

Rapport de stage BUT

# **Conception et fabrication d'une carte de contrôle d'un atténuateur RF ajustable**

Laboratoire d'Astrophysique de Bordeaux

**Par Margot LESTAGE**

Maître de stage : Wilfried D'Anna  
Professeur référent : Frédéric Champion

Date de soutenance : 17 juin 2025

Année Universitaire 2024-2025



Atacama Large  
Millimeter/submillimeter  
Array



## Remerciements

Je souhaite remercier chaleureusement toute l'équipe SEII du Laboratoire d'Astrophysique de Bordeaux, pour m'avoir si bien accueillie et intégrée dans un environnement à la fois stimulant et bienveillant. Dès mon arrivée, j'ai senti une vraie volonté de partager, d'expliquer et de me faire progresser.

Un immense merci à Wilfried D'Anna, mon maître de stage, pour sa disponibilité, sa pédagogie et pour m'avoir fait pleinement confiance dès le début. Grâce à son accompagnement, j'ai pu toucher à de nombreux aspects du projet, prendre en autonomie, et mieux comprendre ce que signifie évoluer dans un cadre professionnel exigeant, mais passionnant.

Je remercie aussi Stéphane Gauffre, pour m'avoir guidée avec patience sur les parties plus complexes de l'électronique analogique. Ses explications et ses conseils m'ont été d'une grande aide. Merci également à Hervé Soulié, toujours là pour m'épauler dans le routage et la fabrication de la carte : j'ai beaucoup appris grâce à lui, notamment sur les bonnes pratiques concrètes à appliquer sur le terrain.

Merci à Valentin Hazard, pour son soutien sur la partie logicielle. Il m'a aidée à mieux comprendre la logique du programme et son intégration dans l'ensemble du système.

Un grand merci aussi à Vincent Carlier, qui m'a permis de participer aux manipulations expérimentales. C'est grâce à lui que j'ai pu rendre plus concrètes toutes les notions théoriques vues pendant le stage. Merci à Théo Chicard et Théotime Sergeant pour leur bonne humeur, leur accueil chaleureux et leur soutien au quotidien.

Je n'oublie pas Benjamin Quertier, chef de projet, dont le leadership bienveillant et motivant m'a marquée. Il m'a montré que gérer un projet, ce n'est pas seulement suivre un planning ou distribuer les tâches, c'est aussi créer un environnement où chacun peut progresser.

Merci aussi à François Augereau, pour m'avoir laissé utiliser la salle de SAE lors de ces cours, pour ses conseils sur l'assemblage de CMS et pour son aide après la phase de soudure.

Enfin, je tiens à remercier Frédéric Champion, mon professeur référent, qui est venu me voir au sein du laboratoire.

## Résumé

Ce stage avait pour objectif la conception et la réalisation d'une carte électronique de contrôle pour l'atténuateur RF ADRF5720. Après une phase de compréhension du cahier des charges et de conception, les logiques ont été simulées puis routées sur Fusion 360. L'étape suivante a été la fabrication et l'assemblage de la carte, suivis des tests fonctionnels qui ont permis de valider les tensions, les commandes logicielles et le bon fonctionnement global. Ce projet a renforcé mes compétences en électronique, programmation Arduino et gestion de projet. La mise en service du système opérationnel est en attentes mais permettra son intégration dans un banc de test plus large pour caractériser des ADC.

## Abstract

This internship was focused on the design and development of an electronic control board for the ADRF5720 attenuator. After understanding all the specifications and completing the design phase, all the logic circuits have been simulated then routed with Fusion 360. The next step was the manufacturing, the assembly and functional test which validated the different voltages, the software command and the overall operation of the board. This project enhanced my skills in electronic, Arduino computing and project management. The deployment of the operating system is pending but its integration will help in the characterization of ADC within a larger bench test.

## Tables des matières

<b>Remerciements.....</b>	<b>2</b>
<b>Résumé.....</b>	<b>3</b>
<b>Abstract.....</b>	<b>3</b>
<b>Index des figures.....</b>	<b>6</b>
<b>Introduction.....</b>	<b>8</b>
<b>Présentation du cadre du stage.....</b>	<b>9</b>
1. Présentation du LAB.....	9
2. Présentation de l'équipe SEII.....	10
2.1. Projets spatiaux.....	10
2.2. Projets terrestres.....	10
3. Présentation du projet ALMA 2030.....	11
3.1. Qu'est ce que ALMA.....	11
3.2. WSU (Wideband Sensitivity Upgrade) et WIFP (WSU IF Processor).....	13
<b>Déroulement du stage et travaux réalisés.....</b>	<b>15</b>
1. Contexte et objectifs du projet.....	15
2. Présentation du banc de test.....	15
3. Travaux réalisés sur la chaîne RF.....	19
4. Cahier des charges du module.....	21
5. Organisation logicielle.....	23
5.1. Conception initiale : logigramme et structuration.....	23
5.2. Firmware Arduino (logiciel embarqué).....	23
5.3. Software Python.....	27
5.4. Structure générale du code.....	27
6. Phase de simulation.....	28
6.1. Simulation électronique.....	28
6.2. Simulation logicielle.....	32
7. Phase de conception avancé.....	33
7.1. Schéma électronique.....	33
7.2. Routage.....	33
8. Phase de fabrication.....	34
8.1. Fabrication de la carte.....	34
8.2. Approvisionnement en composants.....	35
9. Assemblage.....	35
10. Vérification.....	36
10.1. Câblage personnalisé pour l'alimentation.....	37
10.2. Vérification progressive de la carte.....	37
10.3. Tests avancés avec le code.....	37
11. Mise en service du système (tests pratiques à venir).....	38
<b>Compétences acquises.....</b>	<b>39</b>

Compétences renforcées.....	39
Nouvelles compétences acquises.....	39
<b>Conclusion et perspectives.....</b>	<b>40</b>
<b>Annexes.....</b>	<b>41</b>
<b>Glossaire.....</b>	<b>72</b>
<b>Sitographie.....</b>	<b>75</b>

## Index des figures

Figure 1 : Image du site ALMA au Chili.....	11
Figure 2 : Chronologie du projet ALMA.....	11
Figure 3 : Différente visualisation de la galaxie Centaurus A.....	12
Figure 4 : Schéma de l'observatoire ALMA.....	12
Figure 5 : Schéma de la partie Back End et Front End.....	13
Figure 6 : Schéma du module DGSPOT.....	14
Figure 7 : Photo du banc de test Stratix 10.....	16
Figure 8 : Schéma global du banc de test Stratix 10.....	16
Figure 9 : Schéma des connexions du bancs de test au sein du bâtiment du LAB.....	17
Figure 10 : Schéma de la chaîne RF intégrée au banc.....	18
Figure 11 : Emplacement du module de contrôle dans la chaîne RF.....	19
Figure 12 : Photo du banc lors des mesures de la chaîne RF.....	20
Figure 13 : Schéma du banc lors des mesures de la chaîne RF.....	20
Figure 14 : Schémas mesure du spectre du générateur de bruit.....	20
Figure 15 : Spectre des différentes mesures sur la chaîne RF.....	21
Figure 16 : Image de la carte d'évaluation ADRF5720.....	22
Figure 17 : Schéma de principe du projet.....	22
Figure 18 : Extrait de code - fonction processCommand().....	24
Figure 19 : Extrait de vérification de valeur dans CmdAtt().....	25
Figure 20 : Extrait du code - fonction sendAtt().....	25
Figure 21 : Exemple de codage binaire des atténuations.....	26
Figure 22 : Extrait du code - fonction CmdAtt().....	26
Figure 23 : Extrait de l'envoi et l'interprétation de commande.....	28
Figure 24 : Schéma simulation étage transistor +3.3V.....	29
Figure 25 : Résultat de simulation de l'alimentation +3.3V.....	30
Figure 26 : Schéma simulation étage transistor -3.3 V.....	30
Figure 27 : Résultat de simulation de l'alimentation -3.3 V.....	31
Figure 28 : Simulation de l'étage à transistor avec inverseur de tension.....	32
Figure 29 : Photo de la carte assemblée.....	36
Figure 30 : Premier logigramme.....	43
Figure 31 : Logigramme finale.....	44
Figure 32 : Schéma simulation étage transistor +3.3V.....	54
Figure 33 : Résultat de simulation de l'alimentation +3.3V.....	54
Figure 34 : Schéma simulation étage transistor -3.3 V.....	55
Figure 35 : Résultat de simulation de l'alimentation -3.3 V.....	55
Figure 36 : Schéma simulation étage transistor -3.3 V avec inverseur de tension.....	56
Figure 37 : Résultat de simulation de l'alimentation -3.3 V avec inverseur de tension.....	56
Figure 38 : Première séquence de test.....	58
Figure 39 : Deuxième séquence de test.....	59

Figure 40 : Première commande d'atténuation (0.5dB) avec LEDs.....	60
Figure 41 : Deuxième commande d'atténuation (31.5dB) avec LEDs.....	60
Figure 42 : Simulation complète du fonctionnement.....	61
Figure 43 : Première version du schéma électronique.....	63
Figure 44 : Schéma final de la carte de contrôle.....	64
Figure 45 : Premier routage de la carte.....	65
Figure 46 : Couche TOP et BOTTOM de la version 2.....	65
Figure 47 : Couche TOP de la version finale.....	66
Figure 48 : Couche BOTTOM de la version finale.....	66
Figure 49 : Visualisation de la couche TOP COPPER.....	67
Figure 50 : Visualisation du solder paste TOP.....	67
Figure 51 : Vue 3D de la couche TOP.....	68
Figure 52 : Vue 3D couche TOP sans les packages des composants.....	68
Figure 53 : Vue 3D couche TOP avec les packages des composants.....	68
Figure 54 : Vue 3D couche TOP sans les packages des composants.....	69
Figure 55 : Vue 3D couche BOTTOM sans les packages des composants.....	69
Figure 56 : Vue 3D couche TOP avec les packages des composants.....	69
Figure 57 : Vue 3D couche BOTTOM avec les packages des composants.....	69
Figure 58 : Pistolet à dénuder utilisé lors des câblages.....	70

## Introduction

Depuis mon stage de troisième au sein du laboratoire de l'Intégration du Matériaux au Système (IMS) , j'ai toujours été attirée par les environnements de recherche et plus particulièrement par les laboratoires. Passionnée par l'univers, les étoiles et l'astronomie en général, j'ai tout de suite été séduite par l'opportunité de rejoindre le Laboratoire d'Astrophysique de Bordeaux (LAB), où l'électronique est mise au service de l'exploration spatiale.

Ce stage m'a été proposé par Willy D'Anna, enseignant en SAE lors de ma première année à l'IUT de Bordeaux. Après une visite du laboratoire organisée pour notre classe, au cours de laquelle j'ai pu découvrir les équipements et les projets, j'ai manifesté mon envie d'y faire mon stage.

J'ai intégré le laboratoire au sein de l'équipe SEII (Systèmes Électroniques et Informatique Instrumentale), ma mission principale a été la conception d'une carte électronique de contrôle pour un atténuateur RF ajustable. Cette carte s'inscrit dans le développement d'un banc de caractérisation d'un convertisseur analogique-numérique (ADC) utilisé dans des chaînes d'acquisition de signaux astronomiques.

Ce stage s'est révélé particulièrement formateur par la diversité des compétences mises en jeu : j'ai dû apprendre à utiliser des outils nouveaux comme LTSpice, Wokwi ou encore Fusion 360, me familiariser avec les contraintes spécifiques au secteur public pour la fabrication de cartes et développer une certaine autonomie.

Ce rapport montre la progression de mes missions et des connaissances acquises, en abordant notamment la problématique suivante : comment développer un outil de pilotage de contrôle d'un atténuateur RF dans un banc de test en environnement de laboratoire ?

Dans un premier temps, je présenterai le cadre général du stage, le laboratoire et son fonctionnement. Ensuite, je détaillerai les différentes étapes de la conception de la carte de contrôle, avant d'aborder les manipulations et tests effectués, les difficultés rencontrées et les solutions envisagées.

## Présentation du cadre du stage

### 1. Présentation du LAB

Le Laboratoire d'Astrophysique de Bordeaux (LAB) est une unité mixte de recherche (UMR) rattachée à l'Université de Bordeaux et au Centre National de la Recherche Scientifique (CNRS). Il fait partie de l'Observatoire Aquitain des Sciences de l'Univers (OASU), lui-même rattaché à l'Institut National des Sciences de l'Univers (INSU) du CNRS.

Les domaines de recherche du LAB sont très variés. Ils vont de l'étude du milieu interstellaire à la formation des étoiles et des systèmes planétaires, en passant par la recherche de la vie, le climat planétaire et l'habitabilité. À plus grande échelle, certaines équipes s'intéressent à des objets massifs comme les amas d'étoiles et galaxies.

Le LAB est impliqué dans des projets de recherche d'ampleur nationale et internationale. Il collabore avec des agences prestigieuses telles que la NASA (*National Aeronautics and Space Administration*, États-Unis), l'ESA (*European Space Agency*, Europe) et l'ESO (*European Southern Observatory*, Europe), l'IRAM (*Institut de RadioAstronomie Millimétrique*), le LATMOS (*Laboratoire Atmosphères, Milieux, Observations Spatiales*), l'IRAP (*Institut de Recherche en Astrophysique et Planétologie*), le LIRA (*Laboratoire d'Instrumentation et de Recherche en Astrophysique*), le CNES (*Centre National d'Études Spatiales*) et d'autres.

L'organisation au sein du Laboratoire d'Astrophysique de Bordeaux (LAB) repose sur plusieurs pôles aux missions complémentaires, répartis entre activités scientifiques, techniques et administratives.

L'organisation du LAB se fait comme suit :

- ❖ 5 équipes scientifiques travaillant sur des thématiques différentes:
  - AMOR (Astrochimie Moléculaire et ORigine des systèmes planétaires)
  - ASP (Atmosphères et Surfaces Planétaires)
  - ECLIPSE (Exoplanets, CLImates and Planetary Systems Evolution)
  - FEMIS (Formation d'Étoiles et Milieu InterStellaire)
  - M2A (Métrologie de l'espace, Astrodynamique, Astrophysique)
- ❖ 3 équipes techniques qui sont spécialisées en :
  - SEII pour la conception et réalisation de systèmes embarqués
  - CSSD (Calculs Scientifiques et Science des Données) pour le développement de logiciels et interfaces de pilotage d'instruments
  - Mécanique pour la fabrication de composants pour les missions spatiales et les observatoires au sol
- ❖ 1 service administratif, chargé de la gestion logistique, financière et des ressources humaines du laboratoire.

Les équipes scientifiques ont pour objectif de produire des avancées dans la compréhension de l'univers à différentes échelles.

Les équipes techniques, quant à elles, soutiennent les recherches en développant des outils et des instruments innovants pour les observatoires terrestres ou les missions spatiales.

## 2. Présentation de l'équipe SEII

Mon stage s'est déroulé au sein de l'équipe technique SEII (Systèmes Électroniques et Informatique Instrumentale). Cette équipe est spécialisée dans le développement de cartes électroniques, de systèmes de numérisation et de modules de pilotage embarqués.

L'équipe SEII est impliquée dans deux grandes catégories de projets : les missions spatiales et les projets d'observation terrestre. Elle développe les systèmes électroniques nécessaires au fonctionnement des instruments scientifiques embarqués ou installés sur les observatoires.

### 2.1. Projets spatiaux

- ❖ Instrument ChemCam (rover Curiosity, NASA) : conception des systèmes électroniques de pilotage et d'acquisition pour un spectromètre laser (analyse la composition chimique des roches martiennes) dans la cadre de la mission MSL (Mars Science Laboratory).
- ❖ Instrument SuperCam (rover Perseverance, NASA) : augmentation des capacités de ChemCam avec de nouveaux modes de spectroscopie (Raman, infrarouge) et intégration d'un microphone dans le cadre de la mission MARS 2020.
- ❖ Instrument MIRS (Martian Moon eXploration, JAXA) : Conception des systèmes embarqués pour le spectro-imageur infrarouge qui permettra de caractériser la composition du système Martien.
- ❖ Radar WISDOM (rover Rosalind Franklin, ESA) : conception des systèmes embarqués pour un radar d'exploration du sous-sol martien dans le cadre de la mission ExoMars.
- ❖ Spectromètre HIFI (satellite Herschel, ESA) : instrument pour l'étude du milieu interstellaire froid à haute résolution spectrale.

### 2.2. Projets terrestres

- ❖ SKA (Square Kilometre Array) : ce projet international vise à construire le plus grand radiotélescope au monde, réparti entre l'Afrique du Sud (SKA-Mid) et l'Australie (SKA-Low). Le LAB contribue au développement de la chaîne de réception pour des fréquences comprises entre 50 MHz et 14 GHz.
- ❖ ALMA (Atacama Large Millimeter/submillimeter Array, Chili) : participation au programme ALMA2030, avec la conception de cartes électroniques pour la numérisation à large bande (jusqu'à 40 GS/s) et le traitement numérique du signal.

### 3. Présentation du projet ALMA 2030

#### 3.1. Qu'est ce que ALMA

ALMA (Atacama Large Millimeter/submillimeter Array) est un réseau de radiotélescopes situé sur le plateau de Chajnantor à environ 5000 m d'altitude, dans le désert d'Atacama au Chili. Cet observatoire est le fruit d'une collaboration entre l'ESO (European Southern Observatory), le NRAO (National Radio Astronomy Observatory) et le NAOJ (National Astronomical Observatory of Japan) en partenariat avec le Chili.



Figure 1 : Image du site ALMA au Chili

L'étude du projet ALMA a débuté en 1995 mais la mise en service effective s'est faite en 2013 avec l'ouverture du site aux scientifiques.

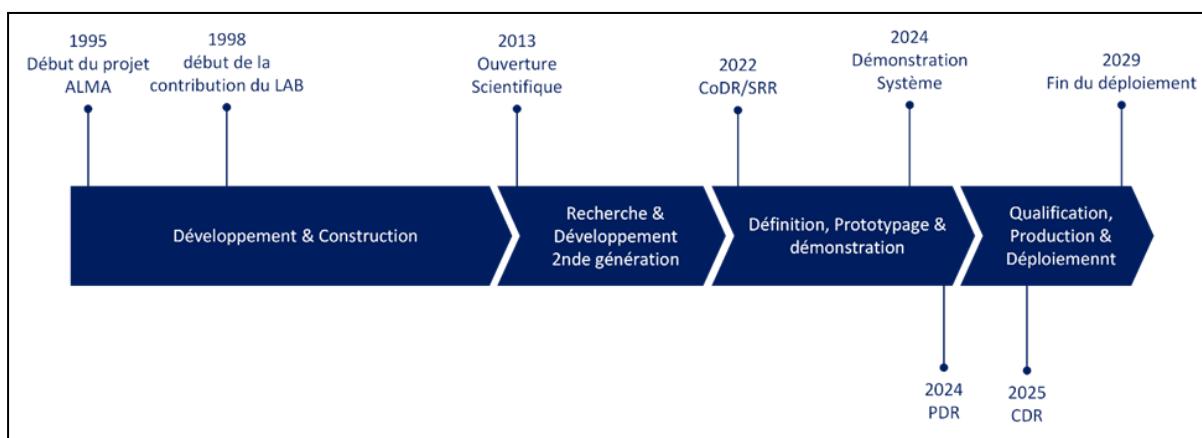


Figure 2 : Chronologie du projet ALMA

ALMA est composé de 66 antennes (54 antennes de 12 mètres de diamètre et 12 antennes de 7 mètres) dont l'espacement est compris entre 16 kilomètres et 160 mètres. Ces

antennes sont capables d'observer dans une bande de fréquences allant de 30GHz à 950GHz. L'objectif est de regarder la formation stellaire, les nuages moléculaires ainsi que l'univers primordial, ce télescope dû à sa bande de fréquence est le plus puissant pour observer l'Univers froid.

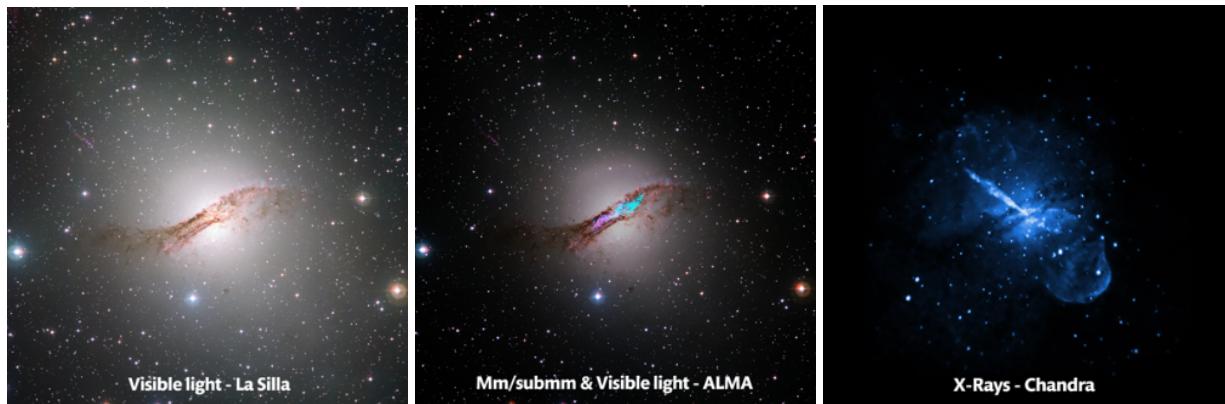


Figure 3 : Différente visualisation de la galaxie Centaurus A

Les rayons X permettent de voir des éléments plus chaud alors que les ondes millimétriques et submillimétriques permettent de visualiser les éléments plus froids.

Le signaux collectés par la parabole affluent vers le Front-End qui est composé de 10 récepteurs avec une plage de fréquences spécifique (entre 30GHz et 950GHz), puis le BackEnd permet la numérisation des données et le corrélateur sert au traitement du signal des antennes.

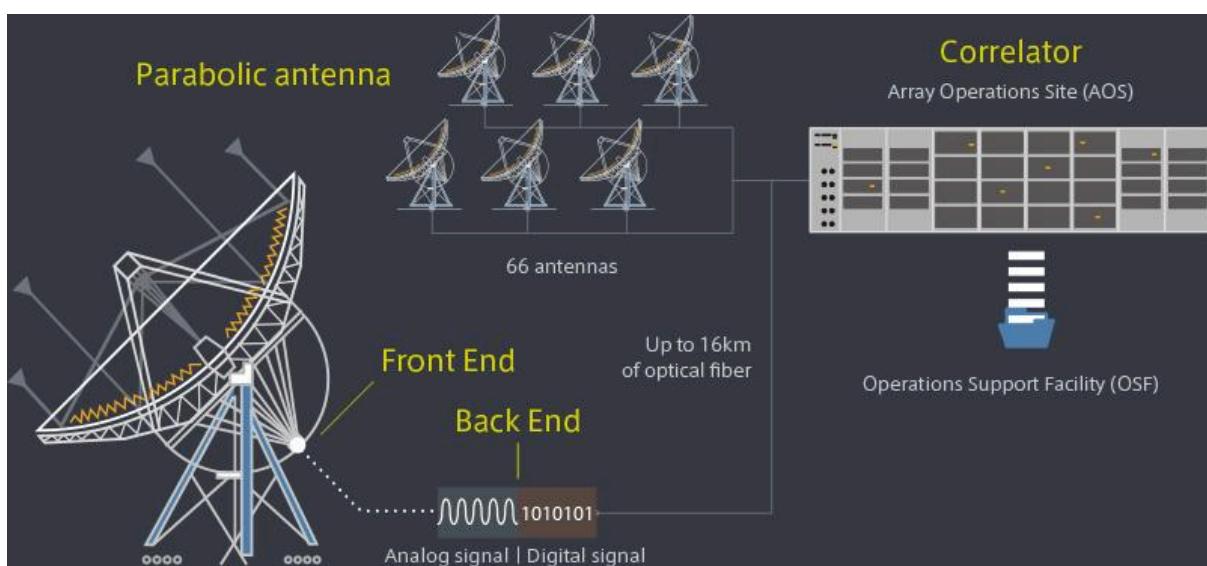


Figure 4 : Schéma de l'observatoire ALMA

### 3.2. WSU (Wideband Sensitivity Upgrade) et WIFP (WSU IF Processor)

ALMA 2030 est une campagne d'amélioration de l'observatoire. Dans ce cadre, le WSU a pour objectif une amélioration de la bande passante, passage d'une bande passante de 2GHz (2GHz à 4GHz) à 18GHz (2GHz à 20GHz).

