POST : username / password
321     We first check if username existe and after password.
322     If it's a bad username or pass we flash the error.
323
324     :return: The login page.
325     :rtype: render_template
326     """
327
328     # Creating a variable called current_time that is equal to the current time.
329     current_time, _ = get_time()
330     # This code is getting the user's IP address and returning it.
331     ip = request.environ.get('HTTP_X_REAL_IP', request.remote_addr)
332
333     # check for authorized ip
334     for i in authorize_ip:
335         if ip == i:
336             logging.info("create local user" + str(ip))

```

```

336         session['user_id'] = 3
337         return redirect(url_for('page1'))
338
339     # This code is checking to see if the user is logged in.
340     # If they are, then they will be directed to the page 1.
341     if session.get("user_id") is not None:
342         return redirect(url_for('page1'))
343
344     # This code is checking to see if the request method is a POST. If it is, it
345     # will run the code below.
346     if request.method == 'POST':
347         # This code is removing the user_id from the session.
348         session.pop('user_id', None)
349
350         username = request.form['username']
351         password = request.form['password']
352
353         try:
354             # This code is using a list comprehension to search for a user with a
355             # given username.
356             user = [x for x in users if x.username == username][0]
357             # The above code is checking if the user exists and if the password is
358             # correct.
359             if user and user.password == password:
360                 # Login accepted
361                 session['user_id'] = user.id
362                 logging.info("New login username : " + username + " ip : " + str(ip))
363                 return redirect(url_for('page1'))
364             elif user.password != password:
365                 # bad password, logging it and flash the error
366                 logging.warning("bad password : " + username + " ip : " + str(ip))
367                 flash("Bad password")
368
369             return redirect(url_for('login'))
370
371         except IndexError:
372             # bad username, logging it and flash the error
373             logging.warning("bad username : " + username + " ip : " + str(ip))
374             flash("Bad username")
375             return redirect(url_for('login'))
376
377     return render_template("login.html", time=current_time, warning=warning)
378
379 @app.route("/login2", methods=['POST', 'GET'])
380 def login2():
381     """The login2 function is the second page of the login page.
382     It is only for the touch screen. It is called when the user want to go in the
383     settings page.
384     It checks if the user has entered the correct password. If the user has entered
385     the correct
386     password, the user is redirected to the settings page.
387
388     :return: The login2.html template is being returned.
389     :rtype: render_template
390     """
391
392     # Creating a function that will return the current time.
393     current_time, _ = get_time()
394     # This code is getting the user's IP address and returning it.
395     ip = request.environ.get('HTTP_X_REAL_IP', request.remote_addr)
396
397     # The above code is checking if the IP address of the user is in the list of
398     # authorized IP addresses.
399     if ip not in authorize_ip:
400         return redirect(url_for('login'))
401
402     global mot_de_pass
403
404     # This code is checking to see if the request method is a POST. If it is, it
405     # will run the code below.
406     if request.method == 'POST':

```

```

401         button_click = request.form.get('btn')
402
403         # Checking if the button click is a number between 0 and 9.
404         if button_click in ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]:
405             # The above code is adding in the list all the buttons that are clicked.
406             mot_de_pass.append(button_click)
407
408         # Checking if the button click is 10 (Effacer)
409         if button_click == "10":
410             # Clear the list
411             mot_de_pass = []
412
413         # Checking if the button click is 11 (Submit)
414         if button_click == "11":
415             # The code above is taking the list of characters and joining them
416             # together to form a string.
417             mot_de_pass = "".join(mot_de_pass)
418             # This code is using a list comprehension to find the user named "local"
419             # and then returning that user.
420             user = [x for x in users if x.username == "local"][0]
421             # The above code is checking if the password entered by the user is the
422             # same as the password stored in the
423             # database.
424             if mot_de_pass == user.password:
425                 mot_de_pass = []
426                 return redirect(url_for('settings', authorized=1))
427
428             mot_de_pass = []
429
430         return render_template('login2.html', mdp=mot_de_pass, time=current_time,
431                               warning=warning)
432
433 @app.route("/page1", methods=['POST', 'GET'])
434 def page1():
435     """This function is used to display the page 1
436     It is the first page of the application.
437
438     :return: The html page is being returned.
439     :rtype: render_template
440     """
441
442     # Creating a function that will return the current time.
443     current_time, _ = get_time()
444
445     # This code is checking to see if the user is logged in. If they are not logged
446     # in, then they are redirected to the
447     # login page.
448     if session.get("user_id") is None:
449         return redirect(url_for('login'))
450
451     # This code is checking if all the elements in the list are equal to "on".
452     if all(elem == "on" for elem in buttonSts_p1):
453         # Button "all on" in page 2 is turn to "on"
454         buttonSts_p2[0] = "on"
455     else:
456         buttonSts_p2[0] = "off"
457
458     # This code is checking to see if the request method is a POST. If it is, it
459     # will run the code below.
460     if request.method == 'POST':
461         # Checking if the button has been pressed.
462         if request.form['button_p1'] == '1':
463             if buttonSts_p1[0] == "on":
464                 buttonSts_p1[0] = "off"
465                 color[0] = "#333333"
466             else:
467                 buttonSts_p1[0] = "on"
468                 color[0] = "#FFFFFF"
469         elif request.form['button_p1'] == '2':
470             if buttonSts_p1[1] == "on":
471                 buttonSts_p1[1] = "off"

```

```

467         color[1] = "#333333"
468     else:
469         buttonSts_pl[1] = "on"
470         color[1] = "#FFFFFF"
471     elif request.form['button_pl'] == '3':
472         if buttonSts_pl[2] == "on":
473             buttonSts_pl[2] = "off"
474             color[2] = "#333333"
475         else:
476             buttonSts_pl[2] = "on"
477             color[2] = "#FFFFFF"
478     elif request.form['button_pl'] == '4':
479         if buttonSts_pl[3] == "on":
480             buttonSts_pl[3] = "off"
481             color[3] = "#333333"
482         else:
483             buttonSts_pl[3] = "on"
484             color[3] = "#FFFFFF"
485     elif request.form['button_pl'] == '5':
486         if buttonSts_pl[4] == "on":
487             buttonSts_pl[4] = "off"
488             color[4] = "#333333"
489         else:
490             buttonSts_pl[4] = "on"
491             color[4] = "#FFFFFF"
492     elif request.form['button_pl'] == '6':
493         if buttonSts_pl[5] == "on":
494             buttonSts_pl[5] = "off"
495             color[5] = "#333333"
496         else:
497             buttonSts_pl[5] = "on"
498             color[5] = "#FFFFFF"
499     elif request.form['button_pl'] == '7':
500         if buttonSts_pl[6] == "on":
501             buttonSts_pl[6] = "off"
502             color[6] = "#333333"
503         else:
504             buttonSts_pl[6] = "on"
505             color[6] = "#FFFFFF"
506     elif request.form['button_pl'] == '8':
507         if buttonSts_pl[7] == "on":
508             buttonSts_pl[7] = "off"
509             color[7] = "#333333"
510         else:
511             buttonSts_pl[7] = "on"
512             color[7] = "#FFFFFF"
513     elif request.form['button_pl'] == 'page_2':
514         return redirect(url_for('page2'))
515     else:
516         pass
517
518     gpio_modif()
519
520     return render_template('page1.html', button=buttonSts_pl, color=color,
521                             time=current_time, warning=warning)
522
523 @app.route("/page2", methods=['POST', 'GET'])
524 def page2():
525     """This function is used to display the page 2
526     It is the second page of the application.
527
528     :return: The page is being rendered.
529     :rtype: render_template
530     """
531
532     # Creating a variable called current_time that is equal to the current time.
533     current_time, _ = get_time()
534
535     # This code is checking to see if the user is logged in. If they are not logged
536     # in, then they are redirected to the
537     # login page.

```



