

**SESSION 2017 - 2018**



CITÉ SCOLAIRE

**ÉMILE ZOLA**

# **BACCALAURÉAT** **TECHNOLOGIQUE**

**SÉRIE :** The logo for the STI 2D SIN series, featuring a stylized green and grey sphere, the text 'STI 2D', and a purple square with 'SIN'.

**ÉPREUVE DE PROJET EN ENSEIGNEMENT  
SPÉCIFIQUE À LA SPÉCIALITÉ**

**AUTOBLIND**

**THÉO GICQUEL - TSTI2D**

**PROFESSEURS RÉFÉRENTS : M.GASTON MME.CELLARD**

# SOMMAIRE

## Travaux communs

### Introduction

- Présentation du projet.....P1

### Conception

- Phase de préconception.....P1
- Conception préliminaire.....P1-4
- Choix du matériel.....P5
- Choix des cartes.....P6
- Choix des capteurs.....P7
- Améliorations thermiques.....P7-8
- Amélioration du confort.....P8-9

### Réalisation du projet

- Répartition des tâches.....P9
- Communication.....P9

## Travaux Personnels

### Réalisation du projet

- Présentation générale.....P10
- Interprétation des données séries.....P11
- Préparation du serveur web.....P12
- Affichage des variables.....P13
- Modification des ordres.....P13-14
- Transmission des ordres.....P14-15
- Développement de l'interface utilisateur.....P16-17
- Conclusion.....P18

### Annexes

- Arborescence des fichiers serveur.....P19
- Diagramme du réseau local.....P19
- Logiciels et outils utilisés.....P20
- Remerciements.....P20

## I Introduction

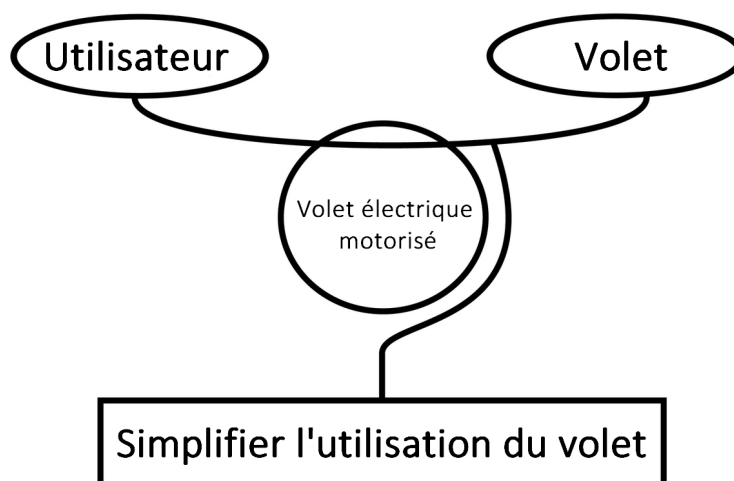
Un volet possède pour premier objectif de limiter la luminosité intérieure, nous avons donc décidé d'élargir les compétences d'un volet motorisé afin d'en limiter les pertes thermiques et énergétiques. De plus, le volet a été conçu dans le but d'optimiser le confort de l'utilisateur. Le volet peut très bien être utilisé en entreprise, chez un particulier ou dans un bâtiment public.

## II Conception

Lors de la conception du projet, nous avons pensé à un panneau solaire afin de l'alimenter de façon autonome. Pour commander le volet nous avons décidé qu'un programme s'occuperait de le contrôler de façon automatique ainsi qu'une application Android. Pour l'isolation, nous avons choisi d'appliquer de la mousse expansive dans les lames du volet. Nous avons aussi jugé qu'il était nécessaire d'afficher les données récoltées pour contrôler automatiquement le volet, soit la température intérieure, extérieure, la luminosité et la réserve de la batterie. Nous avons, après discussion, choisi d'effectuer un volet motorisé vertical et non pour fenêtre de toit. Ensuite, nous avons jugé qu'il était nécessaire d'ajouter un site internet avec version mobile et ordinateur pour contrôler le volet, ainsi qu'un interrupteur et une télécommande infrarouge.

### 1) Conception préliminaire

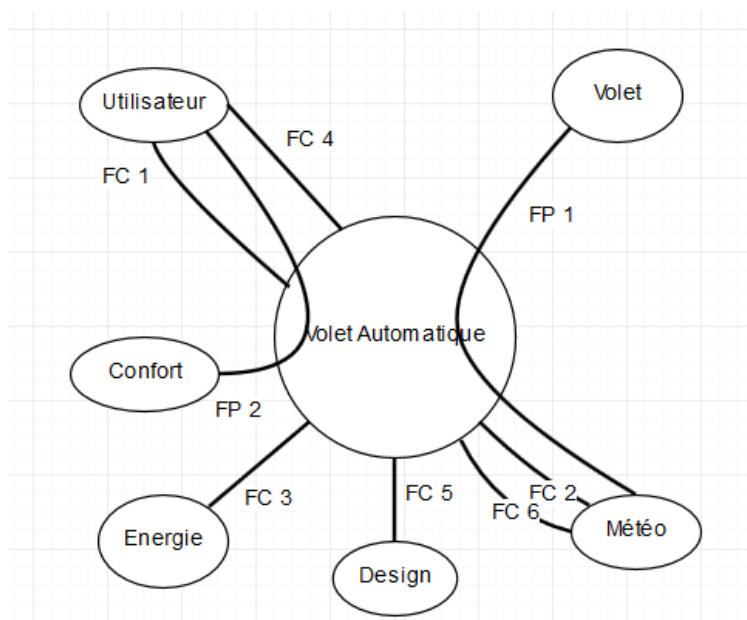
Pour concevoir notre projet nous avons dans un premier temps effectué la méthode APTE, nous avons donc réalisé une « Bête à cornes ».



