INGENIA SE | COURSE 2022-2023

Model Structure, Use and Management

Hell-ix Team

May 23, 2023

Contents

1 Introduction **3**
- 1.1 Purpose of the Document . . . 3
- 1.2 Scope and Overview of the Model . . . 3
- 1.3 Document Organization . . . 3

2 System Overview **4**
- 2.1 Description of the Autonomous Drone Racing Competition System . . . 4
- 2.2 Functional Architecture . . . 4
- 2.3 Physical Architecture . . . 4

3 Model Structure **5**
- 3.1 Subsystem Breakdown and Interconnections . . . 5
  - 3.1.1 Environment . . . 6
  - 3.1.2 Trials . . . 8
  - 3.1.3 Hell-ix . . . 9
- 3.2 Functional Architecture Description . . . 16

4 Model Use and Management **18**
- 4.1 Model Initialization and Configuration . . . 18
- 4.2 Model Parameters . . . 19

1. Introduction

1.1 Purpose of the Document

The purpose of this document is to explain the model structure, use and management of the system designed by Hell-ix for the Autonomous Drone Racing Challenge 2022-2023. It serves as a guide for understanding and using the MBSE files contained in the Hell-ix group GitHub repository `https://github.com/Ingenia-SE/Hell-ix/tree/main/Hell-ix_Project`.

1.2 Scope and Overview of the Model

The model contains the whole Autonomous Drone Racing Competition system, taking into account users, the drone, the ground station and all other elements. Two main architectures have been modeled by using the modeling tool MATLAB System Composer: a functional one that describes functional dependencies (actions) and a physical one that describes the hardware components and their connections.

1.3 Document Organization

The present document is aimed to provide a reference for the visualization, comprehension and edition of the Autonomous Drone Racing Competition system model. To be able to use and modify the model, it is required to have the software MATLAB installed. Additionally, the following add-ons should be installed:

- Simulink
- System Composer
- Requirements Toolbox

The model is organized into multiple files, each containing distinct sets of information:

- **Requirements:** Requirements are assigned to the different subsystems and components of the model in order to ensure design traceability. There are three requirements files:
  - Software requirements: they are included in `software requirements.slrqx` file.
  - Stakeholder requirements: they are included in `stakeholder requirements.slrqx` file.
  - System requirements: they are included in `system requirements.slrqx` file.
  - Requirements traceability: the information related with the requirements assignation is included in the file `Hell_ix_System_Architecture~mdl.slmx`.
- **System Physical Architecture:** The physical architecture describes the hardware components together with their interconnections that enable to understand the physical model of the system. It is included in file `Hell_ix_System_Architecture.slx`. `Competition.xml` file includes information about the different stereotypes that have been created in the model, together with their assigned properties.
- **Functional Architecture:** The functional architecture describes the functional dependencies of the system as actions (each block involves certain hardware and software that

enable the action to be performed). It is complementary to the main system architecture and defines how the system will behave. It is included in file `Hell_ix_Functional_Architecture.slx`. File `Functional_Physical_Allocation.mldatx` establishes traceable and directed relationships between the architectural elements of the physical and functional models.

2. System Overview

In this section a general system overview is shown, together with a brief description of each of the architectures that conform the model and how they are integrated.

2.1 Description of the Autonomous Drone Racing Competition System

Figure 3.1 illustrates the comprehensive model architecture, comprising the Droning Team, the Hell-ix team, the environment, and the trials. These four interdependent architectures are meticulously designed to operate in harmony, as they exert mutual influence on each other. The environment primarily encompasses the materials and individuals involved in the competition, significantly impacting both the trials and the architectures of both teams. Conversely, the teams directly influence the competition by defining its parameters and shape the environment by designing obstacles and participating as individuals. Finally, the trials have a profound impact on both teams, as their work is rooted in adherence to the competition rules, and on the environment, as the obstacles must be conceived and constructed in accordance with these regulations.

2.2 Functional Architecture

The functional architecture describes the functional dependencies of the system as actions (each block involves certain hardware and software that enable the action to be performed). The functional model depicts the three principal functional divisions: User input, obstacle race, and freestyle test. User input serves as a prerequisite for initiating both the obstacle race and freestyle test. Subsequently, in order to successfully navigate the race path, the drone must execute various actions during the obstacle race and freestyle tests, which will be elaborated upon in this document.

This architecture is displayed in figure 3.15 for further comprehension.

2.3 Physical Architecture

The physical architecture describes the hardware components together with their interconnections that enable to understand the physical model of the system. The physical model encompasses three distinct components: the Hell-ix drone system (refer to Figure 3.7), the materials architecture (refer to Figure 3.3), and the people architecture (refer to Figure 3.4).

As shown in the Figure 3.1, there exists a crucial interdependence among the three components. The outputs from the environment play a vital role in providing essential information to both the trials and teams, thereby ensuring the successful completion of the competition. The control managers of each team are responsible for executing the appropriate program at the ground base (exec_prog outputs). This information is then transmitted to the drones via the Radio PA, enabling them to perform in the race. The drones' end output is subsequently received by the trials as an input, and the ultimate winner is determined within the trials block. Furthermore, the environment generates outputs such as the start of the race through the horn and chronometer, which are processed as inputs within the trials block. In

the freestyle competition, the chronometer also signals the end of the competition, as it lasts for one minute. Once the allotted time elapses, the trials block receives this output from the environment. Moreover, the obstacles, as outputs from the environment, are received by both the teams' blocks for the drones to adapt to the different races and the trials' block to define the competition rules.

3. Model Structure

In this section, the model structure of the system is described in detail in order to enable its easy comprehension for further amelioration and as documentation for users both from and not from the team.

3.1 Subsystem Breakdown and Interconnections

This section includes a detailed description of the different subsystems of the model, displayed in Figure 3.1

