Chapter 3

Flying Arena

This chapter describes in detail the implementation process of the devised experimental environment. Section 3.1 recalls the overall design adopted for the system. Section 3.2 explains how the information provided by the Optitrack cameras is processed and integrated into the testing setup. Section 3.3 presents some UAVs incorporated into the testing environment and describes how the PX4 was configured and merged into the framework. This section also details the capabilities of the QGroundControl software and how the PX4 and the radio controllers provide solutions to protect the users and the drones in case of failures. Section 3.4 introduces the modules developed to enable effortless offboard control of the vehicles. Finally, Section 3.5 presents a complete overview of the Flying Arena, through detailed diagrams, summarizing all the information presented over the chapter.

3.1 Architecture of the Flying Arena

The overall architecture of the Flying Arena was presented and discussed in Chapter 2. A slightly different representation of that architecture is depicted in Fig. 3.1.

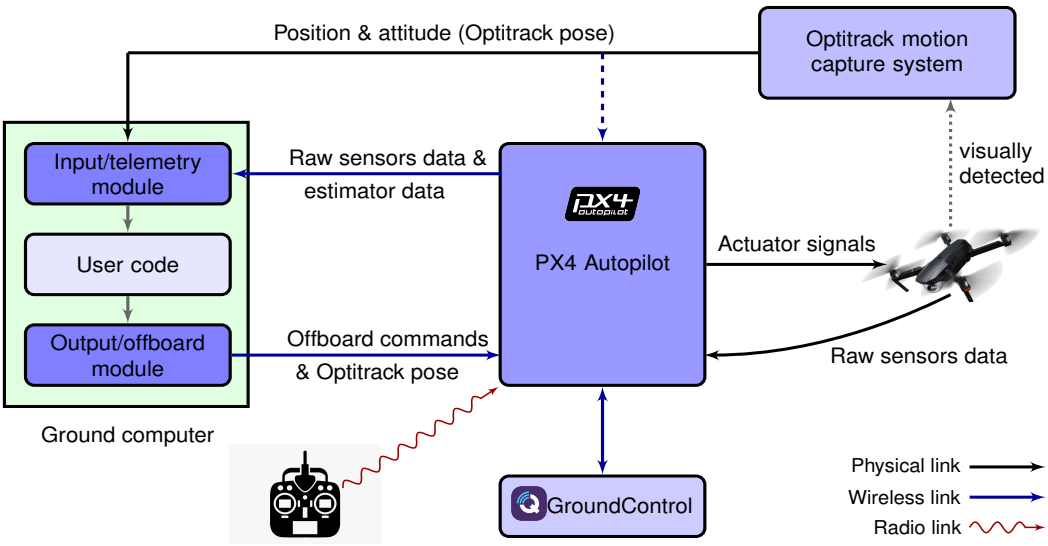

Figure 3.1: Overall architecture of the ISR Flying Arena.

The different shades of blue represent the amount of work that was required to incorporate each block into the setup. The more intense the blue, the longer the time spent setting and coding the block and its interconnections. The figure shows that, although the Optitrack motion capture system and the PX4 autopilot are commercially available products, their inclusion in the system was not immediate. These products required significant and careful configuration along with the development of solutions to decode, convert, and process the information they provide. The input and output modules were the most difficult to implement because they handle and automate communication with the Optitrack and the PX4, according to specific messaging protocols. The detailed implementation of all the blocks represented in Fig. 3.1 is presented over the following sections.

3.2 Optitrack motion capture system

The ISR Flying Arena consists in an indoor test space of dimensions 7m $\times$ 4m $\times$ 2.5m and a set of eight Optitrack motion capture cameras, positioned according to Fig. 3.3. The rapid development and prototyping environment was designed according to the NED (North-East-Down) coordinate system, which is standard in aeronautical applications. The adopted inertial NED coordinate frame, represented in Fig. 3.3, is centered in the arena, at ground level, and has the North axis along the widest side of the arena, the East axis along the shortest side, and the Down axis pointing towards the ground.

Figure 3.2: ISR Flying Arena.

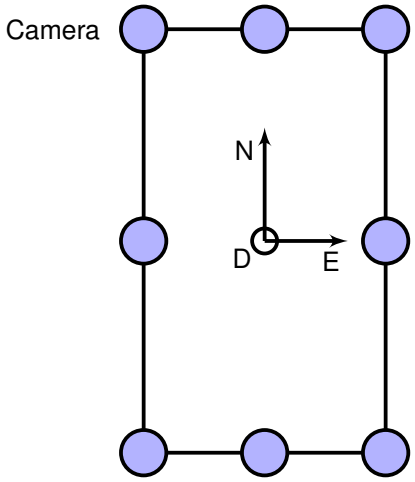

Figure 3.3: Cameras (represented as blue circles) and inertial coordinate frame of the arena.

