INTERNSHIP REPORT SEMESTER 8

Presented by **Paul WALCZAK**

Assistant engineering internship

Presented and supported at Brest, the 05 September 2023

Jury composition :

ENIB tutor :	Alexis MICHEL	Directeur de l'ENIB
Company tutor :	Vincent KLYVERTS TOFTERUP	Responsable du développement des drones

Administrative information :

Company :	Upteko ApS
School :	École Nationale d'Ingénieurs de Brest (ENIB)
Dates :	du 06 Mars 2023 au 04 Aôut 2023
Student mail :	p9walcza@enib.fr

Acknowledgement

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to my internship supervisor Vincent KLYVERTS TOFTERUP, whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project especially in writing this report.

Furthermore, I would also like to acknowledge with much appreciation the crucial role of the staff at Upteko, who gave the permission to use all required equipment and the necessary materials to complete the task. A special thanks goes to my colleagues who helped me in many ways and were always willing to help when I ran into trouble.

I would also like to thank my school for providing me with this wonderful opportunity to apply my academic knowledge in a real-world scenario. The experience has been invaluable and I am grateful for the skills I have acquired and the insights I have gained.

Finally, I express my warm thanks to my family and friends for their understanding, endless patience and encouragement when it was most required. Their moral support has made this journey easier.

Table Of Content

List of Figures

List of Tables

INTRODUCTION

This report provides a detailed account of my internship experience at Upteko, a company that is experiencing rapid growth and is characterized by its diverse and dynamic work environment. Upteko operates in a multitude of fields, including but not limited to hardware, software, simulation, fundraising, and professional events. This diversity offers a unique learning experience, presenting an opportunity to gain exposure to a wide range of disciplines and technologies. It also poses its own set of challenges, as the breadth of the company's operations can sometimes be challeging for trainees.

The report begins with a comprehensive presentation of the company, outlining its field of activity, history, legal and social model, economic model, and type of work organization. This is followed by a detailed description of the projects I was involved in during my internship, including the `STM32 IMU project`, the `LogViewer project`, and the `Collision Avoidance SITL project`. Each project section provides an overview of the tasks I undertook, the challenges I encountered, and the solutions I developed.

Throughout the report, I aim to highlight the knowledge and skills I acquired during my internship, as well as the insights I gained into the complexities and peculiarities of working in such a multifaceted environment. The report also includes a list of codes used in the projects, providing a technical perspective on the work carried out.

Chapter 1

Company presentation

1.1 Field of activity

1.1.1 Wind energy

Upteko has been manually inspecting wind-turbines since 2018. Today, they are using this data and knowledge to build a system to automatically inspect wind turbines Upteko's autonomous drone system will provide high-quality aerial data for all inspections and maintenance requirements, in a safer, more efficient manner, without causing operational downtime. Their drones are pre-programmed with geo-referenced 3D trajectories for the inspection of wind turbines. While flying the planned routes, several images are acquired. The actual number depends on the camera specification, flight altitude and post-processing performance (overlap between images). All images are georeferenced using the UAV position at the acquisition time and these images are afterwards stitched together and in one picture on which the locations of the damages are shown. Depending on the client, many usages and services can be provided by Upteko such as windturbine inspection, runway inspections or payloads delivery.

1.1.2 Oil and Gas

Upteko is working on a solution for detecting oil spills on the surface of the water. They are developing payloads for their drone system that can help detect the signature of oil floating on water surfaces. This payload can be installed on Lärke and can be used to automatically or manually detect oil spills. Upteko have developed an application to detect pirate attacks on off-shore oil rigs and large commercial ships. Pirates use small ships to attack these oil rigs and large commercial ships at night. They take the crew and contents hostage. There are no mobile long-range detection tools with aerial insight to give the crew information about nearby pirates. This results not only in huge loss to the

oil industry but also endangers the crew at huge life risks. The total cost of piracy to the shipping industry, from insurance and time lost, is estimated at over €6B/year. Upteko's multipurpose drone system has a feature called perimeter control. This feature allows any member of the crew to investigate surrounding vessels approaching the premises. A built-in feature within the drone system is added to allow a crew member to operate and control the aerial camera in any direction with no prior drone experience. The geothermic cameras and the infrared sensors help to detect the approach of any pirates faster even during night-time.

1.1.3 Offshore and Maritime

Upteko has been consistently involved with the maritime industry since its inception. They are developing a drone system solution to live on the ship, to connect port operations on the ground and at the sea, with insight from the sky. This system includes a drone and a charging station for the drone that will be installed on the ship. The drone can autonomously perform a variety of tasks and then fly back and charge its batteries while staying protected against the weather. With an in-depth understanding of the challenges that the maritime industries face in attempting long and expensive inspections and other operational tasks, Upteko's software and hardware drone system allows a 100% automatic inspection of a ship in less than 2 hours. Having a permanent drone on a ship will be extremely helpful in a number of cases that can range from Search and Rescue (SAR) operations, vessel docking, dry dock inspections, fire hazard detection and situational awareness among other functionalities.

1.2 History

Upteko™ was founded in 2018 by Mads Joergensen, Benjamin Mejnertz, and Sebastian Duus to pursue the opportunity of developing drone applications for the maritime sector. For Benjamin and Sebastian, it started as a hobby, competing in RC Helicopter competitions around the world and later became a business worth pursuing. Mads came onboard, and together they created Upteko and built a great team of experienced drone pilots, software- and hardware engineers, and business developers. Today Upteko has offices located in Copenhagen, Odense, and Skanderborg. Upteko is about using their creativity and initiative to become leaders in the drone industry. To develop fitting so-

lutions to your needs, they value collaboration and co-creation highly among external parties. Through several years of collaborating with their customers, complementary assets providers, and even competitors, they have achieved priceless partnership within the maritime and other sectors. They are continuously on the hunt for new collaborations.

1.3 Legal and social model

In Denmark, most companies operate under a legal structure of a private limited company (Anpartsselskab, ApS) as Upteko. The country has strong social welfare systems and labor laws, which mandate fair treatment and protection of employees. Danish companies usually have a strong focus on maintaining a healthy work-life balance and uphold high ethical standards, both socially and environmentally.

1.4 Economic model

Denmark follows a mixed-market capitalist system, combining free market principles with a strong regulatory oversight. Upteko operates under a model of sustainable growth, focusing on long-term stability rather than short-term profits. This includes social responsibility, ethical business practices, and environmental sustainability.

1.5 Type of work organization

Denmark has a distinctive workplace culture, often characterized by a flat organizational structure. This implies low power distance, high levels of trust, and extensive collaboration between different levels of the organization. Danish companies typically have strong communication and decision-making practices, promoting employee empowerment and autonomy. Upteko is organised in three main sectors : `Technical Development`, `Finance` and `Product management`. Here is the actual organization chart of the company :

