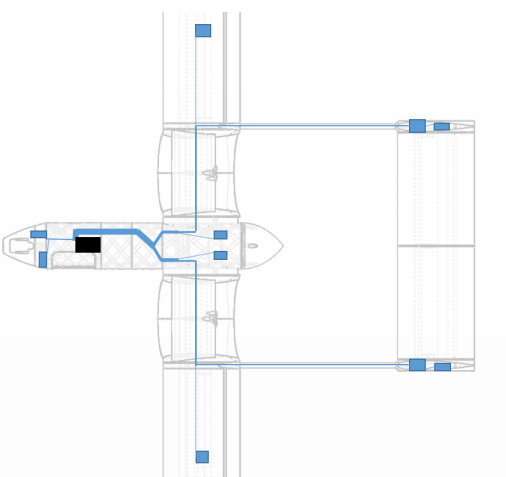
Technical report
Computational Engineering and Design group, FEPS
University of Southampton

Custer UAV avionics instructions

13th April 2021

Juraj Mihalik, jm1e16@soton.ac.uk
Supervised by Prof A.J. Keane



Abstract

In this report, the avionics installed in the Custer UAV is described. Besides the hardware overview, wiring diagrams and radio controls, the pre-flight instructions, equipment checklist and list of useful parameters are included. On top of that, basic overview of the QGroundControl control station is provided, as well as overview of the custom stabilisation module and instructions for HITL simulation with the Custer airframe in X-Plane.

Contents

1 Hardware overview

1.1 Controller

The installed controller box consists of the Pixhawk 4 controller and a power distribution board shielded by an aluminium frame (Figure 1.1). The factory supplied micro SD card was replaced by a faster SanDisk Extreme U3 32GB micro SD card for more reliable data logging.

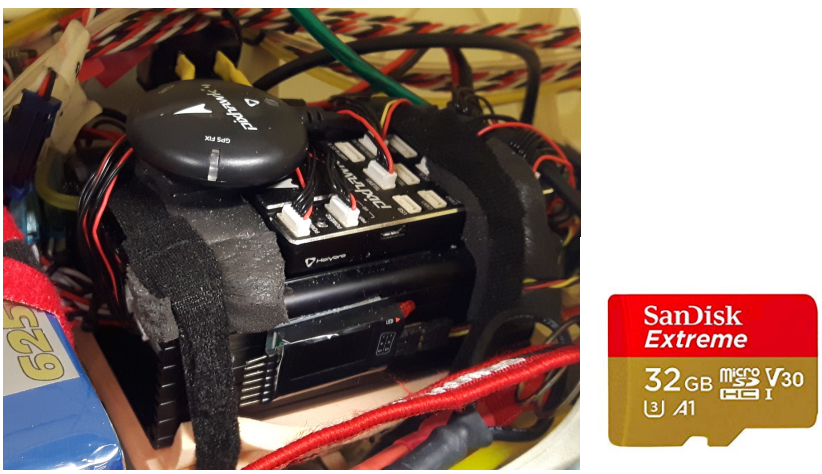


Figure 1.1: Controller box.

1.2 Power source

The electric motors are powered by a 6S Overlander LiPo of capacity 6250 mAh. The avionics systems are powered by a 2S Overlander LiPo of capacity 5000 mAh. W note that we use exclusively high voltage digital servos (Futaba S3470SV, Savox SC–1268SG, HiTEC D940TW, MKS HBL380) that are suitable for this kind of power source. The ignition is powered by a separate 2S NanoTech LiFe battery with capacity of 2500 mAh. The whole battery bundle shown in Figure 1.2 is removable for easier charging and swapping for a spare one.

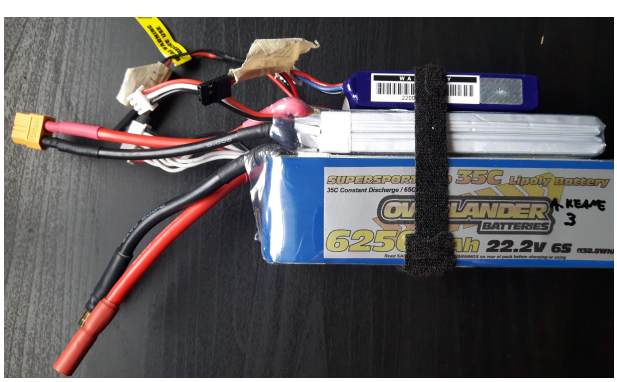


Figure 1.2: Battery bundle.

1.3 Front engine

The OS GF30 front engine on the mount with vibration damping with the ignition and fuel intake lines visible is shown in Figure 1.3. For ignition we use recommended OS IG07 module, and the fuel used is a 40:1 mixture of petrol and 2 stroke fully synthetic motor oil.

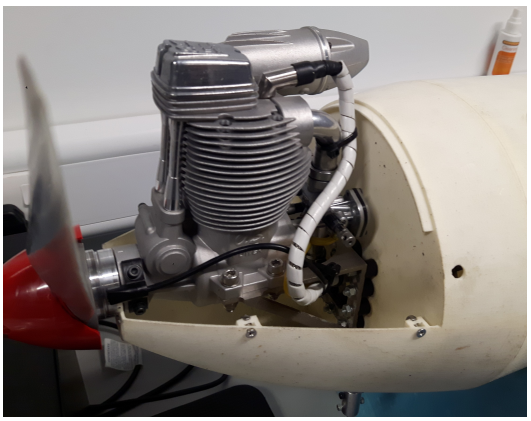


Figure 1.3: Front engine on the mount with vibration damping.

1.4 Electric motors systems

To spin the 14"x7 three bladed MasterAirscrew propellers in the Custer ducts we use the Hackers A50 12sV3 480KV brushless electric motors seen on Figure 1.4. The speed controllers used to operate these motors are the Jeti MasterSPIN 99 Pro OPTO ESCs with a large heat sinks necessary due to lack of cooling around them in the current aircraft design. ** Later swapped for the MasterMezon95 Opto ESCs for more telemetry data.

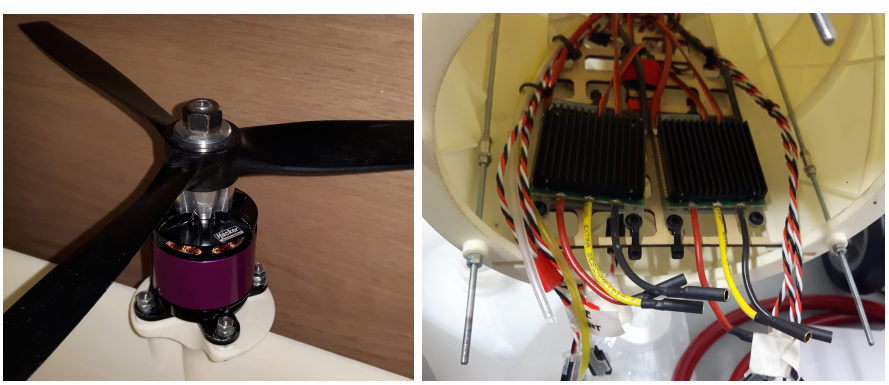


Figure 1.4: Electric motor and speed controllers.

1.5 Ground control station

A ground control station (GCS) consists of a Mac laptop with the QGroundControl (QGC) software installed, and a telemetry transceiver module connected by USB, as shown in Figure 1.5.

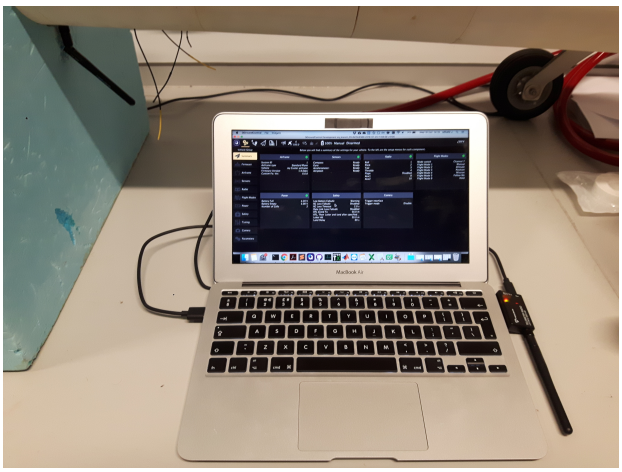


Figure 1.5: Ground control station.

