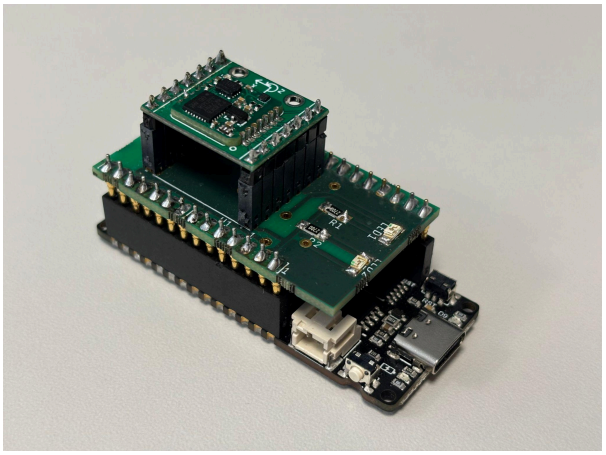
Rapport de SAÉ – GEII3-FA ESE S6 G330

SAÉ *“IMU WIRED”*



Encadrant : Monsieur **Philippe LUCIDARME**

Tonin OLAGNON - Killian GALOPIN

4 boulevard de Lavoisier, 49000 Angers

Année 2024-2025

Glossaire

CMS : Composant Monté en Surface
ESE : Électronique et Systèmes Embarqués
FA : Formation en Alternance
GEII : Génie Électrique et Informatique Industrielle
IUT : Institut Universitaire de Technologie
SAÉ : Situation d'Apprentissage et d'Évaluation
SoC : Système sur Puce
ESP32-WROOM-32 : Microcontrôleur d'Espressif Systems intégrant des capacités Wi-Fi et Bluetooth, utilisé dans les systèmes embarqués et l'Internet des objets.
IMU : Unité de Mesure Inertielle
Accéléromètre triaxial : Accéléromètre à trois axes mesurant l'accélération linéaire sur trois axes.
Gyroscope triaxial : Gyroscope à trois axes mesurant la vitesse angulaire et les rotations sur trois axes.
Magnétomètre : Capteur détectant les champs magnétiques, permettant de déterminer l'orientation absolue.
UART : Transcepteur Asynchrone Universel, protocole de communication
I²C : Circuit Inter-Intégré, protocole de communication
SPI : Interface Périphérique Série, protocole de communication
USB-C : Type de connecteur et standard de câble USB
Tkinter : Bibliothèque Python utilisée pour la création d'interfaces graphiques utilisateur
IHM : Interface Homme-Machine
Wi-Fi : Wireless Fidelity
Bluetooth : Technologie de communication sans fil à courte portée
PCB : Circuit Imprimé
IP65/IP67 : Niveaux de protection contre la poussière et l'eau

Décharge de responsabilités

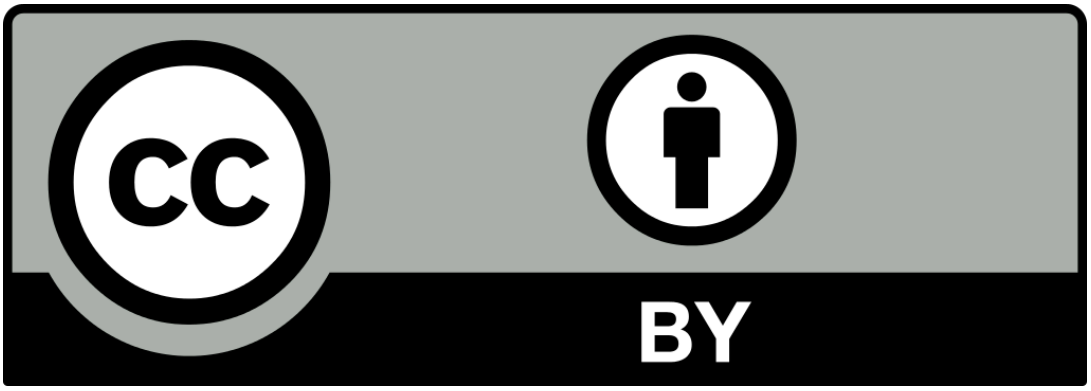
Ce rapport présente le travail réalisé par un groupe d'étudiants dans le cadre d'un projet pédagogique. Les auteurs et l'Université d'Angers ne garantissent pas que l'information, les documents, la méthodologie et le matériel présentés dans ce document soient complets, conformes à l'état de l'art et 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

OLAGNON TONIN et *GALOPIN KILLIAN auteurs du présent rapport, publions et divulguons celui-ci sous la Licence Creative Commons suivante « CC BY » pour le monde entier et pendant la durée légale de protection des droits d'auteur. Cette licence autorise la représentation, la reproduction, la modification, la création d'œuvres dérivées et l'utilisation y compris à des fins commerciales sous réserve de mentionner les noms et prénoms des auteurs.*