```

537 if session.get("user_id") is None:
538     return redirect(url_for('login'))
539
540 # This code is checking to see if the request method is a POST. If it is, it
541 # will run the code below.
542 if request.method == 'POST':
543     # All on
544     if request.form['button_p1'] == '1':
545         buttonSts_p2[0] = "on"
546         for i in range(8):
547             buttonSts_p1[i] = "on"
548             color[i] = "#FFFFFF"
549     # All off
550     elif request.form['button_p1'] == '2':
551         buttonSts_p2[1] = "off"
552         for i in range(8):
553             buttonSts_p1[i] = "off"
554             buttonSts_p2[i] = "off"
555             color[i] = "#333333"
556     # 1 / 2
557     elif request.form['button_p1'] == '3':
558         buttonSts_p2[0] = "off"
559         buttonSts_p2[1] = "off"
560
561         for i in range(0, 8, 2):
562             color[i] = "#FFFFFF"
563             color[i + 1] = "#333333"
564
565             buttonSts_p1[i] = "on"
566             buttonSts_p1[i + 1] = "off"
567     # Left
568     elif request.form['button_p1'] == '4':
569         buttonSts_p2[0] = "off"
570         buttonSts_p2[1] = "off"
571
572         for i in range(0, 8):
573             if i < 3:
574                 color[i] = "#FFFFFF"
575                 buttonSts_p1[i] = "on"
576             else:
577                 color[i] = "#333333"
578                 buttonSts_p1[i] = "off"
579     # Right
580     elif request.form['button_p1'] == '5':
581         buttonSts_p2[0] = "off"
582         buttonSts_p2[1] = "off"
583
584         for i in range(0, 8):
585             if i > 3 and i != 7:
586                 color[i] = "#FFFFFF"
587                 buttonSts_p1[i] = "on"
588             else:
589                 color[i] = "#333333"
590                 buttonSts_p1[i] = "off"
591
592     elif request.form['button_p1'] == 'page_1':
593         return redirect(url_for('page1'))
594     else:
595         pass
596
597     gpio_modif()
598
599 return render_template('page2.html', button=buttonSts_p2, color=color,
600 time=current_time, warning=warning)
601
602 @app.route("/settings", methods=['POST', 'GET'])
603 def settings():
604     """The settings page is used to change the settings of the application.
605
606     :return: The settings.html template is being returned.
607     :rtype: render_template

```

```

607     """
608
609     id = session.get("user_id")
610     authorized = request.args.get('authorized')
611
612     # Get the id for the session.
613     # If == None --> go to th page login.
614     # If == 3 (local) --> go to the page login 2 or settings if password ok
615     # If == 1 (elo) --> remove the id / error 418
616     if id is None:
617         return redirect(url_for('login'))
618     elif id == 3 and authorized != "1":
619         return redirect(url_for('login2'))
620     elif id == 1:
621         session.pop("user_id")
622         abort(418)
623
624     # Calling a function that will return the current time.
625     current_time, _ = get_time()
626     # This code is getting the user's IP address and returning it.
627     ip = request.environ.get('HTTP_X_REAL_IP', request.remote_addr)
628
629     global mode_manuel
630     global mode_heure_manuel
631     global cam_connect
632
633     t = Table()
634     t2 = Table2()
635
636     # This code is checking to see if the request method is a POST. If it is, it
637     # will run the code under it.
638     if request.method == 'POST':
639
640         # This code is getting the data from the form and putting it into variables.
641         btn_tbl_1 = request.form.get('btn_tbl_1')
642         btn_tbl_2 = request.form.get('btn_tbl_2')
643         nouvelle_heure = request.form.get('time')
644         supprimer = request.form.get('supp')
645
646         manuel = request.form.get('manuel')
647         heure_man = request.form.get('heure_manuel')
648         heure_man_btn = request.form.get('heure_manuel_btn')
649
650         modif_mdp = request.form.get('modif_mdp')
651         password = request.form.get('password')
652         new_password = request.form.get('new_password')
653
654         if btn_tbl_1:
655             # The above code is converting the number of table's cell into a row and
656             # column number.
657             btn_click = float(btn_tbl_1)
658             row = int(btn_click)
659             column = round((btn_click - row) * 10)
660
661             # This code is calling "modif_table".
662             t.modif_table(row, column)
663             t.set_table()
664
665         if btn_tbl_2:
666             # The above code is converting the number of table's cell into a row and
667             # column number.
668             btn_click = float(btn_tbl_2)
669             row = int(btn_click)
670             column = round((btn_click - row) * 10)
671
672             # This code is calling "modif_table" for table 2.
673             t2.modif_table(row, column)
674             t2.set_table()
675
676         if nouvelle_heure:
677             # Adding a custom line to the table.
678             t.add_custom_line(nouvelle_heure)

```