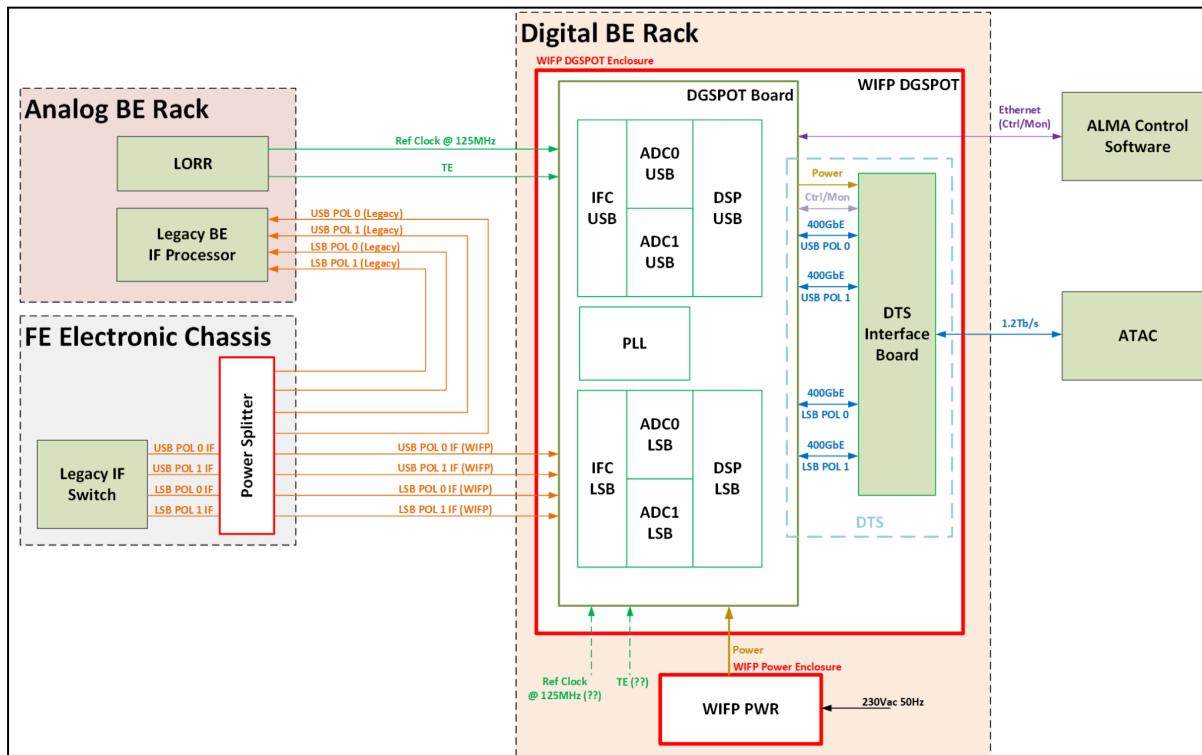


Figure 5 : Schéma de la partie Back End et Front End

Le WIFP est le projet proposé par le LAB pour contribuer au WSU.

- Côté Front End le module “Power splitter” permet de séparer la puissance des 4 signaux provenant de l'antenne vers le back-end actuel et vers le futur back-end (Module DGSPOT)
- Côté Back End, le module WIFP DGSPOT consiste à numériser le signal IF provenant du Front-end (signal provenant du ciel), traiter les signaux numérisés (grâce à des FPGA), puis mettre en paquet les signaux traités afin de les transmettre vers le super corrélateur ALMA (ATAC) via 4 liens optiques à 400GB/s soit à une vitesse globale de 1,2TB/s. D'autre part, un module d'alimentation “WIFP PWR” est nécessaire au fonctionnement de celui-ci.

Le DGSPOT est un module du WIFP. Il regroupe trois fonctions essentielles :

- ❖ la numérisation des signaux analogiques via des ADC 6 bits à 40GS/s
- ❖ le traitement des signaux numérisés
- ❖ la transmission des données numérisées via des liaisons optiques à très haut débit (jusqu'à 400Gb/s)



Figure 6 : Schéma du module DGSPOT

Le module DGSPOT intègre 4 ADC, ce qui donne un débit total de 960GS/s ( $4 \times 6$  bits  $\times$  40GS/s). Chaque ADC a pour but de numériser soit l'une des deux polarisations électromagnétiques soit la voie USB (Upper Side Sand) ou LSB (Lower Side Band).

## Déroulement du stage et travaux réalisés

Au cours de mon stage, j'ai eu l'opportunité de participer à plusieurs projets aux objectifs variés, allant de la conception complète d'un système électronique à des travaux expérimentaux intégrés dans des bancs de test dans le cadre du programme ALMA 2030. Mes autres travaux sont abordés dans l'Annexe G.

### *1. Contexte et objectifs du projet*

Dans le cadre du projet ALMA 2030, un besoin est apparu sur le banc de test dédié à la caractérisation des ADC. Le générateur de bruit utilisé dans la chaîne RF possède un atténuateur manuel et ne peut pas être contrôlé à distance, ce qui le rend peu pratique à utiliser dans un contexte de test automatisé.

Pour répondre à ces contraintes, j'ai travaillé sur la conception d'une carte électronique permettant de contrôler à distance l'atténuateur RF numérique ADRF5720 ayant une bande passante de 9kHz à 40GHz. Ce module comprenant la carte électronique et une Arduino Uno devait s'insérer dans la chaîne RF afin de permettre un ajustement de la puissance du signal d'entrée de l'ADC.

### *2. Présentation du banc de test*

Le module que j'ai conçu devait être intégré dans un banc de test développé pour le projet ALMA, visant à caractériser des ADC 6 bits à 40GS/s. Ce banc, utilisé en interne au laboratoire, est couramment désigné sous le nom de banc Stratix 10, en référence au modèle de FPGA (Field Programmable Gate Array) qu'il utilise pour piloter et récupérer les données de l'ADC.

Il permet de tester les performances de ces composants, en reproduisant un environnement aussi proche que possible des conditions réelles d'utilisation.

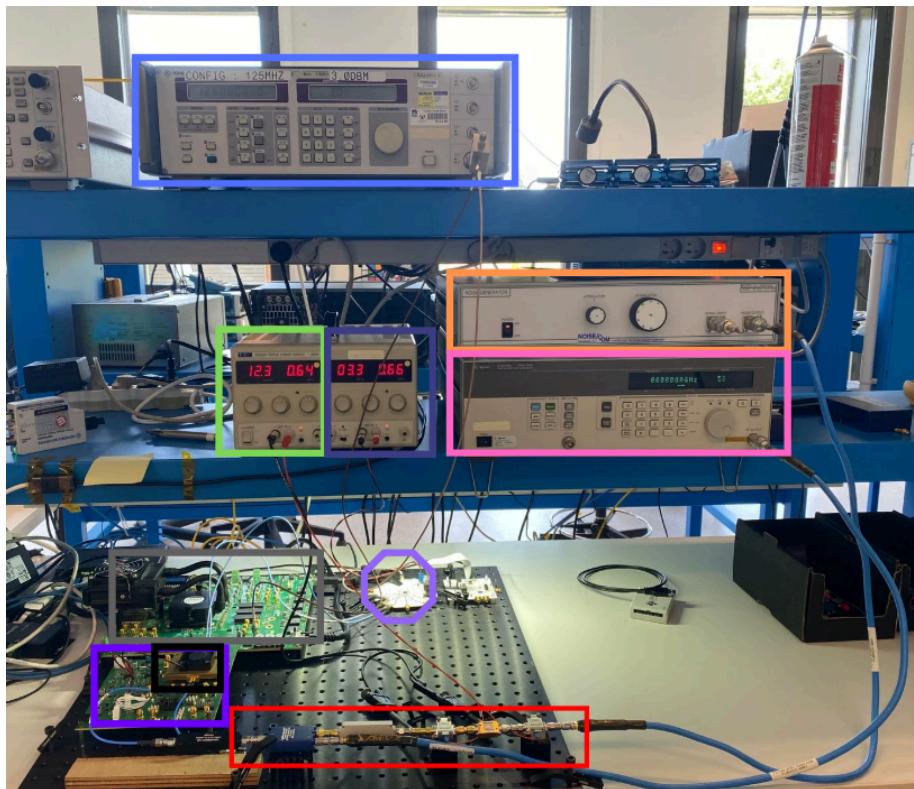


Figure 7 : Photo du banc de test Stratix 10

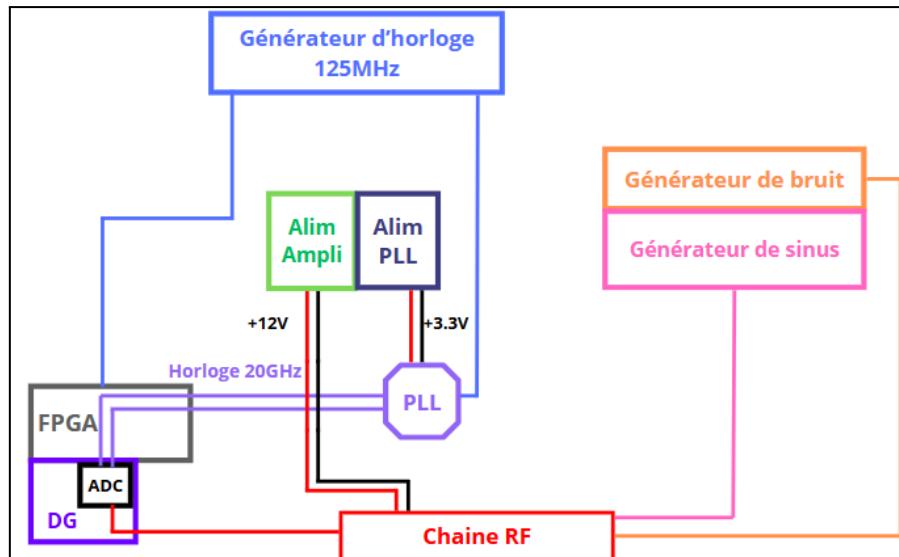


Figure 8 : Schéma global du banc de test Stratix 10

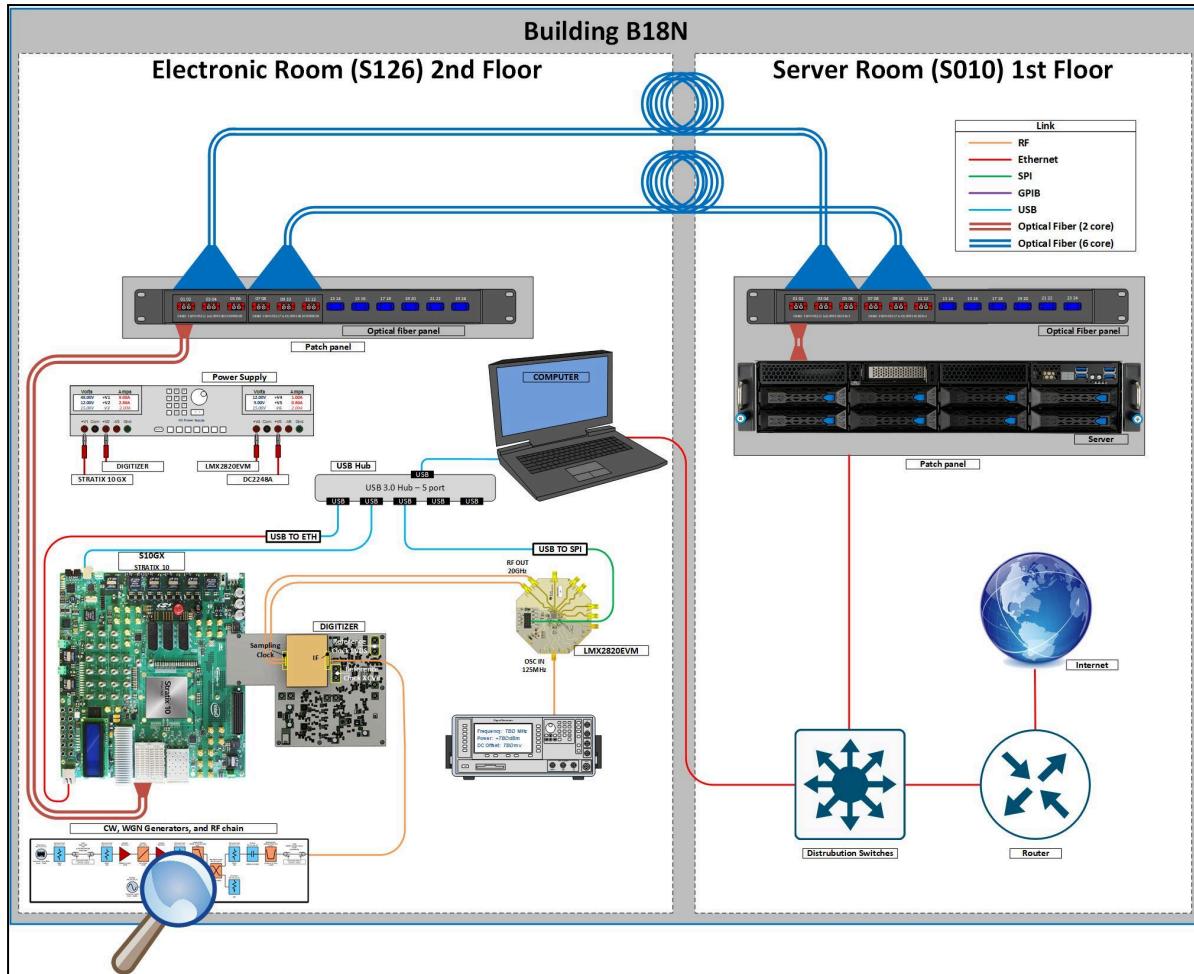


Figure 9 : Schéma des connexions du bancs de test au sein du bâtiment du LAB

Ce banc se compose de plusieurs sous-ensembles :

- ❖ L'ADC, composant au cœur de l'analyse, est monté sur une carte DG (DiGitzer) , conçue en interne. Cette carte assure :
  - l'alimentation de l'ADC (via une alimentation 12 V doublée d'une masse)
  - son interface physique avec les autres composants, notamment le FPGA
- ❖ Le FPGA Stratix 10 joue un rôle central : il reçoit les données numérisées issues de l'ADC, les traite ou les stocke et peut également contrôler certains paramètres de configuration.
- ❖ Un GUI (Graphic User Interface) est un code software écrit en python permettant de récupérer et de traiter les informations reçues par le FPGA.
- ❖ Une PLL (Phase-Locked Loop) est utilisée pour fournir le signal d'horloge de 20GHz (clock) nécessaire au bon fonctionnement de l'ADC.

Une PLL est un circuit électronique capable de générer un signal d'horloge très haute fréquence à partir d'un signal d'entrée plus faible.

Dans notre cas, la PLL reçoit une horloge d'entrée à 125 MHz puis elle délivre une horloge à 20 GHz vers l'ADC, elle est alimentée en 3,3 V et préprogrammée pour fonctionner à cette fréquence de sortie. Cette étape est essentielle pour garantir un échantillonnage correct et stable par l'ADC.

- ❖ Le générateur de bruit, utilisé pour créer un signal large bande (2GHz à 18Ghz) de type gaussien, est placé à l'entrée de la chaîne. Ce signal permet de mimiquer celui reçu venant du ciel (très atténuer). Ce générateur est manuel et réglable par potentiomètre.
- ❖ Un générateur de sinus permettant d'injecter une raie dans la bande afin de mimiquer une source astro
- ❖ La chaîne RF désigne tous les éléments analogiques situés entre le générateur de bruit et l'ADC.

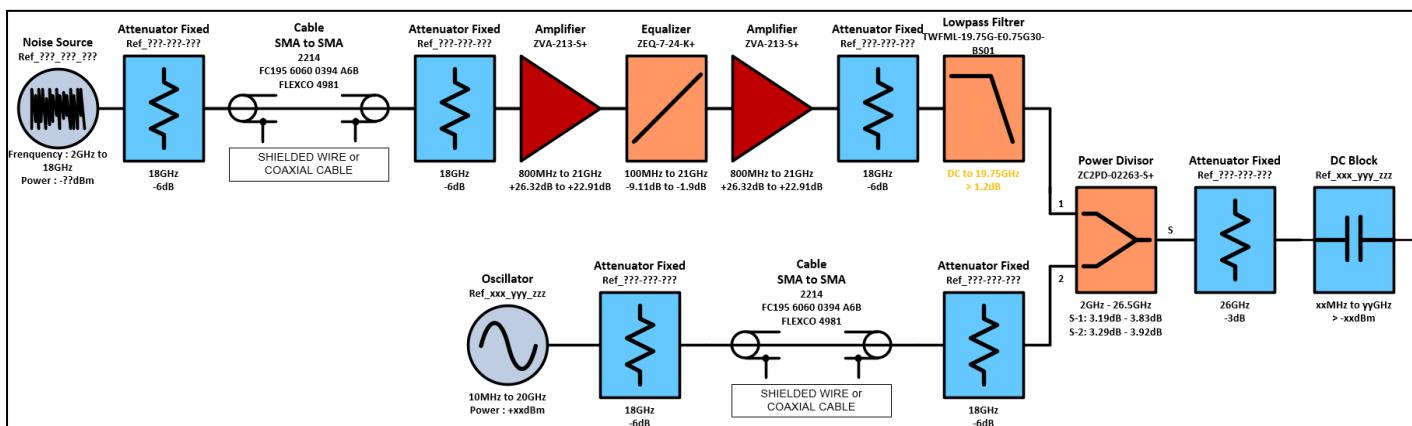


Figure 10 : Schéma de la chaîne RF intégrée au banc

C'est dans cette chaîne RF que mon module de contrôle d'un atténuateur RF numérique devait être inséré. Plus précisément, il s'intercale immédiatement après le générateur de bruit, pour permettre :

- d'ajuster numériquement le niveau du signal
- de reproduire des tests
- de commander à distance l'atténuation, sans intervention manuelle

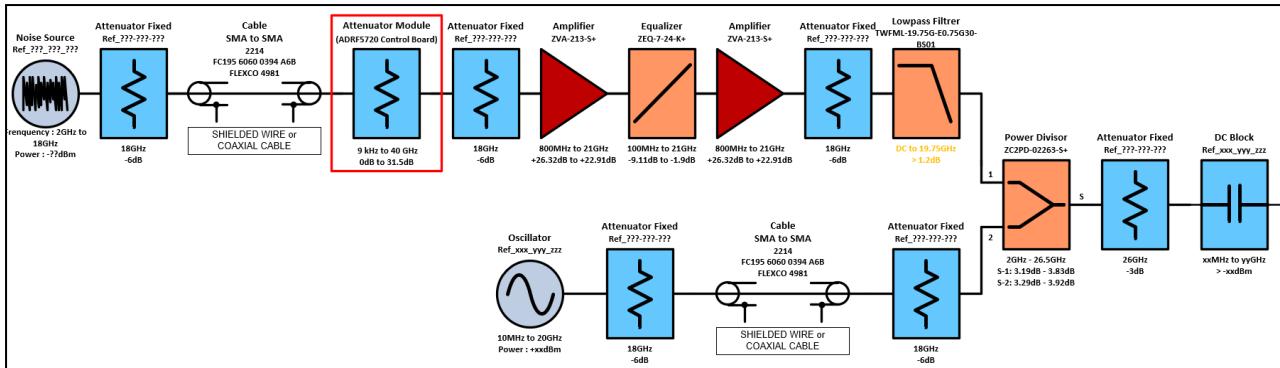


Figure 11 : Emplacement du module de contrôle dans la chaîne RF

### 3. Travaux réalisés sur la chaîne RF

Avant d'entamer la conception du module de contrôle, j'ai pris part à plusieurs manipulations sur la chaîne RF existante, dans l'objectif d'améliorer ses performances et de m'approprier son fonctionnement. Cette étape était indispensable pour assurer la compatibilité du futur module avec les conditions réelles du banc de test.

Au fil de l'analyse, nous avons constaté que certains composants de la chaîne, notamment le power divider et un DC block étaient limités à des fréquences maximales de 18 GHz et 8 GHz, ce qui n'était pas suffisant au regard des exigences du projet ALMA 2030, qui impose un fonctionnement fiable jusqu'à 20 GHz.

J'ai donc procédé à leur remplacement par des versions compatibles avec des fréquences plus élevées, en respectant les précautions liées à ce type de matériel sensible (utilisation de bracelet antistatique, manipulation soignée des connecteurs SMA avec clé dynamométrique, etc...).