Tonin OLAGNON

Galopin Killian



Remerciements

Tout d'abord, nous tenons à remercier tout particulièrement et à témoigner toute notre reconnaissance aux personnes suivantes, pour leur soutien dans la concrétisation de ce projet :

- Monsieur Philippe LUCIDARME, responsable projet, pour ses conseils éclairés, sa patience, sa disponibilité et sa confiance qu'il nous a accordée lors de ce projet ; ainsi que pour la mise à disposition de tous les moyens disponibles afin de mener à bien ce projet.
- Madame Géraldine DENECHEAU, aidante à l'enseignement GEII, pour son aide et son temps lors de la réalisation du projet.
- Le département *GEII* de l'*IUT* d'Angers-Cholet pour sa coopération professionnelle tout au long de cette expérience et pour s'être mis à notre disposition lors de la réalisation de ce projet.

Sommaire

I. Introduction

Les systèmes embarqués et les capteurs transforment de nombreux secteurs, et leur utilisation est devenue un élément clé des technologies modernes. Les centrales inertielles sont utilisées dans l'industrie cinématographique pour capturer les mouvements en trois dimensions et donner vie à des personnages animés, créant un réalisme toujours plus impressionnant.

Au-delà du cinéma, ces capteurs jouent un rôle fondamental dans des domaines tels que l'aéronautique, où ils assurent la stabilisation et la navigation des avions et des satellites, ou encore l'automobile, notamment dans les systèmes d'aide à la conduite et de correction de trajectoire. Même les smartphones en sont équipés, permettant des fonctionnalités comme la détection de l'orientation de l'écran ou le suivi des mouvements dans la réalité augmentée.

C'est dans ce contexte que notre projet s'inscrit. L'objectif est de concevoir un système capable de capturer et d'afficher des données angulaires à l'aide d'un capteur inertiel et d'un microprocesseur dédié. En récupérant ces informations en temps réel, nous souhaitons démontrer comment ces technologies peuvent être intégrées dans des applications concrètes.

Ce projet illustre l'importance des capteurs inertiels dans le traitement du mouvement et la transmission des données, ouvrant la voie à de nombreuses améliorations et applications potentielles. À travers ce rapport, nous reviendrons sur les choix techniques effectués, la conception matérielle et logicielle, ainsi que les défis rencontrés tout au long du développement.

II. Cahier des charges

Objectif du Projet

Le projet consiste à développer un système embarqué capable de collecter, traiter et afficher des données angulaires à partir d'une centrale inertielle (IMU), en utilisant un microprocesseur pour le traitement et une interface graphique Python pour l'affichage des données.

Livrables du projet

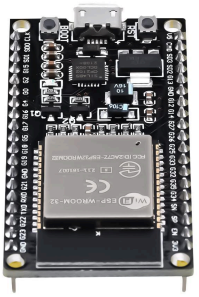
- Phase 1 : Sélection des composants : Choix du capteur, microprocesseur et autres composants nécessaires.
- Phase 2 : Conception et fabrication de la carte électronique : Élaboration du schéma de la carte et routage des composants.
- Phase 3 : Montage et test fonctionnel du système : Assemblage de la carte et validation du bon fonctionnement des composants.
- Phase 4 : Programmation du microprocesseur : Développement du code pour la gestion des capteurs et de la communication des données.
- Phase 5 : Développement de l'interface graphique en Python : Création d'une interface pour l'affichage dynamique des données de l'IMU.
- Phase 6 : Modélisation d'une forme dynamique sur l'IHM Python : Visualisation interactive des données sous forme de formes géométriques animées sur l'IHM.

III. Choix des composants

Microprocesseur : ESP32-WROOM-32

L'ESP32 est une série de microprocesseurs de type système sur une puce (SoC) d'Espressif Systems, basée sur l'architecture Xtensa LX6 de Tensilica et intégrant la gestion du Wi-Fi et du Bluetooth. Il est très utilisé dans les systèmes embarqués et dans le domaine de l'IoT. Il permet également d'utiliser les protocoles de communication UART et I2C.

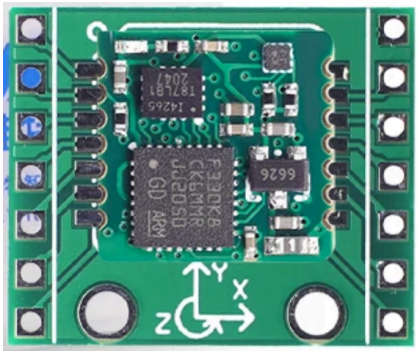
et SPI...