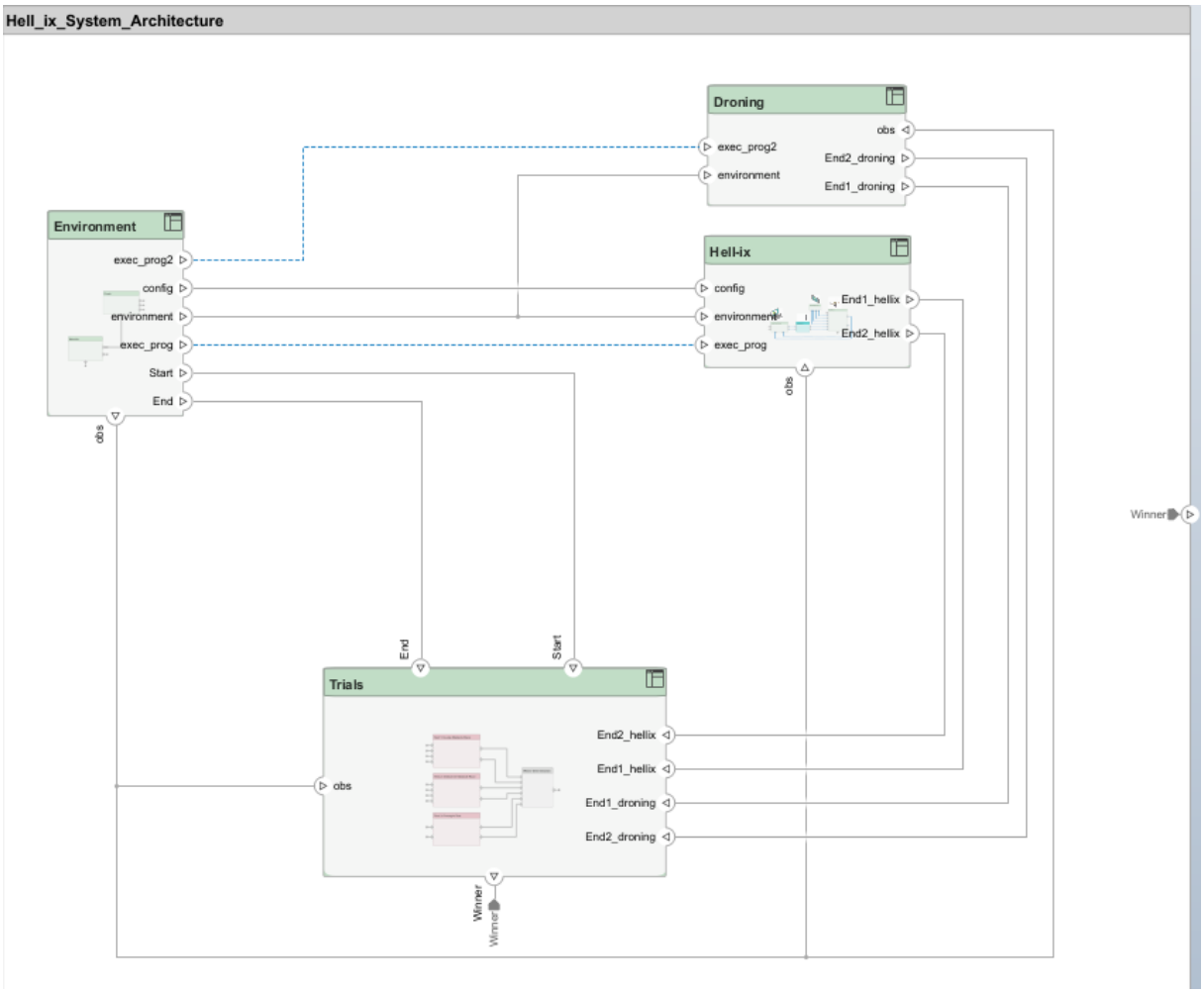

Figure 3.1: General system model.

3.1.1 Environment

The environment block refers to part of the model other than the drone teams and the competition trials. It is made up of two correlated blocks, as shown in Figure 3.2: materials and people. As explained before, the materials outputs are the obstacles, start and end moments of the race, established by the horn and the chronometer. The start signal is received as an input by the people block, so that the program developed for that specific competition part is executed and configured.

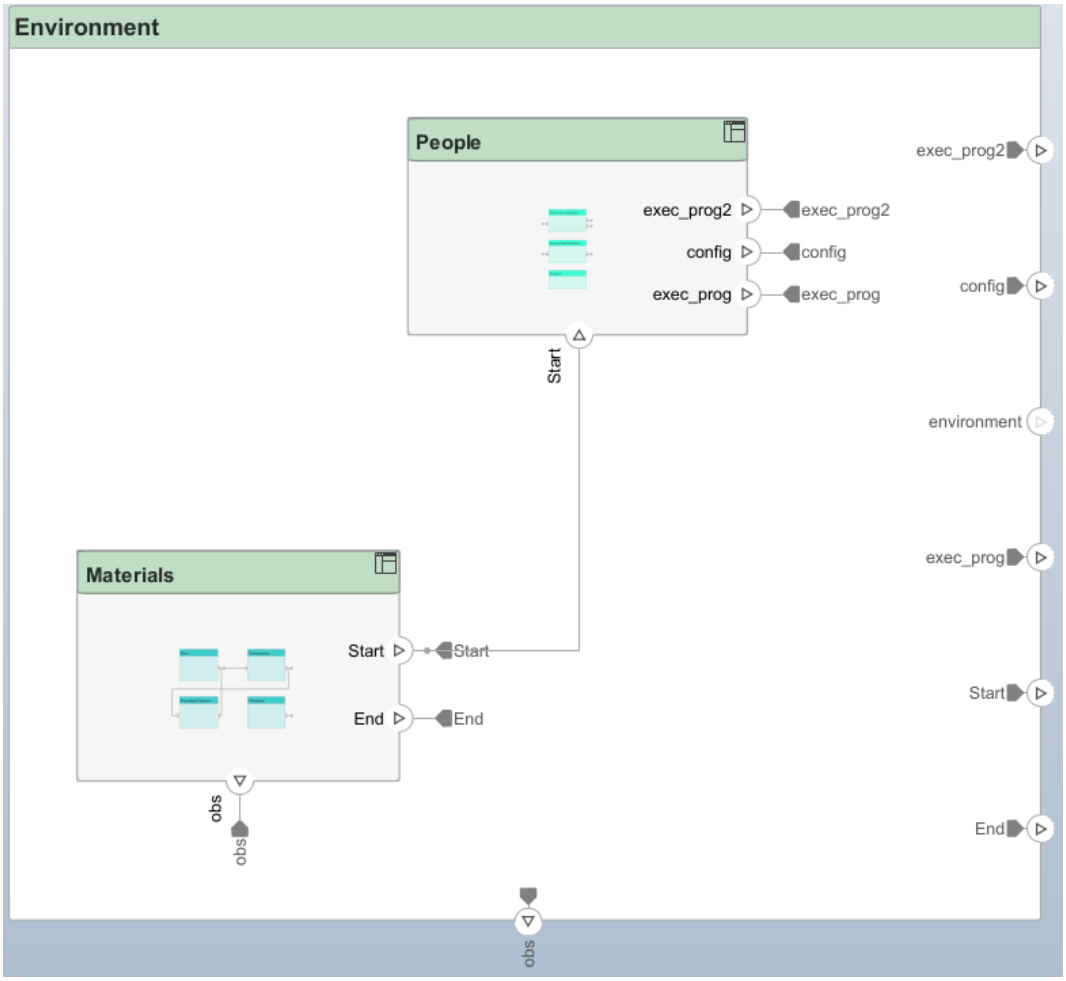

Figure 3.2: Environment subsystem model.

Materials

The materials block, depicted in Figure 3.3, comprises four interconnected components: the horn, chronometer, recording cameras, and obstacles. These four elements must operate harmoniously in order to ensure the smooth functioning of the system.

The horn emits a start output, which serves as an input to both the chronometer and the recording cameras. The chronometer uses this input to initiate the measurement of competition time, while the recording cameras commence capturing the performance of the drones.

While the obstacles within the materials block do not receive any input, their output is crucial for the successful operation of other subsystems, such as the trials and the drone's

teams. The obstacles play a vital role in determining the challenges and rules that the drones must navigate during the competition.

