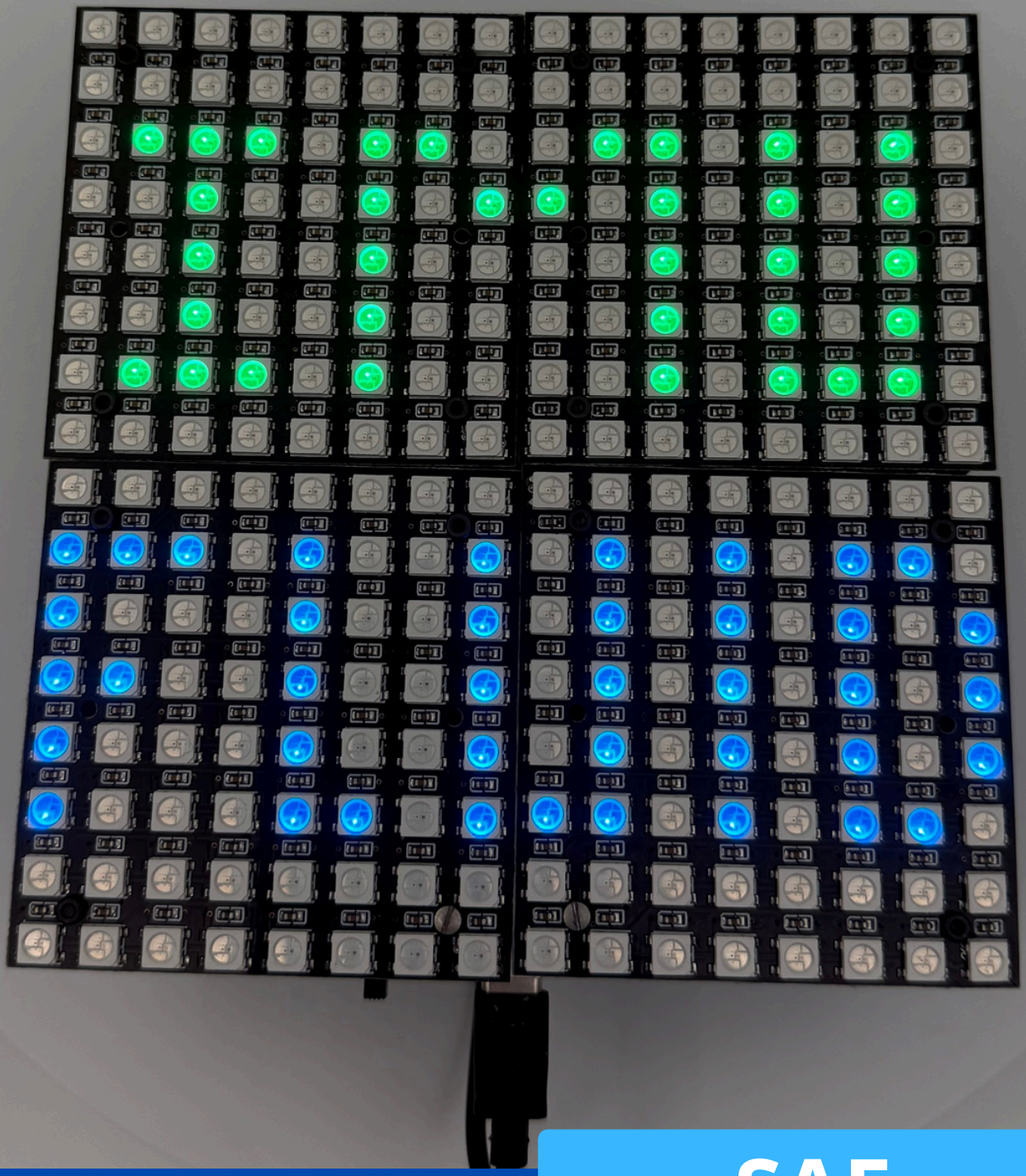
INSTITUT UNIVERSITAIRE DE TECHNOLOGIE
UNIVERSITÉ D'ANGERS

université angers



SAE

IMU FLUID

SAILLARD Paul
CASTRE Léo
G330

Encadré par
Philippe LUCIDARME
& Hélène BONNIN
4 Boulevard Lavoisier, 49100 Angers

2024-2025

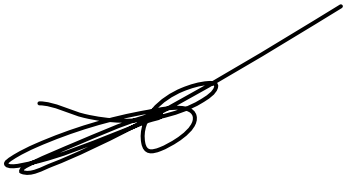
Décharge de responsabilités

Ce rapport présente le travail réalisé par un groupe d'étudiants dans le cadre d'un projet pédagogique. Les auteurs et l'Université d'Angers ne garantissent pas que l'information, les documents, la méthodologie et le matériel présentés dans ce document soient complets, conformes à l'état de l'art et exacts ni n'assurent en toutes circonstances la sécurité des biens, des personnes et des utilisateurs. Les auteurs et l'Université d'Angers ne seront pas tenus responsables des dommages éventuels qui pourraient résulter de l'utilisation du contenu du présent rapport.

Licence CC Attribution

SAILLARD Paul, CASTRE Léo, auteurs du présent rapport, nous opposons à la diffusion publique de ce rapport et de notre travail, en dehors du cadre pédagogique et d'évaluation.

Remerciements



Nous tenons tout d'abord à remercier M. LUCIDARME et Mme BONNIN pour avoir accepté de nous accompagner dans ce projet. Mais aussi d'avoir mis à disposition des matrices de LED ainsi qu'une centrale inertielle, ce qui nous a permis d'obtenir un rendu final de qualité. Enfin, sa disponibilité nous a beaucoup aidé lors de la réalisation de ce projet.

Nous tenons également à remercier Mme DENECHEAU pour l'envoi en fabrication de notre carte électronique ainsi que pour les différentes empreintes dessinées pour le projet.

Sommaire

Glossaire

Centrale inertielle (IMU - Inertial Measurement Unit) :
Module électronique regroupant plusieurs capteurs (accéléromètre, gyroscope, magnétomètre, capteur d'angle) permettant d'analyser les mouvements et l'orientation dans l'espace.

Diagramme de GANTT :
Diagramme de gestion de projet permettant de visualiser les différentes étapes et leur avancement au cours du temps.

EAGLE :
Logiciel utilisé pour la conception et le routage des circuits imprimés du projet.
Arduino IDE : Environnement de développement utilisé pour programmer l'ESP32 C6 et tester le comportement du système.

Entretoises mécaniques :
Composants servant à fixer les matrices de LED au circuit imprimé en assurant la solidité structurelle.

ESP32 C6 :
SoC utilisé pour les fonctionnalités de la centrale inertielle, avec des possibilités de connexions WiFi et de faible consommation. C'est l'un des plus petit SOC de la gamme.

I2C (Inter-Integrated Circuit) :
Protocole de communication série utilisé pour relier les capteurs et l'ESP32 C6 avec un minimum de câblage.

LED :
"Light-emitting diode" ou DEL : "Diode électroluminescente".

RGB :
Les LED RGB ("Red Green Blue") peuvent changer de couleur en fonction du pourcentage ajouté à chacune des trois couleurs primaires. En effet, de multiples combinaisons existent et peuvent être utilisées.

Routage (PCB) :
Processus de conception du circuit imprimé permettant d'organiser et de connecter les composants électroniques entre eux.

SoC (System on Chip) :
Puce qui rassemble (circuit intégré) différents composants, comme : des microprocesseurs, des mémoires (EEPROM, flash, ...), des périphériques d'interface (Wifi, BlueTooth, ...), etc.

TDAH (Trouble du Déficit de l'Attention avec ou sans Hyperactivité) :
Trouble neurologique affectant l'attention et l'impulsivité, pour lequel le projet IMU FLUID propose une solution interactive et visuelle.

UART (Universal Asynchronous Receiver-Transmitter) :
Interface de communication série utilisée comme alternative à I2C pour échanger des données entre composants électroniques.

Introduction

Imaginez un outil qui transforme le mouvement en une expérience visuelle captivante. Et si cette technologie pouvait aussi aider les personnes atteintes de Trouble du Déficit de l'Attention avec ou sans Hyperactivité (TDAH) ? Dans le cadre de ce projet, il a été décidé de concevoir cette technologie : un module électronique capable de simuler le mouvement d'un fluide.

L'objectif principal de cette réalisation est de proposer un outil interactif qui transforme le mouvement en une expérience visuelle captivante, tout en ayant un impact bénéfique sur la concentration. Ce projet s'inscrit dans une démarche innovante visant à associer l'électronique à une problématique actuelle : le Trouble du Déficit de l'Attention avec ou sans Hyperactivité. En effet, selon le journal Le Monde, environ trois pour cent de la population mondiale est concernée par ce trouble. Bien que plusieurs outils existent déjà pour aider à améliorer l'attention, comme les fidget spinners, peu d'entre eux utilisent l'électronique.

Figure 1 - Fidget spinner

Ainsi, notre projet consiste à concevoir un dispositif intégrant des LED et un affichage dynamique afin de reproduire de manière réaliste les mouvements d'un fluide. L'idée est d'utiliser un capteur de position angulaire pour détecter les mouvements de l'utilisateur et d'adapter en temps réel l'animation réaliste. L'un des principaux défis techniques de ce projet repose sur la gestion fluide des animations et l'optimisation du code pour assurer une réactivité optimale du système.

La première approche envisagée consistait à utiliser des LED bleues en cercle afin d'obtenir des vagues d'eaux. Cependant, afin d'enrichir notre apprentissage en conception électronique et en programmation, il a été décidé d'opter pour un affichage sur des matrices de LED RGB carré, permettant un affichage plus dynamique. Cette solution, bien que plus complexe à mettre en œuvre, offre un potentiel créatif plus vaste et permet d'obtenir un effet plus immersif.

Enfin, un premier croquis du projet a été réalisé afin d'imaginer son design et son ergonomie. À travers ce projet, nous souhaitons démontrer le potentiel de l'électronique dans l'amélioration du bien-être et de l'attention, tout en développant nos compétences en conception matérielle et logicielle.

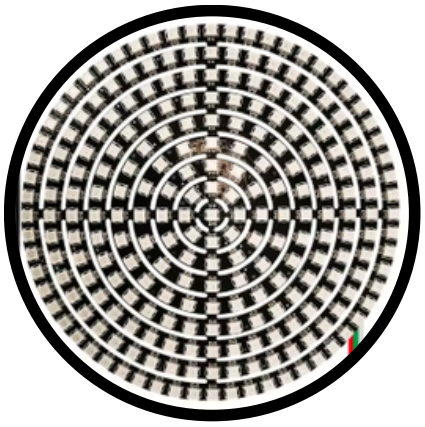


Figure 2 - Croquis prévisionnel

1) Cahier des charges

1.1) Contexte :

Ce projet s'inscrit dans le cadre de la SAE du semestre 6, il a pour objectif la conception et la réalisation d'un module de simulation visuelle de fluide ainsi que la possibilité de l'adapter à différents usages tels qu'un jeu de snake ou une boussole.

1.2) Fonctions principales :

- Mesure de la position angulaire du module
- Affichage et gestion d'animation d'un fluide

1.3) Interactions avec l'utilisateur :

- Interrupteur deux positions pour l'allumage
- Matrices de LED pour l'affichage
- Capteur de position angulaire pour la détection de l'inclinaison

1.4) Schéma fonctionnel du projet :

Les fonctions principales du projet se résument en seulement deux entrées. La première est le capteur qui communique en I2C et la deuxième est le switch d'alimentation. Enfin, une sortie visuelle sur les matrices de LED, le tout géré par un SOC.