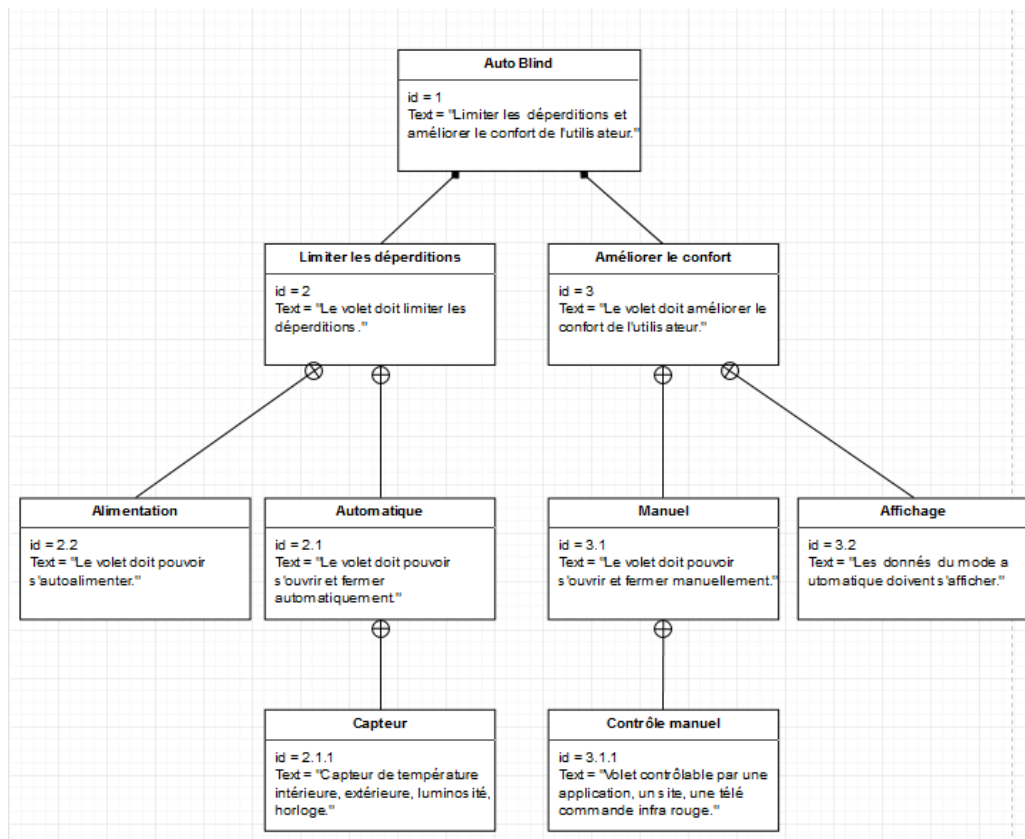
Nous avons ensuite effectué le cahier des charges fonctionnelles, où nous avons listé chaque fonction classée par un ordre de priorité.

REP	FONCTIONS	CRITERES	NIVEAUX	FLEXIBILITE
FP 1	Limiter les pertes énegétiques	Pertes thermiques	Faible	F1
		Dépenses électriques	Faible	F1
FP 2	Améliorer le confort	Affichage	Lisible +/- 0,5m	F1
		Automatisation		F1
FC 1	Contrôler le volet manuellement	Pouvoir s'ouvrir et se fermer		F1
FC 2	Contrôler le volet automatiquement	Pouvoir s'ouvrir et se fermer		F2
FC 3	Assurer sa propre autonomie	Alimentation électrique	Autonome	F2
FC 4	Être contrôlable a distance	Site Internet	Distance Réseau local	F2
		Application	Distance Réseau local	F2
		Télécommande infra rouge	Distance Pièce	F2
FC 5	Être esthétiquement correct			F3
FC 6	Résister aux aléas climatiques			F3
FC 7	Capter les valeurs nécessaire au mode automatique	Capteur de température extérieur	+/- 3°C	F1
		Capteur de température intérieur	+/- 3°C	F1
		Capteur de luminosité	+/- 100lm	F2
		Horloge	+/- 10min	F2

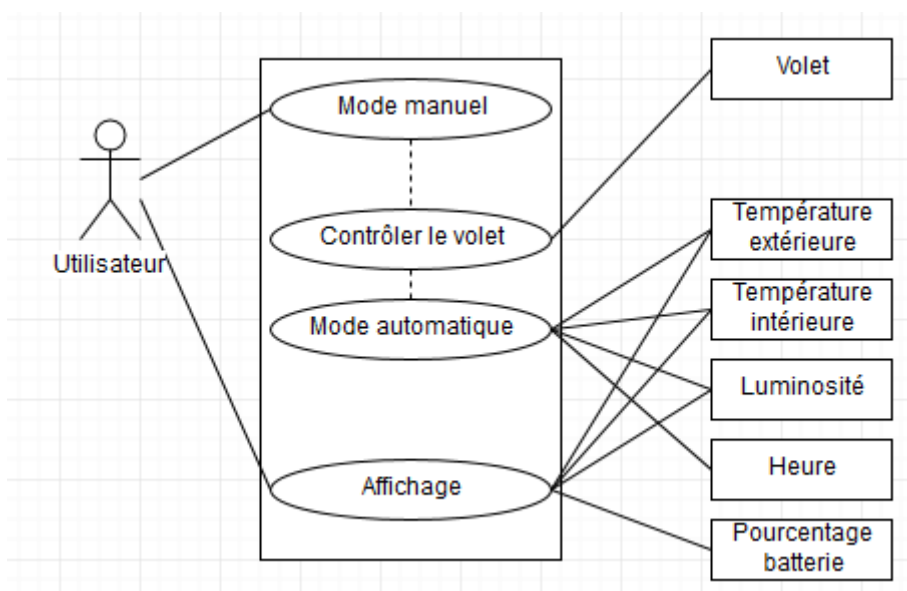
À partir du cahier des charges nous avons effectué le diagramme pieuvre suivant pour comprendre les liaisons entre chaque élément.



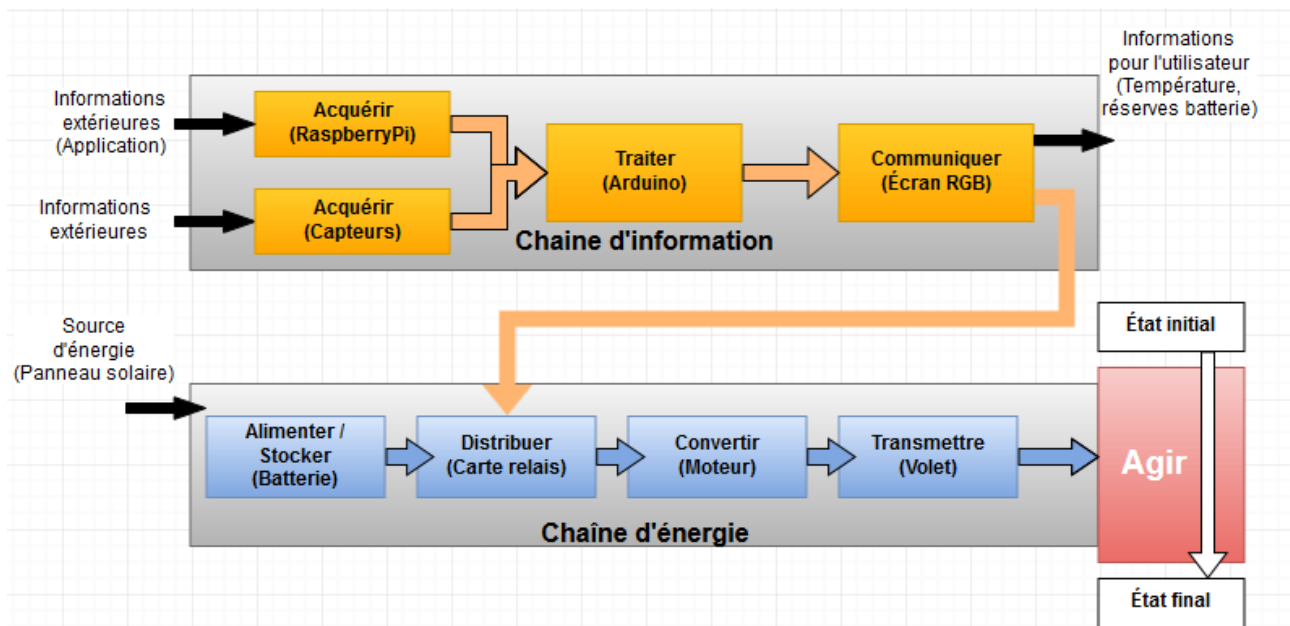
Nous avons ensuite réalisé un diagramme d'exigence (SysML) :