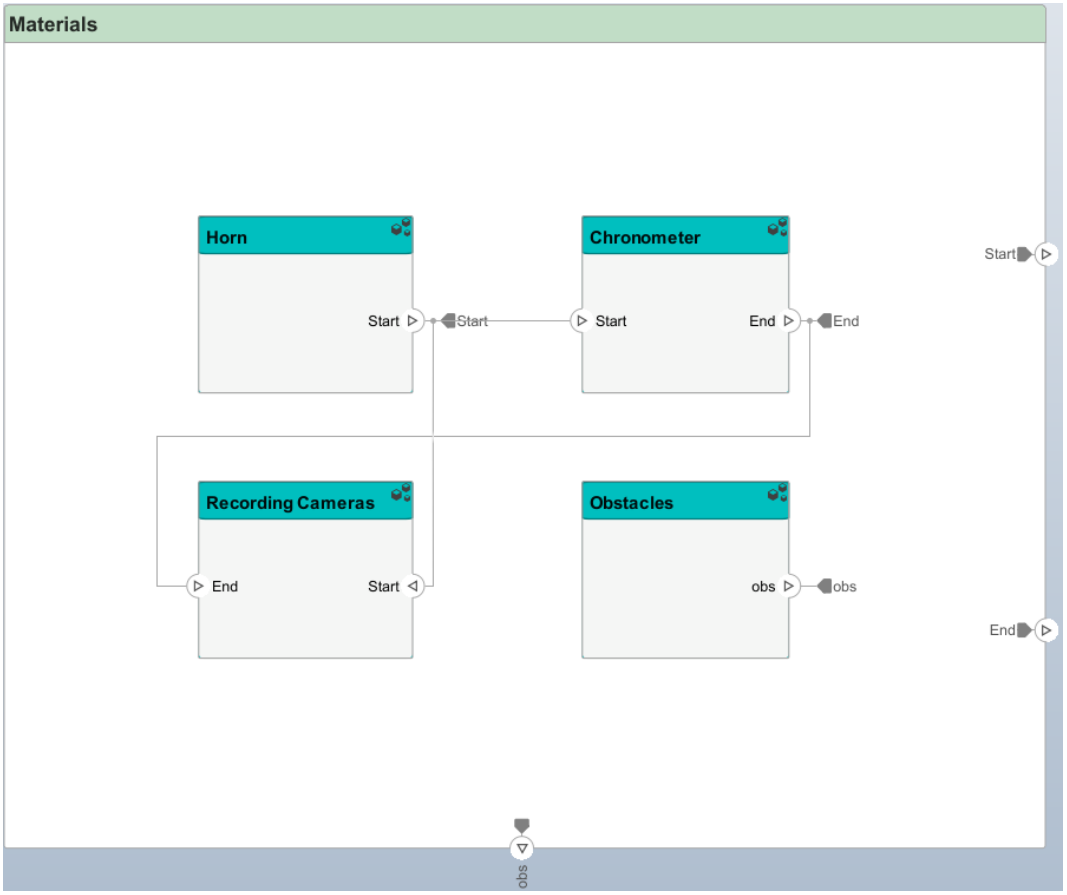

Figure 3.3: Materials subsystem model.

People

The people architecture consists of members from both the Hell-ix and Droning teams, as well as the audience attending the competition, as displayed in the Figure 3.4. This three subsystems inside the people block, do not interact with each other, as they are independent, even though their outputs are crucial for other blocks.

The Hell-ix team members and the Droning team members are considered equivalent in terms of their subsystems. However, a crucial role within both teams during the competition is played by the Control Manager. The Control Manager is responsible for receiving the start signal as an input and subsequently executing the program specific to the competition. Additionally, the Control Manager is responsible for configuring the program accordingly.

The remaining members of the team function in collaboration with the Control Manager, providing support and assistance. One particular task they undertake is operating the auxiliary mobile phone control, which complements the overall control system during the competition. This architecture is displayed in 3.5.

Figure 3.4: People subsystem model.

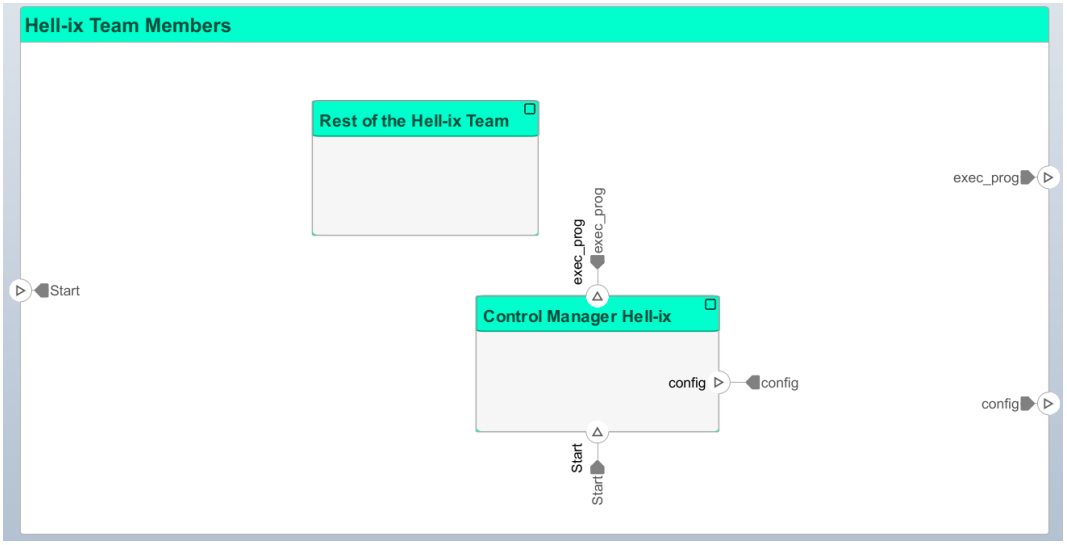

Figure 3.5: Hell-ix team members model.

3.1.2 Trials

The Trials' block architecture, illustrated in Figure 3.6, consists of three distinct and independent blocks. Each block is activated when the specific test is received as an input in the Trials block.

In Test 1, the In-Line Obstacle Race, and Test 2, the Semicircle Obstacle Race, the blocks receive inputs such as the race's start moment, information about the obstacles, and the end times for both teams. These inputs are used to calculate and determine the points earned by each team as an output.

In Test 3, the Freestyle Test, the block simply receives the start and end moments from the chronometer. Based on this information, it determines the points earned by each team as an output.

Lastly, the Winner Determination block takes as inputs the points earned by both teams in all three competitions. It then processes this information to determine the overall winner of the competition, which is provided as an output.