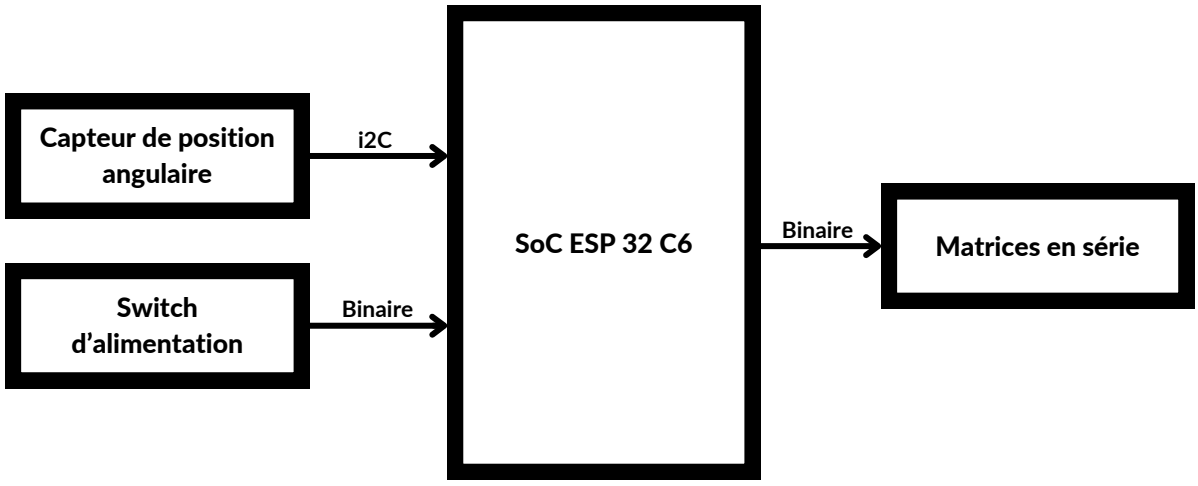


Figure 3 - Schéma bloc fonctionnel de l'architecture du circuit électronique

1.5) Schéma structurel du projet :

La structure du projet a été optimisé afin d'avoir le moins d'alimentation et de bus de communication possible. Les modules peuvent tous fonctionner en 5V ou 3.3V alors que les matrices ne peuvent fonctionner qu'en 5V. Il a alors été décidé de retirer le régulateur 5V-3.3V initialement prévu afin d'alimenter l'ensemble en 5V. Le bus I2C a également été choisi pour sa simplicité d'utilisation.

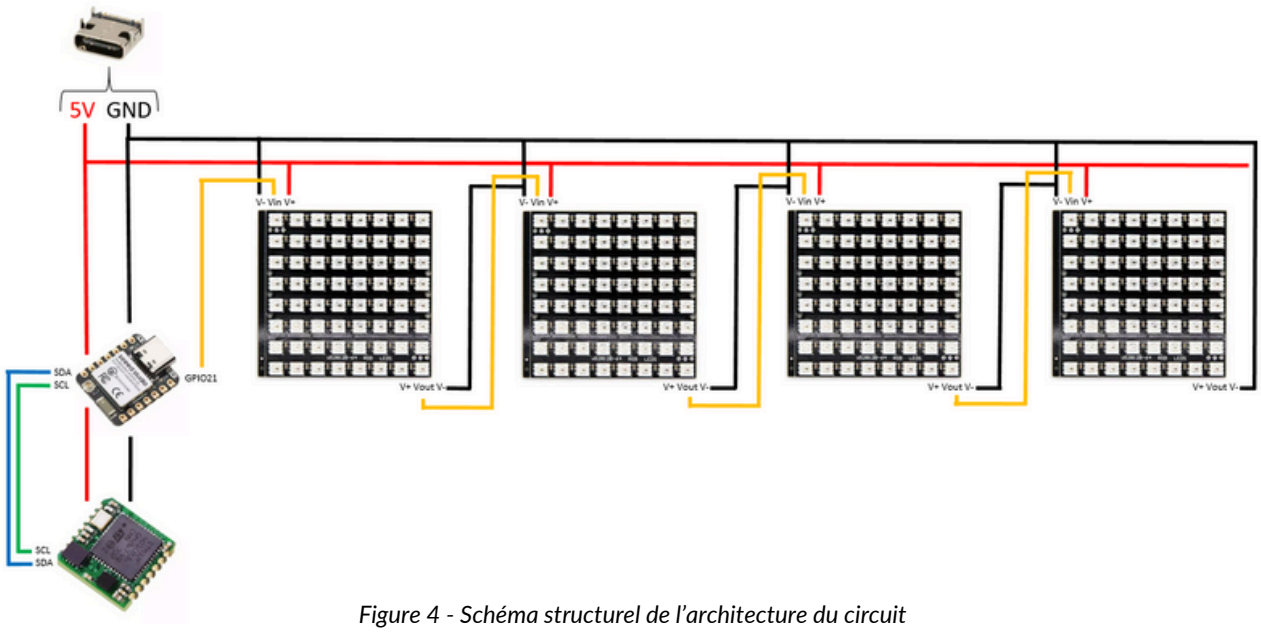


Figure 4 - Schéma structurel de l'architecture du circuit électronique

1.6) Diagramme de GANTT :

Avant de commencer le projet, des livrables ont été définies et un diagramme de Gantt estimant les délais nécessaires pour chaque livrable a été initialisé.

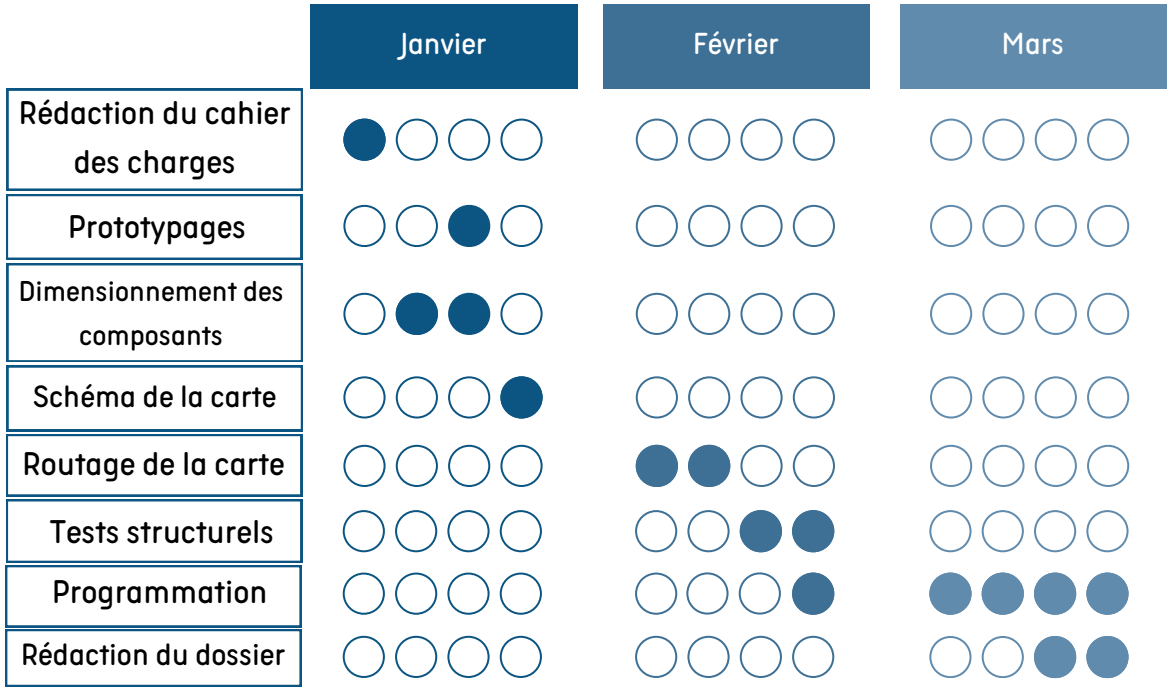


Figure 5 - Diagramme de Gantt prévisionnel

2) Choix technologiques

2.1) WT931

Figure 6 - WT931

Le premier module est le centre du projet IMU FLUID. En effet le composant WT931 est une centrale inertielle qui inclue quatre différents capteurs. Ce sont des capteurs angulaire, gyroscopique, d'accélération et magnétique. Dans ce projet, seul le capteur angulaire est utilisé. Il fonctionne avec les protocoles I2C et UART. Il est alimenté en 5V.

2.2) ESP32 C6

Le second module est l'ESP32 C6, C'est un des plus petit SOC de la gamme ESP32. Sa taille est donc parfaite pour l'intégrer dans le projet IMU FLUID. L'ESP32 reçoit les valeurs angulaires du capteur WT931 à l'aide du protocole I2C et envoie ensuite une trame en binaire vers les matrices. Il est alimenté en 5V.

Figure 7 - ESP32C6

2.3) Matrice LED WS2812b

Figure 8 - Matrice WS2812b

Le troisième module est une matrice de LED. Quatre matrices 8x8 WS2812b sont utilisées en série pour avoir une matrice de 16x16. Ces matrices permettent de visualiser les valeurs retournées par le capteur.

2.4) Connecteur USB-C

Pour l'alimentation de la carte, l'utilisation d'un connecteur USB-C s'imposait comme idée la plus professionnelle. En effet, ce connecteur permet d'utiliser des batteries externes ou encore des câbles de chargeur de téléphone. Il offre donc plus de polyvalence et convient parfaitement pour l'alimentation continue de systèmes électroniques.

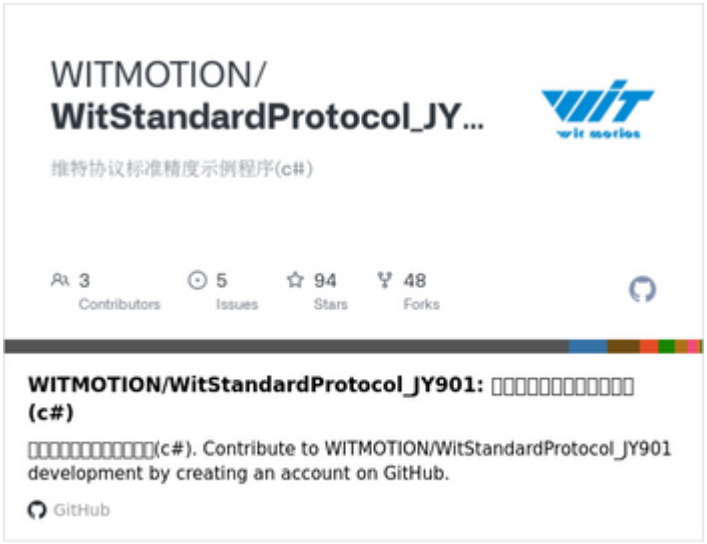
Figure 9 - USB-C

3) Prototypages

3.1) Centrale inertielle

Avant de faire une carte électronique il est important de prototyper chacun des composants importants de la carte.

Premièrement, la centrale inertielle n'étant pas beaucoup documentée sur internet il a tout de même fallu trouver un code permettant de récupérer les valeurs des différents capteurs sur le moniteur série de l'IDE Arduino.



