



## SG05 – Makair

*Barreteau Paul*

### Dossier technique du projet – partie individuelle

<b>I.</b>	<b>PRESENTATION DU PROJET .....</b>	<b>2</b>
II.1	SA CREATION : .....	2
II.2	BUT DU PROJET : .....	3
II.3	INTRODUCTION .....	6
<b>II.</b>	<b>CONTROLLER LE DEBIT D'AIR .....</b>	<b>8</b>
II.4	CAHIER DES CHARGES : .....	8
II.5	ANALYSE : .....	9
II.6	REALISATION : .....	11
II.1.1	Intégration.....	11
II.1.2	Choix des résistances de tirages.....	12
II.1.3	Code .....	12
II.7	TEST .....	13
II.1.4	Problèmes rencontrés.....	13
<b>III.</b>	<b>COMMANDER LA TURBINE : .....</b>	<b>15</b>
II.8	CAHIER DES CHARGES : .....	15
II.9	ANALYSE : .....	16
II.11	REALISATION : .....	17
II.12	TEST .....	18
III.1.1	Problèmes rencontrés.....	18
<b>IV.</b>	<b>BILAN DE LA REALISATION PERSONNELLE – CONCLUSION.....</b>	<b>19</b>

# **I. Présentation du projet**

## **II.1 Sa création :**



En 2020 pour contrer le covid 19 et la forte affluence en réanimation Mr GOURRAUD Pierre Antoine à développer un respirateur artificiel à bas cout pour faciliter l'accès aux hôpitaux à des assistant respiratoire. Le Makair est un projet open source ce qui laisse le libre accès aux informations pour le fabriquer sois même de la partie électrique/électronique à la programmation.

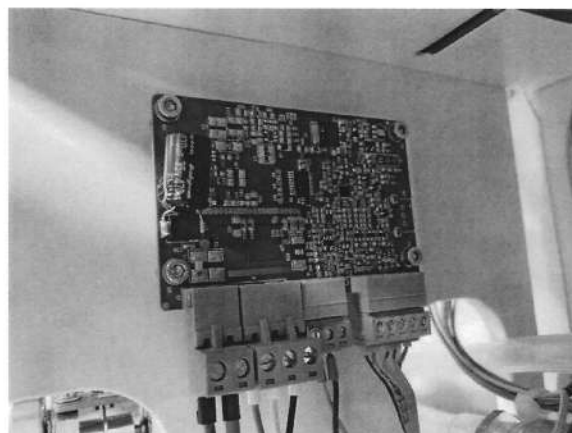
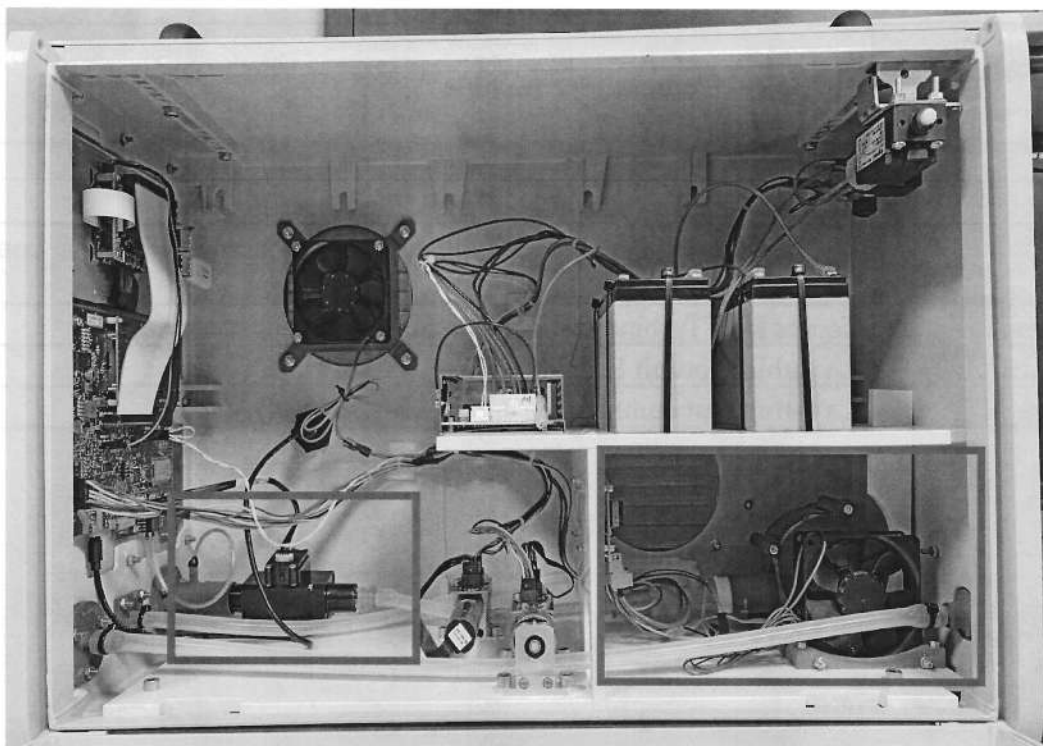
## II.2 But du projet :

Le système envisagé consiste à améliorer sur certains points, décrits ci-dessous, la version V1 qui a été prêtée à l'établissement.

Le système doit pouvoir :

- Superviser la batterie de façon à indiquer son niveau de charge
- Assurer la régulation du débit d'air fourni au patient
- Gérer la température de l'intérieur du boîtier