Capteur : Centrale inertielle WT931

Le module utilisé dans ce projet est le WT931, une centrale inertielle développée par WitMotion. Il s'agit d'une IMU (Inertial Measurement Unit) intégrant plusieurs capteurs permettant de mesurer avec précision les mouvements et l'orientation dans l'espace. Contrairement à un capteur inertiel isolé, ce module complet embarque également un traitement des données, facilitant son intégration dans les systèmes embarqués.

Le détail de son fonctionnement est présenté dans le chapitre IV : "Fonctionnement du capteur"



Une comparaison des capteurs aurait été bienvenue

IV. Fonctionnement du capteur

Le module WT931 est une centrale inertielle intégrant plusieurs capteurs permettant l'acquisition et le traitement des mouvements dans l'espace. Il repose sur la combinaison d'un accéléromètre triaxial, d'un gyroscope triaxial et d'un magnétomètre, offrant ainsi une mesure complète des accélérations linéaires, des vitesses angulaires et de l'orientation absolue.

Le module WT931 exploite un ensemble de capteurs pour assurer un suivi précis du mouvement :

- **Accéléromètre triaxial :** qui mesurent les accélérations linéaires selon les trois axes. Il permet notamment de détecter l'inclinaison en exploitant la projection de la gravité.
- **Gyroscope triaxial :** qui relèvent les vitesses angulaires, essentielles pour estimer les variations d'orientation en temps réel. Toutefois, ces mesures sont sujettes à une dérive progressive. Pas tout à fait
- **Magnétomètre :** qui détecte le champ magnétique terrestre, facilitant ainsi l'orientation absolue et permettant de corriger la dérive du gyroscope.

Le module envoie ensuite ces informations via l'interface I²C, mais une transmission en UART ou SPI peut également être utilisée en fonction des besoins du projet.

→ AHRS

V. Réalisation de la carte électronique

Le module WT931 communique avec le microprocesseur via une liaison I²C pour l'acquisition des données. Une autre méthode de communication possible est la transmission des données en UART via une connexion RX/TX, selon les besoins du système.

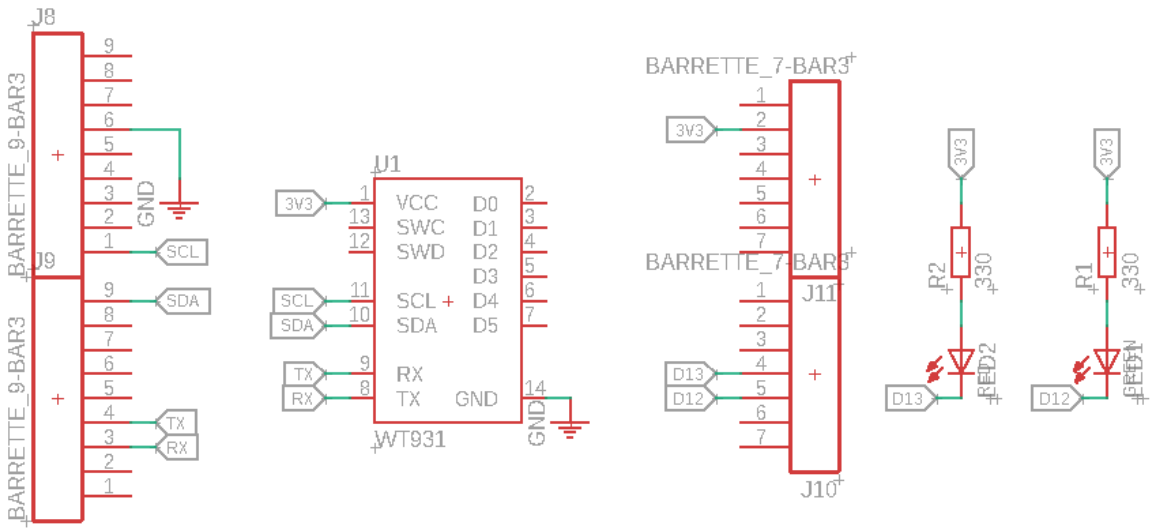
Afin de faciliter les connexions et d'assurer une intégration pratique des différents composants, des borniers ont été mis en place sur la carte. Ces borniers permettent de relier facilement le capteur WT931 au microprocesseur et d'autres périphériques, simplifiant ainsi l'assemblage et la modification du circuit en fonction des configurations souhaitées.

Une fois les données collectées, elles sont transmises via une liaison USB-C en utilisant une communication UART vers USB. Ce mode permet d'envoyer les informations vers un ordinateur afin de pouvoir traiter ces données.

Enfin, afin d'améliorer IHM, deux LEDs ont été ajoutées pour fournir un retour visuel sur l'état du système. Dans un premier temps, elles pourront permettre de vérifier le bon fonctionnement de la carte et aussi de visualiser dynamiquement le fonctionnement du système.

