

# **Projet EDACUB (Enregistreur de Données Autonome à Consommation Ultra Basse)**

**Projet E.6.2 (BTS SN2)**

**Session 2023**

**Revue de Projet 2**

---

**Lycée Alphonse Benoît**

Pomaro Brice

Layouni Noa

Guillot Kilian

Tnifas Abdessalam



Projet EDACUB (Enregistreur de Données Autonome à Consommation Ultra Basse) .....	1
PRÉSENTATION GÉNÉRALE DU SYSTÈME .....	5
A quoi sert le projet EDACUB ? .....	5
Objectif du Système .....	6
Choc .....	6
Diagrammes Sysml généraux.....	7
Diagramme d'exigences .....	7
Diagramme de cas d'utilisation .....	8
Architecture matérielle logicielle.....	9
Matériel utilisé cette année pour le projet.....	9
Choix de la STM32.....	10
Moyens de Communications.....	11
Organisation .....	11
Partie EC 1 : GUILLOT Kilian .....	12
Introduction :.....	12
Présentation du projet : .....	12
Objectifs de la revue de projet : .....	12
Présentation de la partie du candidat : .....	13
Tableau récapitulatif de mes tâches :.....	14
Planification Prévisionnelle : .....	14
Mise en œuvre du projet :.....	16
Schéma structurel :.....	16
IBD du système EDACUB :.....	17
Partie Accéléromètre :.....	18
Voici le schéma structurel :.....	19
Voici le programme en question : .....	19
Partie Microcontrôleur : .....	22
Partie Alimentation : .....	23
Partie Information visuelle :.....	26
Partie Information branchement :.....	27
Système de détection de la carte SD :.....	27
Partie saisie de schéma Kicad : .....	28
Liste des composants utilisés : .....	30

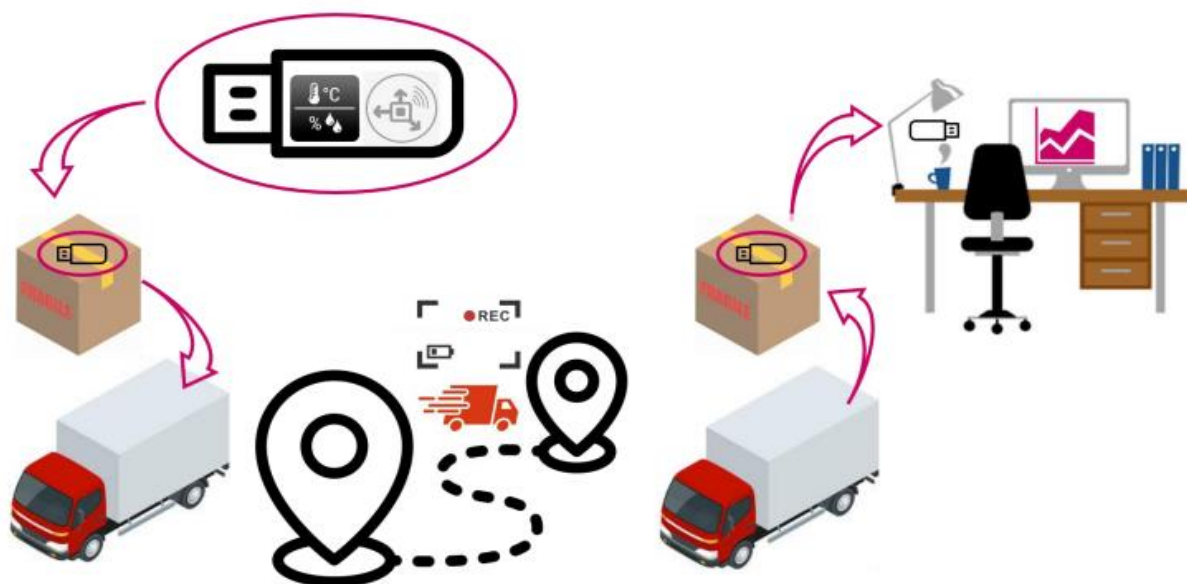
Matériel utilisé lors de l'assemblage de la carte : .....	31
Partie IR 1 : POMARO Brice .....	34
Présentation du rôle du candidat .....	34
Objectifs principaux.....	34
Planification prévisionnelle.....	35
Diagrammes SysML individuels.....	36
Qu'est-ce que STM32 ? .....	38
Compte rendu d'activités - Revue de projet 1 .....	41
Capteur choc .....	41
Capteur Température .....	45
Ajustement de durée de sommeil en cas de détection de choc(s).....	46
Sauvegarde des données dans une carte SD externe.....	50
Câblage – Partie « physique » .....	51
Conclusion.....	52
Partie EC2 : TNIFAS Abdessalam.....	53
Présentation du rôle du candidat .....	53
L'IBD de la carte : .....	54
Prise en main du logiciel «STM32Cube IDE » : .....	55
Pourquoi avoir choisi la STM32F11RE? .....	56
Test de la carte en faisant clignoter la LED CMS : .....	56
Hold Master et No Hold Master:.....	58
Code pour obtenir la température : .....	58
Code pour obtenir l'humidité: .....	59
Schéma Fritzing: .....	61
Relevé de trame:.....	62
Vérification avec l'appareil qui mesure l'humidité relative, .....	65
la température ainsi que le CO2 : .....	65
Journal de bord: .....	71
Planification Partielle:.....	72
Planification Réelle : .....	73
Schéma structurel du projet : .....	74
Routage:.....	75
Routage de la carte couche top: .....	75
Routage de la carte couche bottom: .....	75
Affichage 3D: .....	75
La Soudure : Les étapes.....	76

1 <sup>ère</sup> étape : Liste de matériel et distribution du matériel.....	76
2 <sup>ème</sup> étape : Nettoyage de la carte avec de l'alcool isopropylique.....	77
3 <sup>ème</sup> étape : Association du stencil sur le PCB.....	77
4 <sup>ème</sup> étape : Enduire la pâte à braser .....	78
5 <sup>ème</sup> étape : Placement des composants CMS .....	78
6 <sup>ème</sup> étape : Vérification .....	78
7 <sup>ème</sup> étape : Passage au four à refusions.....	79
Nos recherches concernant les composants au début du projet : .....	79
Annexe des composants si vous êtes intéressés (ils ont peut être expirés): .....	80
Conclusion : .....	81
PARTIE IR2 : LAYOUNI Noa.....	82
Introduction.....	82
Objectif personnel .....	83
J'ai donc émis à l'intelligence artificielle les instructions suivantes :.....	83
Voici le code exemple de ChatGPT : .....	83
Résultat : .....	86
Page web.....	87
Contraintes des chocs :.....	87
Contraintes de la température et de l'humidité : .....	87
Contraintes de la date : .....	88
Choix de la librairie.....	88
Comparatif des librairies JS .....	88
Highcharts.js : .....	88
D3.js : .....	89
Étude du code des courbes.....	90
Site Web .....	92
Site Web avec serveur .....	92
Site Web sans serveur .....	93
Interface Web .....	93
Planification prévisionnelle.....	95

# PRÉSENTATION GÉNÉRALE DU SYSTÈME

## A quoi sert le projet EDACUB ?

Le projet **EDACUB** consiste à réaliser un dispositif, sous forme d'une « **clé USB** », permettant de mesurer et enregistrer les conditions environnementales d'un colis lors de son acheminement tels que la température, l'humidité relative et les chocs durant une longue période afin d'assurer la traçabilité du produit lors de son transport et de son envoi.



La clé USB récupère les données des capteurs via une communication I2C puis les transmet à l'écran de l'ordinateur sur lequel la clé sera branchée.

L'entreprise **Metraware** qui se situe au 3 Impasse du Luberon, 84240 Cabrières-d'Aigues, nous demande de mettre en œuvre une clé USB qui puisse détecter les **chocs**, c'est à dire un suivi pour enregistrer les données sur une longue période (72h) par rapport aux conditions de transports de colis fragiles et dans un second temps rendre ces données consultables à travers une interface WEB (servie sans serveur).

Ce dispositif permet alors la traçabilité des colis durant leur acheminement en offrant la possibilité de suivre les conditions de transport.

Afin de satisfaire la demande de Metraware, nous sommes donc une équipe de 4 personnes (2 IR, 2 EC) pour réaliser et mettre en œuvre le projet EDACUB.

## Objectif du Système

- Enregistrer des informations telles que :
  - L'Humidité relative
  - La Température
  - Les chocs
  
- Avoir accès à ces informations sur l'ordinateur, cela permet d'être averti en cas de colis subissant un éventuel choc, une hausse de température ou une humidité anormale.

Ce système permet en résumé d'avertir l'utilisateur des éventuels aléas durant le transport.

## Choc

### *Définition d'un choc :*

Un choc est la rencontre plus au moins violente et brusque entre deux corps ou plusieurs corps en mouvement.

Maintenant essayons de définir le niveau d'un choc.

Mouvements	Accélération en g, en comptant pour 1 g la gravité terrestre
Train ou métro en démarrage ou freinage normal	+/- 0,1 à 0,2, jusqu'à (-)0,5 en fort freinage
Départ d'un avion de ligne Airbus A320	0,4
Départ de montagnes russes lancées	0,5 à 3,3
Attraction « Le Rotor » (force centrifuge)	1,5
Une Bugatti Veyron passant de 0 à 100 km/h en 2,4 s.	1,55 <sup>d</sup>
Un patineur de vitesse courte piste (virage de 9m de rayon à 54 km/h)	2,6
Record mondial de dragster Top Fuel <sup>e</sup> avec ¼ mile (402,25 m) couvert en 4,4 s.	4,2
Formule 1, décélération maximale avec un freinage brusque.	≥ 5
Luge, maximum relevé au Whistler Sliding Centre.	5,2
Formule 1, accélération latérale maximale en virage <sup>f</sup> .	5 à 6
Virage le plus serré caractéristique dans un avion de voltige ou dans un avion de chasse.	9 à 12 en linéaire (environ)
Record de résistance humaine de John Stapp dans un chariot propulsé à l'aide de fusées.	46,2
Mort ou lésions sévères probables.	> 50
Missile antibalistique Sprint.	100
Exposition très brève d'un homme au moment d'un crash <sup>14</sup> .	> 100
Record de résistance humaine (Kenny Bräck, 2003, lors d'une course automobile, il est propulsé dans les grillages).	214
Canon spatial avec un fût de 1 km et une vitesse à la bouche de 6 km/s (en supposant une accélération constante).	1 800
Résistance au choc d'une montre de poignet <sup>20</sup> .	> 5 000
Accélération maximale d'un piston dans le moteur d'une Formule 1 standard <sup>21</sup> .	8 600
Circuits électroniques de classe militaire destinés aux obus d'artillerie <sup>22</sup> .	15 500
Balle 9 × 19 mm Parabellum (valeur moyenne sur la longueur du canon) <sup>9</sup> .	31 000
Balle 9 × 19 mm Parabellum à son pic d'accélération <sup>h</sup> .	190 000
Accélération moyenne d'un proton dans le Large Hadron Collider (Grand collisionneur de hadrons) <sup>23</sup> .	190 000 000
Accélération avec l'accélérateur laser-plasma <sup>24</sup> .	8,9 × 10 <sup>20</sup>

# Diagrammes Sysml généraux

## Diagramme d'exigences

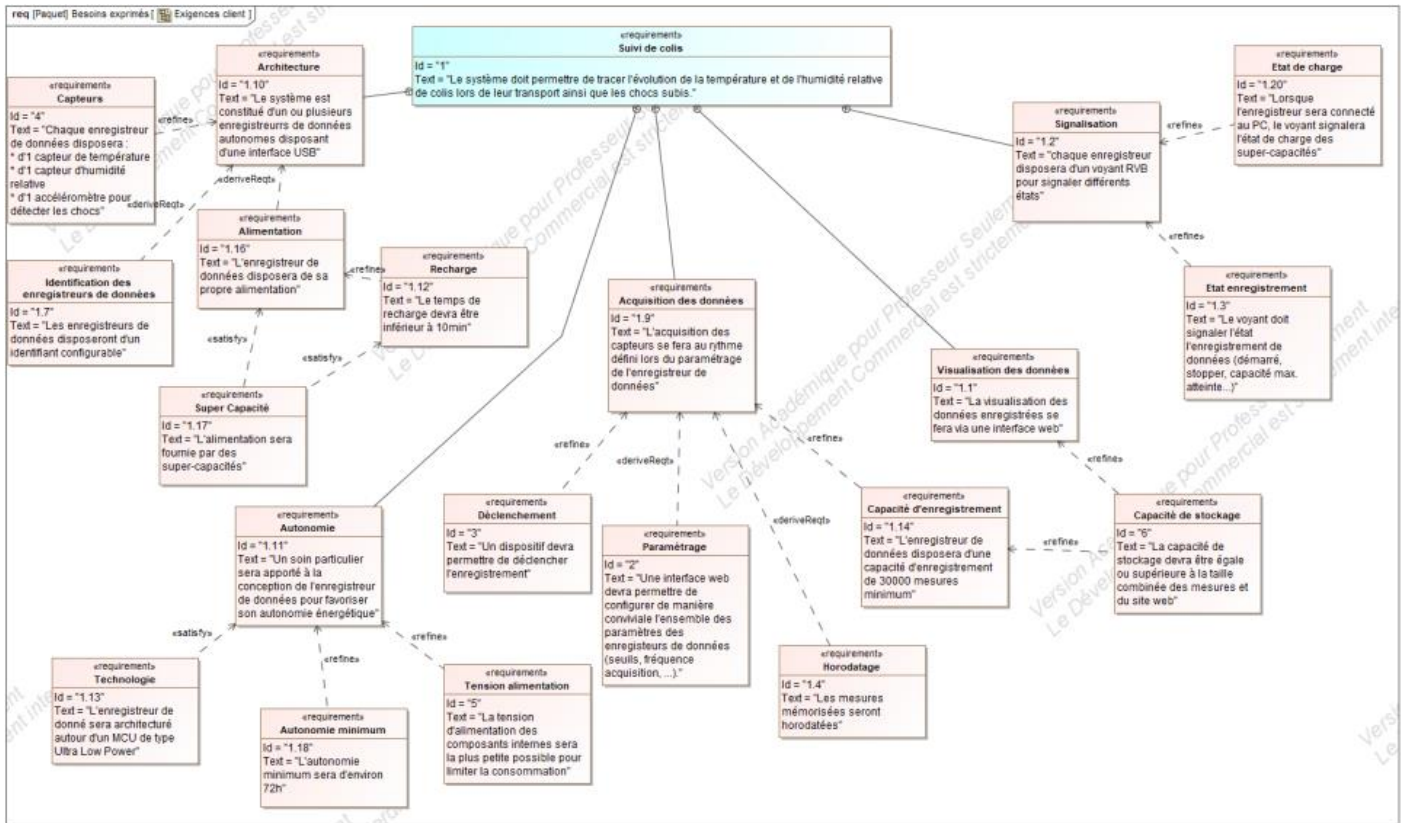
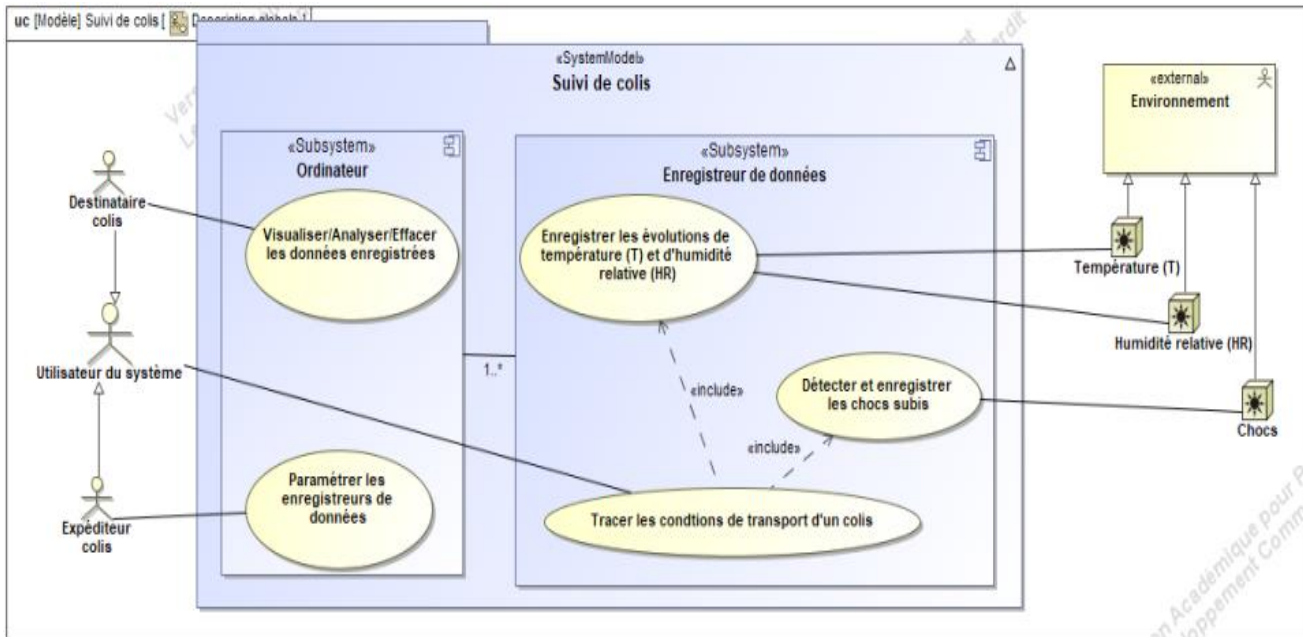
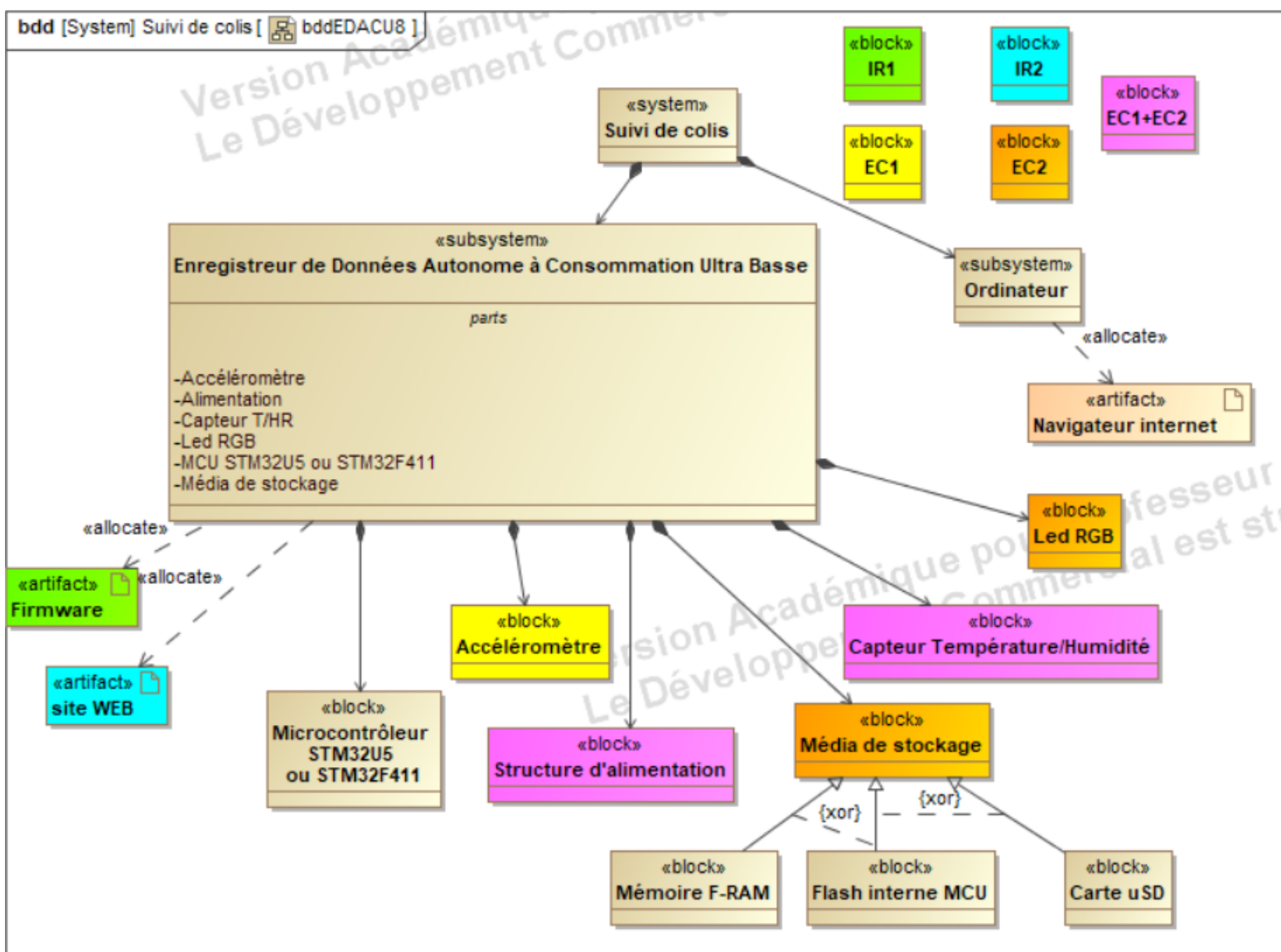


Diagramme de cas d'utilisation





## Architecture matérielle logicielle



## Matériel utilisé cette année pour le projet

- Carte NUCLEO-F411RE 
- Accéléromètre ADXL375 
- Capteur SHT21 
- Mémoire FRAM 
- Quartz 
- Super condensateur 
- Bouton poussoir 

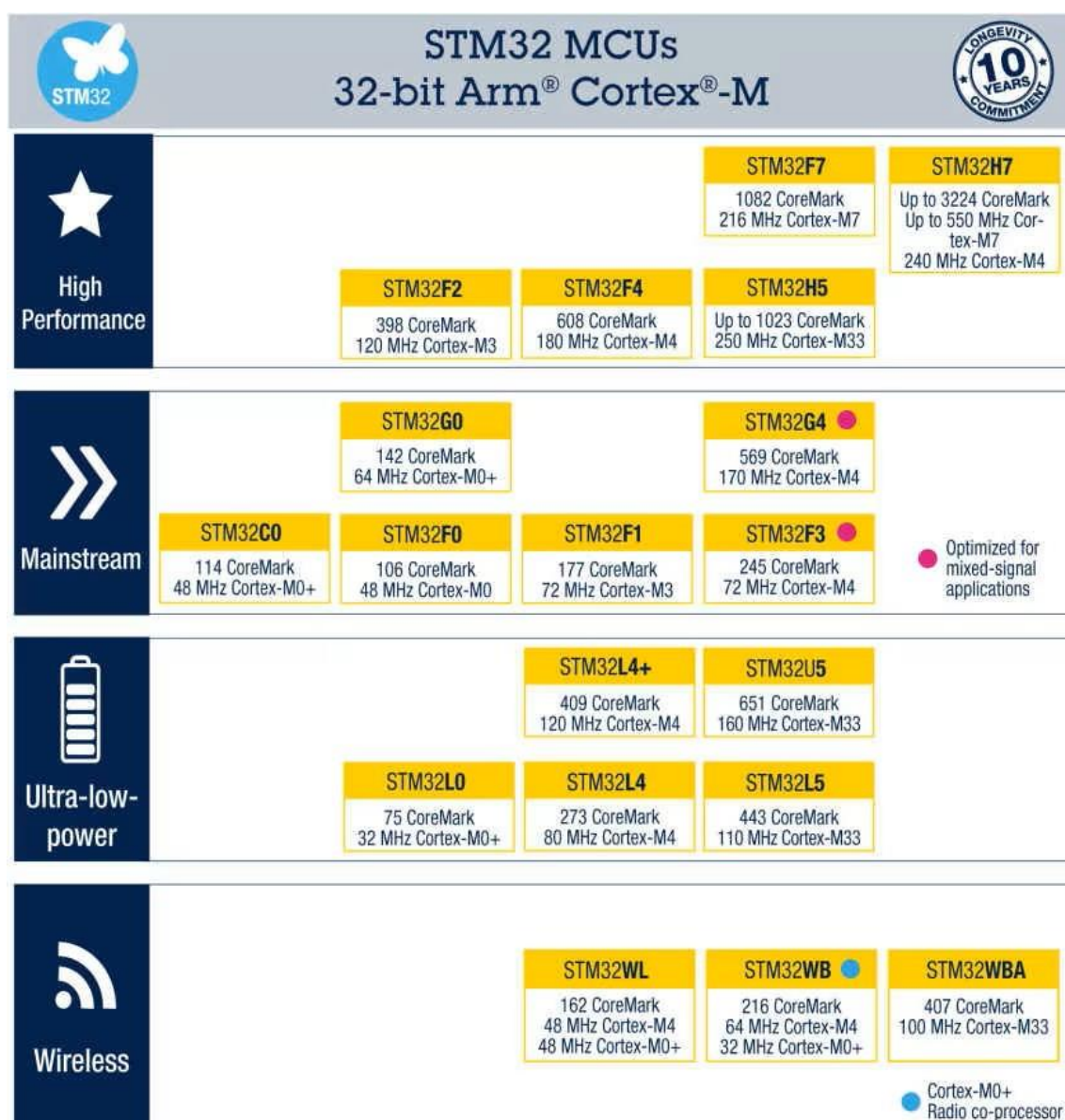
## Choix de la STM32

Pour notre STM32, nous devons initialement en choisir une de la gamme « **Ultra Low Power** » comme indiqué précédemment dans ma revue en raison de sa faible consommation qui en fait sa force, afin de pouvoir respecter le cahier des charges (*voir diagramme d'exigences page 6 bloc Autonomie – Autonomie minimum*) qui spécifie donc 72h minimum environ d'autonomie.

Le choix se portait sur la **STM32U5**, issue donc de la gamme « Ultra Low Power » mais n'a finalement pas été retenue en vue du faible contenu de la communauté et des forums sur cette carte, ne donnant pas suffisamment d'informations ou d'aide concernant les difficultés qui peuvent être rencontrées.

Notre choix va donc finalement se porter sur la **STM32F11RE** de la gamme « **High Performance** » (*cf image ci-dessous*) qui possède une communauté largement plus active que la **STM32U5** avec des bibliothèques **OpenSource** très utiles à notre projet.

Nous allons en revanche devoir sacrifier une exigence du projet qui concerne l'autonomie de 72h qui nous sera probablement plus possible d'assurer.



## **Moyens de Communications**

Pendant le projet notre moyen principal de communication est un Drive que nous avons créé via la plateforme gratuite Google Drive qui nous permet de tenir un Journal de Bord commun et de nous partager les fichiers importants et nos trouvailles.



Pour la communication avec les professeurs et les collègues nous utilisons Discord.



## **Organisation**

Souvent nous écrivons à l'avance ce que nous voulons faire dans le journal de bord et à la fin de la séance nous rajoutons ou enlevons des actions que nous avons réalisées.

Nous avons réalisé une planification avec l'aide de Microsoft Project.

# Partie EC 1 : GUILLOT Kilian

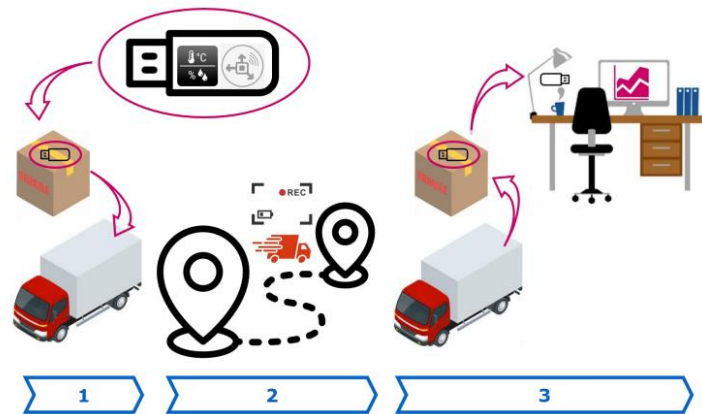
## Introduction :

### Présentation du projet :

De nos jours beaucoup de personnes commandent des produits sur internet, il est donc normal de se dire si notre colis a été transporté dans les normes, ils peuvent mettre plusieurs jours à arriver jusqu'à notre domicile.