The Optitrack motion capture system provides high-frequency and low-latency measurements of the position and attitude of the UAVs. It works according to the diagram of Fig. 3.4. First, the Optitrack cameras track special passive markers placed on the body of the vehicles. Then, this tracking data is sent, via Ethernet, to a computer running the Optitrack's Motive software. By feeding the tracking data to its advanced solvers and to its high-level filters, the Motive computes the position and attitude of the vehicles with a positional error less than 0.3mm and a rotational error less than 0.05°. Finally, the Motive sends the position and attitude data to a router that broadcasts it to the local network. Note that the Optitrack system computes the pose data according to a ENU (East-North-Up) inertial frame.

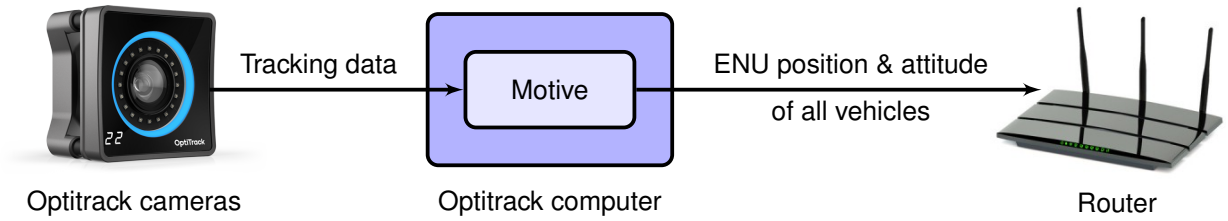

Figure 3.4: Diagram of the information flow of the Optitrack system.

The Optitrack cameras track the markers by detecting reflected infrared light. Consequently, the user has to mask, before each set of experiments and using the Motive software, all the bright spots in the arena that can be mistaken with passive markers. Additionally, the user has to calibrate the Optitrack motion capture system on a weekly basis, because the calibration accuracy naturally deteriorates over time due to ambient factors, such as fluctuation in temperature. These processes are documented in the digital repository that complements this thesis.

After being broadcasted to the local network, the pose data provided by the Optitrack system needs to be decoded and processed, so user programs and the extended Kalman filter of the PX4 can fuse it with the measurements provided by the inertial sensors to produce estimates for the position and attitude of the UAV. The decoding and processing stages developed are represented in Fig. 3.5 and Fig. 3.6. These solutions are implemented using the Robot Operating System (ROS) middleware and the MAVLink-Router [15], a library that transforms ROS topics into MAVLink streams and routes them to other endpoints, such as the PX4.

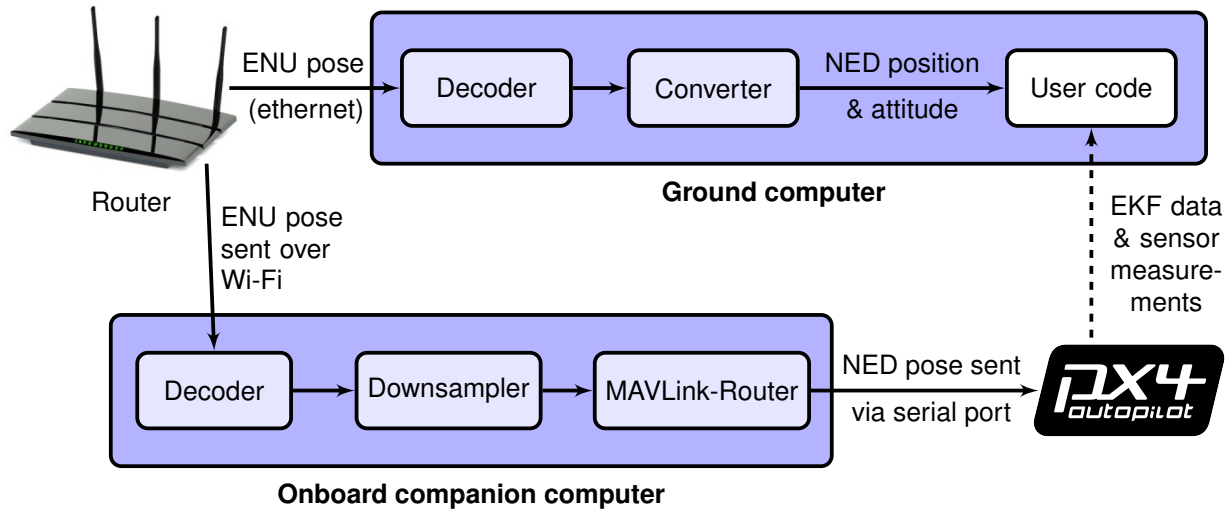

Figure 3.5: Diagram with the processing steps of the Optitrack data in the presence of an onboard companion computer.

