UCLouvain

École polytechnique de Louvain

Conception d'un dispositif d'entraînement au contrôle respiratoire pour la préparation à la radiothérapie

Auteur: **Maurine Poncelet**
Promoteurs: **Benoit Herman, Geneviève Van Ooteghem**
Lecteurs: **Jean-Pierre Raskin, Régis Lomba**
Année académique 2025–2026
Master [120] : ingénieur civil biomédical

Table des matières

Résumé

Les mouvements respiratoires des patients atteints d'une tumeur thoracique lors de leur traitement par radiothérapie représentent un enjeu majeur en matière de précision des radiations. La ventilation mécanique assistée non invasive, développée aux Cliniques universitaires Saint-Luc, permet de contrôler activement le cycle respiratoire et d'offrir des plateaux d'apnée pendant lesquels l'irradiation de la tumeur peut être plus précise. Néanmoins, une telle pratique peut induire une sensation inhabituelle qui peut engendrer de l'anxiété et rendre difficile la tolérance du dispositif. Un entraînement préalable est donc apparu comme une étape incontournable.

Ce projet porte sur la création d'un dispositif capable de reproduire l'action de la MANIV, dans une version portable et à bas coût. L'appareil vise à accompagner le patient dans la découverte et l'entraînement à la sensation de pression positive dans les voies respiratoires avant ses séances de radiothérapie. Le prototype fonctionne avec une turbine centrifuge contrôlée par un algorithme de régulation PID et adopte une architecture pneumatique similaire aux systèmes de CPAP.

Le dispositif développé a traversé une série de tests effectués sur différentes pressions cibles. Ils se sont d'abord déroulés au laboratoire, où de très bons résultats ont révélé une précision de $\pm 0.15 cmH_2O$ et une erreur moyenne inférieure à 0.17 cmH_2O. D'autres tests se sont ensuite déroulés à l'hôpital sur poumon artificiel *EasyLung*, où quelques ajustements logiciels ont dû être effectués pour obtenir des résultats concluants. Un dernier test a été fait sur sujet sain afin de prouver la faisabilité d'une utilisation sur un être humain.

Les résultats obtenus confirment que le prototype présente des résultats encourageants et constitue un premier pas prometteur vers un dispositif d'entraînement cliniquement utilisable, sous réserve de développements complémentaires.

Remerciements

La réalisation de ce mémoire n'aurait pas été possible sans le soutien de nombreuses personnes, à qui je tiens à exprimer ma sincère gratitude.

Je remercie en premier lieu mon promoteur, Monsieur Benoît Herman, pour sa disponibilité, sa pédagogie et ses conseils avisés tout au long de ce travail. Mes remerciements vont également à Madame Geneviève Van Ooteghem, ma co-promotrice, pour son enthousiasme, son écoute, ses conseils et le matériel qu'elle a mis à ma disposition pour la réalisation du prototype.

Je remercie Monsieur Nicolas Audag, kinésithérapeute respiratoire, pour le temps qu'il m'a accordé et les éclairages cliniques qu'il m'a apportés, ainsi que Madame Alicia Hidoud et l'équipe biomédicale des Cliniques Universitaires Saint-Luc pour leur accompagnement bienveillant lors des tests sur poumon artificiel.

Un remerciement tout particulier à Monsieur Simon De Jaeger pour son aide technique précieuse lors de la phase de prototypage, notamment pour la réalisation du circuit électrique.

Je remercie Monsieur Régis Lomba et Monsieur Jean-Pierre Raskin d'avoir accepté de lire ce mémoire et de participer au jury.

Enfin, je remercie chaleureusement mon entourage pour son soutien tout au long de cette année, et plus particulièrement Sophie et mes parents pour la relecture attentive de ce travail.

Liste des abréviations

ADC	Analog-to-Digital Converter (convertisseur analogique-numérique)
APRV	Airway Pressure Release Ventilation
cmH_2O	Centimètre d'eau
CPAP	Continuous Positive Airway Pressure (ventilation en pression positive continue)
ESC	Electronic Speed Controller (contrôleur électronique de vitesse)
HMEF	Heat and Moisture Exchange Filter (filtre échangeur de chaleur et d'humidité)
I2C	Inter-Integrated Circuit
LSB	Least Significant Bit (bit de poids faible)
MANIV	Ventilation Mécanique Assistée Non Invasive
OP-VENT	Open-Source Ventilator ([1])
PID	Proportional-Integral-Derivative (proportionnel-intégral-dérivé)
PVP1	The People's Ventilator Project ([2])
PWM	Pulse-Width Modulation (modulation de largeur d'impulsion)
RPM	Real-Time Position Management
SCL	Serial Clock
SDA	Serial Data
SLM	Standard Liters per Minute (litres standards par minute)
SpO2	Saturation pulsée en oxygène

Introduction

Le cancer demeure aujourd'hui l'une des principales causes de mortalité dans le monde, à mesure que le nombre de cas ne cesse d'augmenter. Face à cette réalité, la radiothérapie occupe une place centrale parmi les traitements disponibles. Son efficacité repose toutefois sur une contrainte majeure : la capacité à délivrer les rayonnements avec une exactitude constante tout au long du traitement. Or, cette précision reste vulnérable aux mouvements du patient, susceptibles de déplacer la tumeur et d'exposer les tissus sains à une irradiation. Parmi ces mouvements, les plus contraignants sont ceux liés à la respiration, en particulier pour les tumeurs situées à proximité du circuit respiratoire.

Pour faire face à ce défi, les chercheurs des Cliniques Universitaires Saint-Luc et de l'UCLouvain ont développé une approche innovante : la ventilation mécanique assistée non invasive. En prenant le contrôle du cycle respiratoire du patient à l'aide d'un respirateur médical, cette technique permet de stabiliser et de moduler activement le schéma ventilatoire, offrant des plateaux de pression reproductibles qui constituent des fenêtres de tir idéales pour le faisceau d'irradiation. Des travaux publiés par l'équipe ont démontré la faisabilité et l'efficacité de cette approche, montrant notamment que la MANIV améliore significativement la reproductibilité des mouvements respiratoires au sein d'une même séance et entre les différentes séances, et ce sans événement indésirable. [3, 4]

Cependant, la MANIV impose au patient une contrainte respiratoire à laquelle il n'est pas habitué, la sensation liée à la pression positive insufflée dans les poumons peut générer de l'anxiété et nuire à la tolérance de la machine, en particulier chez des patients déjà fragilisés par leur maladie. Un entraînement préalable permettrait de familiariser le patient avec cette sensation avant la séance de radiothérapie, d'améliorer sa tolérance et d'augmenter les chances de succès du traitement. À ce jour, aucun dispositif spécifiquement conçu à cet effet n'existe sur le marché.

C'est dans ce contexte que s'inscrit l'objectif de ce projet. Le but est de concevoir, implémenter et valider un dispositif d'entraînement respiratoire portable et à bas coût, capable de reproduire la sensation de la MANIV afin de préparer le patient avant ses séances à l'hôpital. Ce dispositif doit être suffisamment simple pour être utilisé de manière autonome par le patient à domicile et suffisamment précis pour reproduire fidèlement la sensation imposée par le respirateur médical.

Ce document présente l'ensemble de la démarche de conception et de validation du prototype. Le premier chapitre établit le contexte clinique du travail, en décrivant le rôle de la radiothérapie dans le traitement du cancer, les enjeux liés aux mouvements respiratoires, et le fonctionnement de la MANIV. Le deuxième chapitre détaille l'approche de conception adoptée et les spécifications techniques du dispositif. Les chapitres trois et quatre décrivent respectivement l'implémentation matérielle du prototype et le développement de l'algorithme de régulation en pression. Enfin, le dernier chapitre présente une discussion critique des résultats.

Chapitre 1

Contexte

1.1 Le cancer

Le cancer est une maladie omniprésente dans notre monde qui touche chaque année des millions de personnes. Le nombre de cas annuels diagnostiqués est en constante augmentation. Selon les données de la Global Burden of Disease Study 2023, on estime qu'il y a eu 18,5 millions de nouveaux diagnostics de cancer dans le monde en 2023, contribuant à un total de 271 millions d'années de vie ajustées sur l'incapacité.[5] Les projections indiquent que le nombre de nouveaux diagnostics et de décès par cancer devrait augmenter considérablement entre 2024 et 2050, avec une augmentation estimée respectivement de 60,7 % et 74,5 %.[5] Ces projections sont confirmées par d'autres études internationales, qui estiment qu'en l'absence d'intervention, le nombre de cas pourrait atteindre 29,5 millions et les décès 16,3 millions d'ici 2040.[6, 7] Face à cette réalité, la radiothérapie constitue un pilier indispensable pour répondre à cette maladie.[6]

1.2 La radiothérapie

La radiothérapie est l'une des options thérapeutiques les plus utilisées pour traiter les cancers.[8] Elle repose sur l'utilisation de rayonnements à haute énergie pour détruire les cellules cancéreuses tout en préservant au maximum les tissus sains.[9] Ces rayons vont agir en endommageant l'ADN des cellules cancéreuses, ce qui va les empêcher de se diviser et de proliférer. Elle peut être utilisée seule ou en combinaison avec d'autres voies de traitement comme la chirurgie ou la chimiothérapie. Les traitements sont adaptés en fonction du type de cancer, de son stade ainsi que de sa localisation.[10] La radiothérapie a deux buts principaux ; le premier est curatif pour les cancers plutôt solides et le second est palliatif afin d'améliorer la qualité de vie des patients ainsi que diminuer les symptômes.[8]

1.2.1 Importance de la précision

Si la radiothérapie est autant utilisée aujourd'hui, c'est grâce à son efficacité. Cette efficacité est directement liée à la précision des rayonnements sur la tumeur. Cette précision est essentielle pour deux raisons : cibler au mieux la tumeur pour détruire les cellules cancéreuses et éviter au maximum d'endommager les tissus sains, d'autant plus lorsque ce sont des organes à risque.[11]

Cette précision peut être grandement compromise par les mouvements du patient. Ceux-ci peuvent être classés en deux catégories : les mouvements volontaires, liés par exemple à une modification de posture, et les mouvements involontaires, tels que la respiration ou les micro-mouvements internes des organes. Différentes stratégies sont utilisées pour limiter leur impact, notamment l'immobilisation du patient, la surveillance du positionnement, l'adaptation des marges de traitement ou encore les techniques de gestion dynamique du mouvement.[12]

Parmi les mouvements involontaires, la respiration est l'un des plus importants, en particulier pour les tumeurs qui se situent au niveau du thorax (foie, pancréas, sein, poumon). Elle peut entraîner des déplacements significatifs qui sont variables d'une séance à l'autre. Il est donc indispensable de les prendre en compte pour maintenir la précision du traitement.[12]

1.3 Le gating respiratoire

Pour répondre à cette difficulté, des stratégies de gestion respiratoire ont été développées. Parmi elles, le "gating respiratoire" consiste à synchroniser l'irradiation de la tumeur avec une phase précise du cycle respiratoire, afin de réduire les marges nécessaires autour de la tumeur et de limiter l'irradiation des tissus sains.[12]

Traditionnellement, pour compenser l'incertitude liée aux mouvements respiratoires, on élargit les volumes de traitement avec des marges internes puis des marges de sécurité supplémentaires. Cette méthode est efficace, mais elle augmente le volume de tissus sains irradiés et peut donc limiter l'augmentation de la dose.[13]

Le gating respiratoire a donc été développé comme une alternative permettant de mieux maîtriser le mouvement de la tumeur. Deux approches principales se distinguent : le gating en respiration libre, qui ne demande pas de participation active du patient, et le blocage de respiration, qui repose au contraire sur une apnée volontaire ou assistée.[14]

1.4 Types de gating

Il existe différentes approches pour la synchronisation respiratoire. Deux approches principales se distinguent. L'une analyse la respiration naturelle du patient alors que l'autre demande sa participation active.[14]

1.4.1 Le gating en respiration libre (Free-breathing gating) :

Cette méthode ne demande aucune participation du patient. Elle consiste en l'analyse du cycle respiratoire du patient, et donc déclencher les radiations lors d'une des phases respiratoires. Un exemple de système utilisé pour le gating en respiration libre est le système RPM. Il fonctionne avec une caméra infrarouge qui va capter les mouvements d'un petit boîtier muni de réflecteurs placé sur l'abdomen du patient. Le faisceau de radiations est interrompu entre chaque cycle et la dose est délivrée par petites fractions lors d'une fenêtre de tir, souvent choisie pendant la phase d'expiration où le mouvement résiduel est le plus faible et reproductible.[15] Cette méthode, bien qu'efficace, rallonge considérablement le temps de l'intervention car le faisceau n'est activé que pendant une partie du cycle respiratoire.

1.4.2 Le blocage de respiration (Breath-hold) :

Ici, on va demander au patient de bloquer sa respiration après une grande inspiration. On profite de cette apnée pour effectuer les acquisitions d'images et délivrer les radiations. Elle peut être réalisée de manière active via une valve qui bloque temporairement le flux d'air, ou de manière volontaire par le patient lui-même.[14]

1.5 Bénéfices cliniques

Ces deux techniques apportent plusieurs bénéfices considérables. Elles permettent notamment de mieux protéger les organes à risque et de réduire l'irradiation des tissus sains.[14] En choisissant la phase respiratoire adéquate, on éloigne la cible des zones critiques. Ensuite, on observe également une diminution de la toxicité cardiaque et pulmonaire. Le gating en inspiration profonde augmente le volume pulmonaire et éloigne le cœur de la paroi thoracique, ce qui permet de réduire les doses reçues par le cœur et les poumons.[16, 17] Enfin, en réduisant l'irradiation des tissus sains, ces méthodes peuvent offrir la possibilité d'envisager une augmentation de la dose délivrée à la tumeur dans certaines situations cliniques, sans accroître nécessairement le risque de complications.

1.6 Importance de l'entraînement

Ces bénéfices dépendent cependant grandement de la régularité et de la reproductibilité de la respiration du patient ; ce qui est assez complexe car la respiration est une constante très variable. Sans intervention externe ou entraînement, les patients éprouvent des difficultés à maintenir un schéma respiratoire stable d'un jour à l'autre. L'entraînement du patient est donc essentiel pour maximiser l'efficacité. Différents outils d'entraînement ont été étudiés comme le guidage audio. Cette méthode consiste à utiliser des consignes vocales enregistrées pour indiquer au patient quand inspirer et expirer. Si cette technique permet de rendre le rythme respiratoire régulier, elle entraîne paradoxalement des mouvements du ventre plus amples et plus irréguliers. Ces mouvements compliquent le ciblage précis de la zone à traiter, et donc restreignent la fenêtre de tir. De plus, les instructions verbales sont plutôt difficiles à suivre pour les patients.[18]

Il y a aussi le retour visuel. Le schéma respiratoire est montré au patient en temps réel. Les patients sont entraînés à maintenir l'amplitude de leur respiration entre des seuils prédéfinis. Cette méthode offre un bien meilleur contrôle de l'amplitude du mouvement, même si elle présente parfois de légères variations de fréquence.[18] Ces méthodes d'entraînement peuvent néanmoins présenter certaines limites. Elles nécessitent une participation active du patient et supposent que celui-ci soit capable de comprendre les consignes et de reproduire la manoeuvre demandée. Elles peuvent donc être difficiles à appliquer chez les patients présentant une mauvaise tolérance, une anxiété importante ou une capacité limitée à contrôler leur respiration.[19]

Pour répondre à ces limitations, l'utilisation de la ventilation mécanique assistée non-invasive a été introduite comme une approche permettant de réduire la dépendance à la coopération respiratoire volontaire du patient.

1.7 La Ventilation Mécanique Assistée Non Invasive en Radiothérapie

1.7.1 Introduction

Afin de pallier les limites des méthodes classiques, une approche innovante de la MANIV a été développée et évaluée par les chercheurs de l'UCLouvain et des Cliniques Universitaires Saint-Luc. Leur démarche consiste à utiliser la MANIV pour stabiliser, mais aussi pour moduler activement le modèle respiratoire du patient. L'objectif est de prendre le contrôle complet du cycle respiratoire à l'aide d'un respirateur artificiel, offrant ainsi un compromis idéal entre la respiration spontanée aléatoire et l'anesthésie générale.[3, 4]

1.7.2 Équipements

Le dispositif principal utilisé est un ventilateur mécanique médical. Le modèle Bellavista 1000® est connecté au patient via un réseau de tuyaux ainsi qu'un masque englobant le nez et la bouche. Cela constitue une interface non-invasive pour le patient. La mise en place de ce matériel répond à un protocole strict visant à assurer la sécurité du patient et la précision de la ventilation.[20] Voici le protocole :

- **L'interface patient** : le masque facial est strictement nominatif. Son choix de taille (Small, Medium ou Large) est crucial pour éviter les fuites d'air et se détermine à l'aide d'un étalon spécifique mesurant la distance entre le menton et la racine du nez du patient. Une fois mis en place avec des lanières ajustables, un test d'étanchéité est réalisé par le patient lui-même.
- **Le circuit respiratoire** : l'installation requiert un double circuit de ventilation (tubulures larges) et un double circuit de pression (tubulures fines). Un filtre anti-microbien est placé à l'extrémité du patient à chaque utilisation.
- **Le monitorage** : La sécurité et la tolérance du patient sont assurées par une surveillance continue comprenant un capnographe (mesure du CO2 expiré) relié par un adaptateur respiratoire, ainsi qu'un saturomètre pour surveiller la saturation pulsée en oxygène (SpO2) et la fréquence cardiaque. L'apport en oxygène est branché sur une prise murale et géré par le ventilateur.
- **Calibration** : avant chaque utilisation quotidienne, le système subit un « Circuit Test » pour vérifier l'absence de fuite, suivi d'une calibration précise du capteur de CO2.[20]

1.7.3 Modes de ventilation

L'approche MANIV repose sur l'alternance de différents modes ventilatoires selon la phase du traitement (préparation, observation, irradiation) et les besoins cliniques.

La phase préparatoire :Le mode AVM (Adaptative Ventilation Mode)

Le mode AVM, qui peut être assimilé à la respiration spontanée, est utilisé lors de la mise en place du patient. Dans ce mode, le patient reste le déclencheur de sa propre respiration. La machine s'adapte et délivre l'air requis au gré de la respiration du patient sans imposer de contrainte de volume ou de pression. Ce mode présente un double intérêt :

1. Il permet de mesurer les paramètres ventilatoires spontanés du patient pour personnaliser les réglages ultérieurs.

2. Il permet une hyperoxygénation préparatoire sans effort (la fraction inspirée en oxygène, FiO2, est généralement réglée à 60%).[20]

La modulation continue : Les modes contrôlés en Volume et Superficiel

Ces modes visent à imposer un modèle respiratoire régulier. Le mode contrôlé en volume fixe le volume courant et la fréquence respiratoire dans des conditions physiologiques, réduisant ainsi drastiquement la variabilité de la période respiratoire par rapport à la respiration libre. Le mode superficiel accélère la fréquence respiratoire (jusqu'à 30 respirations par minute) tout en diminuant proportionnellement le volume courant, ce qui réduit significativement l'amplitude du mouvement interne moyen des organes.[20]

Le mode de traitement par gating : Le mode APRV

C'est sur ce mode que repose la stratégie de "gating" proprement dite. Le mode APRV prend le contrôle total de la respiration du patient. Le ventilateur génère de manière mécanique une alternance entre deux niveaux de pression positive. La pression haute contraint mécaniquement l'inflation pulmonaire et impose un plateau d'apnée inspiratoire. La pression basse permet l'exhalation. Ces plateaux de fin d'inspiration se répètent (généralement 3 fois par minute) et durent plus de 10 secondes. Ces plateaux hautement reproductibles matérialisent la « fenêtre de tir » temporelle au cours de laquelle le faisceau d'irradiation est activé.[20]

1.7.4 Difficultés liées à l'état du patient et à la tolérance de la MANIV

La prise en charge d'un cancer s'accompagne fréquemment d'un stress psychologique important. Les patients peuvent présenter de l'anxiété, une détresse émotionnelle ou une fatigue mentale liées au diagnostic, aux examens, aux rendez-vous hospitaliers et à l'anticipation des traitements. Cette détresse n'est pas anodine, car elle peut influencer la capacité du patient à suivre correctement les consignes médicales et à s'adapter à une procédure thérapeutique exigeante.[21]

Dans le cadre de la radiothérapie, cette situation est particulièrement importante puisque le patient doit déjà faire face à un environnement technique, à des contraintes de positionnement et à une coopération respiratoire précise. Un niveau de stress élevé peut donc nuire à la stabilité du schéma ventilatoire, à la compréhension des consignes et à la reproductibilité des manœuvres respiratoires attendues. Des travaux ont d'ailleurs montré qu'une détresse plus marquée chez les patients traités par radiothérapie peut être associée à des

rendez-vous manqués et à une moins bonne adhérence au parcours de soins.[21]

Cette difficulté peut être encore accentuée par l'utilisation de la ventilation mécanique assistée non invasive. Le port d'un masque facial, la sensation de flux d'air imposé et la contrainte respiratoire peuvent générer un inconfort, une sensation de confinement, voire une intolérance chez certains patients. Dans la littérature, l'utilisation de la ventilation non invasive est justement limitée par des facteurs comme l'inconfort, la claustrophobie et la mauvaise tolérance du masque. Dans le cas de la MANIV, l'insufflation de pression positive dans les poumons peut être ressentie comme inhabituelle et déstabilisante, ce qui risque de compliquer l'acceptation du dispositif et le maintien d'un comportement respiratoire régulier.[22]

Ces difficultés rencontrées par les patients justifient l'intérêt d'un dispositif d'entraînement préalable. En familiarisant le patient avec la sensation de pression, le rythme imposé et la présence du masque à l'aide d'un dispositif d'entraînement, il deviendrait possible de réduire l'appréhension, d'augmenter la tolérance et d'améliorer la reproductibilité du geste respiratoire. L'entraînement constitue ainsi une étape intermédiaire essentielle entre la respiration spontanée et l'utilisation clinique de la MANIV.

Chapitre 2

Design

2.1 Approche de conception

La conception du dispositif d'entraînement respiratoire s'est déroulée en trois phases : la définition des besoins, l'exploration des solutions disponibles et le développement du prototype.

La première phase a consisté à identifier les besoins utilisateurs en concertation avec l'équipe médicale des Cliniques Universitaires Saint-Luc. Cette étape a mené à la définition des spécifications techniques présentées dans la section suivante, qui constituent le cahier des charges du dispositif.

La deuxième phase fut la recherche des différentes solutions déjà existantes. Cette phase a donné lieu à un état de l'art des différents respirateurs open source disponibles. L'objectif était de rassembler les ressources pertinentes et pouvant servir d'inspiration à l'implémentation du dispositif et ne pas devoir partir de zéro. Cette analyse, présentée en section 2.3, a permis de retenir la principale référence du projet qui est le ventilateur de Garmendia et al., en raison de sa simplicité, de son faible coût et de son potentiel lié aux contraintes du projet.[23]

La troisième phase est le coeur du travail, car il s'agit du développement itératif du prototype. Il a été développé sur base de l'architecture retenue lors de l'état de l'art tout en l'adaptant aux spécifications définies en phase une. Elle comprend la sélection des composants, leur caractérisation, la conception du circuit, l'implémentation de l'algorithme puis la validation en laboratoire et sur poumon artificiel.

2.2 Spécifications du dispositif

2.2.1 Besoins utilisateurs

La définition des spécifications du dispositif se fait en analysant les besoins des deux catégories de personnes qui seront amenées à utiliser l'appareil : les médecins, qui vont le prescrire et le configurer, et les patients, qui seront les

utilisateurs premiers.

Du côté du clinicien, le besoin principal est de pouvoir paramétrer précisément le comportement ventilatoire que la machine adoptera. Les paramètres qui doivent être modifiables par le médecin sont : la durée d'inspiration, la pression de maintien, la durée de ce maintien de pression et la durée de l'expiration. Il est important que ces paramètres soient adaptables à chaque patient pour reproduire au mieux l'activité de la MANIV en hôpital, et, pour certains, être adaptés au long de la phase d'entraînement (comme la durée du maintien de pression). Il est ainsi indispensable que le réglage de ces paramètres soit uniquement accessible au personnel médical en charge du patient, afin d'éviter toute modification non contrôlée par le patient. Enfin, le clinicien a besoin d'être certain que le dispositif ne sera pas utilisé de manière excessive ou incorrecte entre les séances.

Du côté du patient, l'objectif est d'avoir un appareil qu'il peut utiliser seul, sans formation technique particulière. L'interface et le fonctionnement doivent être simples et intuitifs pour garantir une expérience positive auprès de chacun. Étant destiné à un usage à domicile, le dispositif doit être compact et portable. Enfin, des mécanismes de sécurité doivent être intégrés pour protéger le patient en cas de comportement respiratoire anormal durant la séance, ou de problème technique.[19]

2.2.2 Spécifications techniques

Compte tenu de ces besoins, les spécifications techniques suivantes ont été définies en concertation avec l'équipe des Cliniques Universitaires Saint-Luc et listées dans le tableau 2.1. Elles sont organisées en quatre catégories : le profil ventilatoire, les sécurités, la mécanique du circuit d'air, et les contraintes d'usage.

TABLE 2.1 – Spécifications techniques du dispositif[19]

Catégorie	Spécification	Valeur / Contrainte
Profil ventilatoire	Durée de la phase inspiratoire	150 – 300 ms (prédéfinie)
	Pression maximale maintenue	$\leq 20\ cmH_2O$
	Durée du maintien de pression (plateau)	≤ 20 secondes
	Durée de la phase expiratoire	Variable (prédéfinie par le clinicien)
	Mode d'expiration	Non assistée — simple relâchement de pression
Sécurités	Dépassement du seuil inspiratoire	Arrêt immédiat de la session
	Liberté expiratoire	Expiration libre à tout moment, sans résistance
Circuit pneumatique	Architecture des voies d'air	Double circuit : entrée et sortie séparées
	Source d'air	Air ambiant — aucun lien avec l'oxygène médical
Contraintes d'usage	Portabilité	Dispositif compact, usage domiciliaire
	Interface patient	Utilisation simple, sans formation technique
	Accès aux réglages	Réservé au clinicien (accès protégé)
	Limitation des sessions	Blocage automatique après un nombre prédéfini d'entraînements

2.2.3 Précisions sur les spécifications techniques

La durée d'inspiration courte (150–300 ms) correspond au profil imposé par le MANIV, où la phase de remplissage doit être rapide pour structurer un cycle respiratoire régulier et reproductible.

Le plateau de pression est ici défini à 20 secondes alors qu'il est de 30 secondes sur le MANIV. Ce choix est justifié par le fait que le dispositif sert d'outil d'entraînement pour le patient. Les 30 secondes sur le MANIV sont essentielles, car c'est l'intervalle de temps pendant lequel la tumeur recevra les radiations. Le dispositif d'entraînement, lui, sert à habituer le patient au mécanisme et à l'entraîner à recevoir de tels plateaux de pressions. Il n'est donc pas nécessaire d'avoir des plateaux de pressions aussi longs lors de cet entraînement. De plus, contrairement au MANIV clinique qui dispose d'un apport en oxygène médical, le prototype fonctionne à l'air ambiant. En l'absence d'oxygène supplémentaire, il est prudent de limiter la durée du plateau afin d'éviter toute diminution de l'apport d'oxygène du patient. La limite en pression de ce plateau $\leq 20\ cmH_2O$ est une indication, car chaque patient a une pression définie pour lui, mais ces pressions ne dépassent jamais 20 cmH_2O.

La double voie pneumatique est une contrainte fonctionnelle importante : elle permet de séparer le flux entrant (air sous pression délivré au patient) du flux sortant (expiration), évitant toute réinhalation d'air expiré et assurant une expiration libre sans résistance.

L'absence de connexion à l'oxygène médical est un choix délibéré qui simplifie considérablement l'usage à domicile. Le dispositif fonctionne à l'air ambiant, ce qui le rend autonome et ne le soumet pas aux contraintes de prescription et de gestion des gaz médicaux.

Le blocage après un certain nombre de sessions est une mesure à la fois de sécurité et de conformité clinique. Elle évite que le patient ne s'entraîne de manière non encadrée au-delà du programme défini par le médecin et garantit un suivi régulier.[19]

2.3 État de l'art des respirateurs open source

Afin d'optimiser la conception du dispositif, un état de l'art des respirateurs open source a permis d'identifier plusieurs solutions techniques susceptibles de servir de référence. L'analyse s'est concentrée sur trois projets répondant, au moins partiellement, au cahier des charges défini pour ce mémoire : le ventilateur décrit par Garmendia et al., OP-VENT et PVP1. Leur étude comparative a permis d'évaluer la compatibilité de leurs caractéristiques avec les besoins du projet et de mettre en évidence les adaptations nécessaires.

2.3.1 Ventilateur à support de pression non invasif, bas coût - Garmendia et al.

Le ventilateur présenté par Garmendia et al. dans l'article *Low-cost, easy-to-build noninvasive pressure support ventilator for under-resourced regions* constitue une première référence pertinente en raison de sa simplicité de conception et de son faible coût.[23] Le dispositif repose sur une architecture open source relativement accessible, fondée sur une turbine haute pression, deux capteurs de pression différentielle et un microcontrôleur Arduino Nano chargé d'analyser les signaux respiratoires et de commander les phases inspiratoires et expiratoires. Cette configuration présente l'avantage d'être reproductible avec des composants courants, ce qui en facilite la fabrication et l'adaptation.

Le fonctionnement repose sur une logique de pression support, avec détection des efforts inspiratoires du patient et déclenchement des pressions correspondantes. Ce principe est intéressant dans le cadre du présent mémoire, dans la mesure où il montre qu'un contrôle respiratoire peut être obtenu à l'aide d'une architecture simple et peu coûteuse. Toutefois, ce ventilateur demeure conçu pour une assistance respiratoire générale et ne répond pas directement aux contraintes spécifiques d'un dispositif d'entraînement pré-thérapeutique. Des éléments tels qu'un temps inspiratoire strictement défini, une durée d'utilisation limitée ou encore un verrouillage après un certain nombre de sessions devraient donc être ajoutés ou modifiés.

2.3.2 OP-VENT

OP-VENT repose sur une architecture plus sophistiquée, centrée sur une électrovanne proportionnelle contrôlée en boucle fermée.[1] Ce choix technique permet un contrôle précis du débit et de la pression, ce qui constitue un avantage majeur pour obtenir un profil ventilatoire stable et reproductible. L'interface utilisateur a été volontairement simplifiée afin de limiter la complexité d'utilisation tout en conservant les réglages nécessaires au paramétrage clinique.

Les caractéristiques particulièrement intéressantes de ce prototype sont la séparation nette entre la voie inspiratoire et la voie expiratoire, la présence de mécanismes de sécurité intrinsèques, ainsi que la possibilité de moduler plusieurs paramètres ventilatoires. Le dispositif offre ainsi une base solide pour l'obtention d'un cycle respiratoire contrôlé. En revanche, son architecture dépend d'une source externe d'air comprimé, ce qui réduit son autonomie et sa portabilité dans un contexte d'utilisation à domicile ou hors environnement hospitalier. Malgré cette limite, OP-VENT constitue une référence intéressante pour la logique de contrôle et pour l'intégration des sécurités.