GitHub - WITMOTION/WitStandardProtocol_JY901: 维特协议标准精度示例程序(c#)

Sur ce lien GitHub, le code basique pour récupérer les valeurs de chacun des différents capteurs est disponible sous plusieurs formats (Arduino, IOS, C++, Java, Python, STM32...). Pour le projet IMU Fluid, le code Arduino à été utilisé. Pour l'utilisation il faut supprimer les parties de code non fonctionnelles.
Si cette explication n'est pas claire, n'hésitez pas à utiliser la librairie effectuée lors de ce projet qui vous donnera le même résultat !

Vous trouverez ci-dessous une image du câblage entre la centrale inertielle et l'ESP32C6 ainsi de ce que retourne le moniteur série lors de la lecture des valeurs sur les capteurs.

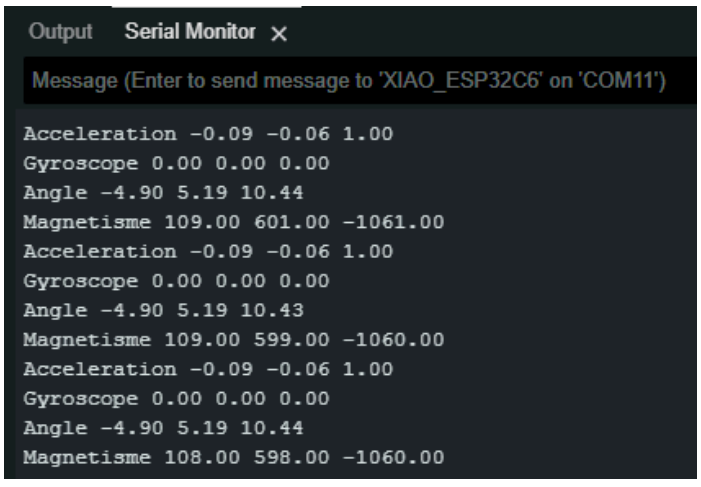


Figure 10 - Lecture des valeurs dans le moniteur série

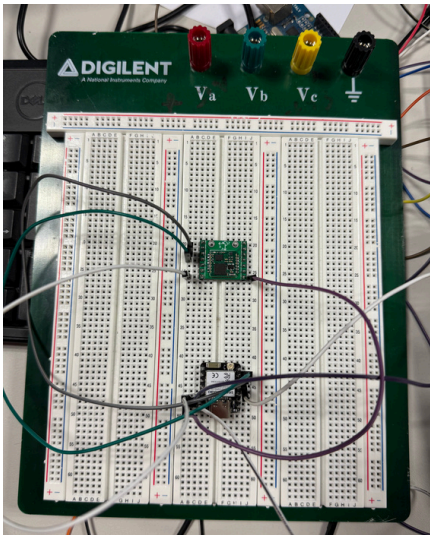


Figure 11 - Prototypage du capteur

3.2) Matrices de LED

Deuxièmement, il est primordial de comprendre comment les matrices fonctionnent. Un simple code à donc été envoyé pour vérifier que les LED des matrices étaient bien à la suite. (voir ci-dessous)

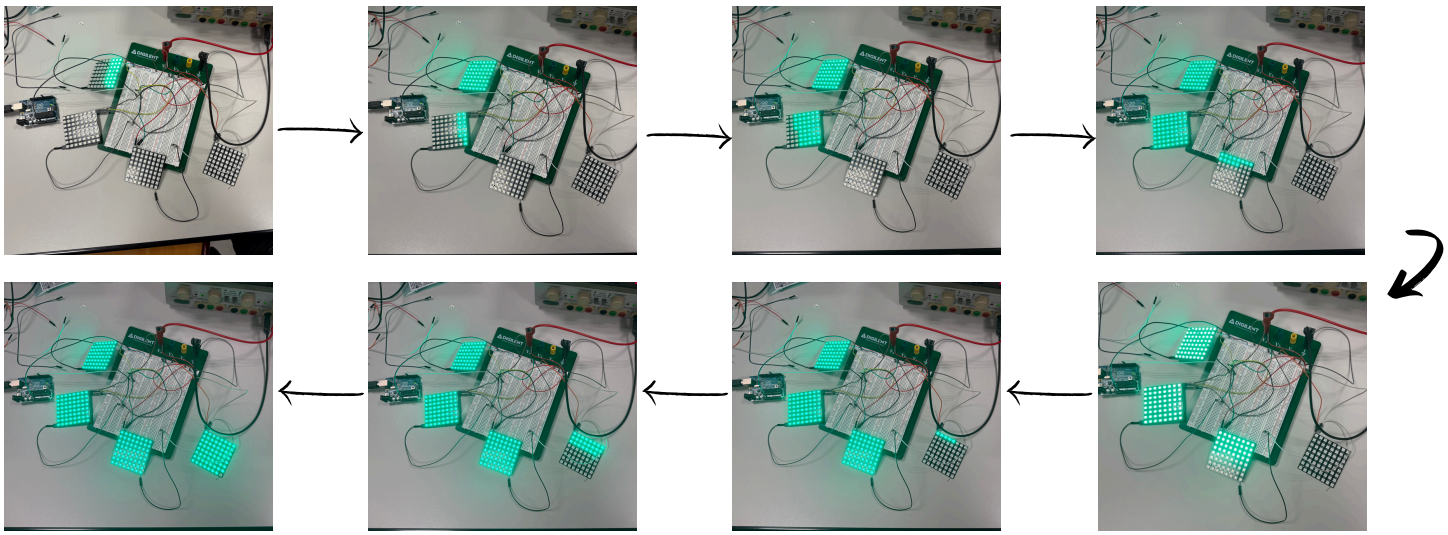


Figure 12 - Lien entre les matrices

La prochaine étape est de lier les données envoyées par la centrale inertielle avec l'affichage des matrices. Pour cela, en fonction des angles reçues les LED s'allument à droite, en haut, à gauche et en bas.

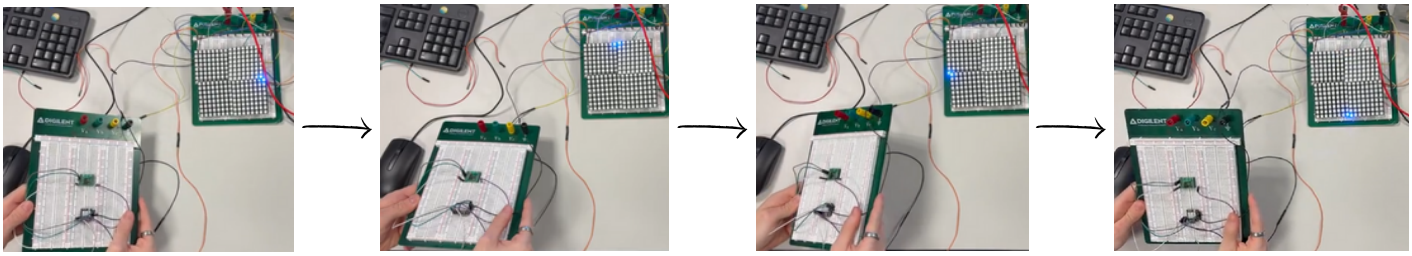


Figure 13 - Lien entre les matrices et la centrale inertielle

Une fois la validation des valeurs angulaires effectuée, il est possible d'intégrer le code final permettant au fluide de se déplacer sur la matrice 16x16. Seul des photos seront représentées dans ce dossier, n'hésitez pas à aller voir la vidéo pour voir le visuel du prototypage ou bien le rendu final.

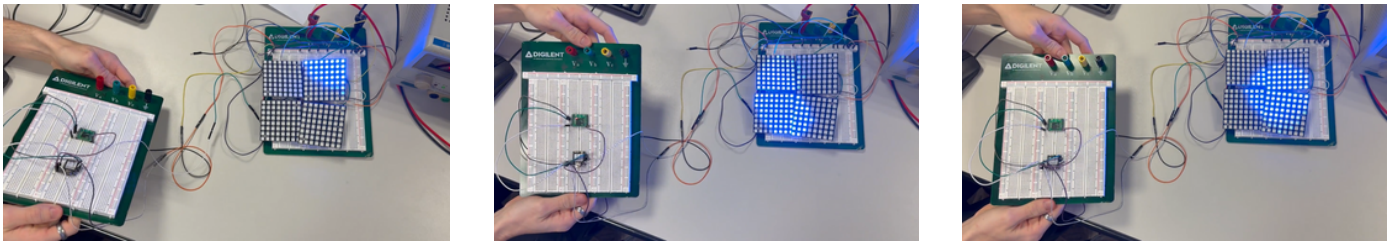


Figure 14 - Prototypage final

4) Réalisations

4.1) Schéma structurel sur Eagle

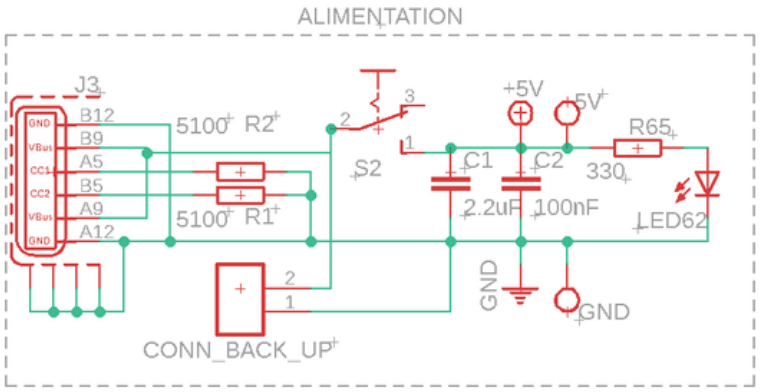


Figure 15 - Schéma alimentation

Partie Alimentation :

L'alimentation de la carte est en 5V. Le connecteur USB-C a deux pins de GND ainsi que deux pins VCC. Les signaux A5 et B5 sont reliés à la masse par une résistance de 5.1kOhm car il n'y a pas de transfert de données via ce câble. Une LED d'alimentation est utilisée pour vérifier le fonctionnement de la carte. Un interrupteur est aussi implémenté pour éteindre ou allumer la carte.

Partie WT931:

La centrale inertielle WT931 est alimentée en 5V. Les signaux SCL et SDA sont les broches permettant la communication I2C avec l'ESP32 C6 en envoyant les valeurs des différents capteurs présents sur la centrale inertielle. Ces broches sont reliées au 5V par des résistances de pull-up pour éviter d'avoir des niveaux logiques incertains. Enfin, les broches UART RX et TX ont aussi été ajoutées pour avoir une solution de secours si l'I2C ne fonctionne pas grâce à un adaptateur.