Afin d'évaluer l'impact de ces modifications, j'ai mené plusieurs séries de mesures à l'aide d'un analyseur de spectre :

- ❖ une première mesure de référence a été effectuée avec la configuration d'origine de la chaîne RF
- ❖ une mesure après chaque modification (remplacement du power divider, puis du DC block)

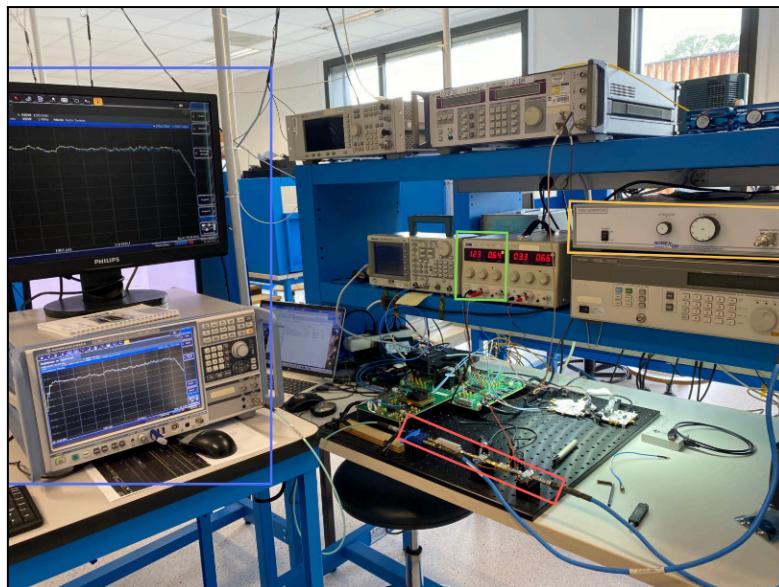


Figure 12 : Photo du banc lors des mesures de la chaîne RF

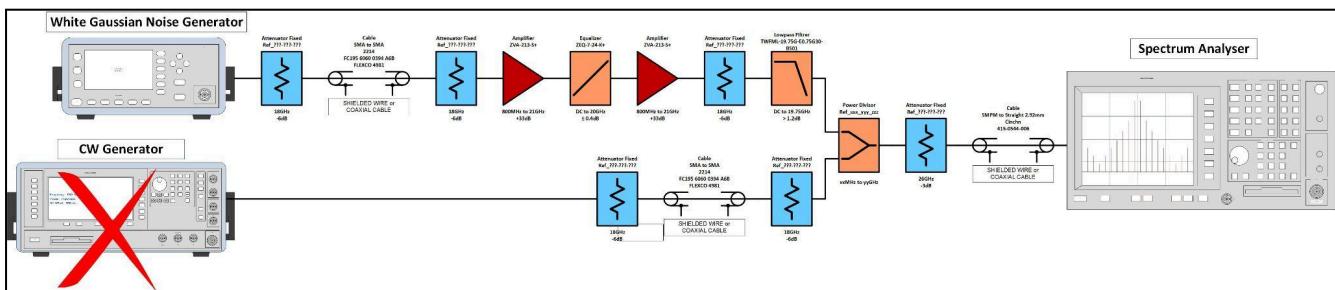


Figure 13 : Schéma du banc lors des mesures de la chaîne RF

- ❖ une mesure du générateur de bruit seul afin d'isoler son comportement

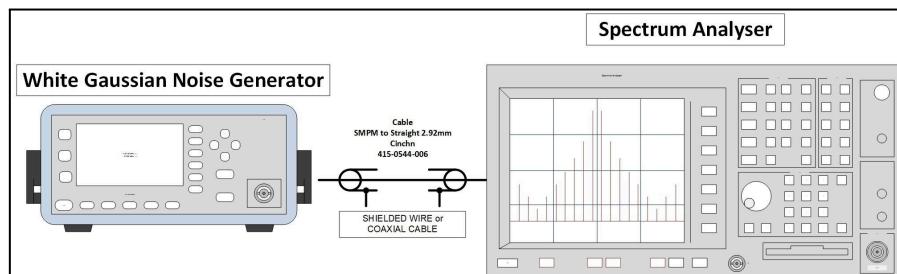
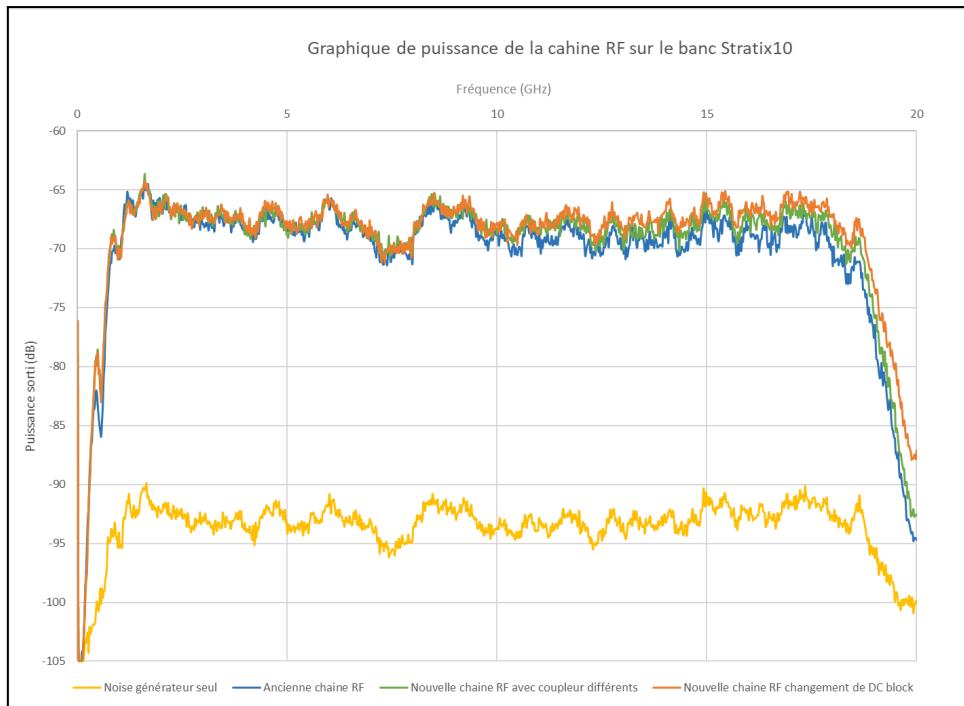


Figure 14 : Schémas mesure du spectre du générateur de bruit

Les données issues de l'analyseur ont été exportées au format .csv puis exploitées sous Excel pour visualiser les différences entre les configurations.



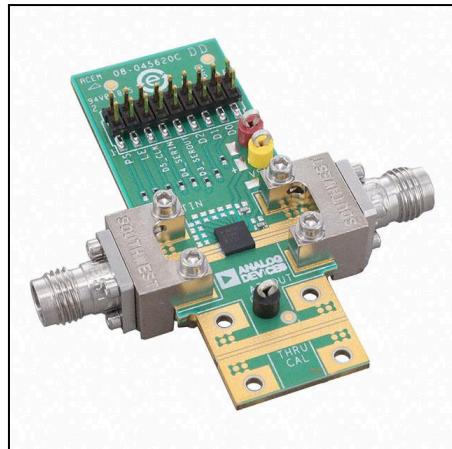
**Figure 15 : Spectre des différentes mesures sur la chaîne RF**

Les résultats montrent que la nouvelle configuration (courbe orange) offre un spectre légèrement plus linéaire et plus stable qu'auparavant (courbe bleu), avec une atténuation plus homogène sur l'ensemble de la bande. Toutefois, il est important de noter que la chaîne présente encore une baisse marquée au-delà de 18 GHz. Cette limitation ne vient plus des composants remplacés, mais du générateur de bruit lui-même (modèle Noise/COM NCB21BA), dont les performances chutent au-delà de cette fréquence.

Pour garantir une réponse réellement fiable jusqu'à 20 GHz, un générateur de bruit plus performant serait nécessaire. Cependant, ce type d'équipement fonctionnant à 20GHz représente un investissement non négligeable de l'ordre des milliers d'euros. C'est pourquoi nous considérons que lors de l'achat du nouvel équipement son fonctionnement devrait être semblable.

#### 4. Cahier des charges du module

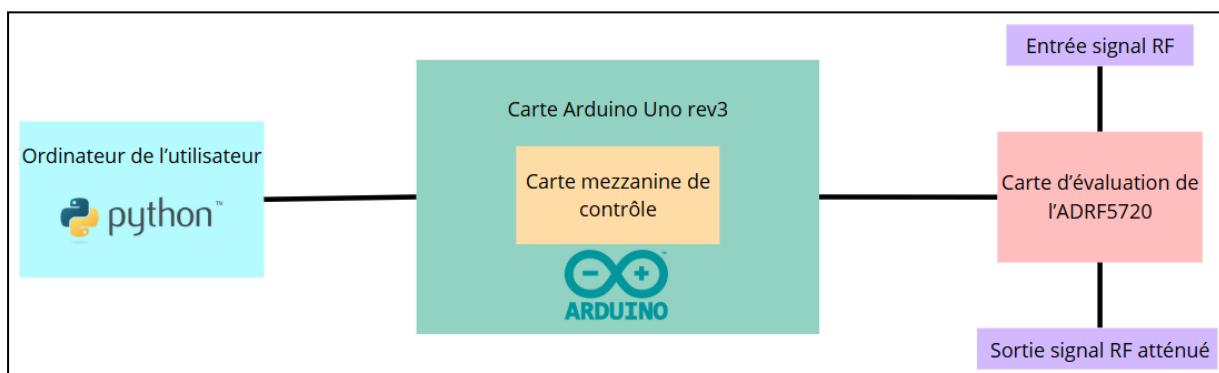
Le module que je devais concevoir avait pour but de piloter la carte d'évaluation d'un atténuateur RF numérique ADRF5720, via une carte Arduino Uno. L'ADRF5720 permet d'ajuster le niveau du signal sur une plage de 0 à 31.5 dB, par pas de 0.5 dB sur une bande de 9kHz à 40GHz.



**Figure 16 : Image de la carte d'évaluation ADRF5720**

Pour cadrer ce projet, j'ai rédigé un cahier des charges inspiré de ceux réalisés à l'IUT, en m'appuyant sur les consignes orales de mon maître de stage. Vous trouverez le cahier des charges complet, détaillé du projet ici : [CDC\\_Carte\\_contrôle\\_ADRF5720](#)

Le projet devait comporter un software Python côté PC qui communiquera avec une carte Arduino Uno et un firmware. Cette carte permettra d'envoyer les commandes à la carte d'évaluation de l'ADRF5720 via une carte adaptée conçue par mes soins selon les spécificités du composant ADRF5720.



**Figure 17 : Schéma de principe du projet**

L'ADRF5720 est un atténuateur ajustable qui peut être utilisé via deux modes : mode série et mode parallèle. Pour mon projet, j'ai opté pour ce dernier car étant plus simple à gérer notamment en termes de timing des différents signaux. C'est un composant qui fonctionne en logique +3.3V avec une double alimentation ( $VDD = +3.3V$  et  $VSS = -3.3V$ ) ainsi que les bits d'atténuation. Cela implique que ma carte de contrôle devra changer la logique +5V en une logique +3.3V.

## 5. Organisation logicielle

### 5.1. Conception initiale : logigramme et structuration

Avant toute phase de codage, j'ai réalisé un logigramme fonctionnel décrivant les différentes étapes du système de la réception des commandes jusqu'aux différentes erreurs .

Cette étape m'a permis de structurer clairement :

- ❖ Les états principaux du système
- ❖ La séquence logique d'alimentation de l'atténuateur RF ADRF5720
- ❖ Les fonctions principales à implémenter (traitement des entrées utilisateur, conversion binaire, gestion des broches)
- ❖ Les cas d'erreurs et limites à anticiper, notamment les entrées hors plage ou les valeurs non décimales d'atténuation

Ce travail préparatoire a fortement contribué à la clarté, la cohérence et la robustesse du programme dès les premières simulations. Le logigramme final est disponible en [Annexe A.1](#), ainsi que les logigrammes préliminaires.

### 5.2. Firmware Arduino (logiciel embarqué)

Le programme embarqué sur l'Arduino est responsable de la commande de l'atténuateur RF via les broches numériques. Il s'agit d'un code bas niveau assurant la fiabilité de l'exécution des commandes, même en cas de coupure de communication avec le PC. Le code complet du firmware est disponible en [Annexe A.2](#) avec le dictionnaire de commande en [Annexe A](#).

#### Fonctions logicielles clés

Le pilotage de l'atténuateur ADRF5720 s'appuie sur plusieurs fonctions critiques dans le firmware. Celles-ci assurent différentes fonctions et son architecture logicielle garantit la lisibilité et la facilité de maintenance du système.

##### Analyse et redirection des commandes : processCommand(String command)

La fonction processCommand() joue un rôle central : elle reçoit une commande texte transmise par liaison UART via l'ordinateur (script Python), puis l'analyse pour déterminer l'action à effectuer. Elle extrait d'abord le type de commande (par exemple "power" ou "att") et une éventuelle valeur numérique qui suit.

Ce découpage en deux parties permet d'isoler la logique d'analyse syntaxique (parsing) dans une fonction unique, rendant le code plus modulaire. En commençant par cette étape, on réduit les risques de traitements incohérents ou redondants dans les fonctions secondaires.

```

void processCommand(String command) {
    int spaceIdx = command.indexOf(' '); // Trouver le premier espace
    String commandType = command.substring(0, spaceIdx); // Action demandée
    String value = command.substring(spaceIdx + 1, '\n'); // Valeur : power ou att

    if (spaceIdx == -1) { // Pas d'espace (fonction de monitoring)
        if(command == "power"){ // Demande de l'état de l'alimentation
            | MonPower();
        }
        else {
            if(command == "att"){ // Demande de la valeur de l'atténuation
                | MonAtt();
            }
            else{
                | Serial.println("ERRE0");
            }
        }
    }
    else if(commandType == "power"){ // Commence par "power"
        | CmdPower(value);
    }
    else if (commandType == "att"){ // Commence par "att"
        | CmdAtt(value);
    }
    else{
        | Serial.println("ERR1"); // Possède un espace mais n'est pas repertorié
    }
}

```

Figure 18 : Extrait de code - fonction processCommand()

Cette architecture garantit que chaque commande reçue est vérifiée, puis transmise à la fonction appropriée uniquement si sa syntaxe est correcte. Cela améliore à la fois la sécurité et la robustesse globale du firmware.

#### Contrôle d'atténuation : CmdAtt(String value)

La fonction CmdAtt() est l'une des plus critiques du programme, car elle pilote directement la configuration de l'atténuation RF. Elle commence par vérifier que l'alimentation du composant est activée, une précaution indispensable car aucune atténuation peut être faite si l'atténuateur ADRF5720 est éteint.

Elle effectue ensuite une série de validations sur la valeur reçue :

- ❖ La présence éventuelle de caractères non numériques ou de virgules (,) est interdite.
- ❖ La valeur doit être un multiple de 0.5 dB et comprise entre 0 dB et 31.5 dB.
- ❖ Le format doit utiliser le point (.) comme séparateur décimal, conformément aux normes anglo-saxonnes interprétées par toFloat() et ne peut pas en contenir plusieurs.

Une fois la vérification réussie, la valeur est transmise à sendAtt() pour être envoyée aux broches.

```
// Vérification de la validité de l'atténuation
if (attVal >= 0.0 && attVal <= 31.5 && fmod(attVal * 2, 1) == 0) {
    att = attVal; // Mémoriser la valeur
    sendAtt(attVal); // Envoyer la valeur aux broches
}
else {
    Serial.println("ERR4");
}
```

**Figure 19 : Extrait de vérification de valeur dans CmdAtt()**

Ce traitement permet de prévenir toute commande non prise en charge, écrite par inadvertance par exemple, tout en offrant une réponse utilisateur claire en cas d'erreur.

#### Conversion et envoi : sendAtt(float attValue)

Une fois la commande validée, la fonction sendAtt() est responsable de sa conversion binaire, puis de son envoi physique sur les broches numériques D0 à D5. Elle convertit d'abord la valeur en un entier, représentant le nombre de pas de 0.5 dB (par exemple, 1.5 dB devient 3, soit 000011 en binaire). Chaque bit de ce mot est ensuite appliqué sur les sorties numériques correspondantes.

Pour permettre un latching correct, une impulsion est ensuite appliquée sur la broche LE (Latch Enable). Ce mécanisme est requis par la logique de l'ADRF5720, qui ne prend en compte l'état des broches qu'à la montée du signal LE.

En parallèle, les LEDs connectées à chaque ligne permettent de visualiser en temps réel les bits envoyés.

```
void sendAtt(float attValue) {
    // Multiplier par 2 pour les pas de 0.5dB puis convertie de float à integer
    int binValue = int(attValue * 2);

    // Écrire chaque bit sur les broches D0 à D5
    for (int i = 0; i < 6; i++) {
        int bitValue = (binValue >> i) & 1;
        digitalWrite(pins[i], bitValue);
        digitalWrite(Led[i], bitValue);
    }

    delayMicroseconds(5);
    digitalWrite(LE, HIGH); // Latch Enable haut
    delayMicroseconds(10);
    digitalWrite(LE, LOW); // Latch Enable bas

    Serial.print("Atténuation en binaire envoyée : ");
    Serial.print(binValue, BIN);
    Serial.print(" ou en dB : ");
    Serial.print(att);
    Serial.println("dB");
}
```

**Figure 20 : Extrait du code - fonction sendAtt()**

Cette logique permet d'envoyer avec l'information d'atténuation, tout en étant sûr de respecter les exigences temporelles du circuit intégré.

Atténuation (dB)	Code binaire 6 bits	Broches activées
0	000 000	Aucune
1.5	000 011	D0 et D1
10	010 100	D4 et D2
22.5	101 101	D5, D3, D2 et D0
31.5	111 111	D0 à D5

**Figure 21 : Exemple de codage binaire des atténuations**

Le tableau complet de la logique d'atténuation est disponible en [Annexe B.1](#).

#### Gestion sécurisée de l'alimentation : CmdPower(String value)

La fonction CmdPower() assure le contrôle de l'alimentation du système, dans le respect de la séquence d'alimentation décrite par la datasheet. Lors de l'allumage, la tension positive VDD est activée avant VSS. À l'inverse, lors de l'arrêt, VSS est désactivée avant VDD. Cette logique empêche tout dommage au composant RF.

En parallèle, un translateur de niveau (piloté par la broche OE) est activé lorsque l'alimentation est présente, garantissant la compatibilité des signaux 5V de l'Arduino avec les entrées 3,3V du composant ADRF5720.

Lors de l'extinction, l'atténuation est automatiquement remise à 0, aussi bien en mémoire (att = 0.0) que physiquement (sendAtt(0)), afin d'assurer une remise à zéro fiable du système.

```
// Contrôle de l'alimentation
void CmdPower(String value){
    if (value == "1" || value == "on") {
        digitalWrite(VDD, HIGH);
        delay(10);
        digitalWrite(VSS, HIGH);
        powerOn = true;
        Serial.println("Alimentation allumée");
        digitalWrite(OE,HIGH); // Allume le translateur de niveau

    }
    else if (value == "0" || value == "off") {
        digitalWrite(VSS, LOW);
        delay(10);
        digitalWrite(VDD, LOW);
        powerOn = false;
        Serial.println("Alimentation éteinte");
        digitalWrite(OE,LOW); // Eteint le translateur de niveau

        att = 0.0;      // Réinitialiser l'atténuation
        sendAtt(0);    // Forcer la valeur 0 physiquement
    }
    else {
        Serial.println("ERR2");
    }
}
```

**Figure 22 : Extrait du code - fonction CmdAtt()**

Ces fonctions ne se contentent pas d'être fonctionnelles : elles structurent le système et le rendent fiable, évolutif et maintenable, dans un contexte de développement matériel sensible.

### 5.3. *Software Python*

En complément du firmware embarqué, un script Python a été développé pour envoyer des commandes à l'Arduino via une liaison série. Ce programme permet une communication en temps réel, une automatisation des tests, et une interface simple pour l'utilisateur final. Pour plus de précision, le code Python entier et le fichier .json se trouvent en [Annexe A.3](#) et [Annexe A.4](#).

Le programme Python assure plusieurs fonctions :

- ❖ Envoyer des commandes (att, power, etc.) à l'Arduino.
- ❖ Recevoir et interpréter les réponses, y compris les messages d'erreur.
- ❖ Automatiser des séries de mesures avec une progression d'atténuations.
- ❖ Utiliser un fichier de configuration JSON pour adapter facilement les paramètres (port COM, débit, messages d'erreur personnalisés).

Cette architecture rend le système flexible et facilement reconfigurable sans modifier le code source.

### 5.4. *Structure générale du code*

Le programme est découpé en plusieurs fonctions pour assurer une clarté et maintenabilité du programme. Tout d'abord, l'importation de la bibliothèque "pyserial" est essentielle pour la communication avec l'Arduino. C'est pourquoi le code inclut un commentaire indiquant la commande exacte à exécuter dans le terminal pour installer cette bibliothèque : "pip install pyserial"

#### Lecture de la configuration JSON

Le fichier Parameter.json contient les paramètres de port série, la vitesse de communication (baudrate) et un dictionnaire d'erreurs. Cette séparation permet d'adapter facilement le système à différents environnements.

#### Envoi d'une commande et lecture de la réponse

La fonction write\_read() envoie une commande série à l'Arduino, puis affiche la réponse. Elle vérifie si celle-ci correspond à une erreur définie dans le fichier JSON et affiche un message explicite à l'utilisateur.

```
# Fonction qui envoie une commande à l'Arduino et affiche la réponse ou erreur
def write_read(serialArduino, command, erreurs):
    serialArduino.write((command + '\n').encode()) # Envoie la commande
    print("Commande envoyée :", command)
    time.sleep(1)

    if serialArduino.in_waiting: # Message dans la liste serial
        data = serialArduino.read(serialArduino.in_waiting).decode().strip()
        if data in erreurs:
            print("Erreur détectée :", erreurs[data])
        else:
            print("Réponse de l'Arduino :", data)
```

Figure 23 : Extrait de l'envoi et l'interprétation de commande

### Automatisation d'une série de mesures

La fonction mesureSerie() permet d'envoyer automatiquement une séquence d'atténuations, en précisant :

- ❖ la valeur de départ
- ❖ le pas entre chaque valeur (ex : 0.5 dB)
- ❖ le délai entre deux envois

Cela permet de balayer automatiquement toute la plage d'atténuation, très utile pour la calibration, la visualisation ou l'enregistrement des mesures.

### Lancement du programme

La fonction start\_program() initialise la liaison série à partir des paramètres JSON, et propose une interface utilisateur simple pour :

- ❖ envoyer des commandes manuelles
- ❖ lancer une série
- ❖ quitter le programme

Une boucle finale permet à l'utilisateur de redémarrer l'application si besoin.

## 6. Phase de simulation

La phase de simulation constitue une étape cruciale dans le développement d'un nouveau projet. Elle permet non seulement de déboguer le système avant son implémentation physique, mais aussi de vérifier le comportement des composants électroniques ainsi que des éléments logiciels critiques. Cette démarche préventive garantit une meilleure fiabilité du projet final et facilite la détection précoce d'éventuels dysfonctionnements.

### 6.1. Simulation électronique

Pour la simulation des composants électroniques, notamment l'étage à transistor, j'ai utilisé le logiciel de simulation LTSpice. Bien que je n'aie jamais utilisé ce logiciel auparavant,



Atacama Large  
Millimeter/submillimeter  
Array



j'ai rapidement su l'appréhender en autonomie. Cette simulation m'a permis d'analyser la polarisation des transistors et les interactions entre composants, assurant ainsi la conformité du montage avant la réalisation physique.

### Simulation +3.3V

L'objectif de cette première simulation était de créer un étage capable de délivrer une tension de +3.3V lorsque le signal numérique de commande passe à 5V, tout en garantissant une sortie à 0V lorsque ce signal est à l'état bas. Ce type de conversion est indispensable pour adapter les niveaux logiques entre l'Arduino et l'ADRF5720, tout en évitant d'endommager ce dernier.

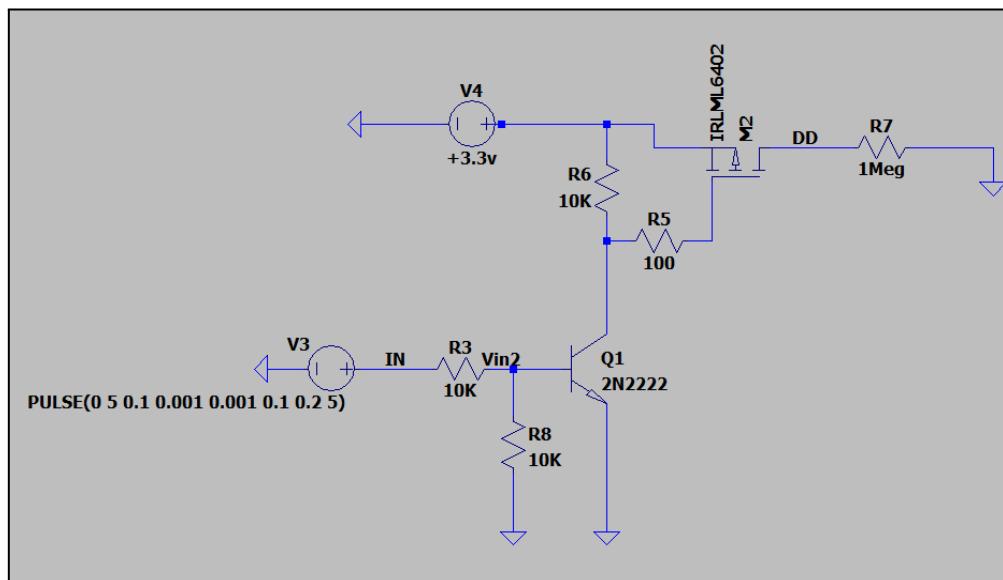


Figure 24 : Schéma simulation étage transistor +3.3V

Le montage repose sur une logique de commutation bipolaire/MOS, permettant de délivrer une tension de +3.3V à partir d'un signal numérique de 5V, tout en assurant un découplage correct entre les niveaux logiques. Le transistor bipolaire NPN sert d'interface logique et le transistor PMOS assure la commutation de la tension d'alimentation vers la sortie.

### Fonctionnement analogique du montage

→ Lorsque le signal de commande est à 0V, aucun courant ne traverse la base du NPN. Celui-ci reste bloqué et la grille du PMOS est alors tirée vers le +3.3V par la résistance de 10 kΩ. La tension entre la grille et la source du PMOS est donc nulle :  $V_{GS} = 3.3V - 3.3V = 0V$  ce qui maintient le PMOS en état bloqué. La sortie est alors à 0V.

→ Lorsque le signal de commande passe à 5V, une tension de polarisation  $V_{BE}=5V$  est appliquée entre la base et l'émetteur du NPN, qui devient saturé. Le transistor NPN devient alors conducteur et relie le collecteur à la masse. La grille du PMOS est alors tirée à 0V,

tandis que sa source reste à +3.3V. On a donc :  $VGS = 0V - 3.3V = -3.3V$ . Cette tension négative rend le PMOS conducteur, autorisant le passage du courant de la source vers le drain. La sortie bascule alors à +3.3V.

