Internship Report

Flying a Drone with Your Mind: FlexEEG Motor Imagery

Guilhem Merlet



Illustration from the EEG Drone Control Program GUI [1]

School	ENSTA Bretagne
Internship Institution	NTNU
Internship Period	04/05/2026 – 04/09/2026
ENSTA Supervisor	Luc Jaulin
NTNU Supervisor	Marta Molinas

ENSTA IP PARIS

NTNU

Abstract

This internship report presents the work carried out within the Department of Cybernetics at the Norwegian University of Science and Technology (NTNU), as part of the second year of the engineering programme at ENSTA. The placement builds on the master's thesis work of Daniel Nicolai Grønn, who developed an experimental platform for controlling Crazyflie 2.1 drones using a brain-computer interface (BCI) based on EEG motor imagery.

The main objective was to revisit this platform, rectify its malfunctions and improve its reliability, so that it could serve as a solid foundation for evaluating BCI control strategies. This work involved restoring the Loco Positioning System (LPS), resolving a critical compatibility issue between the anchor communication modes, and then fine-tuning the PID controller parameters to stabilise the drones' flight, both in altitude and in the horizontal plane. A rigorous experimental approach, based on the analysis of real flight data, enabled the identification of settings offering the best compromise between convergence speed, stability and positioning accuracy.

Finally, comprehensive documentation (User Guide Version 3.0) and a practical demonstration were produced to facilitate the transfer of this knowledge to future interns, who will continue this project by incorporating more advanced EEG classification methods.

Résumé

Ce rapport de stage présente les travaux réalisés au sein du département de cybernétique de la Norwegian University of Science and Technology (NTNU), dans le cadre de la deuxième année du cursus ingénieur à ENSTA. Le stage s'inscrit dans la continuité du travail de master de Daniel Nicolai Grønn, qui a développé une plateforme expérimentale permettant de piloter des drones Crazyflie 2.1 à l'aide d'une interface cerveau-machine (BCI) fondée sur l'imagerie motrice EEG.

L'objectif principal était de reprendre en main cette plateforme, d'en corriger les dysfonctionnements et d'en améliorer la fiabilité, afin qu'elle puisse servir de base solide à l'évaluation de stratégies de contrôle par BCI. Ce travail a consisté à restaurer le système de positionnement local (Loco Positioning System), à résoudre un problème critique de compatibilité entre les modes de communication des ancres, puis à affiner les paramètres du correcteur PID afin de stabiliser le vol des drones, tant en altitude que dans le plan horizontal. Une démarche expérimentale rigoureuse, basée sur l'analyse de données de vol réelles, a permis d'identifier des réglages offrant le meilleur compromis entre rapidité de convergence, stabilité et précision de positionnement.

Enfin, une documentation complète (guide utilisateur version 3.0) et une démonstration pratique ont été réalisées afin de faciliter la transmission de ces connaissances aux prochains stagiaires, qui poursuivront ce projet en intégrant des méthodes de classification EEG plus avancées.

Contents

Abstract i

Résumé ii

1 Introduction 1
- 1.1 Presentation of NTNU and the Department of Engineering Cybernetics . . . 1
- 1.2 Internship Topic . . . 1
- 1.3 Internship Objectives . . . 2

2 Theory 3
- 2.1 Brain-Computer Interfaces (BCI) . . . 3
  - 2.1.1 Definition and Principle . . . 3
  - 2.1.2 EEG as a Signal Acquisition Method . . . 3
  - 2.1.3 Motor Imagery . . . 3
- 2.2 Micro Quadcopters . . . 4
- 2.3 PID Controller . . . 5
  - 2.3.1 Effect of the Proportional Gain . . . 5
  - 2.3.2 Effect of the Integral Gain . . . 5
  - 2.3.3 Effect of the Derivative Gain . . . 5

3 System Setup and Initial Validation 6
- 3.1 Graphical User Interface (GUI) . . . 6
  - 3.1.1 What is a GUI? . . . 6
  - 3.1.2 Shell script update . . . 6
  - 3.1.3 GUI Structure . . . 7
- 3.2 Loco Positioning System . . . 8
  - 3.2.1 What is the Loco Positioning System? . . . 8
  - 3.2.2 Repair of the Node Holders . . . 8
  - 3.2.3 Positioning of the Stand Nodes . . . 9
  - 3.2.4 Resolving the issue with anchor detection . . . 9
- 3.3 Initial Test Result . . . 11

4 Improvement of the PID controller 12
- 4.1 Fine-tuning the PID controller at altitude . . . 12
  - 4.1.1 Methodology for Altitude PID Tuning . . . 12

4.1.2 Insufficient Thrust: Battery Discharge and Mechanical Issues 12
4.1.3 Results 13
4.2 Fine-tuning the PID controller in (x, y) 15
4.2.1 Methodology for PID Tuning in (x, y) 15
4.2.2 Results 16

5 Documentation and Knowledge Transfer **19**
5.1 Updating the User Guide 19
5.2 Demonstrating the real system 19
5.3 Relevance for future work 20

6 Conclusion **21**
6.1 Conclusions on the work carried out 21
6.2 Personal conclusion 22

Bibliography **23**

A Internship Assessment Report **24**

Chapter 1

Introduction

1.1 Presentation of NTNU and the Department of Engineering Cybernetics

The Norwegian University of Science and Technology (NTNU), located in Trondheim, is Norway's largest university in the fields of science, engineering, and technology. Founded in 1996, NTNU plays a major role in Norwegian research and innovation [2]. Among NTNU's leading research departments are:

- The Department of Engineering Cybernetics, which focuses on automation, robotics, control systems, and autonomous technologies.
- The Department of Computer Science, which conducts research in artificial intelligence, cybersecurity, software engineering, and data science.
- The Department of Marine Technology, one of NTNU's flagship departments, whose research covers ships, offshore structures, marine energy, and autonomous underwater vehicles.

I completed my internship in the Department of Engineering Cybernetics.

1.2 Internship Topic

When I joined the Department of Engineering Cybernetics at NTNU under the supervision of Marta Molinas, I was introduced to research in the field of electroencephalography (EEG). The projects supervised by Marta Molinas are based on the use of OpenBCI EEG headsets [3], which record the electrical activity of the brain in order to develop innovative applications, such as robot control or dream analysis.

Among these projects, I chose to work on the following topic: "Flying a Drone with Your Mind: FlexEEG Motor Imagery."

Brain-computer interfaces (BCIs) are technologies that aim to translate brain activity into commands for external devices. One of the main approaches used in BCI research is Motor Imagery (MI), in which users imagine performing a movement, such as moving a hand, without actually moving it. This mental activity generates specific EEG patterns that can be detected

and classified by algorithms. These patterns can then be converted into commands for controlling external systems.

The objective of this project is to demonstrate the potential of EEG-based Motor Imagery for drone swarm control.

1.3 Internship Objectives

This internship is a continuation of the work presented in Daniel Nicolai Grønn's Master's thesis [1]. In this thesis, Daniel Nicolai Grønn presents the development of a drone flight environment that can be used with EEG-based Brain-Computer Interface (BCI) technologies. His main objective was to demonstrate, using a real-time application, a research platform for evaluating BCI-based control strategies. This work resulted in the creation of a comprehensive experimental platform based on:

- a hardware infrastructure comprising Crazyflie 2.1 drones and the Loco Positioning System;
- a modular software architecture facilitating future developments;
- graphical interfaces enabling trajectory planning and collision risk detection;
- a design flexible enough to be reused and adapted for new research projects.

Nevertheless, several improvements were identified, particularly the tuning of the controller and Kalman filter parameters to improve swarm stability.

My internship objectives were:

- to understand the concepts and challenges associated with BCI and motor imagery using EEG;
- to take over and understand the work carried out by Daniel Nicolai Grønn by ensuring the proper functioning of the experimental platform, resolving software dependency issues, and restoring the anchor-based positioning system;
- to improve the flight stability of single and multiple drone configurations;
- to connect a new EEG headset to the platform and analyse the results obtained.