Ce suivi est alors un synonyme de transparence, de qualité, de sécurité et ou de maîtrise ; le projet EDACUB est là pour répondre à ce besoin.

Le projet EDACUB (Enregistreur de Données Autonome à Consommation Ultra Basse) a pour but de concevoir un dispositif qui ressemble à une clé USB qui permet d'enregistrer des températures, de l'humidité, des chocs pour assurer le suivi des conditions de transport de colis.



### Objectifs de la revue de projet :

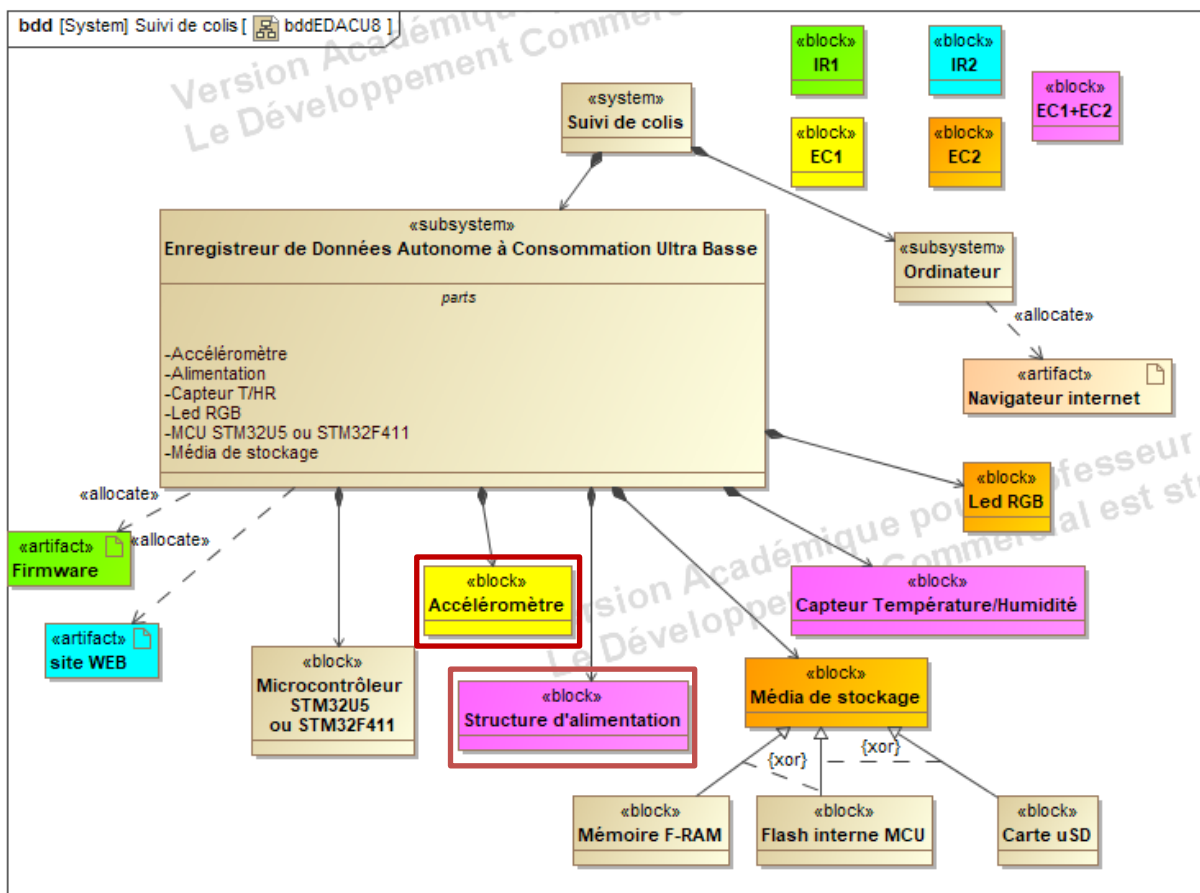
Dans cette revue de projet nous allons faire un point sur ce qui a été fait depuis l'ancienne revue, ce qui a été amélioré, ce qui a été gardé et ce qui a été changé, pour, par la suite, améliorer certains points suite à vos interventions.

Présentation de la partie du candidat :

Tout d’abord, l’entreprise Metraware qui se situe au 3 Impasse du Luberon à Cabrières d’Aigues, nous a demandé de mettre en œuvre un dispositif permettant d’enregistrer la température et l’humidité à l’intérieur des entrepôts lors de son stockage, et les chocs lors d’un voyage d’un colis.

Afin de pouvoir satisfaire leur demande, nous sommes avec mon équipe et moi sur le projet EDACUB.

Il m’a été donné comme tâche de réaliser un détecteur de choc qui serait dans un mode où il consommerait un minimum d’énergie pour permettre une grande autonomie, et qu’il puisse réveiller le microcontrôleur quand il détecte un choc.



= Ma partie

## Tableau récapitulatif de mes tâches :

Étudiant n°3  EC1	<b>Détecter les chocs et assurer la recharge du super condensateur</b> <ul style="list-style-type: none"> <li>Prendre en main l'outil de développement et son environnement.</li> <li>Prise en charge des structures de détection de chocs et de recharge d'un super condensateur.</li> <li>Travail en collaboration avec l'étudiant EC2 pour effectuer des mesures de consommation précises, permettant le dimensionnement du super condensateur, et avec l'étudiant IRI pour la gestion des chocs, et les configurations des entrées sorties pour minimiser la consommation.</li> <li>Production d'un schéma individuel puis d'un schéma de synthèse avec l'étudiant EC2.</li> <li>Effectuer un routage individuel de cette carte et produire les fichiers afin que la fabrication du PCB soit sous-traitée.</li> <li>Câbler la carte et effectuer les essais.</li> <li>Documenter la mise en service de la carte finalisée.</li> </ul>	<b>Installation/Configuration :</b> <ul style="list-style-type: none"> <li>Carte NUCLEO dans l'environnement de développement Stm32CubeIDE.</li> </ul> <b>Mise en œuvre :</b> <ul style="list-style-type: none"> <li>Choisir et tester un accéléromètre, adapté au projet. Le mettre en œuvre sur l'outil de développement.</li> <li>Définir un seuil au-delà duquel le choc sera pris en compte par le datalogger.</li> <li>En lien avec l'étudiant EC2, effectuer des mesures de consommation de l'ensemble des composants constituant la carte. Puis choisir en commun un super condensateur (capacité, tension, dimensions, ...).</li> <li>Proposer/tester/valider une structure permettant d'assurer la recharge du super condensateur à partir du port USB d'un PC.</li> <li>Proposer un schéma structurel des parties que vous avez en responsabilité.</li> </ul> <b>Réalisation :</b> <ul style="list-style-type: none"> <li>Après validation des schémas individuels, proposer avec l'étudiant EC2 un schéma de synthèse.</li> <li>Concevoir individuellement un circuit imprimé devant être fabriqué industriellement.</li> </ul> <b>Documentation :</b> <ul style="list-style-type: none"> <li>Schéma de câblage rapide (Fritzing) pour documenter la phase d'essais.</li> <li>Documents de fabrication de la carte (KiCAD). Ces documents devront avoir un niveau de qualité permettant une fabrication industrielle du circuit imprimé.</li> <li>Schéma structurel avec contours IBD.</li> <li>Liste complète des composants avec leur source d'approvisionnement, code commande et prix.</li> <li>Programme en C/C++ accompagné des commentaires et diagrammes nécessaires à sa compréhension.</li> <li>Fiche de mise en service.</li> <li>Fiche de dépannage.</li> </ul>
-------------------------	---	---

## Planification Prévisionnelle :

Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
<b>Planifiées manuellement</b>	<b>Projet EDACUB</b>	<b>261 heures</b>	<b>Mar 03/01/23</b>	<b>Mar 30/05/23</b>	
Planifié automatiquement	Lecture du contrat	12 heures	Mar 03/01/23	Lun 09/01/23	
Planifié automatiquement	Planification	12 heures	Mar 10/01/23	Lun 16/01/23	2
Planifié automatiquement	Prise en main de l'outil de développement stm32	12 heures	Mer 08/02/23	Mer 15/02/23	3
<b>Planifié automatiquement</b>	<b>Mise en œuvre de la structure du système de détection de chocs</b>	<b>46 heures</b>	<b>Mer 15/02/23</b>	<b>Mar 14/03/23</b>	<b>4</b>
Planifié automatiquement	Lecture de la documentation de l'accéléromètre	10 heures	Mer 15/02/23	Mar 21/02/23	
Planifié automatiquement	Mise en œuvre d'un programme utilisant l'accéléromètre ADXL375	24 heures	Mar 21/02/23	Mar 07/03/23	6
Planifié automatiquement	Tester l'accéléromètre ADXL375	12 heures	Mar 07/03/23	Mar 14/03/23	7

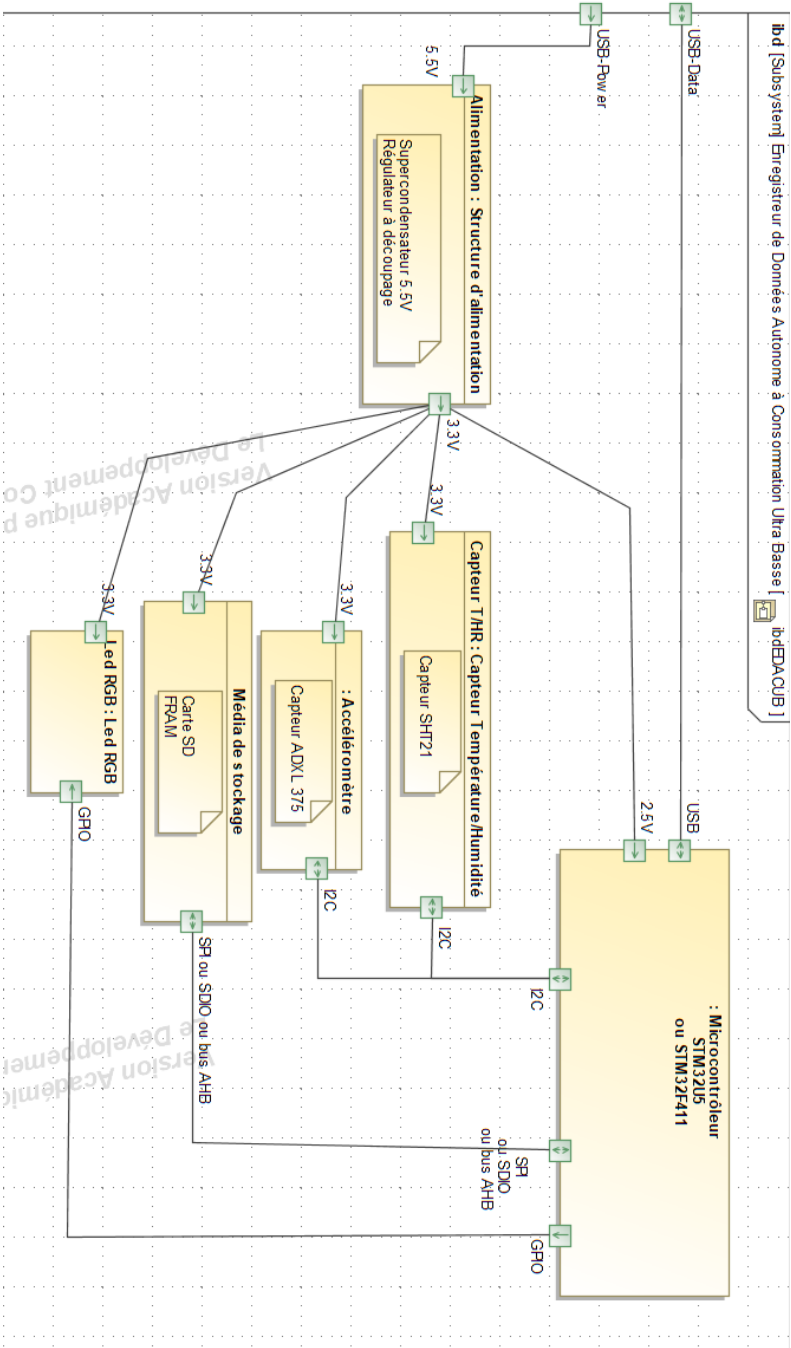
<b>Planifié automatiquement</b>	<b>Mise en œuvre du système de recharge</b>	<b>76 heures</b>	<b>Mar 14/03/23</b>	<b>Mer 26/04/23</b>	<b>8</b>
Planifié automatiquement	Proposition d'un schéma structurel	10 heures	Mar 14/03/23	Mar 21/03/23	
Planifié automatiquement	Test la structure	10 heures	Mar 21/03/23	Lun 27/03/23	10
Planifié automatiquement	Tester la structure de recharge du système	12 heures	Lun 27/03/23	Lun 03/04/23	11
Planifié automatiquement	Validation de la structure	10 heures	Lun 03/04/23	Lun 10/04/23	12
Planifié automatiquement	Mise en œuvre de schéma structurel des différents composants	10 heures	Lun 10/04/23	Mer 12/04/23	13
Planifié automatiquement	Modification des éventuels erreurs	24 heures	Mer 12/04/23	Mer 26/04/23	14
<b>Planifié automatiquement</b>	<b>Réalisation d'un circuit imprimé</b>	<b>56 heures</b>	<b>Mer 26/04/23</b>	<b>Mar 30/05/23</b>	<b>15</b>
Planifié automatiquement	Saisie de schéma du circuit imprimé	24 heures	Mer 26/04/23	Mer 10/05/23	
Planifié automatiquement	Routage du circuit	12 heures	Mer 10/05/23	Mer 17/05/23	17
Planifié automatiquement	Câblage du circuit imprimé	10 heures	Mer 17/05/23	Mer 24/05/23	18
Planifié automatiquement	Test du circuit imprimé	10 heures	Mer 24/05/23	Mar 30/05/23	19





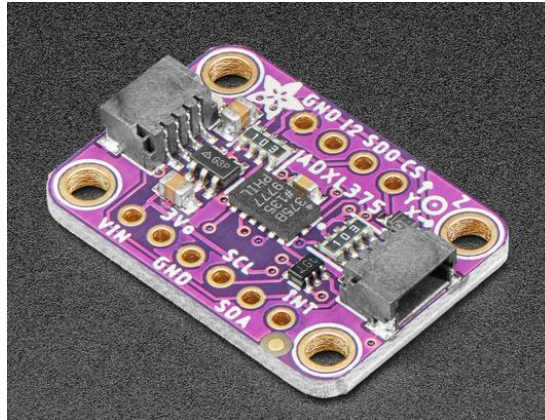


IBD du système EDACUB :



### Partie Accéléromètre :

Dans ma partie j'ai dû mettre en œuvre un dispositif permettant de détecter lorsque notre colis se prend un choc, pour réaliser cela j'ai donc conçu un programme avec un accéléromètre.



Pour me permettre de réaliser ce programme je me suis déjà penché sur la documentation de ce composant, nous avons préféré choisir l'accéléromètre ADXL 375 car il a une gamme de valeur plus grande pour nous permettre de faire ce que nous voulions.

Par la suite grâce aux documentations données par les professeurs, j'ai pu alors utiliser un certain programme de l'ADXL345 et d'une librairie écrite par l'auteur ([dépôt wollewald/ADXL345\\_WE Github](https://github.com/wollewald/ADXL345_WE)), j'ai pu alors utiliser le programme qui s'appelle auto sleep qui permet de faire des mesures lorsqu'il est dans ce mode, de plus il permet de faire une interruption pour réveiller le microcontrôleur.

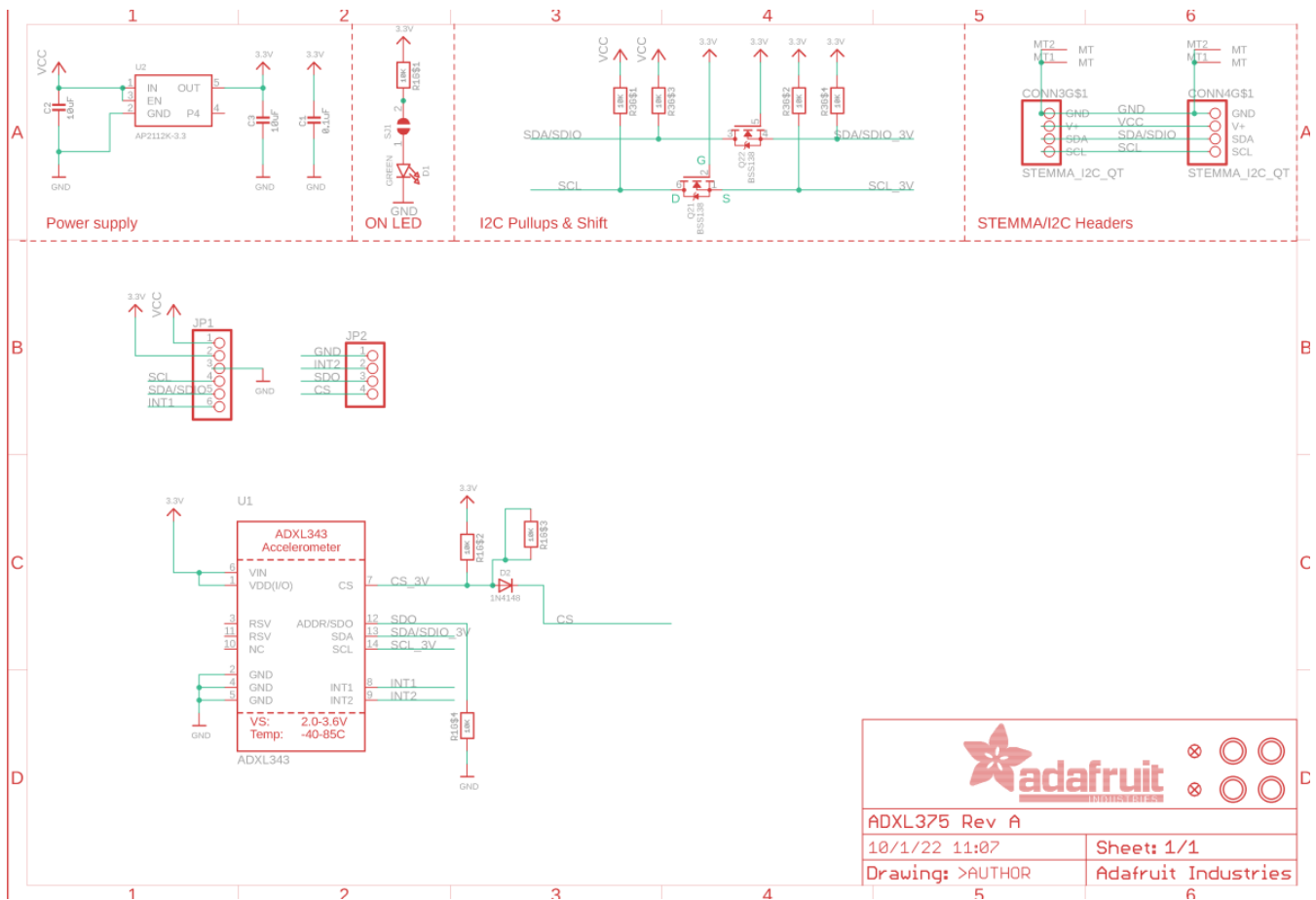
Nous avons préféré prendre l'ADXL 345 car l'ADXL 375 avait une trop grande gamme de valeur pour l'utilisation que nous faisons.

De plus, il y a plus de documentation sur l'ADXL 345, sur Arduino par exemple il y a plus d'exemple de programme que pour l'ADXL 375.

Lors de mes tests, j'ai pu utiliser ce composant avec une carte Arduino, grâce au level Shifter la communication en I2C est possible avec la carte Arduino.

Ce mode permet aussi d'avoir la consommation d'énergie pour avoir une grande autonomie pour notre système.

Voici le schéma structurel :



Voici le programme en question :

```
#include<Wire.h>
#include<ADXL345_WE.h>
#define ADXL345_I2CADDR 0x53 // 0x1D if SDO = HIGH
const int int2Pin = 2;
volatile bool in_activity = false;
/* There are several ways to create your ADXL345 object:
 * ADXL345_WE myAcc = ADXL345_WE() -> uses Wire / I2C Address = 0x53
 * ADXL345_WE myAcc = ADXL345_WE(ADXL345_I2CADDR) -> uses Wire / ADXL345_I2CADDR
 * ADXL345_WE myAcc = ADXL345_WE(&wire2) -> uses the TwoWire object wire2 / ADXL345_I2CADDR
 * ADXL345_WE myAcc = ADXL345_WE(&wire2, ADXL345_I2CADDR) -> all together */
ADXL345_WE myAcc = ADXL345_WE(ADXL345_I2CADDR);

void setup() {
  Wire.begin();
  Serial.begin(9600);
  pinMode(int2Pin, INPUT);
  Serial.println("ADXL345_Sketch - Auto Sleep");
  Serial.println();
  if (!myAcc.init()) {
    Serial.println("ADXL345 not connected!"); }
  /* Insert your data from ADXL345_calibration.ino and uncomment for more precise results */
  // myAcc.setCorrFactors(-266.0, 285.0, -268.0, 278.0, -291.0, 214.0);
  /* Choose the data rate Hz
  ADXL345_DATA_RATE_3200 3200
  ADXL345_DATA_RATE_1600 1600
```

```

ADXL345_DATA_RATE_800 800
ADXL345_DATA_RATE_400 400
ADXL345_DATA_RATE_200 200
ADXL345_DATA_RATE_100 100
ADXL345_DATA_RATE_50 50
ADXL345_DATA_RATE_25 25
ADXL345_DATA_RATE_12_5 12.5
ADXL345_DATA_RATE_6_25 6.25
ADXL345_DATA_RATE_3_13 3.13
ADXL345_DATA_RATE_1_56 1.56
ADXL345_DATA_RATE_0_78 0.78
ADXL345_DATA_RATE_0_39 0.39
ADXL345_DATA_RATE_0_20 0.20
ADXL345_DATA_RATE_0_10 0.10 */
myAcc.setDataRate(ADXL345_DATA_RATE_25);
Serial.print("Data rate: ");
Serial.print(myAcc.getDataRateAsString());
/* Choose the measurement range
ADXL345_RANGE_16G 16g
ADXL345_RANGE_8G 8g
ADXL345_RANGE_4G 4g
ADXL345_RANGE_2G 2g */
myAcc.setRange(ADXL345_RANGE_4G);
Serial.print(" / g-Range: ");
Serial.println(myAcc.getRangeAsString());
Serial.println();
attachInterrupt(digitalPinToInterrupt(int2Pin), in_activityISR, RISING);
/* The following settings are similar to the settings in ADXL345_activity_inactivity_interrupt.ino */
/* Three parameters have to be set for activity:
1. DC / AC Mode:
ADXL345_DC_MODE - Threshold is the defined one (parameter 3)
ADXL345_AC_MODE - Threshold = starting acceleration + defined threshold
2. Axes, that are considered:
ADXL345_000 - no axis (which makes no sense)
ADXL345_00Z - z
ADXL345_0Y0 - y
ADXL345_0YZ - y,z
ADXL345_X00 - x
ADXL345_X0Z - x,z
ADXL345_XY0 - x,y
ADXL345_XYZ - all axes
3. Threshold in g */
myAcc.setActivityParameters(ADXL345_DC_MODE, ADXL345_XY0, 0.8);
/* Four parameters have to be set for in activity:
1. DC / AC Mode:
see activity parameters
2. Axes, that are considered:
see activity parameters
3. Threshold in g
4. Inactivity period threshold in seconds (max 255) */
myAcc.setInactivityParameters(ADXL345_DC_MODE, ADXL345_XY0, 0.8, 10);
/* You can choose the following interrupts:
Variable name: Triggered, if:
ADXL345_OVERRUN - new data replaces unread data
ADXL345_WATERMARK - the number of samples in FIFO equals the number defined in FIFO_CTL
ADXL345_FREEFALL - acceleration values of all axes are below the threshold defined in THRESH_FF
ADXL345_INACTIVITY - acc. value of all included axes are < THRESH_INACT for period > TIME_INACT
ADXL345_ACTIVITY - acc. value of included axes are > THRESH_ACT
ADXL345_DOUBLE_TAP - double tap detected on one incl. axis and various defined conditions are met
ADXL345_SINGLE_TAP - single tap detected on one incl. axis and various defined conditions are met
ADXL345_DATA_READY - new data available
Assign the interrupts to INT1 (INT_PIN_1) or INT2 (INT_PIN_2). Data ready, watermark and overrun are
always enabled. You can only change the assignment of these which is INT1 by default.
You can delete interrupts with deleteInterrupt(type); */
myAcc.setInterrupt(ADXL345_ACTIVITY, INT_PIN_2);
myAcc.setInterrupt(ADXL345_INACTIVITY, INT_PIN_2);
/* Auto sleep is connected with activity and inactivity. The device goes in sleep when inactivity is

```

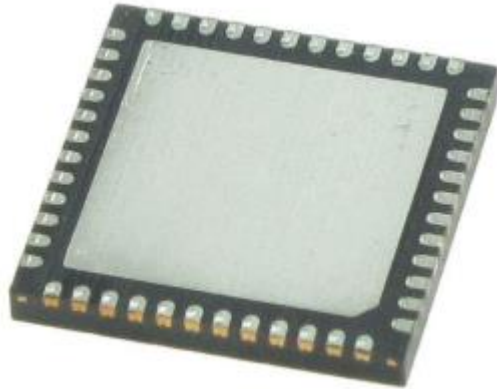
```

detected. The link bit must be set, if you want to use auto sleep. The library sets the link bit
automatically. When the ADXL345 goes into sleep mode it wakes up periodically (default is 8 Hz).
Choose the wake up frequency:
ADXL345_WUP_FQ_1 = 1 Hz
ADXL345_WUP_FQ_2 = 2 Hz
ADXL345_WUP_FQ_4 = 4 Hz
ADXL345_WUP_FQ_8 = 8 Hz */
myAcc.setAutoSleep(true, ADXL345_WUP_FQ_1);
// alternative: myAcc.setAutoSleep(true/false) without changing the wake up frequency.}
void loop() {
  if ((millis() % 300) == 1) {
    xyzFloat g = myAcc.getGValues();
    Serial.print("g-x = ");
    Serial.print(g.x);
    Serial.print(" | g-y = ");
    Serial.print(g.y);
    Serial.print(" | g-z = ");
    Serial.println(g.z);
  }
  if(in_activity == true) {
    byte intSource = myAcc.readAndClearInterrupts();
    if(myAcc.checkInterrupt(intSource, ADXL345_ACTIVITY)){
      Serial.println("Activity!");
      if(!myAcc.isAsleep()){ //check the asleep bit
        Serial.println("I am awake!");
      }
    }
  }
  if(myAcc.checkInterrupt(intSource, ADXL345_INACTIVITY)){
    Serial.println("Inactivity!");
    if(myAcc.isAsleep()){
      Serial.println("I am sleeping...");
    }
  }
  myAcc.readAndClearInterrupts();
  in_activity = false;
}
}
void in_activityISR() {
  in_activity = true;
}

```

### Partie Microcontrôleur :

Pour le microcontrôleur il nous a été donné le STM32F411CEU6, nous avons préféré prendre ce microcontrôleur plutôt que le STM32U5, certes le STM32U5 est plus performant sur la consommation mais il est aussi plus compliqué à prendre en main comparé au STM32F411CEU6, de plus la documentation à son sujet n'est pas trop fournie, nous avons eu peu d'information pour nous permettre de l'utiliser dans notre projet.



Ce microcontrôleur est très utilisé pour des systèmes à ultra basse consommation, c'est ce que nous avons absolument besoin.

Il intègre plusieurs liaisons I2C, SPI, des GPIO, sachant que nous allons utiliser un accéléromètre en liaison I2C, et d'autre capteur qui utiliseront aussi des liaisons I2C, SPI, c'est exactement ce qu'il nous faut.

