STATION D'ACCUEIL ARDUINO

Jimmy HOCHART | Paul BADEUILLE
jimmy.hochart.elv@eilco-ulco.fr | paul.badeuille.elv@eilco-ulco.fr

EILCO Calais
Projet

Sommaire

Introduction

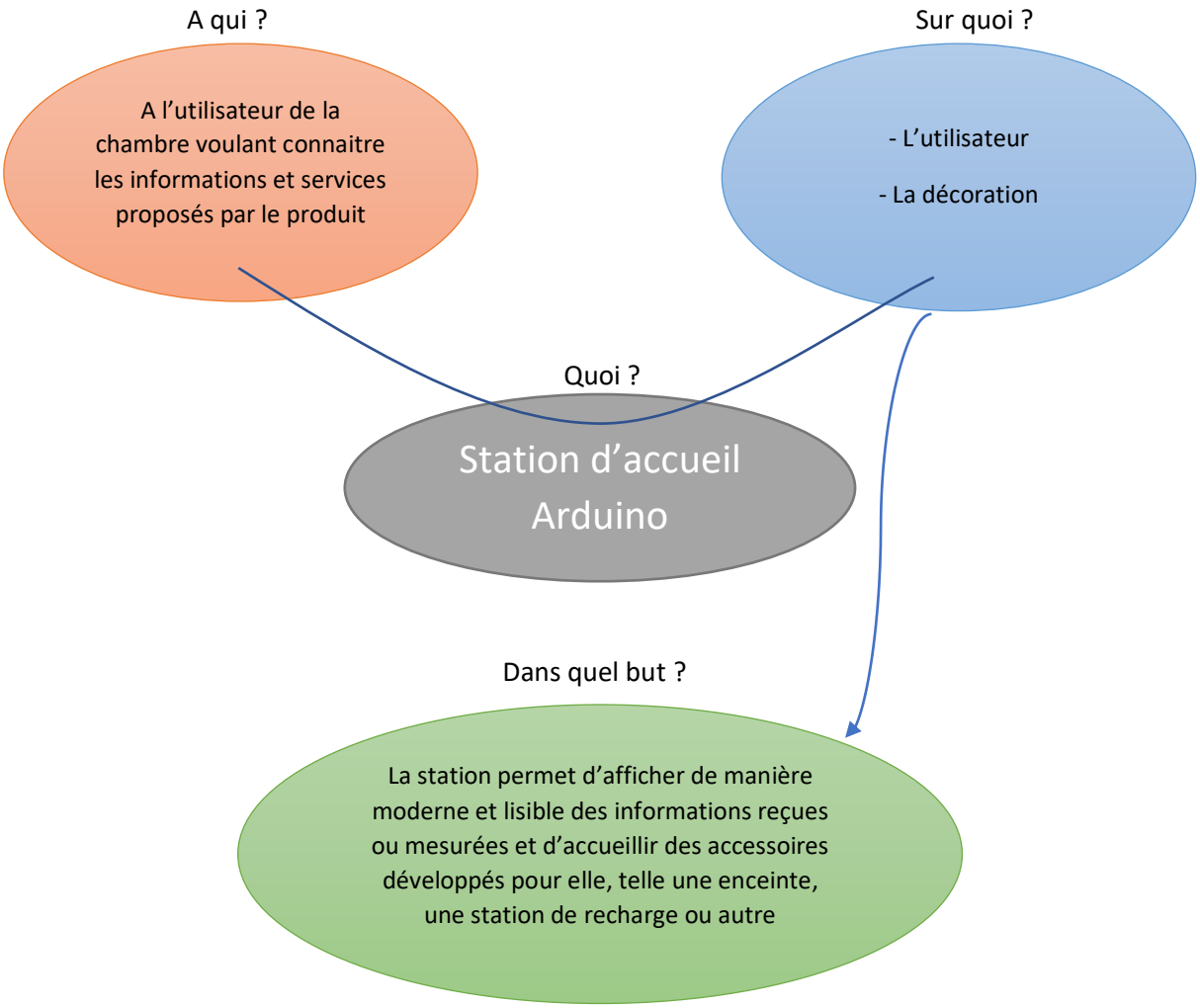
Notre projet en électronique pour le semestre 4 découle d'un besoin survenu il y a quelque temps. Nous voulions avoir un réveil personnalisable qui pourrait subvenir à quelques besoins basiques tels que l'affichage de l'heure ou de la date et qui pourrait être modulable pour permettre de s'en servir à notre guise. De ce fait, nous avons décidé de nous orienter sur la création d'une station d'accueil. Cette dernière sera sur une base d'Arduino, ce qui nous permettra par la même occasion de pratiquer le code C++ de manière appliquée à un problème réel. Si le produit était commercialisé, il serait orienté pour une population fuyant l'obsolescence programmée des produits, et aimant des produits polyvalents, modifiables, high-tech et contemporains.

