Instruction Manual For

PlainFlightController V2.1.0

Revision History

Version	Comments	Date
V05-00	V2.1 updates, neopixel LED for Wavershare Zero/Tiny ESP32's, added new IMU orientation.	14/12/2025
V04-00	V2.0 updates for C++ and many new features.	1/3/2025
V03-00	Updates for web server interface and advanced rudder mixer.	23/6/2024
V02-00	Updates for new levelled mode, battery monitor, ESC calibration and LED outputs.	12/2/2024
V01-02	Initial release.	24/1/2024
V00-05	Development	15/12/2023

Contents

1. Introduction

1.1 Background

PlainFlightController stabilisation software is for the RC pilot who wants to get the most from their model, needs to master an unstable aircraft, or simply counteract environmental conditions for an enjoyable flight.

Originally created as a home project it quickly became more with its performance, ease of build and low budget parts. These qualities led to it being refined and posted on GitHub for other hobbyists to have a go and enjoy.

While PlainFlightController has been developed for small electric powered aircraft it could easily be modified and used on other small radio-controlled craft. This can be done with little effort as the code is broken down into logical classes and is well commented for those that want to understand or modify for their own purposes.

The flight controller hardware is based on the Seeed Studio XIAO ESP32 boards and the ever-popular MPU6050 IMU. It's simple to build, easy to program via Arduino IDE and cheap when compared to many commercially available flight controllers.

1.2 PlainFlightController V2.x Release

V2.0 brings many improvements and upgrades to the V1 code base. The main points to note are as follows:

1.2.1 C++ Rewrite

V2.0 has undergone a complete rewrite into C++. I felt this was required as a good step to tidy and structure the original V1 code base as when adding in the WiFi Configurator it became untidy and complicated.
By using C++ we can also use a model base class from which all model types are built on and leads to new model types being easily added without the need to change the core code. We also get the structural benefits and encapsulation that C++ brings, plus easy instantiation of the model type and features you require through a simple Config.hpp file.

1.2.2 WiFi Configurator

Rather than having to reprogram your flight controller each time you want to adjust the gains/rates, you can now open a web-based configurator on your smart device to set gains, rates, level trims, servo trims, angles, and battery scale. Perfect for when out flying and you need to make a quick change.

1.2.3 Multicopters

Following requests for my Chinook and BiCopter code, V2.0 now includes 7 different mutlicopter models. These being Quad X, Quad +, BiCopter, Chinook, TriCopter, SingleCopter and DualCopter. The only limitation with the default code is a maximum of 4 motors and 4 servos, but this could be modified by yourself for other combinations of up to a total of 8 outputs.

1.2.4 Self-Levelled Prop Hang Mode

For models that are capable of prop-hanging; a new mode activated by a Tx switch on channel 9 can be used to make rudder and elevator maintain nose up auto levelled prop hang. Additionally, a tail sitter option can be set to swap roll and yaw to make a basic tail sitter VTOL.

1.2.5 Acro Trainer Mode

This flight mode allows full aerobatics in rate mode, but when pitch and roll sticks are centred the model will return to and maintain level flight. This is similar to the horizon mode of several other flight controllers.

1.2.6 Neopixel LED

To support the Waveshare ESP32 Zero and Tiny boards which have higher usable pin count and are cheaper to buy. The option of using a neopixel LED is available via Config.hpp.

1.2.7 IMU Orientation

A new IMU orientation has been added that allows the IMU to be rolled to the right by 90 degrees. This allows the flight controller to be mounted in narrow fuselages. More orientations will come in further releases.

1.3 Specifications

PlainFlightController has the following specifications as detailed in Table 1 as standard:

Table 1 - Specifications

Feature	Detail
Model Mixes	Plane Full House (aileron/flaps/elevator/rudder) Plane Full House V-Tail (aileron/flaps/V-tail mix) Plane V-Tail (V-tail mix for rudder/elevator) Plane Rudder/Elevator (rudder/elevator) Plane Advanced Rudder/Elevator (rudder/elevator) Flying Wing (elevons/rudder) Quad X Quad + Bicopter Chinook Singlecopter Dualcopter Tricopter
Flight Modes	Pass Through (Fixed wing only) Gyro Rate Self-Levelled Acro Trainer Heading Hold (Fixed wing only) Prop Hang Self-Levelled (Fixed wing only, Plane or tail sitter modes)
Actuators	4 servos, 4 motors (Or other combination of the 8 with modification)
Actuator Refresh Rates	50Hz, 100Hz, 150Hz, 200Hz, 250Hz, 300Hz, 350Hz Oneshot125 (limited to 2KHz) (Or custom with modification)
Motors	Fixed Wing: Standard throttle response, single or twin motors. Or, throttle/PIDF yaw based differential thrust can be configured for twin motors for any fixed wing model mixer.
Radio Protocols	Sbus
Battery Monitor	1-3s Lipo as standard, actively reduces power to motor when voltage drops below a defined threshold.
Failsafe	Fixed Wing: Automatic transition to self-levelled mode and cuts throttle. Multicopter: Cuts motors.
LED	Flight mode indication. Stanard LED or neopixel.
Target	Seeed Studio XIAO ESP32S3. Or, any other ESP32S3 based Arduino.
IMU	MPU6050 (GY-521 breakout board)
Control Loop	Stable 1KHz
Additional Features	Multicopters have an interpretation of BetaFlight's Air Mode and Anti-Gravity to maintain full PID control at minimum & maximum throttle settings.

1.4 Scope

PlainFlightController software is only intended to be used on small electrically powered model aircraft, with possible adaptions for other small radio-controlled craft i.e. Hovercrafts, hydrofoils, cars etc.

No warranty or guarantee is given with this software or accompanying documentation, refer to license agreement for full details on using this software.

2. How To Assemble

It is recommended that the flight controller is built onto PCB prototype board for rigidity and robustness, but for the lightest weight application it can be built with 'flying wires' connecting between the components.
Figure 1 and Figure 2 detail the wiring diagrams of both 5V (BEC powered) and 3.3V (1s Lipo powered) systems. They both show connections for a 'full house' plane with 4 servo outputs and 2 motors. This wiring configuration can be used for any of the 'Model Mixers' in the software as all servo/motor outputs are given.

Note: Servo/motor power supplies have been omitted for clarity.
Note: The ESP32S3 I/O's are not 5V tolerant and needs the series resistors to drop voltage / current limit.
Note: 470uF 16V capacitor protects the ESP32S3 from voltage spikes.

Figure 1 – 5V Wiring Diagram

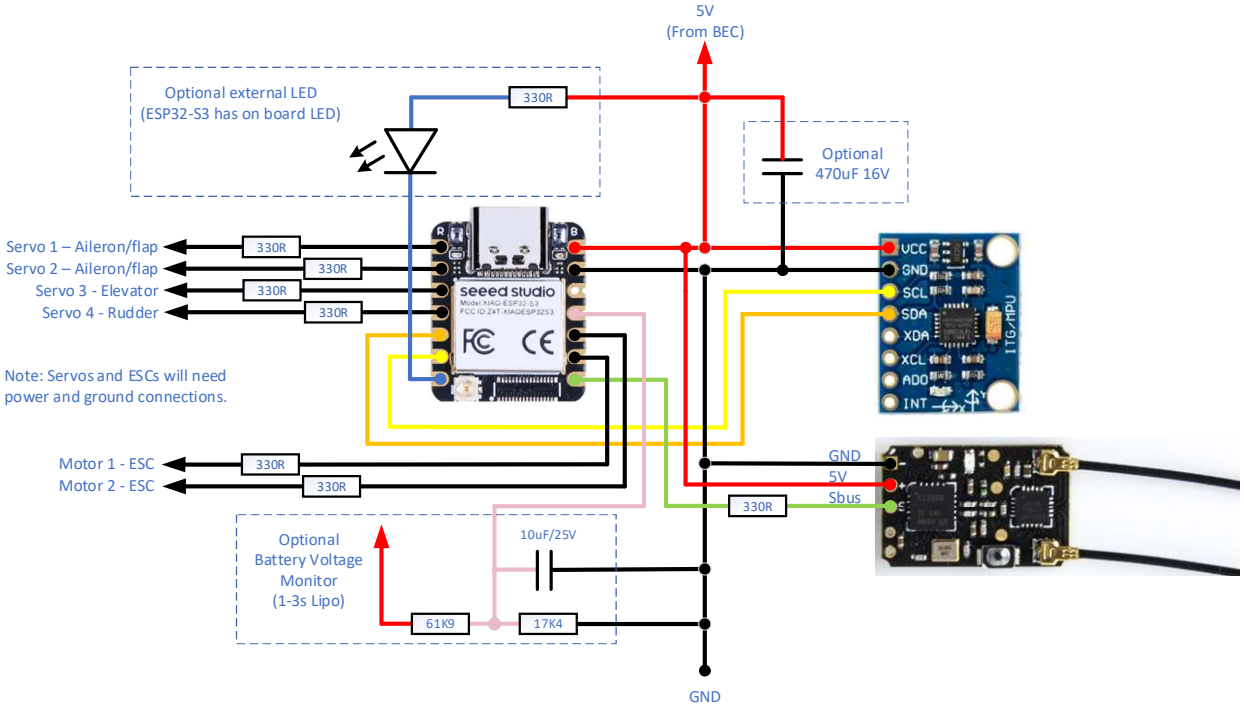
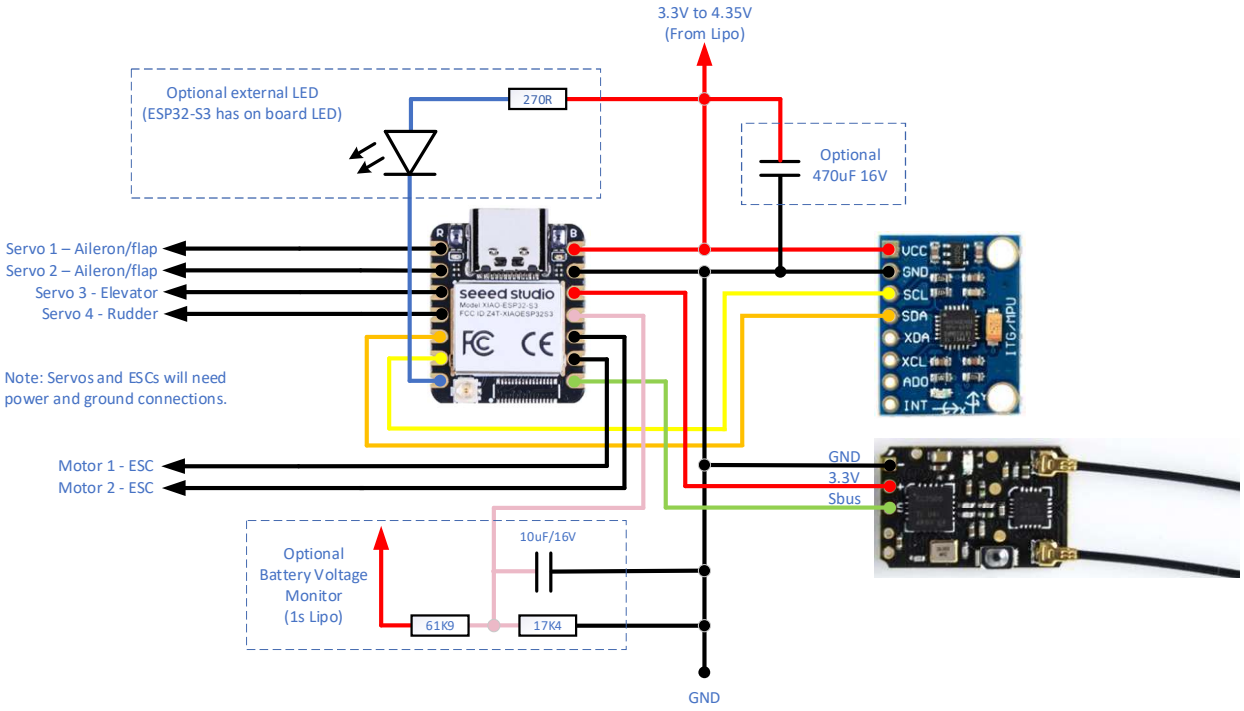


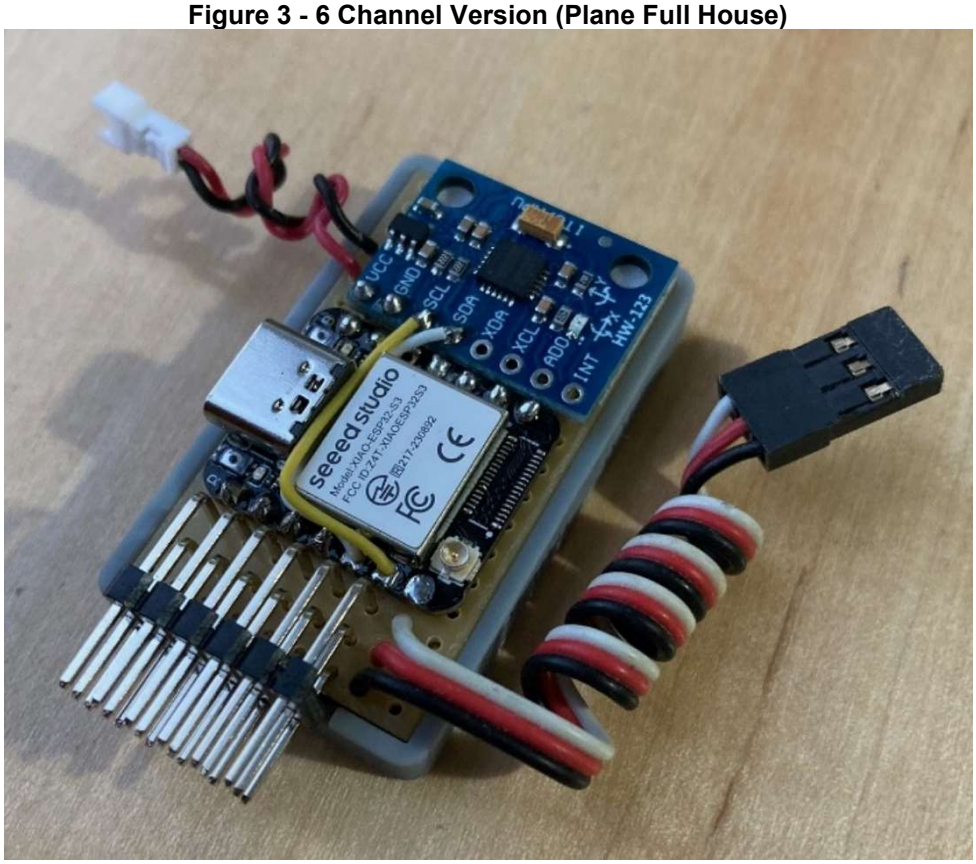
Figure 2 - 3.3V Wiring Diagram



Note: Strongly recommend 470uF 16V capacitor on 3.3V systems supply rail if using BLHeli ESC's to absorb voltage spikes, and generally recommended to prevent the possibility of brown out on 3.3V and 5V systems.

Figure 3 details a typical build on prototype board. This is the 5V version and all resistors are 0805 package soldered onto the back of the prototype board. Note this version does not use the external LED, but does use battery monitoring input.

Figure 3 - 6 Channel Version (Plane Full House)



2.1 Components

To build the flight controller you will need the items listed in Table 2. These can be obtained from a good hobbyist electronics store, eBay, or Amazon.

Table 2 – 5V Flight Controller Components

Component	Quantity	Comment
Seeed Studio XIAO ESP32S3	1	XIAO ESP32C3 is not directly supported but can be used if you modify the software. Its clock speed is lower and may affect control loop times and you will also be limited to 6 actuator outputs.
GY-521 (MPU6050 breakout)	1	
330 Ohm resistor	8	For servo/motor and receiver; These are only required when interfacing 5V components. A 1s lipo powered model with 3.3V servos/ESC will not require them. The external LED option always requires a series resistor.
10uF 16V/25V capacitor	1	Used to filter/smooth battery voltage and helps remove any electrical voltage spikes in its sampling. If not fitted increase weighted filter in the software.
LED	1	Optional LED for external use. Can use LED_BUILTIN of Arduino.
Sbus Receiver	1	Standard 100Kb Sbus receiver of your choice. Inverted Sbus used, but non-inverted can be used with simple code modification.
30AWG Silicone Wire	-	Multiple colours for wiring of components.
61K9 0.25W	1	If not using exact values software can be modified. Ensure potential divider output does not exceed 3.3V.

Component	Quantity	Comment
17K4 0.25W	1	If not using exact values software can be modified. Ensure potential divider output does not exceed 3.3V.
PCB Prototype Pad Board (2.54mm/0.1 inch pitch)	~42x22mm	Use what you are comfortable with or have. Or build the controller with 'flying wires'.
1mm double sided foam tape	20mm	Used to soft mount gyro.

3. Arduino IDE Installation

For this installation process it is assumed you will be using the Seeed Studio XIAO ESP32-S3. The installation process should be the same for other ESP32 Arduino boards. This document examples are based on Arduino version 2.3. Other versions may vary in layout/appearance.

3.1 Arduino IDE Installation

For your operating system download and install the newest version of Arduino IDE.

https://www.arduino.cc/en/software

Once installed launch the Arduino application.

