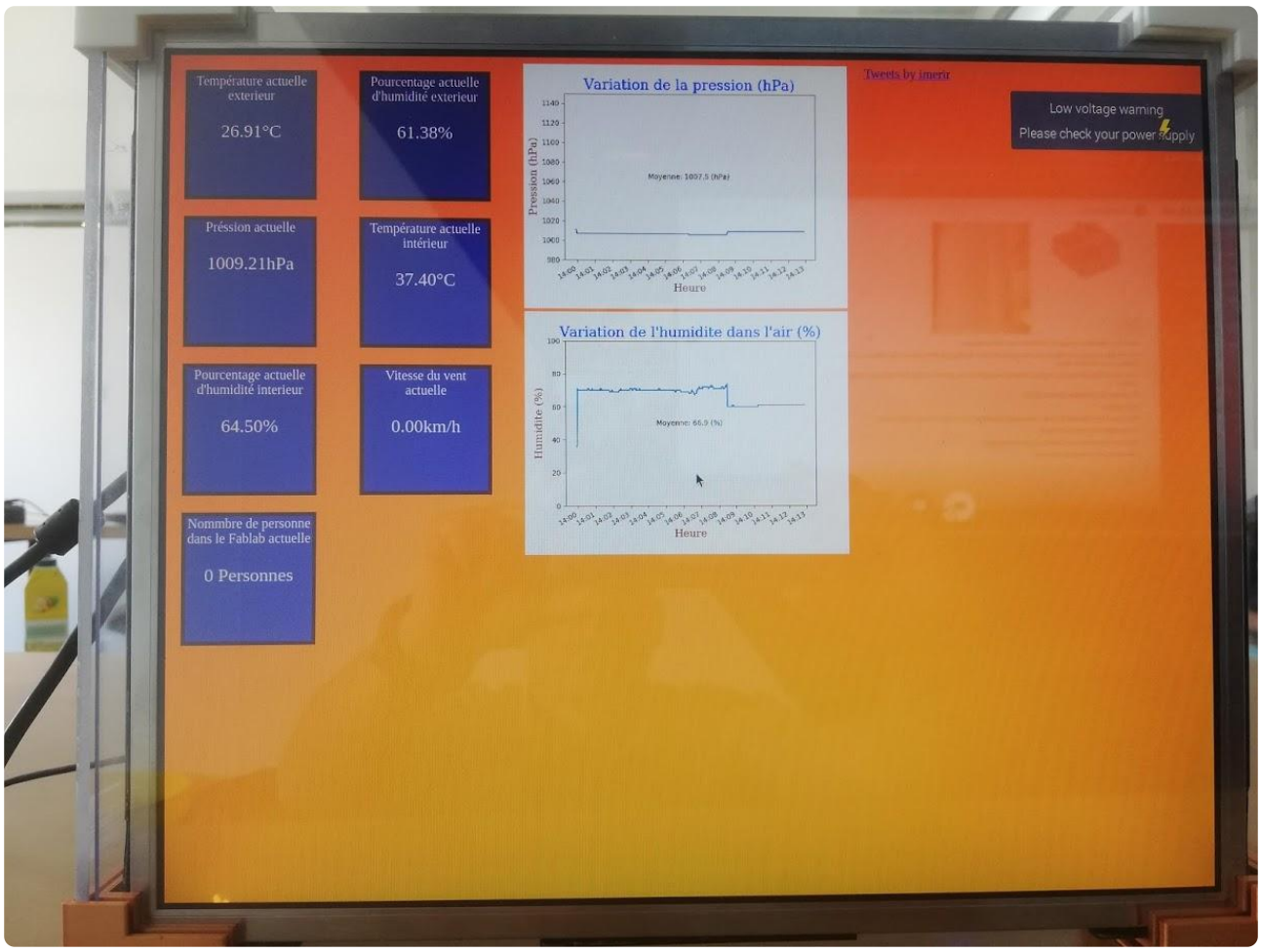


écran

Auteur : Mikel Viale · **Publié le** 04/06/2021 · 3 vues · 4 téléchargements PDF

Le projet consiste à mettre en place un écran pour afficher des données venant d’autres périphériques (thermomètre, station météo, etc..), le flux twitter de l’IMERIR ainsi qu’un comptage du nombre de personnes au sein du FabLab.