I. CAHIER DES CHARGES FONCTIONNEL

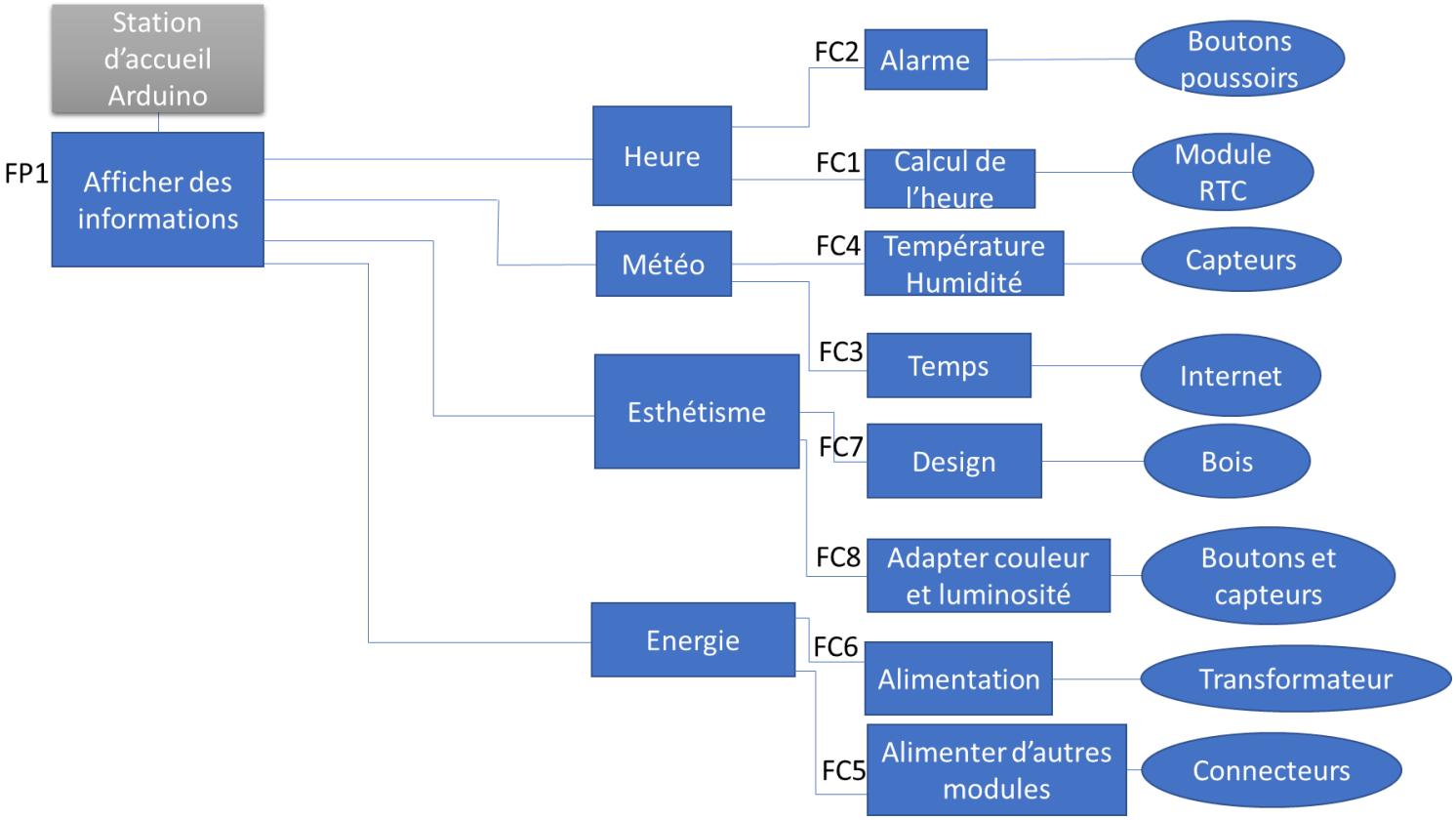
A) Expression fonctionnelle du besoin

Nom		Critère	Niveau	Flexibilité
FP1	Afficher des informations continuellement	Visibilité	Distance	2m
FC1	Déterminer l'heure/date exacte	Horloge intégrée	Précision : seconde	0
FC2	Permettre de programmer une alarme	Mémorisation de l'heure	Précision : seconde	0
FC3	Recevoir des informations sur la météo	Afficher le temps	Pluie, nuage, soleil, orage	0
FC4	Mesurer le température et l'humidité ambiante	- Température en °C - Humidité en %	- 10°C<temp<50°C - 20%<hum<80%	- ±2°C - ± 5%
FC5	Transmettre de l'énergie à d'autres modules	De manière sécurisée	5V	0
FC6	Alimentation en énergie	Transformateur secteur	230V/5V	0
FC7	S'adapter à la décoration	Design	Sobre	0
FC8	S'accommoder à l'utilisateur	- Adapter l'intensité de l'écran - Couleur(s) d'affichage	- Luminosité ambiante - En fonction de la pièce	- ±5s - 10 couleurs

