OKLAHOMA STATE UNIVERSITY
School of Mechanical & Aerospace Engineering

MAE 5010 - Autopilot Design

Course Project (2019 Fall)
Autopilot design and test

This document formalizes requirements for MAE 5010 course project. The course project consists of designing and deploying a simple autopilot for use in research applications. Students will work on assigned teams of approximately 4 students to achieve shared infrastructure objectives and individual research goals.

Contents

1 Project scope and work areas

In this project, your group will design and implement an estimator and controller on a provided flight vehicle using the hardware. Individually, you use this autopilot to address one of your research goals.

1.1 Work areas

This section contains more detail on these individual areas.

Flight mechanics model Define the dynamics and model the aerodynamics, choose a reference flight condition (S+L cruise), then generate a simplified model (using either block diagram and assumptions (a.la. Beard) or linearization)

Controller design Your team is assigned either a lateral-directional controller OR a longitudinal controller. Identify another team who is working on the other controller type; you will integrate their controller. There is no need for you to design both longitudinal and lateral controllers.

Choose a controller structure, based on either the lateral-directional A/P discussed in class or either of the longitudinal forms. Select the gains using your choice of classical control techniques (either root locus or loop shaping), or modern control theory (eg, pole placement, LQR), and verify in simulation.

Estimator design Design an estimator to generate a state estimate based on the sensors available on your Raspberry Pi/Navio2 board combination.

Customization (individual portion) In this phase, you will customize your autopilot implementation and gather in-flight data that serves your research. One approach is to pick the block closest to your research interests and upgrade it. Examples may include

- Envelope expansion: Specializing the autopilot to a different condition: e.ge, landing flare to conduct automated landings, aerobatic motion, etc.
- Adding a parameter uncertainty to include a robustness analysis
- Running a custom estimator routine in parallel with the traditional estimation approach to compare in-flight performance
- Incorporating a more intelligent trajectory planner (e.g, Dubins path, etc)
- Multi-aircraft coordinated control (e.g, stabilizing two aircraft in formation)
- Incorporating a more involved MAVlink messager.
- Flight testing of a sensor, e.g, camera-based target detection, radar-based obstacle avoidance,etc[1]

Students are encouraged to submit a 1/2 page description of their plans for feedback; detailed feedback will not be possible on your topic without this submission. For students not already assigned to a research project, you are encouraged to discuss with the instructor areas of current research and resource availability to support related work.

2 Platform & Resources

2.1 Equipment

A flight platform is provided. This unit is based on a commercial RC airframe with a Raspberry Pi 3B+ and Navio2 daughter board integrated. The airframes provide traditional control surfaces: elevator, aileron, throttle, rudder, and flaps. A 900MHz band radio is included for telemetry streams to ground. If the individual portion of your project requires additional equipment, discuss these with your advisor and the instructor.

[1]Sensor installation must not require airframe modifications and must be able to be moved to a different test aircraft within 3 minutes.

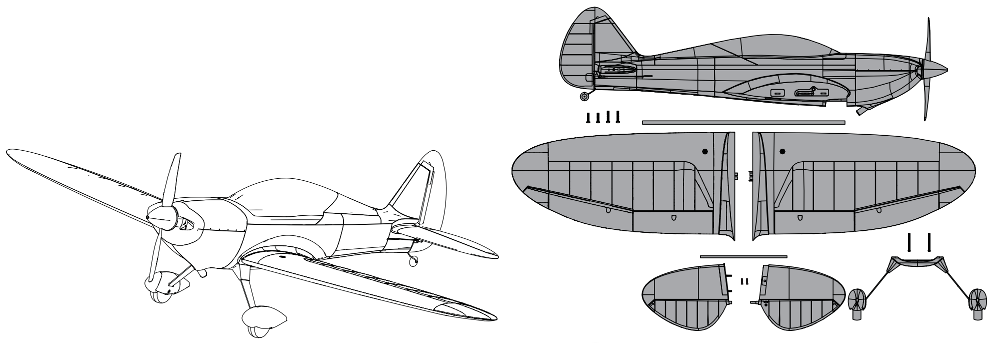

Figure 1: Airframe.

2.2 Software

The software framework follows the structure of the autopilots discussed in our class, seen in Fig. 2.