Chapter 2

Theory

2.1 Brain-Computer Interfaces (BCI)

2.1.1 Definition and Principle

A brain-computer interface (BCI) is a system that enables communication between the brain and an external device, without requiring any muscular output [4]. This technology has recently been used in clinical practice, for example in patient rehabilitation. There are different types of BCIs (active, reactive, and passive systems) [5], depending on how the user's brain activity is triggered and used. Reactive BCIs, such as those based on the P300 signal or SSVEP, depend on brain responses caused by external stimuli, like a flashing light or a sound. Active BCIs, on the other hand, are controlled by the user's own voluntary brain activity, without needing any outside stimulus.

2.1.2 EEG as a Signal Acquisition Method

Electroencephalography (EEG) is the most common choice for BCI applications because it is non-invasive, safe, portable, relatively inexpensive, and offers a high temporal resolution compared to other brain-imaging methods. EEG records the summed electrical activity of large populations of cortical neurons at the scalp; pyramidal neurons contribute most strongly to the recorded signal because their electric fields are stably oriented perpendicular to the cortical surface, unlike other neuron types whose fields tend to cancel out. Because the recording electrodes are located outside the skull, only the synchronized activity of large neural ensembles produces a signal strong enough to be detected, which limits the spatial precision achievable with EEG relative to invasive recording techniques.

2.1.3 Motor Imagery

One of the most important paradigms in BCI research is Motor Imagery (MI). MI refers to the brain activity generated when a person mentally rehearses a movement, without actually moving any muscles. This happens because the brain uses similar neural pathways to plan a movement and to actually perform it. As a result, simply imagining a movement produces changes in sensorimotor brain rhythms that are similar to, but usually weaker than, those produced by performing the movement itself. These changes are typically seen as event-related desynchronisation

(ERD) or synchronisation (ERS) in the mu (8–13 Hz) and beta (13–30 Hz) frequency bands, in the sensorimotor cortex [6, 7].

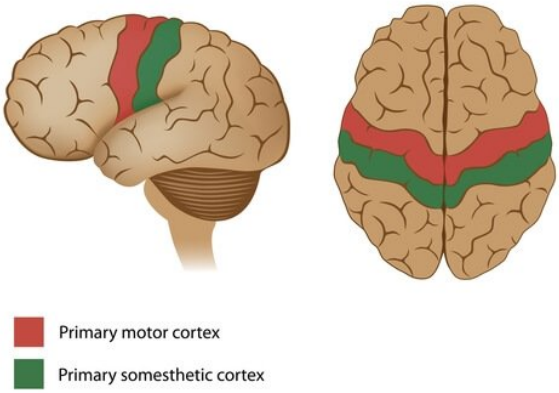


Figure 2.1: Primary motor and somatosensory cortices involved in motor imagery

2.2 Micro Quadcopters

The drones used in this project are Bitcraze Crazyflie 2.1 micro-quadcopters [8], designed for education and research. Each drone weighs only a few tens of grams and is equipped with four brushed motors and a central flight control board. Their main advantages are their small size, low cost, and safety, which make them well suited to indoor experiments.

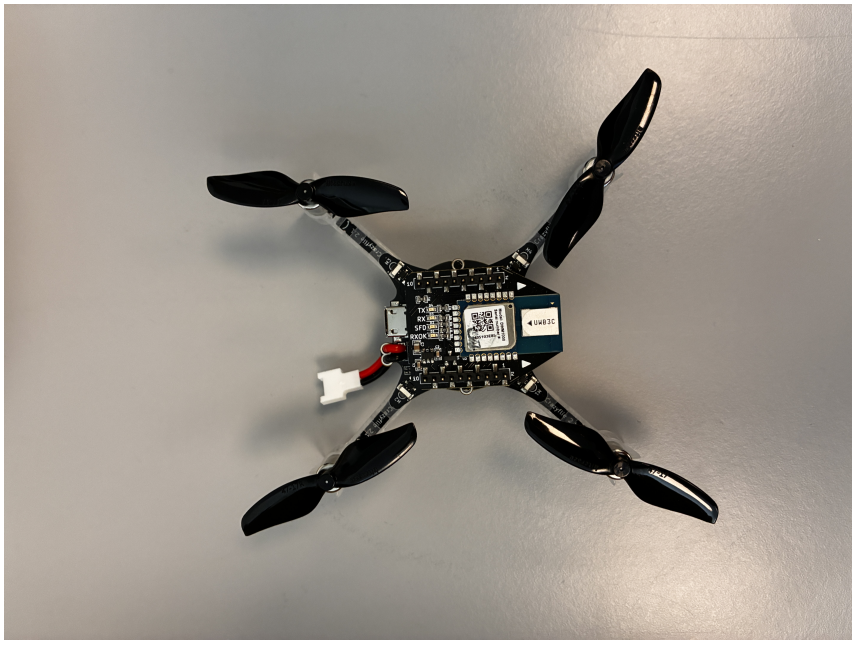


Figure 2.2: Crazyflie 2.1 micro-quadcopter

Furthermore, the Crazyflie ecosystem is fully open source: its firmware, client software and libraries provide a range of localisation and control tools, which makes it possible to prototype complete unmanned aerial vehicle (UAV) applications.

2.3 PID Controller

A Proportional–Integral–Derivative (PID) controller is one of the most widely used control algorithms in industrial automation [9]. Its purpose is to minimize the error between a desired value, called the *setpoint*, and the measured output of a system. The controller continuously computes the error and adjusts the control input using three different actions: proportional, integral, and derivative.

The control law of a PID controller is expressed as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}, \tag{2.1}$$

where:

- $u(t)$ is the control signal,
- $e(t)$ is the error between the setpoint and the measured output,
- K_p is the proportional gain,
- K_i is the integral gain,
- K_d is the derivative gain.

2.3.1 Effect of the Proportional Gain

The proportional term produces an output proportional to the current error. Increasing the proportional gain K_p makes the system respond faster by reducing the rise time. However, if K_p is too high, the system may exhibit overshoot and oscillations, which can reduce its stability.

2.3.2 Effect of the Integral Gain

The integral term accumulates the error over time. Its main purpose is to eliminate the steady-state error that remains after the proportional action. Increasing K_i improves the accuracy of the system in steady state. However, an excessively large integral gain can increase overshoot, lengthen the settling time, and even cause instability due to integral windup.

2.3.3 Effect of the Derivative Gain

The derivative term depends on the rate of change of the error. It predicts the future behavior of the system and provides a damping effect. Increasing K_d generally reduces overshoot and shortens the settling time, improving the overall stability of the system. However, a large derivative gain amplifies measurement noise and may produce an irregular control signal.

A well-tuned PID controller achieves a compromise between fast response, low overshoot, short settling time, and high stability.

Chapter 3

System Setup and Initial Validation

3.1 Graphical User Interface (GUI)

3.1.1 What is a GUI?

A Graphical User Interface (GUI) is an interface that allows users to interact with software through visual elements instead of using only command lines. It usually includes elements such as windows, buttons, menus, and graphical displays. The main objective of a GUI is to make the interaction with a program easier and more intuitive for the user.

In this project, the GUI is used to visualize information and interact with the drone control system. It allows the user to monitor data, configure parameters, and launch different functions of the software without directly modifying the source code or using terminal commands.

3.1.2 Shell script update

In this part of the project, the software developed by Daniel Nicolai Grønn [1] was downloaded from GitHub. To simplify the installation process, the author provided a shell script (.sh) that automatically performs the main configuration steps. This script updates the system, installs the required dependencies, creates and configures a Python virtual environment, installs the project libraries, and displays the commands needed to launch the application.

However, despite this automation, the project could not be directly executed on the development machine. The Python environment required by the project was not compatible with the Python version installed on the system (Python 3.12), which prevented the software from running.

The origin of this issue was related to several dependencies that had not been updated since the creation of the project. This caused compatibility problems during the automatic installation process. Six different problems were identified and fixed: an unpinned version of NumPy, a conflict between cfclient and NumPy, missing graphical interface dependencies required by cfclient, the unavailability of the pkg_resources module in recent versions of setuptools, an incompatibility between vispy and NumPy 2.x, and a compatibility issue between cfclient and a recent version of cflib.