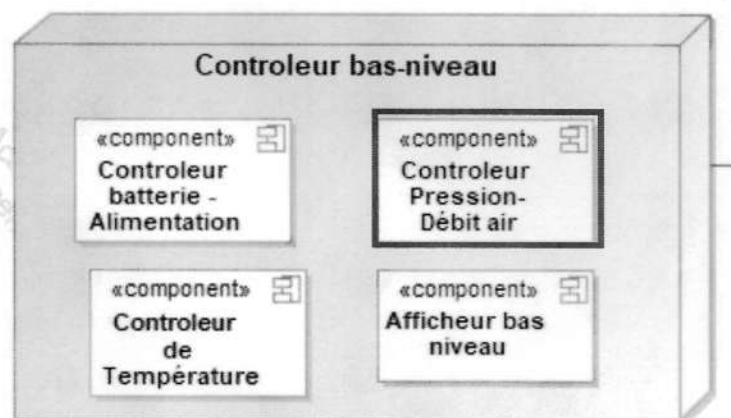
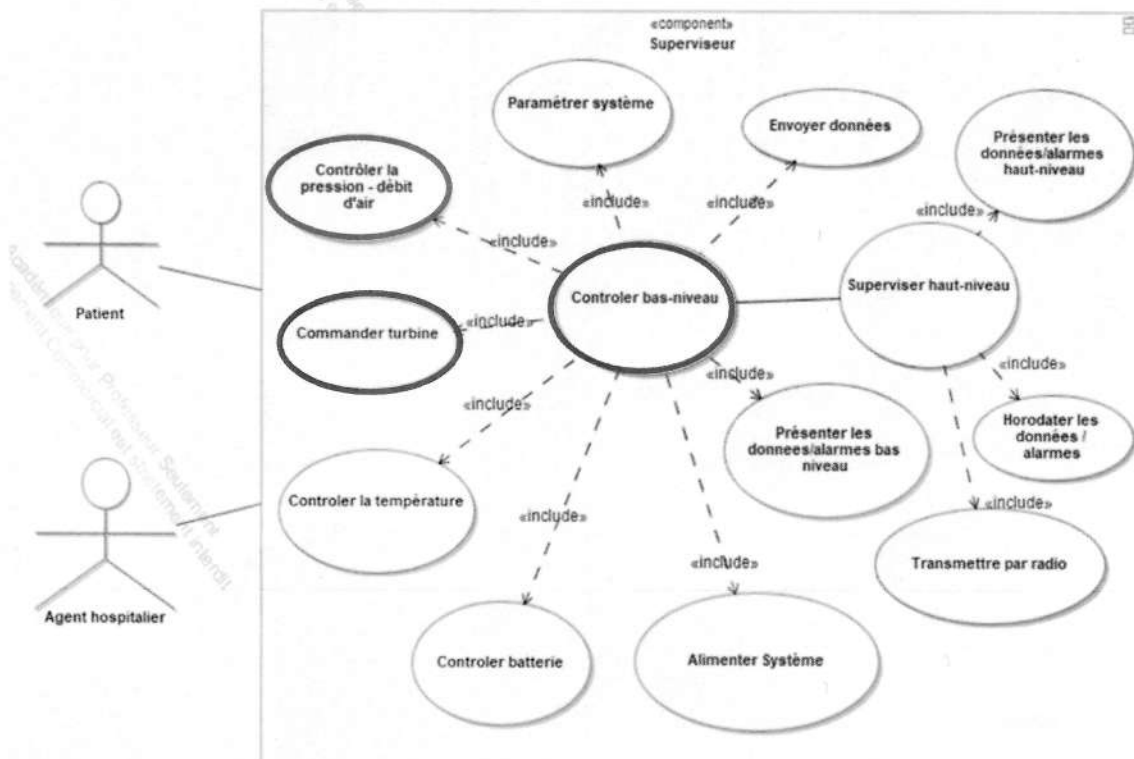
Ces informations doivent pouvoir être affichées « en local » sur un écran embarqué, et sur un écran déporté plus grand et offrant plus d'ergonomie de lecture.



<b>Nom du cas d'utilisation</b>	<b><u>UCBN3</u> – Contrôler la pression-débit d'air</b>
<b><u>Pré-condition(s)</u></b>	Le MakAir est allumé et configuré. Il est opérationnel.
<b>Scénario nominal</b>	De manière automatique et complètement autonome, le MakAir surveille la pression et le débit d'air
<b>Séquencement</b>	Toutes les <u>TAir</u> secondes, le débit d'air est contrôlé (unités cmH2O)
<b>Post-condition</b>	Les données sont correctement collectées et envoyées au superviseur haut niveau
<b>Exigences</b>	Les données sont envoyées au même rythme que la collecte, à la période <u>TAir</u> , (unités cmH2O)

<b>Nom du cas d'utilisation</b>	<b><u>UCBN4</u> – Commander turbine</b>
<b><u>Pré-condition(s)</u></b>	Le MakAir est allumé et configuré. Il est opérationnel.
<b>Scénario nominal</b>	De manière automatique et complètement autonome, le MakAir commande la turbine
<b>Séquencement</b>	Toutes les <u>TTurbine</u> secondes, la turbine est commandée
<b>Post-condition</b>	La turbine fournit le débit d'air
<b>Exigences</b>	La turbine est commandée en PWM – 24 volts

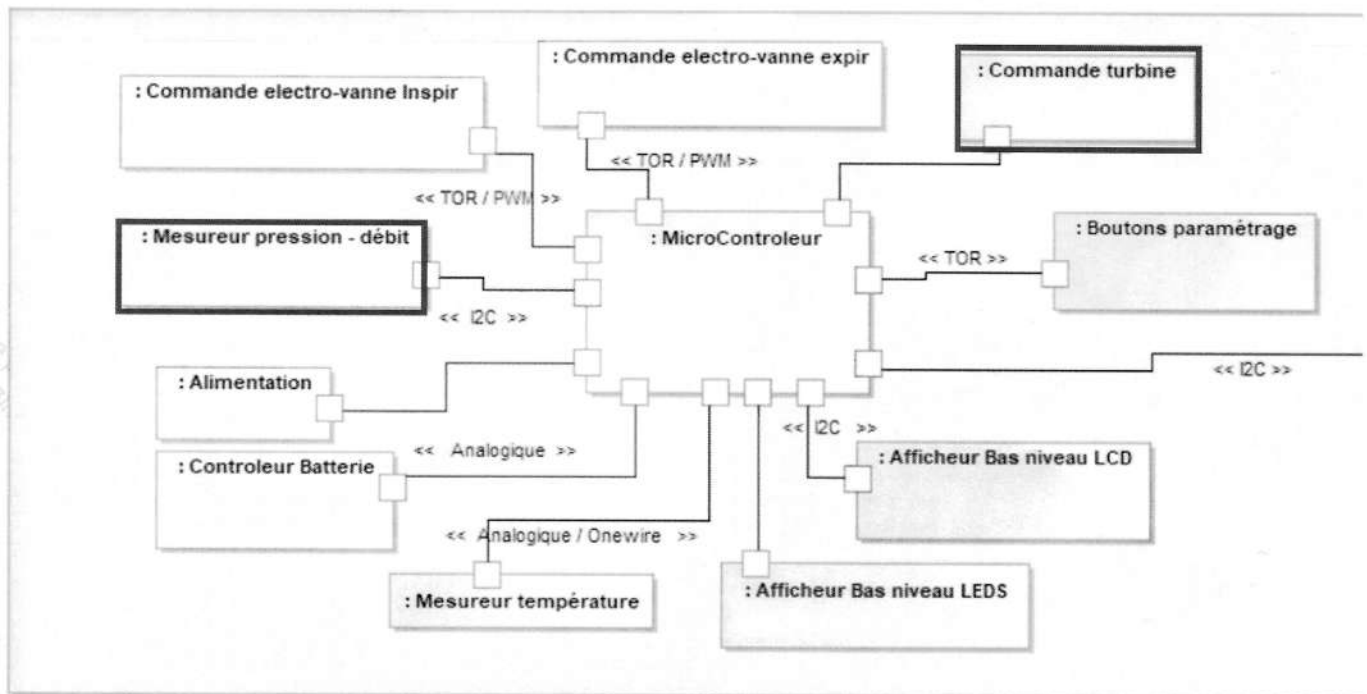
## 11.5.1 - Diagramme des cas d'utilisations

