



General Build Guide

Synerduino Ardu 2560

Fixwing

Latest Version - 2024

For more Information:
www.synerflight.com



TABLE OF CONTENTS

1 INTRODUCTION

- Synerduino Ardu 2560.....#
- Synerduino Kit Components.....#

2 ASSEMBLY

- Tools and Materials.....
- Synerduino Shield Preparation.....
- Arduino Board Preparation.....
- Battery, Motor, ESC, and Propeller Installation.....
- GPS and Bluetooth Configuration.....

3 SOFTWARE SETUP

- ESC Calibration.....#
- Arduino Firmware Config.....#
- FlyWiiGUI Groundstation Installation.....

4 TUNING

- Tuning and Parameter Adjustments.....#

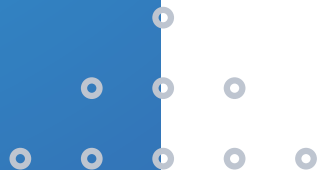
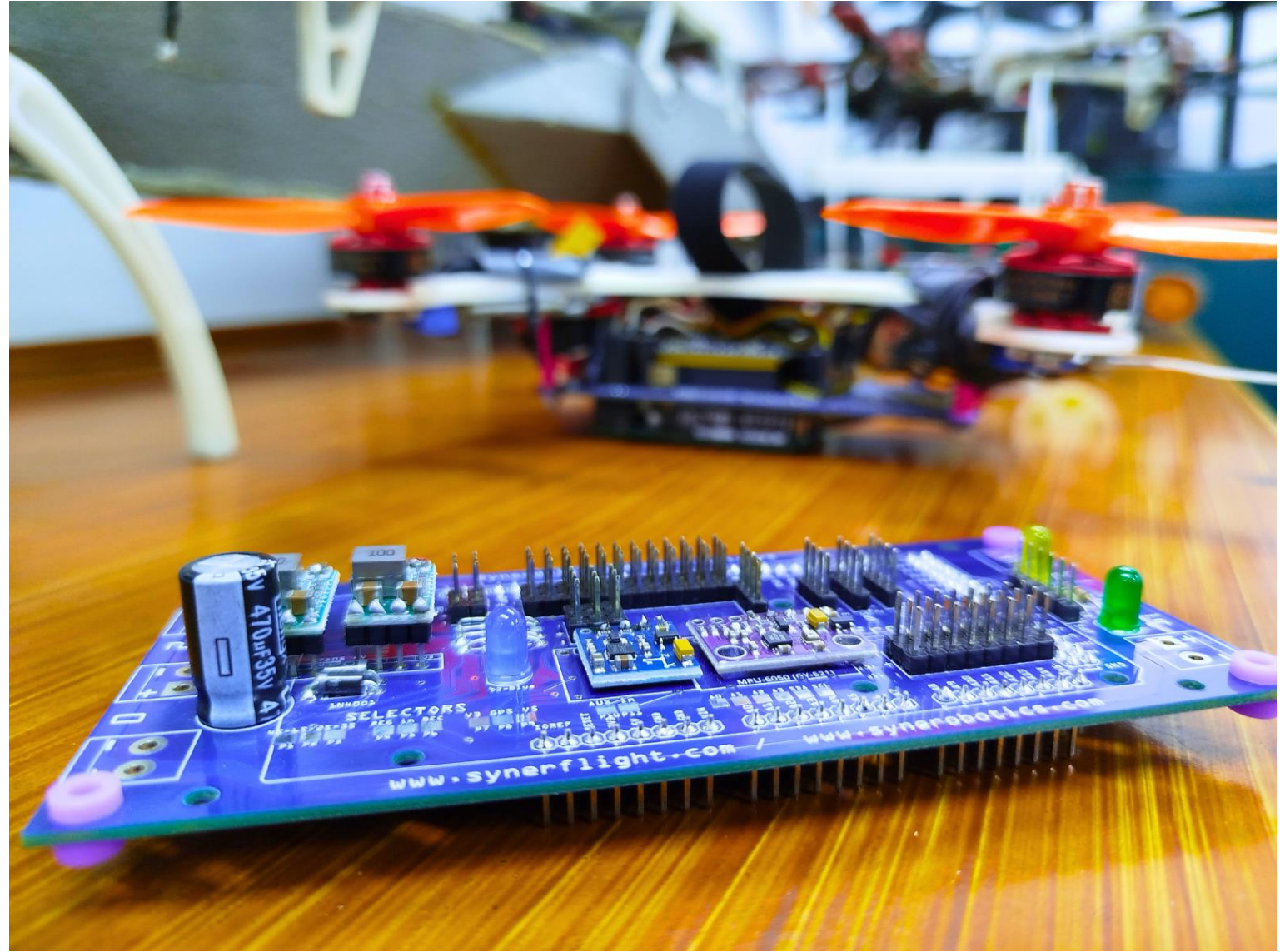
5 PRE-FLIGHT

- Missions.....#
- Safety Precautions and Recommendations.....#



INTRODUCTION

Synerduino Ardu 2560 brings back the classic feel of Arduino, reminiscent of the 2012-2014 era. With its compatibility with the iconic 2560 board and the Arduino IDE, configuring your projects through Sketch .ino files has never been more straightforward. Built on top of the widely used Arduino boards, Synerduino Ardu 2560 is designed with the academic environment in mind, making it a perfect choice for educational and hobbyist projects alike.



