LABWORK - QUADROTOR

2018-2019 version

INTRODUCTION

This document gives all the necessary information for the development and test of control, navigation and guidance algorithms with the quadrotor facility system of ISAE-SUPAERO. The final goal is to fly the algorithms with the real system (Fig 1.) after validation with a fully representative simulator SimoDrones (annex 1).

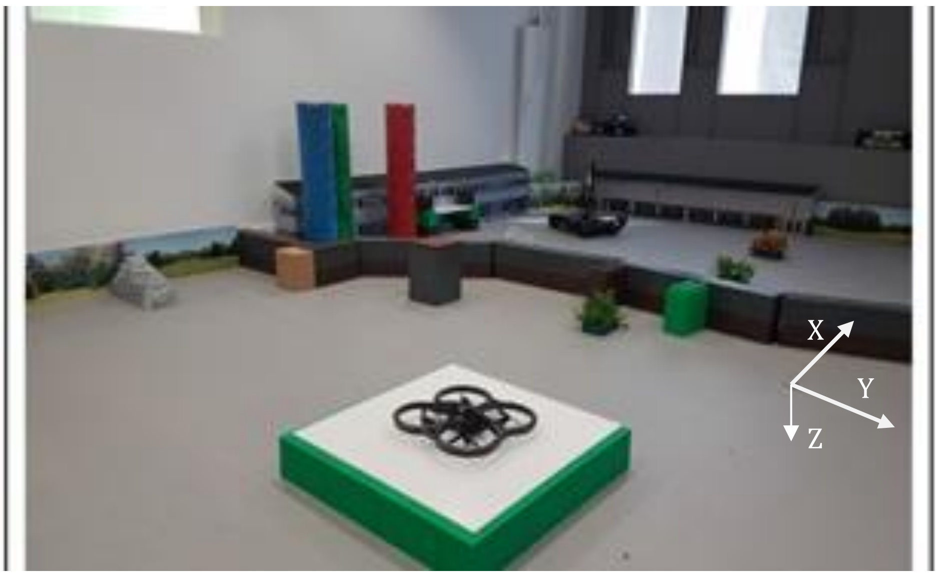


Fig 1 : The Autonomous System Platform at ISAE (room 07.009)

The system is based on a commercial low cost quadrotor (Bebp 2.0 by Parrot). For our quadrotor the control law is basically of the *cascade* type: a fast and robust attitude control loop is embedded in the Bebop controller and higher level control loops (control, guidance, navigation) are done trough a Matlab/Simulink real time program.

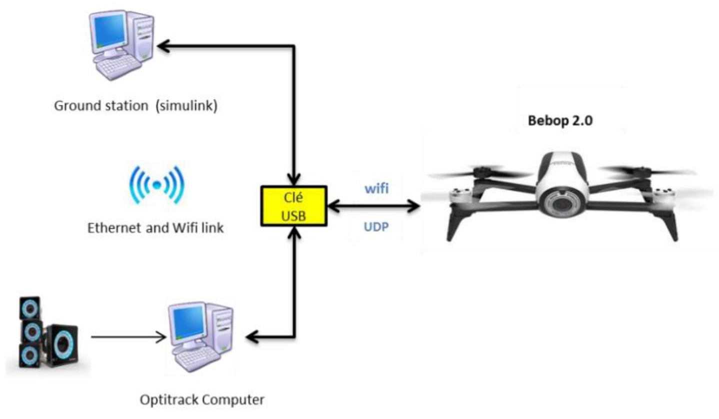
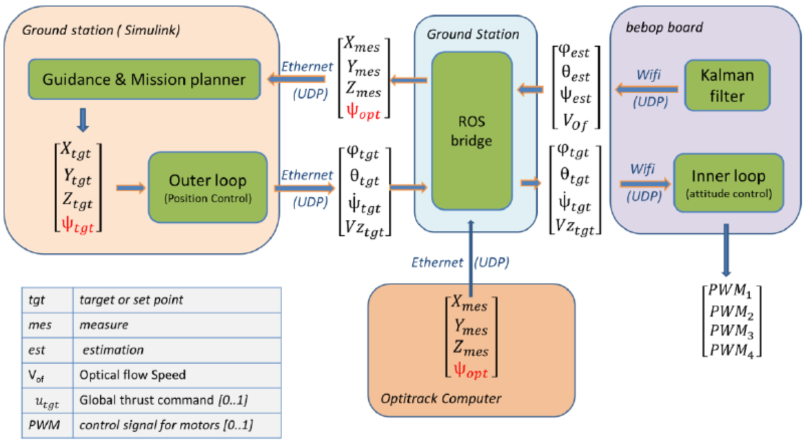


Fig 2: Hardware system of the quadrotor facility of ISAE

One can see on Fig 2. that the hardware environment involves several computers. One is the “ground station” where most of the development is done. The other is the Optitrack Computer (explained later) and the last one is the Bebop embedded controller.



tgt	target or set point
mes	measure
est	estimation
V_{of}	Optical flow Speed
u_{tgt}	Global thrust command [0..1]
PWM	control signal for motors [0..1]

Fig 3 : Software architecture

Each computer is in charge of a separate task. The Fig 3. gives the detail of the most important variables exchanged between each computer. The higher level loop (guidance and navigation) is developed in the ground station with Matlab/Simulink.

