

|  |
| --- |
|  |

Table des matières

[1) Introduction 3](#_Toc61433522)

[2) Besoins et objectifs du projet 4](#_Toc61433523)

[I. Objectifs et contraintes 4](#_Toc61433524)

[II. Objectifs techniques 4](#_Toc61433525)

[III. Délai et organisation 5](#_Toc61433526)

[3) Gestion de projet 6](#_Toc61433527)

[I. L’équipe 6](#_Toc61433528)

[II. Planification des tâches 6](#_Toc61433529)

[III. Répartition des tâches et des technologies 8](#_Toc61433530)

[4) Développement technique 9](#_Toc61433531)

[I. Solutions matérielles 9](#_Toc61433532)

[II. Solutions logicielles 10](#_Toc61433533)

[5) La réalisation du projet 11](#_Toc61433534)

[I. La maquette 11](#_Toc61433535)

[II. Le code : partie 1 : squelette Arduino 11](#_Toc61433536)

[III. Les différents composants 12](#_Toc61433537)

[IV. Le mode manuel 12](#_Toc61433538)

[V. Le mode automatique 12](#_Toc61433539)

[VI. Le mode alarme initial 13](#_Toc61433540)

[VII. L’assemblage du squelette 13](#_Toc61433541)

[VIII. Le code : partie 2 : mise en place du contrôle à distance 14](#_Toc61433542)

[VIII.1) Node-Red 14](#_Toc61433543)

[VIII.2) L’ESP 15](#_Toc61433544)

[VIII.3) L’AT Mega 15](#_Toc61433545)

[VIII.4) Les problèmes rencontrés et solutions apportées 16](#_Toc61433546)

[6) Bilan du projet 17](#_Toc61433547)

[I. Bilan collectif 17](#_Toc61433548)

[II. Conclusion 17](#_Toc61433549)

# Introduction

Ce rapport est la synthèse de notre projet de fenêtre intelligente réalisé dans le cadre de l’uv IFD2 du semestre d’automne 2020. Nous avons eu 2 mois et demi pour le réaliser afin de pouvoir mettre en pratique nos apprentissages en termes de programmation de cartes électroniques type Arduino en langage C.

De plus, nous avons pu exploiter lors du projet de l’utilisation de différents composants électroniques afin de mener à bien ce projet tels que des capteurs, un écran etc…

# Besoins et objectifs du projet

## Objectifs et contraintes

L’objectif principal du projet est de réaliser un objet de domotique composé au minimum de 2 capteurs, d’une carte WEMOS ainsi qu’un écran OLED.

De ce fait, nous avons décidé de mettre en œuvre un combo fenêtre/volet intelligent qui est capable :

* De fonctionner de manière autonome ou manuel, que ce soit le volet ou la fenêtre
* De faire office d’alarme
* D’être piloté depuis la fenêtre en elle-même ou depuis une interface accessible par internet
* D’afficher les informations relatives à son fonctionnement mais aussi à son environnement (la température, l’humidité ainsi que la quantité de lumière).

Cependant, étant donné les circonstances liées à la crise sanitaire et ses contraintes nous nous avons été obligés de nous organiser différemment et ce notamment à cause des réductions des heures de cours et du fait que certains membres du groupe n’étaient pas familiers avec les systèmes de type Arduino. Ainsi nous avons dû répartir les taches efficacement et assurer un suivi des avancées avec des discussions en groupe régulières pour pouvoir mener ce projet à bout. Compte tenu des moyens à notre dispositions et du temps disponibles, nous avons jugé suffisant de mettre au point une maquette taille réduite, qui a largement les moyens de démontrer les capacités de notre système.

## Objectifs techniques

Les objectifs techniques liés à notre fenêtre intelligente se basent sur l’autonomie du système ainsi que sa capacité à être contrôlable soit à distance via un ordinateur ou un smartphone.  Nous avons jugé aussi très important que le système puisse être contrôlé en physique et en hors connexion. Cette dualité de type contrôle nous a cependant imposé de nombreux choix technique pour la réalisation de ce projet.

Concernant l’autonomie de son fonctionnement, elle devra être basée sur les informations recueillies par les différents capteurs comme l’humidité, la température ainsi que l’intensité lumineuse. En fonction de ces informations elle devra pouvoir décider d’ouvrir ou fermer la fenêtre, ou bien de dérouler le volet.

En plus du contrôle à distance nous voulions donner à l’utilisateur la possibilité d’accéder en temps réel, à la visualisation des informations mesurées, du changement de mode automatique ou manuelle de la fenêtre ainsi que la mise en marche du mode alarme.

## Délai et organisation

Le délai de réalisation du projet est de 2 mois et demi. Le délai étant assez court, nous avons donc dû être très organisés pour éviter toute perte de temps. Nous avons dû anticiper les temps où nous ne pouvions être ensemble et se répartir des tâches personnelles que nous mettions en commun lors de nos sessions CrunchLab. Une des contraintes principales que nous avons rencontré lors de ce projet est due à la communication, il s’est révélé parfois difficile d’expliquer des aspects techniques par des moyens de communications à distance.

# Gestion de projet

## L’équipe

Notre équipe est composée de quatre personnes. Nous suivons tous actuellement le même cursus. L’équipe est donc la suivante :

-          Charles Dugas, Tronc Commun en troisième semestre.