2.3.3 PVP1

PVP1 (The People's Ventilator Project) se distingue par son caractère entièrement open source, sa modularité et sa vocation de plateforme de recherche.[2] Le dispositif utilise une commande en pression associée à une interface clinique conçue pour permettre des réglages précis et intuitifs. Cette architecture en fait un système particulièrement intéressant pour des applications nécessitant un haut niveau de flexibilité et de reproductibilité.

Le PVP1 prend en charge des modes ventilatoires contrôlés et permet d'ajuster plusieurs paramètres, notamment la pression cible, la fréquence respiratoire et le temps de montée en pression. Il présente également des dispositifs de sécurité robustes ainsi qu'une bonne stabilité de fonctionnement sur un grand nombre de cycles. En revanche, son format et ses dépendances techniques le rapprochent davantage d'un ventilateur de recherche ou de soins que d'un dispositif compact spécifiquement conçu pour l'entraînement préalable à la MANIV. Par conséquent, son utilité dans le cadre de ce prototype se concentre sur l'excellence de son architecture logicielle et ses facultés d'adaptation.

2.3.4 Analyse synthétique

L'étude de ces trois dispositifs montre qu'aucun ne correspond exactement aux besoins définis précédemment, mais chacun apporte des éléments utiles à la conception finale. Le ventilateur de Garmendia et al. se distingue toutefois comme la solution la plus proche de l'objectif visé, en particulier en raison de son caractère low-cost, de sa simplicité de fabrication et de son architecture relativement compacte et portable. À ce titre, il constitue la principale source d'inspiration pour le développement du dispositif, tandis qu'OP-VENT et PVP1 apportent des références complémentaires, notamment en matière de contrôle ventilatoire, de sécurité et de modularité logicielle.

Cette analyse comparative confirme ainsi la nécessité de développer une solution spécifique, pensée non comme un respirateur clinique complet, mais comme un outil d'entraînement adapté à la préparation au MANIV. Un tel dispositif doit conserver une maîtrise fine du profil respiratoire tout en restant simple, compact et suffisamment intuitif pour un usage encadré.

TABLE 2.2 – Analyse comparative des respirateurs open source

Critère	Garmendia et al.	OP-VENT	PVP1
Actionneur	Turbine centrifuge (WS7040, 15V)	Électrovanne proportionnelle	Valve proportionnelle industrielle
Source d'air	Turbine autonome	Air comprimé externe	Air comprimé externe
Microcontrôleur	Arduino Nano	ATMega1284P	Raspberry Pi
Capteurs	2 capteurs de pression différentielle	Pression et débit	Pression et débit
Modes ventilatoires	BiPAP, CPAP	Volume contrôlé, pression contrôlée	Pression contrôlée (PCV), SIMV
Régulation	Seuil de débit	PID imbriqué (débit + pression)	PID (pression)
Interface	Boutons rotatifs, écran LCD	Écran LCD, encodeurs rotatifs	Écran tactile
Coût	Faible	Faible	∼$1700 USD
Open source	Oui (hardware + software)	Oui	Oui
Portabilité	Bonne	Limitée (source externe)	Limitée (source externe)
Pertinence	Élevée	Moyenne	Moyenne

Chapitre 3

Implémentation du prototype

3.1 Architecture générale du système

Le dispositif repose sur une architecture à fuite intentionnelle, inspirée du principe de fonctionnement de certains systèmes de ventilation en pression positive continue (CPAP). L'architecture générale du système distingue le circuit pneumatique illustré à la figure 3.1 et le circuit électrique présenté sur la figure 3.2.

Le flux d'air est généré par une turbine centrifuge contrôlée par un contrôleur de vitesse électronique (ESC), lui-même piloté par un microcontrôleur Arduino. L'air produit circule ensuite dans une tubulure souple reliée à la turbine par des connecteurs adaptés. À la sortie de cette tubulure, un clapet anti-retour assure l'unidirectionnalité du flux inspiratoire et limite tout reflux d'air expiré vers la turbine.

Une valve de fuite réglable est placée en aval du clapet. Elle permet au patient d'expirer librement lorsque la turbine est à l'arrêt. Mais elle constitue aussi une sécurité passive en assurant en permanence une voie de sortie d'air. Lors des phases d'insufflation et de maintien en pression, le débit généré par la turbine compense cette fuite afin de maintenir la pression cible dans le circuit. Le circuit intègre ensuite un capteur de pression qui permet de mesurer la pression appliquée dans la ligne respiratoire et de tenir compte des pertes de charge liées à la longueur de la tubulure.

Un capteur de débit est ensuite placé à proximité du patient afin de mesurer le flux réellement échangé au niveau du masque. Cette position permet de suivre à la fois les phases inspiratoires et expiratoires du cycle respiratoire. Enfin, un filtre échangeur de chaleur et d'humidité (HMEF) est placé au plus près du patient, entre le capteur de débit et le masque facial. Il contribue au confort respiratoire en réchauffant et humidifiant l'air inspiré tout en assurant une barrière bactérienne et virale entre le circuit et les voies aériennes.

L'ensemble du système est piloté par un microcontrôleur Arduino Mega, qui assure l'acquisition des capteurs et la génération du signal PWM destiné à l'ESC de la turbine. Un convertisseur analogique-numérique 16 bits est intercalé entre le capteur de pression et le microcontrôleur afin d'améliorer la résolution de mesure pour les faibles pressions mises en jeu.

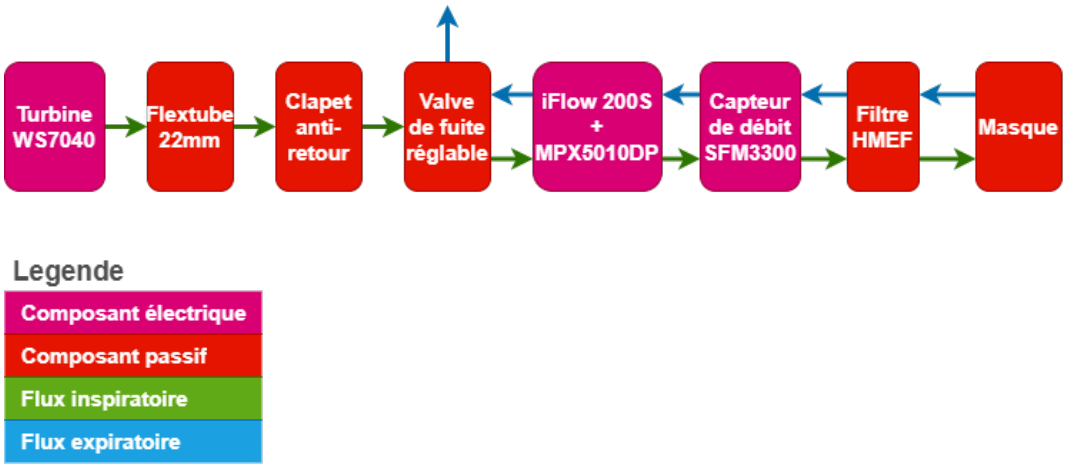


FIGURE 3.1 – Circuit pneumatique du prototype

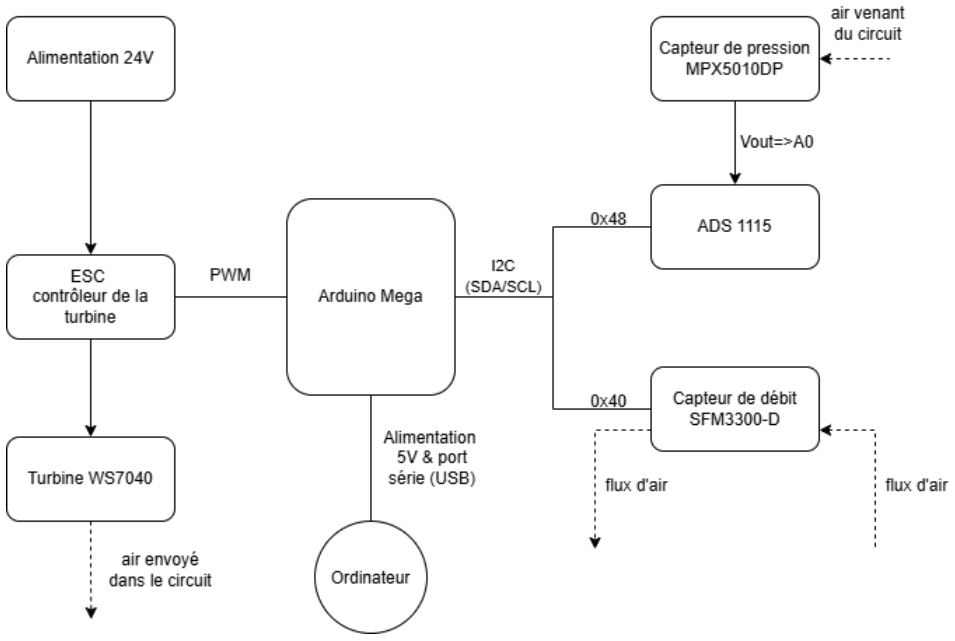


FIGURE 3.2 – Circuit électrique du prototype

3.2 Composants matériels

Une fois l'architecture générale du dispositif définie, il convient de détailler les composants qui le constituent ainsi que les raisons qui ont conduit à leur sélection. Cette analyse permet de préciser le rôle de chaque élément dans le système et de justifier sa compatibilité avec les contraintes techniques, fonctionnelles et économiques du projet. Elle assure ainsi la transition entre la conception globale et la réalisation concrète du dispositif.

3.2.1 La turbine

La turbine constitue l'élément moteur du système. Elle génère le flux d'air nécessaire à l'inspiration du patient. Le modèle retenu est la WS7040, une turbine centrifuge de type blower initialement conçue pour des applications de ventilation forcée. Elle est branchée à une alimentation 24V et sa vitesse de rotation est modulée par un signal PWM issu du microcontrôleur, transmis à la turbine via un ESC (Electronic Speed Controller).

Ce modèle a été retenu pour plusieurs raisons. D'une part, ses caractéristiques aérodynamiques permettent de générer des pressions et des débits compatibles avec les exigences thérapeutiques visées. D'autre part, ce type de turbine est déjà utilisé dans des travaux de recherche sur les respirateurs open source, notamment dans l'article de référence de ce projet ; ce qui facilite la comparaison des résultats. Enfin, son faible encombrement et son coût modéré sont cohérents avec l'objectif de développer un dispositif accessible.

3.2.2 Tuyauterie

Le tuyau assure le transport de l'air entre la turbine et le patient. Il doit être suffisamment long pour permettre au patient d'être installé confortablement sans que la contrainte du circuit n'exerce de traction sur le masque. Le modèle utilisé est le Flextube d'Intersurgical, une tubulure annelée en polyéthylène. Son diamètre standardisé de 22 mm assure la compatibilité avec les connecteurs ISO médicaux des autres composants du circuit, sauf pour la connexion avec la turbine, pour laquelle une pièce sur mesure imprimée en 3D a été réalisée.

3.2.3 Clapet anti-retour

Le clapet anti-retour est positionné au bout de la tubulure et garantit l'unidirectionnalité du flux dans la branche inspiratoire. Son rôle est d'empêcher l'air chaud et humide, expiré par le patient, de retourner vers la turbine ; ce qui pourrait l'endommager. Ce composant est purement mécanique et passif : il s'ouvre sous l'effet de la surpression générée par la turbine lors de l'inspiration, et se referme naturellement lors de l'expiration.

3.2.4 Valve de fuite réglable

La valve de fuite est un composant médical standard utilisé dans les circuits CPAP. Elle se présente sous la forme d'un raccord tubulaire transparent équipé d'un orifice latéral dont l'ouverture est ajustable via une molette. Ce réglage permet d'adapter le débit de fuite aux caractéristiques du patient et aux paramètres de pression cible de façon à ce que la turbine puisse toujours compenser la fuite et maintenir la pression souhaitée. La position de la molette constitue donc un paramètre de réglage clinique du dispositif.

Choix du système d'expiration

La phase expiratoire, et surtout sa gestion, est un aspect critique de la conception du dispositif. En effet, la plupart des systèmes de ventilation sur le marché n'ont pas de phase de maintien en pression, l'expiration a lieu directement après la fin de l'inspiration. Il est donc indispensable de trouver une solution permettant au système de garder sa pression positive lors de la phase de maintien en pression et de permettre l'expiration du patient à la fin de celle-ci, tout en garantissant une possibilité au patient d'expirer si jamais il rencontre un problème. Plusieurs solutions ont été envisagées avant d'aboutir à celle retenue.

La première option envisagée est l'utilisation d'une électrovanne pilotée, maintenue fermée pendant les phases d'insufflation et de plateau, puis ouverte par commande de l'Arduino lors de la phase expiratoire. Cette approche offre un contrôle précis et une synchronisation parfaite avec le cycle ventilatoire. Cependant, une telle vanne ne permet pas au patient d'expirer en dehors de la phase expiratoire pré-définie, ce qui pourrait altérer la sécurité du patient en cas de problème. De plus, l'acquisition d'une électrovanne médicale compatible avec les connecteurs ISO du circuit représenterait un coût supplémentaire, ce qui va à l'encontre de l'objectif de bas coût du dispositif.

Une alternative serait l'utilisation d'une valve de surpression. Celle-ci serait réglée à une pression légèrement supérieure à celle de maintien, ce qui servirait également de sécurité intégrée, puisque la valve s'ouvrirait automatiquement si la turbine se mettait à insuffler une pression trop élevée. L'inconvénient de cette solution est que l'expiration se ferait contre une résistance, obligeant le patient à forcer pour expirer ce qui pourrait s'avérer une difficulté et un stress supplémentaire.

Il a également été envisagé de se servir d'un circuit respiratoire utilisé en hôpital, qui possède une valve expiratoire contrôlable intégrée comme le circuit ASTRAL de ResMed. Si cette solution est employée en clinique sur des ventilateurs médicaux professionnels, son adaptation au prototype s'est avérée complexe. Elle nécessite une source d'air de contrôle supplémentaire pour

piloter la valve, ce qui alourdit considérablement l'architecture du dispositif.

La dernière piste explorée s'inspire du principe de fonctionnement des appareils CPAP. Elle consiste à intégrer une fuite intentionnelle, réglable et permanente dans le circuit, juste avant le masque. Pendant les phases d'inspiration et de maintien en pression, la turbine génère un débit suffisant pour compenser cette fuite et maintenir la pression cible. Lorsque la turbine s'arrête à la fin de la deuxième phase, le patient expire librement à travers cet orifice. Cette solution est mécaniquement simple, elle ne nécessite aucun composant électronique supplémentaire, garantit une certaine sécurité au patient puisqu'il peut expirer par cette fuite en cas de problème. C'est cette solution qui a été retenue pour le prototype. Elle répond ainsi aux exigences fondamentales qui sont le maintien de la pression pendant le plateau, l'expiration libre en fin de cycle et la sécurité passive permanente pour le patient.

3.2.5 Capteur de pression différentielle

La mesure de pression est assurée par la combinaison de trois éléments : le capteur de débit proximal iFlow 200S, le capteur de pression différentielle MPX5010DP, et le convertisseur analogique-numérique ADS1115.

L'iFlow 200S est un capteur de débit proximal médical fonctionnant sur le principe de la différence de pression. Il intègre deux prises de pression latérales ainsi que des connecteurs ISO 22mm/15mm compatibles avec les autres éléments du circuit. Dans ce travail, il n'est pas utilisé pour sa fonction première de mesure de débit, mais comme pièce d'interface entre le circuit pneumatique et le capteur de pression MPX5010DP. Son utilisation a permis d'éviter la conception et la fabrication d'une pièce de raccordement spécifique, tout en garantissant une connexion étanche et standardisée avec le reste du circuit.

Le MPX5010DP est un capteur de pression différentielle avec une plage de mesure de 0 à 10 kPa (soit 0 à 1019 cmH_2O), largement suffisante pour les pressions thérapeutiques visées. Sa sortie est analogique et linéaire, variant de 0.2V à 4.7V sous une alimentation de 5V, selon la relation :

$$P\,(\text{kPa}) = \frac{\dfrac{V_{out}}{V_{cc}} - 0{,}04}{0{,}09} \tag{3.1}$$

L'ADC interne de l'Arduino Mega, d'une résolution de 10 bits, offrirait une résolution en pression d'environ 0.54 cmH_2O, ce qui serait insuffisant pour une régulation précise à des pressions cibles de l'ordre de 10 cmH_2O. C'est pourquoi un convertisseur analogique-numérique externe 16 bits, l'ADS1115, a été intercalé entre le capteur et le microcontrôleur. Ce module communique via le protocole Inter-Integrated Circuit (I^2C) à l'adresse 0x48 et offre une

résolution effective de 0.008 cmH_2O, soit une amélioration d'un facteur 68 par rapport à l'ADC interne.

3.2.6 Capteur de débit

Le capteur de débit SFM3300-D est positionné à la suite de la valve de fuite, à proximité du masque. Il mesure le débit d'air réellement insufflé au patient, après déduction de la fuite calibrée. Ce capteur communique avec le microcontrôleur via I^2C à l'adresse 0x40. Il présente une plage de mesure de $\pm$200 slm et une résolution de 0.008 L/min dans les conditions de ce travail.

Sa position implique qu'il est exposé à un flux bidirectionnel : lors de la phase inspiratoire, il mesure un débit positif correspondant à l'air insufflé ; lors de la phase expiratoire, il mesure un débit négatif correspondant à l'air expiré par le patient à travers la fuite. Cette caractéristique permet d'utiliser ce capteur comme indicateur de la phase respiratoire du patient.

3.2.7 Filtre HMEF

Le filtre échangeur de chaleur et d'humidité (HMEF) Hygrovent S est placé au plus proche du patient, entre le capteur de débit et le masque. Il assure une protection bactérienne et virale entre le circuit et les voies aériennes du patient, ce qui est indispensable dans un contexte clinique. Il agit également en tant qu'échangeur passif de chaleur et d'humidité, il capte lors de l'expiration la chaleur et l'humidité de l'air expiré par le patient, puis les restitue lors de l'inspiration suivante, améliorant ainsi le confort respiratoire.

3.2.8 Masque facial

Le masque facial constitue l'interface entre le circuit pneumatique et le patient. Il doit assurer une étanchéité suffisante pour permettre la montée en pression dans le circuit, tout en restant confortable pour le patient lors des séances d'entraînement. Le masque utilisé est un masque facial standard de ventilation non invasive, équipé d'un harnais de maintien ajustable. L'étanchéité du masque est un paramètre critique. Une fuite excessive autour du masque s'additionne à la fuite intentionnelle de la valve et peut compromettre la capacité de la turbine à atteindre et maintenir la pression cible.

3.2.9 Microcontrôleur

Le microcontrôleur Arduino Mega 2560 assure l'ensemble des fonctions de contrôle et d'acquisition du système. Il génère le signal PWM de commande de l'ESC de la turbine sur la pin 9, et communique avec les deux capteurs (SFM3300-D et ADS1115) via I^2C sur les pins 20 (SDA) et 21 (SCL). Sa fréquence d'horloge de 16 MHz et sa mémoire flash de 256 Ko sont largement suffisantes pour exécuter la boucle de régulation PID à une fréquence

de 20 Hz, tout en assurant la communication série avec un ordinateur pour l'enregistrement des données.

Le choix de la plateforme Arduino se justifie par sa simplicité de mise en œuvre, la richesse de son écosystème de bibliothèques et sa compatibilité directe avec les composants utilisés. Dans un contexte de prototypage académique, ces avantages priment sur les performances brutes d'une architecture plus complexe.

3.3 Caractérisation des composants

Avant d'assembler le circuit complet et de développer la logique de contrôle, une phase de caractérisation expérimentale des composants principaux a été réalisée. Cette étape est essentielle pour vérifier le bon fonctionnement de chacun des composants individuellement, s'assurer de la fiabilité des mesures ainsi que récolter certaines données nécessaires à la future régulation.

Les capteurs de débit et de pression ont d'abord été validés à l'arrêt, sans aucun flux, afin de déterminer leurs bruits de mesure respectifs (offsets). Ensuite la turbine, a été caractérisée en circuit ouvert et fermé pour analyser son comportement et définir la relation entre le signal PWM et les débits et pressions produits.

3.3.1 Caractérisation du capteur de débit SFM3300-D

Protocole expérimental

La validation du capteur est essentielle pour vérifier son bon fonctionnement, et déterminer la valeur renvoyée à débit nul. Le montage pour ce test est très simple puisque les deux orifices du capteur sont laissés ouverts, en s'assurant qu'il n'y ait pas de courant d'air dans la pièce pour avoir un débit réellement nul. Le capteur est connecté à l'Arduino Mega, avec une alimentation de 5V. Une séquence d'initialisation adaptée aux spécifications de la datasheet Sensirion [1] a été implémentée avant le démarrage des acquisitions.

Le sketch Arduino développé pour ce test lit la valeur brute du capteur toutes les 100 ms pendant une durée de 5 minutes. Les grandeurs enregistrées sont le temps écoulé en millisecondes, la valeur brute en LSB et le débit calculé en L/min selon la formule :

$$Q\,(\mathrm{L/min}) = \frac{raw - offset}{120} \tag{3.2}$$

où raw est la valeur brute lue, offset est l'offset nominal (32768 LSB) et 120 est le facteur d'échelle fourni par la datasheet.

1. Datasheet disponible en annexe B

Résultats

Le test a été lancé sur une durée de 5 minutes, soit 3000 lectures. Les résultats sont présentés dans le tableau 3.1.

TABLE 3.1 – Résultats de la validation du capteur de débit SFM3300-D à débit nul

Grandeur	Valeur mesurée
Offset mesuré (moyenne)	32767.22 LSB
Écart-type	±2 LSB
Débit équivalent minimum	−0.033 L/min
Débit équivalent maximum	+0.033 L/min

Interprétation

Les résultats obtenus confirment le bon fonctionnement du capteur. L'offset mesuré de 32767.22 LSB est pratiquement identique à la valeur nominale fournie par la datasheet (32768 LSB), ce qui confirme que le capteur est bien calibré. La valeur nominale de 32768 LSB sera donc conservée dans tous codes développés par la suite.

La résolution théorique du capteur, définie comme la variation de débit correspondant à 1 LSB, est de 0.008 L/min. Les mesures confirment ce résultat : sur l'ensemble des 3000 lectures, le capteur n'alterne qu'entre les valeurs 32764 et 32768 LSB, soit un écart de 4 LSB correspondant à ±0.033 L/min. Ce comportement est caractéristique d'un bruit électronique de mesure inférieur à 4 LSB, ce qui signifie que le capteur ne peut pas distinguer des variations de débit inférieures à 0.033 L/min dans ces conditions.

Cette résolution est largement suffisante pour l'application visée : les débits mis en jeu lors de l'entraînement respiratoire sont de l'ordre de plusieurs dizaines de L/min, ce qui place le bruit de fond à moins de 0.1% de la grandeur mesurée. Son impact sur la qualité des mesures est donc négligeable.

3.3.2 Validation du capteur de pression MPX5010DP

Protocole expérimental

Le capteur est connecté au canal A0 de l'ADS1115, lui-même relié à l'Arduino Mega. Les deux orifices du capteur sont laissés ouverts pour que la pression différentielle appliquée soit nulle. L'Arduino lit la tension de sortie du capteur toutes les 200 ms pendant une durée de 2 minutes via l'ADS1115. La pression est calculée à partir de la tension mesurée selon la relation :

$$P\,(\mathrm{cmH_2O}) = \frac{\dfrac{V_{out}}{V_{cc}} - 0{,}04}{0{,}09} \times 101{,}97 \tag{3.3}$$

où V_{out} est la tension mesurée par l'ADS1115 et V_{CC}=5 V est la tension d'alimentation du capteur. Le code complet de ce test est disponible en annexe E.

Résultats

Le capteur a été testé sur une durée de 2 minutes. Les résultats sont présentés dans le tableau 3.2.

TABLE 3.2 – Résultats de la validation du capteur de pression MPX5010DP à pression nulle

Grandeur	Valeur mesurée
Offset mesuré (moyenne)	0.520 $\mathrm{cmH_2O}$
Écart-type	$\pm$0.042 $\mathrm{cmH_2O}$
Pression minimum	0.397 $\mathrm{cmH_2O}$
Pression maximum	0.708 $\mathrm{cmH_2O}$
Plage de variation	0.312 $\mathrm{cmH_2O}$

Interprétation

Les résultats obtenus confirment que le capteur fonctionne correctement. On observe un écart-type de $\pm 0.042 cmH_2O$ qui correspond à un bruit de mesure

plutôt faible.

L'offset mesuré de 0.520 cmH_2O est dû à un léger écart entre la tension de sortie mesurée et la valeur théorique fournie dans la datasheet. En effet, à pression différentielle nulle, la datasheet indique une tension de sortie théorique de 0.200 V, or celle mesurée est de 0.2023 V. On a donc un écart de 0.0023 V qui correspond à notre écart de pression de 0.520 cmH_2O. Cet écart vaut 1.1% par rapport à la valeur théorique, ce qui rentre dans la plage de tolérance du capteur (±5%). Cet offset sera systématiquement soustrait à toutes les mesures de pression effectuées par la suite.

3.3.3 Caractérisation de la turbine - Débit VS PWM

La turbine WS7040 est le moteur du système, sa caractérisation est donc essentielle pour connaître les relations entre le signal PWM envoyé et les valeurs produites, que ce soit en pression ou en débit. La caractérisation a aussi permis d'évaluer sa dynamique de réponse, indispensable pour dimensionner le futur régulateur PID. La première mesure que l'on va évaluer est le débit par rapport aux signaux PWM.

Protocole expérimental

Le montage se fait ici en circuit ouvert. Le ventilateur est connecté au capteur de débit par une pièce sur mesure imprimée en 3D. Le deuxième orifice du capteur de débit est laissé vide, l'air peut donc sortir librement. Le signal PWM a été augmenté par paliers successifs de 0 à 255 (soit 0 à 100%), avec un délai de stabilisation de 4 secondes à chaque palier. À chaque palier, 30 mesures ont été prises, puis moyennées afin de réduire l'influence du bruit de mesure. Le test a été réalisé en une seule série. Le montage et le code complet de ce test sont disponibles en annexe F.

Résultats

Les résultats sont présentés à la figure 3.3 et résumés dans le tableau 3.3.

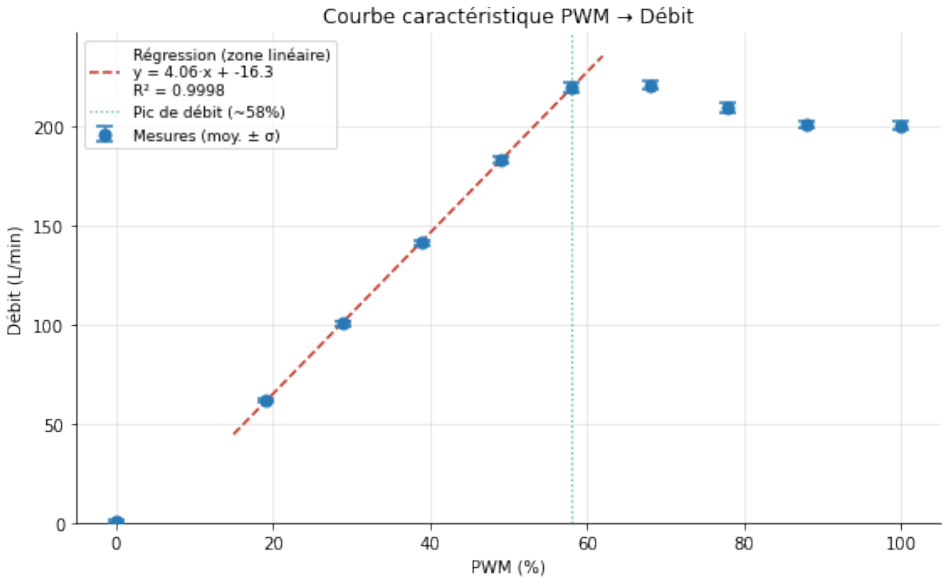


FIGURE 3.3 – Caractérisation débit VS PWM

TABLE 3.3 – Synthèse de la caractérisation débit vs PWM — Turbine WS7040

Grandeur	Valeur
Seuil de démarrage	PWM = 31 (12%)
Zone linéaire	PWM 19% à 58%
Débit maximum	220.5 L/min à PWM = 68%
Régression linéaire (zone linéaire)	$Q = 4.06 \times PWM\% - 16.27$
Débit à pleine puissance (PWM = 100%)	200.4 L/min

Interprétation

On observe 3 phases distinctes. La première correspond à l'absence de débit lorsque le signal PWM n'est pas suffisant pour lancer la turbine. Donc même en présence d'un signal PWM, la turbine ne se met pas en route tant qu'elle n'a pas atteint son seuil de démarrage qui a été identifié à PWM = 31 (12%). Ce seuil est un paramètre important à prendre en compte lors de l'implémentation du régulateur PID.

La deuxième phase est la relation linéaire entre le signal PWM et le débit de sortie. Cette phase se situe dans la zone de PWM allant de 19%, où le débit est de 61.9 L/min à 58% où le débit est de 219.6 L/min. Cette régression linéaire suit l'équation :

$$Q, (\text{L/min}) = 4.06 \times PWM\% - 16.27 \quad (R^2 = 0.9998) \tag{3.4}$$

La troisième phase concerne les signaux PWM supérieurs à 58%. Le débit se stabilise autour de 220 L/min puis se met à diminuer pour atteindre 200.4 L/min à puissance maximale (PWM = 100%). Ce phénomène de saturation puis de légère chute pourrait s'expliquer par une limitation en courant de l'ESC aux vitesses élevées, ou par des effets aérodynamiques propres à la turbine. Des tests supplémentaires seraient nécessaires pour confirmer ces hypothèses.

La phase 2 est donc la plus intéressante pour le prototype car elle permet d'obtenir les débits souhaités en modulant le signal PWM envoyé.

3.3.4 Caractérisation de la turbine - Pression VS PWM

Protocole expérimental

Ce test vise à caractériser la relation entre le signal de commande PWM et la pression générée par la turbine dans trois configurations différentes de circuit, afin d'évaluer l'influence de la valve de fuite sur le comportement du système. Le montage est le suivant : la turbine est connectée via une tubulure souple au capteur de débit proximal iFlow 200S. Les deux prises de pression latérales de l'iFlow 200S sont utilisées différemment : l'une est connectée à l'orifice P1 du capteur MPX5010DP afin de mesurer la pression dans le circuit, tandis que l'autre est obturée. L'orifice P2 du MPX5010DP est laissé ouvert à l'atmosphère, de sorte que la mesure correspond bien à une pression différentielle par rapport à la pression ambiante. La valve de fuite réglable est connectée à l'iFlow 200S et obstruée par la main de l'autre côté. Les trois configurations testées sont :

— **Circuit fermé** : la valve de fuite est fermée, aucun air ne s'échappe. Cette configuration permet de mesurer la pression statique maximale que la turbine peut générer à chaque palier de PWM.

— **Fuite semi-ouverte** : la valve de fuite réglable est semi-ouverte, simulant une résistance intermédiaire au flux d'air.
— **Fuite complètement ouverte** : la valve de fuite est ouverte au maximum, correspondant aux conditions d'utilisation clinique du dispositif.

Le montage et le code complet de ce test sont disponibles en annexe G.