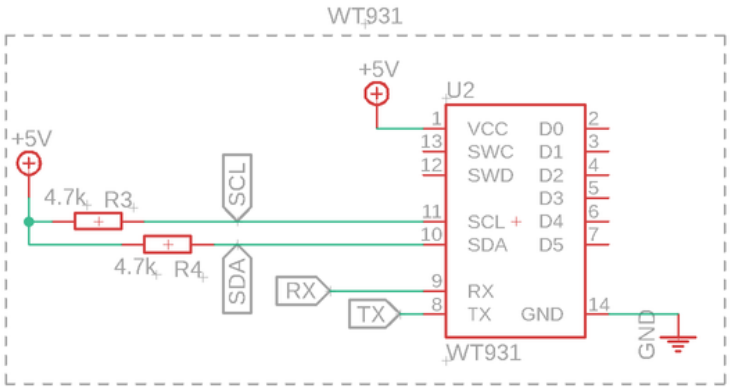


Figure 16 - Schéma WT931

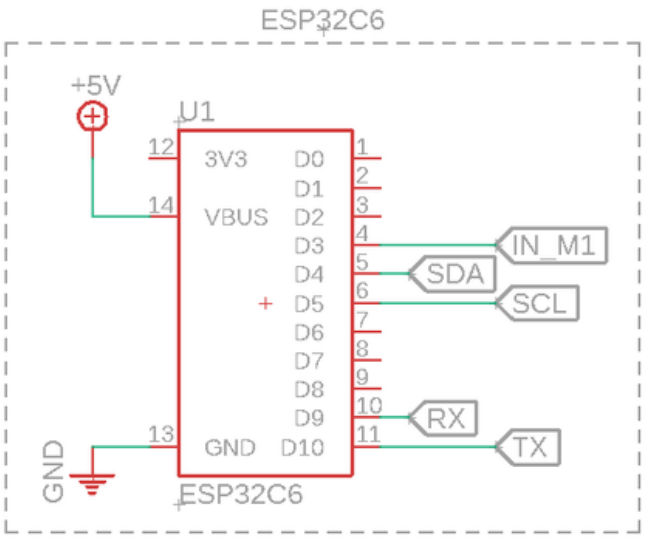


Figure 17 - Schéma ESP32C6

Partie ESP32 C6:

L'ESP32 C6 est alimenté en 5V. Les valeurs reçues de la centrale inertielle sont traitées par les broches SCL et SDA. Les broches RX et TX sont aussi présentes en cas de problème sur le protocole I2C. Après traitement, la broche digitale D3 (GPIO21) s'occupe d'envoyer les trames vers les matrices 8x8 en série.

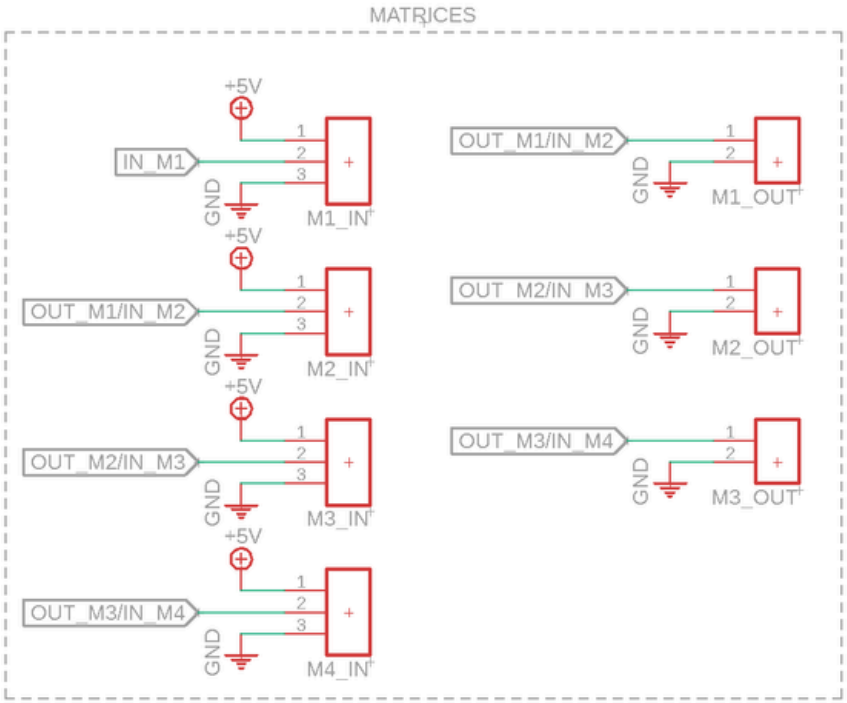


Figure 18 - Schéma matrices

Partie Matrices

Comme quatre matrices de LED sont ajoutées en série sur la carte, il faut les brancher pour faire en sorte que la trame binaire se perpétue sur l'ensemble des LED. Pour cela, il est important de brancher le "Vout" de la première matrice vers le "Vin" de la suivante et répéter ce schéma jusqu'à la quatrième. Pour cela il a été décidé de surélever les matrices au dessus du PCB et de faire passer les signaux sur la carte grâce à des connecteurs. Pour l'alimentation des matrices, seul le signal et la masse sont envoyés de matrice en matrice. Le 5V étant dépendant sur chaque matrice, il est possible de gérer les LED qui crament en série ou alors une alimentation insuffisante en fonction du code implémenté dans l'ESP32 (et de la couleur des LED !)

Figure 19 - Schéma entretoises

Partie Mécanique :

Pour finir, des trous sont percés dans le PCB pour y ajouter ensuite des entretoises mécaniques. Cela permet de fixer les matrices de LED au PCB à l'aide de vis, de rondelles et de boulons pour assurer une structure mécanique complète et solide. Plus de visuels seront disponibles dans la suite du dossier.

4.2) Routage de la carte sur Eagle

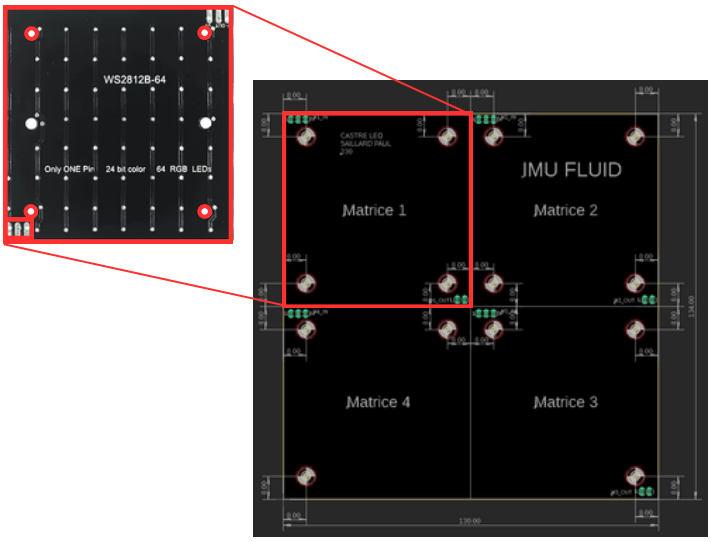


Figure 20 - Routage entretoises

Partie Dimensionnement de la carte :

Afin de dimensionner correctement la carte, il a été décidé de se baser sur la taille des matrices de LED. En effet, une matrice mesure environ 67 par 65 mm et le module est constitué de 4 matrices. La taille totale de la carte est alors logiquement de 134 par 130 mm. Des trous ainsi que des connecteurs ont également été placé de façon à pouvoir fixer et connecter simplement les matrices à la carte.

Partie Alimentation :

L'alimentation de la carte se fait en 5V via le connecteur USB-C. Une LED d'alimentation est utilisée afin de vérifier le fonctionnement de la carte et un connecteur et un connecteur deux pins afin de pouvoir facilement alimenter la carte sans passer par le connecteur USB. Un interrupteur est également présent afin d'éteindre ou d'allumer la carte. Cette partie alimentation a été placée en bas de la carte par soucis d'esthétique mais cela empêche de placer des entretoises de fixation sur les matrices 3 et 4.

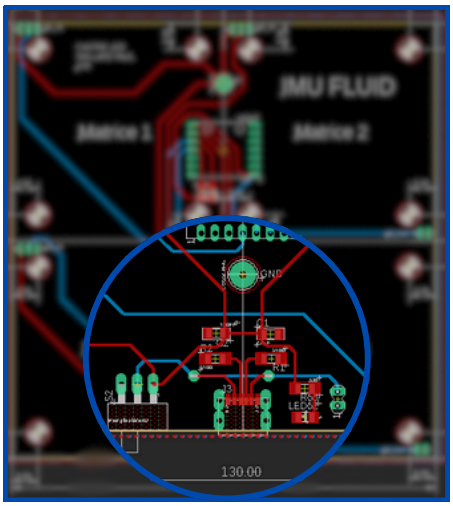


Figure 21 - Routage alimentation

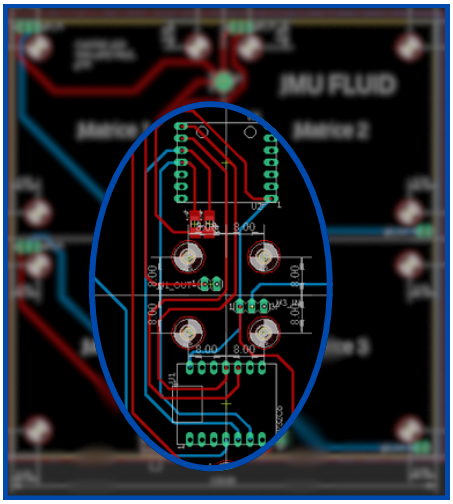


Figure 22 - Routage modules

Partie ESP32 C6 et WT931 :

L'ESP32 C6 et le capteur WT931 ont été placés aux centre de la carte entre les matrices afin de ne pas encombrer les fixations des matrices mais surtout pour que le capteur se trouve au maximum au centre de la carte afin de ne pas fausser les mesures d'angles sur l'axe X comme Y.

4.3) Assemblage de la carte

La face TOP contient l'intégralité des composants et des modules de la carte. En effet, le but est de placer ces derniers en dessous des matrices pour avoir un projet propre et ainsi augmenter l'immersion lors de l'utilisation du module complet IMU FLUID.

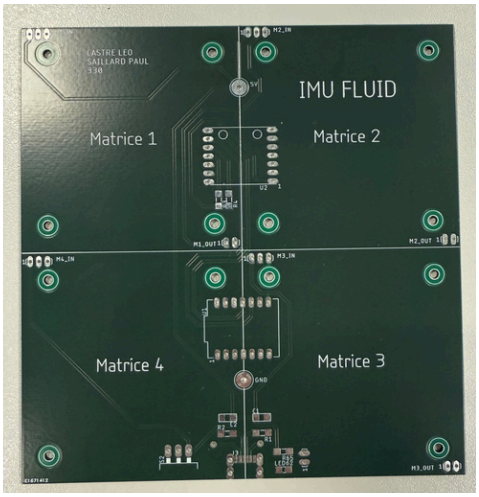


Figure 23 - Carte côté TOP

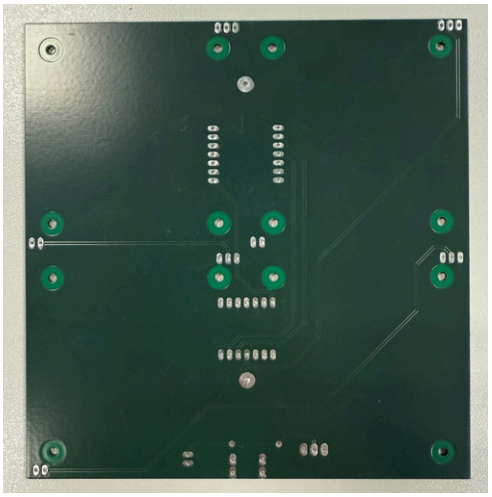


Figure 24 - Carte côté BOTTOM

Ensuite une batterie de test très simple est mise en place pour vérifier les alimentations et les signaux de la carte avant l'assemblage des composants.