SYNERDUINO ARDU 2560

ABOUT THE BOARD



Power

- Input Voltage from Arduino Board: 3.3-5V
- PWM Power Rail Regulated – 5V at 1.5A
- Drone Power Input Voltage – 12.6V (3S) or 25.2V (6S)
- Power Distribution Lines – 80A

Properties

- Dimensions: 128 x 62 x 28 mm LWH / (V1.1)135mm x 62mm x 28mm
- Weight: 46.1g
- 4 Solder Pads for 4 ESCs and Motors
- 15 3-Pin Digital Headers
- 8 3-Pin Analog Headers
- 5 4-Pin Serial Headers

Sensors

- Gyroscope + Accelerometer: MPU6050
- Magnetometer: QMC5883 / HMC5883
- Barometer: BMP280

BOARD VERSIONS

Synerduino Arduino Shield V2 -2024

- MPU9250
- QMC5883
- BMP 280



Synerduino Kwad Shield V1 -2021

- MPU9250
- MAG 9250
- BMP 180



Synerduino Kwad Shield Beta -2020

- L3G4200D
- ADXL345
- BMP 180
- MMC5883



PIN LAYOUT

Aux ADC in

Description: Auxiliary input for connecting additional sensors or components that output analog signals, allowing the board to read and process external analog data.

Note: Input Voltage: 3.3-5V

Power Input

Description: This is the main power input for the board, designed for a 3-cell (3S) - 4-cell (4S) LiPo battery - 11.1V and 14.8V respectively. It powers the ESCs, servos, and other components on the board.

Soldering Note: ESCs should only be soldered on the top side of the board, ensuring the solder joints do not penetrate through to the bottom.

Jumper Pads Selector Zone

Description: These are PWM (Pulse Width Modulation) output pins used to control the motors through ESCs (Electronic Speed Controllers) or servos.

ESC / Servo PWM Out

Description: These are 36 PWM (Pulse Width Modulation) output pins used to control the motors through ESCs (Electronic Speed Controllers) or servos.

Serial Pins

Description: These are 12 serial pins for communication with external devices or modules like GPS or telemetry systems using a UART interface.

GPS Serial Pins

Description: These are 6 dedicated pins for connecting a GPS module's TX and RX (Transmit and Receive) lines for serial communication.

GPS LED

Description: This LED blinks or stays lit depending on whether the GPS is locked (has found satellites) or is searching for a signal.

Status LED

Description: A general-purpose status indicator for the board. It could be used to indicate power, initialization, or operational status.