```

676         t.set_table()
677         t.set_colonne_heure()
678
679     if supprimer:
680         # Deleting the custom line in the table 1.
681         t.del_custom_line(supprimer)
682
683     if manuel:
684         # This code is inverting the manual mode.
685         mode_manuel = 1 - int(manuel)
686
687     if heure_man_btn:
688         # This code is inverting the manual mode.
689         mode_heure_manuel[0] = 1 - int(heure_man_btn)
690
691         if mode_heure_manuel[0] == 1:
692             # Converting the time from the website into a datetime object.
693             # Then it is subtracting the current time from the time from the
694             # website.
695             # Then it is converting the timedelta object into a string.
696             x = datetime.fromisoformat(heure_man)
697             diff = x - datetime.strptime(datetime.now().strftime("%H:%M"),
698                                         "%H:%M")
699             diff = timedelta(seconds=diff.seconds)
700
701             # Store the new day (0 is monday).
702             mode_heure_manuel[1] = x.weekday()
703             # Store the delta time
704             mode_heure_manuel[2] = diff
705
706         else:
707             # The above code is setting mode_heure_manuel to 0.
708             mode_heure_manuel = [0, 0, 0]
709
710         # Creating a function that will be used to get the current time.
711         current_time, _ = get_time()
712
713     if modifier_mdp:
714         # This code is searching the list of users for the user with the same
715         # username as the one who is trying to
716         # change their password.
717         user = [x for x in users if x.username == modifier_mdp][0]
718         # The above code is checking if the user exists and if the password is
719         # correct.
720         if user and user.password == password:
721             # Checking if the user has selected the local option and if the new
722             # password is not only number.
723             if modifier_mdp == "local" and not new_password.isdigit():
724                 flash("New password for local user can only contains numbers")
725             else:
726                 logging.info("Modification password : " + modifier_mdp + " new : "
727                             + new_password + " ip : " + str(ip))
728                 user.password = new_password
729
730                 # Save the new password on the pickle file
731                 with open('password.pickle', 'rb') as f:
732                     passw = pickle.load(f)
733                 with open('password.pickle', 'wb') as f:
734                     passw[int(user.id) - 1] = new_password
735                 pickle.dump(passw, f, pickle.HIGHEST_PROTOCOL)
736
737             elif user.password != password:
738                 # bad password, logging it and flash the error
739                 logging.warning("Try to modify the password but introduce wrong
740                                password : " + modifier_mdp + " ip : " + str(ip))
741                 flash("Wrong password")
742
743     table = t.get_table()
744     heure = t.get_colonne_heure()
745     new_heure = t.get_new_colonne_heure()
746
747     table2 = t2.get_table()

```

```

741     name_t2 = t2.get_name()
742
743     return render_template('settings.html', table=table, heure=heure,
744                            new_heure=new_heure, table2=table2,
745                            name_t2=name_t2, mode_manuel=mode_manuel,
746                            mode_heure_manuel=mode_heure_manuel[0],
747                            cam_conn=cam_connect, time=current_time, warning=warning)
748
749 @app.route('/camera', methods=['POST', 'GET'])
750 def camera():
751     """The camera is only here to receive data from the camera. The script is
752     checking mac address to verify if the
753     sender is a RPi camera. If it is, the server is checking if the data is
754     corresponding with the ELO protocol
755     (the first byte is 0x55 and the last byte is 0xAA). If it is, the server is
756     checking if the data is corresponding.
757
758     :return: The server is returning the string '$,RPWCOK,002,ok,0*' or
759     '$,RPWCER,005,error,0*' to the camera.
760     :rtype: str
761     """
762
763     global cam_connect
764     global buttonSts_pl
765     global color
766     global cam_send_turn_off
767     global number_of_people
768
769     # This code is getting the user's IP address and returning it.
770     ip = request.environ.get('HTTP_X_REAL_IP', request.remote_addr)
771     # Calling a function that will return the current time.
772     current_time, current_day = get_time()
773     # This code is retrieving the user's id from the session.
774     id = session.get("user_id")
775
776     # Check if the user have an id
777     # if yes => 404
778     if id is not None:
779         abort(404)
780     # else check is mac add
781     else:
782         # This code is using the getmacadd function to get the mac address of the ip
783         address.
784         mac_add = getmacadd(ip)
785
786         # The above code is checking to see if the MAC address of the camera is in
787         the list of MAC addresses of the
788         # cameras that are connected to the system.
789         # If the MAC address is not in the list, error 404.
790         if not any(mac_add in x for x in cam_connect) or mac_add is False:
791             abort(404)
792
793     # Checking if the request method is POST. If it is, it will run the code below.
794     if request.method == 'POST':
795         data = request.form.get('data')
796         # Checking if the data is not empty.
797         if data:
798             # Verify if data received corresponding with ELO communication protocol
799             if data[0] == "$" and data[:: -1][0] == "*":
800                 # The above code is reading the data from the file and splitting it
801                 into a list.
802                 data = data.split(',')
803                 # RPWCSD is the header (Raspberry Pi Wi-Fi Camera Send Data)
804                 if data[1] == "RPWCSD":
805                     # Length of the data
806                     length = int(data[2])
807                     # Time when it was sent
808                     h = data[3][0:5]
809                     # Temperature of the camera
810                     cam_temp = int(data[3][5:length - 1])
811                     # data : 1 entry, 0 exit

```



```

861         zip(array_t2, array_comp)]
862
863         # If the value is 0, it will be "off",
864         # otherwise it will be "on".
865         buttonSts_p1 = ["off" if x == 0 else "on"
866                         for x in array_t2_or]
867         # If the value is 0, it will be "#333333",
868         # otherwise it will be "#FFFFFF".
869         color = ["#333333" if x == 0 else "#FFFFFF"
870                 for x in array_t2_or]
871
872         # Reset cam_send_turn_off
873         cam_send_turn_off = []
874
875         # Turn on the light
876         gpio_modif()
877         logging.info(" *** Camera turn on the light.
878                     *** ")
879
880         elif d == "0" and number_of_people == 0:
881             # Add the current_time in cam_send_turn_off.
882             # After TURNOFFTIME if any other action is
883             # arrive turn off all light
884             cam_send_turn_off.append(current_time)
885
886             break
887
888         # This code is checking the list of camera IP addresses and if
889         # the IP address is not in the list, it
890         # will add it to the list.
891         for i, x in enumerate(cam_connect):
892             if mac_add in x and ip not in x:
893                 cam_connect[i][1] = ip
894                 logging.info(" *** New camera : " + ip + " *** ")
895
896         # Return OK to the camera with respecting the protocol of
897         # communication ELO.
898         return '$,RPWCOK,002,ok,0*'
899
900         # Return error to the camera with respecting the protocol of communication ELO.
901         return '$,RPWCER,005,error,0*'
902
903 @app.before_first_request
904 def activate_job():
905     """activate_job : fonction to run job on background of the web server (with
906     thread)
907
908     :return: None
909     :rtype: None
910     """
911
912     def run_job():
913         """run_job : run background job (run every minute).
914         Automatically turn off the light.
915         Check temperature of the Raspberry Pi
916
917         :return: None
918         :rtype: None
919         """
920
921         global mode_manuel
922         global mode_heure_manuel
923         global cam_connect
924         global cam_send_turn_off
925         global TURNOFFTIME
926         global number_of_people
927
928         current_time, _ = get_time()
929         saved_time = current_time
930
931         logging.debug(" *** Starting while loop *** ")
932
933         # While loop that will run forever.

```

