Final Review Meeting

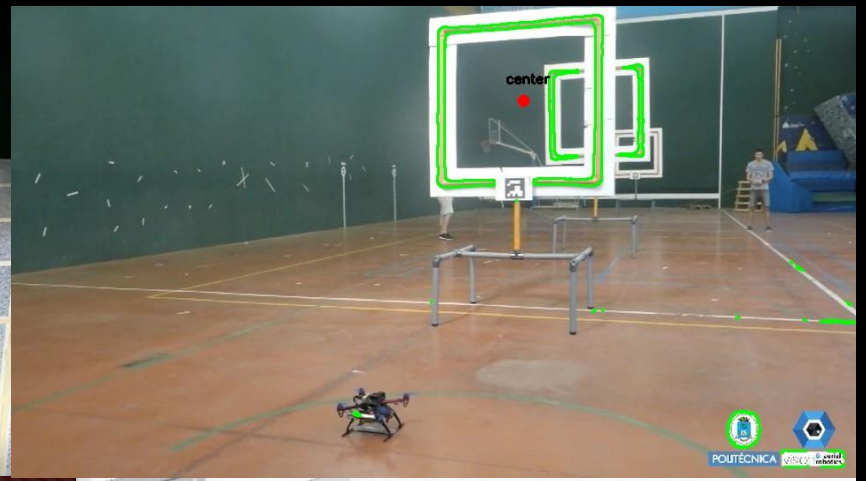
INGENIA-SE Autonomous Drone Racing Challenge Hell-ix Project

May 23, 2023

Contents

1. **Introduction**
2. **Objectives**
3. **Structure**
4. **Execution**
5. **Numbers**
6. **Reports by Areas**
   - 6.1. Competition Delegate Report
   - 6.2.Tooling Manager Report
   - 6.3. Configuration Manager Report
   - 6.4. Documentation Manager Report
   - 6.5. Communication Manager Report
   - 6.6. Modeling Manager Report
   - 6.7. Simulation Manager Report
   - 6.8. Programming Manager (Python) Report
   - 6.9. Programming Manager (MATLAB) Report
   - 6.10. Implementation Manager Report
   - 6.11. Test Manager Report
7. **Sustainability and Impact Analysis**
8. **Conclusions**

1. Introduction

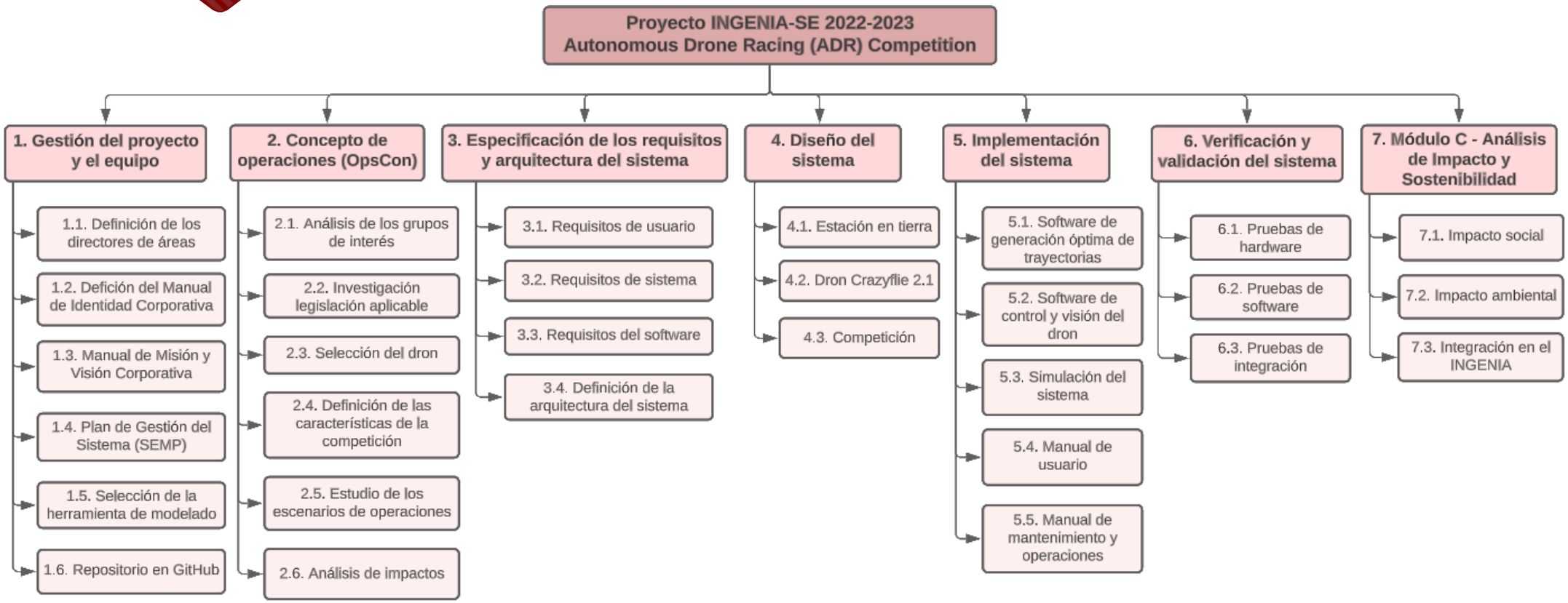


2. Objectives

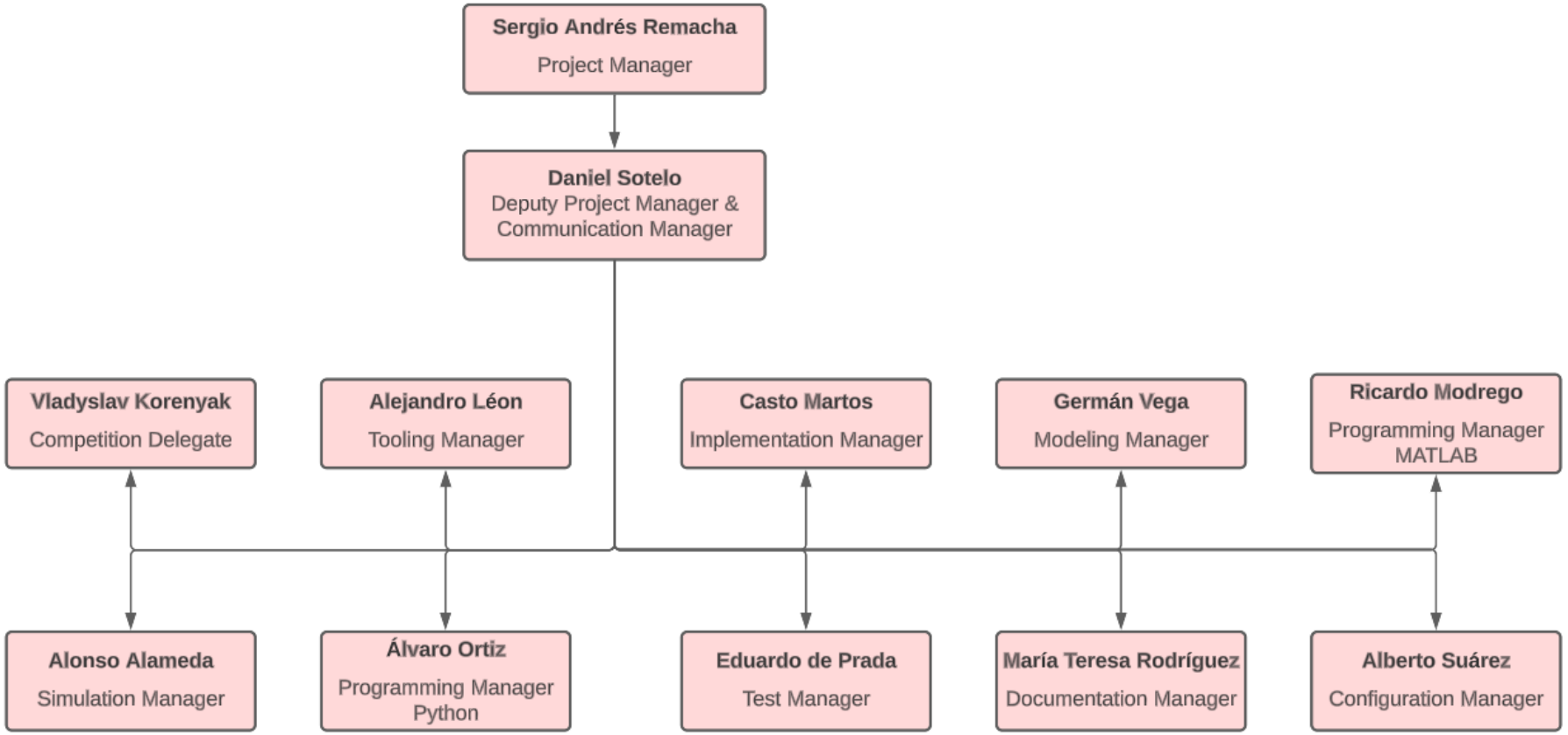
Winning the competition

Acquisition of technical, personal and organizational skills

3. Structure

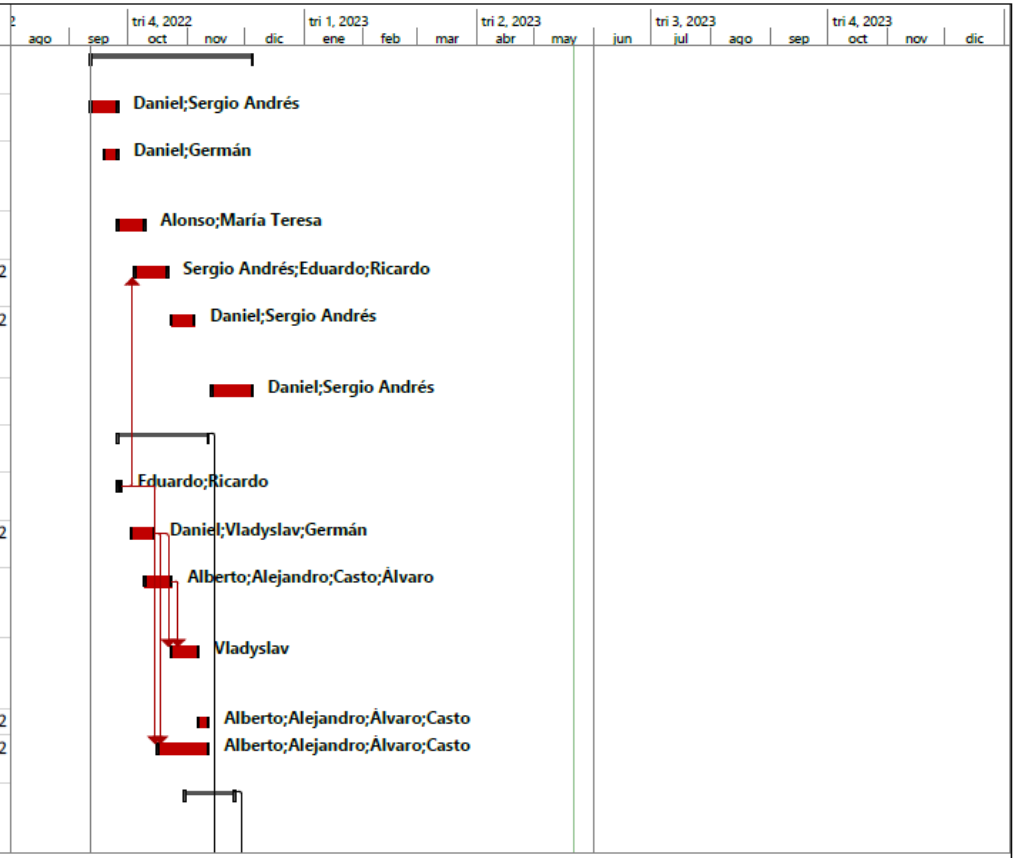

Proyecto INGENIA-SE 2022-2023
Autonomous Drone Racing (ADR) Competition
1. Gestión del proyecto y el equipo
1.1. Definición de los directores de áreas
1.2. Defición del Manual de Identidad Corporativa
1.3. Manual de Misión y Visión Corporativa
1.4. Plan de Gestión del Sistema (SEMP)
1.5. Selección de la herramienta de modelado
1.6. Repositorio en GitHub
2. Concepto de operaciones (OpsCon)
2.1. Análisis de los grupos de interés
2.2. Investigación legislación aplicable
2.3. Selección del dron
2.4. Definición de las características de la competición
2.5. Estudio de los escenarios de operaciones
2.6. Análisis de impactos
3. Especificación de los requisitos y arquitectura del sistema
3.1. Requisitos de usuario
3.2. Requisitos de sistema
3.3. Requisitos del software
3.4. Definición de la arquitectura del sistema
4. Diseño del sistema
4.1. Estación en tierra
4.2. Dron Crazyflie 2.1
4.3. Competición
5. Implementación del sistema
5.1. Software de generación óptima de trayectorias
5.2. Software de control y visión del dron
5.3. Simulación del sistema
5.4. Manual de usuario
5.5. Manual de mantenimiento y operaciones
6. Verificación y validación del sistema
6.1. Pruebas de hardware
6.2. Pruebas de software
6.3. Pruebas de integración
7. Módulo C - Análisis de Impacto y Sostenibilidad
7.1. Impacto social
7.2. Impacto ambiental
7.3. Integración en el INGENIA

3. Structure


Sergio Andrés Remacha
Project Manager
Daniel Sotelo
Deputy Project Manager & Communication Manager
Vladyslav Korenyak
Competition Delegate
Alejandro Léon
Tooling Manager
Casto Martos
Implementation Manager
Germán Vega
Modeling Manager
Ricardo Modrego
Programming Manager MATLAB
Alonso Alameda
Simulation Manager
Álvaro Ortiz
Programming Manager Python
Eduardo de Prada
Test Manager
María Teresa Rodríguez
Documentation Manager
Alberto Suárez
Configuration Manager

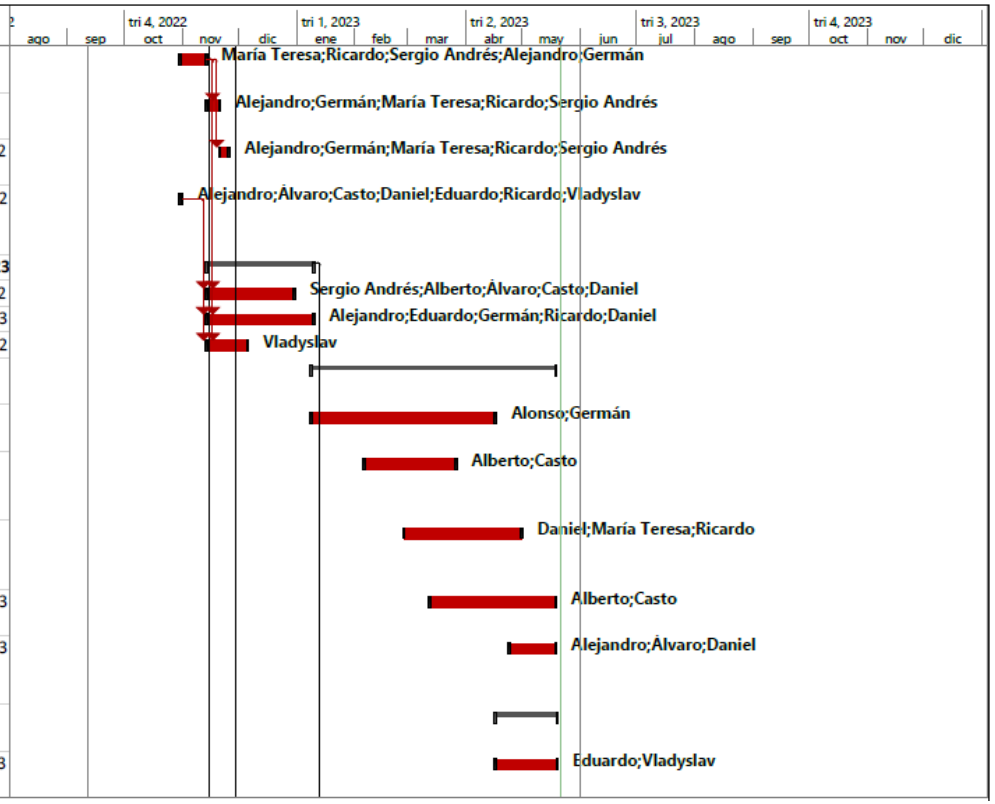
4. Execution

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**1: Gestión del proyecto y del**	**61 días**	**lun 12/09/22**	**dom 04/12/22**
2	1.1. Definición de los directores de	11 días	lun 12/09/22	dom 25/09/22
3	1.2. Definición del Manual de Identidad	6 días	lun 19/09/22	dom 25/09/22
4	1.3. Manual de Misión y Visión	11 días	lun 26/09/22	dom 09/10/22
5	1.4. Plan de Gestión del	13 días	mié 05/10/22	vie 21/10/22
6	1.5. Selección de la herramienta de modelado	10 días	lun 24/10/22	vie 04/11/22
7	1.6. Repositorio de GitHub	16 días	lun 14/11/22	dom 04/12/22
8	**2. Concepto de Operaciones**	**35 días**	**lun 26/09/22**	**vie 11/11/22**
9	2.1. Análisis de los grupos de interés	2 días	lun 26/09/22	mar 27/09/22
10	2.2. Investigación de la legislación	10 días	lun 03/10/22	vie 14/10/22
11	2.5. Estudio de los escenarios de operaciones	11 días	lun 10/10/22	dom 23/10/22
12	2.4. Definición de las características de la competición	11 días	lun 24/10/22	dom 06/11/22
13	2.3. Selección del c	5 días	lun 07/11/22	vie 11/11/22
14	2.6. Análisis de impactos	20 días	lun 17/10/22	vie 11/11/22
15	**3. Especificación de los requisitos y arquitectura del**	**20 días**	**lun 31/10/22**	**vie 25/11/22**



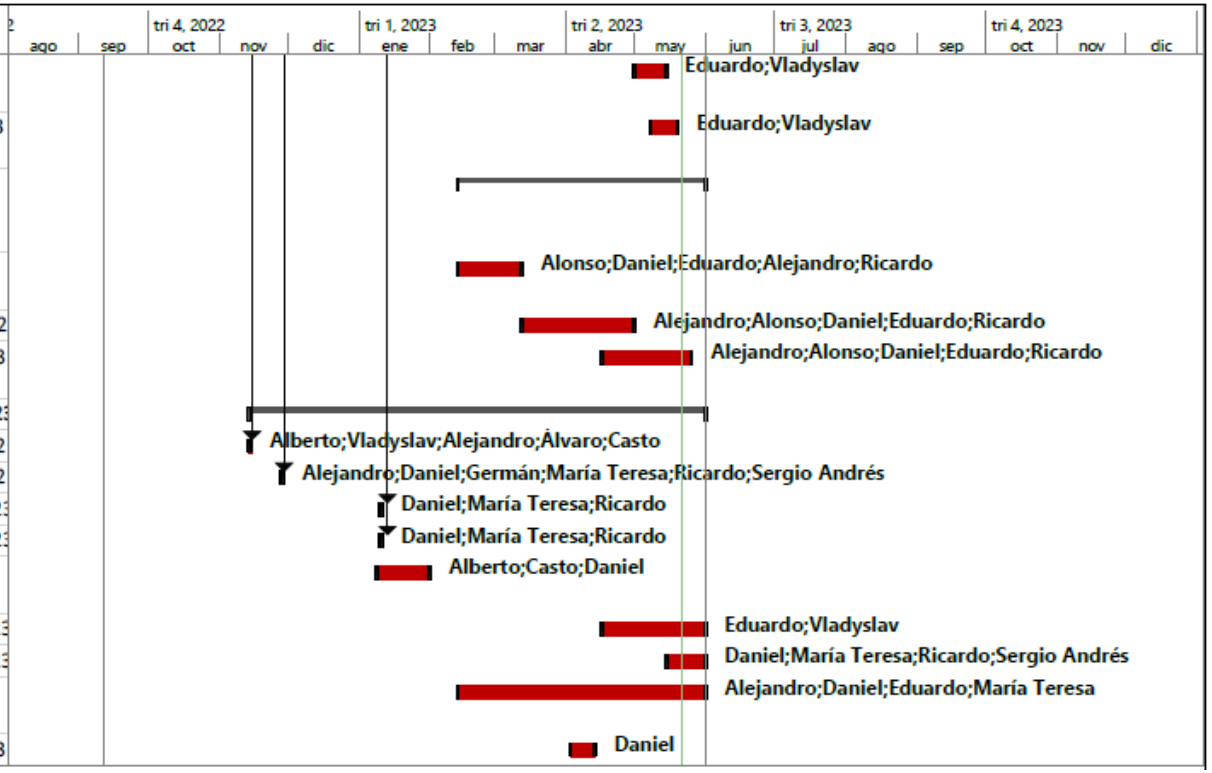
4. Execution

Id	Nombre de tarea	Duración	Comienzo	Fin
16	3.2. Requistos de sistema	11 días	lun 31/10/22	dom 13/11/22
17	3.1. Requisitos de usuario	6 días	lun 14/11/22	dom 20/11/22
18	3.3. Requisitos del software	5 días	lun 21/11/22	vie 25/11/22
19	3.4. Definición de la arquitectura del sistema	1 día	lun 31/10/22	lun 31/10/22
20	**4. Diseño del sistema**	**41 días**	**lun 14/11/22**	**lun 09/01/23**
21	4.1. Estación en tie	35 días	lun 14/11/22	vie 30/12/22
22	4.2. Dron Crazyflie	41 días	lun 14/11/22	lun 09/01/23
23	4.3. Competición	16 días	lun 14/11/22	lun 05/12/22
24	**5. Implementación del sistema**	**94 días**	**lun 09/01/23**	**jue 18/05/23**
25	5.3. Simulación del sistema	71 días	lun 09/01/23	dom 16/04/23
26	5.5. Manual de mantenimiento y operaciones	36 días	lun 06/02/23	dom 26/03/23
27	5.1. Software de generación óptima de	46 días	lun 27/02/23	dom 30/04/23
28	5.4. Manual de usuario	49 días	lun 13/03/23	jue 18/05/23
29	5.2. Software de control y visión del dron	19 días	lun 24/04/23	jue 18/05/23
30	**6. Verificación y validación del**	**25 días**	**lun 17/04/23**	**vie 19/05/23**
31	6.2. Pruebas de software	25 días	lun 17/04/23	vie 19/05/23