Résultats

Les résultats sont présentés à la figure 3.4 et résumés dans le tableau 3.4.

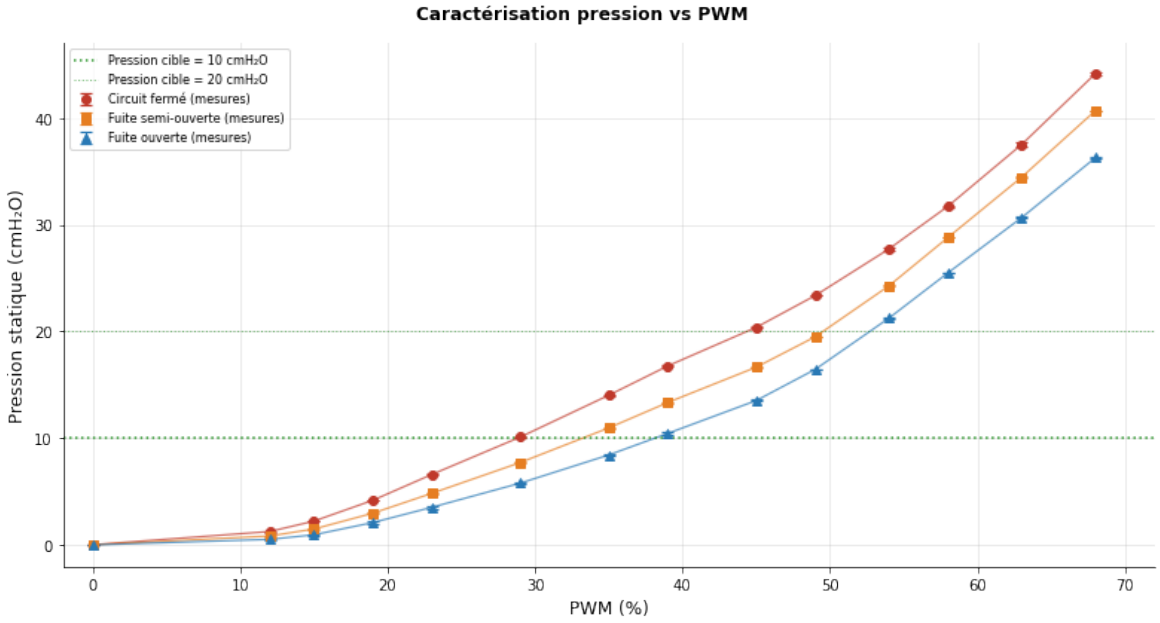


FIGURE 3.4 – Caractérisation de la pression par rapport au signal PWM - Comparaison selon l'ouverture de la fuite

TABLE 3.4 – Synthèse de la caractérisation pression vs PWM selon la configuration du circuit

Configuration	Circuit fermé	Fuite semi-ouverte	Fuite ouverte
Pression à PWM = 15%	19.7 cmH_2O	12.4 cmH_2O	6.3 cmH_2O
Pression à PWM = 29%	96.4 cmH_2O	73.7 cmH_2O	50.7 cmH_2O
Pression à PWM = 68%	436.2 cmH_2O	402.1 cmH_2O	345.9 cmH_2O
PWM pour 10 cmH_2O	∼13%	∼14%	∼16%
PWM pour 20 cmH_2O	∼15%	∼16%	∼19%
Répétabilité (écart-type moyen)	8.2 cmH_2O	3.1 cmH_2O	1.8 cmH_2O

Interprétation

La figure 3.4 montre que l'ouverture de la valve de fuite réduit significativement la pression générée à PWM équivalent. À titre d'exemple, à PWM = 29%, la pression passe de 96.4 cmH_2O en circuit fermé à 73.7 cmH_2O avec la fuite semi-ouverte, et à 50.7 cmH_2O avec la fuite complètement ouverte. Ce résultat est physiquement cohérent : la fuite constitue une voie de sortie pour l'air, ce qui réduit la pression d'équilibre dans le circuit.

Pour les pressions thérapeutiques visées (10 à 20 cmH_2O), les trois configurations nécessitent des signaux PWM très faibles, compris entre 13% et 19% selon l'ouverture de la valve. Cela confirme que la turbine dispose d'une large marge de puissance par rapport aux besoins cliniques.

On note également que la répétabilité s'améliore lorsque la fuite est ouverte : l'écart-type moyen passe de 8.2 cmH_2O en circuit fermé à 1.8 cmH_2O avec la fuite ouverte. Cela s'explique par le fait qu'en circuit fermé, les légères variations de débit de la turbine se traduisent directement en variations de pression, alors qu'en présence d'une fuite, ces variations sont en partie absorbées par le flux d'air sortant. La présence de la valve de fuite contribue donc non seulement à la sécurité du dispositif, mais aussi à la stabilité de la mesure de pression.

Dans le contexte du dispositif final, c'est la configuration fuite ouverte qui correspond aux conditions réelles d'utilisation. Les résultats confirment que pour atteindre et maintenir une pression cible de 10 à 20 cmH_2O, la turbine doit être commandée avec un signal PWM de l'ordre de 16 à 19%, ce qui servira de point de départ pour le réglage du régulateur PID.

3.3.5 Caractérisation de la turbine - Temps de réponse

Protocole expérimental

Le temps de réponse de la turbine a été mesuré par des essais en échelon : à partir d'un état de repos (PWM = 0), un signal PWM constant est appliqué instantanément, et la réponse en pression est enregistrée toutes les 50 ms pendant 5 secondes. Trois amplitudes d'échelon ont été testées : 0 à 75, 0 à 125 et 0 à 175, correspondant respectivement à des pressions finales d'environ 100, 235 et 450 cmH_2O en circuit fermé. Chaque échelon a été répété 3 fois afin d'évaluer la répétabilité. Le temps de montée est défini comme l'intervalle entre le moment où la pression atteint 10% de sa valeur finale (t_{10}) et celui où elle atteint 90% (t_{90}). Le montage et le code complet de ce test sont disponible en annexe H.

Résultats

Les résultats sont présentés à la figure 3.5 et résumés dans le tableau 3.5.

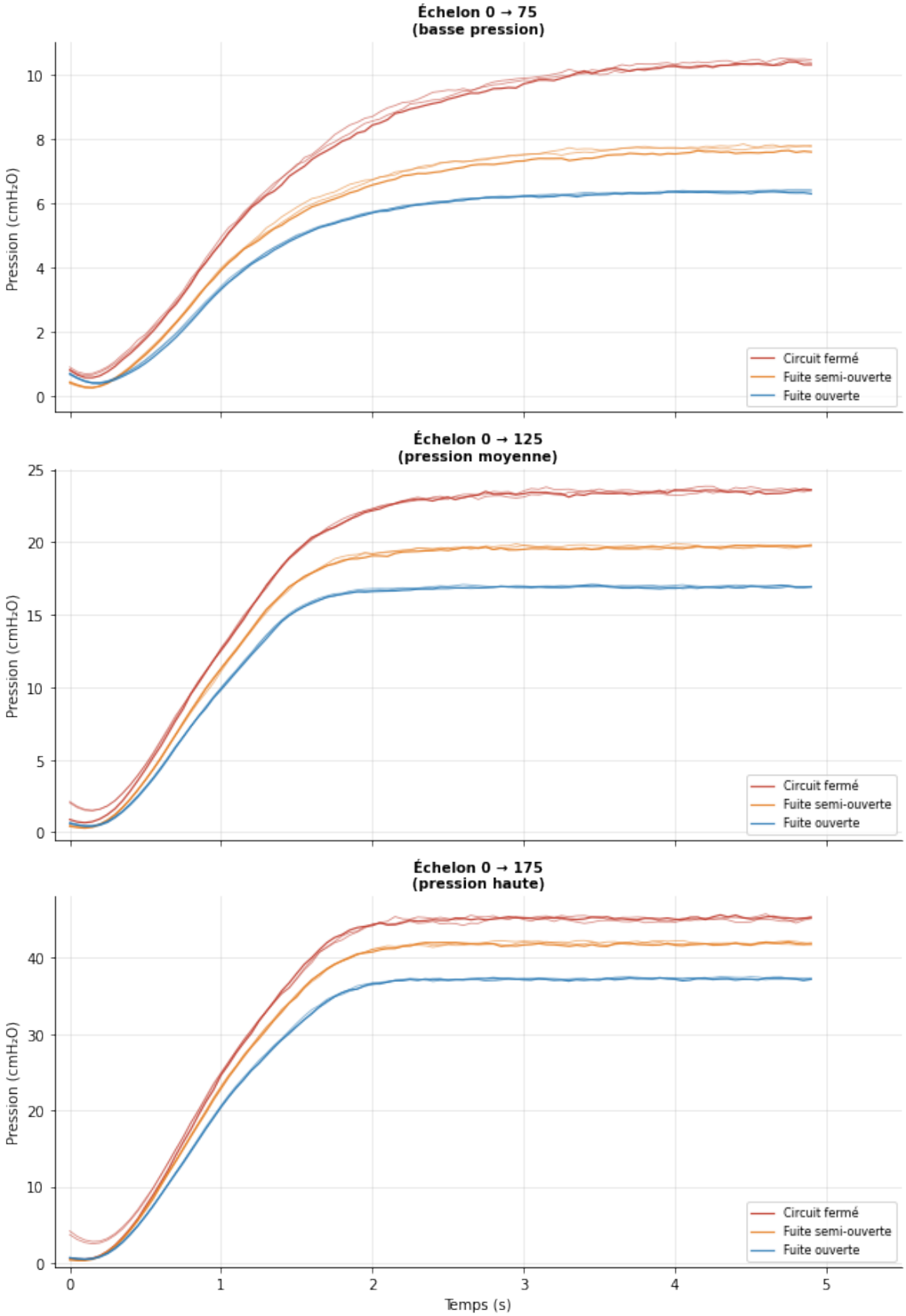


FIGURE 3.5 – Temps de montée en pression de la turbine - Comparaison selon l'ouverture de la fuite

TABLE 3.5 – Temps de montée en pression de la turbine WS7040 selon la configuration de la valve de fuite (moyenne sur 3 répétitions)

Échelon	Configuration	P_{fin} (cmH_2O)	t_{10} (s)	t_{90} (s)	$t_{montée}$ (s)
0 → 75	Circuit fermé	10.39	0.334	2.518	2.184
	Fuite semi-ouverte	7.70	0.401	2.219	1.818
	Fuite ouverte	6.37	0.000	2.034	2.034
0 → 125	Circuit fermé	23.57	0.366	1.784	1.417
	Fuite semi-ouverte	19.72	0.400	1.601	1.201
	Fuite ouverte	16.93	0.400	1.501	1.101
0 → 175	Circuit fermé	45.08	0.383	1.651	1.268
	Fuite semi-ouverte	41.87	0.400	1.668	1.268
	Fuite ouverte	37.25	0.417	1.651	1.234

Interprétation

Les résultats mettent en évidence plusieurs observations.

Premièrement, le temps de montée tend à diminuer lorsque l'amplitude de l'échelon augmente, mais cette tendance n'est pas strictement monotone sur l'ensemble de la plage testée : elle est marquée entre les échelons 0 → 75 et 0 → 125, mais le temps de montée se stabilise ensuite autour de 1.1 à 1.3 s entre les échelons 0 → 125 et 0 → 175, quelle que soit la configuration de valve. Ce comportement est cohérent avec le fait qu'à une vitesse de rotation plus élevée, la turbine développe un couple plus important, ce qui lui permet d'accélérer plus rapidement, jusqu'à ce que d'autres facteurs deviennent limitants.

Deuxièmement, l'effet de la configuration de la valve de fuite dépend de l'amplitude de l'échelon. À basse pression (0 → 75), les trois configurations présentent des temps de montée différenciés (1.8 à 2.2 s), la fuite semi-ouverte étant la plus rapide. À pression moyenne et haute (0 → 125 et 0 → 175), les temps de montée des trois configurations convergent vers des valeurs proches (1.1 à 1.3 s), suggérant qu'à haut régime, la dynamique propre de la turbine domine sur l'effet du débit de fuite.

Troisièmement, la répétabilité reste excellente pour l'ensemble des configurations et des amplitudes testées. Les trois répétitions donnent des temps de montée très proches, avec un écart maximal inférieur à 0.1 s.

Dans le contexte du dispositif, pour couvrir la plage thérapeutique 10-20 cmH_2O, un temps de montée de l'ordre de 1 à 2 secondes est à anticiper selon la pression cible, ce dont le régulateur PID devra tenir compte.

Chapitre 4

Régulation en pression

Une fois l'architecture du dispositif définie et les différents composants caractérisés, il est possible d'établir la stratégie de contrôle pour piloter le système. Cette stratégie vise un objectif principal, maintenir une pression cible définie dans le circuit pendant une durée déterminée. Cette régulation est assurée par un algorithme de contrôle en boucle fermée, dont le principe, l'implémentation et les performances sont détaillés dans ce chapitre.

4.1 Principe du régulateur PID

4.1.1 Justification du choix

La régulation de pression dans le circuit nécessite un mécanisme de contrôle capable d'ajuster en temps réel le signal envoyé à la turbine, de façon à maintenir la pression mesurée au plus proche de la pression cible. Plusieurs stratégies de contrôle peuvent s'envisager.

Une première approche serait d'utiliser le système en boucle ouverte. A partir de la courbe de caractérisation pression par rapport au signal PWM établie au chapitre précédent, il serait possible de déterminer directement le signal PWM correspondant à la pression cible et de l'envoyer à la turbine. Cette approche est simple à implémenter mais présente des limitations majeures. La plus contraignante est que cette méthode ne prend pas en compte les perturbations extérieures susceptibles de faire dériver la pression réelle comme les variations du débit de fuite, les fuites autour du masque, l'échauffement de la turbine en cours de fonctionnement, et les fluctuations de la tension d'alimentation. De plus, elle ignore complètement la compliance pulmonaire du patient. La compliance pulmonaire est la variation de volume pulmonaire produite par une variation de pression. Ce paramètre varie significativement d'un patient à l'autre et peut évoluer chez un même patient au fil des séances. Un signal PWM fixé pour un patient donné produirait donc des pressions très différentes chez un autre patient, rendant l'approche en boucle ouverte

cliniquement inadaptée.

Un régulateur en boucle fermée est donc indispensable. En mesurant en permanence la pression réelle dans le circuit et en ajustant le signal de commande en conséquence, le régulateur s'adapte automatiquement aux caractéristiques des patients et aux perturbations du circuit. Parmi les régulateurs classiques, le régulateur PID est le plus largement utilisé en automatique industrielle et médicale.[24] Il présente plusieurs avantages pour ce projet, il est simple à implémenter sur le microcontrôleur, robuste face aux perturbations, et ne nécessite pas de modèle mathématique précis du système. Ce type de régulateur a d'ailleurs déjà été utilisé dans des respirateurs open source.[23] C'est donc le régulateur PID qui a été retenu pour contrôler la pression dans le circuit.

4.1.2 Rappel théorique

Un régulateur PID calcule en permanence le signal de commande $u(t)$ à partir de l'erreur $e(t)$.[25] L'erreur se définit comme la différence entre la cible et la valeur mesurée. Pour ce dispositif ce sera la différence entre la pression cible et la pression mesurée dans le circuit :

$$e(t) = P_{cible} - P_{mesurée}(t) \tag{4.1}$$

Le signal de commande est la somme de trois termes :

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau), d\tau + K_d \cdot \frac{de(t)}{dt} \tag{4.2}$$

où $K_p, K_i et K_d$ sont respectivement les gains proportionnel, intégral et dérivé. Chacun de ces trois termes joue un rôle dans le comportement du régulateur.[25]

Le **terme proportionnel** ($K_p \cdot e(t)$) applique une correction proportionnelle à l'erreur instantanée cible.[25] Plus l'erreur est grande, plus la correction est importante. Un gain K_p élevé accélère la réponse mais peut provoquer des oscillations autour de la cible.

Le **terme intégral** ($K_i \int_0^t e(\tau), d\tau$) accumule l'erreur au cours du temps. Il permet d'éliminer l'erreur statique résiduelle que le terme proportionnel ne peut pas corriger seul. Un gain K_i trop élevé peut cependant provoquer un phénomène dit de *windup*, où la valeur accumulée devient excessive et déstabilise le système.[25]

Le **terme dérivé** ($K_d \cdot \frac{de(t)}{dt}$) réagit à la vitesse de variation de l'erreur. Il anticipe l'évolution du système et amortit les oscillations.[25] En revanche, il est très sensible au bruit de mesure. Une variation rapide du bruit du capteur est interprétée comme une variation rapide de l'erreur, ce qui peut générer des corrections non désirées. Pour cette raison, le terme dérivé a été désactivé ($K_0 = 0$) dans l'implémentation finale, le filtre passe-bas appliqué à la mesure

de pression étant insuffisant pour atténuer suffisamment le bruit haute fréquence.

Dans un système numérique comme l'Arduino, le PID est implémenté en temps discret avec une période d'échantillonnage T_e.[26] Les équations deviennent :

$$e_k = P_{cible} - P_k \tag{4.3}$$

$$u_k = K_p \cdot e_k + K_i \cdot T_e \sum_{j=0}^{k} e_j + K_d \cdot \frac{e_k - e_{k-1}}{T_e} \tag{4.4}$$

où l'indice k désigne la valeur à l'instant $k \cdot T_e$.

La période d'échantillonnage retenue pour ce projet est $T_e = 50ms$. Le choix de cette valeur est guidé par la règle empirique classique en contrôle numérique, qui stipule que la période d'échantillonnage doit être au moins 10 fois inférieure au temps de montée du système.[25] Le temps de montée de la turbine mesuré au chapitre précédent étant d'environ 2 secondes, la période maximale admissible serait de 200 ms. Une valeur de 50 ms a été retenue, offrant un rapport de sécurité de 4 par rapport à cette limite.

4.2 Implémentation sur Arduino

4.2.1 Architecture générale

Le régulateur PID fonctionne avec une boucle de contrôle s'exécutant toutes les 50 ms sur l'Arduino Mega. À chaque itération, le microcontrôleur effectue les opérations suivantes dans l'ordre :

1. Lecture de la pression mesurée
2. Calcul de l'erreur $e_k = P_{cible} - P_{mesurée}$
3. Calcul des trois termes P, I et D
4. Calcul du signal de commande u_k
5. Conversion en signal PWM et envoi à la turbine

4.2.2 Filtre passe-bas

La mesure brute du capteur MPX5010DP présente un bruit de mesure résiduel qui, amplifié par le terme dérivé du PID, peut générer des corrections erratiques. Un filtre passe-bas exponentiel est donc appliqué à chaque lecture de pression, selon la relation :

$$P_{filtrée,k} = \alpha \cdot P_{brute,k} + (1 - \alpha) \cdot P_{filtrée,k-1} \tag{4.5}$$

où $\alpha \in [0, 1]$ est le coefficient de lissage. Une valeur α proche de 1 donne peu de filtrage (réponse rapide mais bruitée), tandis qu'une valeur proche de 0 donne un filtrage fort (signal lisse mais réponse lente).[27] Le coefficient de lissage $\alpha = 0.2$ a été déterminé empiriquement. Cette valeur offre un bon compromis entre réactivité et stabilité.

4.2.3 Anti-windup

Pour éviter l'emballement intégral, l'intégrale est bornée entre deux valeurs limites $[-I_{max}, +I_{max}]$. Cette contrainte, appelée anti-windup, empêche l'intégrale d'accumuler une valeur excessive lorsque le système est saturé (par exemple lors du démarrage, quand la pression est très éloignée de la cible).[25] La limite anti-windup $I_{max} = 100$ a été fixée de façon à ce que le terme intégral ne puisse pas à lui seul saturer le signal de commande ($PMW_{max} = 175$), tout en lui laissant une marge d'action suffisante pour éliminer l'erreur statique.

4.2.4 Pré-charge de l'intégrale au démarrage

Lors du démarrage du régulateur, la pression dans le circuit est nulle et l'erreur est maximale. Si l'intégrale part de zéro, le terme intégral met plusieurs secondes à accumuler une valeur suffisante pour que la turbine atteigne un régime utile. Pour accélérer la montée en pression initiale, l'intégrale est initialisée à une valeur non nulle lors du démarrage, ce qui force le régulateur à envoyer immédiatement un signal PWM significatif à la turbine. Cette valeur a été déterminée empiriquement, des valeurs trop faibles ne permettaient pas un démarrage rapide de la turbine, tandis que des valeurs trop élevées provoquaient un dépassement trop important de la cible lors de la phase de montée en pression. Une valeur de 60 a été retenue pour l'ensemble des tests en laboratoire. Si cette valeur offre un bon compromis général, elle n'est pas nécessairement optimale pour chaque pression cible. Ce point est discuté plus en détail dans la section 4.3.3.

4.2.5 Seuil minimum de PWM

La turbine WS7040 a un seuil de démarrage à PWM = 31 (12%), en dessous duquel elle ne génère aucun débit. Pour éviter que le régulateur n'envoie un signal trop faible qui ne ferait pas tourner la turbine, une contrainte est appliquée. Si le signal calculé est non nul mais inférieur à ce seuil, il est automatiquement augmenté à la valeur minimale de démarrage.

Le code complet du régulateur est disponible en annexe I.

4.3 Réglage des paramètres du PID et résultats

4.3.1 Méthode de réglage

Le réglage des paramètres du PID s'est fait empiriquement, par essais successifs. Cette approche est justifiée par la difficulté à modéliser le comportement du système. Le débit de la fuite, les pertes de charge du circuit, et par la suite la compliance pulmonaire sont difficiles à caractériser. L'approche adoptée s'inspire de la **méthode d'amortissement au quart d'amplitude**[25], adaptée aux contraintes spécifiques du système.

1. Tout d'abord on initialise K_i et K_d à 0, et le gain proportionnel K_p est augmenté progressivement jusqu'à obtenir une montée en pression rapide sans oscillations excessives.
2. Ensuite, le gain dérivé K_d est normalement augmenté progressivement jusqu'à réduire l'overshoot à un niveau acceptable. Cependant, dans le cas présent, l'activation du terme dérivé s'est avérée contre-productive. Sa sensibilité aux variations rapides de l'erreur amplifiait le bruit de mesure du capteur de pression, générant des corrections instables sur la turbine malgré le filtre passe-bas appliqué. Le terme dérivé a donc été désactivé ($K_d = 0$).
3. Enfin on augmente progressivement le gain intégral K_i jusqu'à éliminer l'erreur statique résiduelle et obtenir un temps de convergence acceptable.

4.3.2 Évolution du réglage

Il a fallu plusieurs itérations avant d'obtenir les paramètres optimaux. L'évolution des paramètres et le comportement associé observé sont résumés dans le tableau 4.1.

TABLE 4.1 – Évolution des paramètres PID et comportements observés

Version	Kp	Ki	Kd	Étape	Comportement observé
V1	8.0	0.0	0.0	Réglage K_p	Montée rapide mais oscillations importantes autour de la consigne.
V2	3.0	0.0	0.0	Réglage K_p	Oscillations éliminées. Erreur statique résiduelle persistante.
V3	3.0	0.0	1.0	Test K_d	Corrections instables dues à l'amplification du bruit de mesure. K_d diminué progressivement jusqu'à être désactivé.
V4	3.0	0.8	0.0	Réglage K_i	Erreur statique réduite. Temps de convergence encore élevé.
V5	2.0	1.3	0.0	Réglage K_i + K_p	Montée rapide, pas d'overshoot, erreur statique nulle. Paramètres retenus.

Note : Ce tableau présente une synthèse des étapes clés du réglage. En pratique, les paramètres ont été modifiés de façon progressive par pas de 0.1, ce qui a nécessité un nombre d'itérations plus important que les 5 versions présentées ici.

4.3.3 Résultats finaux

Les paramètres sélectionnés sont $K_p = 2.0, K_i = 1.3, K_d = 0$. Ils ont été validés sur différentes pressions cibles couvrant la plage thérapeutique visée : 5, 10, 15, 16, 17, 18, 19 et 20 cmH_2O. Pour chacune de ces pressions, une session de 3 cycles a été testée, comprenant l'inspiration, 20 secondes de plateau de pression et 5 secondes d'expiration. Les résultats sont présentés aux figures 4.1 et 4.2. Ils sont également résumés dans le tableau 4.2. Les figures supplémentaires sont disponibles dans l'annexe K.

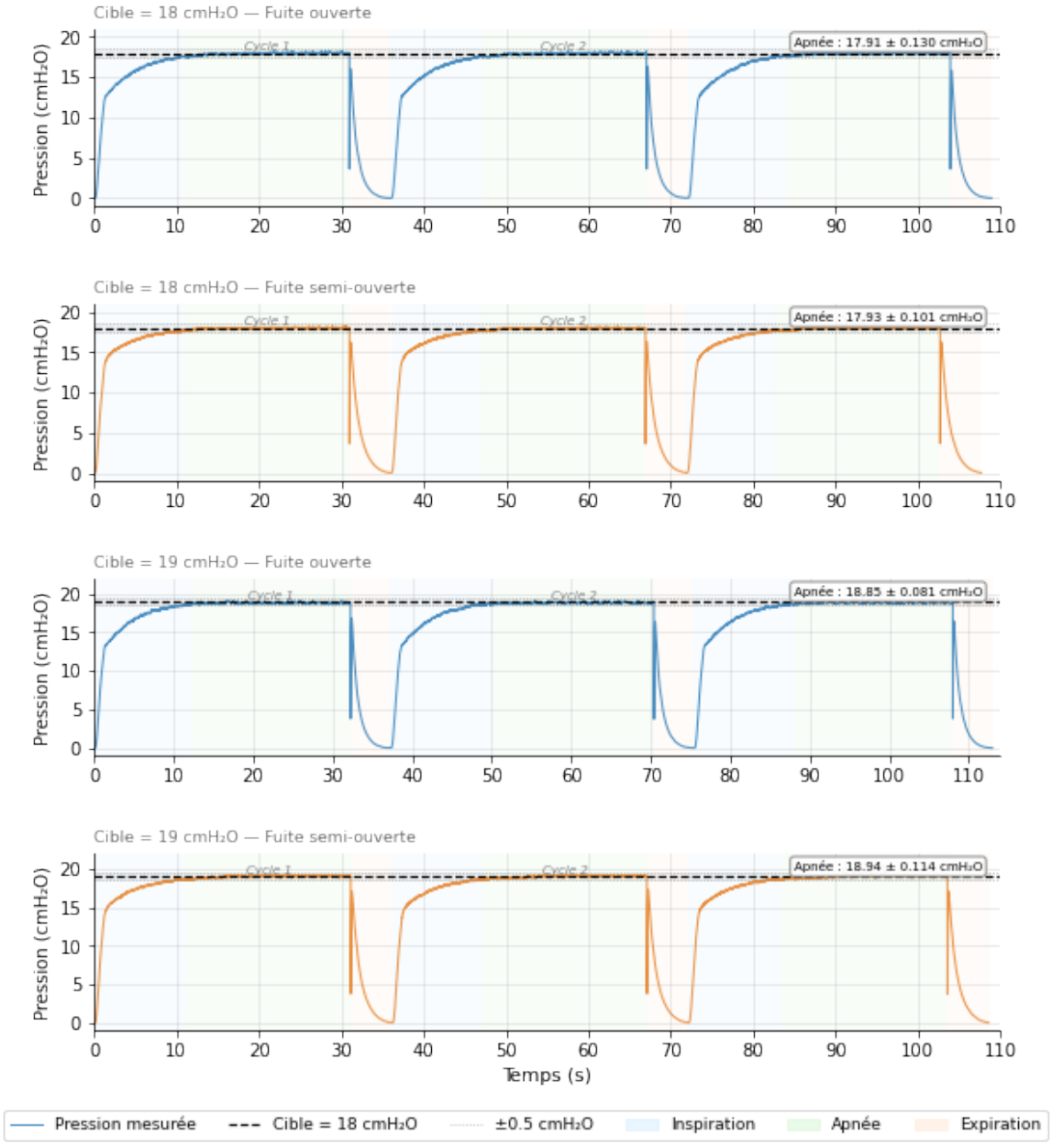


FIGURE 4.1 – Régulation PID en laboratoire - fuite ouverte et fuite semi-ouverte - 18 et 19 cmH_2O

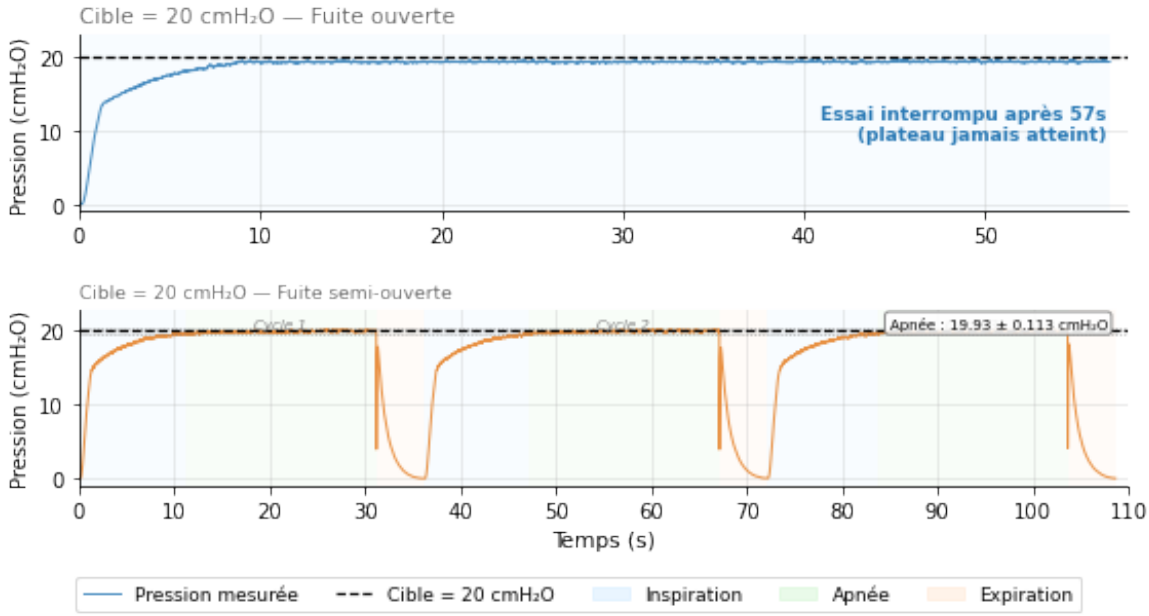


FIGURE 4.2 – Régulation PID en laboratoire - fuite ouverte et fuite semi-ouverte - 20 cmH_2O

TABLE 4.2 – Précision et erreur du régulateur PID en apnée pour les deux configurations de valve de fuite (moyenne sur 3 cycles, apnée 20 s).