Several elements have been provided in in a supplied software environment. The software environment provided is a Python environment running on a Linux-based kernel with the RT PREEMPT patch. This environment provides shared multi-process access to several values you will use:

$$\hat{x} = [(x, y, z), (V_T, \alpha, \beta), (\phi, \theta, \psi), (p, q, r)]^T \tag{1}$$

$$y = [(a_x, a_y, a_z), (p, q, r), (m_x, m_y, m_z), p_{bar}, (x_n, y_e, z_d), (v_n, v_e, v_d)]^T \tag{2}$$

$$\text{servo} = [\text{man/auto, rcin, servosout}]^T \tag{3}$$

2.2.1 Software integration

A manual over-ride circuit has been developed that allows students to integrate their code within an airborne environment. The entirety of your estimation and feedback code must run inside a designated section in this Python script,[2] and no modifications may be made to the manual over-ride portions of this code. The two elements of your software may be summarized as follows:

Estimator read y, write $\hat{x}$. Your estimator will be the only code that writes/updates $\hat{x}$. y is updated by a sensor read routine.

Controller read $\hat{x}$. IFF `man/auto` flag is set to auto, update `servosout`. The manual over-ride code also writes to `servo`. **Your controller *must not* update `servo` when the `man/auto` flag is set to manual.** This functionality will be tested, and no credit will be provided to submissions whose operation interferes with the manual over-ride circuit. It will be helpful to compute an intermediate value

$$u = [\delta_e, \delta_a, \delta_t, \delta_r]^T$$

that is then converted to servo pwm values.

[2]C drivers are provided for future upgrades, however, an environment containing the above variables is not currently available.

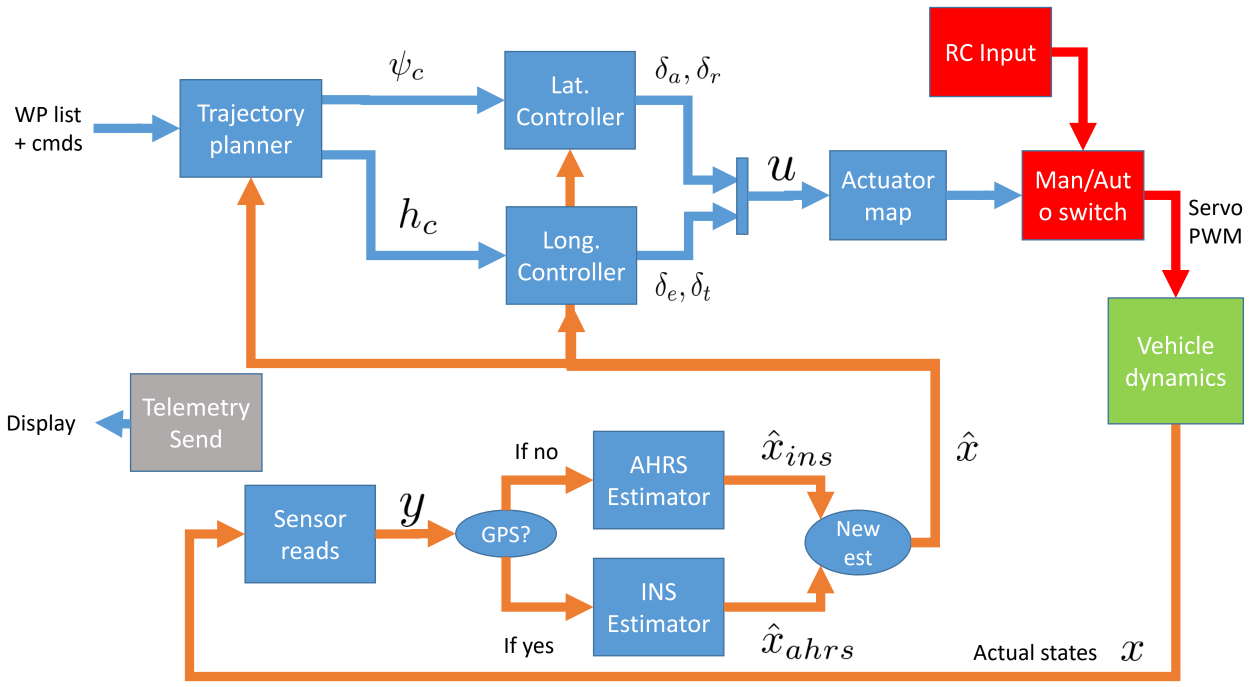

Figure 2: Autopilot layout.