```

923 while True:
924     # Wait a minute.
925     while saved_time == current_time:
926         current_time, current_day = get_time()
927
928     saved_time = current_time
929
930     # The above code is checking if the current time is 00:00 (midnight).
931     # If it is, it sets the mode_manuel to 0 and the mode_heure_manuel to
932     # [0, 0, 0].
933     if current_time == "00:00":
934         number_of_people = 0
935         mode_manuel = 0
936         mode_heure_manuel = [0, 0, 0]
937         logging.info(" *** Reset midnight ***")
938
939     # check for automatic off on the table 1 or if camera requested a shutdown
940     if not mode_manuel:
941         # Getting all the value of the table 1.
942         t = Table()
943         table = t.get_table()
944         table_heure = t.get_colonne_heure()
945
946         # Check for automatic turn off
947         for i in range(len(table_heure)):
948             # Check if current_time is equal to one of the times in the table.
949             if current_time == table_heure[i]:
950                 # Ff the box in the table is blue.
951                 if table[i][int(current_day)] > 1:
952                     # Turn all off.
953                     logging.info(" *** Server turn all off automatically ***")
954
955                     for x in range(8):
956                         buttonSts_p1[x] = "off"
957                         buttonSts_p2[x] = "off"
958                         color[x] = "#333333"
959
960         # Checks if the camera has requested a shutdown
961         for time_turn_off in cam_send_turn_off:
962             # Compares the current time with the time of the requests add to
963             # TURNOFFTIME.
964             time_add = datetime.strptime(time_turn_off, '%H:%M') + TURNOFFTIME
965             time_add = time_add.strftime("%H:%M")
966             if current_time == time_add:
967                 # Clear the requests list
968                 cam_send_turn_off = []
969
970             # Split the current time
971             current_h, current_min = [x for x in current_time.split(":")]
972
973             # Find out what time slot you are in
974             for j in reversed(range(len(table_heure))):
975                 h, m = table_heure[j].split(":")
976                 if current_h == h and current_min >= m:
977                     # If camera as authorisation
978                     if table[j][int(current_day)] == 1 or
979                     table[j][int(current_day)] == 3:
980                         # Turn all off.
981                         logging.info(" *** Camera turn all the light off
982                         ***")
983
984                         for x in range(8):
985                             buttonSts_p1[x] = "off"
986                             buttonSts_p2[x] = "off"
987                             color[x] = "#333333"
988
989             # Update GPIO
990             gpio_modif()
991
992     # The above code is reading the temperature of the CPU only if it's
993     # running on a RPi.
994     if rpi:

```

```

990         with open('/sys/class/thermal/thermal_zone0/temp', 'r') as ftemp:
991             global warning
992             temp = int(ftemp.read()) / 1000
993             # This code is checking the temperature and if it is greater
             # than 60 it will send a warning.
994             if temp > 60:
995                 logging.warning(" *** Temp = " + str(int(temp)) + "°C *** ")
996                 warning = "Temp = " + str(int(temp)) + "°C"
997             else:
998                 warning = ""
999
1000         # The above code is checking to see if the cameras are online.
1001         for i in range(len(cam_connect)):
1002             ip_cam = cam_connect[i][1]
1003             cam_connect[i][0] = myping(ip_cam)
1004
1005         # This code is creating a thread that will run the run_job function.
1006         thread = threading.Thread(target=run_job)
1007         thread.start()
1008
1009
1010 if __name__ == "__main__":
1011     logging.info(" *** Starting server *** ")
1012     app.run(host='0.0.0.0', port=80, debug=False)
1013     GPIO.cleanup()
1014     logging.info(" *** Server stopped *** ")
1015

```



```
1  """wsgi.py | Robin Forestier | 2021 / 2022
2
3  This file is only used to run the server by Gunicorn.
4  """
5
6  # This is importing the app from the main.py file.
7  from main import app
8
9  if __name__ == "__main__":
10     # Running the app.
11     app.run()
12
```



```

1  """table.py | Robin Forestier | 15.03.2022
2
3  This file contains the two class for create the two table in the settings page.
4  """
5
6  # Importing the module `pickle` which allows saving and load python objects.
7  import pickle
8
9
10 # The Table class is a container for the data in a table
11 class Table:
12     def __init__(self):
13         """The function is used to initialize the class"""
14
15         # The function is used to load the new_colonne_heure variable from the
16         # pickle file
17         try:
18             self.colonne_heure = self.get_colonne_heure()
19             self.new_colonne_heure = self.get_new_colonne_heure()
20             self.table = self.get_table()
21         except FileNotFoundError:
22             self.colonne_heure = []
23             self.new_colonne_heure = []
24             self.table = []
25
26     def create(self):
27         """This function is used to create the table if it doesn't exist"""
28         self.colonne_heure = [{"{:02d}:00".format(i)} for i in range(24)]
29         self.table = [[0 for i in range(7)] for j in range(24)]
30
31     def modify_table(self, row, column):
32         """If the value of the cell is 3, change it to 0. Otherwise, add 1 to the
33         value of the cell
34
35         :param row: The row of the cell you want to change
36         :type row: int
37         :param column: The column of the cell you want to change
38         :type column: int
39         :return: The modified table.
40         :rtype: list
41         """
42
43         if self.table[row][column] == 3:
44             self.table[row][column] = 0
45         else:
46             self.table[row][column] += 1
47
48         return self.table
49
50     def add_custom_line(self, heure):
51         """Add a new line to the table if it doesn't already exist
52
53         :param heure: the time you want to add
54         :type heure: str
55         :return: The table and the list of the hours.
56         :rtype: list
57         """
58
59         existing_heure = self.get_colonne_heure()
60
61         if heure not in existing_heure:
62             self.colonne_heure.append(heure)
63             self.colonne_heure.sort()
64
65             self.set_colonne_heure()
66
67             self.new_colonne_heure.append(heure)
68             self.new_colonne_heure.sort()
69
70             self.set_new_colonne_heure()
71
72             index = self.colonne_heure.index(heure)

```

```

71         self.table.insert(index, [0 for i in range(7)])
72
73         self.set_table()
74
75         return self.table, self.colonne_heure
76
77     def del_custom_line(self, heure):
78         """It deletes the line of the table corresponding to the given heure
79
80         :param heure: the hour you want to delete
81         :type heure: str
82         """
83         index = self.new_colonne_heure.index(heure)
84         self.new_colonne_heure.pop(index)
85
86         index = self.colonne_heure.index(heure)
87         self.colonne_heure.pop(index)
88
89         self.table.pop(index)
90
91         self.set_new_colonne_heure()
92         self.set_colonne_heure()
93         self.set_table()
94
95     def set_table(self):
96         """This function is used to save the table to a pickle file"""
97
98         with open('./table/settings/table.pickle', 'wb') as f:
99             pickle.dump(self.table, f, pickle.HIGHEST_PROTOCOL)
100
101     def get_table(self):
102         """It loads the table from the pickle file
103
104         :return: The table
105         :rtype: list
106         """
107
108         try:
109             with open('./table/settings/table.pickle', 'rb') as f:
110                 # The protocol version used is detected automatically, so we do not
111                 # have to specify it.
112                 self.table = pickle.load(f)
113
114             return self.table
115         except FileNotFoundError:
116             self.create()
117             self.set_table()
118             self.set_colonne_heure()
119
120             self.get_table()
121
122     def set_colonne_heure(self):
123         """The function is used to save the value of the variable "colonne_heure" in
124         a file called "heure.pickle" """
125
126         with open('./table/settings/heure.pickle', 'wb') as f:
127             pickle.dump(self.colonne_heure, f, pickle.HIGHEST_PROTOCOL)
128
129     def get_colonne_heure(self):
130         """The function is used to load the value of the variable "colonne_heure"
131         from a file called "heure.pickle
132
133         :return: The list of the hours
134         :rtype: list
135         """
136
137         try:
138             with open('./table/settings/heure.pickle', 'rb') as f:
139                 # The protocol version used is detected automatically, so we do not
140                 # have to specify it.
141                 self.colonne_heure = pickle.load(f)
142
143             return self.colonne_heure

```