All these corrections were integrated into the project's requirements.txt file. As a result, the installation script can now automatically create a functional and reproducible environment

without requiring manual modifications from the user.

3.1.3 GUI Structure

The interface is divided into several sections that correspond to the main steps of using the software. The first section provides access to the project documentation through the README file, which explains the general operation of the application.

The second section is dedicated to the creation of an EEG recognition model. It includes the functions required to record a dataset, train a machine learning model, and evaluate its performance on a test dataset.

The interface also includes a module for controlling a single drone. This module allows the user to launch the cfclient software, which is used to configure the drone and monitor its status, as well as a demonstration mode that can be used to test the system with or without a connected drone.

Another section is dedicated to swarm control. The user can define a flight path using a graphical editor and then launch a simulation interface to visualize and execute the swarm's movements.

Finally, the last module allows the user to run any Python script from the project directly through a file explorer. This feature makes development, testing, and debugging easier.

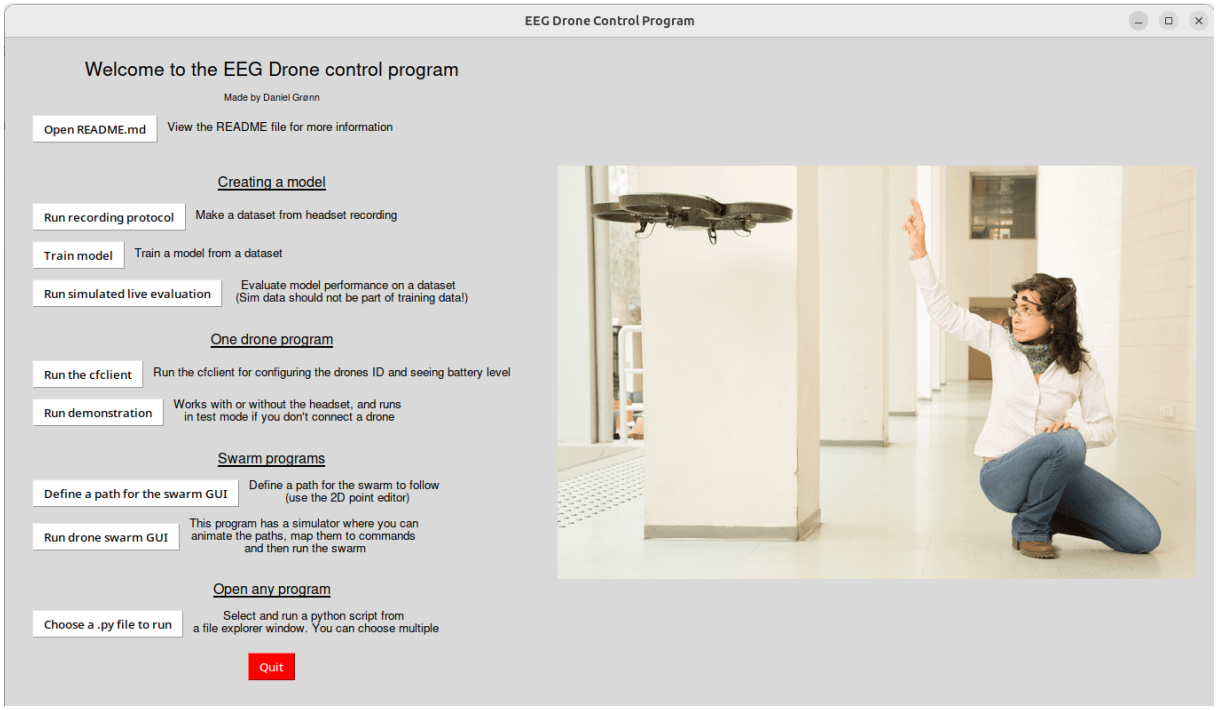


Figure 3.1: Graphical user interface home page

For the work carried out, the GUI features I used most were "Run the cfclient" to check the drones' battery levels and adjust the drone settings, "Define a path for the swarm GUI", which allowed me to define the waypoints the drone flies through, and finally "Run drone swarm GUI", which allowed me to run the test.

3.2 Loco Positioning System

3.2.1 What is the Loco Positioning System?

A Loco Positioning System (LPS) is an indoor positioning system developed by Bitcraze [10] for drones such as the Crazyflie. It is a widely used solution for indoor drone applications, where GPS signals are not available.

The system is based on Ultra-Wideband (UWB) technology. Several fixed devices, called anchors, are placed at known positions in the room, usually between four and eight. The drone is equipped with a UWB tag that communicates with these anchors.

The system measures the travel time of radio signals between the tag and each anchor. Depending on the positioning mode, it uses either Two-Way Ranging (TWR) or Time Difference of Arrival (TDoA) to estimate the distance between the drone and the anchors. By combining these distances through trilateration, the system calculates the three-dimensional position of the drone in real time, with an accuracy of a few centimeters.

Finally, the estimated position is combined with data from the drone's onboard Inertial Measurement Unit (IMU), which includes an accelerometer and a gyroscope. A filtering algorithm, usually a Kalman filter, merges these measurements to provide a stable and accurate estimate of the drone's state. This information is then used for precise position control during flight.

3.2.2 Repair of the Node Holders

During my first visit to the laboratory, I observed that three of the node holders were damaged and could therefore no longer be used to mount the anchors.



Figure 3.2: The three damaged anchor holders

In order to address this issue, I selected one of the node holders that was still in good condition and used it as a reference to redesign the part using Inventor. By carefully measuring its dimensions, I was able to reconstruct a version nearly identical to the original design created by Daniel Nicolai Grønn. Once the CAD model was completed, the part was 3D printed using the printers available at NTNU.

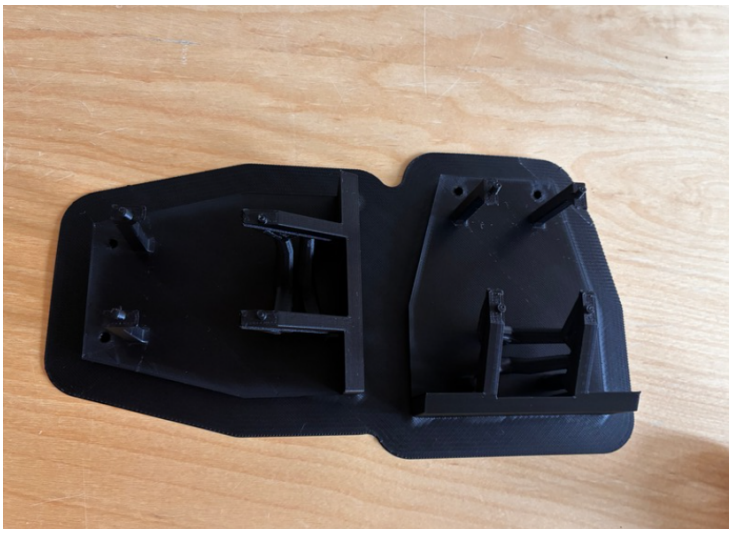


Figure 3.3: New anchor holders 3D-printed at NTNU

3.2.3 Positioning of the Stand Nodes

To ensure accurate and repeatable positioning of the anchors, I marked the exact location of each node stand directly on the floor of the laboratory. The system supports up to 8 anchors, so these markings allow each stand to be placed at the same position every time, which is essential for maintaining the accuracy of the Loco Positioning System, since the anchor coordinates used in the localization algorithm must correspond precisely to their real position in the room.

3.2.4 Resolving the issue with anchor detection

I then did my first tests to make the drone follow a trajectory, but each time, the drone moved in a strange and uncontrolled way instead of following the path. The trajectory was intended to allow the drone to fly upwards to the point (0, 0, 2) and then move along the x-axis to (1, 0, 2), as shown in Figure 3.4.