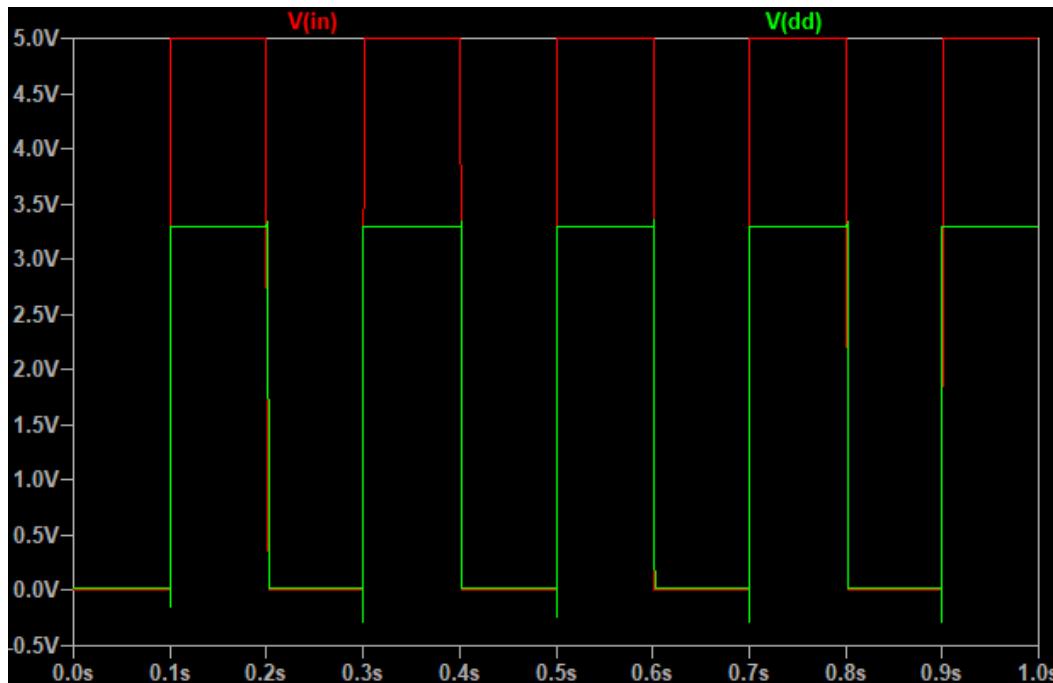


Figure 25 : Résultat de simulation de l'alimentation +3.3V

### Simulation -3.3V

Après avoir validé la commutation du +3.3 V, une seconde simulation a été menée pour produire une tension de -3.3 V à partir d'un signal logique 5 V. L'objectif est similaire : obtenir -3.3 V à la sortie lorsque l'entrée est à 5 V et 0 V lorsque l'entrée est à 0 V. Ce type de montage est essentiel dans mon système à double alimentation.

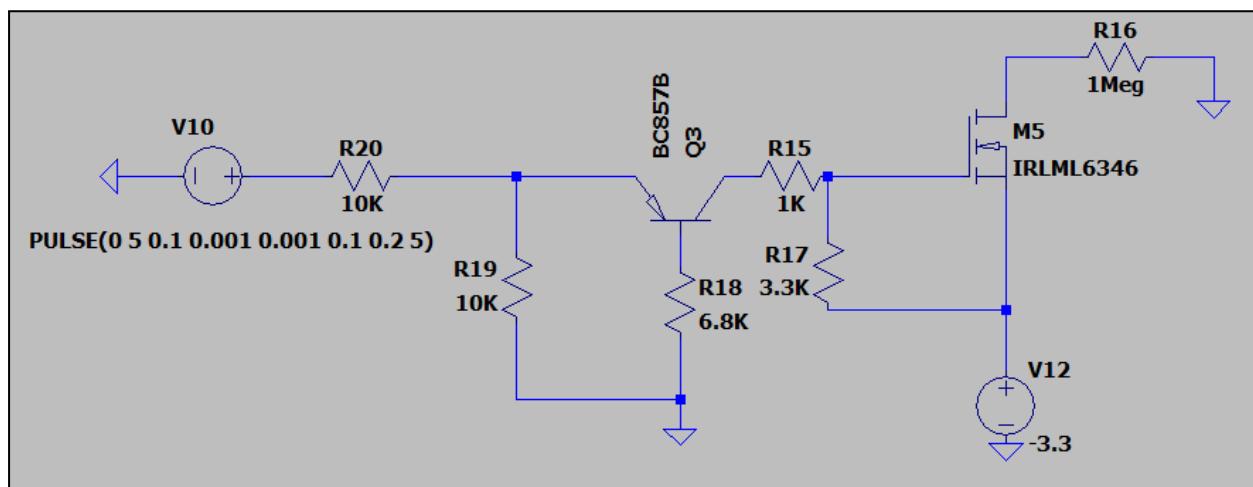


Figure 26 : Schéma simulation étage transistor -3.3 V

### Fonctionnement analogique du montage

→ Lorsque le signal de commande est à 0 V, le PNP est bloqué : aucun courant ne traverse la jonction base-émetteur. La grille du NMOS est alors tirée vers la source (-3.3 V) via la résistance de 3.3 kΩ. On a donc  $V_{GS} = -3.3V - (-3.3V) = 0V$ , le NMOS est non conducteur. La sortie reste à 0 V.

→ Lorsque l'entrée passe à 5 V, une tension  $V_{EB}=5V$  est appliquée entre l'émetteur et la base du PNP, le transistor devient saturé. Son collecteur est tiré à la masse et la grille du NMOS se retrouve à 0 V. On a alors  $V_{GS} = 0V - (-3.3V) = +3.3V$ , ce qui rend le NMOS conducteur. Le courant circule de la source (-3.3 V) vers le drain et la sortie bascule à -3.3 V.

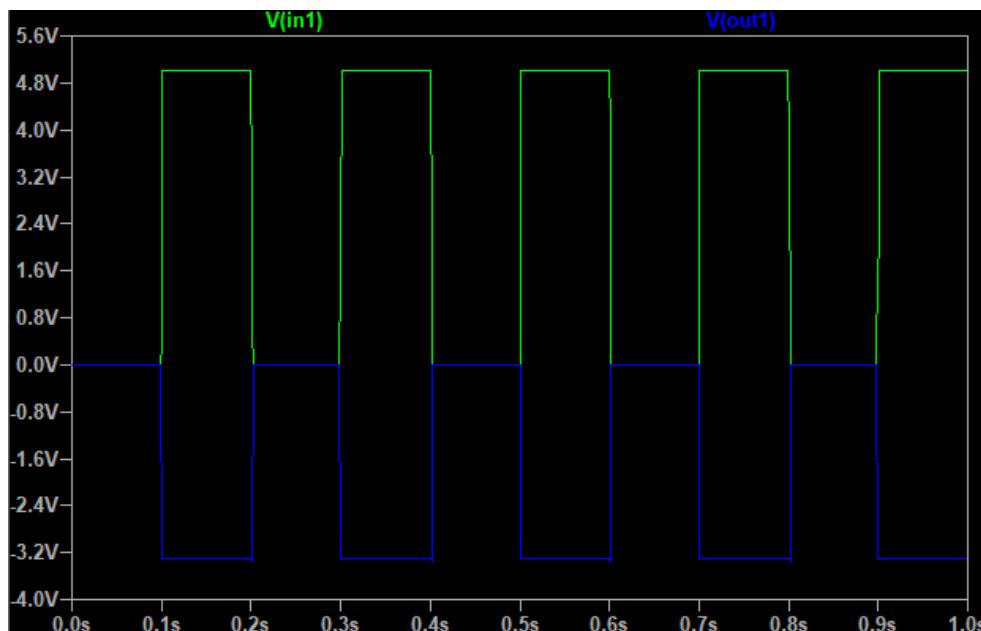


Figure 27 : Résultat de simulation de l'alimentation -3.3 V

### Simulation inverseur de tension

Pour compléter la chaîne d'alimentation, une dernière simulation a été réalisée afin de générer une tension de -3.3 V à partir du +3.3 V, nécessaire à l'alimentation de la broche VSS de certains composants. Dans cette simulation, l'inversion de tension se fait via le composant LTC660, un équivalent du LM2663 qui est prévu pour le projet.

Ce circuit a ensuite été intégré à la simulation complète de l'étage à transistor, permettant de valider le comportement réel de l'ensemble du système tel qu'il sera implanté sur la carte.

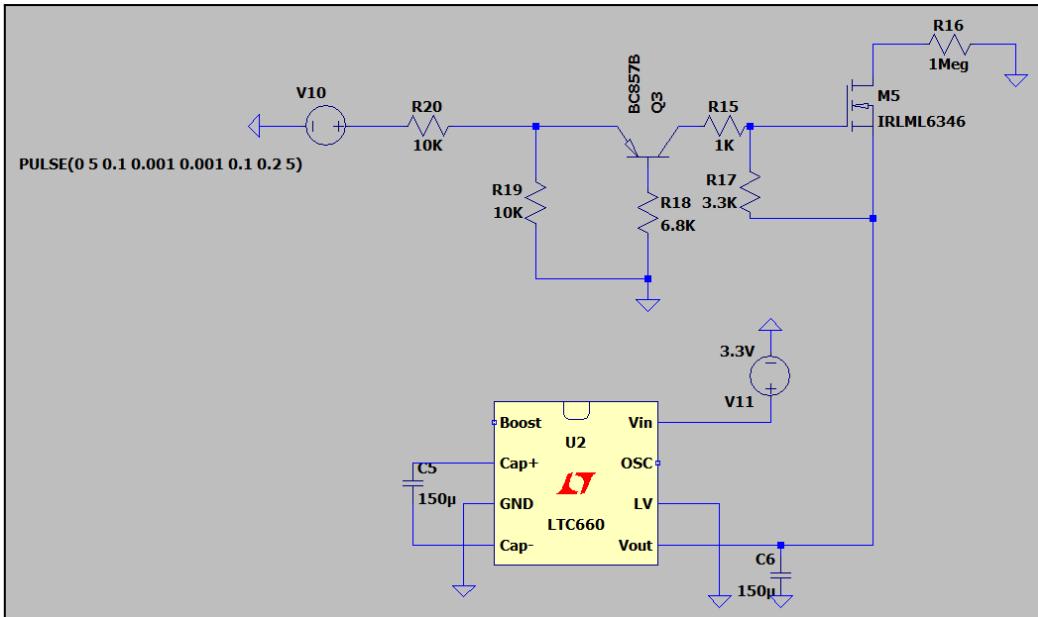


Figure 28 : Simulation de l'étage à transistor avec inverseur de tension

Cette dernière simulation a permis de confirmer la cohérence des choix effectués pour l'alimentation car j'ai retrouvé les mêmes courbes que précédemment. L'ensemble des essais ainsi que des explications sur les transistors, sont disponibles en [Annexe C](#).

## 6.2. *Simulation logicielle*

La simulation logicielle a été une étape essentielle pour valider le fonctionnement de la partie commande et monitoring avant toute implantation sur la carte. Elle a été réalisée avec l'outil [Wokwi](#), qui permet de simuler le comportement d'un Arduino ainsi que d'autres composants (LEDs, capteurs, etc.).

L'objectif de cette simulation était de tester la logique embarquée dans le firmware, pour s'assurer que les messages envoyés au microcontrôleur étaient cohérents, que l'utilisateur recevait les bons retours sur l'état du système. De plus, cela permet de vérifier qu'aucune action critique (comme l'activation d'une atténuation) ne pouvait être faite sans que le composant soit alimenté.

Ces tests m'ont permis de faire évoluer le code de manière significative. Initialement, le programme ne comportait que peu de protections, ce qui pouvait engendrer des comportements indésirables ou des erreurs dites silencieuses. Une liste d'erreurs a ainsi été construite au fur dans le code C et Python pour mieux informer et guider l'utilisateur.

Des captures d'écran des tests et des cas d'erreur sont disponibles en [Annexe D](#), illustrant les différentes situations simulées et les améliorations qui en ont découlé.

## 7. Phase de conception avancé

Après validation des simulations et des étages critiques, cette phase a concerné la création du schéma électrique et le routage de la carte. Le travail a été réalisé sur Fusion 360, utilisé ici dans sa version électronique.

### 7.1. Schéma électronique

La conception du schéma a été en grande partie réalisée en autonomie, avec un accompagnement ponctuel et des vérifications par mes collègues. J'ai dû faire face à plusieurs défis, notamment l'adaptation des modèles électroniques aux exigences du logiciel Fusion 360. Pour cela, j'ai recherché les composants adaptés en tenant compte des courants, des tensions et des contraintes d'encombrement, en utilisant notamment le site [Components Search Engine](#) qui m'a permis d'importer les bibliothèques nécessaires. Ces bibliothèques incluent les packages, l'encombrement et souvent la vue 3D, ce qui facilite la suite du travail.

Une fois les composants récupérés, j'ai réalisé la schématique en intégrant les contraintes et résultats des simulations précédentes. Au cours du stage, plusieurs versions du schéma ont été élaborées afin d'améliorer continuellement le projet. Vous trouverez l'ensemble de ces schémas, dont la version finale, en [Annexe E.1](#).

### 7.2. Routage

La première étape a été de dimensionner la carte, en visant une taille proche de celle d'un Arduino, en excluant la partie connecteur USB qui aurait gêné pour son installation.

Le routage final intègre des plans d'alimentation distincts pour les différentes tensions (+3.3 V interne ou externe, VSS, VDD, GND, 5 V), ainsi que des points de test.

Au début du projet, nous avions envisagé d'utiliser la technique des vias stitching, qui consiste en une disposition dense de vias sur la carte pour améliorer la dissipation thermique et réduire les échauffements. Toutefois, cette idée a été abandonnée car la carte ne devrait pas générer de chaleur importante, les signaux qu'elle traite étant de faible puissance cela aurait augmenté le coût sans réelle raison.

Sur les conseils de mon maître de stage, j'ai également ajouté une sérigraphie fonctionnelle sur la carte. Celle-ci inclut les noms des composants, les repères pour les condensateurs polarisés, les noms des signaux sortants des connecteurs, les indications sur les points de test et un tableau de vérité pour le jumper permettant de choisir entre une alimentation +3.3V interne ou externe.

Par ailleurs, une sérigraphie plus esthétique a été ajoutée avec le nom de la carte « ADRF5720 control board », le logo du laboratoire, ainsi que la mention « Designed by Margot Lestage » au dos de la carte.

Enfin, en dehors de la sérigraphie, j'ai dessiné en cuivre sur Fusion 360 un cadre numéroté 1 et 2 pour identifier les faces top et bottom de la carte, ce qui est une pratique courante dans le routage professionnel.

Cette phase de conception avancée a été suivie et validée par plusieurs collègues, vous trouverez l'ensemble des versions de routage en [Annexe E.2](#).

La partie schématique et l'électronique analogique ont été vérifiées par Stéphane Gauffre, ce qui m'a permis de ne pas passer à côté de subtilités liées aux composants et à leurs usages. La partie routage, quant à elle, a été contrôlée par Hervé Soulié, qui m'a apporté des conseils précieux sur la gestion des plans d'alimentation et les bonnes pratiques de routage. Le suivi chronologique s'est fait à travers quelques revues de projet et réunions où chaque étape du schéma et du routage était expliquée et discutées en détail.

## 8. Phase de fabrication

Cette phase de mon projet couvre l'ensemble du processus allant de la génération des fichiers de fabrication sur Fusion 360 jusqu'aux premières vérifications de la carte physique.

### 8.1. Fabrication de la carte

Le choix a été fait dès le départ de faire fabriquer la carte par une entreprise externe, le laboratoire ne disposant pas des moyens techniques nécessaires, notamment pour la sérigraphie. À la différence de l'IUT, où l'on utilise des procédés manuels (insoleuse, révélateur, gravure chimique, etc.) les finitions attendues ici justifient le recours à une production professionnelle.

Il a été décidé de produire trois exemplaires de la carte : une pour un assemblage manuel (réalisé par moi-même), une autre assemblée par une entreprise pour validation fonctionnelle et une dernière à titre de réserve.

J'ai pris l'initiative de rechercher un fabricant et j'ai proposé Beta Layout, également connu sous le nom PCB Pool. Cette proposition a été validée par l'équipe, notamment parce qu'il s'agit d'une entreprise européenne. En effet, le LAB étant un organisme public, il ne peut effectuer d'achats auprès de fournisseurs chinois. J'ai ensuite exporté les fichiers Gerber depuis Fusion 360, créé un devis sur leur site au nom de mon maître de stage, puis transmis ces éléments pour commande. La visualisation des fichiers Gerber sont disponibles ici : [Annexe E.3](#) avec des explications sur la fabrication.

## 8.2. *Approvisionnement en composants*

Parallèlement, j'ai préparé une première commande de composants sur Mouser, en m'appuyant sur mon routage. Lorsqu'un composant n'était pas disponible, j'ai recherché un équivalent. Cependant, la commande a été bloquée pendant un moment en raison d'un problème administratif entre un autre laboratoire de l'Université de Bordeaux et Mouser, ce qui avait alors suspendu toutes les livraisons vers les laboratoires de l'UB.

Pour pallier ce contretemps, j'ai reconstitué des paniers chez Farnell, RS et Digikey. C'est finalement Digikey qui proposait le plus de composants d'origine (et non des équivalents), donc j'ai finalisé un panier chez eux, transmis à mon maître de stage pour commande. Par chance, la commande Mouser est finalement arrivée à temps, ce qui m'a permis de poursuivre la fabrication sans trop de retard.

## 9. *Assemblage*

L'assemblage s'est déroulé à la fois à l'IUT et au laboratoire, l'étape SMT (composants montés en surface) a été réalisée à l'IUT et les composants THT (composants traversants) ont été soudés au LAB. En effet, le four de refusion du laboratoire n'étant pas fonctionnel, nous avons préféré profiter des équipements de l'IUT, plus adaptés et parfaitement calibrés pour cette étape critique.

L'assemblage des composants SMT a suivi la procédure standard de l'IUT :

- ❖ Application de la pâte à braser à l'aide d'un applicateur pneumatique
- ❖ Placement automatique des composants via la machine Pick and Place
- ❖ Vérification du positionnement à la loupe binoculaire
- ❖ Passage de la carte au four de refusion pour souder les composants
- ❖ Inspection visuelle des soudures réalisées à la binoculaire
- ❖ Reprise manuelle des soudures jugées imparfaites

Deux composants ont demandé une attention particulière :

- ❖ Le TXS0108EPWRG4, un convertisseur logique avec un pas très fin, nécessitait un contrôle précis pour éviter tout court-circuit entre broches.
- ❖ Les LEDs au format 0402, très petits (environ 1 mm × 0,5 mm), que j'avais choisis initialement sans mesurer la difficulté de manipulation. Ce choix m'a confronté à des contraintes d'assemblage réelles et formatrices. Notamment car elles sont à contacts sous le boîtier et ont dû être repositionnées, après le passage au four, avec l'aide de Monsieur Augereau, enseignant à l'IUT.

Une fois la partie CMS terminée, j'ai poursuivi l'assemblage au laboratoire :

- ❖ Nettoyage minutieux de la carte à l'alcool pour éliminer les résidus de flux
- ❖ Séchage à l'aide d'un spray dé poussiérisant à gaz neutre

- ❖ Soudure manuelle des composants traversants (THT), réalisée sous binoculaire afin de garantir une qualité optimale (forme du monticule, quantité d'étain, propreté)
- ❖ Nettoyage final pour retirer les derniers résidus de flux, qui sont être légèrement gras

Voici le rendu de ma suite suite à l'assemblage de tous les composants :

**Figure 29 : Photo de la carte assemblée**

## 10. Vérification

Une fois la carte assemblée, j'ai entamé la phase de vérification, structurée en plusieurs étapes : rédaction d'un protocole de test, mise en place des outils de mesure (alimentation de table, Arduino Uno, oscilloscope...), puis réalisation concrète des essais.

J'ai rédigé un document de vérification personnalisé, inspiré de ceux utilisés à l'IUT mais adapté aux besoins spécifiques de mon projet. Chaque test y suit une trame claire :

- Nom de l'essai (ex. : Alimentation externe / EXIG\_ALIMENTATION\_EXTERNE)
- But de l'essai
- Moyens utilisés
- Procédure d'essai
- Résultats attendus
- Résultats obtenus

Ce document m'a permis de suivre rigoureusement chaque vérification, de valider les différents étages de la carte et d'assurer une traçabilité complète.

Il est disponible ici :  ADRF5720\_VERIF et une partie en [Annexe F](#).

### 10.1. Câblage personnalisé pour l'alimentation

Avant de commencer les essais, j'ai réalisé mes propres câbles d'alimentation à partir de bobines. Après avoir coupé les fils à la bonne longueur, j'ai :

- Dénudé une extrémité (2 mm)
- Serti une fiche métallique à l'aide d'une pince adaptée
- Protégé à l'aide d'une gaine thermorétractable pour éviter les courts-circuits
- Dénudé l'autre extrémité (3 cm), replié le fil, vissé dans un embout banane
- Protégé la zone avec la gaine plastique de l'embout banane

Pour l'alimentation principale (+3.3 V, 5 V et GND), j'ai utilisé un connecteur triple, tressé les trois fils pour limiter les perturbations électromagnétiques et appliqué une gaine thermique en amont de la fin de câble pour laisser de la liberté de mouvement. Enfin, j'ai étiqueté chaque fiche banane pour éviter toute erreur de branchement pour les alimentations. Vous trouverez des images de cette étape dans l'[Annexe F.1](#).

### 10.2. Vérification progressive de la carte

Une fois les câbles prêts, j'ai commencé par la vérification des tensions d'alimentation et des niveaux logiques sur les points critiques (points de test, broches de connecteurs, etc.), qui sont décrites dans le document précédent.

J'ai ensuite contrôlé les composants actifs comme les LEDs. L'une d'entre elles était grillée, probablement lors de l'assemblage, lorsqu'on les a ressoudés. Étant donné son format 0402, mon maître de stage a préféré la remplacer lui-même, même si j'aurais aimé relever le défi.

Après validation des tests statiques, la carte a été connectée à une Arduino Uno pour tester l'ensemble du système dans des conditions proches de l'utilisation réelle. Cette étape m'a permis de valider l'interaction entre la carte et le code embarqué.

### 10.3. Tests avancés avec le code

Ces tests ont également été essentiels pour valider les programmes Arduino et Python développés en amont. Si le code Arduino avait déjà été partiellement simulé, ce n'était pas le cas du code Python. Grâce à cette phase, j'ai pu :

- ❖ Identifier et corriger certaines incohérences logiques
- ❖ Améliorer l'interface utilisateur (textes, options, clarté des messages)
- ❖ Ajouter des protections logicielles
- ❖ Mieux documenter le code, à destination des futurs utilisateurs

Cette étape finale de vérification a été cruciale, notamment pour sécuriser l'utilisation du composant ADRF5720, un composant sensible et coûteux, que je ne pouvais pas me permettre d'endommager. L'ensemble des vérifications a donc permis de garantir le bon fonctionnement de la carte avant de passer à des tests plus avancés avec la carte d'évaluation de l'ADRF5720.

## 11. Mise en service du système (tests pratiques à venir)

La mise en service complète de mon système n'a pas encore pu être réalisée, mais elle constituera l'étape finale du projet. Cette phase consistera à intégrer la carte conçue dans l'ensemble de la chaîne RF, avec le composant ADRF5720 en fonctionnement réel.

Les objectifs de cette mise en service seront :

- ❖ Vérifier la conformité de l'atténuation de l'ADFR5720 avec les valeurs mesurées via un power meter
- ❖ Confirmer la stabilité et la fiabilité du système en fonctionnement continu

- ❖ Utiliser l'ensemble pour effectuer des tests approfondis sur le comportement de l'ADC dans des conditions réalistes

Cette dernière étape permettra de valider non seulement le bon fonctionnement de la carte, mais aussi son intégration dans l'environnement cible pour lequel elle a été développée.