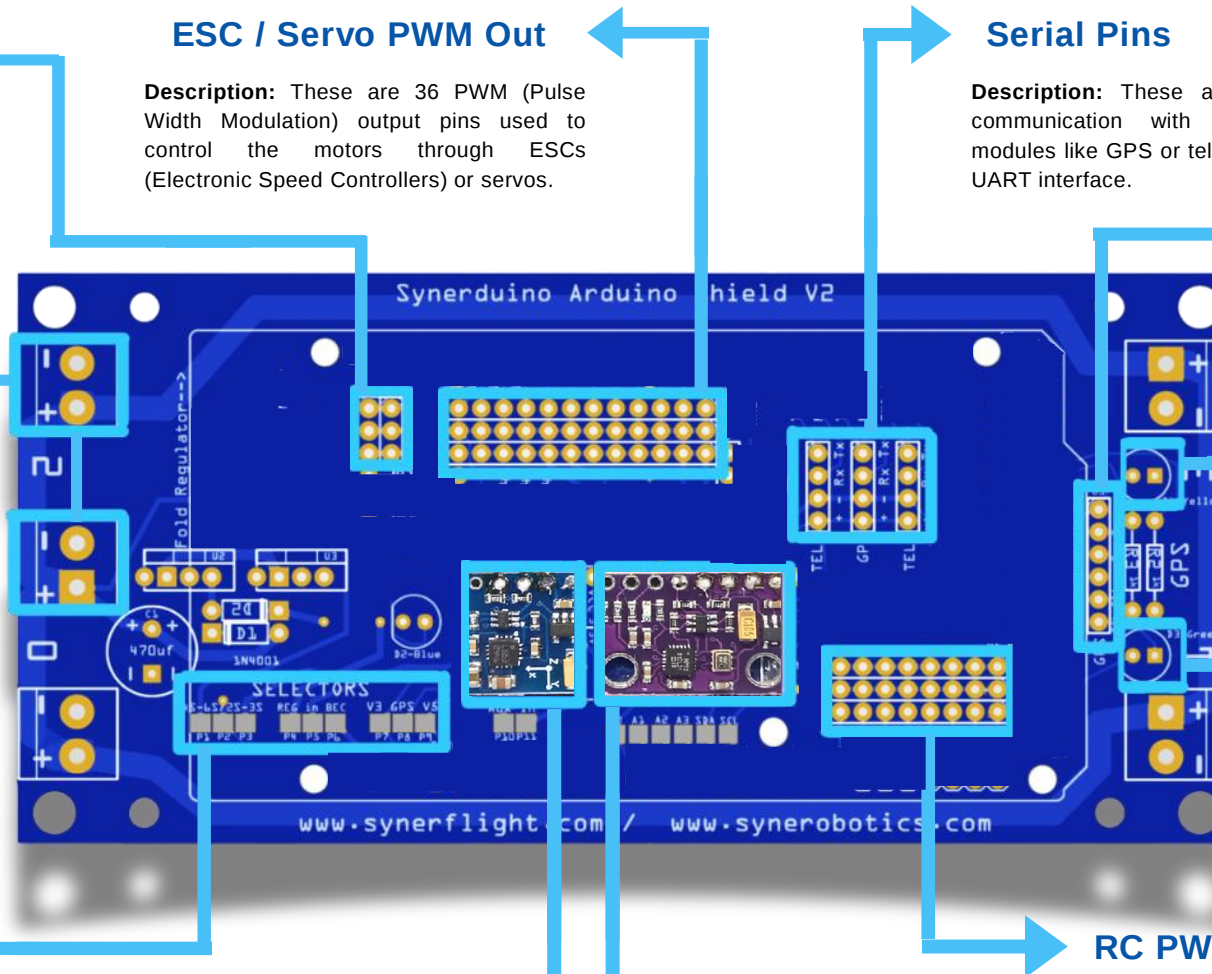
RC PWM In

Description: These are 24 pins which accept PWM signals from an RC (radio control) receiver, allowing manual control via an RC transmitter.

MAG 5883

Note: These modules may vary depending on the manufacturer or version. Some versions might use different sensor combinations that could exclude certain components, like the magnetometer.

IMU MPU-9250



KEY FEATURES OF THE BOARD

Power Terminals (+ and -)

These terminals supply the voltage needed for the entire drone system, ensuring that the right amount of power is distributed across the board and connected components.

Capacitor

Ensures that power distributed to pins and components remains stable, reducing the risk of erratic behavior during flight due to power fluctuations.

GPIO and I/O Headers

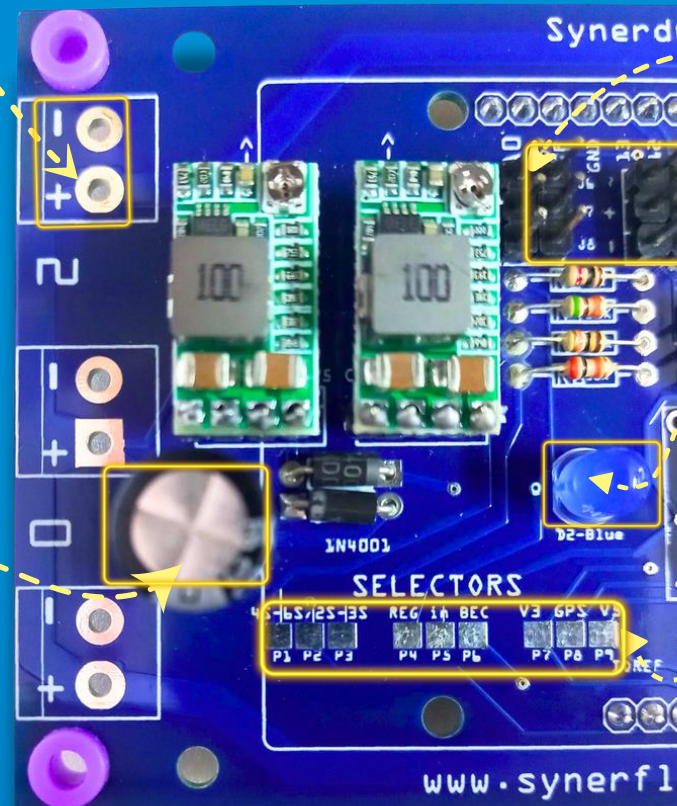
Each pin corresponds to a specific function, such as reading throttle, aileron, elevator, and rudder signals from the receiver, processing sensor data, or controlling motors, making them critical for the drone's operation.