Pression cible (cmH_2O)	Précision fuite ouverte ($\pm$ cmH_2O)	Erreur fuite ouverte (cmH_2O)	Précision fuite semi-ouverte ($\pm$ cmH_2O)	Erreur fuite semi-ouverte (cmH_2O)
5	$\pm$0.113	$-$0.108	$\pm$0.118	+0.100
10	$\pm$0.116	$-$0.097	$\pm$0.156	+0.168
15	$\pm$0.109	$-$0.086	$\pm$0.109	$-$0.069
16	$\pm$0.111	$-$0.062	$\pm$0.112	$-$0.077
17	$\pm$0.121	$-$0.078	$\pm$0.111	$-$0.074
18	$\pm$0.130	$-$0.087	$\pm$0.101	$-$0.069
19	$\pm$0.081	$-$0.146	$\pm$0.114	$-$0.064
20	**N/A** — plateau non atteint		$\pm$0.113	$-$0.069

Les résultats que présente le régulateur sont très positifs. La précision en apnée est excellente, inférieure à $\pm 0.16 cmH_2O$ sur l'ensemble des pressions cible

testées. La répétabilité est également excellente, le régulateur atteint la pression cible de manière similaire lors des trois cycles, signe que le comportement du régulateur est reproductible. L'erreur moyenne reste quant à elle sous un seuil assez bas, de $0.17 cmH_2O$ en valeur absolue. On observe une tendance à une légère sous-régulation (erreur négative) à partir de 15 cmH_2O, cohérente entre les deux configurations de valve. Les paramètres du PID et la précharge intégrale étant identiques pour l'ensemble des pressions testées, il est probable qu'un jeu de paramètres unique ne soit pas également optimal sur toute la plage de fonctionnement, ce qui pourrait expliquer les petites variations d'erreur observées d'une pression cible à l'autre.

Limite de la fuite ouverte

Les tests ont mis en évidence une limitation physique du système, la turbine n'arrive pas à atteindre une pression cible de 20 cmH_2O lorsque la fuite est entièrement ouverte. Elle arrive à atteindre une cible de 19 cmH_2O mais au delà, le débit généré par la turbine n'est pas suffisant pour compenser le débit de fuite et atteindre la pression souhaitée. Cette limitation se justifie par le fait que le modèle de la turbine ait été sélectionné avant que la valve de fuite ne soit adoptée comme solution pour la phase expiratoire.

Cette limitation n'est toutefois pas très problématique car il suffit de refermer légèrement la valve de fuite pour que la turbine atteigne sa cible. Réduire l'ouverture de la valve implique une réduction du débit de fuite et donc demande un débit moins important à la turbine pour une même cible. Cela a été prouvé lors des tests en configuration semi-ouverte, où la cible de 20 cmH_2O a été atteinte. La position de la valve constitue ainsi un paramètre de réglage clinique du dispositif, ajustable par le clinicien en fonction de la pression prescrite.

De plus, lors des tests sur poumons artificiels (présentés à la section 4.4), le dispositif a atteint sans problème la cible de 20 cmH_2O avec la fuite ouverte. Cela suggère que la présence d'une charge mécanique en bout de circuit, telle qu'un poumon artificiel, modifie le comportement du système et permet à la turbine d'atteindre des pressions plus élevées qu'en circuit rigide.

Temps de montée en pression

Le principal point faible de ces résultats de laboratoire concerne le temps de montée en pression, qui se situe entre 10 et 12 secondes. Cela peut en partie s'expliquer par la valeur de la pré-charge de l'intégrale qui est identique pour toutes les pressions cibles. Ce n'est pas optimal car les pressions cibles élevées nécessiteraient une valeur plus élevée qui démarrerait la turbine plus rapidement.

Des tentatives d'optimisation de cette pré-charge ont été testées au laboratoire, mais il s'est avéré que la relation entre la valeur de la pré-charge et la pression cible n'était pas linéaire. Des recherches et tests supplémentaires sont

nécessaires afin de trouver la valeur optimale pour chaque pression cible. Des résultats concluants se sont quand même révélés, par exemple une pré-charge de 150 testée à 20 cmH_2O a réduit le temps de montée à 2.5 secondes, tout en intégrant un léger overshoot de 0.6 cmH_2O.

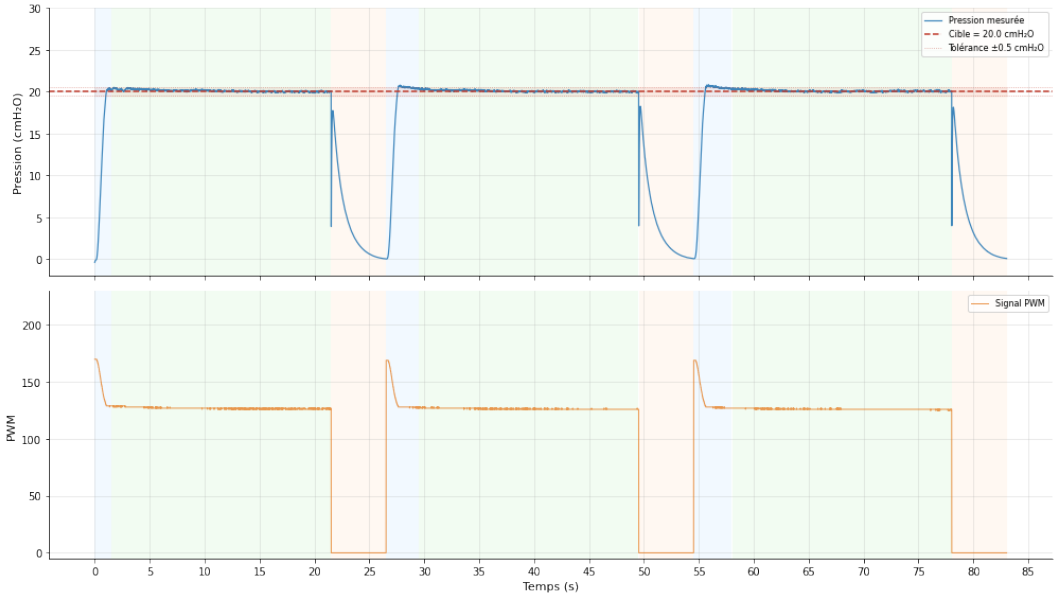


FIGURE 4.3 – Cycles avec pré-charge de 150, fuite semi-ouverte et une cible de 20 cmH_2O

Ce temps élevé de montée en pression n'est cependant pas trop problématique car il a été réglé lors des tests sur poumons artificiels (section 4.4). Les temps de montée en conditions réelles sont autour de 2-3 secondes ce qui est plutôt acceptable, et bien meilleur que les performances en laboratoire.

4.4 Validation sur poumon artificiel

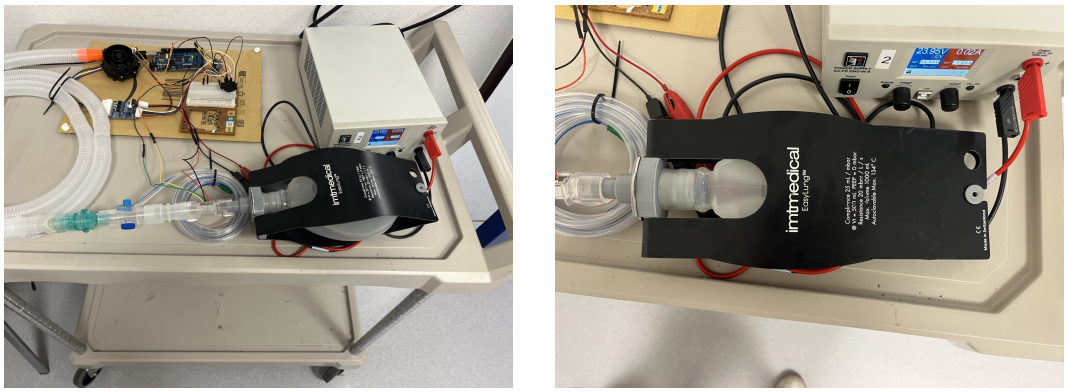
4.4.1 Contexte et dispositif de test

Les tests en laboratoire ont confirmé le bon fonctionnement du régulateur. Mais un poumon est bien différent d'un simple circuit fermé. Les voies aériennes sont caractérisées par une compliance qu'il faut impérativement prendre en compte. Une phase de validation sur poumon artificiel a donc été réalisée dans les locaux du service de radiothérapie de l'hôpital.

Le poumon artificiel utilisé est le modèle *EasyLung* de la société IMT Medical. Ses caractéristiques principales sont les suivantes : une compliance de 25 mL/mbar à un volume courant de 500 mL et une pression expiratoire positive nulle (PEEP = 0 mbar), une résistance de 20 mbar/(L/s) et un volume maximal de 1000 mL. Ce dispositif est couramment utilisé pour l'évaluation et

la certification des respirateurs médicaux[28], ce qui en fait une référence fiable pour valider les performances du prototype dans des conditions contrôlées et reproductibles, avant de recourir à des tests sur sujets humains. Le montage utilisé pour ces tests est le même que lors des essais en laboratoire, et illustré à la figure 4.4.

Afin de valider les mesures fournies par les capteurs du prototype, le montage expérimental a également intégré un manomètre de référence mis à disposition par l'hôpital. Cet instrument, utilisé en routine clinique pour la vérification des respirateurs médicaux, a permis de comparer en temps réel les valeurs de pression mesurées par le MPX5010DP avec une mesure de référence indépendante. Une comparaison visuelle des deux affichages au cours des essais a montré des valeurs sensiblement concordantes, validant qualitativement la fiabilité de la chaîne de mesure de pression du prototype.



(a) Circuit expérimental (b) Poumon artificiel EasyLung

FIGURE 4.4 – Configuration pour les tests sur poumon artificiel

4.4.2 Mise en évidence des oscillations de pression

Lors du premier essai sur le poumon artificiel, des oscillations de pression importantes ont été observées lors de la phase du plateau de pression. Ces oscillations sont clairement visibles sur la figure 4.5, qui présente l'évolution temporelle de la pression mesurée et du signal PWM lors de ce premier essai. La pression dans le circuit n'était pas maintenue à une valeur constante, mais oscillait périodiquement avec une amplitude de $\pm 2.3 cmH_2O$ et une période d'environ 0.42 secondes. Ces oscillations étaient également visibles sur le poumon artificiel. Elles se manifestaient par des mouvements rythmiques de la membrane du *EasyLung*, similaires à des mini-cycles respiratoires.

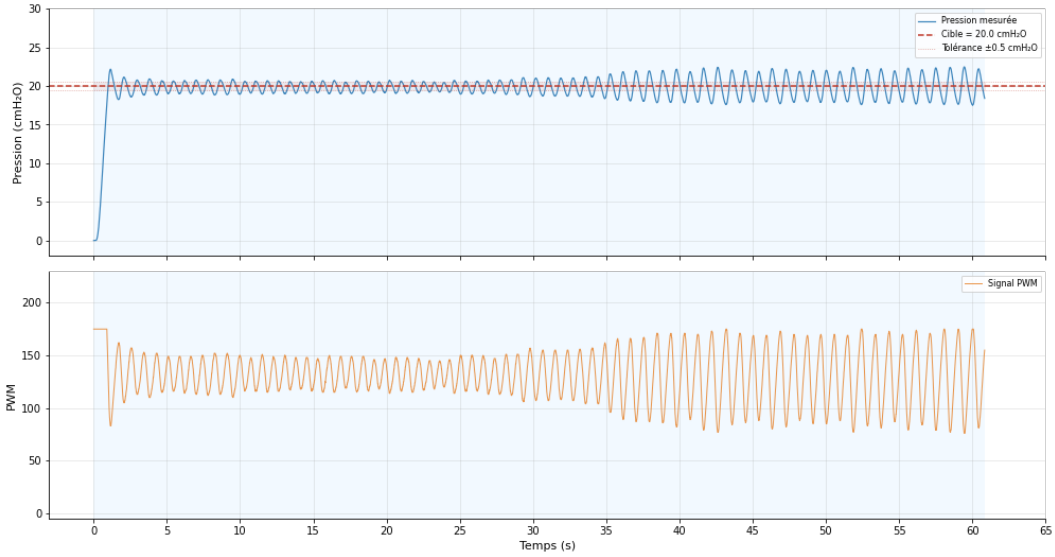


FIGURE 4.5 – Oscillations de pression observées lors du premier essai sur poumon artificiel, pression cible = 20 cmH_2O.

Ce phénomène a été identifié après discussion avec le personnel médical comme cliniquement problématique. En effet, l'objectif du dispositif est de reproduire fidèlement la sensation que perçoit le patient lors d'une ventilation mécanique assistée non invasive, afin de l'y préparer et de l'y habituer avant la séance de radiothérapie. Or, un respirateur médical professionnel est capable de maintenir une pression positive continue et stable pendant le plateau de pression et ne génère pas d'aussi importantes variations de pression. Proposer au patient un entraînement avec des oscillations importantes reviendrait à l'habituer à une sensation fondamentalement différente de celle qu'il ressentira lors de la vraie intervention, ce qui rendrait le dispositif inutile thérapeutiquement.

D'un point de vue physique, ces oscillations peuvent s'expliquer par différents facteurs. La turbine centrifuge possède une inertie mécanique, elle ne peut pas modifier instantanément sa vitesse de rotation en réponse aux corrections du régulateur. Cela introduit un retard entre la commande et son effet sur la pression. Un autre facteur serait le phénomène d'instabilité due au retard pur, bien connu en automatique. Le régulateur PID, en cherchant à corriger continuellement l'erreur, génère des corrections successives qui s'amplifient en raison de ce retard, créant un cycle d'oscillations. Ce phénomène est inhérent à l'architecture du système.[25]

4.4.3 Modifications du régulateur

Face à ce constat, plusieurs modifications ont été apportées au régulateur pour essayer de réduire sans compromettre la précision de la pression cible. Les adaptations efficaces sont décrites ci-dessous.

Le plus gros changement est l'adoption d'un mode de contrôle hybride pour séparer la phase inspiratoire de la phase d'apnée. En théorie, lorsque la pression cible est atteinte et stable, le débit nécessaire pour la maintenir est presque constant. Il n'est donc pas nécessaire de recalculer la commande de la turbine continuellement, ce qui est la source des oscillations. L'idée est donc que le régulateur PID soit actif lors de l'inspiration et tente d'atteindre la pression cible et de se stabiliser. Une fois atteinte, le signal PWM est enregistré en tant que PWM de stabilisation. Ce signal est alors envoyé en continu pendant toute la phase d'apnée, sans que le PID recalcule de nouvelles valeurs.

Pour éviter que le PWM mémorisé ne soit faussé par une valeur instantanée bruitée, il est calculé comme la moyenne des dix dernières valeurs de PWM enregistrées pendant la phase de convergence. Un offset fixe de +10, déterminé empiriquement, est ensuite ajouté à cette moyenne. Il permet de compenser la légère sous-estimation du PWM nécessaire au maintien de la pression observée lors des premiers tests. Cette approche permet d'obtenir une valeur de PWM de stabilisation plus stable et cohérente d'un cycle à l'autre.

Pour compenser les petites variations de pression qui pourraient tout de même survenir dans le circuit (comme des fluctuations de la turbine ou des mouvements respiratoires du patient), une correction proportionnelle de faible gain ($K_{p,apnée}$=0.8) est conservée tout au long de la phase d'apnée. Cette correction s'active uniquement si l'erreur dépasse la zone morte, qui est un seuil minimal fixé à ± 0.2 cmH_2O. Donc tant que la pression reste dans cette zone, on garde le signal $PWM_{stabilisation}$ constant et le dispositif ne produit plus d'oscillations.

Certains paramètres ont aussi connu quelques ajustements. Le coefficient du filtre passe-bas a été réduit de $\alpha = 0.2$ à $\alpha = 0.05$, pour lisser davantage le signal venant du capteur de pression. Et le terme dérivé K_d, qui avait été initialement désactivé lors de l'optimisation en laboratoire, a été réactivé à $K_d = 0.3$. Le poumon artificiel apporte un comportement différent en introduisant la compliance pulmonaire au circuit. Cela crée des variations de pression plus rapides et plus prononcées, ce qui rend le terme dérivé utile pour amortir les transitions. Sa sensibilité au bruit, qui avait été problématique en laboratoire, est ici compensée par le filtrage plus agressif.

Le code complet de la deuxième version du régulateur est disponible en annexe J.

4.4.4 Résultats après modification

Les modifications décrites ci-dessus ont été validées expérimentalement sur le poumon artificiel EasyLung pour plusieurs pressions cibles comprises entre

16 et 20 cmH_2O. Les résultats sont présentés à la figure 4.6 et résumés dans le tableau 4.3.

TABLE 4.3 – Performances du régulateur PID v2 sur poumon artificiel EasyLung

Pression cible (cmH_2O)	Précision plateau (cmH_2O)	Erreur moyenne (cmH_2O)	Temps montée (s)
16	±0.055	+0.43	2.5
17	±0.070	+0.10	1.9
18	±0.085	−0.19	2.0
19	±0.100	+0.18	2.0
20	±0.050	−0.06	2.4

Les résultats montrent que les oscillations ont été considérablement réduites par rapport aux premiers essais, passant d'une amplitude de $\pm 2.3 cmH_2O$ à moins de $\pm 0.1 cmH_2O$. La précision obtenue sur le plateau est cliniquement satisfaisante pour l'ensemble des pressions testées, bien qu'une erreur moyenne légèrement élevée de 0.43 cmH_2O soit observée à 16 cmH_2O.

Ces résultats valident le fonctionnement du dispositif sur poumon artificiel et confirment que le mode de contrôle hybride adopté constitue une solution efficace pour résoudre le problème d'oscillations inhérent à l'architecture à turbine centrifuge.

4.4.5 Comportement face aux perturbations

Afin d'évaluer la robustesse du régulateur, une perturbation externe a été appliquée au système lors du test à 18 cmH_2O. L'objectif était d'observer le comportement du dispositif dans le cas où un patient essayerait d'expirer lors du plateau de pression. Pour reproduire cette situation, le poumon artificiel a été comprimé manuellement à plusieurs reprises pendant la phase d'apnée du troisième cycle.

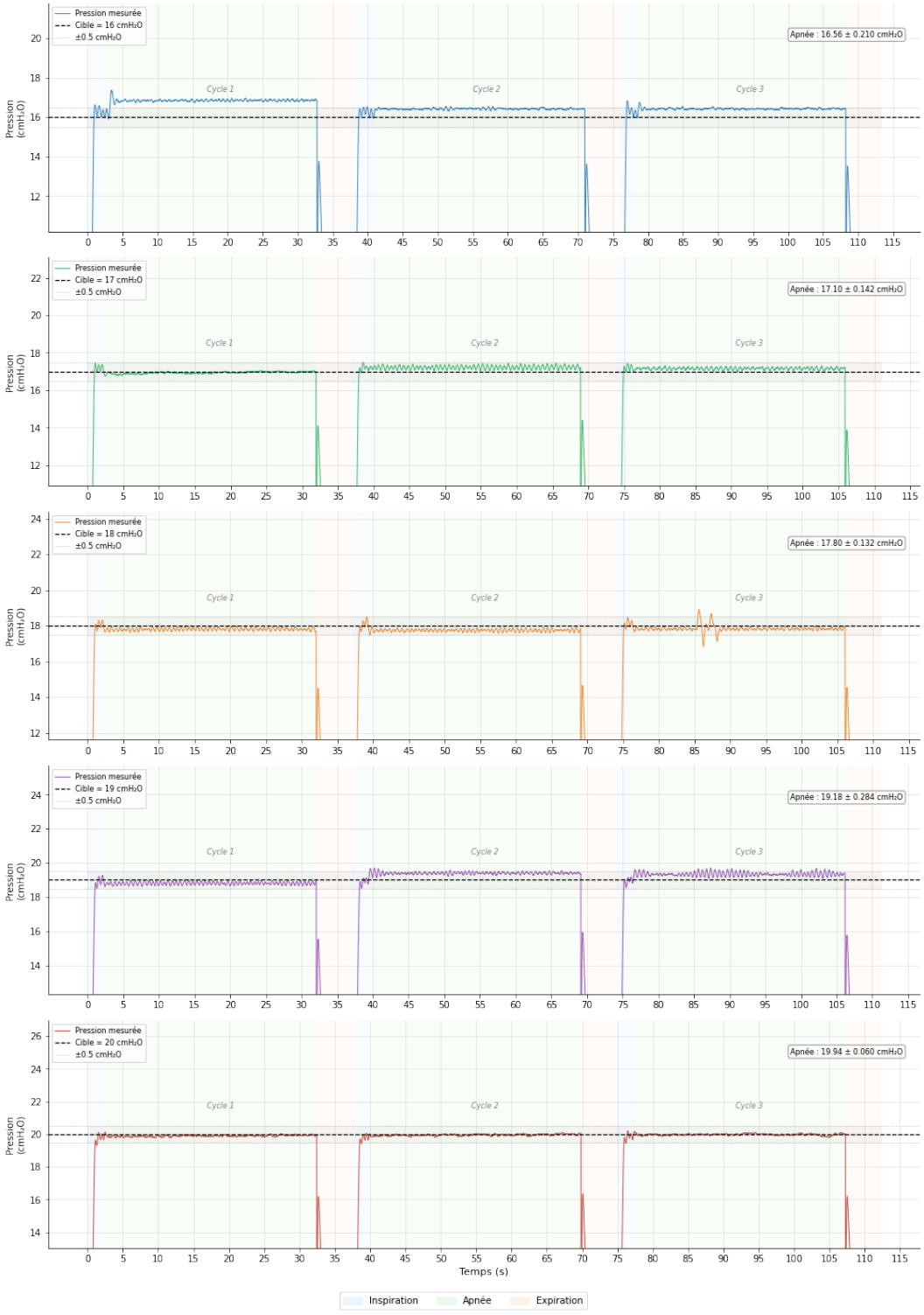


FIGURE 4.6 – Performances sur easylung - comparaison des pressions 16, 17, 18, 19 et 20 cmH_2O

La figure 4.7 montre la réponse du régulateur à ces perturbations. On observe que chaque compression provoque une chute transitoire de la pression, suivie d'un retour rapide vers la valeur cible. Le régulateur parvient à compenser efficacement les perturbations et à rétablir la pression cible en moins de 2 secondes, sans déstabilisation durable du système. Ce comportement confirme la robustesse du mode de contrôle hybride face aux perturbations externes

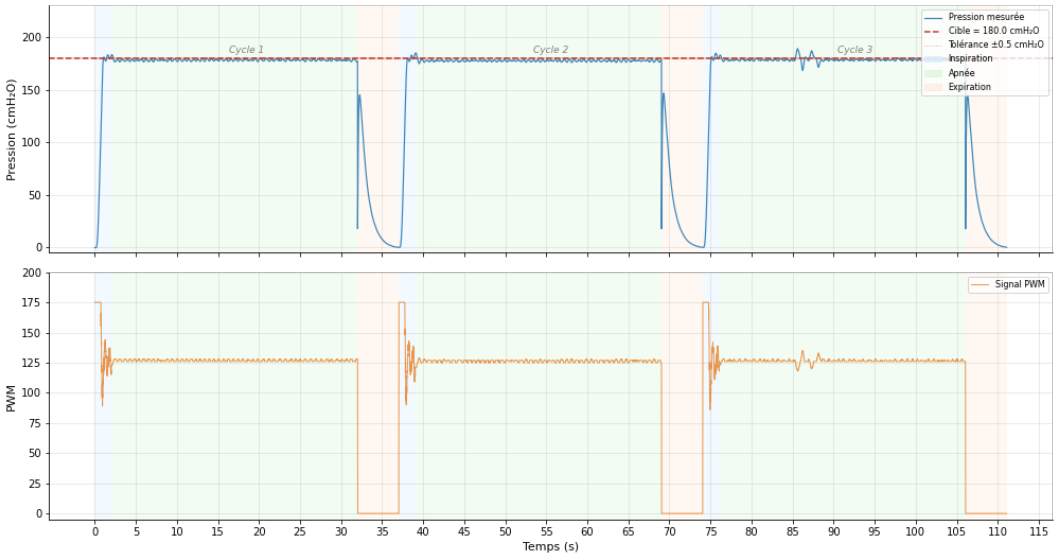


FIGURE 4.7 – Réponse du régulateur PID V2 à des perturbations manuelles lors du troisième cycle

Test préliminaire sur sujet simple

En plus des tests sur poumon artificiel, un test préliminaire a été réalisé sur un sujet sain [1]. Cet essai avait pour but d'évaluer le comportement et les performances du dispositif sur un être humain et d'avoir un retour sur la sensation que celui-ci induit à l'utilisateur. Ce test a été réalisé avec les mesures de sécurité nécessaires, bouton d'urgence à disposition et possibilité d'enlever le masque à tout moment.

Le montage utilisé est identique à celui des tests sur poumon artificiel, avec l'ajout du capteur de débit SFM3300-D en bout de circuit, directement avant le filtre et le masque. Ce capteur, non exploitable lors des tests en circuit ouvert ou sur poumon artificiel, permet de mesurer le débit réel au niveau du masque, qu'il soit inspiratoire ou expiratoire.

Deux pressions cibles (15 et 17 cmH_2O) ont été testées sur un seul cycle, avec un plateau de pression de 15 secondes. Les résultats sont présentés aux figures 4.8 et 4.9.

1. Le sujet ayant participé à cet essai est l'auteure de ce travail.

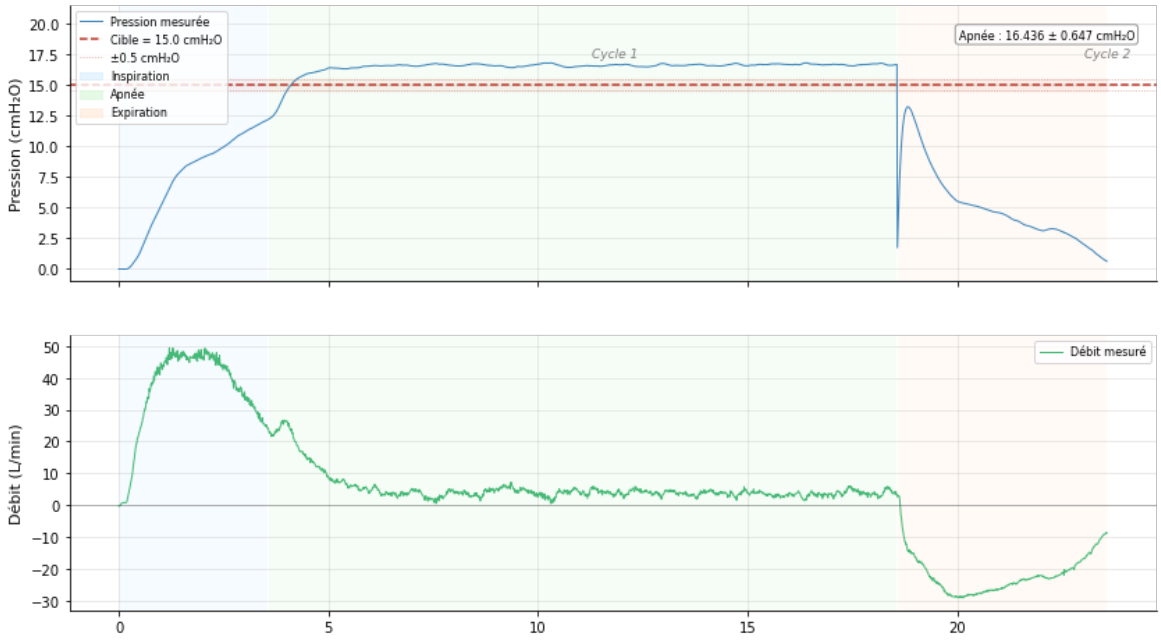


FIGURE 4.8 – Test du dispositif sur sujet sain, pression cible de 15 cmH_2O

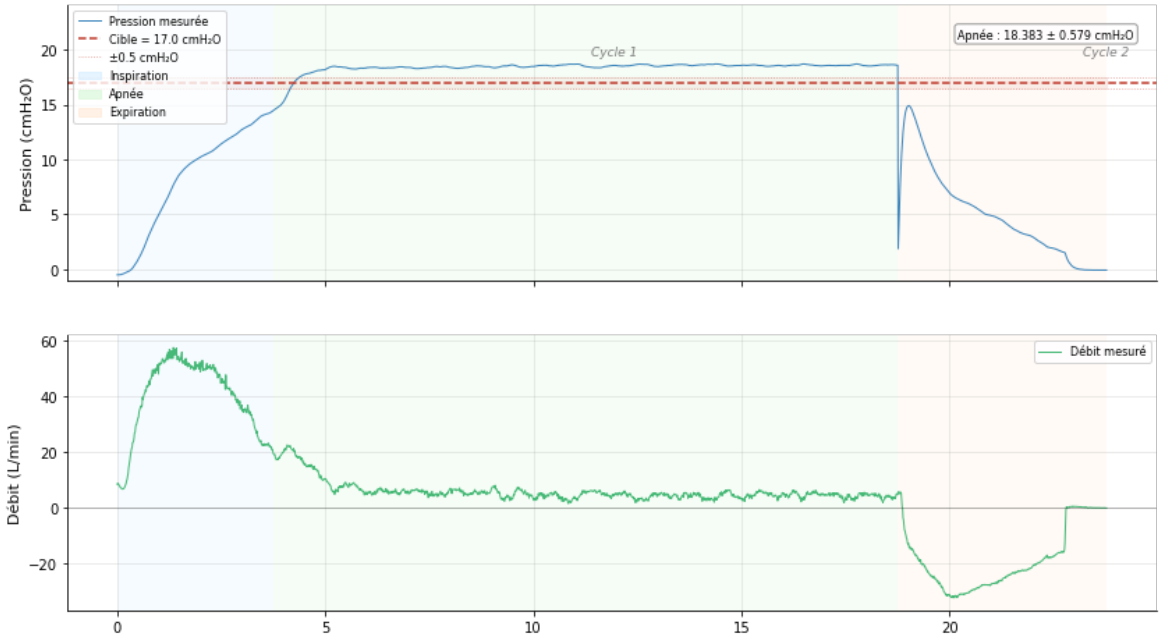


FIGURE 4.9 – Test du dispositif sur sujet sain, pression cible de 17 cmH_2O

Pour les deux essais, on observe bien la présence d'un plateau de pression maintenu pendant la durée prescrite. Le temps de montée en pression est d'environ 5 secondes, ce qui est légèrement plus élevé qu'avec le poumon artificiel. Pour l'essai à 15 cmH_2O, la pression moyenne mesurée est de 16.4

cmH_2O avec une précision de ± 0.647 cmH_2O. Pour l'essai à 17 cmH_2O, on a le même constat. La pression moyenne mesurée est de 18.4 cmH_2O avec une précision de 0.579 cmH_2O. Ces différences de pression correspondent à un écart de 1.4 cmH_2O par rapport à la cible, ce qui est assez conséquent. Ce comportement est probablement dû à la mémorisation du PWM de stabilisation en début d'apnée, qui tend à légèrement surestimer le signal de commande nécessaire au maintien de la pression. La précision est également moins bonne que lors des essais sur poumon artificiel. Ce n'est pas étonnant car un poumon humain ne se comporte pas aussi parfaitement qu'un artificiel. La respiration spontanée du sujet introduit des perturbations continues sur la pression dans le circuit auxquelles le régulateur doit s'adapter en permanence.

Pour ce qui est de la sensation ressentie lors de l'utilisation du dispositif, elle a été jugée comparable à celle produite par le ventilateur médical Bellavista utilisé lors d'une session d'observation clinique en novembre 2025. Ce retour, bien que subjectif et basé sur un souvenir, constitue un premier indicateur encourageant quant à la pertinence du dispositif comme outil d'entraînement préparatoire à la MANIV.