```

141         except FileNotFoundError:
142             self.create()
143             self.set_table()
144             self.set_colonne_heure()
145
146             self.get_colonne_heure()
147
148     def set_new_colonne_heure(self):
149         """The function is used to save the new_colonne_heure variable in a pickle
150         file"""
151
152         with open('./table/settings/new_heure.pickle', 'wb') as f:
153             pickle.dump(self.new_colonne_heure, f, pickle.HIGHEST_PROTOCOL)
154
155     def get_new_colonne_heure(self):
156         """The function is used to get the new_colonne_heure from the pickle file
157
158         :return: The new_colonne_heure variable is being returned
159         :rtype: list
160         """
161
162         try:
163             with open('./table/settings/new_heure.pickle', 'rb') as f:
164                 # The protocol version used is detected automatically, so we do not
165                 # have to specify it.
166                 self.new_colonne_heure = pickle.load(f)
167
168                 return self.new_colonne_heure
169         except FileNotFoundError:
170             self.set_new_colonne_heure()
171             self.get_new_colonne_heure()
172
173 # It inherits from Table and overrides the __init__ method.
174 class Table2(Table):
175     def __init__(self):
176         """The __init__ function is called when an instance of the class is created.
177         used to initialize the attributes of the class.
178         """
179
180         try:
181             self.table = self.get_table()
182             self.name = self.get_name()
183         except FileNotFoundError:
184             self.table = []
185             self.name = []
186
187     def create(self):
188         """Create a table with 8 rows and 7 columns and create the name column."""
189
190         self.table = [[0 for i in range(7)] for j in range(8)]
191         self.name = ["Ligne 1", "Ligne 2", "Ligne 3", "Ligne 4", "Ligne 5", "Ligne
192         6", "Ligne 7", "couloir"]
193
194     def modif_table(self, row, column):
195         """Invert the value of a cell in the table
196
197         :param row: The row of the cell you want to change
198         :type row: int
199         :param column: The column of the cell you want to change
200         :type column: int
201         :return: The modified table
202         :rtype: list
203         """
204
205         if self.table[row][column] == 1:
206             self.table[row][column] = 0
207         else:
208             self.table[row][column] = 1
209
210         return self.table

```

```

211 def set_table(self):
212     """The function opens the pickle file and dumps the table into the pickle
    file"""
213
214     with open('./table/settings/table2.pickle', 'wb') as f:
215         pickle.dump(self.table, f, pickle.HIGHEST_PROTOCOL)
216
217 def get_table(self):
218     """Load the table from the pickle file
219
220     :return: A table object
221     :rtype: list
222     """
223
224     try:
225         with open('./table/settings/table2.pickle', 'rb') as f:
226             # The protocol version used is detected automatically, so we do not
227             # have to specify it.
228             self.table = pickle.load(f)
229
230             return self.table
231     except FileNotFoundError:
232         self.create()
233         self.set_table()
234         self.set_table()
235
236         self.get_table()
237
238 def set_name(self):
239     """This function is used to set the name of the user"""
240
241     with open('./table/settings/name.pickle', 'wb') as f:
242         pickle.dump(self.name, f, pickle.HIGHEST_PROTOCOL)
243
244 def get_name(self):
245     """Get the name of the user
246
247     :return: The name of the user
248     :rtype: list
249     """
250
251     try:
252         with open('./table/settings/name.pickle', 'rb') as f:
253             # The protocol version used is detected automatically, so we do not
254             # have to specify it.
255             self.name = pickle.load(f)
256
257             return self.name
258     except FileNotFoundError:
259         self.create()
260         self.set_table()
261         self.set_name()
262
263         self.get_colonne_heure()
264
265
266 if __name__ == '__main__':
267     pass
268

```

```
1  """flask_test.py | Robin Forestier | 15.03.2022
2
3  This file is used to test the Flask installation.
4  """
5
6  # Importing the Flask class from the flask module.
7  from flask import Flask
8
9  # This is creating an instance of the Flask class and assigning it to the variable
   `app`.
10 app = Flask(__name__)
11
12
13 @app.route('/')
14 def index():
15     """Create the index page.
16
17     :return: The index page.
18     :rtype: str
19     """
20
21     return f"Hello World"
22
23
24 if __name__ == '__main__':
25     # Running the app and create a local server.
26     app.run(host='0.0.0.0', port=80, debug=True)
27
```



```

1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4      <meta http-equiv="refresh" content="60">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <meta charset="UTF-8">
7      <meta name="theme-color" content="black">
8
9      <meta name="author" content="Robin Forestier">
10     <meta name="description" content="Page 1 - controle des lumières en knx">
11
12     <title>Light Control</title>
13     <link rel="icon" type="image/x-icon" href="/static/img/LogoElo.ico">
14     <link rel="shortcut icon" href="/static/img/LogoElo.ico" type="image/x-icon">
15     <link rel="stylesheet" href="../static/settings.css/">
16 </head>
17 <body>
18     <!-- create the header -->
19     <div class="header">
20         <p><a href="/page1">Retour</a></p>
21         <p>{{ time }} </p>
22         <p class="warning">{{ warning }} </p>
23         <h1>Settings</h1>
24         {% with messages = get_flashed_messages() %}
25         {% if messages %}
26             <!-- show the error message -->
27             <div class="error">
28                 {% for message in messages %}
29                     {{ message }}
30                 {% endfor %}
31             </div>
32         {% endif %}
33         {% endwith %}
34     </div>
35
36     <!--Table 1-->
37     <form method="post">
38         <!--Title-->
39         <h2>Sélection des heures </h2>
40         <table class="table_1">
41             <!-- header -->
42             <tr>
43                 <th id="hour"></th>
44                 <th>Lundi</th>
45                 <th>Mardi</th>
46                 <th>Mercredi</th>
47                 <th>Jeudi</th>
48                 <th>Vendredi</th>
49                 <th>Samedi</th>
50                 <th>Dimanche</th>
51             </tr>
52
53             {% for tab in table %}
54             {% set iteration = loop.index0 %}
55             <tr>
56                 <!-- hour -->
57                 <td>{{ heure[iteration] }}</td>
58                 <!-- row of btn -->
59                 {% for t in tab %}
60                     <td id="n{{ t }}"> <button type="submit" name="btn_tbl_1"
61                         value="{{ iteration }}.{{ loop.index0 }}"> &#8203;
62                     </button> </td>
63                 {% endfor %}
64             </tr>
65             {% endfor %}
66         </table>
67     </form>
68
69     <hr>
70
71     <!-- Table 2 -->
72     <form method="post">

```