3.2 Arduino Library Installation

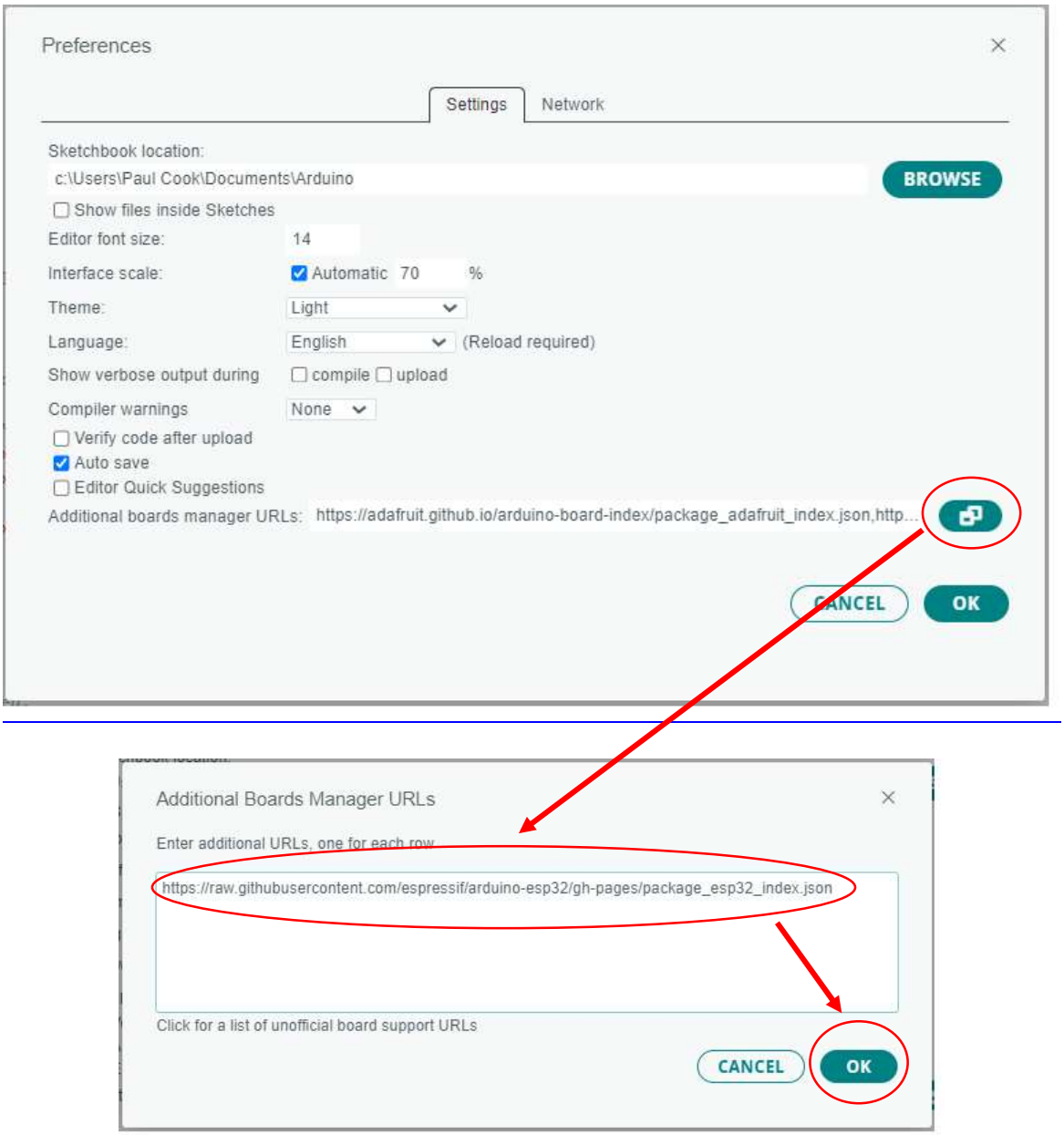
To compile the ESP32 XIAO boards within Arduino you will need to install the 'Espressif Systems ESP32' board package and the 'MPU6050 by Electronic Cats' into your Arduino IDE.

3.2.1 Espressif Systems ESP32 Installation

First you need to map the 3rd party library location . To do this copy the URL given below then navigate to **File > Preferences**, and in the pop-up window fill **'Additional Boards Manager URLs'** with the following URL as shown by Figure 4:

https://espressif.github.io/arduino-esp32/package_esp32_index.json

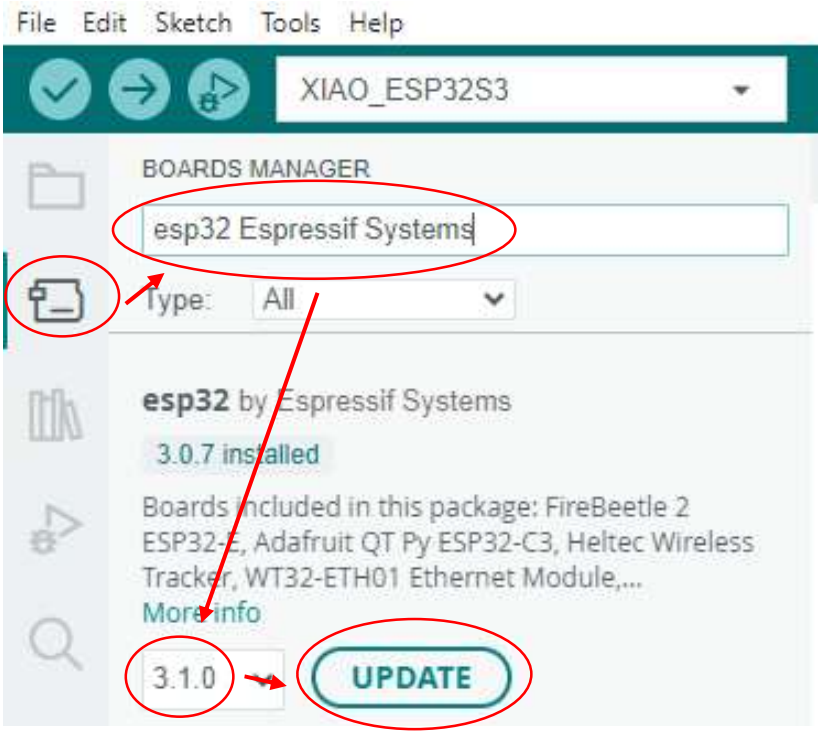
Figure 4 - Espressif Systems URL Mapping



Note: Click **'OK'** to accept and close all pop-up windows.

Now to actually install the 'Espressif Systems ESP32' board package into the Arduino IDE; Navigate to **Boards Manager** icon, type the keywords '**esp32 Espressif Systems**' in the search box, select version 3.1.0 and click the Update/Install button as indicated by Figure 5.

Figure 5 - ESP32 Espressif Systems Board Package 3.1.0

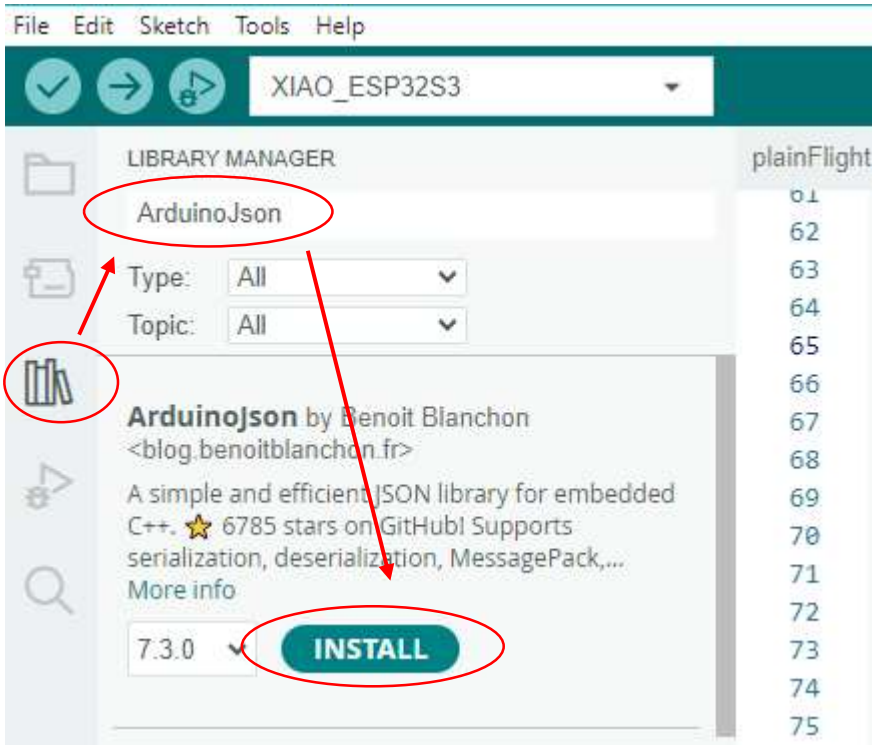


Should you have any problems installing this please refer to:

https://docs.espressif.com/projects/arduino-esp32/en/latest/installing.html

3.2.2 ArduinoJson

Install the ArduinoJson library as this is needed for the file system. Library version 7.3.0 has been used and tested with PlainFLightController V2.0.0.



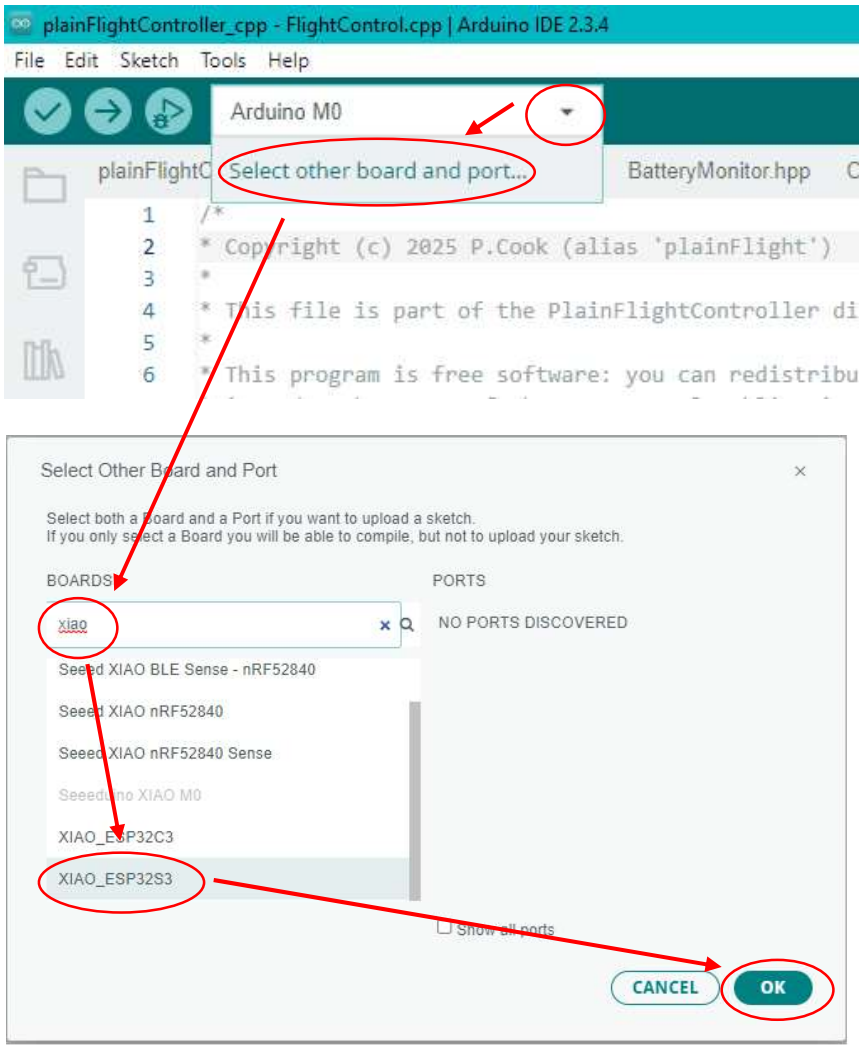
3.2.3 MPU6050 By Electronic Cats Installation - Redacted

PlainFlightController V2.0.0 onwards no longer requires this library.

3.3 XIAO Board Selection

To compile and program the XIAO ESP32S3 you need to select the correct board. Figure 6 details how to navigate to drop down board menu and **'select other board and port...'**, from the pop up window find and select **'XIAO_ESP32S3'** from the list of boards.

Figure 6 - XIAO ESP32S3 Board Selection

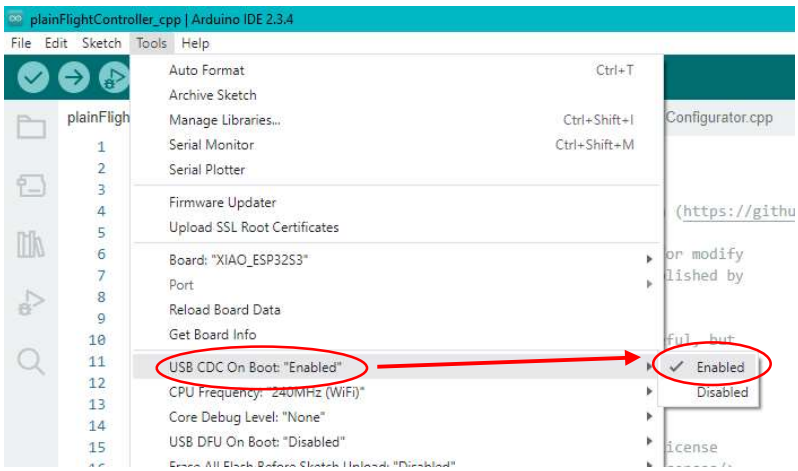


3.4 IDE ESP32 Tools Configuration

3.4.1 USB CDC On Boot

For serial communications to work you must ensure that USB CDC OnBoot is enabled. This is found via Tools->USB CDC On Boot->Enabled.

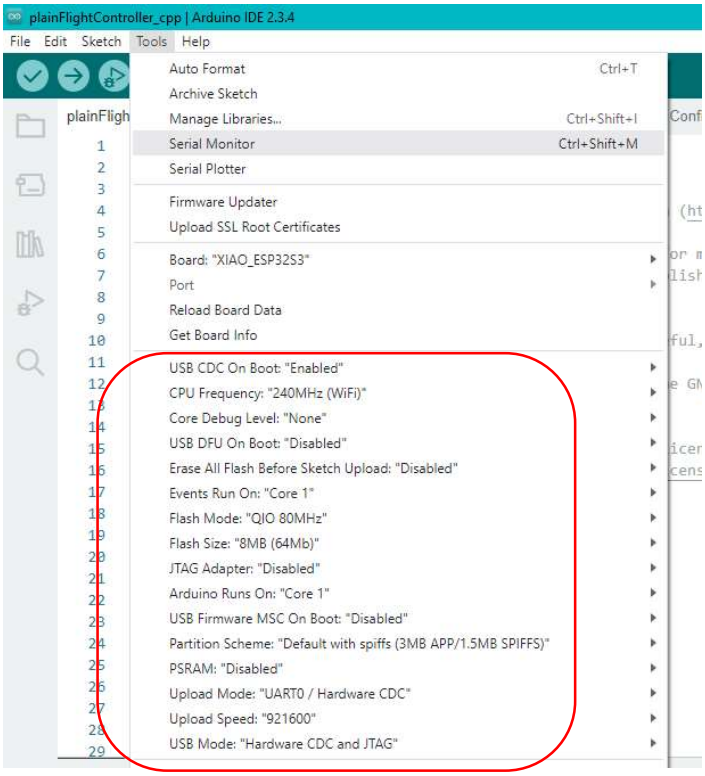
Figure 7 - Enable USB CDC On Boot



3.4.2 Other Tool Configurations

The other setting should default but ensure all other settings are as follows, particularly the Flash Mode of QIO:

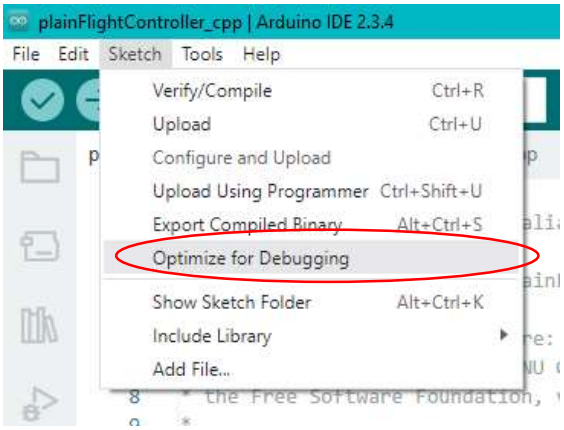
Figure 8 - Default Settings



3.4.3 Optimise For Debugging – Disabled

You must ensure that 'Optimise For Debugging' is **disabled**. If it is enabled, you will not consistently achieve a stable 1Khz loop rate.

Figure 9 - Disable Optimise For Debugging



3.5 Port Selection

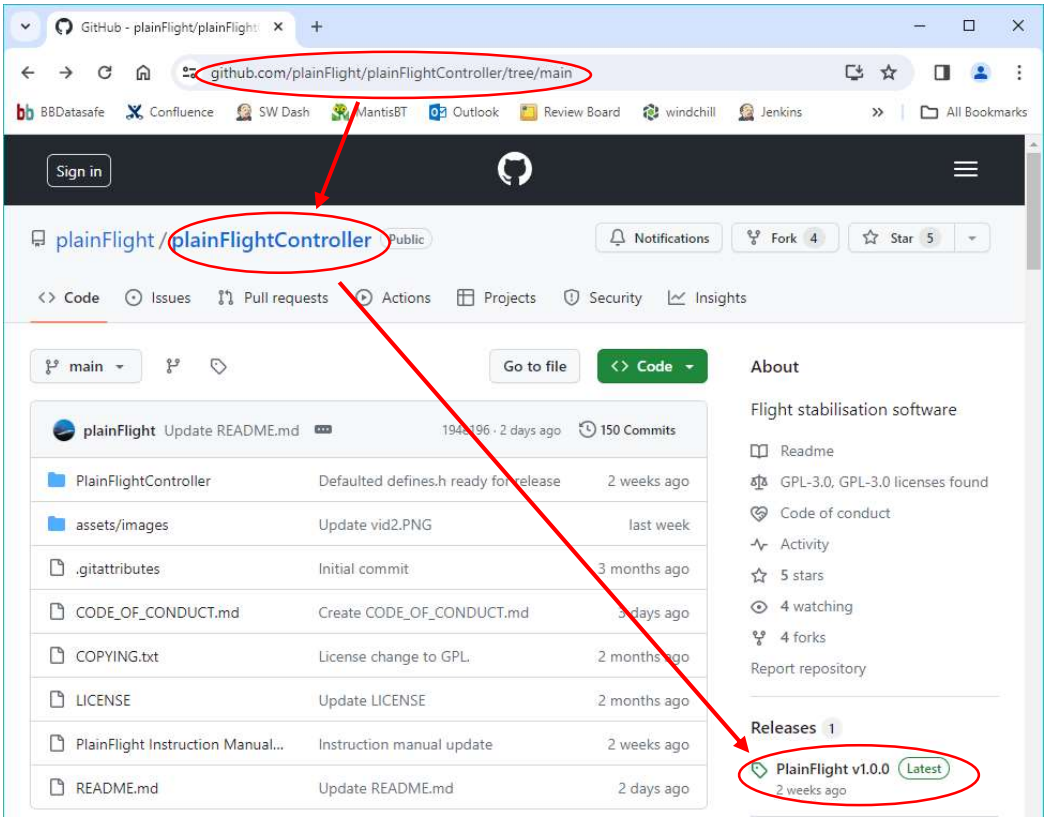
Connect the XIAO_ESP32S3 to your computer via USB. Wait for the drivers to be automatically installed then navigate to **Tools > Port** and select the serial port name of the connected XIAO_ESP32S3.