A simulation model (SimoDrones) has been developed, very accurate for moderate velocities and stable atmospheric conditions. Main physical parameters are given in Annex II. A control law must be first validated with the simulation model and then tested in the real system. The same Matlab/Simulink file is used for both simulation and real time experimentation. The parameters of the model are initialized with the script *SimulationStartUpScript.m* .

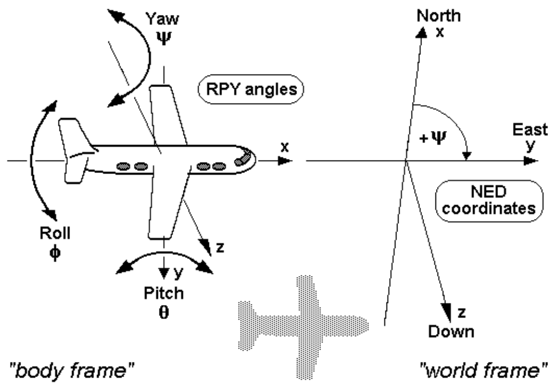


Fig 4 : Reference frames

The setup uses three different frames (figure 4):

- The inertial frame (bebop): attached to the body of the quadrotor (subscript b).
- The local frame (NED): aka "earth frame" or "world frame", attached to the magnetic north,
- The local frame (opt): same convention than NED but attached to the optitrack system as in figure 1.

The bebop is stabilized with an inner loop in charge of attitude control (θ and ϕ in rad) yaw speed control ($Vyaw$ in rad/s) and vertical velocity control (Vz in m/s), based on the bebop sensor's measurements. Higher level control (xyz position and yaw) will require more precise measurement than the IMU, provided by the optitrack system.

A safe auto-takeoff and auto-land is programmed through Boolean variables:

- the variable **TakeOff** equal to 1will generate an automatic takeoff of the drone and a stabilization at approximately 1m ,of altitude. During this phase the vertical velocity target must be set to zero.

- the variable **Land** equal to 1will generate an automatic landing. During this phase the vertical velocity target must be set to zero.

An internal variable (**bebop_state**) tells us in which state is the bebop:

0	1	2	3	4
Landed	Taking off	Hovering	Flying	Landing During this phase the vertical speed reference must be set to zero

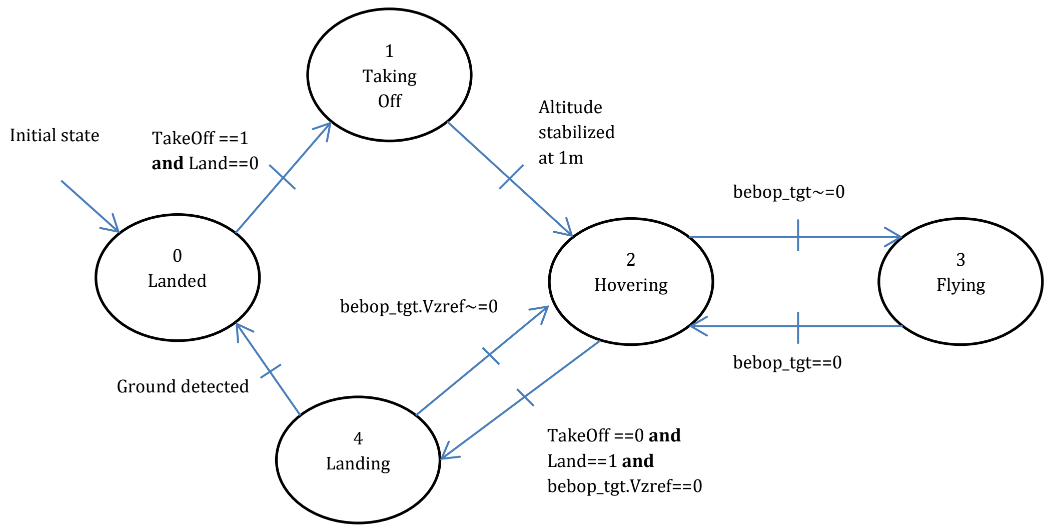


Figure 5 : bebop_state state machine

The state machine is explained in fig 5. In state 2 we will provide from the Matlab/Simulink ground station reference values for:

- the pitch and roll angle (in rad)
- the vertical velocity (in m/s)
- the yaw speed (in rad/s)

TASK I : ALTITUDE CONTROL, SIMULATION

1. Simplified model

Give a simplified model of the system in the vertical axis only. Program a test in order to identify a model for the vertical speed. You will program an automatic take off, a negative step of vertical velocity (0.1 m/s during 3s) and an automatic landing.

2. Design and test of the controller in simulation

Based on the simplified model you will design **a "cascade controller" or a PID controller**. The requirements are as follows:

- Overshoot less than 5% for a standard step input
- steady state error less than 3 cm
- Maximum vertical speed 1 m/s

The controller can be designed on a simplified Simulink model and/or in a dedicated Matlab script or on a trial / error basis.