Lors de nos tests nous avons alors utilisé la carte NUCLEO STM32F411RE.



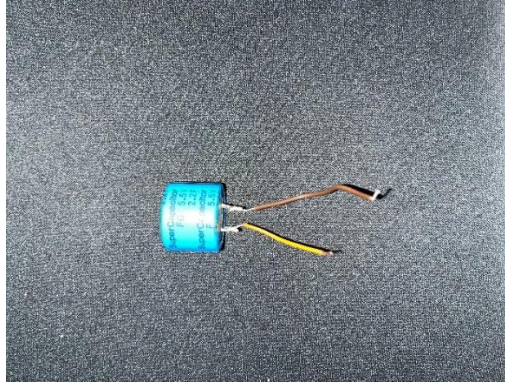


### Partie Alimentation :

Pour la partie alimentation du système, nous avons eu à notre disposition un super condensateur de 5,5 V et 2.2 F.

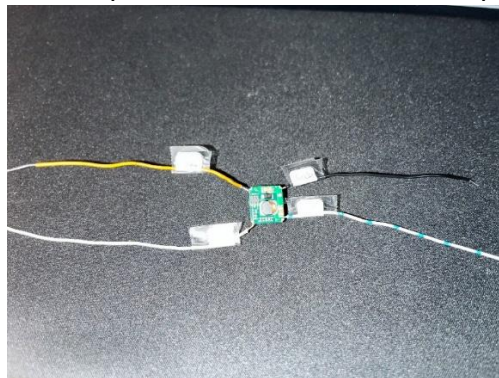
Nous avons utilisé celui-ci car nous voulions avoir une bonne autonomie et aussi pour la contrainte de taille, car notre projet devra ressembler à une clé USB.

Nous avons qu'un seul supercondensateur, un deuxième aurait pris beaucoup de place, on le rappelle, notre objectif est, que notre projet ait la taille d'une clé USB.



Pour alimenter notre système grâce à ce super condensateur nous avons dû utiliser un régulateur de tension pour transformer notre 5.5V en 3.3V, car nos composants fonctionnent seulement en 3.3V, nous avons utilisé un régulateur à découpage car nous l'avions déjà utilisé dans d'autre projet, il est performant, et surtout il est disponible dans le commerce.

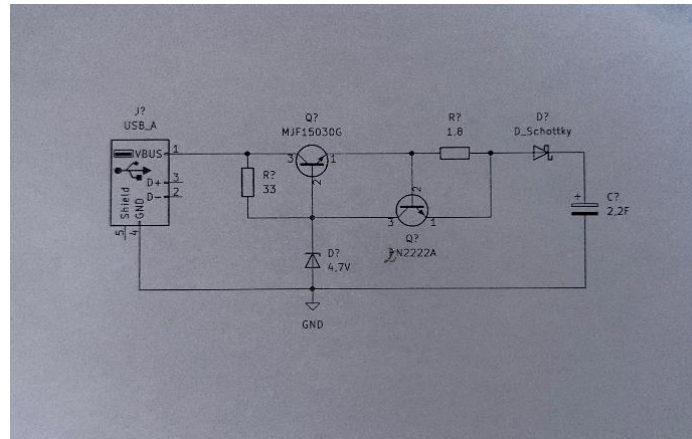
La particularité est qu'il ne chauffe pas énormément cela aurait pu fausser les mesures.



Par la suite, nous avons fait des tests sur le système en le chargeant via un adaptateur sur les prises murales pour regarder l'allure de la tension de charge et de l'intensité que pouvait tirer le supercondensateur

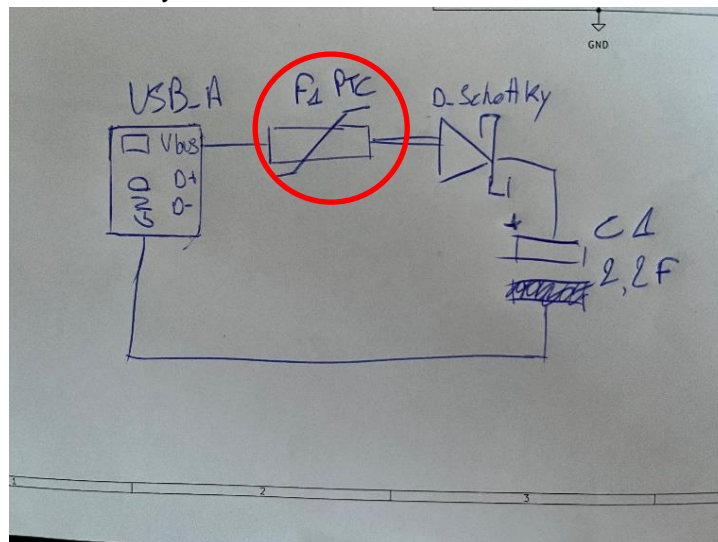


Nous avons donc testé le système une première fois avec un câblage qui nous permettait de limiter l'intensité que pouvait tirer le super condensateur et aussi de limiter la tension d'entrée pour le super condensateur.



Mais avec le câblage ci-dessus le régulateur de tension pouvait fournir du 3.3V que pendant 25 minutes or notre contrat nous demandait 72 heures.

Alors nous avons testé un autre système avec un fusible réarmable PTC (Positive température coefficient), une diode Schottky.



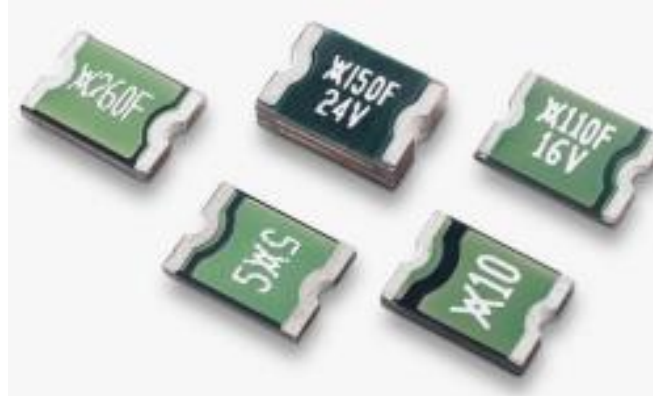
Le fusible réarmable sert à limiter l'intensité que tire le supercondensateur, la diode Schottky elle sert à éviter un retour de courant.

Avec ce système nous avons eu un meilleur résultat que le précédant mais il n'est toujours pas en accord avec nos attentes, le système a tenu 6 heures avec juste le régulateur de tension qui utilisait de l'énergie.

Malheureusement c'est le meilleur résultat que nous pouvons faire avec les directives qui nous ont été données.



Nous y avons aussi mis un fusible réarmable CTP (Coefficient en température positif) ou PTC dans les normes anglaise, lorsqu'il va commencer à chauffer un peu trop, le fusible voit sa résistance augmenter à un point où le courant ne passe plus, lorsqu'il va refroidir sa résistance baisse et le courant peut alors passer à travers le composant



Nous avons utilisé un programme qui a été fait par un ancien élève nous permettant de savoir combien de temps notre système a pu fournir du 3.3V.

```

unsigned long startTime;
unsigned long totalTime[4];
const int PIN = A0; /*Déclaration de la broche analogique qui servira à faire la mesure*/
int value;
unsigned long stopTime;
unsigned long total;

void setup() {
  Serial.begin(115200);
  startTime = millis();          /*définition de l'heure de début*/
}

void loop() {
  value = analogRead(PIN);
  Serial.print("reg 3.3V : ");
  Serial.println (value);
  if((value < 675){ /*si la tension est très faible alors*/
    stopTime = millis(); /*définition de l'heure de fin*/
    total = stopTime - startTime; /*calcul de la durée en ms*/

    totalTime[0] = total/3600000; /*conversion des ms en h*/
    totalTime[1] = (total % 3600000) / 60000; /*conversion des ms en m*/
    totalTime[2] = ((total % 3600000) % 60000) / 1000; /*conversion des ms en s*/
    totalTime[3] = ((total % 3600000) % 60000) % 1000;

    Serial.print("Durée totale :");
    Serial.print(totalTime[0]);
    Serial.print("h");
    Serial.print(totalTime[1]);
    Serial.print("min");
  }
}

```

```

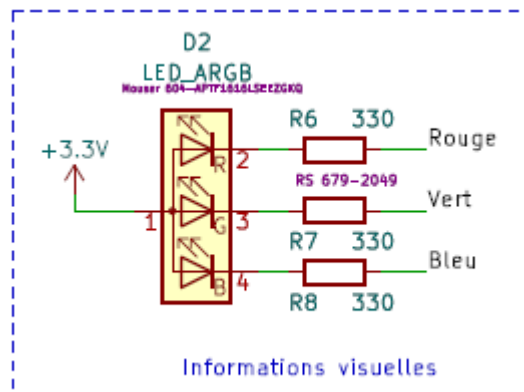
Serial.print(totalTime[2]);
Serial.print("s");
Serial.print(totalTime[3]);
Serial.println("ms");

while(1);
}
}

```

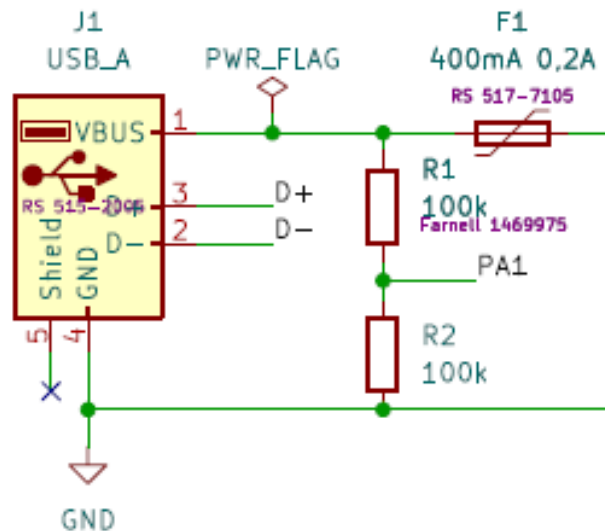
### Partie Information visuelle :

Pour pouvoir informer du bon fonctionnement du système nous y avons ajouté un système de LED qui permet d'indiquer à l'utilisateur l'état du système.



Lorsque le système sera activé par l'utilisateur une petite lumière verte clignotera tous les x temps, la LED rouge permettra d'annoncer à l'utilisateur de savoir si le colis a eu un problème avant même de brancher le système sur un ordinateur, le LED bleu est encore à définir.

Partie Information branchement :



Pour nous permettre de commencer la communication entre l'ordinateur et notre système, il faut que notre stm32 puisse détecter si notre clé est branchée ou non.

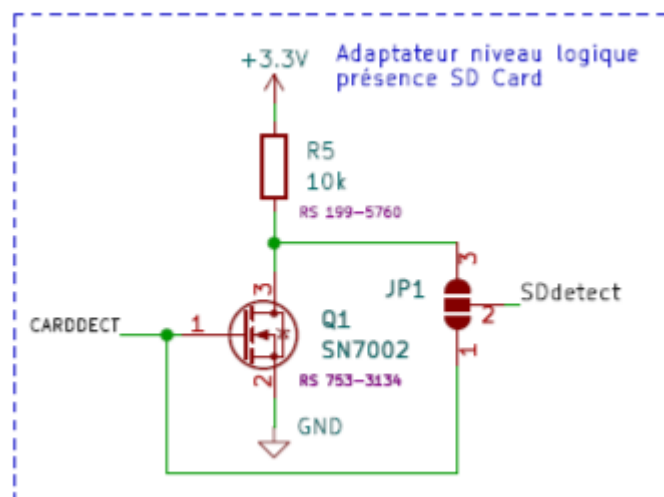
Pour se faire, nous avons utilisé une broche de notre stm32 pour pouvoir par la suite détecter si elle est branchée.

Lorsque notre système sera branché sur le port USB celui-ci délivrera 5V, cette tension est trop forte pour notre système qui peut accepter au maximum 3.3V.

Donc, nous avons mis un pont diviseur de tension qui divisera la tension par 2 pour délivrer du 2.5V qui est accepté par la carte.

Logiciellement, nous allons dire au stm32 que quand il reçoit un courant de 2.5V ou moins sur sa broche PA1 alors oui il est connecté à l'ordinateur et il peut donc commencer la communication.

Systeme de détection de la carte SD :



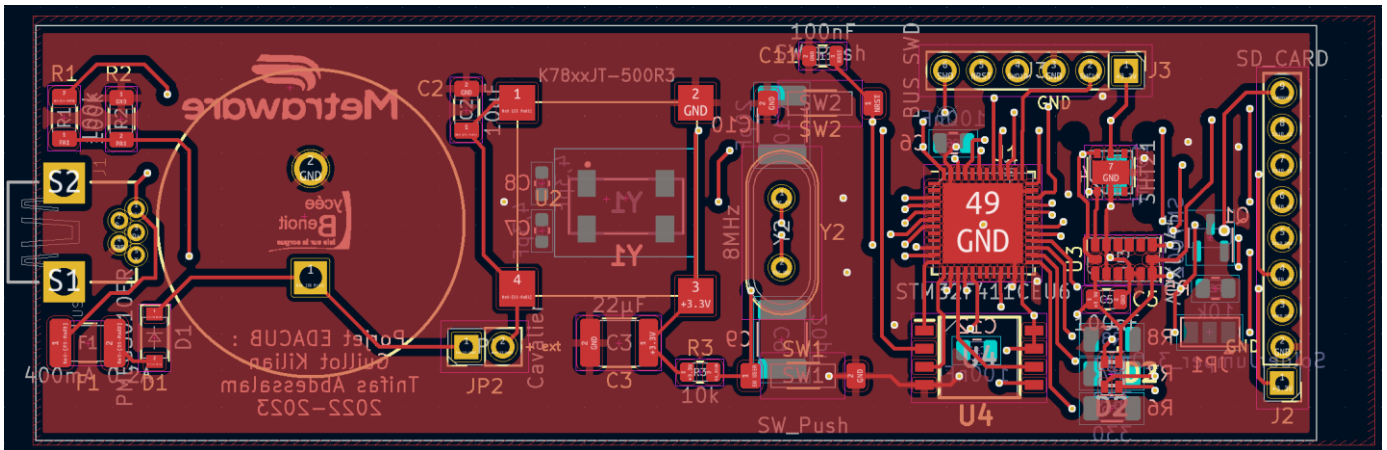
En fonction du composant utilisé pour la carte SD un niveau logique est délivré, il peut être à l'état haut ou à l'état bas, lorsque nous allons souder tous nos composants nous allons souder JP1 en fonction du niveau logique dont nous nous avons besoin.

Partie saisie de schéma Kicad :

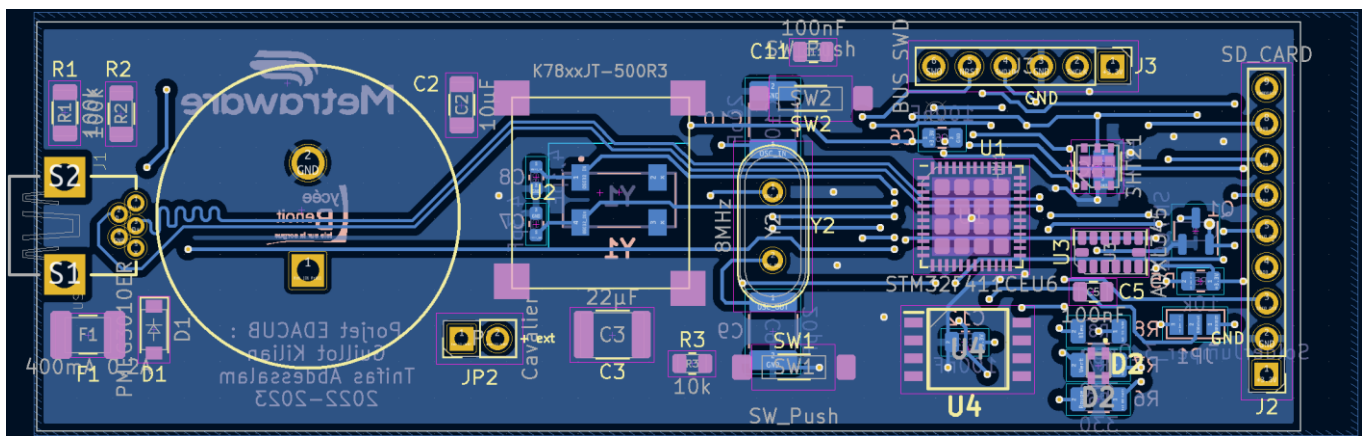
Pour le routage de la carte nous nous sommes appuyés sur les normes d'une clé USB.  
Nous y avons mis alors nos composants dessus.

Voici le résultat :

Vue de dessus



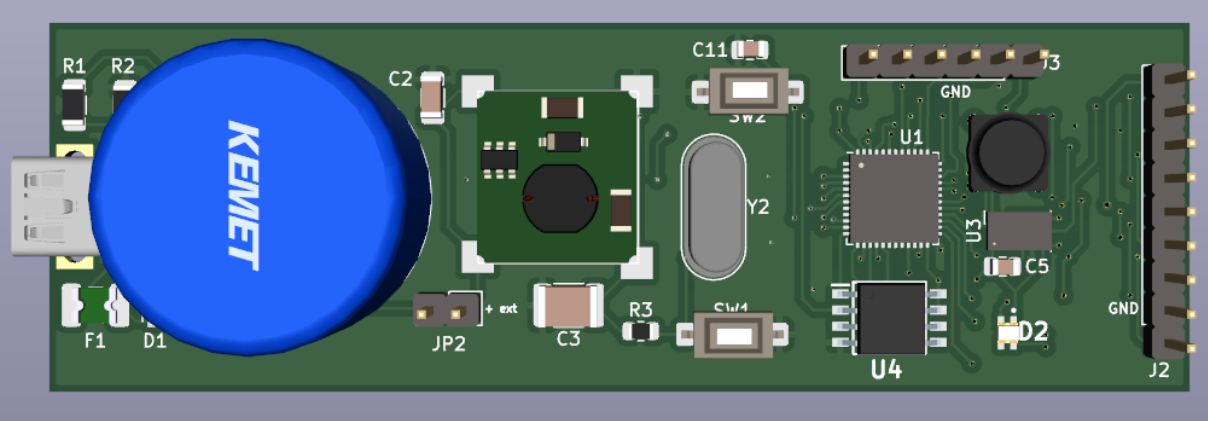
Et voici la vue de dessous



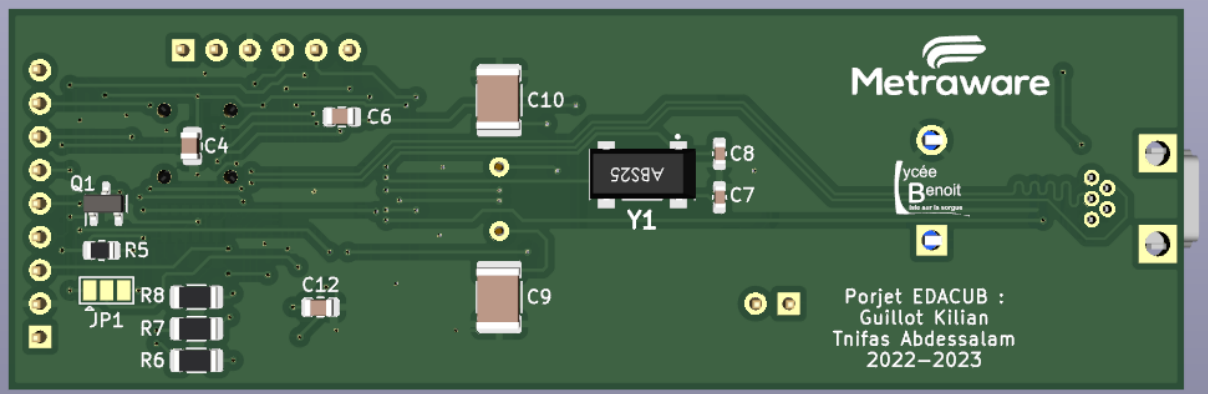
Lors de la mise en place des composants, trouver de la place a été un combat, comme on peut le voir les composants sont assez proches les uns des-autres, cependant, aucuns composants ne gênent son voisin.

Voici une vue 3D pour se rendre compte de l'apparence finale de la carte :

Vue de dessus



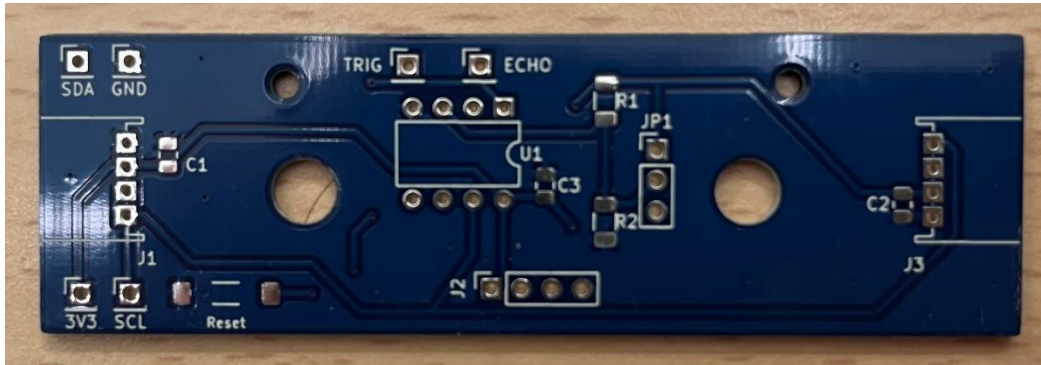
Et voici la vue de dessous





Matériel utilisé lors de l'assemblage de la carte :

Lors de la commande de nos cartes un problème est survenu, les cartes étaient coincées en douane, donc voici ci-dessous un exemple de carte qui a, à peu près, le même gabarie de notre vraie carte.



Pour déposer la pâte à brasé sur notre carte, nous disposons d'un stencil qui correspond à notre carte pour en déposer sur toutes les zones grises de la carte.



La pâte à brasé utilisée.



Une fois la pâte à brasé mise, nous plaçons les composants avec une pince brucelles et une loupe électronique.





Lorsque les composants sont placés à leur place respective il est l'heure de les passer au four à refusion





Lorsque nous allons commencer, le four va d'abord entrer en préchauffage, c'est-à-dire qu'il va amener le circuit entre 100°C et 160°C par la suite il va monter à 170°C pour se préparer à la phase de refusion, il va ensuite atteindre rapidement les 230°C pour souder les composants à la carte.

Ce processus est là pour souder les composants en CMS (composant monté en surface), pour les composants traversants nous utilisons la station de soudage.



# Partie IR 1 : POMARO Brice

## Présentation du rôle du candidat

Dans ma partie je m'occupe de la partie *firmware STM32* (programmation de la carte STM32) avec comme buts :

- L'acquisition de la température et de l'humidité à partir d'un capteur *Sensirion (SHT21)*.
- La détection de chocs à partir d'un capteur de chocs *Analog Devices (ADXL375)*.
- La journalisation -- avec horodatage depuis la RTC interne du STM32F411 -- de ces informations sur un média de stockage qui serait idéalement la mémoire flash interne du microcontrôleur. (D'autres mémoires de stockage sont aussi à notre disposition tels que la RAM interne, une carte SD et une mémoire de type F-RAM).

Le développement de ce projet s'effectuera en langage C sur un environnement de développement intégré (IDE) de chez *STMicroElectronics* appelé *STM32CubeIDE*.

## Objectifs principaux

L'application globale de l'enregistreur de donnée prendra la forme d'une machine à états pour gérer les différents modes (acquisition, restitution, charge, veille...).

Elle impliquera de :

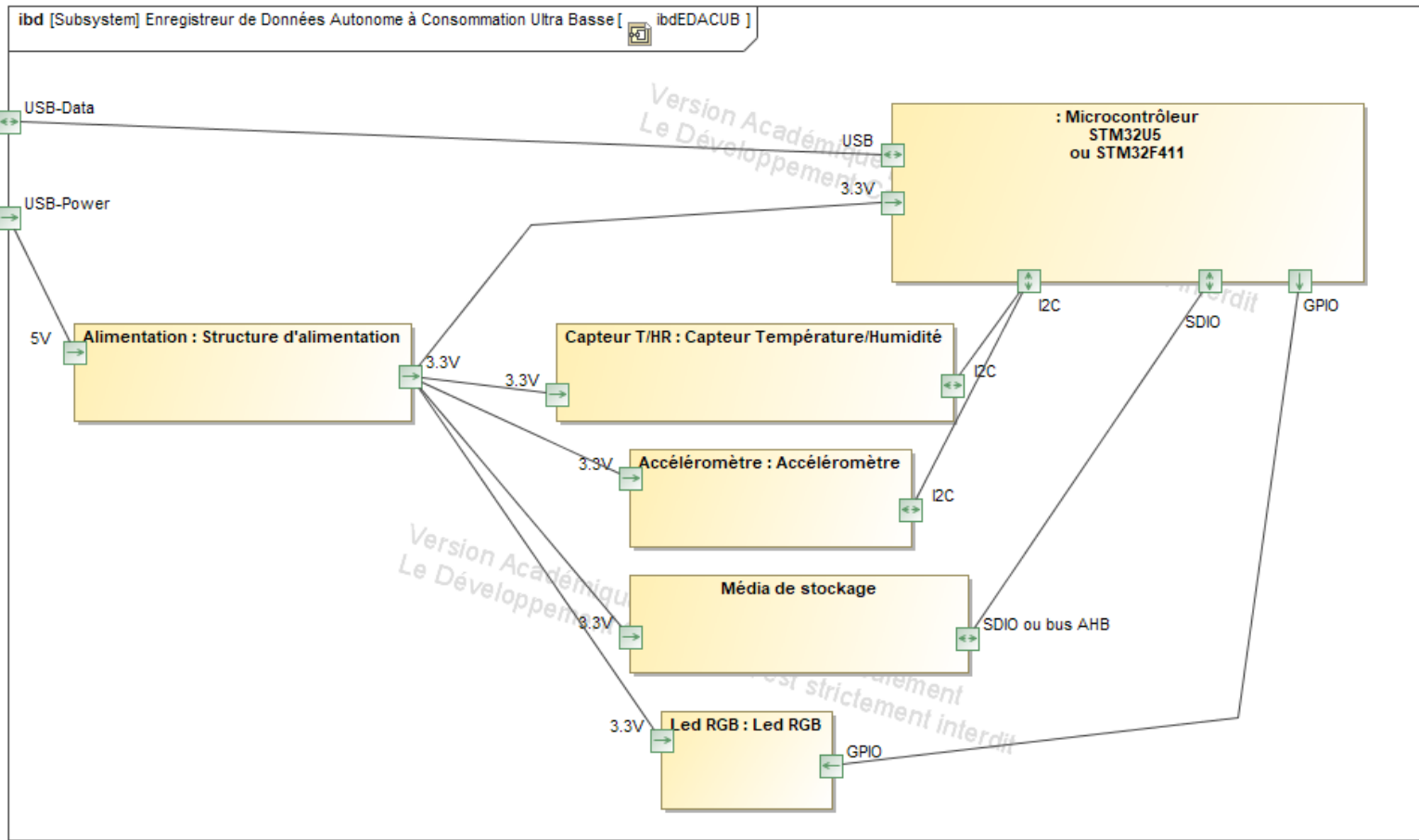
- Concevoir un driver en C pour exploiter le capteur de *température* et d'*humidité relative*
- Concevoir un driver en C pour exploiter *l'accéléromètre*
- Gérer le mode veille du *STM32* pour limiter autant que possible la *consommation*
- Mettre en œuvre le *middleware FatFS* et l'adapter au média de stockage choisi pour permettre au STM32 de stocker les valeurs mesurées ainsi que les fichiers du site web embarqué dans un système de fichiers pris en charge par la majorité des systèmes d'exploitation
- Mettre en place une *connectivité USB avec un profil MSC Device* (Media Storage Class Device) pour faire détecter le système à base de STM32 comme un média de stockage lorsqu'il est connecté à un ordinateur
- Mettre en œuvre l'OS temps réel *FreeRTOS*

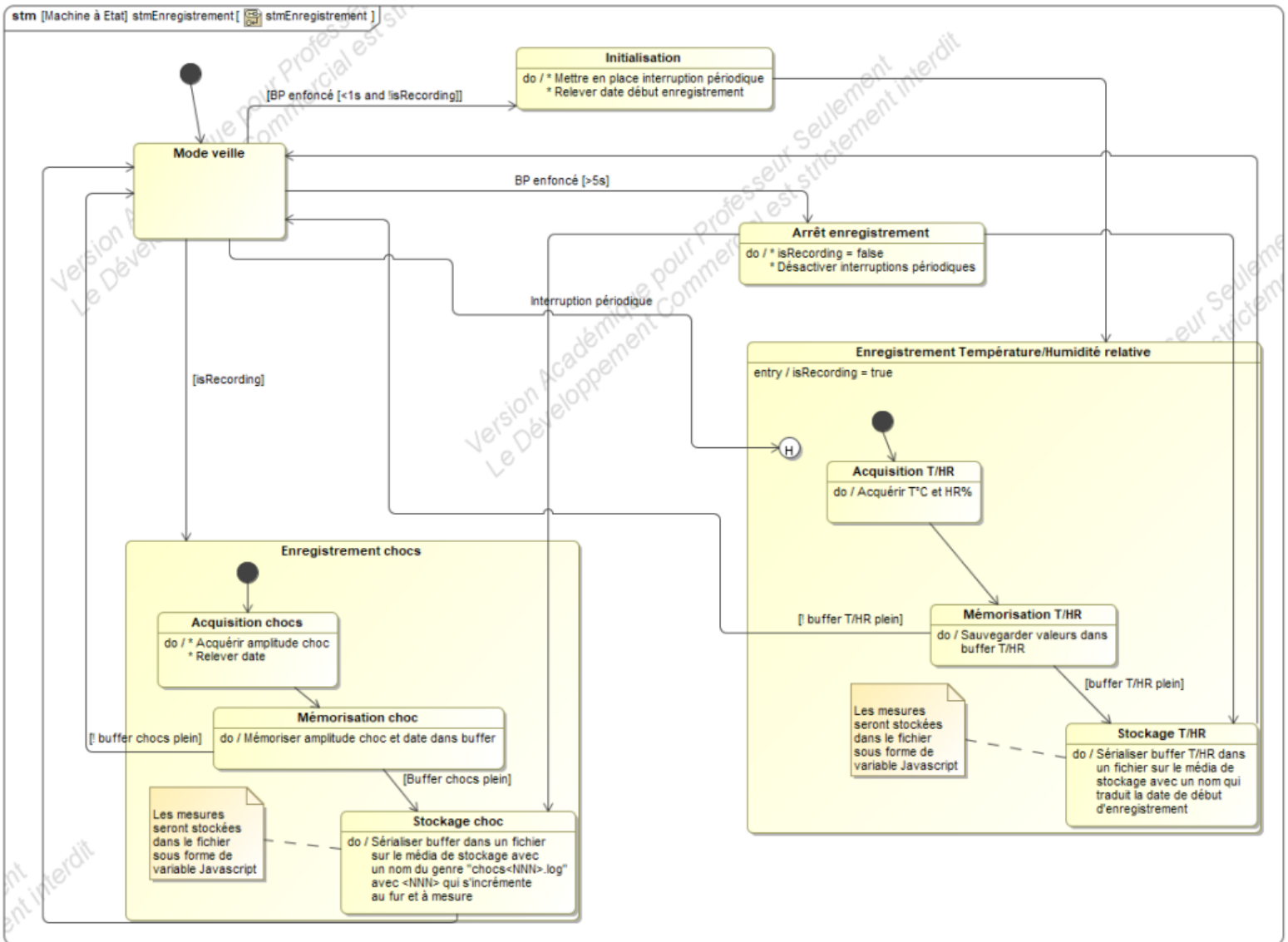
L'enregistreur sera architecturé autour du micro-contrôleurs (MCUs) de la gamme *High Performance* des STM32 de chez ST Electronics (*STM32F411*) dont j'expliquerai le choix ultérieurement. La fréquence des mesures doit être configurable entre 10s et 15min. Les mesures seront horodatées.