Ce qui nous permet donc de réaliser le diagramme des cas d'utilisation (SysML) :

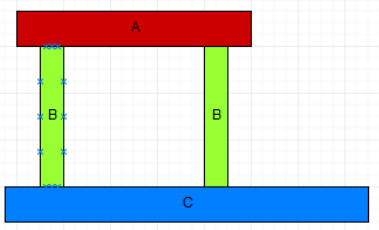


Maintenant que nous connaissons les objectifs du projet et les interactions extérieures, nous avons réalisé une chaîne d'énergie et une chaîne d'information pour connaître le fonctionnement interne du projet.



## 2) Choix du matériel

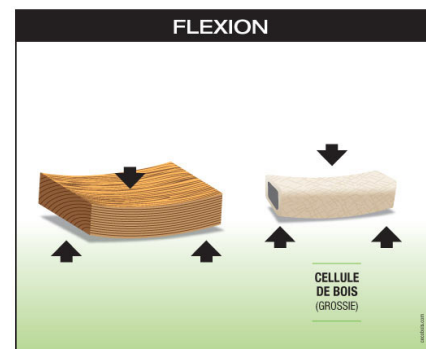
Représentation simplifiée de la maquette :



Nous avons choisi d'utiliser des tablettes en pin pour réaliser le prototype, les propriétés du bois étant intéressant dans notre cas :

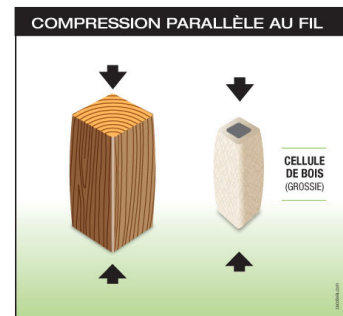
### a) Flexion

La planche sur la partie supérieure (A) ainsi que sur la partie inférieure (C) devra résister à certains efforts de flexions et ne pas se briser.



### b) Compression

Les piliers (B) maintenant la partie supérieure sur la partie inférieure devront résister à certains efforts de compression (et de flexion) et ainsi rester stable.



Nous avons choisi des équerres renforcées avec nervure afin de fixer les piliers (B), et les différents éléments entre eux (C).



Nous avons choisi des équerres de fixation destinées à la base pour des étagères, les équerres devront tenir la planche supérieure (A) aux piliers (B), de plus les équerres peuvent supporter une charge de 300 kilogrammes garantie par le constructeur, ce qui permet de tenir sans difficulté les différents éléments sur la planche A.

### 3) Choix des cartes

Modèle	Arduino Uno	Arduino Mega	Arduino Leonardo	Arduino Micro
Microcontrôleur	ATmega328	Atmega2560	ATmega32u4	ATmega32u4
Fréquence de calcul	16MHz	16MHZ	16MHz	16MHz
Mémoire Flash	256KB	256KB	32KB	32KB
Nb Broches Numérique	14	54	20	20
Nb Broches entrées analogiques	6	16	12	12
Compatibilité Shield Grove Mega	Non	Oui	Non	Non

Nous avons choisi d'utiliser une Arduino MEGA, car d'une part, ce modèle dispose du plus grand nombre de broches par rapport aux autres modèles parce qu'il s'agit de la seule carte compatible avec le shield grove mega qui permet de connecter et d'utiliser un grand nombre de capteurs grove très facilement.

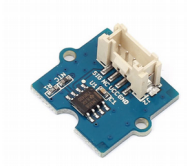
Modèle	PI Model A	PI Model B	PI2 Model B	PI3 Model B
Processeur	700MHz Monocore	700MHz Monocore	900MHz Qadricore	1200MHz Quadricore
Mémoire Vive	256Mb	512Mb	1000Mb	1000Mb
Nb Ports USB	1	2	4	4
Sans-Fil	Non	Non	Non	Wifi & Bluetooth
Ethernet RJ45	Non	Oui	Oui	Oui
Nb Broches GPIO	26	26	40	40

Pour le choix du modèle de Raspberry PI, nous nous sommes orientés vers le modèle PI3 model B, car celui-ci possédait le processeur le plus performant, ce qui permet de gérer un maximum de requêtes HTTP.

De plus, le modèle 3B est le seul équipé d'une interface réseau sans fil, ce qui permet de placer le dispositif du volet n'importe où dans une habitation sans avoir besoin de le relier via un câble ethernet.



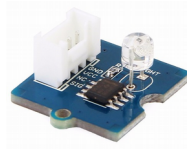
#### 4) Choix des capteurs



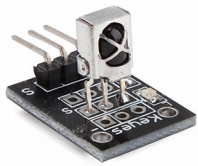
Les capteurs de température sont des Grove Temperature Sensor V1.2 utilise une thermistance et délivre une valeur analogique.



Notre capteur de tension, Phidgets Precision Voltage Sensor 1135, mesure le différentiel et délivre une valeur analogique.



Le capteur de luminosité Grove Light Sensor v1.2 relève la luminosité avec une photorésistance, il délivre une valeur analogique.

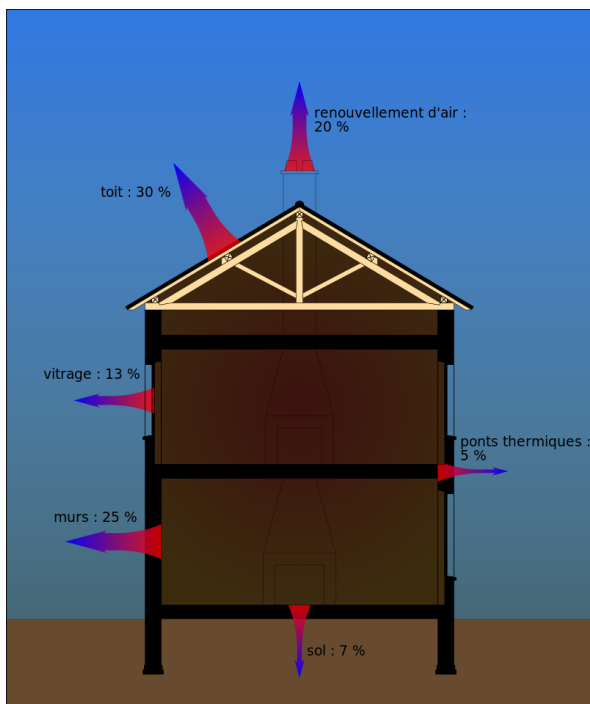


Notre récepteur infrarouge, FontaineRs 10Pcs KY-022 KY022 délivre une valeur numérique.