Tests			Bon	Mauvais
Test à l'ohmmètre	Continuité des vias	Vias généraux	✔	
		Vias plan de masse	✔	
	Continuité des alimentations	5V	✔	
		GND	✔	
	Continuité des pistes		✔	
Test visuel	Qualité des brasures		✔	

Figure 25 - Tests carte avant assemblage

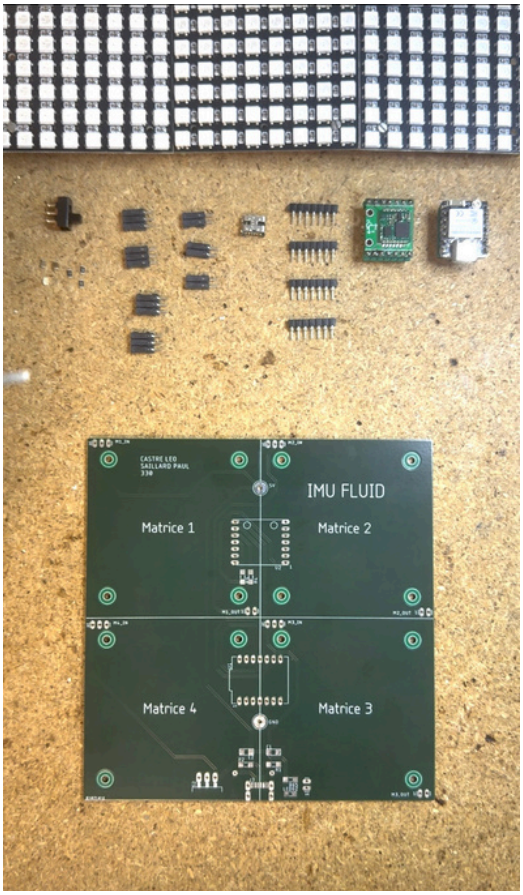


Figure 26 - Plan timelapse

Soudure des composants :

Dans l'optique d'avoir une vidéo finale, les résistances et condensateurs CMS ont été soudés au fer à souder. Le but étant de faire un timelapse complet lors de l'ajout de tous les composants sur la carte électronique. Si le timelapse vous intéresse n'hésitez pas à aller voir la vidéo du projet IMU FLUID ! Ci-dessous la carte finale soudée :

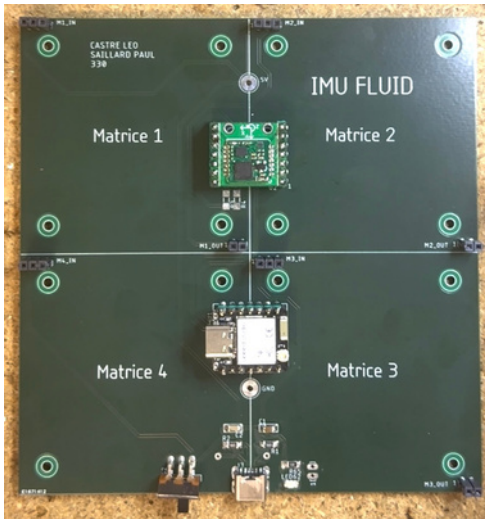


Figure 27 - Carte assemblée côté TOP

Un seconde batterie de test est effectuée pour vérifier la soudure des composants. Aucun problème n'a été relevé sur la carte, aucune retouche n'a été effectuée.

Test			Bon	Mauvais
Visuel			✔	
Test à l'ohmmètre	Continuité des vias	Vias généraux	✔	
		Vias plan de masse	✔	
	Continuité des alimentations	Vcc des composants	✔	
		GND des composants	✔	
	Continuité des pistes		✔	

Figure 28 - Tests après assemblage de la carte

Tests des alimentations des composants soudés sur la carte :

Composants		Tests	
Nom du composant	**Nom de la pin**	**V attendue**	**V mesurée**
WT931	Vcc	5V	4.86V
	GND	0V	0.02V
ESP32 C6	Vbus	5V	4.93V
	GND	0V	0.07V
USB-C	Vbus	5V	4.98V
	GND	0V	0.04V
Matrice WS2812b	V+	5V	4.90V
	V-	0V	0.12V

Figure 29 - Tests alimentations modules

Enfin, les matrices ont été ajoutées sur la carte et fixées à l'aide de vis, de rondelles et de boulons. La structure finale est très solide grâce au bloc composé de matrices et de la carte. Ci-dessous des photos de l'ensemble :

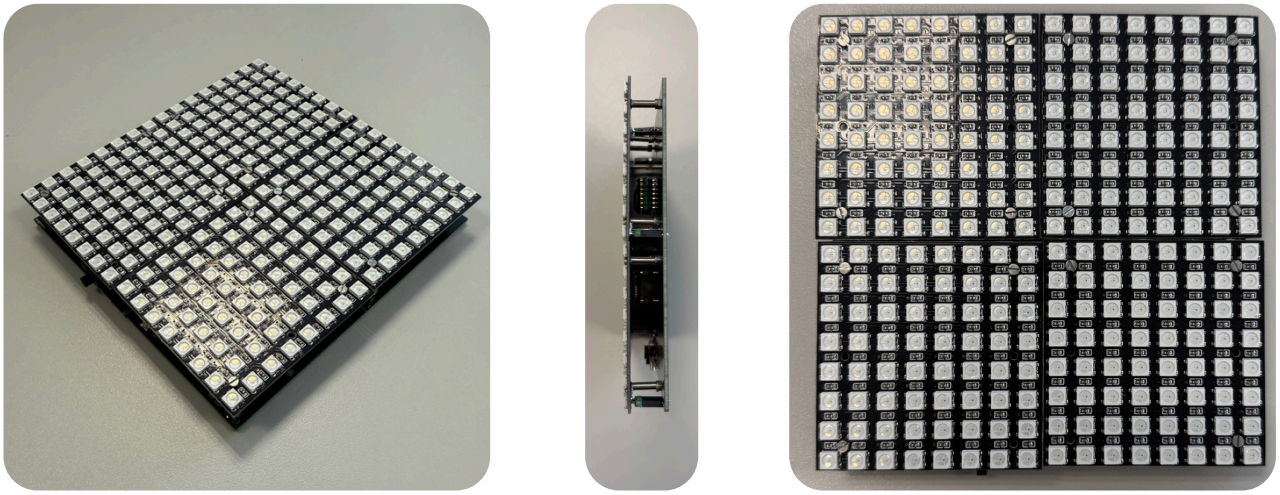


Figure 30 - Carte finale

4.4) Programmation du SoC

Organigramme du code principal :

Pour commencer la programmation du projet, un organigramme a été réalisé afin de facilité l'écriture et la découpe du programme par étapes. Ensuite, il a été traduit en code informatique faisant appel à différentes librairies.

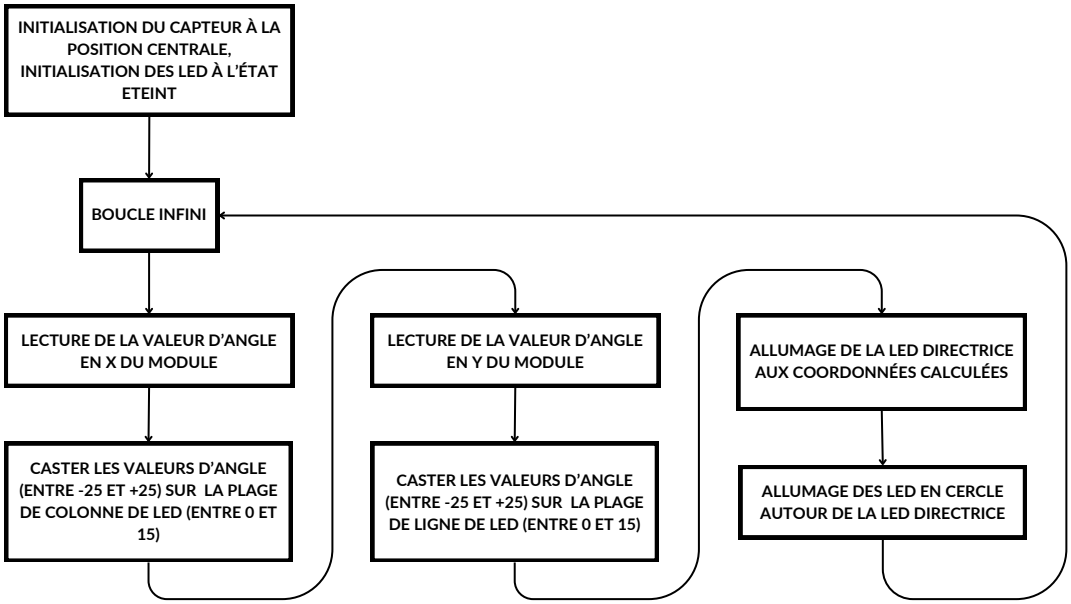


Figure 31 - Organigramme du fonctionnement général du code en C

Les librairies :

Afin d'optimiser et de professionnaliser le code principal, plusieurs librairies ont été utilisées ou créées. Les librairies REG.h et wit_c_sdk.h étaient fournies par le constructeur du module WT931 et servent à communiquer simplement en I2C avec celui-ci ainsi qu'à configurer ses registres internes. Celle appelée LibrairieCapteurs.h a été créée afin de regrouper les fonctions de lecture, de traitement et de stockage des valeurs retournées par les différents capteurs du module. La librairie Adafruit_Neopixel.h est une librairie fournie par adafruit et qui permet de communiquer facilement avec plusieurs types de matrices/LED RGB, ce qui est parfait pour commander les WS2812b. Enfin, la librairie Matrices.h a été créée afin d'y réunir l'ensemble des fonctions d'affichage sur les matrices, d'adaptation de la taille des matrices ainsi que de calcul des coordonnés en fonction des angles.

```
#include <REG.h> // Librairie de paramétrage des regisres du module
#include "LibrairieCapteurs.h" // Librairie de récupération des valeurs d'axes des capteurs
#include <wit_c_sdk.h> // Librairie de configuration du protocole I2C
#include <Adafruit_NeoPixel.h> // Librairie de communication avec les matrices
#include "Matrices.h" // Librairie d'ordonnancement des matrices
```

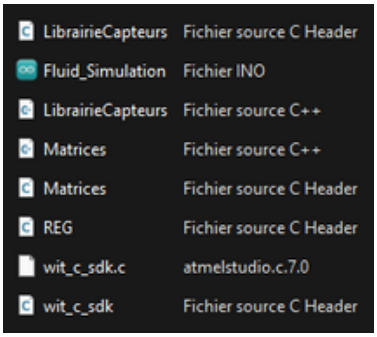


Figure 32 - Librairies utilisées pour la simulation et l'affichage du fluide

Le code principal :

L'utilisation de librairies a permis de réduire considérablement la taille du code principal atteignant seulement une dizaine de ligne. Il réunit la définition des variables de paramétrage, l'initialisation du module et des matrices dans le setup ainsi que la commande du mouvement du fluide dans le loop (la trainée, la position d'angle et de LED ainsi que la commande d'affichage de celles-ci).

```
#define PIN 21 // Pin de communication de l'ESP32 avec les matrices
#define NUMPIXELS 256 // Nombre total de LEDs
#define GRID_SIZE 16  // Taille de la grille de LEDs
#define CIRCLE_RADIUS 4 // Taille du cercle du fluide
#define SENSI_ANGLE 20 // Maximum de l'angle pris en compte pour le mappage des valeurs d'angle en position

void setup() {
    wt931_init(); // Initialiser les capteurs du module WT931
    Matrices_init(); // Initialiser les matrices
}

void loop() {
  trainee(); // Gérer la trainée (dégradé de bleu) derrière le fluide lors du mouvement

  int posx = conf_posx(read_angle('x'), SENSI_ANGLE); // Stocker l'angle lu en fonction de la sensibilité choisi
  int posy = conf_posy(read_angle('y'), SENSI_ANGLE); // Stocker l'angle lu en fonction de la sensibilité choisi

  cmd_LEDs(posx, posy, SENSI_ANGLE, GRID_SIZE, CIRCLE_RADIUS); // Allumer la LED correspondante à la position obtenue (ligne/colonne) après mappage

  delay(10); // Attendre 10 ms
}
```