1.6 RC transmitter

Originally, we used the Taranis X9D Plus transmitter with the X8R receiver as shown in Figure 1.6.

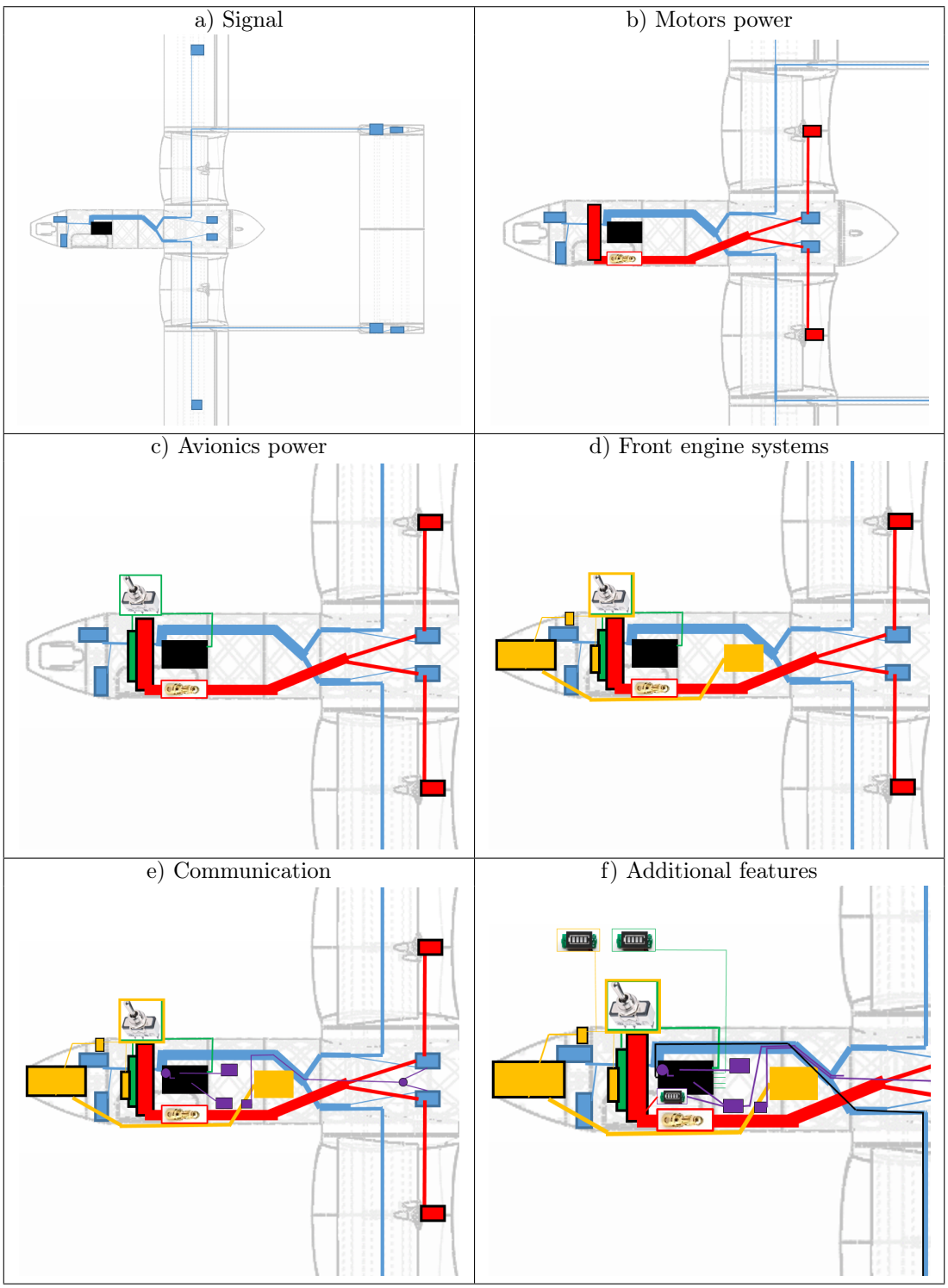
Figure 1.6: Taranis X9D Plus with the X8R receiver.

This was later replaced by the Futaba T18MZ transmitter with the R7008SB receiver as shown in Figure 1.7.

Figure 1.7: Futaba T18MZ with the R7008SB receiver.

In both cases, the receiver connects to Pixhawk using the SBUS connector. The functionality of both transmitters for Custer is the same and the model setup files can be found in the provided attachments (see Table 10.3). It is recommended (even though not necessary) to re-run the radio calibration in the QGC when changing between the transmitters as the PWM limits are not matching exactly. In the Futaba transmitter, switching between the PX4 flight modes using the SE,SF,SH sticks is not supported (as this was only used in quadcopter and wasn't needed in Custer) (see Section 4). Moreover, the SH and SF stick functions are interchanged as SH is a button on Taranis and a lever on Futaba and vice-versa.

2 Wiring diagrams



a) Signal is sent from the controller to the actuators through servo wires as shown in blue.

b) The power from the 6S LiPo battery to the electric motors is led through the bullet connectors and ESCs as shown in red. The bullet connectors constitute a switch.

c) The avionics power from the 2S LiPo battery is led through a switch to the controller as shown in

green. The controller is then powering all the servos through the servo wires. This holds also for the Opto ESCs that require power input with their signal.

d) The front engine is powered by the circuit shown in orange. The power from the 2S LiFe battery is led through the switch to the ignition. The fuel is passed from the fuel tank in the centre of the fuselage to the front engine by a fuel hose shown in yellow as well. The throttle level is operated by a servo already depicted before in blue.

e) The communication is provided by the items shown in purple. This includes GPS module, telemetry transceiver and RC receiver connected to the controller, and a separate Jeti duplex receiver connected to the ESCs through a Jeti expander.

f) This figure shows additional features incorporated in the avionics system. This includes two LiPo sensors led to the fuselage wall to indicate status of the avionics and ignition batteries. An additional FrSky telemetry LiPo sensor is included to pass the main battery status from its balance lead to the RC receiver, sending the info to the RC transmitter. We can also see featured an extra servo lead from the RC receiver to the back of the fuselage for redundant RC receiver (not included at the moment as this interfered with the information on channel 16). Three extra power outputs are provided from the back side of the controller to be able to power any additional devices such as cameras or optical sensors (Figure 2.1). The pitot tube hoses led from the front of the controller to the left wing (port) are shown in black. The static pressure hose is left unconnected as the currently used pitot tube does not have a static pressure port.

The signal outputs from the controller can be seen in Figure 2.1. We can see that the rudder channel output is split three ways for the two rudders in the tail and the nose wheel. The port rudder signal is next passed through a signal reverser due to the opposite servo mounting orientation. This could be replaced in future by reprogramming the corresponding digital servo using a servo programmer. ** It has been reprogrammed using the Futaba transmitter.