Nous utiliserons un Raspberry Pi 3 pour traiter et afficher ces données.

Lien du GitLab : https://gitlab.imerir.com/vini.praion/screen-project

Étapes du projet

ÉTAPE 1

Démontage et conception du nouveau châssis de l'écran

Après avoir récupéré un moniteur d'ordinateur, nous l'avons démonté.

Une fois débarrassés de son châssis, nous avons poncé sa partie métallique afin de la peindre, cette plaque nous servira de support pour notre électronique.

Nous avons ensuite prototypé un nouveau châssis pour accueillir l'écran, qui sera composé des 2 plaques en plexiglass et 2 équerres.

On peut voir sur les images situées plus haut :

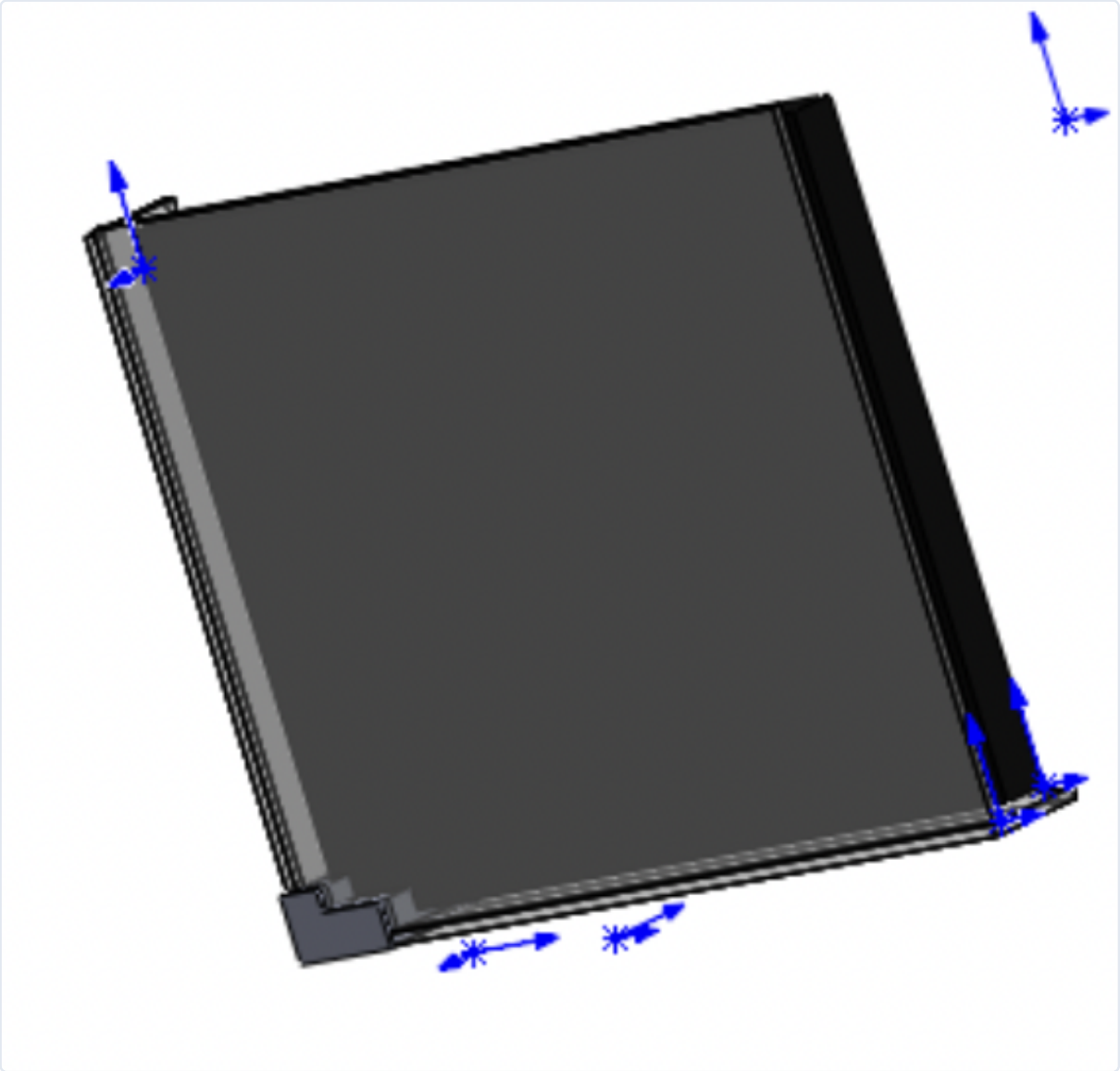
1) Le moniteur

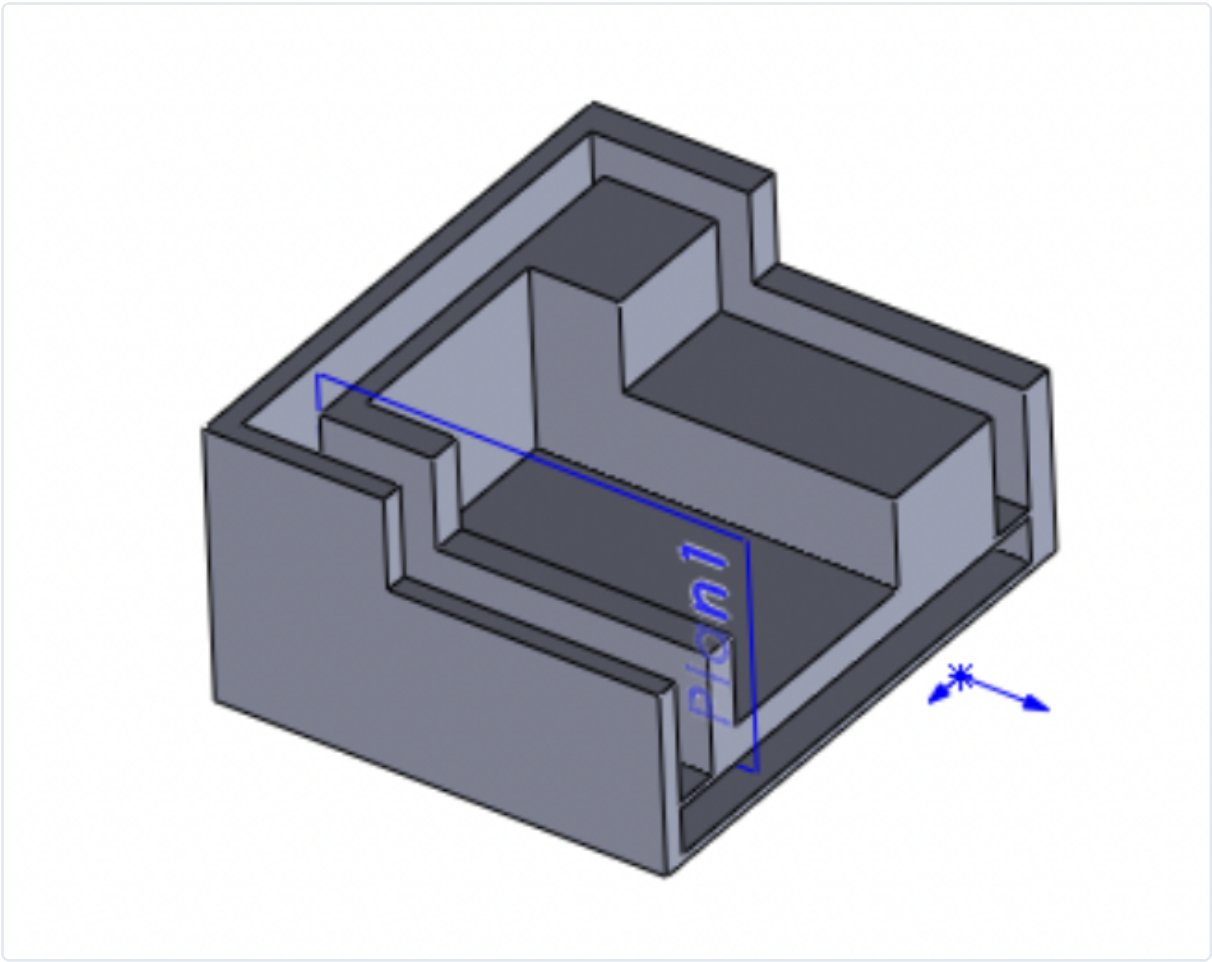
2) Le support où seront fixés les parties électroniques (Raspberry Pi 3, interface vidéo avec alimentation..)