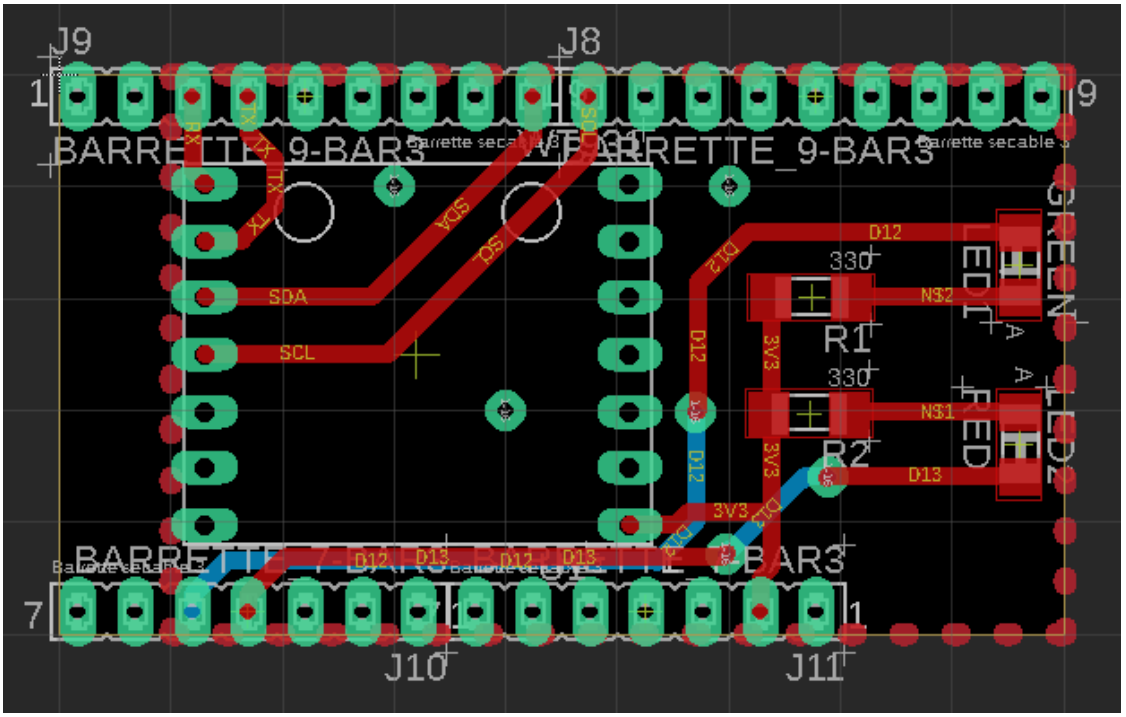
Qui a fait toute? Géraldine ?

- ## Réalisation du schéma structurel

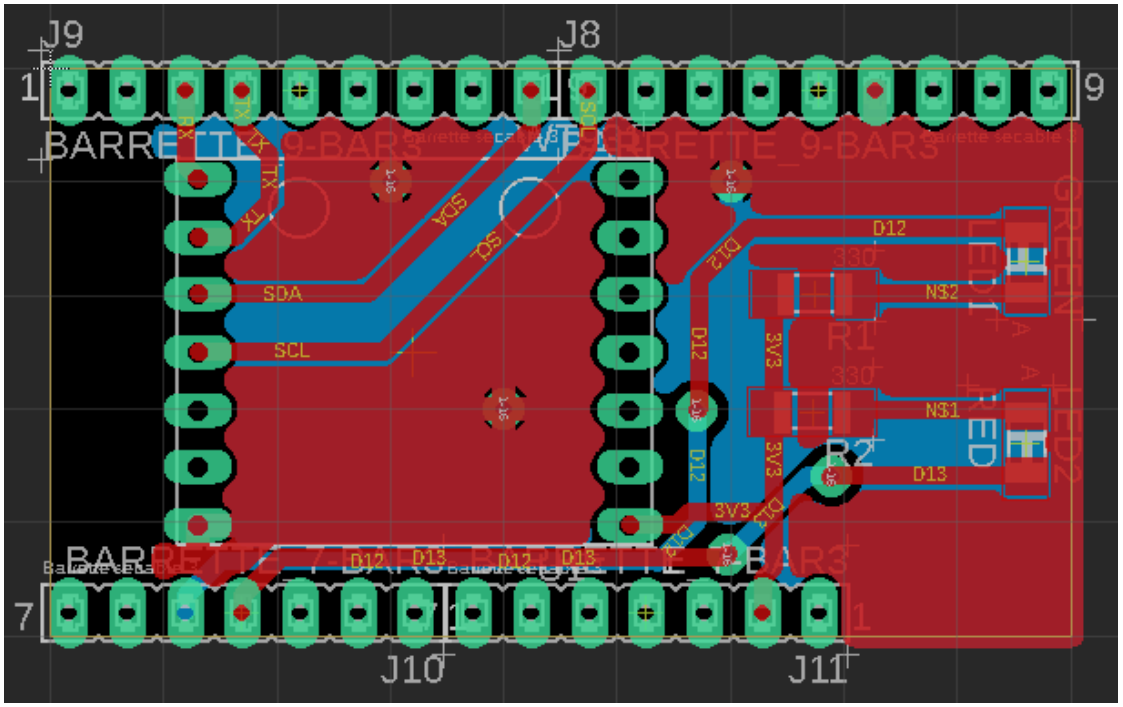


- Réalisation du routage

Routage :