Chapitre 5

Discussion

5.1 Performances du prototype par rapport aux spécifications

Les spécifications techniques définies au chapitre 2 en concertation avec l'équipe des Cliniques Universitaires Saint-Luc constituent le cahier des charges auquel le prototype doit répondre. Le tableau 5.1 confronte chacune de ces spécifications aux résultats obtenus lors des phases de validation expérimentale.

TABLE 5.1 – Comparaison des spécifications techniques et des performances obtenues

Spécification	Valeur cible	Résultat obtenu	Statut
Durée phase inspiratoire	150 – 300 ms	2 – 12 s selon la pression et la configuration	**Non**
Pression maximale maintenue	$\leq$ 20 mbar (20 cmH_2O)	5 – 20 cmH_2O, précision $\pm$0.1 cmH_2O en labo, $\pm$0.2 cmH_2O sur poumon artificiel	**Oui**
Durée du plateau	$\leq$ 20 secondes	Durée paramétrable, validée jusqu'à 30 s	**Oui**
Durée de l'expiration	Variable, prédéfinie par le clinicien	Paramétrable dans le code	**Oui**
Mode d'expiration	Non assistée, relâchement de pression	Expiration libre via valve de fuite	**Oui**

Suite page suivante...

Spécification	Valeur cible	Résultat obtenu	Statut
Sécurité — dépassement de seuil	Arrêt immédiat de la session	Implémentée, seuil paramétrable	**Oui**
Liberté expiratoire	Expiration libre à tout moment	Assurée par la valve de fuite permanente	**Oui**
Architecture pneumatique	Double circuit entrée/sortie séparées	Circuit unique avec fuite calibrée	**Non**
Source d'air	Air ambiant, sans oxygène médical	Turbine centrifuge sur air ambiant	**Oui**
Portabilité	Dispositif compact, usage domiciliaire	Prototype de laboratoire, transportable, pas encore encapsulé	**Partiel**
Interface patient	Simple, sans formation technique	Non implémentée dans le prototype actuel	**Non**
Accès aux réglages	Réservé au clinicien	Via port série Arduino uniquement	**Partiel**
Limitation des sessions	Blocage automatique après N sessions	Non implémentée dans le prototype actuel	**Non**

où **Oui** indique une spécification atteinte, **Partiel** une spécification partiellement atteinte, et **Non** une spécification non atteinte dans le prototype actuel.

L'analyse du tableau met en évidence que le prototype répond correctement aux spécifications les plus critiques sur le plan clinique : la pression cible est maintenue avec une précision satisfaisante sur toute la plage thérapeutique, la durée du plateau est paramétrable et respectée, et le patient dispose à tout moment d'une voie d'expiration libre via la valve de fuite.

Deux spécifications sont partiellement atteintes. La portabilité du dispositif est respectée car le dispositif a été déplacé jusqu'aux Cliniques universitaires Saint-Luc pour les tests sur poumon artificiel. Mais le transport reste très délicat car le dispositif est encore fragile. L'absence d'une armature pouvant contenir les composants électroniques le rend peu adapté à un usage domiciliaire en l'état. Pour ce qui est de l'accès aux réglages, il se fait, pour l'instant, via le

port série de l'Arduino, ce qui nécessite un ordinateur ainsi que les connaissances adaptées. Un accès protégé pour le réglage des paramètres reste encore à réfléchir et sera discuté dans la section 5.3.

Quatre spécifications n'ont par contre pas été atteintes dans le prototype actuel. La plus importante est la durée de la phase inspiratoire, dont la valeur visée était entre 150 et 300 ms pour reproduire fidèlement l'action du MANIV. Ces valeurs ne sont pas atteignables avec une turbine centrifuge en raison de son inertie mécanique. En effet, la turbine ne peut pas modifier instantanément sa vitesse de rotation, ce qui pénalise la vitesse de montée en pression. L'architecture en double circuit, permettant de séparer les flux inspiratoires et expiratoires n'a pas été implémentée. Le prototype repose sur un circuit unique avec valve de fuite calibrée, qui constitue une solution alternative. Enfin, l'interface patient et le blocage automatique des sessions n'ont pas été implémentés dans ce prototype mais constituent des perspectives de développements futurs identifiées dans la section 5.3.

5.2 Limitations actuelles

Le prototype présente plusieurs limitations qui devront être adressées avant d'envisager une utilisation clinique.

5.2.1 Limitations techniques

La principale limitation technique du prototype est liée au choix de la turbine centrifuge WS7040. Ce composant, sélectionné en début de projet pour sa disponibilité et son faible coût, présente deux contraintes majeures qui ont impacté les performances du dispositif.

D'une part, son inertie mécanique limite les variations rapides de vitesse de rotation, ce qui résulte en un temps de montée supérieur à celui attendu. D'autre part, sa puissance est insuffisante pour compenser le débit de fuite et atteindre 20 cmH_2O avec la valve complètement ouverte, nécessitant une réduction de la fuite pour les pressions les plus élevées. Ces deux limitations proviennent de l'architecture à turbine et ne peuvent pas être résolues via le logiciel. Une première piste d'amélioration serait de remplacer la turbine actuelle par un modèle présentant une meilleure dynamique de réponse et une puissance accrue. Une approche plus avancée consisterait à combiner une turbine avec une valve proportionnelle[1], afin de moduler le débit d'air de manière plus précise, au prix d'une architecture plus complexe.

Il faut cependant rester conscient que le prototype a été conçu comme un dispositif à bas coût, destiné à un usage à domicile et non comme un respirateur médical certifié. Il n'est pas directement comparable avec des ventilateurs médicaux tels que le Bellavista et les performances obtenues doivent être évaluées au regard des spécifications définies pour ce prototype spécifique.

5.2.2 Limitations de validation

Sur le plan de la validation, le prototype en est encore à un stade préliminaire.

Les premiers tests réalisés sur humain avaient pour seul but de confirmer sa potentielle faisabilité dans un contexte plus réel. Il n'y a eu qu'une seule séance, avec uniquement deux pressions cibles testées, cela ne représente absolument pas une validation clinique formelle. Dans un premier temps, des tests supplémentaires et plus rigoureux sur poumon artificiel seraient nécessaires. Par la suite, une validation clinique approuvée par un comité d'éthique, avec un protocole strict et détaillé, devrait être menée. Ces tests demanderaient un panel de participants comprenant des patients atteints de cancer, ainsi que des sujets sains.

La validation qualitative quant à elle est complètement subjective et ne constitue pas une preuve suffisante. Elle repose sur le souvenir de la sensation perçue lors de l'essai de la MANIV, datant de plusieurs mois. Une évaluation rigoureuse de la similitude de sensation entre le prototype et la MANIV nécessiterait un protocole de comparaison, idéalement avec des patients ayant déjà eu recours à la MANIV.

5.3 Perspectives d'amélioration

Le prototype développé confirme la faisabilité d'un dispositif d'entrainement à la MANIV, portable et à bas coût. L'appareil est encore dans les premières phases de développement, plusieurs axes d'amélioration ont donc été identifiés, tant sur le plan technique que sur le plan de l'expérience utilisateur et de l'intégration clinique. Ces perspectives font partie des prochaines étapes de développement du dispositif.

5.3.1 Améliorations techniques du prototype

Plusieurs modifications doivent être apportées pour que l'appareil puisse être utilisé par le patient en toute sécurité.

Tout d'abord, l'encapsulation des composants tels que le ventilateur, son driver, l'Arduino et tous les câbles dans une boîte fermée, afin qu'ils ne soient pas directement accessibles et pour les protéger. Dans cette boîte, chaque composant serait fixé à sa place. Toute la partie tuyauterie serait évidemment située en dehors de cette boîte.

Ensuite, l'alimentation de laboratoire doit être remplacée par une alimentation domestique, pour que la machine puisse fonctionner en étant simplement branchée aux prises standard que l'on retrouve dans les habitations.

Une amélioration éventuelle serait d'ajouter un filtre en sortie de ventilateur, afin de filtrer une première fois l'air sortant du ventilateur et de compléter

l'action du filtre HMEF déjà en place.

Il convient aussi de discuter de l'utilité et de l'importance du capteur de débit dans le dispositif final. Bien que celui-ci ait été très utile dans les phases de développement du dispositif et les phases de test, sa présence dans le produit final n'est pas essentielle. En effet, la mesure essentielle pour le bon fonctionnement de la machine est la pression dans le circuit, et non le débit. On pourrait donc le supprimer de l'architecture finale, ce qui simplifierait l'appareil et réduirait ses coûts de production.

Il serait également pertinent d'intégrer un mode de respiration assistée, similaire au mode AVM (Adaptative Ventilation Mode) utilisé sur le ventilateur Bellavista. Dans la version actuelle du dispositif, le seul moyen pour le patient de respirer lorsque le circuit est connecté est par la valve de fuite, ce qui rend la respiration spontanée assez difficile et inconfortable. Ajouter un mode de respiration libre permettrait au patient de s'installer à son aise, et respirer à travers le masque jusqu'à ce qu'il se sente prêt à lancer un cycle d'entraînement. Il pourrait fonctionner en détectant les intentions respiratoires du patient via le capteur de pression, et donc en activant la turbine au bon moment. Cette approche est décrite dans l'article de Garmendia et al. comme base de fonctionnement de leur ventilateur[23], et constituerait une évolution du prototype vers un dispositif plus complet et plus confortable pour le patient.

Enfin, la dernière étape mais pas des moindres est l'encadrement réglementaire du dispositif. La question de savoir si ce dispositif entre dans la catégorie des dispositifs médicaux au sens du règlement européen MDR (Medical Device Regulation, UE 2017/745) est délicate. Selon l'article 2 de ce règlement, un dispositif médical est défini comme tout instrument destiné à être utilisé chez l'être humain à des fins, entre autres, de diagnostic, de prévention, de contrôle, de traitement ou d'atténuation d'une maladie.[29] Bien que l'appareil n'ait pas de visée thérapeutique, il contribue indirectement à la prévention de complications liées à une potentielle intolérance à la MANIV, ce qui rejoint la notion de "prévention d'une maladie" et placerait le dispositif au rang d'appareil médical.

Il pourrait également être classifié comme accessoire de dispositif médical au sens du point 2 du même article, qui définit tout article "destiné à être utilisé avec un ou plusieurs dispositifs médicaux donnés pour permettre une utilisation de ce ou ces derniers conforme à sa destination ou pour contribuer spécifiquement et directement à la fonction médicale" de celui-ci.[29] En effet, le dispositif développé est explicitement conçu pour préparer le patient à l'utilisation du ventilateur médical Bellavista, contribuant ainsi directement à l'efficacité de la MANIV.

Quelle que soit la conclusion sur cette classification, une démarche de certification selon le MDR devrait être envisagée avant toute utilisation clinique.

Cela implique la démonstration de la conformité du dispositif aux exigences générales en matière de sécurité et de performances définies à l'annexe I du règlement MDR[29], notamment en ce qui concerne la sécurité électrique, la biocompatibilité des matériaux en contact avec le patient, et l'évaluation clinique formelle telle que définie à l'article 2, point 44 du même règlement. Cette réflexion réglementaire devra être menée en concertation avec les équipes médicales et réglementaires des Cliniques Universitaires Saint-Luc.

5.3.2 Interface et expérience utilisateur

Au-delà des aspects techniques, l'expérience utilisateur représente un axe d'amélioration majeur, que ce soit pour le patient ou pour le personnel médical.

Du côté du clinicien, il est indispensable d'intégrer un système lui permettant d'encoder les paramètres déterminants, tels que la pression cible, la marge d'erreur acceptée, la durée d'expiration, et cela sans que le patient ne puisse modifier ces paramètres. Plusieurs solutions sont envisageables, comme une clé d'accès uniquement détenue par le médecin, qui donnerait accès à des paramètres supplémentaires par rapport à ceux auxquels le patient a accès. À la place de cette clé, il pourrait y avoir un code à entrer pour accéder aux fonctionnalités supplémentaires, mais cela impose la présence d'une interface permettant de saisir ce code. Une autre idée serait d'intégrer un port USB sur la machine, de telle sorte que le médecin puisse s'y connecter avec son ordinateur, s'identifier et accéder aux paramètres pour les modifier via son ordinateur. Encore une autre possibilité serait de conserver l'encodage des paramètres via Arduino, et de demander à l'ingénieur biomédical de l'hôpital de programmer la machine selon les instructions du médecin. Une courte formation à l'implémentation Arduino pourrait être proposée si nécessaire.

Du côté du patient, une interface simple d'utilisation est la priorité. Les composants seraient donc tous placés dans une boîte fermée, sur laquelle il y aurait un petit écran ainsi que quelques boutons permettant aux utilisateurs de naviguer à travers les fonctionnalités. L'interface se composerait d'un bouton de démarrage permettant de lancer la session, et d'un bouton stop pour arrêter la session à tout moment. Un autre bouton permettrait de sélectionner le nombre de cycles à effectuer dans la session, et un bouton supplémentaire pour choisir la durée de l'apnée pour la session. Le nombre de cycles et la durée d'apnée seraient modifiables par le patient afin qu'il puisse lui-même moduler son entraînement selon les recommandations. Des sécurités seraient bien sûr intégrées pour que le patient ne puisse pas effectuer un nombre infini de sessions et de cycles, ni des apnées trop longues.

Un retour visuel pourrait également être intégré au boîtier du dispositif, sous forme de LEDs par exemple. Il y aurait une LED de couleur différente pour chaque phase du cycle (inspiration, apnée, expiration), ce qui permettrait

au patient de suivre visuellement le déroulement du cycle sans avoir besoin d'instructions verbales.

5.3.3 Programme d'entraînement et suivi clinique

Une perspective de développement particulièrement intéressante serait un programme d'entraînement personnalisé en complément de l'appareil. Ce programme pourrait être intégré dans une application mobile, ou présenté sous forme de brochure pour les patients moins à l'aise avec les outils numériques, ou ne disposant pas d'un smartphone. Il serait élaboré en concertation avec les professionnels de santé, afin de s'adapter au mieux aux patients et à leurs besoins. Son objectif serait d'habituer progressivement les patients à être soumis à une apnée forcée. Le nombre de cycles et la durée des apnées augmenteraient au fil des jours, jusqu'à atteindre la cible définie par le médecin.

Ce type de progression graduelle est cohérent avec les principes de la réhabilitation pulmonaire, qui recommandent une augmentation progressive de la charge d'entraînement afin d'améliorer la tolérance et les performances respiratoires des patients.[30] Des protocoles d'entraînement au *Deep Inspiration Breath Hold* recommandent déjà une pratique quotidienne à domicile, avec une augmentation progressive de la durée des apnées jusqu'à la cible prescrite, ce qui rejoint le principe d'un programme structuré d'habituation à l'apnée avant la radiothérapie.[31]

Dans le cadre d'une application, un assistant vocal pourrait y être intégré. Il serait connecté au dispositif et chargé de guider le patient tout au long de la session, de lui expliquer ce qu'il se passe, de l'encourager lorsqu'il suit bien les instructions et de le corriger lorsqu'il commet des erreurs. Cet assistant servirait à guider le patient dans l'utilisation d'un dispositif potentiellement impressionnant et à le rassurer.

On pourrait également intégrer à cette application un système de suivi des sessions, en enregistrant les données de chaque session. Le médecin pourrait alors consulter ces données et adapter l'entraînement si nécessaire.

Conclusion

L'objectif de ce projet était de concevoir un prototype fonctionnel de dispositif d'entraînement respiratoire à bas coût et destiné à l'usage domiciliaire. Il devait correspondre aux caractéristiques de la MANIV, une méthode qui utilise la ventilation mécanique assistée pour gérer le cycle respiratoire de patients ayant recours à des séances de radiothérapie. Le caractère anxiogène potentiel de cette procédure justifie la mise en place d'une phase de préparation dédiée aux patients.

Le prototype développé repose sur un système à fuite calibrée inspirée des CPAP, pilotée par un microcontrôleur Arduino Mega et régulée par un algorithme PID. Les phases de caractérisation expérimentale ont permis de valider le bon fonctionnement de chaque composant du circuit et d'établir les relations entre les signaux de commande et les grandeurs physiques produites. Le régulateur PID a été réglé et validé sur une plage de pressions thérapeutiques de 5 à 20 cmH_2O, avec une précision en plateau inférieure à ± 0.15 cmH_2O en laboratoire.

Le dispositif a ensuite été testé sur le poumon artificiel *EasyLung* aux Cliniques Universitaires Saint-Luc afin de le confronter à des conditions d'utilisation plus réalistes. De nouveaux paramètres comme la compliance pulmonaire ont engendré des oscillations de pressions problématiques. Ce problème a été résolu par l'introduction d'un mode de contrôle hybride distinguant la phase d'inspiration régulée par le PID et la phase d'apnée désormais à commande fixe. Ces modifications ont drastiquement réduit les oscillations ce qui a permis d'obtenir des plateaux de pressions stables, prometteurs pour la suite.

Enfin, un test préliminaire sur sujet sain a permis de confirmer la faisabilité d'une utilisation sur être humain et d'obtenir un premier retour qualitatif encourageant. La sensation produite par le dispositif a été jugée comparable à celle du ventilateur Bellavista utilisé en clinique, ce qui constitue un premier indicateur de la pertinence du prototype comme outil d'entraînement.

Les résultats obtenus confirment la faisabilité d'un tel dispositif, et les performances produites sont très encourageantes. Plusieurs limitations ont été identifiées telles que le temps de montée en pression lié à l'inertie de la turbine ou l'absence d'interface utilisateur, qui constituent les priorités de développement pour les versions futures. La prochaine étape sera la réalisation de tests sur un plus grand nombre de sujets sains, puis sur des patients, dans le cadre d'un protocole éthique formalisé, afin de valider l'efficacité du dispositif

comme outil de préparation à la MANIV.

Ce projet s'inscrit dans une démarche plus large visant à améliorer la qualité et la précision de la radiothérapie pour les patients atteints de cancer. En facilitant l'accès à la MANIV grâce à un outil d'entraînement simple, portable et abordable, il contribue à l'objectif de rendre ce traitement innovant accessible au plus grand nombre de patients, indépendamment de leur état général ou de leur capacité à tolérer spontanément la ventilation mécanique.

Bibliographie

Conformément aux lignes directrices de l'EPL sur l'utilisation de l'IA générative, je déclare avoir eu recours à Claude (Anthropic, [32]) pour la reformulation linguistique, la mise en forme LaTeX, la génération de code et l'aide à la navigation à travers certaines sources (toutes consultées et vérifiées personnellement)

[1] W. J. Dally, "OP-VENT : A Low-Cost, Easily Assembled, Open-Source Medical Ventilator," *GetMobile : Mobile Comp. and Comm.*, vol. 25, pp. 12–18, Mar. 2022.

[2] J. LaChance, M. Schottdorf, T. J. Zajdel, J. L. Saunders, S. Dvali, C. Marshall, L. Seirup, I. Sammour, R. L. Chatburn, D. A. Notterman, and D. J. Cohen, "PVP1—The People's Ventilator Project : A fully open, low-cost, pressure-controlled ventilator research platform compatible with adult and pediatric uses," *PLOS ONE*, vol. 17, p. e0266810, May 2022.

[3] G. Van Ooteghem, D. Dasnoy-Sumell, M. Lambrecht, G. Reychler, G. Liistro, E. Sterpin, and X. Geets, "Mechanically-assisted non-invasive ventilation : A step forward to modulate and to improve the reproducibility of breathing-related motion in radiation therapy," *Radiotherapy and Oncology*, vol. 133, pp. 132–139, 2019.

[4] G. Van Ooteghem, D. Dasnoy-Sumell, J. A. Lee, and X. Geets, "Mechanically-assisted and non-invasive ventilation for radiation therapy : A safe technique to regularize and modulate internal tumour motion," *Radiotherapy and Oncology*, vol. 141, pp. 283–291, 2019.

[5] Q. Luo and D. P. Smith, "Global cancer burden : progress, projections, and challenges," *The Lancet*, vol. 406, pp. 1536–1537, Oct. 2025.

[6] H. Zhu, M. L. K. Chua, I. Chitapanarux, O. Kaidar-Person, C. Mwaba, M. Alghamdi, A. R. Mignola, N. Amrogowicz, G. Yazici, Z. Bourhaleb, H. Mahmood, G. M. Faruque, M. Thiagarajan, A. Acharki, M. Ma, M. Harutyunyan, H. Sriplung, Y. Chen, R. Camacho, Z. Zhang, and M. Abdel-Wahab, "Global radiotherapy demands and corresponding radiotherapy-professional workforce requirements in 2022 and predicted to 2050 : a population-based study," *The Lancet Global Health*, vol. 12, pp. e1945–e1953, Dec. 2024.

[7] "Cancer Atlas."

[8] M. Abdel-Wahab, S. S. Gondhowiardjo, A. A. Rosa, Y. Lievens, N. El-Haj, J. A. Polo Rubio, G. B. Prajogi, H. Helgadottir, E. Zubizarreta, A. Meghzifene, V. Ashraf, S. Hahn, T. Williams, and M. Gospodarowicz, "Global Radiotherapy : Current Status and Future Directions—White Paper," *JCO Global Oncology*, pp. 827–842, June 2021.

[9] M. V. Studio, "Radiothérapie."

[10] E. Sterpin, "Engineering challenges in proton therapy." Notes de cours, Université catholique de Louvain, 2026. Cours donné à l'École Polytechnique de Louvain.

[11] "Une dosimétrie précise pour un traitement du cancer de qualité," Feb. 2020.

[12] P. Giraud and R. Garcia, "Contrôle de la respiration en radiothérapie : principaux aspects techniques et intérêts cliniques," *Bulletin du Cancer*, vol. 97, no. 7, pp. 847–856, 2010.

[13] A. Paumier, A. Crespeau, S. Krhili, M. Georgin-Mège, C. Tuchais, J. Mesgouez, P. Cellier, A. Lisbona, F. Denis, and D. Autret, "Étude dosimétrique des différentes techniques de gestion du mouvement respiratoire pour l'irradiation thoracique en conditions stéréotaxiques," *Cancer/Radiothérapie*, vol. 16, pp. 263–271, July 2012.

[14] P. Giraud and A. Houle, "Respiratory Gating for Radiotherapy : Main Technical Aspects and Clinical Benefits," *International Scholarly Research Notices*, vol. 2013, no. 1, p. 519602, 2013. _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1155/2013/519602.

[15] J. P. Santoro, E. Yorke, K. A. Goodman, and G. S. Mageras, "From phase-based to displacement-based gating : a software tool to facilitate respiration-gated radiation treatment," *Journal of Applied Clinical Medical Physics*, vol. 10, pp. 132–141, Oct. 2009.

[16] A. J. Hayden, M. Rains, and K. Tiver, "Deep inspiration breath hold technique reduces heart dose from radiotherapy for left-sided breast cancer," *Journal of Medical Imaging and Radiation Oncology*, vol. 56, pp. 464–472, Aug. 2012.

[17] C. Bergom, A. Currey, N. Desai, A. Tai, and J. B. Strauss, "Deep Inspiration Breath Hold : Techniques and Advantages for Cardiac Sparing During Breast Cancer Irradiation," *Frontiers in Oncology*, vol. 8, p. 87, Apr. 2018.

[18] V. R. Kini, S. S. Vedam, P. J. Keall, S. Patil, C. Chen, and R. Mohan, "Patient training in respiratory-gated radiotherapy," *Medical Dosimetry : Official Journal of the American Association of Medical Dosimetrists*, vol. 28, no. 1, pp. 7–11, 2003.

[19] G. Van Ooteghem, "Communication personnelle." Entretien oral, 2026. Communication réalisée dans le cadre du projet de mémoire.

[20] Service de radiothérapie oncologique des Cliniques universitaires Saint-Luc, *Mode d'emploi MANIV*. Cliniques universitaires Saint-Luc, Bruxelles,

Belgique, July 2020. Référence documentaire RDTH-DSQ-0123, version 2.0, date d'application : 28 juillet 2020.

[21] J. L. Anderson *et al.*, "The long-lasting relationship of distress on radiation oncology-specific clinical outcomes," *Advances in Radiation Oncology*, vol. 4, no. 2, pp. 199–206, 2019.

[22] G. Cammarota, R. Simonte, P. De Marchis, *et al.*, "Comfort during non-invasive ventilation," *Frontiers in Medicine*, vol. 9, p. 874250, 2022.

[23] O. Garmendia, M. A. Rodríguez-Lazaro, J. Otero, P. Phan, A. Stoyanova, A. T. Dinh-Xuan, D. Gozal, D. Navajas, J. M. Montserrat, and R. Farré, "Low-cost, easy-to-build noninvasive pressure support ventilator for under-resourced regions : open source hardware description, performance and feasibility testing," *European Respiratory Journal*, vol. 55, June 2020.

[24] K. J. Åström and T. Hägglund, *PID Controllers : Theory, Design, and Tuning*. Research Triangle Park, NC : Instrument Society of America, 2 ed., 1995.

[25] H. K. Khalil, *Control Systems : An Introduction*. Michigan Publishing, 2023.

[26] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Pearson, 8 ed., 2019.

[27] "Filtering Noise from Sensor Readings : A Simple Low Pass Filter | The Aspiring Roboticist," Jan. 2013.

[28] IMT Analytics AG, "Easylung — test lung." `https://imtanalytics.com/products/easylung`, 2024. Consulté le 2 août 2026.

[29] Parlement européen et Conseil de l'Union européenne, "Règlement (ue) 2017/745 du parlement européen et du conseil du 5 avril 2017 relatif aux dispositifs médicaux," 2017.

[30] R. Gloeckl, B. Marinov, and F. Pitta, "Practical recommendations for exercise training in patients with COPD," *European Respiratory Review*, vol. 22, pp. 178–186, June 2013.

[31] "Deep Inspiration Breath Hold (DIBH) - Peter MacCallum Cancer Centre."

[32] Anthropic, "Claude (sonnet 5)," 2026. Modèle de langage utilisé comme assistance pour la reformulation, le formatage LaTeX, la génération de code et l'exploration bibliographique.

Annexe A

Datasheet — Capteur de pression MPX5010DP

Freescale Semiconductor
Data Sheet: Technical Data

MPX5010
Rev 13, 10/2012

Integrated Silicon Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated

MPX5010
MPXV5010
MPVZ5010
Series

0 to 10 kPa (0 to 1.45 psi)
(0 to 1019.78 mm H_2O)
0.2 to 4.7 V Output

The MPxx5010 series piezoresistive transducers are state-of-the-art monolithic silicon pressure sensors designed for a wide range of applications, but particularly those employing a microcontroller or microprocessor with A/D inputs. This transducer combines advanced micromachining techniques, thin-film metallization, and bipolar processing to provide an accurate, high level analog output signal that is proportional to the applied pressure. The axial port has been modified to accommodate industrial grade tubing.

Features

- 5.0% Maximum Error over 0° to 85°C
- Ideally Suited for Microprocessor or Microcontroller-Based Systems
- Durable Epoxy Unibody and Thermoplastic (PPS) Surface Mount Package
- Temperature Compensated over -40° to +125°C
- Patented Silicon Shear Stress Strain Gauge
- Available in Differential and Gauge Configurations
- Available in Surface Mount (SMT) or Through-hole (DIP) Configurations

Application Examples

- Hospital Beds
- HVAC
- Respiratory Systems
- Process Control
- Washing Machine Water Level Measurement (Reference AN1950)
- Ideally Suited for Microprocessor or Microcontroller-Based Systems
- Appliance Liquid Level and Pressure Measurement

ORDERING INFORMATION

Device Name	Case No.	# of Ports			Pressure Type			Device Marking
		None	Single	Dual	Gauge	Differential	Absolute	
Unibody Package (MPX5010 Series)								
MPX5010DP	867C			•		•		MPX5010DP
MPX5010GP	867B		•		•			MPX5010GP
MPX5010GS	867E		•		•			MPX5010D
MPX5010GSX	867F		•		•			MPX5010D
Small Outline Package (MPXV5010 Series)								
MPXV5010DP	1351			•		•		MPXV5010DP
MPXV5010G6U	482	•			•			MPXV5010G
MPXV5010GC6T1	482A		•		•			MPXV5010G
MPXV5010GC6U	482A		•		•			MPXV5010G
MPXV5010GC7U	482C		•		•			MPXV5010G
MPXV5010GP	1369		•		•			MPXV5010GP
Small Outline Package (Media Resistant Gel) (MPVZ5010 Series)								
MPVZ5010G6U	482	•			•			MPVZ5010G
MPVZ5010G7U	482B	•			•			MPVZ5010G
MPVZ5010GW6U	1735		•		•			MZ5010GW
MPVZ5010GW7U	1560		•		•			MZ5010GW

Operating Characteristics

Table 1. Operating Characteristics (V_S = 5.0 Vdc, T_A = 25°C unless otherwise noted, P1 > P2. Decoupling circuit shown in Figure 3 required to meet specification.)

Characteristic		Symbol	Min	Typ	Max	Unit
Pressure Range		P_{OP}	0	—	10 1019.78	kPa mm H_2O
Supply Voltage(1)		V_S	4.75	5.0	5.25	Vdc
Supply Current		I_o	—	5.0	10	mAdc
Minimum Pressure Offset(2) @ V_S = 5.0 Volts	(0 to 85°C)	V_{off}	0	0.2	0.425	Vdc
Full Scale Output(3) @ V_S = 5.0 Volts	(0 to 85°C)	V_{FSO}	4.475	4.7	4.925	Vdc
Full Scale Span(4) @ V_S = 5.0 Volts	(0 to 85°C)	V_{FSS}	4.275	4.5	4.725	Vdc
Accuracy(5)	(0 to 85°C)	—	—	—	±5.0	$\%V_{FSS}$
Sensitivity		V/P	—	450 4.413	—	mV/mm mV/mm H_2O
Response Time(6)		t_R	—	1.0	—	ms
Output Source Current at Full Scale Output		I_{O+}	—	0.1	—	mAdc
Warm-Up Time(7)		—	—	20	—	ms
Offset Stability(8)		—	—	±0.5	—	$\%V_{FSS}$

1. Device is ratiometric within this specified excitation range.
2. Offset (V_{off}) is defined as the output voltage at the minimum rated pressure.
3. Full Scale Output (V_{FSO}) is defined as the output voltage at the maximum or full rated pressure.
4. Full Scale Span (V_{FSS}) is defined as the algebraic difference between the output voltage at full rated pressure and the output voltage at the minimum rated pressure.
5. Accuracy (error budget) consists of the following:

   Linearity: Output deviation from a straight line relationship with pressure over the specified pressure range.

   Temperature Hysteresis:Output deviation at any temperature within the operating temperature range, after the temperature is cycled to and from the minimum or maximum operating temperature points, with zero differential pressure applied.

   Pressure Hysteresis:Output deviation at any pressure within the specified range, when this pressure is cycled to and from the minimum or maximum rated pressure, at 25°C.

   TcSpan: Output deviation over the temperature range of 0° to 85°C, relative to 25°C.

   TcOffset:Output deviation with minimum rated pressure applied, over the temperature range of 0° to 85°C, relative to 25°C.

   Variation from Nominal:The variation from nominal values, for Offset or Full Scale Span, as a percent of V_{FSS}, at 25°C.
6. Response Time is defined as the time for the incremental change in the output to go from 10% to 90% of its final value when subjected to a specified step change in pressure.
7. Warm-up Time is defined as the time required for the product to meet the specified output voltage after the Pressure has been stabilized.
8. Offset Stability is the product's output deviation when subjected to 1000 hours of Pulsed Pressure, Temperature Cycling with Bias Test.