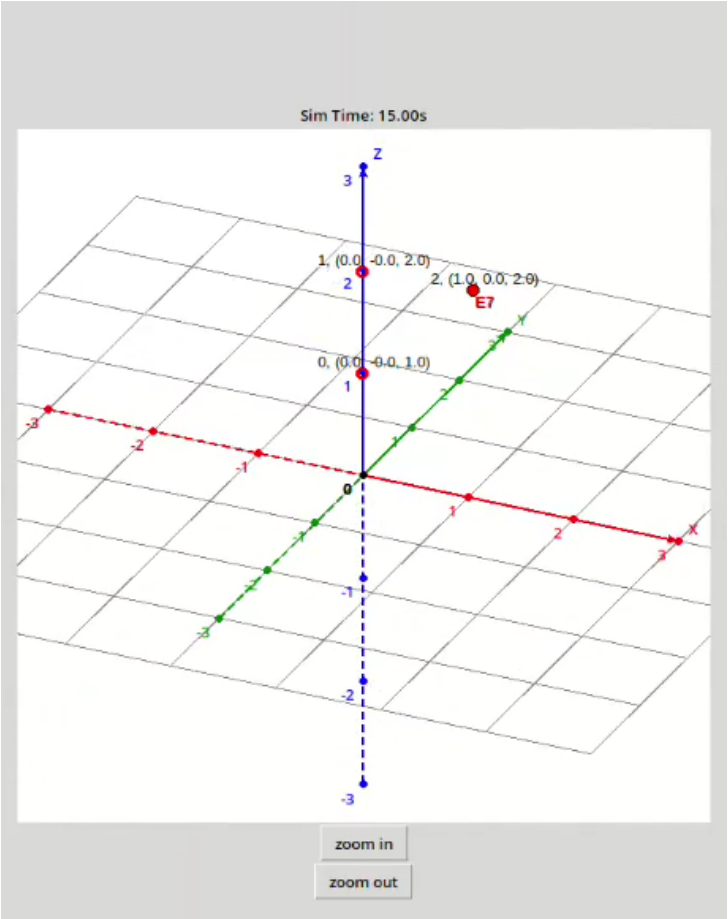


Figure 3.4: First flight path to test the system

After doing a lot of research to understand why this was happening, I found that the drone was not detecting the anchors at all. The problem was that the drone was set to TWR (Two-Way Ranging) mode, while the anchors were set to TDoA2 (Time Difference of Arrival) mode. Because these two modes do not work together, the drone could not receive correct position data from the anchors, which explains why it was flying incorrectly during the tests. Once the drone was switched to the same mode as the anchors, they were correctly detected (Figure 3.5).

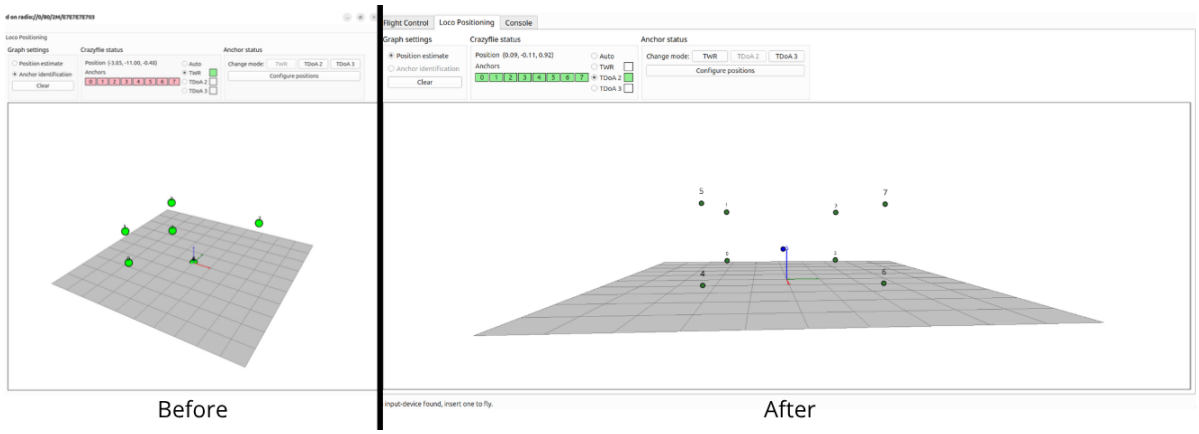


Figure 3.5: Anchors not detected / detected

3.3 Initial Test Result

After solving the setup problems and the issues related to the anchors, I ran the same trajectory test again, and this time it was successful. The drone followed the planned path correctly, which showed that the Loco Positioning System was now working properly. I filmed this successful test and uploaded the video to YouTube as proof of this result: click here to watch the video[1].

This first part of my internship therefore allowed me to fix the initial problems with the laboratory setup and to obtain a working positioning system, which was an important step before moving on to more advanced tests.

Now that the positioning system was working correctly, the next step was to focus on tuning the PID controller in order to improve the stability of the drone during flight.

[1] https://youtube.com/shorts/3qZQ36JsB9g

Chapter 4

Improvement of the PID controller

Although the drone was now able to follow the planned trajectory, its flight was not completely stable and some oscillations could still be observed during the tests. This lack of stability comes from the PID controller, which is responsible for correcting the drone's position and orientation in real time based on the data received from the Loco Positioning System. In order to improve the overall performance and precision of the flight, it was therefore necessary to work on tuning the PID controller, by adjusting its parameters to reduce oscillations and make the drone react in a more stable way.

4.1 Fine-tuning the PID controller at altitude

4.1.1 Methodology for Altitude PID Tuning

To tune the PID coefficients for altitude control in a rigorous and systematic way, I used the "take off" function available in the cfclient software [11]. This function makes the drone fly in a stationary hover at an altitude of 0.5 m, regardless of its (x, y) position, which made it possible to focus specifically on the altitude behavior of the drone without being influenced by horizontal movements. During each flight, I collected several types of data in real time, including the position estimated by the Kalman filter, the battery state, and the levels of the motors and the thrust. This data collection allowed me to precisely observe how the drone reacted for a given set of PID parameters, for example by identifying oscillations, overshoot, or a slow response time in the altitude estimation. By analyzing this data after each test, I was able to progressively adjust the PID coefficients in a methodical way, testing one parameter at a time in order to clearly understand its individual effect on the stability of the flight. This structured approach, based on real flight data rather than visual observation alone, made it possible to tune the controller more accurately and to obtain more reliable and repeatable results.

4.1.2 Insufficient Thrust: Battery Discharge and Mechanical Issues

At the beginning of the PID tuning process, I worked with Crazyflie drone number 3. The hover flights were never consistent, and the drone kept crashing at different moments during the flight, even though it was supposed to stay at a stable altitude of 0.5 m. To understand this problem, I collected and analyzed the flight data. Several interesting observations came out

of this analysis. First, the battery voltage shows a large initial drop at the start of the flight, regardless of whether the battery is fully charged or has already been used. This voltage drop is caused by the internal resistance of the battery. After this initial drop, the voltage continues to decrease, but at a slower and more constant rate, and the drone tends to crash once voltage drops into the low 3.2–3.3V range. The data also shows that the motor and thrust values remain consistently higher throughout the flight compared to a healthy drone, and reach saturation only in the final moments, coinciding with the voltage bottoming out, which explains why the crash always occurs around that same point.

At first, I assumed that this problem was simply due to the batteries discharging too quickly. However, after testing a different drone with the same type of battery, I observed that it was able to maintain the required altitude for a longer period of time. This suggests that the battery discharge is not the only factor involved, and that it is also related to the mechanical efficiency of the drone, meaning its ability to generate optimal lift. Drone number 7 showed a better lift efficiency, so I used it for the rest of the tuning process. Figure 4.1 compares the battery voltage and the maximum motor thrust of the two drones.