The diagram of Fig. 3.5 shows the set of modules developed to deliver the Optitrack data to the user and the PX4, in the cases in which there is an onboard companion computer connected, through a serial port, to the autopilot. On the top part, the position and attitude of the vehicles, in ENU coordinates, is received by the ground computer through an Ethernet link. Then, a decoder block reads this information

and publishes it into a ROS topic. The decoder block was built using the VRPN (Virtual Reality Peripheral Network) client, a ROS node that connects to the VRPN server used by the Optitrack system (to stream data to the local network) and exposes the information over a ROS topic. After being decoded, the position and attitude data is converted to the NED coordinate frame, the one adopted for the testing setup, and is finally made available to the user. On the bottom part of Fig. 3.5, the companion computer receives the position and attitude data generated by the Optitrack system via Wi-Fi. The decoder block is equivalent to the one implemented in the ground computer. It was also programmed using the VPRN client that publishes the received pose information into a ROS topic. After the decoding block, the position and attitude data is submitted to a downsampling process. The Optitrack system provides positioning data to the local network at a frequency of 180 Hz. In order to avoid exhausting the bandwidth of the communication channel, which could cause delays in the communication with the PX4 and loss of packets, the downsampling block republishes the pose data into a new ROS topic, dropping two of every three messages received. Therefore, the new ROS topic receives new position and attitude updates at a frequency of 60 Hz. Finally, by using the MAVLink-Router library, the new ROS topic is transformed into a MAVLink stream and is sent, through a serial port, to the PX4 autopilot. During this step, the position and attitude are converted from ENU coordinates, used by the Optitrack system and the ROS middleware, to NED coordinates, used by the MAVLink protocol and the PX4 autopilot. Note that the sensor measurements and the output of the extended Kalman filter of the PX4 are sent, via Wi-Fi, to the ground computer. This gives the users freedom to implement its own estimation algorithms, by fusing the position and attitude data retrieved from the Optitrack system with the sensor measurements received from the PX4, or to simply use the output state estimates of the EKF of the PX4.

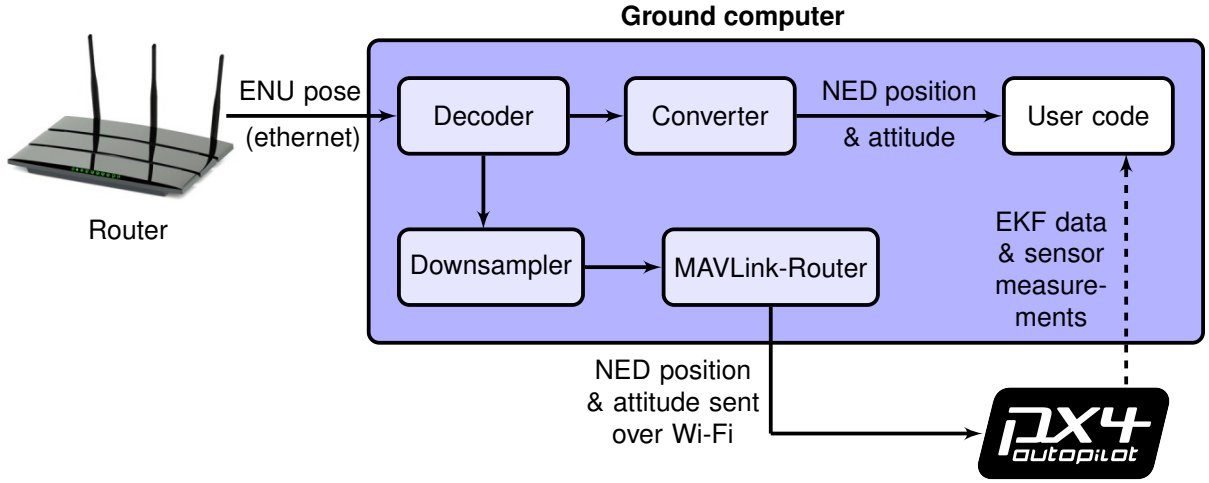

Figure 3.6: Diagram with the processing steps of the Optitrack data in the absence of an onboard companion computer.

The diagram of Fig. 3.6 shows the layout devised to deliver the Optitrack position and attitude of each vehicle to the user programs and to the PX4, in the cases in which there is no onboard companion computer. The blocks adopted are the same ones developed to the setup of Fig. 3.5. The only difference is that all the blocks run on an offboard computer. It should be noted that there is an independent process,

for each vehicle, that runs the procedure described in Fig. 3.6. This gives flexibility to the designed setup. For example, if a group of researchers is performing an experiment with multiple UAVs, they can have one ground computer per vehicle, one ground computer for all the vehicles, or a compromise between these two options.