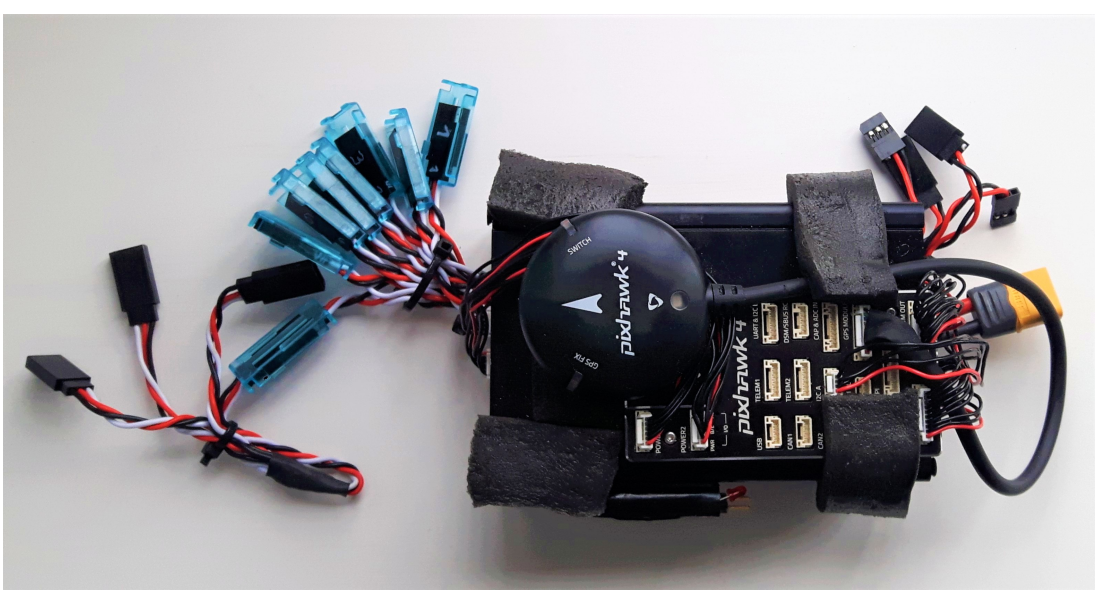


Figure 2.1: The controller box with the signal outputs.

The wiring guide for how to connect up the equipment with the Pixhawk in the controller box can be found in the PX4 documentation pages [3] or in the manufacturer's manual.

3 Equipment checklist

3.1 Equipment necessary to flight the aircraft

	Item	Note	Tick
1	RC transmitter	charge	
2	Ground station (laptop + QGC + telem)	charge	
3	LiPo packs	charge	
4	LiPo chargers + power leads + XT60, servo and bullet connectors		
5	Toolbox + assembly bolts		
6	Manual fuel pump		
7	Fuel + 2 stroke synthetic oil (40:1)		
8	Motor starter + 12V car battery	charge	
9	12V car battery for LiPo charging	charge	
10			
11			
12			
13			
14			

3.2 Optional equipment

	Item	Note	Tick
1	Jeti-Box for ESC telemetry	charge	
2	Spare ESCs, spare servos		
3	Spare servo wires and connector clips		
4	Camera + SD card	charge	
5	DC-AC converter for 240V power socket + laptop charger + RC charger		
6	Spare Pixhawk and receiver for HITL simulation in Xplane		
7			
8			
9			
10			

4 Radio Controls

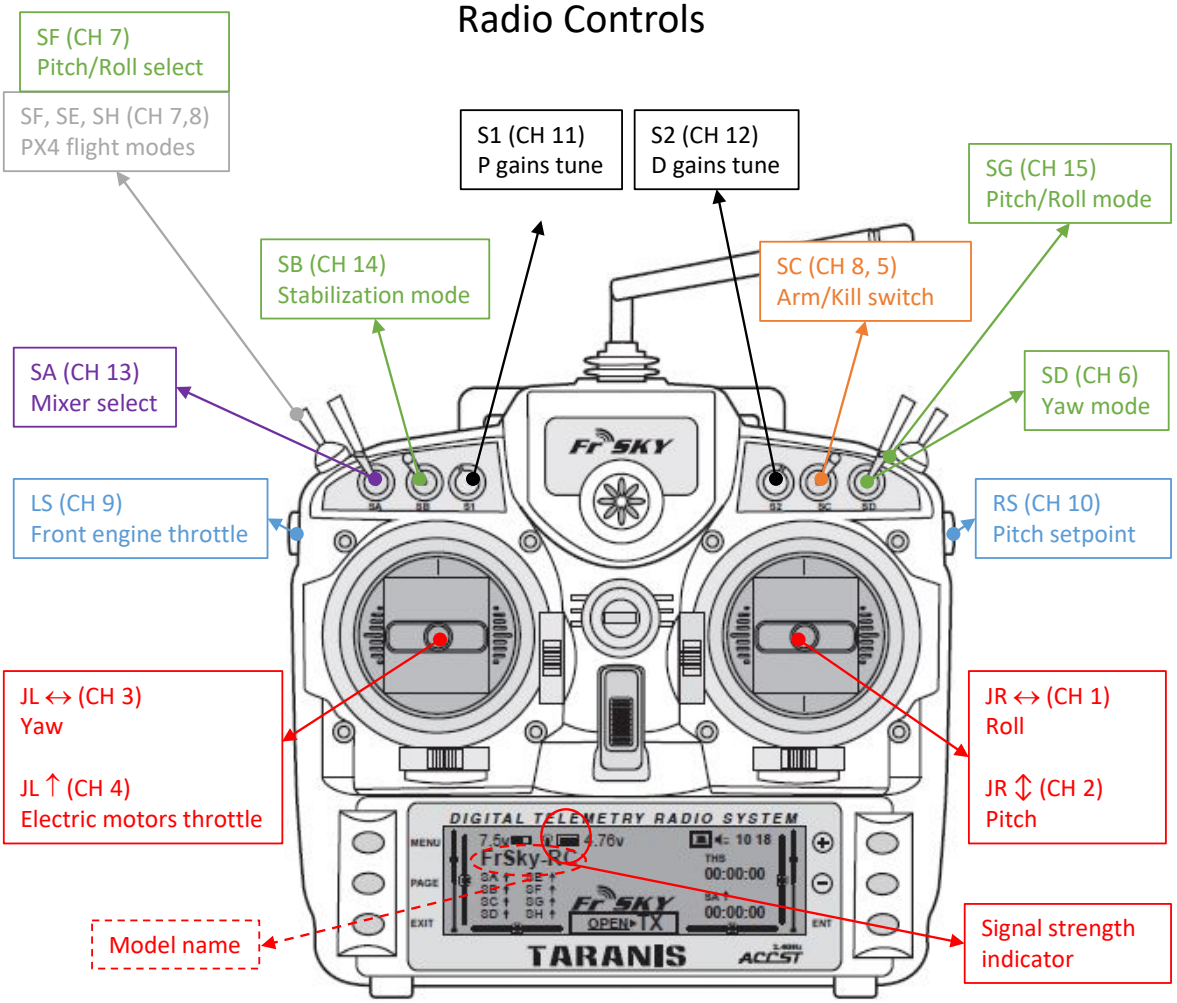
4.1 Mixer

The Pixhawk in manual (pass-through) mode maps the radio inputs to the actuator outputs in a customizable mixer file [9]. The mixer logic for Custer UAV is defined as follows.

Pixhawk 4	
RC IN	PWM OUT
+ Throttle el motors (CH 4) +0.0·Roll (CH 1) +0.3·Yaw (CH 3) *	CH 1 Motor left
+ Throttle el motors (CH 4) −0.0·Roll (CH 1) −0.3·Yaw (CH 3) *	CH 2 Motor right
+ Throttle engine front (CH 9)	CH 3 Engine front
− Roll (CH 1)	CH 4 Aileron left
+ Roll (CH 1)	CH 5 Aileron right
+ Pitch (CH 2) −0.3·Roll (CH 1) **	CH 6 Elevator left
− Pitch (CH 2) −0.3·Roll (CH 2) **	CH 7 Elevator right
− Yaw (CH 3)	CH 8 Rudders

* The amount of roll and yaw mixing for differential thrust can be modified using the `MY_LQR_MOTORONSP` and `MY_LQR_MOTORONSR` parameters respectively.
** The amount of roll added to split the elevators can be modified using the `MY_LQR_TAILERONS` parameter. (For more info on parameters see Section 6.)