-          Christian Do, Tronc Commun en troisième semestre.

-          Antoine Bonzi, Tronc Commun en troisième semestre.

-          Elios Cama, Tronc Commun en troisième semestre.

Malgré notre base commune du tronc Commun, nos expériences personnelles et professionnelles sont différentes. Charles Dugas et Antoine Bonzi ont tous deux participé à des cours de Science de l’ingénieur dans leurs lycées, ce qui les ont avantagés dans la réalisation de ce projet, car ils avaient déjà des connaissances sur le sujet des cartes Arduino. De plus, ce semestre nous avons tous choisi des matières différentes, ce qui a développé chez chacun des savoirs distincts mais complémentaires. Chacun de nous a ses propres centres d’intérêts dans des technologies, ce qui a été très stimulant pour notre curiosité lors de nos discussions. Cependant, malgré l’avantage de certains, nous avons tenu à préserver un certain équilibre dans les taches pour que tout le monde puisse apprendre de cette expérience, chose parfois compliqué du fait de la complexité des méthodes à utiliser dans ce projet et du temps à notre disposition pour se familiariser avec.

Il est bon de noter que la plupart d’entre nous avions déjà travaillé ensemble, et ainsi, nous avions déjà nos affinités et nous savions comment chacun travaillait, ce qui a largement facilité nos communications.

## Planification des tâches

Le temps est quelque chose qui nous à très stressés et nous avions raison de l’être. Ainsi dès la première semaine de cours de IFD2 et la présentation du projet, nous avons formé ce groupe pour répondre aux objectifs de l’UV le plus vite possible car le tout devait être rendu pour la semaine des finaux, ainsi trouver un projet cohérent avec les demandes et dont chacun pouvait participer fut notre premier objectif.

Durant les premières semaines de cours, nous avons assimilé des connaissances tout en réfléchissant chacun à des idées de projets correspondant à nos attentes. Et à la suite d’un brainstorming nous nous sommes mis d’accord sur notre idée de projet de fenêtre et volet connecté, sobrement nommé IWindows. Juste après nous avons établi les principales idées : comment nous allions réaliser la maquette, les matériaux et les composants nécessaires et l’architecture logicielle nécessaire. Ce jour-là, nous avons récupéré tout le matériel nécessaire pour ce que nous voulions faire dans l’idée de le compléter si nécessaire lors des séances du CrunchLab.

Pour être sûr que tout le monde était accordé sur les objectifs et les étapes à effectuer et donc pour perdre le moins de temps possible, nous avons organiser une réunion téléphonique le week-end du Samedi 12 Décembre pour prévoir tout ce que nous allions faire lors de la séance de la semaine suivante.

Pour ce projet nous voulions être le plus efficace possible. Par conséquent nous nous sommes concentrés dans un premier temps dans la mise en place des bases, pour ensuite venir les compléter avec ce qu’il manquera. En effet tout au long de ce défi, nous avons changé maintes fois d’idées et de solutions techniques pour arriver à notre but, que ce soit par ce que notre idée n’était pas réalisable ou que nous n’avions pas le matériel à disposition.

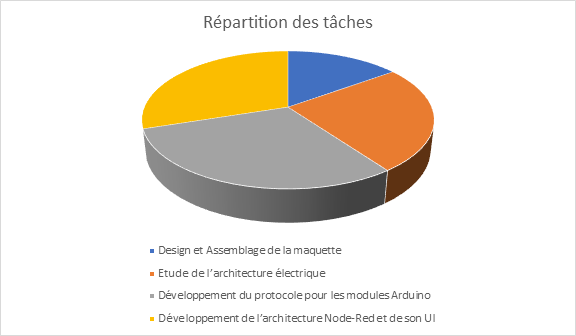
La première étape de cette stratégie fut la réalisation d’un prototype fonctionnel basique c’est-à-dire sans commande à distance pour avoir un squelette au projet.  Ainsi de manière parallèle nous avons réalisé la fabrication de la maquette, un premier schéma électronique pour faire fonctionner toute notre maquette et une base de notre programme Arduino.

Ces nombreuses tâches nous ont amenés à réserver plusieurs créneaux horaires sur les deux jours du Mercredi 16 et du Jeudi 17. Cette organisation nous permettait d’être à l’heure dans les délais mais aussi de préparer la suite. C’est pourquoi nous avons pris la décision que nous pourrions peaufiner les détails de la maquette pendant les vacances des fêtes de fin d’année et se consacrer à la suite à la rentrée. Nous avons confié la maquette à Charles Dugas pour les vacances car il détient du matériel technique à son domicile.

Durant ces vacances, tout le monde s’était vu attribuer des tâches différentes pour mettre en place la seconde phase de notre projet, c’est-à-dire la mise en place de l’automatisation ainsi que de la communication à distance. De plus, nous nous sommes alors mis d’accord sur le fait qu’il fallait finir l’architecture Node-Red durant la semaine, et qu’il fallait que la réalisation du projet soit finie avant la fin de la semaine. Aussi, nous avons pris la décision de commencer la rédaction du rapport durant cette première semaine de Janvier. Tout ceci nous permettrait de préparer la soutenance prévue pour la semaine du 11 Janvier.