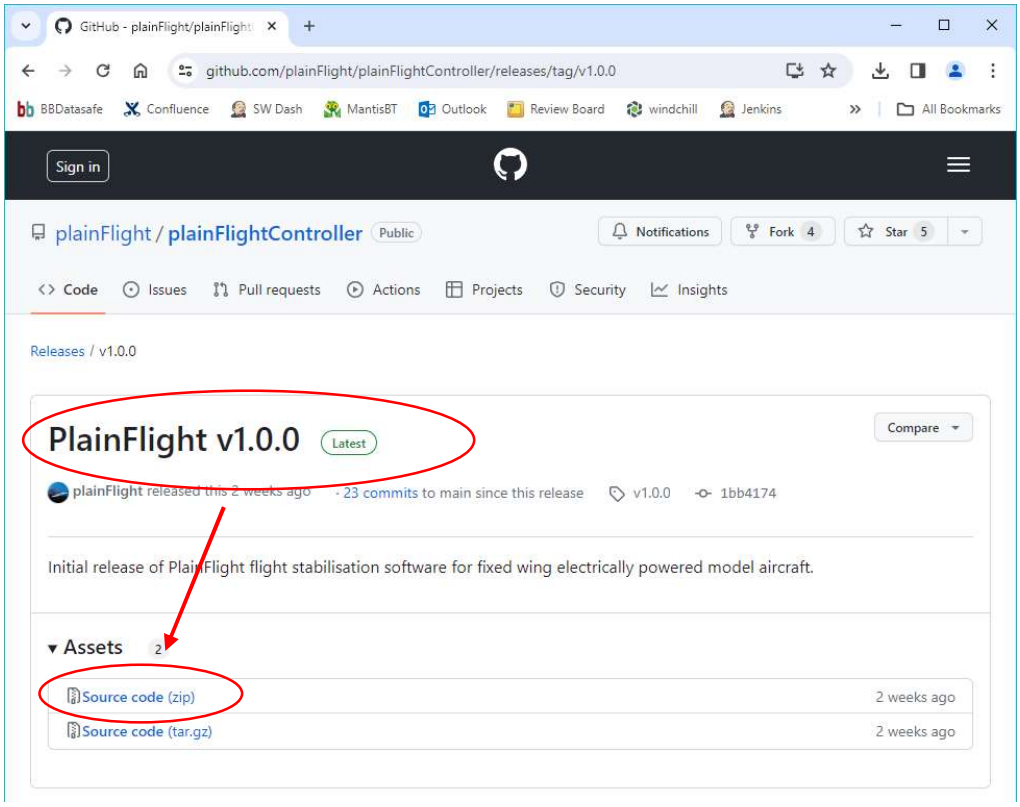
Note: The XIAO boards often comes up with the wrong name, so select the named board that appears on USB connection.

4. Downloading PlainFlightController Software

From github.com search 'plainFlightController' to find the project home page. To download the flight controller software, click on release v2.0.0 (v1.0.0 shown in example below) then select 'Download ZIP'. This will download the Arduino code, instructions and license to your computer. Figure 10 details the sequence for downloading the software.

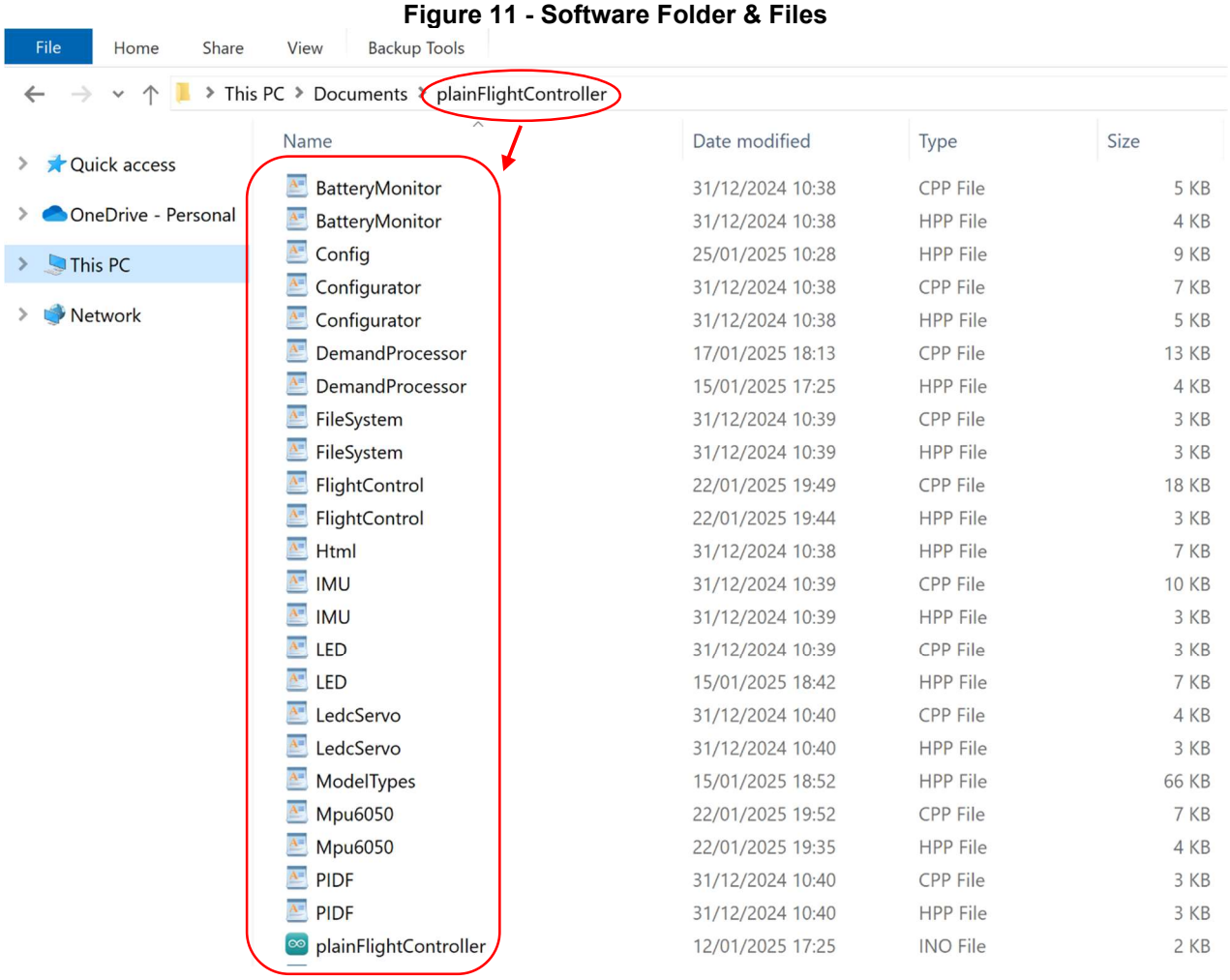
Figure 10 - PlainFlightController Software Download





Once downloaded unzip the files at a suitable location. Note the software files should appear inside a folder called ‘PlainFlightController’. If they do not create it and place the software files inside it, as detailed by Figure 11, or it will not compile.

Figure 11 - Software Folder & Files



5. How To Program

CAUTION: Always remove the flight battery when programming. The motor may start unexpectedly if you do not.

CAUTION: If you have built the 3.3V system never plug USB and 1s lipo flight battery in at the same time as the 5V USB supply will back feed the 4.2V lipo battery.

Open the Arduino project by double clicking ’PlainFlightController.ino’ file. The Arduino IDE should open and all of the software files will appear as tabs inside the IDE.

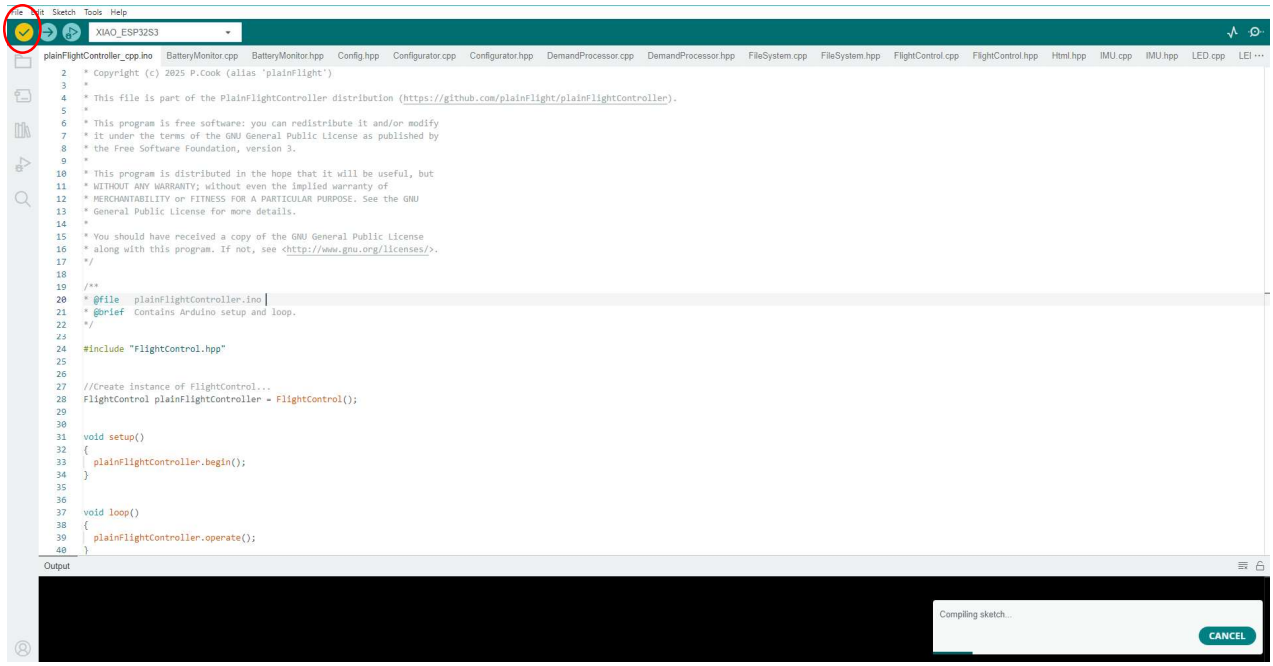
Ensure the correct target board is selected (XIAO_ESP32S3) as detailed in section 3.3.

Plug in your XIAO ESP32S3 to your computer via the USB and select the correct port as detailed in section 3.5.

You can now select the upload button to program your XIAO ESP32S3 with PlainFlightController software. Figure 12 shows how the Arduino IDE should appear with all software files, correct board selected, and upload button highlighted.

The Upload can take several minutes, especially if it’s the first time the software has been compiled, a progress bar will be given to indicate the upload status.

Figure 12 - PlainFlightController Upload



Should your XIAO board fail to program, disconnect power/USB and hold down the boot button on the XIAO board. Now reapply power to enter boot mode ready for programming.

6. How To Configure Your Model (Config.hpp)

The flight controller is configured via the constant expressions contained within Config.hpp file. The file is well commented, but the following sections describe these in detail.

6.1 Model Mixer

There is currently a choice of 13 model types to choose from, 6 of which are fixed wing configurations and 7 are multicopter configurations. For the fixed wing configurations differential thrust can be set (if using 2 motors) and self-levelled prop-hang modes can be set.

The model type is set via the constant expressions within Config.hpp. Set the model type you require by setting the constexpr to true.

Figure 13 - Model Type Setting

```
//Model type to instantiate, set one to true...
static constexpr bool PLANE_FULL_HOUSE                = true;
static constexpr bool PLANE_FULL_HOUSE_V_TAIL         = false;
static constexpr bool PLANE_ADVANCED_RUDDER_ELEVATOR  = false;
static constexpr bool PLANE_RUDDER_ELEVATOR           = false;
static constexpr bool PLANE_V_TAIL                    = false;
static constexpr bool PLANE_FLYING_WING               = false;
static constexpr bool QUAD_X_COPTER                   = false;
static constexpr bool QUAD_P_COPTER                   = false;
static constexpr bool BI_COPTER                       = false;
static constexpr bool CHINOOK_COPTER                  = false;
static constexpr bool TRI_COPTER                      = false;
static constexpr bool DUAL_COPTER                     = false;
static constexpr bool SINGLE_COPTER                   = false;
```

Note: that multicopter mixers only have rate and self-levelled/acro-trainer modes (Pass through is not possible). They also have my own interpretation/implementation of 'Air Mode' and 'Anti Gravity' which are always active and do not require any tuning parameters.

6.1.1 PLANE_FULL_HOUSE

This mixer offers 2 servos for aileron/flap functions, rudder and elevator, single or dual motor. Flaps have 3 positions and are controlled via Aux1.

Figure 14 - PLANE_FULL_HOUSE

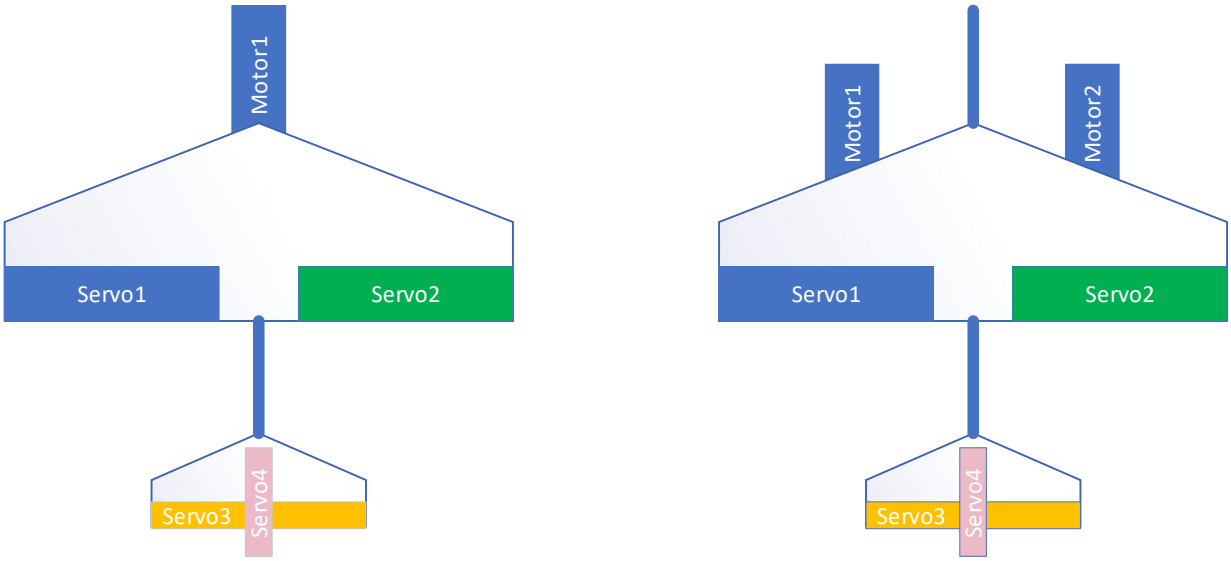


Table 3 - Plane Full House Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle	D8	
Motor 2	Throttle	D9	If fitted on model.
Servo 1	Left aileron/flap	D0	Aux1 operates 3 position flap
Servo 2	Right aileron/flap	D1	Aux1 operates 3 position flap
Servo 3	Elevator	D2	
Servo 4	Rudder	D3	

Note: To enable differential thrust uncomment the define USE_DIFFERENTIAL_THRUST.
Note: To enable Oneshot125 for BLHeli type ESCs set motor RefreshRate to IS_ONESHOT125.
Note: Pusher configuration has no impact on the flight controller mixer.

6.1.2 PLANE_FULL_HOUSE_V_TAIL

This mixer offers 2 servos for aileron/flap functions, and mixed rudder/elevator for a V-tail model. Single or dual motors can be used and flaps have 3 positions and are controlled via Aux1.

Figure 15 - PLANE_FULL_HOUSE_V_TAIL

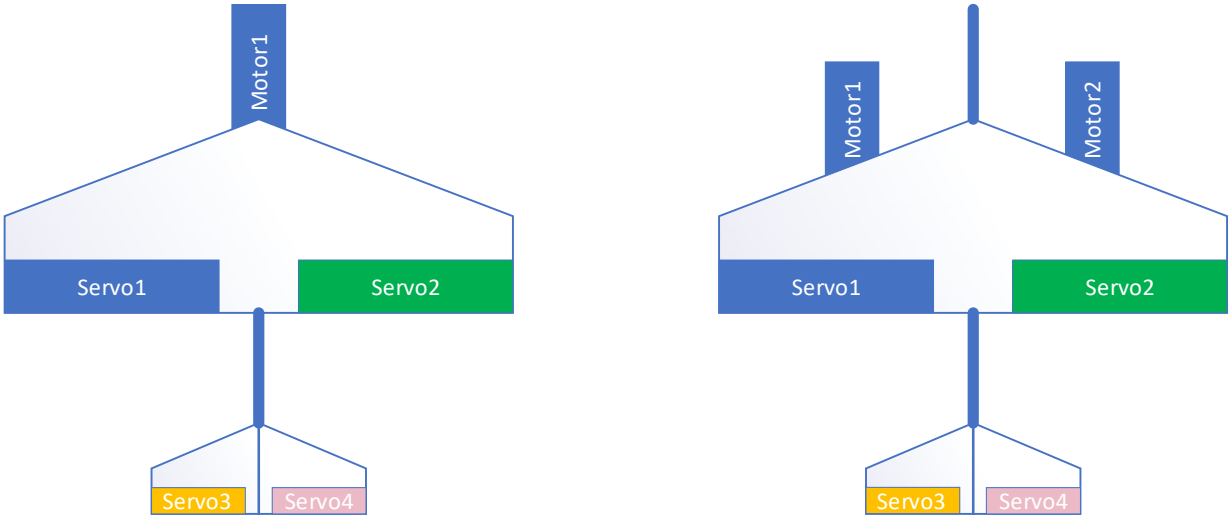


Table 4 - Plane V-Tail Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle	D8	
Motor 2	Throttle	D9	If fitted on model.
Servo 1	Left aileron/flap	D0	Aux1 operates 3 position flap
Servo 2	Right aileron/flap	D1	Aux1 operates 3 position flap
Servo 3	Elevator/rudder	D2	
Servo 4	Elevator/rudder	D3	

Note: To enable differential thrust uncomment the define USE_DIFFERENTIAL_THRUST.
Note: To enable Oneshot125 for BLHeli type ESCs set motor RefreshRate to IS_ONESHOT125.
Note: Pusher configuration has no impact on the flight controller mixer.

6.1.3 PLANE_RUDDER_ELEVATOR

This mixer offers rudder and elevator control with single or dual motors. Rudder is controlled by the roll axis Tx input and is used to stabilise roll axis when in rate or self-levelled modes.