#### 5) Améliorations thermiques

##### a) Déperdition thermique

La déperdition thermique est la perte d'énergie thermique par les parois d'un bâtiment et ses échanges de fluide avec l'extérieur, la chaleur se propage d'un milieu chaud vers un milieu froid.



Le schéma ci-contre représente toutes les déperditions thermiques dans un bâtiment, nous pouvons donc remarquer que la plupart des déperditions, ou pertes thermiques, ont lieu au niveau des murs, du toit et de la cheminée ce qui représente à eux seul 75 % de pertes thermiques.

Sur cette photo, nous remarquons en rouge les pertes de chaleur, nous remarquons donc que le toit, la cheminée et les murs sont correctement isolés, ce qui nous retire 75 % de pertes thermiques, il ne reste donc sur l'image que les ponts thermiques (pouvant être évité avec une isolation extérieure) et fenêtres.

Les lames du volet sont remplies d'une mousse expansive permettant une isolation thermique et acoustique (dans notre cas une Rubson Mousse expansive Power) afin d'éviter tout transfert thermiques directement sur la vitre, le volet ayant donc un rôle d'isolant.



Analyse thermographique d'un bâtiment relativement bien isolé.

[Projetvert.fr](http://Projetvert.fr)

### **b) Pertes énergétiques**

Le fait que le volet puisse s'autoalimenter n'est en soit pas un grand gain énergétique, mais permet néanmoins de légères économies. Le fait que le volet s'ouvre de façon automatique lorsqu'il fait jour dehors permet d'éviter à ce que l'utilisateur ait besoin d'allumer les lumières alors qu'il fait jour. L'afficheur, affichant la température extérieure et intérieure, permet à l'utilisateur de prendre connaissance (si il ne possède pas déjà de thermostat) de la température intérieure et extérieure, et donc de réduire le chauffage ou ouvrir les fenêtres s'il fait trop chaud. À l'échelle individuelle les gains semblent minimes, mais à plus grande échelle, les gains sont significatifs.

## **6) Amélioration du confort**

L'utilisateur peut simplement commander son volet ou ses volets de plusieurs façons, soit avec une télécommande infra rouge, soit avec une application sur son téléphone, soit depuis un navigateur web (ordinateur, tablette, téléphone) ou depuis un interrupteur, de cette façon l'utilisateur n'a donc pas à arrêter sa tâche courante pour contrôler le volet. De plus, le volet se comporte de façon automatique, il n'est donc pas nécessaire à l'utilisateur de se préoccuper du volet.

### **a) Affichage**

L'affichage des informations est effectué par l'écran LCD (contrôlé via l'Arduino), ainsi que par le site internet et l'application (contrôlé par la carte Raspberry PI). L'afficheur va donc afficher pendant 5 secondes la Température intérieure en degrés Celsius, puis pendant 5 secondes la température extérieure en degrés Celsius puis, pendant 5 secondes la pourcentage de batterie et pendant 5 secondes l'éclairement extérieur en Lumen.

### **b) Automatisation**

Le mode automatique du volet peut être activé à la fois par la télécommande Infra rouge, le site web et l'application mobile. Une fois le mode automatique activé, l'Arduino va considérer que si la température extérieure est supérieure à 35°C, le volet se fermera afin de conserver la fraîcheur à l'intérieur du bâtiment. Si l'écart entre la température extérieure et intérieure est supérieur à 20 degrés celsius, le volet se fermera. Lorsque l'éclairement est supérieur à 600 lumen le volet s'ouvrira afin que l'utilisateur puisse profiter de la lumière naturelle. Pour cela, nous nous sommes basés sur le tableau ci-dessous.

Nous avons choisi la valeur de 600lux, puisque qu'elle est inférieure à celle d'une zone d'ombre et supérieure à valeur de la fin d'un coucher de soleil et supérieure à une lumière artificielle.

Donc, le volet ne se fermera pas s'il y a de l'ombre pendant une journée ensoleillée, ou resteras ouvert en pleine nuit s'il se trouve sous un lampadaire.

	Eclairage en Lux
Pleine lune	0,5
Lumière d'une bougie	10 ~ 15
Rue de nuit	30 ~ 100
Pièce intérieure, lumière artificielle	50 ~ 300
Pièce intérieure, lumière du jour	100 ~ 700
Extérieur à l'ombre	700 ~ 7000
Extérieur coucher du soleil	500 ~ 5000
Extérieur nuageux	5000 ~ 20000
Extérieur soleil	20000 ~ 70000

### III Réalisation du projet

#### 1) Répartition des tâches

##### Hugo Tamboise

Chef de projet, responsable de l'usinage et de la mise en place des éléments mécaniques et électriques sur le volet.

S'assure également du contrôle qualité générale et de l'assistance Arduino.

##### Hugo Magnier

Responsable de la mise en place des capteurs, de la programmation de la carte Arduino et de son comportement en mode automatique.

##### Théo Gicquel

Responsable de l'interprétation, du stockage et de l'affichage des données transmises de l'Arduino au Raspberry Pi, ainsi que du contrôle du volet via l'interface du site web hébergé.

##### Benjamin Roger

Développement de l'application Android et assistance au design de l'interface utilisateur.

##### Michael Forques

Responsable de l'ajout du contrôle du volet sur l'Arduino via une télécommande infrarouge et assistance sur l'application Android.

#### 2) Communication

La Raspberry PI 3 model B communique les ordres à l'Arduino Mega via ses broches.

La carte Arduino envoie des informations à la Raspberry via le port série.

La Raspberry communique avec l'application via son serveur hébergé localement.

## Présentation Générale

Ma partie du projet se concentre sur la création d'une interface pour les utilisateurs permettant un accès aux informations du volet ainsi que du contrôle de celui-ci, le tout hébergé au sein de la carte raspberry PI.

j'ai tout d'abord établi les besoins auxquels je devais répondre : **interpréter les données provenant du port série de l'arduino, donner des ordres à l'arduino, et centraliser ces tâches sur une interface utilisateur.**

Pour interpréter les données séries, j'ai décidé d'utiliser le langage interprété python avec la bibliothèque Serial permettant l'interaction avec le port série.