B) Diagramme : Bête à cornes



C) Diagramme : F.A.S.T



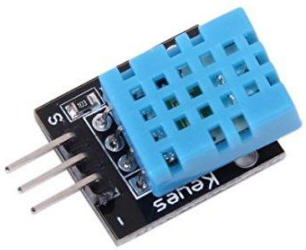
II. REALISATION DU PROJET

A) Matériel utilisé

Voici le matériel que nous avons utilisé pour mener notre projet :

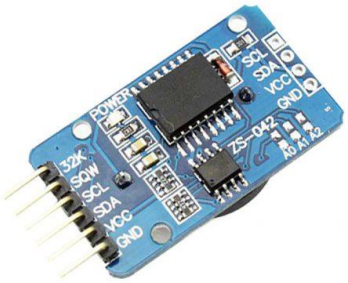
- Arduino UNO R3 : C'est une carte électronique programmable, utilisant le microprocesseur ATmega328P. Elle se programme en langage C++ grâce au logiciel Arduino IDE qui se charge de compiler le programme en langage machine et de l'envoyer dans la carte. On peut également communiquer avec la carte en série. Le programme se compose d'une fonction setup() initialisée une fois au démarrage et d'une fonction loop() qui est exécutée en boucle. Dans cette fonction, il y a appel à d'autres fonctions créées dans le programme. Chaque programme Arduino comporte au minimum ces deux fonctions. Elle dispose de 6 pins analogiques et 14 pins digitaux. Cette carte est open-source sous licence GNU GPL v3.

- DHT11 : Nous utilisons ce module afin de mesurer l'humidité dans l'air. Il s'agit d'un capteur résistif qui varie en fonction du taux d'humidité dans l'air. Il dispose également d'un capteur NTC (thermistance) pour mesurer la température mais nous n'utiliserons pas celui-ci par la suite.

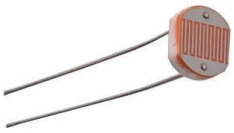


- BMP180 : Dans notre projet ce module permet de mesurer la pression atmosphérique (+-0,002hPa) et la température car ce capteur est plus précis que le DHT11 (+-2°C dans les valeurs extrêmes). On se sert de la pression pour estimer la météo (pluie, nuage ou soleil). Une fonction permet également l'altitude relative par rapport au niveau de la mer qui ne nous sera pas utile ici.

- RTC DS3231 : Ce module nous aide à déterminer l'heure et la date précisément, on l'utilise principalement pour sa capacité de fonctionnement hors tension grâce à la pile de sauvegarde. Il utilise un TCXO (oscillateur compensé en température) afin de mesurer l'heure et ainsi il peut renvoyer précisément l'heure et la date.

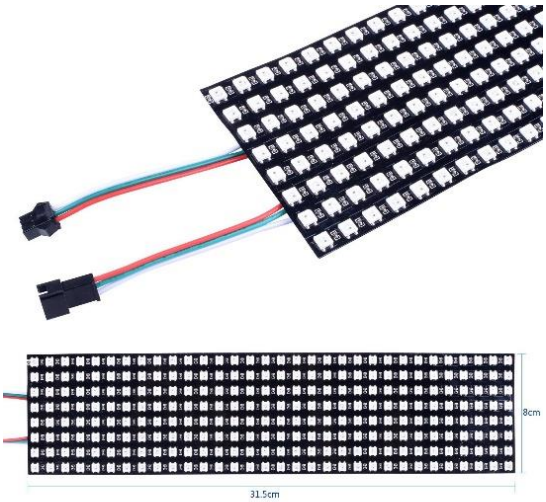


- Photorésistance : La photorésistance est un dipôle dont la résistance dépend de la lumière qu'il reçoit. La partie sensible du capteur est une piste de sulfure de cadmium en forme de serpent : l'énergie lumineuse reçue fait diminuer la résistance du composant. Cela va nous permettre de faire varier l'intensité lumineuse de notre matrice LED.

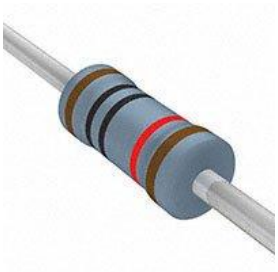


- Buzzer : Le buzzer est contrôlé soit avec une impulsion haute pour émettre un son, soit par une absence d'impulsion pour ne pas émettre de son. On peut également lui donner une fréquence et une durée mais nous n'en avons pas l'utilité dans le projet. Nous nous en servons pour faire l'alarme du réveil.