## Planification prévisionnelle

Nom de la tâche	Durée	Début	Fin	Progrès
<b>IR 1</b>	<b>174h</b>	<b>03/01/23</b>	<b>17/04/23</b>	
Prendre connaissance du matériel	1h	04/01/2023	04/01/2023	100%
S'approprier le fonctionnement de la chaîne de développement STM32	10h	04/01/2023	11/01/2023	100%
Concevoir/coder/tester un driver pour exploiter le capteur de T/HR	10h	16/01/2023	18/01/2023	100%
Concevoir/coder/tester un driver pour l'accéléromètre	20h	23/01/2023	31/01/2023	100%
Etudier / Mettre en oeuvre le middleware FatFS (système de fichiers) sur le STM32U5 ou STM32F411 et l'adapter au média de stockage choisi (Flash interne, F-RAM, carte uSD)	30h	06/02/2023	21/02/2023	50%
S'approprier la structure, le fonctionnement de FreeRTOS et le mettre en oeuvre dans le STM32U5 ou le STM32F411	10h	22/02/2023	28/02/2023	0%
Etudier / Mettre en oeuvre le middleware USB Device avec profil MSC (Media Storage Class)	15h	06/03/2023	13/03/2023	100%
Concevoir / Coder / Tester l'application de l'enregistreur de données	10h	15/03/2023	21/03/2023	0%
Définir avec IR2 le format des données enregistrées (structure, précision des valeurs, horodatage)	2h	27/03/2023	27/03/2023	75%
Collaborer étroitement avec EC1 et EC2 pour déterminer les ressources à mettre en oeuvre (GPIOs, Interruptions, ...)	5h	03/04/2023	04/04/2023	20%
Assurer la gestion de version logicielle (Forge logicielle Framagit)	5h	10/04/2023	11/04/2023	50%

# Diagrammes SysML individuels

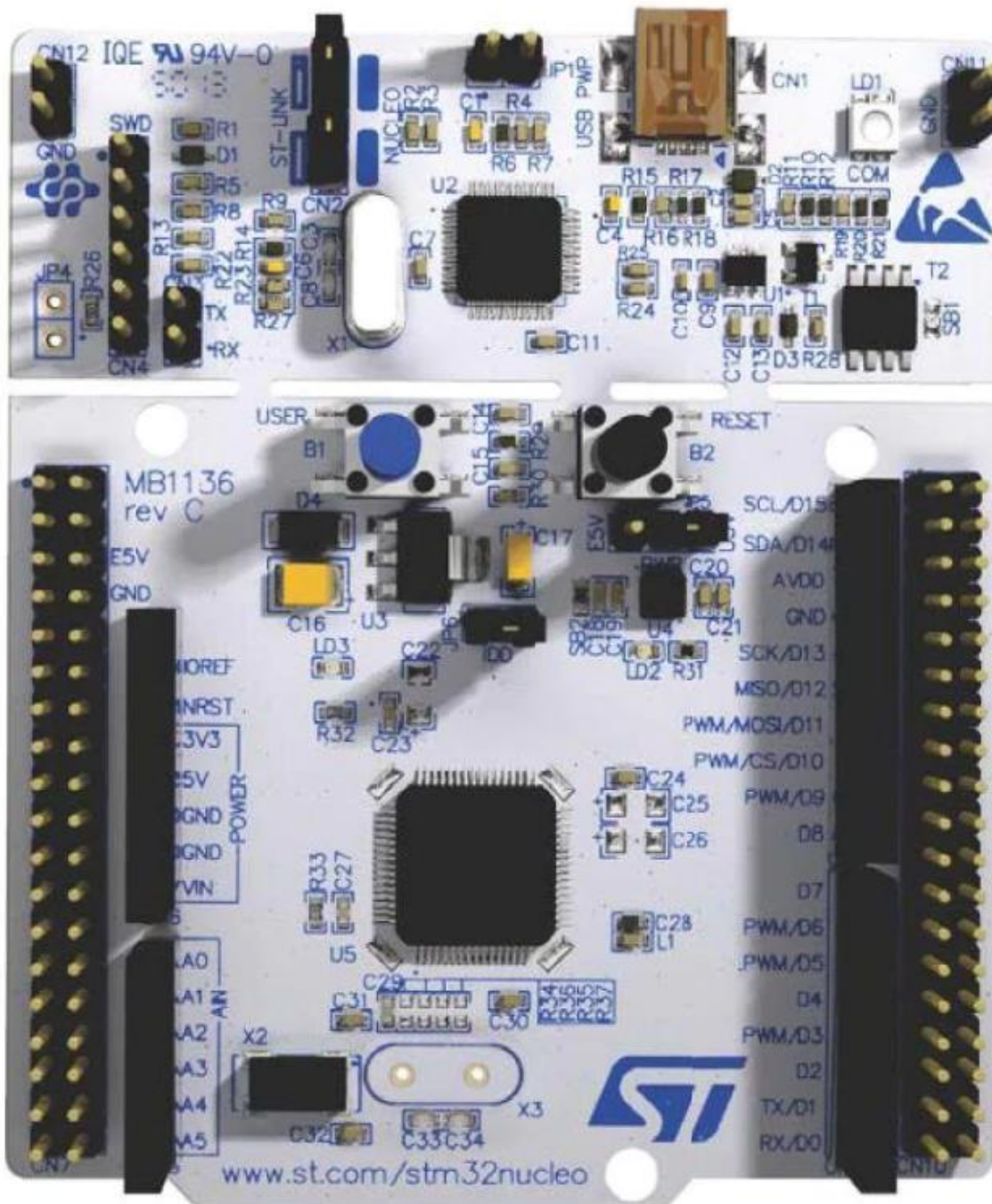






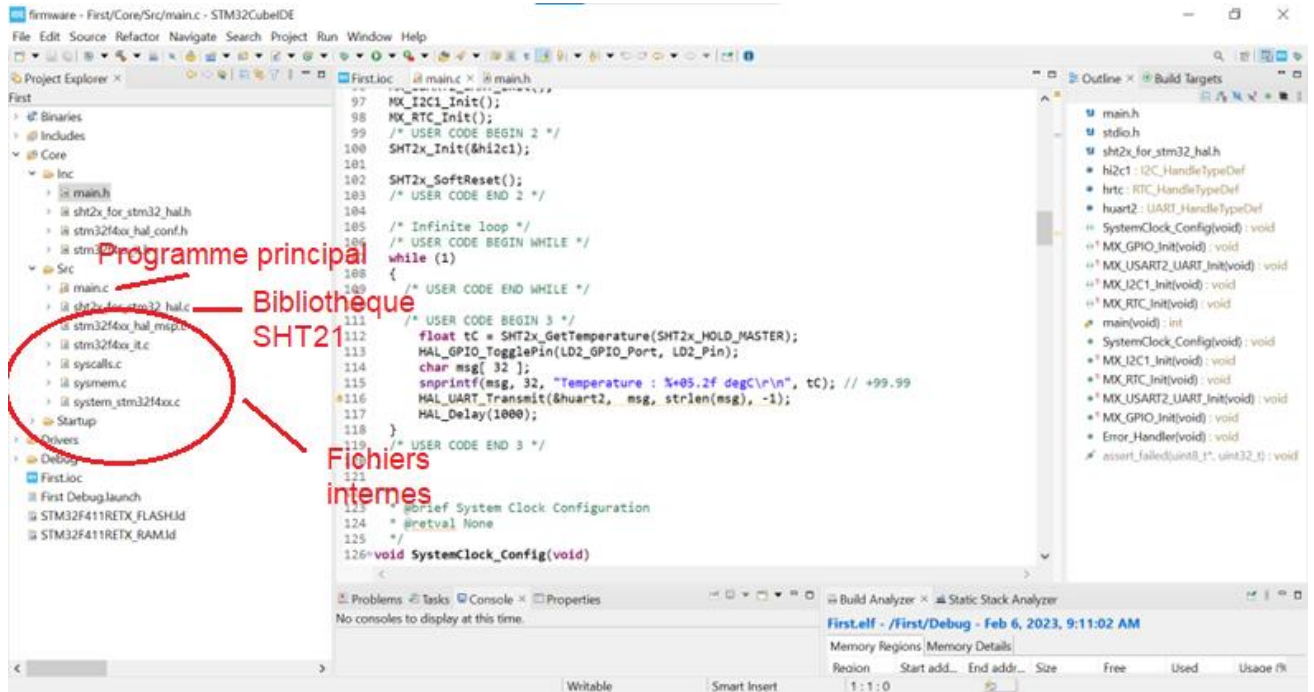
## Qu'est-ce que STM32 ?

Le STM32 est une famille de circuits intégrés de microcontrôleurs et je vais personnellement utiliser durant 90% de mon temps de projet le logiciel *STM32CubeIDE* qui est un environnement de développement intégré consacré au développement sur STM32.

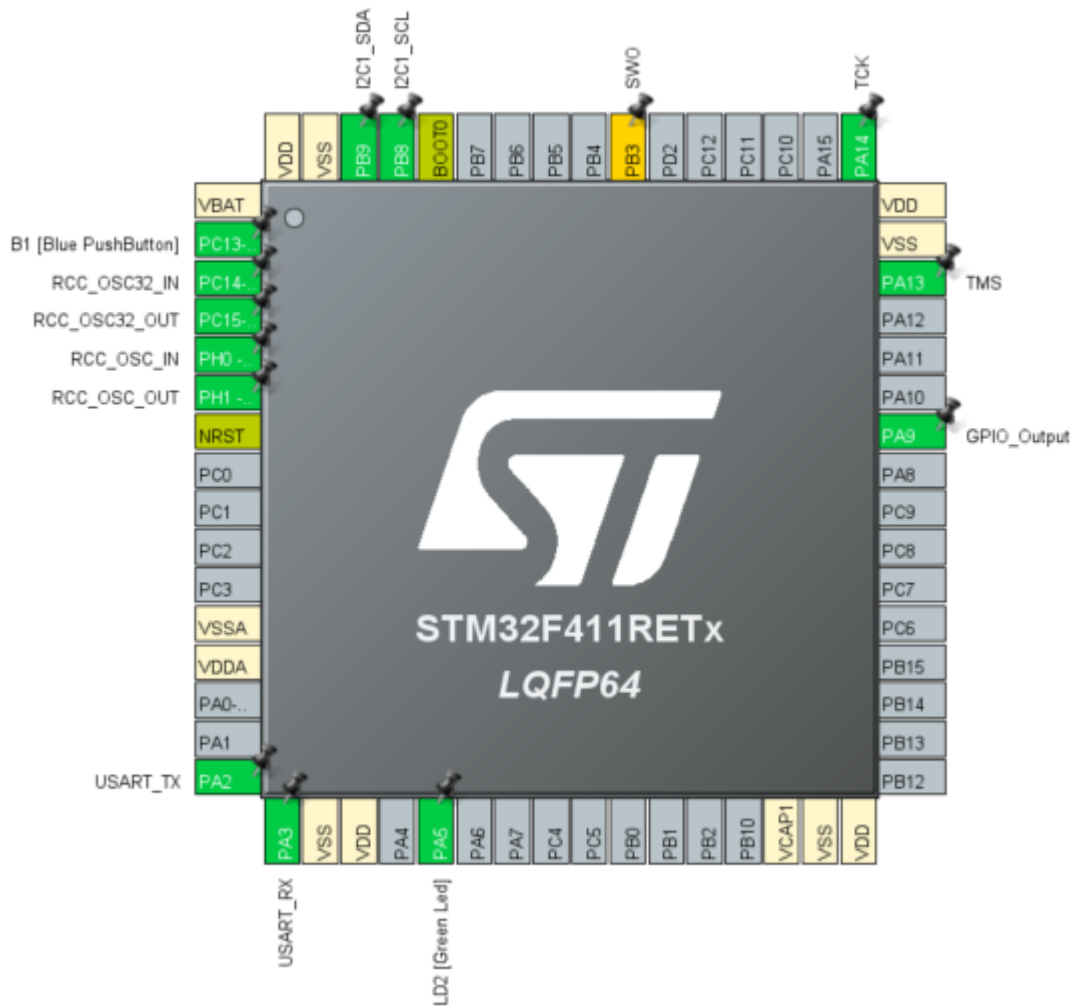


L'environnement de développement STM32 peut être caractérisé en 2 parties :

D'une part le côté programmation, avec les nombreux fichiers, main.c, main.h, les includes, les bibliothèques, les fichiers internes etc..



Et d'un autre côté, le fichier IOC qui correspond à une interface graphique configurable, permettant d'activer ou de désactiver certaines broches et/ou fonctionnalités du STM32F411, nous permettant la configuration d'interruptions, d'allumage de LED's, de communications SPI et I2C et bien plus encore. Cette interface graphique possède également une auto-génération de code, car en effet lorsqu'on active une broche, le logiciel nous propose d'auto-générer le code, du fait de la quantité de lignes de code que ça nous obligerait à écrire, afin de faciliter et diminuer la charge de travail.



On peut voir ici les broches activées sur la **STM32** en **vert** celles qui sont désactivées (car inutilisées) en **gris** et les autres couleurs qui subsistent sont des broches déjà activées par défaut qui ne peuvent être désactivées (comme le bouton Reset ou l'alimentation par exemple)



# Compte rendu d'activités - Revue de projet 1

## Capteur choc

- Datasheet

L'analyse de la datasheet était un élément important de notre activité, il fallait analyser les registres afin de choisir les meilleures options pour l'accéléromètre pour drainer la consommation.

- Adresse I2C

### **Register 0x00—DEVID (Read Only)**

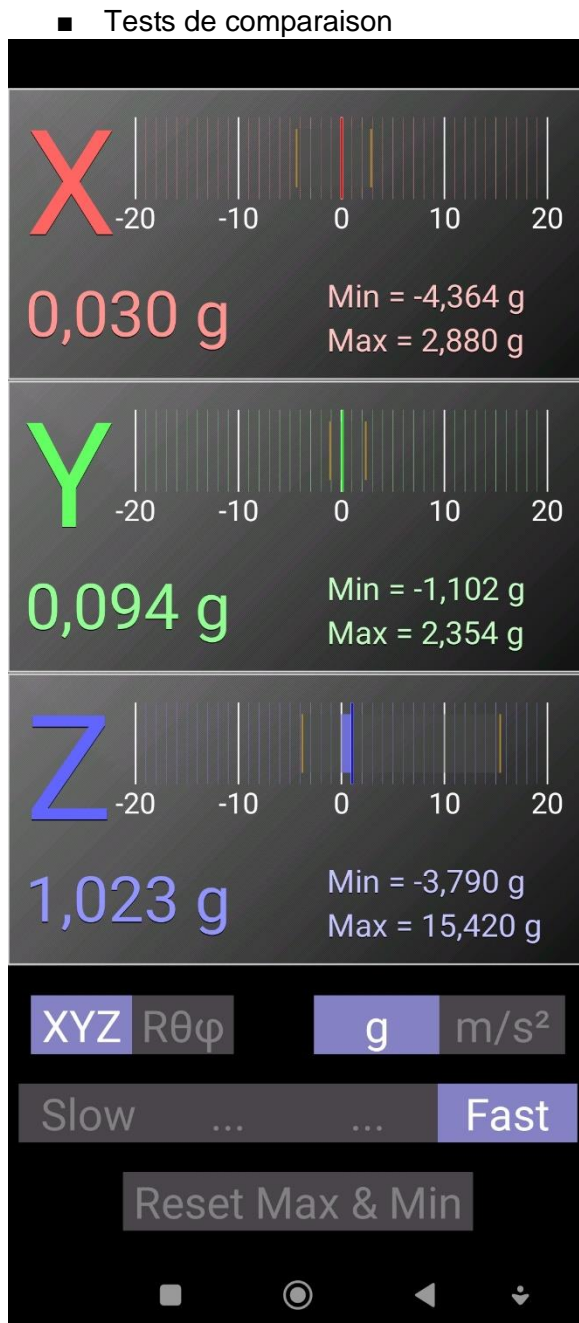
D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	0	1	0	1

L'adresse du composant est fixe : 0xE5 en hexadécimal.

- Registres

Address		Register Name	Access Type	Reset Value	Description
Hex	Decimal				
0x00	0	DEVID	R	11100101	Device ID
0x01 to 0x1C	1 to 28	Reserved	N/A	N/A	Reserved; do not access
0x1D	29	THRESH_SHOCK	R/W	00000000	Shock threshold
0x1E	30	OFSX	R/W	00000000	X-axis offset
0x1F	31	OFSY	R/W	00000000	Y-axis offset
0x20	32	OFSZ	R/W	00000000	Z-axis offset
0x21	33	DUR	R/W	00000000	Shock duration
0x22	34	Latent	R/W	00000000	Shock latency
0x23	35	Window	R/W	00000000	Shock window
0x24	36	THRESH_ACT	R/W	00000000	Activity threshold
0x25	37	THRESH_INACT	R/W	00000000	Inactivity threshold
0x26	38	TIME_INACT	R/W	00000000	Inactivity time
0x27	39	ACT_INACT_CTL	R/W	00000000	Axis enable control for activity and inactivity detection
0x2A	42	SHOCK_AXES	R/W	00000000	Axis control for single shock/double shock
0x2B	43	ACT_SHOCK_STATUS	R	00000000	Source of single shock/double shock
0x2C	44	BW_RATE	R/W	00001010	Data rate and power mode control
0x2D	45	POWER_CTL	R/W	00000000	Power saving features control
0x2E	46	INT_ENABLE	R/W	00000000	Interrupt enable control
0x2F	47	INT_MAP	R/W	00000000	Interrupt mapping control
0x30	48	INT_SOURCE	R	00000010	Interrupt source
0x31	49	DATA_FORMAT	R/W	00000000	Data format control
0x32	50	DATA_X0	R	00000000	X-Axis Data 0
0x33	51	DATA_X1	R	00000000	X-Axis Data 1
0x34	52	DATA_Y0	R	00000000	Y-Axis Data 0
0x35	53	DATA_Y1	R	00000000	Y-Axis Data 1
0x36	54	DATA_Z0	R	00000000	Z-Axis Data 0
0x37	55	DATA_Z1	R	00000000	Z-Axis Data 1
0x38	56	FIFO_CTL	R/W	00000000	FIFO control
0x39	57	FIFO_STATUS	R	00000000	FIFO status

Il existe une grande quantité de registres permettant diverses actions sur le capteur de chocs, comme par exemple les registres 0x1E, 0x1F et 0x20 qui sont des registres permettant de gérer l'offset des axes X, Y et Z respectivement, c'est-à-dire de compenser l'erreur potentielle de la valeur renvoyée par le capteur (par exemple, s'il est à plat, la valeur de X et Y doivent être à 0 et Z à 1, et sans l'utilisation de ces 3 registres, les valeurs renvoyées peuvent ne pas être 100% précises.



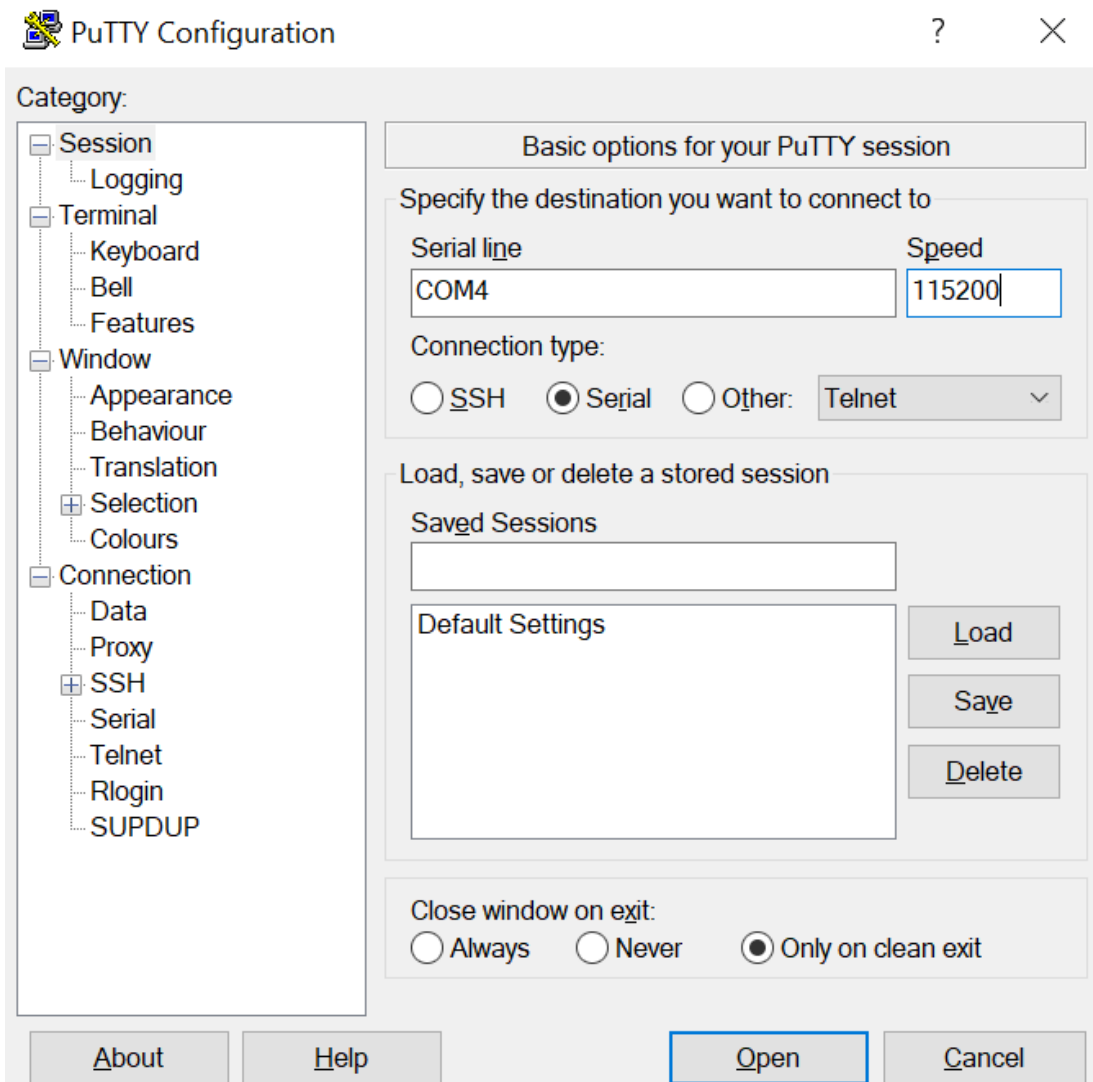
J'ai installé une application sur mon téléphone qui utilise un capteur de choc interne au téléphone pour avoir une idée vague (résultats à prendre avec des pincettes, ce n'est qu'un téléphone) des valeurs renvoyées sur le capteur. Pour ce test, mon téléphone est stable, sur une table à plat. J'ai effectué un test de choc en le laissant tomber d'une dizaine de centimètres vers le sol. On peut observer qu'il a reçu un choc de plus de 15g sur l'axe des Z, et 2g sur l'axe des Y et des X.

- Configuration logiciel CubeIDE

The screenshot shows the 'Pinout & Configuration' window in STM32CubeIDE. The left sidebar lists various components, with 'I2C1' highlighted in blue and circled in red. A red arrow points from this highlight to the text 'Activation de l'I2C1 sur la STM32'. The main area shows the 'I2C1 Mode and Configuration' section with 'I2C' selected in the 'Mode' dropdown. Below this is the 'Configuration' section, which includes a 'Reset Configuration' button and tabs for 'Parameter Settings', 'User Constants', 'NVIC Settings', 'DMA Settings', and 'GPIO Settings'. The 'GPIO Settings' tab is active, displaying a table of pin configurations. The table has columns for 'Pin Name', 'Signal on Pin', 'GPIO output', 'GPIO mode', 'GPIO Pull-u', 'Maximum o', 'User Label', and 'Modifie'. Two rows are visible: PB8 with signal 'I2C1\_SCL' and PB9 with signal 'I2C1\_SDA'. Both 'I2C1\_SCL' and 'I2C1\_SDA' are circled in red, with a red arrow pointing to the text 'Signal Clock et Signal Data sur les broches PB8 et PB9'.