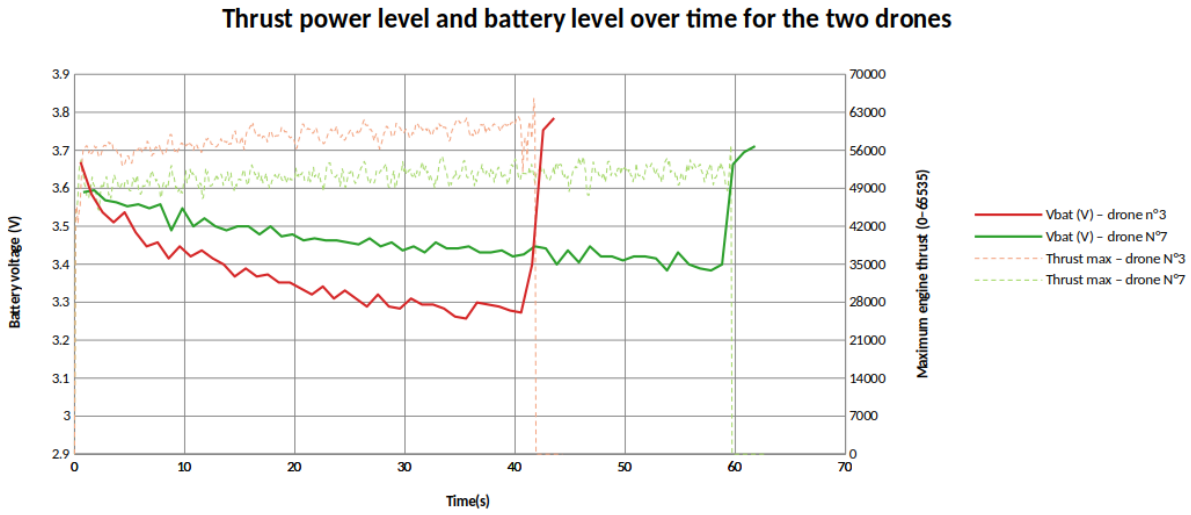


Figure 4.1: Thrust power level and battery level over time for the two drones

4.1.3 Results

Here are the results of the various tests carried out to fine-tune the PID controller's parameters. The altitude responses are shown in Figure 4.2 and the mean altitudes in Figure 4.3.

Test R1 (kpZ=2, kiZ=0.5) — Initial setting: pure PI controller without a derivative term. The flight reached an altitude close to the setpoint (0.5 m) but exhibited persistent oscillation in steady state, which did not dampen over time.

Test R2 (kpZ=2, kiZ=0.3, kdZ=0.5) — Hypothesis tested: the oscillation stems from excessive integral action. The kiZ was therefore reduced and an initial derivative term introduced to dampen the response. Result: the amplitude of the oscillation remains virtually identical to that in R1, which invalidates the hypothesis of a simple excess of kiZ and suggests that the phenomenon has another cause (measurement or system dynamics).

Test R3 (kpZ=2, kiZ=0, kdZ=1.0) — A deliberately extreme test to reach a definitive conclusion: complete removal of the integral action and doubling of the derivative gain, in order to isolate the contribution of the corrector to the observed oscillation. As the engine thrust oscillated in phase with the measured altitude, the origin of the oscillation in the controller was confirmed. The oscillation does indeed diminish, but at the cost of a significant static error: the drone stabilises well below the setpoint, behaviour expected of a pure PD corrector which cannot cancel out a permanent deviation between commanded thrust and required thrust.

Test R4 (kpZ=2, kiZ=0.1, kdZ=1.0) — Reintroduction of a low kiZ value to correct the static error from R3, whilst maintaining kdZ=1.0. However, this flight was carried out with a battery that was already significantly discharged (3.79 V → 3.18 V), leading to saturation of the motor controls. The result (altitude remaining low) is therefore not useful for assessing the settings: it reflects a physical thrust limit, not an effect of the gains.

Test R5 (kpZ=2, kiZ=0.1, kdZ=1.0) — A repeat of R4 under identical conditions with a fully charged battery, in order to isolate the actual effect of the tuning from that of the battery. Result: altitude stabilised at 0.48 m, with moderate overshoot.

Test R6 (kpZ=2, kiZ=0.15, kdZ=1.2) — An attempt at further refinement based on R5. Smaller oscillations than in R5, with an average value of 0.5 m, which is the best result obtained. These settings were therefore kept for the rest of the internship.

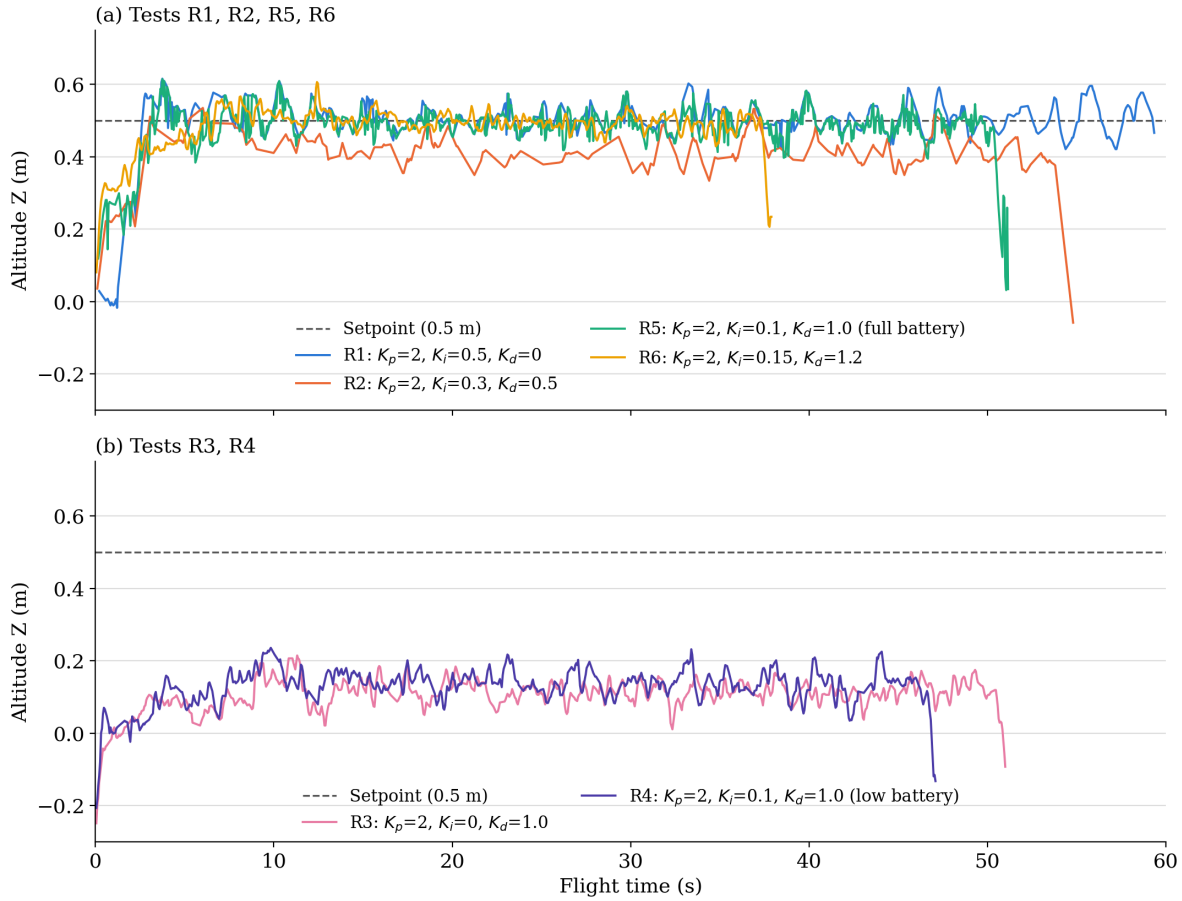


Figure 4.2: Altitude response $Z(t)$ for the different controller parameters (setpoint 0.5 m)