- Matrice LED WS2812B : Cette matrice fonctionne sur le même principe que les LEDs adressées, chaque LED dispose d'un contrôleur intégré. Grâce à un fil de données il nous est possible de contrôler chaque LED indépendamment. Également l'intensité de chaque LED peut être contrôlée indépendamment afin de faire une multitude de combinaison de couleurs. Il y a 256 LEDs.



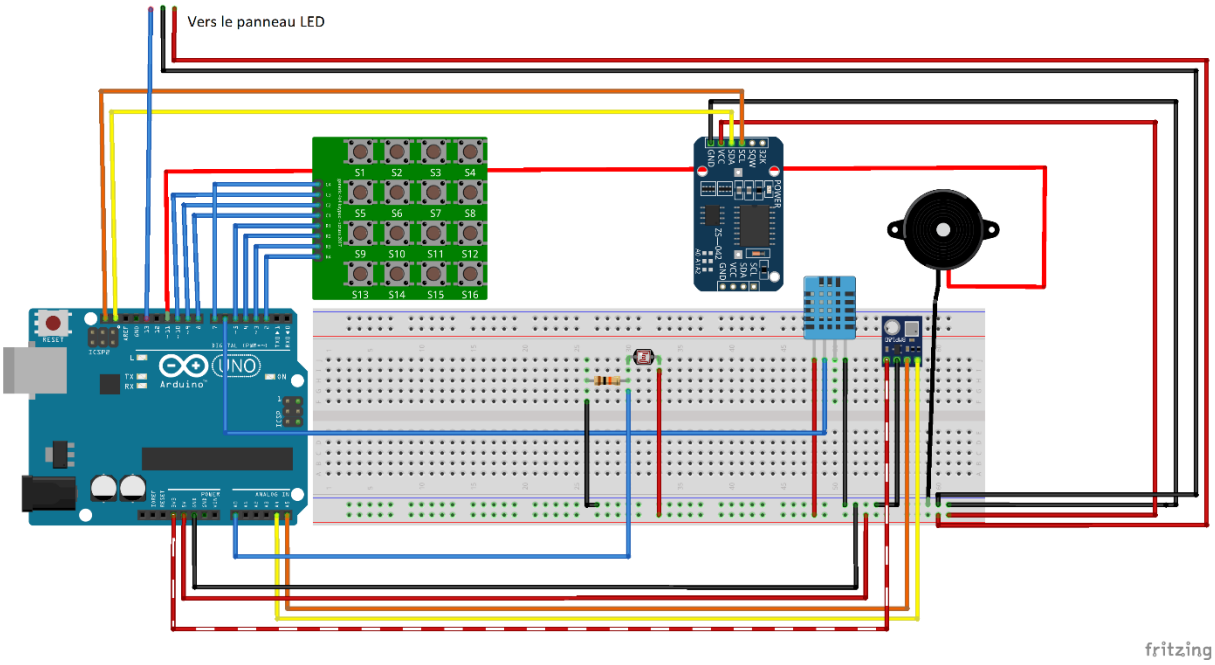
- Résistance 10kOhm : Cette résistance sera couplée à la photorésistance afin de créer un pont diviseur de tension pour l'Arduino.



- Matrice de boutons 4x4 : Chaque bouton est adressé selon un quadrillage, donc en ligne et en colonne. Chaque bouton peut donc être détecté indépendamment sans rebond. Ainsi cela permet à l'utilisateur d'interfacer avec la station.

B) Câblage des composants

Nous avons modélisé le schéma de notre câblage sur le logiciel Fritzing qui permet de faire des représentations schématiques illustrées.

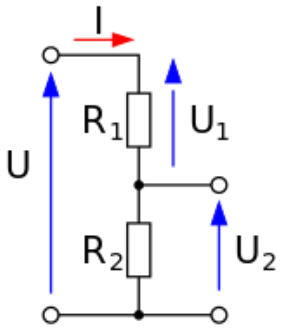


Nous avons utilisé une plaque de prototypage afin de relier tous les composants à l'Arduino. Les capteurs utilisés communiquent de différentes façons. Certains communiquent par les ports analogiques (majoritairement les dipôles résistifs) comme la photorésistance. Une valeur analogique peut prendre une infinité de valeurs dans un intervalle donné. Dans le cas de l'Arduino elles seront dans l'intervalle [0 ; 1023]. Ces valeurs sont fixées à partir d'une tension entre 0V et 5V.

Les ports utilisés par la matrice de bouton ou par le capteur d'humidité sont des ports digitaux c'est-à-dire qu'ils permettent seulement de lire des valeurs binaires, soit 0 soit 1.