Cette planification des tâches régulières nous a permis de respecter les délais sans avoir à se précipiter. Nous allons maintenant nous intéresser à la répartition des tâches dans l’équipe.

## Répartition des tâches et des technologies



Tandis que Christian Do et Elios Cama se sont plutôt focalisé sur les deux tâches de l’assemblage, du design de la maquette et de l’étude de l’architecture électrique, Charles Dugas lui s’est intéressé à l’Arduino et de la mise en coordinations de différents éléments de la maquette, pendant qu’Antoine Bonzi a réalisé la partie contrôle à distance avec l’ESP et la mise en place de l’interface node-red. Malgré une répartition des tâches qui peut paraître assez stricte, nous avons un peu tous touché à toutes les tâches car tous ces domaines sont interdépendants et pour être sûr que tout s’assemble, il a fallu que tout le monde se coordonne efficacement.

# Développement technique

## Solutions matérielles

#### Arduino WEMOS and RAMPS - 3D Printing Stack ExchangeCarte WEMOS :

C’est la carte que nous avons choisi d’utiliser dans notre projet car elle comporte à la fois un module Arduino AT mega 2560 et un module ESP8266. Les Arduino sont des cartes électroniques qui comportent un microcontrôleur programmable. Tandis que le module ESP permet d’établir une connexion avec un réseau Wifi.

### 

#### Moteurs :

Nous avons utilisé deux moteurs dans notre réalisation. Le premier sert à faire pivoter donc à ouvrir et fermer la fenêtre, et l’autre sert à enrouler et dérouler le volet. Les deux moteurs sont des steppers, ce qui veut dire qu’ils fonctionnent pas à pas. Nous avions besoins de ce type de moteur afin de fournir un nombre de pas nécessaire aux moteurs lors de leur activation pour que la fenêtre s’arrête exactement où on le voulait et que le volet s’arrête quand il est enroulé. La référence du moteur servant à ouvrir la fenêtre est 14HS11 et celle du volet est un Neocene 2T354201.

#### Boutons poussoirs :

Nous avons utilisé cinq boutons poussoirs dans notre projet. Nous les avons vissés dans le bois de notre maquette. Leurs fonctions sont les suivantes : activation du mode automatique, du mode manuel, du mode alarme, de l’ouverture de la fenêtre, de l’enroulage du volet. Ils assureront le fonctionnement en physique de la maquette.

#### Capteur infrarouge passif :

Nous avons utilisé un Capteur de mouvement PIR HC-SR501 afin de pouvoir enclencher notre système d’alarme lorsqu’un mouvement est détecté.

Pour son fonctionnement, le capteur détecte les rayonnements infrarouges dans son champ de vision et en déduit une présence ou un mouvement.

#### Alimentation-moteurs :

Elles seront nécessaires pour alimenter les deux moteurs car nécessitent à la fois un ampérage que la carte ne peut pas fournir mais aussi une tension supérieure à 5V.

#### 0.96 128x64 I2C Interface White Color OLED Display Module for Arduino - Free shipping - DealExtremeEcran Oled :

L’écran Oled était un composant fondamental et obligatoire dans le cadre de la réalisation de ce projet. En effet, il permet d’afficher les informations envoyées par les capteurs directement sur la maquette. L’écran est un Oled 12C de dimensions 128x64 pixels.

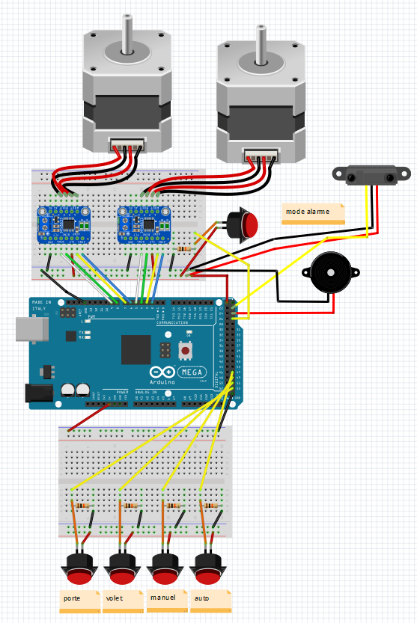
#### Capteur température de température DHT11 :

Nous avons utilisé un capteur de température et d’humidité de haute précision permettant l’obtention de beaucoup de donnée en temps réel.

#### Photorésistance LDR :

Les photorésistances sont des composants dont la résistivité dépend de la lumière ambiante. Quand une photorésistance est illuminée sa résistance diminue. On en a utilisé une pour avoir la quantité lumineuse extérieure.

#### Buzzer PIEZO :

Le buzzer est un composant qui réagit à la piézoélectricité et qui ici permet d’émettre un son. Nous utilisons un buzzer dans notre projet car il permet d’émettre le son d’une alarme lorsque celle-ci est activée.