Enfin, une autre étape importante sera d'intégrer mon code Python directement dans l'interface graphique (GUI) déjà utilisée pour le banc de test. Cela permettra de regrouper toutes les fonctions du banc, y compris le contrôle de l'atténuation de la chaîne RF, dans une seule interface. Ce sera beaucoup plus pratique pour les utilisateurs au quotidien.

Cette partie est d'autant plus importante que la carte que j'ai développée servira à tester l'efficacité et le bon fonctionnement de chaque module DGSPOT. D'ici la fin du projet ALMA, prévue en 2030, mon système devrait être utilisé au moins une centaines de fois pour valider environ une centaine de modules. C'est très motivant de savoir que le travail que j'ai réalisé pendant ce stage pourra vraiment servir sur le long terme.

## Compétences acquises

Au cours de ce stage, j'ai consolidé un certain nombre de compétences déjà abordées durant ma formation, tout en développant de nouvelles.

### Compétences renforcées

J'ai pu approfondir et perfectionner plusieurs compétences techniques :

- ❖ Soudure : aussi bien en THT (traversant) qu'en SMT (montage en surface), sur des composants de tailles variées y compris des packages très petits (type 0402).
- ❖ Schématique et routage de circuits imprimés : grâce à l'utilisation de Fusion 360 Electronics, j'ai affiné ma compréhension des règles de conception électronique, des plans d'alimentation et des contraintes d'encombrement.
- ❖ Utilisation de composants analogiques : notamment des transistors, en comprenant mieux leur fonctionnement, leur configuration et leur intégration dans un circuit.
- ❖ Programmation Arduino : amélioration de mes compétences en C embarqué, avec la gestion de signaux numériques, la création de protections logicielles, la gestion d'erreurs, etc.
- ❖ Python : utilisé pour des scripts de test, d'analyse ou de communication.
- ❖ Rédaction technique : notamment la documentation des tests et la préparation de rapports de vérification.

### Nouvelles compétences acquises

J'ai également acquis des compétences que je n'avais encore jamais pratiquées :

- ❖ Rédaction d'un cahier des charges : comprendre et formuler les objectifs, contraintes et besoins d'un projet technique.
- ❖ Rédaction de documents utilisateurs : création de fiches explicatives pour la prise en main et l'utilisation de la carte que j'ai conçue.
- ❖ Utilisation de fichiers JSON : dans le cadre de la communication ou de la configuration de certains scripts.
- ❖ Fabrication de câbles personnalisés : choix des connecteurs, sertissage, câblage propre et adapté aux besoins spécifiques du projet.

Ce bilan technique me permet de mesurer les progrès réalisés durant ce stage. Il m'a permis de gagner en autonomie et en capacité d'adaptation face à des outils ou des technologies que je ne connaissais pas encore.

## Conclusion et perspectives

Mon projet de carte de contrôle de l'atténuateur ADRF5720 est dans sa phase de finalisation. La mise en service complète du système est prévue pour ma dernière semaine de stage. Ce projet m'a énormément apporté, aussi bien sur le plan technique que sur celui de la gestion de projet. En effet, durant ce projet j'ai beaucoup gagné en autonomie car j'en été responsable avec notamment : la prise de contact avec des entreprises pour la fabrication, la création de devis, la recherche de fournisseurs de composants et la gestion des commandes. J'ai également suivi les étapes de fabrication par mail, ce qui m'a permis de vraiment m'approprier le projet. J'ai aussi pu valoriser des compétences acquises à l'IUT qui ont été renforcées lors de ce stage.

En parallèle, j'ai participé à plusieurs tests, notamment sur le bruit parasite ("spurious") d'un convertisseur ADC et sur le Time Event (signal d'impulsion pour synchroniser les numérisations). C'est sur ces derniers que j'ai été le plus impliquée. J'ai manipulé les protocoles de test, utilisé le GUI et contribué à l'analyse de certains dysfonctionnements. Par exemple, certaines numérisations montraient des "glitch", des défauts de reconstitution du signal, alors que d'autres non. Cela nous a poussés à mener de nombreux tests pour en comprendre la cause. Ce genre de situation m'a permis de vivre concrètement les incertitudes d'un environnement de recherche.

Au-delà de l'aspect technique, j'ai beaucoup apprécié l'environnement du laboratoire : l'ambiance, le cadre et les projets m'ont confortée dans l'idée que ce type de structure peut me correspondre. Mais j'aimerais aussi découvrir un cadre industriel, d'où ma volonté de trouver une alternance dans ce domaine pour ma troisième année de GEII.

L'équipe SEII m'a fait part de sa satisfaction concernant mon travail et une possibilité d'alternance au LAB a même été évoquée si je suis admise à l'ENSEIRB-MATMECA, ce qui renforce davantage ma motivation. Ce stage a été très formateur et a confirmé mon envie de poursuivre dans l'électronique embarquée en école d'ingénieur. J'ai beaucoup appris : sur l'astrophysique, l'électronique, la programmation, mais aussi sur les contraintes liées à la gestion de projets de recherche comme les deadlines, les financements, les réunions techniques, etc. Cela a tout de même renforcé mon envie de possiblement poursuivre dans cette voie.

## Annexes

<u>Annexe A – Codes sources</u>	42
<u>A.1 – Logigramme code Arduino et dictionnaire de commandes</u>	42
<u>A.2 – Code Arduino (firmware final)</u>	45
<u>A.3 – Script Python (interface PC)</u>	49
<u>A.4 – Fichier JSON de configuration</u>	51
<u>Annexe B – Données techniques d’atténuation</u>	53
<u>B.1 – Tableau de correspondance logique SPI / atténuation</u>	53
<u>Annexe C – Simulations électroniques</u>	54
<u>C.1 – Etudes des transistors</u>	54
<u>C.2 – Simulation finale (étages de commutation)</u>	54
<u>Annexe D – Simulations logicielles sur Wokwi</u>	58
<u>D.1 – Simulations fonctionnelles Arduino</u>	58
<u>Annexe E – Schémas et routages PCB</u>	64
<u>E.1 – Versions de schéma électrique</u>	64
<u>E.2 – Versions de routage</u>	65
<u>E.3 – Gerber &amp; 3D</u>	68
<u>Annexe F – Tests et validations</u>	71
<u>F.1 - Fabrication des câbles de test</u>	71
<u>F.2 - Problèmes rencontrés et corrections</u>	72

## Annexe A – Codes sources

### A.1 – Logigramme code Arduino et dictionnaire de commandes

Les codes développés dans le cadre de ce projet sont le fruit d'une réflexion préalable structurée à l'aide de logigrammes.

Dès le premier logigramme, j'ai distingué deux grandes catégories de commandes, celles de monitoring et celles de contrôle qui sont essentielles pour mon projet.

Voici comment ces commandes étaient écrites dans le premier logigramme.

- ❖ Commandes de monitoring (préfixées par mon) :
  - mon hist : affiche l'historique des atténuations appliquées
  - mon status : affiche l'atténuation actuelle et l'état de l'alimentation
- ❖ Commandes de contrôle(préfixées par cmd) :
  - cmd power "0/1" : active (1) ou désactive (0) l'alimentation de l'ADRF5720
  - cmd att "valeur" : envoie une valeur d'atténuation entre 0 et 31,5 dB par pas de 0,5 dB

Vous trouverez ci-dessous le premier logigramme.

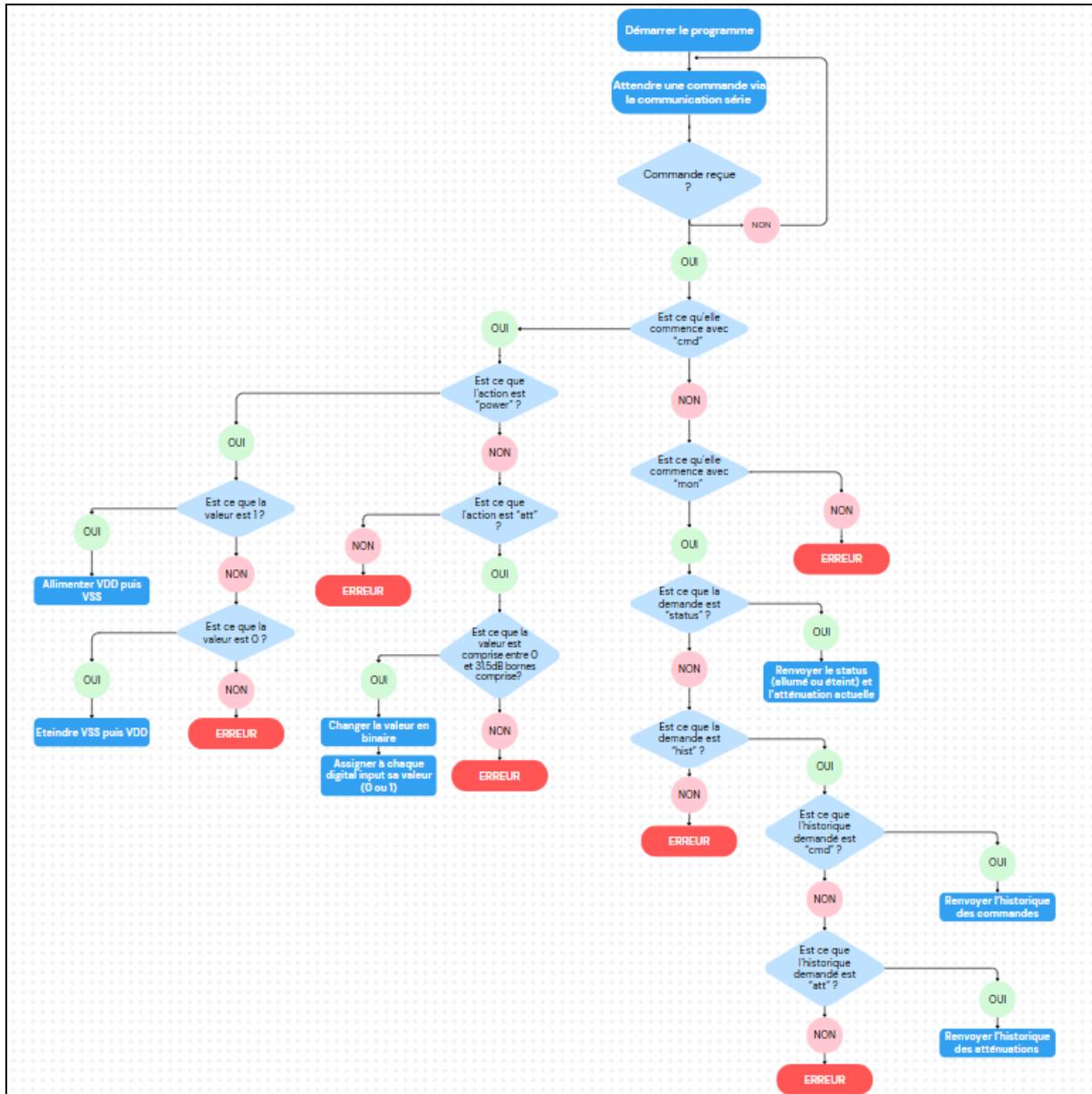


Figure 30 : Premier logigramme

Voici le logigrammes finales de mon projets :

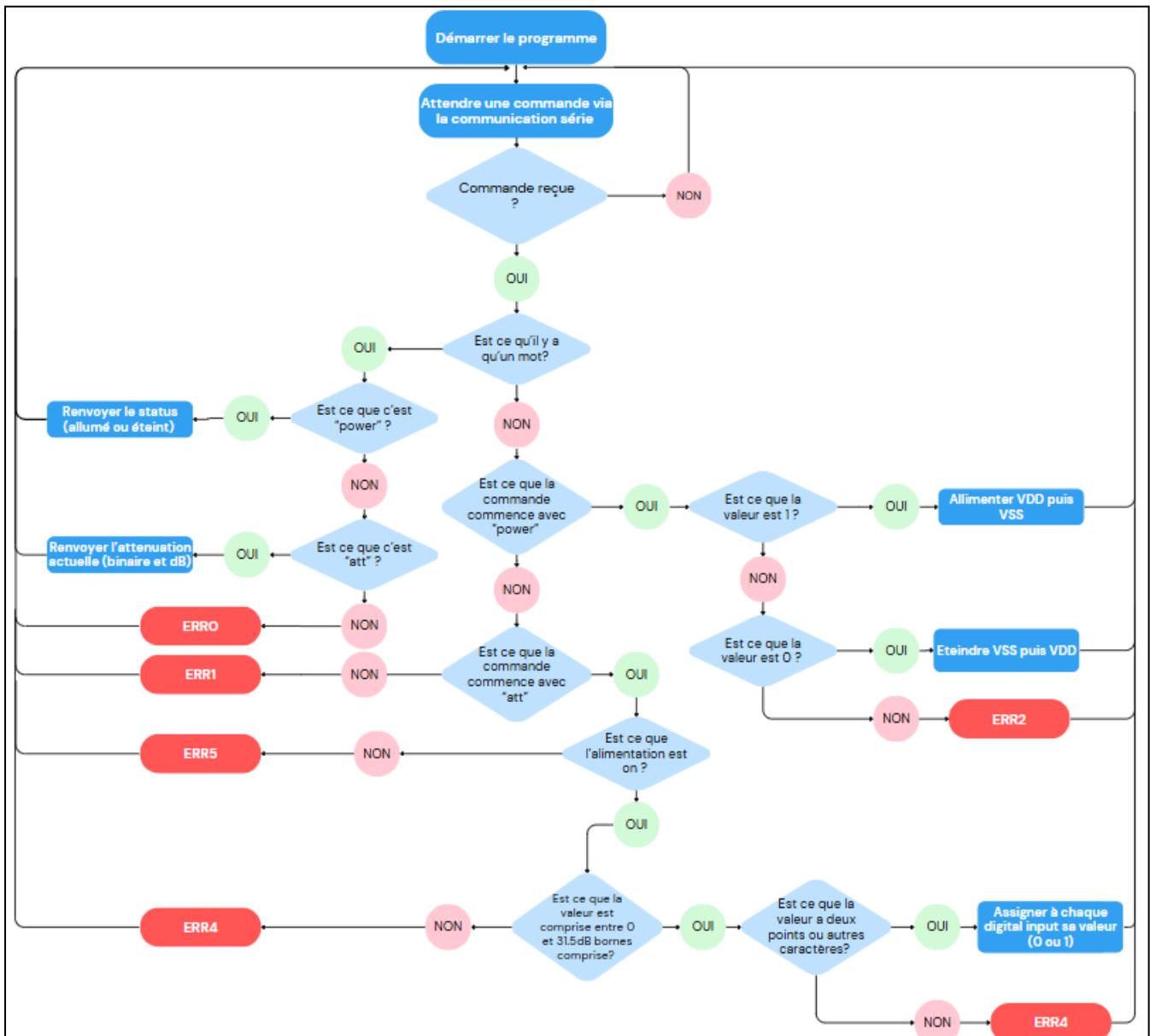


Figure 31 : Logigramme finale

Ce logigramme permet de voir les différentes commandes de mon système et renvoie des erreurs lorsque la commande est fausse, l'erreur a un numéro en fonction de son origine et dans le code json ce dictionnaire d'erreur est décrit pour l'expliquer à l'utilisateur.

Ces schémas m'ont permis non seulement de poser la logique du programme mais aussi de définir le dictionnaire finale de commandes à destination de l'utilisateur :

1er paramètre	2ème paramètre	Valeur d'entrée	Commentaire valeur	Retourne	Description valeur retourné
power	N.A	N.A	N.A	Chaine de caractère	Etat de l'alimentation
	on/off	0 ou 1	Booléen 0 = power OFF 1 = power ON	N.A	N.A
att	N.A	N.A	N.A	Chaine de caractère	Valeur de l'atténuation
	Valeur atténuation	0.0 à 31.5	Float 0.0 = 0dB 31.5 = 31.5dB (multiple de 0.5)	N.A	N.A

## A.2 – Code Arduino (firmware final)

Ce code “[Code\\_ADRF\\_V5.ino](#)” est chargé sur la carte Arduino Uno pour piloter le composant ADRF5720. Il gère :

- ❖ La séquence d'alimentation correcte du composant
- ❖ L'envoi de la valeur d'atténuation en binaire via des broches numériques (D0 à D5)
- ❖ Le contrôle via port série à l'aide de commandes textuelles
- ❖ L'affichage de l'état via LEDs pour visualiser les bits actifs

Vous trouverez toutes les versions de mon code Arduino ici : [Code\\_C](#) ou le code en texte ici :

```
const int VDD = 4; // Pin Vdd (3.3V)
const int VSS = 5; // Pin Vss (-3.3V)
const int LE = 7; // Pin LE (Latch Enable)
const int OE = 6; // Pin LE (Output Enable du translateur de niveau)

// Variables pour stocker l'état
bool powerOn = false; // État de l'alimentation (Allumé ou Éteint)
float att = 0.0; // Valeur actuelle de l'atténuation en dB

// Pins de contrôle pour l'ADRF5720 (D0 à D5)
const int pins[] = {8, 9, 10, 11, 12, 13}; // Pins D0 à D5

// Pins des LEDs
const int Led[] = {A5, A4, A3, A2, A1, A0}; // Pins A0 à A5

void setup() {
    // Initialisation de la communication série
    Serial.begin(9600);

    // Initialisation des lignes de commande de l'alim et du LE
    pinMode(VDD, OUTPUT);
    pinMode(VSS, OUTPUT);
```



Atacama Large  
Millimeter/submillimeter  
Array



```

pinMode(LE, OUTPUT);

// Éteindre par défaut les lignes d'alimentation et de latch
digitalWrite(VSS, LOW);
delay(5);
digitalWrite(VDD, LOW);
digitalWrite(LE, LOW);

// Initialisation des pins de l'ADRF5720 en mode OUTPUT
for (int i = 0; i < 6; i++) {
    pinMode(pins[i], OUTPUT);
    digitalWrite(pins[i], LOW); // Initialiser les pins D0 à D5 à LOW
}

// Initialisation des LEDs en mode OUTPUT
for (int i = 0; i < 6; i++) {
    pinMode(Led[i], OUTPUT);
    digitalWrite(Led[i], LOW); // Initialiser les pins D0 à D5 à LOW
}

void loop() {
    // Vérifier si une commande est disponible via la communication série
    if (Serial.available()) {
        String command = Serial.readStringUntil("\n"); // Lire toute la commande jusqu'à un saut de ligne
        command.trim(); // Nettoyer les espaces inutiles au début et à la fin

        processCommand(command); // Traiter la commande reçue
    }
}

void processCommand(String command) {
    int spaceIdx = command.indexOf(' '); // Trouver le premier espace
    String commandType = command.substring(0, spaceIdx); // Action demandée
    String value = command.substring(spaceIdx + 1, '\n'); // Valeur : power ou att

    if (spaceIdx == -1) { // Pas d'espace (fonction de monitoring)
        if(command == "power"){ // Demande de l'état de l'alimentation
            MonPower();
        }
        else {
            if(command == "att"){ // Demande de la valeur de l'atténuation
                MonAtt();
            }
            else{
                Serial.println("ERRO");
            }
        }
    }

    else if(commandType == "power"){ // Commence par "power"
        CmdPower(value);
    }
}

```

```

else if (commandType == "att"){ // Commence par "att"
    CmdAtt(value);
}

else{
    Serial.println("ERR1"); // Possède un espace mais n'est pas repertorier
}
}

// Afficher le statut de l'alimentation
void MonPower(void){
    String powerState; // Déclarer une variable pour l'état de l'alimentation sous forme de texte

    // Vérifier si l'alimentation est allumée ou éteinte
    if (powerOn == true) {
        powerState = "Allumé"; // Mettre "Allumé" dans powerStateText
    }
    else {
        powerState = "Éteint"; // Mettre "Éteint" dans powerStateText
    }
    Serial.print("État de l'alimentation : ");
    Serial.println(powerState);
}

// Afficher le statut actuel de l'atténuation
void MonAtt(void){
    Serial.print("Atténuation actuelle : ");
    Serial.print(att);
    Serial.println(" dB");
}

// Contrôle de l'alimentation
void CmdPower(String value){
    if (value == "1" || value == "on") {
        digitalWrite(VDD, HIGH);
        delay(10);
        digitalWrite(VSS, HIGH);
        powerOn = true;
        Serial.println("Alimentation allumée");
        digitalWrite(OE,HIGH); // Allume le translateur de niveau
    }
    else if (value == "0" || value == "off") {
        digitalWrite(VSS, LOW);
        delay(10);
        digitalWrite(VDD, LOW);
        powerOn = false;
        Serial.println("Alimentation éteinte");
        digitalWrite(OE,LOW); // Eteint le translateur de niveau
    }
    att = 0.0; // Réinitialiser l'atténuation
    sendAtt(0); // Forcer la valeur 0 physiquement
}
else {
}

```



Atacama Large  
Millimeter/submillimeter  
Array



```

Serial.println("ERR2");
}

// Contrôle de l'atténuation
void CmdAtt(String value){
    // Vérifie si l'alimentation est allumée
    if (!powerOn) {
        Serial.println("ERR5");
        return; // Retourne sans faire d'autres actions si l'alimentation n'est pas allumée
    }

    int pointCount = 0;

    // Vérifie si la chaîne contient des caractères invalides (autres que des chiffres et un seul point)
    for (int j = 0; j < value.length(); j++) {
        if (value[j] == '.') {
            pointCount++; // Incrémente le compteur du point
        if (pointCount > 1) { // Si plus d'un point
            Serial.println("ERR3");
            return;
        }
    }
    else if (!isDigit(value[j])) { // Si ce n'est pas un chiffre
        Serial.println("ERR3");
        return;
    }
}

// Vérifie si la chaîne ne contient pas de virgule
if (value.indexOf(',') != -1) { // Si il y a une virgule
    Serial.println("ERR3"); // Utilisez un point "." pas une virgule ","
    return;
}

float attVal = value.toFloat(); // Conversion en float

// Vérification de la validité de l'atténuation
if (attVal >= 0.0 && attVal <= 31.5 && fmod(attVal * 2, 1) == 0) {
    att = attVal; // Mémoriser la valeur
    sendAtt(attVal); // Envoyer la valeur aux broches
}
else {
    Serial.println("ERR4");
}

// Envoyer l'atténuation convertie en binaire à l'ADRF5720
void sendAtt(float attValue) {
    int binValue = int(attValue * 2); // Multiplier par 2 pour les pas de 0.5dB puis convertie de float à intégrer

    // Écrire chaque bit sur les broches D0 à D5
    for (int i = 0; i < 6; i++) {

```



Atacama Large  
Millimeter/submillimeter  
Array



```

int bitValue = (binValue >> i) & 1;
digitalWrite(pins[i], bitValue);
digitalWrite(Led[i], bitValue);
}

delayMicroseconds(5);
digitalWrite(LE, HIGH); // Latch Enable haut
delayMicroseconds(10);
digitalWrite(LE, LOW); // Latch Enable bas

Serial.print("Atténuation en binaire envoyée : ");
Serial.print(binValue, BIN);
Serial.print(" ou en dB : ");
Serial.print(att);
Serial.println("dB");
}

```

### A.3 – Script Python (interface PC)

Ce script Python “[ADRF5720\\_V5.py](#)” permet de dialoguer avec l’Arduino via port série pour :

- ❖ Contrôler manuellement l’alimentation et l’atténuation
- ❖ Lancer automatiquement une série de valeurs d’atténuation avec délais

Les différentes versions de ce code sont disponible ici : [Code\\_Python](#) ou ci-dessous en texte.

```

import serial
import time
import json

# Chargement des paramètres à partir du fichier JSON (port, baudrate, dictionnaire d'erreurs, etc.)
def read_parameter(file="Parameter.json"):
    with open(file, "r") as f:
        return json.load(f)

# Envoie une commande série à l'Arduino et affiche la réponse ou une éventuelle erreur
def write_read(serialArduino, command, erreurs):
    serialArduino.write((command + '\n').encode()) # Envoi de la commande
    print("\n > Commande envoyée :", command)
    time.sleep(0.1) # Délai nécessaire pour laisser le temps à l'Arduino de répondre

    if serialArduino.in_waiting: # Vérifie si des données sont disponibles en lecture
        data = serialArduino.read(serialArduino.in_waiting).decode().strip() # Lecture et décodage de la réponse
        if data in erreurs: # Si la réponse correspond à une erreur connue
            print("Attention, erreur :", erreurs[data])
        else:
            print("> Réponse de l'Arduino :", data)

# Envoie automatiquement une série de valeurs d'atténuation avec délai entre chaque
def mesureSerie(serialArduino, erreurs):