The procedure described over Figures 3.5 and 3.6 is completely automatic. When performing an offboard experiment, the user fills and runs a configuration file, that will be presented and discussed in Chapter 5, that launches in the background the processes that decode and manage the Optitrack data. This enables the users to focus on high-level algorithms. The Optitrack position and attitude of the vehicles, the sensors measurements, and the output of the extended Kalman filter of the PX4 are automatically made available to them in the user code block.

3.3 Quadrotors, PX4, and QGroundControl

In order to perform experiments, it is necessary to integrate vehicles into the testing environment. This is primarily achieved by installing a PX4 autopilot in the vehicles and by configuring it appropriately. As stated in Chapter 2, the PX4 acts as an interface between the offboard modules and the flying capabilities of the vehicle. It decodes data from the sensors, encodes data to the actuators, provides autonomous controllers and estimators, supports safety features, and enables communication with external systems through the MAVLink protocol. In this thesis, two different quadcopters were integrated into the testing setup. The first one was the Intel Aero Ready to Fly Drone [16] exhibited in Fig. 3.7. The second one was Snapdragon, a custom built quadcopter assembled by the researchers of Institute for Systems and Robotics, illustrated in Fig. 3.8. The Intel Aero quadcopter is an example of a vehicle that features an onboard companion computer, whereas the Snapdragon is a lighter drone, that does not have any onboard computer assisting the PX4. Both drones were used in the tests documented in Chapter 6.

Figure 3.7: Intel Aero Ready to Fly Drone.

Figure 3.8: Snapdragon quadrotor.

To incorporate these two vehicles into the testing setup, a three-step procedure was followed:

1. In the first step, each PX4 was configured according to the physical properties of the UAV and in conformity with the sensors and avionics installed in the vehicle. This configuration procedure is introduced in Section 3.3.1. It includes the specification of the air-frame of the drone, the calibration of the sensors, the configuration of the failsafe features, and the optimal tuning of the internal controllers of the PX4.

2. In the second step, the extended Kalman filter of the PX4 was tuned according to the properties of the Optitrack system and the communication latency measured. This procedure is presented in Section 3.3.2.

3. Finally, in the last step, a set of modules were developed to enable communication with each PX4 and to automate the reception of odometry updates and the sending of offboard commands to the vehicle. These modules are introduced in Section 3.4 and explored in greater detail in Chapter 5.

3.3.1 Configuring the PX4

The PX4 of the Intel Aero and the Snapdragon quadcopters was configured according to the official guide [17] provided in the PX4 website. The configuration was assisted by the QGroundControl software. Since it was an extensive procedure, only the most relevant steps of the configuration, that are related with important features of the Flying Arena, will be addressed.

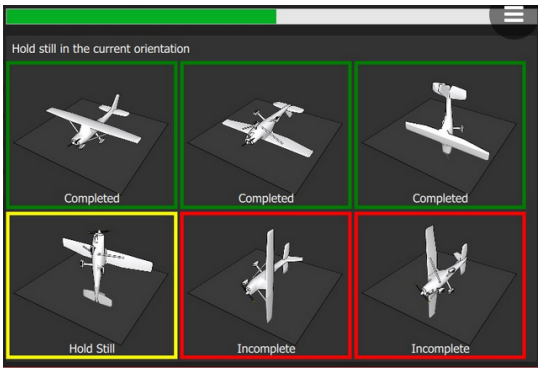

Figure 3.9: Calibration of the accelerometer using the QGroundControl.

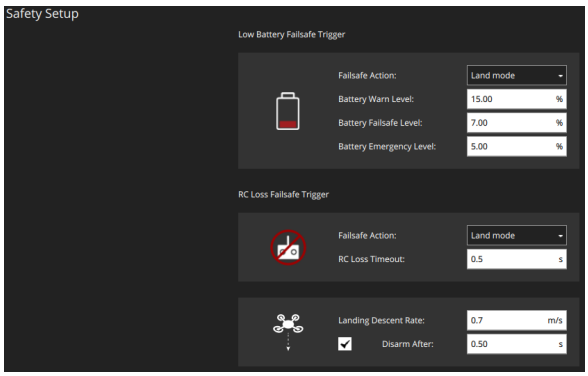

Figure 3.10: Failsafes configuration using the QGroundControl.

After uploading the most recent firmware version of the PX4 to the flight controller board, it was necessary to calibrate the sensors of the vehicle. This was a straightforward process due to a visual calibration setup provided by QGroundControl software. In order to calibrate the inertial sensors, it is only necessary to place and hold the vehicle according to a series of orientations requested by the calibration routine. Fig. 3.9 shows the setup provided by the QGroundControl software to enable the calibration of the accelerometer. The PX4 performs pre-flight sensor quality and estimator checks to verify if there is a good enough position estimate to arm and operate the vehicle. Whenever there is a poor position estimation and a calibration of the sensors is required, the users can resort to this setup to perform it.