Figure 33 - Code principal du projet

La fonction de gestion de la trainée :

Afin de rendre le mouvement du fluide plus réaliste, un dégradé de couleur ou une trainée a été ajoutée. En effet, cela rend le mouvement du fluide plus dynamique et simule parfaitement la diminution de la quantité d'eau derrière une flaque qui se déplacerait. Cela se fait par la diminution de l'intensité des LED auparavant allumées, c'est à dire que lorsqu'une LED allumée se déplace vers la droite, elle laisse derrière elle, une LED allumée à une plus faible intensité.

```
// Fonction de gestion de la trainée
void trainee(void){
  // Effacer les anciennes LEDs (traînée)
  for (int i = 0; i < NUMPIXELS; i++) {
    // Diminuer l'intensité de la couleur de la traînée
    uint32_t color = lastColors[i];
    uint8_t r = (color >> 16) & 0xFF;
    uint8_t g = (color >> 8) & 0xFF;
    uint8_t b = color & 0xFF;

    // Appliquer un effet de dégradé (réduire l'intensité)
    r = max(r - 10, 0);  // Réduire l'intensité rouge
    g = max(g - 10, 0);  // Réduire l'intensité verte
    b = max(b - 10, 0);  // Réduire l'intensité bleue

    // Re-assigner la couleur diminuée
    lastColors[i] = pixels.Color(r, g, b);
    pixels.setPixelColor(i, lastColors[i]);
  }
}
```

Figure 34 - Fonction de gestion de la trainée

```
// Fonction de lecture des valeurs angulaires
float read_angle(unsigned char axe) {
    float fAngle[3]; // Tableau de stockage des valeurs d'angle
    WitReadReg(AX, 12); // Lecture des registres des capteurs
    delay(10); // Attendre 10 ms

    // Conversion des valeurs brutes en valeurs physiques
    for (int i = 0; i < 3; i++) {
        fAngle[i] = sReg[Roll + i] / 32768.0f * 180.0f; // Conversion en degrés
    }

    // Retourne la valeur demandée selon l'axe spécifié
    switch (axe) {
      case 'x': return fAngle[0];
      case 'y': return fAngle[1];
      case 'z': return fAngle[2];
      default: return 'N';
    }
}
```

Figure 35 - Fonction de lecture et de stockage des valeurs lues sur le capteur d'angle

La fonction de lecture des angles :

La fonction ci-contre permet de lire directement les registres du capteur de position angulaire afin d'y récupérer les valeurs sur les axes X, Y et Z, les convertir en degré puis les stocker dans le tableau fAngle[]. A noter que cette fonction existe en trois autres exemplaires dans la librairie pour le capteur magnétique, le capteur gyroscopique et l'accéléromètre.

Les fonctions de gestion des positions en X et en Y (colonne/ligne) :

Les deux fonctions ci-dessous sont similaires, elles permettent de sélectionner une plage de valeurs à utiliser lors du mouvement du module. En effet, si une sensibilité de 20 est choisie alors l'affichage sur les extrémités correspondra aux valeurs -20 et 20 degrés. Les valeurs d'angles du capteur allant de -180 à 180 degrés, toutes les valeurs situées entre -180 et -20 et entre 20 et 180 seront alors bloquées et ramenées respectivement à -20 et 20. Toutes les valeurs situées entre -20 et 20 seront alors par la suite utilisées pour commander les LED (entre 0 et 15) que ce soit en ligne (Y) ou en colonne (X).

```
// Fonction de calibration de la colonne (x) de la LED à allumer en fonction de l'angle en x
int conf_posx(int x, char sensi){
  int pos_x = 0 - x; // Stocker l'angle en X
  if (pos_x < -sensi) { // Si X est inférieur à la sensibilité choisie
    pos_x = -sensi; // X est égal à la sensibilité
  }
  else if (pos_x > sensi) { // Si X est supérieur à la sensibilité choisie
    pos_x = sensi; // X est égal à la sensibilité
  }
  return pos_x; // Retourner la position en X
}

// Fonction de calibration de la ligne (y) de la LED à allumer en fonction de l'angle en y
int conf_posy(int y, char sensi){
  int pos_y = 0 - y; // Stocker l'angle en Y
  if (pos_y < -sensi) { // Si Y est inférieur à la sensibilité choisie
    pos_y = -sensi; // Y est égal à la sensibilité
  }
  else if (pos_y > sensi) { // Si Y est supérieur à la sensibilité choisie
    pos_y = sensi; // Y est égal à la sensibilité
  }
  return pos_y; // Retourner la position en Y
}
```

Figure 36 - Fonctions de gestion des positions en X et en Y (colonne/ligne)

La fonction de commande des LED :

Une fois les valeurs entre -20 et 20 degrés récupérées (pos_x et pos-y), elles sont mappées dans la fonction de commande des LED ci-dessous entre 0 et 15 afin de correspondre aux 16 LED de longueur des matrices. Une fois la LED directrice correctement placée, un cercle est affiché autour de celle-ci. Enfin, toutes les LED y compris celles de la trainée sont affichées dans la couleur choisie.

```
// Fonction de commande des LEDs
void cmd_LEDs(int pos_x, int pos_y, char sensi, unsigned char GRID_SIZE, unsigned char CIRCLE_RADIUS){
  unsigned char led_column = map(pos_x, -sensi, sensi, 0, GRID_SIZE-1); // Position réelle de la colonne de la LED entre 0 et 15
  unsigned char led_line = map(pos_y, -sensi, sensi, 0, GRID_SIZE-1); // Position réelle de la ligne de la LED entre 0 et 15

  // Allumer un cercle autour de la LED directrice
  for (int r = -CIRCLE_RADIUS; r <= CIRCLE_RADIUS; r++) {
    for (int c = -CIRCLE_RADIUS; c <= CIRCLE_RADIUS; c++) {
      // Calculer la distance à la position centrale du cercle
      if (r * r + c * c <= CIRCLE_RADIUS * CIRCLE_RADIUS) {
        int x = led_line + r;
        int y = led_column + c;

        // Assurer que les coordonnées sont dans les limites de la matrice 16x16
        if (x >= 0 && x < GRID_SIZE && y >= 0 && y < GRID_SIZE) {
          lastColors[getLEDIndexFrom16x16(x, y)] = pixels.Color(0, 75, 100);  // Bleu ciel
          pixels.setPixelColor(getLEDIndexFrom16x16(x, y), pixels.Color(0, 0, 150)); // Bleu
        }
      }
    }
  }
  pixels.show(); // Afficher les LEDs avec la traînée
}
```

Figure 37 - Fonction de commande des LED en fonction des coordonnées calculées

Conclusion

Ce projet IMU FLUID qui clôture nos trois années de BUT GEII nous a permis d'explorer une approche innovante réunissant électronique et interaction visuelle captivante.

Les choix effectués, notamment l'intégration du capteur WT931 et les matrices de LED WS2812b ont permis d'acquérir de nombreuses compétences matérielles et logicielles. En effet, plusieurs étapes se sont révélées difficiles à mettre en place de façon professionnel comme la gestion des animations en temps réel ou la disposition des composants sur le PCB pour assurer un assemblage compact et fonctionnel.

À travers la conception de ce module, nous avons su relever ces nombreux défis matériels et logiciels. En effet, l'utilisation du capteur WT931 et de l'ESP32 C6 a nécessité une compréhension approfondie de la création de librairies en C++, la gestion de valeurs bruts et les contraintes techniques associées.

Certaines difficultés techniques ont été rencontrées puis corrigées au fil du développement. Par exemple, des ajustements ont été nécessaires sur la configuration de l'alimentation des modules afin de simplifier et gagner de la place sur le PCB. La gestion des signaux entre l'ESP32 C6 et les LED a également été optimisée afin de garantir une synchronisation fluide des animations.

Outre l'aspect technique très complet, IMU FLUID possède un potentiel d'application dans des domaines variés, notamment pour l'aide aux personnes atteintes de TDAH, ou encore dans des usages ludiques comme le jeu SNAKE.

Perspectives

IMU FLUID est un module complètement programmable. Si vous trouvez que l'animation de fluide n'est pas assez captivante pour une personne atteinte de Trouble du Déficit de l'Attention avec ou sans Hyperactivité (TDAH), alors plusieurs utilisations existent.

Figure 38 - Jeu de SNAKE

Il est possible de penser au jeu "SNAKE" pour continuer à capter l'attention d'une personne. Effectivement, les valeurs retournées par le capteur (grâce aux axes X et Y) permettent de contrôler la direction que prend le serpent. De plus, les LED étant RGB, le serpent peut être de couleur verte et la pomme recherchée par le serpent de couleur rouge. Une première version d'un code "SNAKE" complètement autonome est disponible dans la vidéo finale.

Comme deuxième point d'amélioration, un boitier peut être un moyen pour rendre le projet plus professionnel. En effet, toute la carte a été conçue en vue d'un boitier. Cependant, par manque de temps aucune ébauche de boitier n'a été dessiné.

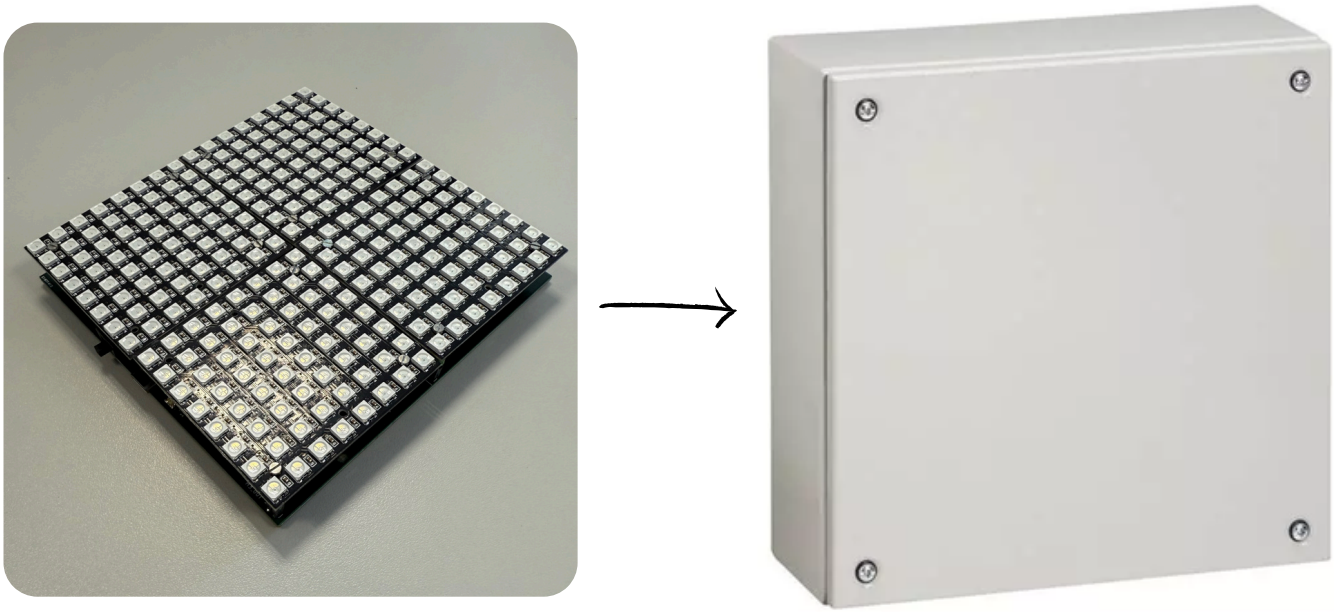


Figure 39 - Exemple de boitier

Bibliographie

Manuel d'utilisation du module WT931 : https://manuals.plus/fr/witmotion/wt931-inclinometer-sensor-manual