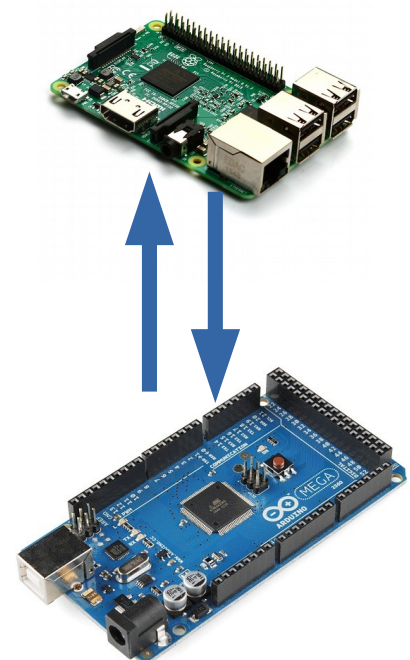
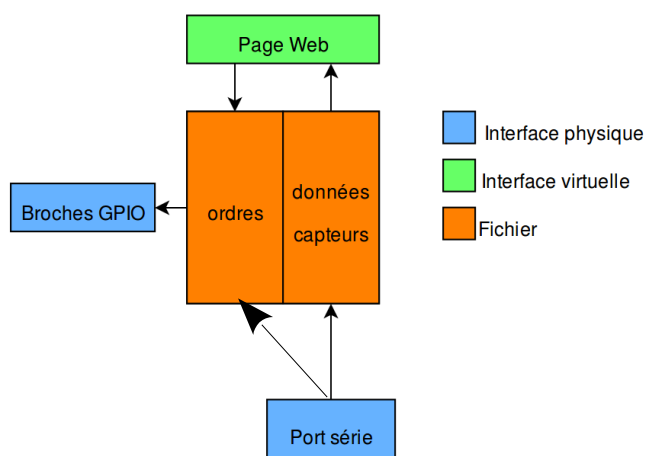
Ce programme possède pour objectif d'acquérir les requêtes envoyées par l'Arduino, de les séparer en différentes variables, et ensuite de synchroniser chaque variable dans son fichier individuel.

Pour ce qui est d'afficher les variables à l'utilisateur, j'ai préféré utiliser un serveur web au lieu d'un programme, car cela permet aux informations d'être consultables par n'importe quelle machine pouvant être connectée au réseau local et par ma familiarité avec les langages orientés web tels que le PHP, le HTML, et le CSS.

Le principe de l'interface utilisateur du site web consiste à afficher sur plusieurs pages web le contenu des fichiers des variables récupérées auparavant via le port série.

Pour que l'utilisateur puisse donner des ordres au volet, le procédé inverse se produit : le site web récupère l'ordre de l'utilisateur à partir d'une variable et le stocke dans un fichier.

Puis, un autre programme python s'occupe de lire les fichiers correspondant aux ordres à donner au volet et allume ou éteint les broches GPIO en conséquence.



## Interprétation des données séries

Les données telles que la température des capteurs ou le niveau de la batterie sont organisées dans une chaîne de caractères agencés de cette façon

```
température intérieure|température extérieure|charge batterie|luminosité|etat volet|mode|
```

Par exemple, dans le cas où :

- la température intérieure serait de 20°C
- la température extérieure serait de 18°C
- la batterie à 99 % de capacité
- la luminosité à 200 lumens
- le volet serait ouvert
- le mode serait manuel

la chaîne de caractères ressemblerais à : `|20|18|99|200|ouvert|manuel|`

Pour cela, j'utilise le langage de programmation interprété Python pour séparer les caractères présents entre les barres verticales ( "|") avec la fonction `split()`.

Cependant, je n'avais pas initialement remarqué que l'Arduino plaçait un caractère de retour chariot à la fin de chaque ligne. J'ai remédié à cela en plaçant une variable supplémentaire inutile nommée "dump" parmi les variables des capteurs.

```
# lecture d'une ligne sur le port serie.  
response = serialport.readline()  
  
# separation de la ligne recue en variables  
tempint,tempext,batterie,luminosite,etat,mode,dump = response.split("|")
```

J'ai ensuite créé une fonction nommée `envoi_serveur()` qui permet de stocker une variable dans un fichier en écrasant la valeur précédemment stockée dans ce dernier.

```
# déclaration de la fonction  
def envoi_serveur(fichier, variable) :  
    # ouverture du fichier en mode écriture (write)  
    contenu = open(fichier, "w")  
    # suppression de l'ancien contenu  
    contenu.truncate()  
    # écriture des données  
    contenu.write(variable)
```

Puis finalement, j'ai spécifié l'emplacement des fichiers de données ainsi que les paramètres du port série en accord avec l'Arduino, puis unifié ces deux extraits dans un seul programme final s'exécutant en continu.

```
#!/usr/bin/env python2
import serial

#paramètres port série
serialport = serial.Serial("/dev/ttyACM0", 9600, timeout=30)

# emplacement des fichiers
d_tempint = "/var/www/html/tempint.dat"
d_tempest = "/var/www/html/tempest.dat"
d_luminosite = "/var/www/html/luminosite.dat"
d_batterie = "/var/www/html/batterie.dat"
d_etat = "/var/www/html/etat.dat"
d_mode = "/var/www/html/mode.dat"

def envoi_serveur(fichier, data):
    contenu = open(fichier, 'w')
    contenu.truncate()
    contenu.write(data)

while True:
    # lecture d'une ligne sur le port série.
    response = serialport.readline()

    #séparation de la ligne recue en variables
    tempint,tempest,batterie,luminosite,etat,mode,dump = response.split("|")

    # appel des fonctions
    envoi_serveur(d_tempint,tempint)
    envoi_serveur(d_tempest,tempest)
    envoi_serveur(d_batterie,batterie)
    envoi_serveur(d_luminosite,luminosite)
    envoi_serveur(d_etat,etat)
    envoi_serveur(d_mode,mode)
```

### **Préparation du serveur web**

Mon prochain objectif étant d'afficher les informations des fichiers dans une page web, il était obligatoire d'installer un serveur http. J'ai donc installé un serveur apache, car il s'agit du serveur le plus stable et le plus utilisé au sein de l'environnement Linux.



Le rôle du serveur http apache est de permettre l'affichage de pages html et php via le protocole http sur le réseau local.

Nous souhaitions initialement rendre le serveur accessible en dehors du réseau local, mais les ports du router du lycée n'autorisant pas les requêtes http en entrée, cette fonctionnalité n'a donc pas pu être incluse.

Les fichiers présents dans /var/www/html/ sont donc disponibles à la consultation via un navigateur web.


j'ai ensuite installé le module php afin qu'il soit possible interpréter les fichiers php.