Another important aspect that was configured in the PX4 autopilot are the failsafe modes. These safety features protect the user and the equipment when something goes wrong. Whenever a failsafe is triggered, the PX4 performs a pre-selected action, such as transition to hover mode, land the vehicle immediately, or return the vehicle to the home position. Fig. 3.10 presents the safety setup page provided by the QGroundControl to configure the failsafes. The low battery level failsafe was set to warn the user if the battery capacity drops below 15% and to transition to auto-land mode when the battery capacity drops below 7%. The RC loss failsafe is triggered if the communication link with the RC transmitter is lost and was also configured to set the flight mode to auto-land. Whenever the auto-land mode is engaged, the vehicles will vertically descent to the ground at a rate configured to 0.7 m/s. The RC transmitter, shown in Fig. 3.11, is the primarily safety link of the Flying Arena because it provides a last-resort solution to stop the most unpredictable problems through its safety switches configured to immediately put the vehicles in hover mode or shutdown the motors. Some failsafe settings cannot be configured through the QGroundControl safety setup page. These must be configured by editing the internal parameters of the PX4. Examples of this are the position loss failsafe and the offboard loss failsafe. The former is triggered if the quality of the position estimate drops below acceptable levels, whereas the latter is triggered if the offboard communication link is lost during an offboard experiment. These failsafes were configured by editing, respectively, the COM_POSCTL_NAVL and the COM_OBL_RC_ACT parameters of the PX4. If a position loss failsafe is triggered, the PX4 is set to shutdown the motors of the vehicle because a significant drop in the quality of the estimates usually means that the PX4 stop receiving data from the Optitrack and, therefore, the position and attitude estimates are no longer accurate enough to continue the experiment or even land the drone. If an offboard loss failsafe is triggered, the PX4 was configured to activate the auto-land mode. This failsafe usually occurs when the user program that sends offboard commands to the vehicle suddenly stops working. Since there is an independent process sending the Optitrack data to the vehicle, the estimator of the PX4 still produces valid estimates to safely perform an auto-landing maneuver.

Figure 3.11: Radio Controller used in the experiments.

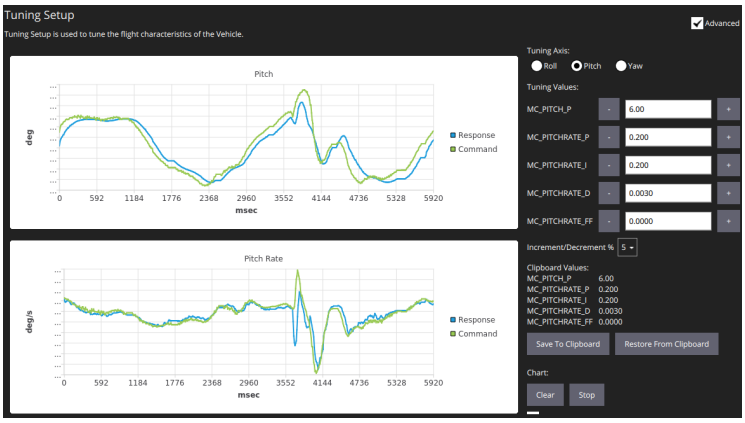

Figure 3.12: PID tuning setup available in the QGroundControl software.

Finally, the internal PID controllers of the PX4 were slightly tuned. This procedure was performed using the tuning setup of the QGroundControl, exhibited in Fig. 3.12. Since the Intel Aero is a commercial product and the Snapdragon was pre-assembled by the Institute for Systems and Robotics, these vehicles were already tuned and flight tested. However, for safety reasons, the controller gains were set to conservative values. Therefore, in this final step, the gains were better adjusted to each quadcopter.

3.3.2 Tuning the extended Kalman filter

To finish the configuration of the vehicles, the extended Kalman filter of the PX4 was tuned according to the properties of the testing setup and the Optitrack motion capture system. First, the sources of position, velocity, and attitude measurements used by the estimator were defined by configuring the bits of the EKF2_AID_MASK parameter. Since the Flying Arena is an indoor environment, a global positioning system such as the GPS is unavailable, so the source of horizontal and vertical position data was set to be the Optitrack. The Optitrack system has the added advantage of providing position information with significantly more precision and higher rates than the GPS. However, the Optitrack system does not provide velocity measurements of the vehicles. Consequently, the estimator was set to not use any source of external velocity data. Finally, two different sources of attitude measurements were configured to be used by the estimator. For roll and pitch estimation, the extended Kalman filter is automatically programmed to depend only on the IMU sensors available onboard, discarding the roll and pitch data retrieved from the Optitrack. This is due to the fact that the inertial sensors measurements are sufficient to produce satisfying low-latency and low-drift roll and pitch estimates. For yaw estimation, the EKF2_AID_MASK parameter was configured so that the estimator uses the yaw measurements provided by the Optitrack and discards the readings of the magnetometers because these are disturbed by the electric motors and affected by magnetic anomalies of the indoor environment. In summary, the extended Kalman filter of the PX4 was configured to use the position and yaw measurements given by the Optitrack motion capture system and is automatically programmed rely on the accelerometers and gyroscopes to produce low-latency and low-drift estimates of the roll and pitch angles.