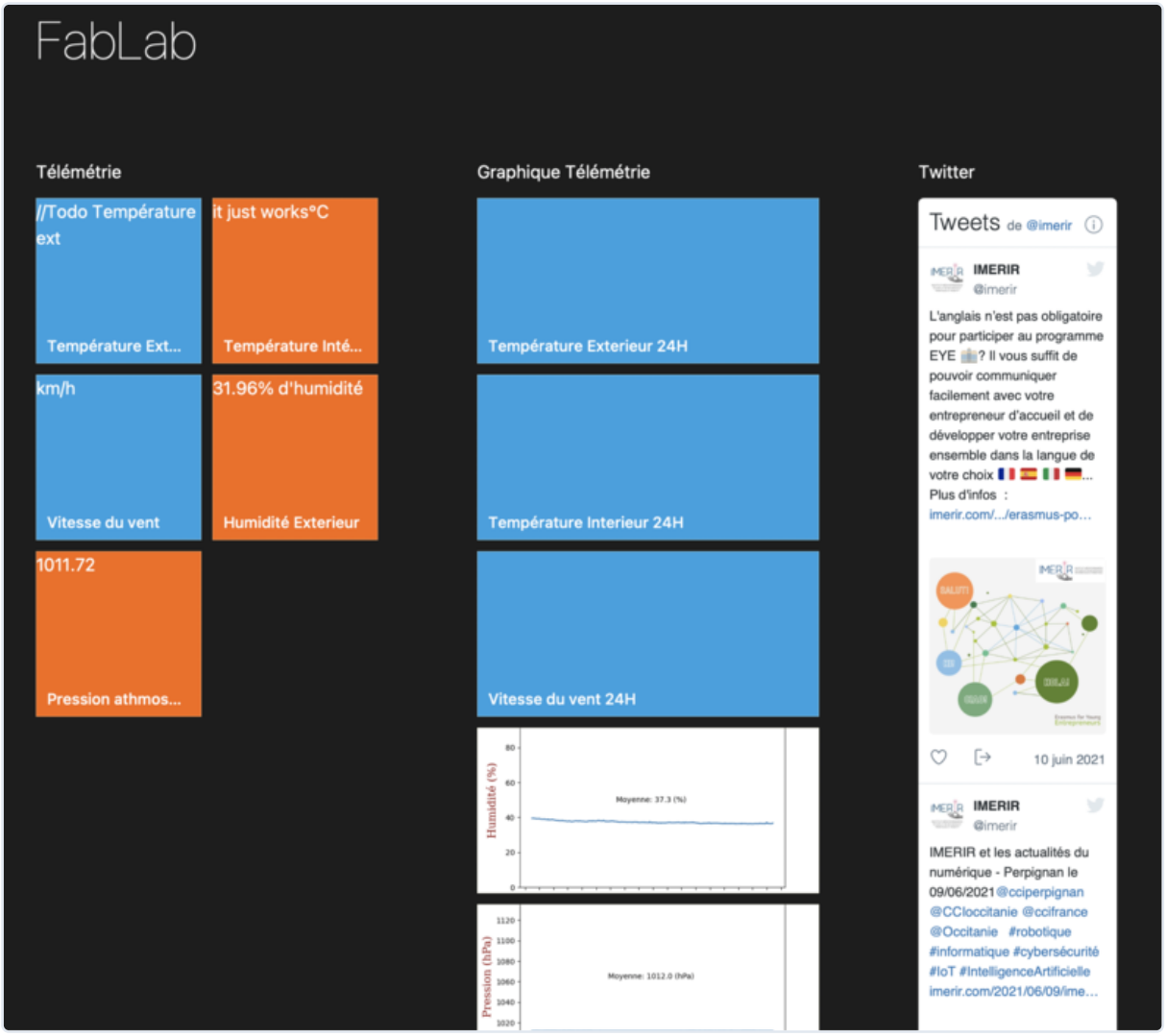
3) Conception du châssis sur Solidworks

4) Equerre qui maintiendra en position 2 plaques de plexiglass

5) Le résultat de la conception Solidworks




Plan1


FabLab
Télémétrie
//Todo Température ext
Température Ext...
it just works°C
Température Inté...
km/h
Vitesse du vent
31.96% d'humidité
Humidité Exterieur
1011.72
Pression athmos...
Graphique Télémétrie
Température Exterieur 24H
Température Interieur 24H
Vitesse du vent 24H
Humidité (%)
Moyenne: 37.3 (%)
Pression (hPa)
Moyenne: 1012.0 (hPa)
Twitter
Tweets de @imerir
IMERIR
@imerir
L'anglais n'est pas obligatoire pour participer au programme EYE ? Il vous suffit de pouvoir communiquer facilement avec votre entrepreneur d'accueil et de développer votre entreprise ensemble dans la langue de votre choix ...
Plus d'infos :
imerir.com/.../erasmus-po...
10 juin 2021
IMERIR
@imerir
IMERIR et les actualités du numérique - Perpignan le 09/06/2021@cciperpignan @CCIoccitanie @ccifrance @Occitanie #robotique #informatique #cybersécurité #IoT #IntelligenceArtificielle
imerir.com/2021/06/09/ime...

ÉTAPE 2

Initialisation du Raspberry Pi 3

Dès que le Raspberry Pi 3 est allumé, il va exécuter son script de communication MQTT que nous allons voir plus bas, ainsi qu'un script toutes les 60 secondes qui va générer des graphiques à l'aide des données récupérées via cette communication MQTT.

```
import logging
import threading
import time
import schedule
import os
import mosquittosub

def mosqui():
mosquittosub.run()

def rune():
while True:
os.system('sudo python /home/pi/Mosquitto/hydrograph.py')
print("Run Hydro Graph")
os.system('sudo python /home/pi/Mosquitto/pressiongraph.py')
print("Run Pression Graph")
time.sleep(60)

threading.Thread(target=rune).start()
mosqui()
```

ÉTAPE 3

Communication par protocole MQTT

Afin d'intéragir avec les autres périphériques (station météo, thermomètre, etc..) de façon sans-fil, nous utilisons le protocole MQTT (Message Queuing Telemetry Transport). C'est un protocole réseau conçu pour les connexions avec une bande passante limitée, et le choix ici est justifié vu le peu d'informations que nous souhaitons récupérer (valeur du capteur et date/heure).