## Affichage des variables

Une fois le "terrain" préparé, j'ai écrit un premier morceau de code php me permettant de récupérer les informations contenues dans les différents fichiers grâce à la fonction php `file_get_contents()` puis de les afficher avec la fonction `echo`.

```
<?php
// récupération des données
$etat      = file_get_contents('etat.dat');
$tempint   = file_get_contents('tempint.dat');
$tempext   = file_get_contents('tempext.dat');
$lumin     = file_get_contents('lumin.dat');
$ouverture = file_get_contents('ouverture.dat');
$mode      = file_get_contents('mode.dat');
$stop      = file_get_contents('stop.dat');
$batterie  = file_get_contents('batterie.dat');

// affichage des données à partir des variables
echo "Température intérieure : " . $tempint . "°C<br>";
echo "Température extérieure : " . $tempext . "°C<br>";
echo "Luminosité : " . $lumin . " lumen<br>";
echo "Etat : " . $etat . "<br>";
echo "Ouverture : " . $ouverture . "<br>";
echo "Mode : " . $mode . "<br>";
echo "Charge batterie : " . $batterie . "%<br>";
?>
```



```
Température intérieure : 20°C
Température extérieure : 18°C
Luminosité : 200 lumen
Etat : ouvert
Ouverture : haut
Mode : manuel
Charge batterie : 99%
```

Résultat dans le navigateur

## Modification des ordres

J'ai ensuite réalisé un autre script php me permettant cette fois-ci de modifier le contenu d'un fichier via la fonction `file_put_contents()` qui écrase le contenu avec une variable donnée.

```
<?php
file_put_contents('mode.dat', $_GET['mode']);
header("Location: {" . $_SERVER['HTTP_REFERER'] . "}");
?>
```

Ici, la fonction `$_GET['']` correspond à la variable qui écrase le contenu du fichier, et correspond directement à la variable fournie via l'URL du navigateur.

Par exemple, avec une URL telle que:

<http://192.168.43.120/setmode.php?mode=auto> est entrée via le navigateur, la fonction `file_put_contents()` remplacera `$_GET['mode']` par la valeur de la variable `mode` de l'url et écrira donc dans le fichier `mode.dat` la chaîne de caractères 'auto'.

```
file_put_contents($_GET['fichier'], $_GET['valeur']);
```



```
file_put_contents('mode.dat' , 'auto') ;
```

J'ai également ajouté la fonction `header()` qui redirige le navigateur vers la page précédente une fois le fichier écrasé.

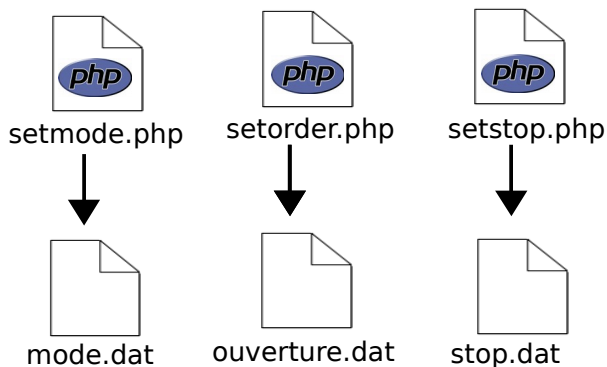
J'ai ensuite réutilisé plusieurs programmes similaires pour modifier les ordres de fermeture/ouverture, et d'arrêt du moteur du volet.



```
<?php
file_put_contents('ouverture.dat',$_GET['ouverture']);
header("Location: {$_SERVER['HTTP_REFERER']}");
?>
```

```
<?php
file_put_contents('stop.dat',$_GET['stop']);
header("Location: {$_SERVER['HTTP_REFERER']}");
?>
```

Ainsi, chaque ordre à donner au volet possède un fichier php et un fichier de stockage dédié.



### **Transmission des ordres**

Pour transmettre les ordres au raspberry PI, nous nous sommes mis d'accord sur l'usage de ses broches GPIO reliées à plusieurs pins de l'Arduino au lieu du port série, car cela permet l'actualisation continue des ordres à transmettre au volet sans occuper le port série.

Pour cela, j'ai encore une fois utilisé le langage Python, car une librairie (ou module) spécifique aux broches GPIO y est déjà présente au sein de l'environnement natif.

Par exemple, lorsque le mode de fonctionnement souhaité par l'utilisateur est manuel, la broche 35 sera alimentée, dans le cas contraire où le volet devrait se fermer et s'ouvrir automatiquement, la broche 35 ne serait pas alimentée, de plus cela permet que le volet reste indépendant dans le cas où le raspberry PI ne fonctionnerait pas.



Emplacement de  
l'environnement d'exécution

Importation des modules

Configuration des broches

Définition des fichier

Boucle Continue

Lecture des fichiers

Cas où le mode est  
manuel

Cas où le mode est  
automatique

```
#!/usr/bin/env python2
import RPi.GPIO as GPIO
import time

time.sleep(2)

# définition des broches en mode circuit
GPIO.setmode(GPIO.BOARD)

GPIO.setup(35, GPIO.OUT) # sortie, AUTO/ MANUEL
GPIO.setup(36, GPIO.OUT) # 36 sortie, OUVERT/ FERME
GPIO.setup(37, GPIO.OUT) # 37 sortie, ARRET MOTEUR

d_etat = "/var/www/html/ouverture.dat"
d_mode = "/var/www/html/mode.dat"
d_stop = "/var/www/html/stop.dat"

while True:
    mode = open(d_mode).read()
    ouverture = open(d_etat).read()
    arret = open(d_stop).read()

    if mode == "manuel":
        GPIO.output(35, 1)

        if ouverture == "haut":
            GPIO.output(36, 1)

        else :
            GPIO.output(36, 0)

        if arret == "1":
            GPIO.output(37, 1)

        else :
            GPIO.output(37, 0)

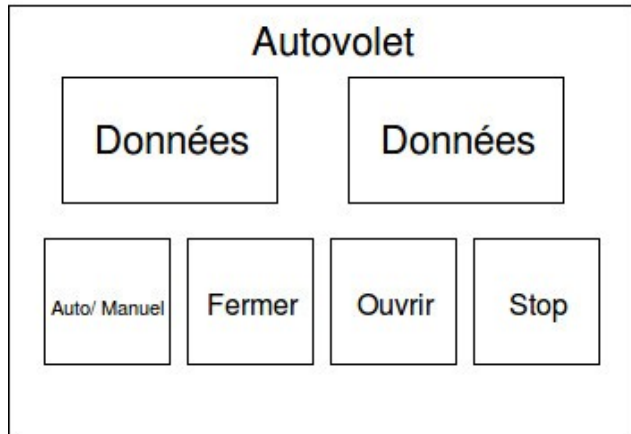
    if mode == "auto":
        GPIO.output(35, 0)
        GPIO.output(37, 0)
```