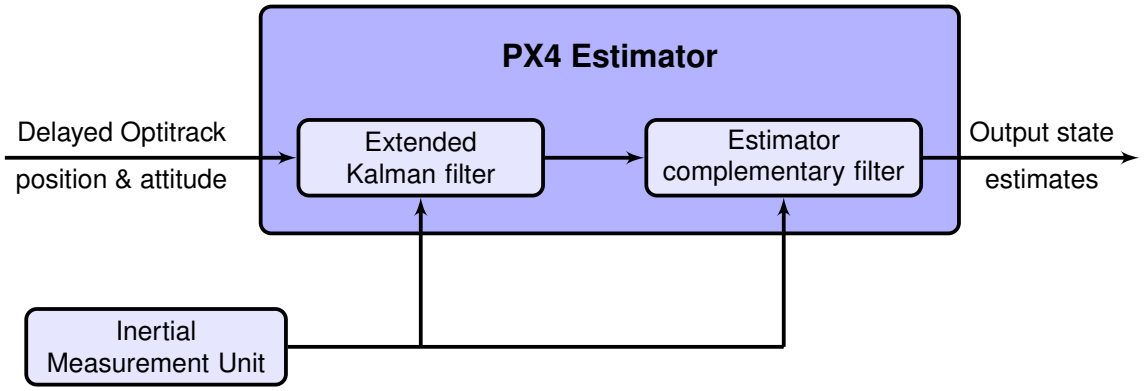

Figure 3.13: Data fusion process of the estimator of the PX4.

To enable the fusion of low-latency measurements of the accelerometers and gyroscopes with the position and yaw measurements received from the Optitrack system, that reach the PX4 with some delay due to communication overhead, the estimator fuses the data in a delayed time horizon. This means that the estimates are computed considering the time delay of the Optitrack data relative to the IMU. Then, an output complementary filter propagates the estimates forward to current time. This procedure is represented in Fig. 3.13. Consequently, it was necessary to tune the extended Kalman filter with the correct time delay between the arrival of the Optitrack and the IMU measurements. A rough estimate of the delay was obtained from the logs by checking the time offset between the IMU and the Optitrack data. Then, this value was further refined by performing a set of experiments with distinct delay values, and by verifying the resulting estimator innovations. The time delay obtained, 20ms, corresponds to the one that yielded the smallest innovations. This value was assigned to the EKF2_EV_DELAY internal parameter of the PX4.

3.4 Ground Computers

Once all the steps described in the last sections have been completed, that is, once: i) the PX4 autopilot is adapted to the physical properties of the vehicle; ii) the sensors of the drone are calibrated; iii) the failsafe modes and the radio controller switches are defined; iv) the internal controllers of the PX4 are tuned; v) the Optitrack position and attitude data is automatically being decoded and sent to the PX4 autopilot; and vi) the extended Kalman filter is correctly configured to use the position and yaw data from the Optitrack system, the UAVs are ready to fly. In the devised environment, the vehicles receive offboard control commands from user programs. These programs follow an object-oriented approach and rely on the input and output modules to communicate with the PX4, as shown in Fig.3.14. Each vehicle is represented by an instance of a class, whose attributes store the current state of the vehicle (such as the position, attitude and velocity) and the most recent readings of the sensors. The input or telemetry module consists on a set of methods that, running in background threads, keep the attributes of the class up-to-date. The output or offboard module comprises the methods that send offboard commands and control references to the PX4. Since these modules are extensive and common to both the real and the simulation environment, they will only be explored in detail in Chapter 5.