4. Execution

Id	Nombre de tarea	Duración	Comienzo	Fin
32	6.1. Pruebas de hardware	11 días	lun 01/05/23	dom 14/05/23
33	6.3. Pruebas de integración	10 días	lun 08/05/23	vie 19/05/23
34	**7. Módulo C - Análisis de impacto y sostenibilidad**	**78 días**	**lun 13/02/23**	**mié 31/05/23**
35	7.2. Impacto ambiental	21 días	lun 13/02/23	dom 12/03/23
36	7.1. Impacto social	36 días	lun 13/03/23	dom 30/04/2
37	7.3. Integración en el INGENIA	29 días	lun 17/04/23	jue 25/05/23
38	**8. Documentación fir**	**143 días**	**lun 14/11/22**	**mié 31/05/2:**
39	8.1. OpsCon	1 día	lun 14/11/22	lun 14/11/22
40	8.2. SyRS	1 día	lun 28/11/22	lun 28/11/22
41	8.3. System design	1 día	mar 10/01/2:	mar 10/01/2:
42	8.4. System model	1 día	mar 10/01/2:	mar 10/01/2:
43	8.5. System implementation	17 días	lun 09/01/23	mar 31/01/23
44	8.6. Test report	33 días	lun 17/04/23	mié 31/05/2:
45	8.7. Project report	13 días	lun 15/05/23	mié 31/05/2:
46	8.8. Sustainability and impact	78 días	lun 13/02/23	mié 31/05/23
47	8.9. Poster	9 días	lun 03/04/23	jue 13/04/23



5. Numbers

Budget: 705.73 €

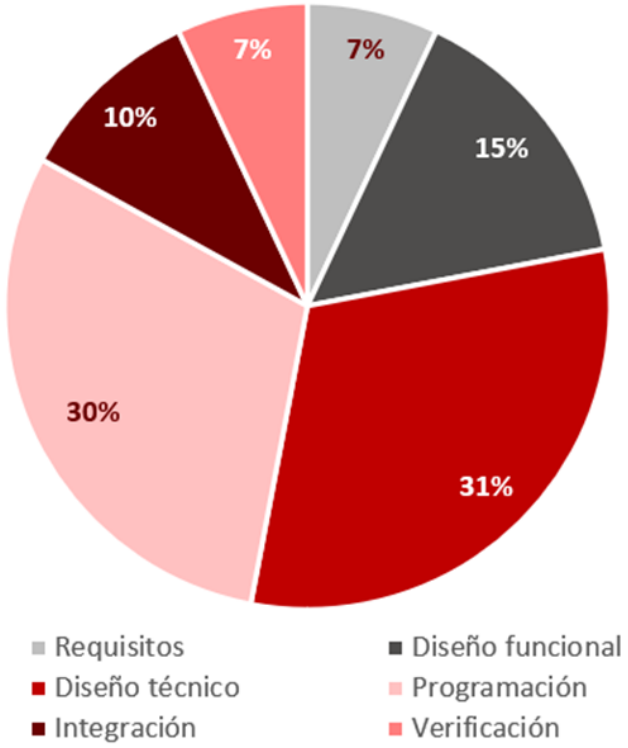
Crazyflie 2.1	260.00 €
AI-deck 1.1	260.00 €
Flow deck v2	58.75 €
LED deck	26.25 €
Female deck header	2.50 €
Spare male headers	3.13 €
Battery holder deck	6.25 €
250 mAh battery	7.50 €
250 mAh battery with 500 mA USB charger	11.88 €
4x spare motors mounts	7.50 €
2x Thust upgrade bundle	37.50 €

Initial date: 12/09/2022

Final date: 23/05/2023

Number of participants: 12

Competition tests: 3



6. Reports by Areas

Project Manager

Competition Delegate

Tooling Manager

Configuration Manager

Documentation Manager

Deputy Project Manager
Communication Manager

Modeling Manager

Simulation Manager

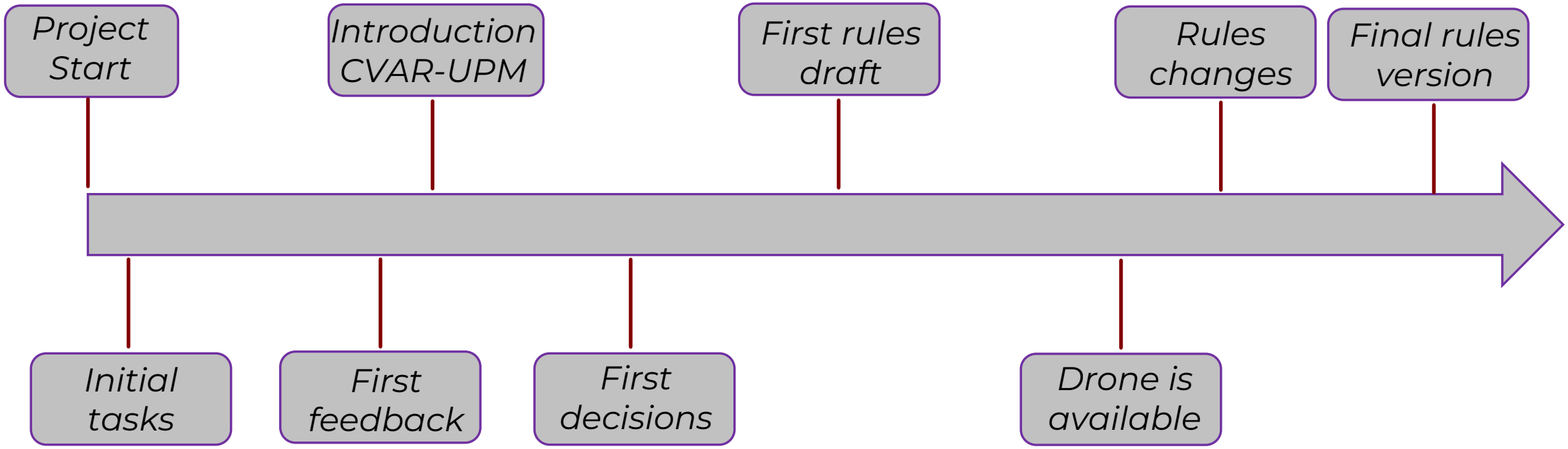
Programming Manager Python

Programming Manager MATLAB

Implementation Manager

Test Manager

6.1. Competition Delegate Report
Overview

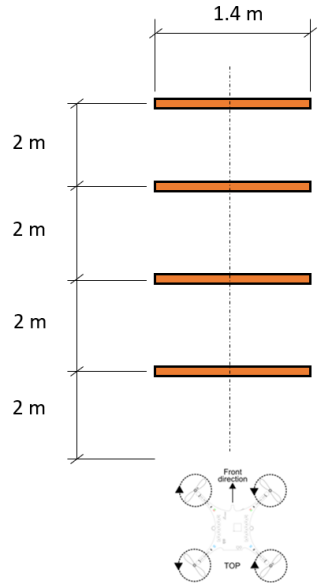


6.1. Competition Delegate Report
Competition rules

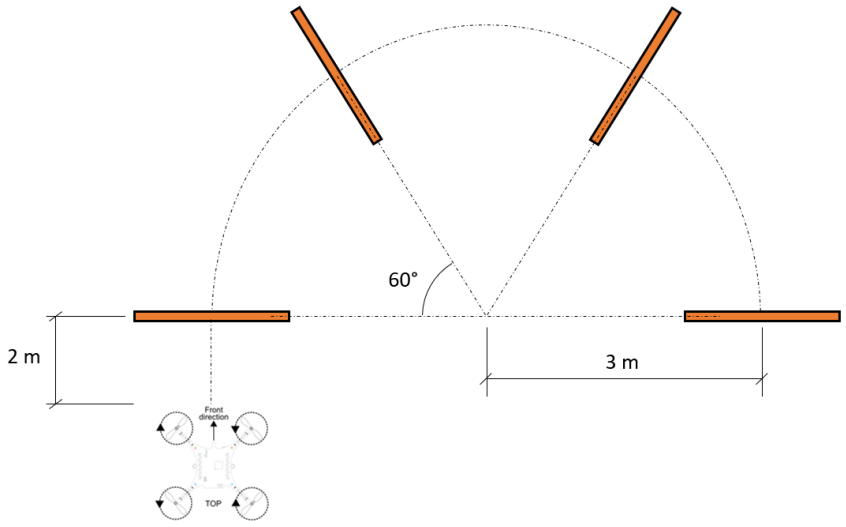
2 preliminary tests
- *Speed*
- *One obstacle*

3 challenges
- *4 obstacles in line*
- *4 obstacles in a circle*
- *Freestyle*

4 consecutive obstacles in line



4 consecutive obstacles disposed in a semicircle



6.1. Competition Delegate Report
Competition rules

2 preliminary tests	*Speed*	*Eliminatory*
	One obstacle	*Eliminatory*
3 challenges	*4 obstacles in line*	*1 point*
	4 obstacles in a circle	*1 point*
	Freestyle	*1 point*

6.2. Tooling Manager Report
Planned Work

Model program selection: Matlab System Composer

Criteria	MATLAB System Composer	Capella	Eclipse Papyrus	Cameo Systems Modeler	Modelio
Accesibility (20%)	10	8	8	0	4
Integration of SysML (20%)	9	3	5	8	4
Usability (40%)	9	9	6	6	5
Community (20%)	7	9	7	5	3
Weighted Total	**8.8**	7.6	6.4	5	4.2

6.2. Tooling Manager Report
Planned Work

Drone adquisition process made by communication manager

Drone selection: Crazy Flie

Crazy Flie

Recommended by CAR

		Crazy Flie		Ryze Tello	
Factor	Weighing	Mark	Weighing	Mark	Weighing
Weight	0.01	8	0.08	7	0.07
Battery duration	0.06	5	0.3	7	0.42
Size	0.01	7	0.07	8	0.08
Autonomous driving possibilities	0.5	10	5	5	2.5
Price	0.1	6	0.6	8	0.8
Hardware accesories	0.16	9	1.44	2	0.32
Code that already existed	0.16	9	1.44	9	1.44
Final mark	1		8.93		5.63

Ryze Tello

6.2. Tooling Manager Report
Planned Work

Crazyflie modules:

- AI Deck: camera vision (detect obstacles and correct position deviations)
- Flow Deck: 3D positioning
- Led ring: colorful lights

Led ring

Flow Deck

AI Deck

6.2. Tooling Manager Report
Planned Work

Programming environment: Visual Studio Code

- Most of the team already had experience
- Easy to use
- Free
- Available

6.2. Tooling Manager Report
Planned Work

Simulation software selection: Webots

- Gazebo: good when flying the real drone, not that good at simulating the drone flight.
- Webots: it has more options to control a Crazyflie simulation than Gazebo.

GAZEBO

Webots™
robot simulation

6.2. Tooling Manager Report
Planned Work

Location to test our drone: CAR drone cage

- Big space
- No additional allowance needed to fly
- On campus
- Security net
- 2 cameras
- 4 obstacles (1,4 x 1,4 metres doors) given by CAR

6.2. Tooling Manager Report
Planned Work

Presentations software: Microsoft PowerPoint

- All team members already had experience
- Working in several devices at the same time via Microsoft Teams
- Easy to use
- Free
- Available

6.2. Tooling Manager Report
Tooling Selection

Chosen tools:

- **Model program selection:** Matlab System Composer
- **Drone selection:** Crazy Flie
- **Programming environment:** Visual Studio
- **Simulation software selection:** Webots
- **Location to test our drone:** CAR drone cage
- **Presentations software:** Microsoft PowerPoint

Team members think that all tooling choices have been right except one: the drone.

6.2. Tooling Manager Report
Analysis

Crazy Flie or Ryze Tello

Lack of time → simplify
Lack of time → not depend on low-availability products

Ryze Tello would have coped better with this issue,
and the client's requirements could have been met with this drone too.

6.3. Configuration Manager Report
Planned Work

Configuration management tool: GitHub

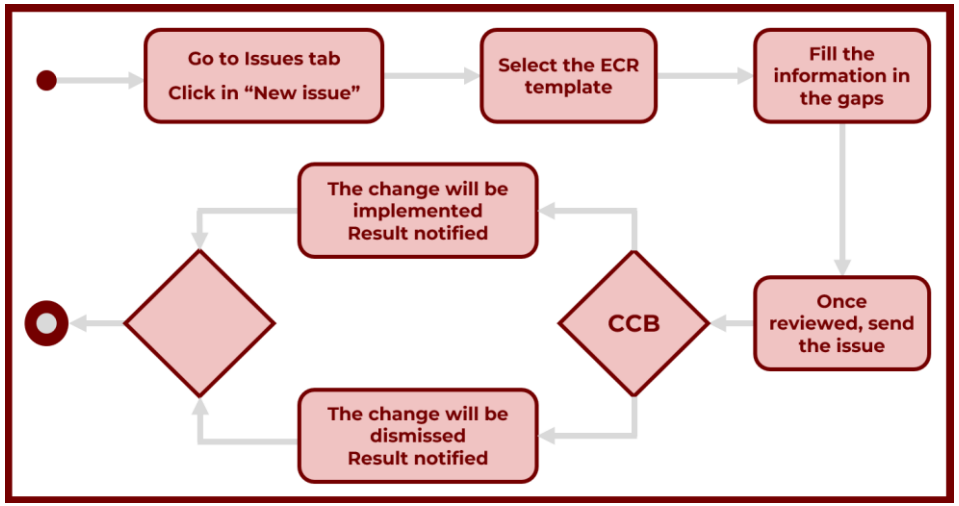
- Hell-ix repository creation
- Participants Integration
- Tool understanding and formation
- Monitoring

Repository contents:

- Documentation folder
- Hellix Project folder
- Simulation folder
- Software folder
- Docs
- imgs

6.3. Configuration Manager Report
Work Done

ECR: Engineering Change Request



name	about	title	labels	assignees
Engineering Change Request	Submit an ECR Form.	yy/nn Request Name	enhancement	

Project:	Hell-ix	**ECR ID#:**	yy/nn
Requestor:	Requestor's name	**Date:**	dd/mm/aaaa
Project Managers:	Sergio Andrés Remacha / Daniel Sotelo	**Priority:**	Priority level
Configuration Managers:	Alberto Suárez / María Teresa Rodríguez	**CI Target:**	Target

Change request type:

- ☐ New Requirement
- ☐ Requirement Change
- ☐ Other

ECR Classification:

- ☐ Major
- ☐ Minor

Requested change description: give a description about the request you're about to submit.

Requested change reason: argument why this should be taken account of.

6.3. Configuration Manager Report
Analysis

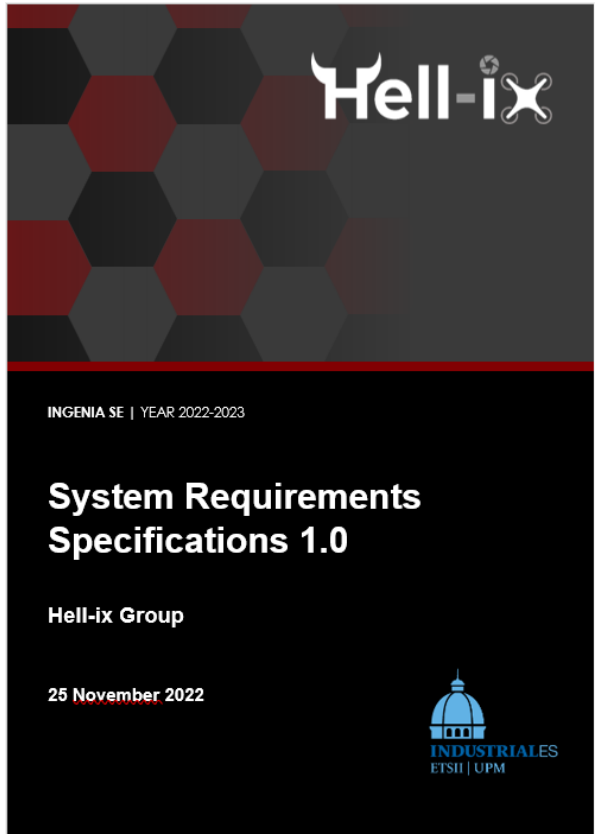
Benefits:

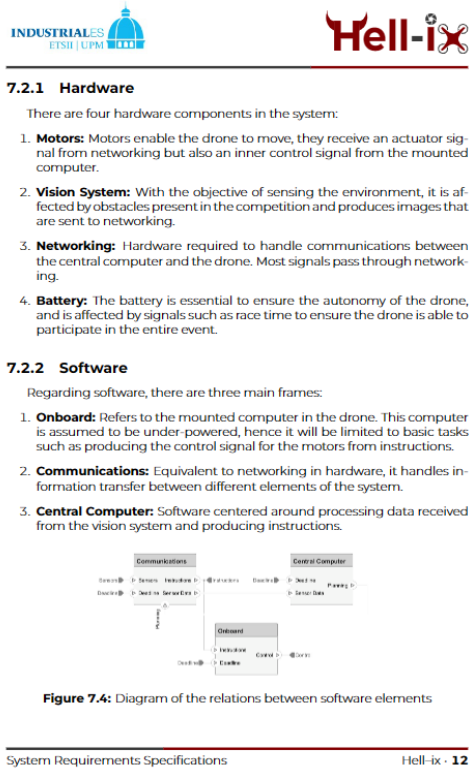
- **Effective collaboration:** It has allowed multiple developers to work simultaneously in the same repository.
- **Advanced version control:** Precise tracking of changes and the ability to roll back to previous versions.
- **Integration with popular tools:** Easily integrates with other tools.

6.4. Documentation Manager Report
Planned Work

TEMPLATE FOR STANDARDIZING DOCUMENTS

- Hell-ix logo
- Hell-ix motives for consistency and team identity
- Document version
- Date
- Standardized format





INDUSTRIALES ETSII | UPM — Hell-ix

7.2.1 Hardware

There are four hardware components in the system:

1. **Motors:** Motors enable the drone to move, they receive an actuator signal from networking but also an inner control signal from the mounted computer.
2. **Vision System:** With the objective of sensing the environment, it is affected by obstacles present in the competition and produces images that are sent to networking.
3. **Networking:** Hardware required to handle communications between the central computer and the drone. Most signals pass through networking.
4. **Battery:** The battery is essential to ensure the autonomy of the drone, and is affected by signals such as race time to ensure the drone is able to participate in the entire event.

7.2.2 Software

Regarding software, there are three main frames:

1. **Onboard:** Refers to the mounted computer in the drone. This computer is assumed to be under-powered, hence it will be limited to basic tasks such as producing the control signal for the motors from instructions.
2. **Communications:** Equivalent to networking in hardware, it handles information transfer between different elements of the system.
3. **Central Computer:** Software centered around processing data received from the vision system and producing instructions.

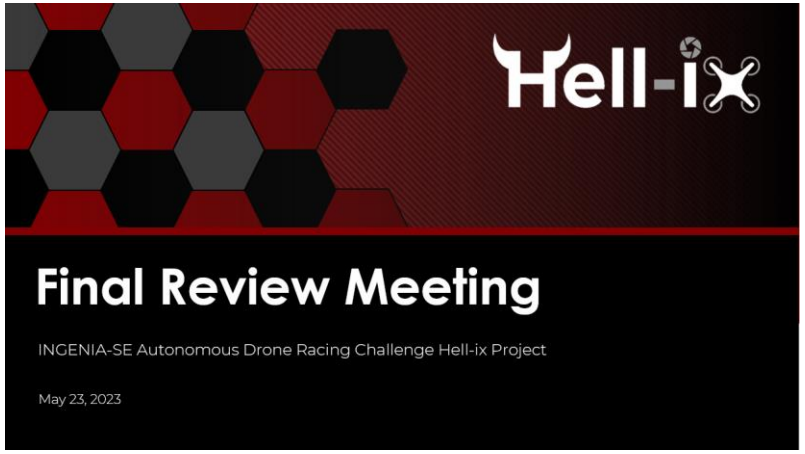
Figure 7.4: Diagram of the relations between software elements

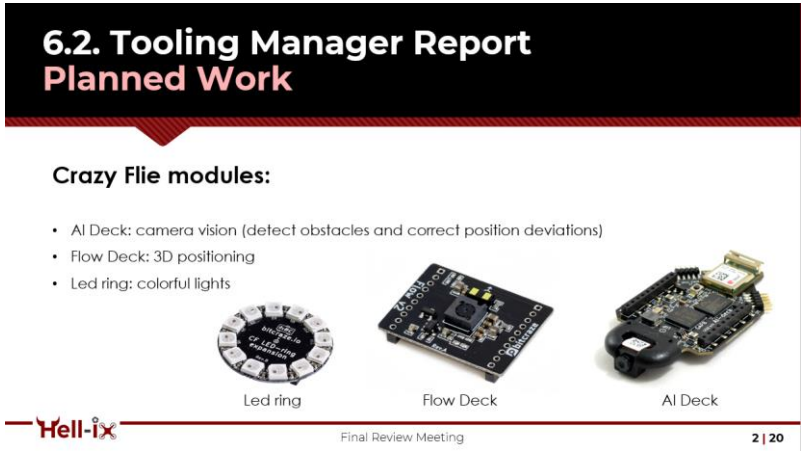
System Requirements Specifications — Hell-ix · 12

6.4. Documentation Manager Report
Planned Work

TEMPLATE FOR STANDARDIZING PRESENTATIONS

- Hell-ix logo
- Hell-ix motives for consistency and team identity
- Date
- Standardized format