## Développement de l'interface utilisateur

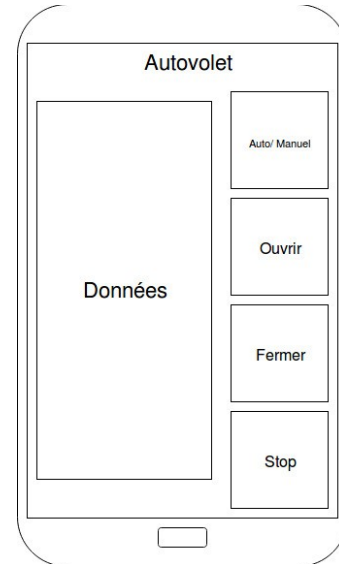
L'interaction entre l'utilisateur et le volet techniquement possible, il fallait ensuite rendre cette interaction facile et intuitive pour toutes les personnes.

j'ai tout d'abord réalisé deux croquis d'interface :

- un pour les ordinateurs

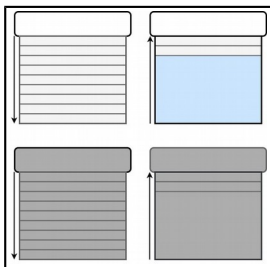


- un pour les téléphones



Pour correspondre à ces croquis, j'ai choisi modifier la page php d'affichage déjà existante, et d'y rajouter des liens vers les pages php sous forme d'images pour leur donner un aspect de bouton.

J'ai d'abord réalisé plusieurs images :



Images d'état du volet

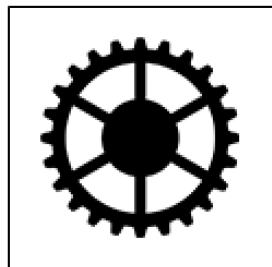


Image du mode automatique

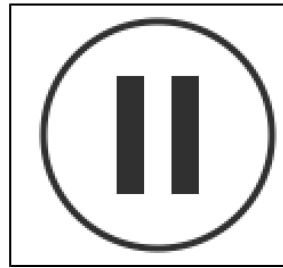


Image du bouton d'arrêt

puis les a définis dans des liens individuels via des variables.

(éléments dédiés à l'apparence purement esthétique omis pour raison de lisibilité )

```
//bouton haut
$ico_hautInactif='<div class="bouton"><a href="setorder.php?order=haut"></a><p>OUVERTURE</p></div>';
$ico_hautActif='<div class="bouton"><p>OUVERTURE</p></div>';
$ico_hautGris='<div class="bouton"><p>OUVERTURE</p></div>';

// Bouton Bas
$ico_basInactif = '<div class="bouton"><a href="setorder.php?order=bas"></a><p>FERMETURE</p></div>';
$ico_basActif = '<div class="bouton"><p>FERMETURE</p></div>';
$ico_basGris = '<div class="bouton"><p>FERMETURE</p></div>';

// Bouton Automatisation
$ico_manuel= '<div class="bouton"><a href="setmode.php?mode=auto"></a><p>AUTO : OFF</p></div>';
$ico_auto= '<div class="bouton"><a href="setmode.php?mode=manuel"></a><p>AUTO : ON</p></div>';

// Bouton Stop
$ico_stop0 = '<div class="bouton"><a href="setstop.php?stop=1"></a><p>STOP ? </p></div>';
$ico_stop1 = '<div class="bouton"><a href="setstop.php?stop=0"></a><p>STOP</p></div>';
```

On peut ainsi afficher n'importe quel bouton en ajoutant simplement une courte instruction telle que `echo $ico_auto;`

Ensuite, je me suis focalisé sur l'affichage des boutons selon l'état des ordres présents au sein du serveur.

Pour cela, j'ai rajouté à la page le code ci-contre.

Lors de l'exécution, le serveur vérifie l'état de la variable `$mode`. Si il est automatique, le bouton affiché permettras de passer en mode manuel et inversement.

Cette façon, de gérer l'affichage m'a permis de répondre très rapidement aux attentes des autres membres du groupe sur l'apparence de l'interface, et également d'en faciliter la compréhension.

```
if ($mode == 'manuel') {
    echo $ico_manuel;

    if ($stop == 0){
        if ($ouverture == 'haut'){
            echo $ico_hautActivable;
        }

        if ($ouverture == 'bas'){
            echo $ico_basActivable;
        }

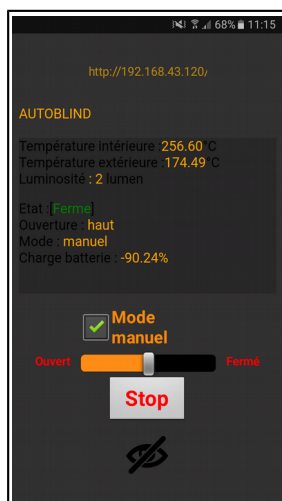
        echo $ico_stop0;
    }

    if ($stop == 1) {
        echo $ico_hautActivable;
        echo $ico_basActivable;
        echo $ico_stop1;
    }
}

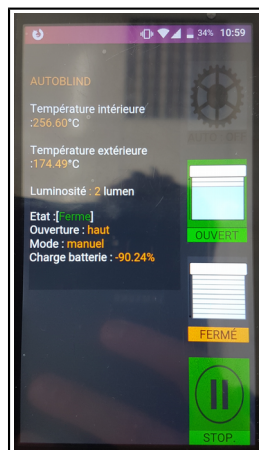
if ($mode == 'auto') {
    echo $ico_auto;
}
```

Pour finir, il était nécessaire de produire un rendu graphique de l'interface satisfaisant, pour cela, j'ai utilisé du CSS, qui permet de gérer l'apparence et l'agencement des éléments de la page web.

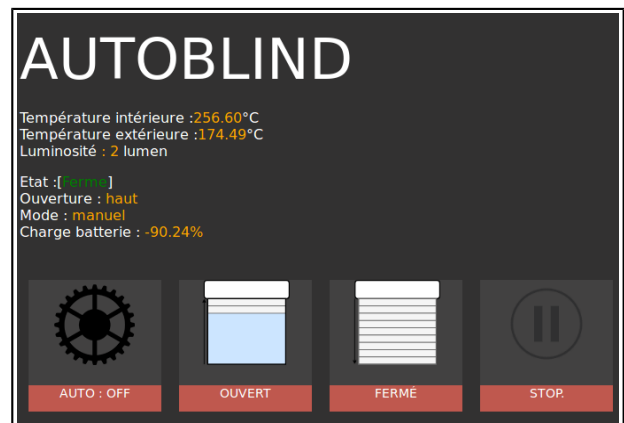
Pour des raisons évidentes de lisibilité, j'ai préféré faire impasse sur l'explication du CSS.(plus de 150 lignes)



Rendu final sur l'application mobile



Rendu final sur navigateur mobile



Rendu final de l'interface pour ordinateur

# CONCLUSION

Pour conclure, ma partie s'est avérée fonctionnelle d'un point de vue technique et répondant aux besoins attendus du système.