LED Indicator

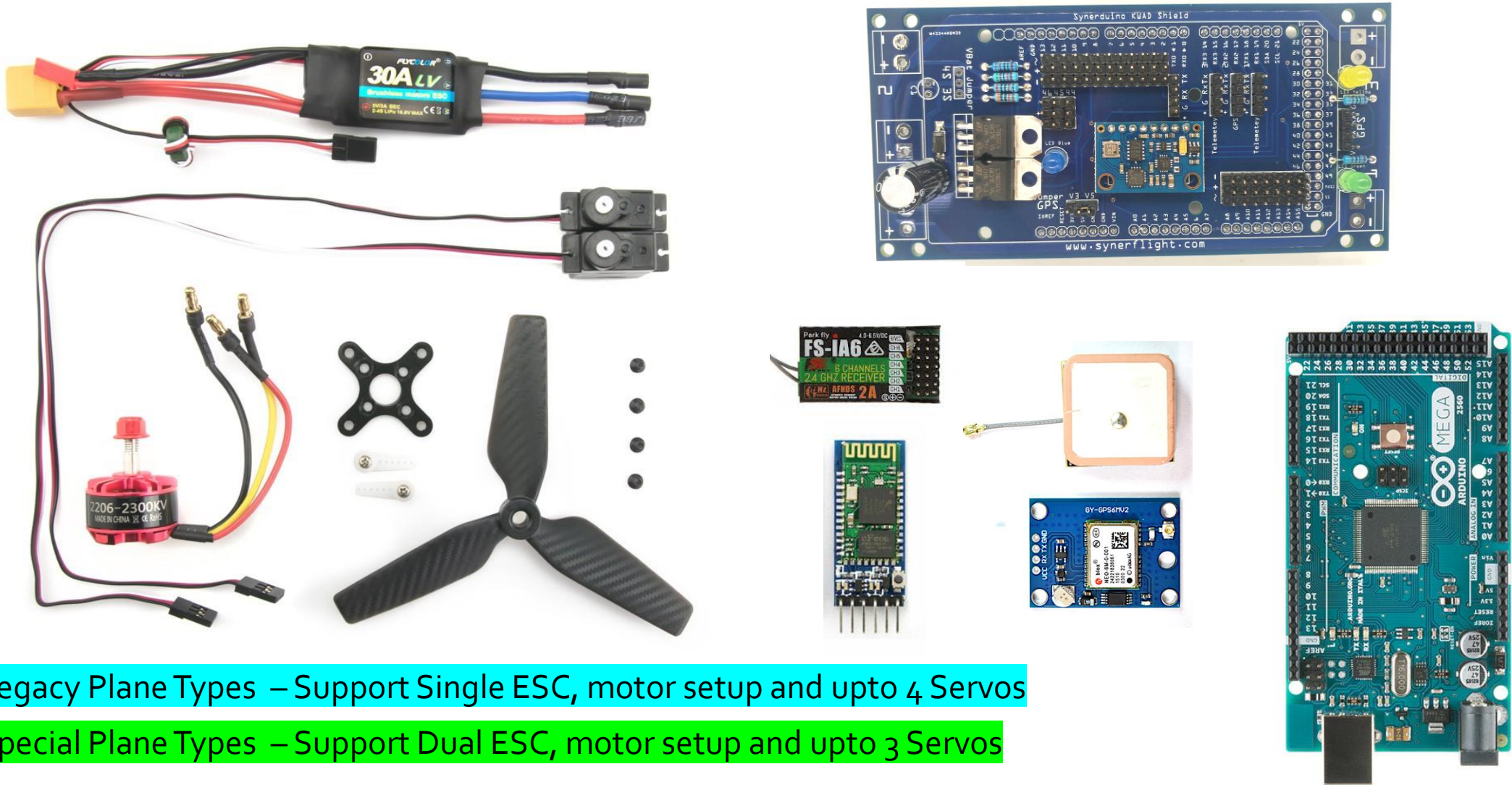
Provides a quick visual confirmation that the board and connected pins are functioning correctly, which is particularly useful during pre-flight checks.

Selector Pins

Selector pins customize the board's power management, allowing you to configure ESCs, GPS modules, and sensors according to flight needs.



SYNERDUINO KIT COMPONENTS



Legacy Plane Types – Support Single ESC, motor setup and upto 4 Servos

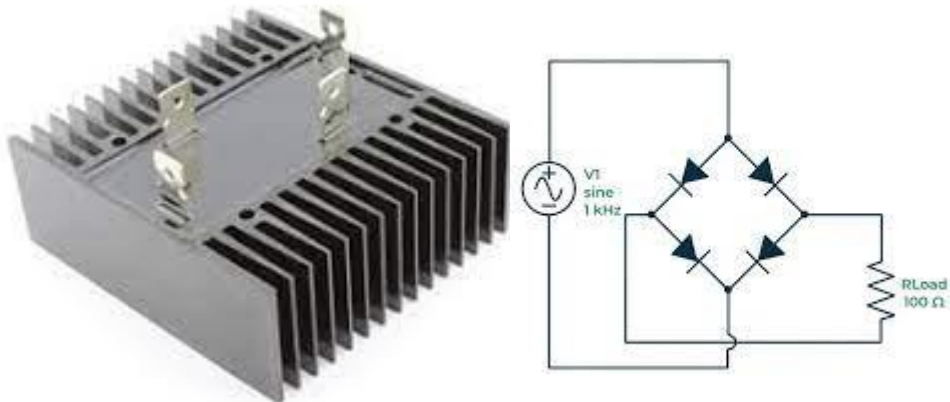
Special Plane Types – Support Dual ESC, motor setup and upto 3 Servos