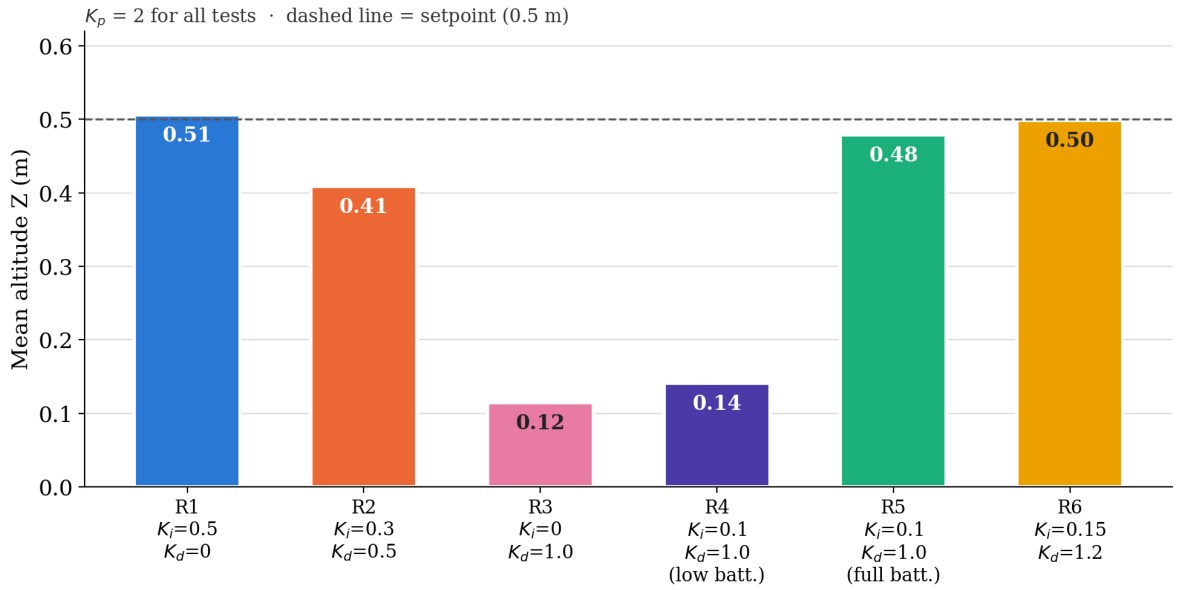


Figure 4.3: Mean altitude Z in steady state for each test

4.2 Fine-tuning the PID controller in (x, y)

4.2.1 Methodology for PID Tuning in (x, y)

The objective of this work was to evaluate the performance of the (x, y) position controller when regulating the drone to a fixed reference position (0, 0).

The reference point was selected directly on the graphical user interface (GUI). By clicking on the workspace, the desired target position could be defined before launching the controller.

Figure 4.4 presents the GUI used to select the reference point.

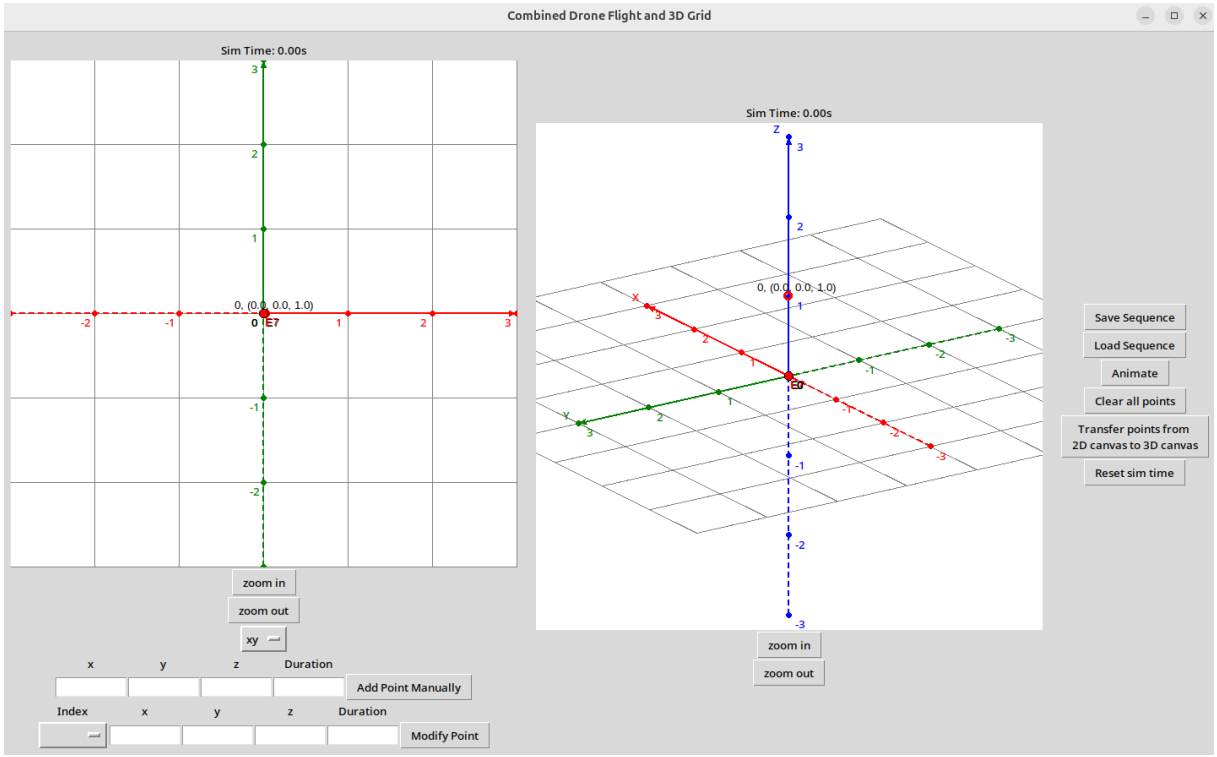


Figure 4.4: Define the waypoint (0, 0)

The tuning of the controller was carried out using an iterative approach. After each experiment, the drone behaviour was analysed, and the controller parameters were adjusted according to the observed response. The objective was to obtain a stable regulation with a good compromise between convergence speed and positioning accuracy.

The proportional gain was kept constant during the tuning process, while the derivative gain was progressively modified to evaluate its influence on the dynamic response of the controller. For each iteration, the drone was commanded to move towards the reference point, and its position was recorded over time.

To compare the controller performance, two indicators were calculated:

- the mean control error, corresponding to the average distance from the reference;
- the Root Mean Square (RMS) control error, which gives more weight to larger errors and provides a better evaluation of the overall regulation quality.

The recorded trajectories and the calculated errors were then compared to identify the most suitable controller parameters.

4.2.2 Results

The proportional gain was fixed at Kp = 2 throughout all experiments. This value was selected because it provided a good balance between responsiveness and stability. Lower proportional gains resulted in a slower convergence towards the reference, while higher values increased oscillations and made the drone behaviour less stable. Therefore, Kp = 2 was considered an appropriate compromise and was kept constant to isolate the effect of the derivative gain.

The integral gain (Ki) was intentionally set to 0. Since the objective of the controller was to regulate the drone to a fixed position rather than continuously track a moving reference, no significant steady-state error was observed that required integral compensation. Moreover, introducing an integral term could lead to integral windup, increasing overshoot and slowing the stabilization process. For these reasons, only a PD controller was used.

The derivative gain was then progressively increased from Kd = 0 to Kd = 0.3 and finally to Kd = 0.5. The purpose of increasing Kd was to analyse how additional damping affected the drone dynamics. A small derivative gain is expected to reduce oscillations while maintaining a fast response, whereas a larger derivative gain may over-damp the system and reduce its responsiveness.

Figure 4.5 compares the convergence of the drone towards the reference position for the three tested values of Kd.