Radio Controls



SF, SE, SH: PX4 flight modes (not supported in Futaba)
The Custer control is not set up in PX4, used only with quadcopter
SC: Arm/Kill switch
÷ disarmed, ↑ armed, ↓ kill (same as disarmed for Custer)
SA: Mixer select
↑ standard plane, ÷ add tailerons, ↓ add differential thrust and tailerons
SB: Stabilization mode
↑ manual, ÷ stabilized attitude, ↓ ~~PX4 control (not set up for Custer, used only with quadcopter)~~ **stabilized altitude
SD: Yaw mode
↑ yaw stabilization off, ÷ yaw damping only (D), ↓ full yaw stabilization (P,D)
(SB must be in stabilized)
SG: Pitch/Roll mode
↑ pitch/roll stabilization off, ÷ pitch/roll damping only (D), ↓ full pitch/roll stabilization (P,D)
(SB must be in stabilized)
SF: Pitch/Roll select (SH on Futaba)
↑ applies the mode selected above on pitch, ↓ applies the mode selected above on roll
(SB must be in stabilised)
S1: P gains tune
tune the proportional gain, -1...$base^{-1}$, 0...$base^{0}$, 1...$base^{1}$, ($base = 10$ default)
S2: D gains tune
tune the derivative gain, -1...$base^{-1}$, 0...$base^{0}$, 1...$base^{1}$, ($base = 10$ default)
LS: Front engine throttle (LLS on Futaba)
-1(down)...0, 1(up)...1
RS: Pitch setpoint (RLS on Futaba)
-1(down)...0, 1(up)...tht_sp_m, ($tht\_sp\_m = 40$ default)
JL ↑: Electric motors throttle
-1(down)...0, 1(up)...1

5 Pre-flight instructions

5.1 General notes and possible issues

5.1.1 RC transmitter

- Before switching on the transmitter, make sure that the arming switch *SC* is in the disarmed middle position ÷ and the throttle levels are zero (down↓).
- Switch on the transmitter and press any key to ignore the switch warnings. Make sure that in the Menu the *pixhawk v5R2* model is selected. (Ignore this in Futaba, make sure *custer01* is selected.)
- After powering on the aircraft, you can verify that the connection is established observing the signal strength indicator.
- The connection may be lost if the transmitter is too close to the aircraft radio antenna.
- The vehicle may not be fully responsive to RC commands, if it was powered on before the transmitter. In such case try to restart in the correct order.
- The vehicle takes few seconds to boot after powering on before it starts to respond to RC commands.

5.1.2 Arming

- The vehicle needs to be armed in order to provide motors and engine outputs.
- Arm by putting the arming switch *SC* to upward position ↑. Disarm by putting it to the middle position ÷.
- To verify the vehicle is armed, try spinning the motors. Alternatively, use the front engine throttle stick and observe the throttle servo moving. Alternatively, notice the throttle servo level moved from zero to idle level. Alternatively, observe the vehicle status in the QGC.
- The arming will be denied without any warning, if the throttle levels are not zero. If this happens, put the arming stick back to disarmed and try again.
- The arming will be denied if the USB is connected or if the avionics battery is low.
- The arming may not work if the vehicle was powered on before the transmitter was on.
- The arming may not work if the arming switch was already on when turning on the transmitter.
- The electric motors may not spin even in armed state if the main LiPo was connected before powering on the Pixhawk.
- For another reasons of failed arming, see the warnings in QGC. (The PX4 firmware performs various consistency checks before arming is allowed. Even though most of them were disabled to prevent this behaviour, there may be more surprises. If such event should be a reason for denied arming, you should be informed in the information window in QGC or as a background text in the Fly view of the QGC. To resolve the issue make sure your transmitter is connected, try a few restarts. See if the vehicle status responds for example to the *kill* command. Try arming using the slider in the Fly view in QGC. Use google to resolve the warnings.)
- The vehicle will not disarm on it's own under any circumstances (the landing detector and termination circuit are disabled).
- The data logger starts automatically on arming.

5.2 Pre-assembly instructions

Before assembling the aircraft, use QGC to make sure that all the sensors are calibrated, and load the correct firmware version as needed. Make sure that the vehicle info corresponds to the aircraft you are flying. Any firmware or calibration issues should be resolved now, as it is easier to deal with the fuselage on it's own than with the assembled aircraft full of fuel. If the airspeed sensor is not calibrated, you can do this after assembling.

1. GCS on → Power on.
2. Check system info in QGC, sensors calibrated, disarmed status check, battery status check, response to console inputs check, parameters check, pwm info check.(Section 7)
3. Power off →GCS off.

5.3 Assembly instructions

1. Connect the wings, tighten the bolts.
2. Connect servo wires and motor wires on the back of the fuselage (easily accessible without the tail on).
3. Connect the tail with the wires, tighten the bolts.
4. GCS on → RC on → Power on → Main LiPo on → Arm.
5. Check spin directions adding throttle on RC. (Both propellers should spin outwards, i.e. the starboard wing cw and port wing ccw looking from tail view, can also infer from the propeller geometry). Fix issues by reordering the motor wires.
6. Disarm → Main LiPo off.
7. Check control surfaces deflections are as expected by p,q,r inputs on the RC, fix issues by checking the servo connections.
8. Check the feedback responses (perturb the aircraft in stabilised mode and observe reactions), check no vibrations are transmitted to the control surfaces. (Bear in mind the filter needs few seconds to boot.)
9. Check GPS satellites are connected (using QGC or green light flashing on the GPS module). (Not essential for flight but good for log.)
10. If the airspeed sensor not calibrated, do it now.
11. RC range check.
12. RC-loss failsafe check.
13. Power off → RC off → GCS off.
14. Fuel up.
15. Close the fuselage end cone with bolts (only after you checked the spin directions!).

5.4 Start instructions

1. Pitot tube push in, telemetry antenna fold out.
2. GCS on → RC on → Power on → Main LiPo on → Arm.
3. Fuselage hatch close.
4. Engine start.
5. Check GPS satellites, logger status, no filtering error warnings in the console.
6. Check no vibrations get transmitted in the stabilised mode, turn the D-gain knob to estimate the admissible operating range without vibrations. (Bear in mind the filter needs few seconds to boot.)
7. Perform the flight. Watch the Jeti-Box telemetry for ESC temperature and main LiPo voltage.
8. Disarm → Main LiPo off → Power off → RC off.

5.5 Flight instructions in stabilised mode

- Cruise: Use the front engine and zero pitch setpoint.
- Hover: Use the electric motors on half throttle and pitch setpoint 20 degrees (middle stick position). Front engine idle or slight thrust. Vary the motors thrust for ascent/descent and pitch angle for speed up/slow down. ** Switch the altitude stabilisation on if desired using the SB stick down.
- Hover landing: Decrease the motors throttle slightly. Increase the throttle to slow down the descent. Even on full throttle the upward acceleration will be tiny so descend slowly. Add full throttle on the front engine to abort landing.
- Short take-off: Use full throttle on front engine, after the initial boost add full motors throttle and pitch up.

6 Parameters

6.1 List of parameters

The most important parameters that we use to set up the controller properties follow. It is advisable to check that they attain the desired values before each flight.