HIGH POWER HARDWARE

For those running additional sensors, servos and other 5V or 6V components Please use an standalone BEC to power extra hardware



Rectifiers are useful as reverse polarity protection at source. Can be place before the ESC, Servo, Synerduino board rating must be higher than the combine current off all the electronics and motor current draw



For ESCs that have 6V – 8V BEC, to prevent damage to Synerduino PWM Power Rail its recommended that the PWM Power wire is disconnected



High power servos that required 6V-12V

Its required you use an External BEC or Power supply to power Large servos



Disconnect the Red PWM servo wire to



this can be redirected to the power input rail or high power servo input

SYNERDUINO KIT COMPONENTS

Hardware

- Synerduino shield
- Arduino2560 MEGA / Uno 328
- Fixwing (Airplane) Synerduino Plane Airframe
- (1x Legacy) (2x Special) Brushless Motor 1500kv - 2300kv
- (1x Legacy) (2x Special) ESCs 30A 2s-4s
- (1x Legacy) (2x Special) Props 5045 or 8045 depending on setup
- (4x Legacy) (3x Special) 9g Micro Servos and control Rod & horn set
- GPS
- Bluetooth

SYNERDUINO KIT COMPONENTS



**WHAT'S
NEW?**

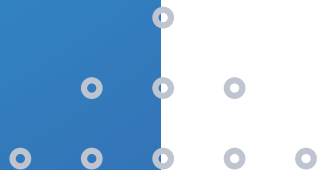
Features:

- Compatible with MultiWii (open source RC multi rotor flying platform)
- Compatible with Arduino Mega 2560 and Uno
- Ground Station with Flywii GUI or Synerflight App
- IMU 10DOF
- Supports 3S/4S Batteries
- 4 Output ESC Pads
- Mode Selection Pads (V1.1)
- ADC sensor input (V1.1)
- Highly customizable

Technical Specifications:

- Physical Dimensions: 128 x 62 x 28 mm LWH
- Weight: 46.1g
- 4 Solder Pads for 4 ESCs and Motors
- 15 3-Pin Digital Headers
- 8 3-Pin Analog Headers
- 5 4-Pin Serial Headers

ASSEMBLY



TOOLS AND MATERIALS



PLIERS

Used for gripping, bending, and cutting wires or components during the assembly process.



TAPES

Electrical and double-sided tapes used for securing wires and insulating electrical connections.



SOLDERING SET

Essential for soldering components and making secure connections between wires and circuit boards.



HEX DRIVER SET

Utilized for tightening or loosening hex screws commonly found in drone frames and components.



CUTTER

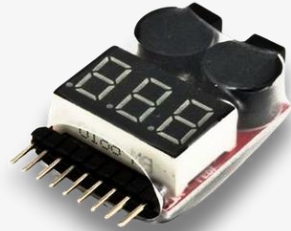
Handy for cutting zip ties, wires, or other materials to the desired length during assembly.



ZIP TIES

Used for bundling and securing wires, ensuring neat and organized cabling inside the drone.

TOOLS AND MATERIALS



BATTERY ALARM CHECKER

Monitors battery voltage, providing warnings when the battery is low to prevent damage or crashes.



LI-PO BATTERY CHARGER

Safely recharges Li-Po batteries, ensuring optimal battery health and longevity.



PVC GLUE

Used for assembling or reinforcing non-electrical parts of the drone frame and components.