Nous stockons ensuite les valeurs récupérées au sein d'un fichier CSV (Coma Separated Values).

```
# coding: utf-8
""" Souscription au topic "fablab/#" sur le broker Eclipse Mosquitto.
Utilise une autenthification login/mot-de-passe sur le broker
"""
import paho.mqtt.client as mqtt_client
from datetime import datetime
import time

# Configuration
MQTT_BROKER = "IP"
MQTT_PORT = "port"
KEEP_ALIVE = 20 # intervale en seconde

def on_log( client, userdata, level, buf ):
print( "log: ",buf)

def on_connect( client, userdata, flags, rc ):
print( "Connexion: code retour = %d" % rc )
print( "Connexion: Statut = %s" % ("OK" if rc==0 else "échec") )



def on_message( client, userdata, message ):
print( "Reception message MQTT..." )
print( "Topic : %s" % message.topic )
print( "Data : %s" % message.payload )
now = datetime.now() # current date and time
ts = time.time()
date_time = now.strftime("%Y/%m/%d, %H:%M:%S")
date_time_csv = now.strftime("%Y%m%d%H%M%S")
```

```
file1 = open('/var/www/html/data/data-' + message.topic.replace('/', '-') +
'.txt',"a")
file1.write( 'date: ' + str(ts) + '{ ' + 'topic: ' + message.topic + ' , ' + 'message: '
+ message.payload + ' } \n' )
file1.close()
file2 = open('/var/www/html/data-csv/data-' + message.topic.replace('/', '-') +
'.csv', "a")
file2.write(str(ts) + ',' + message.payload + '\n')
file2.close()

def run():
# Client(client_id=””, clean_session=True, userdata=None, protocol=MQTTv311,
transport=”tcp”)
client = mqtt_client.Client( client_id="client007" )

# Assignation des fonctions de rappel
client.on_message = on_message
client.on_connect = on_connect
#client.on_log = on_log

# Connexion broker
client.connect( host=MQTT_BROKER, port=MQTT_PORT, keepalive=KEEP_ALIVE )
client.subscribe( "fablab/#" )

# Envoi des messages
client.loop_forever()
```

ÉTAPE 4

Génération de graphiques

Nous souhaitons mettre en place des graphiques qui montrent l'évolution des valeurs pour chaque capteur, afin de suivre un historique sur une certaine intervalle, plutôt qu'avoir uniquement à disposition la valeur du capteur sur l'instant T.