## Solutions logicielles

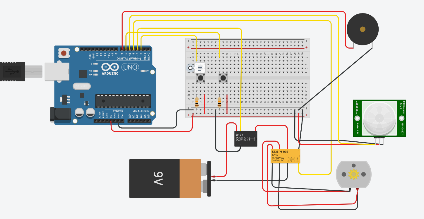
Pour réaliser ce projet nous avons utilisés de nombreux logiciels. Premièrement, lors de la phase de mise en place du squelette de notre maquette, nous avons procédé à de nombreux test Tinkercad pour être sûr que ce que nous voulions faire était faisable. Nous avons ensuite réalisé un modèle Fritzing que nous avons suivi au début pour éviter toute erreur de câblage lors du montage de celle-ci. Pour la partie codage nous avons utilisé un mélange d’Arduino IDE et de PIO sur Visual Studio. Le premier nous servait à contrôler l’AT Mega tandis que VS nous a servi à contrôler l’ESP. Le fait d’avoir ces deux interfaces différentes pour les deux modules nous permettais de mieux s’y retrouver et de rester organisé. Nous aurions pu mettre en place un Github pour le stockage et le partage du code mais nous avons préféré une approche plus néophyte avec Onedrive, qui nous permettais de partager et sauvegarder à la fois le code mais aussi toutes les ressources nécessaires au projet tel que le rapport, les images… Pour ce qui concerne le partage d’information et parce que nous n’avions que très peu de temps ensemble en présentiel, nous avons créé un serveur discord nous permettant d’échanger des informations et d’avoir un suivi régulier sur les tâches de chacun.

Figure 1 : Modèle Fritzing initial du projet

Figure 2 : Modèle Tinkercad précédant la fabrication initiale

# La réalisation du projet

## La maquette

Figure 4 : Premier assemblage

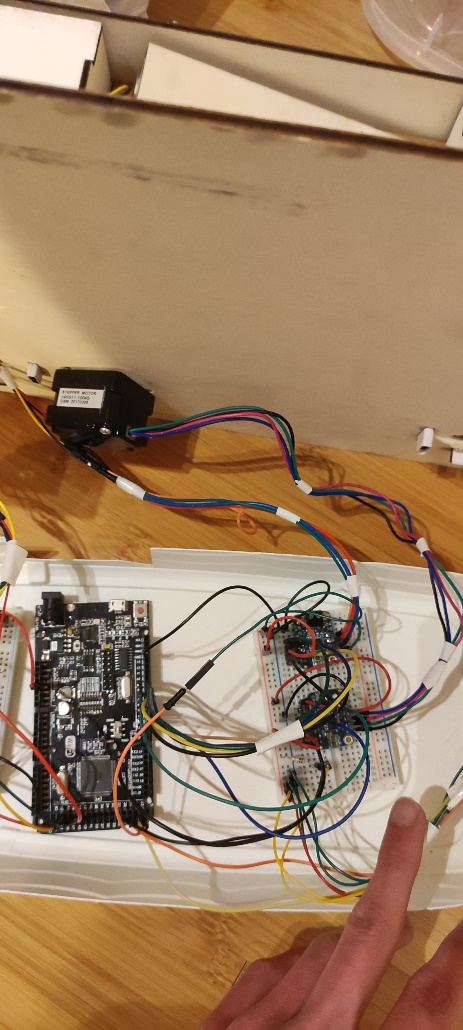
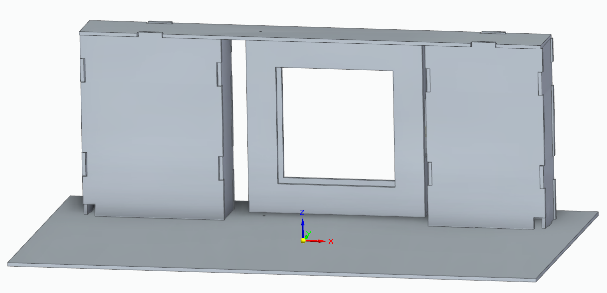
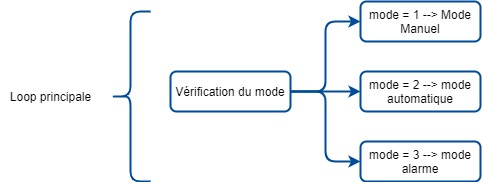
 Pour réaliser celle-ci, nous avons décidé de la réaliser sous découpe laser en gardant à l’esprit qu’elle devait être démontable et assemblable facilement. Ainsi la solution fut de découper les pièces avec des petits ergots pour permettre l’emboitement des pièces les unes entre les autres. Nous avons choisi comme base une grande plaque de contreplaqué pour assembler les différents éléments de la maquette. Sur celle viendra donc se placer les murs, la fenêtre et les différents boutons et capteurs ainsi que l’écran OLED. Nous avons eu pas mal d’évolution à effectuer sur celles tout au long de la réalisation du projet et ce dès la découpe des pièces au laser, il se trouve que nous avions pris trop de jeu entre les pièces que l’on a dû combler avec du scotch pour éviter que tout ne bouges trop et compromette l’intégrité de la maquette. De plus par rapport au dessin original, il a fallu percer de nombreux trous pour pouvoir installer les différentes capteurs et boutons qui n’étaient pas prévus à l’origine. La solution retenue pour faire actionner la fenêtre et le volet sont des moteurs steppers, un est placé directement sur l’axe de la fenêtre pour la faire pivoter et l’autre est placé à la parallèle de la fenêtre et enroule sur son axe un rideau. Concernant le câble-management on a trouvé plus simple et même plus propre de garder la partie câblage des moteurs sur breadboard sachant qu e l’on ne peut pas souder les modules de gestion des moteurs. En revanche la partie de gestion des boutons sera soudée du fait de la densité de câblage et des composants qui causaient des faux contacts.