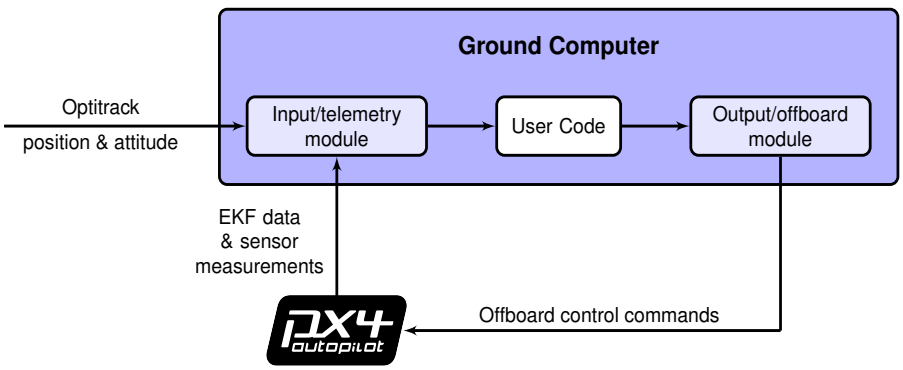

Figure 3.14: Offboard control process of the vehicles.

3.5 Detailed overview

The diagram of Fig. 3.15 aims to summarize the information presented over this chapter by detailing the flow of data between the different modules of the ISR Flying Arena, for vehicles that count with an onboard companion computer.

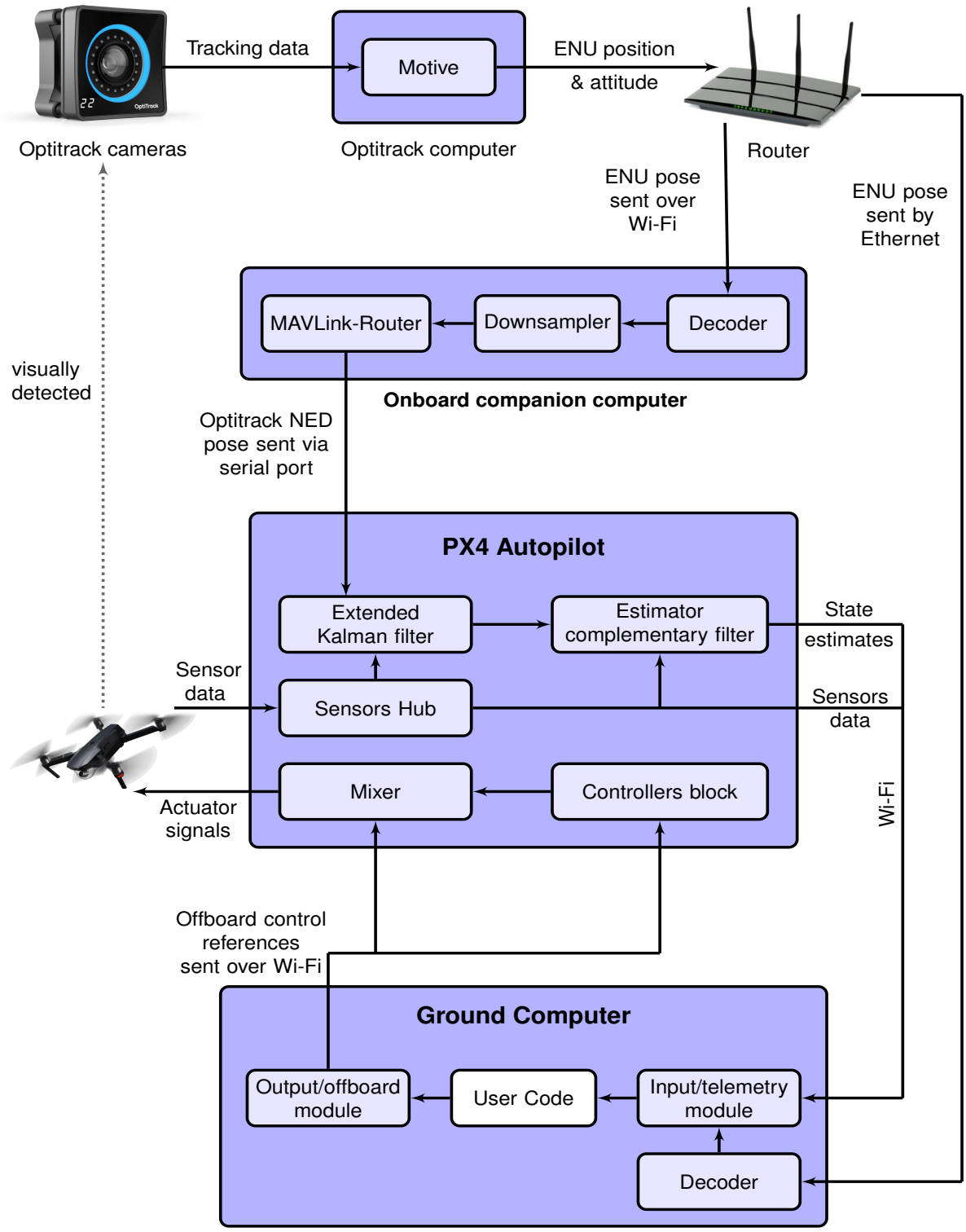

Figure 3.15: Flow of information between the modules of the ISR Flying Arena, for vehicles with an onboard companion computer.