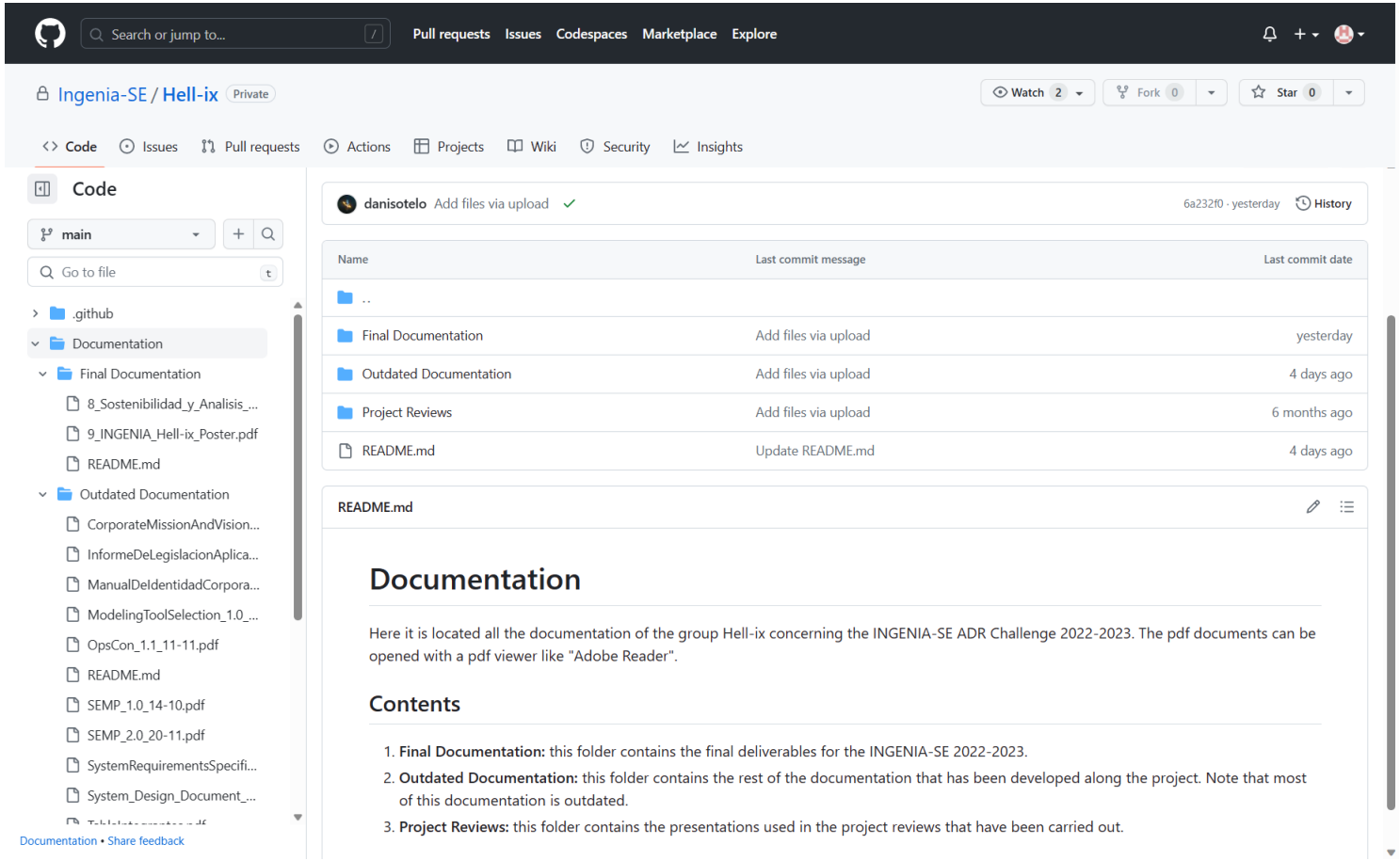




6.4. Documentation Manager Report
Planned Work

Overleaf
- LaTeX
- Collaborative writing
- Real-time collaboration
- Ease of sharing and editing documents

GitHub
- Centralized repository
- Version Control
- Version number identication



6.4. Documentation Manager Report
Work Done

SEPT
DEFINE AREA LEADERS

OCT
CORPORATE IDENTITY MANUAL
CORPORATE MISSION AND VISION
INVESTIGATE LAWS
SEMP 1.0

NOV
MODELING TOOL SELECTION
OPSCON 1.0
PROJECT REVIEW
WRITE CM PLAN
"HELLO WORLD!" PROJECT
SYSREQ 1.0
SEMP 2.0

DEC

JAN
SYSTEM DESIGN 1.0

FEB

MAR

APR

MAY
INGENIA SE POSTER
ENTREGA MÓDULO C
FINAL PRESENTATION

END MAY
ENTREGA MÓDULO C
OPSCON
SYSREQ
SYSTEM DESIGN
SYSTEM MODEL
SYSTEM IMPLEMENTATION
TEST REPORT
PROJECT REPORT
SUSTAINABILITY AND IMPACT ANALYSIS

6.4. Documentation Manager Report
Analysis

Benefits of Overleaf and GitHub in document management:

- Facilitation of collaboration among team members
- Efficient version control and tracking
- Quick access to the latest document versions and ability to view previous versions

Impact on project success:

- Increased efficiency and productivity in document management
- Improved quality and consistency of documents and presentations
- Reduced errors and conflicts in collaborative editing

6.5. Communication Manager Report
Planned Work

Hell-ix Communication Plan and Objectives:

- Ensure appropiate communication with stakeholders
- Ensure appropiate internal communication tools and quality
- Ensure access to experts advice
- Enable easy-channel communications with suppliers

6.5. Communication Manager Report
Work Done

External Communication:

- Logo and Corporative Image
- Moodboard
- Poster

Hardware acquisition: e-mail

- Carlos from CAR and Ricardo, the INGENIA-SE coordinator

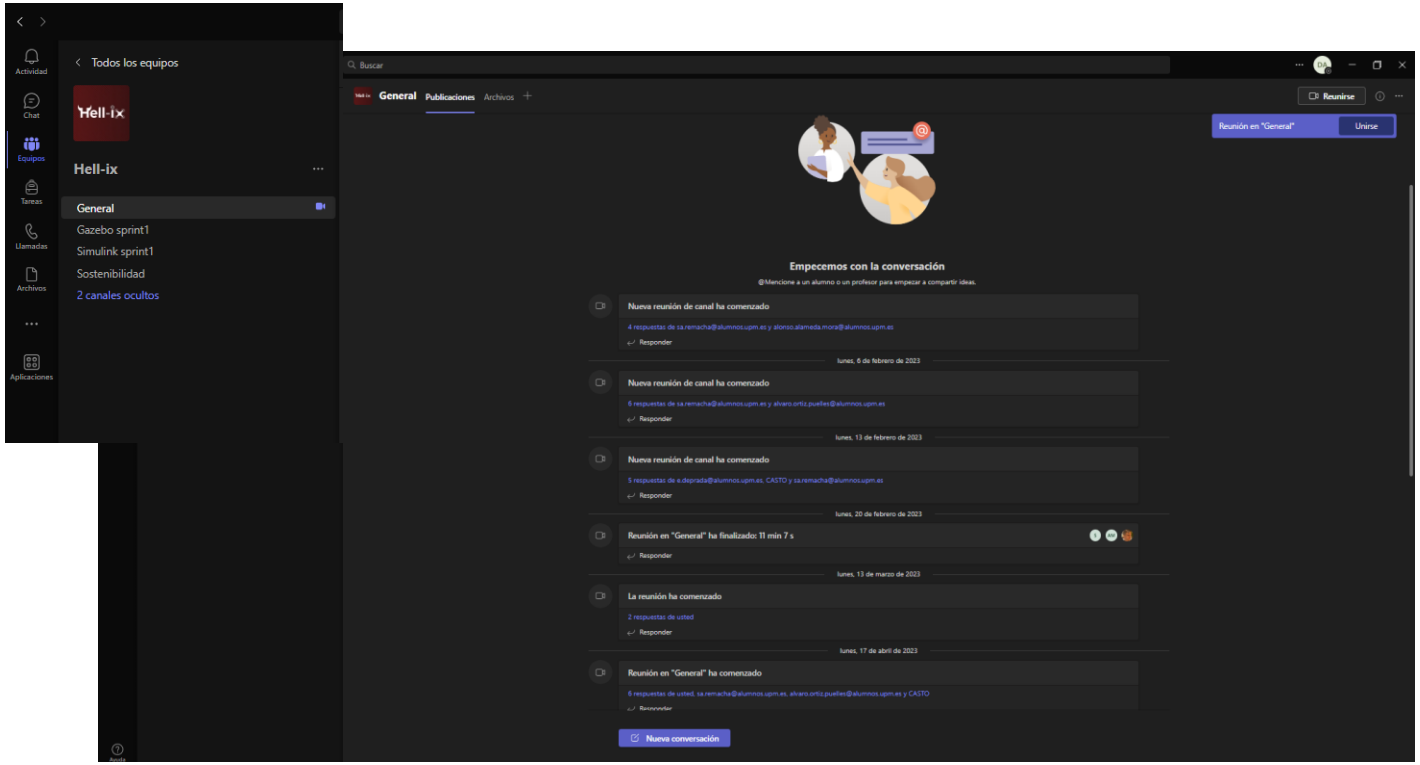
Asks for experts advice:

- CAR Vision Group

6.5. Communication Manager Report
Work Done

Internal Communication:

- Teams
- WhatsApp



6.6. Modeling Manager Report
Planned Work

Initial Objectives

1. Identify all relevant components for the system
2. Define the requirements for the system and its components
3. Create a model of the system, its components and the relations between them
4. Create a Simulink model of the drone that capture its dynamics

Tools Used

MATLAB® + Simulink
Requirements Toolbox
System Composer

6.6. Modeling Manager Report
Planned Work

Previous Research

©2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

CrazyS: a software-in-the-loop platform for the Crazyflie 2.0 nano-quadcopter

Giuseppe Silano, Emanuele Aucone and Luigi Iannelli

Abstract—**In this paper we propose CrazyS, an extension of the ROS (Robot Operating System) package RotorS, aimed to modeling, developing and integrating the Crazyflie 2.0 nano-quadcopter in the physics based simulation environment Gazebo.**

Such simulation platform allows to understand quickly the behavior of the flight control system by comparing and evaluating different indoor and outdoor scenarios, with a details level quite close to reality. The proposed extension expands RotorS capabilities by considering the Crazyflie 2.0 physical model and its flight control system, as well.

A simple case study has been considered in order to show how the package works. The use of open-source software makes the platform available for scientific and educational activities.

I. Introduction

Unmanned Aerial Vehicles (UAVs), although originally designed and developed for defense and military purposes (e.g., aerial attacks or military air covering), during the last years gained increasing interest and attention related to civilian use. Nowadays, UAVs are employed for several tasks and services like surveying and mapping [1], for rescue operations in disasters [2], for buildings inspection [3], geophysics exploration [4], traffic monitoring [5], animal protection [6], agricultural crops [7].

Many existing algorithms for the autonomous control [8], [9] and navigation [10] are provided in the literature, but it is particularly difficult to make the UAVs able to work autonomously in constrained and unknown environments or also indoor. Thus, it follows the need for tools that allow to understand what it happens when some new applications are going to be developed in unknown or critical situations. Simulation is one of such helpful tools, widely used in robotics [11], [12], and whose main benefits are costs and time savings, enabling to carry out and to study complex missions that might be time consuming and risky in real world. Furthermore, bugs and mistakes in simulation cost virtually nothing: it is possible to crash a vehicle several times and thereby getting a better understanding of implemented methods under various conditions. To this aim, simulation environments are able to manage the complexity and heterogeneity of the hardware and applications, to promote the integration of new technologies, to simplify software design, to hide the complexity of low-level communication [13].

Different solutions, typically based on external robotic simulators such as Gazebo [14], V-REP [15], AirSim [16], are available to this purpose. They employ recent advances in computation and graphics (e.g., the AirSim photorealistic environment [17]) in order to simulate physical phenomena (gravity, magnetism, atmospheric conditions) and perception (e.g., providing sensor models) in such a way that the environment realistically reflects the actual world. Definitely, it comes out that complete software platforms able to test different algorithms for UAVs moving in a simulated 3D environment are becoming more and more important.

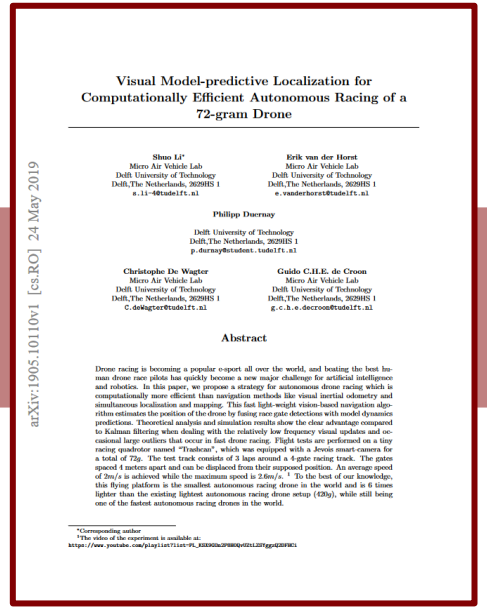
In this paper it is described CrazyS, a software package for modeling, developing and integrating the dynamics and the control architecture of the nano-quadcopter Crazyflie 2.0 (Fig. 1) in the Gazebo simulator. CrazyS is based on the Micro Aerial Vehicles (MAVs) simulation framework RotorS [18] and such contribution illustrates how it is possible to expand the functionalities of the simulation platform in the UAV field, by facilitating the development of different control strategies before testing them on a real platform, possibly with few changes, thanks to software-in-the-loop methodologies [9].

The chosen aircraft, the Crazyflie 2.0, is available on the market for less than $200 and it is ideal for many research areas (e.g., large swarm [19], path planning [20], mixed reality [21], etc.). The source code and the hardware are open, making it possible to go through any part of the system for complete control and full flexibility. New hardware and sensors can be linked through the versatile expansion ports, enabling the addition of the latest sensors. The small size and light weight reduce the need for safety equipment and increase the productivity. For all such reasons it appears valuable to have a realistic and detailed simulator of the Crazyflie dynamic behavior, with the possibility of validating in an easy way the effects of modifying the control architecture for complex missions.

Fig. 1. The Crazyflie 2.0 nano-quadcopter.

Giuseppe Silano, Emanuele Aucone and Luigi Iannelli are with the Department of Engineering, University of Sannio in Benevento, Piazza Roma, 21 – 82100 Benevento, Italy, email: emanuele.aucone@alice.it and {giuseppe.silano, luigi.iannelli}@unisannio.it

Basic Modelling and Control



arXiv:1905.10110v1 [cs.RO] 24 May 2019

Visual Model-predictive Localization for Computationally Efficient Autonomous Racing of a 72-gram Drone

Sihao Li*
Micro Air Vehicle Lab
Delft University of Technology
Delft,The Netherlands, 2629HS 1
s.li-4@tudelft.nl

Erik van der Horst
Micro Air Vehicle Lab
Delft University of Technology
Delft,The Netherlands, 2629HS 1
e.vanderhorst@tudelft.nl

Philipp Duernay
Delft University of Technology
Delft,The Netherlands, 2629HS 1
p.durnay@student.tudelft.nl

Christophe De Wagter
Micro Air Vehicle Lab
Delft University of Technology
Delft,The Netherlands, 2629HS 1
c.dewagter@tudelft.nl

Guido C.H.E. de Croon
Micro Air Vehicle Lab
Delft University of Technology
Delft,The Netherlands, 2629HS 1
g.c.h.e.decroon@tudelft.nl

Abstract

Drone racing is becoming a popular e-sport all over the world, and beating the best human drone race pilots has quickly become a new major challenge for artificial intelligence and robotics. In this paper, we propose a strategy for autonomous drone racing which is computationally more efficient than navigation methods like visual inertial odometry and simultaneous localization and mapping. This fast light-weight vision-based navigation algorithm estimates the position of the drone by fusing race gate detections with model dynamics predictions. Theoretical analysis and simulation results show the clear advantage compared to Kalman filtering when dealing with the relatively low frequency visual updates and occasional large outliers that occur in fast drone racing. Flight tests are performed on a tiny racing quadrotor named "Trashcan", which was equipped with a Jevois smart-camera for a total of 72g. The test track consists of 3 laps around a 4-gate racing track. The gates spaced 4 meters apart and can be displaced from their supposed position. An average speed of $2m/s$ is achieved while the maximum speed is $2.6m/s$. [1] To the best of our knowledge, this flying platform is the smallest autonomous racing drone in the world and is 6 times lighter than the existing lightest autonomous racing drone setup (420g), while still being one of the fastest autonomous racing drones in the world.

*Corresponding author
[1]The video of the experiment is available at:
https://www.youtube.com/playlist?list=PL_EEEHXOkcZPHHOQvfZfLZ2FfqgzdZ2PFHC1

MPC with Visual Integration

Sliding Mode Control: An approach to Control a Quadrotor

Marco Herrera
Facultad de Ingeniería Eléctrica y Electrónica
Escuela Politécnica Nacional-EPN
Quito, Ecuador
marco.herrera@epn.edu.ec,

Alejandro P. Gómez
Universidad de las Fuerzas Armadas
Departamento de Energía Mecánica
Quito, Ecuador
apgomez@espe.edu.ec

William Chamorro
Facultad de Ingeniería Eléctrica y Electrónica
Escuela Politécnica Nacional-EPN
Quito, Ecuador
william.chamorro@epn.edu.ec

Oscar Camacho
Universidad de Los Andes-ULA, Venezuela
Facultad de Ingeniería Eléctrica y Electrónica.
Escuela Politécnica Nacional-EPN
Quito, Ecuador
oscar.camacho@epn.edu.ec

Abstract— **This paper describes the synthesis and application of controller based on sliding mode theory to a quadrotor. A PD sliding surface is considered for vertical take-off and landing aircraft, also changes in angles are done, and some disturbances are included. Therefore, the controller can be implemented using a PD controller as the sliding surface, and adding some algebra the complete controller algorithm is presented. The controller is tested by simulation.**

Keywords— **Sliding Mode Control, Control theory, UAV's, Nonlinear control.**

I. INTRODUCTION

Quadrotor, also known as Unmanned Aerial Vehicles (UAV), is widely used in several applications, in both military and civilian markets. UAV is nonlinear, unstable and vulnerable to external disturbances. It presents an interesting dynamics where the uncertainties on physical parameters and variable dynamic driving as the drone make it an attractive and challenging control problem. Different methodologies and strategies have been recommended to control a quadrotor. The following approaches have been used: linear quadratic regulator (LQR) control [1], proportional-integral derivative (PID) control [1-4], fuzzy logic (FL) control [3].

Sliding Mode Control (SMC) is a simple robust technique that permits designing controllers for linear and nonlinear processes [6-8]. SMC has become one of the selected solutions for many practical control designs such as electronics devices, robotics, chemical processes and so on [8, 9]. SMC consists in two parts: firstly, a discontinuous control law to enforce the error vector toward a decision rule, called sliding surface, during the reaching phase. This part the control is switching on the different sides of the sliding surface equation and secondly, once the error vector is confined to the sliding surface, an equivalent part of the controller acts to follow the dynamics imposed by the equations describing the sliding surface [6].

There are some works of SMC applied to tracking and regulation tasks for quadrotors, let us make a briefly mention of them, for instance: [10], in this paper is proposed a control schema based on backstepping and its combination with fuzzy system for chattering reduction. [11], the authors present a controller approach where it is obtained by dividing the design in two steps: firstly, a high gain observer is considered and secondly, a dynamic sliding mode controller (DSMC) is proposed. [12] in this article a direct adaptive sliding mode control is developed. The proposed controller is applied without considering disturbances and parameter uncertainties. The design of the adaptation laws is based on Lyapunov design principle. [13], in this article, backstepping and sliding mode as control schemes are presented and compared. [14] In this paper a controller based on high order sliding mode control (HOSMC) is designed for trajectory tracking. [15] Describes a feedback linearization-based controller with a high-order sliding mode observer. The observer works as an observer and estimator of the external disturbances effect. [16], in this paper, the principal attention is based on the dynamic modelling of quadrotor taking into account the high-order nonholonomic constraints. Using the Backstepping approach for the synthesis of tracking errors and Lyapunov functions, a sliding mode controller is obtained. [17], in this paper an adaptive backstepping approach is utilized to design the sliding mode control.

The objective of this paper consists in designing a sliding mode control and applying it by simulation in a quadrotor. A PD sliding surface is considered for vertical take-off and landing aircraft, also changes in angles are done, and some disturbances are included. Therefore, the controller can be implemented using a PD controller as the sliding surface, and adding some algebra the complete controller algorithm is presented.

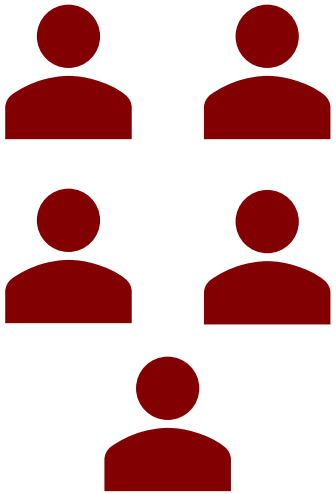
This article is structured as follows. In Section II the quadrotor model is presented. Section III, describes concepts

Advanced Non-Linear Control

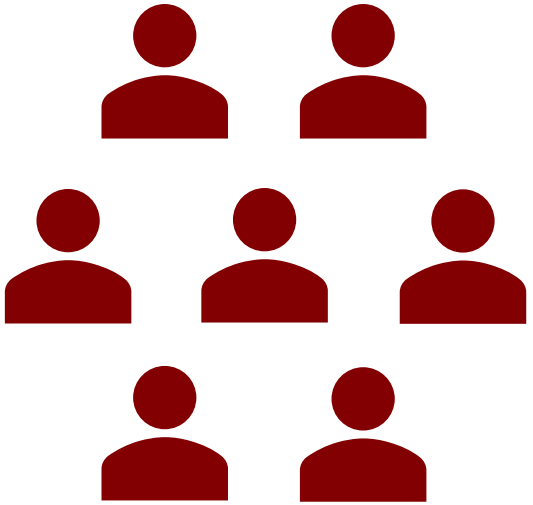
6.6. Modeling Manager Report
Planned Work

Division of Work

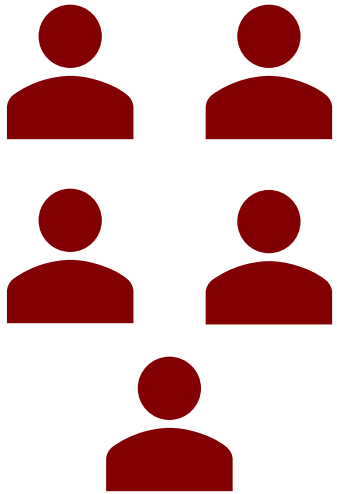
System Requirements



System Composer

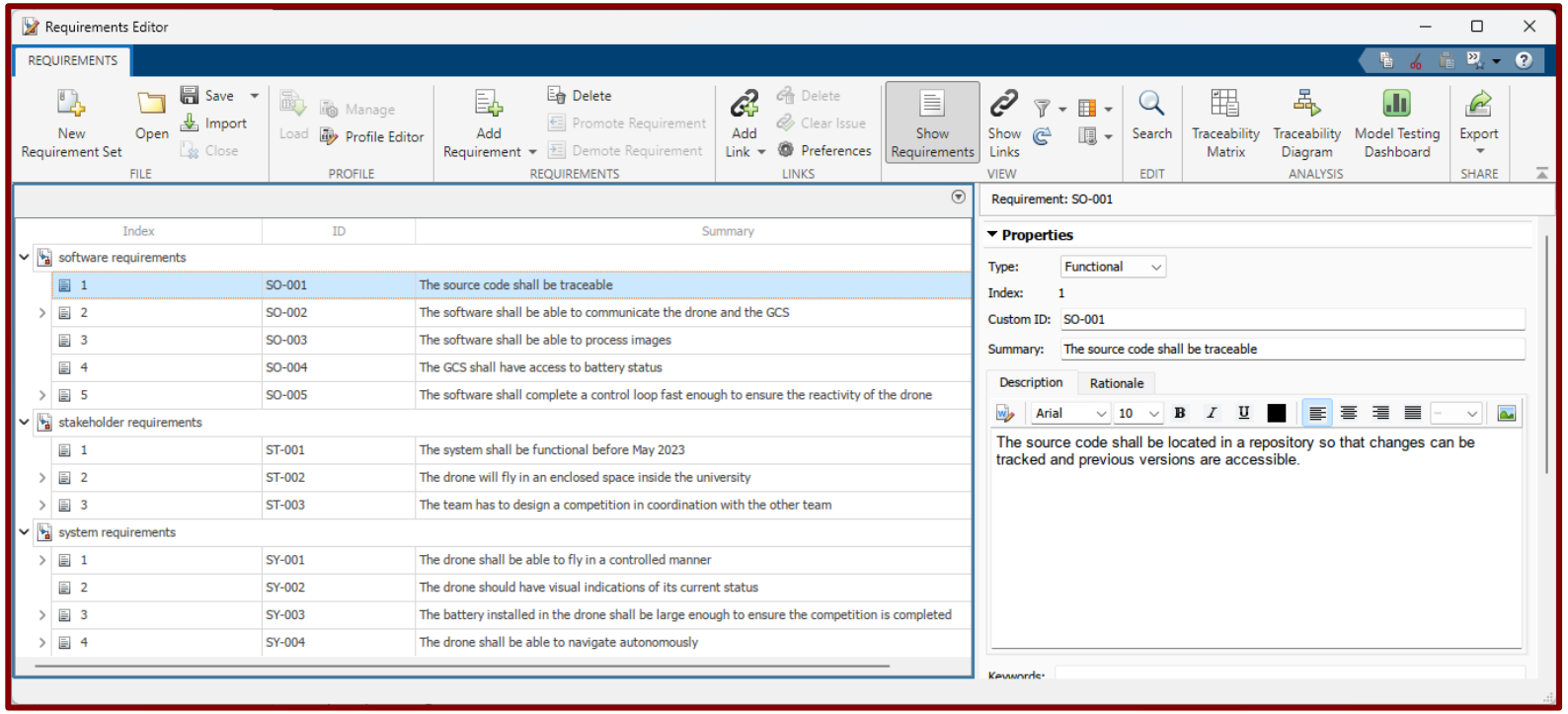


Simulink Model



6.6. Modeling Manager Report
Work Done

Requirements Definition



6.6. Modeling Manager Report
Work Done

Requirements Definition

There are 3 types of requirements:

1. Software Requirements
2. Stakeholder Requirements
3. System Requirements

A total of **33 requirements**

software requirements		
1	SO-001	The source code shall be traceable
2	SO-002	The software shall be able to communicate the drone and the GCS
2.1	SO-002.1	The control system shall have a fail-safe mechanism in case of connection loss
2.2	SO-002.2	The control system shall have an automatic landing mechanism in case of connection loss
2.3	SO-002.3	The control system shall cooperate to estimate the position of the drone
2.4	SO-002.4	The drone shall send images to the GCS
3	SO-003	The software shall be able to process images
4	SO-004	The GCS shall have access to battery status
5	SO-005	The software shall complete a control loop fast enough to ensure the reactivity of the drone
5.1	SO-005.1	The software shall use a compiled programming language
stakeholder requirements		
1	ST-001	The system shall be functional before May 2023
2	ST-002	The drone will fly in an enclosed space inside the university
2.1	ST-002.1	The system must comply to current regulations on drone flights
3	ST-003	The team has to design a competition in coordination with the other team
3.1	ST-003.1	The competition must have a speed test
3.2	ST-003.2	The competition should have an obstacle avoidance test
3.3	ST-003.3	The competition should have a point system
system requirements		
1	SY-001	The drone shall be able to fly in a controlled manner
1.1	SY-001.1	The drone shall be able to lift its own weight plus the components mounted
1.2	SY-001.2	The GCS shall be able to manipulate the drone manually in case of security threats or unexpected events
1.3	SY-001.3	The GCS shall be able to control the movement of the drone automatically
1.4	SY-001.4	The GCS should relay information about the status of the drone
1.5	SY-001.5	The drone shall have automatic lift-off and landing procedures
1.6	SY-001.6	A control algorithm shall run onboard the drone
2	SY-002	The drone should have visual indications of its current status
3	SY-003	The battery installed in the drone shall be large enough to ensure the competition is completed
3.1	SY-003.1	At any moment during the competition, there should be at least two battery replacements for the drone
4	SY-004	The drone shall be able to navigate autonomously
4.1	SY-004.1	The drone shall detect and avoid obstacles in its path
4.1.1	SY-004.1.1	The system shall detect obstacles using a camera mounted on the drone
4.1.2	SY-004.1.2	The drone must react in time to obstacles in its path
4.2	SY-004.2	The drone shall be able to follow a target position
4.3	SY-004.3	The GCS shall generate trajectories for the drone using Bezier curves

6.6. Modeling Manager Report
Work Done

Requirements Documentation

1 The source code shall be traceable

Requirement Type Functional

ID SO-001

Description

The source code shall be located in a repository so that changes can be tracked and previous versions are accessible.

Change Information No change issue detected.

Links

Artifact: Hell_ix_System_Architecture.slx

Linked Item	Link Type
Program Selection	Implemented by
Star	Implemented by
Spiral	Implemented by
Race Trajectory Planner	Implemented by
Path Follower Assisted by Computer Vision	Implemented by
CrazyRadio PA	Implemented by

→

Software/Source Code/SRS/SO-001

Function The source code must be traceable

Description

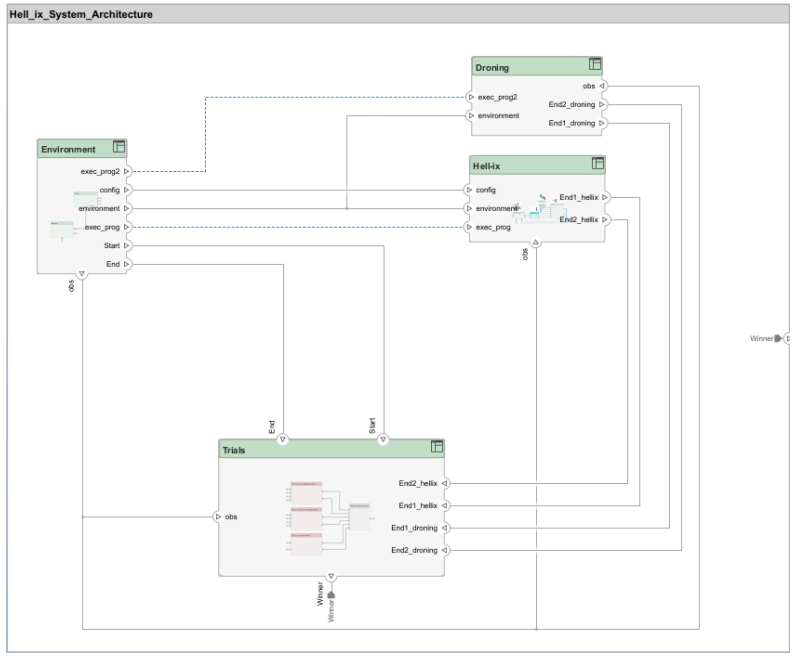
The source code shall be located in a repository so that changes can be tracked and previous versions are accessible.