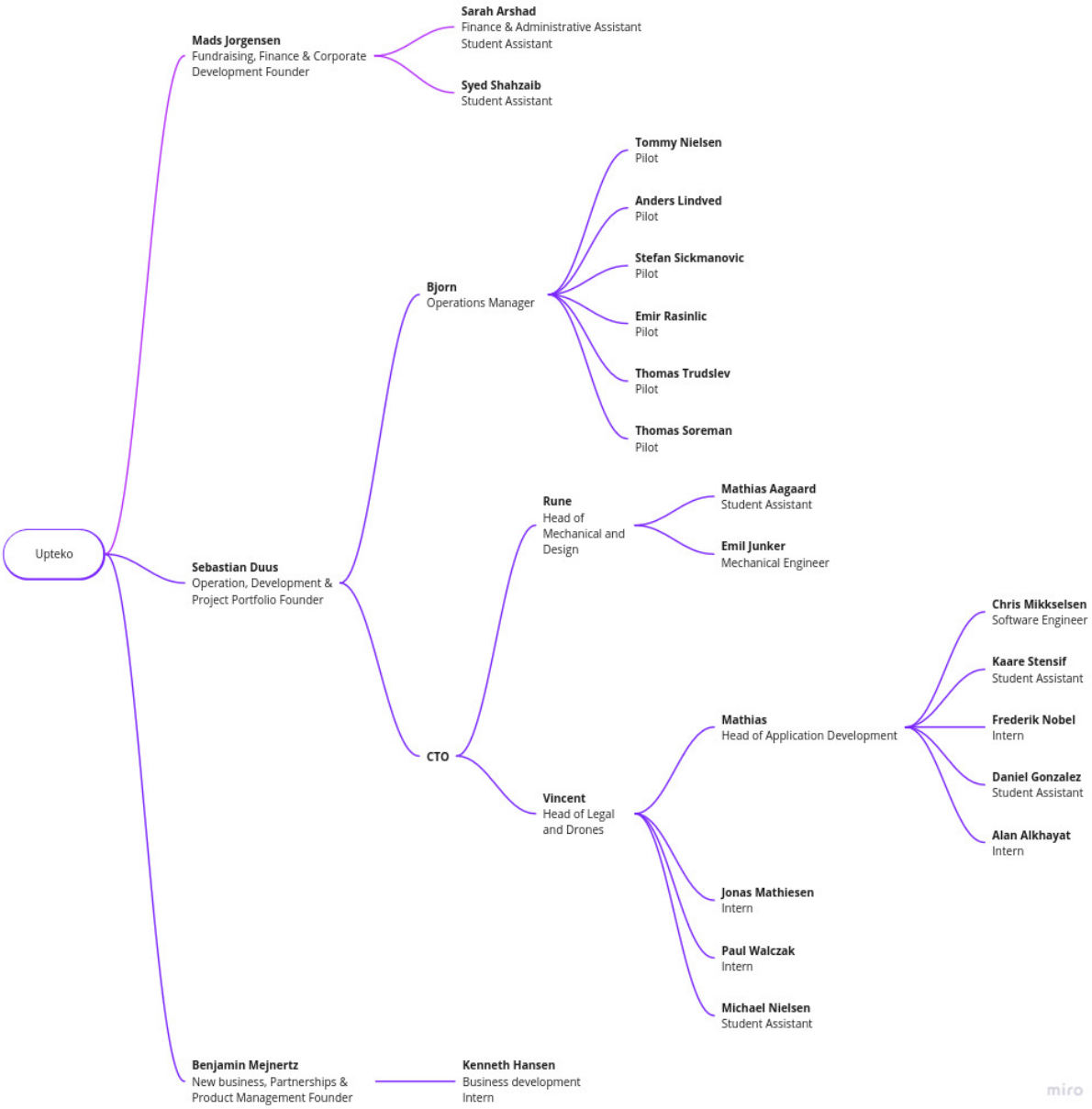


Figure 1.1 – Upteko company organization chart

Actual presence at work and office hours are the same in Denmark as in France. The Danes are highly flexible in this regard, allowing themselves to work 2 or 3 days a week remotely from home. Obviously, they don't do this every week, but it allows them to easily do tasks that don't require any particular communication "in peace" and in a more pleasant way.

Working from home and having flexible work hours offer numerous advantages, including increased productivity, improved work-life balance, and cost savings. However, they also present challenges such as social isolation, blurred boundaries, and communication difficulties. This way Upteko can maximize the benefits of flexible work arrangements and mitigate potential disadvantages, creating a positive and productive remote work environment.

Working with the compact drone development team at Upteko was an enriching experience. Despite our small size, the team was brimming with expertise and passion. The collaborative environment fostered innovation, and our shared enthusiasm made even challenging tasks enjoyable. We learned from each other, solved complex problems, and continually improved our designs. This experience underscored the power of teamwork in technological innovation.

Chapter 2

Projects

2.1 STM32 IMU

2.1.1 PModNav

The purpose is to get the `roll`, `pitch` and `yaw` from the PModNav to plot a 3D visualization of the IMU. The final purpose of it is to create log files from the orientation and attitude of the drone in real-time. By combining data from accelerometers, gyroscopes and magnetometers, an IMU can provide information about the object's position, orientation and angular velocity. This is crucial for tasks such as safety deployement or drone tracking. The STM32-L4796ZG recovers the value of the sensors from the `PModNav` via `SPI Bus` and transmits them to the HMI via `UART` connection.

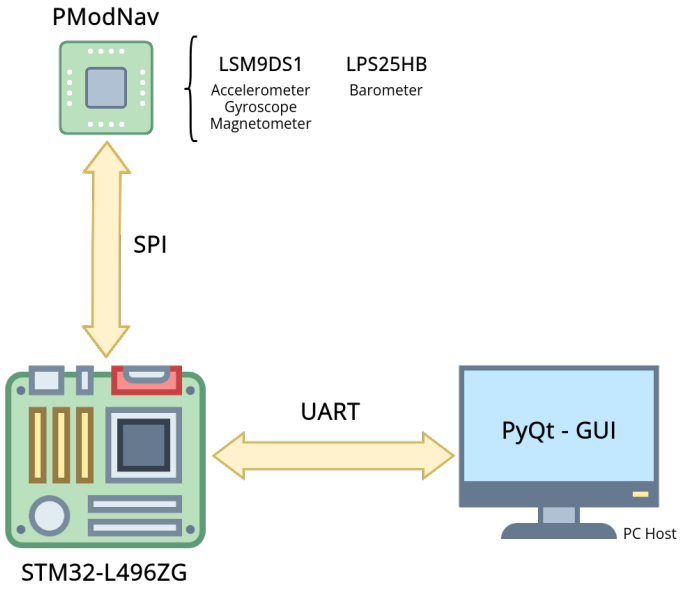


Figure 2.1 – Project communication overview

2.1.1.1 Hardware

The PModNav module is equipped with the `LSM9DS1` [1] sensor, offering 10-Degree Of Freedom (DOF) functionality. It integrates a 3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer, and an LPS25HB digital barometer. This comprehensive sensor suite allows users to obtain orientation-related data and determine the precise position and heading of the module. The module supports various full-scale options for linear acceleration, angular rate, and magnetic field measurements. It follows the Digilent Interface Specification Type 2A and utilizes a 12-pin Pmod connector with an SPI interface.

2.1.1.2 Software and Development Environment

The development environment for the PModNav project is STM32 CubeIDE. The documentation references project sources, including code snippets and libraries, such as the PModNav driver and Madgwick's filter implementation.

2.1.1.3 Data Processing

The project outlines two approaches for deriving object attitudes: Euler angles and quaternions. Euler angles are obtained through the integration of angular velocity and provide information about the roll, pitch, and yaw of an object. However, a challenge known as `Gimbal Lock` [2] arises when using Euler angles directly, resulting in a loss of a degree of freedom when two axes of rotation overlap. To overcome this, quaternions are introduced as a mathematical representation of displacement and rotation. They effectively resolve the Gimbal Lock issue and provide a more robust solution for determining attitudes.