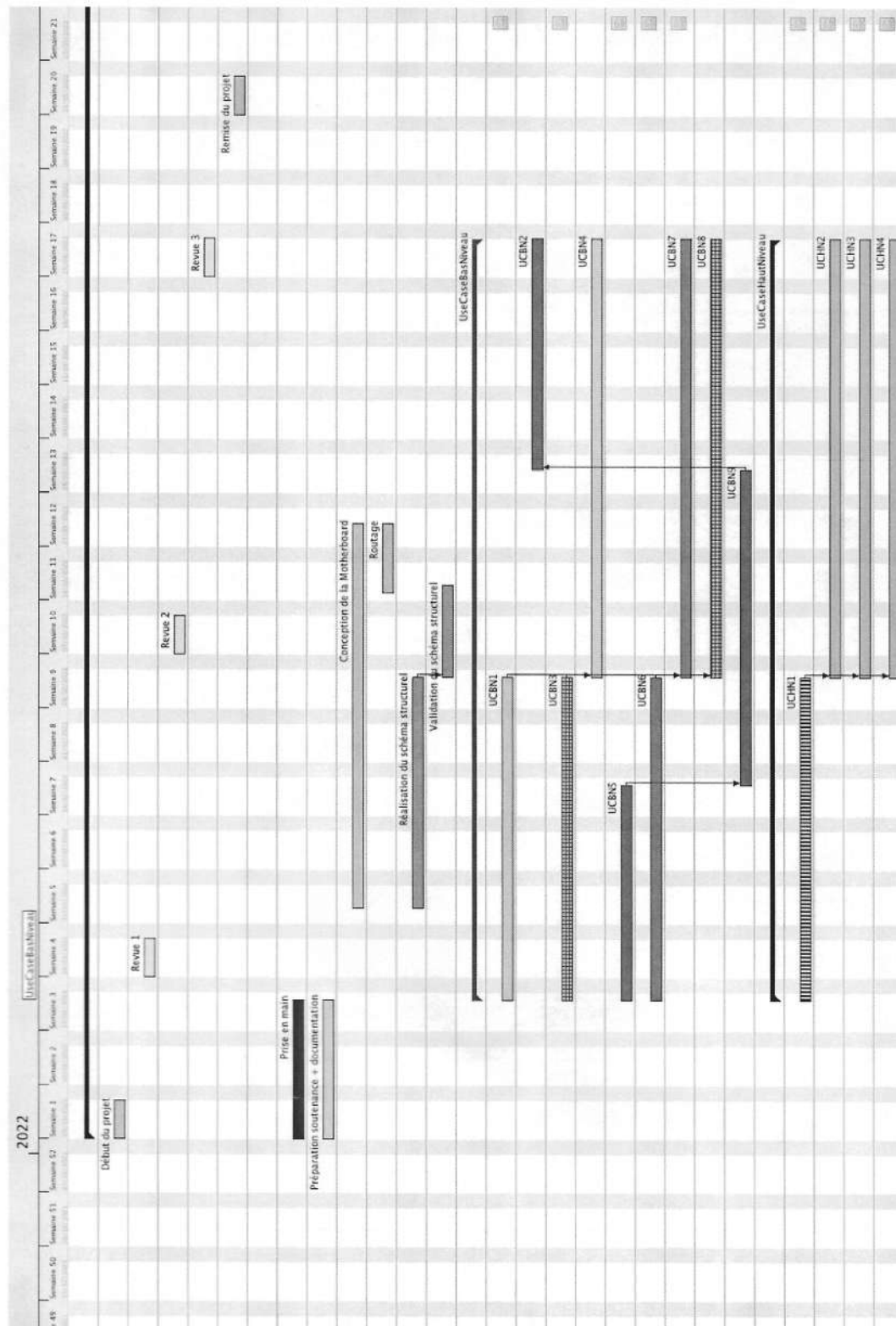


### II.3 Introduction

*Durant ce projet j'ai dû réaliser les tâches suivantes :*

- *Prise en main Cahier des charges.*
- *Découverte de la structure du Makair et des documents techniques.*
- *Séparation des différentes parties du Makair.*
- *Découverte du débitmètre avec codage et test.*
- *Découverte de la carte de puissance et de la turbine avec codage et test.*
- *Réalisation de la nomenclature*

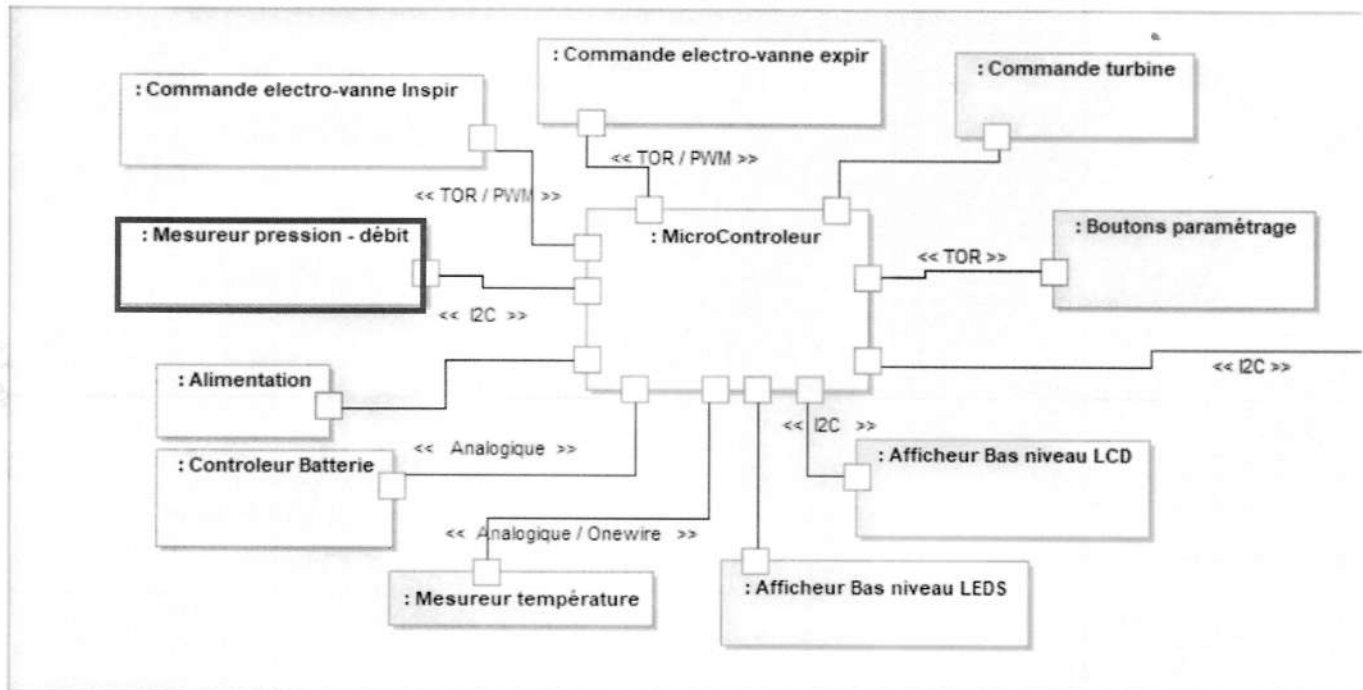




## II. Contrôler le débit d'air

### II.4 Cahier des charges :

Objectif : Mesurer le débit d'air produit par la turbine en sortie du Makair avec un délai entre chaque mesure