Automatic Requirement

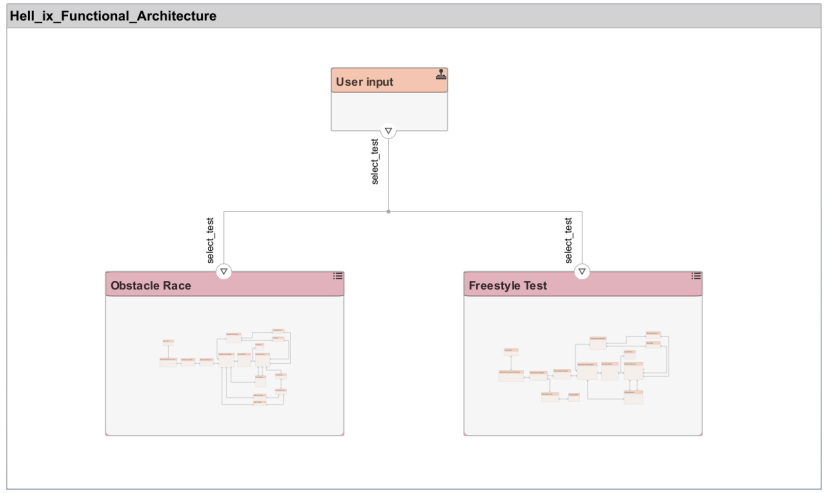
Final Requirement

6.6. Modeling Manager Report
Work Done

System Model



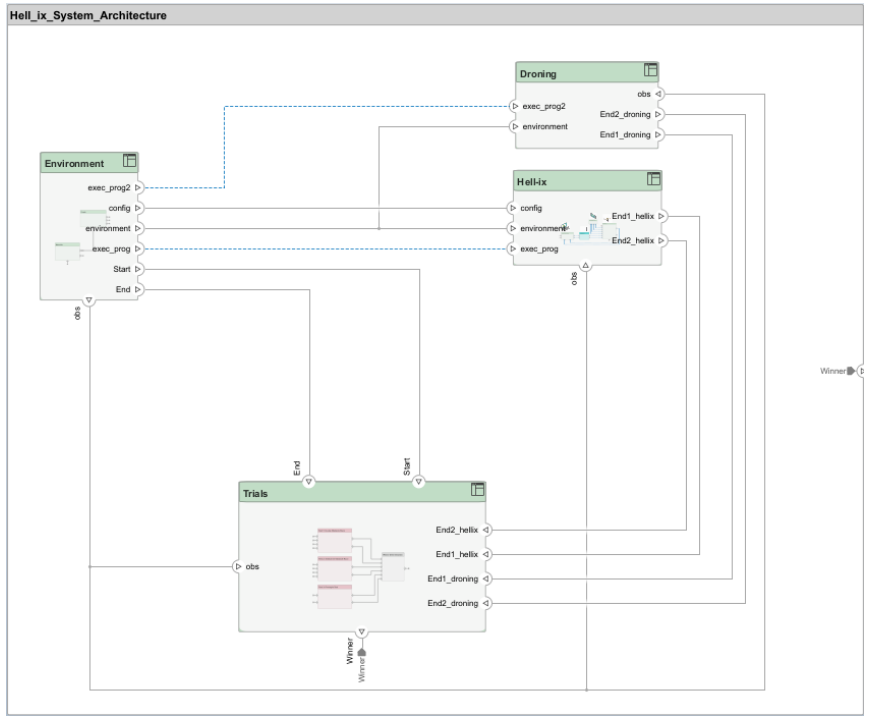
System Architecture

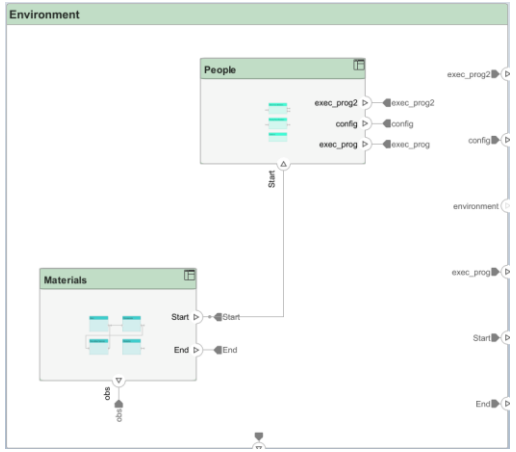


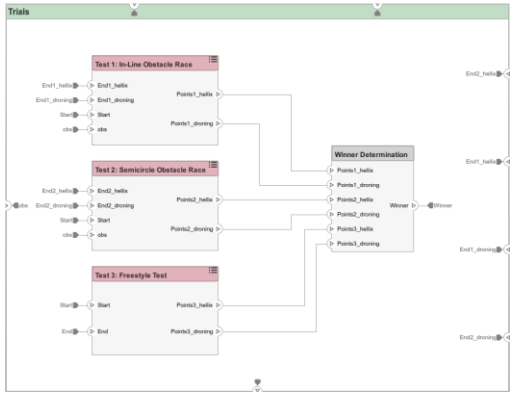
Functional Architecture

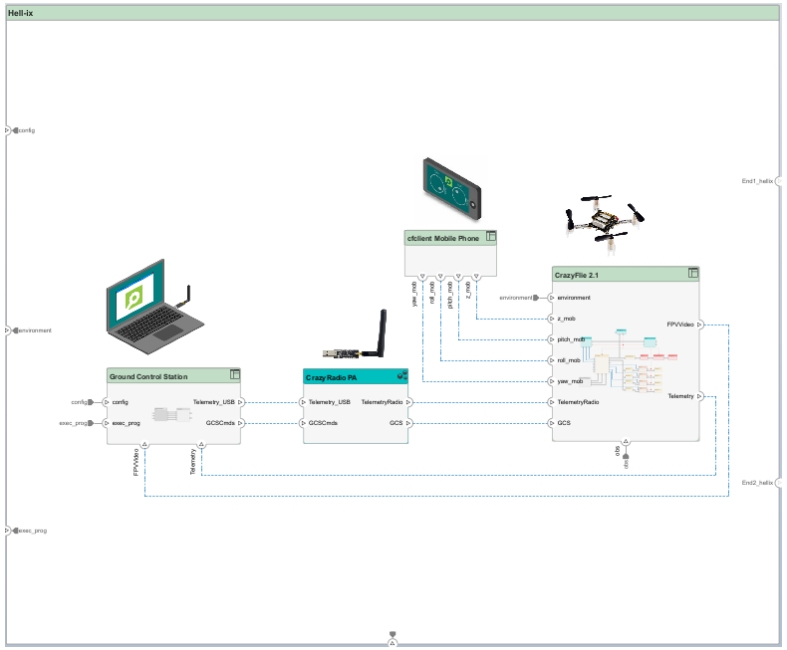
6.6. Modeling Manager Report
Work Done

System Architecture



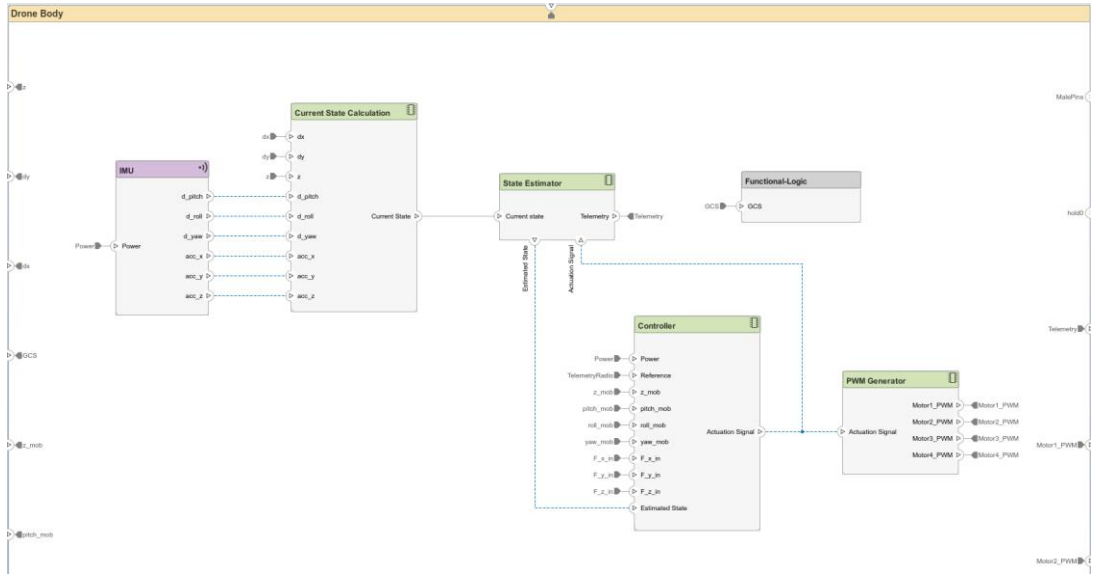


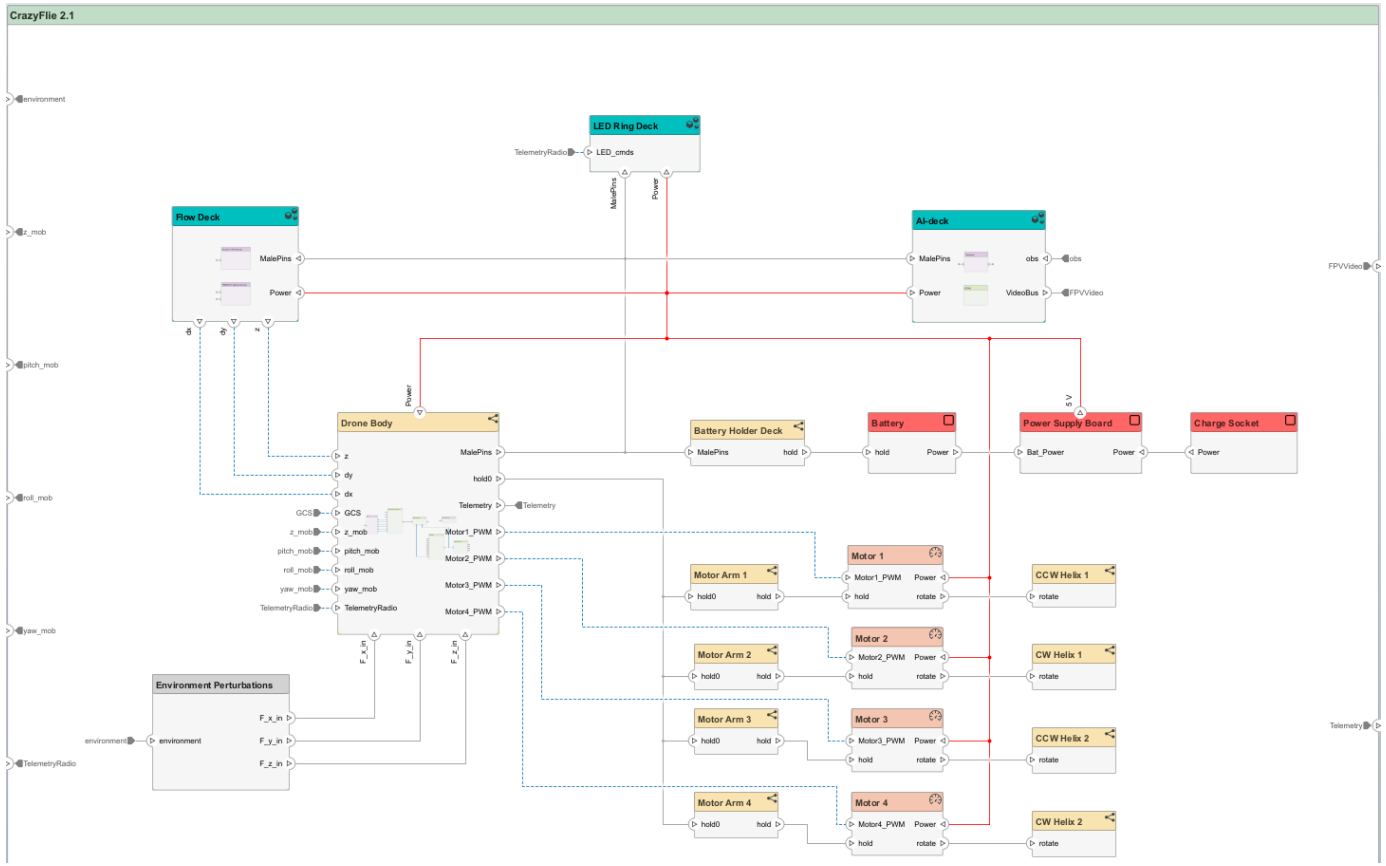




6.6. Modeling Manager Report
Work Done

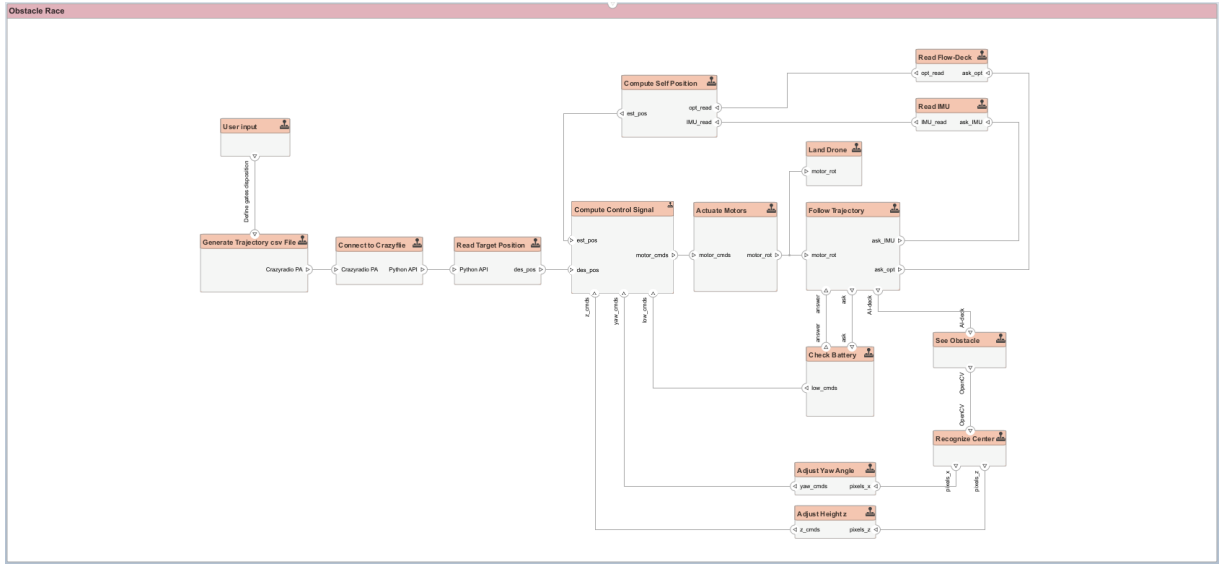
Drone Architecture

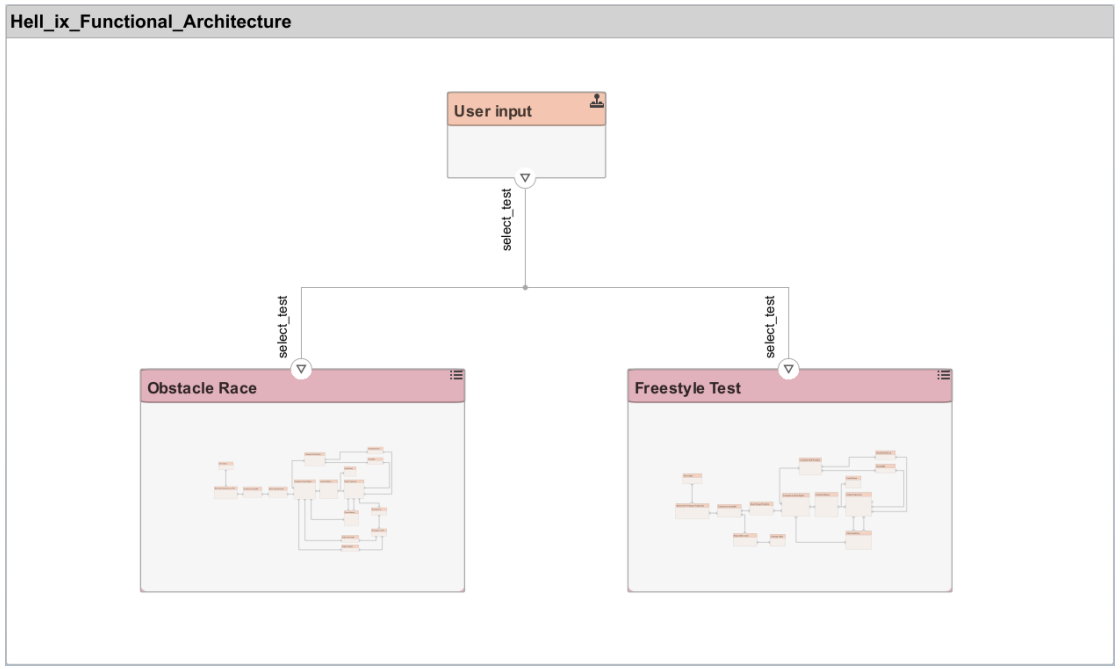




6.6. Modeling Manager Report
Work Done

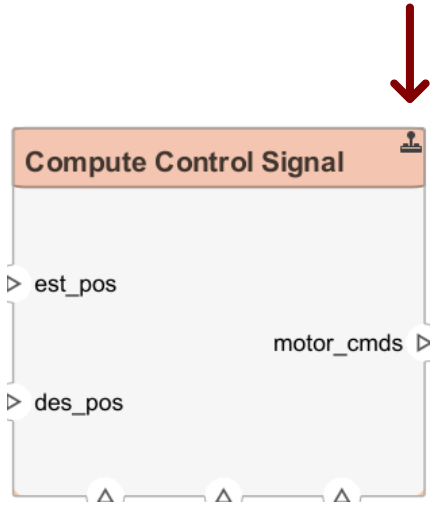
Functional Architecture

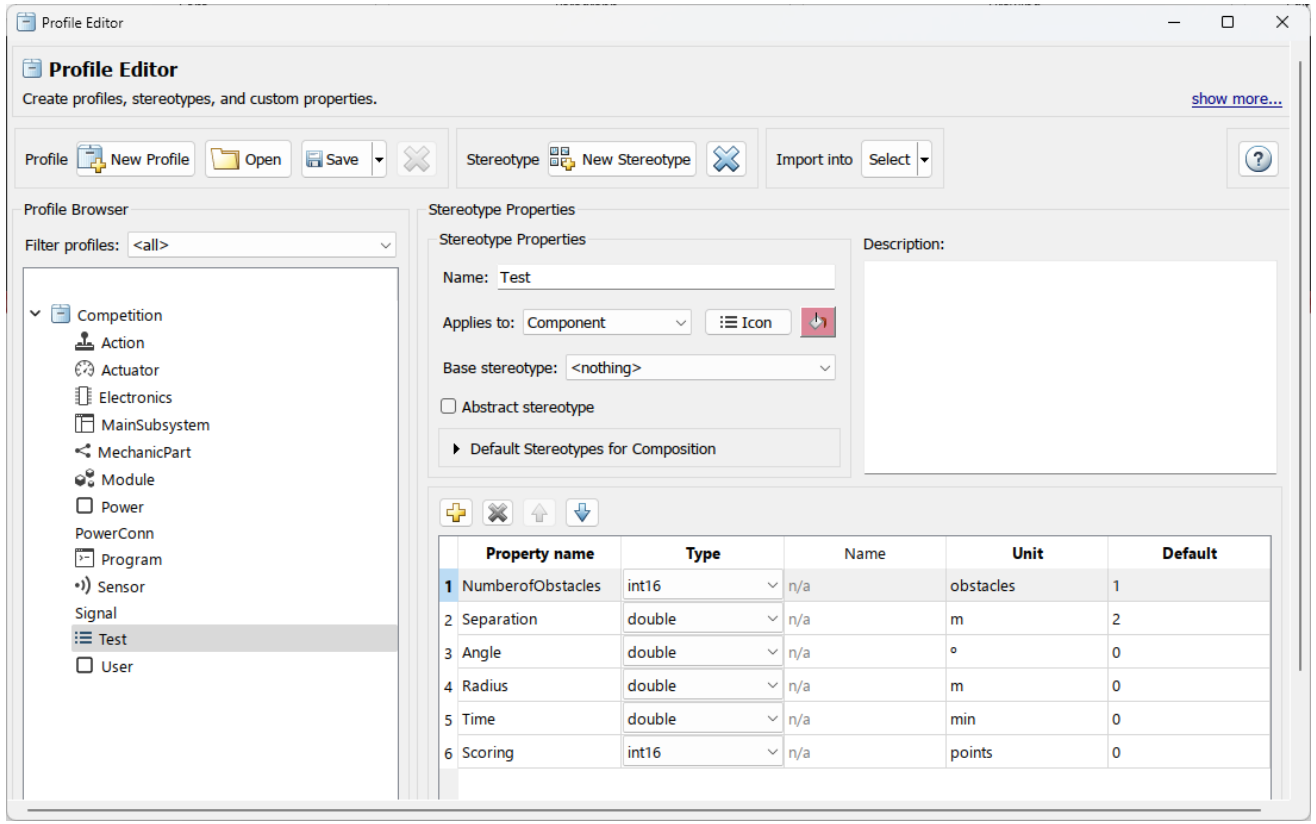




6.6. Modeling Manager Report
Work Done

Stereotypes





6.6. Modeling Manager Report
Work Done

Allocation Matrix

Relations between the components and the functions that are related to their operation.

	Hell_ix_Functional_Arcl	Obstacle Race	Actuate Motors	Compute Contro	Connect to Crazy	Generate Traject	Read Target Pos	Recognize Cent	Read IMU	Check Battery	Adjust Yaw Angl	Read Flow Deck	User input	Adjust Height z	Land Drone	Follow Trajectory	See Obstacle	Compute Self Pc	User input	Freestyle Test	User input	Read IMU	Read Target Pos	Compute Self Pc	Read Flow Deck	Change LEDs	Select the Freest	Read LED cmds	Land Drone	Compute Contro	Check Battery	Follow Trajectory	Actuate Motors	Connect to Crazy
Hell_ix_System_Arc																																		
Droning																																		
Trials																																		
Test 1: In-Lin		↑																																
Test 3: Frees																																		
Test 2: Semic		↑																																
Winner Deter																																		
Hell-ix																																		
CrazyFlie 2.1																																		
LED Ring																										↑		↑						
Motor Arm																																		
Motor 3			↑																														↑	
Battery										↑																					↑			
Motor 4			↑																														↑	
Environm																																		
AI-deck							↑										↑																	
Came																																		
GPA8																																		
Motor 2			↑																														↑	
CW Helix																																		
Battery H																																		
Motor 1			↑																														↑	

6.6. Modeling Manager Report
Work Done

Dynamical Model

Sliding Mode Control: An approach to Control a Quadrotor

Marco Herrera
Facultad de Ingeniería Eléctrica y Electrónica
Escuela Politécnica Nacional-EPN
Quito, Ecuador
marco.herrera@epn.edu.ec,

Alejandro P. Gómez
Universidad de las Fuerzas Armadas
Departamento de Energía Mecánica
Quito, Ecuador
apgomez@espe.edu.ec

William Chamorro
Facultad de Ingeniería Eléctrica y Electrónica
Escuela Politécnica Nacional-EPN
Quito, Ecuador
william.chamorro@epn.edu.ec

Oscar Camacho
Universidad de Los Andes-ULA, Venezuela
Facultad de Ingeniería Eléctrica y Electrónica
Escuela Politécnica Nacional-EPN
Quito, Ecuador
oscar.camacho@epn.edu.ec

***Abstract*— This paper describes the synthesis and application of controller based on sliding mode theory to a quadrotor. A PD sliding surface is considered for vertical take-off and landing aircraft, also changes in angles are done, and some disturbances are included. Therefore, the controller can be implemented using a PD controller as the sliding surface, and adding some algebra the complete controller algorithm is presented. The controller is tested by simulation.**

***Keywords*— Sliding Mode Control, Control theory, UAV's, Nonlinear control.**

I. INTRODUCTION

Quadrotor, also known as Unmanned Aerial Vehicles (UAV), is widely used in several applications, in both military and civilian markets. UAV is nonlinear, unstable and vulnerable to external disturbances. It presents an interesting dynamics where the uncertainties on physical parameters and variable dynamic driving as the drone make it an attractive and challenging control problem. Different methodologies and strategies have been recommended to control a quadrotor. The following approaches have been used: linear quadratic regulator (LQR) control [1], proportional-integral derivative (PID) control [1-4], fuzzy logic (FL) control [5].

Sliding Mode Control (SMC) is a simple robust technique that permits designing controllers for linear and nonlinear processes [6-8]. SMC has become one of the selected solutions for many practical control designs such as electronics devices, robotics, chemical processes and so on [8, 9]. SMC consists in two parts: firstly, a discontinuous control law to enforce the error vector toward a decision rule, called sliding surface, during the reaching phase. This part the control is switching on the different sides of the sliding surface equation and secondly, once the error vector is confined to the sliding surface, an equivalent part of the controller acts to follow the dynamics imposed by the equations describing the sliding surface [8].

There are some works of SMC applied to tracking and regulation tasks for quadrotors, let us make a briefly mention of them, for instances: [10], in this paper is proposed a control scheme based on backstepping and its combination with fuzzy system for chattering reduction. [11], the authors present a controller approach where it is obtained by dividing the design in two steps: firstly, a high gain observer is considered and secondly, a dynamic sliding mode controller (DSMC) is proposed. [12] in this article a direct adaptive sliding mode control is developed. The proposed controller is applied without considering disturbances and parameter uncertainties. The design of the adaptation laws is based on Lyapunov design principle. [13], in this article, backstepping and sliding mode as control schemes are presented and compared. [14] In this paper a controller based on high order sliding mode control (HOSMC) is designed for trajectory tracking. [15] Describes a feedback linearization-based controller with a high-order sliding mode observer. The observer works as an observer and estimator of the external disturbances effect. [16], in this paper; the principal attention is based on the dynamic modelling of quadrotor taking into account the high-order nonholonomic constraints. Using the Backstepping approach for the synthesis of tracking errors and Lyapunov functions, a sliding mode controller is obtained. [17], in this paper an adaptive backstepping approach is utilized to design the sliding mode control.

The objective of this paper consists in designing a sliding mode control and applying it by simulation in a quadrotor. A PD sliding surface is considered for vertical take-off and landing aircraft, also changes in angles are done, and some disturbances are included. Therefore, the controller can be implemented using a PD controller as the sliding surface, and adding some algebra the complete controller algorithm is presented.

This article is structured as follows. In Section II the quadrotor model is presented. Section III, describes concepts

$$X = [x\ \dot{x}\ y\ \dot{y}\ z\ \dot{z}\ \phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi}\ ]$$

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$$

$$U_2 = b(-\Omega_2^2 + \Omega_4^2)$$

$$U_3 = b(-\Omega_1^2 + \Omega_3^2)$$

$$U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)$$

$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$$

$$\ddot{x} = \frac{\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)}{m} U_1$$

$$\ddot{y} = \frac{\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi)}{m} U_1$$

$$\ddot{z} = -g + \frac{\cos(\phi)\cos(\theta)}{m} U_1$$

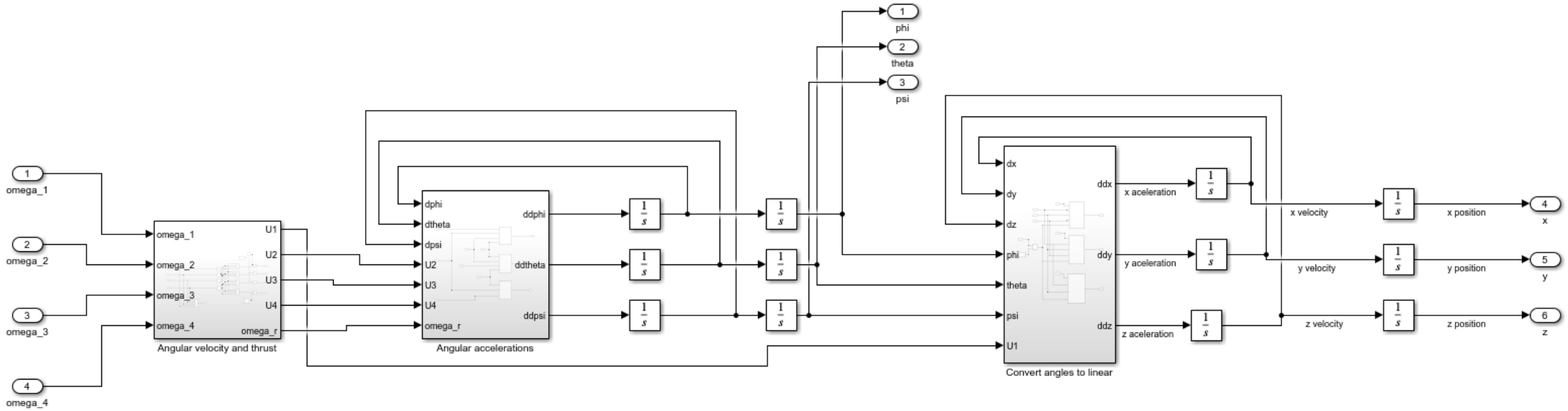
$$\ddot{\phi} = \dot{\phi}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) - \frac{Jr}{I_x}\dot{\theta}\Omega + \frac{l}{I_x}U_2$$

$$\ddot{\theta} = \dot{\phi}\dot{\psi}\left(\frac{I_z - I_x}{I_y}\right) + \frac{Jr}{I_y}\dot{\theta}\Omega + \frac{l}{I_y}U_3$$

$$\ddot{\psi} = \dot{\theta}\dot{\phi}\left(\frac{I_x - I_y}{I_z}\right) + \frac{1}{I_z}U_4$$

6.6. Modeling Manager Report
Work Done

Dynamical Model



6.6. Modeling Manager Report
Work Done

Dynamical Model

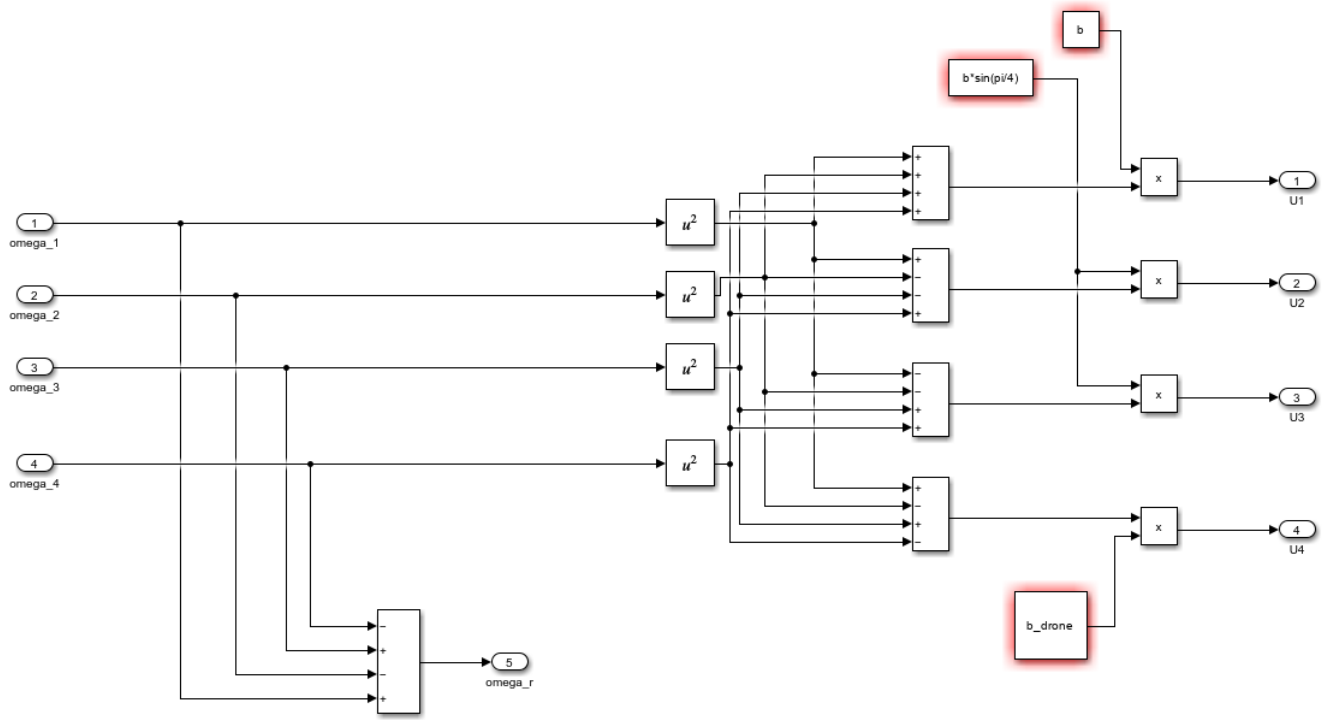
$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$$

$$U_2 = b(-\Omega_2^2 + \Omega_4^2)$$

$$U_3 = b(-\Omega_1^2 + \Omega_3^2)$$

$$U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)$$

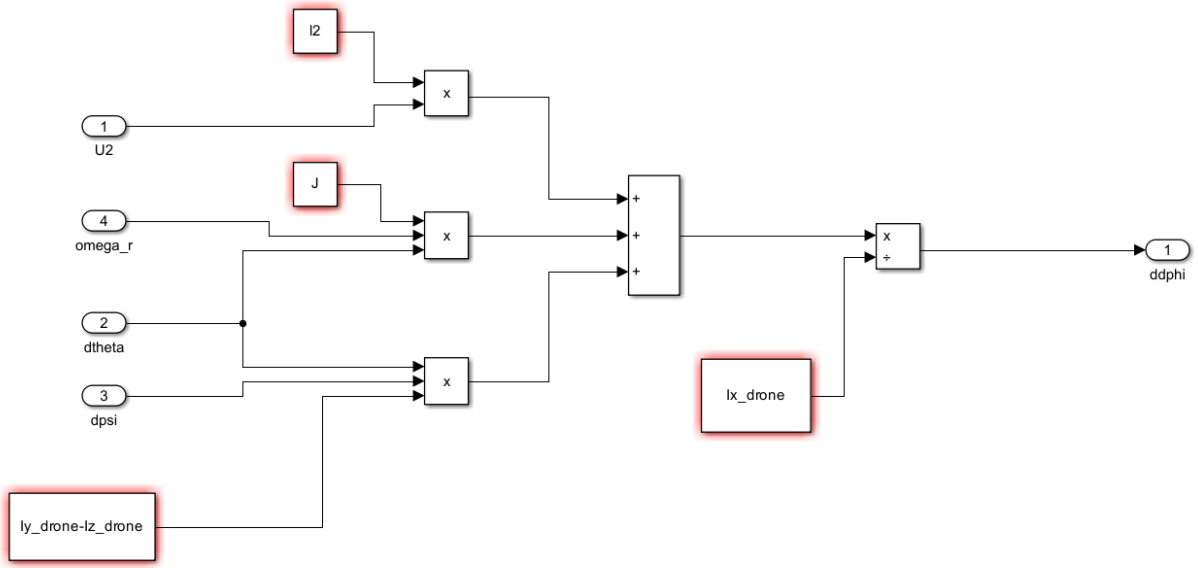
$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$$



6.6. Modeling Manager Report
Work Done

Dynamical Model

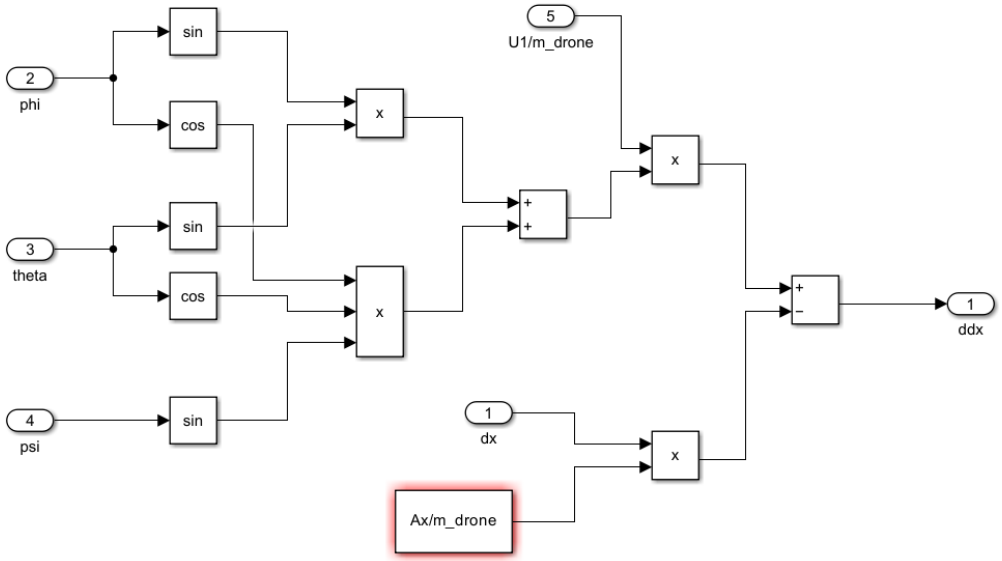
$$\ddot{\phi} = \dot{\phi}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) - \frac{Jr}{I_x}\dot{\theta}.\Omega + \frac{l}{I_x}U_2$$



6.6. Modeling Manager Report
Work Done

Dynamical Model

$$\ddot{x} = \frac{\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)}{m} U_1$$



6.6. Modeling Manager Report
Analysis

Dynamical Model

- The model can capture the dynamics of the drone, but it is not feasible in real life, as filtering and state estimation are required, which are highly complex for the purpose of simulation.

- The Crazyflie already has a working controller, dedicating such effort to recreate the controller would not be reasonable unless we wanted to study new control techniques, but that is out of the scope of the Project.

- Simulation times were extremely high, making it difficult to test the model.

6.6. Modeling Manager Report
Analysis

System Model

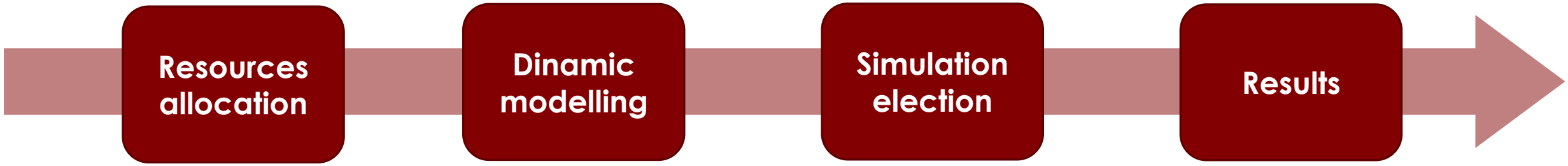
- The model is exhaustive in respect to the components and the relations between them. Making a deeper model without making use of the other System Composer features would be a waste of our time.
- The classification of the blocks allows to rapidly identify stereotypes and their properties.
- The allocation matrix while being complex to make, allows to have a traceability of the relations between system and functional architectures.