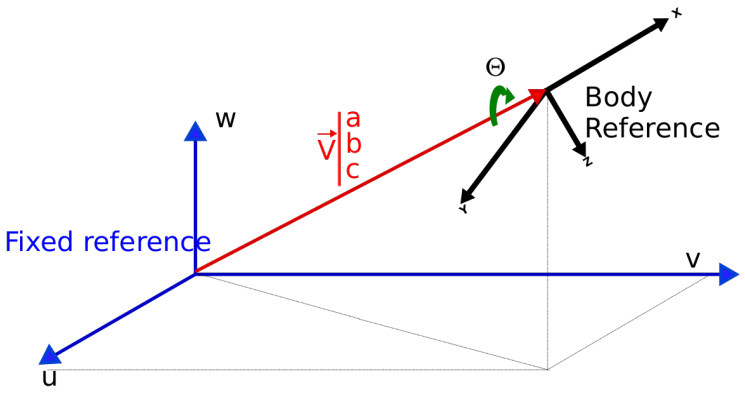


Figure 2.2 – Quaternion representation

$$Q = [q_0, q_1, q_2, q_3]$$
$$Q = \left[cos(\frac{\theta}{2}), a.sin(\frac{\theta}{2}), b.sin(\frac{\theta}{2}), c.sin(\frac{\theta}{2})\right]$$

At each `Te`, we can calculate the new value of the quaternion vector from the velocity :

$$Q_{k+1} = Q_k + \frac{1}{2}.T_e.\omega_k.Q_k$$

For cheap IMUs, it is unavoidable to perform a data fusion to make the accelerometer compensate for the gyroscope defect. If using a `Kalman filter` [3] is possible, there are other (faster) algorithms like Madgwick's. The idea is to compensate for the gyroscope measurement error by modulating its values by the result of a comparison between an estimate of the gravity field and the measured gravity field (with the accelerometer).

> Note: Only 6 DOF from the accelerometer and gyroscope have been use to this project due to too sensitive and hard to calibrate data from the magnetometer

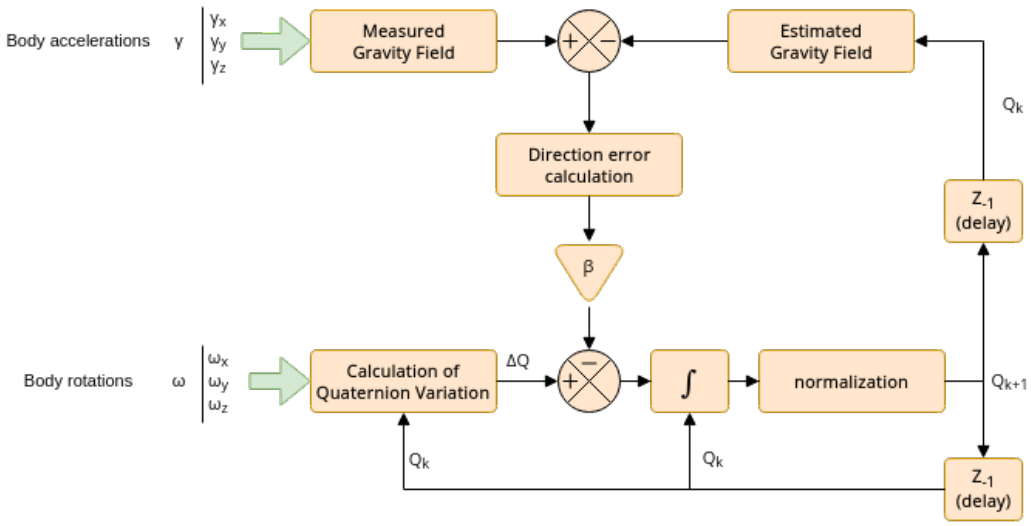


Figure 2.3 – Madgwick filter schematic

I decided to use the most popular open source algorithm to compute rotations, the

`Madgwick's algorithm` [4]. This calculation updates the quaternion, from which the attitudes (Euler angles) can be calculated.

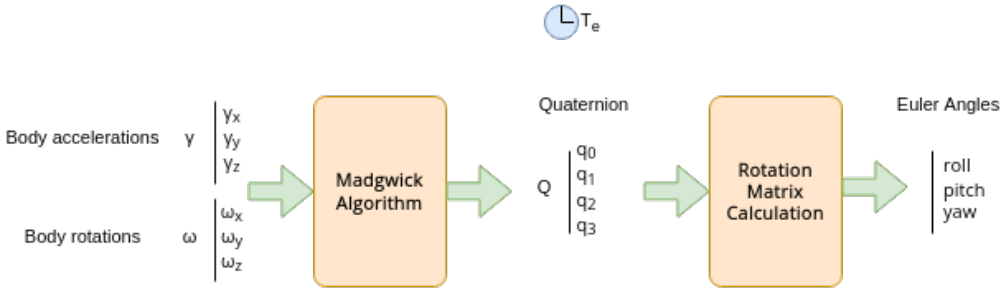


Figure 2.4 – Data transform schematic

Calculation of the Euler Angles from the rotation matrix :

$$
\begin{aligned}
R_{12} &= 2(q_1 q_2 + q_0 q_3) \\
R_{22} &= q_0^2 + q_1^2 - q_2^2 - q_3^2 \\
R_{31} &= 2(q_0 q_1 + q_2 q_3) \\
R_{32} &= 2(q_1 q_3 - q_0 q_2) \\
R_{33} &= q_0^2 - q_1^2 - q_2^2 + q_3^2
\end{aligned}
\qquad
\begin{aligned}
\text{roll} &= \text{atan2}(R_{31}, R_{33}) \\
\text{pitch} &= \sin(R_{32}) \\
\text{yaw} &= \text{atan2}(R_{12}, R_{22})
\end{aligned}
$$

2.1.2 Graphical User Interface

Additionally, a graphical user interface GUI is provided, leveraging `PyQt5` [5] and `PyOpenGL` [6] modules. The GUI manages the main window and handles OpenGL object management. It offers features like port selection and serial communication. The received data is displayed in a textbox within the GUI, facilitating real-time monitoring and analysis. Here is how the GUI is threaded :

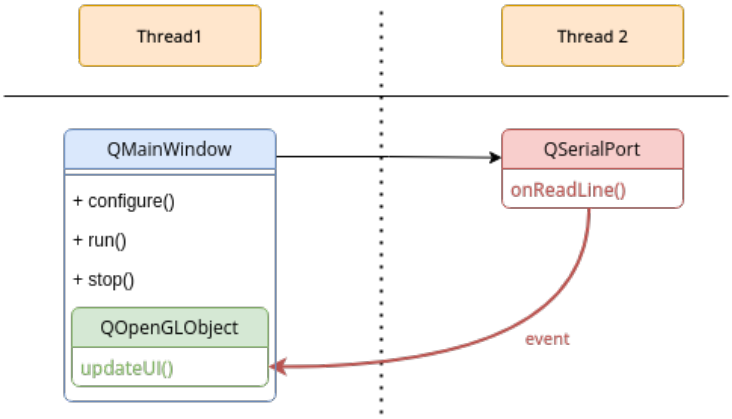


Figure 2.5 – Python GUI threads schematic

The use is rather simple for the communication configuration (cyan box) :

— the port
— the baud rate
— the number of bits per frame
— the number of stop bits
— the parity
— the flow control

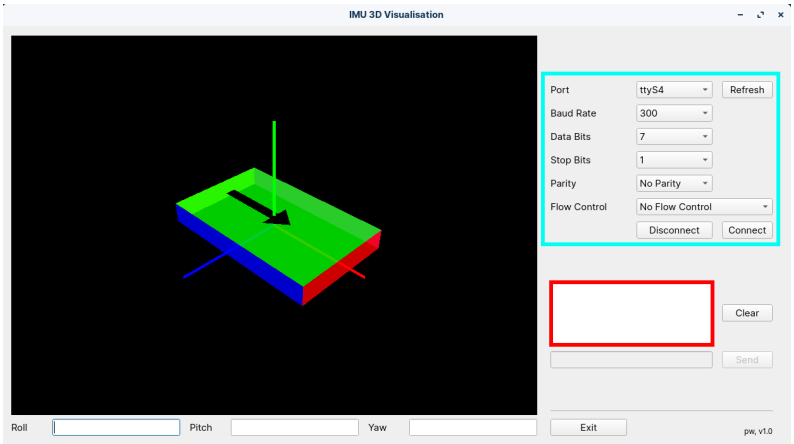


Figure 2.6 – Python GUI window screenshot

Then click on `Connect` to start the serial communication. Every received line appeared in the textBox (red box).