Figure 16 - PLANE_RUDDER_ELEVATOR

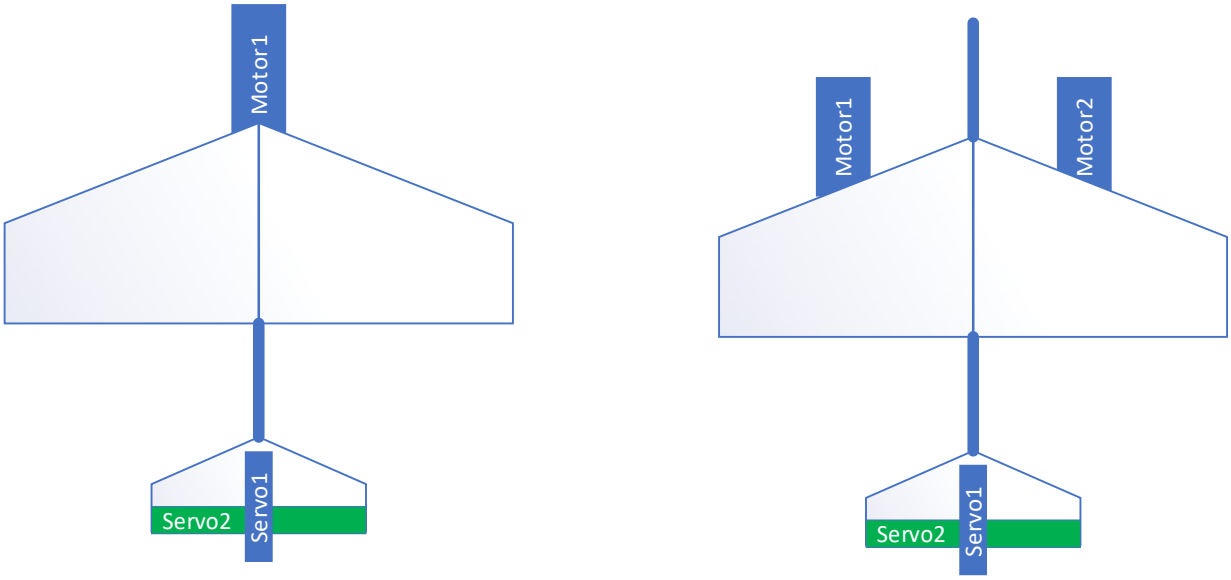


Table 5 - Plane Rudder Elevator Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle	D2	
Motor 2	Throttle	D3	If fitted on model.
Servo 1	Rudder	D0	
Servo 2	Elevator	D1	

Note: To enable differential thrust uncomment the define USE_DIFFERENTIAL_THRUST.
Note: To enable Oneshot125 for BLHeli type ESCs set motor RefreshRate to IS_ONESHOT125.
Note: Pusher configuration has no impact on the flight controller mixer.

6.1.4 PLANE_ADVANCED_RUDDER_ELEVATOR

This mixer offers rudder and elevator control with single or dual motors. Rudder is controlled by the roll axis Tx input and is used to stabilise roll axis when in rate or self-levelled modes. However, rudder is also controlled by the yaw axis Tx input and is used to stabilise yaw axis when in rate or self-levelled modes.

This allows stall turns to be cleanly executed via Tx yaw input and prevents I-gain build up on the roll axis. It also gives a more natural feel to using rudder stick for stall turns and seems to reduce ‘wing rock’ due to the yaw PIDF counter acting roll.

6.1.5 PLANE_V_TAIL

This mixer offers rudder and elevator control with single or dual motors. Use this for rudder/elevator control as flap function is disabled.

Figure 17 - PLANE_V_TAIL

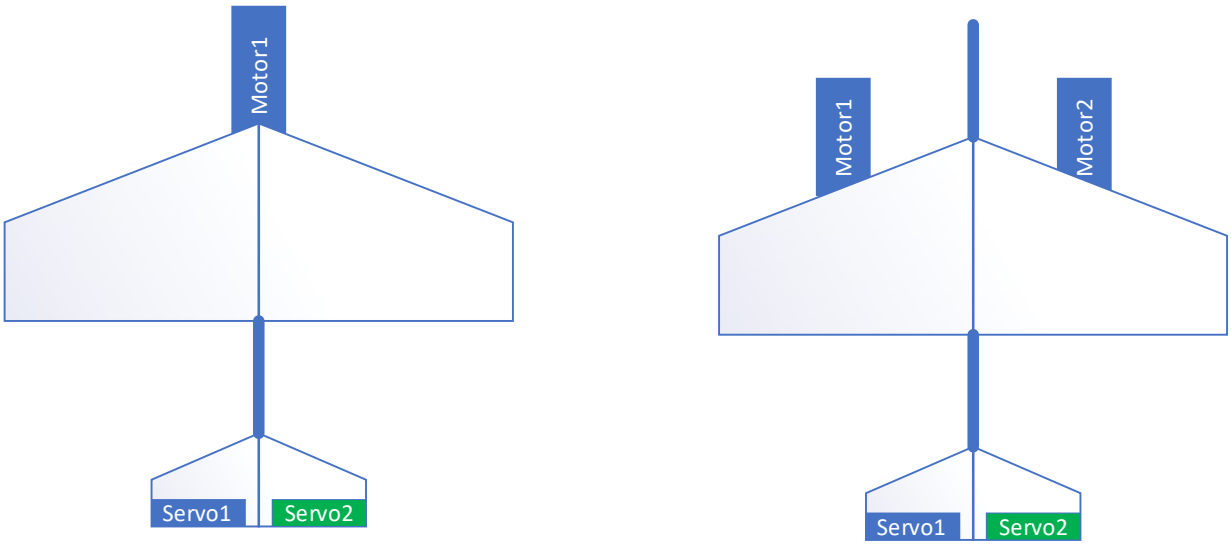


Table 6 - Plane V-Tail Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle	D2	
Motor 2	Throttle	D3	If fitted on model.
Servo 1	Left Rudder/elevator	D0	
Servo 2	Right Rudder/elevator	D1	

Note: To enable differential thrust uncomment the define USE_DIFFERENTIAL_THRUST.
Note: To enable Oneshot125 for BLHeli type ESCs set motor RefreshRate to IS_ONESHOT125.
Note: Pusher configuration has no impact on the flight controller mixer.

6.1.6 PLANE_FLYING_WING

This mixer offers elevon mixing of roll and elevator controls for flying wings. Rudder control is also available if fitted on the model. As standard single or dual motor control is given.

Figure 18 – MIXER_FLYING_WING

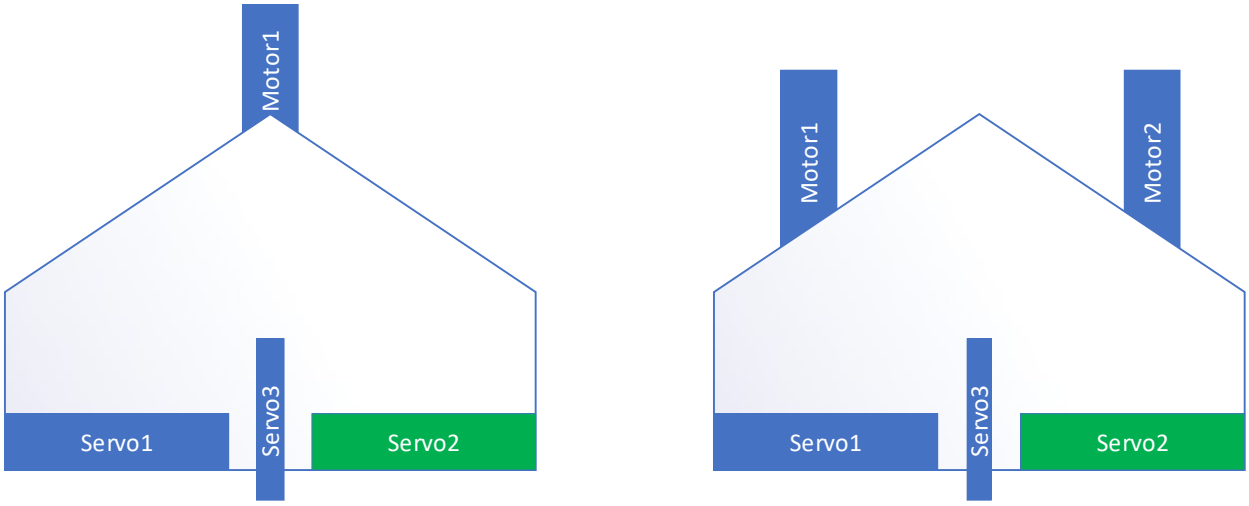


Table 7 - Plane Rudder Elevator Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle	D8	
Motor 2	Throttle	D9	If fitted on model.
Servo 1	Left elevon	D0	
Servo 2	Right elevon	D1	
Servo 3	Rudder	D2	

Note: To enable differential thrust uncomment the define USE_DIFFERENTIAL_THRUST.
Note: To enable Oneshot125 for BLHeli type ESCs set motor RefreshRate to IS_ONESHOT125.
Note: Pusher configuration has no impact on the flight controller mixer.

6.1.7 Multicopters

Do not expect PlainFlightController to fly multicopters with the same level of precision of other open-source flight controllers. However, with correct tuning they can be agile and perform good aerobatics.

Multicopters have selectable flight modes of rate, acro-trainer and self-levelled/acro-trainer. There is also transparent modes similar to BetaFlights ‘Air Mode’ and ‘Anti Gravity’ which are always active. This gives full PID control at throttle extremes and allows you to punch out and free fall under full control with no deviation of the aircraft.

For best performance you are advised to always use motor refresh of IS_ONESHOT125 with Bl-Heli (or similar) ESC’s as this gives the fastest update rates to the ECS’s.

6.1.7.1 Idle Up & Min Motor Settings

When you arm a multicopter the motors will spin up to a speed defined by IDLE_UP_VALUE. You may need to tune this value (with propellers removed) to a speed that makes the motors spins with reasonable energy but not to the point where the model is on the verge of trying to lift off from the ground.

MIN_MOTOR_VALUE will also need to be tuned. This needs to be set to a value where the motors run as slowly as possible but without cutting/staling.

Both IDLE_UP_VALUE and MIN_MOTOR_VALUE can be found in Config.hpp and are crucial to set correctly as these dictate how well the PID controller works at minimum throttle position/idle up value (i.e. Air Mode equivalent).

```
//Multicopter minimum motor speed settings
static constexpr int32_t IDLE_UP_VALUE                  = 300;
static constexpr int32_t MIN_THROTTLE_VALUE             = 100;
```

6.1.7.2 Motor Calibration

For multicopters always calibrate your ESC's before tuning idle up, min motor values and/or the first flight. See calibrate ESC feature in Config.hpp.

6.1.7.3 Motor Audible Control Noise

There is no RC interpolation within PlainFlightController, as a result you will hear ~143Hz noise within the motors when moving the Tx sticks. Do not be alarmed this is just the ESC's adjusting the speed of the motors to suit the newest RC commands.

6.1.7.4 QUAD_X_COPTER

This mixer offers control of 4 motors arrange as a quad X configuration. In config.hpp set QUAD_X_COPTER to true to enable.

Figure 19 – QUAD_X_COPTER

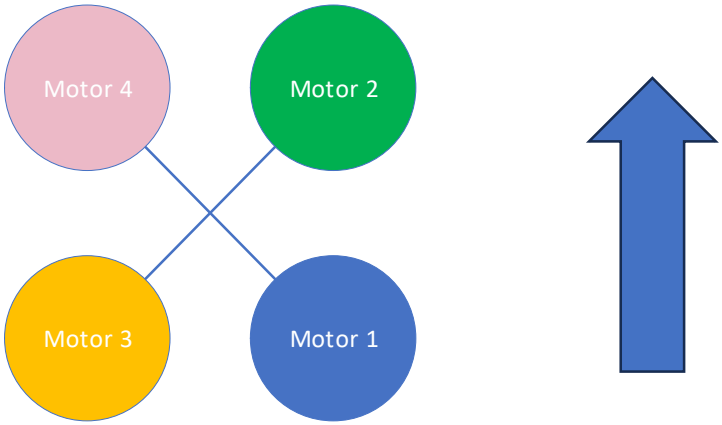


Table 8 – Quad X Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle, roll, pitch, yaw	D0	
Motor 2	Throttle, roll, pitch, yaw	D1	
Motor 3	Throttle, roll, pitch, yaw	D2	
Motor 4	Throttle, roll, pitch, yaw	D3	

6.1.7.5 QUAD_P_COPTER

This mixer offers control of 4 motors arrange as a quad + configuration. In config.hpp set QUAD_P_COPTER to true to enable.

Figure 20 – QUAD_P_COPTER

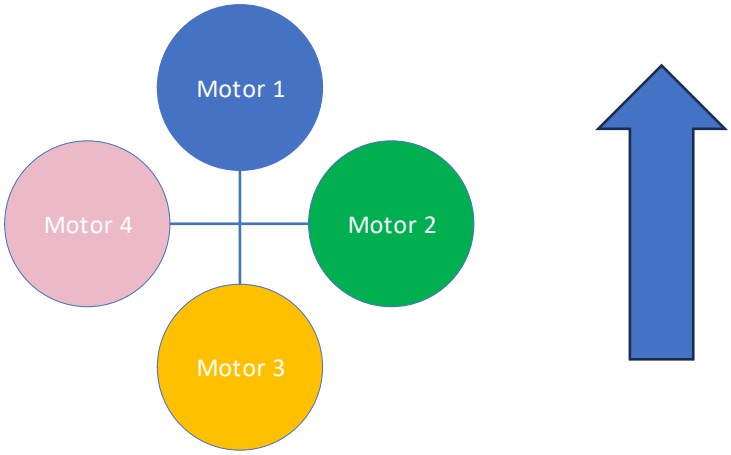


Table 9 – Quad P Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle, roll, pitch, yaw	D0	
Motor 2	Throttle, roll, pitch, yaw	D1	
Motor 3	Throttle, roll, pitch, yaw	D2	
Motor 4	Throttle, roll, pitch, yaw	D3	

6.1.7.6 BI_COPTER

This mixer offers control of 2 motor which are controlled by throttle and roll and 2 servos to mix pitch and yaw controls. In config.hpp set BI_COPTER to true to enable.

Figure 21 – BI_COPTER

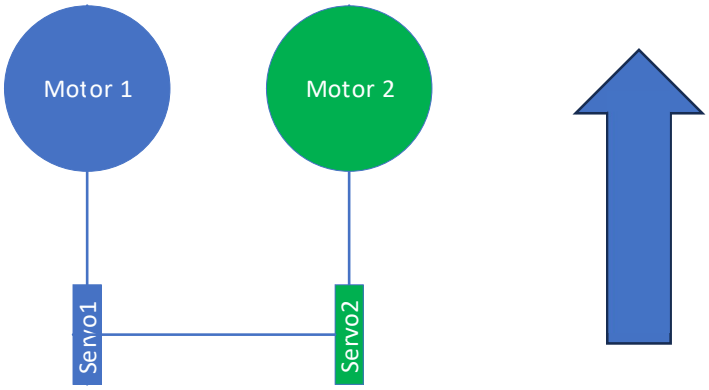


Table 10 – Bicopter Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle, roll	D0	
Motor 2	Throttle, roll	D1	
Servo 1	Pitch, yaw	D2	
Servo 2	Pitch, yaw	D3	

6.1.7.7 CHINOOK_COPTER

This mixer offers control of 2 motor which are controlled by throttle and pitch demands and 2 servo's to mix roll and yaw controls. In config.hpp set CHINOOK_COPTER to true to enable.

Figure 22 – CHINOOK_COPTER

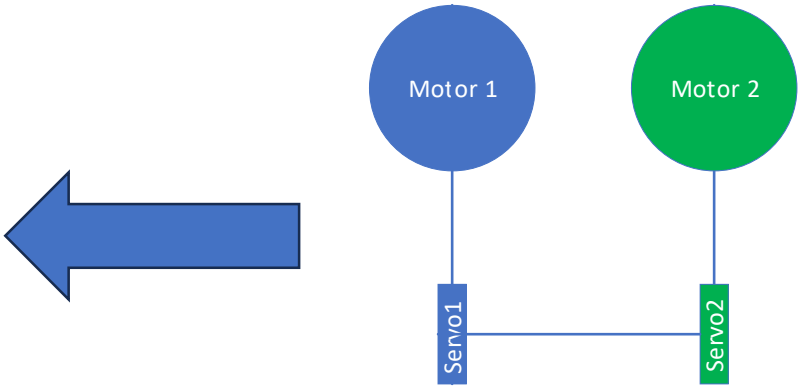


Table 11 – Chinook Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle, pitch	D0	
Motor 2	Throttle, pitch	D1	
Servo 1	Roll, yaw	D2	
Servo 2	Roll, yaw	D3	

6.1.7.8 SINGLE_COPTER

This mixer offers control of 1 motor which is controlled by throttle demand and 4 servos arrange in a + configuration which mix roll, pitch and yaw controls. In config.hpp set SINGLE_COPTER to true to enable.

Note: The yaw servo must have sufficient I-gain to hold heading and overcome motor torque.

Figure 23 – SINGLE_COPTER

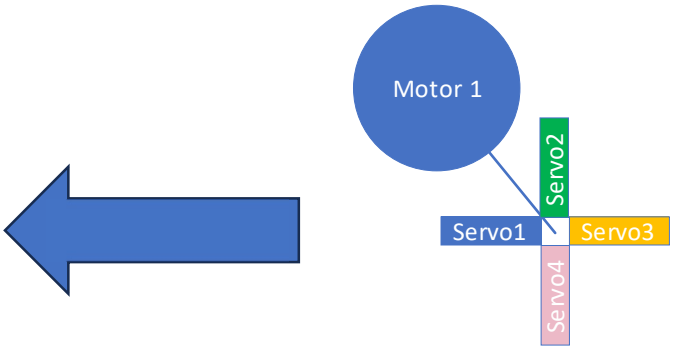


Table 12 - Singlecopter Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle, pitch	D0	
Motor 2	Unused	D1	Must declare as LEDC channel pairs share a common timer which prevents different PWM frequencies i.e. motor refresh and servo refresh.
Servo 1	Roll, yaw	D2	
Servo 2	Pitch, yaw	D3	
Servo 3	Roll, yaw	D8	
Servo 4	Pitch, yaw	D9	

6.1.7.9 DUAL_COPTER

This mixer offers control of 2 motors which are controlled by throttle and yaw demands and 2 servos arrange in a + configuration for Roll and pitch control. Motors must be coaxial design or mounted directly above/below each other. In config.hpp set DUAL_COPTER to true to enable.