6.6. Modeling Manager Report
Analysis

Requirements

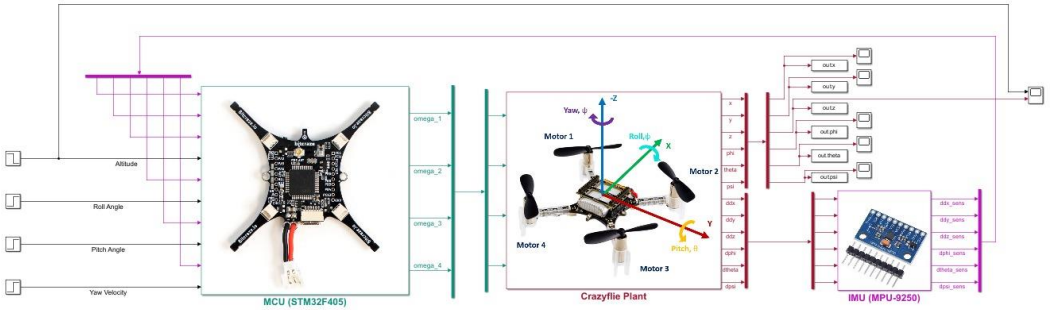
- Using Matlab to define the requirements of the project allows to integrate them easily with the model.
- Links to the blocks allow to have control over which changes should be performed if a requirement is altered during development.
- Automatic documentation accelerates the process of generating reports.

6.7. Simulation Manager Report
Planned Work



6.7. Simulation Manager Report
Work Done

Dinamic Model



Simulation election

GAZEBO

Webots™
robot simulation

Gazebo

Just mind that this model does not fly properly yet.

1- First install gazebo garden: https://ignitionrobotics.org/docs/garden/install

2- Clone this repo: git clone https://github.com/bitcraze/crazyflie_simulation.git

3- Put this repo in your ~/.bashrc and source it in your terminal

```
export IGN_GAZEBO_RESOURCE_PATH="path/to/crazyflie-simulation/gazebo-ignition/"
```

6.7. Simulation Manager Report
Work Done

Tools Used

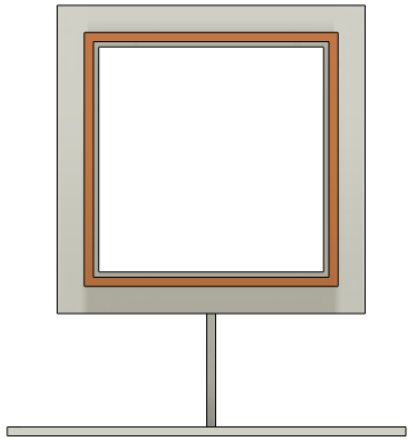
F 360

Webots™
robot simulation

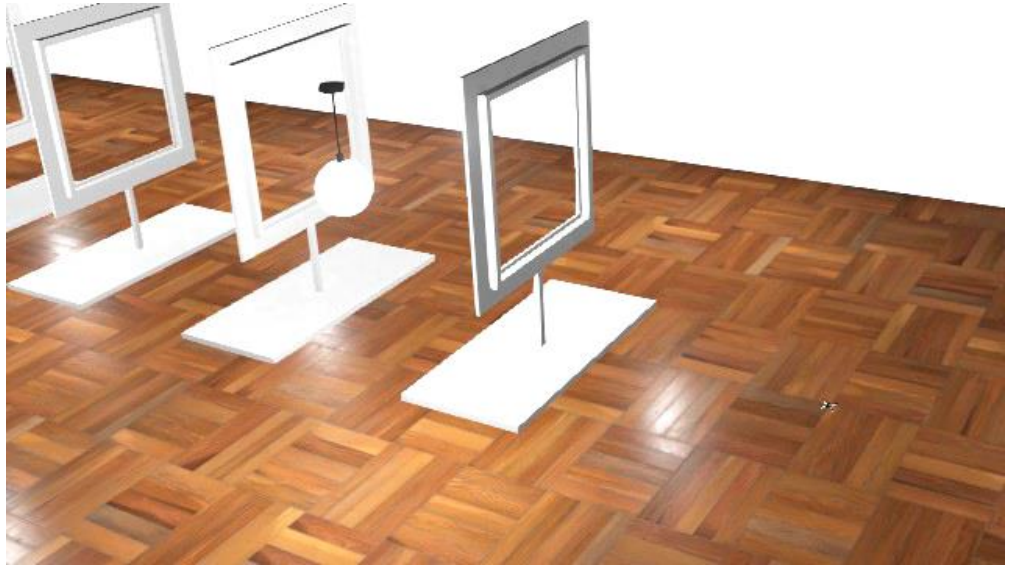
CSV

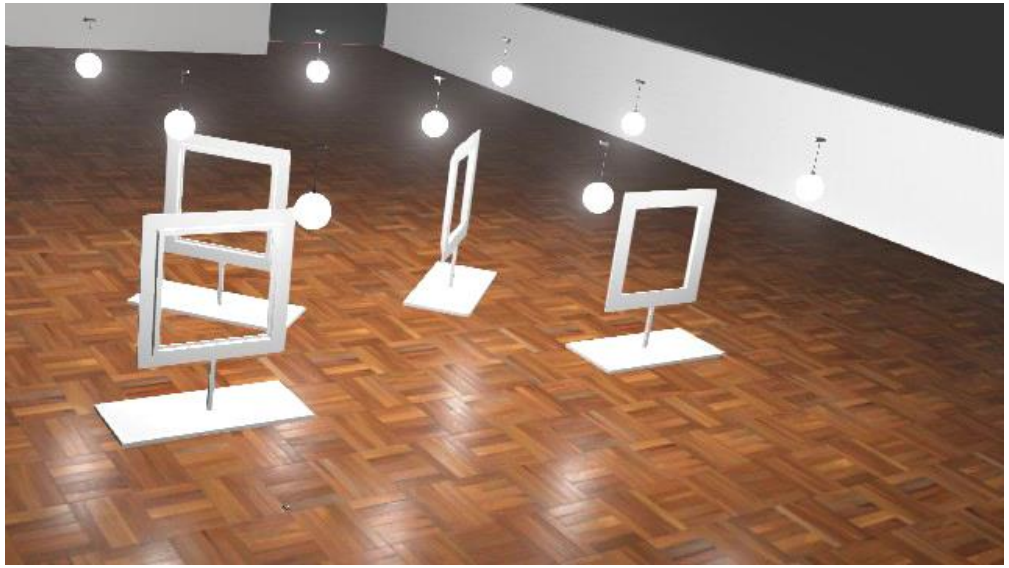
SIMULINK

Designed elements



6.7. Simulation Manager Report
Work Done





6.8. Programming Manager (Python) Report
Planned Work

Goals:

Testing code

- Design tests to verify correct operation
- Study the response of the drone to the change of parameters (waiting and processing time, speed,...)
- Check the correct operation of functions separately

Competition code

- Develop a code capable of meeting the requirements of the competition
- Identify problems and possible improvements

Freestyle code

- Develop programs in which the drone describes trajectories with recognizable shapes
- Design programs that make use of other modules (leds)

6.8. Programming Manager (Python) Report
Planned Work

Connections with other departments:

We depend on:

- *Competition delegation*
- *Modeling and simulation department*
- *Tooling department*

Programming Department

Depend on us:

- *Test department*
- *Implementation department*

6.8. Programming Manager (Python) Report
Work Done

Programs developed:

Testing code

- up_and_down.py
- move_forward.py

Competition code

- csv_follower.py
- csv_follower_vision.py

Freestyle code

- freestyle_star.py
- freestyle_spiral.py

6.8. Programming Manager (Python) Report
Work Done

Tools:

IDE

Programming language

- *Math*
- *Time*
- *Logging*
- *Cmath*
- *Csv*

bitcraze/**crazyflie-lib-python**

Python library to communicate with Crazyflie

OpenCV

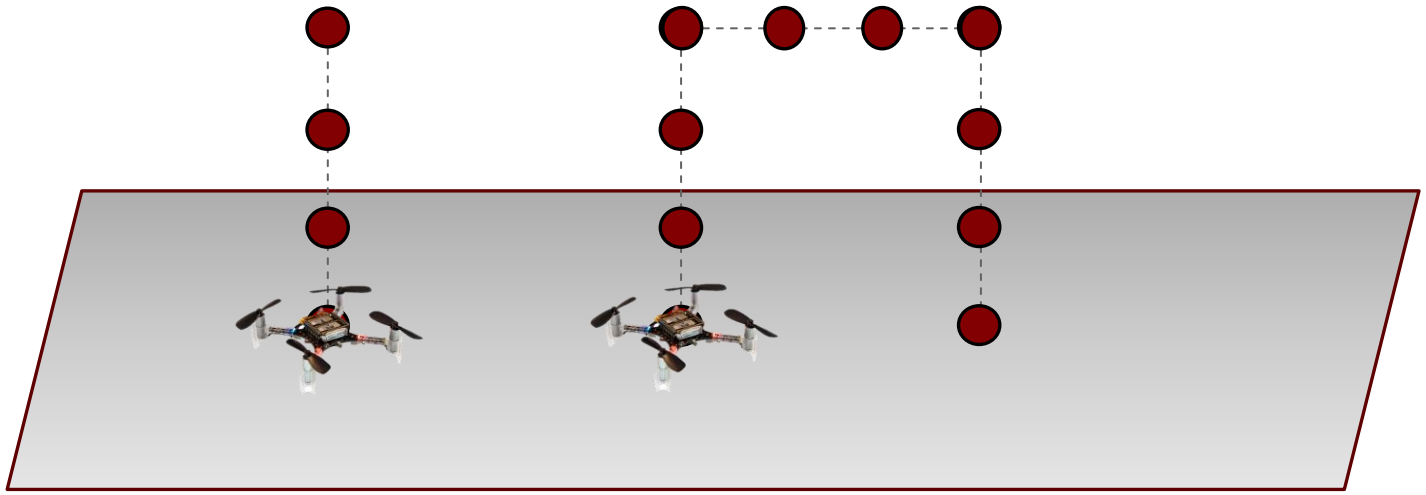
Other libraries

6.8. Programming Manager (Python) Report
Work Done

Programs developed:
Testing code

- up_and_down.py
- move_forward.py

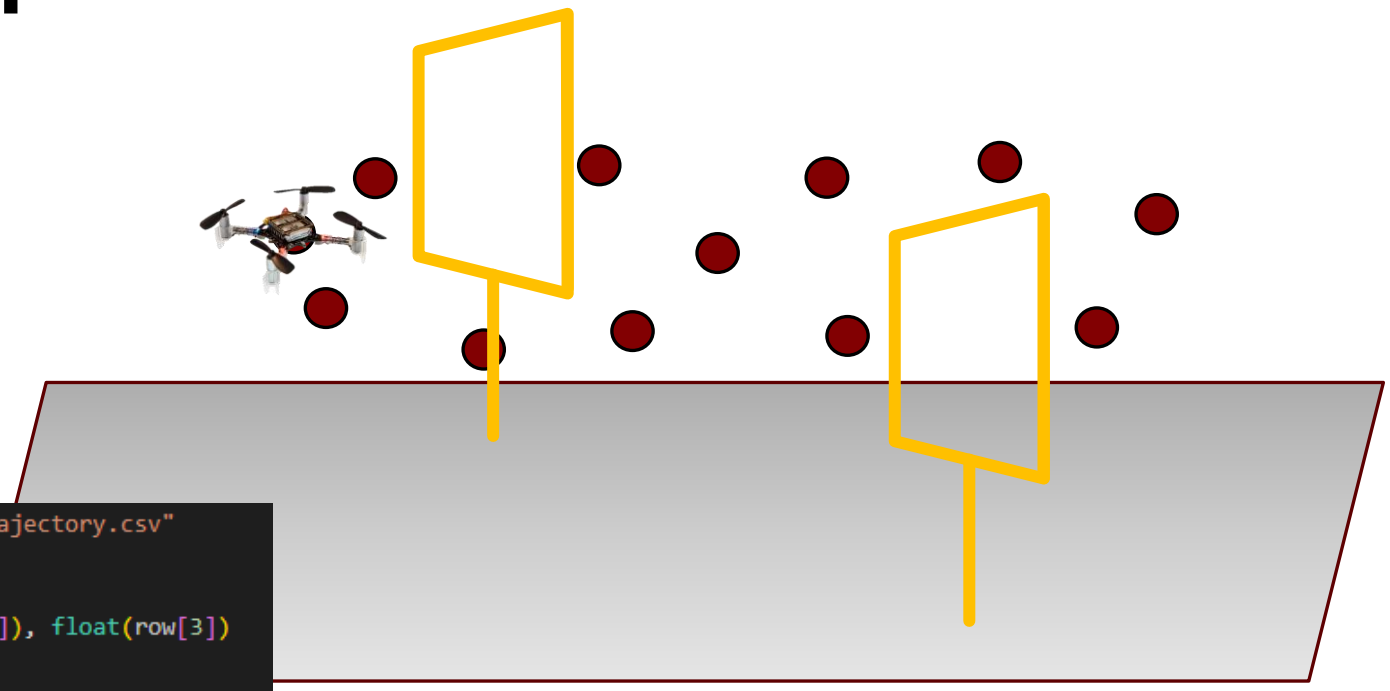
send_hover_setpoint(vx, vy, yawrate, z)



6.8. Programming Manager (Python) Report
Work Done

Programs developed:
Competition code

- **csv_follower.py**
- csv_follower_vision.py



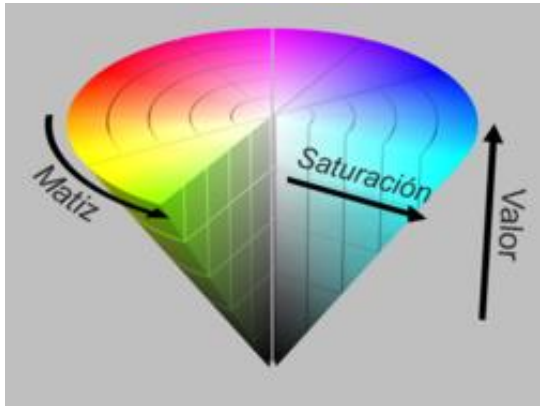
```python
csv_file_path = "C:/Users/Usuario/Desktop/VUELO CRAZYFLIE/trajectory.csv"
csv_reader = csv.reader(open(csv_file_path, 'r'))
for row in csv_reader:
    x, y, z, yaw = float(row[0]), float(row[1]), float(row[2]), float(row[3])
    cf.commander.send_position_setpoint(y, -x, z, yaw)
    time.sleep(0.02)
```

6.8. Programming Manager (Python) Report
Work Done

Programs developed:
Competition code

- csv_follower.py
- **csv_follower_vision.py**

RGB



HSV

6.8. Programming Manager (Python) Report
Work Done

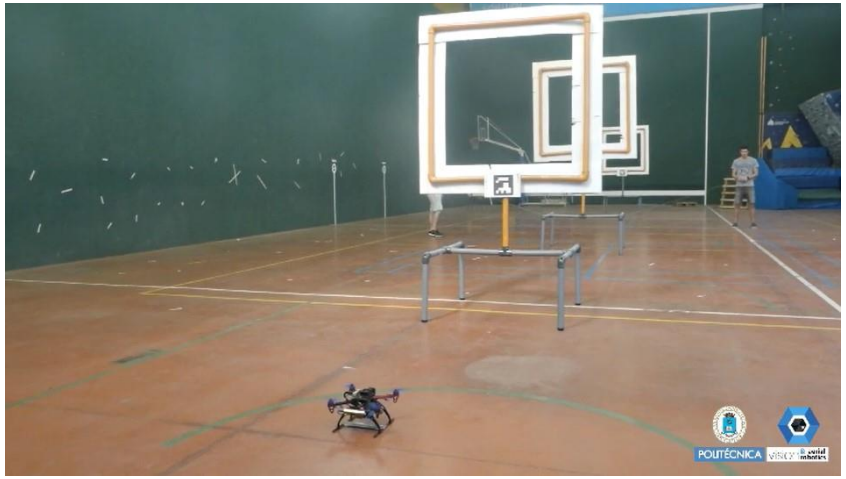
Programs developed:
Competition code

- csv_follower.py
- **csv_follower_vision.py**

6.8. Programming Manager (Python) Report
Work Done

Programs developed:
Competition code

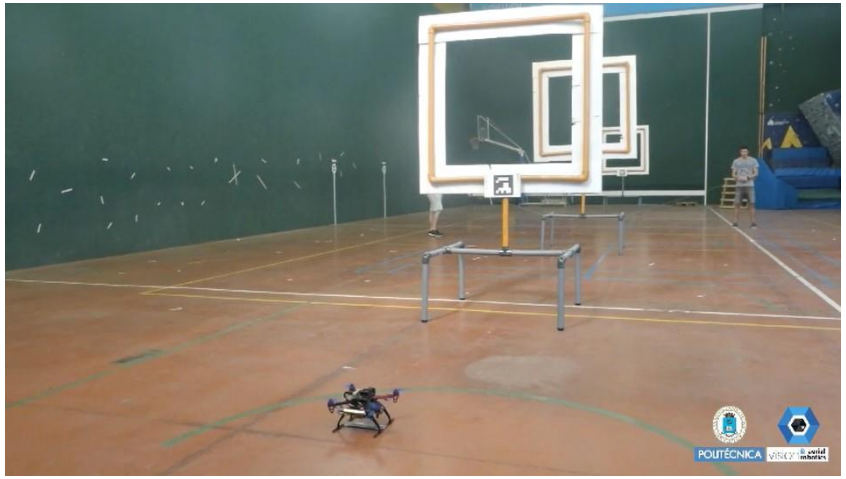
- csv_follower.py
- **csv_follower_vision.py**

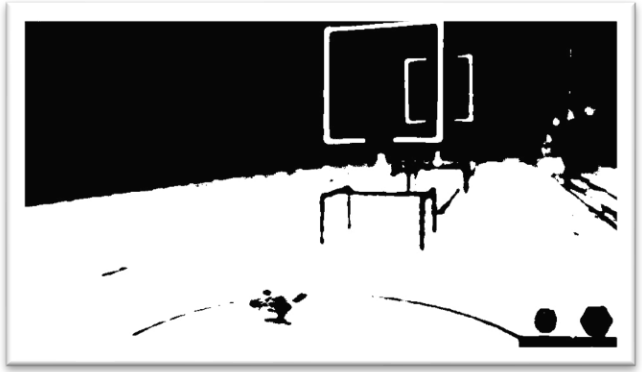


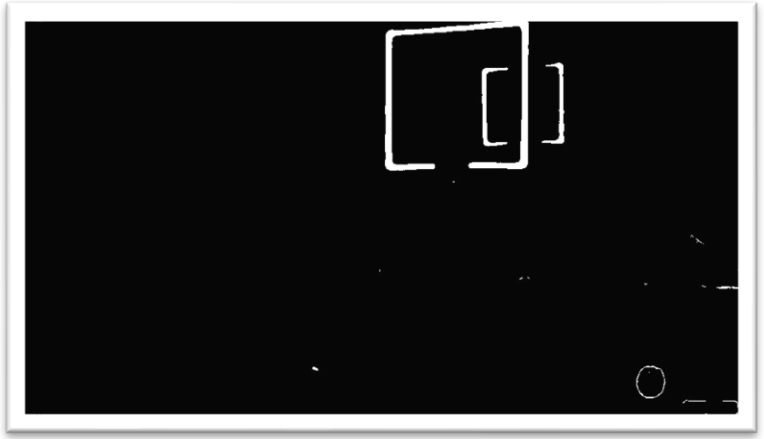
6.8. Programming Manager (Python) Report
Work Done