2.1.2.1 Final result

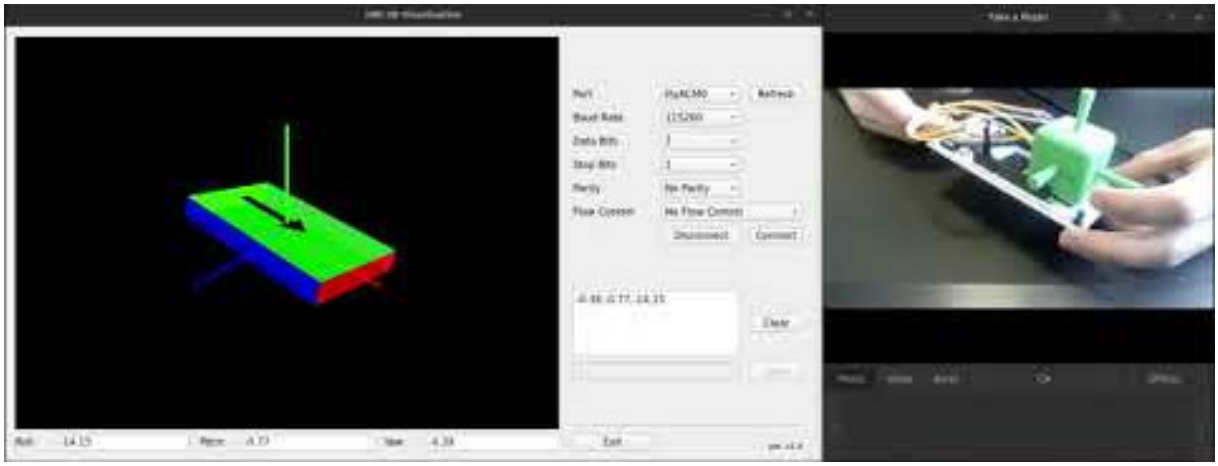


Figure 2.7 – 3D IMU Visualizer demo video

2.1.3 Real-Time Operating System

Drones, or unmanned aerial vehicles, have gained significant popularity and application across various industries. As drones become more advanced and complex, it is crucial to incorporate reliable systems for data logging and analysis. One such system is a blackbox, which serves as a flight data recorder for drones. Here are the reason of having a blackbox on a drone :

— Flight Data Logging: A blackbox records various flight parameters and sensor data during drone flights. This data can include information such as GPS coordinates, altitude, speed, battery voltage, motor RPM, control inputs, and more. Analyzing this data can provide valuable insights into the drone's performance, diagnose issues, and improve flight characteristics.

— Accident Investigation: In the event of a drone crash or incident, a blackbox can provide crucial data for accident investigation and analysis. It helps in understanding the sequence of events leading up to the accident, identifying any anomalies or failures, and determining the cause of the incident. This information can be used for troubleshooting, improving safety measures, and preventing similar incidents in the future.

— System Optimization: By analyzing the flight data recorded by the blackbox, drone manufacturers and developers can optimize the drone's performance, stability, and

efficiency. They can fine-tune flight control algorithms, evaluate sensor accuracy, and identify areas for improvement in terms of power consumption, aerodynamics, and overall system design.

— Regulatory Compliance: In some countries, drone operations may be subject to regulatory requirements. Flight data recording and analysis can help demonstrate compliance with regulations, such as maintaining safe distances from restricted areas or operating within altitude limits. The recorded data can serve as evidence of adherence to guidelines and regulations set by aviation authorities.

Overall, using a blackbox on a drone provides valuable data for performance analysis, troubleshooting, safety improvement, and compliance with regulations. It allows drone operators, manufacturers, and researchers to gain insights into flight characteristics and make informed decisions for enhancing drone performance and safety.

The best way to implement it is to use RTOS, they provide a highly accurate sampling period and easy task management for every kind of application. In my case, I have 2 tasks that should run independantly :

Task Name	Frequency [Hz]	Duration [ms]	Priority
IMU_Step	100 Hz	∼1 ms	High
Data_Log	4 Hz	∼175 ms	Low

Table 2.1 – RTOS tasks description

Beacause `IMU_Step`'s priority is higher than `Data_Log`'s priority and I want it to run at 100 Hz, I decided to use a semaphore on timer interuption to release the `IMU_Step` task.

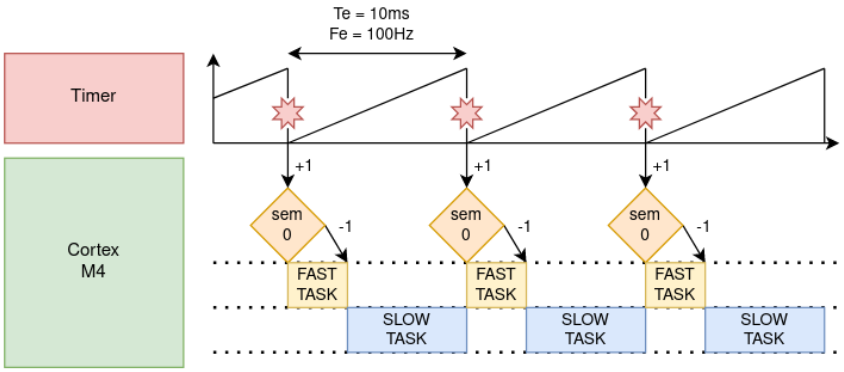


Figure 2.8 – RTOS overall chart

Here is the temporal plot of both tasks :

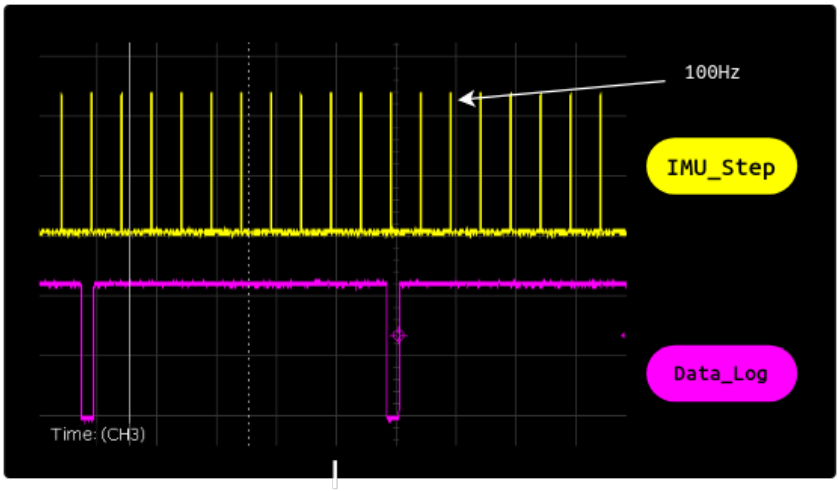


Figure 2.9 – Oscilloscope tasks measurements

Note:Everything work has planned up to 200 Hz for the IMU and the UART data log. However, it will be a big improvement to add a SDCard data storage to get all data

2.1.3.1 SD Card handler

The next improvment is to add the SDCard store feature, here is a Figure 2.11 to how the byte communication should be done for the next step. It's pretty simple : all typed

data are converted to bytes to reduce the memory size of it. The `UART` connection works and allow to send all data at ∼50 Hz and the SDCard store system should use the `SPI` or `SDMMC` connection.