Le module RTC et le capteur de pression communiquent par I^2C (Inter-Integrated Circuit) par les broches SDA (Serial Data Line) et SCL (Serial Clock Line).

Nous avons pu mettre en pratique quelques concepts d'électronique de base, pour utiliser la photorésistance nous avons besoin de lire une tension sur une entrée analogique. Nous utilisons donc un pont diviseur de tension où R_1 est la photorésistance qui varie en fonction de la luminosité ambiante. R_2 est la résistance de 10kOhm.



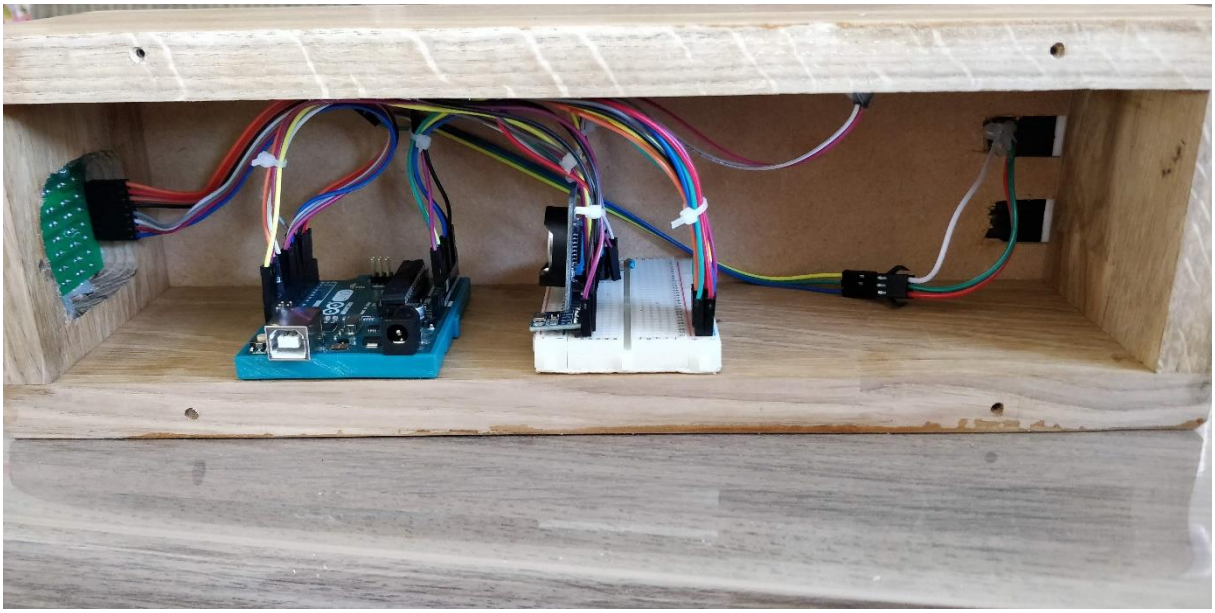
$$\rightarrow U_2 = U\frac{R_2}{R_1+R_2}$$

Nous avons donc la tension U_2 lue par l'Arduino qui est l'image de la luminosité. Si R_1 augmente donc si la luminosité diminue, U_2 diminue.

Source : Wikipédia

Pour plus d'esthétique, nous avons dessoudé et ressoudé les broches de la matrice de boutons afin de les retourner.

Le câblage a été soigné afin d'être le plus compréhensible possible :



C) Programmation

Nous avons utilisé le logiciel Arduino IDE pour réaliser notre programme. L'Arduino se programme en C/C++. Il y a une structure particulière pour le programme. Il y a une fonction *void setup()* qui est exécutée une seule fois au démarrage et ensuite une autre fonction *void loop()* qui est exécutée en boucle. Il y a ensuite toutes les autres fonctions qui sont appelées tout au long du programme. Nous avons eu besoin des librairies suivantes pour pouvoir faire fonctionner tous les composants :

- DHT : pour le capteur de température et d'humidité
- DS3231 : pour le module temps réel
- FastLed : pour la matrice LED
- Adafruit_BMP085 : pour le capteur de pression

- Adafruit_Sensor : pour le capteur de pression

Nous avons réfléchi à la façon la plus simple de pouvoir écrire des chiffres à différents endroits précis de la matrice. Nous avons donc modélisé la matrice LED sur Excel afin d'avoir la position exacte des LEDs à allumer en fonction du chiffre voulu.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	240	239	238	237	236	235	234	233	232	231	230	229	228	227	226	225	224

Nous voyons qu'un chiffre, modélisé par un carré de couleur utilise 35 LEDs. Il y a 4 positions possibles. Nous avons donc défini un tableau (array) de 256 cases pour modéliser la matrice de LEDs. Nous avons créé la fonction *void modifierMatrice(int chiffre, int pos, int afficheur[256])* qui permet de remplir le tableau afficheur passé en paramètre avec le chiffre demandé (qui est créé dans cette fonction dans un tableau de 35 cases) dans la position demandée.