Figure 24 – DUAL_COPTER

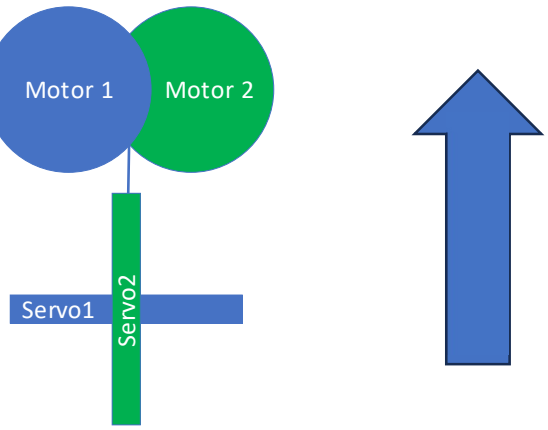


Table 13 – Dualcopter Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle, yaw	D0	Must be coaxial motors, or coaxially mounted.
Motor 2	Throttle, yaw	D1	
Servo 1	Pitch	D2	
Servo 2	Roll	D3	

6.1.7.10 TRI_COPTER

This mixer offers control of 3 motors which are led by throttle, roll and pitch and 1 servos for yaw control. In config.hpp set TRI_COPTER to true to enable.

Note: The yaw servo must be angled s few degrees to counter torque and have I-gain to hold heading and overcome any residual motor torque.

Figure 25 – TRI_COPTER

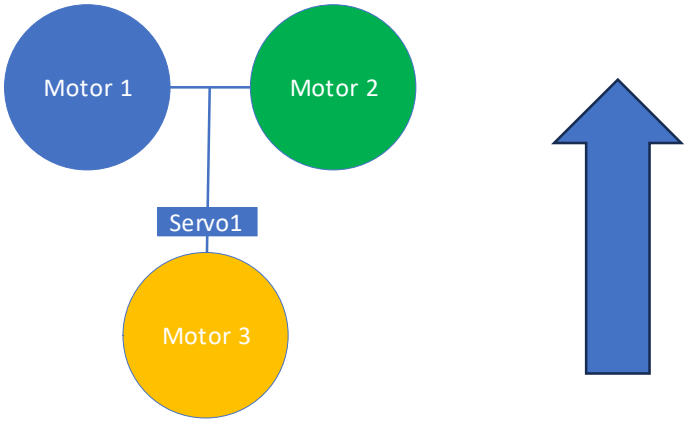


Table 14 – Tricopter Actuators

Actuator	Function	IO Pin	Comment
Motor 1	Throttle, roll, pitch	D0	
Motor 2	Throttle, roll, pitch	D1	
Motor 3	Throttle, roll, pitch	D2	
Motor 4	Unused	D3	Must declare as LEDC channel pairs share a common timer which prevents different PWM frequencies i.e. motor refresh and servo refresh.
Servo 1	Yaw	D8	

6.2 Servo/Motor Refresh Rate

To optimise performance of you model you will need to set servos and esc rates to their highest possible. Always consult the manufacturers manual/instructions on maximum refresh rates as you may permanently damage servos by driving them with too higher rate. In Config.hpp find the following lines of code and update to suit your radio equipment:

Figure 26 - Servo/Motor Refresh Rate Settings

```
//Refresh rates of servos & motors, chose from the following but ensure you servos/ESC are capable of the rate set !
//IS_50Hz, IS_100Hz, IS_150Hz, IS_200Hz, IS_250Hz, IS_300Hz, IS_350Hz, IS_ONESHOT125
//If you are using analog servos use IS_50Hz. If configuring multicopter use IS_ONESHOT125 for BLHeli ESCs.
//Note: higher refresh rates give better output resolution and flight controller reponse.
static constexpr LedcServo::RefreshRate SERVO_REFRESH_RATE = LedcServo::RefreshRate::IS_150Hz;
static constexpr LedcServo::RefreshRate MOTOR_REFRESH_RATE = LedcServo::RefreshRate::IS_ONESHOT125;
```

6.2.1 SERVO_REFRESH_RATE

The ESP32 LEDC PWM timer peripheral is set to 14-bit resolution. This allows a range of servo PWM/PPM refresh rates to be set from 50Hz up to 2KHz and therefore allows use of analogue or digital servos. Check the manufacturers data on your servos and set the appropriate refresh rate for your servos via the RefreshRate::IS_xxxHz enumeration.

CAUTION: Setting a servo refresh rate higher than the manufacturer recommends may prevent them from working correct and even damage them.

Due to how the LEDC PWM peripheral works, the lower the refresh rate results in lower resolution output for the servo i.e. lower resolution equals fewer servo steps over full travel of the servo. Table 15 details the servo resolution relationship and clearly shows that digital servos will offer better precision.

Table 15 - Servo Refresh Verses Resolution

RefreshRate Enumeration	Resolution/Servo Steps
IS_50HZ	819
IS_100HZ	1638
IS_150HZ	2458
IS_200Hz	3276
IS_250HZ	4096
IS_300HZ	4915
IS_350Hz	5461
IS_ONESHOT125 (2KHz implementation)	4096 (ESC output)

6.2.1.1 IS_ONESHOT125

When this is enabled motor outputs will use the Oneshot125 protocol as found on BLHeli ESCs. The oneshot125 protocol is a fast 4KHz PWM/PPM output with pulse width ranging from 125-250us.

This protocol allows faster and more accurate updates to the motors for multicopters, and when used for differential thrust aircraft and may increase stability in prop-hanging type manoeuvres.

Note: While Oneshot125 can output at up to ~4KHz PlainFlightController only outputs at 2KHz, this is done as the maths for the PWM/PPM works out nicely at ~12-bit resolution (4096 steps). Also, the main flight controller loop runs at 1KHz so there is no benefit to update above 1Khz, or at the maximum of 4KHz.

Note: BlHeli ESCs may have had different end points set. Either use the BlHeli App to configure or adjust the ONESHOT125_MIN_TICKS and/or ONESHOT125_MAX_TICKS constant expressions in LedcServo.hpp.

Note: There is no RC interpolation within PlainFlightController, as a result you will hear ~143Hz noise within the motors when moving the Tx sticks. Do not be alarmed this is just the ESC's adjusting the speed of the motors to suit the newest RC commands.

6.3 Additional Configuration

For every model type you may also have the following options to configure. Note that not all configurations can be applied to model types or multicopters.

6.3.1 USE_FLAPS (Fixed Wing Only)

When using PLANE_FULL_HOUSE or PLANE_FULL_HOUSE_V_TAIL that has two aileron servos you can enable the flaps function. Set USE_FLAPS constexpr to true if you wish to use flaps.

Note: do not enable for multicopters.

6.3.2 USE_DIFFERENTIAL_THRUST (Fixed Wing Only)

If your fixed wing model has twin motors then by uncommenting this allows you to use differential thrust via rudder Tx commands. Differential throttle will be gyro stabilised in rate and self-levelled modes.

Note: do not enable for multicopters.

6.3.3 USE_HEADING_HOLD (Fixed Wing Only)

When enabled a heading hold function will be mapped to the rudder. This is done by applying I-gain to the PIDF yaw calculation when heading hold switch is enabled.

The intention of heading hold is to assist tracking during manoeuvres such as run off ground take off's, vertical climbs, loops and prop hanging.

Note: Comment out if you have a transmitter with only 6 channels.
Note: do not enable for multicopters as it is automatically enabled.

6.3.4 USE_LOW _VOLTS_CUT_OFF

When enabled battery voltage will be monitored and the system will react to low battery volts by actively reducing the throttle output(s) and ultimately cutting the throttle.

Motor power reduction starts when the battery falls to 3.4V per cell, and complete throttle cut off occurs at 3.3V per cell. If you are using lithium-ion cells then these values will need to be changed to what the manufacturer recommends and will be approximately 2.7V and 2.5V respectively.

This option is only recommended when using BlHeli type ECSs where low voltage detection is not part of the ESC software.

Note: For this to operate correctly you must have the potential divider hardware connected to D10 analogue input.

6.3.5 USE_250_DEGS_SECOND

When set to true it sets the maximum degrees per second the gyro will output to 250 degrees per second. Recommended for models that can do mild/pattern aerobatics.

CAUTION: Ensure USE_500_DEGS_SECOND is set to false or you will get a compile failure.

6.3.6 USE_500_DEGS_SECOND

When set to true it sets the maximum degrees per second the gyro will output to 500 degrees per second. Only set true if your model can achieve more than 250 degrees/second rates of rotation, as this setting will lower resolution of the sensor output and may reduce performance for models that cannot manoeuvre at these rates.

CAUTION: Ensure USE_250_DEGS_SECOND is set to false or you will get a compile failure.

6.3.7 USE_EXTERNAL_LED

When set to true a second instance of LED will be instantiated on the defined IO pin. This will be used to indicate the flight controller mode and is useful if the flight controller is hidden inside a fuselage.

The LED can be configured to be sink or source configuration, this is done at the point the where the LED object is declared in FlightControl.hpp.

6.3.8 USE_ACRO_TRAINER

For fixed wing model types, when set to true the acro trainer mode will be use instead of self-levelled mode. This mode self-levels when pitch and roll sticks are centred. When pitch and/or roll are not centred rate mode is used. This mode is similar to the horizon mode of other flight controllers. It allows aerobatics but hands off sticks will perform an auto recovery to level.

It is essential your Tx trims are set to neutral, or this mode is unlikely to work. The model should be trimmed mechanically to fly straight and level.

Note: This mode is always used for multicopters.

6.3.9 USE_ONBOARD_NEOPIXEL

Waveshare ESP32-S3 Zero and Tiny have an onboard RGB Neopixel LED. When set to true this will allow the onboard Neopixel to function. Note: Ensure the correct IO pin is set for the Neopixel. For the Waveshare

Zero and Tiny this is pin 21. Also be aware that Neopixel are serial controlled and take a finite amount of time. This may affect the 1KHz loop rate with some configurations.

6.3.10 REVERSE_PITCH_CORRECTIONS

When in self-levelled or rate modes if you find that the pitch correction is in the wrong sense then use this define to change the direction to the correct sense.

Note: If you find that control surfaces are correcting in the right sense, but Tx commands give wrong sense then reverse the output direction within you transmitter.

6.3.11 REVERSE_ROLL_CORRECTIONS

When in self-levelled or rate modes if you find that the roll correction is in the wrong sense then use this define to change the direction to the correct sense.

Note: If you find that control surfaces are correcting in the right sense, but Tx commands give wrong sense then reverse the output direction within you transmitter.

6.3.12 REVERSE_YAW_CORRECTIONS

When in self-levelled or rate modes if you find that the yaw correction is in the wrong sense then use this define to change the direction to the correct sense.

Note: If you find that control surfaces are correcting in the right sense, but Tx commands give wrong sense then reverse the output direction within you transmitter.

6.3.13 CALIBRATE_ESC

CAUTION: ALWAYS REMOVE PROPELLER(S) WHEN CALIBRATING ESC(S). YOU RISK SERIOUS INJURY IF YOU DO NOT.

When set to true, at power on the flight controller will output maximum throttle value and hold for CALIBRATE_HOLD_TIME, then set minimum throttle value. This should calibrate the ESC(s) full range to match the flight controller output.

Note: When calibrating you will need to use the flight battery to power the ESC(s) as the 5V USB supply may not power the ESC. Depending on your ESC(s) you may need to reduce CALIBRATE_ESC_DELAY to prevent it from entering programming mode.

Once calibrating has finished, code execution is halted. You will need to set CALIBRATE_ESC to false and reprogram the flight controller.

Note: Always calibrate multicopter ESCs before the first flight.

Note: For the 3.3V version avoid powering ESC via the USB connection – you may damage your laptop/USB power supply.

6.3.14 REVERSE_SERVO_x

These settings should only need to be used when you have same handed servo horns used on ailerons etc. Consider this a last resort to resolve a mechanical control linkage issue with your model.

6.3.15 USE_PROP_HANG_MODE (Fixed Wing Only)

When set to true Tx channel 9 will be used to activate the self-levelled prop-hang mode where rudder and elevator are used to keep the models nose vertical.

Note: your model must be capable of prop-hanging in the first place !

There are several settings to USE_PROP_HANG_MODE which are described below. These will hopefully be rationalised in future releases.

6.3.15.1 REVERSE_PROP_HANG_x_CORRECTIONS

When prop-hang mode is active, if you find that rudder or elevator is correcting in the wrong sense then set the appropriate constexpr to true to reverse the corrections.

6.3.15.2 PROP_HANG_TAIL_SITTER_MODE

Set to true for tail-sitter mode where roll stick commands the model's yaw, and yaw stick commands the models roll when prop-hang mode is enabled.

6.3.15.3 PROP_HANG_REVERSE_x_DEMAND

When prop-hang mode is active, if you find that yaw/roll controls are working in the correct sense, but the Tx commands are reversed, you can use these settings to reverse the Tx commands for the appropriate control surface.

This problem is generally associated with having tail sitter mode enabled. This will hopefully be rationalised in future releases.

6.3.15.4 IMU_ROLLED_RIGHT_90

When set to true the IMU can be rolled to the right 90 degrees of normal orientation. This allows the flight controller to be mounted on the side of a models fuselage or in narrow fuselages.

6.3.15.5 IMU_ROLLED_180

When set to true the IMU can be rolled 180 degrees of normal orientation i.e. flipped upside down on the roll axis. This allows the flight controller to be mounted on the underside of a model fuselages.

6.3.15.6 ACRO_TRAINER_LEVEL_RATE

Due to the way acro trainer works we may get a large step demand when sticks are centred, and the model is at an extreme angle. This would result in the maximum rate demand being applied and can lead to levelling overshoot particularly on fast acting multicopters. This parameter instead allows us to fine tune how quickly we return to level in degrees per second and is generally set at a lower rate than your flight control rates. Consider it to be a levelling strength parameter.

Caution: Do not exceed and stay well below the degrees per second you have set for the gyro.

Caution: Do not exceed your set control rates degrees per second

Caution: Setting this too high can lead to flight gains over reacting and the craft returning to level faster than your set control rates and gyro degs/sec setting which can cause the Madgwick filter to loose true level – particularly on fast acting multicopters.

Caution: Do not exceed your models maximum degrees per second capabilities or you will get I-gain build up and overshoot level.

6.3.15.7 IDLE_UP_VALUE (Multicopters Only)

This is the motor idle speed when armed. Adjust this if IDLE_UP_VALUE speed is too low/high, for best performance it needs to be set to a speed where the model is a few throttle clicks away from lifting off.

6.3.15.8 MIN_THROTTLE_VALUE (Mutlicopters Only)

Absolute minimum motor speed to prevent motors from stopping/stalling. Adjust this if MIN_THROTTLE_VALUE speed is too low/high. This setting allows the PID controller to work around the IDLE_UP_VALUE to try and give an 'air mode' locked in feeling when at minimum throttle.

6.3.15.9 TX_DEAD_BAND_xxx

Radio control transmitters may have a bit of jitter or inaccuracy around centred positions. This dead band can be used to mask this jitter or inaccuracy.
If you find that certain control surfaces are drifting in rate mode with Tx sticks and trims centred, then try increasing the dead band.

Note: Control surface drift could also be caused by a poor power on gyro calibration i.e. model was moving during calibration, but not by enough to trip a calibration reset.

6.4 DEBUG

Debug defines are used to help problem solve the flight controller. It is recommended that each of these is enabled one at a time to confirm correct operation of the flight controller following its construction. Use Arduino IDE 'Serial Monitor' to view the data, Table 16 details what debug data you can view.

Section 8 How To Test gives greater detail on each debug setting and the data output.

Table 16 - Debug Data Output

Debug	Description
DEBUG_SBUS	Outputs raw Sbus data. Confirms correct wiring and that Sbus data is being received.
DEBUG_RC_DATA	Outputs refactored Sbus data that the flight controller uses.
DEBUG_LOOP_RATE	Outputs the loop rate as float in seconds. You should see a value of 0.001 (up to 0.001002 is normal).
DEBUG_BATTERY_MONITOR	Outputs battery volts as a float and other associated data.
DEBUG_MADGWICK	Outputs 3-axis IMU data from the Madgwick filter. This data is used for self levelled flight mode.
DEBUG_GYRO_CALIBRATION	Outputs data relevant to gyro calibration. If you have a noisy gyro then CALIBRATE_MAX_MOTION may need to increase.
DEBUG_MPU6050	Outputs 6-axis data from the gyro. Can be used to confirm the gyro is connected and working correctly.
DEBUG_MOTOR_OUTPUT	Outputs motors 1 to 4 as signed integers. May help with debugging cross mixing of channels.
DEBUG_SERVO_OUTPUT	Outputs servos 1 to 4 as signed integers. May help with debugging cross mixing of channels.

6.5 USB_BAUD

You should not need to change this but can be lowered if you are having USB connection issues.

6.6 IO Pin Allocation

You should not need to change these unless you are modifying the software for your purposes.

Note: Servo and motor allocation changed depending on the model type selected. The table below is for a PLAN_FULL_HOUSE_. See ModelTypes.hpp for exact allocations.

Seeed Studio XIAO ESP32S3 I/O	ESP32S3 GPIO	PlainFlightController IO Allocation
D0/A0	GPIO1	Servo1
D1/A1	GPIO2	Servo2
D2/A2	GPIO3	Servo3
D3/A3	GPIO4	Servo4
D4/I2C_SDA	GPIO5	I2C SDA
D5/I2C_SCL	GPIO6	I2C SCL
D6/TX	U0TXD	External LED
D7/RX	U0RXD	Sbus
D8/A8/SCK	GPIO7	Motor1
D9/A9/MISO	GPIO8	Motor2
D10/A10/MOSI	GPIO9	Battery Monitor

7. Transmitter Configuration

To obtain full functionality of PlainFlightController you will need an 8-channel transmitter. Channels 1 to 4 are your primary flight controls of throttle, aileron, elevator, rudder and need to be set to 100% travel. Auxiliary channels are configured for switch inputs with an example shown in Figure 27, switches must also be set for 100% travel.

Figure 27 - Typical Tx Switch Setup

Table 17 details the requirements of these switch inputs so that flight controller functions operate correctly.