Maximum Ratings

Table 2. Maximum Ratings(1)

Rating	Symbol	Value	Unit
Maximum Pressure (P1 > P2)	P_{max}	40	kPa
Storage Temperature	T_{stg}	–40 to +125	°C
Operating Temperature	T_A	–40 to +125	°C

1. Exposure beyond the specified limits may cause permanent damage or degradation to the device.

Figure 1 shows a block diagram of the internal circuitry integrated on a pressure sensor chip.

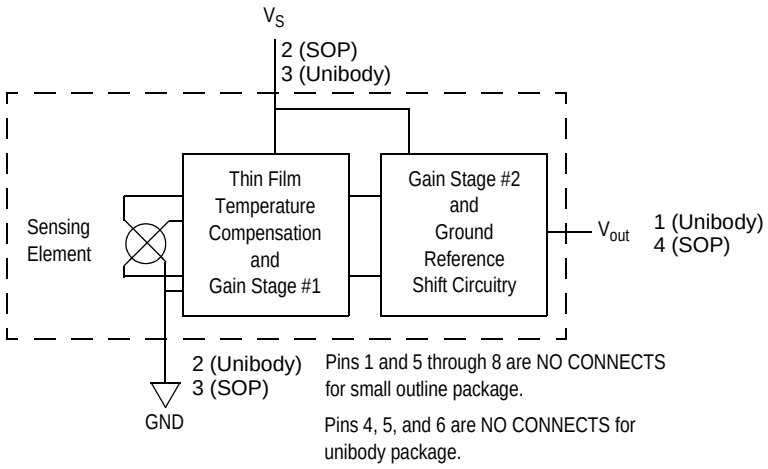


Figure 1. Fully Integrated Pressure Sensor Schematic

ON-CHIP TEMPERATURE COMPENSATION AND CALIBRATION

The performance over temperature is achieved by integrating the shear-stress strain gauge, temperature compensation, calibration and signal conditioning circuitry onto a single monolithic chip.

Figure 3 illustrates the Differential or Gauge configuration in the basic chip carrier (Case 482). A fluorosilicone gel isolates the die surface and wire bonds from the environment, while allowing the pressure signal to be transmitted to the sensor diaphragm.

The MPxx5010G series pressure sensor operating characteristics, and internal reliability and qualification tests are based on use of dry air as the pressure media. Media, other than dry air, may have adverse effects on sensor performance and long-term reliability. Contact the factory for information regarding media compatibility in your application.

Figure 4 shows the recommended decoupling circuit for interfacing the integrated sensor to the A/D input of a microprocessor or microcontroller. Proper decoupling of the power supply is recommended.

Figure 5 shows the sensor output signal relative to pressure input. Typical, minimum, and maximum output curves are shown for operation over a temperature range of 0° to 85°C using the decoupling circuit shown in Figure 4. The output will saturate outside of the specified pressure range.

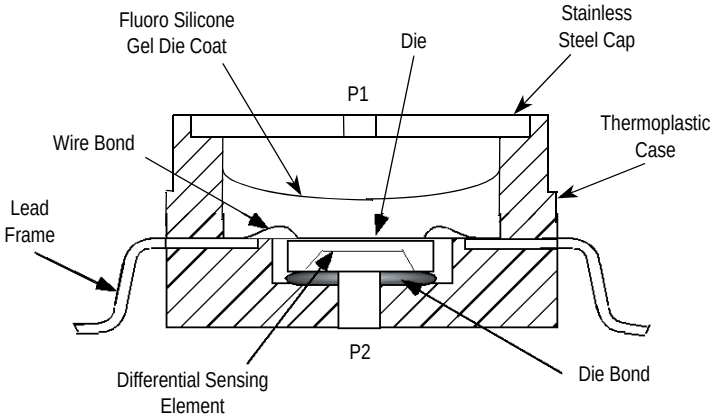


Figure 2. Cross-Sectional Diagram SOP (not to scale)

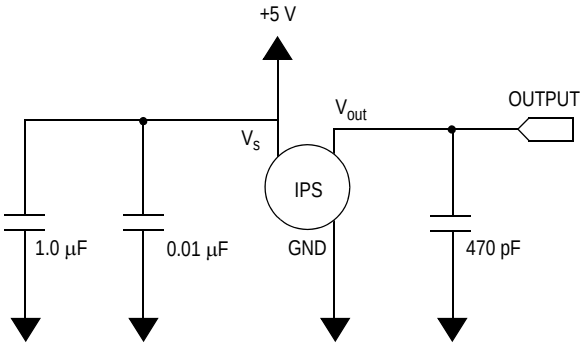


Figure 3. Recommended Power Supply Decoupling and Output Filtering
(For additional output filtering, please refer to Application Note AN1646.)

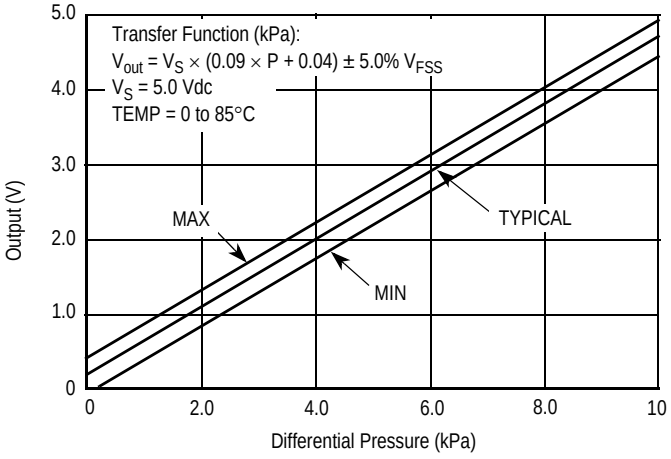
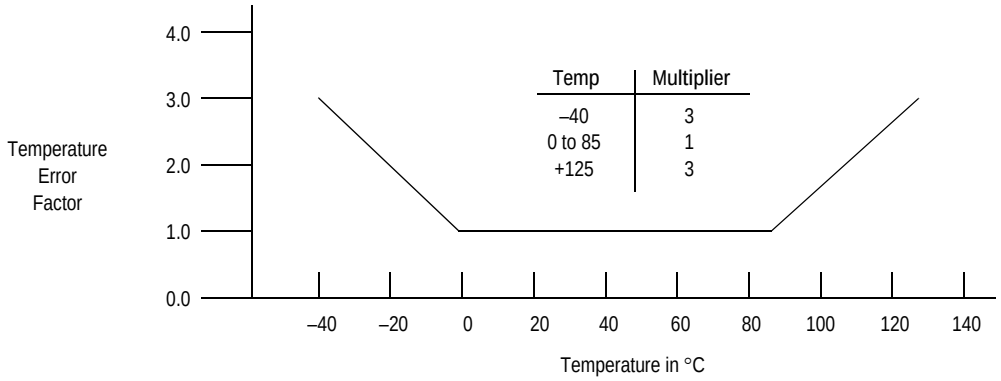


Figure 4. Output vs. Pressure Differential

Transfer Function

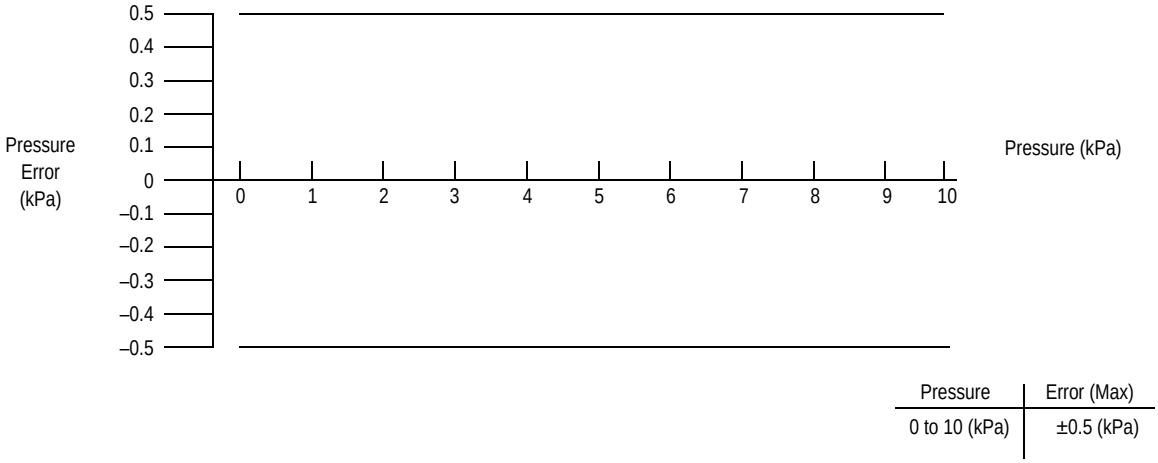
Nominal Transfer Value: $V_{out} = V_S \times (0.09 \times P + 0.04)$
$\pm$ (Pressure Error x Temp. Factor x 0.09 x V_S)
$V_S = 5.0\ V \pm 0.25\ Vdc$

Temperature Error Band



NOTE: The Temperature Multiplier is a linear response from 0° to –40°C and from 85° to 125°C.

Pressure Error Band



Pressure	Error (Max)
0 to 10 (kPa)	±0.5 (kPa)

Annexe B

Datasheet — Capteur de débit SFM3300-D

Datasheet SFM3300-D
Datasheet SFM3300-AW

Digital Flow Meter for medical applications

- Flow range: ± 250 slm, bidirectional
- Small dead space < 10ml
- Single use (-D) version
- Re-use (-AW) version
- Very fast update time (0.5ms)

Product Summary

The SFM3300 series is Sensirion's digital flow meter series designed for medical applications. It measures the flow rate of **air, oxygen and other non-aggressive gases** with superb accuracy. The special design of the flow channel results in a **very small dead space**.

The SFM3300-D is the disposable single-use Version. The re-use version SFM3300-AW **withstands washing and autoclaving procedures.** Both versions are extremely well suited for proximal flow measurements in medical ventilation and other respiratory applications.

The SFM3300 series has been designed with the use by medical professionals in mind. It features **medical cones** for pneumatic connection to standard breathing circuits and a mechanical interface for **easy and reliable electrical reconnection**. The sensor element, signal processing and digital calibration are on a single microchip assuring **very fast signal processing time, best-in-class accuracy** and **superior robustness** to rough handling and adverse conditions.

The well-proven and patented **CMOSens® sensor technology** is perfectly suited for high-quality mass production and is the ideal choice for demanding and cost-sensitive OEM applications.

Applications

- Proximal Flow measurement
- Expiratory flow measurement
- For Ventilation & Anesthesia
- Respiratory measurements
- Metabolic Measurements

OEM options

A variety of custom options can be implemented for high-volume OEM applications (custom flow rates, calibration for other gases etc.). Contact us for more information.

Sensor chip

The SFM3300 series flow meter features a fifth-generation silicon sensor chip. In addition to a thermal mass flow sensor element, the chip contains an amplifier, A/D converter, EEPROM memory, digital signal processing circuitry, and interface. Due to seamless integration of signal acquisition and processing on the single silicon die significant performance and cost benefits are achieved.

1.1 Physical specifications [1]

Parameter	Condition	Value		Unit
Flow range		-250 … +250		slm[2]
		Typical	Max [3]	
Accuracy[4]	span <100 slm span >100 slm offset	3 [5] 7 [5] 0.1 [5]	5 10 0.2	% m.v.[6] % m.v.[6] slm[2]
Noise Level[4,7]	span <25slm span >25slm offset	2.0 [8] 3 [8] 0.1 [8]	2.5 5 0.2	% m.v[6] % m.v[6] slm[2]
Accuracy shift for deviation from Reference temperature 25°C	span offset	0.4 [8] 0.015 [8]	0.5 0.02	% m.v./10°C % m.v./10°C
Resolution (14bit)	span offset		0.07 0.034	% m.v. [6] slm[2]
Pressure drop	@ 60 slm @ 100 slm @ 200 slm	180 / 0.73 [8] 380 / 1.53 [8] 1400 / 5.62 [8]	230 / 0.93 550 / 2.21 1900 / 7.63	Pa / inH_2O Pa / inH_2O Pa / inH_2O

1.2 Ambient conditions

Parameter	Condition	Value	Unit
Calibrated Temperature Range	Dry gas	+10 … +50	°C
Operating Temperature Range[9]	10-95% rel. hum. (non cond.)	+5 … +50	°C
Extended Operating Range	Recommended to use on-sensor heater	-20…+5	°C
Storage Temperature	10-95% rel. hum. (non cond.)	-40 … +70	°C
Shelf Life for SFM3300-D	15°C - 35°C; 30 - 70 % rel. hum. storage in original packaging	3	years
Operating Pressure Range	absolute	0.54 – 1.1	bar
Burst Overpressure	gauge	0.3	bar

[1] Reference conditions are temperature = 25°C, absolute pressure = 966 mbar, horizontal flow and Vdd = 5V
[2] slm: mass flow measured in liters per minute at standard conditions (T = 20 °C, p = 1013.25 mbar)
[3] for "Max" no sensor measured outside of this limits will be shipped and a CpK of 1.33 is targeted
[4] For accuracy, noise level or resolution the total value is the sum of offset and span values
[5] This value corresponds to a CpK of 0.67 (95% of sensors within the “Typical” limit)
[6] %m.v. = % measured value = % of reading
[7] Noise level defined as standard deviation of individual sensor readings, measured at full sampling rate
[8] Average value
[9] Do not exceed these operating conditions by heating the sensor excessively with the external heater, see section 4.4.

1.3 Media compatibility

Parameter	Value
Calibration [10]	Air
Media Compatibility	Air (non-condensing), N_2, O_2, other non- aggressive gases
Wetted Materials –AW version	Si, Si_3N_4, SiO_x, Gold, Epoxy, PPSU, silicone, stainless steel
Wetted Materials –D version	Si, Si_3N_4, SiO_x, Gold, Epoxy, MABS, silicone
RoHS, REACH	RoHS and REACH compliant

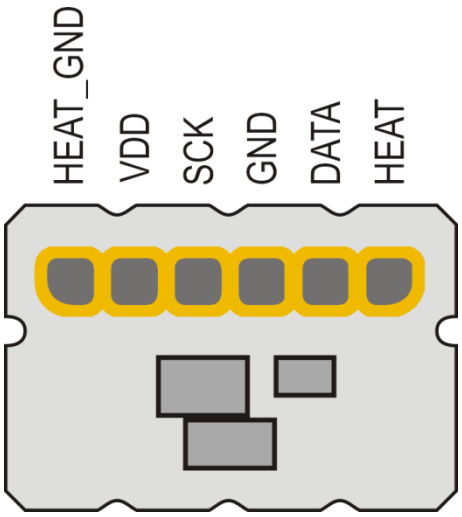
2. Electrical Specifications

2.1 Electrical characteristics

Electrical properties	Condition	Value		Unit
Interface		I^2C		
Default Sensor Address		64 (h40)		
Update Time	14 bit	0.5		ms
Soft Reset Time		80		ms
Start-up Time [11]	Max.	100		ms
I2C bus Clock Frequency	Max.	400		kHz
Supply Voltage		5V±5%		V
Communication Level		Min.	Max.	V
	High	2.5	VDD	
	Low	GND	1.1	
Power Consumption[12]		< 50		mW
Electrical Connector		See section 0 and 3.2		
External Heater Power Rating[13]	Max.	0.5		W
External Heater Resistance	Typ.	51		Ω
Output signal resolution[14]		14		bit
Scale Factor Flow	Air, N2	120		1/slm
Offset Flow		32768		

[10] Contact Sensirion for information about other gases, wider calibrated temperature ranges and higher storage temperatures.
[11] After 4.75V is reached
[12] When the heater resistor on the PCB is not in operation
[13] The heater's purpose is to avoid condensation or icing. One should only apply sufficient power to achieve this purpose. See section 4.4 for more details.
[14] 16 bit with two least significant bits always zero

2.2 Pad layout



2.3 Conversion to Physical Values

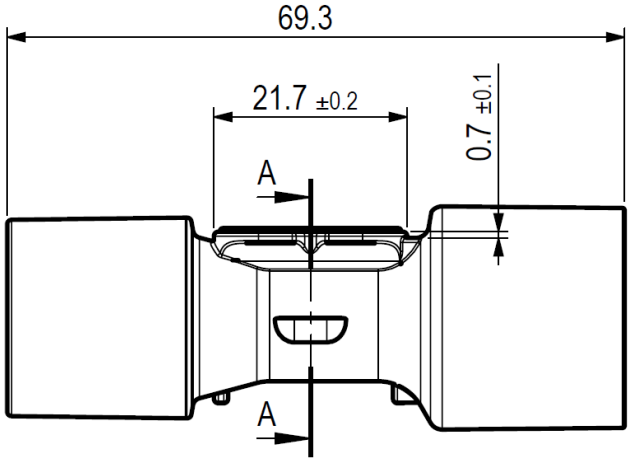
In order to obtain the measured flow in [slm], the measured value needs to be converted using the following formula:

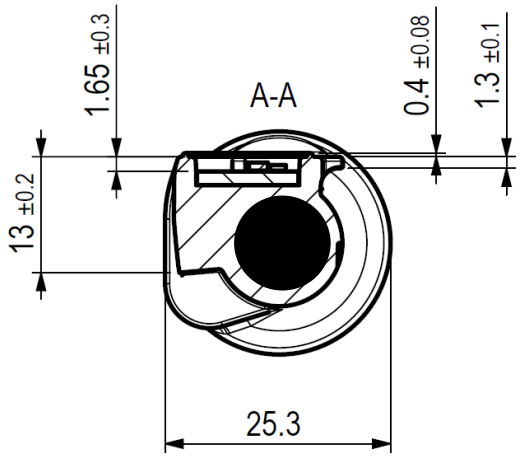
$$flow\ [\text{slm}] = \frac{measured\ value - offset\ flow}{scale\ factor\ flow}$$

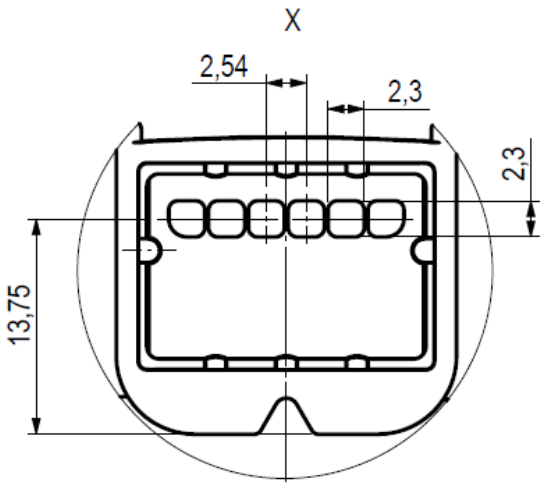
Please note that the first measurement performed directly after chip initialization is not valid.

3. Mechanical Specifications

All dimensions are in millimeters (mm).







3.1 Mechanical fitting

Fittings of the SFM3300 series sensors correspond to the international standard ISO5356-1:2004. Details about this type of connection can be found in the description of the standard.

3.2 Mechanical / Electrical Interface

SFM3300 series sensors have been designed for use in an proximal/expiratory environment. Therefore the sensor connector has been designed for an easy and reliable connection and disconnection. The connector itself is not provided as a standard product by Sensirion but Sensirion can help with an application note including design recommendations.

Dimension	Condition	Value	Unit
Length		69.3	mm
Diameter Flow channel		11.25	mm
Proximal medical cone	Inner	15	mm
Proximal medical socket	Outer	22	mm
Distal medical socket	Inner	22	mm
Distal medical cone	Outer	15	mm
Dead space		<10	ml
Weight		<20	g

Annexe C

Informations constructeur — Turbine WS7040

Les caractéristiques techniques de la turbine centrifuge WS7040 ne font pas l'objet d'une datasheet officielle au format standard. Les informations du constructeur (Wonsmartmotor) sont disponibles à l'adresse suivante :

`https://www.wonsmartmotor.com/news/ws7040-centrifugal-brushless-dc-blower/`

Les caractéristiques principales utilisées dans ce travail sont :

- Tension d'alimentation : 12V ou 24V
- Type : turbine centrifuge brushless
- Commande : signal PWM
- Application : ventilation forcée haute pression

Annexe D

Code Arduino — Validation du capteur de débit SFM3300-D

Ce code a été utilisé pour valider le fonctionnement du capteur de débit SFM3300-D à débit nul, c'est-à-dire avec les deux orifices du capteur ouverts à l'atmosphère en l'absence de tout flux d'air. Il effectue 3000 lectures sur une durée de 5 minutes et calcule automatiquement un résumé statistique en fin d'acquisition (offset, écart-type, minimum, maximum).

Matériel requis :

- Arduino Mega 2560
- Capteur de débit SFM3300-D (Sensirion)

Connexions :

- SFM3300-D : VDD → 5V, GND → GND, SDA → pin 20, SCL → pin 21
- Les deux orifices du capteur sont laissés ouverts à l'atmosphère

Format de sortie :

- Données CSV : `temps_ms, valeur_brute, debit_Lmin`
- Résumé statistique automatique en fin de test

Ce code a été généré par IA (Claude code).

```
/*
 * Ce code a ete genere par Claude code
 * TEST 1 -- Validation du capteur SFM3300-D a debit nul
 * Memoire : Dispositif d'entrainement respiratoire
 *
 * Lit le capteur SFM3300-D toutes les 100 ms pendant 5 minutes
 * et envoie les donnees sur le port serie au format CSV.
 */

#include <Wire.h>

```

```
#define SFM3300_ADDR    0x40     // Adresse I2C du SFM3300-D
#define SCALE_FACTOR    120      // Facteur d'echelle pour l'air (Sensirion)
#define OFFSET_DEFAULT  32768    // Offset nominal

#define INTERVAL_MS     100      // Lecture toutes les 100 ms
#define DUREE_TEST_S    300      // Duree du test : 5 minutes

unsigned long tempsDebut     = 0;
unsigned long derniereLecture = 0;
bool capteurOK               = false;

long     somme          = 0;
long     sommeCarres    = 0;
int      nombreLectures = 0;
uint16_t valeurMin      = 65535;
uint16_t valeurMax      = 0;

void setup() {
  Serial.begin(115200);
  Wire.begin();
  Wire.setClock(50000);  // 50 kHz pour robustesse

  Serial.println("temps_ms,valeur_brute,debit_Lmin");

  capteurOK = initCapteur();

  if (!capteurOK) {
    Serial.println("# ERREUR : capteur non detecte.");
    while (true) delay(1000);
  }

  Serial.println("# Capteur initialise. Acquisition dans 2s...");
  delay(2000);

  tempsDebut      = millis();
  derniereLecture = millis();
}

void loop() {
  unsigned long maintenant  = millis();
  unsigned long tempsEcoule = maintenant - tempsDebut;

  if (tempsEcoule > (unsigned long)DUREE_TEST_S * 1000) {
    afficherStatistiques();
    while (true) delay(1000);
  }

  if (maintenant - derniereLecture >= INTERVAL_MS) {
    derniereLecture = maintenant;

    uint16_t valeurBrute = lireCapteur();
```

```
63      if (valeurBrute == 0xFFFF) {
64        Serial.println("# ERREUR de lecture I2C");
65        return;
66      }
67
68      float debit = ((float)valeurBrute - OFFSET_DEFAULT) /
            SCALE_FACTOR;
69
70      Serial.print(tempsEcoule); Serial.print(",");
71      Serial.print(valeurBrute); Serial.print(",");
72      Serial.println(debit, 4);
73
74      somme       += valeurBrute;
75      sommeCarres += (long)valeurBrute * valeurBrute;
76      nombreLectures++;
77      if (valeurBrute < valeurMin) valeurMin = valeurBrute;
78      if (valeurBrute > valeurMax) valeurMax = valeurBrute;
79    }
80  }
81
82  bool initCapteur() {
83    // Soft reset
84    Wire.beginTransmission(SFM3300_ADDR);
85    Wire.write(0x20); Wire.write(0x00);
86    Wire.endTransmission();
87    delay(500);
88
89    // Commande de mesure continue
90    Wire.beginTransmission(SFM3300_ADDR);
91    Wire.write(0x10); Wire.write(0x00);
92    byte erreur = Wire.endTransmission();
93    if (erreur != 0) return false;
94    delay(500);
95    return true;
96  }
97
98  uint16_t lireCapteur() {
99    Wire.requestFrom(SFM3300_ADDR, 3);
100   if (Wire.available() < 3) return 0xFFFF;
101   byte msb = Wire.read();
102   byte lsb = Wire.read();
103   Wire.read();  // CRC ignore
104   return ((uint16_t)msb << 8) | lsb;
105 }
106
107 void afficherStatistiques() {
108   if (nombreLectures == 0) return;
109
110   float moyenne   = (float)somme / nombreLectures;
111   float variance  = ((float)sommeCarres / nombreLectures)
112                     - (moyenne * moyenne);
113   float ecartType = sqrt(variance);
114   float debitMin  = ((float)valeurMin - OFFSET_DEFAULT) /
```

```
    SCALE_FACTOR;
115   float debitMax  = ((float)valeurMax - OFFSET_DEFAULT) /
    SCALE_FACTOR;
116
117   Serial.println("# RESUME STATISTIQUE");
118   Serial.print("# Nombre de lectures      : ");
119   Serial.println(nombreLectures);
120   Serial.print("# Offset mesure (moyenne) : ");
121   Serial.print(moyenne, 2); Serial.println(" LSB");
122   Serial.print("# Ecart-type (bruit)      : ");
123   Serial.print(ecartType, 2); Serial.println(" LSB");
124   Serial.print("# Valeur min              : "); Serial.println(
    valeurMin);
125   Serial.print("# Valeur max              : "); Serial.println(
    valeurMax);
126   Serial.print("# Debit equivalent min    : ");
127   Serial.print(debitMin, 4); Serial.println(" L/min");
128   Serial.print("# Debit equivalent max    : ");
129   Serial.print(debitMax, 4); Serial.println(" L/min");
130 }
```

Listing D.1 – Sketch Arduino — Validation du capteur de débit SFM3300-D à débit nul

Annexe E

Code Arduino — Validation du capteur de pression MPX5010DP

Ce code a été utilisé pour valider le fonctionnement du capteur de pression MPX5010DP via le convertisseur analogique-numérique ADS1115, à pression nulle (orifices P1 et P2 ouverts à l'atmosphère). Il effectue 600 lectures sur une durée de 2 minutes et calcule automatiquement un résumé statistique en fin d'acquisition (moyenne, écart-type, minimum, maximum).

Matériel requis :

- Arduino Mega 2560
- Capteur de pression MPX5010DP (NXP Semiconductors)
- Convertisseur ADC 16 bits ADS1115
- Librairie Arduino : `Adafruit ADS1X15`

Connexions :

- ADS1115 : VDD → 5V, GND → GND, SDA → pin 20, SCL → pin 21, ADDR → GND (adresse I^2C = 0x48)
- MPX5010DP : broche 1 (Vout) → A0 de l'ADS1115, broche 2 (GND) → GND, broche 3 (Vs) → 5V

Ce code a été généré par IA (Claude code).

```
/*
 * Ce code a ete genere par Claude code
 * VALIDATION MPX5010DP via ADS1115 -- Lecture a pression nulle
 * Memoire : Dispositif d'entrainement respiratoire
 *
 * Formule MPX5010DP (datasheet Freescale) :
 *   P(kPa)   = (Vout / Vcc - 0.04) / 0.09
 *   P(cmH2O) = P(kPa) x 101.97
 *   Plage    : 0 a 10 kPa = 0 a 1019 cmH2O
 *   Sortie   : 0.2V (0 kPa) a 4.7V (10 kPa) sous Vcc = 5V
```

```
 */

#include <Wire.h>
#include <Adafruit_ADS1X15.h>

Adafruit_ADS1115 ads;

#define CANAL_MPX     0           // Canal A0 de l'ADS1115
#define VCC           5.0f        // Tension alimentation MPX5010DP (V)
#define GAIN_ADS      GAIN_ONE    // +/-4.096V
#define LSB_GAIN_ONE  0.0001250f  // 1 LSB = 0.125 mV en GAIN_ONE
#define INTERVAL_MS   200         // Lecture toutes les 200 ms
#define DUREE_TEST_S  120         // Duree du test : 2 minutes

float somme = 0, sommeCarres = 0;
int   nbLectures = 0;
float pMin = 9999, pMax = -9999;

void setup() {
  Serial.begin(115200);
  Wire.begin();
  Wire.setClock(50000);

  if (!ads.begin(0x48)) {
    Serial.println("# ERREUR : ADS1115 non detecte.");
    while (true) delay(1000);
  }
  ads.setGain(GAIN_ADS);

  Serial.println("temps_ms,raw_ADS,vout_V,pression_kPa,pression_cmH2O");
  delay(2000);
}

void loop() {
  static unsigned long tempsDebut     = millis();
  static unsigned long derniereLecture = millis();
  unsigned long maintenant  = millis();
  unsigned long tempsEcoule = maintenant - tempsDebut;

  if (tempsEcoule > (unsigned long)DUREE_TEST_S * 1000) {
    afficherStatistiques();
    while (true) delay(1000);
  }

  if (maintenant - dernierLecture >= INTERVAL_MS) {
    dernierLecture = maintenant;

    int16_t raw  = ads.readADC_SingleEnded(CANAL_MPX);
    float   vout = raw * LSB_GAIN_ONE;
    float   pression_kPa   = (vout / VCC - 0.04f) / 0.09f;
```

```
61      float   pression_cmH2O = pression_kPa * 101.97f;
62
63      Serial.print(tempsEcoule);      Serial.print(",");
64      Serial.print(raw);              Serial.print(",");
65      Serial.print(vout, 5);          Serial.print(",");
66      Serial.print(pression_kPa, 5); Serial.print(",");
67      Serial.println(pression_cmH2O, 3);
68
69      somme       += pression_cmH2O;
70      sommeCarres += pression_cmH2O * pression_cmH2O;
71      nbLectures++;
72      if (pression_cmH2O < pMin) pMin = pression_cmH2O;
73      if (pression_cmH2O > pMax) pMax = pression_cmH2O;
74    }
75  }
76
77  void afficherStatistiques() {
78    if (nbLectures == 0) return;
79    float moyenne   = somme / nbLectures;
80    float variance  = (sommeCarres / nbLectures) - (moyenne *
        moyenne);
81    float ecartType = sqrt(max(0.0f, variance));
82
83    Serial.println("# RESUME STATISTIQUE");
84    Serial.print("# Nb lectures      : "); Serial.println(
        nbLectures);
85    Serial.print("# Pression moyenne : "); Serial.print(moyenne,
        3);
86                                          Serial.println(" cmH2O
                                              ");
87    Serial.print("# Ecart-type       : "); Serial.print(ecartType
        , 3);
88                                          Serial.println(" cmH2O
                                              ");
89    Serial.print("# Pression min     : "); Serial.print(pMin, 3);
90                                          Serial.println(" cmH2O
                                              ");
91    Serial.print("# Pression max     : "); Serial.print(pMax, 3);
92                                          Serial.println(" cmH2O
                                              ");
93  }
```

Listing E.1 – Sketch Arduino — Validation du capteur MPX5010DP à pression nulle

Annexe F

Code Arduino — Caractérisation débit vs PWM

Ce code a été utilisé pour caractériser la relation entre le signal de commande PWM et le débit d'air généré par la turbine WS7040, mesuré par le capteur de débit SFM3300-D en circuit ouvert. Il propose deux modes de fonctionnement : un mode manuel permettant d'explorer librement la plage de PWM, et un mode automatique qui parcourt une série de paliers prédéfinis et enregistre les données au format CSV.