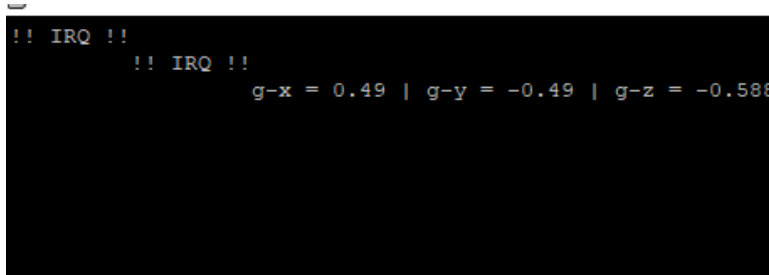
Pin Name	Signal on Pin	GPIO output	GPIO mode	GPIO Pull-u	Maximum o	User Label	Modifie
PB8	I2C1_SCL	n/a	Alternate Fu...	Pull-up	Very High		<input checked="" type="checkbox"/>
PB9	I2C1_SDA	n/a	Alternate Fu...	Pull-up	Very High		<input checked="" type="checkbox"/>

■ Configuration PuTTY



Pour observer nos résultats, nous utilisons PuTTY, un logiciel permettant l'émulation d'un terminal, capable de se connecter en SSH (à distance) ou en Serial, c'est-à-dire par liaison USB. Il faut d'abord vérifier le port USB utilisé par la carte STM32 (dans mon cas, COM4), paramétrer le baud rate à 115200 (le baud rate étant le nombre de symboles par secondes)

■ Résultat du programme



## Capteur Température

- Datasheet
  - Adresse I2C

Command	Comment	Code
Trigger T measurement	hold master	1110'0011
Trigger RH measurement	hold master	1110'0101
Trigger T measurement	no hold master	1111'0011
Trigger RH measurement	no hold master	1111'0101
Write user register		1110'0110
Read user register		1110'0111
Soft reset		1111'1110

La Datasheet présente plusieurs adresses vis-à-vis de la fonction choisie. Parmi celles-ci, voici celles que nous utiliserons :

- Trigger T measurement en mode « no hold master » (0xF3) le mode no hold master signifie que lorsque le capteur prend une mesure, il autorise les autres communications SCL à être utilisées, tandis que le mode « hold master » interdit les autres communications SCL de faire quoi que ce soit.
- Trigger RH measurement en mode “no hold master” (0xF5)
- Soft reset (0xFE)

- Résultat du programme

COM4 - PuTTY

```

Temperature : +25.02 degC
Humidite : +61.08 Hr

Temperature : +25.02 degC
Humidite : +61.08 Hr

Temperature : +25.01 degC
Humidite : +61.05 Hr

Temperature : +25.01 degC
Humidite : +61.00 Hr

Temperature : +25.00 degC
Humidite : +60.91 Hr

Temperature : +25.00 degC
Humidite : +60.76 Hr

Temperature : +24.99 degC
Humidite : +60.65 Hr

Temperature : +24.99 degC
Humidite : +60.52 Hr

Temperature : +24.98 degC
Humidite : +60.38 Hr

Temperature : +24.98 degC
Humidite : +60.27 Hr

Temperature : +24.98 degC
Humidite : +60.08 Hr

Temperature : +24.97 degC
Humidite : +59.94 Hr

Temperature : +24.97 degC
Humidite : +59.82 Hr

Temperature : +24.95 degC
Humidite : +59.71 Hr

```

## Ajustement de durée de sommeil en cas de détection de choc(s)

Durant ce projet, une séance a été consacrée à l'ajustement d'un écart de temps à la suite d'un choc, permettant la soustraction d'un temps en secondes entre le moment où la mesure de température est faite et entre le moment où un choc est mesuré.

En condition normale la température se mesure à intervalle régulier, or, la détection d'un choc fait sortir du mode sommeil. On souhaite donc éviter un délai de mesure de la température pour avoir même dans le cas d'un choc, des mesures de température et d'humidité régulières.

En effet, avant la mise en place de cette sauvegarde, dès qu'un choc se produisait, à partir du choc, le délai était à nouveau le temps du standby paramétré par défaut, ce qui produisait des mesures complètement désordonnées dans le temps.

Je vais par la suite expliquer mon code pour fournir des explications sur le chemin parcouru pour arriver à mes fins.

```
RTC_TimeTypeDef now = {0};
RTC_DateTypeDef today = {0};
```

Pour commencer, on initialise les valeurs de 2 structures spéciales aux STM32F4XX à 0 (valeurs qui seront lues par la suite dans le programme)

```
#typedef struct
{
  uint8_t Hours;          /*!< Specifies the RTC Time Hour.
                          This parameter must be a number between Min_Data = 0 and Max_Data = 12 if the RTC_HourFormat_12 is selected
                          This parameter must be a number between Min_Data = 0 and Max_Data = 23 if the RTC_HourFormat_24 is selected */
  uint8_t Minutes;       /*!< Specifies the RTC Time Minutes.
                          This parameter must be a number between Min_Data = 0 and Max_Data = 59 */
  uint8_t Seconds;       /*!< Specifies the RTC Time Seconds.
                          This parameter must be a number between Min_Data = 0 and Max_Data = 59 */
  uint8_t TimeFormat;    /*!< Specifies the RTC AM/PM Time.
                          This parameter can be a value of @ref RTC_AM_PM_Definitions */
  uint32_t SubSeconds;   /*!< Specifies the RTC_SSR RTC Sub Second register content.
                          This parameter corresponds to a time unit range between [0-1] Second
                          with [1 Sec / SecondFraction +1] granularity */
  uint32_t SecondFraction; /*!< Specifies the range or granularity of Sub Second register content
                          corresponding to Synchronous prescaler factor value (PREDIV_S)
                          This parameter corresponds to a time unit range between [0-1] Second
                          with [1 Sec / SecondFraction +1] granularity.
                          This field will be used only by HAL_RTC_GetTime function */
  uint32_t DayLightSaving; /*!< This interface is deprecated. To manage Daylight
                          Saving Time, please use HAL_RTC_DST_xxx functions */
  uint32_t StoreOperation; /*!< This interface is deprecated. To manage Daylight
                          Saving Time, please use HAL_RTC_DST_xxx functions */
} RTC_TimeTypeDef;
```

Ces structures possèdent de nombreuses valeurs (exemple ci-dessus avec la structure pour l'heure) permettant de lire les heures, minutes, secondes, format de l'heure (12h ou 24h) et les autres valeurs ne seront pas utilisées dans notre programme.

De même pour la structure de la date possédant des valeurs comme le jour de la semaine, le mois, le jour, l'année. Nous verrons par la suite que ces structures posent quelques légers soucis.

```

int DD, MM, YY, WD, hh, mm, ss;

printf("\n\rSaisir la date avec le format jj/mm/aa : ");
scanf("%d/%d/%d", &DD, &MM, &YY);

today.Date = DD;
today.Month = MM;
today.Year = YY;
printf("\n\rSaisir le jour de la semaine (0=Lundi -> 6=Dimanche) : ");
scanf("%d", &WD);
today.WeekDay = WD;

printf("\n\rSaisir l'heure avec le format hh:mm:ss : ");
scanf("%x:%x:%x", &hh, &mm, &ss);
now.Hours = hh;
now.Minutes = mm;
now.Seconds = ss;
printf("\n\rLet's go !\r\n\n");
if (HAL_RTC_SetTime(&hrtc, &now, RTC_FORMAT_BCD) != HAL_OK)
{
    Error_Handler();
}
if (HAL_RTC_SetDate(&hrtc, &today, RTC_FORMAT_BCD) != HAL_OK)
{
    Error_Handler();
}

```

Par la suite, on lit les valeurs entrées par l'utilisateur pour définir un horaire (le jour et la date actuels) en utilisant une variable `today` et une autre `now` qui sont les structures définies plus haut en leur donnant les valeurs précédemment inscrites par l'utilisateur tel que la date, le mois, l'année, l'heure, les minutes et les secondes.

En tout dernier lieu on vérifie que l'heure et la date sont dans le bon format, et cet ordre est le seul possible car si on effectue la vérification de la date avant celui de l'heure, le registre contenant l'heure n'est plus mis à jour donnant ainsi des valeurs incohérentes à celles inscrites par l'utilisateur.

```
int standbyDuration = STANDBY_DURATION * 32768 / 16;
```

Ensuite, j'initialise une variable `standbyDuration` avec la valeur définie par l'utilisateur pour le standby classique qu'on multiplie par 32768 (fréquence d'entrée en Hz de la clock) et qu'on divise par 16 (valeur donnée par défaut dans la configuration de la RTC)

```

▼ Wake UP
    Wake Up Clock          RTCCLK / 16

```

```
struct tm stdcTmSavedTime;
```

Par la suite, j'initialise une structure « `stdcTmSavedTime` » (pour récupérer la dernière mesure de la température) de type « `tm` » qui est une structure de base du langage C.



```

struct tm
{
    int    tm_sec;
    int    tm_min;
    int    tm_hour;
    int    tm_mday;
    int    tm_mon;
    int    tm_year;
    int    tm_wday;
    int    tm_yday;
    int    tm_isdst;
}

```

Cette structure comporte également des paramètres comme les secondes, minutes, heures, jour de la semaine, jour, mois, année etc... Afin de permettre de regrouper toutes les informations nécessaires dans un seul et même endroit au lieu de 2 auparavant.

```

time_t stdcNow;
time_t stdcSavedTime;
struct tm stdcTmNow;

stdcTmNow.tm_hour = now.Hours;
stdcTmNow.tm_min = now.Minutes;
stdcTmNow.tm_sec = now.Seconds;
stdcTmNow.tm_mday = today.Date;
stdcTmNow.tm_mon = today.Month - 1; // tm_mon commence avec Janvier = 0 et Décembre = 11
stdcTmNow.tm_year = today.Year + 100; // tm_year indique une année depuis 1900

```

Ensuite, j'initialise 2 variables de type « time\_t » qui retourne une valeur en secondes.

J'initialise également une structure « stdcTmNow » de type « tm » également, pour prendre la valeur du capteur de choc lorsqu'il y aura un choc détecté surpassant le seuil.

Puis j'affecte mes valeurs précédemment obtenues par l'utilisateur à la variable stdcTmNow en lui passant les paramètres de la structure c « tm ». Quelques modifications sont faites au vu de l'incompatibilité légère concernant certaines valeurs telles que le mois ou l'année. Par exemple, si on avait omis d'enlever 1 au mois, en spécifiant 3 pour le mois de mars, on se serait trouvé en avril au lieu de mars. Et quant à l'année, si on spécifiait 23, on aurait eu 1923 au lieu de 2023.

(La même chose est faite plus bas avec les mêmes affectations, seule la variable stdcTmNow est changée par stdcTmSavedTime pour sauvegarder la dernière heure sur laquelle la dernière mesure de température a été faite.

```

stdcNow = mktime(&stdcTmNow);
stdcSavedTime = mktime(&stdcTmSavedTime);

time_t deltaT = difftime(stdcNow, stdcSavedTime);
printf("deltaT = %d\n", (int)deltaT);

standbyDuration = 32768 / 16 * (STANDBY_DURATION - deltaT);
printf("Nouvelle durée de standby : %d \n", (int)(STANDBY_DURATION - deltaT));

```

Pour la dernière partie du programme de sauvegarde, la variable stdcNow permettra de changer le type de stdcTmNow (qui était précédemment une structure) avec la fonction mktime qui retourne un timestamp (le nombre de secondes écoulées depuis le 1<sup>er</sup> janvier 1970 à 00h00) et la même chose avec stdcSavedTime.

J'initialise ensuite la variable deltaT qui utilise la fonction difftime, permettant littéralement de faire la différence de temps entre deux valeurs décimales.

Pourquoi utiliser mktime ? Tout simplement car c'est bien plus simple de soustraire 2 valeurs décimales plutôt que de soustraire des valeurs dans des structures.

Maintenant, le résultat du programme après instauration de cette nouvelle fonctionnalité :

```
Saisir la date avec le format jj/mm/aa : 23/03/23
Saisir le jour de la semaine (0=Lundi -> 6=Dimanche) : 4
Saisir l'heure avec le format hh:mm:ss : 14:08:00

Let's go !

17/03/2017 @ 14:08:00 :
    Tcelsius : +23.73 / HR% : +49 / G(x:-0.155869, y:-0.654052, z:-1.71186)
deltaT = 4
    Nouvelle durée de standby : 1

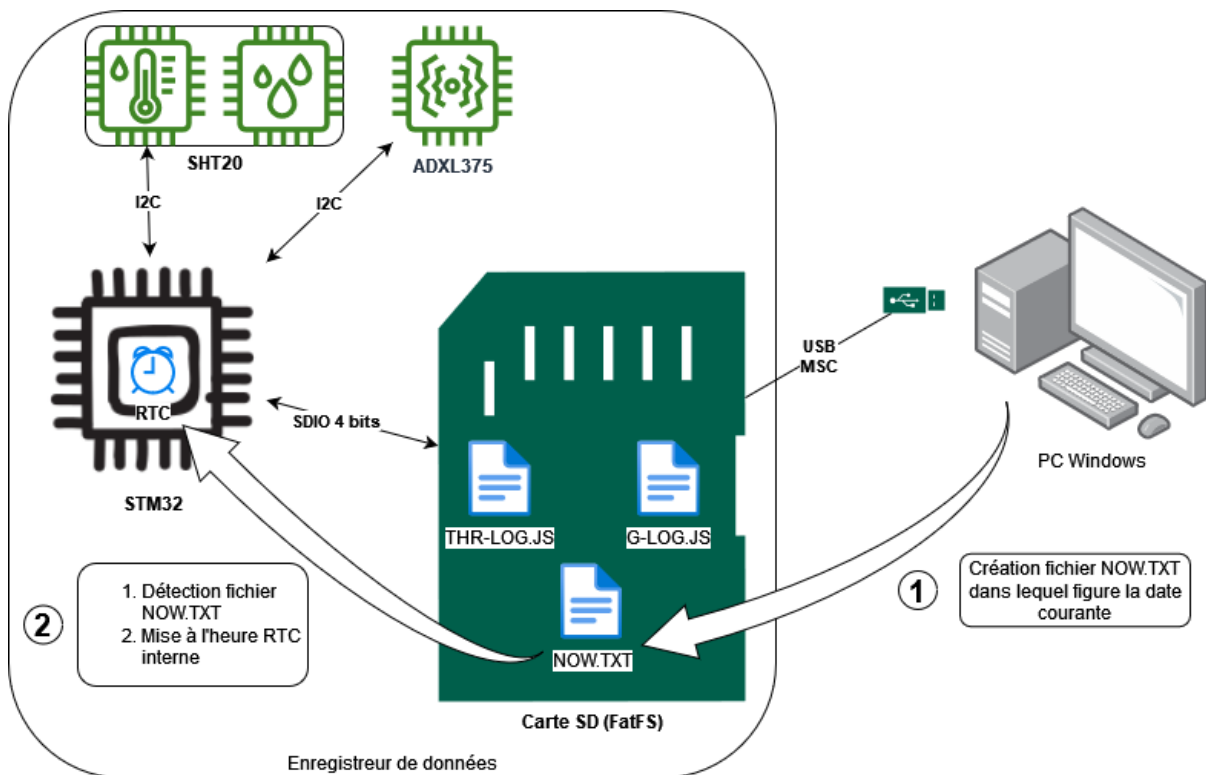
!! Choc TAP-Z TAP-Y TAP-X !! (@14:08:04)

17/03/2017 @ 14:08:05 :
    Tcelsius : +23.73 / HR% : +49 / G(x:-0.302869, y:-0.605052, z:-1.76086)
17/03/2017 @ 14:08:10 :
    Tcelsius : +23.75 / HR% : +48 / G(x:-0.204869, y:-0.654052, z:-1.71186)
█
```

Comme on peut le voir, une fois que le choc est détecté, le deltaT est calculé (la valeur écoulée entre la dernière mesure de température, d'humidité et de choc périodique), dans ce cas il vaut 4, et étant donné que le standby par défaut est 5, il restera 1s de standby avant de reprendre une mesure (donc, prochaine mesure périodique à 14 :08 :05, comme prévu s'il n'y avait pas eu de chocs !

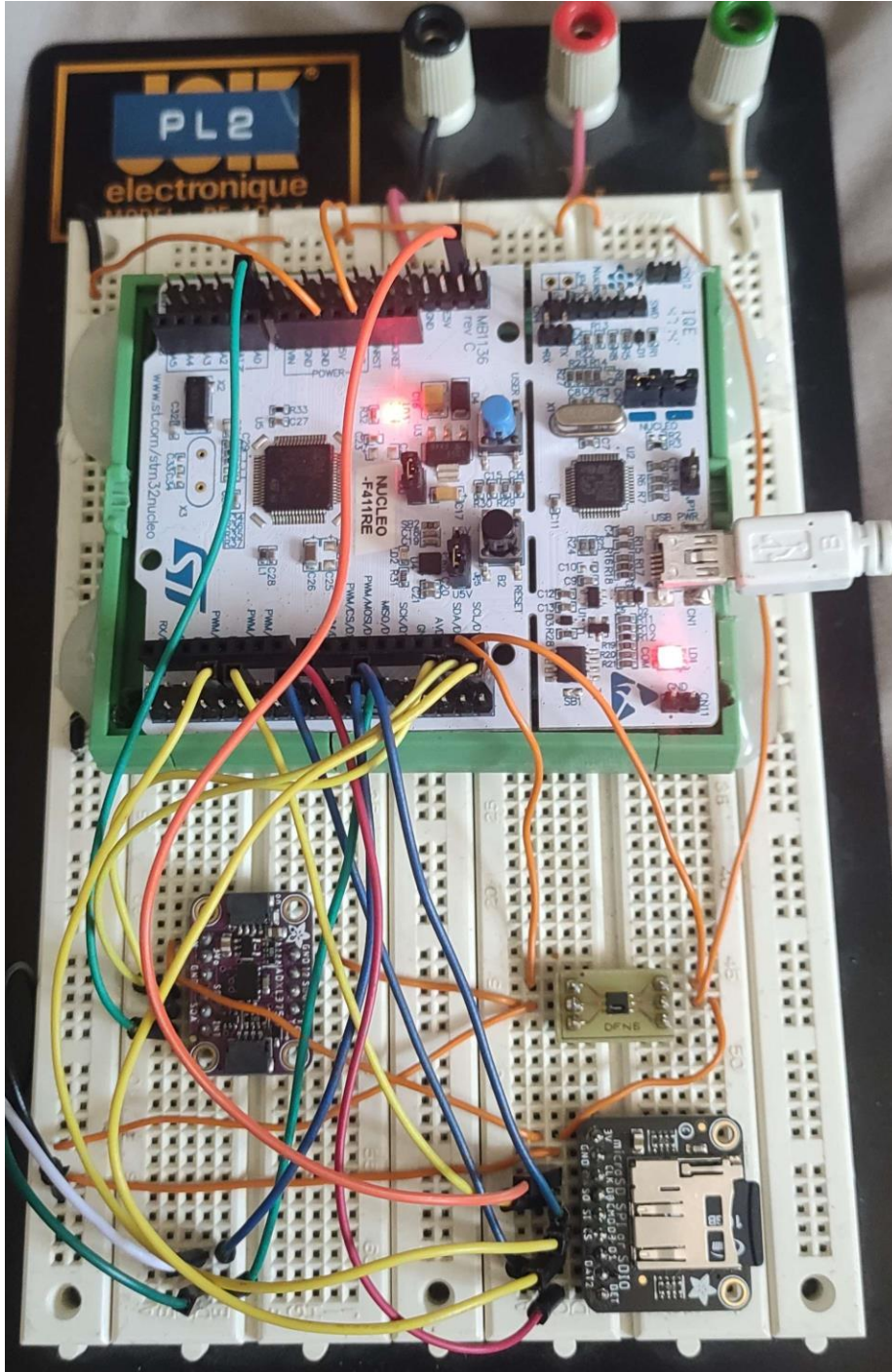
### Sauvegarde des données dans une carte SD externe

Le schéma ci-dessous montre l'étendue de l'évolution du projet proposée concernant la sauvegarde des données. Le programme va créer un fichier « NOW.TXT » qui contiendra la date actuelle, et une fois ce fichier détecté, la RTC interne se mettra à l'heure, proposant des mesures horodatées correctement dans les fichiers de logs THR-LOG et G-LOG.

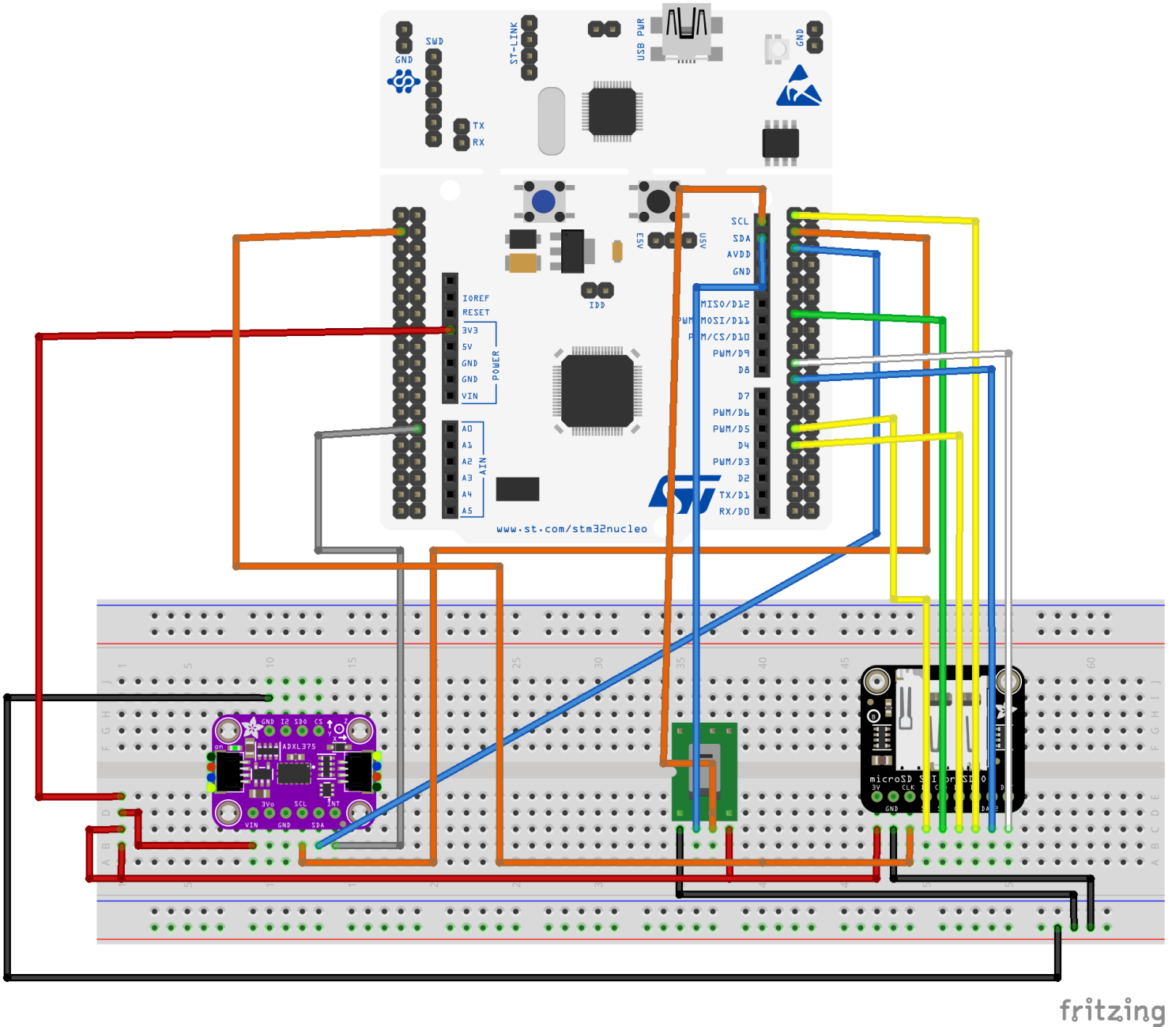


### Câblage – Partie « physique »

Durant ce projet, j'ai consacré une partie de mon temps à faire un câblage propre et efficace pour effectuer mes tests. J'ai donc utilisé une platine d'essai, sur laquelle nous avons collé un support dans lequel la carte STM32F411RE est installée. J'ai ensuite fixé les capteurs (température, chocs) et la SD card sur la platine d'essai afin de pouvoir câbler par la suite au maximum des fils de connexion dits « durs » afin d'en adapter la taille au besoin et d'utiliser ses inconvénients (à savoir d'être difficilement repositionnable) pour faire un câblage propre et plus facilement lisible.



Le schéma Fritzing ci-dessous est associé à ce câblage a de ce fait également été créé pour faciliter la vue des câbles, difficiles à repérer par photo. Dans ce schéma on peut repérer en rouge les câbles liés à l'alimentation en 3V3, en noir les câbles liés à la masse (GND), et les autres couleurs pour éviter les répétitions de couleurs, seront de couleurs assez multicolores.



## Conclusion

Ce projet m'a permis d'apprendre à manier un nouvel environnement de développement qui est STM32CubeIDE. J'ai également appris à lire une datasheet. J'apprends également à gérer un projet et essayer de palier aux imprévus directs qui peuvent surgir.