Parameter	Default	Description
MY_LQR_CTF_OMG	3.0	Angular rates filter cut-off frequency [Hz]
MY_LQR_THT_SP_M	40.0	Pitch setpoint maximum [deg]
MY_LQR_K_SC_P	1.0	Gains scale p
MY_LQR_K_SC_Q	1.0	Gains scale q
MY_LQR_K_SC_R	1.0	Gains scale r
MY_LQR_K_SC_PHI	1.0	Gains scale phi
MY_LQR_K_SC_THT	1.0	Gains scale theta
MY_LQR_K_SC_PSI	1.0	Gains scale psi
MY_LQR_K_SC_CCP	1.0	Gains scale cross coupled (off diagonal) proportional terms
MY_LQR_K_SC_CCD	1.0	Gains scale cross coupled (off diagonal) derivative terms
MY_LQR_MOTORONSP	0.0	Mixer roll to differential thrust
MY_LQR_MOTORONSR	0.3	Mixer yaw to differential thrust
MY_LQR_TAILERONS	0.3	Mixer roll to split elevators
MY_LQR_TUNE_EX	10.0	Gains tuner knob exponential base
MY_LQR_BOOL_SCHD	1	Use the gain scheduling? 1/0
MY_LQR_RC_SCALE	1.0	Scale the RC input to change responsiveness to commands
MY_LQR_TUNE_MOD	210	RC tuning of specified axis or their combination
SDLOG_PROFILE	27	Logging topic profile, log everything

For more detailed description of the parameters used in the custom module, please see the parameters file in Table 10.1.

The custom parameter settings used to fly the Custer UAV in the past can be also loaded from the backup files (Table 10.4) using the QGC. The backup files contain also the sensor calibration values, so given that no changes to the airframe were made, one can just load them to a new Pixhawk and fly. The parameters used in flights are also stored in the log files (Table 10.4), if needed for analysis.

For a full parameters list including their description, please consult the PX4 parameters list documentation pages [11]. We note that mainly the stabilisation parameters my be irrelevant, as we are using our own stabilisation module. The lower level sensor parameters and estimator properties are still used, they however most likely don't need to be modified.

We note that we force-saved the COM_ARM_IMU_ACC, COM_ARM_IMU_GYR and COM_ARM_MAG to high values in order to prevent arming issues on the airfield. The landing detector is disabled by setting COM_DISARM_LAND to -1. The NAV_RCL_ACT is set to 0 as the RC failsafe is taken care of within the custom module.

6.2 Changing parameter values

To change a parameter or check its value use the MAVLink console in QGC (Section 7.4) connected through telemetry or USB. For example, to change the filter frequency for angular rates, type

```
param set MY_LQR_RTS_CTF 5
```

The console output will be

```
param set MY_LQR_RTS_CTF 5
```

if the value was 5 already, or

```
param set MY_LQR_RTS_CTF 5
+ MY_LQR_RTS_CTF: curr: 4.0 -> new: 5.0
```

if you just changed it from 4 to 5.

Another way to change and check parameters is to look them up in the Parameters tab of the Setup view in the QGC (Section 7.2). We note, however, that especially the custom module parameters may be difficult to find this way.

7 QGroundControl overview

The overview provided here is by no means exhaustive and for full description please consult the documentation [14].

7.1 Main Toolbar

The main toolbar with view-select icons and status icons is shown on the top of the screen (Figure 7.1). Use this to navigate between the views and to check the arming state, battery status, GPS signal and warnings.

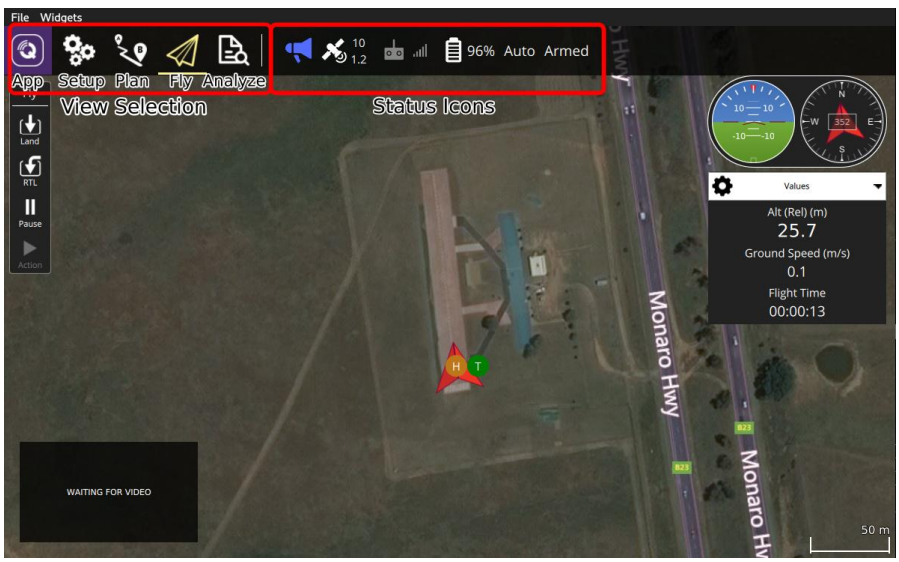


Figure 7.1: Main Toolbar.

7.2 Setup view

Use this to configure a new vehicle prior to flight and view the vehicle summary. The configuration includes airframe setup, sensors calibration, radio setup, flight modes and failsafes setup and parameters setup. The configuration for the Custer UAV needs no changing and is stored in the parameters backup file (Table 10.4). Under the parameters tab in the setup view it is possible to load or save the vehicle setup from/to a backup file.

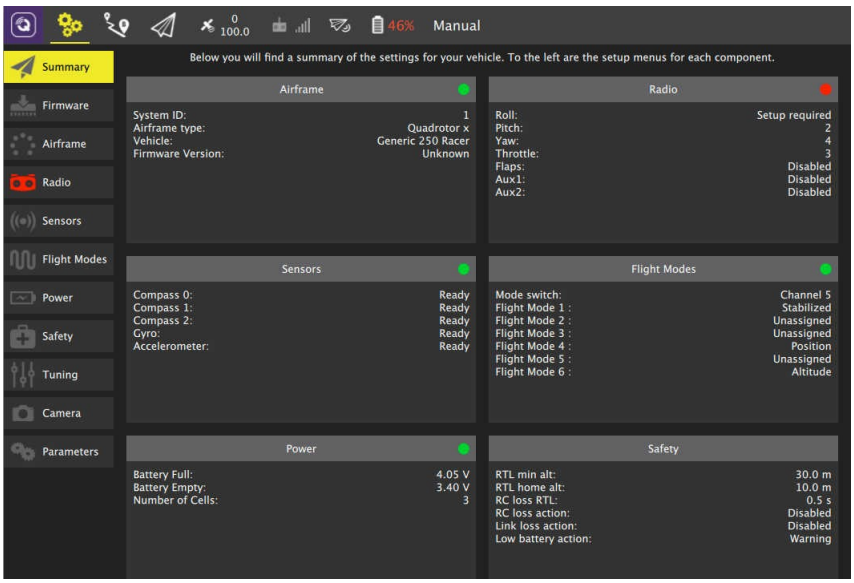


Figure 7.2: Setup view.

7.3 Fly view

The fly view is used to monitor the vehicle when flying. As shown already in Figure 7.1 we can follow the ground speed, altitude, attitude and vehicle position on the map. In this view it is also possible to use the status icons to send an arm/disarm command or mode switching that is however not necessary for the Custer UAV.

7.4 Analyze view

MAVLink Console tab of the Analyze view we use to send commands and communicate with the vehicle. Mainly, we will see here printouts from the custom stabilisation module (Section 8.2), and we can send commands to change the parameter values (Section 6.2). Other useful commands include `free` to see the current system memory status, `top` to monitor the current CPU and memory usage and `pwm info` to see the current PWM outputs to the actuators. The custom stabilisation module status can be checked by `my_LQR_control status`, turned off by `my_LQR_control stop` or started in the Custer mode by `my_LQR_control start -p 2`.