```



Atacama Large  
Millimeter/submillimeter  
Array



```

start = float(input("Atténuation de départ (dB) : "))
step = float(input("Pas entre chaque valeur (multiple de 0.5) : "))
delayseries = float(input("Délai entre chaque mesure (secondes) : "))

max_att = 31.5
count = int((max_att - start) // step) + 1 # Nombre total de pas
last_att = start + step * (count - 1) # Calcul de la dernière valeur d'atténuation

if last_att > max_att:
    last_att -= step # Ajustement si dépassement de la valeur max autorisée par l'ADRF5720

print(f"\n Série prévue : de {start:.2f} dB à {last_att:.2f} dB par pas de {step} dB.")
print("Délai entre mesures :", delayseries, "secondes")

attenuation = start
while attenuation <= max_att:
    command = "att " + str(attenuation) # Construction de la commande à envoyer
    write_read(serialArduino, command, erreurs)
    time.sleep(delayseries)
    attenuation += step

print("\n Fin de la série de mesures")

```

Dans ce début de code nous avons l'importation des bibliothèque nécessaire notamment pour la communication série, le chargement des paramètres provenant du fichier json ainsi que deux fonction :

- ❖ write\_read qui permet d'écrire à l'Arduino et recevoir et lire les données provenant de la liaison série
- ❖ mesureSerie qui permet de faire une série de mesure automatisé avec plusieurs paramètres dont : l'atténuation de début, le délai entre chaque changement et le pas d'atténuation. Cette fonction doit être améliorée pour choisir une atténuation de fin car actuellement elle est de 31.5 par défaut, cela devrait être fait avant le vendredi 13 juin 2025.

```

# Fonction principale du programme
def start_program():
    # Chargement des paramètres json
    parameter = read_parameter()
    erreurs = parameter["errors"]
    port = parameter["port"]
    baudrate = parameter["baudrate"]

    # Initialisation de la communication série avec l'Arduino
    serialArduino = serial.Serial(port, baudrate, timeout=1)
    print(f"\n Connexion réussie sur le port {port} à {baudrate} bauds.")
    time.sleep(1) # Délai d'initialisation de l'Arduino après ouverture du port

    serialArduino.reset_input_buffer() # Nettoyage du buffer d'entrée pour éviter les messages parasites

    # Boucle de commande utilisateur
    while True:

```

```

command = input("\n Entrer une commande (ex : 'att 10', 'serie', 'power on', 'exit') : ").strip().lower()

if command == "exit":
    print("Arrêt du programme.")
    break # Sortie de la boucle principale, retour à la boucle de redémarrage
elif command == "serie":
    mesureSerie(serialArduino, erreurs)
else:
    write_read(serialArduino, command, erreurs)
serialArduino.close() # Fermeture propre du port série

# Première exécution du programme
start_program()

# Boucle permettant de redémarrer le programme sans relancer le script
while True:
    restart = input("\n Souhaitez-vous fermer le programme ? (oui/non) : ").strip().lower()
    if restart == "non":
        start_program()
    else:
        print("Programme terminé.")
        break

```

Cette partie comporte la fonction principale avec l'initialisation de la communication série et la boucle de commande utilisateur ainsi que la boucle de fin de programme et de redémarrage.

#### A.4 – Fichier JSON de configuration

Ce fichier [Parameter.json](#) contient les paramètres de communication et les messages d'erreur personnalisés.

```
{
  "port": "COM4",
  "baudrate": 9600,
  "errors": {
    "ERR0": "Commande inconnue. Utilisez 'power' ou 'att'.",
    "ERR1": "Commande mal formée. Vérifiez la syntaxe après l'espace.",
    "ERR2": "Valeur pour 'power' invalide. Utilisez 'on', 'off', '1' ou '0'.",
    "ERR3": "Valeur d'atténuation invalide. Utilisez un point (pas une virgule).",
    "ERR4": "Atténuation hors limites. Doit être entre 0 et 31.5 par multiple de 0.5.",
    "ERR5": "Impossible de changer l'atténuation. Veuillez allumer l'alimentation."
  }
}
```

Même si ce fichier n'était pas essentiel fonctionnement du projet, sa création m'a permis de mieux comprendre la structure d'un fichier JSON : sa syntaxe, les éléments qu'il peut contenir (noms, types de données, etc.) ainsi que son utilité dans le cadre d'une configuration externe.

Ce fichier permet notamment :



Atacama Large  
Millimeter/submillimeter  
Array



- ❖ de centraliser les messages d'erreur dans un endroit lisible
- ❖ de modifier les intitulés des erreurs si besoin, sans avoir à intervenir dans le code Arduino
- ❖ de spécifier le port de communication série utilisé, ce qui facilite la configuration pour l'utilisateur via le gestionnaire de périphériques

## Annexe B – Données techniques d’atténuation

### B.1 – Tableau de correspondance logique SPI / atténuation

Atténuation (dB)	Code binaire 6 bits	Broche activée
0	000 000	Aucune
0,5	000 001	D0
1	000 010	D1
1,5	000 011	D0, D1
2	000 100	D2
2,5	000 101	D0, D2
3	000 110	D1, D2
3,5	000 111	D0, D1, D2
4	001 000	D3
4,5	001 001	D0, D3
5	001 010	D1, D3
5,5	001 011	D0, D1, D3
6	001 100	D2, D3
6,5	001 101	D0, D2, D3
7	001 110	D1, D2, D3
7,5	001 111	D0, D1, D2, D3
8	010 000	D4
8,5	010 001	D0, D4
9	010 010	D1, D4
9,5	010 011	D0, D1, D4
10	010 100	D2, D4
10,5	010 101	D0, D2, D4
11	010 110	D2, D4
11,5	010 111	D0, D1, D2, D4
12	011 000	D3, D4
12,5	011 001	D0, D3, D4
13	011 010	D1, D3, D4
13,5	011 011	D0, D1, D3, D4
14	011 100	D2, D3, D4
14,5	011 101	D0, D2, D3, D4
15	011 110	D1, D2, D3, D4
15,5	011 111	D0, D1, D2, D3, D4

16	100 000	D5
16,5	100 001	D0, D5
17	100 010	D1, D5
17,5	100 011	D0, D1, D5
18	100 100	D2, D5
18,5	100 101	D0, D2, D5
19	100 110	D1, D2, D5
19,5	100 111	D0, D1, D2, D5
20	101 000	D3, D5
20,5	101 001	D0, D3, D5
21	101 010	D1, D3, D5
21,5	101 011	D0, D1, D3, D5
22	101 100	D2, D3, D5
22,5	101 101	D0, D2, D3, D5
23	101 110	D1, D2, D3, D5
23,5	101 111	D0, D1, D2, D3, D5
24	110 000	D4, D5
24,5	110 001	D0, D4, D5
25	110 010	D1, D4, D5
25,5	110 011	D0, D1, D4, D5
26	110 100	D2, D4, D5
26,5	110 101	D0, D2, D4, D5
27	110 110	D1, D2, D4, D5
27,5	110 111	D0, D1, D2, D4, D5
28	111 000	D3, D4, D5
28,5	111 001	D0, D3, D4, D5
29	111 010	D1, D3, D4, D5
29,5	111 011	D0, D1, D3, D4, D5
30	111 100	D2, D3, D4, D5
30,5	111 101	D0, D2, D3, D4, D5
31	111 110	D1, D2, D3, D4, D5
31,5	111 111	Toutes

## Annexe C – Simulations électroniques

Cette annexe présente les simulations réalisées dans le cadre du projet afin de valider le fonctionnement des circuits de commande d'alimentation pour l'atténuateur numérique ADRF5720. Elles ont été menées avec l'outil de simulation LTspice à différentes étapes du projet.

### C.1 – Etudes des transistors

La première étape de la phase de simulations était de bien comprendre le fonctionnement des transistors bipolaire et MOSFET. Cette compréhension s'est faite comme un rappel des cours de première année mais était nécessaire pour appréhender ce projet.

Pour visualiser et synthétiser ma compréhension de leurs utilisations j'ai réalisé le tableau ci-dessous avec les différents transistors.

Type	Polarité du courant	Tension de commande pour conduction
NPN	Du collecteur vers l'émetteur	Base plus positive que l'émetteur ( $V_{be} > 0.6V$ )
PNP	De l'émetteur vers le collecteur	Base plus négative que l'émetteur ( $V_{eb} < 0.6V$ )
NMOS	Du drain vers la source	Grille plus positive que la source ( $V_{gs} > V_{th}$ )
PMOS	Du source vers le drain	Grille plus négative que la source ( $V_{gs} < V_{th}, < 0$ )

### C.2 – Simulation finale (étages de commutation)

Objectif : Valider le circuit complet à double étage (commande de +3.3 V et -3.3 V) avec le bon séquencement des tensions imposé par l'ADRF5720.

Cela m'a permis de vérifier que les étages à transistor fonctionnent comme je l'attendais pour alimenter l'ADRF5720.

Dans la partie simulation électronique de mon rapport vous avez déjà les simulations et l'explication de ce montage. Mais vous trouverez ici les images de ces simulations :

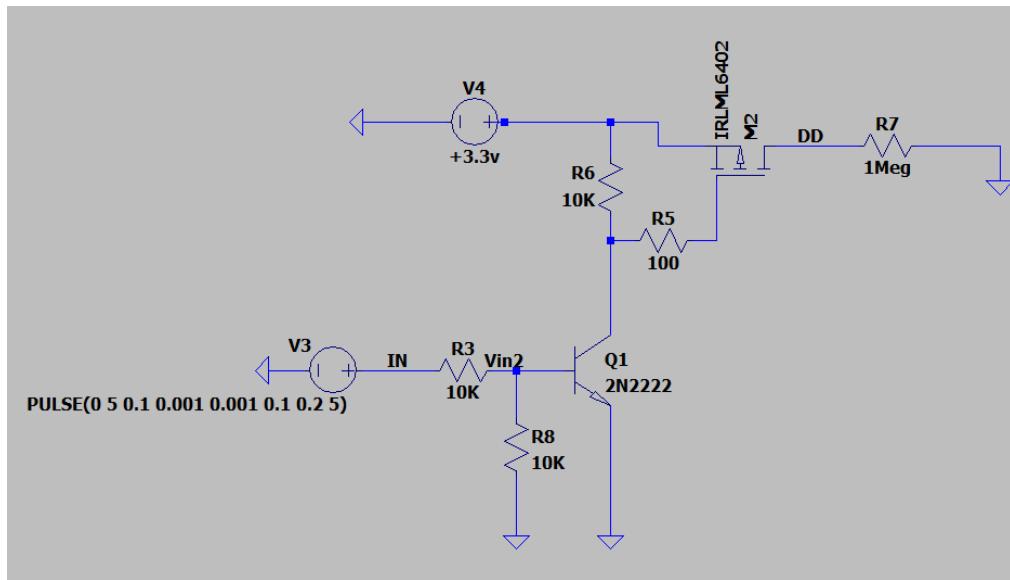


Figure 32 : Schéma simulation étage transistor +3.3V

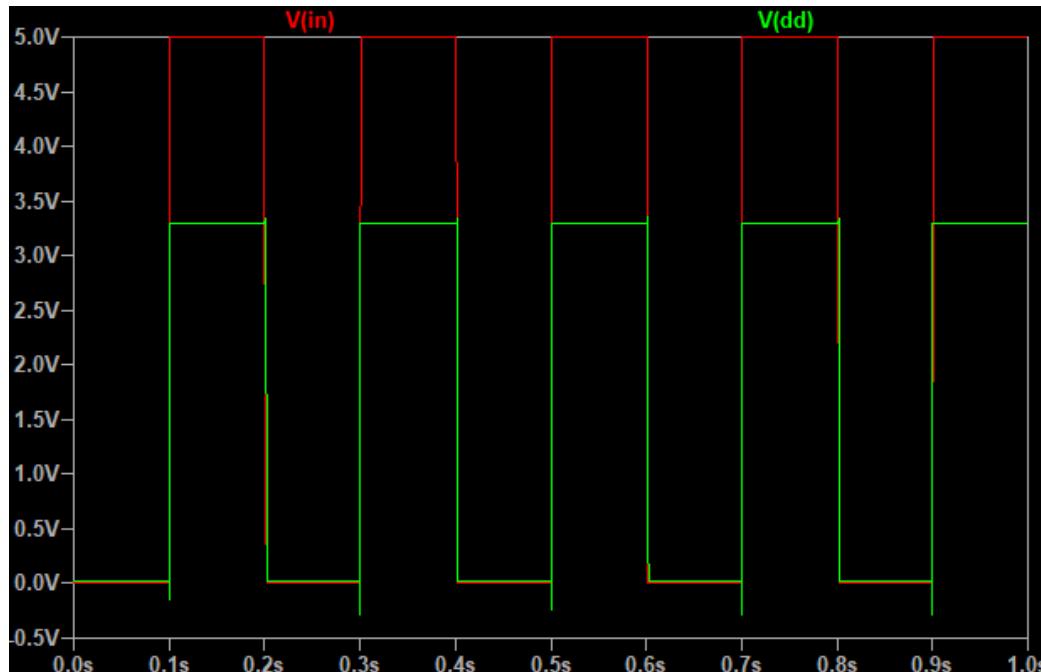


Figure 33 : Résultat de simulation de l'alimentation +3.3V

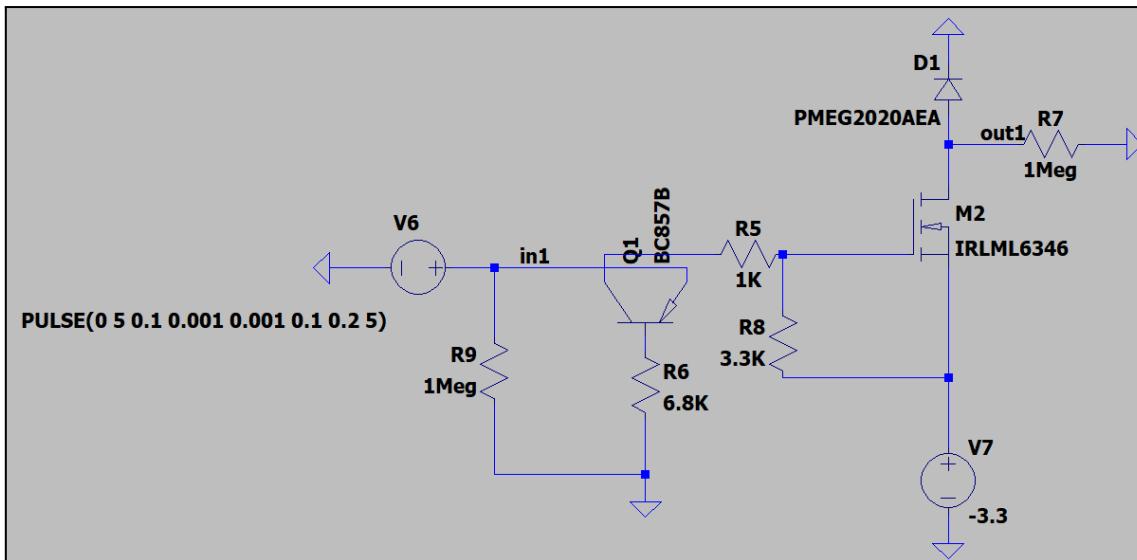


Figure 34 : Schéma simulation étage transistor –3.3 V

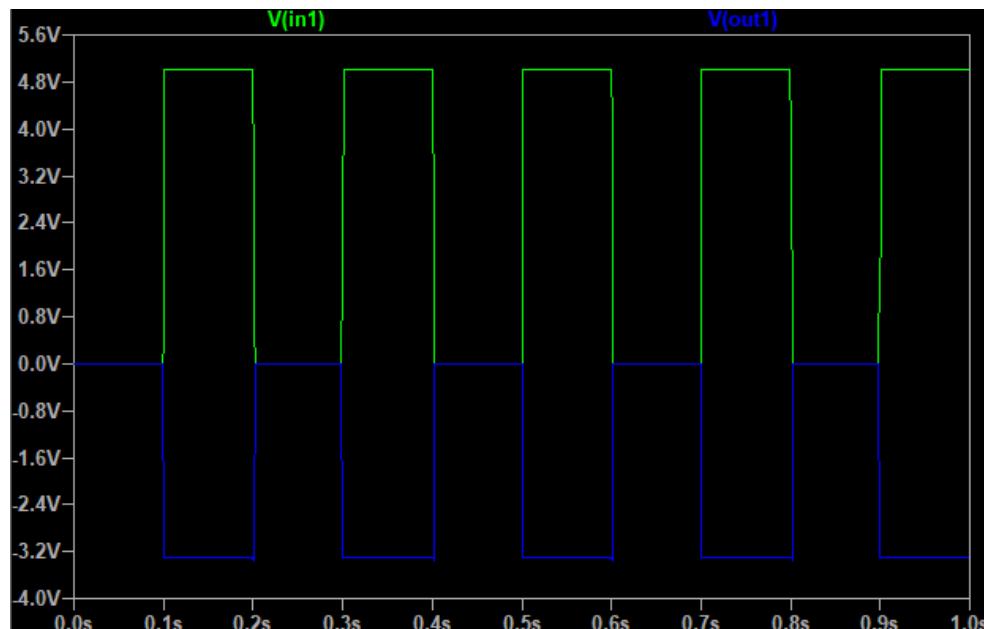


Figure 35 : Résultat de simulation de l'alimentation –3.3 V

### Explication de l'inverseur :

L'entrée  $V_{IN}$  est alimentée par le +3.3 V issu de l'étage de commande de VDD, ce qui garantit que la tension négative est générée uniquement après l'apparition du +3.3 V. Deux condensateurs électrolytiques (au tantale métallique) polarisés de 150  $\mu$ F sont utilisés : le premier entre CAP+ et CAP- qui assure le transfert d'énergie nécessaire à l'inversion de tension et le second, entre  $V_{OUT}$  et la masse pour stabiliser la sortie -3.3 V.

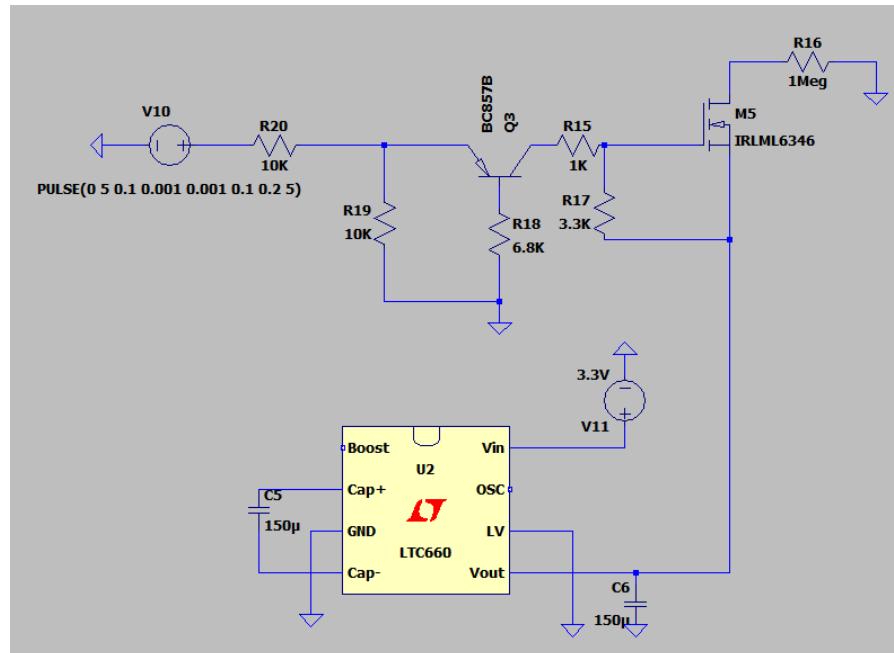


Figure 36 : Schéma simulation étage transistor -3.3 V avec inverseur de tension

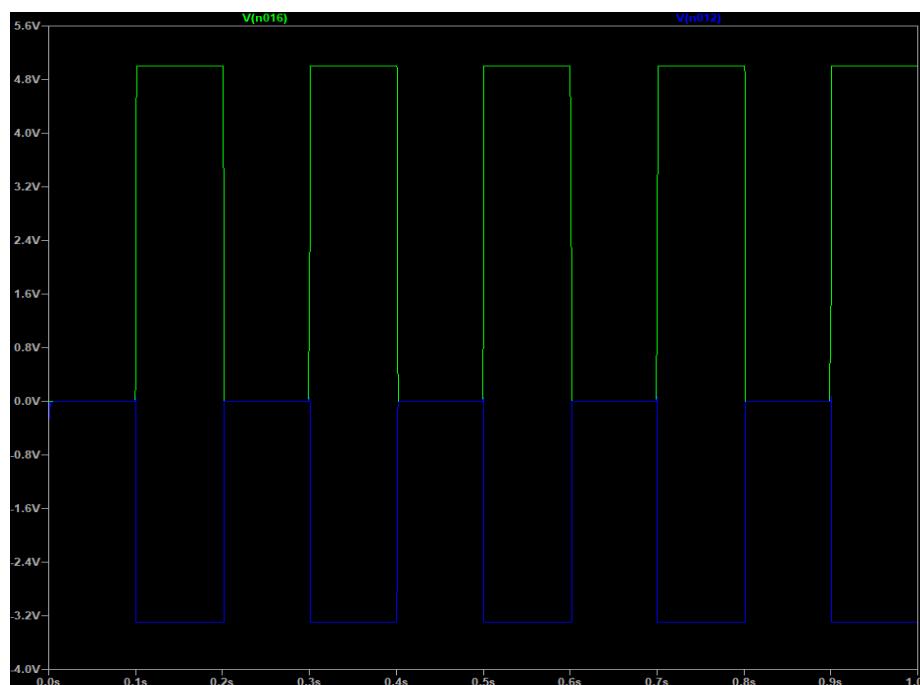


Figure 37 : Résultat de simulation de l'alimentation -3.3 V avec inverseur de tension

## Annexe D – Simulations logicielles sur Wokwi

Cette annexe regroupe les différentes simulations réalisées sur la plateforme Wokwi pour tester et valider les fonctionnalités du programme Arduino développé pour piloter l'atténuateur numérique ADRF5720.