Table 17 - Tx Switch Requirements

Switch	Type	Tx Channel	Usage
Arm Switch	2 position latching	5	>1500us = armed
Mode Switch	3 position	6	Fixed wing: <1250us = pass through >1250us & <1750us = rate mode >1750us = self-levelled or acro trainer mode Multicopters: <1250us = rate mode >1250us & <1750us = acro trainer >1750us = self-levelled mode
Flaps Switch	3 position	7	<1250us = no flaps >1250us & <1750us = flaps 1

			>1750us = flaps 2
Heading Hold Switch	2 position (latching if preferred)	8	>1500us = heading hold active.
Prop-hanging Switch	2 position (latching if preferred)	9	>1500us = self-level prop hang mode

Note: Flight controller will not arm if throttle is high.
Note: Failsafe will be entered if Tx reception is not initiated or is lost.

7.1 Trims

It is essential that your transmitter trims are set to their neutral position, if they are not when you enter rate or levelled modes the flight controller will see the trim(s) offset as a demand and make the model pitch, roll, or yaw.
You will need to set servo control arms/horns perpendicular to the servo body and adjust the linkages mechanically to make the model fly straight and level.

7.2 Arming Switch

Like any flight controller it will need to be armed before the motor(s) will operate. Whilst disarmed all servos will operate in pass through mode i.e. no stabilisation will be applied. Arming is controlled via channel 5.

7.3 Mode Switch

Once armed the flight controller operates in one of 4 main modes of pass-through, rate, self levelled and acro trainer. These modes are controlled via channel 6, a 3 position switch.

7.3.1 Pass Through Mode

When in pass-through mode the transmitter commands are passed directly through to the servos and motor(s). Should you experience any control problems in flight switch back to pass-through mode to disable all PIDF calculations.

7.3.2 Rate Mode

When in rate mode the 3 axis gyro is used to apply stabilisation to roll, pitch and yaw axes. Transmitter stick commands are converted to degrees/second and PIDF calculations are applied for each axes based on stick commands and gyro data, these calculations depend on the gains set for each corresponding axis.

Note: Setting gains too high will likely make the model suffer from oscillations and could make it uncontrollable.

7.3.3 Self-Levelled Mode

Self-levelled mode using both the gyro and accelerometer; this mode combines data via a fusion algorithm known as a Madgwick filter which computes the IMUs attitude relative to gravity. Transmitter stick commands are converted to a desired angle in degrees and PIDF calculations are applied for pitch and roll axes based on stick commands and IMU attitude, these calculations depend on the gains set for each corresponding axis. Aelf-Levelled mode will not allow the pilot to exceed a predefined pitch or roll angle.

7.3.4 Acro Trainer Mode

Settable for fixed wing model types, when set to true in Config.hpp the acro trainer mode will be use instead of self-levelled mode. This mode self-levels the model when pitch and roll sticks are centred but when pitch and/or roll are not centred rate mode is used so rolls and loops can be achieved. This mode is similar to the horizon mode of other flight controllers. It allows aerobatics to be performed, but hands off sticks will perform an auto recovery to level flight.

It is essential your Tx trims are set to neutral, or this mode is unlikely to work. The model should be trimmed mechanically to fly straight and level.

Note: This mode is always used for multicopters.

7.4 Flaps Switch

When enabled in the software a 3-position switch driven by channel 7 will give either, no flaps, half flaps, or full flaps. Currently half flaps result in 25% of servo travel and full flaps 50% of servo travel.

7.5 Heading Hold Switch

When enabled in the software a 2-position switch driven by channel 8 will give a heading hold function to the rudder (yaw axis). This is done by applying I-gain to the PIDF yaw calculation when heading hold switch is enabled.

Caution: Do not attempt to fly with this permanently enabled as the heading hold will fight an aileron turn.

The intention of heading hold is to assist tracking during manoeuvres such as run off ground take off's, vertical climbs, loops, prop hanging and knife-edge flight. The user will need to switch this heading hold function on and off as and when needed during flight manoeuvres.

7.6 Prop-hanging Mode Switch

Self-levelled prop-hanging mode using both the gyro and accelerometer; this mode will try to keep the models nose pointed vertically in prop-hang. Uses the Madgwick filter which computes the IMUs attitude relative to gravity. Transmitter stick commands are converted to a desired angle in degrees and PIDF calculations are applied for pitch and yaw axes based on stick commands and IMU attitude, these calculations depend on the gains set for each corresponding axis. Note that this mode can also be used for tail-sitter VTOLs.

Note: Prop-hanging mode overrides passthrough, gyro rate and self-levelled modes.

Note: You can set a tail sitter mode where yaw and roll controls are swapped when in prop-hang mode.

Caution: Your model must be capable of prop hanging in the first place for this mode to be able to work.

8. How To Test

Following your build of your PlainFlightController controller you will need to test the following items. Testing is done via setting the DEBUG_ constant expressions in Config.hpp to true. Only set one DEBUG_ constant expression to true at one time or the data displayed will be unreadable.

Note: If at any time your flight controller refuses to program, remove USB from your computer, press and hold the boot button on the XIAO ESP32 then insert the USB and after 1 second of it being powered release the boot button. Now reprogram.

Note: Always set DEBUG_ items to false before flying as the data output may slow the control loop and affect flight performance.

Caution: Ensure propeller(s) are removed when connecting USB with flight battery in circuit. Inserting/removing the USB can cause the XIAO to reset and result in undefined behaviour of motor outputs i.e. motors may operate !

Caution: The 1s circuit will back feed 5V USB power to the ESC's, ensure ESC's can tolerate 5V and remove propeller(s) and note that inductive spikes from motors may damage the connected USB device.

8.1.1 LED Status

The LED indicates the status of the flight controller via different flash sequences, these are detailed in Table 18.

Table 18 - LED Sequences

Mode	Sequence	Neopixel Colour
Initialising/calibrating	Fast on/off flash	White
Failsafe	Very fast on/off flash	Purple
Disarmed	Slow 1 second on/off flash	Green
Armed & passthrough mode	1 quick flash with following pause	Red
Armed & rate mode	2 quick flashes with following pause	Red
Armed & self-levelled mode	3 quick flashes with following pause	Red
Armed & acro trainer mode	4 quick flashes with following pause	Red
Waiting to disarm	5 quick flashes with following pause	Blue
WiFi Configuration	6 quick flashes with following pause	Yellow
Fault	7 quick flashes with following pause	Purple
Armed & prop-hang mode	8 quick flashes with following pause	Red

Verify that your Armed switch and Mode switches operate and give all LED sequences shown in Table 18.

Note: There is no flash sequence for heading hold as it is a transparent command used in rate and self-levelled modes.

8.2 DEBUG_GYRO_CALIBRATION

In Config.hpp set DEBUG_GYRO_CALIBRATION to true and program your flight controller.

Note: When the flight controller gyro calibrates at power on, small or sustained movements will cause it to halt and to retry calibration. Calibration will not complete until the flight controller is still.

Place the flight controller on a flat surface, ensure it is still and plug the USB cable into your flight controller and computer.

On Arduino's Serial Monitor you should see messages similar to:

```
Calibration...
Calibration...
Calibration...
Calibration complete...
x: 43    y: 171    z: 55
```

Note: you may see 'Calibration reset !' messages if your flight controller was not still at power on.
Note: Your X, Y and Z calibration values will be different.

If you do not see these messages, then you may have a power or I2C wiring issue between the ESP32 XIAO and the MPU6050.

Alternatively, if you have a noisy gyro you may need to increase CALIBRATE_MAX_MOTION to a larger value. If you do need to increase this value, consider changing the MPU6050 for another. A lot of these MPU6050's are Chinese clones and the quality varies.

Note: The calibration state is only exited on successful calibration. Should you be moving the flight controller or model during calibration then it will not exit, and the flight controller cannot be armed. Calibration state is indicated by a constant 250ms second LED flash.

8.3 DEBUG_MPU6050

In Config.hpp set DEBUG_MPU6050 to true and then reprogram your flight controller.

The Serial Monitor should now be continually displaying gyro and accelerometer data. Confirm that rotating/tilting your flight controller in X, Y and Z axes causes the corresponding axis data to change. When still your Arduino Serial Monitor you should display something similar to:

gx:0.02	gy:0.03	gz:0.03	ax:0.05	ay:-0.05	az:0.99
gx:0.02	gy:0.03	gz:0.03	ax:0.05	ay:-0.05	az:0.99
gx:0.02	gy:0.03	gz:0.03	ax:0.05	ay:-0.05	az:0.99
gx:0.02	gy:0.03	gz:0.03	ax:0.05	ay:-0.05	az:0.99

Note: All data should change as you tilt and rotate the flight controller.

In the unlikely case that some data is stuck and does not change you likely have a bad MPU6050.

8.4 DEBUG_MADGWICK

If the gyro tests in 8.1.1 and 8.3 are successful, then there is no need to perform this test. However, it may be of interest to you.

In Config.hpp comment set DEBUG_MADGWICK to true then reprogram your flight controller.

The Serial Monitor should now be continually displaying roll and pitch data. You will only see yaw data if you have USE_PROP_HANG_MODE set to true.

Confirm that rotating/tilting your flight controller in X, Y and Z axes causes the corresponding axis data to change between 0 and +-180 for roll, 0 and +- 90 degrees for pitch and yaw (asin or atan usage determines range). On your Arduino Serial Monitor you should see data similar to:

Roll: -0.46, Pitch: -1.75
Roll: -0.46, Pitch: -1.74
Roll: -0.46, Pitch: -1.74
Roll: -0.47, Pitch: -1.78

8.5 DEBUG_LOOP_RATE

There is no need to perform this test unless you have modified the software for your own purposes and wish to check that your changes have not affect the 1KHz loop rate.

In Config.hpp set DEBUG_LOOP_RATE to true then reprogram your flight controller.

The Serial Monitor should now be continually displaying floating-point numbers. The 3 columns of data represent the stable loop time, the actual execution time, and the running average loop time. All data is shown in micro seconds (0.001000s = 1.000ms = 1000us = 1.0KHz). The stable loop time may fluctuate by 2us (0.001000 to 0.001002), this is ok.

0.001000	0.000944	0.000864
0.001001	0.000925	0.000864
0.001001	0.000940	0.000864
0.001001	0.000865	0.000864

Note: 1/0.001000 = 1000Hz (1KHz) loop rate.

8.6 DEBUG_SBUS

In Config.hpp set DEBUG_SBUS to true then reprogram your flight controller.

With you transmitter on and having bound the transmitter to your receiver the Serial Monitor should now display 8 channels of Sbus data (9 if USE_PROP_HANG_MODE is enabled), followed by the lost frame and failsafe bits:

172	992	992	985	172	992	992	992	0	0
172	992	992	985	172	992	992	992	0	0
172	992	992	985	172	992	992	992	0	0

172	992	992	985	172	992	992	992	0	0

Note: Channels 8 to 16 will only be displayed if enabled in the software.

Moving your transmitter sticks should cause corresponding channel data to change. Turning off your transmitter should result in lost frame and failsafe bits at the end of the coms data line to be set to '1'.

If no data is displayed, then check your wiring and that your receiver is bound to your transmitter.

If data is displayed but does not change, ensure you have setup your transmitter correctly to output 8 channels for pitch, roll, yaw, throttle, aux1, aux2, aux3 and aux4 (9 channels when using prop-hang mode).

8.7 DEBUG_RC_DATA

In Config.hpp set DEBUG_RADIO_COMMANDS to true then reprogram your flight controller.

In Serial Monitor verify operation by moving the sticks or turning switches on/off, the outputs should be as described by Table 19.Typical data will look like:

pitch: -819	roll: 0	yaw: 0	throttle: -819	flaps: 1	armed: 0	mode: 2
pitch: -819	roll: 0	yaw: 0	throttle: -819	flaps: 1	armed: 0	mode: 2
pitch: -819	roll: 0	yaw: 0	throttle: -819	flaps: 1	armed: 0	mode: 2
pitch: -819	roll: 0	yaw: 0	throttle: -819	flaps: 1	armed: 0	mode: 2
pitch: -819	roll: 0	yaw: 0	throttle: -819	flaps: 1	armed: 0	mode: 2

Note: When prop-hang mode and/or heading hold are enabled you will see these appear in the data.
Note: It is important to verify that your switches are operating correctly so correct flight modes can be set. Stick command outputs depends on the flight mode set, flight mode/state testing is verified in **Error! Reference source not found.**.

Table 19 - Commands

Command	Outputs
Armed	0 and 1
Mode	0, 1, and 2
Aux1 (Flaps)	0, 1, and 2
Aux2 (Heading Hold)	0 and 1
Aux3 (Prop-Hang)	0 and 1
Throttle	Is a signed number that varies between SBUS::MIN_NORMALISED_US and SBUS:: MAX_NORMALISED_US.
Aileron	When disarmed is a signed number that varies between SBUS::MIN_NORMALISED_US and SBUS:: MAX_NORMALISED_US. When armed values depend on flight mode and your set rates or max angles.
Pitch	When disarmed is a signed number that varies between SBUS::MIN_NORMALISED_US and SBUS:: MAX_NORMALISED_US. When armed values depend on flight mode and your set rates or max angles.
Yaw	When disarmed is a signed number that varies between SBUS::MIN_NORMALISED_US and SBUS:: MAX_NORMALISED_US. When armed values depend on flight mode and your set rates or max angles.

8.8 DEBUG_BATTERY_MONITOR

You should only need to use this if you are unable to use the WiFi web configurator as battery voltage calibration can be done on it.

Either way you will need a multimeter to measure DC voltage of your flight battery. If motor(s) is connected ensure your propeller(s) are removed.

In Config.hpp set DEBUG_BATTERY_MONITOR then reprogram your flight controller.

Serial Monitor will now be displaying several items related to the battery monitor, these being the raw ADC value, number of cells detected, battery scaler, and battery voltage. Use your multimeter to verify that the battery voltage given by the flight controller is the same as it reads on your multimeter. It will likely be different due to resistor tolerances, if it differs by more than 0.1V then you can calibrate it via the WiFi web configurator or manually edit the BATTERY_SCALER in Config.hpp. Note if you manually edit it in the code then you will need to erase all flash when programming for it to take effect. Note the calculated voltage is heavily filtered and will slightly lag the battery voltage.

```
ADC: 2141, Cells: 2, Scaler:0.00378, volts:8.10
ADC: 2144, Cells: 2, Scaler:0.00378, volts:8.10
ADC: 2147, Cells: 2, Scaler:0.00378, volts:8.10
ADC: 2141, Cells: 2, Scaler:0.00378, volts:8.10
```

CAUTION: If you are powering from the flight controller from 1s 3.3V lipo battery, do not plug in the flight battery and USB in at the same time, as the USB will feed 5V to the lipo battery. For 1s lipo powered system calibrate the battery voltage via the USB 5V supply.

Caution: Ensure propeller(s) are removed when connecting USB with flight battery in circuit. Inserting/removing the USB may cause the flight controller to reset and output an undefined stated to an ESC.

Caution: The 1s circuit will back feed 5V USB power to the ESC's, ensure ESC's can tolerate 5V and remove propeller(s).

8.9 DEBUG_SERVO_OUTPUT

There is no need to perform this test unless you have modified the software for your own purposes and wish to check that your servo mixing output are working as you intended.

In Config.hpp set DEBUG_SERVO_OUTPUT to true then reprogram your flight controller.

Data is displayed as 'timer ticks' required to reproduce the 1ms to 2ms pulse. Data range will depend on the refresh rate you have chosen. Section 6.2 Servo/Motor Refresh Rate indicates the relation of servo resolution, or range of timer ticks you should expect to see.

```
servo1: 3510, servo2: 3510, servo3: 3510, servo4: 3510
servo1: 3510, servo2: 3510, servo3: 3510, servo4: 3510
servo1: 3510, servo2: 3510, servo3: 3510, servo4: 3510
servo1: 3510, servo2: 3510, servo3: 3510, servo4: 3510
```

8.10 DEBUG_MOTOR_OUTPUT

There is no need to perform this test unless you have modified the software for your own purposes and wish to check that your servo mixing output are working as you intended.

In Config.hpp set DEBUG_MOTOR_OUTPUT to true then reprogram your flight controller.

Data is displayed as 'timer ticks' required to reproduce the 1ms to 2ms pulse. Data range will depend on the refresh rate you have chosen. Section 6.2 Servo/Motor Refresh Rate indicates the relation of servo resolution, or range of timer ticks you should expect to see.

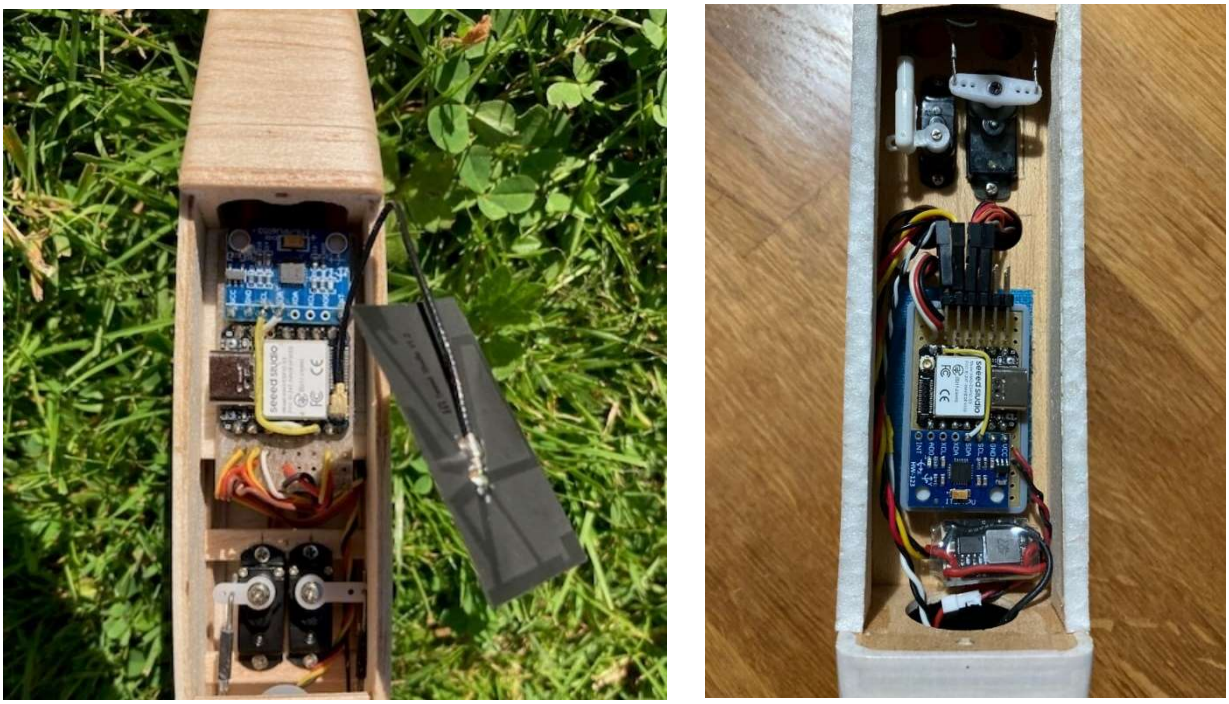
```
motor1: 4090, motor2: 4090, motor3: 4090, motor4: 4090
motor1: 4090, motor2: 4090, motor3: 4090, motor4: 4090
motor1: 4090, motor2: 4090, motor3: 4090, motor4: 4090
motor1: 4090, motor2: 4090, motor3: 4090, motor4: 4090
```

9. How To Install In Your Model

The flight control needs to be mounted with the MPU flat as shown in Figure 28. The gyro Z plane must be parallel to gravity, the X axis must be aligned with the pitch moment, and Y aligned with roll.