In the absence of an onboard companion computer, the decoder, downsampler, and MAVLink-Router modules have to run in a ground computer, as depicted in Fig. 3.16

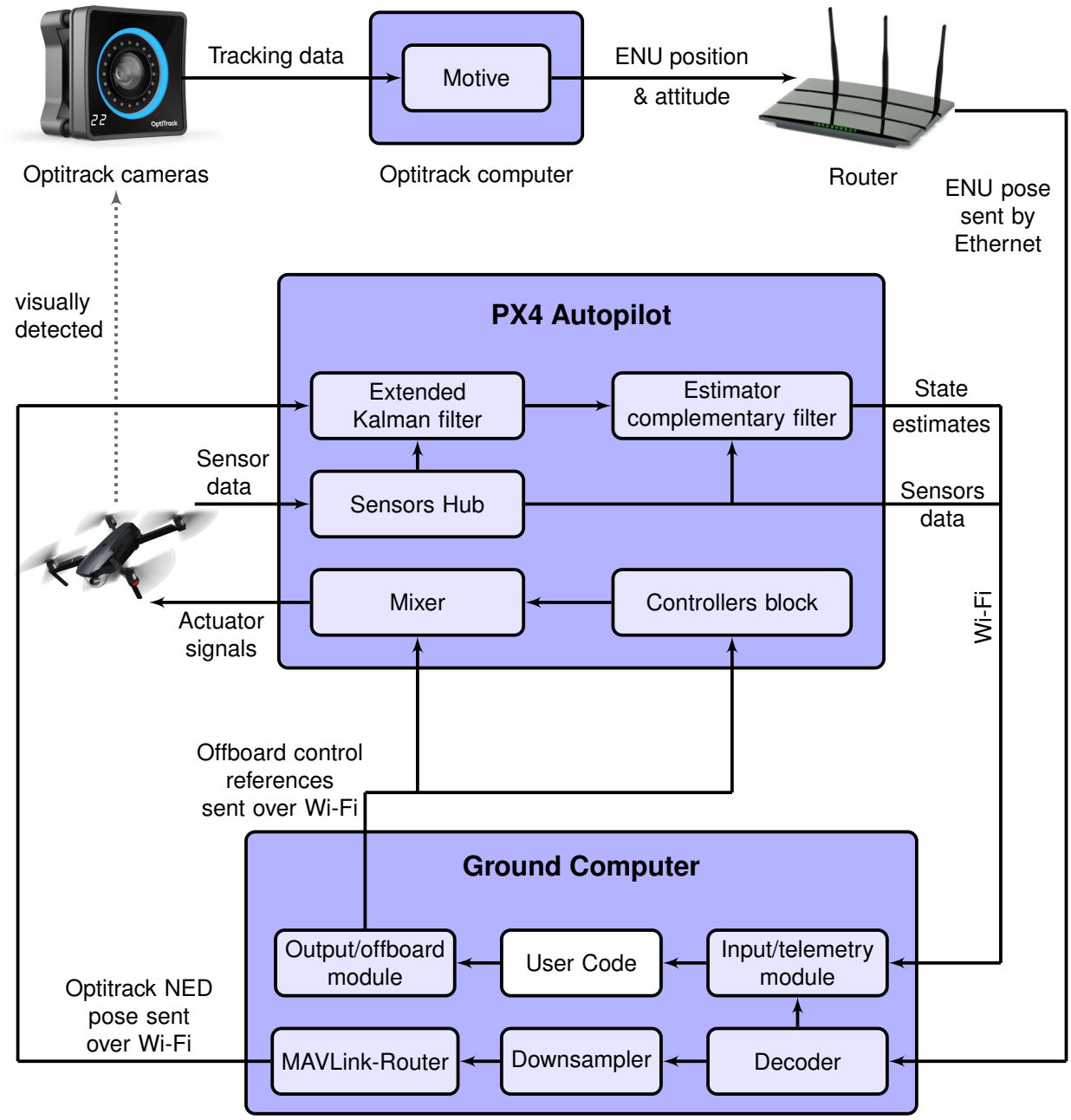

Figure 3.16: Flow of information between the modules of the ISR Flying Arena, for vehicles without an onboard companion computer.

This chapter focused on the configuration of existent commercial hardware components and their integration in the envisioned setup. For this purpose, modules were programmed to decode and process data from the Optitrack system and deliver it to the PX4 of each vehicle. The next two chapters are more focused on the software parts of the Flying Arena. Chapter 4 explores the solutions coded for simulating the testing environment. Chapter 5 presents the solutions coded, in multiple programming languages and using multiple messaging libraries, for automating the communication with the PX4 and the testing of navigation and control solutions.