### D.1 – Simulations fonctionnelles Arduino

Objectif est de valider par simulation :

- ❖ Les commandes power et att
- ❖ Le séquencement correct des broches d'alimentation (VDD avant VSS)
- ❖ Le bon encodage de l'atténuation sur les broches D0 à D5
- ❖ L'affichage en série (Serial Monitor) des états et erreurs
- ❖ La synchronisation des LED avec les bits transmis

Voici le lien pour accéder au site Wokwi : [New Arduino Uno Project - Wokwi Simulator](#)

Version	Fonctionnalité testée	Description / Objectif	Résultat
v1	Commande att	Vérification de l'encodage binaire et latch enable	Conforme
v1	Gestion des erreurs	Simulation d'entrées erronées pour valider le contrôle	Conforme
v2	Synchronisation LEDs	LEDs associées aux broches D0–D5 pour visualiser le codage	Conforme
v3	Fonctionnement global	Simulation complète incluant VDD/VSS (avec un temps entre les deux extrapolé pour le visualiser), OE, LE, et LEDs	Conforme

Version 1 : Commandes de base et gestion d'erreurs

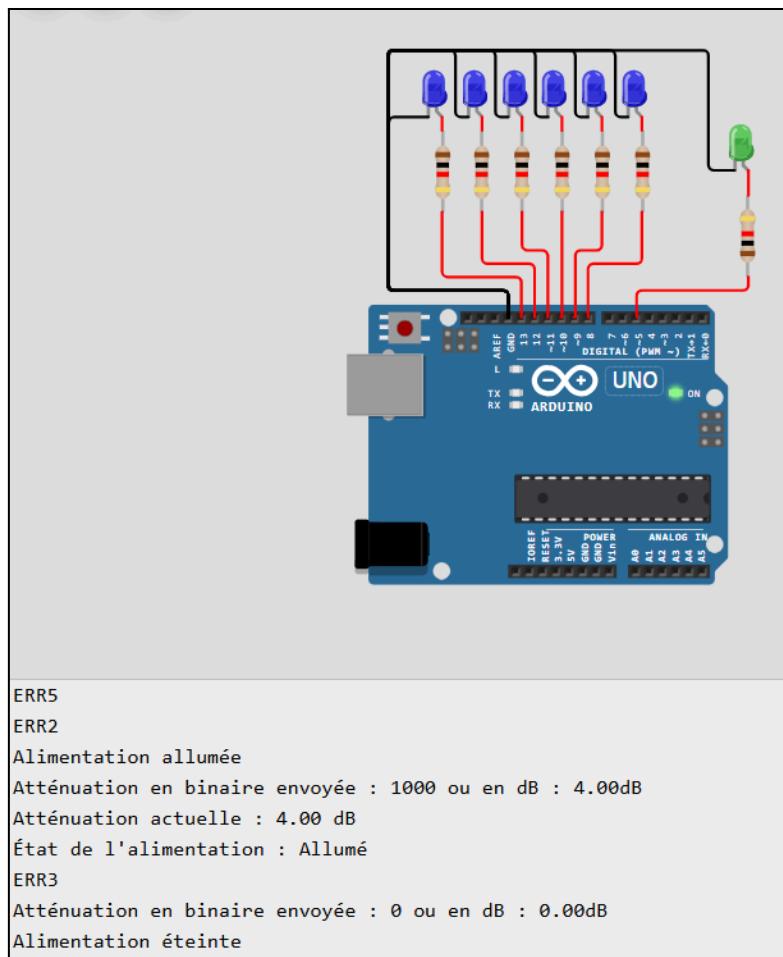


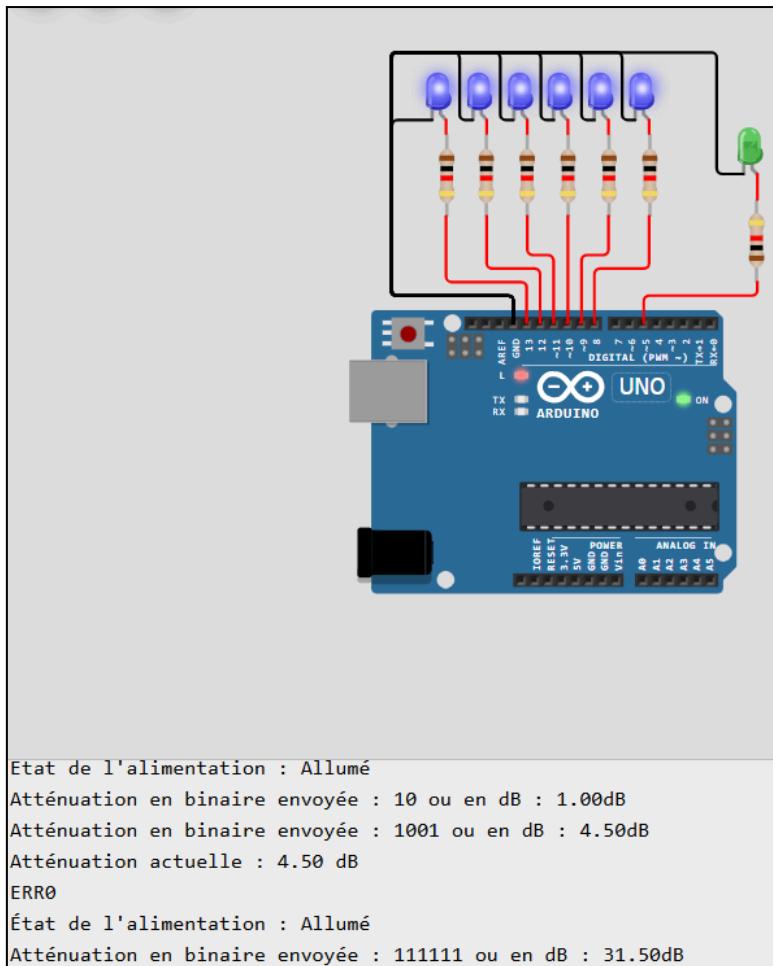
Figure 38 : Première séquence de test

Séquence réalisée :

```

att          // ERR5 : commande sans power
power on!    // ERR2 : syntaxe incorrecte
power on
att 4
att
power
att 4.5.3    // ERR3 : format invalide (trop de points)
att 0
power off

```



**Figure 39 : Deuxième séquence de test**

Séquence réalisée :

```

power on    // (non visible sur la capture)
power
att 10
att 4.5
att
dozhdj    //ERRO : commande inexistante
power
att 31.5

```

Ces tests ont permis de valider le bon fonctionnement des commandes et la détection automatique d'un grand nombre de cas d'erreur : valeurs manquantes, mauvaise syntaxe, entrées non reconnues, etc...

### Version 2 : Synchronisation LEDs avec les broches D0–D5

Sur la deuxième phase de test, j'ai fait cette séquence de commande :

power on

att 0.5 // Seule D0 allumée

att 31.5 // Toutes les LEDs allumées

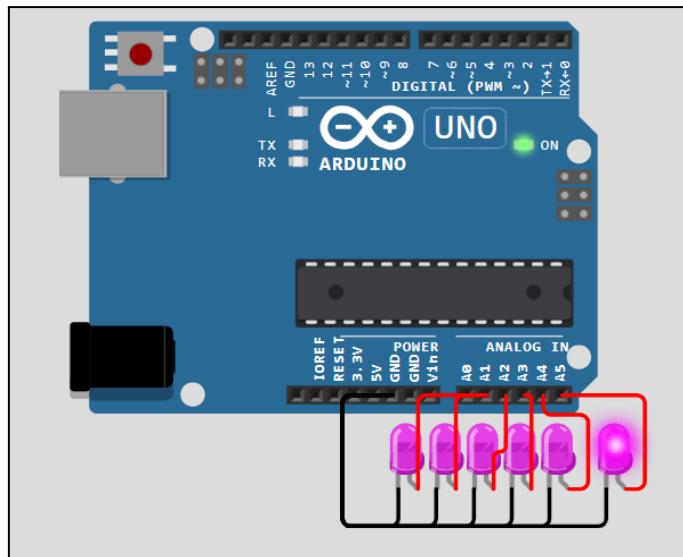


Figure 40 : Première commande d'atténuation (0.5dB) avec LEDs

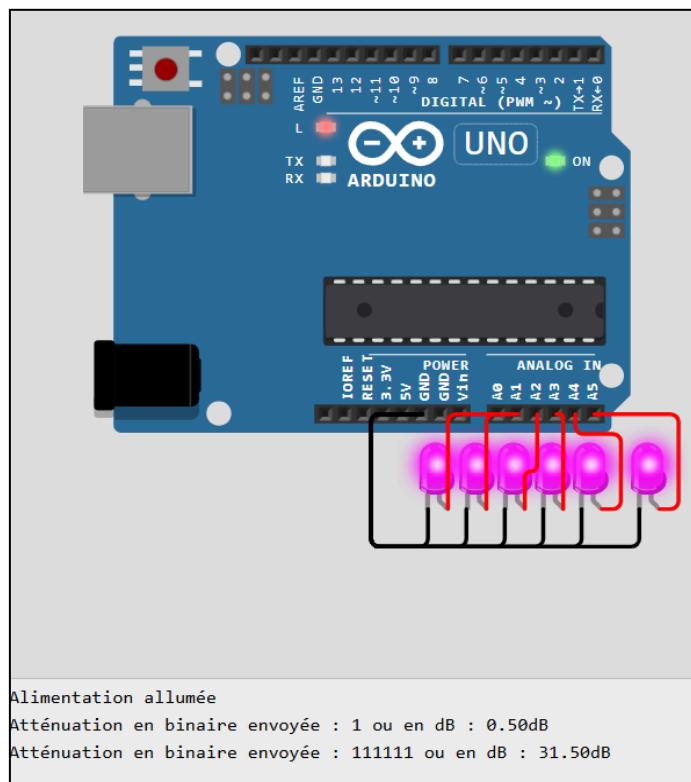


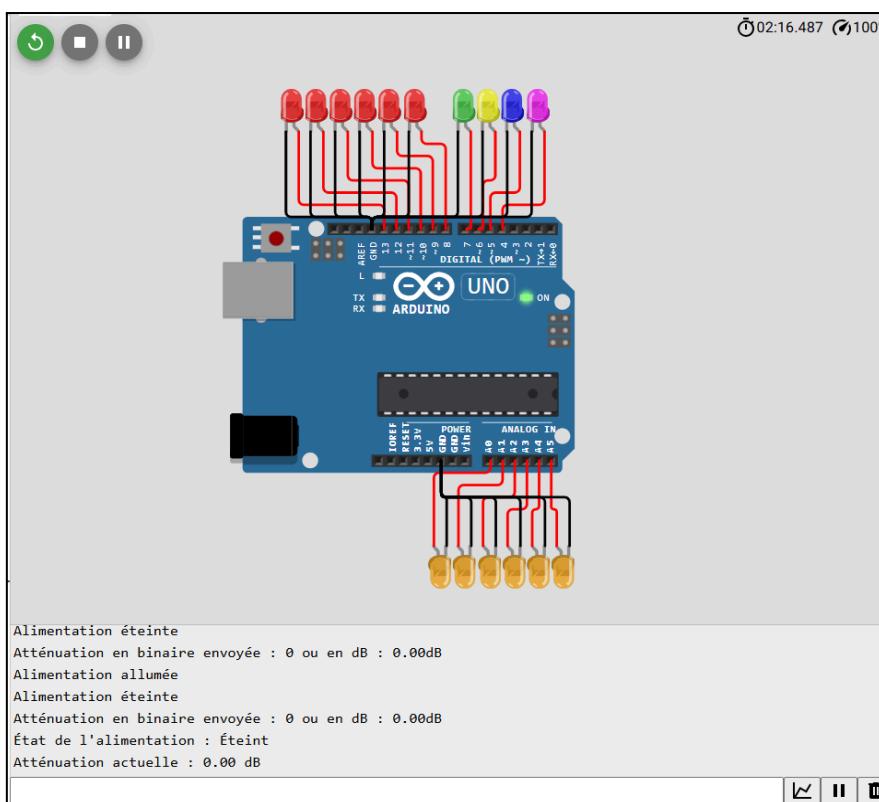
Figure 41 : Deuxième commande d'atténuation (31.5dB) avec LEDs

Cette simulation a permis de vérifier que les LEDs connectées aux broches D0 à D5 s'allument conformément à l'encodage binaire de l'atténuation. C'est un bon outil de visualisation directe.

### Versions 3 : Fonctionnement global

Ce test visait à valider le comportement global du système, en intégrant tous les éléments développés précédemment. Il permet de simuler à la fois :

- ❖ L'atténuation numérique (via les broches D0 à D5)
- ❖ L'activation du Latch Enable (LE)
- ❖ Le contrôle du Output Enable (OE) du translateur de niveau
- ❖ Le bon séquencement des alimentations : activation de VDD avant VSS



**Figure 42 : Simulation complète du fonctionnement**

Organisation des LEDs dans la simulation Wokwi :

- ❖ LEDs rouges : représentent les broches D0 à D5, donc l'atténuation binaire envoyée à l'ADRF5720.
- ❖ LEDs jaunes : simulent les LEDs physiques présentes sur le PCB final.
- ❖ LED verte : associée au Latch Enable (LE).
- ❖ LED jaune (supplémentaire) : représente le OE du translateur de niveau.
- ❖ LED bleue : commande simulée du -3.3 V (VSS).
- ❖ LED rose : commande simulée du +3.3 V (VDD).

Ce test a été concluant et prometteur :

- ❖ Tous les signaux se sont déclenchés dans l'ordre attendu.
- ❖ Les LEDs ont permis de visualiser la logique complète du système.
- ❖ Le moniteur série affichait correctement les états et retours d'erreur.

Cette version a donc marqué une étape importante dans la validation du code, avant de passer à la phase de test matériel réel.

#### Commentaires supplémentaires :

- L'environnement Wokwi s'est avéré pratique pour tester rapidement la logique du code, avant même de disposer du matériel réel.
- Bien que limité à des tensions standard (pas de simulation de -3.3 V par exemple), Wokwi a permis de valider l'ensemble des échanges logiques et des états internes de commande.

## Annexe E – Schémas et routages PCB

Cette annexe regroupe les différentes étapes de conception électronique matérielle du projet : schémas électriques, routage PCB, génération de fichiers Gerber et rendus 3D.

### E.1 – Versions de schéma électrique

Plusieurs versions du schéma ont été réalisées au fil du projet, principalement dans un objectif d'apprentissage progressif et d'amélioration du fonctionnement du circuit.

- Première version réalisée de manière autonome, pour gagner du temps et poser une base fonctionnelle.
- Revue de projet en équipe avec mes collègues de l'atelier : discussions sur des états indéterminés possibles, rôle des résistances, sécurisation des transitions logiques.

Modifications appliquées :

- ❖ Ajout ou modification de résistances de tirage (pull down sur l'étage de commande du +3.3V).
- ❖ Vérification des états de repos des transistors pour éviter tout comportement instable.
- ❖ Amélioration de la lisibilité et de la logique du schéma global.

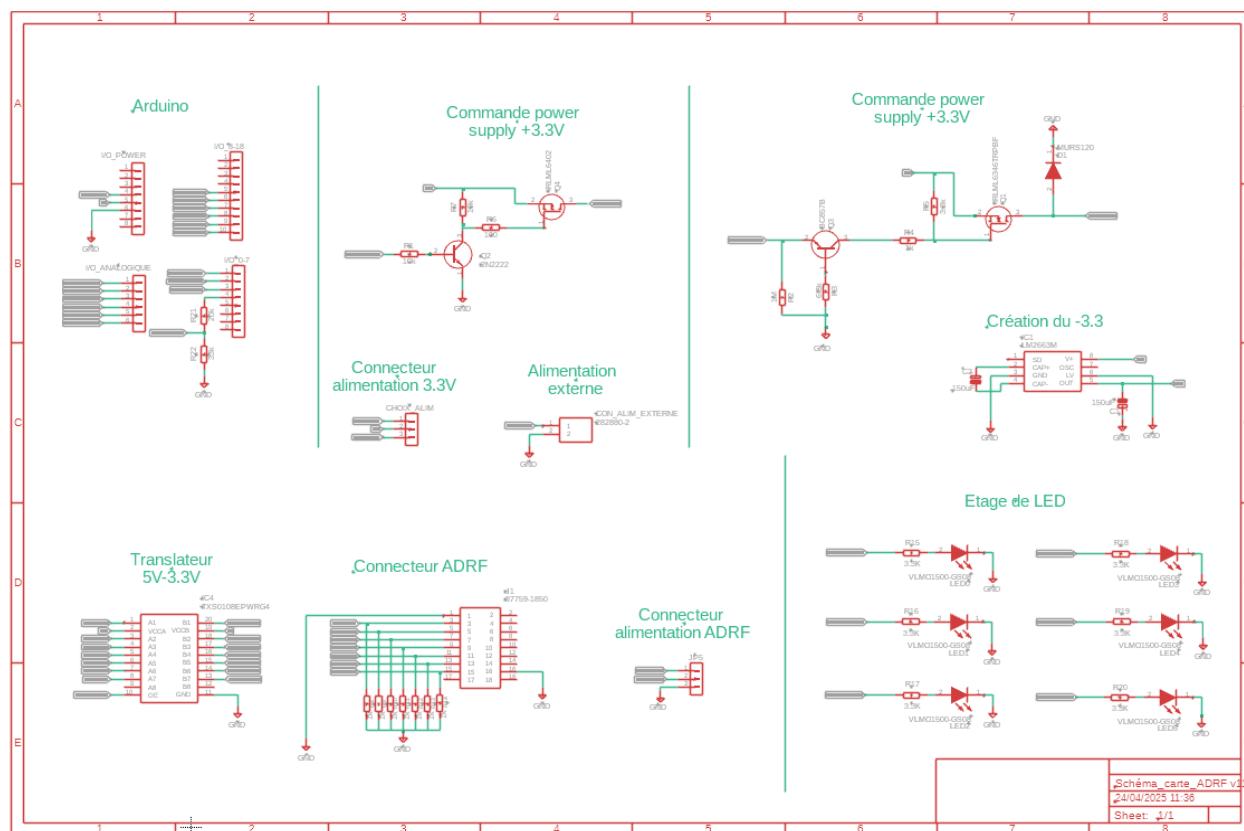


Figure 43 : Première version du schéma électrique

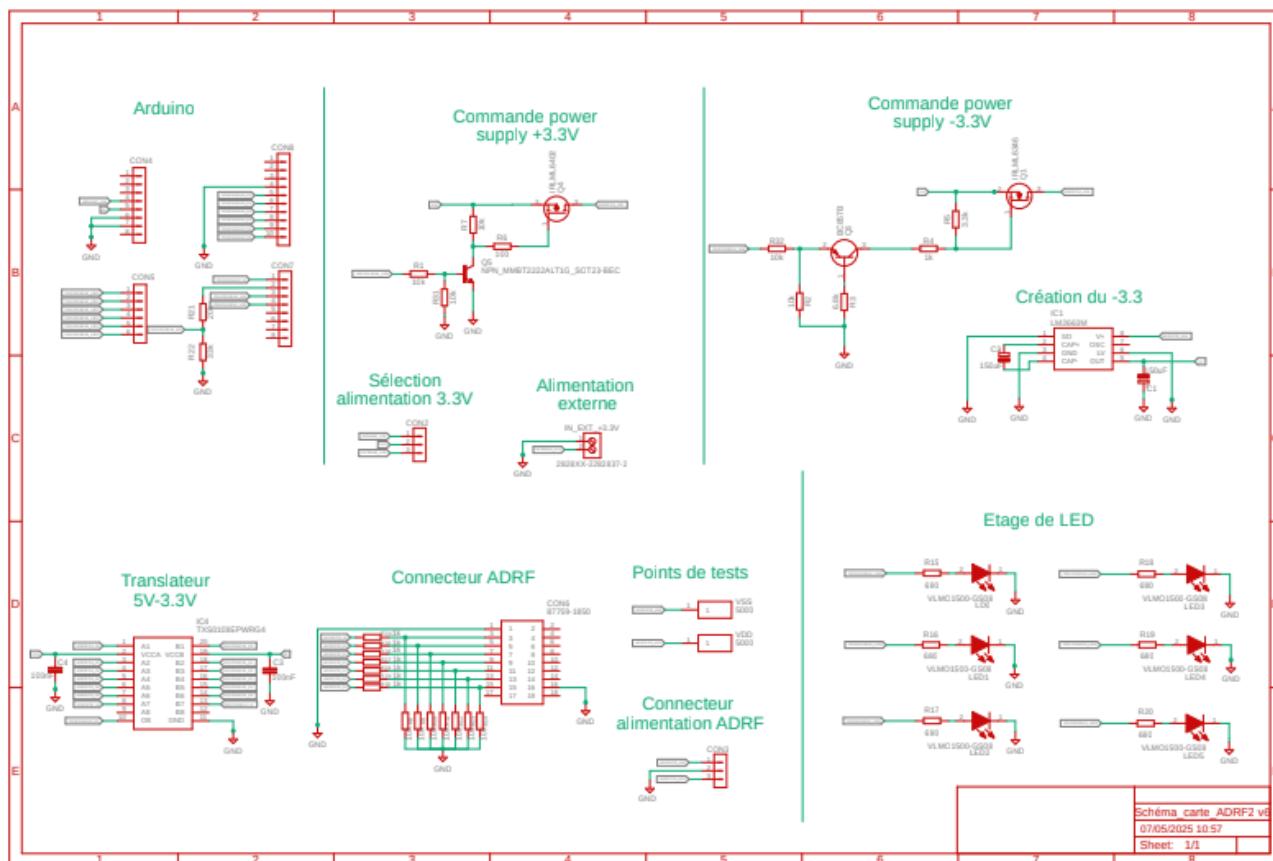


Figure 44 : Schéma final de la carte de contrôle

Vous retrouverez ce schéma au format pdf ici : [PDF Schéma électronique\\_ADRF5720.pdf](#)

## E.2 – Versions de routage

Le processus de routage du circuit imprimé a connu trois itérations distinctes, chacune correspondant à une phase d'expérimentation, de validation ou de prise de décision technique.

### Version 1 : Routage initial

- ❖ Réalisée rapidement dans le but d'avancer sur le projet.
- ❖ Composants placés et connectés de manière fonctionnelle, mais sans réelle réflexion sur la circulation des courants, les plans de masse ou la séparation des alimentations.
- ❖ Cette version m'a permis de m'entraîner sur ce nouvel outil et de poser une première base de travail.

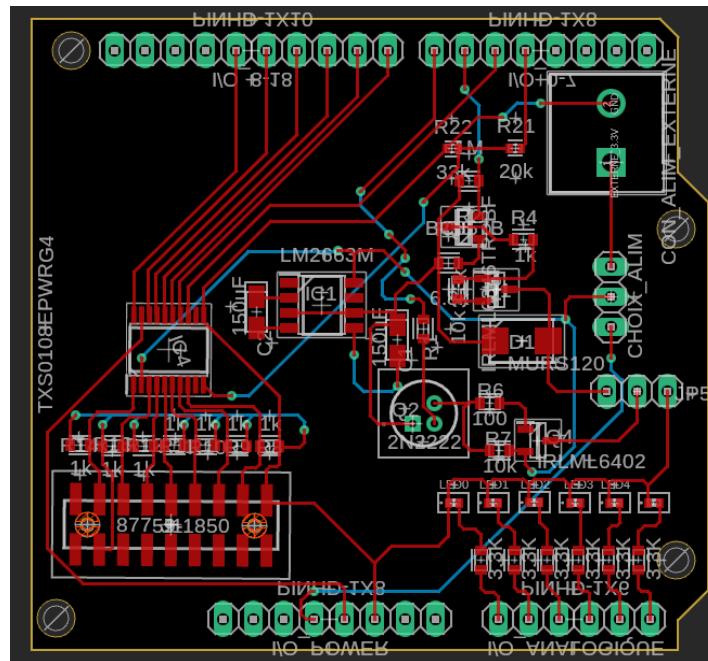


Figure 45 : Premier routage de la carte

### Version 2 : Routage avec plan de masse unique

- ❖ Première tentative de routage plus propre.
- ❖ Ajout d'un plan de masse unique.
- ❖ Les alimentations (+3.3 V / -3.3 V) sont routées avec des pistes classiques.
- ❖ Réalisée pour pouvoir la comparer à une autre version, dans l'attente du retour de mon maître de stage.