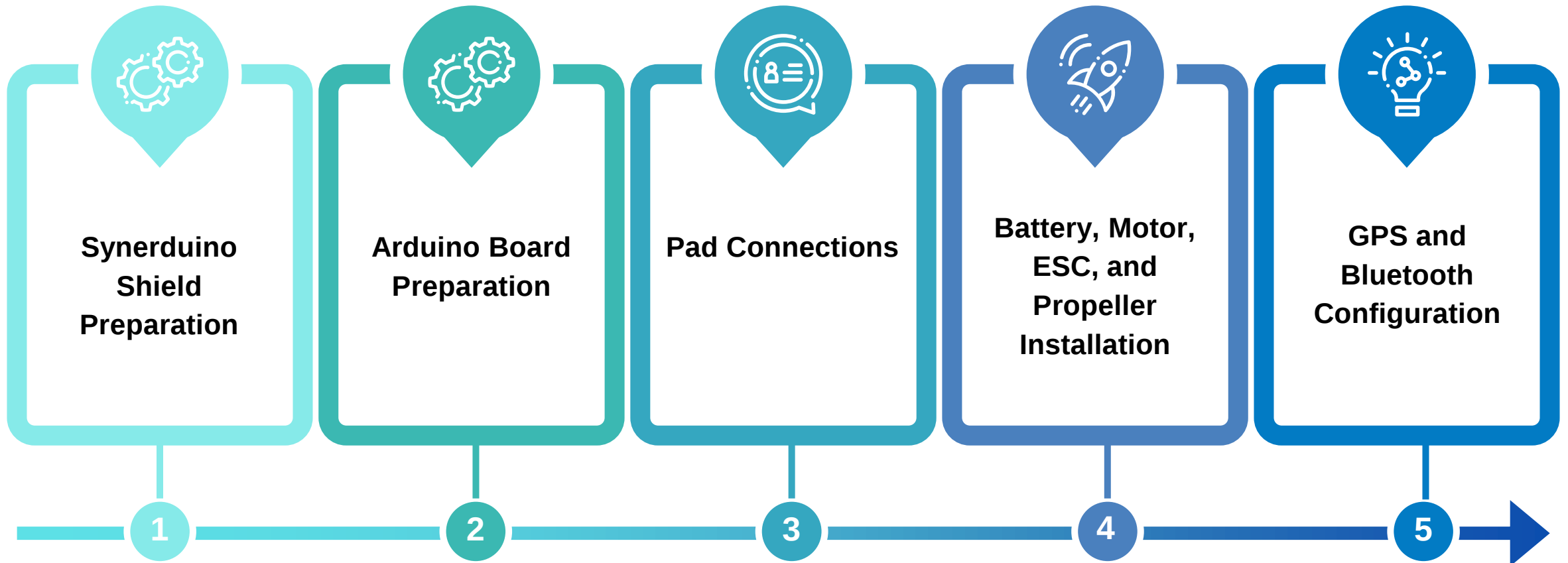


THREAD LOCKER PURPLE

Secures screws and fasteners in place, preventing them from loosening due to vibration during flight.

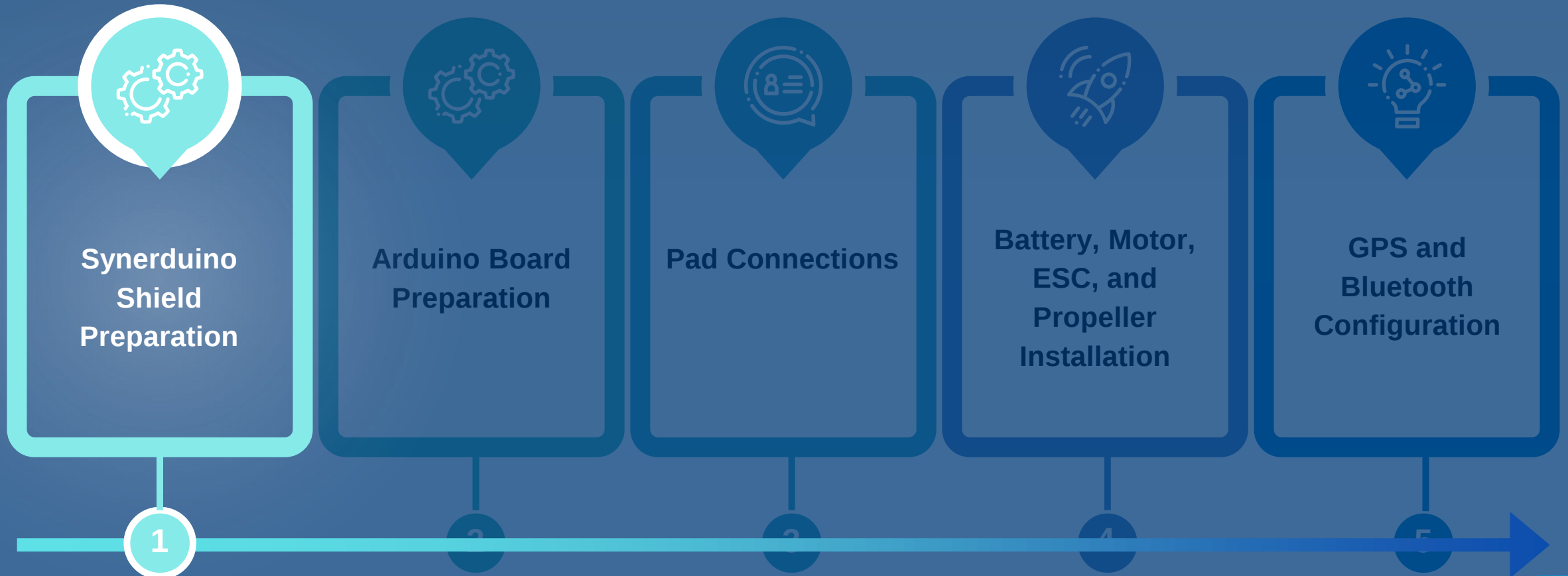
ASSEMBLING PROCESS

This section outlines the essential steps for assembling your Synerduino Drone Kit. Begin by gathering the necessary tools and materials, then prepare the Synerduino shield and the Arduino board. Finally, install the motor, Electronic Speed Controller (ESC), and propeller. Follow these steps carefully to ensure a successful assembly and get your drone ready for flight!



ASSEMBLING PROCESS

This section outlines the essential steps for assembling your Synerduino Drone Kit. Follow these steps carefully to ensure a successful assembly and get your drone ready for flight!