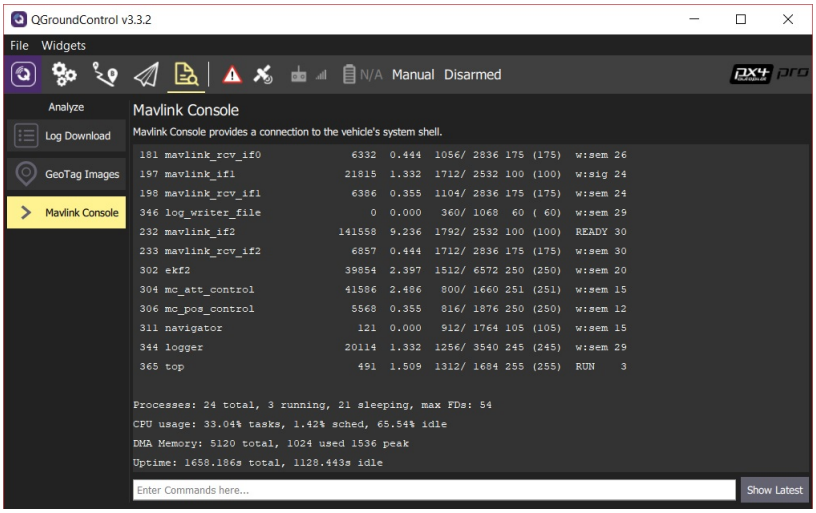


Figure 7.3: MAVLink console in the Analyze view.

Besides of the MAVLink console, the Analyze view features a Log Download tab. We tend to analyze the downloaded logs using an online utility called Flight Review [7], or process them to .csv files using pyulog's `ulog2csv` command in Mac terminal [12]. Data logging starts automatically on arming.

8 Custom controller in PX4

8.1 PX4 Flight stack overview

The software used to control the aircraft on the Pixhawk 4 controller board is the PX4 Drone Autopilot running the NuttX real time operating system. The software architecture of PX4 can be broadly divided into three layers. The abstraction layer provides drivers for communication with the sensors, motor and servo actuators, other peripheral hardware and the operating system. The libraries layer provides lower level supporting tools used by modules to facilitate sensor readings, parameters access, communication, messaging, data logging, actuator controls mixing, state estimation or more complex mathematical operations. The modules are the outermost layer consisting of programs for the aforementioned tasks, but also the control logic, navigation, or mission planning. Each module runs on the processor as a separate task of defined priority with its allocated memory stack. The memory is shared between all modules and they communicate through the uORB asynchronous publish/subscribe message passing API. Since the modules wait for the message updates, it is typically the sensor drivers that define how fast the modules update.

The PX4 code structure is shown in Figure 8.1.

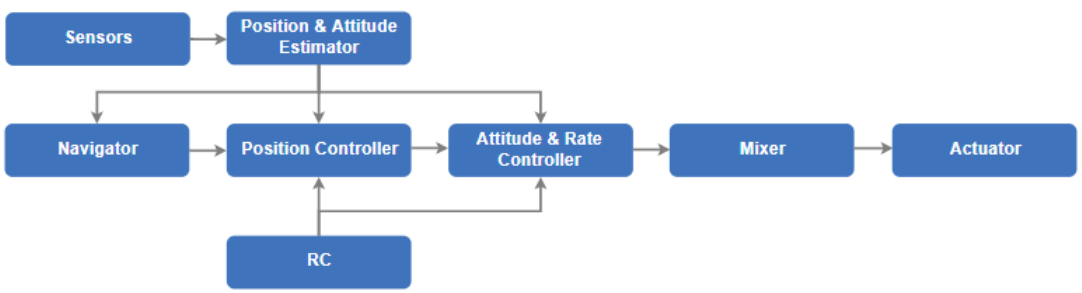


Figure 8.1: PX4 flight stack building blocks.

A custom stabilisation module overrides the PX4 controller as shown in Figure 8.2.

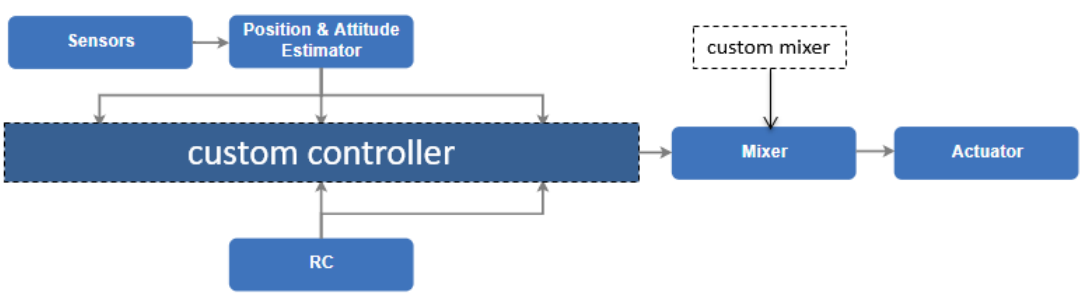


Figure 8.2: PX4 flight stack with the custom module.

8.2 Custom stabilisation module

The main loop of the custom stabilisation module is structured as shown in the code snippet in Figure 8.3. The loop iteration is triggered by receiving new information from the `vehicle_attitude` topic that updates whenever new sensor readings become available. The purpose of the listed functions is explained in the comments. For the complete custom module code please see the `my_LQR_control.cpp` file in Table 10.1.

```
while (!should_exit()) {
    timer_clock();
    read_y_state(); // read state from sensors
    read_setpoints(); // read setpoints from RC sticks
    gains_tune(); // gains tune based on RC knobs
    gains_schedule(); // gains schedule based on pitch angle
    controller_mode(); // manual / full feedback / pitch and yaw on/off/damping based on RC switches
    control_fun(); // computes the actuator controls
    px4_override(); // overrides the controls by the PX4 controller based on RC switches
    supporting_outputs(); // front engine and tailerons and differential thrust to mixer based on RC switches
    rc_loss_failsafe(); // actuator controls to zero if RC loss longer than 2 seconds
    publish_topics(); // published control outputs and custom messages
    printouts(); // printouts to MAVLink console for debuging
    update_parameters();
}
```

Figure 8.3: Custom stabilisation main loop.

The default gains matrices are stored in the `initialize_variables` function. If there is a need to change these, please modify the `K_feedback_y` and `k_scheds` arrays accordingly. We note that while the `K_feedback_y` array stores the whole gains matrix for a single operating point case, the `k_scheds` array contains values for the gains scheduler based on the pitch setpoint angles stored in `tht_ints` array. The first index in `k_scheds` represents the component of the gain matrix `K_feedback_y` ordered as shown in Figure 8.4, while the second index represents the corresponding pitch angle for which this gain is taken. Later, a `k_sw` array was added to store the gains for the motors control for altitude hold in slow flight; only the nonzero components shown in Figure 8.4 need to be considered.

$$\texttt{K\_feedback\_y} = \begin{array}{c} \begin{array}{cccccccccccc} \mathrm{x} & \mathrm{y} & \mathrm{z} & \mathrm{u} & \mathrm{v} & \mathrm{w} & \mathrm{p} & \mathrm{q} & \mathrm{r} & \phi & \theta & \psi \end{array} \\ \begin{pmatrix} & & & & & & \mathrm{pp} & & \mathrm{pr} & \mathrm{p}\phi & & \mathrm{p}\psi \\ & & & & & & & \mathrm{qq} & & & \mathrm{q}\theta & \\ & & & & & & \mathrm{rp} & & \mathrm{rr} & \mathrm{r}\phi & & \mathrm{r}\psi \\ & & & \mathrm{mz} & & \mathrm{mw} & & & & & & \end{pmatrix} \end{array} \begin{array}{c} c_{\mathrm{p}} \\ c_{\mathrm{q}} \\ c_{\mathrm{r}} \\ c_{\mathrm{TM}} \end{array},$$

$$\texttt{k\_scheds(:,j)} = \begin{pmatrix} \mathrm{pp} \\ \mathrm{pr} \\ \mathrm{rp} \\ \mathrm{rr} \\ \mathrm{p}\phi \\ \mathrm{p}\psi \\ \mathrm{r}\phi \\ \mathrm{r}\psi \\ \mathrm{qq} \\ \mathrm{q}\theta \end{pmatrix}, \texttt{k\_sw} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \mathrm{mz} \\ 0 & \mathrm{mw} \end{pmatrix}.$$

Figure 8.4: Gains matrices and the gains schedule array.