La fonction *void afficherMatrice(int afficheur[256], int r, int g, int b, int mode, int secondes)* permet d'afficher la matrice qui a été remplie précédemment. C'est cette fonction qui communique la couleur de chaque LED au panneau. Les composantes de chaque couleur varient entre 0 et 255. C'est l'unique fonction qui communique avec la matrice de LEDs. C'est dans cette procédure qu'on demande d'afficher les symboles supplémentaires (degrés, humidité…) en fonction du mode ainsi que les secondes. La météo est également gérée ici. Si la pression est inférieure à 1000hPa, c'est qu'il va pleuvoir, si elle est supérieure à 1020hPa, c'est qu'il va faire du soleil et si elle est entre 1000hPA et 1020hPa c'est qu'il va faire un temps nuageux. Nous avons pris ces valeurs en observant les baromètres existants. Ces deux fonctions sont dans le fichier séparé *gerer_matrice.h*.

Dans le fichier principal, nous avons plusieurs fonctions. Nous avons tout d'abord les fonctions suivantes :

- *void afficherHumidite()*
- *void afficherTemperature()*
- *void afficherDate()*
- *void afficherHeure()*
- *void afficherAlarm()*

Ces fonctions récupèrent les données auprès des capteurs et variables et les affichent. Afin d'afficher les nombres, composés de deux chiffres, nous les avons décomposés en deux chiffres distincts avec une succession de conditions afin de gérer les cas possibles. Chaque fonction qui demande un affichage communique les composantes de couleurs à afficher et les diminue en fonction de la luminosité ambiante (avec un seuil minimum de 20%). Ces fonctions exécutent les fonctions de *gerer_matrice.h* en leur indiquant le mode dans lequel on est pour savoir quel(s) symbole(s) afficher.

La fonction *void toucherBouton(int i, int j)* permet d'interagir avec les boutons. Les fonctions qui sont exécutées par ces derniers sont les suivantes :

- *void augmenter(int &r)* : Cette fonction permet d'augmenter par palier de 15 (et repart à 0 au-dessus de 255) la composante de couleur passée en paramètre par référence.
- *void diminuer(int &r)* : Cette fonction permet de diminuer par palier de 15 (et repart à 255 en-dessous de 0 la composante de couleur passée en paramètre par référence.
- *void reset(int &r, int &g, int &b)* : Cette fonction permet de remettre la couleur par défaut des LEDs (blanc avec une luminosité moyenne).
- *void changerFonction(int &fonc)* : Cette fonction permet de passer entre les différents modes d'affichage. Chaque mode est associé à un chiffre, de 0 à 4.
- *void augmenterminute(int& alarmm)* : Cette fonction qui peut seulement s'exécuter lorsque l'on se trouve dans le mode de réglage de l'alarme permet d'augmenter le chiffre des minutes (il repasse à 0 lorsque l'on atteint 60).
- *void diminuerminute(int& alarmm)* : Cette fonction qui peut seulement s'exécuter lorsque l'on se trouve dans le mode de réglage de l'alarme permet de diminuer le chiffre des minutes (il repasse à 59 lorsque l'on passe en dessous de 0).
- *void augmenterheure(int& alarmh)* : Cette fonction qui peut seulement s'exécuter lorsque l'on se trouve dans le mode de réglage de l'alarme permet d'augmenter le chiffre des heures (il repasse à 0 lorsque l'on atteint 24).
- *void diminuerheure(int& alarmh)* : Cette fonction qui peut seulement s'exécuter lorsque l'on se trouve dans le mode de réglage de l'alarme permet de diminuer le chiffre des heures (il repasse à 23 lorsque l'on passe en dessous de 0).
- *void changeretat(int &etatalarm)* : Cette fonction permet de mettre l'alarme en marche ou non.

  La fonction *void alarme(int& etatalarm)* est exécutée pour faire sonner le buzzer. Quand l'alarme sonne et le réveil affiche l'heure actuelle (si l'on se trouve dans un autre mode). Pour la désactiver il faut appuyer sur n'importe quel bouton afin que l'Arduino sorte de la fonction.