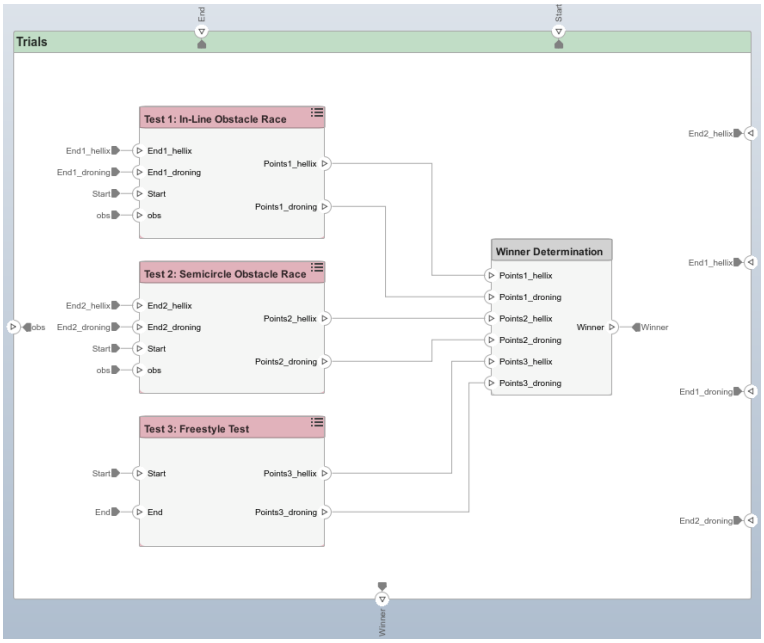

Figure 3.6: Trials subsystem model.

3.1.3 Hell-ix

The Hell-ix drone system is composed of the ground control station, which establishes a connection to the Crazyflie 2.1 drone through the Crazy Radio PA. Additionally, a mobile phone can also be connected to the drone, serving as an auxiliary controller, as displayed in Figure 3.7.

The ground control station serves as a pivotal component in the system and receives various inputs to facilitate effective communication and control of the Crazyflie drone. It receives program execution and configuration inputs from the team control manager, as well as telemetry and FPV (First Person View) video from the drone.

These inputs are processed by the ground control station, resulting in outputs such as Telemetry USB and GCSCmds, which are then received as inputs by the CrazyRadio PA. The CrazyRadio PA acts as a communication interface, enabling the ground control station to establish a connection with the drone. It utilizes the outputs, namely telemetry radio and GCS (Ground Control Station) commands, to facilitate this communication.

Furthermore, the auxiliary mobile phone, equipped with the cfclient app, can emit output signals that include yaw, roll, pitch, and z orders. These outputs are received as inputs by the drone, in conjunction with the radio outputs.

Upon receiving these inputs, the drone performs the competition tasks and subsequently sends outputs in the form of FPV video and Telemetry back to the ground control station. This information is used for more precise monitoring and control in subsequent steps of the competition.

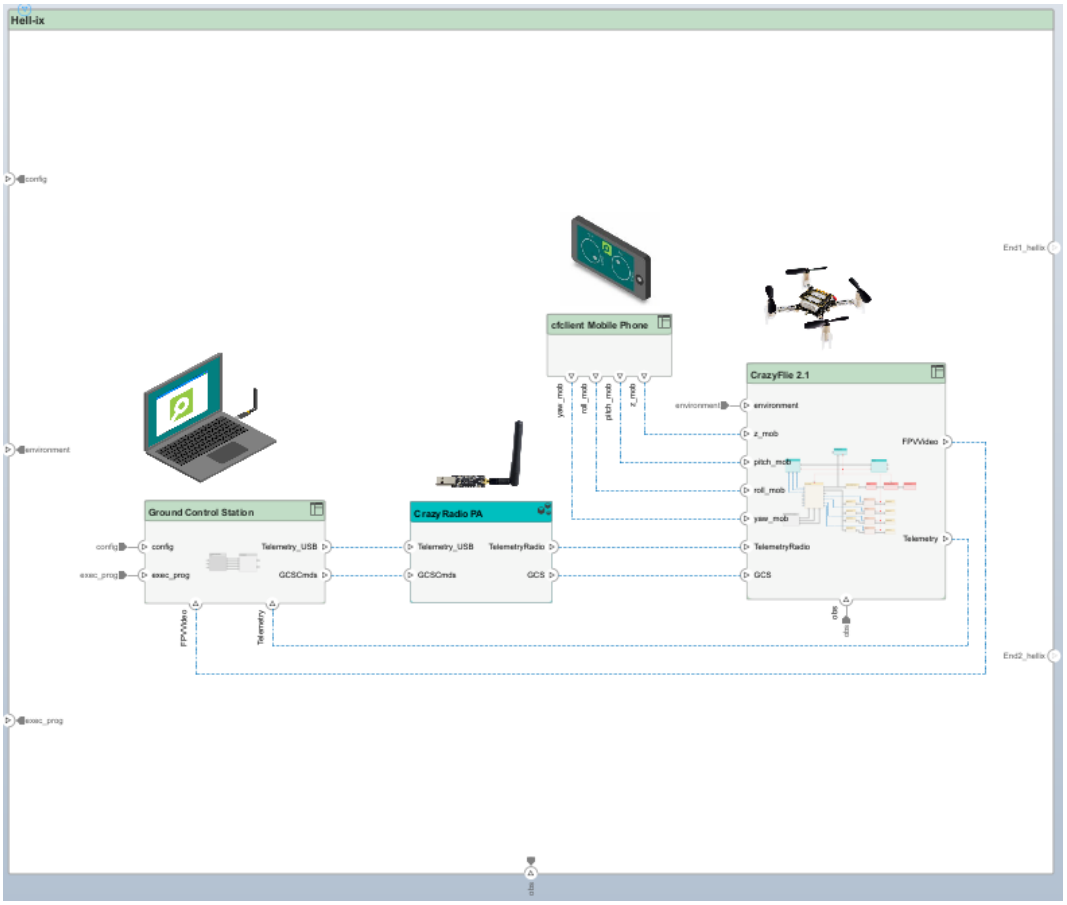

Figure 3.7: Hell-ix subsystem model.

Ground Control Station

As depicted in Figure 3.8, the ground control station comprises two distinct blocks: program selection and commands. The program selection block receives inputs such as FPV video, program execution, and configuration. Based on this input, it generates outputs in the form of x, y, and z commands, as well as yaw and LED configuration (applicable during the freestyle competition).

These outputs from the program selection block are then received as inputs by the commands block. The commands block processes this input and produces outputs in the form of Telemetry USB and GCSC mds. These outputs serve to facilitate communication and control with the drone, enabling the ground control station to effectively monitor and maneuver the drone during the competition.

Crazyflie 2.1

The architecture of the Crazyflie 2.1 drone, as depicted in Figure 3.9, consists of several interconnected components. These include:

- Flow Deck: This component is responsible for providing optical flow sensing capabilities to the drone, enabling it to perceive its movement and position relative to the environment.

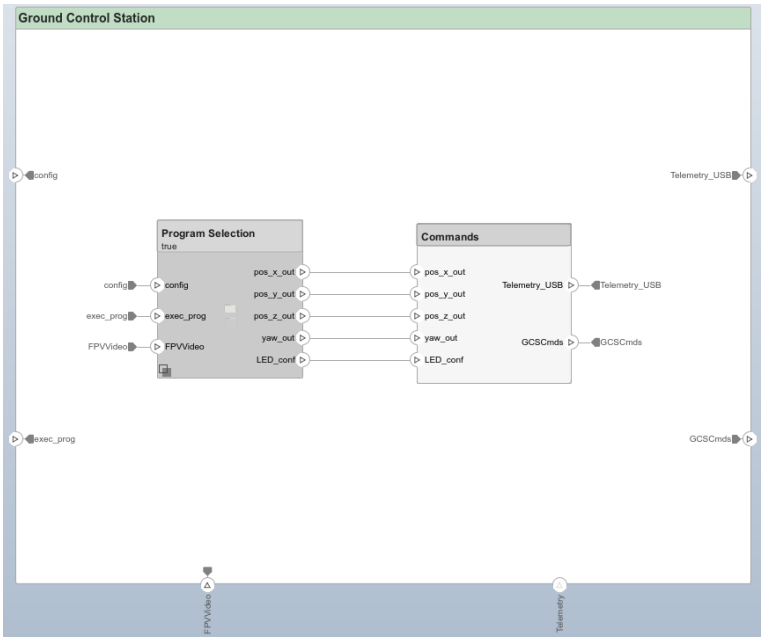

Figure 3.8: Ground control station subsystem model.