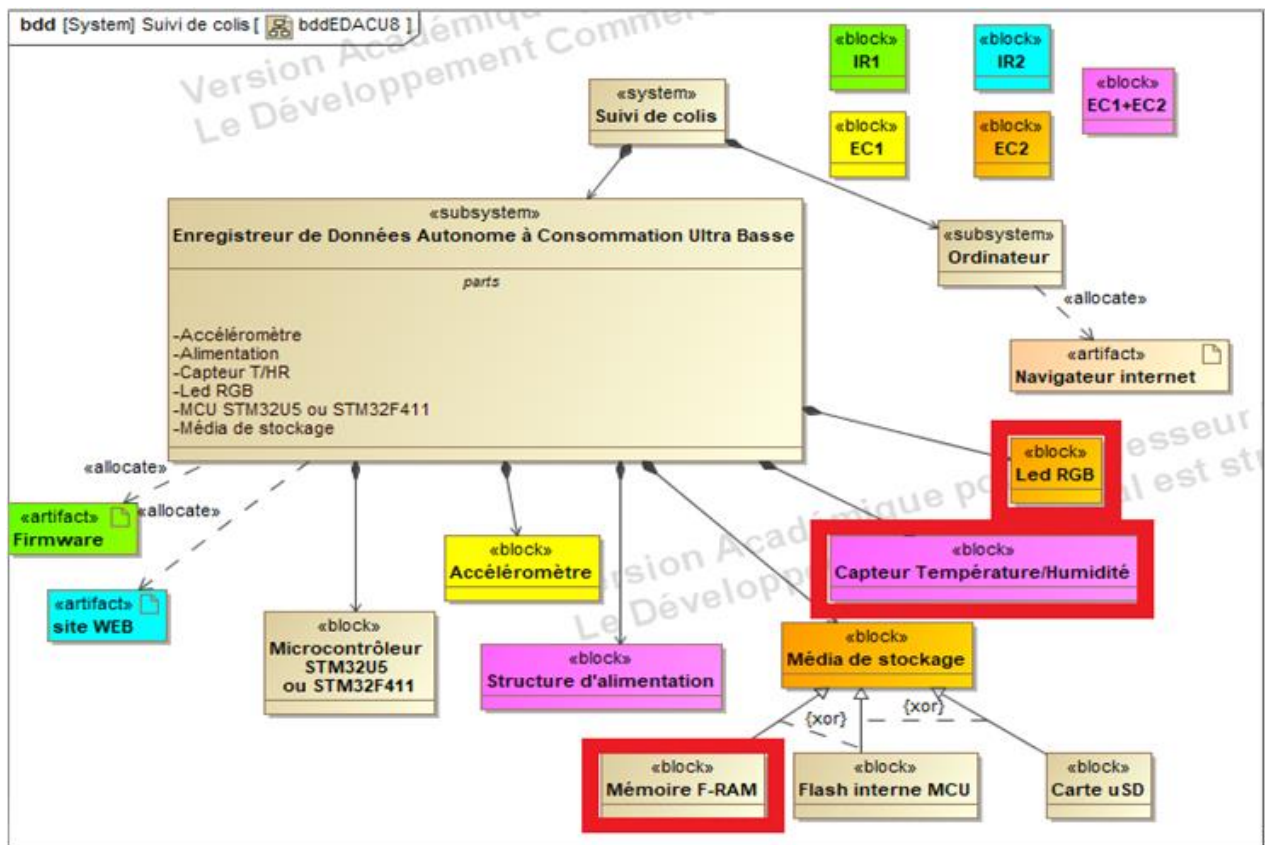
# Partie EC2 : TNIFAS Abdessalam

## Présentation du rôle du candidat

Ma partie concernant le projet vise à enregistrer la température ainsi que l'humidité relative avec le SHT21 pour assurer le suivi des conditions de transports de colis fragiles, j'ajouterai une LED RGB permettant d'indiquer :

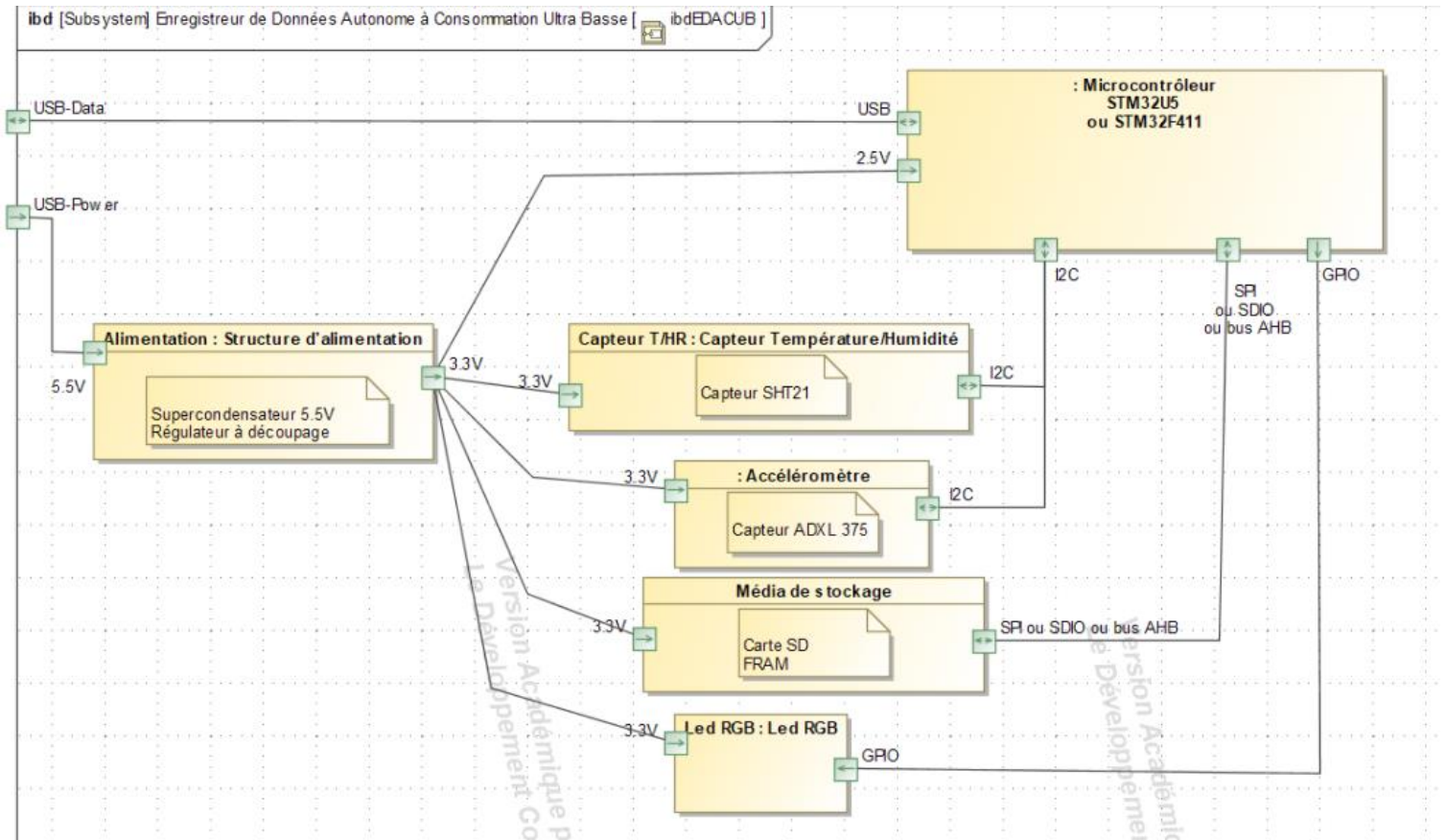
- L'état du colis pendant le trajet s'il reçoit certains chocs ou non.
- L'état de l'enregistrement lorsque le data logger n'est pas connecté au PC (démarré, stoppé, capacité max).

Ensuite la capacité d'enregistrement sera au minimum de 30 000 mesures (1 mesure toutes les 10s pendant 72h), pour que les mesures soient horodatées je choisirai un quartz. Pour finir, je mettrai en œuvre une mémoire FRAM qui sera associée au microcontrôleur.



## L'IBD de la carte :

Je me suis inspiré du schéma structurel pour faire ce diagramme, où l'on aperçoit chaque liaison avec son composant





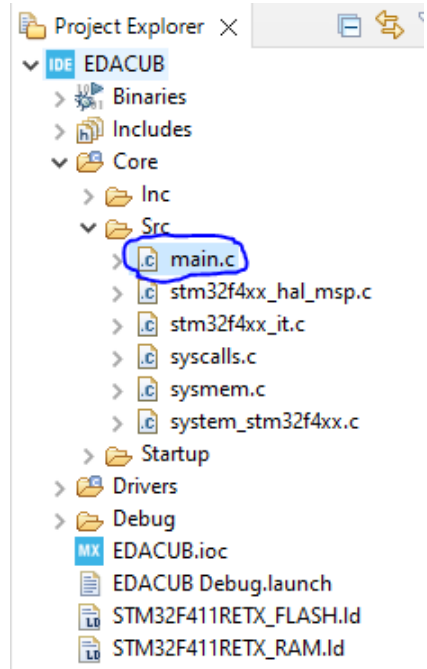


C'est sur ce logiciel que l'on va pouvoir travailler sur la carte NUCLEO, je me suis servi des vidéos mise à disposition pour pouvoir par exemple faire clignoter une LED que vous allez voir plus tard.

### Pourquoi avoir choisi la STM32F11RE?

Notre choix va se porter sur la *STM32F11RE* de la gamme « High Performance » qui possède une communauté largement plus active que la *STM32U5* avec des bibliothèques Open Source très utiles à notre projet. Nous avons plus d'informations sur la *STM32F11RE*.

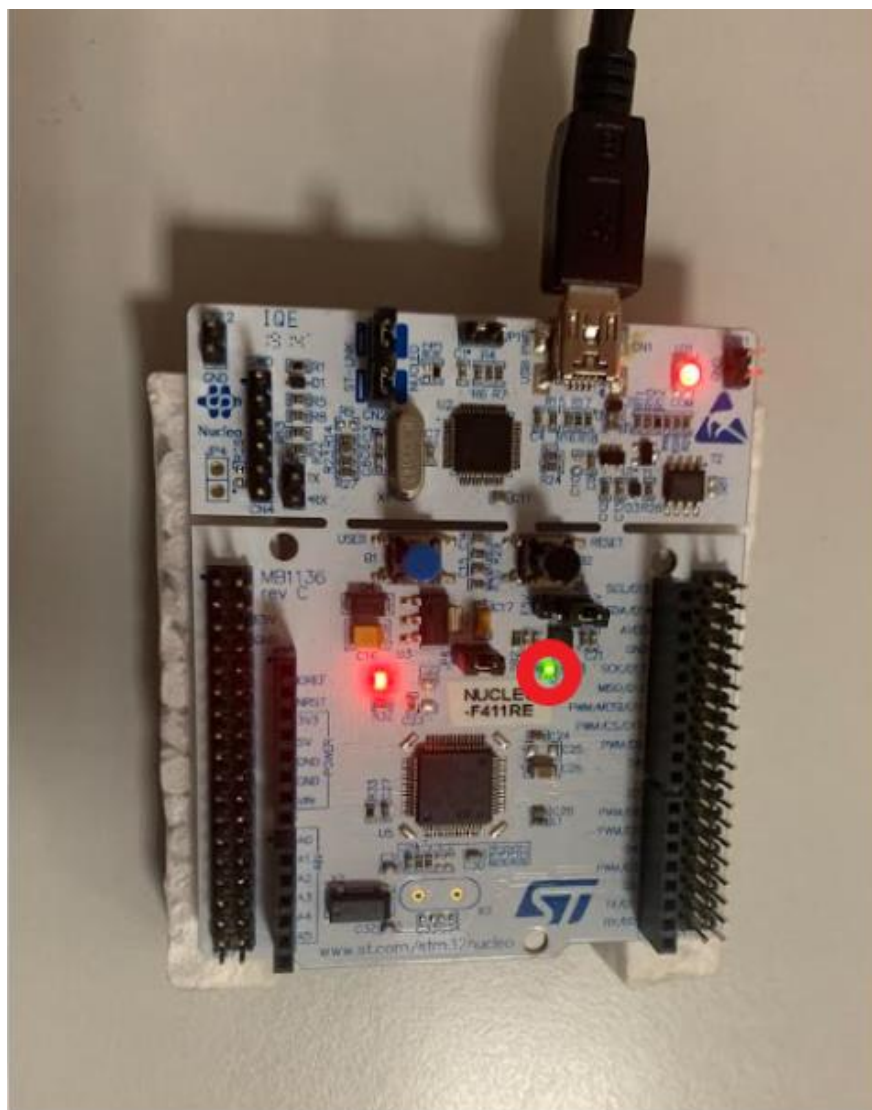
Nous allons en revanche devoir sacrifier une exigence du projet qui concerne l'autonomie de 72h qui nous sera probablement plus possible d'assurer.



### Test de la carte en faisant clignoter la LED CMS :

```
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
  HAL_Delay(500);
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
  HAL_Delay(200);
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
  HAL_Delay(200);
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
  HAL_Delay(200);
}
}
```



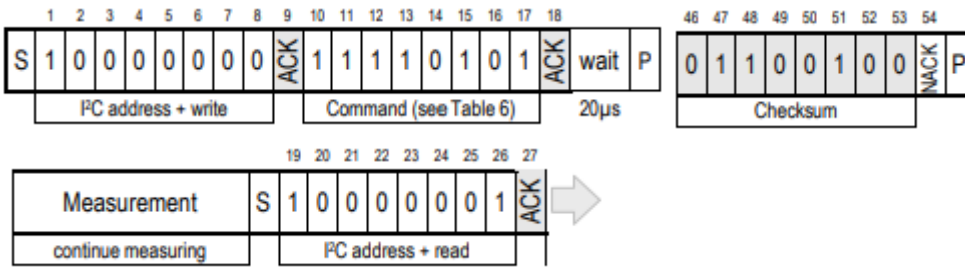
Capteur SHT21 de température et d'humidité:



Pin	Name	Comment	
1	SDA	Serial Data, bidirectional	
2	VSS	Ground	
5	VDD	Supply Voltage	
6	SCL	Serial Clock, bidirectional	
3,4	NC	Not Connected	

Les broches 1 et 6 «SDA et SCL» concernent le bus I2C, la broche 2 est le «GND», et pour finir la broche est l'alimentation, son adresse en hexa est 0x40.

The supply voltage of SHT2x must be in the range of 2.1 – 3.6V, recommended supply voltage is 3.0V. Power supply



Hold Master et No Hold Master:

En mode Hold Master la communication SCL est bloquée par le capteur pendant que le capteur fait la mesure, donc il ne peut rien faire d'autre mais la communication est plus rapide tandis que le mode No Hold Master, la communication SCL reste ouverte aux autres communications pendant la prise de mesure donc les deux sont intéressants mais le No Hold Master pourrait être plus exploité.

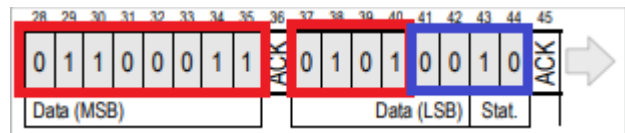
Command	Comment	Code
Trigger T measurement	hold master	1110'0011
Trigger RH measurement	hold master	1110'0101
Trigger T measurement	no hold master	1111'0011
Trigger RH measurement	no hold master	1111'0101
Write user register		1110'0110
Read user register		1110'0111
Soft reset		1111'1110

Comment calculer l'Humidité Relative:

$$RH = -6 + 125 \cdot \frac{S_{RH}}{2^{16}}$$

Ainsi que la Température:

$$T = -46.85 + 175.72 \cdot \frac{S_T}{2^{16}}$$



Valeurs obtenues du capteur SHT21:

Code pour obtenir la température :

```
// Mesure température
    buf[0] = Temp_no_Hold_Master;
    ret = HAL_I2C_Master_Transmit(&hi2c1, SHT21_ADDR, buf, 1,
HAL_MAX_DELAY);
    if (ret !=HAL_OK){
        strcpy((char*)buf, "Erreur Tx\r\n");
    } else {
        HAL_Delay(200); // Delai indispensable
        ret = HAL_I2C_Master_Receive(&hi2c1, SHT21_ADDR, buf, 2,
HAL_MAX_DELAY);
        if (ret !=HAL_OK){
            strcpy((char*)buf, "Erreur Tx\r\n");
        }
    }

```

```

    }
    else {
        valTemp = (((int16_t)buf[0]<<8) | (buf[1]&0b11111100));
    }
temp = (-46.85 + 175.72 * valTemp/65536);

temp *=100;
sprintf((char*)buf,
        "%u.%02u C\r\n",
        ((unsigned int)temp / 100),
        ((unsigned int) temp % 100));

```

36.38	%
21.45	C
36.35	%
21.45	C
36.35	%
21.44	C
36.32	%
21.45	C
36.32	%
21.45	C
36.29	%
21.45	C
36.26	%
21.44	C
36.23	%
21.44	C
36.23	%
21.44	C

### Code pour obtenir l'humidité:

```

// Mesure humidité relative
buf[0] = HR_no_Hold_Master;
ret = HAL_I2C_Master_Transmit(&hi2c1, SHT21_ADDR, buf, 1,
HAL_MAX_DELAY);
if (ret !=HAL_OK){
    strcpy((char*)buf, "Erreur Tx\r\n");
} else {
    HAL_Delay(200); // Delai indispensable
    ret = HAL_I2C_Master_Receive(&hi2c1, SHT21_ADDR, buf, 2,
HAL_MAX_DELAY);
if (ret !=HAL_OK){
    strcpy((char*)buf, "Erreur Tx\r\n");
}
}

```

```

    }
    else {
        valHR = (((int16_t)buf[0]<<8) |
(buf[1]&0b11110000));
    }

```

```
HR = (-6 + 125.00 * valHR/65536);
```

```

36.38 %
21.45 C
36.35 %
21.45 C
36.35 %
21.44 C
36.32 %
21.45 C
36.32 %
21.45 C
36.29 %
21.45 C
36.26 %
21.44 C
36.23 %
21.44 C
36.23 %
21.44 C

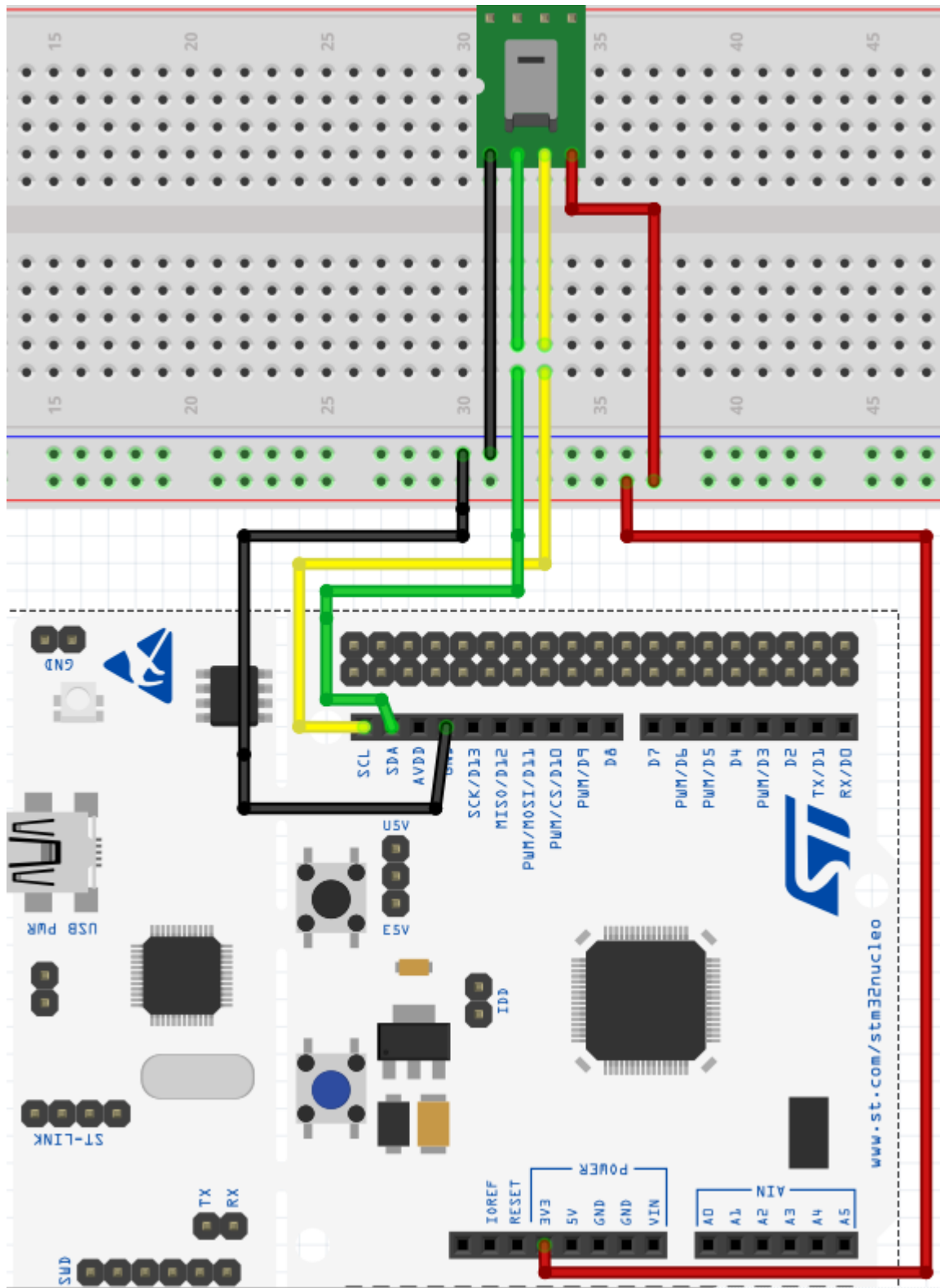
```

```

HR *=100;
sprintf((char*)buf,
        "%u.%02u %%\r\n",
        ((unsigned int)HR / 100),
        ((unsigned int) HR % 100));

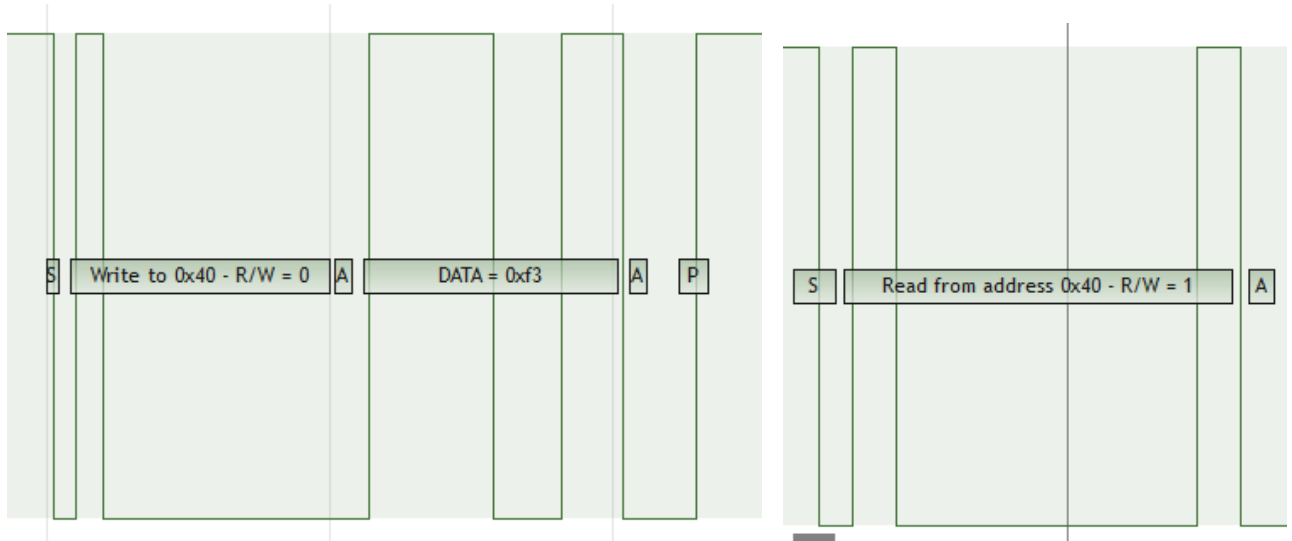
```

## Schéma Fritzing:

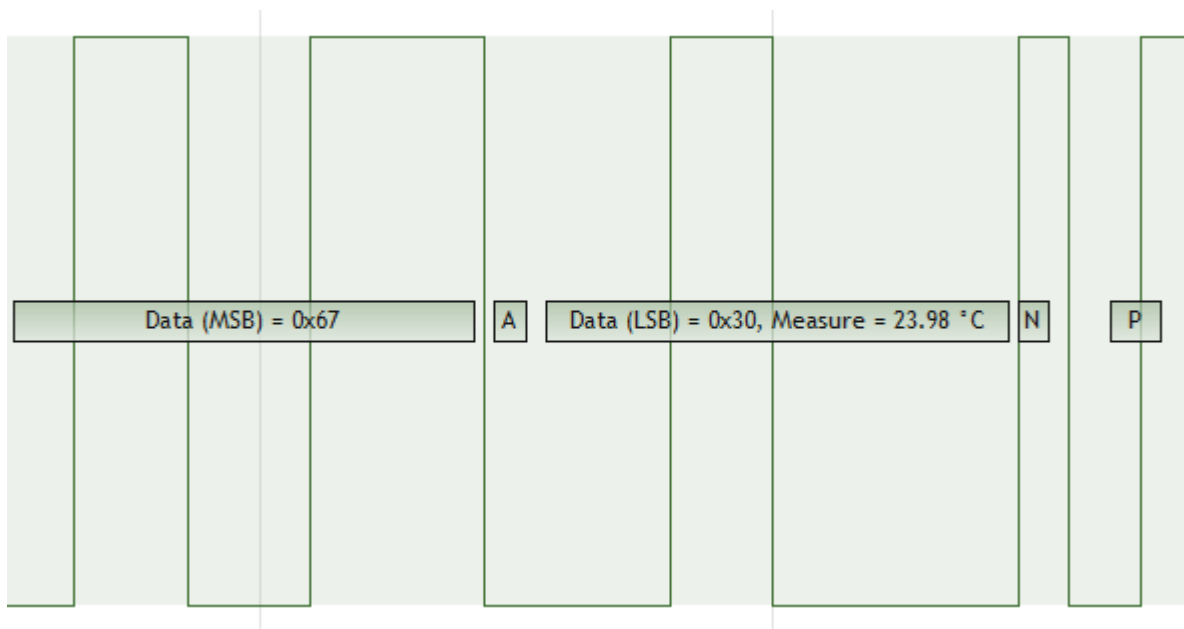




## Relevé de trame:



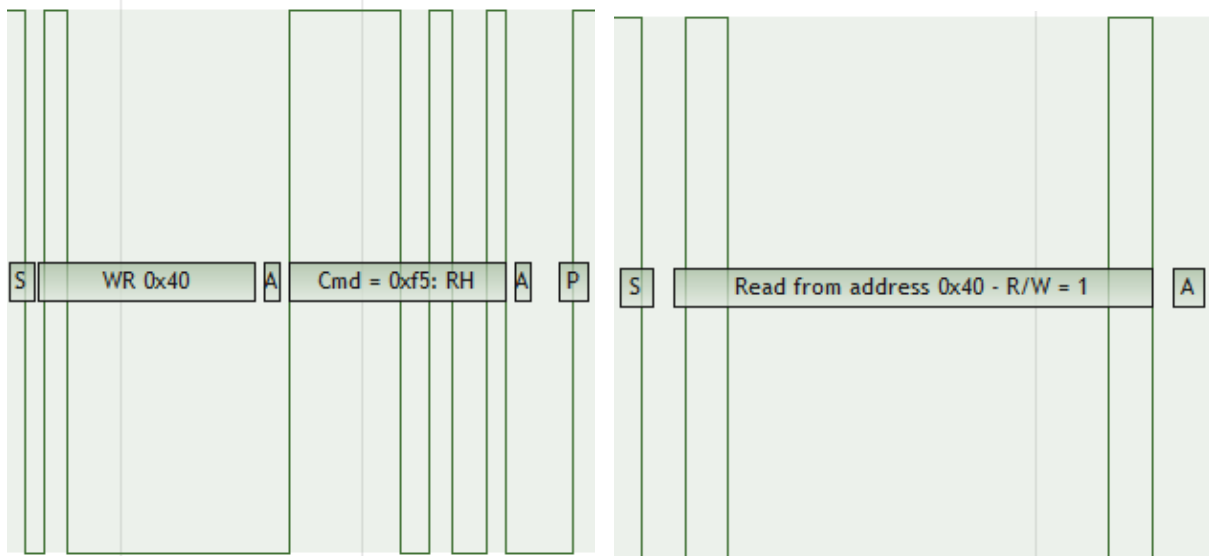
Dans ce chronogramme, il écrit sur l'adresse du capteur «40», suivie d'un acknowledge, commande du «no hold Master, en température» avec F3, ensuite il compte écrire sur l'adresse du capteur.