Figure 6 : Mise en place des boutons et capteurs

Figure 3 : Plan SolidEdge

Figure 5 : Câblage des moteurs

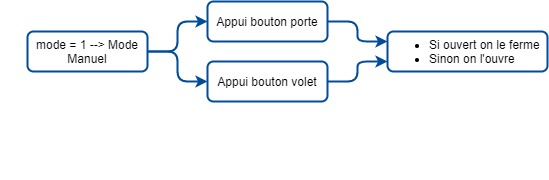
## Le code : partie 1 : squelette Arduino

La première étape du codage de notre projet fut de créer les bases du programme Arduino, l’objectif était dans un premier temps d’arriver à coordonner les différents éléments de la maquette, et donc sans notion de gestion à distance. Nous nous sommes mis d’accord sur un articulation en 3 blocs dans la boucle loop, le mode manuel, automatique et le mode alarme, qui s’activeront selon une variable globale : « mode ».

## Les différents composants

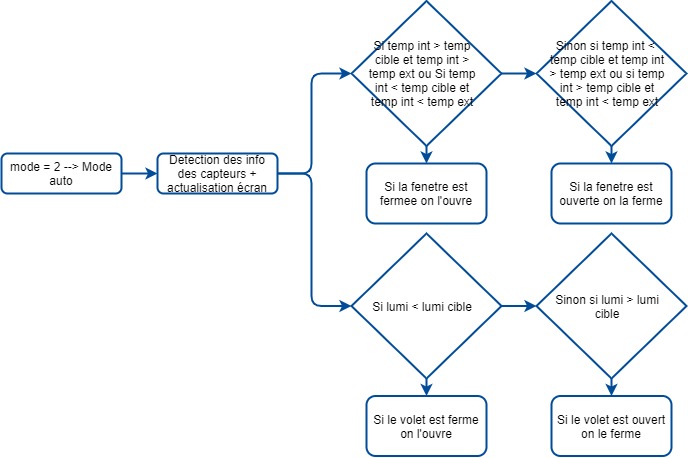
Mais avant d’arriver à ce squelette, chacun a dû s’assurer que nous pouvions utiliser correctement les différents composants individuellement. Ainsi chacun a programmé des composants de manière individuels. L’avantage de procéder ainsi est que l’on a rapidement identifié les composants qui allaient nous poser un problème. De manière générale, ce seront les composants nécessitants des bibliothèques externes. Tout d’abord les moteurs pas à pas dont les bibliothèques Arduino classiques ne fonctionnent pas et dont on a dû personnaliser les bibliothèques, ajouté à la nécessité d’utiliser un module de gestion externe à la carte, font que ces sont des composants que l’on a mis beaucoup de temps à maitriser. Ensuite fut le tour des capteurs de température DHT11 puis de l’écran OLED dont nous avons dû nous renseigner longtemps sur internet avant d’arriver à les exploiter. Un autre problème résidait dans l’obtention des informations de luminosité. Plusieurs options étaient disponibles avec des photodiodes, mais nous avons décidé à la suite de la découverte d’un tuto sur internet[[1]](#footnote-2), d’utiliser une photorésistance pour obtenir les informations de luminosité. Cette dernière méthode étant bien plus simple à mettre en place que les autres.

## Le mode manuel

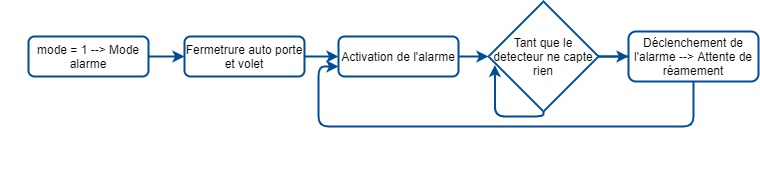
Mode consistant simplement d’ouvrir par suite de l’appui d’un bouton, la fenêtre ou le volet. C’est le premier mode implémenté et c’est celui qui nous a permis de nous assurer du bon fonctionnement de la maquette.

## Le mode automatique

C’est le mode qui nous a permis de vérifier le bon fonctionnement des composants restants, c’est-à-dire l’écran OLED et les capteurs de température et de luminosité. Cependant et malgré l’acquisition des informations d’humidité par les capteurs, nous avons fait le choix de ne pas exploiter cette information pour ce mode car nous l’avons jugé peu utile dans la mesure où si l’on ouvre la fenêtre car il y aurait trop d’humidité, cela aurait des conséquences directes sur la température, créant un réel problème pour le la dissociation des cas dont l’Arduino aurait à choisir. Nous avons donc choisi de privilégier la température sur l’humidité. C’est une des pistes d’amélioration principale de notre projet que nous pourrions poursuivre dans un futur.



## Le mode alarme initial

Pour ce mode, le fonctionnement est un peu différent, une fois le mode alarme lancé, le programme rentre dans une boucle interne à une fonction alarme(), pour a la fois attendre un mouvement et déclencher l’alarme si cela intervient.