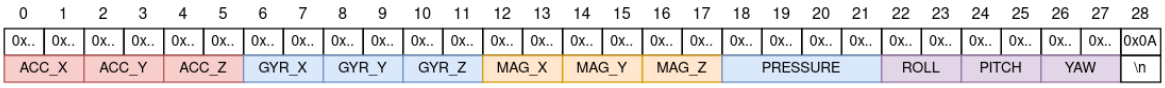


Figure 2.10 – Data frame as byte format

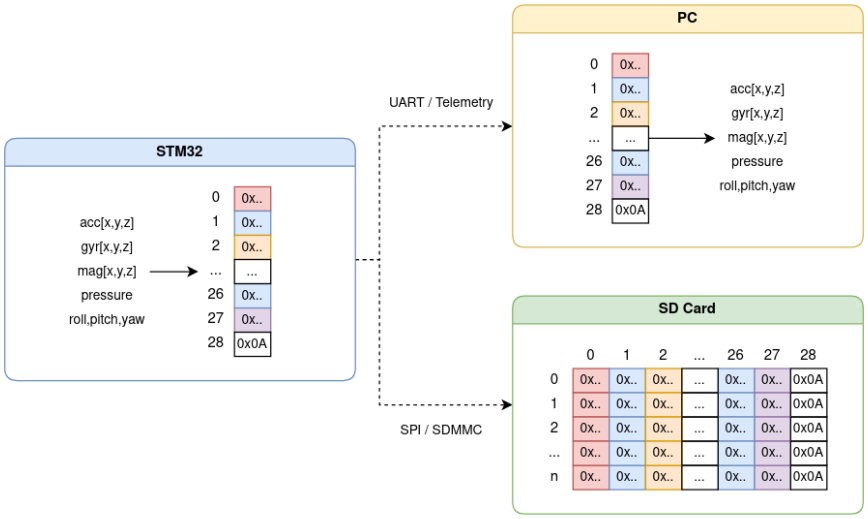


Figure 2.11 – STM32 data communication and storage

2.2 LogViewer

The objective of a log viewer application is to facilitate the efficient viewing and analysis of log data generated by systems or applications. This scientific report highlights the key goals and functionality of a log viewer, which includes log visualization, search and filtering capabilities, log analysis and insights, integration with other tools, and customization options.

2.2.1 Sensors tests

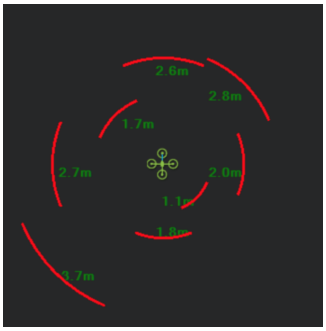
Before using any sensors, make sure they work properly, that communication is correct and that measurements are consistent with the reality of the environment. To check all these prerequisites, it's often easiest to use the manufacturer's proprietary software, which is robust, reliable and easy to use.

A few parameters need to be modified to enable the 8 distance sensors and their data to be taken into account by the collision avoidance built-in algorithm.

Here is the final configuration for a minimum effective distance of 3 meters between the drone and obstacles and a maximum effective distance of 5 meters :

	TowerEvo	Description
PRX_ORIENT	0	Default
PRX_TYPE	6	TeraRangerTowerEvo
SERIAL_BAUD	921600	Serial baud
SERIAL_PROTOCOLE	11	Lidar 360 deg
AVOID_ANGLE	1300	Max lean angle
AVOID_BEHAVE	1	Stop when obstacle detected
AVOID_DIST_MAX	5	Max distance avoidance
AVOID_ENABLE	3	Enable drone reaction
AVOID_MARGIN	3	Min distance avoidance

(a) Ardupilot flight controller configuration



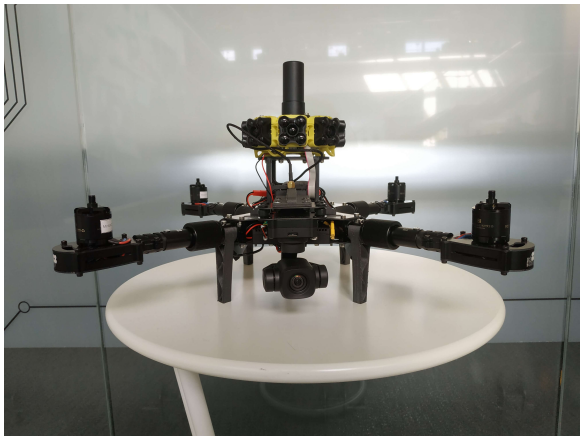
(b) Data acquisition acknowledgement

Table 2.2 – Ardupilot configuration & data acknowledgement

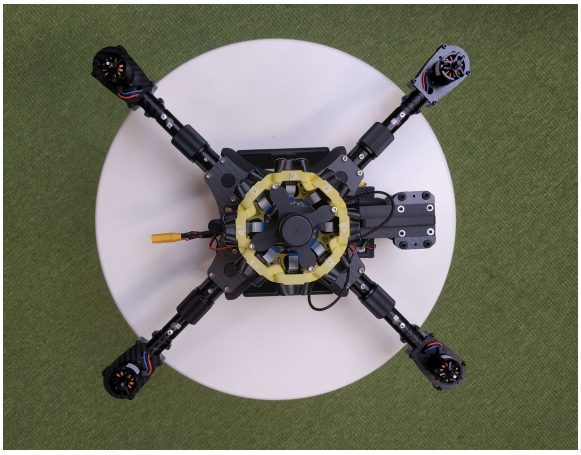
2.2.2 Sensors Integration and drone flight

(a) Front left view

(b) Left view



(c) Front view



(d) Top view

Figure 2.12 – Larke Mini drone pictures with 360° degrees LiDAR mounted

The sensors are not integrated into the hull, requiring a large, bulky tower on top of the Larke Mini drone. The integration with the drone is not aesthetic but functional.

Nevertheless the drone successfully achieved its intended mission objectives. Whether it was aerial mapping or surveying, the drone captured distance data effectively. My colleague Sebastian Duus ensured that the drone was in the optimal position for data acquisition, adjusting altitude and camera settings accordingly.

2.2.3 Python Plotly Visualizer

A graphical user interface is provided, leveraging `Plotly` [7] library. The Logviewer import the `.csv` file as data. Here is the `logfile.csv` header :

timestamp(ms)	PRX.D0	PRX.D45	PRX.D90	PRX.D135	PRX.D180
PRX.D225	PRX.D270	PRX.D315	ATT.Roll	ATT.Pitch	ATT.Yaw

Table 2.3 – Plotly logviewer file header

The application is pretty simple : it use the timestamp as main variable to create animated data. I created 3 main plots in th same window :

— the distance sensors plot

— the roll and pitch attitude

— the yaw plot

The most valuable part of this application is the time slider that can be used to go back in time, pause and restart the view. This way the data analysis is easy-to-use.

2.2.3.1 Final result

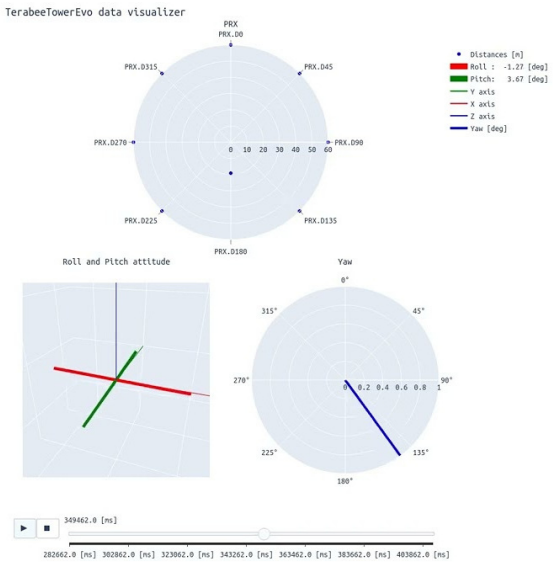


Figure 2.13 – Python LogViewer demo video

2.3 Collision avoidance SITL

2.3.1 Environement description

2.3.1.1 Robot Operating System

`Robot Operating System 2 (ROS2)` is an open-source framework for building robotic systems. It supports distributed systems, allowing communication between different components and nodes running on separate devices. It provides a communication infrastructure for exchanging messages and supports various protocols. ROS2 introduces a quality of service system for controlling communication behavior. It includes real-time capabilities for time-critical applications. It supports multiple programming languages and emphasizes modularity and extensibility. ROS 2 comes with development tools and has an active community and ecosystem. Overall, ROS2 provides a powerful and versatile platform for developing robotic systems. Its distributed architecture, flexible communication infrastructure, real-time capabilities, language support, and extensibility make it a valuable tool for building complex and advanced robots.