Programs developed:
Competition code

- csv_follower.py
- **csv_follower_vision.py**



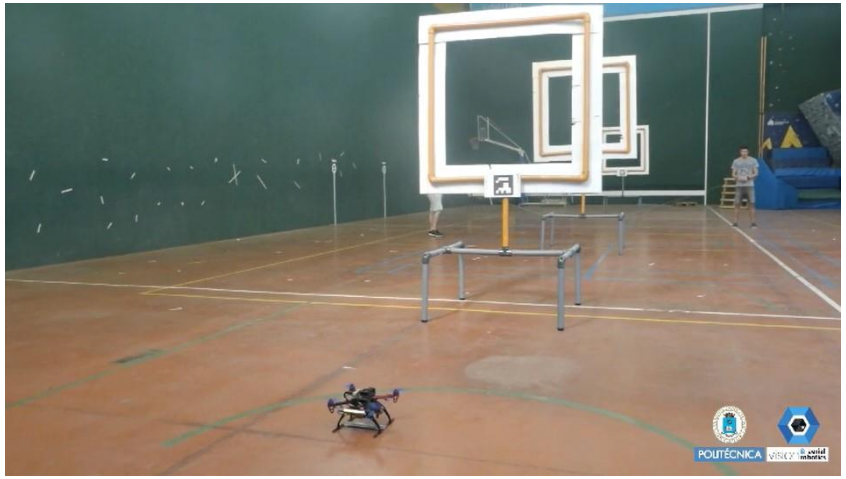


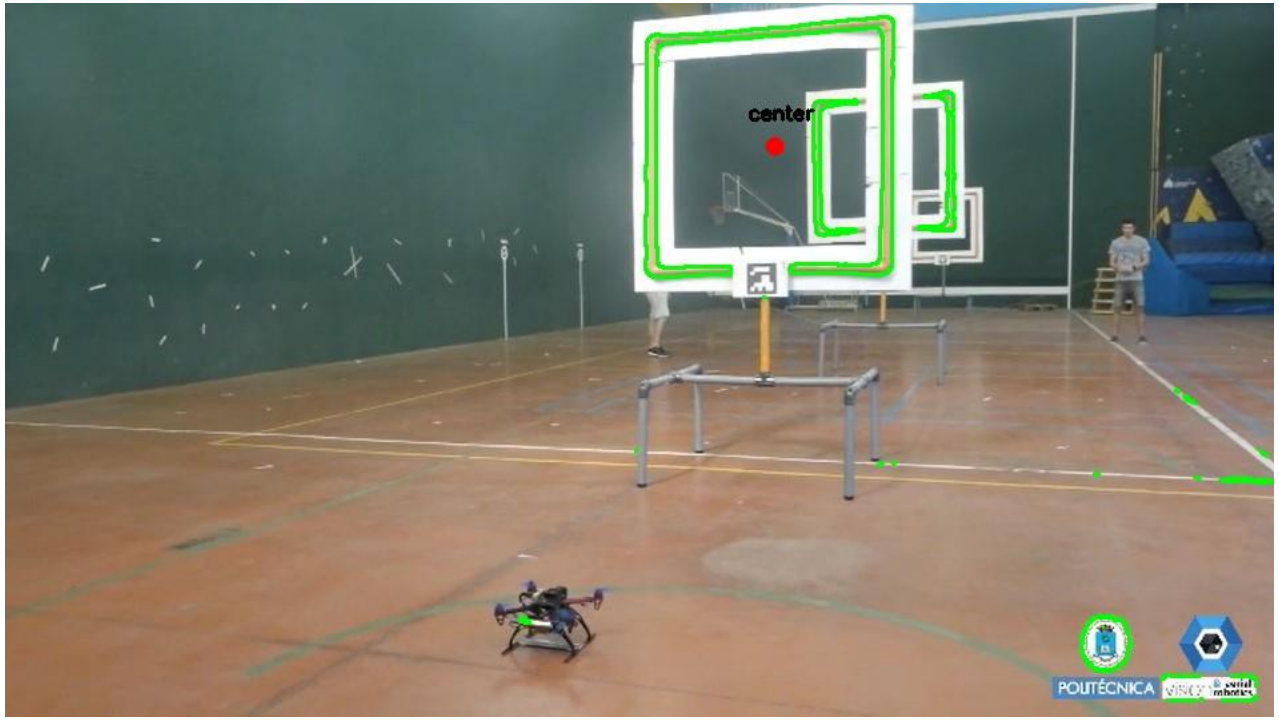


6.8. Programming Manager (Python) Report
Work Done

Programs developed:
Competition code

- csv_follower.py
- **csv_follower_vision.py**

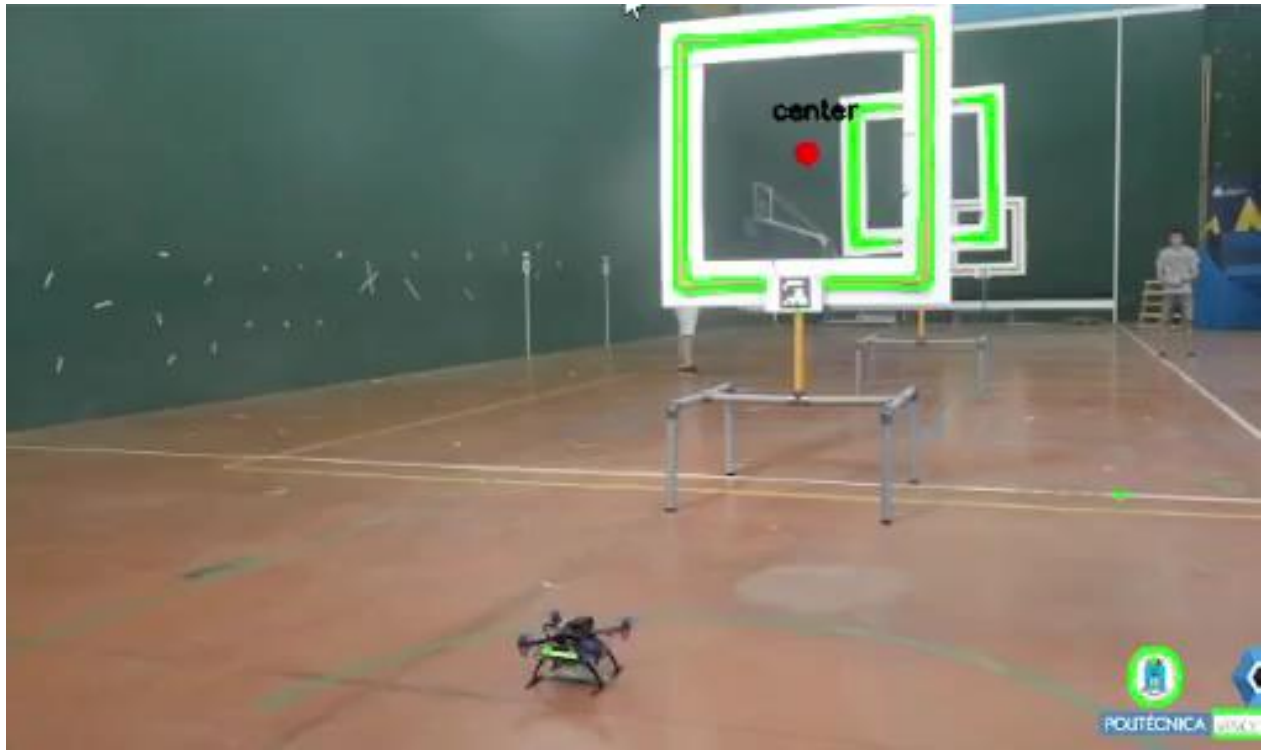




6.8. Programming Manager (Python) Report
Work Done

Programs developed:
Competition code

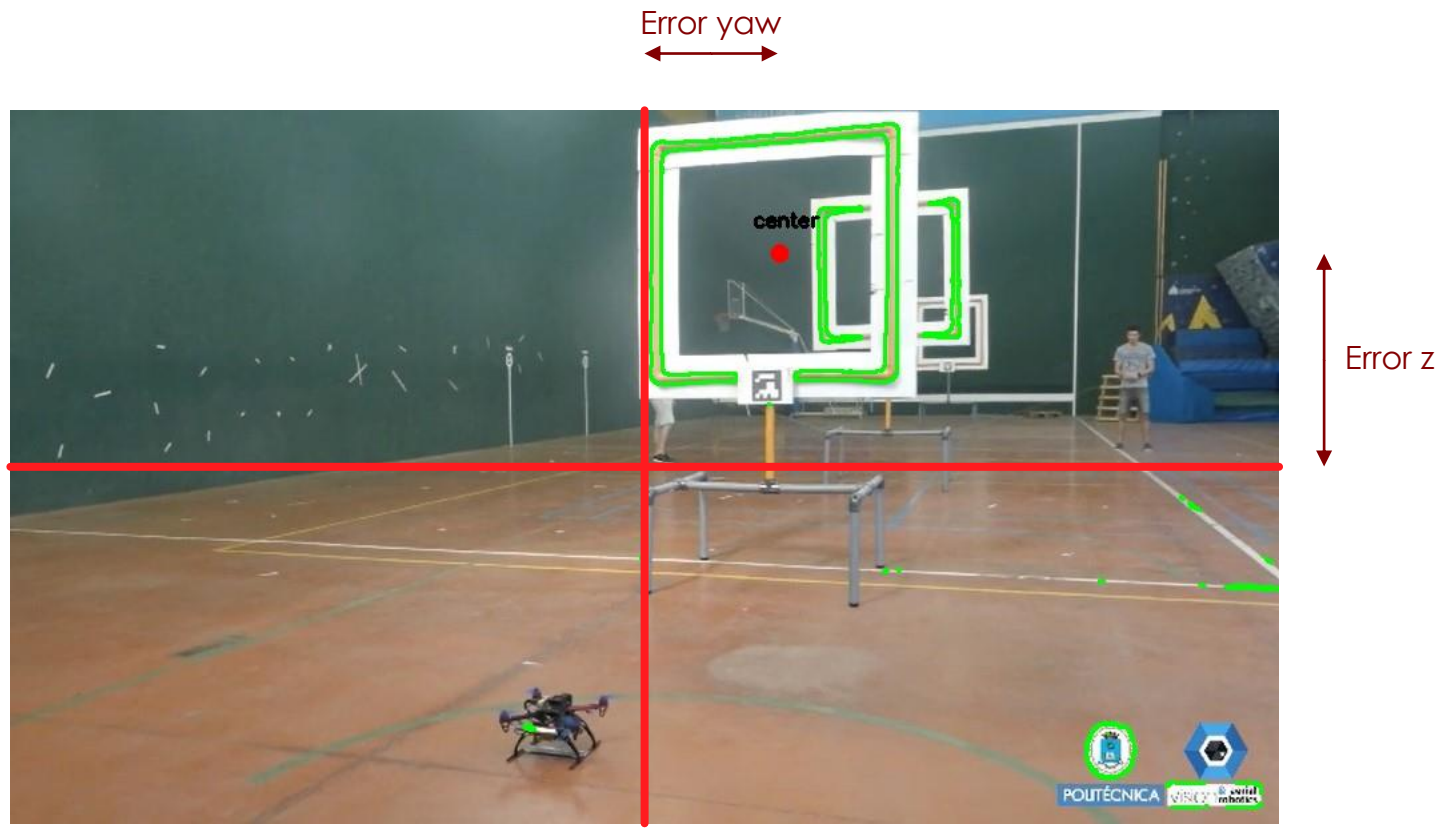
- csv_follower.py
- **csv_follower_vision.py**



6.8. Programming Manager (Python) Report
Work Done

Programs developed:
Competition code

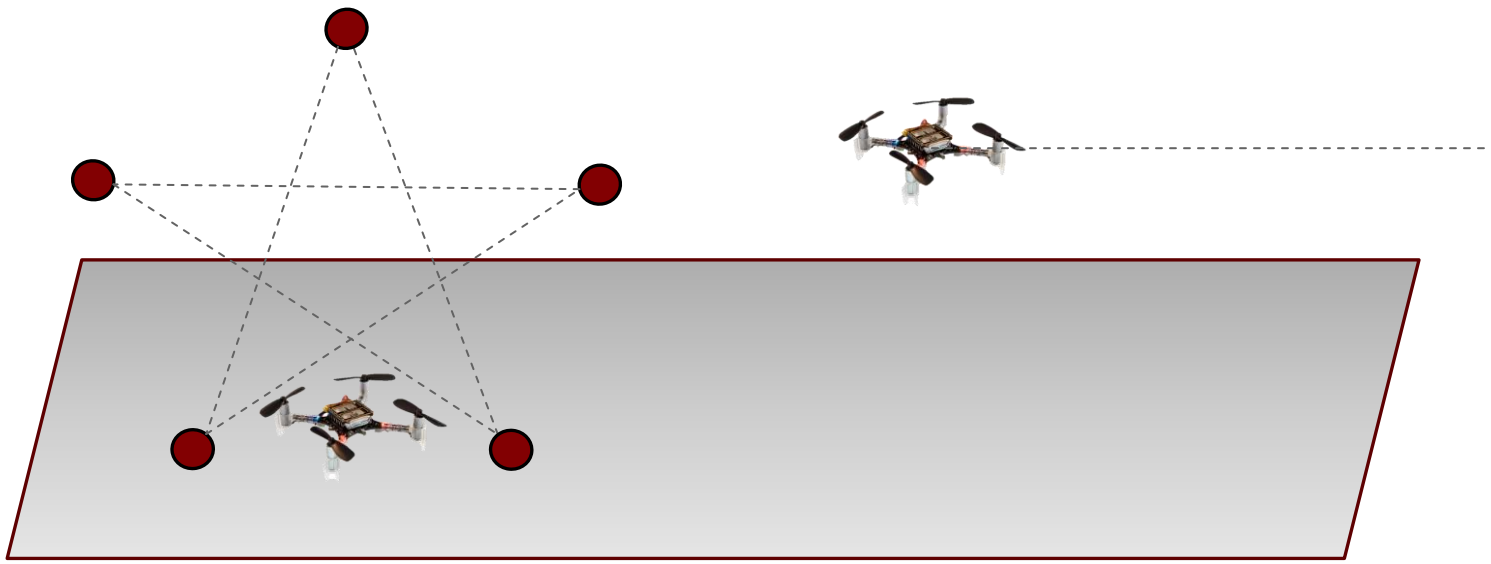
- csv_follower.py
- **csv_follower_vision.py**



6.8. Programming Manager (Python) Report
Work Done

Programs developed:
Freestyle code

- freestyle_spiral.py
- freestyle_star.py



6.8. Programming Manager (Python) Report
Analysis

Problems detected in the process:

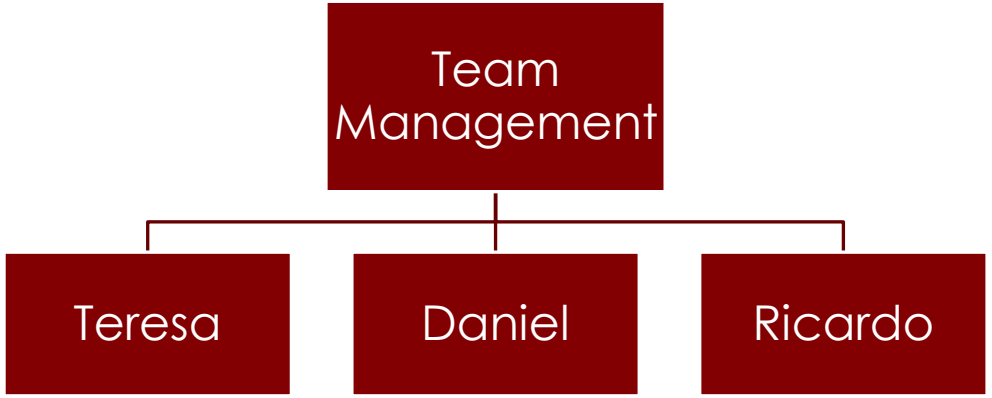
- Height detection error when crossing an obstacle
- It is impossible to connect the LED module at the same time as the Flowdeck

Possible future improvements:

- Create new trajectories for the freestyle test

6.9. Programming Manager (MATLAB) Report

Planned Work



6.9. Programming Manager (MATLAB) Report
Work Done

```
% Normalize the vector in case it is not already normalized
vector_2D = [vector(1), vector(2)];
normal_vector = vector_2D/norm(vector_2D);

% Calculate the perpendicular vector to the one normal to the surface
perp_vector = [-normal_vector(2), normal_vector(1)];

% Calculate the four corners of the square gate
corners = [[center(1:2) + side/2*perp_vector, center(3) - side/2];
           [center(1:2) - side/2*perp_vector, center(3) - side/2];
           [center(1:2) - side/2*perp_vector, center(3) + side/2];
           [center(1:2) + side/2*perp_vector, center(3) + side/2]];

% Plot the four corners, as well as the union of the gate to the ground
for i=1:4
    plot3([corners(i, 1), corners(mod(i, 4) + 1, 1)], ...
          [corners(i, 2), corners(mod(i, 4) + 1, 2)], ...
          [corners(i, 3), corners(mod(i, 4) + 1, 3)], 'w-', ...
          'LineWidth', 2);
end

union = [mean([corners(1,1), corners(2,1)]), ...
         mean([corners(1,2), corners(2,2)]), corners(1, 3)];

plot3([union(1), union(1)], [union(2), union(2)], [union(3), 0], ...
      'w-', 'LineWidth', 2);

end
```

plotGate

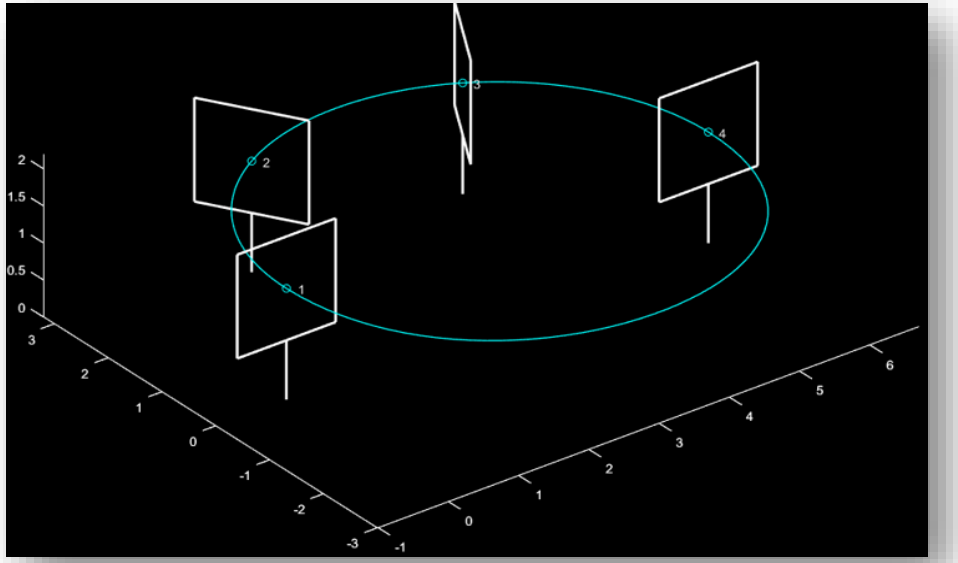
```
% Create a global variable to store the 3D coordinates list
global trajectory;

% Specify the number of gates to pass:
N = 4;
trajectory = zeros(50 * N, 3);

% Create a figure to plot the obtained trajectories and the gates
fig = figure(1);
    set(fig, 'color', [0 0 0], 'name', '3D spline example');
    clf;
    hold on;
    axis equal;
    ax = gca;
    ax.Color = 'black';
    ax.XColor = 'white';
    ax.YColor = 'white';
    ax.ZColor = 'white';
    ax.XLim = [-1, inf];
    ax.YLim = [-3, inf];
    ax.ZLim = [0, inf];
    view(3);

% Specification of the four points where the gates' centers are located.
% They can be modified if the points do not define a circle.

R = 3;

gate_loc = [[  -R*cos(deg2rad(0))+R,   R*sin(deg2rad(0)), 1.5]; ...
            [ -R*cos(deg2rad(60))+R,  R*sin(deg2rad(60)), 1.5]; ...
            [-R*cos(deg2rad(120))+R, R*sin(deg2rad(120)), 1.5]; ...
            [-R*cos(deg2rad(180))+R, R*sin(deg2rad(180)), 1.5]];

% Specification of the velocity vectors of the drone when passing through
% the center of the gates
vel_vec = [[0, 1, 0]; [sqrt(3)/2, 0.5, 0]; [sqrt(3)/2, -0.5, 0];
           [0, -1, 0]];

% Specification of the tension (ratio power-to-weight) of the drone
tension = 1;
```

```
% Specification of the dimensions of the side of the square gate
side = 1.4;

specs = cell(1, N);

if (length(gate_loc) == N) && (length(vel_vec) == N)
    for i = 1:N
        specs{i} = {gate_loc(i, :), vel_vec(i, :), tension};
        % Plot the gates through which the drone has to pass
        plotGate(gate_loc(i, :), vel_vec(i, :), side);
    end
else
    error("Revise that you have correctly defined positions" + ...
        " and velocities for all the gates.");
end

% Plot the trajectory according to the given specifications
hobbysplines(specs, 'debug', true, 'cycle', true, 'color', 'red');

x = trajectory(:, 1);
y = trajectory(:, 2);
trajectory(:, 4) = zeros(1, length(x));
for i = 2:length(x) - 1
    dx = x(i + 1) - x(i - 1);
    dy = y(i + 1) - y(i - 1);
    trajectory(i, 4) = -rad2deg(atan2(dx, dy));
end

writematrix(trajectory,'trajectory.csv')
```

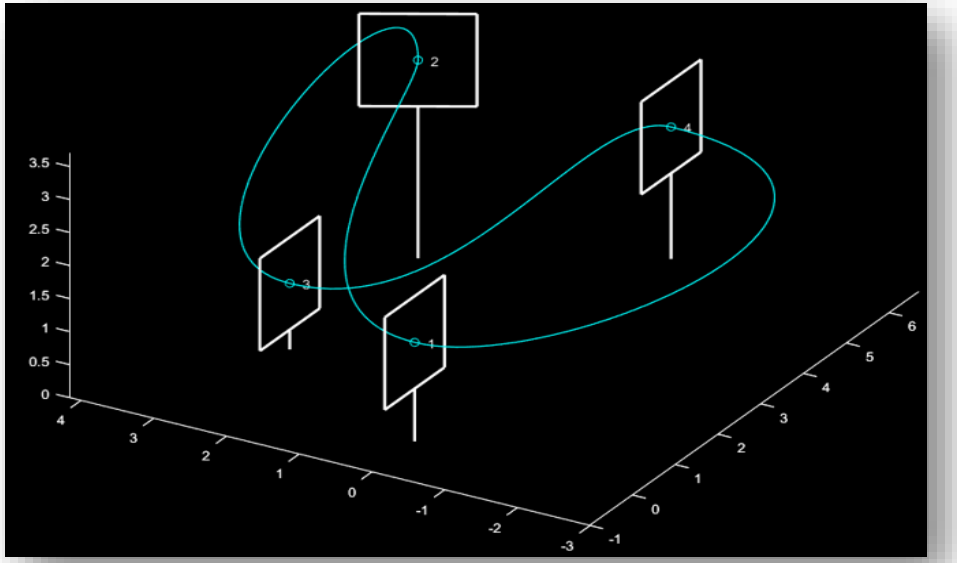
Trajectory_planner

6.9. Programming Manager (MATLAB) Report

Work Done

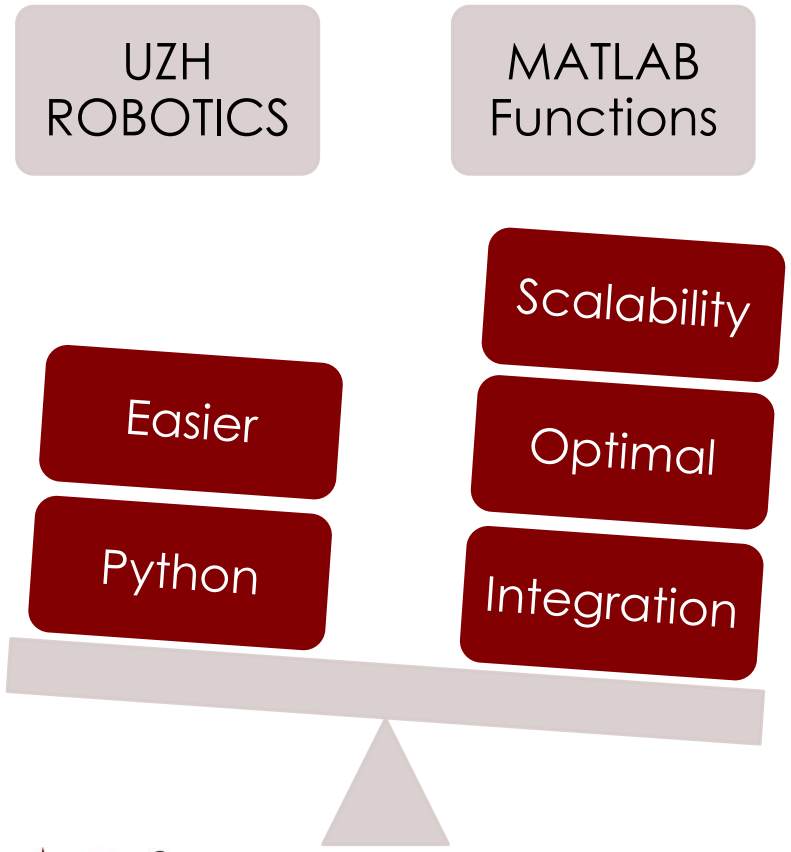


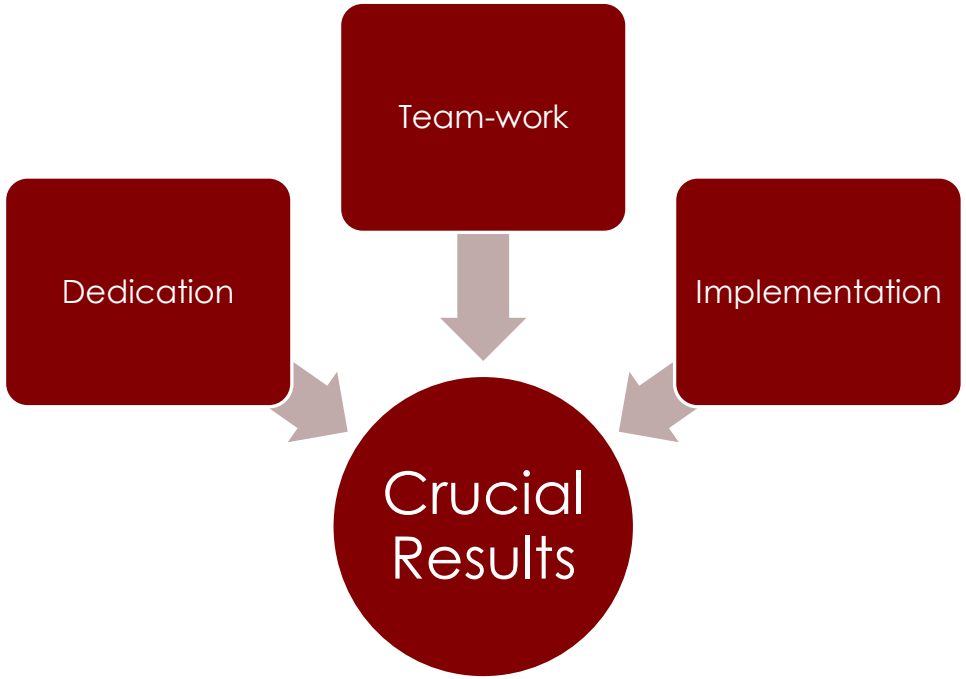
Semi-circular Loop



Alternative Trajectory

6.9. Programming Manager (MATLAB) Report
Analysis





6.10. Implementation Manager Report
Planned Work

INITIAL TASKS

- Team communication
- Requeriments analysis
- Shopping list

GITHUB TASKS

- Simulation
- Software

TECHNICAL TASKS

- Drone

DOCUMENTATION TASKS

- User manual
- Maintenance and Operations Manual

6.10. Implementation Manager Report
Work Done

INITIAL TASKS

6.10. Implementation Manager Report
Work Done

GITHUB TASKS

- Simulation
  - webots
  - README.md

User manual.