Datasheet du module WT931: https://images-na.ssl-images-amazon.com/images/I/B1jDR7-M9VS.pdf

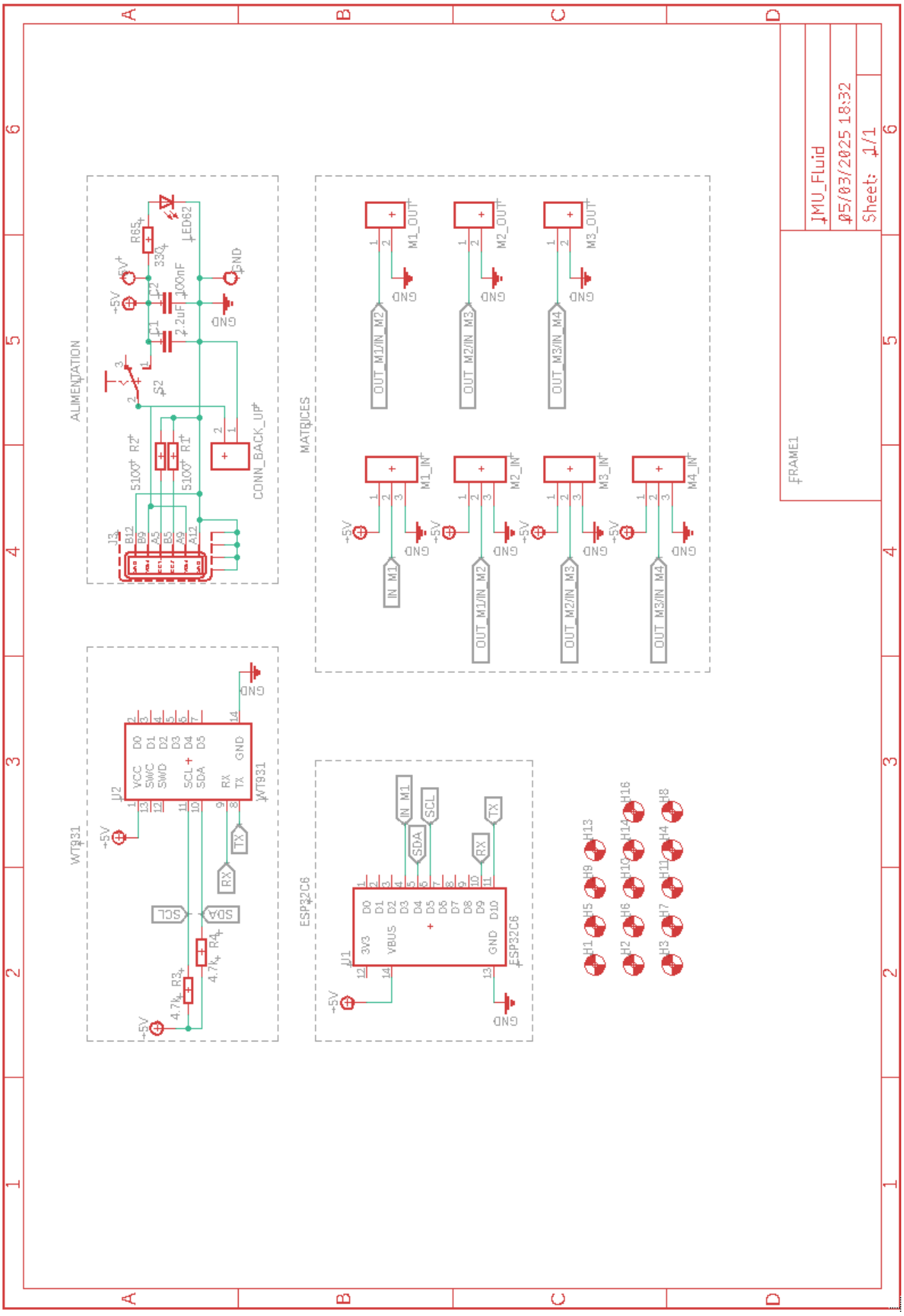
Tutoriels d'utilisation du module WT931 : https://www.youtube.com/playlist?list=PL43tdDrVL_VDDciadEelCtvTh0dZjgHtS

Datasheet de l'esp32 C6 : https://docs.espressif.com/projects/esp-idf/en/stable/esp32c6/get-started/index.html