The software environment also provides a rudimentary telemetry stream through Mavlink messages to update a Mavlink compatible groundstation such as Mission Planner, or QGroundControl. GPS waypoints to follow will be provided.

2.3 Flight test opportunities

3 flight test days will be provided at the OSU UAS Unmanned Aircraft Flight Station in Glencoe, Stillwater. The tests are designed to allow you to explore the performance of your estimator and autopilot.

2.3.1 Test purpose

Estimator in-flight test The above estimator will be flown onboard an aircraft and must record sensor readings and state estimates.

Autopilot flight test The combined autopilot and estimator will be flown on-board an aircraft while controlling the aircraft. The vehicle must record sensors readings, state estimates, auto/manual status, and control surface commands.

2.3.2 Location

GPS Location: lat=36.1632229 lon=-96.836514 Airfield Directions: From all directions, the best way to get to the UAFS is to take Clay road north from the intersection of Hwy 51 (E 6th st.) and N. Clay Road. The UAFS is approximately 3.2 miles, shortly north of the intersection of N. Clay Rd. and E. Airport Rd.

2.3.3 Test day allocation

Test days will nominally be scheduled during class periods and the dates adjusted for ideal flight weather. All test days will have the same structure proceeding in team order. Each team is expected to receive 3 flight opportunities with approximately 5 minutes in-air each. If an on-deck team is not ready within 2 minutes, the following team will be called and the on-deck team recycled.

The first will be a practice day. No grades will be assigned based on performance; you are encouraged to use this to validate your estimator. The second will be an estimator in-flight test day. The remaining day will focus on autopilot testing. However, if a team successfully achieves an element, they may immediately use their next flight opportunity to advance to the next test, provided the code for this next test has been peer-verified. To enter a flight attempt, the team must indicate their desire for a flight to be considered by submitting an evaluation sheet to the instruction and judging team prior to the time they begin a flight attempt.

3 Flight test sequence

The flight test sequence consists of a rectangular circuit involving two different altitudes defined by four 3-D GPS waypoints. This test will be conducted while running your software on a class airframe. There may be multiple aircraft operating, and the waypoints will not be available until the day of the test.

Your submission on flight test days will be a micro-SD card containing your team's operating system and peer-verified code. You will need to boot your software, connect any required communication links, and initiate your code. After a pre-flight check, a manual pilot will initiate the airframe takeoff, and transition flight control to your code. The manual pilot will disengage your code when you indicate completion of your test or when an unsafe condition is created, whichever comes first. You will be returned your microSD card; in some cases, return may be delayed to verify code versions.

4 Submission

The following items will be submitted.

Code Code submissions will be submitted for peer-verification. Each team's code must be verified by at least one other team prior to demonstration.

Group report The group report will summarize the infrastructure development–ie, the shared elements of the project. The appendix must include the autopilot code that was tested. Typical reports are 7-11 pages (not including appendices); no report will be accepted less than 4 or over 13 pages.

Individual report The individual report will present only the research-related study. Include only the customizations and results from your investigation; reference the group report for anything contained in it. No code appendix is required for the individual report. Typical reports are 2-4 pages; no report will be accepted less than 1.5 pages or over 5 pages.

Component	Weight
Group report	70%
Flight test outcome	5%
Individual report	25%

Table 1:

5 Grading and required elements

Your individual project grade will be computed from the following components

5.1 Group Report

Two required elements

1. Autopilot design process is described, focusing on estimator and controller. This description should include performance specifications chosen, reasoning and analysis to support gain choices, and quantitative analysis (e.g., plots) documenting the expected performance. All plots must be labeled and legible.

2. Flight test discussion compares controller performance from expected (via analysis, simulation)

5.2 Code

(Group) code submissions will be dispositioned as follows:

1. Code shows all required elements 55%

2. Code is documented such that a peer may work with 5%

3. Code runs without significant errors and generates output 20%

4. Correctness of resulting output (from plots in group report) 20%

5.3 Flight test

Flight tests will be dispositioned as follows:

1. Code runs

2. Code runs and generates usable output

3. Code runs, generates output, and adequately performs the assigned test sequence.