- LED Ring Deck: The LED ring deck encompasses a ring of LED lights that can be programmed to display various colors and patterns, adding visual appeal and functionality to the drone.
- Artificial Intelligence Deck: The Artificial Intelligence (AI) deck is an additional module that enhances the drone's capabilities by enabling onboard processing and execution of AI algorithms.
- Drone Body: The main body of the Crazyflie 2.1 drone houses and supports all the interconnected components.
- Battery Model Blocks: The drone is powered by multiple batteries, and the battery model blocks represent these power sources.
- Environmental Perturbations Block: This block represents the influence of environmental factors on the drone's flight, such as wind or other disturbances.
- Motor Blocks: The drone comprises several motor blocks that control the individual motors responsible for flight. Each motor block regulates the operation of a specific motor.
- Motor Arm Blocks: The motor arm blocks connect the motors to the drone's body structure, providing stability and facilitating controlled movement.
- Helix Blocks: The helix blocks refer to the propellers or rotor blades that generate the necessary thrust for the drone's flight.

These interconnected components work together to enable the Crazyflie 2.1 drone to perform various functions and maneuvers during the competition.

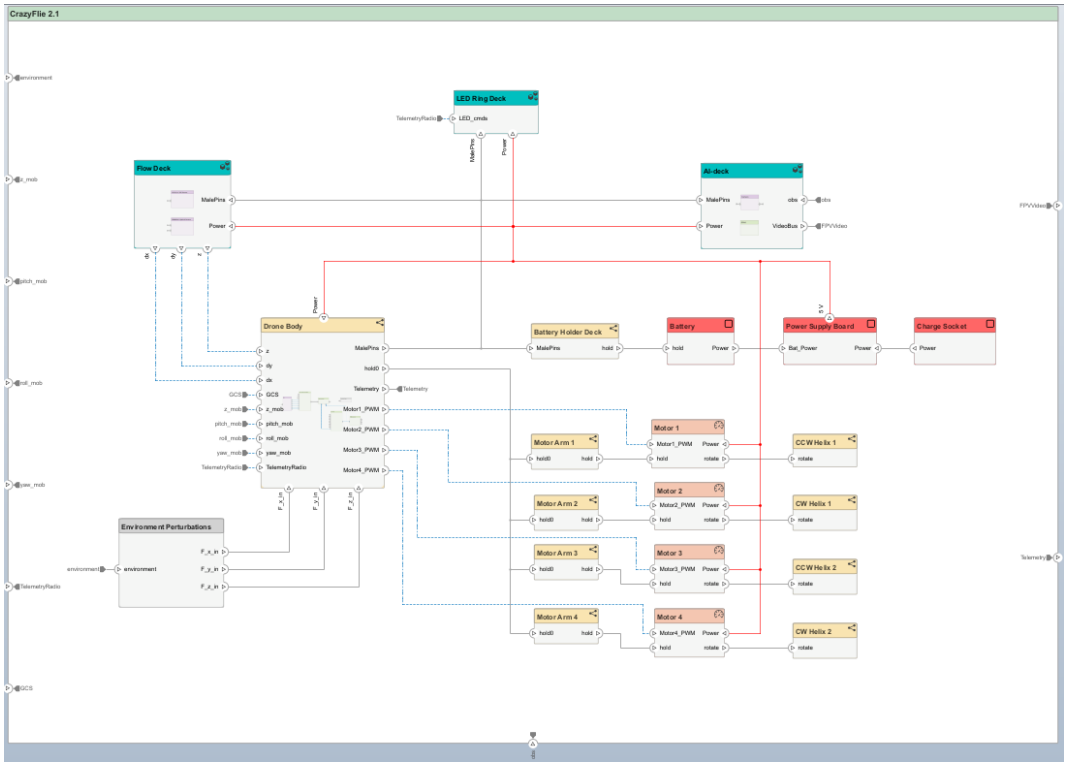

Figure 3.9: Ground control station subsystem model.

The Flow Deck takes inputs such as male pins from the Drone Body module and power from the Power Supply Board module. It utilizes this information to output instructions for dx (lateral movement), dy (longitudinal movement), and z (vertical movement) to the Drone Body.

The LED Ring Deck takes inputs including male pins from the Drone Body module, power from the Power Supply Board module, and LED commands from the Telemetry Radio. It is activated during the freestyle competition, and it does not produce any specific outputs.

The AI Deck receives inputs such as male pins from the Drone Body module, power from the Power Supply Board module, and information about obstacles. It generates the Video Bus used for the FPV (First Person View) Video.

The Drone Body module receives various inputs, including z (vertical movement), dx and dy (lateral and longitudinal movements), Ground Central Station commands, z mobility, pitch, roll, yaw, telemetry radio signals, and forces derived from environmental perturbations (F_x_in, F_y_in, and F_z_in). It outputs male pins for the decks, telemetry signals, motor signals (Motor1_PWM, Motor2_PWM, Motor3_PWM, and Motor4_PWM), and signals for motor arms (hold).

The Battery Modeling is comprised of four blocks, as displayed in the Figure 3.10. The Battery Holder Deck takes inputs from the Drone Body's male pins and outputs the hold signal. This signal is then received by the Battery Model, which produces power as an output. The Power Supply Board module takes inputs from the charge socket and the Battery Model's power output and emits power as an output to all the modules requiring it.

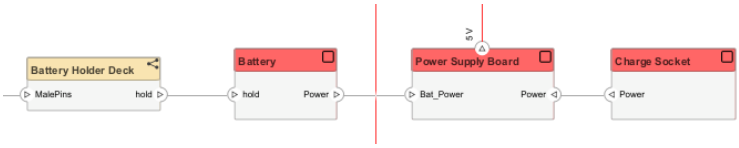

Figure 3.10: Battery modelling modules.

The Environment Perturbations block receives environment information from the Environment module as input and generates resulting forces in the vertical (F_z_in), lateral (F_x_in), and longitudinal (F_y_in) directions. These forces are then received as inputs by the Drone Body module.

The Motors Modeling, depicted in Figure 3.11, consists of three blocks representing the motor arms, motors, and helixes for each motor.

The Motor Arm block receives the hold information from the Drone Body module and produces the hold signal as output. This hold signal is then received by the corresponding motor block.

The Motor block takes inputs such as the hold signal from the Motor Arm block, Motor_PWM from the Drone Body module, and power from the Power Supply Board module. Based on these inputs, the Motor block generates the corresponding rotation command for the helix.