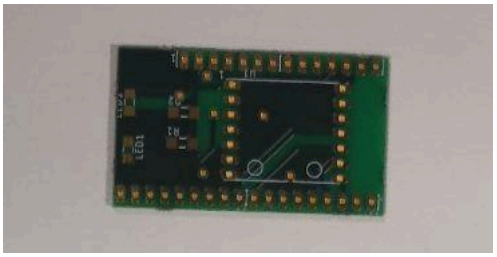
Plan de masse :



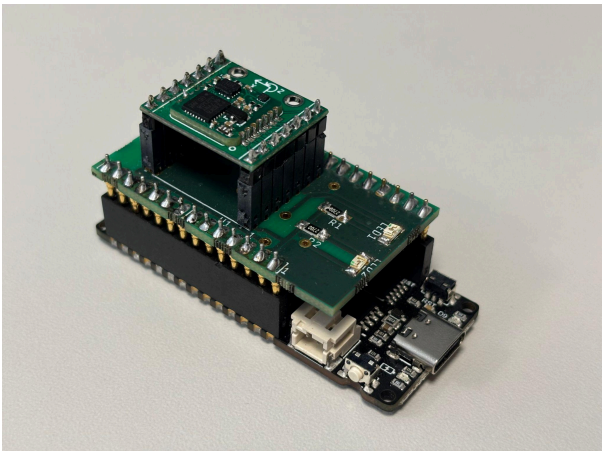
VI. Assemblage et test de la carte

○ Assemblage

Carte pour la liaison entre ESP32 et la centrale inertielle :

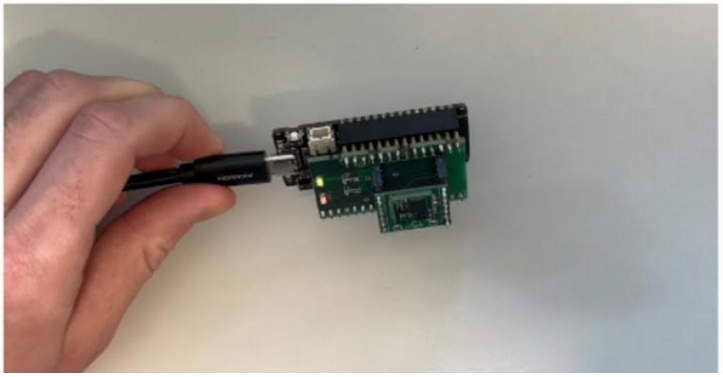


Module assemblée :



○ Tests de fonctionnement

Les premiers tests ont consisté à développer un programme simple allumant les LEDs pour vérifier le bon fonctionnement de la carte.



○ Tests de débogage

Les tests de débogage ont consisté en la récupération des données issues du capteur et leur affichage dans le terminal de l'IDE Arduino. Les premières tentatives ont permis de vérifier que les valeurs issues du capteur étaient lues correctement.

VII. Programmation de la carte

Le système a été modifié pour ne récupérer et afficher que les valeurs angulaires en utilisant les broches I2C. Les données sont lues et affichées en temps réel sur le terminal. Des ajustements ont été réalisés sur la vitesse de communication et les délais des boucles afin d'optimiser les performances. Les valeurs angulaires sont ensuite envoyées via UART par la connexion USB-C pour une transmission fluide des données. Des LEDs s'allument lorsque les valeurs de roll ou de pitch dépassent un seuil critique, simulant ainsi les points limites qu'un avion ne doit pas franchir. Cependant, une calibration est nécessaire sur l'axe de *yaw* après l'intégration du capteur dans la carte, car ce dernier a changé d'environnement, ce qui a affecté la précision des mesures.