The actuator controls are computed in essence as

$$c = c_{sp} + \texttt{K\_feedback\_y} \cdot \Delta y,$$

or in the expanded form as

$$\begin{pmatrix} c_{\mathrm{p}} \\ c_{\mathrm{q}} \\ c_{\mathrm{r}} \\ c_{\mathrm{TM}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ c^{N}_{\mathrm{TM}} \end{pmatrix} + \begin{pmatrix} & & & \mathrm{pp} & & \mathrm{pr} & \mathrm{p}\phi & & \mathrm{p}\psi \\ & & & & \mathrm{qq} & & & \mathrm{q}\theta & \\ & & & \mathrm{rp} & & \mathrm{rr} & \mathrm{r}\phi & & \mathrm{r}\psi \\ & \mathrm{mz} & \mathrm{mw} & & & & & & \end{pmatrix} \begin{pmatrix} \Delta\mathrm{x} \\ \Delta\mathrm{y} \\ \Delta\mathrm{z} \\ \Delta\mathrm{u} \\ \Delta\mathrm{v} \\ \Delta\mathrm{w} \\ \Delta\mathrm{p} \\ \Delta\mathrm{q} \\ \Delta\mathrm{r} \\ \Delta\phi \\ \Delta\theta \\ \Delta\psi \end{pmatrix},$$

where the c_{sp} are the thrust setpoints given directly by the radio control sticks and the Δy vector contains the errors between the desired and measured states. Up to the moment only the attitude and altitude components of `K_feedback_y` matrix are non-zero, hence the position control is inactive. If the gain scheduling is active, the `K_feedback_y` matrix will be changing depending on the pitch setpoint angle θ_{sp} received from the transmitter.

8.2.1 Other functions

It is noted that the custom stabilisation module was created with regard to the flexibility needed when designing the control strategy for the developing Custer UAV platform. Gains scaling is supported parametrically using the `MY_LQR_K_SC_` params shown in Section 6, or by tuning using the knobs on the transmitter as shown in Section 4. The gains can be tuned in groups, or for each axis individually depending on the `MY_LQR_TUNE_MOD` parameter. Full state feedback is supported, including the option for specifying the the cross coupling compensations in the `K_feedback_y` matrix. The gains can be scheduled in the extended range of pitch angle from -120 to 120 degrees. Mixing of the tailerons and differential thrust can be adjusted in flight using the parameters shown in Section 6. Stabilisation of individual axes can be disabled or switched to the damping only mode using the transmitter switches as shown in Section 4. Custom pitch trim input can be specified using the `MY_LQR_CQ_TRM` parameter. Even though most of these functions served mainly for testing and are no longer needed for the Custer UAV operation, the functionality is still preserved until the code is refactored to a lighter version for the final product deployment.

9 Custom controller integration to PX4

9.1 Integration steps

Here we list the steps needed to integrate the custom stabilisation module and supporting component to the PX4 flight stack v1.9.

- Create the custom module files and add them to the `Firmware/src/modules` folder as per instructions in [10]. Use the `actuator_controls_0` message topic to publish the desired control outputs and `actuator_controls_1` to publish additional supporting control outputs. Don't forget to add your module name to the `default.cmake` file in the `Firmware/boards/px4/fmu-v5` folder in order to be included in the compilation.
- Create your custom mixer files and add them to the `Firmware/ROMFS/px4fmu_common/mixers` folder as per instructions in [9]. Add their names to the `CMakeList.txt` file in the same folder. The mixer files are used to map the `actuator_controls` channels to the `actuator_outputs` channels.
- Create your custom airframe files and add them to the `Firmware/ROMFS/init.d/airframes` folder as per instructions in [5]. Add their names to the `CMakeList.txt` file in the same folder. The airframe file should state the custom mixer file to be used and the custom module to be started automatically on the system start-up. It can also predefine a set of default parameters, for example the PWM limits for the airframe can be defined here.
- Create your custom messages definitions files (used in your custom module) and add them to the `Firmware/msg` folder as per instructions in [8]. Add their names to the `CMakeList.txt` file in the same folder.
- Add the custom messages topics to the logger in the `logger.cpp` file in `Firmware/src/modules` folder. Add the `actuator_controls_virtual_fw` topic to the logger as well to record also what controls would the default PX4 stabilisation use.
- Modify the `fw_att_control.cpp` and `mc_att_control.cpp` files in the `Firmware/src/modules` folder such that they publish their actuator outputs to the logged virtual topic `actuator_controls_virtual_fw` instead of the real `actuator_controls_0`. This is achieved by changing the `_actuators_id = ORB_ID(actuator_controls_0)` line accordingly in the `vehicle_status_poll` functions. This topic can then be subscribed to in the custom module and passed through to the real actuator controls based on some RC switch position as desired.
- Increase the update rate of the `actuator_controls_1` topic in the `px4io.cpp` file in `Firmware/src/drivers/px4io` folder in the `task_main` function the same way as they do with the `actuator_controls_0` topic. Do this if you use the `actuator_controls_1` topic for your primary controls, such as the tailerons or differential thrust in the Custer UAV.
- Compile the code by calling `make px4_fmu-v5_default` from the Firmware directory as per instructions in [6]. Use `make clean` before each build if you modified the files in the `Firmware/ROMFS` or `Firmware/boards` folders otherwise the changes will not be registered.
- Load the build file `px4_fmu-v5_default.px4` to the Pixhawk connected by USB through the QGC Firmware utility, or by using `make px4_fmu-v5_default upload` command as per instructions in [6]. For any new airframes to be registered by the QGC, you have to use the QGC Firmware utility the first time this airframe is added, and restart the QGC.
- Select the custom airframe in the QGC Airframes tab, perform the calibrations and set up the parameters as needed.

10 List of files

my_LQR_control - CMakeLists.txt - my_LQR_control_params.c - my_LQR_control.hpp - my_LQR_control.cpp	Folder containing the custom stabilisation module files. The variables are allocated in the .hpp file, the main control logic is in the .cpp file, the custom parameters are in the _params.c file and the .txt file contains instructions for the compiler.
my_LQR_printouts - CMakeList.txt - my_LQR_printouts_params.c - my_LQR_printouts.cpp - my_LQR_printouts.hpp	Supporting module to take care of printouts to the QGC terminal screen.
my_LQR_Custer.main.mix my_LQR_S500_quad.main.mix	Custom mixer files.
2999_my_LQR_Custer 4999_my_LQR_S500_quad	Custom airframe files.
angular_rates_filtered.msg my_LQR_setpoint.msg	Custom message files (uORB topics definitions). We note that we used the setpoint topic to pass also other information than setpoints, as this was more convenient for us than creating a new topic each time a new information had to be logged.
logger.cpp	Modified logger function.
fw_at_control.cpp mc_att_control.cpp	Modified attitude control functions.
px4io.cpp	Modified px4io driver.
px4_fmu-v5_default.px4	The PX4 build file with the custom stabilisation module.

Table 10.1: The files used for the custom Custer UAV stabilisation, following the instructions in Section 9.1.

QGCXPlaneLink.cc	Modified QGC widget for HITL simulation in X-Plane from the QGC source code [13].
QGroundControl	QGC installer file for Mac with the modified HITL widget.
my_LQR_Custer_HIL.main.mix	Custom mixer file for HITL.
1999_my_LQR_Custer_HIL	Custom airframe file for HITL.
CusterXPlane_model - MY_PT60RC_icon.png - MY_PT60RC_paint.png - MY_PT60RC_paint2.png - MY_PT60RC_prefs.txt - MY_PT60RC_prop.png - MY_PT60RC.acf	Custom aircraft definition for X-Plane to be added to the Aircraft/MyPlanes folder of the X-Plane installation as per instructions in [1].