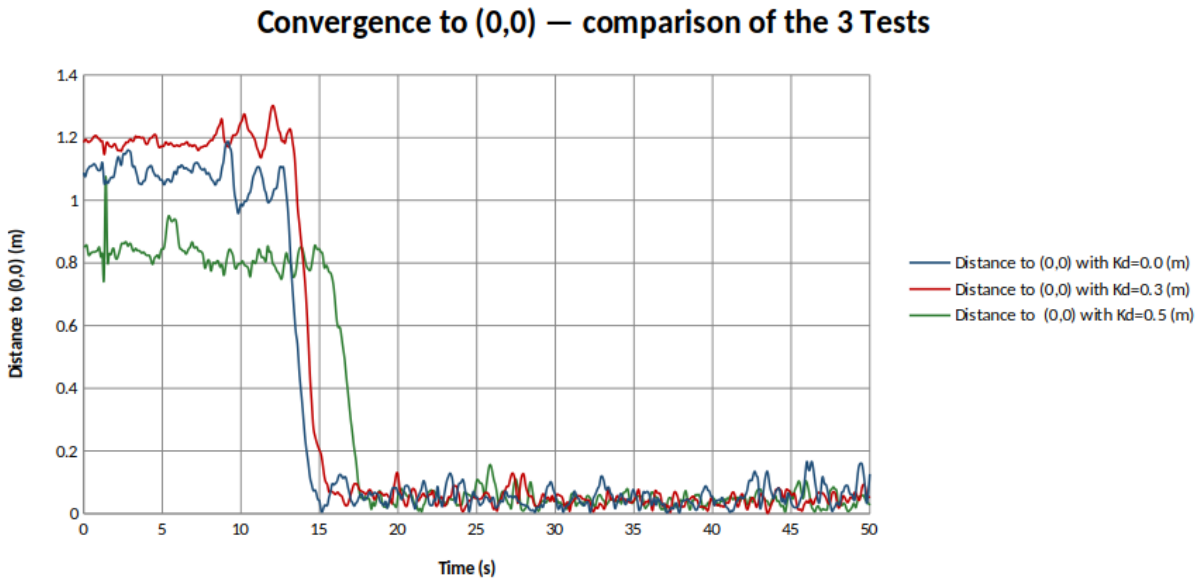


Figure 4.5: Convergence to (0, 0): comparison of the three tests

When Kd = 0, the drone reached the target in approximately 15 seconds, making it the fastest configuration. However, the response presented noticeable oscillations after convergence, showing that the controller lacked damping. These oscillations explain the relatively large control errors measured during the experiment.

Increasing the derivative gain to Kd = 0.3 significantly improved the regulation quality. Although the convergence time increased slightly, the oscillations were strongly reduced and the drone stabilized smoothly around the reference point. This configuration produced the most stable behaviour while maintaining a satisfactory response time.

When the derivative gain was further increased to Kd = 0.5, the controller became more damped. As a result, the drone required approximately 18 seconds to reach the target. While the response remained stable, the stronger damping reduced the system responsiveness, leading to a slower convergence without providing a significant improvement in the final positioning accuracy.

Figure 4.6 compares the mean control error and the RMS control error obtained for the three

derivative gains.

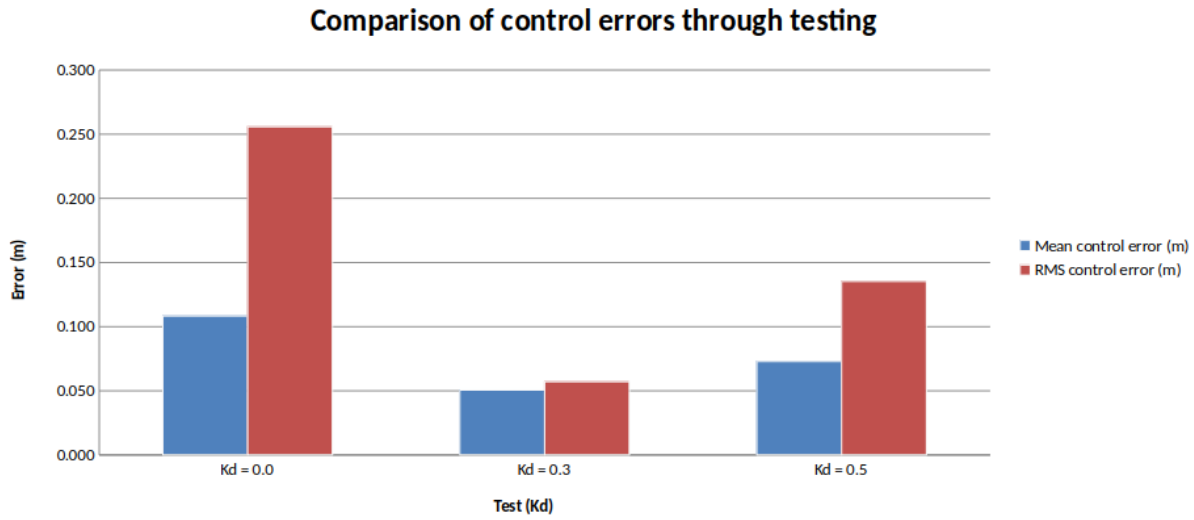


Figure 4.6: Comparison of control errors through testing

The quantitative results confirm the observations made from the trajectories. The controller with Kd = 0.3 achieved the lowest mean error (approximately 0.05 m) and the lowest RMS error (approximately 0.06 m), indicating the best overall regulation performance.

For Kd = 0, the mean error was approximately 0.11 m, while the RMS error reached 0.26 m. The large difference between these two indicators reflects the presence of important oscillations during the experiment.

With Kd = 0.5, the controller remained stable but the errors increased compared with Kd = 0.3. The mean error was around 0.07 m, and the RMS error reached approximately 0.13 m. These results show that excessive derivative action slows down the response without improving the regulation accuracy.

Overall, the experiments demonstrate that Kp = 2 and Kd = 0.3 provide the best compromise between convergence speed, stability and positioning accuracy for the regulation task. The derivative action introduces sufficient damping to reduce oscillations while preserving a fast convergence towards the reference position, making this configuration the most suitable among the tested values.

Chapter 5

Documentation and Knowledge Transfer

A significant part of my internship consisted of making the EEG-drone framework easier to use and easier to hand over to the interns who will continue the project after my departure. This involved two main contributions: writing Version 3.0 of the User Guide, and preparing a practical demonstration of the real system for the next team.

5.1 Updating the User Guide

I wrote Version 3.0 of the User Guide for EEG Drone Swarm Control at BrainKybLab, which focuses on three points that were missing or incomplete in the previous version. First, I added a full installation procedure for the software on Linux, including the driver setup, the required udev rules for the Crazyradio dongle and the Loco Positioning nodes, and the Python environment setup through the provided shell script. This was not documented before, even though several future users are expected to work on Linux machines. Second, I documented the recurring issues with anchor detection in the Loco Positioning System, in particular the mismatch between the TWR and TDoA2/TDoA3 modes, which can silently prevent the anchors from being detected and cause unexplained drone behavior. Third, I added practical advice on using the cfclient software: how to connect a drone by radio address, how to log flight parameters (battery, motor thrust, Kalman position estimate) directly to a file, and how to adjust PID controller settings from the Parameters window. These additions are meant to reduce the time future interns spend debugging setup problems that I already solved, so they can focus directly on the scientific and technical goals of the project.

5.2 Demonstrating the real system

Beyond the written documentation, I prepared a demonstration to show the incoming interns how the system actually behaves in practice, not only how it is supposed to work on paper. It consists of a live demonstration of a Crazyflie drone flying under position control with the Loco Positioning System, showing the calibration steps I performed (PID tuning on the Z axis and on the X/Y axes), the flight logs I collected, and the failure modes I encountered (Kalman estimator

getting stuck before takeoff, drone instability with poorly tuned gains, dependency conflicts on Ubuntu 24.04). Seeing the drone fly and being walked through a real debugging session gives the next interns a faster and more concrete understanding of the framework than the documentation alone can provide.

5.3 Relevance for future work

This handover is particularly important for the interns who will work on integrating improved EEG classification methods with real-time drone control. Their work depends directly on the stability of the underlying flight and positioning system: if the drone's position control or the anchor detection is not reliable, it becomes very difficult to evaluate whether a new classifier is actually improving performance, or whether unrelated flight issues are causing the observed errors. By documenting the setup, the known bugs, and the tuning process, and by demonstrating the working system directly, I have given them a stable and well-understood baseline so they can concentrate on the classification and real-time integration challenges rather than on re-discovering the infrastructure problems I already resolved.

Chapter 6

Conclusion

6.1 Conclusions on the work carried out

This internship set out to take over and improve the experimental BCI-drone platform initially developed by Daniel Nicolai Grønn, with four main objectives: understanding the concepts of BCI and motor imagery, restoring the experimental platform to a working state, improving the flight stability of the drones, and connecting a new EEG headset to the platform.