<b>Nom du cas d'utilisation</b>	<u>UCBN3</u> – Contrôler la pression-débit d'air
<b>Pré-condition(s)</b>	Le MakAir est allumé et configuré. Il est opérationnel.
<b>Scénario nominal</b>	De manière automatique et complètement autonome, le MakAir surveille la pression et le débit d'air
<b>Séquencement</b>	Toutes les TAir secondes, le débit d'air est contrôlé (unités cmH2O)
<b>Post-condition</b>	Les données sont correctement collectées et envoyées au superviseur haut niveau
<b>Exigences</b>	Les données sont envoyées au même rythme que la collecte, à la période TAir, (unités cmH2O)



## II.5 Analyse :

Pour pouvoir répondre au cahier des charges j'ai dû commencer par choisir un débitmètre pour ensuite réaliser une comparaison entre celui de présent dans le makair et un nouveau.



<i>Sensirion</i>	<i>Honeywell Zephyr</i>
<i>Semblable à l'existant</i>	<i>Présent dans le Makair</i>
<i>5Vdc</i>	<i>3,3Vdc à 10Vdc</i>
<i>Plage de débit 240 SLPM</i>	<i>Plage de débit 200 SLPM</i>
<i>Sortie numérique I2C</i>	<i>Sortie numérique I2C</i>
<i>160€</i>	<i>Prix 150€</i>
<i>16 bits de résolution</i>	<i>12 bits de résolution</i>

*Après avoir comparé ces deux débitmètres j'ai dû choisir le plus efficace et simple d'utilisation.*

*J'ai donc choisi le débitmètre déjà présent dans le makair il est fonctionnel avec tout le système sur le plan électronique (compatibilité avec les autres composants, branchement...) comme sur le plan mécanique (diamètre des tuyaux). De plus le fait qu'il soit déjà présent dans le makair m'a permis de le tester dès le début du projet sans à avoir à attendre les délais de commande et de livraison.*

*Ce débitmètre se commande en I2C ce qui signifie que :*

Le bus I<sup>2</sup>C est un bus série synchrone bidirectionnel half-duplex, où plusieurs équipements, un maître et un ou des esclaves, peuvent être connectés au bus.

Les échanges ont toujours lieu entre un seul maître (Nucléo) et un (des) esclave (Débitmètre), toujours à l'initiative du maître (jamais de maître à maître ou d'esclave à esclave). Cependant, rien n'empêche un composant de passer du statut de maître à esclave et réciproquement.

La connexion est réalisée par l'intermédiaire de deux lignes :

- SDA (Serial Data Line) : ligne de données bidirectionnelle,
- SCL (Serial Clock Line) : ligne d'horloge de synchronisation bidirectionnelle.

Il ne faut également pas oublier la masse qui doit être commune aux équipements.

Les 2 lignes sont tirées au niveau de tension VDD à travers des résistances de pull-up (RP).

Ce débitmètre a une précision de 12 bits ce qui signifie qu'il a un quantum de :

$$Q = \frac{V_{dd}}{2^{nb}}$$

$$Q = \frac{5}{2^{12}} = 1,22 \cdot 10^{-3} V = 1,22 mV$$

L'une des autres raisons pour laquelle ce débitmètre a été choisi est qu'il a une documentation détaillée.

### Honeywell Zephyr™ Digital Airflow Sensors

HAF Series—High Accuracy: 10, 15, 20, 50, 100, 200, 300 SLPM

Figure 8. Mounting Dimensions (For reference only: mm [in], continued.)

Port Style: G 3/8 Female Threaded Fitting per ISO 1179

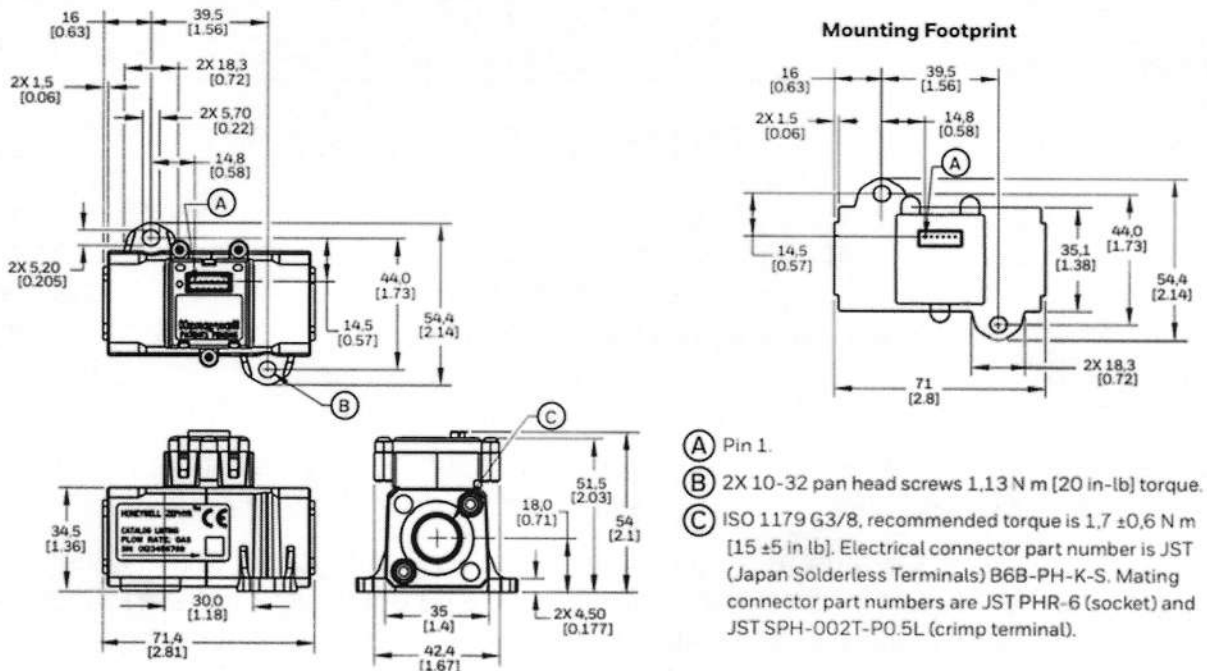


Table 7. Pinout (Digital Function)

Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6
NC	SCL	VVDD	ground	SDA	NC

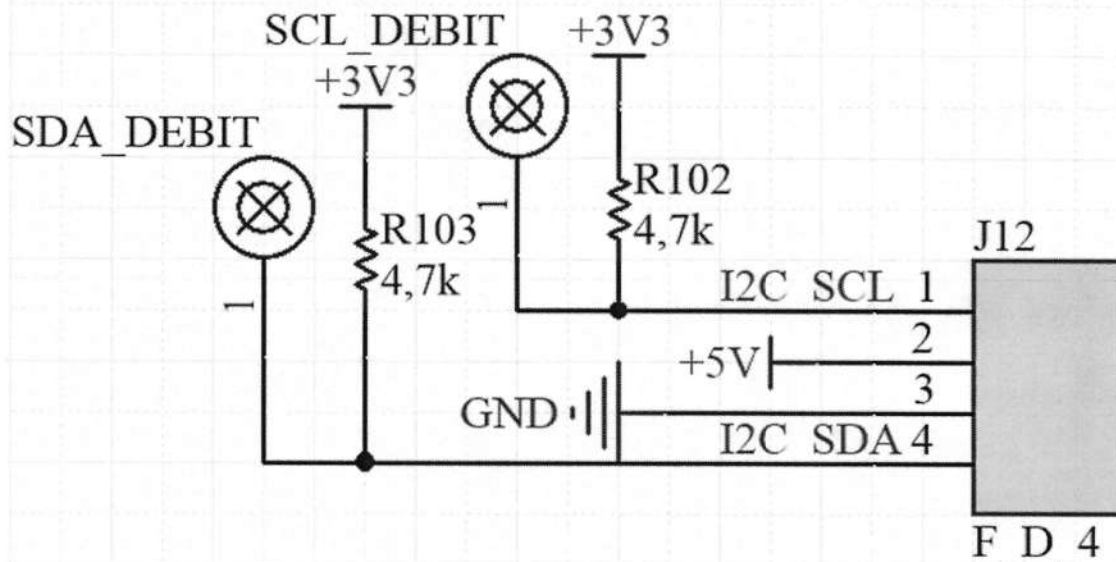
Datasheet du débitmètre avec les dimensions et le brochage.

II.6 Réalisation :

## II.1.1 Intégration

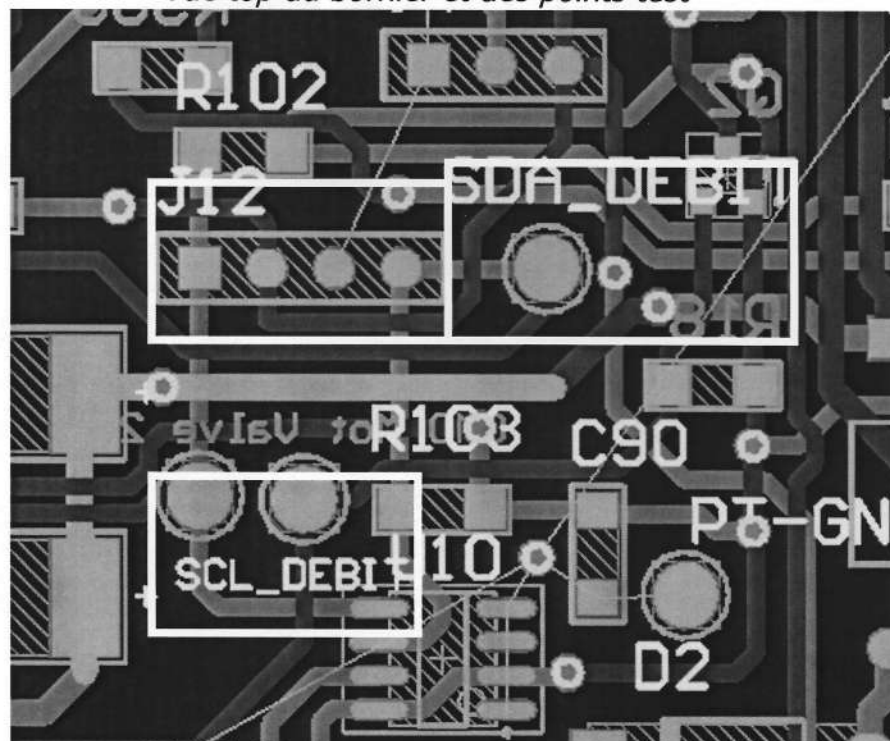
Pour intégrer le débitmètre avec la carte j'ai dû installer un bornier 4 broches pour le SDA, SCL, +5V et la masse. Les broche SDA et SCL ont des points de test pour pouvoir tester la communication et afficher le signal en cas de problème. J'ai aussi dû installer des résistances de tirages (pull-up) pour forcer les états logiques de sortie à 1.