```

71     <!-- Title -->
72     <h2>Sélection des lumières </h2>
73     <table class="table_1">
74         <!-- header -->
75         <tr>
76             <th id="hour"></th>
77             <th>Lundi</th>
78             <th>Mardi</th>
79             <th>Mercredi</th>
80             <th>Jeudi</th>
81             <th>Vendredi</th>
82             <th>Samedi</th>
83             <th>Dimanche</th>
84         </tr>
85
86         {% for tab in table2 %}
87         {% set iteration = loop.index0 %}
88         <tr>
89             <!-- hour -->
90             <td>{{ name_t2[iteration] }}</td>
91             <!-- row of btn -->
92             {% for t in tab %}
93                 <td id="n{{ t }}"> <button type="submit" name="btn_tbl_2"
94                     value="{{ iteration }}.{{ loop.index0 }}"> &#8203;
95                     </button> </td>
96             {% endfor %}
97         </tr>
98         {% endfor %}
99     </table>
100 </form>
101
102 <hr>
103
104 <!-- add a custom hour -->
105 <div class="sett">
106     <!-- title -->
107     <h2>Ajouter une nouvelle heure</h2>
108     <form method="post">
109         <input type="time" id="time" name="time" required>
110         <button type="submit">Submit</button>
111     </form>
112 </div>
113
114 <hr>
115
116 <!-- del a custom hour -->
117 <div class="sett">
118     <!-- title -->
119     <h2>Supprimer une heure</h2>
120     <form method="post">
121         <select name="supp">
122             <!-- basic value but disable -->
123             <option value="" disabled selected>-- Supprimer une heure --</option>
124
125             {% if new_heure %}
126             <!-- custom hour -->
127             {% for new in new_heure %}
128                 <option value="{{ new }}">{{ new }}</option>
129             {% else %}
130
131             {% endfor %}
132             {% endif %}
133         </select>
134         <button type="submit">Submit</button>
135     </form>
136 </div>
137
138 <hr>
139
140 <!-- Manual mode -->
141 <div class="sett">
142     <!-- title -->

```

```

141     <h2>Mode manuel</h2>
142     <form method="post">
143         <!-- Slide button, differ on class -->
144         <button type="submit" name="manuel" class="c{{ mode_manuel }}" value={{
mode_manuel }}></button>
145     </form>
146     <p>Le mode manuel désactive les caméras tant qu'il est activé ou jusqu'à
minuit.</p>
147 </div>
148
149 <hr>
150
151 <!-- Hour manual -->
152 <div class="sett">
153     <!-- title -->
154     <h2>Date et heur manuel</h2>
155     <form method="post">
156
157         {% if mode_heure_manuel == 0 %}
158             {% set var = "required" %}
159         {% else %}
160             {% set var = "" %}
161         {% endif %}
162
163         <input type="datetime-local" id="heure_manuel" name="heure_manuel" {{
var }}>
164         <button type="submit" name="heure_manuel_btn" class="c{{
mode_heure_manuel }}" value={{ mode_heure_manuel }}></button>
165     </form>
166     <p>Le mode date et heure manuelle est activé jusqu'à minuit.</p>
167 </div>
168
169 <hr>
170
171 <!-- Camera infos -->
172 <div class="sett">
173     <h2>Caméras</h2>
174
175     {% for cam in cam_conn %}
176         {% if cam[1] != "" %}
177             {% if cam[0] == True %}
178                 {% set var = "Online" %}
179             {% else %}
180                 {% set var = "Offline" %}
181             {% endif %}
182
183             <p>Caméra {{ loop.index }} : <span class={{ var }}> {{ var }}
</span> IP : {{ cam[1] }}
184                 {% if cam[3] != 0 %}
185                     <span>Température: </span> <span style="color:#f33"> {{
cam[3] }} </span>
186                 {% endif %}
187             </p>
188         {% endif %}
189     {% endfor %}
190 </div>
191
192 <hr>
193
194 <!-- Modification of password -->
195 <div class="modif_pass">
196     <h2> Modifier mot de passe </h2>
197     <form method="post">
198         <div class="select">
199             <select name="modif_mdp" required>
200                 <option value="" disabled selected>-- Modifier mot de passe
--</option>
201                 <option value="elo">elo</option>
202                 <option value="admin">admin</option>
203                 <option value="local">local</option>
204             </select>
205         </div>

```

```
206
207     <div class="password">
208         <input type="password" name="password" required>
209         <label>Password</label>
210     </div>
211
212     <div class="password">
213         <input type="password" name="new_password" required>
214         <label>New password</label>
215     </div>
216
217     <button class="btn" type="submit"> Submit </button>
218 </form>
219 </div>
220
221 <!-- Use for css only -->
222 <div class="space" style="visibility:hidden">.</div>
223
224 </body>
225 </html>
```

```

1  /* sttings.css | ../static/settings.css | Robin Forestier | 12.2021
2  *
3  * Used for settings.html on ../templates/settings.html
4  */
5
6  * {
7      font-family: arial;
8  }
9
10 /* The default CSS for the body of the page. */
11 html,
12 body {
13     height: 100%;
14     width: 100%;
15     height: 100vh;
16     margin: 0;
17     padding: 0;
18     background: linear-gradient(to bottom right, #333 0%, #777 100%);
19     background-attachment: fixed;
20     box-sizing: border-box;
21 }
22
23 /* Header */
24 .header {
25     float: left;
26     width: 100%;
27     height: auto;
28     line-height: 25px;
29     background-color: #333333;
30     color: white;
31     vertical-align: middle;
32     position: relative;
33     z-index: 10;
34 }
35
36 .header h1 {
37     margin-left: 10px;
38 }
39
40 .header p {
41     float: right;
42     font-size: 20px;
43     margin-right: 20px;
44 }
45
46 /* Modification for the button "retour" */
47 .header a {
48     color: #fff;
49     text-decoration: none;
50     padding: 10px;
51     background-color: #777;
52     border-radius: 10px;
53 }
54
55 .warning {
56     color: #F00;
57 }
58
59 /* error for bad password (show in the header but at the second line) */
60
61 .error {
62     position: relative;
63     float: left;
64     width: auto;
65     left: 50%;
66     height: 50px;
67     line-height: 50px;
68     font-size: 20px;
69     color: #FFF;
70     box-shadow: inset -2px -3px rgba(255, 255, 255, 0.2);
71     padding-left: 70px;
72     padding-right: 10px;
73     margin-bottom: 10px;

```