Pour se faire, nous générons un fichier image à l'aide de matplotlib. Nous lui donnons les dates sur l'axes des abscisses, et les valeurs sur l'axe des ordonnées.

Voici un exemple pour la génération du graphique de taux d'humidité dans l'air.

```
# coding: utf-8
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter, FixedLocator, MultipleLocator
import matplotlib.dates as mdates
import numpy as np
import datetime as dt
from datetime import datetime
import csv
import time
from statistics import mean

path = '/var/www/html/'
#path = ''

font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

x = []
y = []

with open(path + 'data-csv/data-fablab-weather-hydro.csv','r') as file:
graph = file.read().replace("\0","")
contents_split = graph.splitlines()
for i in range(len(contents_split)):
xety = contents_split[i].split(",")
x.append(int(float(xety[0])))
```

```
y.append(int(round(float(xety[1]),1)))
timestamp = time.time()
moyenne = mean(y)
now = datetime.fromtimestamp(timestamp)
then = now + dt.timedelta(seconds=len(x))
hours = mdates.drange(now,then,dt.timedelta(seconds=1))
plt.show(block=True)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
plt.gca().xaxis.set_major_locator(mdates.SecondLocator(interval=60))
plt.ylim([0, 100])
plt.plot(hours[:len(y)],y)
plt.gcf().autofmt_xdate()
plt.ylabel('Humidite (%)', fontdict = font2)
plt.xlabel('Heure', fontdict = font2)
plt.title('Variation de l\'humidite dans l\'air (%)', fontdict = font1)
plt.text(0.5,0.5,'Moyenne: ' + str(round(moyenne, 1)) + " (%)
",horizontalalignment='center',
verticalalignment='center', transform = plt.gca().transAxes)
plt.savefig(path + 'graph/hydro-graph.png')
print("Ok SAVE HYDRO")
```

ÉTAPE 5

Développement de l'interface graphique

Finalement, pour mettre en forme ces graphiques et ces valeurs, nous le faisons via un navigateur internet. La mise en place d'un serveur web a été indispensable, car par HTML/Javascript, pas possible de lire localement des fichiers. On a alors opté pour du PHP.

Pour la mise en page, nous avons utilisé "metroui" qui reprend l'interface des tuiles de Windows 8/Windows 10, des exemples sont disponibles à l'adresse suivante : https://metroui.org.ua

Cette page s'actualisera d'elle même toutes les minutes, afin de mettre à jour les dernières données récupérées.

```
<?php
function csvLastValue($path, $unitType)
{
$dateTimeFormat = 'Y-m-d H:i:s';
$handle = fopen($path, "r");
$data = fgetcsv($handle);
$value = $data[count($data) - 1];
echo ($value . $unitType);
fclose($handle);
};
?>

<!DOCTYPE html>
<html lang="fr">

<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no">

<meta name="twitter:site" content="@metroui">
<meta name="twitter:creator" content="@pimenov_sergey">
<meta name="twitter:card" content="summary">
<meta name="twitter:title" content="Metro 4 Components Library">
```

<meta name="twitter:description" content="Metro 4 is an open source toolkit for developing with HTML,
CSS, and JS. Quickly prototype your ideas or build your entire app with responsive grid system,
extensive prebuilt components, and powerful plugins.">
<meta name="twitter:image" content="https://metroui.org.ua/images/m4-logo-social.png">

<meta property="og:url" content="https://metroui.org.ua/index.html">
<meta property="og:title" content="Metro 4 Components Library">
<meta property="og:description" content="Metro 4 is an open source toolkit for developing with HTML, CSS, and JS.
Quickly prototype your ideas or build your entire app with responsive grid system,
extensive prebuilt components, and powerful plugins .">
<meta property="og:type" content="website">
<meta property="og:image" content="https://metroui.org.ua/images/m4-logo-social.png">
<meta property="og:image:secure_url" content="https://metroui.org.ua/images/m4-logo-social.png">
<meta property="og:image:type" content="image/png">
<meta property="og:image:width" content="968">
<meta property="og:image:height" content="504">