## Extensions I2C Debit



*Schéma structurel de l'extension I2c*

*Vue top du bornier et des points test*



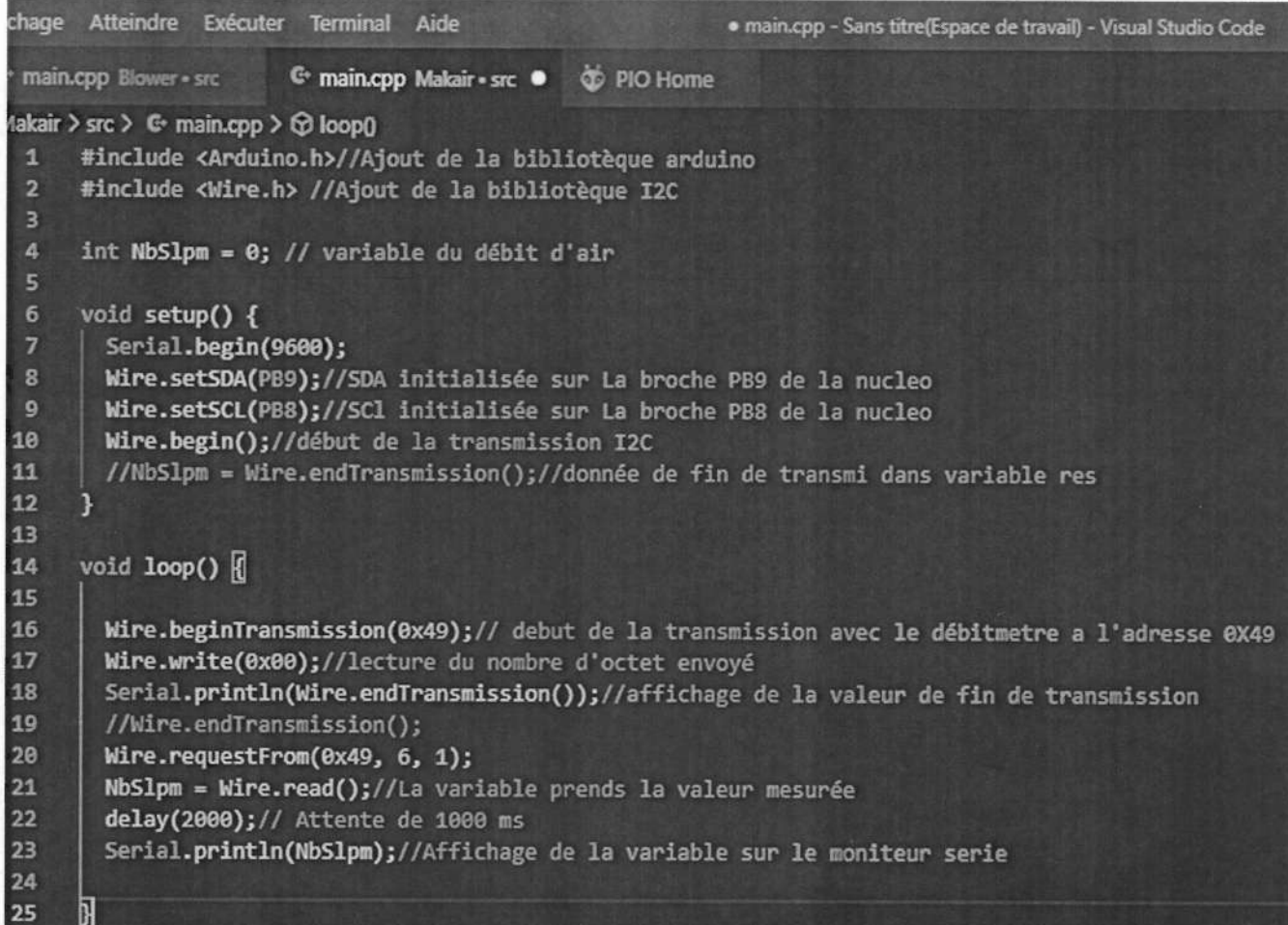
### II.1.2 Choix des résistances de tirages

*Pour choisir les résistances de tirage je me suis référé aux informations données par le constructeur. J'ai donc choisi des résistances de 4,7kohms*

### II.1.3 Code

*J'ai ensuite dû réaliser le code du débitmètre :*

*Le code a pour but de tester le débitmètre avec ses broches et pour afficher la valeur mesurer en SLPM (Standard litre per minutes).*



```
chage Atteindre Exécuter Terminal Aide • main.cpp - Sans titre(Espace de travail) - Visual Studio Code
main.cpp Blower • src main.cpp Makair • src PIO Home
Makair > src > main.cpp > loop()
1 #include <Arduino.h> //Ajout de la bibliothèque arduino
2 #include <Wire.h> //Ajout de la bibliothèque I2C
3
4 int NbSlpm = 0; // variable du débit d'air
5
6 void setup() {
7   Serial.begin(9600);
8   Wire.setSDA(PB9); //SDA initialisée sur La broche PB9 de la nucleo
9   Wire.setSCL(PB8); //SCL initialisée sur La broche PB8 de la nucleo
10  Wire.begin(); //début de la transmission I2C
11  //NbSlpm = Wire.endTransmission(); //donnée de fin de transmi dans variable res
12 }
13
14 void loop() {
15
16   Wire.beginTransmission(0x49); // debut de la transmission avec le débitmetre a l'adresse 0X49
17   Wire.write(0x00); //lecture du nombre d'octet envoyé
18   Serial.println(Wire.endTransmission()); //affichage de la valeur de fin de transmission
19   //Wire.endTransmission();
20   Wire.requestFrom(0x49, 6, 1);
21   NbSlpm = Wire.read(); //La variable prends la valeur mesurée
22   delay(2000); // Attente de 1000 ms
23   Serial.println(NbSlpm); //Affichage de la variable sur le moniteur serie
24
25 }
```

*Code du débitmètre*

II.7 Test

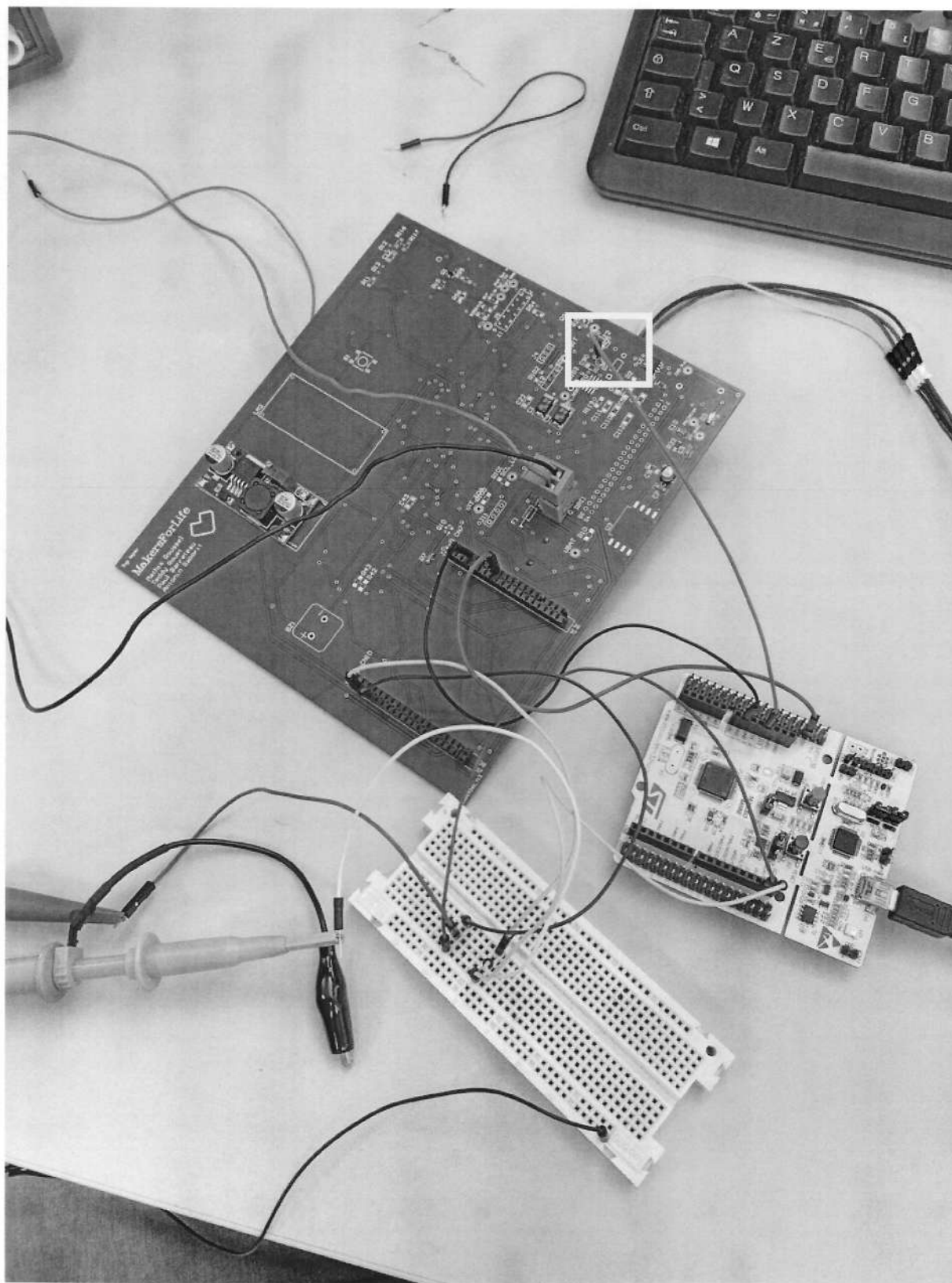
Élément testé :	Débitmètre		
Objectif du test :	Vérifier son intégration dans la carte Vérifier son fonctionnement		
Nom du testeur :	Barreteau Paul	Date : 10/05/2022	
Moyens mis en œuvre :	Logiciel : Visual Studio code	Matériel : Débitmètre	Outil de développement : Visual Studio code
Réalisation :	<p>Procédure du test : Au début du test j'ai commencé par tester le débitmètre sur une carte nucléo seule afin de tester de son fonctionnement.</p> <p>Puis après avoir reçu la carte et soudé les composants nécessaires au fonctionnement de la partie I2C j'ai connecté le débitmètre pour ensuite regarder les trames à l'aide d'un oscilloscope et pour vérifier le bon fonctionnement de mon code</p>		
Élément testé	Résultat attendu	Résultat obtenu	Validation (O/N)
Intégration sur la carte	Intégration sans problème	Bonne intégration mais problème d'alimentation du 3,3v	OUI
Fonctionnement du débitmètre	Bon fonctionnement	Fonctionne correctement et réponds aux attentes du code	OUI
Conclusion du test :	Le test m'a permis de tout vérifier. Le fonctionnement du débitmètre avec son code et son intégration sur la carte sont réussis. Il faudrait comme amélioration une alimentation direct du bornier en 3V3		