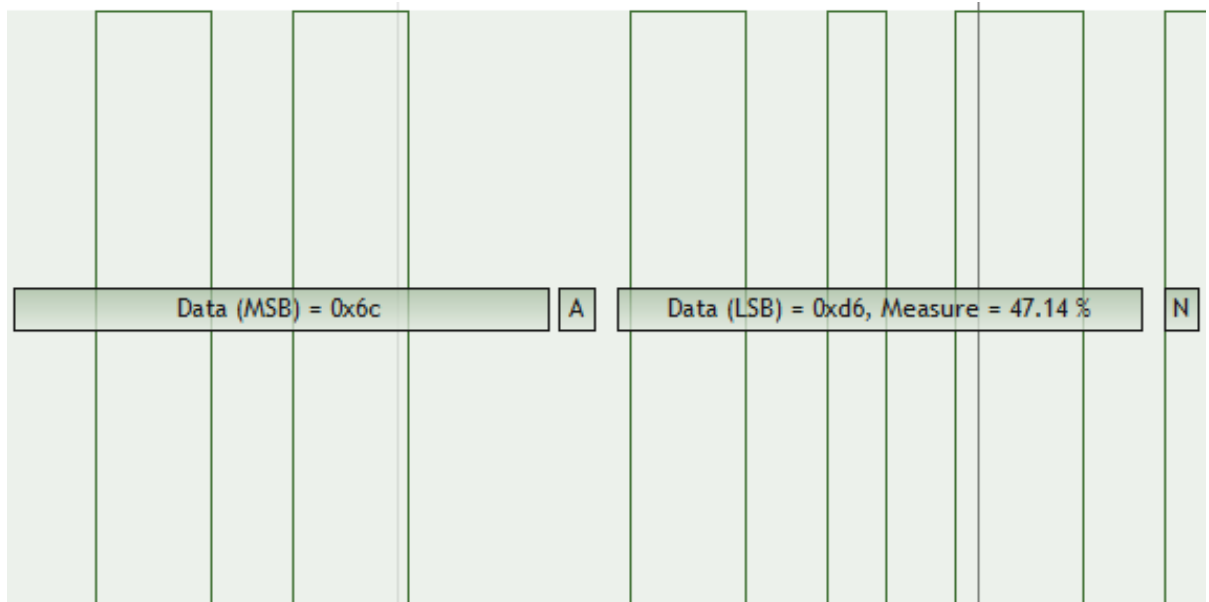
Dans celui-là, il lit sur l'adresse du capteur les valeurs "0x67" et "0x30", ce qui va donner 23,98°C la température.

On prend les 4 octets "6730" =  $26\ 416 / 2^{16} = 0,403 * 175.72 - 46.85 = 23.98^{\circ}\text{C}$ .

$$T = -46.85 + 175.72 \cdot \frac{S_T}{2^{16}}$$



Ici nous retrouvons la commande F5 pour l'humidité relative en mode "no Hold Master", ainsi que l'écriture sur le capteur.



Il lit les datas suivantes: 0x6C et 0xD6 pour l'humidité relative qui est de 47.14%, finit par un Nonacknowledge.

"6CD0"(buf[1]&0b11110000); ,les 4 bits de poids faible sont à 0, le 0 vient de là.

"6CD0" = 27 856 / 2<sup>16</sup> = 0,425 \* 125 - 6 = 47,13%

$$RH = -6 + 125 \cdot \frac{S_{RH}}{2^{16}}$$

On aperçoit les onglets détaillés avec l'adresse du capteur et les résultats obtenus

∨ Views

☰ PacketView

---

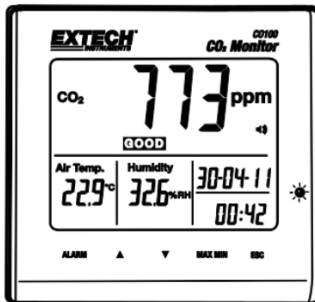
I2C 0x40 W[1]	CH4
Start	
Address	Write to 0x40
Data[0]	0xf3
Stop	
I2C Sensor	CH4
Start	
Address	Write to address 0x40
Command	Temperature Measure (no hold Master)
Stop	
I2C 0x40 R[2]	CH4
Start	
Address	Read from 0x40
Data[0]	0x67
Data[1]	0x30
Nack	
Stop	
I2C Sensor	CH4
Start	
Address	Read from address 0x40
Measure (T)	23.98 °C
NACK	
Stop	
I2C 0x40 W[1]	CH4
Start	
Address	Write to 0x40
Data[0]	0xf5
Stop	
I2C Sensor	CH4
Start	
Address	Write to address 0x40
Command	RH Measure (no hold master)
Stop	
I2C Sensor	CH4
Start	
Address	Read from address 0x40
Measure (RH)	47.14 %
NACK	

Vérification avec l'appareil qui mesure l'humidité relative,

la température ainsi que le CO2 :

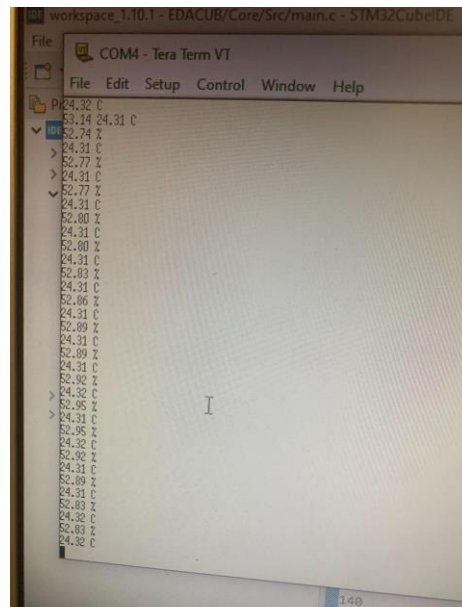
## Testeur de la qualité de l'air intérieur de bureau

Modèle CO100



On reçoit les données suivantes:

- 24.21°C
- 52%



L'appareil nous affiche:

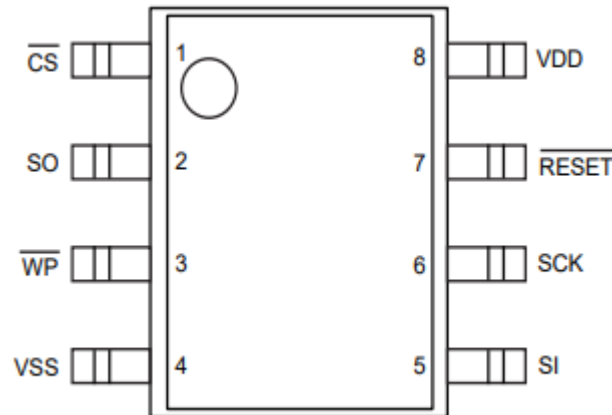
- 22.°C
- 54.3%.

Effectivement il y a un écart de 2°C ainsi que 2%, l'appareil ayant une meilleure précision, les résultats sont donc cohérents.



Mémoire FRAM (MB85RS4MTY):





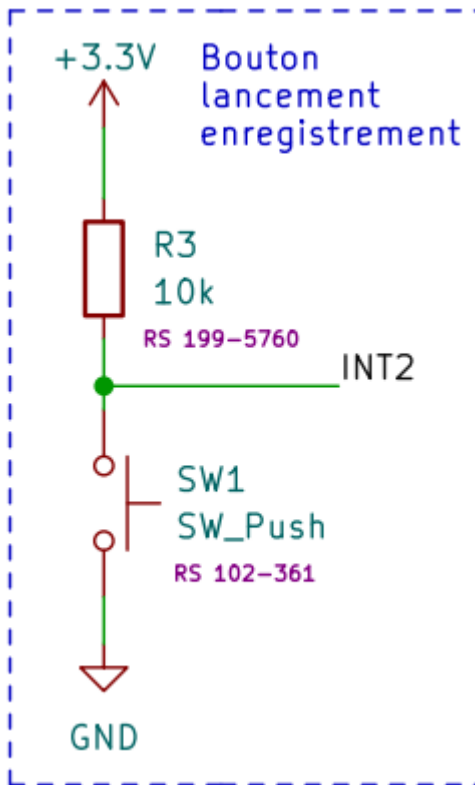
1	$\overline{CS}$	Chip Select pin This is an input pin to make chips select. When $\overline{CS}$ is "H" level, device is in deselect (standby) status and SO becomes High-Z. Inputs from other pins are ignored for this time. When CS is "L" level, device is in select (active) status. CS has to be "L" level before inputting op-code.
3	$\overline{WP}$	Write Protect pin This is a pin to control writing to a status register. The writing of status register (see "■ STATUS REGISTER") is protected in related with $\overline{WP}$ and WPEN. See "■ WRITING PROTECT" for detail.
7	$\overline{RESET}$	RESET pin This is an input pin to make chip reset. When $\overline{RESET}$ is "L" level, device is reset and SPI interface status is initialized.
6	SCK	Serial Clock pin This is a clock input pin to input/output serial data. SI is loaded synchronously to a rising edge, SO is output synchronously to a falling edge.
5	SI	Serial Data Input pin This is an input pin of serial data. This inputs op-code, address, and writing data.
2	SO	Serial Data Output pin This is an output pin of serial data. Reading data of FRAM memory cell array and status register data are output. This is High-Z during standby.

Sa plage d'alimentation est de 1.8 à 3.6V, la température ambiante peut aller de -45° à 125°C, ce composant permettra de stocker les données des capteurs, 524,288 mots à 8 bits.

Bouton poussoir et LED ARBG (Addressable RGB):

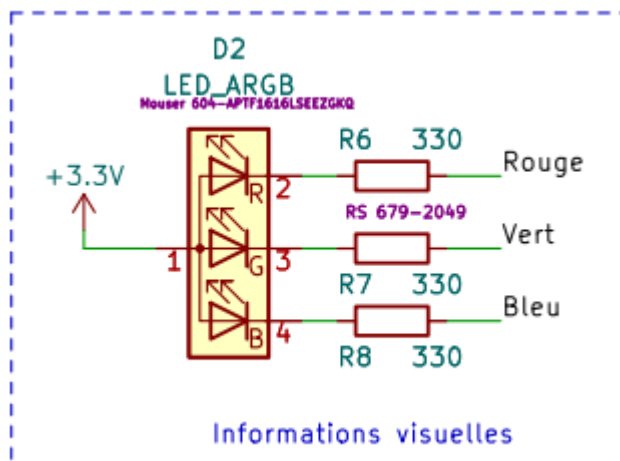


Le schéma est présent ci-dessous :



Relié au 3,3V avec une résistance de 10K $\Omega$ , entre eux l'entrée "INT2" qui sera reliée à la broche 14 du microcontrôleur.

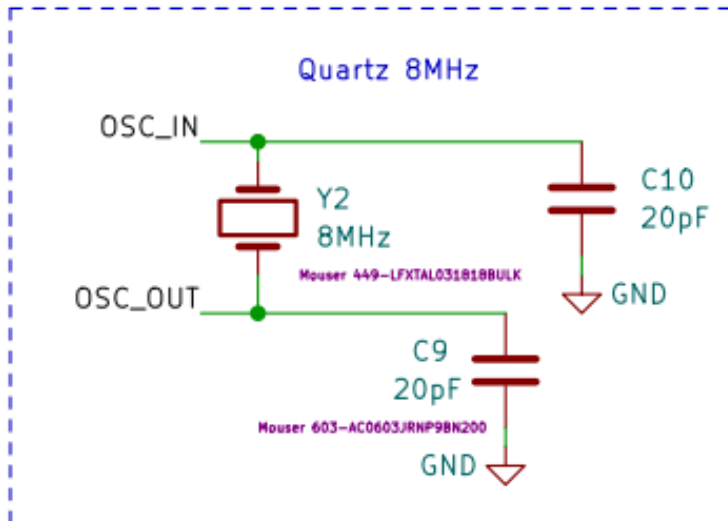
LED ARGB:



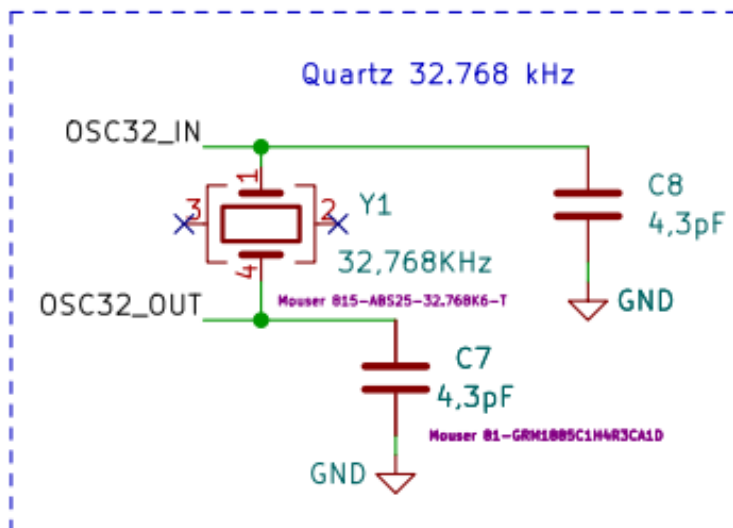
La LED changera de couleur selon le choc du colis pris durant le trajet.

Les deux quartz:

Quartz 8 Mhz:

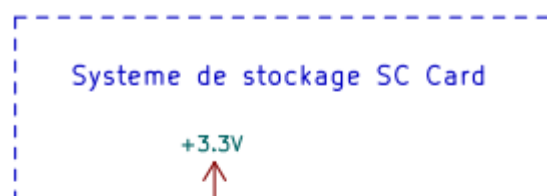


Quartz 32.768 Khz:



Les quartz permettent de donner le tempo à la fréquence choisie.

Nous avons ajouté une carte SD:



Elle accompagnera la mémoire FRAM pour pouvoir stocker les datas.

**Journal de bord:**

<u>Date</u>	<u>TNIFAS</u>	<u>GUILLOT</u>	<u>LAYOUNI</u>	<u>POMARO</u>
03/01/23	J'ai commencé à lire le dossier, et ensuite une petite réunion avec Mr Hortolland pour les tâches à réaliser.	Lecture du dossier et planification du projet	Réunion avec Brice et Mr Defrance pour une présentation complète du projet et des tâches à réaliser.	Réunion avec Noa et Mr Defrance pour une présentation complète du projet et des tâches à réaliser.
04/01/23	Commencement du test avec le <u>ADXL375</u> implanté sur le breakout(avec Kilian)	Découverte du capteur ADXL375, test avec Arduino(avec Abde)	Découverte des trois librairies Javascript capables de tracer des courbes sur une interface web.	Découverte large du logiciel principal de programmation de la carte STM32F411
09/01/23	Codage du capteur sur Arduino pour détecter les chocs du capteur, documentation sur les différentes lignes de code	Codage du capteur sur Arduino pour détecter les chocs du capteur, documentation sur les différentes lignes de code.	Recherche et découverte du fonctionnement de File System Access API pour lire un fichier en local sur une page web sans serveur web.	Découverte d'une librairie pour contrôler le SHT21 et prise en main du GUI permettant le contrôle sur les broches de la carte électronique  Mise hors service d'un capteur SHT21 suite à une erreur de câblage à cause d'une mauvaise indication sur le capteur
10/01/23	Planification d'un tableau pour trouver la concordance du "g", recherche la LED.	Planification d'un tableau pour trouver la concordance du "g".		Création du programme sur l'ADXL375

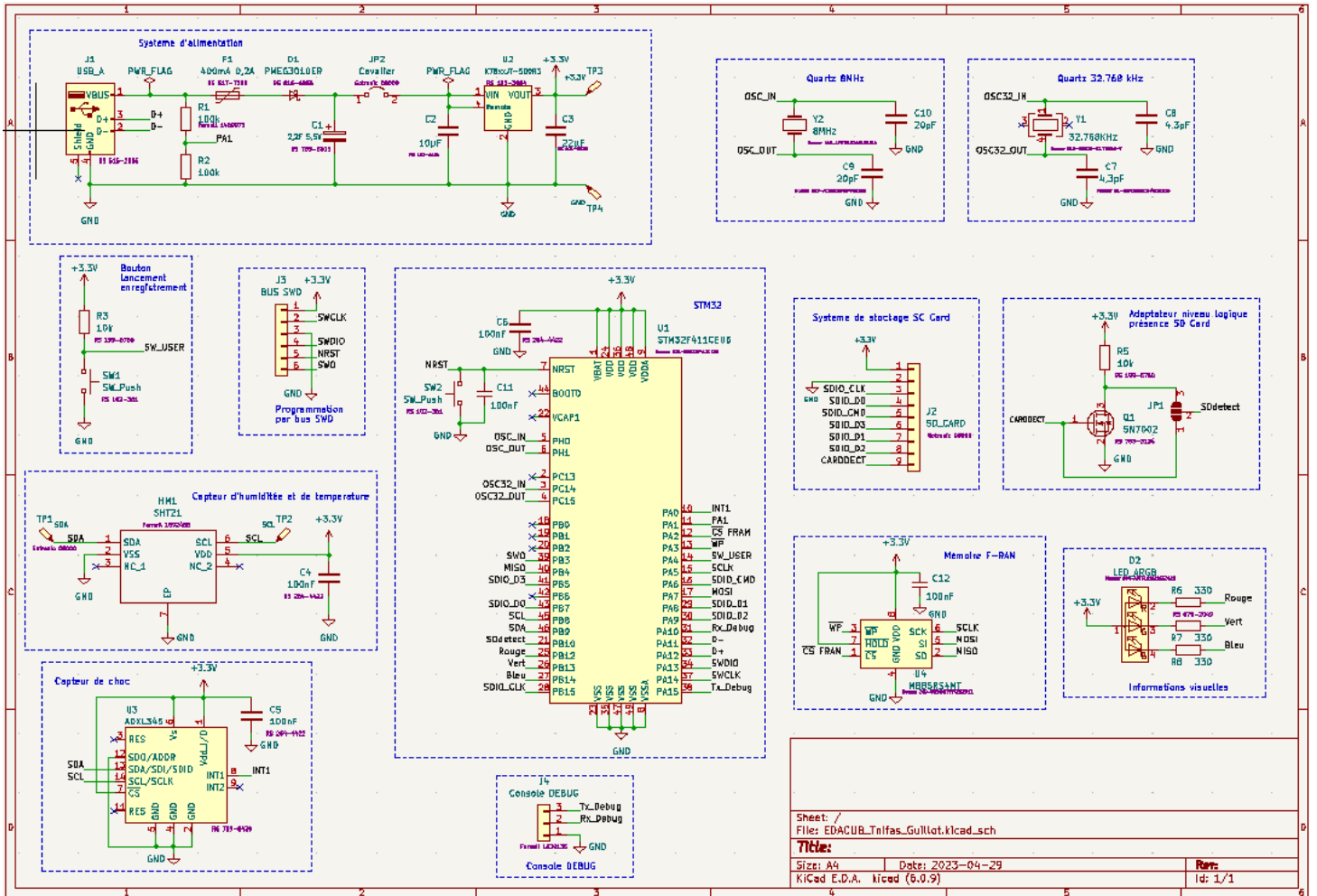
## Planification Partielle:

	→	Planification du projet	480 min?	Mar 03/01/23	Mer 04/01/23	
	→	Lecture du contrat	240 min	Mar 03/01/23	Mer 03/01/23	
	→	Lecture de documentation	240 min?	Mer 04/01/23	Mer 04/01/23	2
	→	Réunion de groupe	60 min	Mar 03/01/23	Mer 03/01/23	
	→	Revue de projet	0 min	Mar 03/01/23	Mer 03/01/23	
	→	Se documenter sur le STM32	120 min	Mar 03/01/23	Mer 03/01/23	
	→	Réunion avec Kilian pour s'occuper du microcontrôleur STM32	420 min?	Mar 03/01/23	Mer 04/01/23	
	→	Choisir une LED et tester sur le microcontrôleur	420 min?	Mar 03/01/23	Mer 04/01/23	
	→	Se documenter sur les mémoires de type FRAM	420 min?	Mar 03/01/23	Mer 04/01/23	
	→	Choisir un quartz de la carte	420 min?	Mar 03/01/23	Mer 04/01/23	
	→	Regroupement avec Kilian pour associer nos 2 parties (dimension du supercondensateur)	420 min?	Mar 03/01/23	Mer 04/01/23	
	→	Router le circuit (Format clef USB)	420 min?	Mar 03/01/23	Mer 04/01/23	
	→	Câbler la carte	420 min?	Mar 03/01/23	Mer 04/01/23	

## Planification Réelle :

♣ Planification du projet			
Lecture du contrat	660 min	Mar 03/01/23	Lun 09/01/23
Lecture de documentation	240 min	Mar 10/01/23	Mar 10/01/23
Se documenter sur le STM32	2220 min?	Mar 10/01/23	Mar 31/01/23
Essai et test sur le capteur ADXL375 de (choc) avec Kilian	2220 min	Mer 01/02/23	Mer 22/02/23
Choisir une LED et la tester sur le microcontrôleur STM32	240 min	Lun 27/02/23	Mar 28/02/23
Essai sur le SHT21	1800 min?	Mar 28/02/23	Mer 15/03/23
Regarder les vidéos	320 min?	Mer 15/03/23	Lun 20/03/23
Se documenter sur les mémoires de type FRAM et les comparer avec les autres type.	840 min	Lun 20/03/23	Lun 27/03/23
Choisir un quartz pour la carte	1740 min?	Lun 27/03/23	Mar 11/04/23
Réunion de groupe	660 min?	Mar 11/04/23	Lun 17/04/23
Proposer un schéma structurel regroupant les deux parties.	2400 min?	Lun 17/04/23	Mar 09/05/23
Router le circuit (Format clef USB)	660 min?	Mar 09/05/23	Lun 15/05/23
Câbler la carte et tester	1080 min	Lun 27/03/23	Mar 04/04/23

# Schéma structurel du projet :

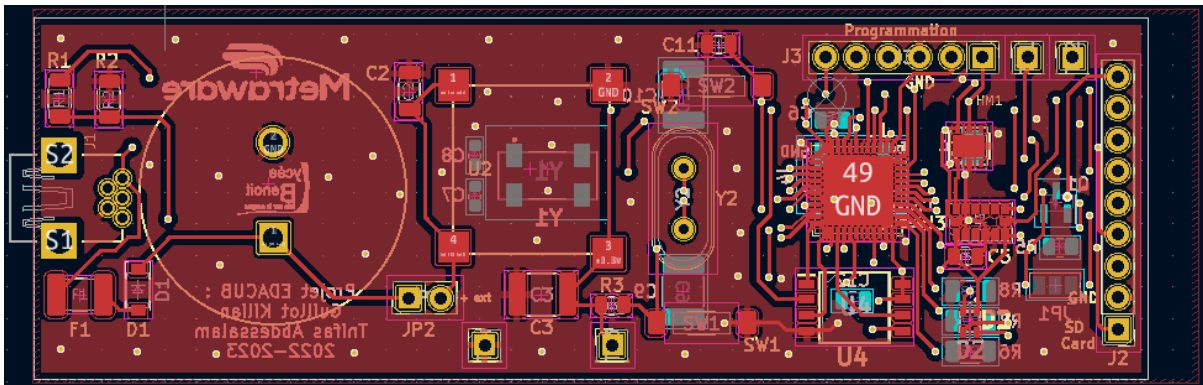


Sheet: /	Date: 2023-04-29	Rev:
File: EDACUB_Tnifas_Gullot.kicad_sch		
Title:		
Size: A4	Date: 2023-04-29	Rev:
KiCad E.D.A. kicad (6.0.9)		Id: 1/1

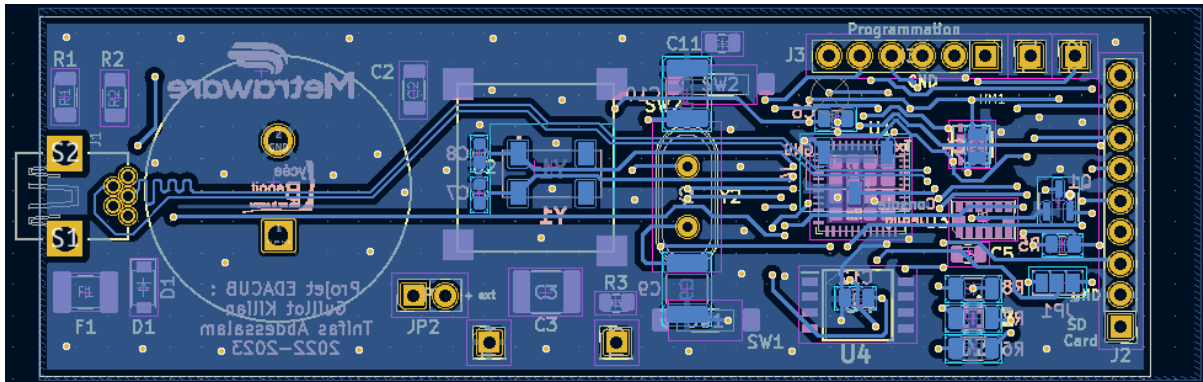


## Routing:

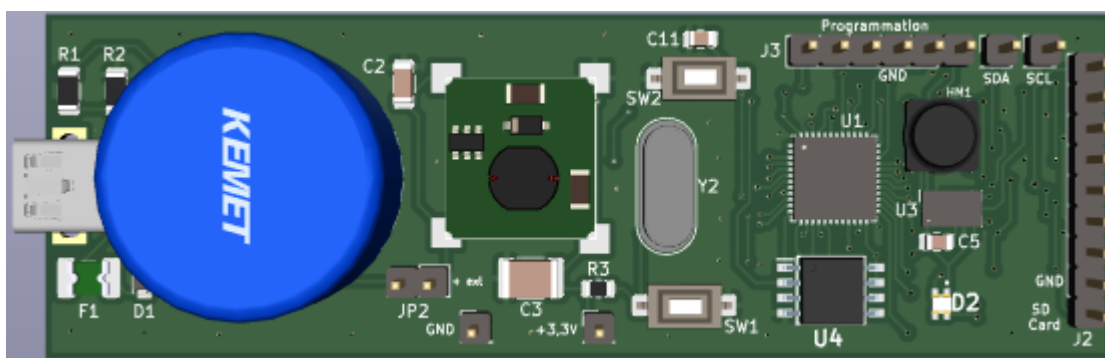
### Routing de la carte couche top:



### Routing de la carte couche bottom:



### Affichage 3D:



## La Soudure : Les étapes.

### 1<sup>ère</sup> étape : Liste de matériel et distribution du matériel

La première étape du soudage est d'établir une liste de matériel qui permet de définir les composants que l'on soudera, puis dans quel ordre on les soudera. Notre professeur nous distribuera les composants.

Liste des composants :	Quantité :	Value	Nom des composants	Vendeur	Code commande	Prix UHT	Prix TTC
C1,	1	2,2F 5,5V	C_Polarized	Rs component	RS 789-8015	6,12	7,34
C2,	1	10µF	C	Rs component	RS 103-4164	0,291	0,349
C3,	1	22µF	C	Rs component	RS 601-0536		
C4, C5, C6, C11, C12,	5	100nF	C_Small	Rs component	RS 264-4422	0,139	0,167
C7, C8,	2	4,3pF	C	Mouser	81-GRM1885C1H4R3CA1D	0,113	
C9, C10,	2	20pF	C	Mouser	603-AC0603JRN9BN200	0,113	
D1,	1	PMEG3010ER	D_Schottky_Small	Rs component	RS 816-6855	0,387	0,464
D2,	1	LED_ARGB	LED_ARGB	Mouser	604-APTF1616LSEEZGKQ	0,818	
F1,	1	400mA 0,2A	Polyfuse	Rs component	RS 517-7105	0,342	0,41
HM1,	1	SHT21	SHT21	Farnell	Farnell 1892485	2,22	
J1,	1	USB_A	USB_A	Rs component	RS 515-2005	2,17	2,6
J2,	1	SD_CARD	Conn_01x09	Gotronic	Gotronic 08000	0,5	0,6
J3,	1	BUS SWD	Conn_01x06	Gotronic	Gotronic 08000	0,5	0,6
J4,	1	Console DEBUG	Conn_01x03	Farnell	Farnell 1629135	8,11 €	
JP1,	1	SolderJumper_3_Open	SolderJumper_3_Open				
JP2,	1	Cavalier	Jumper_2_Open	Gotronic	Gotronic 08000	0,5	0,6
Q1,	1	SN7002	2N7002	Rs component	RS 753-3134		
R1, R2,	2	100k	R	Farnell	Farnell 1469975	0,079	
R3, R5,	2	10k	R	Rs component	RS 199-5760	0,011	0,013
R6, R7, R8,	3	330	R	Rs component	RS 679-2049	0,082	0,098
SW1, SW2,	2	SW_Push	SW_Push	Rs component	RS 102-361	0,568	0,682
TP1,	1	SDA	TestPoint_Probe	Gotronic	Gotronic 08000	0,5	0,6
TP2,	1	SCL	TestPoint_Probe	Gotronic	Gotronic 08000	0,5	0,6
TP3,	1	+3,3V	TestPoint_Probe	Gotronic	Gotronic 08000	0,5	0,6
TP4,	1	GND	TestPoint_Probe	Gotronic	Gotronic 08000	0,5	0,6
U1,	1	STM32F411CEU6	STM32F411CCUx	Mouser	511-STM32F411CEU6	8,75	
U2,	1	K78xxJT-500R3	K78xxJT-500R3	Rs component	RS 193-3954	2,94	3,53
U3,	1	ADXL345	ADXL345	Rs component	RS 709-8439	8,23	9,88
U4,	1	MB85RS4MT	MB85RS2MT	Mouser	249-85S4MTYPPFBCERE1	9,78	
Y1,	1	32,768KHz	Crystal_GND23	Mouser	815-ABS25-32.768K6-T	0,649	
Y2,	1	8MHz	Crystal	Mouser	449-LFXTAL031818BULK	0,62	
	HC	TTC					
Prix total :	56,76 €		68,112				

## 2<sup>ème</sup> étape : Nettoyage de la carte avec de l'alcool isopropylique



La deuxième étape consiste à nettoyer le stencil afin qu'il n'y ait pas d'impureté dessus pour appliquer la pâte dans les meilleures conditions.