  Toutes ces fonctions ne fonctionneraient pas sans les fonctions suivantes :

  void setup() : Cette fonction permet d'activer les capteurs de température, de pression et le module temps réel. Elle permet également d'activer le panneau LED en lui indiquant la broche sur lequel il est connecté, le type de LED, l'ordre des couleurs, le nombre de LED et la luminosité. Elle permet aussi d'initialiser les boutons et de définir le buzzer en tant que sortie pour l'alarme.

  void loop() : Cette fonction est la fonction principale qui est exécutée en boucle. Elle demande l'affichage de telle ou telle information en fonction du mode dans lequel on se trouve. C'est également cette fonction qui appelle la fonction *toucherBouton* en fonction du bouton qui est appuyé. L'appui des boutons est vérifié en continu. La fonction *alarme* est exécutée ici lorsque l'heure actuelle correspond à l'heure programmée pour l'alarme. Elle est exécutée tout le temps que la personne n'ait pas appuyé sur un des boutons du réveil. Dès qu'un bouton est pressé, l'alarme est arrêtée et désactivée.

  Les variables globales, utilisées tout au long du programme sont les suivantes :

  - *CRGB leds[256];* : initialisation du panneau LED
  - *Adafruit_BMP085 bmp;* : initialisation du capteur de pression
  - *DS3231 clock; :* initialisation du module temps réel
  - *DHT dht(6, DHT11);* : initialisation du capteur de température et d'humidité sur le pin 6
  - *RTCDateTime dt;* : initialisation de la variable de sortie du module temps réel

- *int r = 135, g = 135, b = 135, fonc = 0, alarmh = 0, alarmm = 0, etatalarm = 0;* : initialisation des variables au démarrage. La couleur est définie sur un blanc atténué, l'alarme est désactivée (et placée arbitrairement sur 00:00) et l'heure est affichée par défaut.

- *int *afficheur = new int[256];* : initialisation du tableau représentant le panneau de LEDs. Il est passé en pointeur afin qu'il puisse être modifiable dans toutes les procédures du programme.

- *int rangee[] = {2, 3, 4, 5};*
- *int colonne[] = {8, 9, 10, 7};*
- *int col_scan;*

: initialisation de la matrice de boutons (les 8 broches de connexion ainsi que la variable de détection)

D) Finition du produit

Nous voulions que notre réveil puisse s'adapter dans tout type de décoration. Nous avons donc choisi de faire un boitier en chêne. Nous avons découpé six planches à l'aide d'une scie plongeante. Cela conserve un style contemporain tout en restant sobre. Nous avons tout d'abord commencé par réfléchir à la meilleure manière pour rigidifier le panneau LED et le faire tenir droit. La meilleure manière était de le coincer entre une plaque de plexiglass et une plaque de MDF de 3mm. Afin de diffuser la lumière nous avons ajouté une feuille de papier entre le panneau LED et le plexiglass. L'épaisseur total est alors de 10mm. Afin de pouvoir le faire tenir droit sans qu'il bouge nous avons à l'aide d'une défonceuse et d'une fraise droite de 10mm, fait une rainure tout le tour du réveil afin de glisser l'ensemble dedans. Ensuite nous avons voulu incruster les boutons. Nous avons commencé par découper à la scie sauteuse l'espace nécessaire pour passer les boutons. Ensuite, la carte de prototypage des boutons étant un peu plus grande que l'espace occupé par ces derniers, nous avons fraisé une avancée dans le bois afin d'avoir les boutons en surface. Nous avons aussi gravé la correspondance des boutons à l'aide d'une graveuse laser. Ensuite, nous avons percé un trou de 5mm sur le haut du réveil afin de passer la photorésistance. C'est l'endroit le plus propice pour capter la meilleure lumière possible. Nous avons ensuite percé trois trous dans la face arrière. Ces trous ont une double utilité : tout d'abord pouvoir faire passer le câble d'alimentation de l'Arduino mais également de pouvoir faire circuler l'air pour avoir une meilleure précision possible pour les capteurs de température, d'humidité et de pression. La face arrière est vissée afin d'être démontée facilement s'il y a une opération de maintenance à faire sur l'Arduino. Le reste est collé et emboité à l'aide du système Lamello©.

Système Lamello©

Les boutons, la photorésistance, la carte Arduino ainsi que la carte de prototypage ont été collés sur le bois. Les arêtes du bois ont été chanfreinées avec une défonceuse afin d'avoir une meilleure esthétique. Nous avons fini par protéger le réveil et le rendre brillant à l'aide d'une huile pour bois. Voici le rendu final :

E) Répartition des tâches

Concernant la répartition des tâches, nous nous sommes concentrés chacun sur les domaines que nous connaissions le mieux, tout en s'entraidant bien sûr. Nous avons commencé tous les deux par réfléchir sur les fonctionnalités à adopter. Ensuite, Paul s'est plus préoccupé de la partie programmation et de la finition du réveil pour l'esthétique, Jimmy s'est consacré principalement de la partie câblage et de la compatibilité des modules entre eux.

Jimmy a donc principalement amélioré ses compétences en C++ alors que Paul a principalement amélioré ses connaissances sur la partie hardware.

Le code source ainsi que le câblage restent tout de même une création du groupe car nous avons mis en communs nos idées afin de produire un code source complet et optimisé et un câblage le plus compact possible.