<meta name="author" content="Sergey Pimenov">
<meta name="description" content="The most popular HTML, CSS, and JS library in Metro style.">
<meta name="keywords" content="HTML, CSS, JS, Metro, CSS3, Javascript, HTML5, UI, Library,
Web, Development, Framework">

<link href="https://cdn.metroui.org.ua/v4/css/metro-all.min.css?ver=@@b-version" rel="stylesheet">
<link href="start.css" rel="stylesheet">

<title>FABLAB IMERIR</title>
</head>

<body class="bg-dark fg-white h-vh-100 m4-cloak">

```
<div class="container-fluid start-screen h-100">
<h1 class="start-screen-title">FabLab</h1>

<div class="tiles-area clear">
<div class="tiles-grid tiles-group size-2 fg-white" data-group-title="Télémétrie">
<div data-role="tile" class="bg-cyan fg-white" data-size="medium">
<span>//Todo Température ext</span>
<span class="branding-bar">Température Ext</span>
</div>
<div data-role="tile" class="bg-orange fg-white" data-size="medium">
<span>
<?php
csvLastValue("data-csv/data-fablab-thermo-int.csv", "°C");
?>
</span>
<span class="branding-bar">Température Int</span>
</div>
<div data-role="tile" class="bg-cyan fg-white" data-size="medium">
<span>
<?php
csvLastValue("data-csv/data-fablab-wind.csv", "km/h");
?>
</span>
<span class="branding-bar">Vitesse du vent</span>
</div>
<div data-role="tile" class="bg-orange fg-white" data-size="medium">
<span>
<?php
csvLastValue("data-csv/data-fablab-weather-hydro.csv", "% d'humidité");
?>
</span>
<span class="branding-bar">Humidité Ext</span>
</div>
<div data-role="tile" class="bg-orange fg-white" data-size="medium">
<span>
<?php
csvLastValue("data-csv/data-fablab-weather-pression.csv", "hPa");
?>
</span>
<span class="branding-bar">Pression Atmo</span>
```

```
</div>
<div data-role="tile" class="bg-orange fg-white" data-size="medium">
<span>
<?php
//csvLastValue("data-csv/data-fablab-weather-pression.csv", "hPa");
?>
</span>
<span class="branding-bar">Personnes FabLab</span>
</div>

</div>

<div class="tiles-grid tiles-group size-2 fg-white" data-group-title="Graphiques Télémétrie">
<a data-role="tile" data-cover="graph/thermo-ext.png" data-size="wide" href="graph/thermo-ext.png">
<span class="branding-bar">Température Extérieure 24H</span>
</a>
<a data-role="tile" data-cover="graph/thermo-int.png" data-size="wide" href="graph/thermo-int.png">
<span class="branding-bar">Température Intérieure 24H</span>
</a>
<a data-role="tile" data-cover="graph/wind-graph.png" data-size="wide" href="graph/wind-graph.png">
<span class="branding-bar">Vitesse du vent 24H</span>
</a>
<a data-role="tile" data-cover="graph/hydro-graph.png" data-size="wide" href="graph/hydro-graph.png">
<span class="branding-bar">Taux d'humidité</span>
</span>
</a>
<a data-role="tile" data-cover="graph/pression-graph.png" data-size="wide" href="graph/pression-graph.png">
<span class="branding-bar">Pression </span>
</span>
</a>
<a data-role="tile" data-cover="graph/personnes-graph.png" data-size="wide" href="graph/personnes-graph.png">
<span class="branding-bar">Personnes au sein du FabLab</span>
</a>
```

```
</div>

<div class="tiles-grid tiles-group size-2 fg-white" data-group-title="Twitter"
data-size="wide">
<a class="twitter-timeline" data-lang="fr" data-width="500" data-height="800"
data-theme="light" href="https://twitter.com/imerir?
ref_src=twsrc%5Etfw">Tweets by imerir</a>
<script async src="https://platform.twitter.com/widgets.js" charset="utf-8"></
script>
</div>
</div>
</div>


<script src="https://cdn.metroui.org.ua/v4/js/metro.min.js"></script>
<script src="start.js"></script>
<script>setInterval(function(){window.location.reload()}, 60000);</script>
</body>

</html>
```