## 3<sup>ème</sup> étape : Association du stencil sur le PCB

Pour superposer le stencil, il faudrait que je bloque ma carte avec les cartes en appliquant du scotch et laisser un écart pour pouvoir reprendre ma carte quand j'en aurai fini avec le stencil.



C'est la carte qui se rapproche le plus de notre projet, voici son stencil sur lequel on va appliquer la pâte à braser.

#### 4<sup>ème</sup> étape : Enduire la pâte à braser



Après avoir bien posé le stencil, on recouvre les trous du stencil qui sont prédéfinis sur le PCB de la carte ce qui va recouvrir parfaitement les places voulues.

#### 5<sup>ème</sup> étape : Placement des composants CMS

#### 6<sup>ème</sup> étape : Vérification

On vérifie ensuite via une loupe le positionnement des composants et les corriger si cela est nécessaire.



### 7<sup>ème</sup> étape : Passage au four à refusions

On place notre carte dans un four qui monte jusqu'à 90°C reste constant à 90°C entre 60 et 120 secondes puis réalise une nouvelle montée en température au-delà d'environ 217°C, température à laquelle la pâte devient liquide entre 45 et 70 secondes ensuite le four redescend en température ainsi la pâte repasse à l'état solide en refroidissant.



## **Nos recherches concernant les composants au début du projet :**

LED PARAMÈTRES:

CMS

15 mcd, 60 mcd, 10m cd

1204 ou 0808

2mA (le plus faible)

X3

QUARTZ PARAMÈTRES :

Fréquence: 8 MHz

12pF ou 16pF

Radial

C33, C34

Condensateurs PARAMÈTRES :

Capacitor,20pF,5%,50V,NPO,0603,YAGEO

R35, R37

Résistors PARAMÈTRES :

0603R

Pour me renseigner sur ces composants j'ai utilisé différents site:

- RS
- Farnell
- Mouser
- Digi-Key

## **Annexe des composants si vous êtes intéressés (ils ont peut être expirés):**

Lien vers mouser

Led cms:

<https://www.mouser.fr/ProductDetail/Kingbright/APFA3010LSEEZGKQBKC?qs=ZqXcJfGIKsh4RNXwluO8Eg%3D%3D> = 1st

<https://www.mouser.fr/ProductDetail/Kingbright/APTF1616LSEEZGKQBKC?qs=ZqXcJfGIKsh%252BpLJNPLW72w%3D%3D> = 2nd

<https://www.mouser.fr/ProductDetail/Inolux/IN-PI20TAT5R5G5B?qs=stqOd1AaK78nDrJHvECLbg%3D%3D> = 3rd)

Quartz:

(<https://www.mouser.fr/ProductDetail/IQD/LFXTAL031818Bulk?qs=e4%2FAndAAwgILUlepGv4RzA%3D%3D> = 1st

<https://www.mouser.fr/ProductDetail/IQD/LFXTAL024717Bulk?qs=e4%2FAndAAwgIGz%252Bxp%252BByDXyg%3D%3D> = 2nd

<https://www.mouser.fr/ProductDetail/IQD/LFXTAL057135Bulk?qs=e4%2FAndAAwgIkCKEhU7u90A%3D%3D> = 3rd

<https://www.mouser.fr/ProductDetail/IQD/LFXTAL029665Reel?qs=e4%2FAndAAwgKgMVs0u0B2zA%3D%3D> = 4rd)

Capacitor:

(<https://www.mouser.fr/ProductDetail/YAGEO/AC0603JRNPO9BN200?qs=tS%2FAHvPQ%2F55t1h2OAaKY2w%3D%3D> = 1st)

## **Conclusion :**

Pour conclure sachant que c'est un nouveau projet cette année on a dû sacrifier une exigence du projet qui concerne l'autonomie de 72h avec le microcontrôleur, peut être trouver une solution concernant cela ainsi que l'alimentation des capteurs pour baisser la consommation pour pouvoir garantir un délai assez cohérent 24h ect jusqu'à arriver vers 72h.



# PARTIE IR2 : LAYOUNI Noa

## Introduction

Dans cette partie je vais présenter les activités que j'ai réalisées durant le projet dans sa globalité, je devais m'occuper de la partie IR2 concernant le site web embarqué que contiendra la clé USB qui permettra la visualisation des mesures de la température, de l'humidité et les chocs enregistrés afin de permettre voir les différents problèmes rencontrés lors d'une livraison.

Ci-dessous la partie du cahier des charges qui m'était attribuée.

Étudiant n°2  <b>IR2</b>	<p style="text-align: center;"><b>Site WEB embarqué</b></p> <ul style="list-style-type: none"> <li>• S'approprier le fonctionnement d'une librairie Javascript capable de tracer des courbes (Highcharts.js, D3.js ...)</li> <li>• Trouver une solution pour pouvoir lire/écrire un fichier sur un système de fichiers local depuis une page web <b>servie sans serveur web</b> (→ <b>file://</b>) <i>[piste à explorer : File System Access API]</i></li> <li>• Coder sur PC un dashboard web « statique » permettant de visualiser les mesures et paramétrer l'enregistreur de données</li> <li>• Déployer le site web dans le système de fichier FatFS du STM32</li> <li>• Assurer la gestion de version logicielle (forge logicielle Framagit)</li> <li>• Rédiger un manuel de démarrage rapide pour l'installation et l'utilisation des</li> </ul>	<p><b>Installation/Configuration :</b></p> <ul style="list-style-type: none"> <li>• IDE pour développement Web</li> </ul> <p><b>Mise en œuvre :</b></p> <ul style="list-style-type: none"> <li>• HTML/CSS</li> <li>• Programmation Javascript</li> </ul> <p><b>Réalisation :</b></p> <ul style="list-style-type: none"> <li>• Site WEB de visualisation des mesures et paramétrage de l'enregistreur de données</li> </ul> <p><b>Documentation :</b></p> <ul style="list-style-type: none"> <li>• Guide de démarrage rapide</li> <li>• Gestion de version logicielle</li> <li>• Dossier de développement</li> </ul>
--------------------------------	---	---

## Objectif personnel

Mon objectif est de créer un site web en local qui permettra la visualisation des données enregistrées par la partie de l'élève IR2, ce site web mettra en avant 3 courbes évolutives, qui indiqueront :

- La température
- L'humidité relative
- Les chocs

Pour les courbes et leur aspect graphiques j'ai utilisé la librairie D3.js qui permet de rendre les courbes plus esthétiques et de mieux se repérer sur le site.

J'ai dans un premier temps rapidement utiliser ChatGPT pour pouvoir me donner une idée de ce à quoi pourrait ressembler cette courbe et l'allure qu'elle allait avoir.

*J'ai donc émis à l'intelligence artificielle les instructions suivantes :*

« Pourrais-tu me créer une page web en local sur laquelle il serait capable d'enregistrer un fichier en local sur le PC, le fichier enregistrer contiendra soit la date actuelle soit une date choisie à l'aide d'un calendrier. »

*Voici le code exemple de ChatGPT :*

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Températures du 31/03/2023</title>
  <script src="https://d3js.org/d3.v7.min.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 1;
      padding: 0;
    }

    .line {
      fill: none;
      stroke: #dec912;
      stroke-width: 2;
    }
  </style>
</head>
</html>
```

```
.axis path,
.axis line {
  fill: none;
  stroke: #ad1e1e;
  shape-rendering: crispEdges;
}

.tick-label {
  font-size: 12px;
  font-weight: bold;
}

#y-axis-label {
  font-size: 14px;
  font-weight: bold;
  transform: rotate(-90deg);
  transform-origin: left top;
  text-anchor: middle;
  margin-left: -40px;
  margin-top: 70px;
}
</style>
</head>
<body>
<svg width="500" height="500"></svg>

<script>
  // Les données des températures
  const temperatures = [
    { heure: "00h", temperature: 6 },
    { heure: "01h", temperature: 5 },
    { heure: "02h", temperature: 5 },
    { heure: "03h", temperature: 4 },
    { heure: "04h", temperature: 3 },
    { heure: "05h", temperature: 3 },
    { heure: "06h", temperature: 3 },
    { heure: "07h", temperature: 4 },
    { heure: "08h", temperature: 5 },
    { heure: "09h", temperature: 7 },
    { heure: "10h", temperature: 9 },
    { heure: "11h", temperature: 12 },
    { heure: "12h", temperature: 15 },
    { heure: "13h", temperature: 18 },
    { heure: "14h", temperature: 20 },
    { heure: "15h", temperature: 22 },
    { heure: "16h", temperature: 23 },
    { heure: "17h", temperature: 22 },
    { heure: "18h", temperature: 20 },
```

```

    { heure: "19h", temperature: 17 },
    { heure: "20h", temperature: 15 },
    { heure: "21h", temperature: 12 },
    { heure: "22h", temperature: 10 },
    { heure: "23h", temperature: 8 }
  ];

  // Les dimensions du graphique
  const margin = { top: 100, right: -10, bottom: 20, left: 150 };
  const width = 1000 - margin.left - margin.right;
  const height = 500 - margin.top - margin.bottom;

  // Crée les échelles pour l'axe X et l'axe Y
  const xScale = d3.scaleBand()
    .domain(temperatures.map(d => d.heure))
    .range([0, width])
    .padding(0.1);

  const yScale = d3.scaleLinear()
    .domain([0, d3.max(temperatures, d => d.temperature)])
    .range([height, 0]);

  // Crée la fonction pour dessiner la courbe
  const line = d3.line()
    .x(d => xScale(d.heure) + xScale.bandwidth() / 2)
    .y(d => yScale(d.temperature));

  // Crée l'élément SVG qui contiendra le graphique
  const svg = d3.select("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", `translate(${margin.left}, ${margin.top})`);

  // Ajoute l'axe X
  svg.append("g")
    .attr("class", "x-axis")
    .attr("transform", `translate(0, ${height})`)
    .call(d3.axisBottom(xScale))
    .selectAll(".tick text")
    .attr("class", "tick-label");

  // Ajoute le titre de l'axe X
  svg.append("text")
    .attr("id", "x-axis-label")
    .text("Température (°C)");

  // Ajoute l'axe Y

```

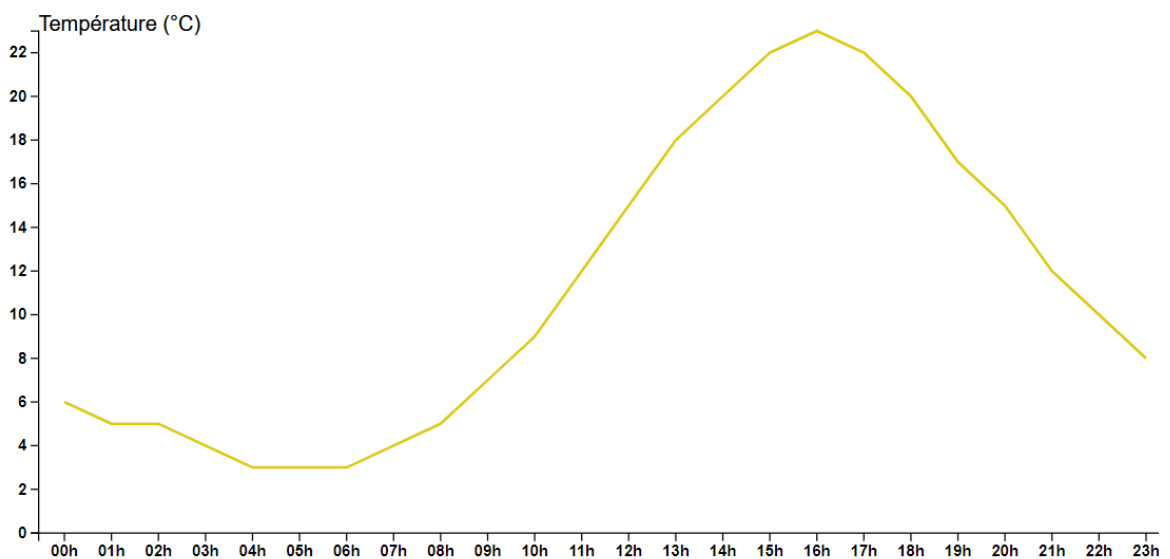
```
svg.append("g")
  .attr("class", "y-axis")
  .call(d3.axisLeft(yScale))
  .selectAll(".tick text")
  .attr("class", "tick-label");

// Ajoute la courbe au graphique
svg.append("path")
  .datum(temperatures)
  .attr("class", "line")
  .attr("d", line);

// Ajoute le titre de l'axe Y
svg.append("text")
  .attr("id", "x-axis-label")
  .text("Température (°C)");

</script>
</body>
</html>
```

## Résultat :



Ce n'était pas le résultat escompté mais je me suis tout de même inspiré d'une partie du code pour la création des axes qui est donc cette part

```
// Crée les échelles pour l'axe X et l'axe Y
const xScale = d3.scaleBand()
  .domain(temperatures.map(d => d.heure))
  .range([0, width])
  .padding(0.1);

const yScale = d3.scaleLinear()
  .domain([0, d3.max(temperatures, d => d.temperature)])
  .range([height, 0]);
```

## Page web

Pour la création de la page web, il y a un ensemble de questions à se poser. Notamment au niveau des contraintes que nous devons respecter :

### Contraintes des chocs :

Savoir si l'on pourra acquérir la valeur de l'amplitude des chocs.  
Lire les données sur une courbe "lollipop".

Détection du choc s'il dépasse un certain seuil que l'on essaiera de pouvoir paramétrer sur la page web.

Au niveau de l'horodatage, la période d'enregistrement des données de chocs débutera lorsque le seuil sera dépassé pour la première fois, l'acquisition des mesures suivantes indiqueront combien de secondes après le début celles-ci ont été enregistrées.

Pour ce qui est des mesures, l'idéal serait de pouvoir indiquer les valeurs des trois axes X, Y et Z, du moins celui ou ceux qui ont connu un choc dépassant le seuil et les afficher seulement eux sur le graphique.

Possibilité de changer de type d'affichage pour les chocs et de ne pas faire une courbe mais plutôt un tableau si nous n'arrivons pas à recueillir les valeurs des amplitudes des chocs.

### Contraintes de la température et de l'humidité :

On utiliserait donc D3.js pour la courbe qui contiendra les valeurs des mesures de température et d'humidité.

En tout, il y aura 30 000 mesures à effectuer sur 72h ce qui est équivalent à une mesure toute les 8s.

Créer une mini-map de la courbe avec D3.js tout comme avec Highcharts.js serait intéressant car nous ne pourrions pas afficher toutes les mesures à l'écran. Et donc, grâce à cela nous allons pouvoir choisir la période de mesures que nous voulons visualiser. La fonction "brush" nous permet de réaliser cette mini-map sur D3.js.

### Contraintes de la date :

Pour que l'horloge intégrée qui est présente dans la clé USB soit bien réglée à l'heure, il faudra pour ma part créer un fichier txt que l'on appellera par exemple "currdate.tkt" que l'on intégrera dans la clé USB.

On inclura alors un f\_stat, ensuite nous lirons le fichier et nous programmerons l'horloge interne.

Sur la page web, on ajoutera des options de paramétrage, ainsi nous pourrions ensuite sauvegarder, effacer les mesures, modifier la fréquence d'acquisition et le seuil de choc et aussi la date.

## Choix de la librairie

Pour savoir quelle librairie JavaScript nous allons utiliser, j'ai dû comparer celles qui m'étaient données en exemple, Highcharts.js et D3.js.

### Comparatif des librairies JS

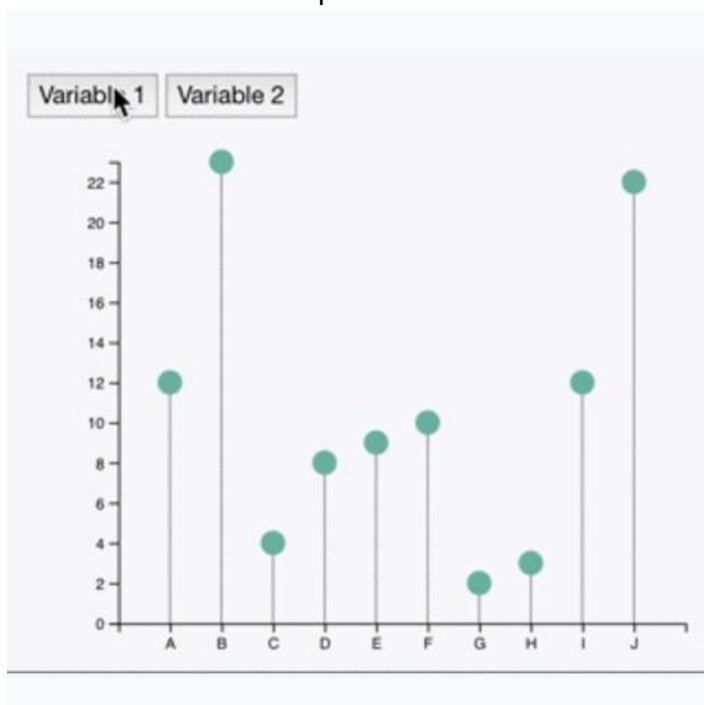
Highcharts.js :

- Différents types de sites possibles grâce à cette librairie comme des maps, des données de stockage ou encore des diagrammes de Gantt.
- Utilisé par 80 des 100 plus grandes entreprises comme Samsung, Sony, BBC, American express, twitter ou encore MasterCard.
- Large choix de personnalisation au niveau du code et de la mise en forme des graphiques.



D3.js :

- Esthétique simpliste.
- Animation stylisée et améliorée.
- Grande palette de couleurs pour les graphiques.
- Création possible de graphiques en colonnes ou camembert.
- Création possible de courbe Lollipop ex :



Mon choix s'est donc tourné vers la librairie D3.js car pour la courbe qui regroupe les données des chocs cela sera plus simple et esthétique avec la courbe Lollipop.

De même pour la température et l'humidité car nous pourrons créer une mini-map pour pouvoir se balader sur la courbe pour étudier particulièrement les données que nous souhaitons.

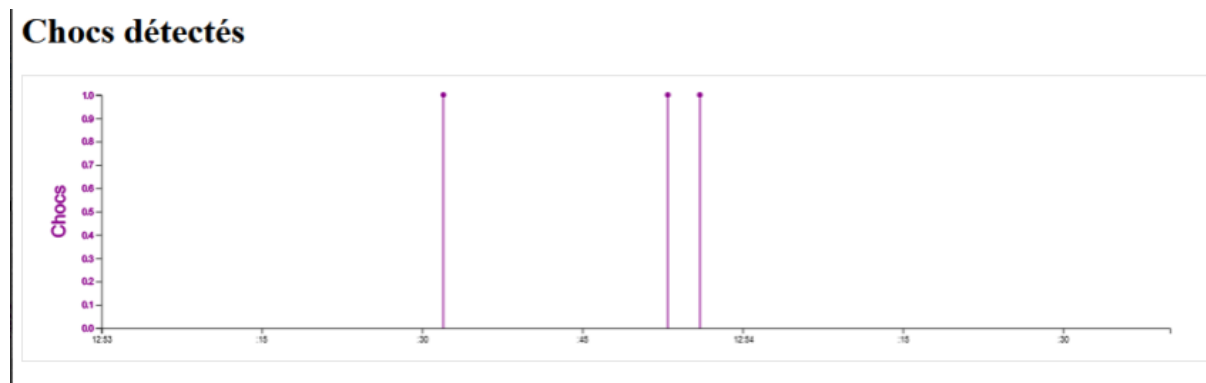


## Étude du code des courbes

La courbe Lollipop est choisie car elle est plus esthétique et simple à comprendre au premier abord.

Ce qui permettra de peut-être pouvoir afficher sur le graphique l'amplitude du choc enregistré.

Un exemple :



La fonction `data()` nous permet de travailler sur le fichier de données indiqué et la fonction `enter()` de sélectionner ce fichier de données.

```

svg.append('g')
  .selectAll("dot")
  .data(data)
  .enter()
  .append("circle")
  .attr("cx", d => { return xTime(d.ts); })
  .attr("cy", d => { return yTemperature(d.T); })
  .attr("r", 3)
  .attr("class", "dotT")
;
    
```

On utilise les données du fichier `G-LOG.js` à l'aide de plusieurs timestamps qui indiquent la date à laquelle un choc a été détecté.

```
1  var chocs = {  
2    'startDate': "2023-03-22T11:53:00.000Z",  
3    'data': `_dt  
4    1679486012  
5    1679486033  
6    1679486036`  
7  };|
```

3 chocs relevés et utilisés en timestamp pour le répertoire chronologiquement sur la courbe.

```
const xTime = d3.scaleTime()  
  .range([0, width]);  
  
const yShocks = d3.scaleLinear()  
  .range([height, 0]);
```

Ici on cale les deux graphiques à la même échelle pour qu'ils représentent la même durée.

## Site Web

Dans ce projet, nous devons créer une page web contenant des graphiques pour afficher les différentes données que nous aurons acquises grâce à la clé USB.

Or, dans le cahier des charges il nous est imposé d'utiliser une page web SANS serveur Web.

Pour essayer de comprendre pourquoi il nous été imposé ceci, j'ai donc dressé deux tableaux contenant les avantages et inconvénients des deux différentes possibilités.

### Site Web avec serveur

Avantages	Inconvénients
<p>Permet de mieux sécuriser les données de son site</p> <p>Améliore les performances du site.</p> <p>Serveur entièrement dédié au site.</p> <p>On peut adapter la mémoire selon la performance requise par le site</p> <p>Plus grande réception de données.</p> <p>Le client dispose de tous les droits sur le serveur.</p>	<p>Partager le même serveur pour plusieurs sites peut affecter les performances</p> <p>Ce type d'hébergement n'est pas adapté pour les sites avec un fort trafic.</p> <p>Il est impossible pour des utilisateurs avancés de personnaliser la configuration de leur serveur web.</p> <p>Non accessible sans connexion internet.</p>

### Site Web sans serveur

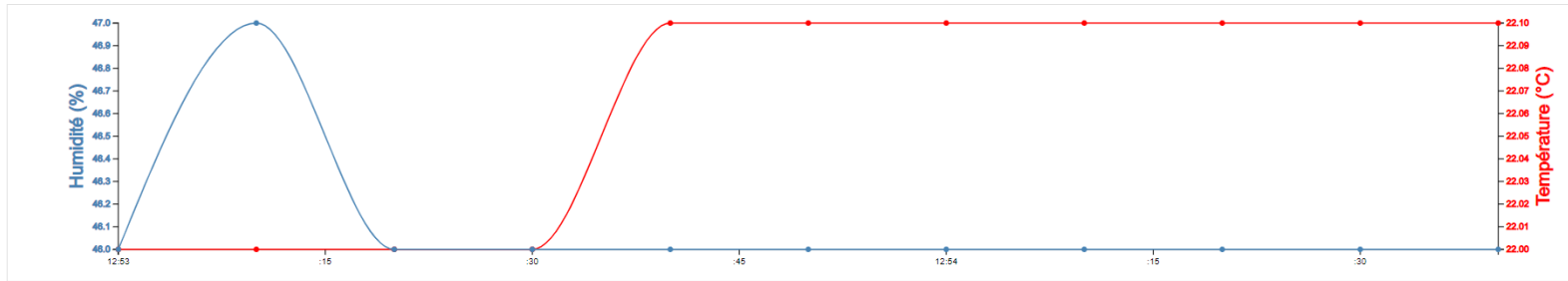
Avantages	Inconvénients
<p>Aucune gestion de serveur nécessaire.</p> <p>L'évolution des données est à la guise du développeur car il gère les sources qu'il veut recevoir.</p> <p>Déploiements et mises à jour rapides.</p> <p>Accessible sans connexion internet.</p>	<p>L'architecture d'un site sans serveur web n'est pas prévue pour de la longue durée.</p>

## **Interface Web**

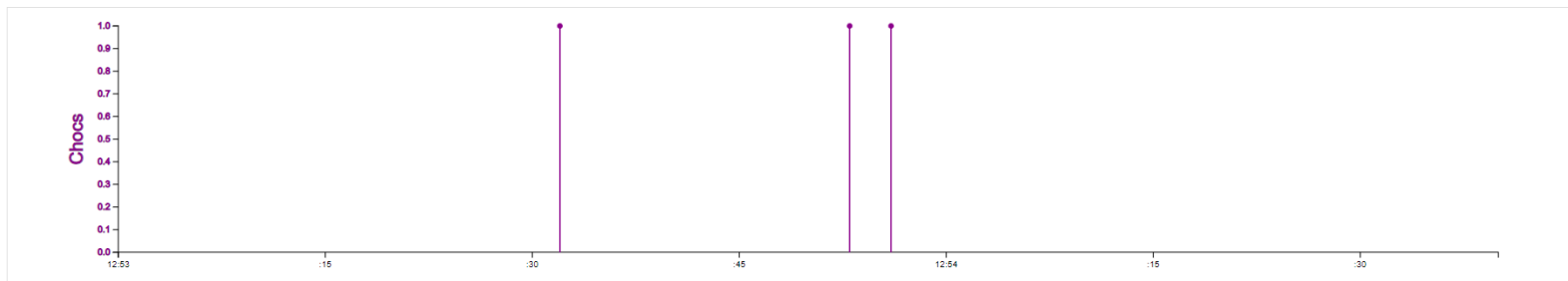
Penchons-nous maintenant sur la partie affichage, graphique et paramétrage du site web.

Pour la partie affichage, les deux courbes seront superposées, la courbe d'humidité et de température au-dessus de celle des chocs, voici une visualisation de cet affichage :

## Evolution de la T° (en °C) et de l'humidité relative (en %)



## Chocs détectés



L'objectif final en ce qui concerne l'affichage est d'intégrer une mini-map comme il est expliqué précédemment, pour pouvoir se balader chronologiquement sur la période de temps que l'on souhaite pour visualiser l'évolution de la courbe.

## Planification prévisionnelle

▲ LAYOUNI	61 h	Mar 03/01/23	Mer 15/02/23		
S'approprier le fonctionnement d'une librairie Javascript capable de tracer des courbes (Highcharts.js, DS3.js...)	25 h	Mar 03/01/23	Lun 16/01/23		Noa Layouni
Trouver une solution pour pouvoir lire/écrire un fichier sur un système de fichiers local depuis une page web servie sans serveur web (-> file://) (piste à explorer : File System Access API)	20 h	Lun 16/01/23	Mer 25/01/23	15	Noa Layouni
Coder sur PC un dashboard web "statique" permettant de visualiser les mesures et paramétrer l'enregistreur de données	10 h	Mer 25/01/23	Mar 31/01/23	16	Noa Layouni
Déployer le site web dans le système de fichier FatFS du STM32	10 h	Mar 31/01/23	Lun 06/02/23	17	Noa Layouni
Assurer la gestion de version logicielle (forge logicielle Framagit)	10 h	Mar 07/02/23	Lun 13/02/23	18	Noa Layouni
Rédiger un manuel de démarrage rapide pour l'installation et l'utilisation des différents constituants	10 h	Lun 13/02/23	Mer 15/02/23	19	Noa Layouni
Collaborer étroitement avec étudiant IR1	1 h	Mer 15/02/23	Mer 15/02/23	20	Noa Layouni