La réalisation de ce projet m'a permis de développer mes compétences en programmation, particulièrement avec le langage PHP .

Travailler au sein d'une équipe m'a permis d'acquérir un sens de coopération ainsi que des connaissances solides en électronique.

## **Points à améliorer**

J'aurais pu néanmoins améliorer l'aspect utilisateur en créant du code CSS plus travaillé qui s'adapterait correctement à la résolution des périphériques mobiles afin de convenir à tous les utilisateurs.

Le système pourrait également évoluer vers un site web accessible via internet.

# ANNEXE

## RÉSEAU LOCAL & ARBORESCENCE DU RASPBERRY PI

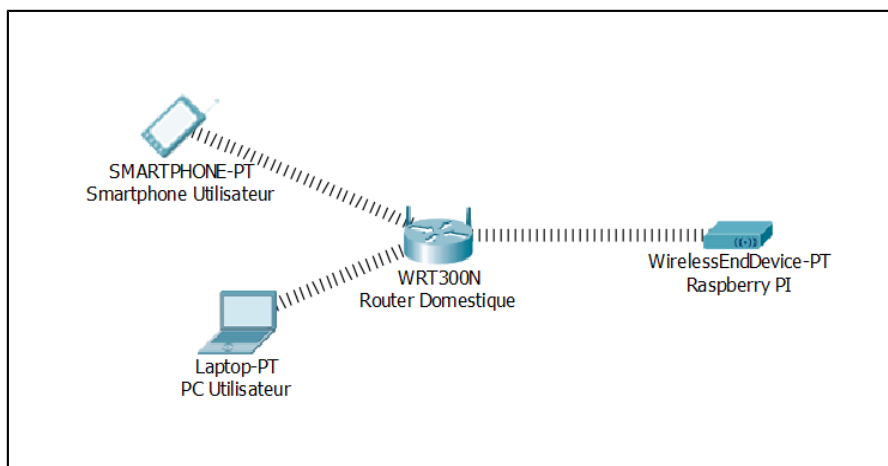
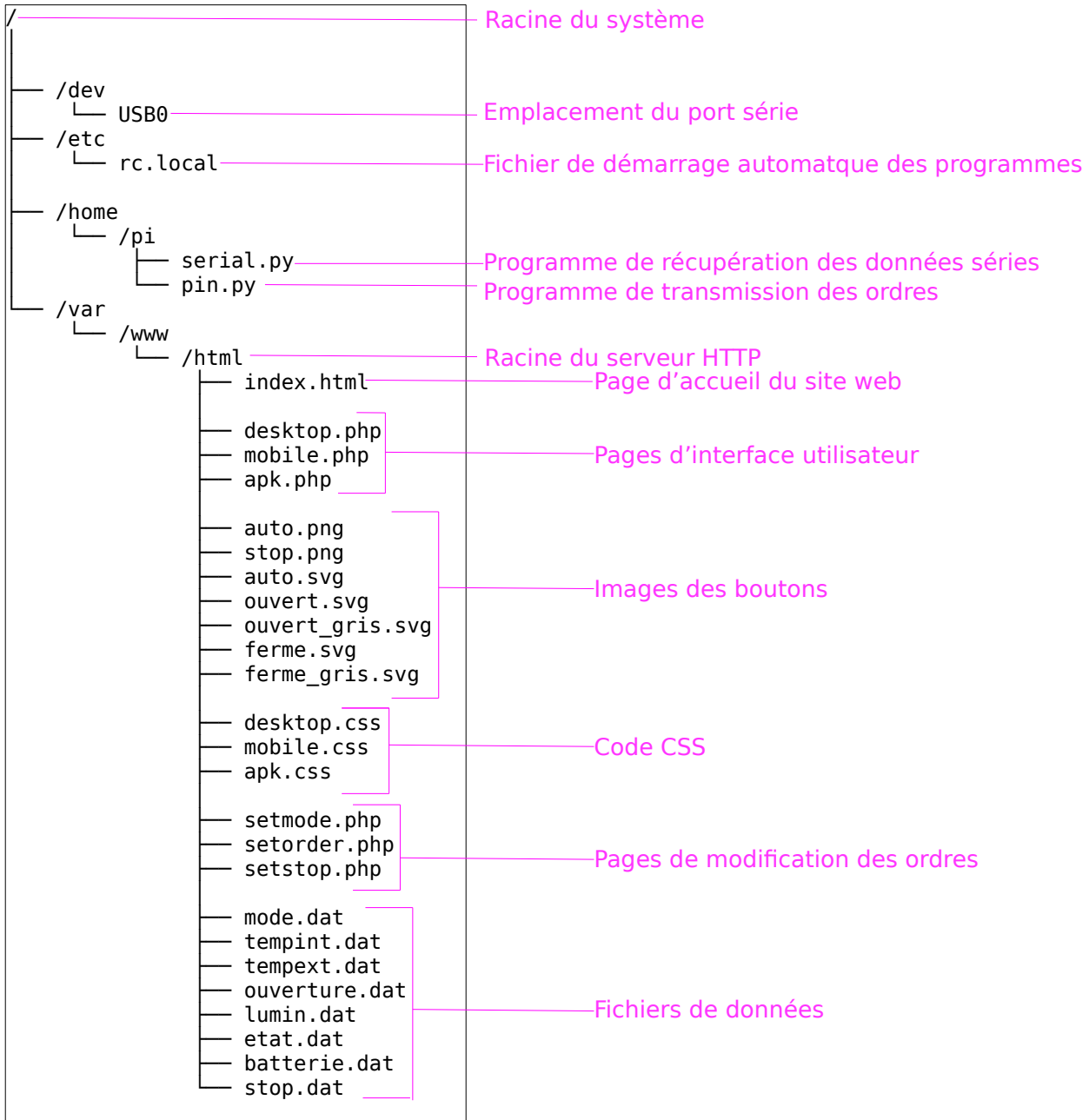


Diagramme du réseau local du raspberry PI.

# **LOGICIELS & OUTILS UTILISÉS**

**Système d'exploitation : Raspbian**  
**Réalisation des schémas : Draw.io**  
**Diagramme Réseau : Cisco Packet Tracer**  
**Traitement de texte : LibreOffice 5.1.6.2**  
**Éditeur de code : GNU Nano, Sublime Text 3**  
**Contrôle de version : Github**  
**Maintenance à distance : SSH**  
**Serveur HTTP : Apache**  
**Langages : PHP, Python 2.7,**  
**Interpréteur Bash**  
**Création d'arborescence : Tree**

## **REMERCIEMENTS**

**Lycée Emile Zola de Châteaudun**