Table 10.2: The customized files used to run the HITL simulation of Custer in X-Plane.

Taranis_backup_2.otx	Taranis OpenTX model setup
custer01.mdl	Futaba model setup

Table 10.3: Files with transmitter model setup.

`custer_fly.params`	Parameters file storing all the parameters used for one of the test flights.
`log_7_2019-9-3-11-11-20.ulog`	Logger file from one of the flight tests.
`doc.px4.io`	User Guide documentation backup for v1.9 PX4 firmware.
`dev.px4.io`	Developer Guide documentation backup for v1.9 PX4 firmware.

Table 10.4: Other files additionally provided for reference.

11 HITL simulation in X-Plane

Additionally, in order to be able to test the custom module in the Hardware In The Loop (HITL) mode with X-Plane simulator, one has follow the instructions in [2].

To be able to use a custom airframe different than the Standard Plane, one has to modify the X-Plane widget in the `qgroundcontrol/src/comm` folder of the QGC source code such that the control inputs are sent to X-Plane as desired. An example of how to achieve this for the Custer airframe using dedicated mixer and airframe files can be seen by observing the files in Table 10.2. We note that the modified QGC has to be compiled with the correct version of the Qt Creator as per instructions in [4] on the Mac/Windows/Linux platform to obtain the custom Mac/Windows/Linux QGC installer respectively. We also note that while the differential thrust effect was achieved by modifying the QGC installation, the split elevator effect is achieved from within the X-Plane aircraft model.

Besides the instructions in [2], we recommend to disable the angular rates filtering in the custom module as the update frequency of the HITL simulation will be already very low. The QGC XPlane HITL widget setting that was found to work is shown in Figure 11.1. We note that the receiving frequency is expected to be somewhere in the range of 10 to 30 Hz and slower rates can cause unstable behaviour of the controller. We also note that the vehicle has to be in the armed state in order to successfully communicate with the X-Plane.

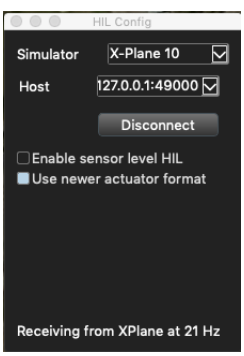


Figure 11.1: QGC X-Plane HITL widget setting.

The used X-Plane settings are depicted in Figure 11.2. We are using the X-Plane version 10 in a demo version which allows 10 minutes of flying after which it needs to be restarted.

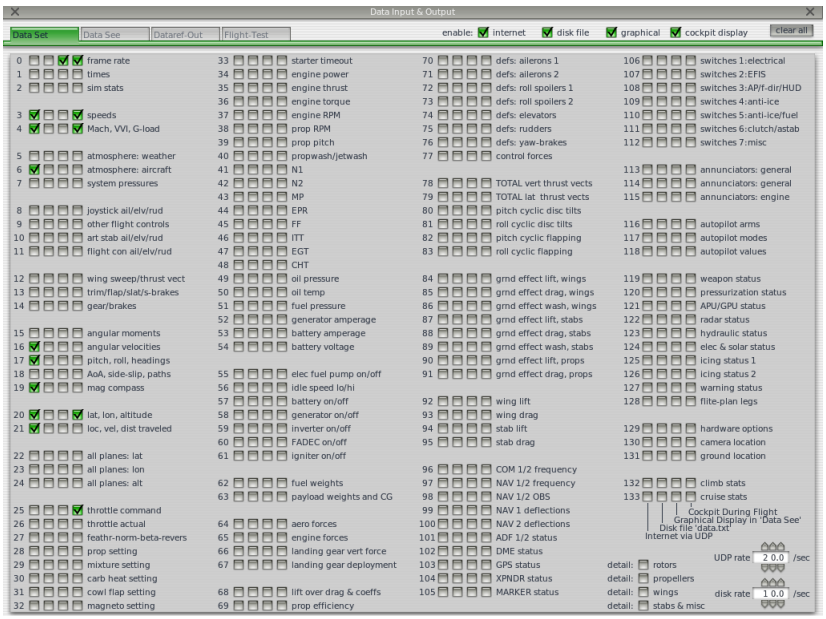


Figure 11.2: X-Plane settings for HITL simulation.

The control commands are sent from the RC transmitter to the receiver the same way as in real flight so no changes are necessary. The Pixhawk can be powered through the USB and only a receiver bounded to the transmitter needs to be connected, as shown in Figure 11.3. In QGroundControl, the `1999_my_LQR_Custer_HIL` airframe has to be selected.

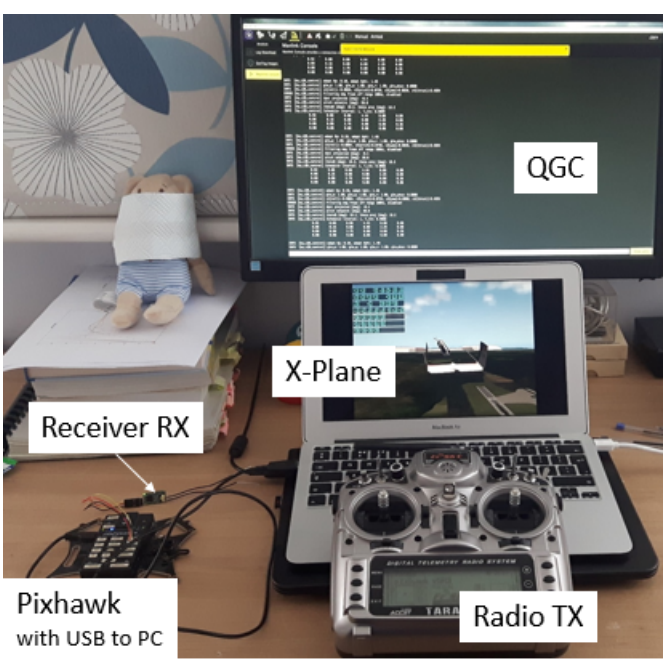


Figure 11.3: HITL wiring.

References

[1] Plane Maker Manual. https://developer.x-plane.com/manuals/planemaker/, retrieved in October 2019 .

[2] Hardware in the Loop Simulation (HITL). https://dev.px4.io/v1.9.0/en/simulation/hitl.html, retrieved in October 2019.

[3] Pixhawk 4 Wiring. https://docs.px4.io/v1.9.0/en/assembly/quick_start_pixhawk4.html, retrieved in October 2019.

[4] QGroundControl Developers Guide. https://dev.qgroundcontrol.com/en/getting_started/, retrieved in October 2019.

[5] Adding a new Airframe. https://dev.px4.io/v1.9.0/en/airframes/adding_a_new_frame.html, retrieved in October 2019.

[6] Building the PX4 Software. https://dev.px4.io/v1.9.0/en/setup/building_px4.html, retrieved in October 2019.

[7] Log Analysis using Flight Review. https://docs.px4.io/v1.9.0/en/log/flight_review.html, retrieved in October 2019.

[8] MAVLink Messaging. https://dev.px4.io/v1.9.0/en/middleware/mavlink.html, retrieved in October 2019.

[9] Mixing and Actuators. https://dev.px4.io/v1.9.0/en/concept/mixing.html, retrieved in October 2019.

[10] First Application Tutorial. https://dev.px4.io/v1.9.0/en/apps/hello_sky.html, retrieved in October 2019.

[11] Parameter Reference. https://dev.px4.io/v1.9.0/en/advanced/parameter_reference.html, retrieved in October 2019.

[12] Flight Log Analysis. https://docs.px4.io/v1.9.0/en/log/flight_log_analysis.html, retrieved in October 2019.

[13] QGroundControl Ground Control Station. https://github.com/mavlink/qgroundcontrol, retrieved in October 2019.

[14] QGroundControl User Guide. https://docs.qgroundcontrol.com/en/, retrieved in October 2019.