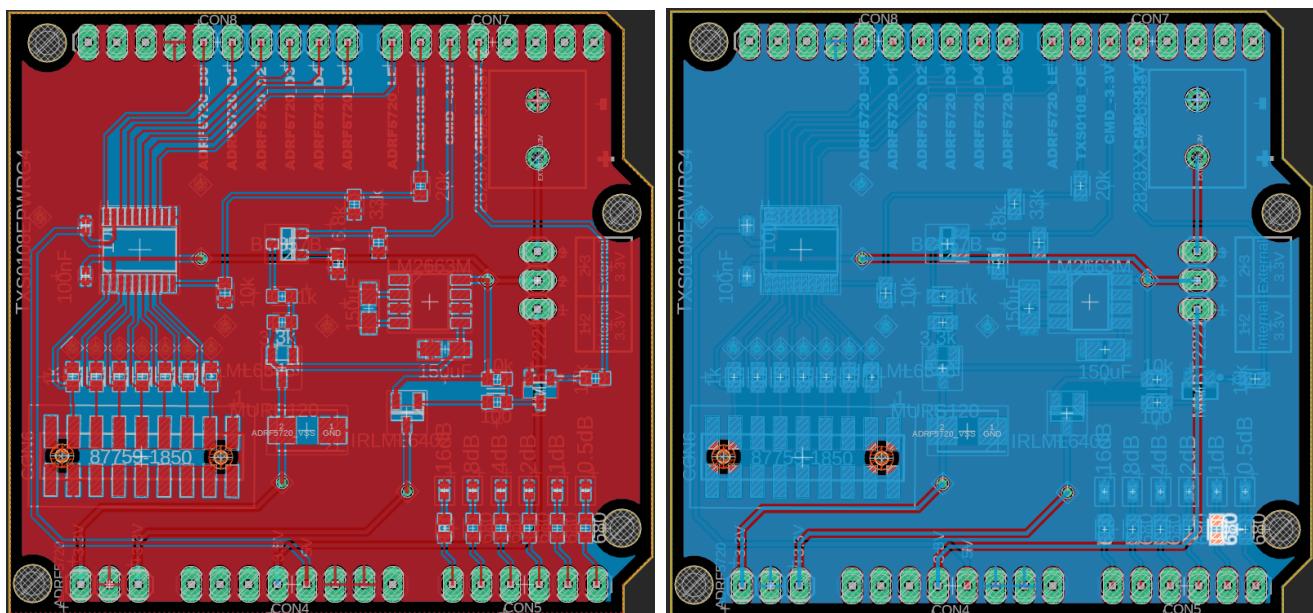


Figure 46 : Couche TOP et BOTTOM de la version 2

### Version 3 : Routage final avec plans d'alimentation

- ❖ Version retenue pour fabrication.
- ❖ Ajout de plans dédiés pour VDD et VSS, en plus du plan de masse.
- ❖ Cette architecture permet une meilleure stabilité électrique, une réduction du bruit et une distribution plus homogène des potentiels.
- ❖ Adoptée après discussion et validation avec mon maître de stage et les collègues.

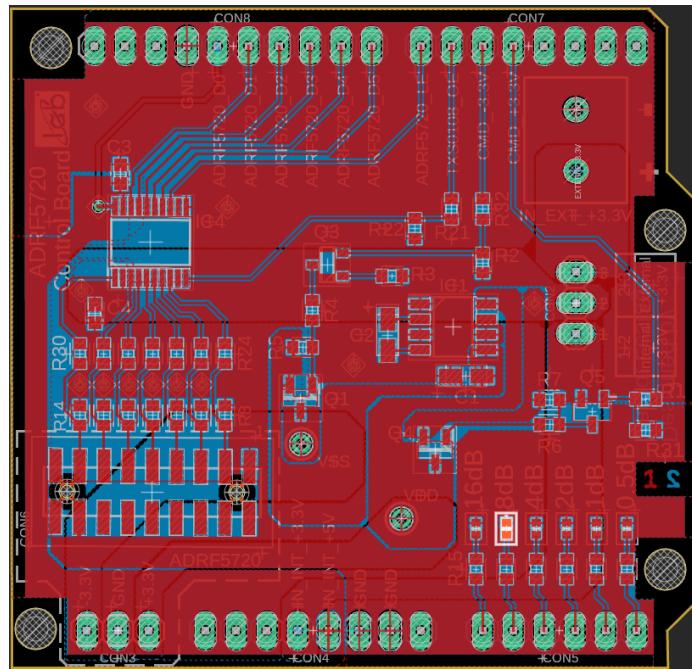


Figure 47 : Couche TOP de la version finale

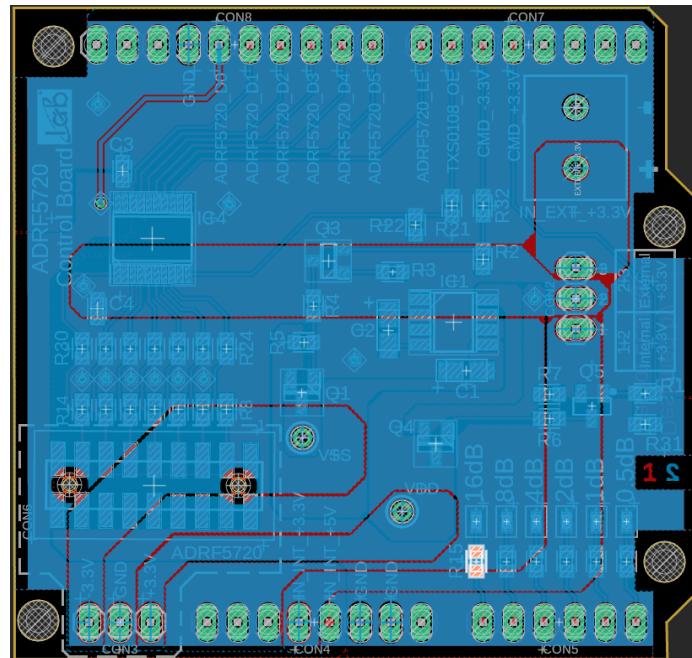
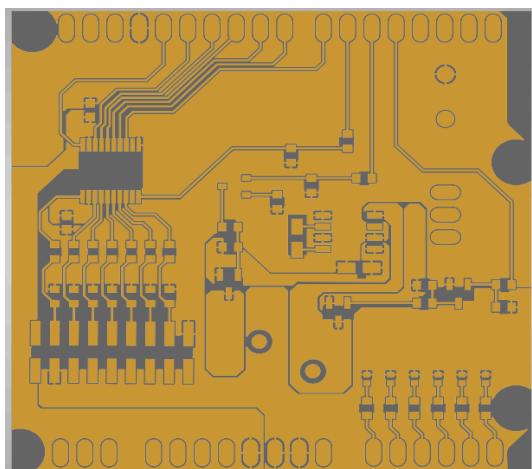


Figure 48 : Couche BOTTOM de la version finale

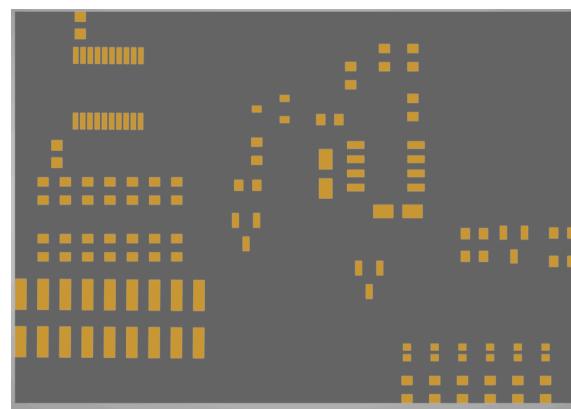
### E.3 – Gerber & 3D

Les fichiers Gerber ont été générés directement à partir de Fusion 360. Ils ont servi à la fois pour la visualisation du PCB et pour sa fabrication.

Pour vérifier la validité des fichiers avant envoi à l'atelier, j'ai utilisé le site [Online Gerber Viewer - PCB Prototype the Easy Way](#), qui permet de visualiser précisément les différentes couches (cuivre, sérigraphie, contours...). Vous trouverez ci-dessous des exemples de ce que j'ai pu visualiser.



**Figure 49 : Visualisation de la couche TOP COPPER**



**Figure 50 : Visualisation du solder paste TOP**

Afin de mieux m'organiser pour l'étape de fabrication, j'ai également constitué un dossier complet de fabrication, contenant :

- ❖ Les fichiers Gerber exportés depuis Fusion 360.
- ❖ La nomenclature des composants (BOM).
- ❖ La sérigraphie (Silkscreen) avec le nom des composants.
- ❖ Un schéma de connexions généré automatiquement depuis Fusion 360 pour faciliter l'assemblage.

Voici le dossier final de fabrication : [Dossier\\_fabrication\\_2025\\_05\\_13](#)

Le rendu de la carte en 3D sur Fusion 360 m'a permis :

- ❖ Vérifier les encombrements des composants (malgré le manque de certaine vue 3D des composants)
- ❖ Réfléchir à la sérigraphie (silkscreen) en tenant compte de la taille et de l'emplacement
- ❖ Vérifier l'accessibilité des connecteurs et du bornier pour s'assurer qu'aucun obstacle mécanique ne gêne leurs utilisations.

De la même manière que pour le routage, j'ai systématiquement vérifié la vue 3D du PCB à chaque étape d'évolution. Cela m'a permis d'anticiper des erreurs d'encombrement ou d'implantation avant même la fabrication.

Vous trouverez ci-dessous les vues 3D correspondantes aux trois versions successives du routage :

### Version 1 :

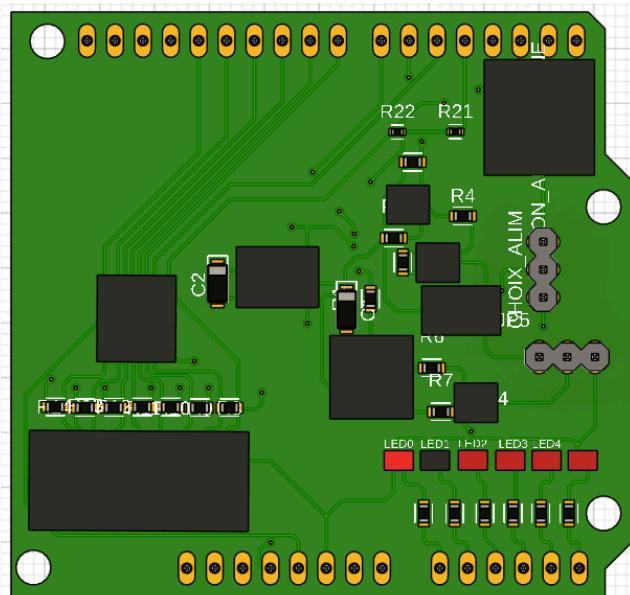


Figure 51 : Vue 3D de la couche TOP

### Version 2 :

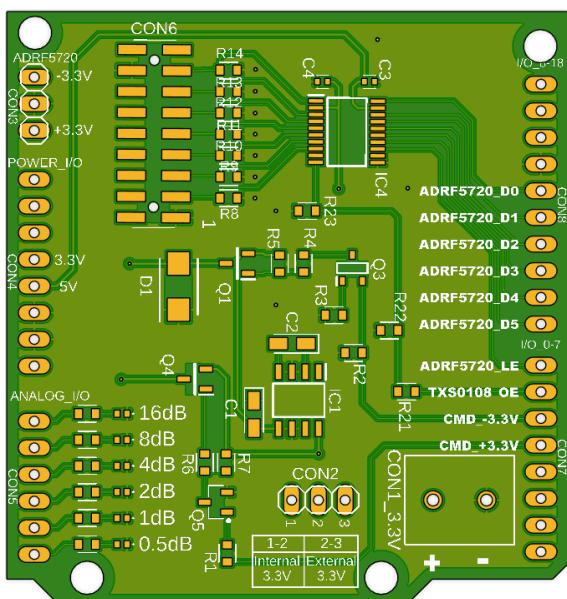


Figure 52 : Vue 3D couche TOP sans les packages des composants

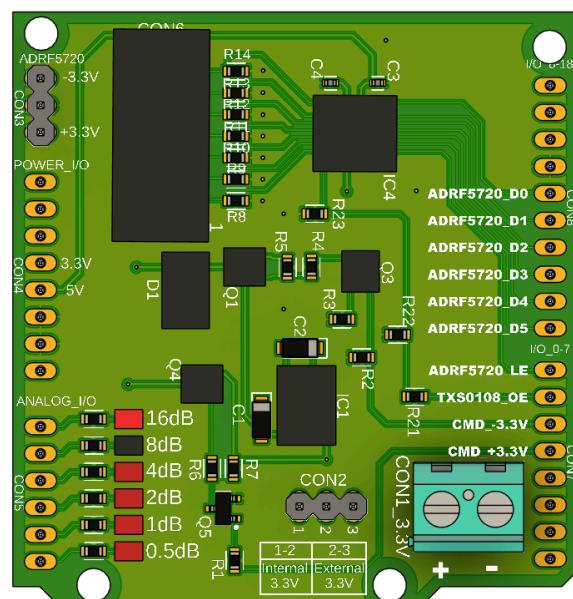
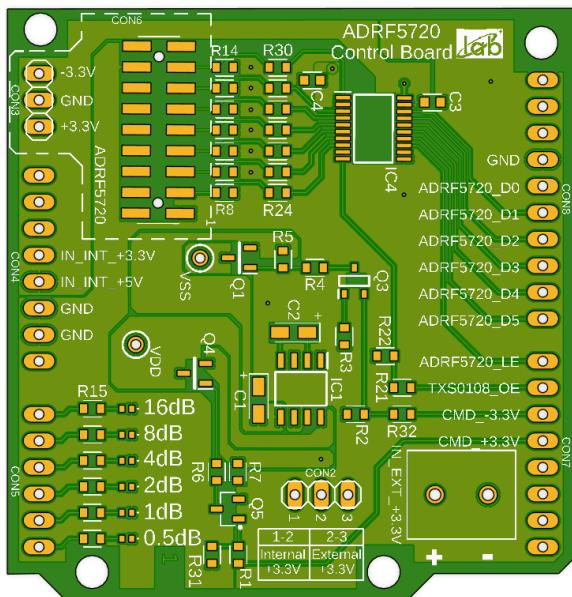
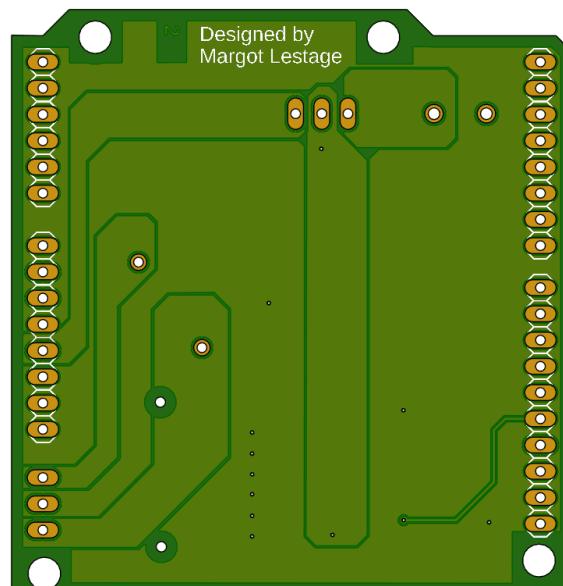


Figure 53 : Vue 3D couche TOP avec les packages des composants

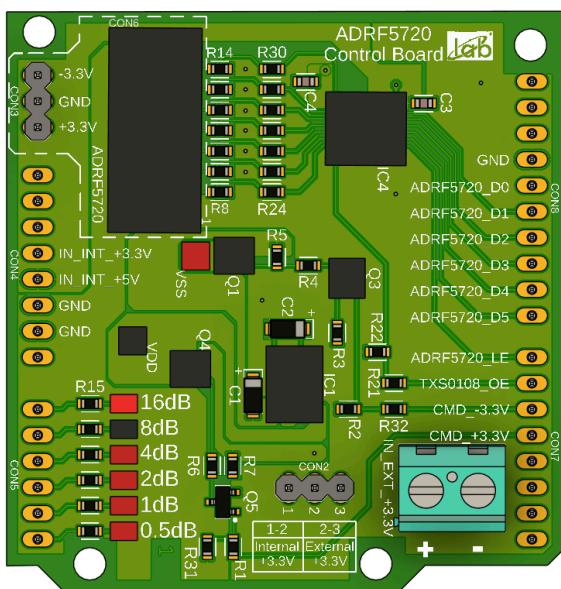
**Version 3 :**



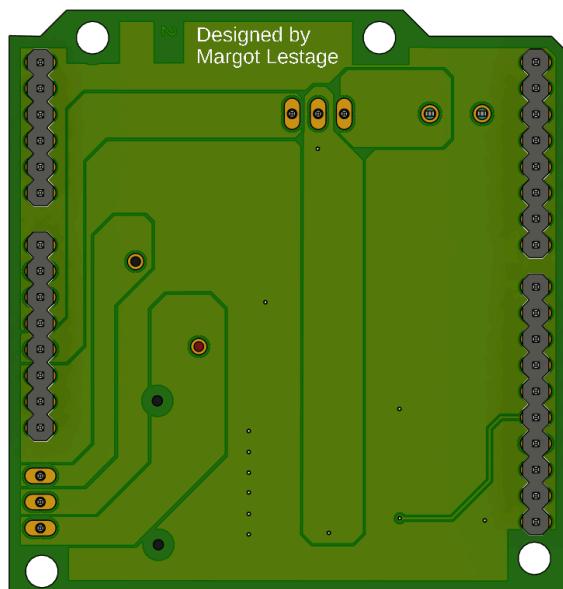
**Figure 54 : Vue 3D couche TOP sans les packages des composants**



**Figure 55 : Vue 3D couche BOTTOM sans les packages des composants**



**Figure 56 : Vue 3D couche TOP avec les packages des composants**



**Figure 57 : Vue 3D couche BOTTOM avec les packages des composants**

## Annexe F – Tests et validations

Tous les tests ont été réalisé suivant un protocole précis, vous trouverez toutes les images et vidéos prise durant cette phase regroupé dans le dossier ci-contre: [Image\\_vérification](#) et les différents protocoles de test sont décrit dans le document suivant : [ADRF5720\\_VERIF](#) Ce document permet de regrouper les différentes étapes de test et montre ma capacité de vérification.

Voici les différentes étapes de test et l'ordre dans lequel j'ai procédé :

1. Vérification de l'impédance entre les alimentations (tests de continuité)
2. Vérification du comportement de la carte de contrôle sans l'Arduino
3. Vérification des commandes avec l'Arduino via le firmware

### F.1 - Fabrication des câbles de test

Pour pouvoir réaliser les vérifications préliminaires (avant de connecter la carte à l'Arduino Uno), j'ai conçu et fabriqué des câbles permettant d'alimenter la carte via une alimentation de table. La fabrication de ces câbles s'est faite selon les étapes suivantes :

- Découpe de la longueur désiré de fil
- Dénudé une extrémité (2 mm)
- Serti une fiche métallique à l'aide d'une pince adaptée
- Protégé à l'aide d'une gaine thermorétractable pour éviter les courts-circuits
- Dénudé l'autre extrémité (3 cm), replié le fil, vissé dans un embout banane
- Protégé la zone avec la gaine plastique de l'embout banane

Cette étape a été très formatrice : c'était la première fois que je réalisais ce type de câblage. J'ai pu découvrir l'utilisation de plusieurs outils que je ne connaissais pas auparavant, comme le pistolet à dénuder ou encore l'appareil chauffant pour gaine thermorétractable.



Figure 58 : Pistolet à dénuder utilisé lors des câblages

## F.2 - Problèmes rencontrés et corrections

### **Court-circuit entre plans d'alimentation :**

Lors de la première série de tests, un court-circuit a été détecté entre le plan +3,3 V et la masse (GND). L'origine du défaut se situait au niveau du condensateur C4, où les deux plans étaient confondus. Ce problème est sûrement dû aux règles de routage qui n'avaient pas été correctement paramétrées dans Fusion 360.

Ce court-circuit a été détecté avant la mise sous tension de la carte, ce qui a évité toute détérioration.

Pour résoudre le problème :

- Une isolation manuelle des deux plans d'alimentation a été réalisée à l'aide d'un cutter sans endommager les pistes voisines.
- Une vérification à l'ohmètre a permis de confirmer que le court-circuit était bien éliminé.

### **LED grillée :**

Un autre problème est apparu lors de la vérification du fonctionnement des LEDs. L'une d'elles ne s'allumait pas lorsqu'elle était alimentée en 5 V.

En testant à l'aide d'un multimètre en mode diode, j'ai confirmé que la LED était grillée, probablement à cause de la chaleur générée lors des ajustements suite à l'assemblage.

Pour corriger cela :

- Nous avons retiré la LED défectueuse à l'aide d'une pince chauffante, adaptée aux composants 0402.
- Une nouvelle LED a été positionnée directement sur l'étain de l'ancienne puis soudée.
- Un test de fonctionnement a validé la réparation.

## Glossaire

### Définitions :

Astrophysique : Branche de l'astronomie qui étudie la nature physique des astres et des phénomènes spatiaux (composition, interactions, évolutions).

Astronomie : Science qui étudie les astres, les corps célestes (y compris la Terre) et la structure globale de l'univers.

Astre : Corps céleste naturel visible, comme une étoile, une planète ou un satellite.

Hardware (matériel) : Ensemble des composants physiques d'un système informatique ou électronique (circuits, cartes, composants).

Software (logiciel) : Ensemble des programmes et applications informatiques exécutés par un système.

Firmware : Logiciel de bas niveau stocké dans une mémoire non volatile, qui contrôle directement le matériel et assure son fonctionnement autonome.

PLL (Phase-Locked Loop) : Boucle à verrouillage de phase utilisée pour générer des signaux synchronisés en fréquence et phase, essentielle dans les circuits de communication et d'horloge.

FPGA (Field Programmable Gate Array) : Circuit intégré programmable utilisé pour le traitement des données numériques en temps réel.

GUI (Graphical User Interface) : Interface graphique utilisateur permettant d'interagir facilement avec un logiciel ou un système.

DC Block : Dispositif supprimant la composante continue d'un signal.

PLL (Phase-Locked Loop) : Circuit électronique générant un signal d'horloge stable et haute fréquence à partir d'une fréquence d'entrée plus faible, essentiel pour synchroniser les composants comme les ADC.

Interférométrie : Technique de mesure et d'imagerie qui combine les signaux reçus par plusieurs antennes pour améliorer la résolution spatiale.

Convertisseur Analogique-Numérique (ADC) : Dispositif qui transforme un signal électrique analogique en une valeur numérique pour traitement informatique.

Module DGSPOT : Module spécifique du projet WIFP chargé de numériser et traiter les signaux issus du Front End d'ALMA avant transmission au corrélateur.

Corrélateur (ATAC) : Dispositif traitant et combinant les signaux numériques provenant des antennes d'un réseau d'interférométrie comme ALMA.

### **Acronymes et sigles :**

LAB : Laboratoire d'Astrophysique de Bordeaux

UMR : Unités Mixtes de Recherche

CNRS : Centre National de la Recherche Scientifique

OASU : Observatoire Aquitain des Sciences de l'Univers

ALMA : Atacama Large Millimeter/submillimeter Array

MSL : Mars Science Laboratory (mission NASA)

ESA : European Space Agency

NASA : National Aeronautics and Space Administration

JAXA : Japan Aerospace Exploration Agency (à rajouter si mentionnée)

LATMOS : Laboratoire Atmosphères, Milieux, Observations Spatiales

LIRA : Laboratoire d'Instrumentation et de Recherche en Astrophysique

INSU : Institut National des Sciences de l'Univers

SEII : Systèmes Embarqués et Informatique Instrumentale

IRAM : Institut de RadioAstronomie Millimétrique

IRAP : Institut de Recherche en Astrophysique et Planétologie

CNES : Centre National d'Études Spatiales

NRAO : National Radio Astronomy Observatory

SKA : Square Kilometre Array



Atacama Large  
Millimeter/submillimeter  
Array



NAOJ : National Astronomical Observatory of Japan

JAO : Joint ALMA Observatory

ADC : Analog to Digital Converter

PCB : Printed Circuit Board (carte électronique)

WIFP : WSU IF Processor (projet spécifique au laboratoire pour ALMA2030)

WSU : Wideband Sensitivity Upgrade (campagne d'amélioration de l'observatoire ALMA)

FPGA : Field Programmable Gate Array

GUI : Graphical User Interface

MMX : Martian Moon eXploration

## Sitographie

SKA (Square Kilometre Array) :

- [Le projet](#)
- [SKA](#)
- [SKAO](#)

ALMA (Atacama Large Millimeter/submillimeter Array) :

- [ALMA Observatory](#)
- [ALMA - Atacama Large Millimeter/submillimeter Array | ESO](#)

Laboratoire d'Astrophysique de Bordeaux : [Laboratoire d'astrophysique de Bordeaux](#)

Interférométrie, définition et concepts : [Définition | Interférométrie | Futura Sciences](#)