F) Mode d'emploi

L'interaction avec l'utilisateur est réalisée par la matrice de boutons, ce qui permet de contrôler plusieurs fonctions sur un espace réduit. Nous avons assigné les boutons comme suit :

HEURE -	HEURE +	VERT -	VERT +
MINUTE -	MINUTE +	ROUGE -	ROUGE +
ALARME ON/OFF		BLEU -	BLEU +
		RESET COULEURS	MODE

Cette correspondance avec les boutons a été reportée sur le côté du réveil à côté de ces derniers à l'aide d'une graveuse laser.

Le bouton Mode permet de changer entre les modes suivants :

- Heure (affichage de l'heure et des secondes, un point correspond à deux secondes, avec une distinction de couleur en fonction des secondes paires ou impaires)
- Date
- Alarme
- Température
- Humidité

- Les boutons Vert -, Vert +, Rouge -, Rouge +, Bleu -, Bleu + permettent de régler la couleur d'affichage de l'écran. L'appui sur un des boutons augmentera ou diminuera l'intensité des composantes couleurs (la luminosité est gérée automatiquement en fonction de la lumière ambiante). Reset Couleurs remettra l'affichage en blanc par défaut.

- Les boutons Heure +, Heure -, Minute +, Minute - permettent lorsque l'on est sur le mode Alarme de programmer l'heure souhaitée.
- Le bouton Alarme ON/OFF permet d'activer et de désactiver l'alarme, l'appui peut être effectué tout le temps, peu importe dans quel mode on se trouve. L'état sera également notifié sur chaque mode du réveil.
- Lorsque l'alarme retentit, la désactivation se fait par une secousse du réveil

La météo (soleil, nuage ou pluie) ainsi que l'état de l'alarme sont affichées dans chaque mode.

G) Diffusion du projet

Ce projet est publié sous licence GNU GPLv3. Il peut être modifié, réutilisé, partagé et redistribué à n'importe quelles fins. Nous pensons que notre travail peut être utile à tous ceux qui souhaitent visualiser notre projet et y apporter des modifications pour l'améliorer ou créer de nouvelles fonctionnalités.

Nous avons également décidé de publier sur GitHub les sources ainsi qu'un tutoriel concis pour donner la possibilité même aux débutants qui veulent réaliser ce projet. Afin de toucher un plus grand public nous avons pris l'initiative de l'écrire en anglais ainsi qu'en français. Les personnes ayant déjà des connaissances sont également concernées car un des points fort de notre projet est qu'il est totalement modifiable et surtout laisse la possibilité d'ajouter sa touche personnelle et ses propres fonctionnalités.

QR Code vers le dépôt GitHub

https://github.com/Boogy62/ArduinoClock

Conclusion

Durant la réalisation de notre projet, de la conception du besoin et l'élaboration du cahier des charges jusqu'à la réalisation du produit fini, nous avons rencontré peu d'obstacles. Un des problèmes que nous avons pu corriger était le fait que les LEDs scintillaient. Nous mettions à jour la matrice avec les chiffres puis nous affichions, en dernier, les symboles. Maintenant tout est affiché en une seule fois. La matrice se met à jour à chaque tour de la fonction principale (donc plusieurs fois par seconde). Désormais, le fait qu'il y ait une unique mise à jour dans la fonction d'affichage permet d'éviter le scintillement. Certaines fonctions n'ont pas pu être réalisées à temps comme la partie enceinte connectée et d'autres ne sont pas possibles sans modification matérielle comme la réception de données venant d'internet qui nécessiterait l'utilisation d'une carte plus performante. Nous avons également quelques idées d'amélioration hors cahier des charges comme par exemple la rotation automatique de l'affichage en fonction du sens dans lequel le réveil est posé.

Une amélioration possible serait par exemple une meilleure gestion de la prévision de météo, avec plus d'états possibles (comme l'orage ou la tempête) par le biais des tables de Moreux mais cela nécessiterait de connaitre le sens du vent. Le cahier des charges est donc bien respecté. Nous pourrions avoir une meilleure précision sur la température et l'humidité en prenant un capteur DHT22 par exemple. Nous avons même fait mieux que le cahier des charges sur la gestion des couleurs et de l'intensité lumineuse. Nous pouvons créer 5832 couleurs différentes (18^3) alors que nous n'en voulions que 10 à la base. De plus, la luminosité de l'affichage s'adapte à l'éclairage ambiant avec une latence presque nulle alors que nous voulions moins de 2 secondes.

Ce projet nous a donc donné un aspect de la gestion d'un projet de A à Z. Ayant travaillés qu'à deux, l'aspect du travail d'équipe était assez limité et donc travailler dans un plus grand groupe serait une envie à combler.