This process applies to all four motors, allowing the drone to successfully adjust its roll, pitch, and yaw movements.

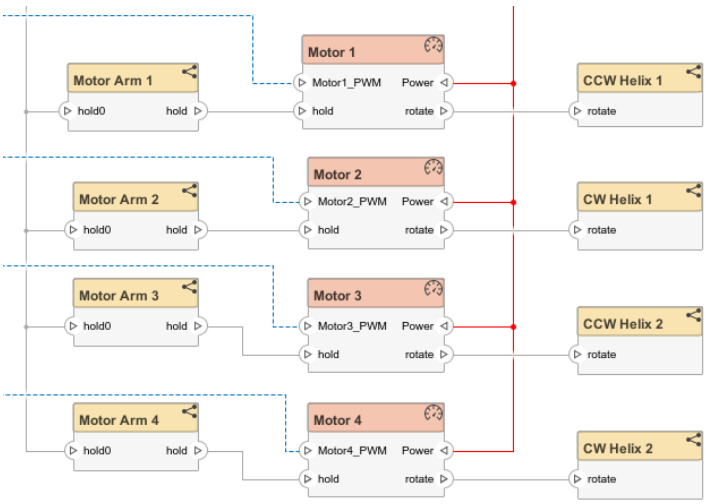

Figure 3.11: Motors modelling modules.

Flow Deck

The Flow Deck module, as depicted in Figure 3.12, consists of two independent sensors: the VL53L1x Time-of-Flight (ToF) Sensor and the PMW3901 Optical Sensor. The VL53L1x ToF Sensor outputs the z signal, which represents the distance measured by the sensor to the floor. This signal provides information about the drone's vertical position relative to the

ground. On the other hand, the PMW3901 Optical Sensor outputs the dx and dy measures. These measures provide information about the drone's lateral (dx) and longitudinal (dy) movements. By combining the outputs from both sensors, the Flow Deck module enables the drone to accurately perceive its position and movements in the vertical, lateral, and longitudinal directions.

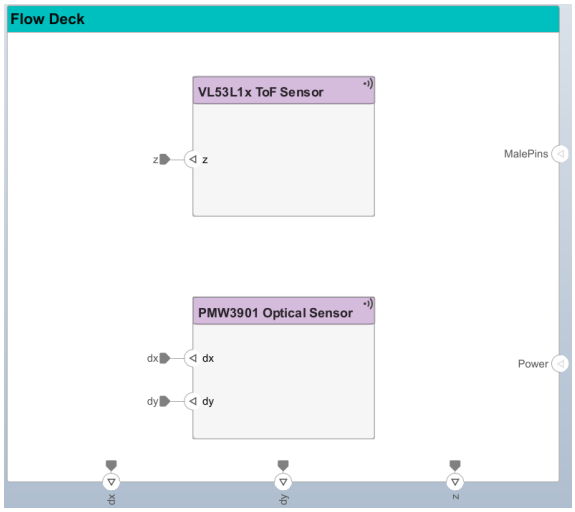

Figure 3.12: Flow deck modelling module.

AI Deck

The AI Deck, as shown in Figure 3.13, is comprised of two main modules: the Camera module and the GPA8 module. The Camera module receives obstacles as input. Using this information, it processes the visual data captured by the camera and generates the out bus for the video bus. This out bus contains the processed video data that it is used for the object recognition. The GPA8 module is responsible for executing the AI-related tasks. It utilizes the processed visual data from the Camera module as input and performs AI algorithms or computations on this data.

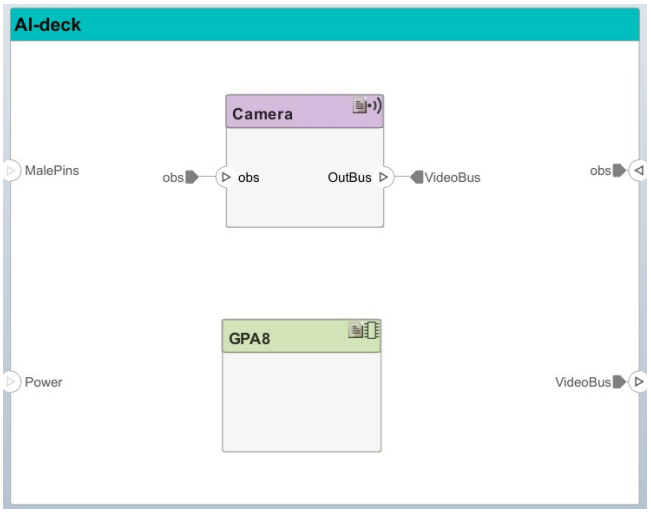

Figure 3.13: AI deck module model.

Drone Body

The drone body comprises the IMU module, the current state calculation module, the state estimator, the controller, the PWM generator and the functional-logic module, as illustrated in the Figure 3.14.

The IMU (Inertial Measurement Unit) module takes power as input and outputs various signals related to the drone's movement, including d_pitch (change in pitch), d_roll (change in roll), d_yaw (change in yaw), acc_x (acceleration in the x-axis), acc_y (acceleration in the y-axis), and acc_z (acceleration in the z-axis). These outputs, along with the dx, dy, and z signals, serve as inputs for the Current State Calculation module.

The Current State Calculation module utilizes the aforementioned signals to determine and output the current state of the drone. This current state is then received by the State Estimator module, along with the actuation signal from the Controller. The State Estimator module uses this information to estimate and output the telemetry and the estimated state of the drone. The estimated state is subsequently used as input by the Controller.

The Controller module takes inputs such as the estimated state, power, telemetry radio signals, z_mob (z mobility), pith_mob (pitch mobility), roll_mob (roll mobility), yaw_mob (yaw mobility), and the forces F_x_in, F_y_in, and F_z_in. Based on these inputs, the Controller generates the actuation signal as its output. This actuation signal is used by both the State Estimator and the PWM Generator module.

The PWM Generator module, utilizing the actuation signal, produces the Motor1_PWM, Motor2_PWM, Motor3_PWM, and Motor4_PWM signals as its output. These signals are crucial for controlling the motors of the drone.

Lastly, the Functional Logic module operates independently by receiving outputs from the Ground Control Station. The specific functions and interactions of this module may vary depending on the requirements and design of the system.

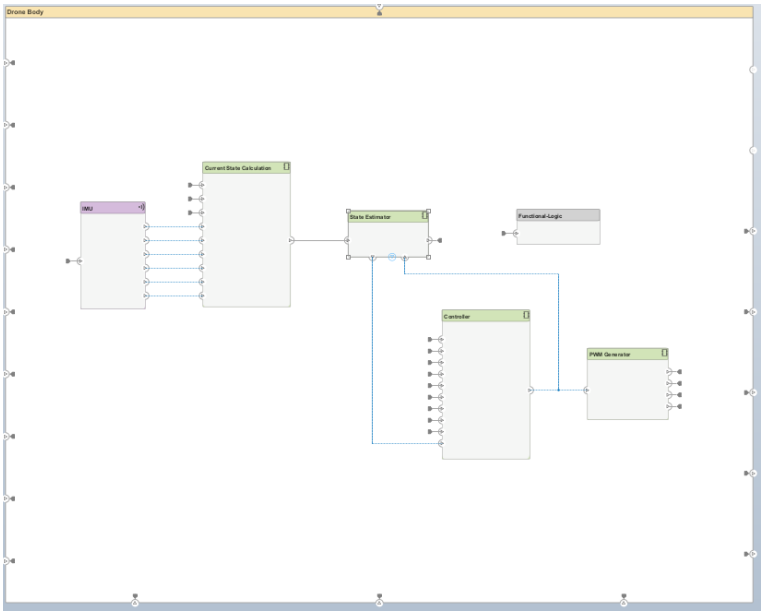

Figure 3.14: Drone body modelling module.