```

74     transform: translate(-50%, 0);
75     border-radius: 15px;
76     background: rgba(255, 0, 0, 0.6);
77 }
78
79 .error:after {
80     content: "❌";
81     font-size: 30px;
82     text-align: center;
83     line-height: 35px;
84     position: absolute;
85     width: 40px;
86     height: 40px;
87     left: 10px;
88     top: 5px;
89     background: rgba(0, 0, 0, 0.3);
90     border-radius: 50%;
91 }
92
93 /* basics sizing */
94
95 hr {
96     border: 0.1px solid #333;
97 }
98
99 form, h2 {
100     position: relative;
101     margin: 0;
102 }
103
104 form>h2, .table_1 {
105     width: 90%;
106     left: 50%;
107     transform: translate(-50%, 0);
108     position: relative;
109 }
110
111 p {
112     color: #fff;
113     font-size: 16px;
114 }
115
116 h2 {
117     color: #ddd;
118 }
119
120 /* Table */
121
122 .table_1 {
123     padding: 5px;
124     color: #aaa;
125     text-align: center;
126 }
127
128 /* Background of table's header */
129 .table_1 th {
130     background: linear-gradient(rgba(0, 0, 0, 0.4),
131                                rgba(0, 0, 0, 0.3) 10%,
132                                rgba(0, 0, 0, 0.4) 40%,
133                                rgba(0, 0, 0, 0.5) 80%,
134                                rgba(0, 0, 0, 0.55) 100%);
135     width: 10%;
136 }
137
138 .table_1 tr {
139     height: 30px;
140 }
141
142 .table_1 td {
143     border: 1px solid;
144     padding: 0;
145 }
146

```

```

147 .table_1 th#hour {
148     width: 10%;
149 }
150
151 .table_1 td#n0 {
152
153 }
154
155 /* Orange */
156 .table_1 td#n1 {
157     background: #F56217;
158 }
159
160 /* Blue */
161 .table_1 td#n2 {
162     background: #0B486B;
163 }
164
165 /* Orange and Blue */
166 .table_1 td#n3 {
167     background: linear-gradient(to bottom right ,
168                                 #F56217 48%, #0B486B 52%);
169 }
170
171 /* Button in cells */
172 .table_1 button {
173     float: none;
174     width: 100%;
175     height: 20%;
176
177     background: transparent;
178     border: none;
179     inline: none;
180 }
181
182 .sett {
183     width: 90%;
184     left: 50%;
185     transform: translate(-50%, 0);
186     position: relative;
187 }
188
189 /* add and del hour */
190
191 .sett input#time {
192     color: #fff;
193     width: 175px;
194     height: 25px;
195     font-size: 16px;
196     border-radius: 5px;
197     background: transparent;
198 }
199
200 /* Manual mode */
201
202 button[class*="c"] {
203     background: #777;
204     width: 60px;
205     height: 30px;
206     border: none;
207     border-radius: 15px;
208     position: relative;
209 }
210
211 button[class*="c"]:after {
212     content: "o";
213     background: #333;
214     color: #777;
215     line-height: 22px;
216     top: 50%;
217     transform: translate(0, -50%);
218     width: 40%;
219     height: 80%;

```

```

220     border-radius: 50%;
221     position: absolute;
222 }
223
224 button.c0:after {
225     content: "x";
226     color: #777;
227     transform: translate(-100%, -50%);
228     background: #333;
229 }
230
231 /* Camera Infos */
232
233 .Online {
234     color: #3f3;
235 }
236
237 .Offline {
238     color: #f33;
239 }
240
241 /* Form and select positioning */
242
243 form>input {
244     margin: 10px 0px 10px 0px;
245 }
246
247 form>button {
248     float: right;
249     width: 130px;
250     height: 40px;
251     font-size: 18px;
252     border-radius: 10px;
253     color: #fff;
254     background : transparent;
255     box-shadow: 0 1px 5px rgba(0, 0, 0, .4);
256     background-image: linear-gradient(rgba(90, 90, 90, 1) 0%,
257                                     rgba(90, 90, 90, 0.3) 25%,
258                                     rgba(90, 90, 90, 0.15) 50%,
259                                     rgba(90, 90, 90, 0.075) 65%,
260                                     rgba(90, 90, 90, 0.037) 80%,
261                                     rgba(90, 90, 90, 0.019) 88%,
262                                     rgba(90, 90, 90, 0) 100%);
263 }
264
265 select {
266     margin: 10px 0px 10px 0px;
267     width: 180px;
268     color: #fff;
269     background: transparent;
270     padding: 5px 0;
271 }
272
273 option {
274     background: #333;
275 }
276
277 /* Date et hour manual */
278
279 .sett input#heure_manuel {
280     color: #fff;
281     width: 175px;
282     height: 25px;
283     font-size: 16px;
284     border-radius: 5px;
285     background: transparent;
286 }
287
288 /* Password modif */
289
290 .modif_pass {
291     position: relative;
292     width: 90%;

```



```

293     left: 50%;
294     transform: translate(-50%, 0);
295     margin-bottom: 20px;
296 }
297
298 .modif_pass .password {
299     position: relative;
300 }
301
302 .modif_pass .password input{
303     width: 100%;
304     padding: 10px 0;
305     font-size: 16px;
306     color: #FFF;
307     margin-bottom: 30px;
308     outline: none;
309     border: none;
310     border-bottom: 1px solid #FFF;
311     background: transparent;
312 }
313
314 .modif_pass .password label {
315     position: absolute;
316     top: 0;
317     left: 0;
318     padding: 10px 0;
319     font-size: 18px;
320     color: #FFF;
321     pointer-events: none;
322     transition: .5s;
323 }
324
325 /* animation text */
326
327 .modif_pass .password input:focus ~ label,
328 .modif_pass .password input:valid ~ label {
329     top: -20px;
330     left: 0;
331     color: #999;
332     font-size: 14px;
333 }
334
335 .space {
336     padding: 10px;
337 }
338
339 /* Only for raspberry pi screen (res : 800px / 480px ) */
340 @media all and (max-width: 800px) and (max-height: 480px) and (min-width: 799px){
341
342     /* Don't show cursor*/
343     * {
344         cursor: none;
345     }
346
347     body {
348         overflow: overlay;
349     }
350
351     .header a {
352         margin-right: 20px;
353     }
354
355     /* Custom scroll bar */
356
357     ::-webkit-scrollbar {
358         width: 25px;
359     }
360
361     ::-webkit-scrollbar-thumb {
362         background-color: #bbb;
363         background-clip: padding-box;
364         border: 0.05em solid #eee;
365         border-radius: 20px;

```