SYNERDUINO BOARD PREPARATION

Aux ADC in

Usage: These pins are used to connect external sensors or devices that output analog signals, such as temperature, pressure, or current sensors.

ESC / Servo PWM Out

Usage: ESCs should be connected to these pins for motor control in a drone or robot.

Serial Pins

Usage: You can use these for connecting devices that need to send/receive serial data.

Power Input

Usage: Connect a 3S LiPo battery to this input. Make sure to match the polarity of the battery with the input terminals to avoid damaging the board.

Soldering Note: ESCs should only be soldered on the top side of the board, ensuring the solder joints do not penetrate through to the bottom.

GPS Serial Pins

Usage: You will connect the GPS module's TX to the RX pin on the board and vice versa, ensuring the correct serial data exchange between the GPS and the microcontroller.

GPS LED

Usage: When the GPS module is properly connected, this LED will inform you of its status. A blinking or steady light usually indicates good GPS reception.

Status LED

Usage: The microcontroller on the board controls this LED, which will light up to signal that the board is functioning correctly.

Jumper Pads Selector Zone

Usage: By soldering the appropriate jumper pads, you can select between different power sources or set the board for the correct number of battery cells.

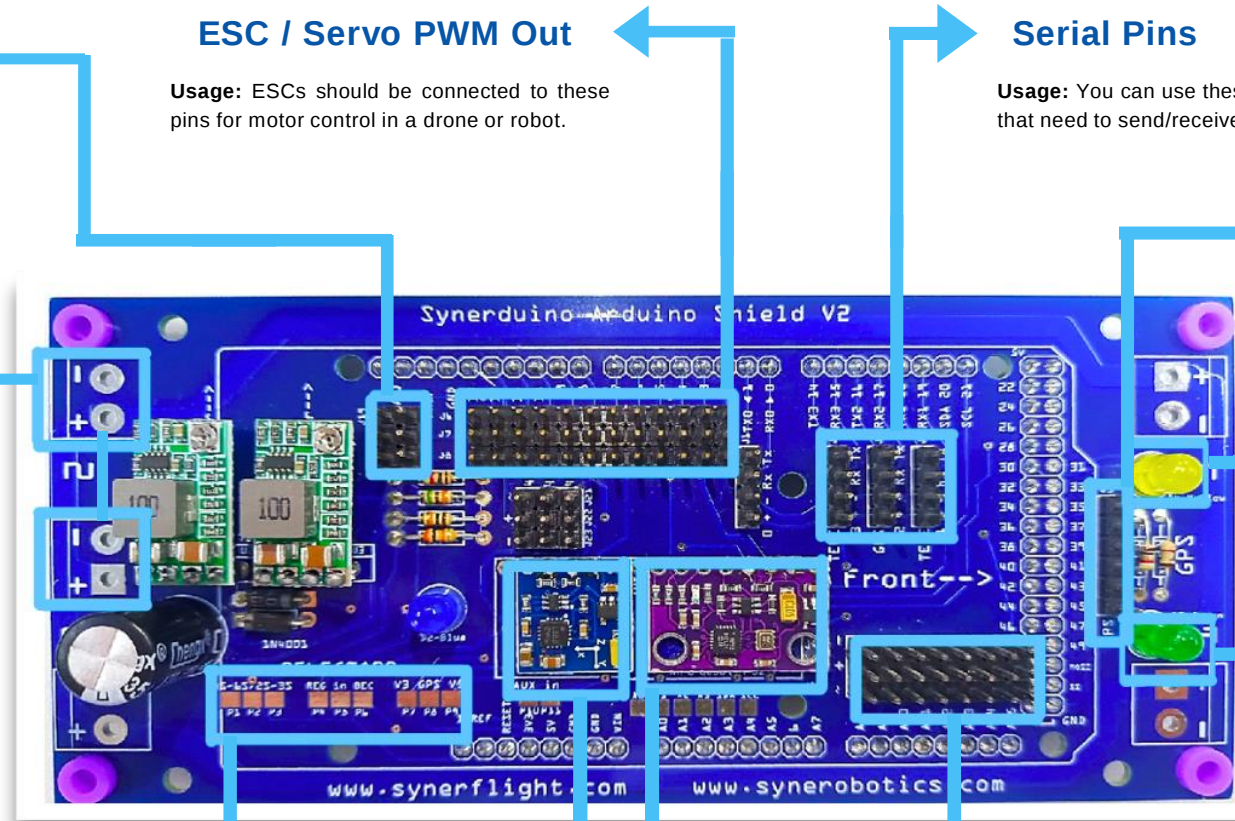
MAG 5883

IMU : MPU-9250

RC PWM In

Usage: Connect the output pins from your RC receiver to these PWM input pins. The signals will then be processed by the microcontroller, which can send commands to the motors through the ESC.