Il faudrait détailler le protocole UART connect la carte communique avec le PC

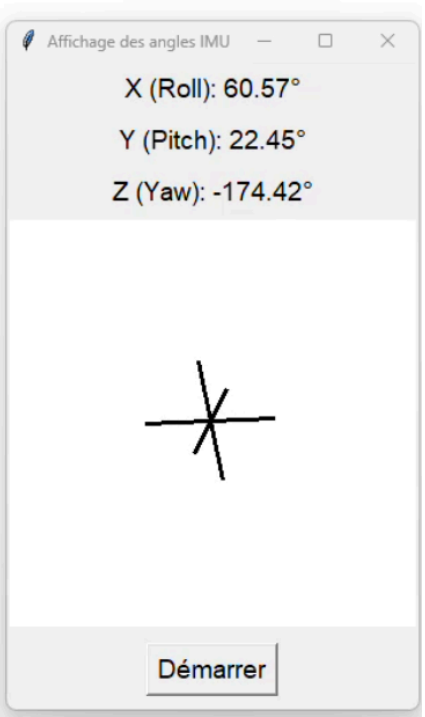
VIII. IHM Python

L'IHM a été réalisée en configurant la lecture des valeurs angulaires via le protocole I2C, avec un affichage dynamique des données sur le terminal. La précision des valeurs affichées a été ajustée pour améliorer la lisibilité. Tkinter a été utilisé pour modéliser des figures géométriques simples à l'écran.

La communication par port série a été configurée en installant la bibliothèque "pyserial" via la commande “sudo apt install pyserial” dans le terminal windows, afin d'assurer une communication fluide entre l’ordinateur et la centrale inertielle.

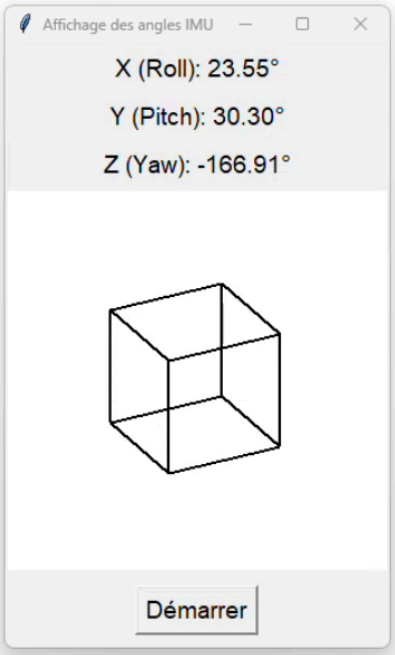
IHM avec une croix disponible dans les trois axes, X,Y et Z :

```
    label_y.config(text=f"Y (Pitch): {y.get():.2f}°")
    label_z.config(text=f"Z (Yaw): {z.get():.2f}°")
    dessiner_avion(x_val, y_val, z_val)

def dessiner_avion(roll, pitch, yaw):
    canvas.delete("all")

    cx, cy = 150, 150
    size = 50
    angle_rad_x = math.radians(roll)
    angle_rad_y = math.radians(pitch)
    angle_rad_z = math.radians(yaw)

    avion_points = [
        (-size, 0, 0), (size, 0, 0), (0, -size, 0), (0, size, 0),
        (0, 0, -size), (0, 0, size)
    ]

    rotated_points = []
    for px, py, pz in avion_points:
        x1 = px * math.cos(angle_rad_z) - py * math.sin(angle_rad_z)
        y1 = px * math.sin(angle_rad_z) + py * math.cos(angle_rad_z)
        z1 = pz

        x2 = x1
        y2 = y1 * math.cos(angle_rad_x) - z1 * math.sin(angle_rad_x)
        z2 = y1 * math.sin(angle_rad_x) + z1 * math.cos(angle_rad_x)

        x3 = x2 * math.cos(angle_rad_y) + z2 * math.sin(angle_rad_y)
        y3 = y2
        z3 = -x2 * math.sin(angle_rad_y) + z2 * math.cos(angle_rad_y)

        screen_x = cx + x3
        screen_y = cy - y3
        rotated_points.append((screen_x, screen_y))

    canvas.create_line(rotated_points[0], rotated_points[1], fill="black", width=3)
    canvas.create_line(rotated_points[2], rotated_points[3], fill="black", width=3)
    canvas.create_line(rotated_points[4], rotated_points[5], fill="black", width=3)


def init_connection():
    if ser and ser.is_open:
        messagebox.showinfo("Connexion", "ESP32 connecté !")
        threading.Thread(target=lire_donnees, daemon=True).start()
    else:
```