2.3.1.2 Gazebo

`Gazebo` is an open-source 3D simulation environment for robotics. It allows developers to create virtual worlds where they can simulate and test robots. Gazebo provides realistic physics-based simulations, visualizes the environment in 3D, and simulates sensors like cameras and LiDARs. Users can model robots and their surroundings, integrate with the Robot Operating System (ROS), and extend its functionality using plugins. Gazebo has a supportive community and offers a wide range of resources for developers. In summary, Gazebo is a valuable tool for simulating and evaluating robotic systems before deploying them in real-world environments.

2.3.1.3 Ardupilot

`Ardupilot` is open-source software used for controlling autonomous vehicles like drones. It includes firmware that runs on the vehicle's autopilot hardware, as well as ground control station Ground Control Station (GCS) software for mission planning and monitoring. Ardupilot supports different flight modes, navigation based on waypoints, and telemetry for communication with the vehicle. It is compatible with various hardware platforms and can be customized and extended. Ardupilot is widely used in drones

and other vehicles, and it has a supportive community and extensive documentation. In summary, Ardupilot is a versatile and customizable software suite for autonomous vehicle control.

2.3.2 Software-In-The-Loop Ardupilot collision avoidance

The collision avoidance feature in Ardupilot is designed to enhance the safety and reliability of these vehicles by detecting and avoiding potential collisions with obstacles or other aircraft.
The collision avoidance system in Ardupilot relies on various sensors and algorithms to perceive the environment and make informed decisions about navigation. Some of the key components and techniques used in the Built-In Ardupilot collision avoidance include:

- Sensor Inputs: Ardupilot can interface with different types of sensors, including proximity sensors, LiDAR, sonar, and cameras. These sensors provide information about the surrounding environment and help in detecting obstacles.
- Obstacle Detection: The collision avoidance system processes the sensor data to identify potential obstacles in the flight path. This can include buildings, trees, other aircraft, or any other objects that may pose a risk of collision.
- Path Planning: Based on the environment map and the current position of the vehicle, Ardupilot generates a safe and collision-free path for the aircraft to follow. It considers factors such as obstacle proximity, vehicle speed, and maneuverability to calculate an optimal trajectory.
- Collision Avoidance Algorithms: Ardupilot employs sophisticated algorithms to predict the future movement of obstacles and determine the best course of action to avoid collisions. These algorithms take into account the dynamics of the vehicle and the obstacles.

Software-In-The-Loop (SITL) is a testing and simulation technique used in software development and engineering. It involves the integration of software components into a simulated environment or virtual platform to assess their functionality and behavior. Using it, developers can identify and resolve potential issues and bugs in the software early in the development cycle, reducing costs and risks associated with late-stage failures.
The Figure 2.14 represent the non-detailed structure of the work environment and the communication between each of its components :

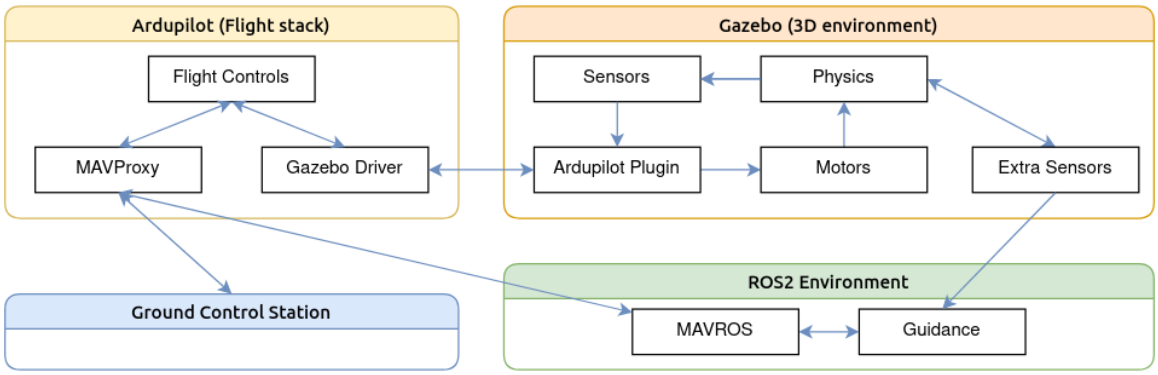


Figure 2.14 – Software-In-The-Loop overall chart[8]

It took a number of steps to obtain a vehicle that could be used in simulation. All the control of the motors and their physical action in Gazebo is already done, but with the integration of the TeraTowerEvo distance sensors.
For the sensors to work as realistically as possible, they need to be parameterized in such a way that the minimum and maximum distances correspond to the datasheet, that the angle of vision is also the same, and to add a little noise to the measurements captured.

Once everything is set up, the data can be easily analyzed in the integrated ROS interface (cf. Figure 2.15):

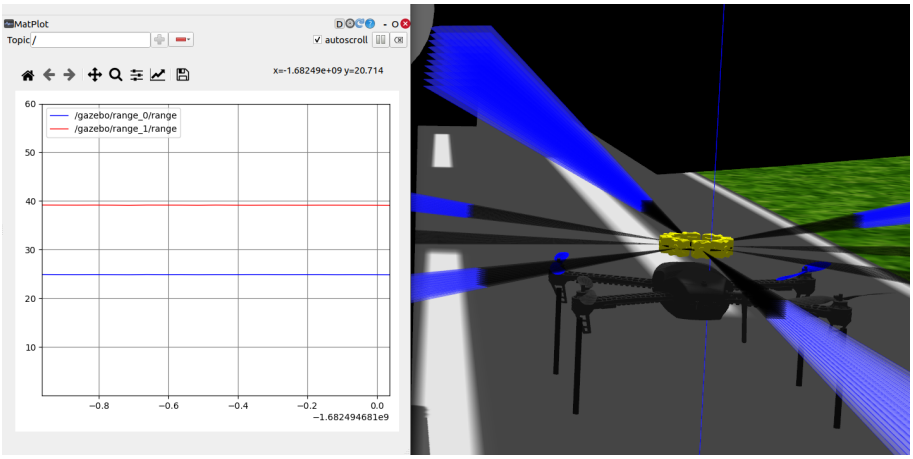


Figure 2.15 – Drone with TowerEvo in Gazebo environment

2.3.3 Collision Avoidance ROS2 Migration

2.3.3.1 ROS1 collision-avoidance description

The aim of this task is to migrate a project from the old version of Robot Operating System (ROS) to the new one ROS2.