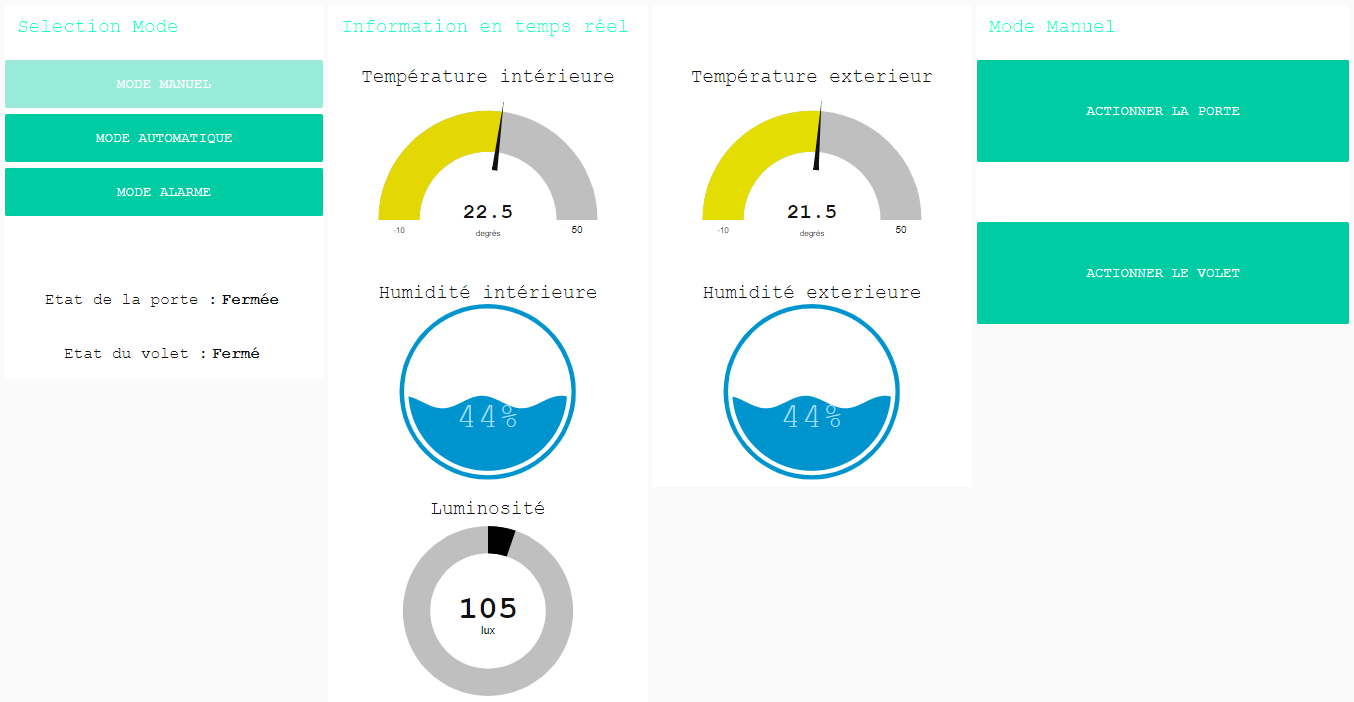
## L’assemblage du squelette

L’assemblage des différentes parties fut relativement aisé dans la mesure où chacun savait qu’il devait coder en respectant le code des autres, c’est-à-dire que les différents modes devaient fonctionner de manière indépendante et sans rentrer en interférence. Cependant quelque chose que nous n’avions pas prévu est qu’il fallait prévoir une interruption possible en plein milieu d’une boucle pour changer de mode, ce qui nous a pris plus de temps que prévu du fait qu’il fallait qu’elle puisse fonctionner de manière transparente et fonctionne quoi qu’il arrive. On a donc mis en place une fonction qui va venir vérifier à chaque itération d’une boucle de n’importe quel mode, qu’aucun des boutons n’a été appuyé, et si c’est le cas ordonne le changement de mode. C’est à la fin de cette partie que nous avons ainsi obtenu une maquette fonctionnant totalement en mode physique.

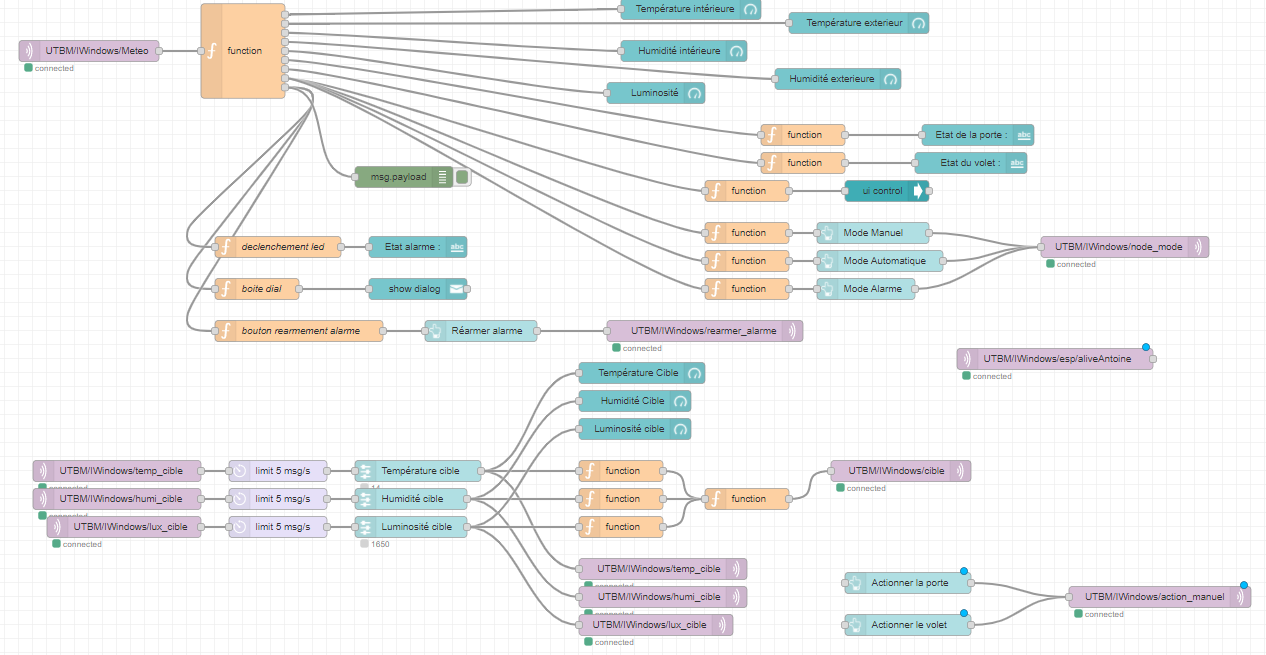
## Le code : partie 2 : mise en place du contrôle à distance