Files required for the simulation.

- Software
  - Freestyle Test
  - Path Follower Assisted by Comput...
  - Race Trajectory Planner
  - Tests
  - README.md

Files required to fly the drone.

Files required for the tests.

User manual.

6.10. Implementation Manager Report
Work Done

TECHNICAL TASKS

- AI-deck
- Led-Ring
- Flow deck

DOCUMENTATION TASKS

User manual

Connect drone to the PC → Run the code to fly the drone

Maintenance and operating Manual

- Replace parts of the drone
- Advices
- Errors

6.10. Implementation Manager Report
Issues

Buying issues

CAR	Buying order w/o Crazyradio PA	Delivery delay (1 month)	Limited implementation time

Causes

- Lack of stock
- High demand
- Delay in hardware acquisition

Consequences

- Possibility of switching to the Tello drone.
- Very limited time for implementation.

6.10. Implementation Manager Report
Issues

AI-deck issues

AI-deck version C	Cable Olimex ARM-USB-TINY-H JTAG required	CAR	Blind drone

Causes

- Outdated AI-deck module (C)
- Limited implementation time

Consequences

- Installation stuck at 99%.
- Can't create a Wifi red to send images.
- Unable to test physically.

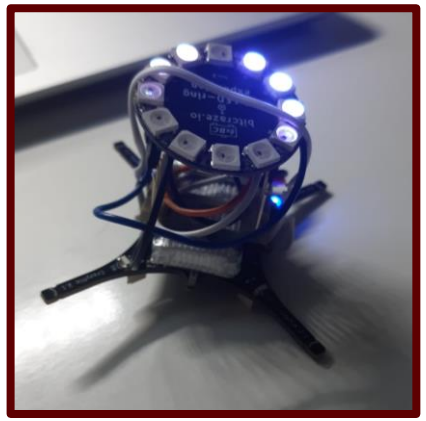
6.10. Implementation Manager Report
Issues

Led-ring issues

Mounting place

Arduino cables (CEI)

Soldering required



Causes

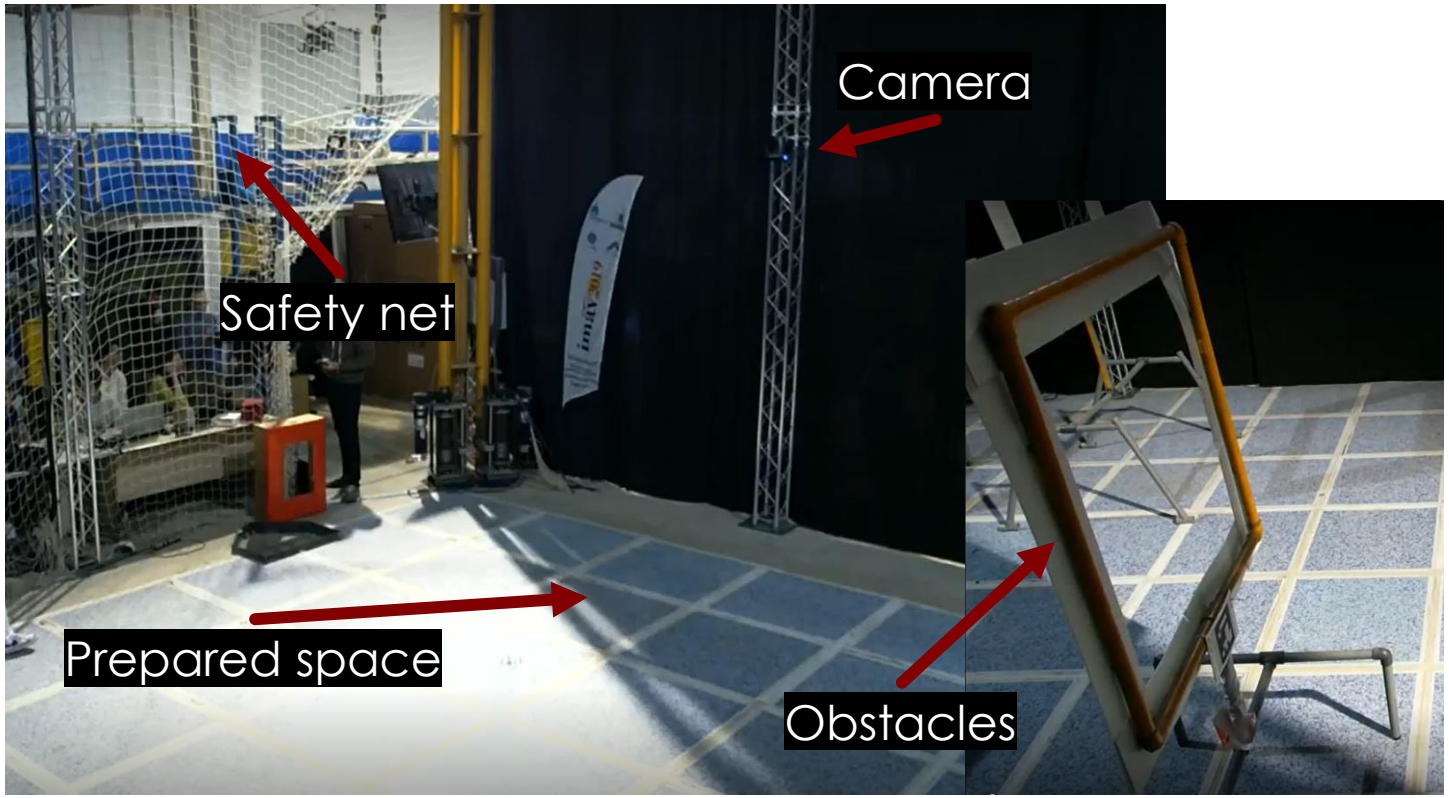
- Led-ring mounting place
- Cable size
- Common pin (Flow-deck and Led-ring)

Consequences

- Simultaneous mounting of the flow-deck and the LED-ring is not possible.
- Uncontrollable without the flow-deck.

6.11. Test Manager Report
Planned Work

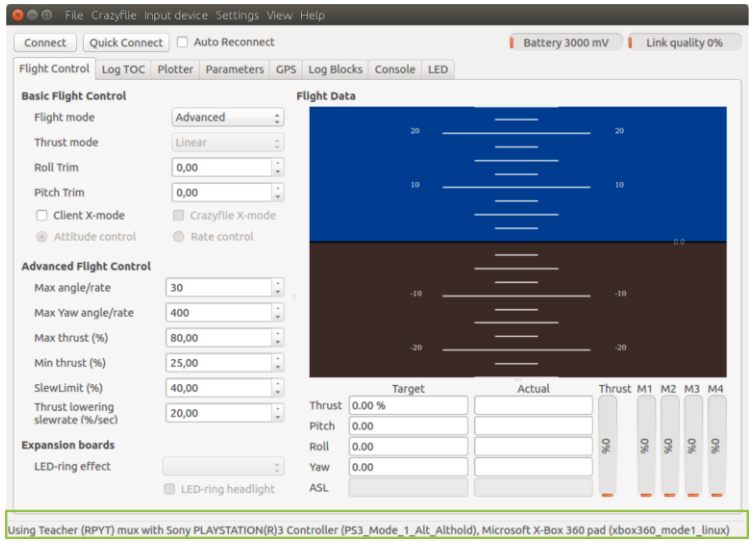
- Test team management
- Test day
- Environment: Prepared space, obstacles, cameras, safety.
- Test planning to verify requirements



6.11. Test Manager Report
Work Done

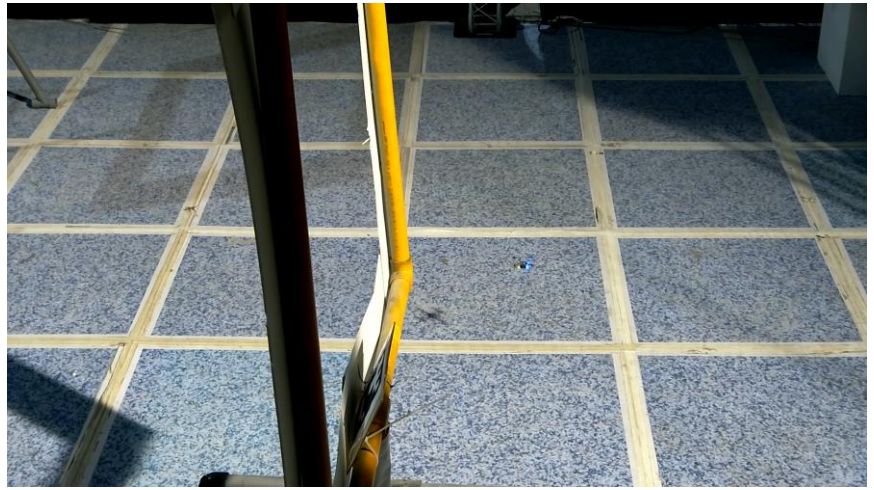
CONNECTIVITY TEST

- Drone–antena connection
- Cfclient



LOAD CAPACITY TEST

- Bearing own weight + components



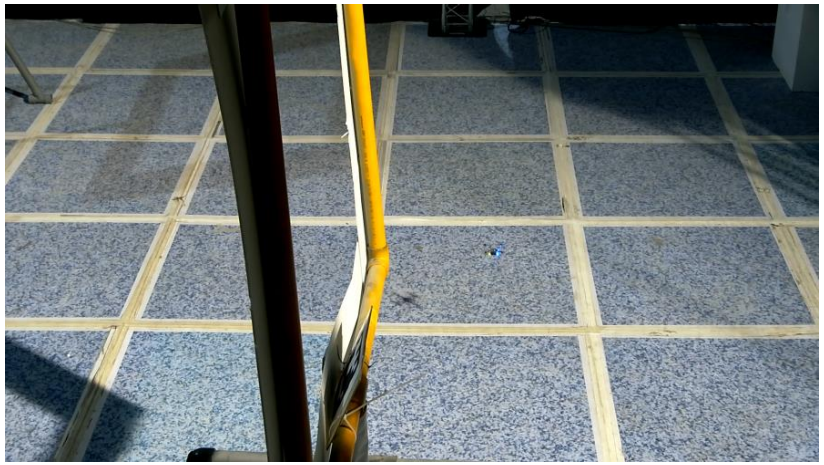
CONTROL TEST BY GROUND STATION (GS)

- GS must be able to control the basic movements of the drone
- GS must be capable of controlling speed and altitude

6.11. Test Manager Report
Work Done

TAKE-OFF AND LANDING TEST

- Takeoff from stationary position
- Safe landing

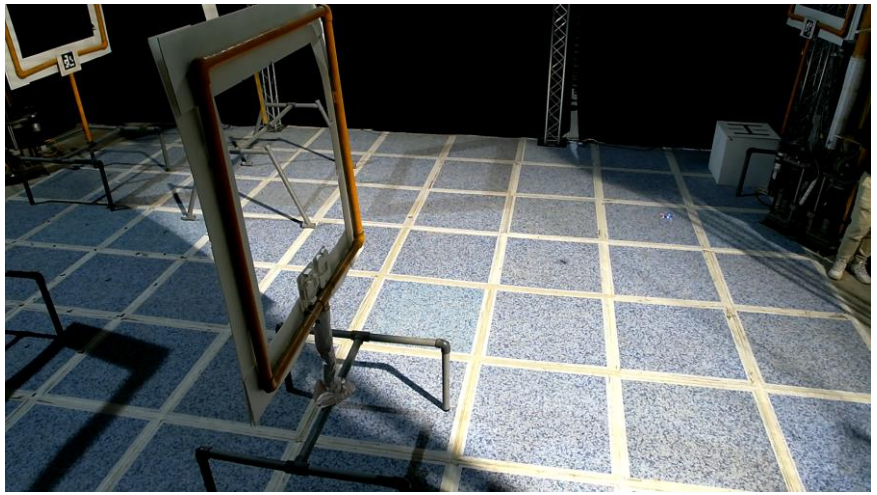


STATIONARY FLIGHT TEST

- Check that the battery lasts as long as necessary for the competition.

CONTROLLED FLIGHT TEST

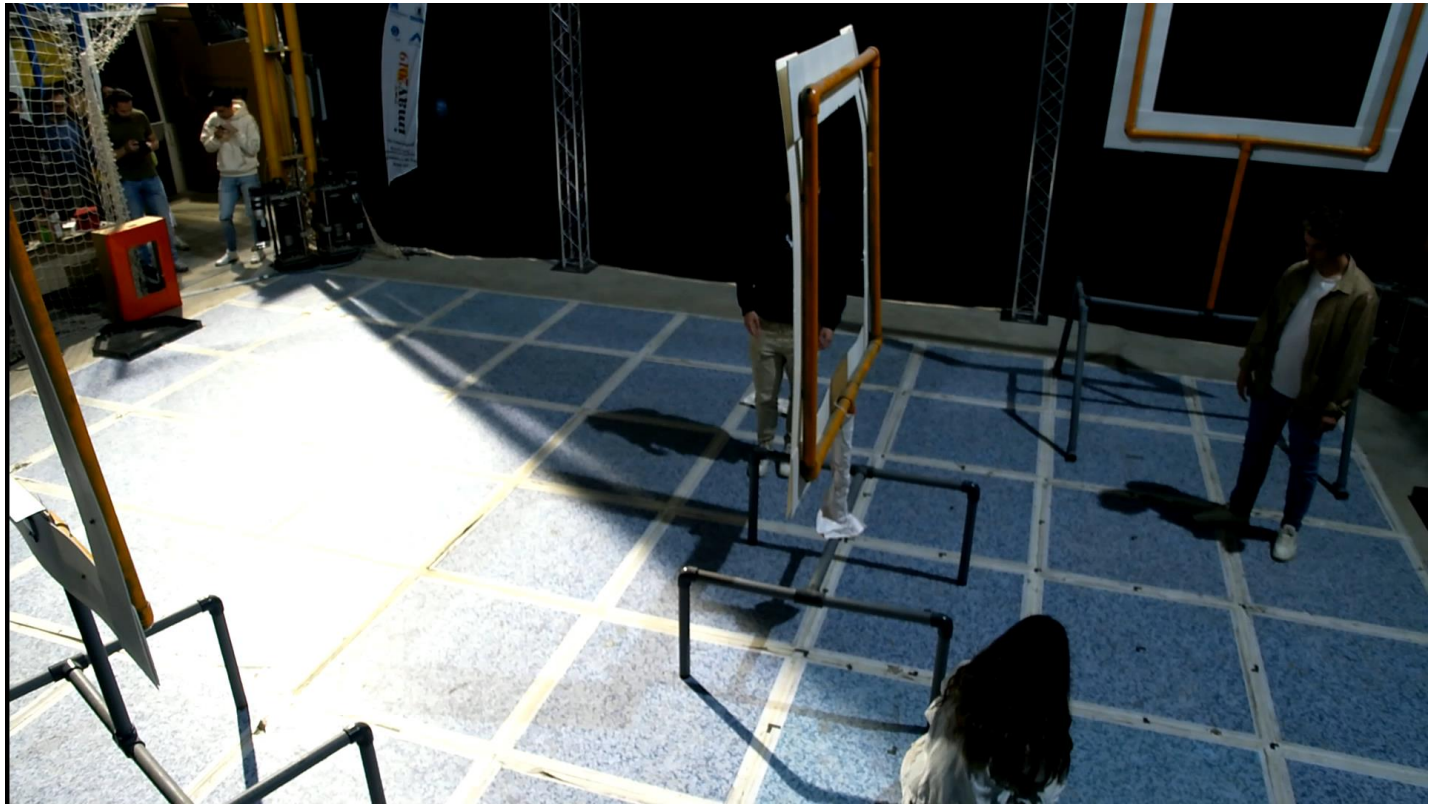
- Follow references within tolerance limits



6.11. Test Manager Report
Work Done

ADVANCED TRAJECTORY TEST

- Complex path and reaching a target position



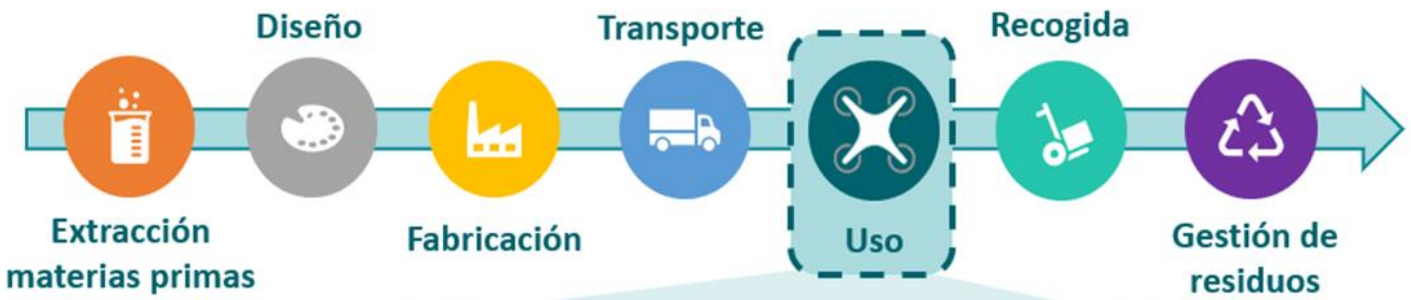
6.11. Test Manager Report
Analysis

REQUIREMENT CODE	RESULT
Drone/Flight/SyRS/SY-001	Checked
Drone/Motors/SyRS/SY-001.1	Checked
Drone/Networking/SyRS/SY-001.2	Not checked
Drone/Networking/SyRS/SY-001.3	Checked
Drone/Onboard Computer/SyRS/SY-001.4	Checked
Drone/Onboard Computer/SyRS/SY-001.5	Checked
Drone/Onboard Computer/SyRS/SY-001.6	Checked
Drone/Flight/SyRS/SY-002	Checked
Drone/Battery/SyRS/SY-003	Checked
Drone/Battery/SyRS/SY-003.1	Checked
Drone/Onboard Computer/SyRS/SY-004	Not checked
Drone/Vision System/SyRS/SY-004.1	Not checked
Drone/Vision System/SyRS/SY-004.1.1	Not checked

REQUIREMENT CODE	RESULT
Drone/Flight/SyRS/SY-004.1.2	Not checked
Drone/Flight/SyRS/SY-004.2	Checked
Drone/Flight/SyRS/SY-004.3	Checked
Software/Source Code/SRS/SO-001	Checked
Networking/Communications/SRS/SO-002	Checked
Networking/Communications/SRS/SO-002.1	Not checked
Networking/Communications/SRS/SO-002.2	Not checked
Networking/Communications/SRS/SO-002.3	Checked
Networking/Communications/SRS/SO-002.4	Not checked
Central Computer/Vision/SRS/SO-003	Checked
Central Computer/Vision/SRS/SO-004	Checked
Central Computer/Vision/SRS/SO-005	Checked
Software/Programming Language/SRS/SO-005.1	Checked

7. Sustainability and Impact Analysis

ÁMBITO AMBIENTAL: ANÁLISIS DEL CICLO DE VIDA DEL DRON



ENTRADAS:

- 277,5 Wh del mix energético español

SALIDAS:

- 41,63 g CO_2 eq
- 0,0954 g NO_x
- 0,0756 g SO_x
- 0,00451 g $PM_{2.5}$

CATEGORÍAS DE IMPACTO:

- Cambio climático
- Acidificación
- Eutrofización

ÁMBITO SOCIAL

- Estudio de la ausencia de **diversidad de género** en este INGENIA: causas y posibles soluciones.

ODS MÁS RELEVANTES

8. Conclusions

It is advisable to simplify when working on complex projects with limited time.

It is advisable to avoid making products with low availability mandatory for the completion of the project.

It is advisable to make decisions concerning supplies as early as possible to avoid delays.

Fluid communication with providers is of the essence.

8. Conclusions

Configuration and basic testing should be done as soon as possible, as it may affect decisions concerning design and implementation.

Risk management should not be taken lightly.

Planification should be as flexible as possible.

Errors, failures and learnt lessons should be documented to avoid them in future projects.

Thank you for your attention