II.1.4 Problèmes rencontrés

Lors de la réalisation de ce test j'ai pu observer un problème d'alimentation sur le 3v3 du bornier. J'ai donc dû utiliser l'alimentation de la nucléo pour alimenter le bornier I2C



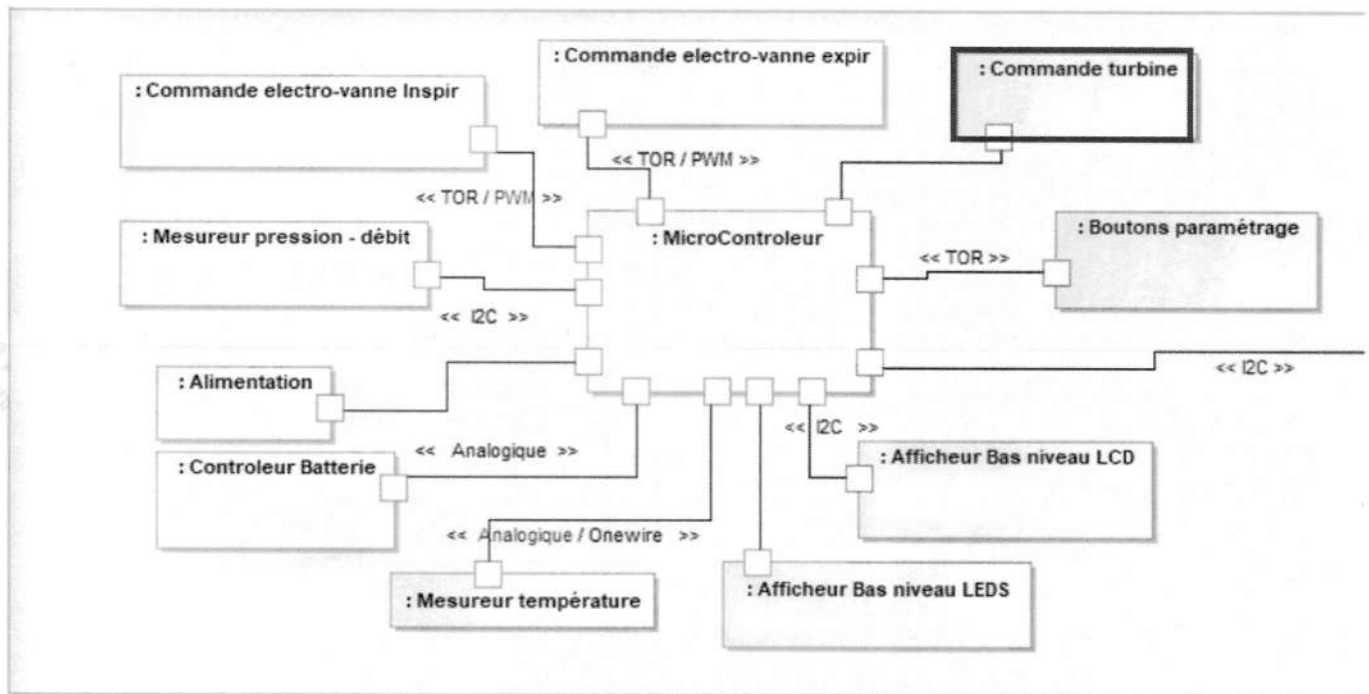
Dans l'encadrer arrivée de l'alimentation externe de la nucléo vers la partie I2C de la carte mère afin de l'alimenter en 3V3