3.2 Functional Architecture Description

The model that depicts the functional architecture of the drone race can be seen in figure 3.15. Each of the actions shown in this model are implemented by a physical component of the drone. These relationships between the two models can be viewed and modified using the Allocation Editor. The model begins with the choice of a test by the user, which can be either an Obstacle Race or a Freestyle Test.

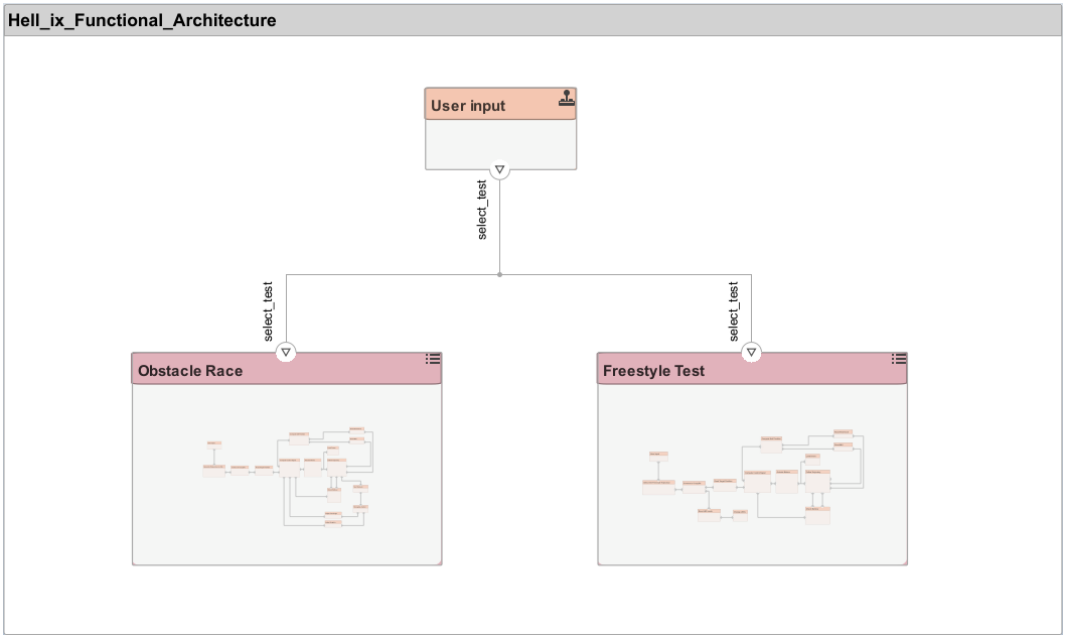

Figure 3.15: Functional Architecture model.

Obstacle Race

In the case that the Obstacle Race is chosen, the position of the gates must then be provided by the user. The Trajectory is calculated by the Ground Control Station and stored in a CSV file as the series of target positions that the drone must follow.

Then, the Ground Control Station connects to the Crazyflie via the Crazyradio PA. The Python API on the GCS subsequently reads the following target position and sends it to the Crazyflie. This desired position des_pos is compared with the estimated position est_pos and the Control Signal is computed. This Control Signal contains the commands that are sent to the motors. At this step, the status of the battery is checked. In the case of the battery level being low, a signal is sent to the Control System Computer at the Ground Control System. The given trajectory is followed until a signal is sent by the Ground Control System to land the drone.

The loop is closed in two different ways. First, the drone has its own Self Position Estimator that uses the information sent by the IMU and Flow-Deck on board the Crazyflie. The IMU provides information about the drone's position, velocity, acceleration and changes in orientation by using different sensors, whereas the Flow-Deck contains information from optical sensors that help the onboard processor estimate the device's velocity, altitude and position changes. By analyzing the optical flow patterns or depth information from these sensors, the drone can maintain a more precise position, stabilize its flight, and perform tasks such as obstacle avoidance or accurate landing.

Second, the camera on the AI-deck is responsible for streaming images to the Ground Control Station. When an obstacle is detected using an algorithm provided by OpenCV, the position of its center is calculated and the yaw angle and height are corrected accordingly.

The Control Signal Computer receives information from all three sources (i.e. the desired position given by the fixed trajectory, the actual position estimated by the drone and the obstacle detection data) and ponderates it to adapt the Crazyflie's trajectory.

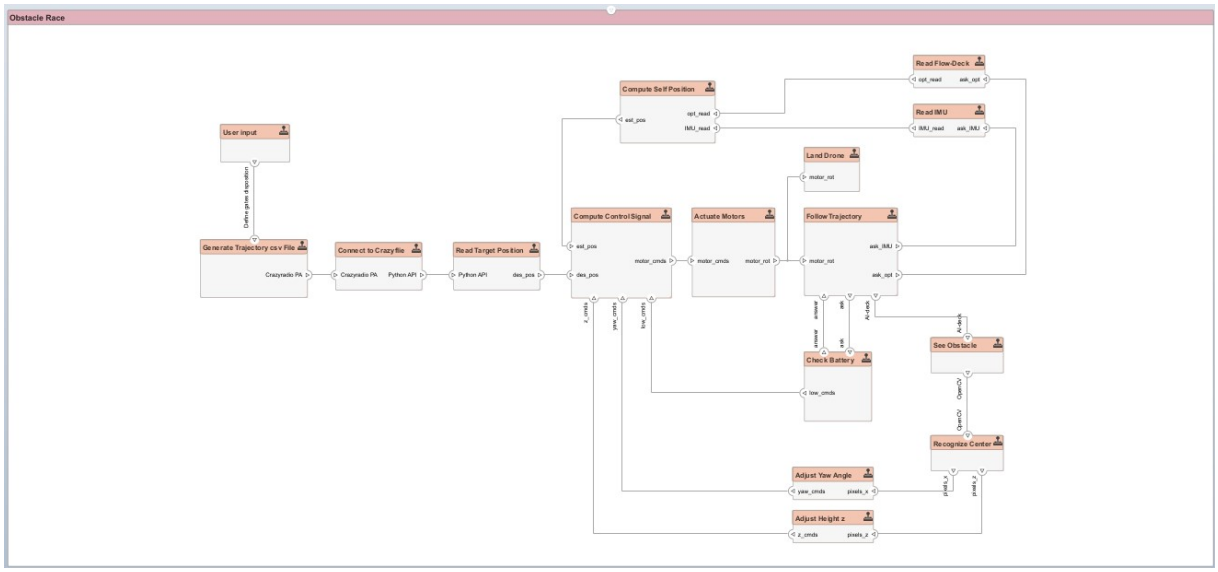

Figure 3.16: Obstacle race model.