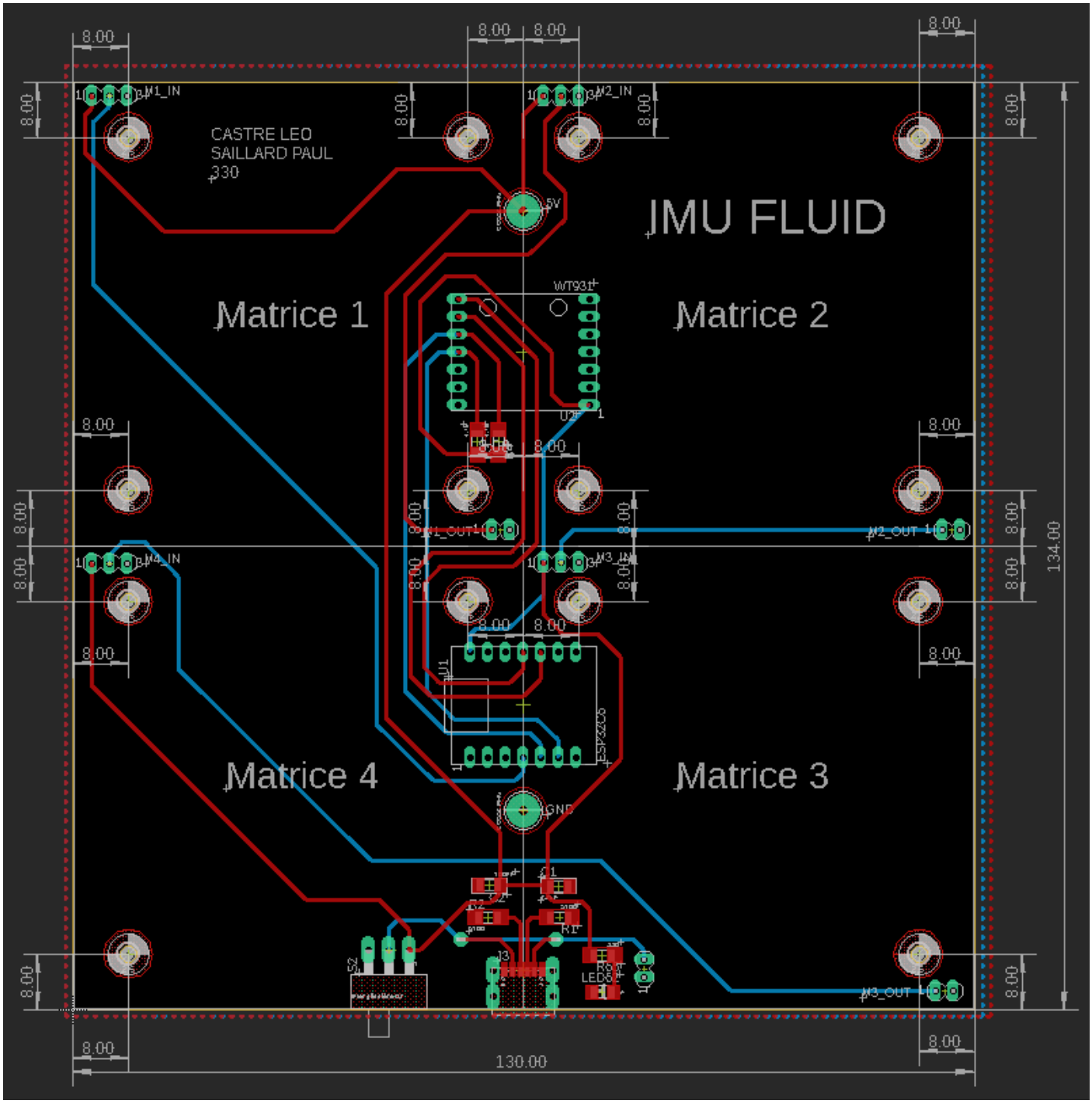
Table des Figures

Annexes

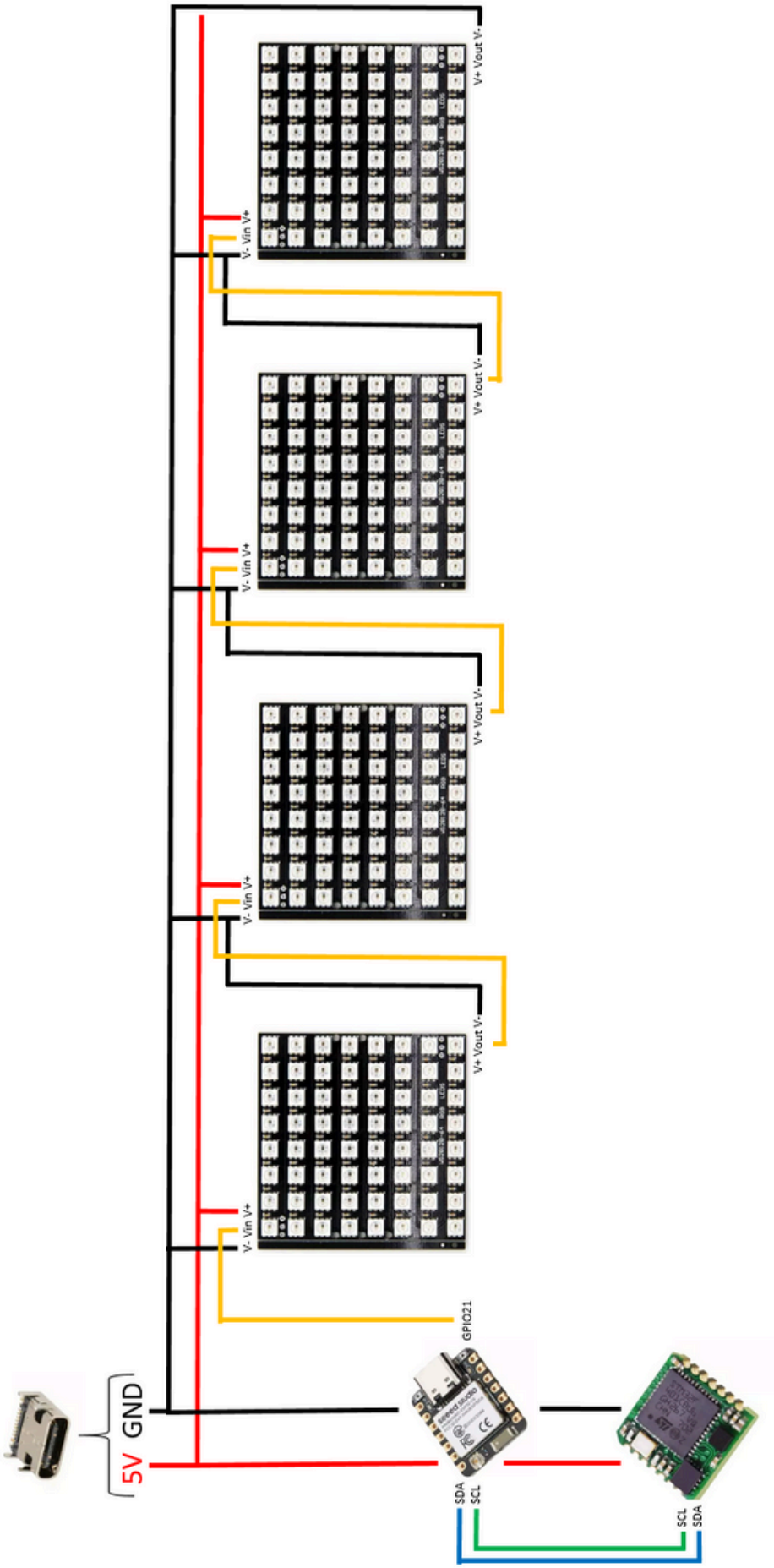
Annexe 1: Schéma structurel



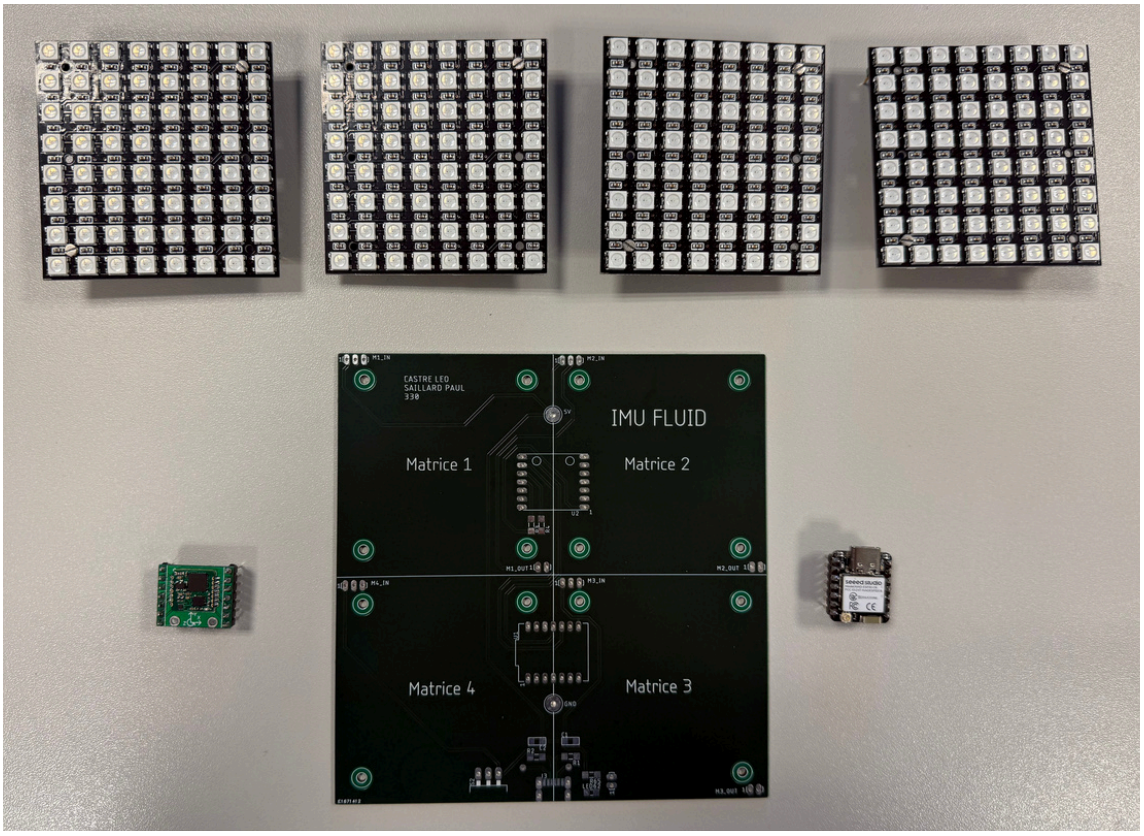
Annexe 2 : Routage



Annexe 3 : Schéma structurel



Annexe 4 : Composants utilisés pour le projet



Annexe 5 : Organigramme du code

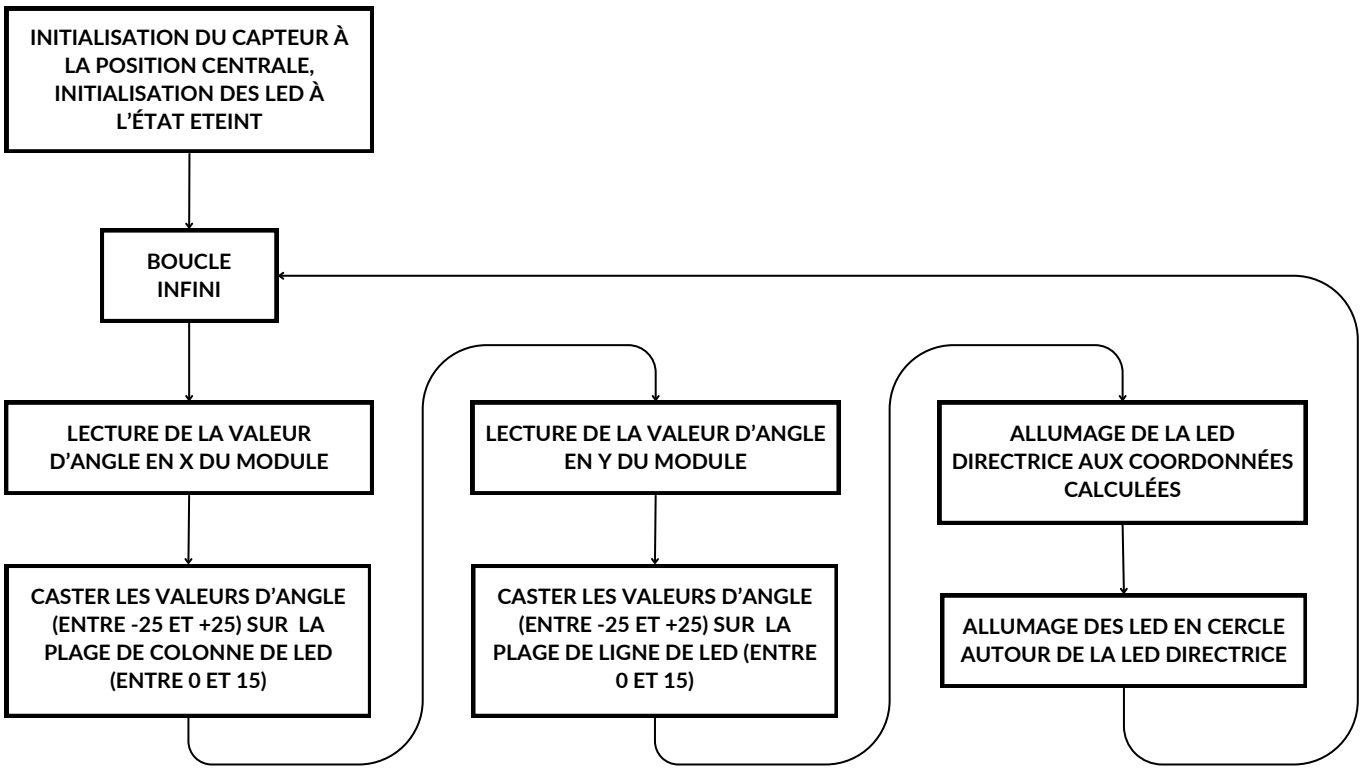
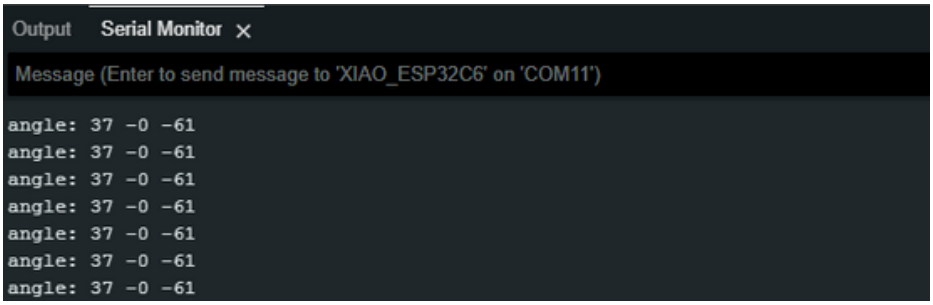


Figure 31 - Organigramme du fonctionnement général du code en C

Annexe 6 : Valeurs retournées par le capteur angulaire en fonction de la position du module IMU FLUID. Ici, seules les deux premières valeurs sont importantes (axes X et Y) :

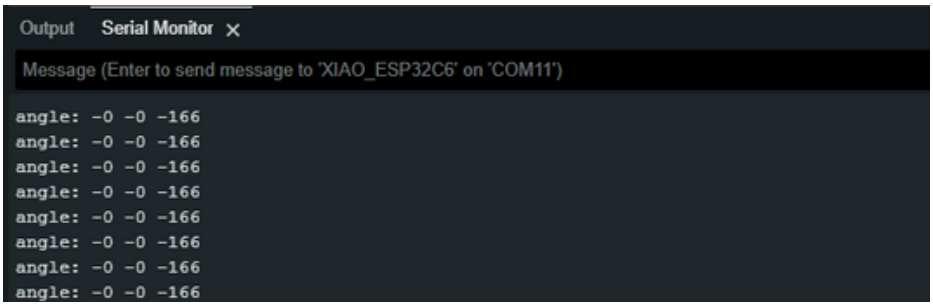
Capteur penché vers le bas



Capteur penché vers le haut

Capteur penché vers la gauche

Capteur penché vers la droite



Capteur posé à plat sur une surface plane

Résumé

En résumé, l'idée principale du projet est de comprendre et d'utiliser la centrale inertielle WT931 de chez WitMotion avec l'utilisation du protocole I2C. De plus, il a été décidé d'y ajouter des matrices de LED pour aider les personnes atteintes de Trouble du Déficit de l'Attention avec ou sans Hyperactivité (TDAH) dans le but d'augmenter leur concentration. Plusieurs prototypages ont été effectués pour tester et choisir définitivement les composants à implémenter dans le projet.

Pour la partie hardware, un schéma électronique sur Eagle a été créé, suivi du routage de la carte sur ce même logiciel ainsi qu'une batterie de tests à effectuer pour s'assurer du bon fonctionnement de la conception du projet. Ensuite, la partie software contient un code s'assurant de tester et de pouvoir utiliser les différentes parties de la carte (centrale inertielle, matrices, ESP32C6) et a été développé sur l'IDE Arduino en C/C++.

Publication

This project concludes the last year of the Electrical Engineering and Industrial Computing degree at the IUT of Angers. IMU FLUID is a four weeks project combining English and Electronic. The main goal of the project was to understand and use an Inertial Measurement Unit (IMU) containing four different sensors.

IMU FLUID needed context, that is why we decided to add a captivating experience with LED matrices for the people with attention deficit disorders to improve their focus and cognitive engagement. Then, if you want to see the result and the steps of the project, launch the video !