Matériel requis :

- Arduino Mega 2560
- Turbine WS7040 et son ESC
- Capteur de débit SFM3300-D (Sensirion)
- Alimentation 24V

Connexions :

- ESC : signal → pin 9 (PWM)
- SFM3300-D : VDD → 5V, GND → GND, SDA → pin 20, SCL → pin 21

Commandes série disponibles :

- `p` / `m` : augmenter / diminuer le PWM de 1
- `P` / `M` : augmenter / diminuer le PWM de 10
- `0` : arrêt de la turbine
- `A` : lancer la séquence automatique de paliers (génère les données CSV)

Photo du montage expérimental :

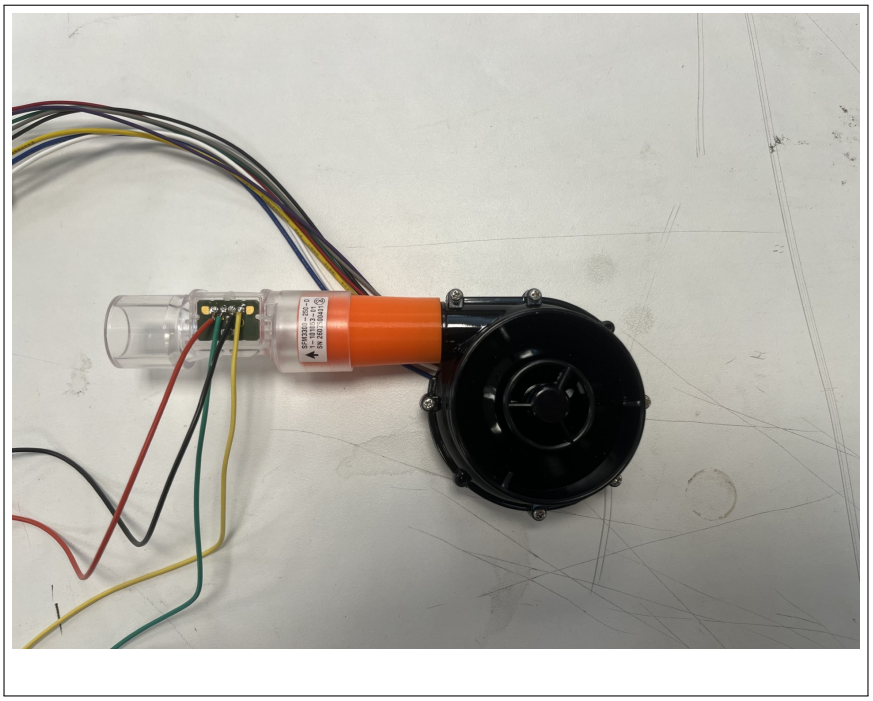


FIGURE F.1 – Montage expérimental pour la caractérisation débit vs PWM

Ce code a été généré par IA (Claude code).

```
/*
 * Ce code a ete genere par Claude code
 * CARACTERISATION DEBIT vs PWM -- Turbine WS7040 + SFM3300-D
 * Memoire : Dispositif d'entrainement respiratoire
 *
 * MODE MANUEL : p/m (+/-1 PWM), P/M (+/-10 PWM), 0 (stop)
 * MODE AUTO   : A -> sequence automatique de paliers -> CSV
 */

#include <Wire.h>

#define SFM3300_ADDR    0x40
#define SCALE_FACTOR    120
#define OFFSET          32768

#define PIN_PWM         9
#define PWM_MIN         0
#define PWM_MAX         255

#define NB_MESURES      30
#define DELAI_PALIER_MS 4000
#define DELAI_MESURE_MS 50

const int NB_PALIERS = 10;
const int PALIERS[NB_PALIERS] = {
    0, 50, 75, 100, 125, 150, 175, 200, 225, 255
```

```
};

int  pwmValue  = 0;
bool capteurOK = false;

void setup() {
  pinMode(PIN_PWM, OUTPUT);
  analogWrite(PIN_PWM, 0);
  Serial.begin(115200);
  Wire.begin();
  Wire.setClock(50000);
  capteurOK = initCapteur();
  if (!capteurOK)
    Serial.println("# ERREUR : SFM3300-D non detecte.");
}

void loop() {
  if (Serial.available()) {
    char c = Serial.read();
    switch (c) {
      case 'p': ajusterPWM(+1);    break;
      case 'm': ajusterPWM(-1);    break;
      case 'P': ajusterPWM(+10);   break;
      case 'M': ajusterPWM(-10);   break;
      case '0': setPWM(0);         break;
      case 'A': modeAutomatique(); break;
    }
  }
  static unsigned long dernierAffichage = 0;
  if (millis() - dernierAffichage >= 200) {
    dernierAffichage = millis();
    if (capteurOK && pwmValue > 0) {
      float debit = lireDebit();
      if (debit > -900) {
        Serial.print("PWM="); Serial.print(pwmValue);
        Serial.print("  Debit="); Serial.print(debit, 2);
        Serial.println(" L/min");
      }
    }
  }
}

void modeAutomatique() {
  Serial.println("pwm_raw,pwm_pct,debit_moy_Lmin,"
                 "ecarttype_Lmin,debit_min_Lmin,debit_max_Lmin"
                   );
  for (int i = 0; i < NB_PALIERS; i++) {
    int palier     = PALIERS[i];
    int palier_pct = map(palier, PWM_MIN, PWM_MAX, 0, 100);
    setPWM(palier);
    delay(DELAI_PALIER_MS);

    float somme = 0, sommeCarres = 0;
```

```
    float dMin = 9999, dMax = -9999;
    int   nbValides = 0;

    for (int m = 0; m < NB_MESURES; m++) {
      float d = lireDebit();
      if (d > -900) {
        somme       += d;
        sommeCarres += d * d;
        if (d < dMin) dMin = d;
        if (d > dMax) dMax = d;
        nbValides++;
      }
      delay(DELAI_MESURE_MS);
    }

    if (nbValides == 0) continue;

    float moy = somme / nbValides;
    float var = (sommeCarres / nbValides) - (moy * moy);
    float std = sqrt(max(0.0f, var));

    Serial.print(palier);     Serial.print(",");
    Serial.print(palier_pct); Serial.print(",");
    Serial.print(moy, 3);     Serial.print(",");
    Serial.print(std, 3);     Serial.print(",");
    Serial.print(dMin, 3);    Serial.print(",");
    Serial.println(dMax, 3);
  }
  setPWM(0);
}

void ajusterPWM(int delta) {
  setPWM(constrain(pwmValue + delta, PWM_MIN, PWM_MAX));
}

void setPWM(int val) {
  pwmValue = constrain(val, PWM_MIN, PWM_MAX);
  analogWrite(PIN_PWM, pwmValue);
}

bool initCapteur() {
  Wire.beginTransmission(SFM3300_ADDR);
  Wire.write(0x20); Wire.write(0x00);
  Wire.endTransmission();
  delay(500);
  Wire.beginTransmission(SFM3300_ADDR);
  Wire.write(0x10); Wire.write(0x00);
  byte err = Wire.endTransmission();
  if (err != 0) return false;
  delay(500);
  return true;
}

```

```
132 float lireDebit() {
133   Wire.requestFrom(SFM3300_ADDR, 3);
134   if (Wire.available() < 3) return -9999.0;
135   byte msb = Wire.read();
136   byte lsb = Wire.read();
137   Wire.read();
138   uint16_t raw = ((uint16_t)msb << 8) | lsb;
139   return ((float)raw - OFFSET) / SCALE_FACTOR;
140 }
```

Listing F.1 – Sketch Arduino — Caractérisation débit vs PWM

Annexe G

Code Arduino — Caractérisation pression vs PWM

Ce code a été utilisé pour caractériser la relation entre le signal de commande PWM et la pression statique générée par la turbine WS7040, mesurée par le capteur MPX5010DP via l'ADS1115, en circuit fermé. Le circuit est obturé en bout de tube, et la pression est relevée pour chaque palier de PWM. Le code permet de réaliser jusqu'à trois séries de mesures successives afin d'évaluer la répétabilité.

Matériel requis :

- Arduino Mega 2560
- Turbine WS7040 et son ESC
- Capteur de pression MPX5010DP (NXP Semiconductors)
- Convertisseur ADC 16 bits ADS1115
- Capteur de débit proximal iFlow 200S
- Alimentation 24V
- Librairie Arduino : `Adafruit ADS1X15`

Connexions :

- ESC : signal → pin 9 (PWM)
- ADS1115 : VDD → 5V, GND → GND, SDA → pin 20, SCL → pin 21, ADDR → GND (adresse I^2C = 0x48)
- MPX5010DP : broche 1 (Vout) → A0 de l'ADS1115, broche 2 (GND) → GND, broche 3 (Vs) → 5V
- P1 du MPX5010DP → prise de pression de l'iFlow 200S
- P2 du MPX5010DP → ouvert à l'atmosphère

Commandes série disponibles :

- `s` : lancer une série de mesures
- `x` : arrêt d'urgence (turbine coupée immédiatement)

Photo du montage expérimental :

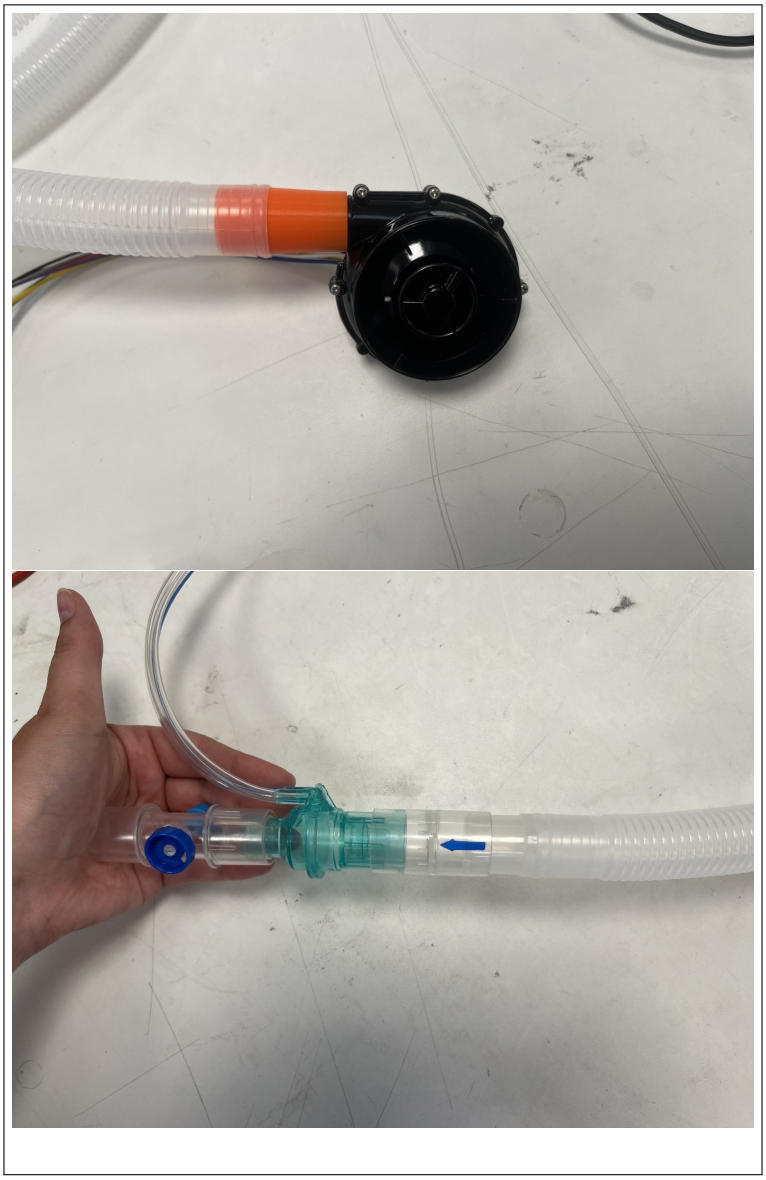


FIGURE G.1 – Montage expérimental pour la caractérisation pression vs PWM

Ce code a été généré par IA (Claude code).

```
/*
 * Ce code a ete genere par Claude code
 * TEST 3 -- Caracterisation pression vs PWM
 * Memoire : Dispositif d'entrainement respiratoire
 *
 * Circuit ferme en bout de tube (obture manuellement).
 * Commandes : 's' -> lancer une serie | 'x' -> arret urgence
```

```
 */

#include <Wire.h>
#include <Adafruit_ADS1X15.h>

Adafruit_ADS1115 ads;

#define CANAL_MPX         0
#define VCC               5.0f
#define LSB_GAIN_ONE      0.0001250f
#define OFFSET_PRESSION   0.052f      // cmH2O -- mesure lors de la validation

#define PIN_PWM           9
#define NB_MESURES        20          // Mesures moyennees par palier
#define DELAI_PALIER_MS   3000        // Stabilisation a chaque palier (ms)
#define DELAI_MESURE_MS   50          // Intervalle entre deux mesures (ms)

const int NB_PALIERS = 14;
const int PALIERS[NB_PALIERS] = {
    0, 31, 40, 50, 60, 75, 90, 100, 115, 125, 140, 150, 163, 175
};

int numeroSerie = 1;

void setup() {
  Serial.begin(115200);
  pinMode(PIN_PWM, OUTPUT);
  analogWrite(PIN_PWM, 0);
  Wire.begin();
  Wire.setClock(50000);
  delay(2000);

  if (!ads.begin(0x48)) {
    Serial.println("# ERREUR : ADS1115 non detecte !");
    while (true) delay(1000);
  }
  ads.setGain(GAIN_ONE);

  Serial.println("serie,pwm_raw,pwm_pct,pression_moy_cmH2O,"
                 "ecarttype_cmH2O,pression_min,pression_max");
  Serial.println("# Tapez 's' pour lancer une serie.");
}

void loop() {
  if (Serial.available()) {
    char c = Serial.read();
    if (c == 's') {
      lancerSerie(numeroSerie);
```

```
      numeroSerie++;
      if (numeroSerie > 3) {
        Serial.println("# 3 series effectuees. Test termine.");
        analogWrite(PIN_PWM, 0);
        while (true) delay(1000);
      }
    }
    if (c == 'x') {
      analogWrite(PIN_PWM, 0);
      Serial.println("# ARRET D'URGENCE.");
      while (true) delay(1000);
    }
  }
}

void lancerSerie(int serie) {
  for (int i = 0; i < NB_PALIERS; i++) {
    int pwm     = PALIERS[i];
    int pwm_pct = map(pwm, 0, 255, 0, 100);

    analogWrite(PIN_PWM, pwm);
    delay(DELAI_PALIER_MS);

    float somme = 0, sommeCarres = 0;
    float pMin  =  9999, pMax = -9999;

    for (int m = 0; m < NB_MESURES; m++) {
      float p = lirePression();
      somme       += p;
      sommeCarres += p * p;
      if (p < pMin) pMin = p;
      if (p > pMax) pMax = p;
      delay(DELAI_MESURE_MS);
    }

    float moy = somme / NB_MESURES;
    float var = (sommeCarres / NB_MESURES) - (moy * moy);
    float std = sqrt(max(0.0f, var));

    Serial.print(serie);    Serial.print(",");
    Serial.print(pwm);      Serial.print(",");
    Serial.print(pwm_pct);  Serial.print(",");
    Serial.print(moy, 3);   Serial.print(",");
    Serial.print(std, 3);   Serial.print(",");
    Serial.print(pMin, 3);  Serial.print(",");
    Serial.println(pMax, 3);
  }
  analogWrite(PIN_PWM, 0);
  delay(2000);
}

float lirePression() {
  static float pression_filtree = 0;
```

```
109   const  float alpha = 0.2f;
110   int16_t raw  = ads.readADC_SingleEnded(CANAL_MPX);
111   float   vout = raw * LSB_GAIN_ONE;
112   float   p    = (vout / VCC - 0.04f) / 0.09f * 10.197f -
          OFFSET_PRESSION;
113   pression_filtree = alpha * p + (1.0f - alpha) *
          pression_filtree;
114   return pression_filtree;
115 }
```

Listing G.1 – Sketch Arduino — Caractérisation pression vs PWM

Annexe H

Code Arduino — Caractérisation du temps de réponse de la turbine

Ce code a été utilisé pour mesurer le temps de réponse en pression de la turbine WS7040 lors d'essais en échelon de PWM. À partir d'un état de repos (PWM = 0), un signal PWM constant est appliqué instantanément et la réponse en pression est enregistrée toutes les 50 ms pendant 5 secondes. Trois amplitudes d'échelon sont disponibles ($0 \rightarrow 75$, $0 \rightarrow 125$ et $0 \rightarrow 175$), chacune répétée automatiquement 3 fois. Le montage est identique à celui du test de caractérisation pression vs PWM : circuit fermé, extrémité du tube obturée manuellement.

Matériel requis :

- Arduino Mega 2560
- Turbine WS7040 et son ESC
- Capteur de pression MPX5010DP (NXP Semiconductors)
- Convertisseur ADC 16 bits ADS1115
- Capteur de débit proximal iFlow 200S
- Alimentation 24V
- Librairie Arduino : `Adafruit ADS1X15`

Connexions :

- Identiques à l'annexe G (caractérisation pression vs PWM)

Commandes série disponibles :

- `1` : lancer l'échelon $0 \rightarrow 75$ (basse pression)
- `2` : lancer l'échelon $0 \rightarrow 125$ (pression moyenne)
- `3` : lancer l'échelon $0 \rightarrow 175$ (pression haute)
- `x` : arrêt d'urgence (turbine coupée immédiatement)

Photo du montage expérimental :

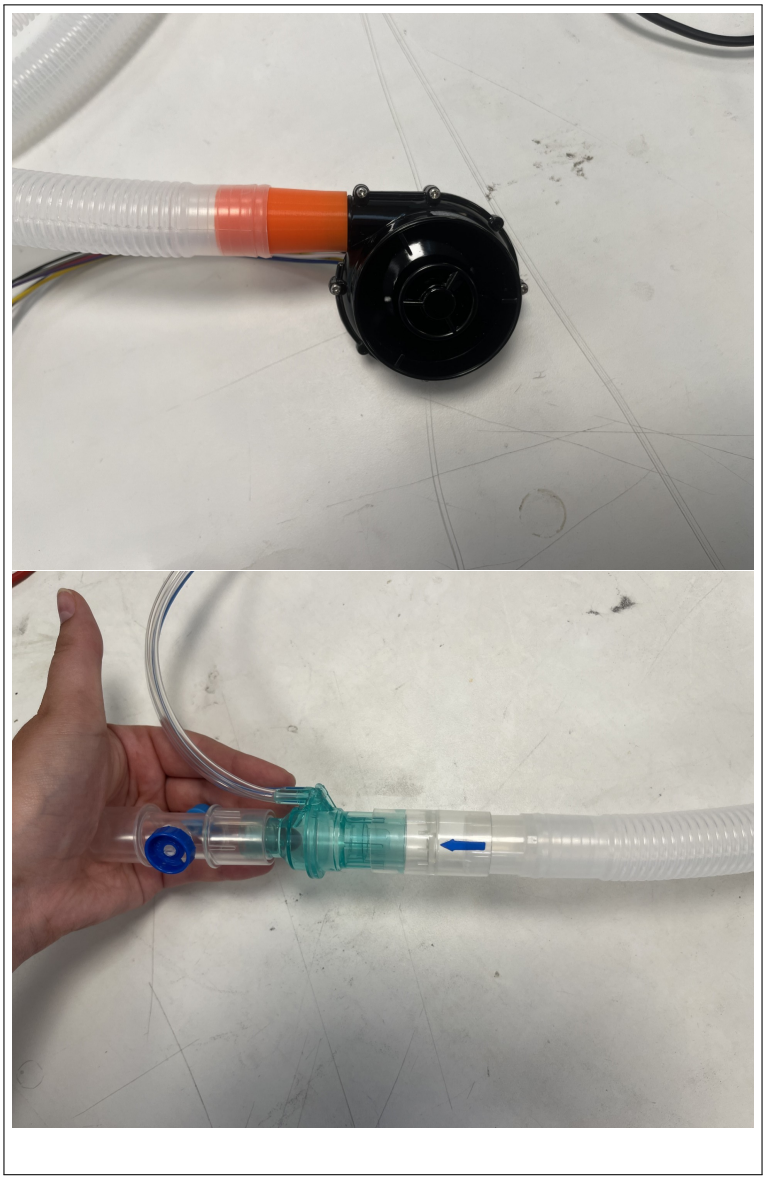


FIGURE H.1 – Montage expérimental pour la caractérisation du temps de réponse

Ce code a été généré par IA (Claude code).

```
/*
 * Ce code a ete genere par Claude code
 * TEST 4 -- Temps de reponse de la turbine WS7040
```

```
 * Memoire : Dispositif d'entrainement respiratoire
 *
 * Circuit ferme en bout de tube (obture manuellement).
 * Commandes : '1' -> echelon 0->75  | '2' -> echelon 0->125
 *             '3' -> echelon 0->175 | 'x' -> arret urgence
 * Chaque echelon est repete 3 fois automatiquement.
 */

#include <Wire.h>
#include <Adafruit_ADS1X15.h>

Adafruit_ADS1115 ads;

#define CANAL_MPX         0
#define VCC               5.0f
#define LSB_GAIN_ONE      0.0001250f
#define OFFSET_PRESSION   0.052f

#define PIN_PWM           9
#define DUREE_ECHELON_MS  5000        // Duree d'enregistrement (ms)
#define DELAI_MESURE_MS   50          // Lecture toutes les 50ms
#define DELAI_REPOS_MS    5000        // Repos entre repetitions (ms)
#define NB_REPETITIONS    3

const int  ECHELONS[3] = {75, 125, 175};

float pressionFiltree = 0;

void setup() {
  Serial.begin(115200);
  pinMode(PIN_PWM, OUTPUT);
  analogWrite(PIN_PWM, 0);
  Wire.begin();
  Wire.setClock(50000);
  delay(2000);

  if (!ads.begin(0x48)) {
    Serial.println("# ERREUR : ADS1115 non detecte !");
    while (true) delay(1000);
  }
  ads.setGain(GAIN_ONE);

  Serial.println("echelon,repetition,temps_ms,pression_cmH2O");
  Serial.println("# Tapez '1', '2' ou '3' pour lancer un echelon.");
}

void loop() {
  if (Serial.available()) {
    char c = Serial.read();
    if (c >= '1' && c <= '3') lancerEchelon(c - '1');
```

```
    if (c == 'x') {
      analogWrite(PIN_PWM, 0);
      while (true) delay(1000);
    }
  }
}

void lancerEchelon(int idx) {
  int pwm_cible = ECHELONS[idx];

  for (int rep = 1; rep <= NB_REPETITIONS; rep++) {
    analogWrite(PIN_PWM, 0);
    delay(DELAI_REPOS_MS);
    reinitFiltre();

    unsigned long t0 = millis();
    analogWrite(PIN_PWM, pwm_cible);

    unsigned long derniereLecture = millis();
    while (millis() - t0 < DUREE_ECHELON_MS) {
      if (millis() - derniereLecture >= DELAI_MESURE_MS) {
        derniereLecture = millis();
        float p = lirePression();
        unsigned long t = millis() - t0;
        Serial.print(pwm_cible); Serial.print(",");
        Serial.print(rep);       Serial.print(",");
        Serial.print(t);         Serial.print(",");
        Serial.println(p, 3);
      }
    }
    analogWrite(PIN_PWM, 0);
  }
}

void reinitFiltre() {
  for (int i = 0; i < 10; i++) {
    int16_t raw  = ads.readADC_SingleEnded(CANAL_MPX);
    float   vout = raw * LSB_GAIN_ONE;
    float   p    = (vout / VCC - 0.04f) / 0.09f * 101.97f
                   - OFFSET_PRESSION;
    pressionFiltree = p;
    delay(20);
  }
}

float lirePression() {
  const float alpha = 0.2f;
  int16_t raw  = ads.readADC_SingleEnded(CANAL_MPX);
  float   vout = raw * LSB_GAIN_ONE;
  float   p    = (vout / VCC - 0.04f) / 0.09f * 10.197f
                 - OFFSET_PRESSION;
  pressionFiltree = alpha * p + (1.0f - alpha) *
     pressionFiltree;
```

```
106   return pressionFiltree;
107 }
```

Listing H.1 – Sketch Arduino — Caractérisation du temps de réponse de la turbine

Annexe I

Code Arduino — Régulateur PID v1 (tests en laboratoire)

Ce code implémente le cycle respiratoire complet du dispositif d'entraînement, avec un régulateur PID actif pendant les phases d'inspiration et d'apnée. Il a été utilisé pour l'ensemble des tests de validation en laboratoire, sur les quatre pressions cibles (5, 10, 15 et 20 cmH_2O).

Matériel requis :

- Arduino Mega 2560
- Turbine WS7040 et son ESC
- Capteur de pression MPX5010DP (NXP Semiconductors)
- Convertisseur ADC 16 bits ADS1115
- Capteur de débit proximal iFlow 200S
- Valve de fuite réglable
- Filtre HMEF Hygrovent S
- Masque facial de ventilation non invasive
- Alimentation 24V
- Librairie Arduino : `Adafruit ADS1X15`

Connexions :

- ESC : signal → pin 9 (PWM)
- ADS1115 : VDD → 5V, GND → GND, SDA → pin 20, SCL → pin 21, ADDR → GND (adresse I^2C = 0x48)
- MPX5010DP : broche 1 (Vout) → A0 de l'ADS1115, broche 2 (GND) → GND, broche 3 (Vs) → 5V

Commandes série disponibles :

- `s` : démarrer la séquence de cycles
- `x` : arrêt d'urgence immédiat (turbine coupée)
- `p` : afficher les paramètres actuels

Format de sortie CSV :

```
temps_ms, cycle, etat, pression_cmH2O, debit_Lmin, pwm
```

où `etat` vaut 1 (inspiration), 2 (apnée) ou 3 (expiration).

Paramètres modifiables en tête de code :

- `PRESSION_CIBLE_cmH2O` : pression cible en cmH_2O
- `NB_CYCLES` : nombre de cycles par session
- `DUREE_APNEE_S` : durée du plateau en secondes
- `DUREE_EXPIRATION_S` : durée de l'expiration en secondes
- `Kp`, `Ki`, `Kd` : gains du régulateur PID
- `PRESSION_MAX_SECURITE` : seuil de sécurité en cmH_2O

Paramètres utilisés lors des tests en laboratoire :

- Kp = 2.0, Ki = 1.3, Kd = 0.0
- Précharge intégrale : `INTEGRALE_PRECHARG` = 60
- Filtre passe-bas : $\alpha = 0.2$
- Tolérance de convergence : ± 0.5 cmH_2O pendant 500 ms
- PWM_MAX = 200