For the validation test you will program a trajectory as follows: the automatic take off, a reference step (20cm and down) and an automatic landing.

TASK II : HORIZONTAL CONTROL, SIMULATION

1. Model identification

The horizontal displacement (x and y) will be controlled with attitude angle (att_tgt). The actual attitude is controlled with an internal loop embedded in the bebop computer you are not in charge of this part. Before starting the design of the controller we need a simplified simulation model.

The simulation models will be obtained by identification. You will analyze standard step input and derive a first order model (between att_tgt and horizontal velocity).

2. Design and test of the controller in simulation

For this axis you will design a **state feedback** controller. The requirements are as follows:

- Overshoot less than 5% for a standard step input
- steady state error less than 3 cm
- Maximum horizontal speed 1,5 m/s

The validation test will be a 20 cm horizontal step.

TASK III : FULL VALIDATION, EXPERIMENTATION

1. **Validation test**

   Test your controllers in a real flight. The provided script *plotdata.m* gives an idea of how to save and plot interesting data.

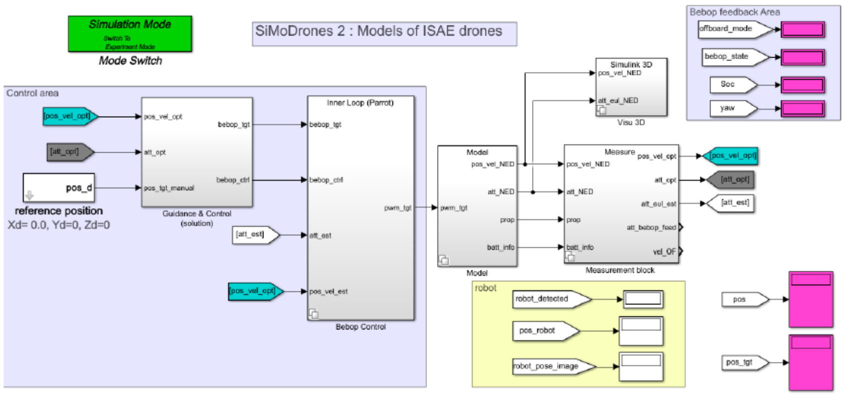
   You may have not enough time to fine tune your controller but the performance (and difference between experimentation and simulation) must be pointed out and explained.

2. **Test in navigation**

   The final test is a full navigation. A block is provided that gives a full reference trajectory.

Annex I : The SimoDrones simulator

The Matlab/Simulink model **SiMoDronesBE.mdl** is the a full simulation model of the quadrotor and its environment. A Matlab script **plot_data.m** is useful for the identification and plots of variables.



Blocks description

1. **Guidance & Navigation** : Generates target positions in the NED frame.
2. **Position Control** : altitude, horizontal position and yaw control. Provides attitude target and thrust.
3. **Attitude Control** : inner attitude loop, embedded in the PX4. It is already coded and cannot be modified during this labwork. Control laws are basically cascade loops with feedforward term.
4. **Model** : Full simulation model of the quadrotor: actuators, quadrotor dynamic, battery.
5. **Measurement Block** : Full model of sensors, including delays and noise.
6. **Simulink 3D** : 3D visual feedback.

List of variables

pos_vel_opt	position (X, Y, Z) in meter (m) and speed (Vx, Vy, Vz) in m/s
att_opt	attitude (roll, pitch, yaw) in radian (rad) and angular speed (p, q, r) in rad/s

att_est	attitude (roll, pitch, yaw) and angular speed (p, q, r) in the body frame estimated by the bebop
Vel_OF	Speeds (Vx, Vy, Vz) estimated by the optical flow in the NED frame

tgt : *target variables (target)*

bebop_tgt	Roll in body frame Pitch in body frame yaw_rate in body frame Vz : vertical speed in NED
Bebop_ctrl	Attitude target (roll, pitch, yax, throttle) send to the inner loop (embedded to the px4). "throttle" corresponds to the mean value sent to the motors.
pos_tgt_manual	position reference (X,Y,Z) in the optitrack frame.

Ned : *variables in the NED frame*

pos_vel_NED	position (X, Y, Z) and velocity (Vx, Vy, Vz) in the NED frame
att_NED	attitude (roll, pitch, yaw) and angular speed (p, q, r) in the NED frame

Other variables

prop	Rotation speed of the propellers.
Batt_info	Battery state (voltage, current, charge)
Pwm_tgt	Actuator reference normalized between 0 and 1
Offboard_mode	Equalm to 1 when the control by Matlab/Simulink is set
Soc	Battery status (between 0% and 100%)

Annex II : Bebop data

Total mass : 510 g

Inertia: I_x = 0.00^18 kgm^2

I_y = 0.0018 kgm^2

I_z =0.0033 kgm^2

Actuator model: 1st order with time constant T= 0.02s

Sampling time: Te=0.02s

Mean total delay: Tr = 0.02s

Propeller thrust efficiency coefficient : Pw = 1.96e-06 N/(rad/s)^2

Propeller torque efficiency coefficient : Mw = 2.6 N.m/(rad/s)^2