Freestyle Test

In the case that the option selected by the user is the Freestyle Test, the necessary input is the selection of the specific freestyle trajectory from a library of different trajectories previously calculated by the team.

The connection to Crazyflie is done in the same way via the Crazyradio PA. Two algorithms in Python are responsible for sending signals that will change the colors of the onboard LEDs and reading the next desired position from the CSV file.

The trajectory is followed by the drone in the same way as in the Obstacle Race, with the exception of the detection of obstacles. In this case, the control signal computer simply receives the desired position specified in the CSV file and read by the Python code and the position estimation calculated by the drone itself using information from the Flow-Deck and the IMU sensors.

The control signal in terms of commands is sent to the motors, which comply with these instructions and follow the trajectory until a signal to land the drone is sent by the Ground Control System. In the same way, the battery status is checked at this step and a signal for low battery is sent to the GCS in the event of the battery level being low.

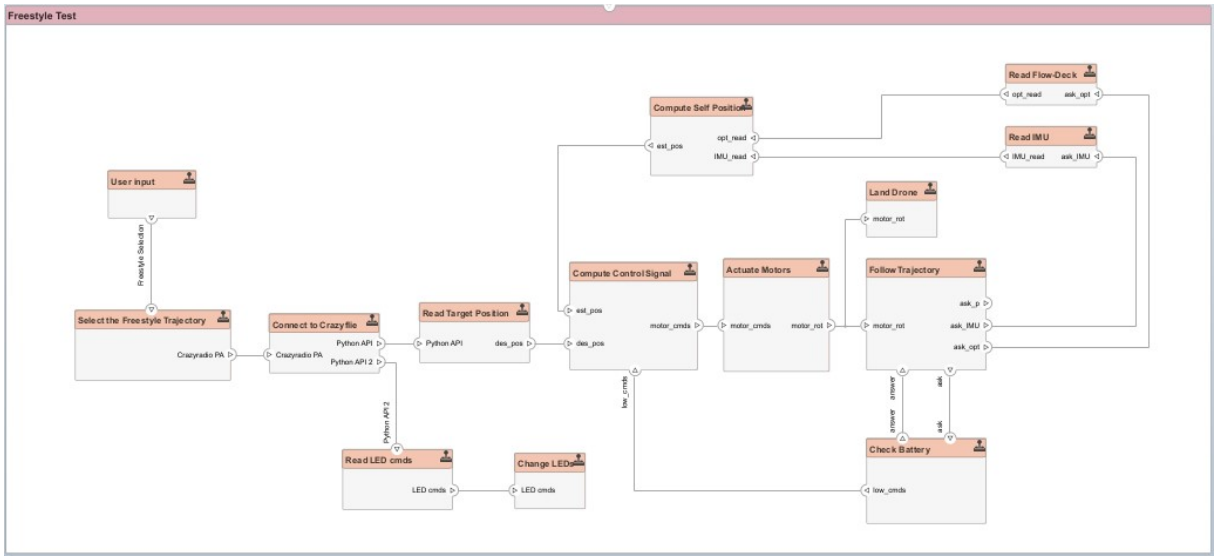

Figure 3.17: Freestyle test model.

4. Model Use and Management

In the present section, it is explained how the model can be initialized and configured in order to be used by any user. After reading the present documentation with the explanation of the current model, together with the steps provided in this section on the model use and management, any user can employ the model and adapt and ameliorate it.

4.1 Model Initialization and Configuration

In order to utilize and modify the model effectively, the installation of MATLAB software is required, along with the necessary Simulink, System Composer, and Requirements Toolbox add-ons. These software components collectively provide the essential tools and functionality required for seamless interaction with the model.

Once all the required software is installed, access the MATLAB Project Archive file called `System_Model`. Then, select the folder you would like to extract the project to. This will automatically do the set-up of the model and open the two main architectures. The model consists of two interconnected architectures that depict the relationships between actions and the associated hardware. To access and modify these relationships, the Allocation Editor in Simulink is utilized. By opening the `Functional_Physical_Allocation.mldatx` file, one can conveniently view and edit these connections. This functionality empowers users to finely adjust the system's behavior by allocating functional requirements to designated hardware components, thereby ensuring seamless performance and integration within the overall system design.

Moreover, within the model, all the software, stakeholder, and system requirements are interconnected with the corresponding physical components responsible for their implementation. To access and modify these relationships, users can use the Requirements Manager, located under the Apps tab. The Requirements Manager provides a comprehensive overview of the various requirements, organized into subcategories. Each requirement is accompanied by a detailed description and a direct link to the specific block within the model that implements the given requirement. This interface allows users to efficiently manage and track the

interdependencies between requirements and their corresponding implementation blocks.

4.2 Model Parameters

Through the profile editor, users are able to generate and modify various profiles. In our model, a total of 13 profiles have been created, as depicted in 4.1. Each profile allows for the assignment of a specific color and icon, as can be seen on the color-coded blocks within the model. Additionally, properties can be defined for each profile. For instance, in 4.2, we can observe the properties assigned to the "Test" profile. Consequently, all components labeled as "Test" within the model (e.g., Test 1, Test 2, and Test 3) inherit these defined properties.

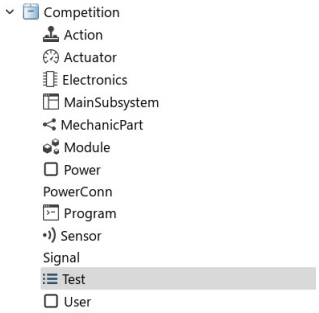

Figure 4.1: Profiles that have been defined in our model using the Profile Editor

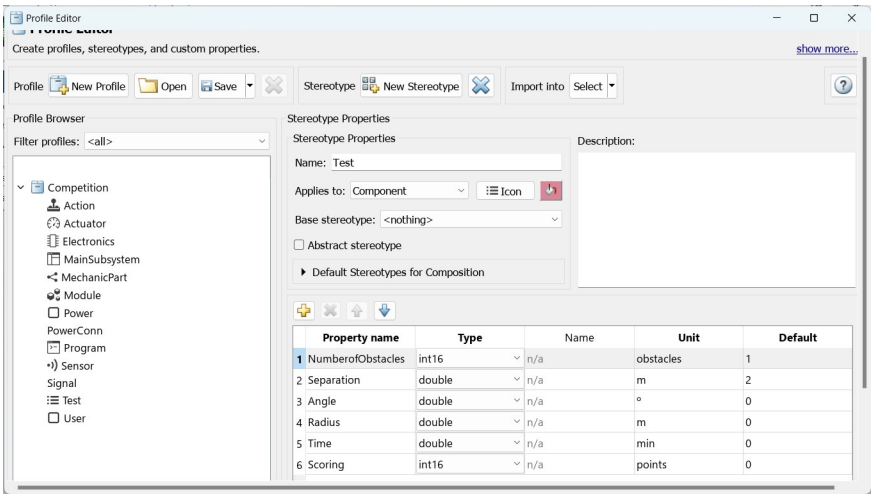

Figure 4.2: Properties defined for the profile category "Test"