Ce code a été généré par IA (Claude code).

```
#Ce code a ete genere par Claude code
#include <Wire.h>
#include <Adafruit_ADS1X15.h>

//         Parametres modifiables


float PRESSION_CIBLE_cmH2O = 20.0;
int   NB_CYCLES            = 3;
float DUREE_APNEE_S        = 20.0;
float DUREE_EXPIRATION_S   = 5.0;
float PRESSION_MAX_SECURITE = 25.0;

float Kp = 2.0;
float Ki = 1.3;
float Kd = 0.0;

#define PERIODE_PID_MS    50
#define INTEGRALE_MAX    100.0
#define INTEGRALE_PRECHARG 60.0

float TOLERANCE_cmH2O  = 0.5;
int   TEMPS_STABLE_MS  = 500;
float ALPHA_FILTRE     = 0.2;

//         Constantes
```

```
#define PIN_PWM          9
#define PWM_MIN          31
#define PWM_MAX          200
#define CANAL_MPX        0
#define VCC_MPX          5.0f
#define LSB_GAIN_ONE     0.0001250f
#define OFFSET_PRESSION  0.0520f
#define SFM3300_ADDR     0x40
#define SCALE_FACTOR     120
#define OFFSET_DEBIT     32768

enum Etat {
  ATTENTE, INSPIRATION, APNEE, EXPIRATION, TERMINE, URGENCE
};

Adafruit_ADS1115 ads;
Etat etatCourant = ATTENTE;
int cycleCourant = 0;
float erreurPrecedente = 0, integrale = 0, pressionFiltree = 0;
unsigned long tPrecedentPID = 0, tDebutPhase = 0, tStable = 0;
bool stableDetecte = false, capteurSFM_OK = false;

void setup() {
  Serial.begin(115200);
  pinMode(PIN_PWM, OUTPUT);
  analogWrite(PIN_PWM, 0);
  Wire.begin();
  Wire.setClock(50000);
  delay(2000);
  if (!ads.begin(0x48)) {
    Serial.println("# ERREUR : ADS1115 non detecte !");
    while (true) delay(1000);
  }
  ads.setGain(GAIN_ONE);
  Serial.println("# ADS1115 OK.");
  capteurSFM_OK = initSFM3300();
  if (capteurSFM_OK) {
    Serial.println("# SFM3300-D OK.");
  } else {
    Serial.println("# ATTENTION : SFM3300-D non detecte. Debit
       non mesure.");
  }
  Serial.println("temps_ms,cycle,etat,pression_cmH2O,debit_Lmin
     ,pwm");
  afficherParametres();
  Serial.println("# Tapez 's' pour demarrer, 'x' pour arret d'
     urgence.");
}

void loop() {
  if (Serial.available()) {
    char c = Serial.read();
    if (c == 'x') { arretUrgence(); return; }
```

```
75      if (c == 's' && etatCourant == ATTENTE) demarrerSequence();
76      if (c == 'p') afficherParametres();
77    }
78    switch (etatCourant) {
79      case INSPIRATION: gererInspiration(); break;
80      case APNEE:       gererApnee();       break;
81      case EXPIRATION:  gererExpiration();  break;
82      default: break;
83    }
84  }
85
86  void gererInspiration() {
87    unsigned long maintenant = millis();
88    float pression = lirePression();
89    if (pression > PRESSION_MAX_SECURITE) {
90      Serial.println("# SECURITE : pression maximale depassee !")
          ;
91      arretUrgence(); return;
92    }
93    int pwm = calculerPID(pression);
94    analogWrite(PIN_PWM, pwm);
95    float erreur = abs(pression - PRESSION_CIBLE_cmH2O);
96    if (erreur <= TOLERANCE_cmH2O) {
97      if (!stableDetecte) { stableDetecte = true; tStable =
          maintenant; }
98      else if (maintenant - tStable >= (unsigned long)
          TEMPS_STABLE_MS) {
99        Serial.println("# Pression atteinte. Passage en phase d'
            apnee.");
100       etatCourant = APNEE; tDebutPhase = maintenant;
101       stableDetecte = false;
102     }
103   } else { stableDetecte = false; }
104   enregistrerDonnees(maintenant, 1, pression, pwm);
105 }
106
107 void gererApnee() {
108   unsigned long maintenant = millis();
109   float pression = lirePression();
110   if (pression > PRESSION_MAX_SECURITE) {
111     Serial.println("# SECURITE : pression maximale depassee !")
          ;
112     arretUrgence(); return;
113   }
114   int pwm = calculerPID(pression);
115   analogWrite(PIN_PWM, pwm);
116   if (maintenant - tDebutPhase >= (unsigned long)(DUREE_APNEE_S
        * 1000)) {
117     Serial.println("# Plateau termine. Passage en expiration.")
          ;
118     etatCourant = EXPIRATION; tDebutPhase = maintenant;
119     analogWrite(PIN_PWM, 0);
120     integrale = 0; erreurPrecedente = 0; pressionFiltree = 0;
```

```
121   }
122   enregistrerDonnees(maintenant, 2, pression, pwm);
123 }
124
125 void gererExpiration() {
126   unsigned long maintenant = millis();
127   float pression = lirePression();
128   if (maintenant - tDebutPhase >= (unsigned long)(
          DUREE_EXPIRATION_S * 1000)) {
129     cycleCourant++;
130     if (cycleCourant >= NB_CYCLES) {
131       Serial.println("# Sequence terminee !");
132       Serial.print("# "); Serial.print(NB_CYCLES);
133       Serial.println(" cycles effectues.");
134       etatCourant = TERMINE;
135     } else {
136       Serial.print("# Debut du cycle ");
137       Serial.print(cycleCourant + 1);
138       Serial.print("/"); Serial.println(NB_CYCLES);
139       etatCourant   = INSPIRATION;
140       tDebutPhase   = millis(); tPrecedentPID = millis();
141       integrale     = INTEGRALE_PRECHARG;
142     }
143   }
144   enregistrerDonnees(maintenant, 3, pression, 0);
145 }
146
147 void demarrerSequence() {
148   for (int i = 0; i < 20; i++) {
149     int16_t raw  = ads.readADC_SingleEnded(CANAL_MPX);
150     float   vout = raw * LSB_GAIN_ONE;
151     float   p    = (vout / VCC_MPX - 0.04f) / 0.09f * 101.97f
152                    - OFFSET_PRESSION;
153     pressionFiltree = p;
154     delay(20);
155   }
156   Serial.println("# Demarrage de la sequence.");
157   Serial.print("# "); Serial.print(NB_CYCLES);
158   Serial.print(" cycles | Pression cible : ");
159   Serial.print(PRESSION_CIBLE_cmH2O); Serial.print(" cmH2O");
160   Serial.print(" | Apnee : "); Serial.print(DUREE_APNEE_S);
          Serial.print("s");
161   Serial.print(" | Expiration : "); Serial.print(
          DUREE_EXPIRATION_S);
162   Serial.println("s");
163   cycleCourant = 0;
164   integrale    = INTEGRALE_PRECHARG;
165   erreurPrecedente = 0; stableDetecte = false;
166   tDebutPhase = millis(); tPrecedentPID = millis();
167   etatCourant = INSPIRATION;
168   Serial.println("# Cycle 1 / debut de l'inspiration.");
169 }
170
```

```
171 void arretUrgence() {
172   analogWrite(PIN_PWM, 0);
173   etatCourant = URGENCE; integrale = 0;
174   Serial.println("# !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
175   Serial.println("# ARRET D'URGENCE     Turbine coupee");
176   Serial.println("# !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
177   Serial.println("# Tapez 's' pour redemarrer.");
178   etatCourant = ATTENTE;
179 }
180
181 int calculerPID(float pression) {
182   unsigned long maintenant = millis();
183   float dt = (maintenant - tPrecedentPID) / 1000.0f;
184   tPrecedentPID = maintenant;
185   if (dt <= 0 || dt > 1.0) dt = PERIODE_PID_MS / 1000.0f;
186   float erreur = PRESSION_CIBLE_cmH2O - pression;
187   float termP  = Kp * erreur;
188   integrale += erreur * dt;
189   integrale  = constrain(integrale, -INTEGRALE_MAX,
        INTEGRALE_MAX);
190   float termI = Ki * integrale;
191   float derivee = (erreur - erreurPrecedente) / dt;
192   float termD   = Kd * derivee;
193   erreurPrecedente = erreur;
194   float sortie = termP + termI + termD;
195   int pwm = (int)constrain(sortie, 0, PWM_MAX);
196   if (pwm > 0 && pwm < PWM_MIN) pwm = PWM_MIN;
197   return pwm;
198 }
199
200 float lirePression() {
201   int16_t raw  = ads.readADC_SingleEnded(CANAL_MPX);
202   float   vout = raw * LSB_GAIN_ONE;
203   float   p    = (vout / VCC_MPX - 0.04f) / 0.09f * 10.197f
204                  - OFFSET_PRESSION;
205   pressionFiltree = ALPHA_FILTRE * p + (1.0f - ALPHA_FILTRE) *
        pressionFiltree;
206   return pressionFiltree;
207 }
208
209 float lireDebit() {
210   if (!capteurSFM_OK) return -9999.0f;
211   Wire.requestFrom(SFM3300_ADDR, 3);
212   if (Wire.available() < 3) return -9999.0f;
213   byte msb = Wire.read(); byte lsb = Wire.read(); Wire.read();
214   uint16_t raw = ((uint16_t)msb << 8) | lsb;
215   return ((float)raw - OFFSET_DEBIT) / SCALE_FACTOR;
216 }
217
218 bool initSFM3300() {
219   Wire.beginTransmission(SFM3300_ADDR);
220   Wire.write(0x20); Wire.write(0x00); Wire.endTransmission();
        delay(500);
```

```
  Wire.beginTransmission(SFM3300_ADDR);
  Wire.write(0x10); Wire.write(0x00);
  byte err = Wire.endTransmission();
  if (err != 0) return false;
  delay(500);
  return true;
}

void enregistrerDonnees(unsigned long maintenant, int etat,
                        float pression, int pwm) {
  float debit = lireDebit();
  Serial.print(maintenant);       Serial.print(",");
  Serial.print(cycleCourant + 1); Serial.print(",");
  Serial.print(etat);             Serial.print(",");
  Serial.print(pression, 3);      Serial.print(",");
  if (debit > -900) Serial.print(debit, 3); else Serial.print("
     N/A");
  Serial.print(","); Serial.println(pwm);
}

void afficherParametres() {
  Serial.println("#

     ");
  Serial.println("# PARAMETRES ACTUELS");
  Serial.println("#

     ");
  Serial.print("# Pression cible       : ");
  Serial.print(PRESSION_CIBLE_cmH2O); Serial.println(" cmH2O");
  Serial.print("# Nombre de cycles     : "); Serial.println(
     NB_CYCLES);
  Serial.print("# Duree apnee          : ");
  Serial.print(DUREE_APNEE_S); Serial.println(" s");
  Serial.print("# Duree expiration     : ");
  Serial.print(DUREE_EXPIRATION_S); Serial.println(" s");
  Serial.print("# Securite (P max)     : ");
  Serial.print(PRESSION_MAX_SECURITE); Serial.println(" cmH2O")
     ;
  Serial.print("# Kp / Ki / Kd         : ");
  Serial.print(Kp); Serial.print(" / ");
  Serial.print(Ki); Serial.print(" / "); Serial.println(Kd);
  Serial.print("# Offset pression      : ");
  Serial.print(OFFSET_PRESSION); Serial.println(" cmH2O");
  Serial.print("# Offset debit         : "); Serial.println(
     OFFSET_DEBIT);
  Serial.println("#

     ");
}
```

Listing I.1 – Sketch Arduino — Régulateur PID v1 (tests en laboratoire)

Annexe J

Code Arduino — Régulateur PID v2 (mode hybride)

Ce code implémente une version améliorée du régulateur PID, intégrant un mode de contrôle hybride pour la phase d'apnée. Il a été utilisé lors des tests sur poumon artificiel EasyLung aux Cliniques Universitaires Saint-Luc, ainsi que lors des tests préliminaires sur sujet sain.

Principales différences par rapport à la version v1 :

- **Mode hybride en apnée** : le signal PWM de stabilisation est mémorisé à la fin de l'inspiration (moyenne des 10 dernières valeurs + offset de +10) et envoyé de façon fixe pendant l'apnée. Une correction proportionnelle de faible gain ($K_{p,apnée} = 0.8$) est appliquée uniquement si la pression dépasse la zone morte.
- **Zone morte** : aucune correction n'est apportée si l'erreur reste inférieure à ± 0.2 cmH_2O, évitant toute réaction au bruit de mesure.
- **Filtre passe-bas plus agressif** : $\alpha = 0.1$ au lieu de 0.2, pour lisser davantage le signal de pression.
- **Terme dérivé activé** : $K_d = 0.3$, rendu possible par le filtrage plus fort, pour amortir les transitions en présence de la compliance pulmonaire.
- **PWM_MAX augmenté** : 200 au lieu de 175, pour permettre d'atteindre les pressions les plus élevées sur poumon artificiel.

Matériel requis : identique à l'annexe I.

Connexions : identiques à l'annexe I.

Commandes série disponibles :

- `s` : démarrer la séquence
- `x` : arrêt d'urgence immédiat
- `p` : afficher les paramètres actuels

Paramètres utilisés lors des tests sur poumon artificiel :

- Kp = 2.0, Ki = 1.3, Kd = 0.3

— $K_{p,apnée} = 0.8$, zone morte = ± 0.2 cmH_2O
— Filtre passe-bas : $\alpha = 0.1$
— Précharge intégrale : `INTEGRALE_PRECHARG` = 60
— Tolérance de convergence : 0.5 pendant 1000 ms
— Durée du plateau : 20 secondes (labo) / 30 secondes (hôpital)

Ce code a été généré par IA (Claude code).

```
#Ce code a ete genere par Claude code
#include <Wire.h>
#include <Adafruit_ADS1X15.h>

//        Parametres modifiables


float PRESSION_CIBLE_cmH2O  = 18.0;
int   NB_CYCLES             = 3;
float DUREE_APNEE_S         = 20.0;
float DUREE_EXPIRATION_S    = 5.0;
float PRESSION_MAX_SECURITE = 25.0;

// PID phase inspiration
float Kp = 2.0;
float Ki = 1.3;
float Kd = 0.3;

// Mode hybride phase apnee
float Kp_APNEE   = 0.8;
float ZONE_MORTE = 0.2;

#define PERIODE_PID_MS      50
#define INTEGRALE_MAX      100.0
#define INTEGRALE_PRECHARG  60.0

float TOLERANCE_cmH2O = 0.5;
int   TEMPS_STABLE_MS = 1000;
float ALPHA_FILTRE    = 0.1;

//        Constantes


#define PIN_PWM         9
#define PWM_MIN         31
#define PWM_MAX         200
#define CANAL_MPX       0
#define VCC_MPX         5.0f
#define LSB_GAIN_ONE    0.0001250f
#define OFFSET_PRESSION 0.0520f
#define SFM3300_ADDR    0x40
#define SCALE_FACTOR    120
#define OFFSET_DEBIT    32768

enum Etat { ATTENTE, INSPIRATION, APNEE, EXPIRATION, TERMINE,
```

```
    URGENCE };

Adafruit_ADS1115 ads;
Etat etatCourant  = ATTENTE;
int  cycleCourant = 0;
float erreurPrecedente = 0, integrale = 0, pressionFiltree = 0;
unsigned long tPrecedentPID = 0, tDebutPhase = 0, tStable = 0;
bool stableDetecte = false, capteurSFM_OK = false;
int pwm_stabilise = 50;

void setup() {
  Serial.begin(115200);
  pinMode(PIN_PWM, OUTPUT);
  analogWrite(PIN_PWM, 0);
  Wire.begin(); Wire.setClock(50000); delay(2000);
  if (!ads.begin(0x48)) {
    Serial.println("# ERREUR : ADS1115 non detecte !");
    while (true) delay(1000);
  }
  ads.setGain(GAIN_ONE);
  capteurSFM_OK = initSFM3300();
  Serial.println("temps_ms,cycle,etat,pression_cmH2O,debit_Lmin,pwm");
  afficherParametres();
  Serial.println("# Tapez 's' pour demarrer, 'x' pour arret d'urgence.");
}

void loop() {
  if (Serial.available()) {
    char c = Serial.read();
    if (c == 'x') { arretUrgence(); return; }
    if (c == 's' && etatCourant == ATTENTE) demarrerSequence();
    if (c == 'p') afficherParametres();
  }
  switch (etatCourant) {
    case INSPIRATION: gererInspiration(); break;
    case APNEE:       gererApnee();       break;
    case EXPIRATION:  gererExpiration();  break;
    default: break;
  }
}

void gererInspiration() {
  unsigned long maintenant = millis();
  float pression = lirePression();
  if (pression > PRESSION_MAX_SECURITE) {
    Serial.println("# SECURITE : pression maximale depassee !");
    arretUrgence(); return;
  }
  int pwm = calculerPID(pression);
  analogWrite(PIN_PWM, pwm);
```

```
91
92    float erreur = abs(pression - PRESSION_CIBLE_cmH2O);
93    if (erreur <= TOLERANCE_cmH2O) {
94      if (!stableDetecte) { stableDetecte = true; tStable =
          maintenant; }
95      else if (maintenant - tStable >= (unsigned long)
          TEMPS_STABLE_MS) {
96        static int historique_pwm[10] = {0};
97        static int idx_pwm = 0;
98        historique_pwm[idx_pwm % 10] = pwm;
99        idx_pwm++;
100       int somme = 0, count = 0;
101       for (int i = 0; i < 10; i++) {
102         if (historique_pwm[i] > 0) { somme += historique_pwm[i
              ]; count++; }
103       }
104       pwm_stabilise = (count > 0) ? somme / count + 10 : pwm +
            10;
105       Serial.print("# PWM stabilise memorise : ");
106       Serial.println(pwm_stabilise);
107       etatCourant = APNEE; tDebutPhase = maintenant;
108       stableDetecte = false;
109       integrale = 0; erreurPrecedente = 0;
110     }
111   } else { stableDetecte = false; }
112   enregistrerDonnees(maintenant, 1, pression, pwm);
113 }
114
115 void gererApnee() {
116   unsigned long maintenant = millis();
117   float pression = lirePression();
118   if (pression > PRESSION_MAX_SECURITE) {
119     Serial.println("# SECURITE : pression maximale depassee !")
          ;
120     arretUrgence(); return;
121   }
122   float erreur = PRESSION_CIBLE_cmH2O - pression;
123   int pwm;
124   if (abs(erreur) < ZONE_MORTE) {
125     pwm = pwm_stabilise;
126   } else {
127     float correction = Kp_APNEE * erreur;
128     pwm = (int)constrain(pwm_stabilise + correction, PWM_MIN,
          PWM_MAX);
129   }
130   analogWrite(PIN_PWM, pwm);
131   if (maintenant - tDebutPhase >= (unsigned long)(DUREE_APNEE_S
        * 1000)) {
132     Serial.println("# Plateau termine. Passage en expiration.")
          ;
133     etatCourant = EXPIRATION; tDebutPhase = millis();
134     analogWrite(PIN_PWM, 0);
135     integrale = 0; erreurPrecedente = 0; pressionFiltree = 0;
```

```
136   }
137   enregistrerDonnees(maintenant, 2, pression, pwm);
138 }
139
140 void gererExpiration() {
141   unsigned long maintenant = millis();
142   float pression = lirePression();
143   if (maintenant - tDebutPhase >= (unsigned long)(
          DUREE_EXPIRATION_S * 1000)) {
144     cycleCourant++;
145     if (cycleCourant >= NB_CYCLES) {
146       Serial.println("# Sequence terminee !");
147       etatCourant = TERMINE;
148     } else {
149       etatCourant    = INSPIRATION;
150       tDebutPhase    = millis(); tPrecedentPID = millis();
151       integrale      = INTEGRALE_PRECHARG;
152     }
153   }
154   enregistrerDonnees(maintenant, 3, pression, 0);
155 }
156
157 void demarrerSequence() {
158   for (int i = 0; i < 20; i++) {
159     int16_t raw  = ads.readADC_SingleEnded(CANAL_MPX);
160     float   vout = raw * LSB_GAIN_ONE;
161     pressionFiltree = (vout / VCC_MPX - 0.04f) / 0.09f * 10.197
          f
162                       - OFFSET_PRESSION;
163     delay(20);
164   }
165   cycleCourant = 0;
166   integrale    = INTEGRALE_PRECHARG;
167   erreurPrecedente = 0; stableDetecte = false;
168   static int historique_pwm[10] = {0};
169   static int idx_pwm = 0;
170   for (int i = 0; i < 10; i++) historique_pwm[i] = 50;
171   idx_pwm = 0;
172   tDebutPhase = millis(); tPrecedentPID = millis();
173   etatCourant = INSPIRATION;
174   Serial.println("# Cycle 1 / debut de l'inspiration.");
175 }
176
177 void arretUrgence() {
178   analogWrite(PIN_PWM, 0);
179   integrale = 0; etatCourant = ATTENTE;
180   Serial.println("# ARRET D'URGENCE     Turbine coupee");
181   Serial.println("# Tapez 's' pour redemarrer.");
182 }
183
184 int calculerPID(float pression) {
185   unsigned long maintenant = millis();
186   float dt = (maintenant - tPrecedentPID) / 1000.0f;
```

```
187   tPrecedentPID = maintenant;
188   if (dt <= 0 || dt > 1.0) dt = PERIODE_PID_MS / 1000.0f;
189   float erreur = PRESSION_CIBLE_cmH2O - pression;
190   float termP  = Kp * erreur;
191   integrale += erreur * dt;
192   integrale  = constrain(integrale, -INTEGRALE_MAX,
          INTEGRALE_MAX);
193   float termI = Ki * integrale;
194   float derivee = (erreur - erreurPrecedente) / dt;
195   float termD   = Kd * derivee;
196   erreurPrecedente = erreur;
197   float sortie = termP + termI + termD;
198   int pwm = (int)constrain(sortie, 0, PWM_MAX);
199   if (pwm > 0 && pwm < PWM_MIN) pwm = PWM_MIN;
200   return pwm;
201 }
202
203 float lirePression() {
204   int16_t raw  = ads.readADC_SingleEnded(CANAL_MPX);
205   float   vout = raw * LSB_GAIN_ONE;
206   float   p    = (vout / VCC_MPX - 0.04f) / 0.09f * 10.197f
207                  - OFFSET_PRESSION;
208   pressionFiltree = ALPHA_FILTRE * p + (1.0f - ALPHA_FILTRE) *
          pressionFiltree;
209   return pressionFiltree;
210 }
211
212 float lireDebit() {
213   if (!capteurSFM_OK) return -9999.0f;
214   Wire.requestFrom(SFM3300_ADDR, 3);
215   if (Wire.available() < 3) return -9999.0f;
216   byte msb = Wire.read(); byte lsb = Wire.read(); Wire.read();
217   uint16_t raw = ((uint16_t)msb << 8) | lsb;
218   return ((float)raw - OFFSET_DEBIT) / SCALE_FACTOR;
219 }
220
221 bool initSFM3300() {
222   Wire.beginTransmission(SFM3300_ADDR);
223   Wire.write(0x20); Wire.write(0x00); Wire.endTransmission();
          delay(500);
224   Wire.beginTransmission(SFM3300_ADDR);
225   Wire.write(0x10); Wire.write(0x00);
226   byte err = Wire.endTransmission();
227   if (err != 0) return false;
228   delay(500);
229   return true;
230 }
231
232 void enregistrerDonnees(unsigned long maintenant, int etat,
233                         float pression, int pwm) {
234   float debit = lireDebit();
235   Serial.print(maintenant);       Serial.print(",");
236   Serial.print(cycleCourant + 1); Serial.print(",");
```

```
237   Serial.print(etat);              Serial.print(",");
238   Serial.print(pression, 3);       Serial.print(",");
239   if (debit > -900) Serial.print(debit, 3); else Serial.print("
          N/A");
240   Serial.print(","); Serial.println(pwm);
241 }
242
243 void afficherParametres() {
244   Serial.println("#

          ");
245   Serial.println("# PARAMETRES ACTUELS");
246   Serial.println("#

          ");
247   Serial.print("# Pression cible       : ");
248   Serial.print(PRESSION_CIBLE_cmH2O); Serial.println(" cmH2O");
249   Serial.print("# Nombre de cycles     : "); Serial.println(
          NB_CYCLES);
250   Serial.print("# Duree apnee          : ");
251   Serial.print(DUREE_APNEE_S); Serial.println("s");
252   Serial.print("# Duree expiration     : ");
253   Serial.print(DUREE_EXPIRATION_S); Serial.println("s");
254   Serial.print("# Securite (P max)     : ");
255   Serial.print(PRESSION_MAX_SECURITE); Serial.println(" cmH2O")
          ;
256   Serial.println("# --- PID Inspiration ---");
257   Serial.print("# Kp / Ki / Kd         : ");
258   Serial.print(Kp); Serial.print(" / ");
259   Serial.print(Ki); Serial.print(" / "); Serial.println(Kd);
260   Serial.println("# --- Mode hybride Apnee ---");
261   Serial.print("# Kp apnee             : "); Serial.println(
          Kp_APNEE);
262   Serial.print("# Zone morte           : +/-"); Serial.println(
          ZONE_MORTE);
263   Serial.print("# Alpha filtre         : "); Serial.println(
          ALPHA_FILTRE);
264   Serial.println("#

          ");
265 }
```

Listing J.1 – Sketch Arduino — Régulateur PID v2 en mode hybride

Annexe K

Figures supplémentaires — Résultats PID

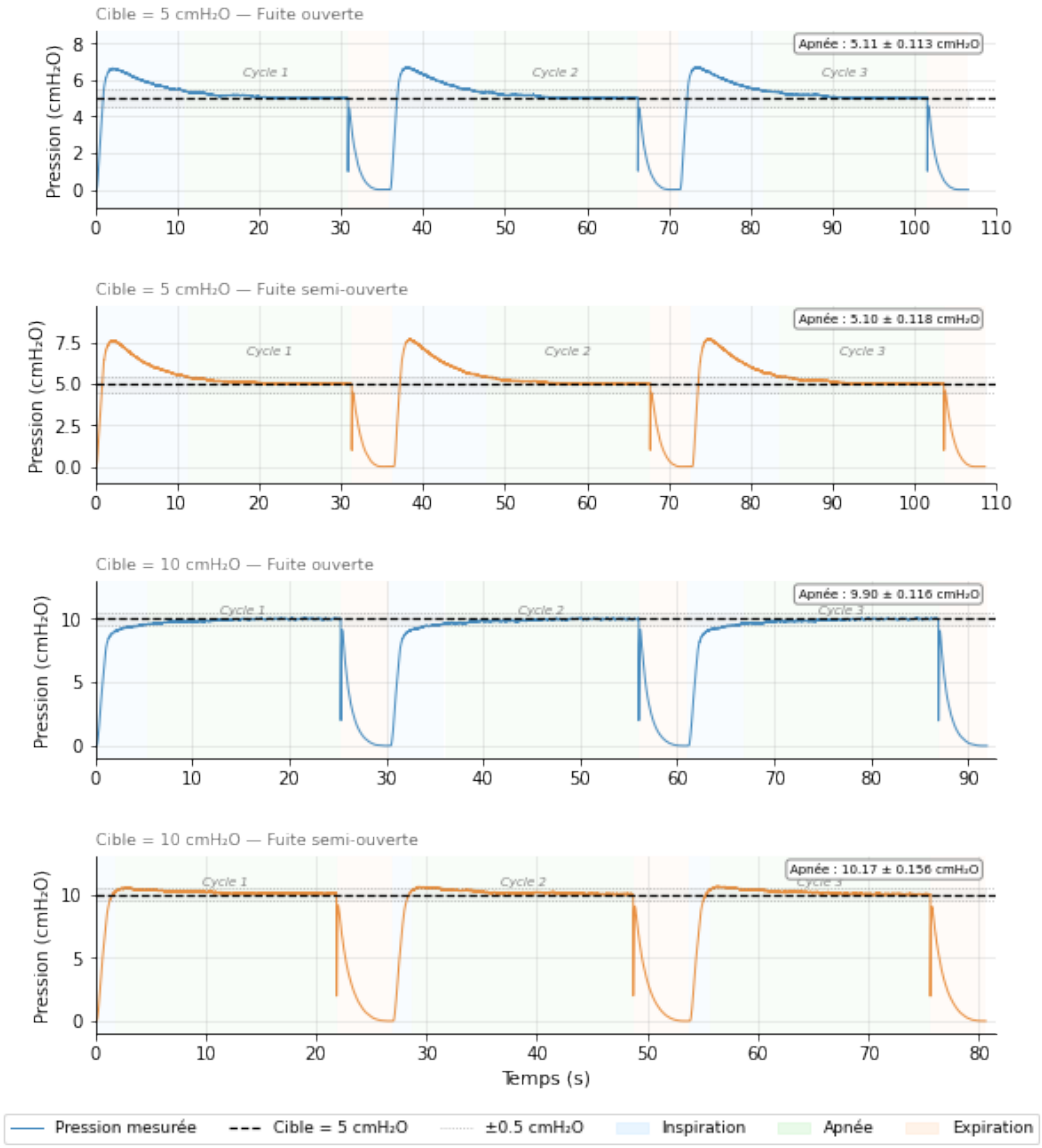


FIGURE K.1 – Résultats du PID pour les pressions 5 et 10 cmH_2O

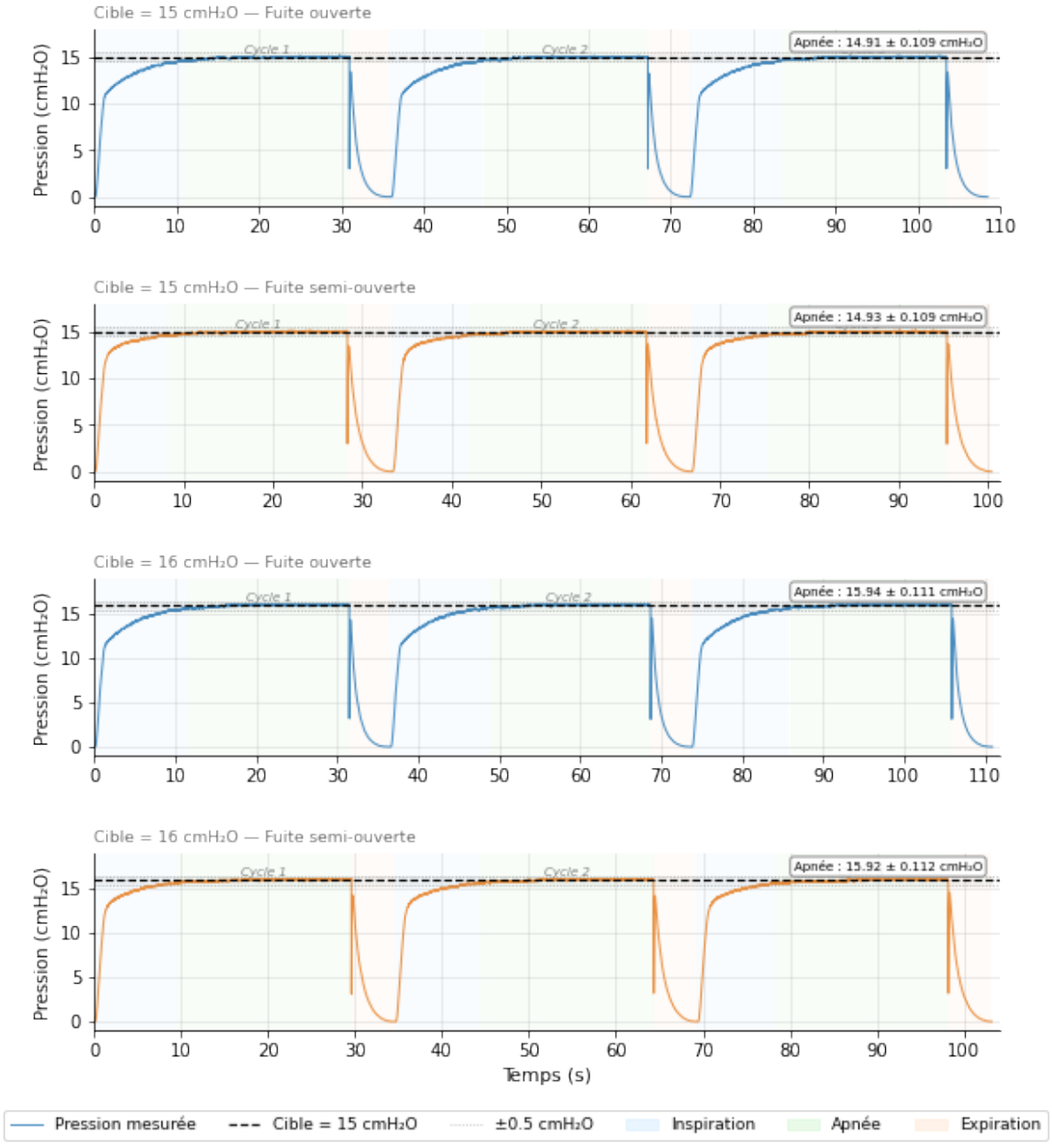


FIGURE K.2 – Résultats du PID pour les pressions 15 et 16 cmH_2O

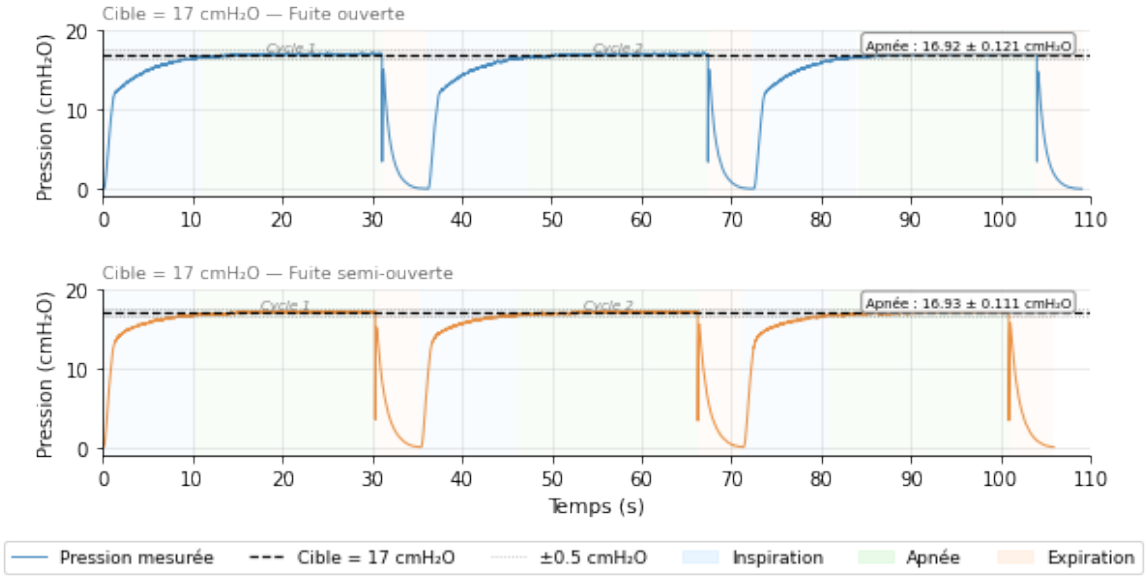


FIGURE K.3 – Résultats du PID pour la pression 17 cmH_2O

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**