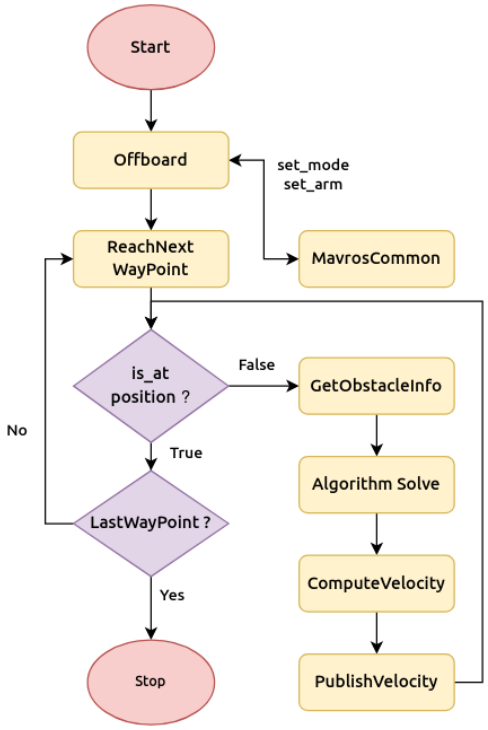


Figure 2.16 – Collision Avoidance flowchart

The Figure 2.16 represent how the collision avoidance logic is working. It's basically a loop that covers all the waypoints to reach the final destination.The data acquired by the distance sensors are continuously recorded to enable the algorithm to detect obstacles and change the speed and direction of the drone in real time. As soon as data are acquired, this action chain start : `GetObstacleInfo`, `Algorithm Solve`, `ComputeVelocity` and `PublishVelocity`.

It use the `potential field method` [9], it's an algorithm used in robotics for motion planning. It represents the environment as a field of attractive and repulsive potentials. The agent is guided towards the goal by attractive potentials and repelled by repulsive potentials generated by obstacles. The agent follows the gradient of the potential field to navigate towards the goal while avoiding obstacles. It is a simple and intuitive approach but can have limitations in complex environments and local minima trapping. Extensions and modifications exist to address these challenges.

2.3.3.2 ROS migration

Migration is a complex process that requires a thorough understanding of the environment in which you're moving and the environment in which you're moving. What's more,

programming languages (python here) often evolve between 2 ROS releases, so language errors add to the difficulty. I successfully migrate all the files as espected but unfortunatly after 3 weeks working on this tasks, I wasnt able to debug and test the project due to the environment complexity.

My colleagues are way more experienced in ROS2 and migration, so we decided together to stop my part here and let them continue to debug and test the project.
I did, however, learn a lot of principles, such as Quality Of Services (QOS) and Micro Air Vehicle Link (MAVLink) message communication, which are essential for flying robotics projects.

Chapter 3

Gantt Task Organization

FILENAME	UPTEKO - Assistant Engineer Intership	COMPANY	Upteko Aps
NAME	Paul Walczak	DATE	24/05/23

	TASK	DURATION	STATUT	WEEK NUMBER 10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	**STM32**																holidays							company holidays	
1.1	**Terabee sensor communication**	1	100 %																						
1.1.1	UART communication	1	100 %																						
1.2	**PModNav**	7	100 %																						
1.2.1	SPI communication	1	100 %																						
1.2.2	IMU parameters	2	100 %																						
1.2.3	Sensors calibration	2	100 %																						
1.2.4	AHRS algorithm implementation	2	100 %																						
1.2.5	Tests	2	100 %																						
1.2.6	Documentation	1	100 %																						
1.3	**Python GUI**	2	100 %																						
1.3.1	Serial communication	1	100 %																						
1.3.2	QT Designer	2	100 %																						
1.3.3	OpenGL	2	100 %																						
1.3.4	Documentation	1	100 %																						
1.4	**RTOS**	5	83 %																						
1.4.1	RDF-868 configuration	1	100 %																						
1.4.2	FreeRTOS implementation	4	100 %																						
1.4.3	FreeRTOS & SD Card	4	50 %																						
1.4.4	IMU merge	2	100 %																						
2	**Collisions Avoidance SITL**																								
2.1	**Ardupilot collision avoidance built-in**	3	64 %																						
2.1.1	Learn Gazebo environment	2	80 %																						
2.1.2	Learn ROS1/2 environment	2	75%																						
2.1.3	Ardupilot parameters	2	50 %																						
2.1.4	Software-In-The-Loop	2	50 %																						
2.2	**Collision Avoidance ROS2 Migration**	3	65 %																						
4.1	Rewrite	2	100 %																						
4.2	3D environment	2	100 %																						
4.3	Ardupilot environment	1	100 %																						
4.4	Debug	2	25 %																						
4.5	Tests	0	0 %																						
3	**Log Viewer**																								
3.1	**Python GUI**	2	100 %																						
3.1.1	Sensors tests	1	100 %																						
3.1.2	Drone integration	1	100 %																						
3.1.3	Drone flight	1	100 %																						
3.1.4	Python Plotly lib	2	100 %																						
5	**Others**		100 %																						
5.1	Production line assembly	1	100 %																						
5.2	Drone disassembly	1	100 %																						
5.3	MiniPC comparisioin sheet	1	100 %																						
6	**Internship Report**																								
6.1	**Internship Report**	10	100 %																						
6.2	Writing	9	100 %																						
6.3	Latex	8	100 %																						
6.4	Reading	2	100 %																						

Figure 3.1 – Gantt chart

Conclusion

The internship at Upteko was a unique and enriching experience that provided me with the opportunity to learn a lot about myself and professionally. The company's rapid growth and the constant influx of new trainees created a dynamic and stimulating work environment. However, this also presented certain challenges. The most significant of these was the difficulty in fully assimilating and mastering new technologies due to the breadth and complexity of the tasks at hand.

Despite the occasional frustrations, the experience was invaluable. It allowed me to apply the theoretical knowledge I had acquired in my academic studies to practical, real-world problems. The complexity of the tasks and the need for quick adaptation to new technologies and disciplines were significant learning experiences that will undoubtedly be beneficial in my future career.

In conclusion, the internship was a journey of discovery and learning, offering an overview into the real-world applications of my academic knowledge. By the end of my 5 months internship, I was involved in 3 main projects and their documentation. All of them will be use as a solid foundation for future development and intership subject. Despite the challenges, I am grateful for the experience and the skills I have acquired, and I look forward to applying these in my future career.

Bibliography

[1] Digilent. « Vivado pmodnav c library ». (), [Online]. Available: https://github.com/Digilent/vivado-library/tree/master/ip/Pmods/PmodNAV_v1_0/drivers/PmodNAV_v1_0.

[2] OpenSource. « Gimbal lock explained ». (), [Online]. Available: https://en.wikipedia.org/wiki/Gimbal_lock.

[3] OpenSource. « Kalman filter - wikipédia ». (), [Online]. Available: https://en.wikipedia.org/wiki/Kalman_filter.

[4] x-ioTechnologies. « Open source imu and ahrs algorithms - x-io technologies ». (), [Online]. Available: https://x-io.co.uk/open-source-imu-and-ahrs-algorithms/.

[5] QtCompany. « Pyqt5 documentation ». (), [Online]. Available: https://www.riverbankcomputing.com/static/Docs/PyQt5/.

[6] OpenGLGroup. « Pyopengl documentation ». (), [Online]. Available: https://mcfletch.github.io/pyopengl/index.html.

[7] PlotlyGroup. « Plotly official website ». (), [Online]. Available: https://plotly.com/.

[8] IntelligentQuads. « Sitl explained ». (), [Online]. Available: https://www.youtube.com/watch?v=W2ncr0DKWHE&ab_channel=IntelligentQuads.

[9] A. Azzabi. « An advanced potential field method proposed for mobile robot path planning ». (), [Online]. Available: https://doi.org/10.1177/0142331218824393.

[10] J. Lee. « Fatfs sd library ». (), [Online]. Available: https://github.com/eziya/STM32_SPI_SDCARD/.