The IMU/MPU6050 can face forwards or backwards i.e. X axis +ve facing forward to the nose or towards the tail. However, depending on the orientation set you may need to configure pitch/roll corrections as described in section 6 How To Configure Your Model.

Figure 28 - Flight Controller Installation



The flight controller should be mounted on foam tape or Velcro to help absorb any vibrations and ensure that no wires or objects can vibrate or knock onto the flight controller when in flight. Ensure that the WiFi antenna is not surrounded by other wiring or metallic objects or you may experience connection issues. Be aware the WiFi aerial and connector has been found to be quite susceptible to damage from heavy arrivals so try to secure or stow it.

10. How To Tune

Firstly, if you have never tuned a flight controller or do not understand PIDF then seek assistance or training as this document assumes you have knowledge of flight controllers and understand PIDF control systems.

To tune your flight controller at minimum you will have to set rates and PIDF parameters for each axis. The following sections detail how this can be done.

10.1 WiFi Web Configurator

PlainFlightController V2.0.0 onwards supports a WiFi web-based configurator to allow easy tuning of flight parameters on any smart device with a web browser. This means configuration of the following parameters can be done when out flying your model without the need to completely reprogram the ESP32:

Table 20 - Web Configurable Parameters

Parameters	Comments
PIDF	Proportional, integral, differential and feed forward for roll, pitch and yaw axis.
Rates	Maximum degrees per second of rotation for roll, pitch and yaw axis.
Max Self-Levelled Angles	Maximum allowed bank angles for pitch and roll when in self-levelled mode.
Levelled Trim	Fine trim self-levelled flight with pitch and roll trims.
Servo Trims	Used to fine tune the servo horn to centre position. Not to be used for incorrect mechanical linkage adjustment – you will lose range of servo motion if not mechanically set correct.
Battery Scaling	Fine tune the battery voltage monitor.

Note: If you are unable to use the PIDFs, Rates and trim parameters in defines.h are used for the initial setup only. Changing these parameters and programming the ESP32 will not change flight settings as they did for previous versions of PlainFlightConroller. All changes must be made through the web interface to take effect.

10.1.1 How To Connect

To connect your smart device to your flight controller you must follow these easy steps:

1. Model is disarmed (LED slow 1 second on/off flash)
2. Throttle is moved to fully open position

When the model is disarmed and throttle set to maximum the WiFi access point is started by the ESP32, this is indicated by 5 repeating quick LED flashes. Once established, go to your WiFi settings on your smart device and you should see ‘PlainFlight’ appear in the list of WiFi providers.

On selecting ‘PlainFlight’, for the first ever connection you will be prompted to enter a password (12345678). WiFi connection can be instantaneous, or take up to ~15 seconds.
Should it not connect, close the throttle, and wait until ‘PlainFlight’ disappears from your smart device, then reopen the throttle to reinitialise.

When your smart device shows connection is established open your web browser and enter the address 192.168.4.1 and the configurator interface should load. Should it not load your virus checker may be blocking. The web interface has been tested on iPhone, iPad, Android phone, and Windows 10, with Chrome, Edge, and Safari browsers.

Table 21 - Web Configurator Information

Parameters	Data
WiFi Provider	PlainFlight
Password	12345678
Web Address	192.168.4.1

The following devices have been tested with the web configurator:

Table 22 - Tested Smart Devices/Browsers

Device	Browser	Comments
Laptop (Win 10)	Microsoft Edge	Works well
Iphone SE	Safari	Iphone takes long time to connect to WiFi. Safari sometimes reports ‘could not parse’
Iphone SE	Chrome	Iphone takes long time to connect to WiFi.
Samsung Galaxy	Chrome	Works well
Ipad (~2020)	Safari	Works well

Note: I have found my dated iPhone to be unreliable in establishing a WiFi connection, but all other devices generally work first time. I have also found that the WiFi connector on the XIAO to be very fragile and can fail rendering the WiFi inoperable.

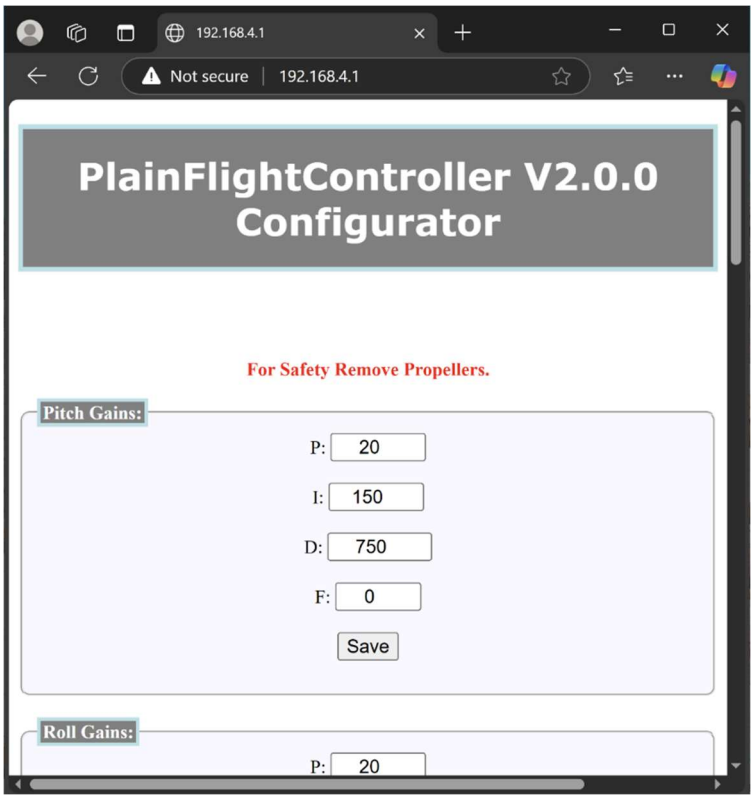
Caution: Ensure the WiFi aerial is connected to your ESP32-S3 or you may permanently damage it.

Caution: For safety it is strongly advised to remove all propellers before entering WiFi configurator mode.

Note: The password (12345678) is used to prevent unsecure network warnings from your smart device.
Note: Keep WiFi aerial away from other wiring or metal objects to ensure reliable connection.
Note: Keep your radio transmitter ~1m away from you model and smart device when using WiFi to help reduce any connection issues.

On successful connection you should be presented with the PlainFlightController Configurator as shown in Figure 29. The configurator is based on HTML forms for data entry, this breaks down data items into groups for easy data entry. Values shown in the web configurator on loading are those of your model it has connected to.

Figure 29 - Web Configurator On MS Edge Browser



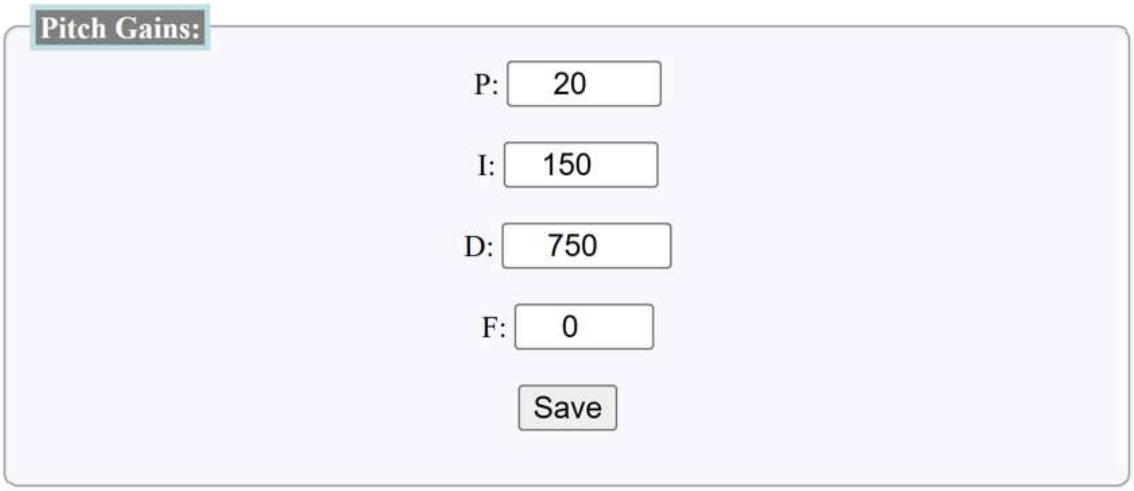
10.1.2 PIDF Configuration

Proportional, Integral, Differential, and Feed Forward for Pitch, Roll, Yaw axis can be set to allow easy tuning of your model's gain parameters.

To tune pitch, roll and yaw, you will be presented with three separate forms for Pitch Gain, Roll Gains, and Yaw Gains. The Pitch Gains example given in Figure 30 clearly shows P, I, D, and F that can be configured. Enter the required values you want then press the save button associated with the gains you have just changed.

On saving the form will reload and show the values saved.

Figure 30 - Pitch Gains Form



10.1.2.1 Rate Configuration

Rate Configuration allows you to set the maximum degrees per second that your model will rotate for Pitch, Roll and Yaw.

Note the configurator will not allow you to set beyond the maximum value you have set within the PlainFlightController code (250 or 500 degs/s). Also your model should be capable of rotating at the rates you are setting.

Enter the values you require, then press save.

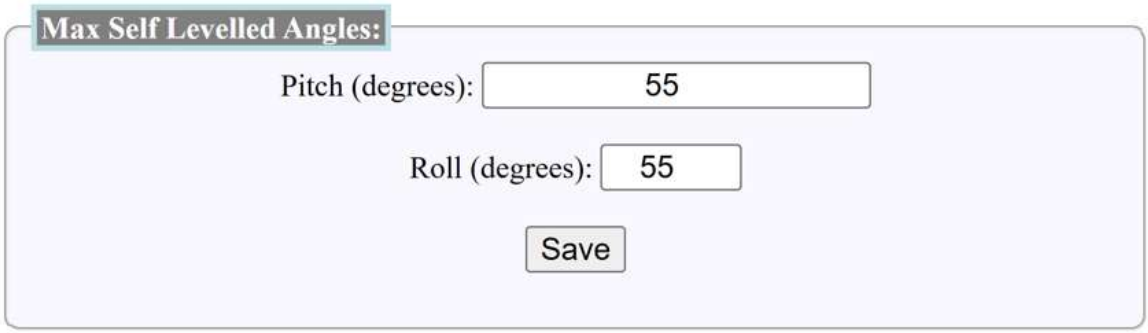
Figure 31 – Rates Form

10.1.2.2 Max Self-Level Angle Configuration

This form allows you to set the maximum pitch and roll angles that are allowed when flying in self-levelled mode. Setting these too low will result in your model feeling unresponsive and may make turns impossibly large. It is recommended that you use value around 60 degrees.

On editing the values, press save to store the new settings.

Figure 32 - Max Self Levelled Angles Form



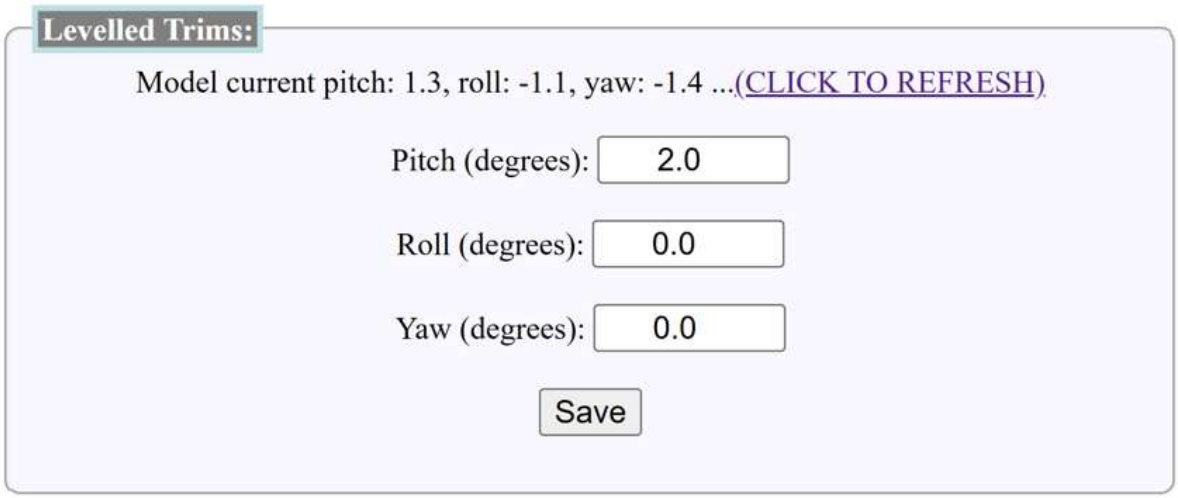
10.1.2.3 Self-Levelled Trim Configuration

If you find that your model does not maintain level flight when flying in self-levelled mode, then adjust the Levelled Trims values. Note these values are signed and depends on the orientation of you flight controller as to what signed value you require.

The easiest way to initially set these trims is to place you model on a flat/level surface or position the model in its flying attitude. Now use the 'Model current' values as your trims but make them the opposite sign. i.e. with current pitch of 1.3 degrees, apply a pitch trim of -1.3.

On editing the values, press save to store the new settings.

Figure 33 - Levelled Trims Form



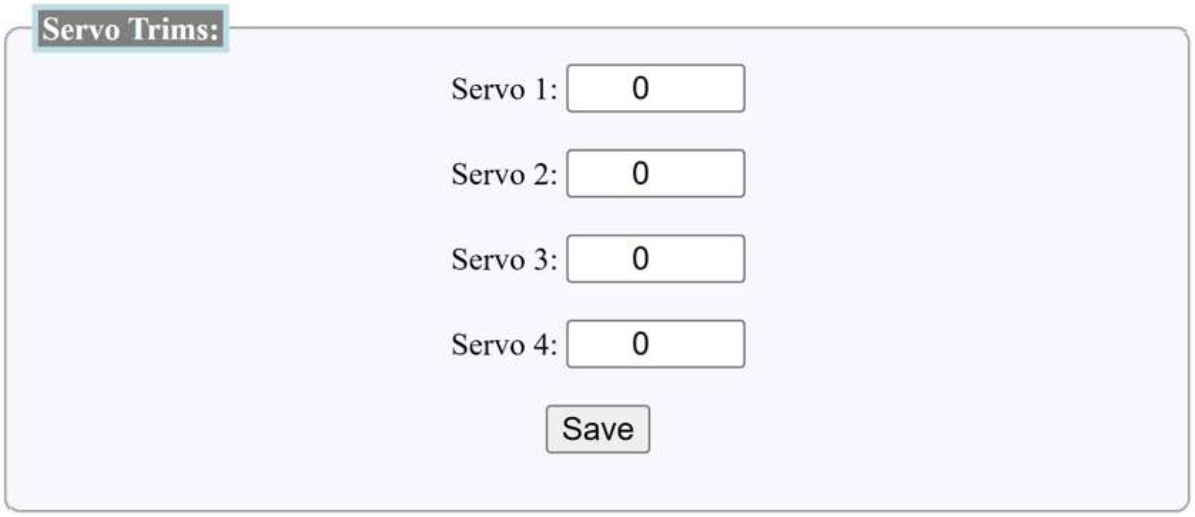
10.1.2.4 Servo Trim Configuration

Note: Servo Trims are only to be used for centering the servo horn due to bad servo spline alignment. Always mechanically adjust control surfaces to be neutral with the servo horn at 90 degrees to the servo body. If you do not do this and use servo trims then you will reduce servo travel in one direction.

Servo trim depends on what model configuration you have selected, see section 6.1 Model Mixer for definitions of servo numbers and what control surface they relate to.

On entering trim values, press save and you will instantly see the new trim value take effect on the servo.

Figure 34 - Servo Trims Form

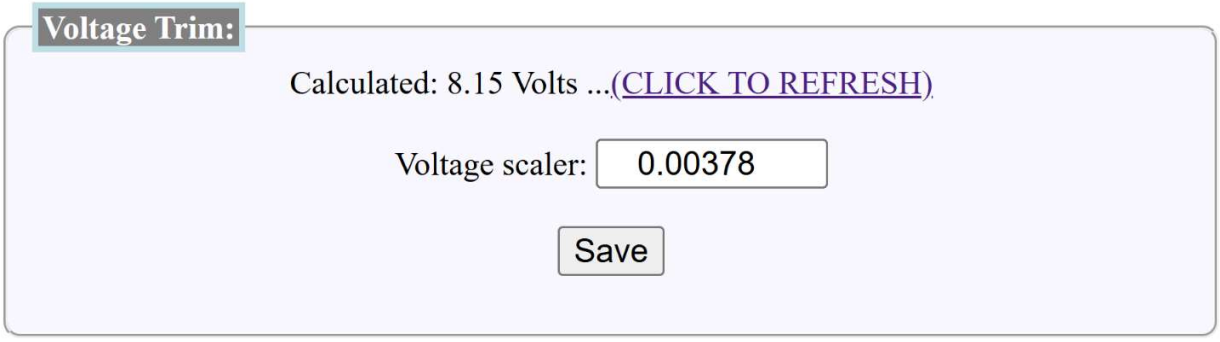


10.1.2.5 Battery Voltage Trims

If you are using USE_LOW_VOLTS_CUT_OFF to monitor the battery voltage, then you will likely need to calibrate the battery voltage due to resistor tolerances with the battery trim function.

On entering trim values, press save and you will instantly see the new trim value take effect on the calculated voltage. You can refresh the page to get the latest calculation, it is wise to do this as the voltage filtering may delay a true reading.