### III. Commander la turbine :

#### II.8 Cahier des charges :

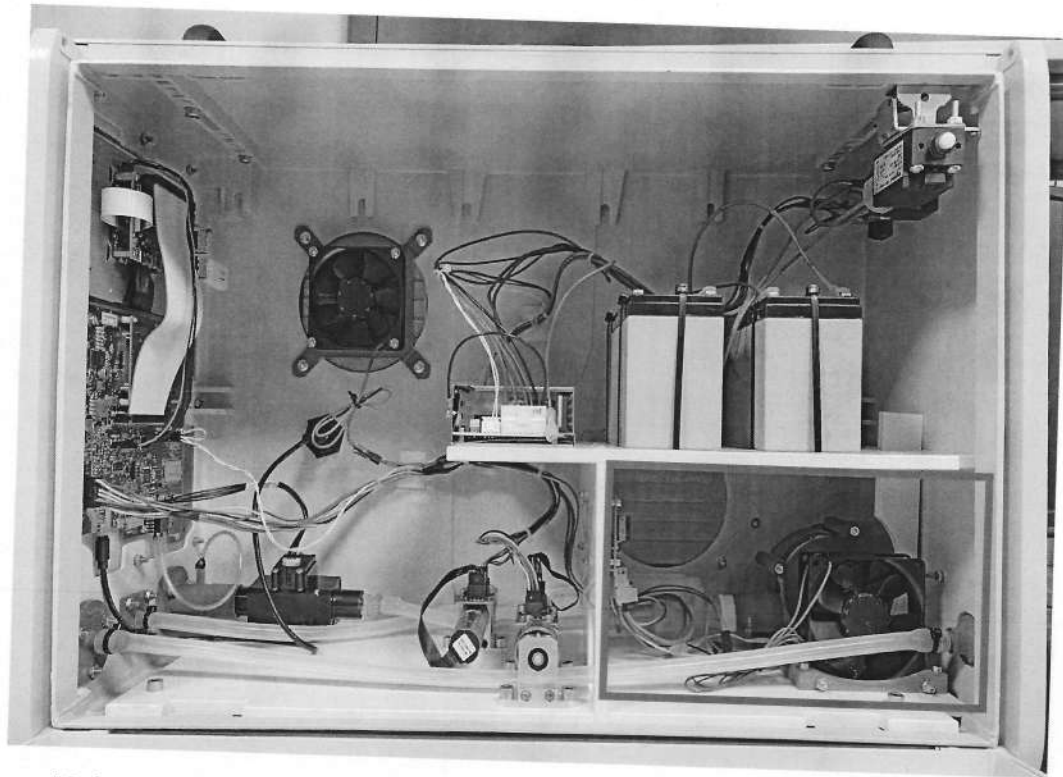
Objectif : Commander la turbine pour qu'elle puisse fournir de l'air au patient avec des intervalles de temps entre chaque période



<b>Nom du cas d'utilisation</b>	<u>UCBN4</u> – Commander turbine
<b>Pré-condition(s)</b>	Le MakAir est allumé et configuré. Il est opérationnel.
<b>Scénario nominal</b>	De manière automatique et complètement autonome, le MakAir commande la turbine
<b>Séquencement</b>	Toutes les <u>TTurbine</u> secondes, la turbine est commandée
<b>Post-condition</b>	La turbine fournit le débit d'air
<b>Exigences</b>	La turbine est commandée en PWM – 24 volts

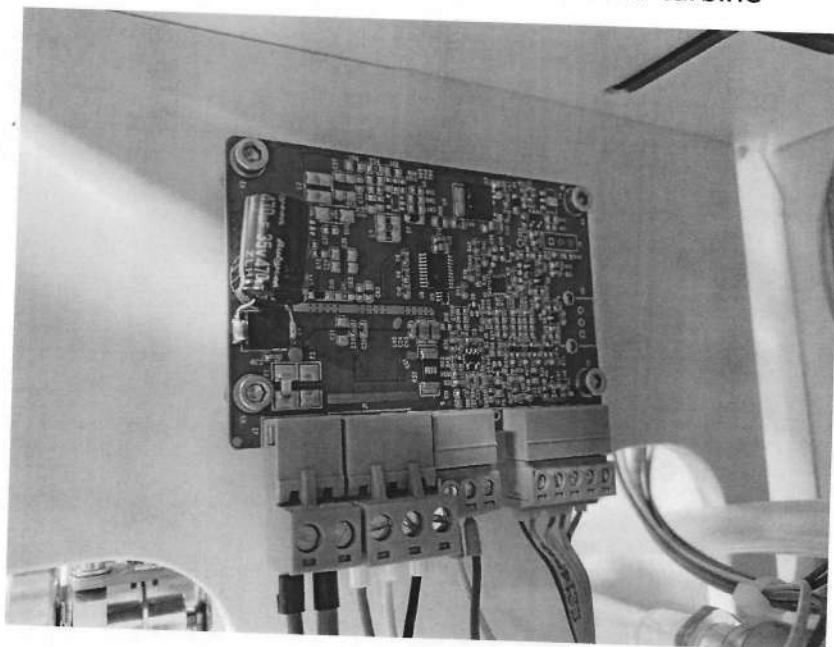
## II.9 Analyse :

Pour réaliser cette tâche j'ai commencé par étudier la partie du makair où se situait la turbine et sa carte de commande.



Makair avec encadré en rouge la turbine et sa carte de commande

Zoom sur la carte de commande de la turbine





## II.11 Réalisation :

Pour réaliser cette partie j'ai commencé par tester le makair en état de marche pour regarder le fonctionnement de la turbine avec les servo-moteur et le débitmètre.

J'ai ensuite commencé un code en PWM pour commander la turbine depuis la carte de puissance

```
Affichage  Atteindre  Exécuter  Terminal  Aide  • main.cpp - Sans titre(Espace de travail) - Visual Studio Code

G+ main.cpp Blower • src  G+ main.cpp Makair • src  PIO Home

Blower > src > G+ main.cpp > loop()
1  #include <Arduino.h>
2
3
4  int Broche_PWM = D13;
5  int Valeur_PWM = 1;//Valeur du rapport cyclique en decimal (255*N%)
6
7  void setup() {
8  |  pinMode(Broche_PWM, OUTPUT);//Choix de la broche PWM
9  |  }
10
11 void loop() {
12 |  for(int T = 0; T < 10 ;T++){//Boucle for pour faire 10 tours
13 |  |  Valeur_PWM = Valeur_PWM + 5;//Augmente la valeur du rapport cyclique a chaque tour
14 |  |  analogWrite(Broche_PWM, Valeur_PWM);//Envoie des données à la turbine
15 |  |  }
16 |  }
```

II.12 Test

Élément testé :	Turbine		
Objectif du test :			
Nom du testeur :	Barreteau Paul	Date :	
Moyens mis en œuvre :	Logiciel : Visual Studio code	Matériel : Turbine	Outil de développement : Visual Studio code
Réalisation :			
Élément testé	Résultat attendu	Résultat obtenu	Validation (O/N)
Conclusion du test :			

III.1.1 Problèmes rencontrés

- *Présentez un historique des problèmes rencontrés pendant la phase de tests unitaires,*
- *Présentez les solutions mises en œuvre pour remédier aux problèmes rencontrés.*

## **IV. Bilan de la réalisation personnelle – Conclusion**

Pour conclure ce projet la partie du cahier des charges sur mesurer le débit d'air est testée et réussie. Si je devais l'améliorer je changerais la carte mère pour ne pas avoir à utiliser l'alimentation de la nucléo pour alimenter la partie I2C.

La partie sur la turbine n'étant pas finis je ne peux donc pas conclure dessus. Je dois donc la finir et la tester.

D'un point de vue personnel ce projet m'a permis de découvrir l'intérêt d'être chef de groupe. J'ai pu apprendre à gérer une équipe et à ordonner les tâches. J'ai aussi pu découvrir la façon de réaliser un projet dans un milieu comme celui-là (médical)