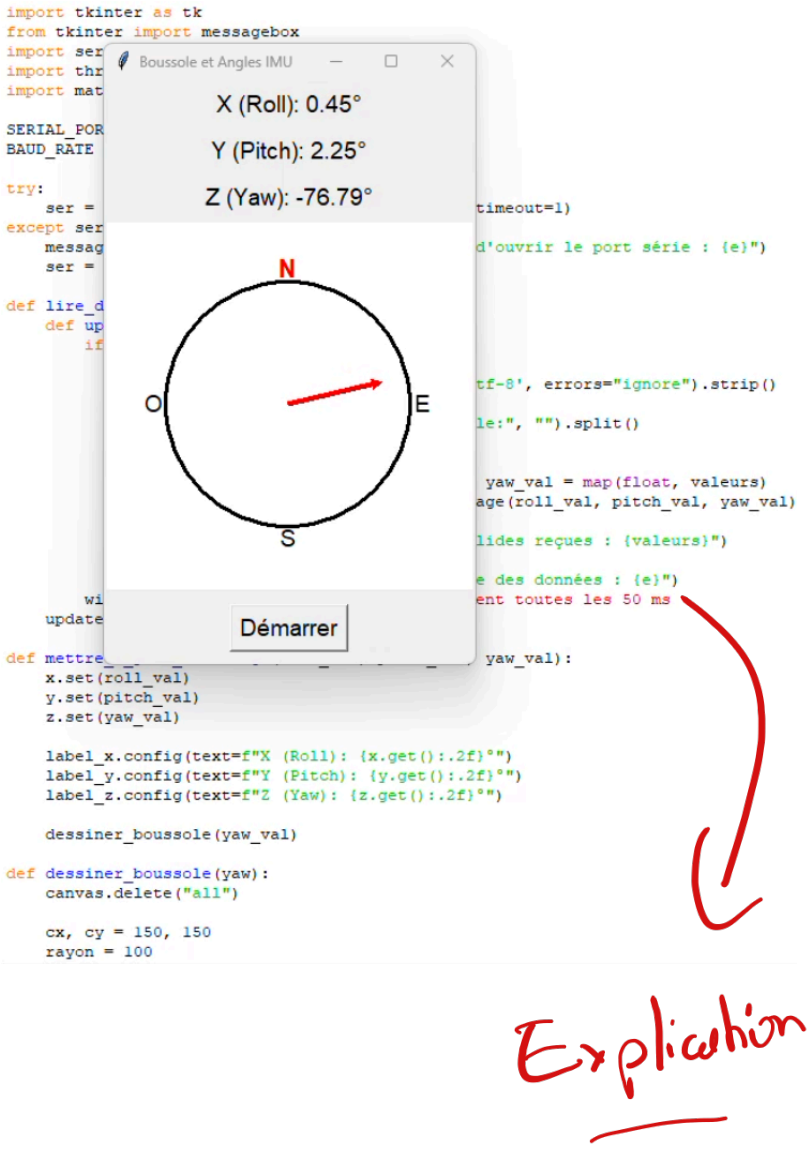
IHM avec un cube disponible dans les trois axes, X,Y et Z :

```
import tkinter as tk
from tkinter import messagebox
import serial
import threading
import math

SERIAL_PORT = "COM4"
BAUD_RATE = 115200

try:
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=0)
except serial.SerialException as e:
    messagebox.showerror("Erreur", f"Impossible d'ouvrir le port série : {e}")
    ser = None

def lire_donnees():
    def update():
        if ser and ser.is_open:
            try:
                ligne = ser.readline().decode('utf-8').strip()
                if ligne.startswith("angle:"):
                    valeurs = ligne.replace("angle:", "").split()
                    if len(valeurs) == 3:
                        try:
                            x_val, y_val, z_val = map(float, valeurs)
                            mettre_a_jour_affichage(x_val, y_val, z_val)
                        except ValueError:
                            print(f"Données invalides reçues : {valeurs}")
            except Exception as e:
                print(f"Erreur lors de la lecture des données : {e}")
        window.after(10, update)
    update()

def mettre_a_jour_affichage(x_val, y_val, z_val):
    x.set(x_val)
    y.set(y_val)
    z.set(z_val)
    label_x.config(text=f"X (Roll): {x.get():.2f}°")
    label_y.config(text=f"Y (Pitch): {y.get():.2f}°")
    label_z.config(text=f"Z (Yaw): {z.get():.2f}°")
    dessiner_cube(x_val, y_val, z_val)

def dessiner_cube(roll, pitch, yaw):
    """Dessine un cube en perspective avec axes X, Y et Z dynamiques"""
    canvas.delete("all")

    cx, cy = 150, 150
    size = 50
    angle_rad_x = math.radians(roll)
```



Idem

Une boussole a été développée, fonctionnant correctement pour les directions nord et sud, mais des problèmes sont apparus pour l'est et l'ouest. Cette erreur devrait être résolue grâce à un calibrage du capteur pour corriger les erreurs de rotation.



IX. Conclusion et perspectives

○ Conclusion

Ce projet a permis de concevoir un système embarqué intégrant une centrale inertielle pour acquérir et exploiter les données de mouvement en temps réel. L'architecture retenue repose sur un ESP32-C6-WROOM-1-N8, qui assure la gestion du capteur WT931 via une liaison I²C et une transmission des données en UART over USB. Cette configuration garantit un transfert efficace des mesures inertielles vers un système externe pour analyse et affichage.