### VIII.1) Node-Red

Nous avons commencé par créer une interface de contrôle à distance de la fenêtre sur Node-Red. L’objectif de cette interface est de pouvoir accéder en temps réel aux informations telles que les températures et humidités (intérieur et extérieur), les lux et les états de la fenêtre et du volet. Nous voulions aussi pouvoir contrôler les modes (manuel, automatique et alarme) et effectuer des actions telles qu’actionner la fenêtre, le volet ou même changer les valeurs cibles pour le mode auto.



Pour la réaliser nous avons dû créer le schéma bloc ci-dessous, en paramétrant la taille, la place, … de chacun d’eux. Cependant le point le plus important de cette programmation n’est pas l’aspect esthétique mais la partie envoie et réception de données. En effet, l’information affichée sur l’écran doit à la fois changée si une action est faite en réel et à la fois être capable d’effectuer un changement en réel si elle est changée depuis l’interface Node-Red. Pour cela, l’information doit être transmise via un serveur MQTT (nous avons fait le choix d’un serveur MQTT public « mqtt.fluux.io »), afin que l’ESP puisse les récupérer à la carte Arduino et inversement. Ainsi nous avons créé des topics, dans le serveur MQTT, pour différencier le trajet des informations.

A titre d’exemple, le topic Météo permet de récupérer les informations météo des capteurs, envoyées par la carte vers l’esp puis sur le serveur MQTT. Une fois récupérée sur le serveur, une fonction sépare les parties du message pour les envoyer à chaque partie de l’interface pour la mettre à jour. En effet toutes ces informations sont les seules à n’avoir qu’une seule direction c’est-à-dire envoyée uniquement de la carte jusqu’à l’interface, j’ai nommé les variables d’état de la fenêtre, du volet, de l’alarme. A contrario il existe d’autre variables qui ne vont que de node-red à la carte, par exemple les valeurs cibles qui sont réglées sur l’interface et dont on doit les envoyer pour que la carte puisse les prendre en compte. De plus il existe aussi des variables qui font des allers-retours telles que le mode.

### VIII.2) L’ESP

L’ESP fait le lien entre le serveur Mqtt et la Wemos. Une fois l’ESP connecté au wifi et au serveur Mqtt, il s’inscrit sur les topics permettant de recevoir les informations de l’interface. Une fonction lui permet de détecter dès qu’un message est changé sur le serveur. Une fois détecté, dépendant du topic du message, il envoi ce message à la carte par son port Serial avec un caractère supplémentaire au début du message pour que la carte différencie les informations. Les messages envoyés par l’ESP prennent la forme [caractère différentiation#info1#info2#.....#]

De plus il fait également le lien entre la Wemos et le serveur. Une fonction détecte s’il y a un message arrivant sur son port Serial depuis la carte, si c’est le cas, il le publie automatiquement sur le serveur Mqtt dans le topic Météo.

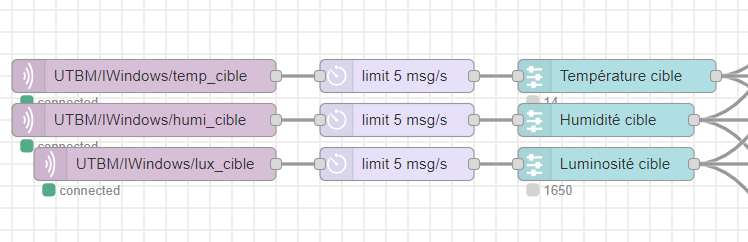
### VIII.3) L’AT Mega

À la suite de l’envoi des messages depuis l’ESP, la carte détecte si un message arrive sur son port Serial3 (port permettant la communication avec l’esp) grâce au serialEvent3(), et traite le message grâce au caractère de différenciation de topic pour ensuite effectuer les actions et modifier des valeurs si nécessaire. Parmi les variables reçues il y a le mode choisi sur node-red et qui s’il diffère du mode actuel sera modifié dans la carte. Ce traitement va se passer au même endroit que la détection d’appui des boutons pour éviter toute confusion.