Acronyms

DOF Degree Of Freedom. 14, 15

GCS Ground Control Station. 25

GUI Graphical User Interface. 5, 16, 17

LiDAR Light Detection And Ranging. 5, 23, 25, 26

MAVLink Micro Air Vehicle Link. 29

QOS Quality Of Services. 29

ROS Robot Operating System. 25, 28

ROS2 Robot Operating System 2. 25, 28

RTOS Real-Time Operating System. 6, 19

SITL Software-In-The-Loop. 26

APPENDIX

```
/* Private typedef
   -----------------------------------------------------------*/
extern UART_HandleTypeDef hlpuart1;
extern SPI_HandleTypeDef hspi1;
extern uint8_t divider;


uint8_t divider = 0;
char buffer[50];
uint8_t size = 0;

osThreadId Slow_TaskHandle;
osThreadId Fast_TaskHandle;
osSemaphoreId BinSemHandle;

void MX_FREERTOS_Init(void) {
  /* definition and creation of BinSem */
  osSemaphoreDef(BinSem);
  BinSemHandle = osSemaphoreCreate(osSemaphore(BinSem), 1);

  /* definition and creation of Slow_Task */
  osThreadDef(Slow_Task, Start_Slow_Task, osPriorityBelowNormal,0,128);
  Slow_TaskHandle = osThreadCreate(osThread(Slow_Task), NULL);

  /* definition and creation of Fast_Task */
  osThreadDef(Fast_Task, Start_Fast_Task, osPriorityAboveNormal,0,128);
  Fast_TaskHandle = osThreadCreate(osThread(Fast_Task), NULL);

  HAL_TIM_Base_Start_IT(&htim3);
}

void Start_Slow_Task(void const * argument)
{
  /* Infinite loop */
  osDelay(100);
```

```
35   for(;;)
36   {
37     if(divider >= 2) {
38       //NAV_SendUartAngles(&nav, &hlpuart1);  // IMU 3D viz
39       NAV_SendUartRawFrame(&nav, &hlpuart1);  // Data logger
40
41       divider = 0;
42     }
43     osDelay(10);
44   }
45 }
46
47 void Start_Fast_Task(void const * argument)
48 {
49   /* Infinite loop */
50   for(;;)
51   {
52     osSemaphoreWait(BinSemHandle, osWaitForever);
53     divider ++;
54     NAV_Step(&nav, &hspi1, &hlpuart1);
55   }
56 }
```

Code 3.1 – freertos.c

```
1 #ifndef INC_PMODNAV_H_
2 #define INC_PMODNAV_H_
3
4 /*********** Function declarations *************/
5 void NAV_WriteSPI(SPI_HandleTypeDef *hspi, GPIO_TypeDef *csPort,
     uint16_t csPin, uint8_t addr, uint8_t val);
6 uint8_t NAV_ReadSPI(SPI_HandleTypeDef *hspi, GPIO_TypeDef *csPort,
     uint16_t csPin, uint8_t addr);
7 void NAV_ReadRegister(SPI_HandleTypeDef *hspi, GPIO_TypeDef *csPort,
     uint16_t csPin, uint8_t addr, uint8_t nBytes, uint8_t *data);
8
9 void NAV_InitAG(SPI_HandleTypeDef *hspi, GPIO_TypeDef *csPort, uint16_t
     csPin);
10 void NAV_InitMAG(SPI_HandleTypeDef *hspi, GPIO_TypeDef *csPort, uint16_t
     csPin);
11 void NAV_InitALT(SPI_HandleTypeDef *hspi, GPIO_TypeDef *csPort, uint16_t
     csPin);
```

```
12
13 void NAV_CalibrateIMU(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi);
14
15 void NAV_Step(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi,
       UART_HandleTypeDef *uart);
16
17 void NAV_GetData(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi);
18 void NAV_GetRawData(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi);
19
20 void NAV_ReadAccelG(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi);
21 void NAV_ReadAccel(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi);
22 float NAV_ConvertRawToG(PmodNAV *InstancePtr, int16_t rawVal);
23 void NAV_ReadGyroDps(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi);
24 void NAV_ReadGyro(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi);
25 float NAV_ConvertRawToDps(PmodNAV *InstancePtr, int16_t rawVal);
26 void NAV_ReadMagGauss(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi);
27 void NAV_ReadMag(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi);
28 float NAV_ConvertRawToGauss(PmodNAV *InstancePtr, int16_t rawVal);
29
30 void NAV_ReadPressurehPa(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi);
31 void NAV_ReadPressure(PmodNAV *InstancePtr, SPI_HandleTypeDef *hspi);
32
33 void NAV_PrintAccelG(PmodNAV *InstancePtr, UART_HandleTypeDef *uart);
34 void NAV_PrintAccelRaw(PmodNAV *InstancePtr, UART_HandleTypeDef *uart);
35 void NAV_PlotUart3Floats(UART_HandleTypeDef *uart, float a, float b,
       float c);
36 void NAV_PrintGyroDps(PmodNAV *InstancePtr, UART_HandleTypeDef *uart);
37 void NAV_PrintMagGauss(PmodNAV *InstancePtr, UART_HandleTypeDef *uart);
38 void NAV_PrintPressurehPa(PmodNAV *InstancePtr, UART_HandleTypeDef *uart
       );
39
40 void NAV_SendUartAngles(PmodNAV *InstancePtr, UART_HandleTypeDef *uart);
41 void NAV_SendUartRawFrame(PmodNAV *InstancePtr, UART_HandleTypeDef *uart
       );
42
43 void NAV_MadgwickAHRS(PmodNAV *InstancePtr);
44
45 #endif /* INC_PMODNAV_H_ */
```

Code 3.2 – PModNav.h

Assistant engineering internship

Intern: Paul WALCZAK
Semester: Semestre 8
Company: Upteko ApS
Company tutor:Vincent KLYVERTS TOFTERUP
ENIB tutor: Alexis MICHEL

Subject : Mechatronics, robotics, embedded electronics, and software development

Key words : STM32, drone, RTOS, embedded system, AHRS, telemetry

Company : Upteko is a rapidly growing company, founded in 2018 by Mads Joergensen, Benjamin Mejnertz, and Sebastian Duus, with a mission to develop drone applications for the maritime sector. The company's work environment is diverse and dynamic, encompassing various fields such as hardware, software, simulation, fundraising, and professional events. Upteko's primary innovation is a multipurpose drone system designed for maritime use, capable of performing a wide range of tasks autonomously. This system includes a drone and a charging station, allowing for 100% automatic ship inspections in less than two hours, among other functionalities. The company's flexible work culture, which includes the possibility of remote work, contributes to a positive and productive work environment. Despite its rapid growth, Upteko maintains a close-knit team of experienced drone pilots, software and hardware engineers, and business developers across its offices in Copenhagen, Odense, and Skanderborg.

Work done : During my internship at Upteko, I had the opportunity to immerse myself in the world of drone technology through various projects. One of the main projects I worked on was the STM32 IMU project. The goal was to extract roll, pitch, and yaw data from the PModNav to create a 3D visualization of the IMU, which was crucial for tasks such as safety deployment and drone tracking. I managed to set up the STM32-L4796ZG to recover the sensor values from the PModNav and transmit them to the HMI. I also spent a significant amount of time migrating files in ROS2. Despite the complexity of the environment and the evolution of programming languages, I successfully migrated all the files. This task allowed me to learn valuable principles such as Quality Of Services and Micro Air Vehicle Link message communication. Additionally, I worked on a Software-In-The-Loop (SITL) project. This involved integrating TeraTowerEvo distance sensors into a simulation vehicle and parameterizing them for realistic operation. These projects not only deepened my understanding of drone technology but also allowed me to apply my academic knowledge to real-world problems. The experience was challenging but incredibly rewarding, and I am grateful for the opportunity.

Conclusion : My abroad internship provided me with a unique opportunity to apply my academic knowledge to real-world problems, particularly in the field of drone technology. Despite the occasional hurdles, the experience was invaluable, offering me a deeper understanding of the workings of a rapidly growing company and the complexities of working in a multifaceted environment. I am grateful for the skills I have acquired and the insights I have gained, and I look forward to applying these in my future career.

STUDENT SIGNATURE	SCHOOL SUPERVISOR SIGNATURE	STAMP OF THE COMPANY & SIGNATURE