L'intégration matérielle et logicielle a nécessité une approche méthodique, notamment pour le traitement des signaux issus des capteurs inertiels comme l'optimisation des fréquences d'échantillonnage. L'ajout de LEDs de diagnostic a permis de valider le bon fonctionnement de la carte et d'offrir un retour visuel dynamique lors des tests de mouvement.

Ce travail a mis en évidence les contraintes liées à l'implémentation de capteurs inertiels dans des systèmes embarqués, ainsi que les optimisations nécessaires pour garantir une exploitation robuste des données. L'expérience acquise dans la gestion des interfaces de communication, le traitement des signaux des capteurs et l'intégration matérielle constitue une base solide pour des améliorations et extensions futures.

○ Perspectives d'améliorations

Les évolutions possibles de ce projet portent sur l'amélioration des performances, la réduction de l'encombrement et l'optimisation des fonctionnalités pour une meilleure intégration dans des applications embarquées.

- **Miniaturisation du PCB**
  Une des améliorations majeures serait de réduire la taille du circuit imprimé en limitant le nombre de composants et en optimisant leur disposition. l'ESP32-C6-MINI-1 représente une alternative plus compacte et plus performante par rapport au modèle actuel. Son intégration faciliterait l'optimisation de l'espace tout en conservant les fonctionnalités essentielles, notamment la connectivité WiFi et Bluetooth. De plus, plutôt que d'utiliser la carte complète du WT931, il serait plus avantageux d'intégrer uniquement la puce inertielle directement sur notre circuit.

- **Communication WiFi pour un suivi à distance**
  L'ESP32 intégrant un module WiFi, il serait intéressant d'exploiter cette fonctionnalité pour transmettre les données en temps réel vers une interface web ou mobile. Cela offrirait un suivi distant sans nécessité de connexion filaire, rendant le dispositif plus flexible et mobile. → Socket ou Websocket

- **Ajout d'un écran pour un affichage en temps réel**
  L'intégration d'un écran OLED ou LCD permettrait d'afficher directement les valeurs angulaires sans passer par un ordinateur, améliorant ainsi l'autonomie et l'ergonomie du système.

- **Conception d'un boîtier robuste et étanche**
  Afin de garantir une utilisation fiable dans des environnements difficiles, la conception d'un boîtier étanche (IP65 ou IP67) protégerait l'électronique contre la poussière et l'humidité, augmentant ainsi la durabilité du dispositif.

↳ Parler des perturbations magnétiques ?

X. Bibliographie

Vidéo tutoriel WitMotion :
https://www.youtube.com/watch?v=dXVu5U45bMM&list=PL43tdDrVL_VDDciadEelCtvTh0dZjgHtS

Documents techniques et codes exemples du fabricant :
https://github.com/AiriYokochi/WT931/tree/master

Documentation Python :
https://docs.python.org/3/
https://docs.python.org/3/tutorial/

Développer une IHM Python :
https://vincent.developpez.com/cours-tutoriels/python/tkinter/apprendre-creer-interface-graphique-tkinter-python-3/

Documentation Tkinter :
https://infoforall.fr/python/python-act130.html
https://docs.python.org/3/library/tkinter.html
https://realpython.com/python-gui-tkinter/

XI. Résumé et mots clés/Abstract and keywords

Résumé :

Le projet “IMU WIRED” a pour objectif de concevoir et réaliser un système embarqué capable de mesurer et d'afficher des données angulaires en temps réel à l’aide d’une centrale inertielle (IMU) et d’un microprocesseur. Le capteur utilisé est la centrale inertielle WT931, qui intègre un accéléromètre, un gyroscope et un magnétomètre pour mesurer les mouvements et l'orientation dans l’espace. Le microprocesseur ESP32-WROOM-32 est utilisé pour gérer les données via une interface I2C et les transmettre par UART sur un ordinateur. Une interface graphique en Python, développée avec Tkinter, permet d’afficher ces données de manière dynamique et interactive. Le système a été testé et les performances optimisées pour garantir une communication fluide et un affichage précis des informations.

Mots clés :

Électronique - centrale inertielle - Python - IHM - Affichage dynamique - microprocesseur

Abstract :

The "IMU WIRED" project involves developing an embedded system that captures and visualizes angular data in real-time using an Inertial Measurement Unit (IMU) and a microcontroller. The IMU sensor, the WT931, combines a triaxial accelerometer, triaxial gyroscope, and magnetometer to measure 3D motion and orientation. Data is processed by an ESP32-WROOM-32 microcontroller, which interfaces with the IMU using I2C and sends data to an external device via UART over USB. A Python-based graphical user interface (GUI) built with Tkinter provides real-time dynamic visualization of the data. The project also includes system assembly, testing, and troubleshooting to ensure reliable data transmission and accurate sensor readings.

Keywords :

Electronics - inertial navigation sensor (INS) - Python - HMI - Dynamic Display - Microprocessor