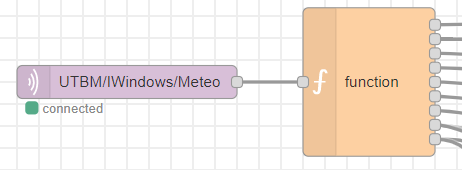
Pour ce qui est de l’envoi de données via le topic Météo, une fois les informations des capteurs DHT11 mis à jour, la carte écrit sur son port Serial3 le message, sous la forme [infos1#infos2#...#], comportant toutes les informations nécessaires pour que l’interface Node-Red s’actualise à l’ESP qui le publie sur le serveur.

### VIII.4) Les problèmes rencontrés et solutions apportées

Dans cette partie nous avons rencontrés quelques problèmes, principalement d’un point de vue praticité.

Premièrement, la fréquence d’envoie des données sur le serveur depuis l’interface était trop grande et donc la carte n’arrivait pas à traiter intégralement chaque message et un décalage se creusait jusqu’à l’arrêt de lecture des informations. Pour y remédier nous avons ajouté une limitation du nombre de messages pouvant être publier sur le serveur, comme ci-dessous, et ainsi la carte a maintenant le temps de décoder chaque message et entre temps la boucle permettant le fonctionnement de la maquette pour s’effectuer.

Un second problème auquel nous avons dû faire face est un problème de praticité concernant l’envoi d’informations depuis la carte. En effet nous voulions pour objectif d’envoyer les informations seulement si elles sont changées et donc d’envoyer la météo en continue (toutes les 3 secondes) mais uniquement parfois l’état de la fenêtre, du volet, de l’alarme et le mode si elles ont changé. Or nous nous sommes très vite rendu compte que cette méthode n’était pas viable. Ainsi nous avons alors pris la décision de regrouper toutes ces informations dans un seul message du côté de la carte même si elles n’ont pas de rapport avec la météo, elles seront envoyées toutes les 3 secondes dans un seul topic pour que l’interface actualise tout d’un coup.

 De ce fait, le message reçu sur ce topic doit être réparti dans différents blocs de l’interface et nous avons donc utilisé une fonction split avec autant de sortie de nécessaire pour tout récupérer comme il faut.

Un autre problème concernait les valeurs cibles du mode automatique, nous voulions qu’elles arrivent dans un seul topic, cependant nous avons eu du mal à actualiser le topic au changement d’une des valeurs cible. La solution fut de stocker les valeurs cibles sur le flow à l’aide de flow.set et de renvoyer la variable cible formatée dès l’actualisation d’une des valeurs.

Le dernier problème auquel on a fait face venait du mode alarme, du fait de son fonctionnement par boucle sans fin à l’intérieur de la fonction alarme (), la détection des messages venant de l’ESP n’est plus possible. Il a fallu alors redesigner entièrement cette fonction pour qu’elle puisse tourner comme avant mais cette fois en utilisant la boucle loop d’arduino.

# Bilan du projet

## Bilan collectif

La réalisation de ce projet d’étude dans le cadre de l’UV d’IFD2 a été bénéfique pour tout le groupe. En effet, tout au long du projet nous avons pu apprendre à travailler en équipe ainsi qu’exploiter les points forts de chacun. Le fait que pour certain la programmation de domotiques soit nouveau a permis de développer la curiosité vis-à-vis de ce domaine. De plus, malgré la crise sanitaire, nous avons dû apprendre à nous adapter afin de pouvoir maintenir le bon déroulement du projet du début à la fin. Le délai étant court, nous avons développé nos capacités à rechercher, trouver ainsi que mettre en œuvre les points que nous avons besoin pour résoudre les différents problèmes que nous avons pu rencontrer lors du projet comme la découverte de node-red, la programmation sur carte électronique, la correction de bugs etc…

La découverte du CrunchLab fut aussi très positif puisque nous avons pu découvrir les ressources que pouvait nous apporter ce lieu afin de travailler dans de bonnes conditions pour la réalisation de ce projet mais éventuellement pour de futurs projets.

Enfin, réussir à mettre en place et réaliser nos idées et objectifs nous a permis d’avoir la satisfaction de mener au bout un projet du début à la fin ce qui est pour chacun de nous le point le plus important de ce projet.

## Conclusion

En conclusion, la réalisation de ce projet a été très enrichissant pour nous en termes d’apprentissage et découverte du domaine de la domotique. Pouvoir mettre en pratique ainsi qu’adapter nos connaissances et tout ce que nous avons pu apprendre en cours pour réaliser le projet du papier à sa mise en service nous a permis de renforcer tous ces acquis que cela soit en termes d’informatique, programmation, électronique, électrotechnique et bien plus encore.

Concernant nos objectifs, ils sont pour la plupart rempli car nous avons obtenu un système fonctionnel, contrôlable à distance et en physique. Il possède aussi toutes les fonctionnalités que nous voulions. Cependant un des points d’amélioration pourrait être une meilleure gestion du mode automatique avec peut être un algorithme de gestion plus poussé et des capteurs plus précis.

1. Lux mètre à partir d’un LDR : https://www.allaboutcircuits.com/projects/design-a-luxmeter-using-a-light-dependent-resistor/ [↑](#footnote-ref-2)