Figure 35 - Battery Voltage Trim



10.2 Reprogramming ESP32-S3 To Tune Parameters

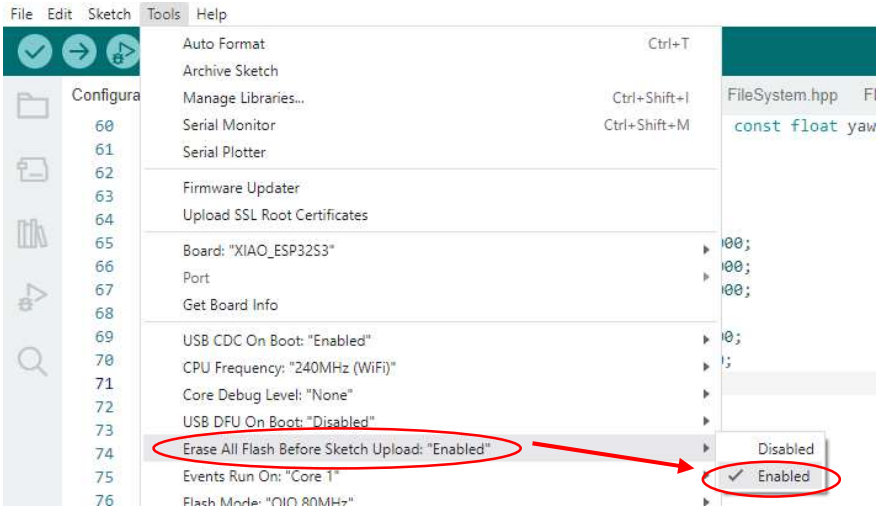
If you are unable to use the web configurator then you can still set all parameters by reprogramming the ESP32-S3. In Configurator.hpp are the default parameters, these can be manually edited and set to what you require:

Figure 36 - Default Parameters

```
//Default values for file system
//Rates
static constexpr  int32_t MAX_PITCH_RATE_DEGS_x100  = 20000;
static constexpr  int32_t MAX_ROLL_RATE_DEGS_x100   = 20000;
static constexpr  int32_t MAX_YAW_RATE_DEGS_x100    = 15000;
//Max angles allowed when in levelled mode (angles * 100)
static constexpr  int32_t MAX_PITCH_ANGLE_DEGS_x100 = 5500;
static constexpr  int32_t MAX_ROLL_ANGLE_DEGS_x100 = 5500;
//Trims
static constexpr  float LEVELLED_ROLL_TRIM        = 0.0f;
static constexpr  float LEVELLED_PITCH_TRIM       = 0.0f;
static constexpr  int32_t SERVO_TRIM              = 0;
//Gains
static constexpr  int32_t PITCH_P_GAIN              = 25;
static constexpr  int32_t PITCH_I_GAIN              = 50;
static constexpr  int32_t PITCH_D_GAIN              = 0;
static constexpr  int32_t PITCH_F_GAIN              = 18;
static constexpr  int32_t ROLL_P_GAIN               = 25;
static constexpr  int32_t ROLL_I_GAIN               = 50;
static constexpr  int32_t ROLL_D_GAIN               = 0;
static constexpr  int32_t ROLL_F_GAIN               = 18;
static constexpr  int32_t YAW_P_GAIN                = 25;
static constexpr  int32_t YAW_I_GAIN                = 50;
static constexpr  int32_t YAW_D_GAIN                = 0;
static constexpr  int32_t YAW_F_GAIN                = 10;
//Battery scaler
static constexpr float BATTERY_SCALER               = 0.00357f;
```

However, for any edits to take effect you will need to erase all flash before programming. This forces the file system to write the default parameters to a new file. In Arduino IDE make sure the following setting is enabled to force full flash erase, set Tools->Erase All Flash Before Sketch Upload->Enabled:

Figure 37 - Force Full Flash Erase



10.3 Rate Mode Tuning

Gyro rate mode is the most challenging to tune as you have three axis (pitch, roll and yaw) and four main parameters to tune per axis (PIDF).

Note: There is no gain attenuation based on throttle position as you will find on other flight controllers. To date I have found that conservative P with higher I-gains resolve this and gives excellent results. Adding D also can significantly improve the performance and reduces ‘bobble’ in flight by dampening P-gain, it also reduces/removes speed oscillations and ultimately allows slightly higher P-gains to be used – I have been amazed by how well D term works on servos, see my YouTube video on this.

Note: You will only need to use feed forward on servos.

To those who believe D-gain has no effect on servos you are partially correct. Servos will not respond to instantaneous pulses of D to speed the servo response, but the D term will prevent P-gain overshoot, softens ends of manoeuvres and disturbances and reduces/removed speed oscillations. This is of course with correct tuning.

Table 23 - Gyro Rate Mode Gains

Gain	Comment
xxx_F_GAIN	Feed forward does most of the work during a manoeuvre and the P and I terms are there to ensure that the degrees per second demand is met or maintained. With rate PID gains set to zero, set F-gain so you have at least 80% servo travel for full stick deflection when compared to pass-through mode. Note: As an example; When flying with PIF gains, perform a roll at maximum stick deflection and stop that roll quickly when level. If you notice bounce back (wings rolled back slightly) then F-gain is a little too high. If you notice or feel that the wings rolled more than you commanded, then feedforward is too low. Note: Multicopters generally do not need to use feed forward on motors.
xxx_P_GAIN	Increasing the P-gain results in faster corrections to disturbances. Too high a value will result in oscillation and possibly an uncontrollable situation – change to pass-through mode if you're ever in this situation. Airspeed will also affect this parameter and possible oscillations, so set this value conservatively for all axis. Note: Typical values that I have used vary between 45 and 100, but it will depend on control surface area, deflection and servo response times.
xxx_I_GAIN	I-gain gives 'stiffness' or 'heading hold' to each axis. Initially set the I gain to the same value as your P. However, do not be concerned to exceed the P-gain value. Note: Typical values that I have used vary between 100 and 300, but it will depend on control surface area, deflection and servo response times. **Note**: For fixed wing models, I-gain on yaw is only used when heading hold and/or prop hang mode are enabled.
xxx_D_GAIN	D-gain is used to stop P-gain overshoot and helps reduce oscillations due to the P term. Get your model flying with PIF initially, if you feel it is on the verge of oscillation, 'bobbles' in flight, or oscillates when at speed then add D-gain to the affected control axis to soften the settling of the control surface. I have found D-gains in the order of 250 to 1000 are effective. See my video on D-gains for fixed wing.

Note: Always set gains conservatively to avoid uncontrolled behaviour on first flights.

10.4 Self-Levelled Mode Tuning

As of PlainFlightController v1.1.0 There are no PIDF parameters to tune for levelled mode. It uses the rates and gains set for rate mode. However, a few parameters relate to its operation as described by 10.1.2.3 Self-Levelled Trim Configuration.

The algorithm implemented gives a very locked in flight feel as it uses the gyro rate controller and provides a snappy return to level if the pitch/roll stick is released quickly.

10.5 Heading Hold (Fixed Wing Only)

When heading hold is enabled you will have to tune yaw I-gain for it.

I-gain gives the 'heading hold' feature to the yaw axis. Typical values that I have used have been between 100 and 300. However, it will depend on control surface area, deflection, and servo response times.

Note: Heading hold has no effect on Multicopters as yaw I-gain is always enabled.

10.6 Model Gain Examples

The following sections give examples from some of my models to give you a rough idea of gains and gain ratios for different model types.

10.6.1 FLYING_WING with USE_DIFFERENTIAL_THROTTLE

~12” span (A4 paper size) twin motor (1102-10000kv) flying wing powered from 1s 650mAh lipo. Large elevon surfaces took a bit of tuning but found that D-gain dampened roll P-gain overshoot. 1.7g digital servos are running at SERVO_REFRESH_300HZ giving quickest possible update rate. Will hover hands off when using heading hold with differential thrust motors. With USE_LOW_VOLTS_CUT_OFF set to 3.4V limit and 3.3V cut off, flight times are around 8 minutes.

Note: Strongly advise use of 470uF 16V capacitors on power input of 1s BLHeli ESC’s to prevent voltage spikes getting to the ESP32-S3 processor – I learnt the hard way.

Table 24 - Example Of Settings Used By A4

Settings	Value
PLANE_FLYING_WING	true
USE_DIFFERENTIAL_THRUST	true
USE_PROP_HANG_MODE	true
PROP_HANG_TAIL_SITTER_MODE	true
USE_LOW_VOLTS_CUT_OFF	true
USE_250_DEGS_SECOND	true
Motor RefreshRate	IS_ONESHOT125
Servo RefreshRate	IS_300Hz
PITCH P	30
PITCH I	50
PITCH D	350
PITCH F	18
ROLL P	25
ROLL I	50
ROLL D	550
ROLL F	15
YAW P	50
YAW I	50
YAW D	550
YAW F	10
ROLL RATE (Degrees/Sec)	230
PITCH RATE (Degrees/Sec)	200
YAW RATE (Degrees/Sec)	150

10.6.2 PLANE_FULL_HOUSE

A 9.5” span all balsa sheet model with ailerons and elevator, powered by a 1102-10000kv brushless motor and 1s 300mAh LiHv. This model uses the Waveshare ESP32-S3 Tiny with built in RGB Neopixel LED. The MPU6050 IMU is a smaller version PCB than previously used and is located under the canopy.
Model has great vertical performance and with plainFlightController on board gives precision aerobatics.

Table 25 – Example Of Settings Used BY Pea 9.5" Aerobat

Settings	Value
PLANE_FULL_HOUSE	true
USE_ONBOARD_NEOPIXEL	true
USE_250_DEGS_SECOND	true
Motor RefreshRate	IS_ONESHOT125
Servo RefreshRate	IS_100Hz
PITCH P	25
PITCH I	100
PITCH D	0
PITCH F	20
ROLL P	25
ROLL I	120
ROLL D	0
ROLL F	18
ROLL RATE (Degrees/Sec)	225
PITCH RATE (Degrees/Sec)	160

10.6.3 PLANE_FULL_HOUSE

This is an old FPV model I build back in 2015, balsa construction results in a stiff and responsive airframe. Has a non-symmetrical aerofoil that would cause it to ‘balloon’ when closing the throttle and when travelling at speed. Total of 10 degrees dihedral gives stability and allows rudder to steer the model. The high mounted motor can cause the nose to pitch down if throttle is applied quickly.

Adding PlainFlightController to this model has resolved any ‘ballooning’, rudder coupling and power pitching issues. This model is now so locked in it feels like a mini pattern ship to fly (it is stupidly overpowered with the Tmotor F40 too). 4-point rolls are fantastic, and with heading hold (and a lot of throttle) it will knife edge hands off.

Initially tuned with PIF gains only, with these settings I found that aileron corrections would cause the wing to ‘bobble’ when disturbed (not oscillate, just observe fast aileron corrections), not a problem and what you would expect for a flight controller. The yaw corrections would do the same for the rudder when travelling at speed. This bobbling was resolved by adding D-gain to both aileron and rudder which smooths any P-gain overshoot around the set point (neutral). This is now an incredibly smooth, locked in model to fly and is an absolute pleasure for aerobatics, which it was not originally designed for.

Four off 4g digital MG BlueArrow servos are running at SERVO_REFRESH_150HZ giving a reasonable response rate, and USING_ONESHOT125 to operate a BLHeli ESC.

Table 26 - Example Of Settings Used By OldFPV

Settings	Value
PLANE_FULL_HOUSE	true
USE_LOW_VOLTS_CUT_OFF	true
USE_FLAPS	true
USE_HEADING_HOLD	true
USE_250_DEGS_SECOND	true
Motor RefreshRate	IS_ONESHOT125
Servo RefreshRate	IS_150Hz
PITCH P	65
PITCH I	175
PITCH D	0
PITCH F	18
ROLL P	35
ROLL I	150
ROLL D	750
ROLL F	15
YAW P	40
YAW I	200
YAW D	600
YAW F	45
ROLL RATE (Degrees/Sec)	230
PITCH RATE (Degrees/Sec)	120
YAW RATE (Degrees/Sec)	100

10.6.4 PLANE_FULL_HOUSE

My original test bed during development of PlainFlightController was this donated and tatty Horizon Hobby Super Cub.

Now fitted with an AXI 2808 and some hastily added ailerons, it has made this into a nice slow fun-fly model. Nothing is straight or square on it and the wing flexes terribly when pulling manoeuvres. However, despite these problems its now a nice model that flies straight and true. Its party piece being hands off knife edge the length of our field when using heading hold, and very low slow knife edge (no heading hold). Flying this in self-levelled mode using only throttle and rudder controls to perform touch and goes has also been quite good fun. With heading hold this model will hold itself in prop-hang hands off, though lack of airflow over ailerons does mean that it drifts off after several seconds.

The PI gains are quite high on this model for 2 reasons, the original servos are quite slow to respond (even though SERVO_REFRESH_150HZ is set) and there is so much flex in the airframe that control corrections are absorbed before the model responds. To date I have not found D-gain to give any beneficial effect on this model and believe this is because of the flex in the model.

Settings	Value
PLANE_FULL_HOUSE	true
USE_FLAPS	true
USE_HEADING_HOLD	true
USE_PROP_HANG_MODE	true
USE_250_DEGS_SECOND	true
Motor RefreshRate	IS_150Hz
Servo RefreshRate	IS_150Hz
PITCH P	100
PITCH I	300
PITCH D	0
PITCH F	18
ROLL P	75
ROLL I	150
ROLL D	0
ROLL F	25
YAW P	80
YAW I	300
YAW D	0
YAW F	45
ROLL RATE (Degrees/Sec)	180
PITCH RATE (Degrees/Sec)	180
YAW RATE (Degrees/Sec)	100

10.6.5 PLANE_ADVANCED_RUDDER_ELEVATOR

Pip is a simple 16” span rudder elevator model. Its non-symmetrical aerofoil makes it quite buoyant and wants to climb when throttle is quickly reduced (effective down thrust removed). Its light weight also makes it susceptible to weather conditions.

Adding PlainFlightController instantly removes any pitch issues related to speed and down thrust and also makes it fly rock solid in windy conditions.

Being rudder elevator, it was found that the wings would rock a little due to corrections in windy conditions. This just looked odd and poorly tuned. As a result, PLANE_ADVANCED_RUDDER_ELEVATOR was implemented to allow separate PIDF controls for roll and pitch to be applied to rudder. In effect one cancels the other out and gives a more stable flight. However, it is a little harder to tune as result. PLANE_ADVANCED_RUDDER_ELEVATOR also has benefits for stall turns as yaw PIDF controller is directly controlling yaw rather than roll PIDF controlling yaw which would lead to roll I-gain build up and a roll response when exiting the stall turn.

Settings	Value
PLANE_ADVANCED_RUDDER_ELEVATOR	true
USE_LOW_VOLTS_CUT_OFF	true
USE_HEADING_HOLD	true
USE_250_DEGS_SECOND	true
Motor RefreshRate	IS_ONESHOT125
Servo RefreshRate	IS_150Hz
PITCH P	65
PITCH I	175
PITCH D	650
PITCH F	18
ROLL P	35
ROLL I	150
ROLL D	750
ROLL F	15
YAW P	40
YAW I	150
YAW D	600
YAW F	45
ROLL RATE (Degrees/Sec)	220
PITCH RATE (Degrees/Sec)	180
YAW RATE (Degrees/Sec)	100

10.6.6 QUAD_X_COPTER

OK don't expect any PlainFlightController multicopters to fly as good as BetaFlight. However, be surprised at how well your home built quad frame can fly, even aerobatics.

Consider the gains given to be a basic tune for a low powered quad it can certainly be improved. However, this model happily loops, rolls and stall turns with the settings given.

D-gain was crucial to getting any sense out of making it fly nicely. Add I-gain until each axis feel locked in, be slightly sparing with P-gain. Add D-gain to overcome any 'bobbles'.

Settings	Value
QUAD_X_COPTER	true
USE_LOW_VOLTS_CUT_OFF	true
USE_HEADING_HOLD	true
USE_250_DEGS_SECOND	false
Motor RefreshRate	IS_ONESHOT125
PITCH P	20
PITCH I	150
PITCH D	950
PITCH F	0
ROLL P	20
ROLL I	150
ROLL D	950
ROLL F	0
YAW P	45
YAW I	150
YAW D	250
YAW F	0
ROLL RATE (Degrees/Sec)	360
PITCH RATE (Degrees/Sec)	360
YAW RATE (Degrees/Sec)	200

11. Pre-Flight Checks

Strongly recommend that the following pre-flight check are carried out as a minimum. Not doing so may result in the loss of your model. Perform any additional checks you feel are necessary or wise to ensure safety and reliability of your model.

Note: It is assumed that you have experience in using and tuning flight controllers and understand PIDF control systems. If you are not confident seek advice or training.

Caution	Check	Comments
	Ensure servos are centred and model control surfaces are mechanically trimmed to neutral positions.	Set servo horns to correct spline location. Use TRIM_SERVOx to centre servos before flying.
	Ensure flight controller is firmly attached to model and in correct orientation.	Attach with double sided foam take or Velcro. Make sure no lose items can rub or strike the flight controller when in flight. Make sure all wired connections are secure.
	Ensure pitch, roll, and yaw gyro corrections operate in the correct sense when in rate mode.	Use REVERSE_xxx_GYRO to reverse pitch, roll, yaw corrections for rate mode.
	Ensure pitch and roll corrections operate in the correct sense when in self-levelled mode.	Use REVERSE_xxx_IMU to reverse pitch and/or roll corrections for self-levelled mode.
	Ensure pitch, roll, and yaw operate in the correct sense to Tx commands.	If gyro and self-levelled corrections work correctly but Tx commands are the wrong sense, then reverse direction of affected channel(s) in your transmitter software.
	Ensure power supply is sufficient for increased current demands of servos.	Move model around rapidly in rate mode for 10 seconds to ensure power supply does not brown out. With a helper; repeat rapid movements whilst pulsing throttle between low and full to ensure motor load does not affect power supply.
	Set rate mode gains to sensible values.	Default gains given may not be appropriate for you model. Set gains so you can just see corrections occurring for first flight.
	For first flight, fly model in pass through mode until you are confident.	For first tuning flights only change out of pass-through mode when you have sufficient height to try and recover the model in case gains are set too high tuned.

12. Disclaimer & License

Do not expect this software to be comparable to or outperform other more established flight controller projects such as ArduPilot, inav, betaFlight etc. This code shall also be considered as highly experimental and is not designed or written to any safety critical, or mission critical standards.

It is given/shared for free with the knowledge and understanding that this open-source flight controller software is only for small hobby based electrically powered model aircraft, or other small hobby radio-controlled vehicles. It is intended to be used or modified to suit your needs for small models and is NOT to be used on any manned vehicles.

The author(s) shall not be held responsible or accountable for any damage, injury or loss that may be inflicted or incurred as a result of the use or miss use of this software. You use and/or modify this software entirely at your own risk and it must be used within accordance of your country's laws and/or regulations.