```
366     }
367
368     ::-webkit-scrollbar-track {
369         background-color: transparent;
370     }
371
372     ::-webkit-scrollbar-track-piece {
373         background: transparent;
374         margin-top: 10px;
375         margin-bottom: 10px;
376     }
377
378     /* Buttons */
379     ::-webkit-scrollbar-button:single-button {
380         background-color: transparent;
381         display: block;
382         border-style: solid;
383         height: 20px;
384         width: 20px;
385     }
386
387     /* Up */
388     ::-webkit-scrollbar-button:single-button:vertical:decrement {
389         border-width: 0px 12px 16px 12px;
390         border-color: transparent transparent #bbb transparent;
391     }
392
393     ::-webkit-scrollbar-button:single-button:vertical:decrement:hover {
394         border-color: transparent transparent #999 transparent;
395     }
396
397     /* Down */
398     ::-webkit-scrollbar-button:single-button:vertical:increment {
399         border-width: 16px 12px 0 12px;
400         border-color: #bbb transparent transparent transparent;
401     }
402
403     ::-webkit-scrollbar-button:vertical:single-button:increment:hover {
404         border-color: #999 transparent transparent transparent;
405     }
406 }
```

```

1  <html>
2  <head>
3      <meta name="viewport" content="width=device-width, initial-scale=1.0">
4      <meta charset="UTF-8">
5      <meta name="theme-color" content="black">
6      <meta name="author" content="Robin Forestier">
7      <meta name="description" content="Login2 - ">
8      <title>Login 2</title>
9      <link rel="stylesheet" href="../static/login2.css/">
10     <link rel="icon" type="image/x-icon" href="/static/img/LogoElo.ico">
11     <link rel="shortcut icon" href="/static/img/LogoElo.ico" type="image/x-icon">
12 </head>
13 <body>
14     <!-- Create the header -->
15     <div class="header">
16         <p><a href="/page1">Retour</a></p>
17         <p>{{ time }} </p>
18         <p class="warning">{{ warning }} </p>
19         <h1>Login</h1>
20     </div>
21
22     <div class="log">
23         <!-- Title -->
24         <h2>Mot de passe</h2>
25
26         <!-- Create the password -->
27         <div class="mdp">
28             <p>
29                 {% if mdp|length == 0 %}
30                 <!-- Password is empty -->
31                 {{ " o o o " }}
32                 {% endif %}
33                 {% for m in mdp %}
34                 <!-- Show the number of characters by * -->
35                 {{ "*" }}
36                 {% endfor %}
37             </p>
38         </div>
39
40         <!-- Create the keyboard -->
41         <form method="post">
42             <div class="clavier">
43                 <button type="submit" class="btn" name="btn" value="1"> <span
44                     class="btn_num">1</span> </button>
45                 <button type="submit" class="btn" name="btn" value="2"> <span
46                     class="btn_num">2</span> </button>
47                 <button type="submit" class="btn" name="btn" value="3"> <span
48                     class="btn_num">3</span> </button>
49                 <button type="submit" class="btn" name="btn" value="4"> <span
50                     class="btn_num">4</span> </button>
51                 <button type="submit" class="btn" name="btn" value="5"> <span
52                     class="btn_num">5</span> </button>
53                 <button type="submit" class="btn" name="btn" value="6"> <span
54                     class="btn_num">6</span> </button>
55                 <button type="submit" class="btn" name="btn" value="7"> <span
56                     class="btn_num">7</span> </button>
57                 <button type="submit" class="btn" name="btn" value="8"> <span
58                     class="btn_num">8</span> </button>
59                 <button type="submit" class="btn" name="btn" value="9"> <span
60                     class="btn_num">9</span> </button>
61                 <button type="submit" class="btn" name="btn" value="10"> <span
62                     class="btn_num">Effacer</span> </button>
63                 <button type="submit" class="btn" name="btn" value="0"> <span
64                     class="btn_num">0</span> </button>
65                 <button type="submit" class="btn" name="btn" value="11"> <span
66                     class="btn_num">Submit</span> </button>
67             </div>
68         </form>
69     </div>
70 </body>
71 </html>

```



```

1  /* login2.css | Robin Forestier
2  *
3  * Used for login2.html on ../templates/login2.html
4  */
5
6
7  /* This is the CSS code for the background of the page. */
8  html,
9  body {
10     height: 100%;
11     width: 100%;
12     height: 100vh;
13     margin: 0;
14     padding: 0;
15     background: linear-gradient(to bottom right, #333 0%, #777 100%);
16     background-attachment: fixed;
17     box-sizing: border-box;
18 }
19
20 /* Header */
21 .header {
22     float: left;
23     width: 100%;
24     height: auto;
25     line-height: 25px;
26     background: #333;
27     color: white;
28     vertical-align: middle;
29     z-index: 100;
30     position: relative;
31     margin-bottom: 15px;
32 }
33
34 .header h1 {
35     margin-left: 10px;
36 }
37
38 .header p {
39     float: right;
40     font-size: 20px;
41     margin-right: 20px;
42 }
43
44 .header a {
45     color: #fff;
46     text-decoration: none;
47     padding: 10px;
48     background-color: #777;
49     border-radius: 10px;
50 }
51
52 .warning {
53     color: #F00;
54 }
55
56 /* Keyboard Position */
57
58 .log {
59     position: absolute;
60     /* Centrage */
61     bottom: 20px;
62     left: 50%;
63     width: 400px;
64     transform: translate(-50%, 0);
65     /* background transparent mais plus foncé */
66     background: rgba(0,0,0,.2);
67     /* BOX */
68     box-sizing: border-box;
69     box-shadow: 0 15px 25px rgba(0,0,0,.6);
70     padding: 20px;
71     border-radius: 15px;
72 }
73

```

```

74  /* Title Login */
75  h2 {
76      color: #FFF;
77      font-size: 30pt;
78      margin: 0;
79      padding: 0;
80      margin-bottom: 10px;
81      text-align: center;
82  }
83
84  .log .user {
85      position: relative;
86  }
87
88  /* Password preview */
89  .mdp {
90      width: 100%;
91      text-align: center;
92      font-size: 25px;
93      color: #fff;
94  }
95
96  .mdp>p{
97      margin: 15px;
98  }
99
100 form {
101     margin: 0;
102 }
103
104 /* Keyboard display */
105 .clavier {
106     display: grid;
107     grid-template-columns: repeat(3, 1fr);
108     gap: 10px;
109     width: 100%;
110 }
111
112 /* Custom buttons */
113 .btn {
114     padding: 5px;
115     text-align: center;
116     height: 50px;
117     border-radius: 20px;
118     background: transparent;
119     box-shadow: 0 1px 5px rgba(0, 0, 0, .4);
120     background-image: linear-gradient(
121         rgba(90, 90, 90, 1) 0%,
122         rgba(90, 90, 90, 0.3) 25%,
123         rgba(90, 90, 90, 0.15) 50%,
124         rgba(90, 90, 90, 0.075) 65%,
125         rgba(90, 90, 90, 0.037) 80%,
126         rgba(90, 90, 90, 0.019) 88%,
127         rgba(90, 90, 90, 0) 100%);
128 }
129
130 button>span {
131     color: #ddd;
132     font-size: 20px;
133 }
134
135 /* Only for raspberry pi screen (res : 800px / 480px ) */
136 @media all and (max-width: 800px) and (max-height: 480px) and (min-width: 799px){
137     /* Don't show cursor */
138     * {
139         cursor: none;
140     }
141 }

```