The first part of the work consisted of making the platform operational again. This required resolving a cascade of software dependency conflicts that prevented the project from running on an up-to-date Python environment, repairing the physical infrastructure of the Loco Positioning System (redesigning and 3D-printing damaged anchor holders), and diagnosing a critical incompatibility between the drone's ranging mode (TWR) and the anchors' mode (TDoA2), which had been silently preventing any correct positioning. Once these issues were fixed, the platform was validated through a successful trajectory-following test, confirming that the Loco Positioning System was functioning correctly again.

The second part focused on improving flight stability through a rigorous, data-driven tuning of the PID controller. On the altitude axis, several configurations were tested and compared using real flight logs, which also revealed that flight inconsistencies were not only caused by battery discharge but also by differences in mechanical efficiency between drones. The best compromise was obtained with kpZ=2, kiZ=0.15, kdZ=1.2. On the horizontal (x, y) plane, a similar iterative approach identified kp=2, kd=0.3 as the setting offering the best trade-off between convergence speed, stability, and positioning accuracy, based on mean and RMS control error comparisons.

Finally, in order to ensure the continuity of the project, this work was documented in a new version (3.0) of the User Guide, including a full Linux installation procedure, a description of the anchor-detection issue and its resolution, and practical guidance on using the cfclient software. A live demonstration of the working system was also prepared for the incoming interns.

Some of the initial objectives could not be completed within the internship. Restoring the platform and obtaining a stable flight with a single drone took most of the available time, so the connection of the new EEG headset, flight tests with several drones and the tuning of the Kalman filter parameters were not carried out. These points remain the priorities for the next interns, who can now build on a working and documented platform.

Overall, the internship achieved its main goal: transforming a partially non-functional exper-

imental platform into a stable, well-documented base for BCI-driven drone control. The natural continuation of this work, as identified from the outset, is the integration of more advanced EEG motor-imagery classification methods, which can now be developed on top of a reliable and properly characterized flight platform.

6.2 Personal conclusion

To conclude this report, here is an analysis and a reflection on the work carried out and on my overall experience within the Cybernetics Department at NTNU. By joining NTNU, I discovered a university research environment as well as a different facet of the engineering profession, one more focused on experimentation and exploration than on production.

By taking over and getting to grips with the work already carried out by Daniel Nicolai Grønn, I was able to position myself as an engineer capable of understanding an existing system, diagnosing its weaknesses, and then proposing concrete improvements, whether on restoring the hardware platform or on fine-tuning the control laws.

From a technical standpoint, I experienced getting a full software environment running again (resolving dependency conflicts, repairing 3D-printed mechanical parts, reconfiguring an ultra-wideband positioning system), as well as the methodical tuning of a PID controller based on real flight data. While my engineering education had already taught me the importance of a rigorous methodology for obtaining reliable results, this internship showed me that such rigor is just as essential in research, where one must know how to isolate a single variable at a time in order not to confuse the effect of a tuning parameter with that of an external factor, as I observed during my tuning tests disrupted by battery charge level.

This internship allowed me to confirm my interest in robotics and control systems, while discovering a new field of application, brain-computer interfaces, which I had not been exposed to before. I wanted to work on both the hardware infrastructure and the software/control side, and this project allowed me to do exactly that. However, I would like to further deepen my knowledge of signal processing and machine learning as applied to EEG, which remains an area I have yet to explore.

I particularly enjoyed splitting my time between foundational work on the platform (repairs, documentation, tuning) and the lab testing phases with the drone, where the results of the tuning become directly visible. Carrying out this internship abroad, within a research university, also allowed me to discover a way of working different from that of a company — more autonomous and more focused on long-term knowledge transfer, since a significant part of my work consisted of documenting my findings for the interns who will continue this project.

Finally, this internship at NTNU allowed me not only to immerse myself in research on robotics and brain-computer interfaces, but also to carry out a concrete project in a field that interests me, while laying the groundwork needed for this work to be effectively picked up by future interns.

Bibliography

[1] D. N. Grønn, "[master's thesis on an EEG-based drone control platform]," Master's thesis, Norwegian University of Science and Technology (NTNU), Department of Engineering Cybernetics, Trondheim, Norway, n.d.

[2] Norwegian University of Science and Technology, "About NTNU." `https://www.ntnu.edu/about`. Accessed: 2026-09-29.

[3] OpenBCI, "OpenBCI: open-source brain-computer interface tools." `https://openbci.com/`. Accessed: 2026-09-29.

[4] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain-computer interfaces for communication and control," *Clinical Neurophysiology*, vol. 113, no. 6, pp. 767–791, 2002.

[5] T. O. Zander and C. Kothe, "Towards passive brain-computer interfaces: applying brain-computer interface technology to human-machine systems in general," *Journal of Neural Engineering*, vol. 8, no. 2, p. 025005, 2011.

[6] G. Pfurtscheller and F. H. L. da Silva, "Event-related EEG/MEG synchronization and desynchronization: basic principles," *Clinical Neurophysiology*, vol. 110, no. 11, pp. 1842–1857, 1999.

[7] G. Pfurtscheller and C. Neuper, "Motor imagery and direct brain-computer communication," *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1123–1134, 2001.

[8] Bitcraze AB, "Crazyflie 2.1." `https://www.bitcraze.io/products/crazyflie-2-1/`. Accessed: 2026-09-29.

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

[10] Bitcraze AB, "Loco positioning system." `https://www.bitcraze.io/documentation/system/positioning/loco-positioning-system/`. Accessed: 2026-09-29.

[11] Bitcraze AB, "Crazyflie PC client (cfclient) documentation." `https://www.bitcraze.io/documentation/repository/crazyflie-clients-python/master/`. Accessed: 2026-09-29.

Appendix A

Internship Assessment Report

The assessment report completed and signed by the NTNU supervisor, Prof. Marta Molinas, is reproduced on the following pages.

RAPPORT D'EVALUATION
ASSESSMENT REPORT

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :
At the end of the internship, please return this report via mail or email to:

ENSTA Bretagne – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
☎ 00.33 (0) 2.98.34.87.70 / stages@ensta-bretagne.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name* Norwegian University of Science and Technology, Brain Cybernetics lab.

Adresse / *Address* O.S: Bragstad plass 2D, 7491, Trondheim, Norway

Tél / *Phone (including country and area code)* +47 94287670

Nom du superviseur / *Name of internship supervisor* Maria Marta Molinas Cabrera, Professor, NTNU

Fonction / *Function* Professor

Adresse e-mail / *E-mail address* marta.molinas@ntnu.no

Nom du stagiaire accueilli / *Name of intern* Guilhem Merlet

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
*Please attribute a mark from **A (excellent)** to **F (very weak).***

MISSION / *TASK*

❖ La mission de départ a-t-elle été remplie ? A B C D E F
Was the initial contract carried out to your satisfaction?

❖ Manquait-il au stagiaire des connaissances ? ☑ oui/*yes* ☐ non/*no*
Was the intern lacking skills?

Si oui, lesquelles ? / *If so, which skills?* _____

ESPRIT D'EQUIPE / *TEAM SPIRIT*

❖ Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / *Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)*

A B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* The peformance was excellent (A) and I am very satisfied of the work done by the intern.

COMPORTEMENT AU TRAVAIL / *BEHAVIOUR TOWARDS WORK*

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?

Did the intern live up to expectations? (*Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?*

A B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'est –il rapidement adapté à de nouvelles situations ? A B C D E F
(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations? A B C D E F
(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication ? A B C D E F
Was the intern open to listening and expressing himself /herself?

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

OPINION GLOBALE / *OVERALL ASSESSMENT*

❖ La valeur technique du stagiaire était : A B C D E F
Please evaluate the technical skills of the intern:

III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?

Would you be willing to host another intern next year? ☑ oui/*yes* ☐ non/*no*

Fait à _____, le _____
In O.S: Bragstad plass 2D, 7491, Trondheim, Norway, on September 08, 2026

molina Digitally signed by molinas Date: 2026.09.08 12:26:47 +02'00'

Signature Entreprise _____ Signature stagiaire
Company stamp _____ *Intern's signature*

Merci pour votre coopération
We thank you very much for your cooperation