IMU : MPU-9250 Mag QMC5883/QMC5883P & BMP280



Synerduino Kwad Shield **BETA GY801**

Note : surface mount your solder ESC wire make sure it doesn't penetrate to the bottom of the board

ESC is
Solder on
Top side
only

ESC is
Solder on
Top side
only

ESC / Servo PWM Out

Serial Pins

Set jumper to the
Battery Cell
count (Soldered) 3S
- 4S

Power
input
3S 11.1V
4S 14.8V

GPS LED

GPS Serial Pins

Status LED

ESC is
Solder on
Top side
only

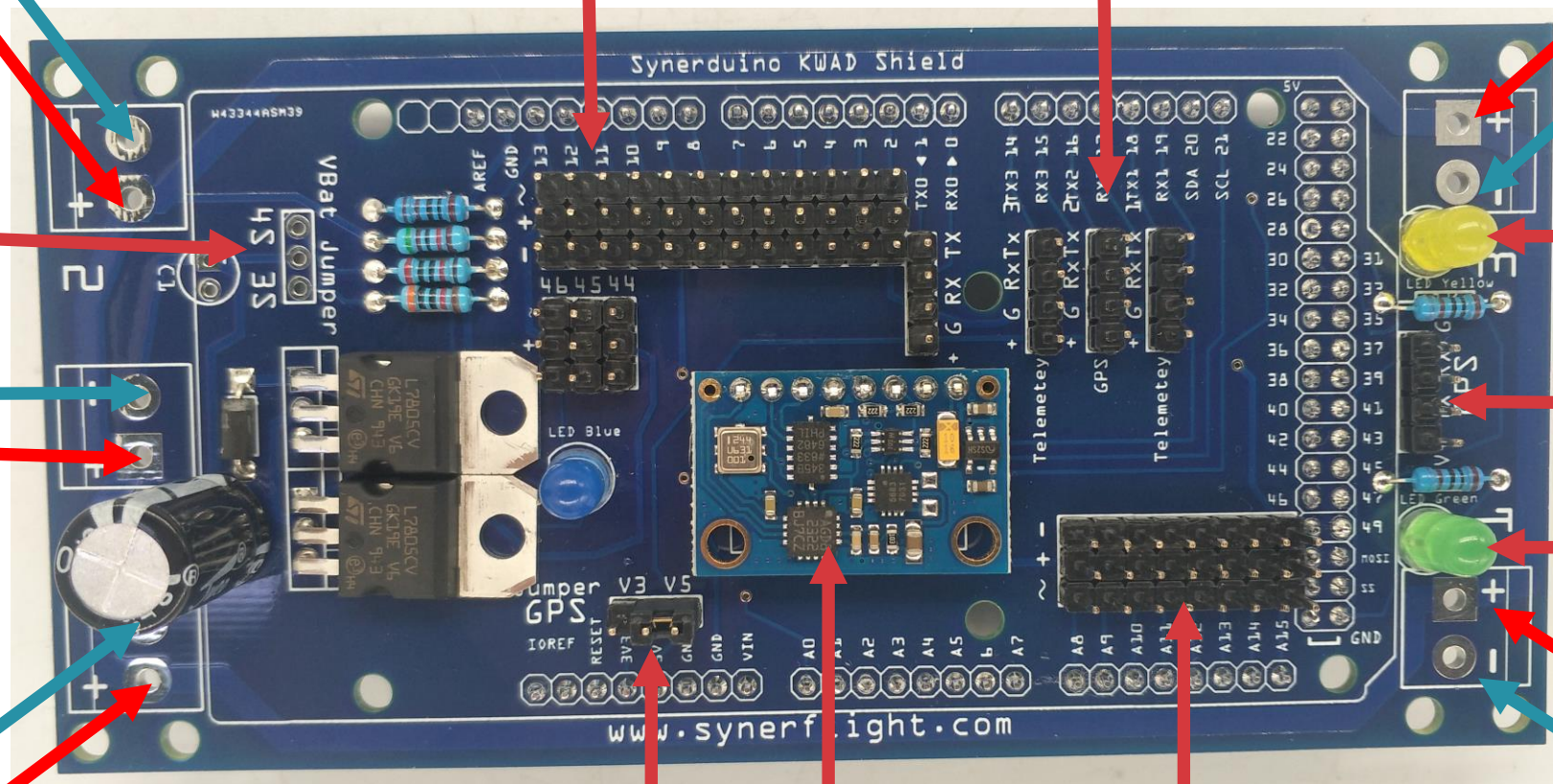
ESC is
Solder on
Top side
only

GPS Voltage Jumper

RC PWM in

For improve performance IMU must be protected from the Environment

IMU : L3G4200D Gyro / ADXL345 Accelerometer / BMP180 - 85 Baro / MMC5883 Mag



Synerduino Kwad Shield V1.1 GY91

ESC is
Solder on
Top side
only

Note : surface mount your solder ESC wire make sure it doesn't penetrate to the bottom of the board

Aux ADC in

ESC / Servo PWM Out

Serial Pins

ESC is
Solder on
Top side
only

Power
input
3S 11.1V

GPS LED

GPS Serial Pins

Status LED

ESC is
Solder on
Top side
only

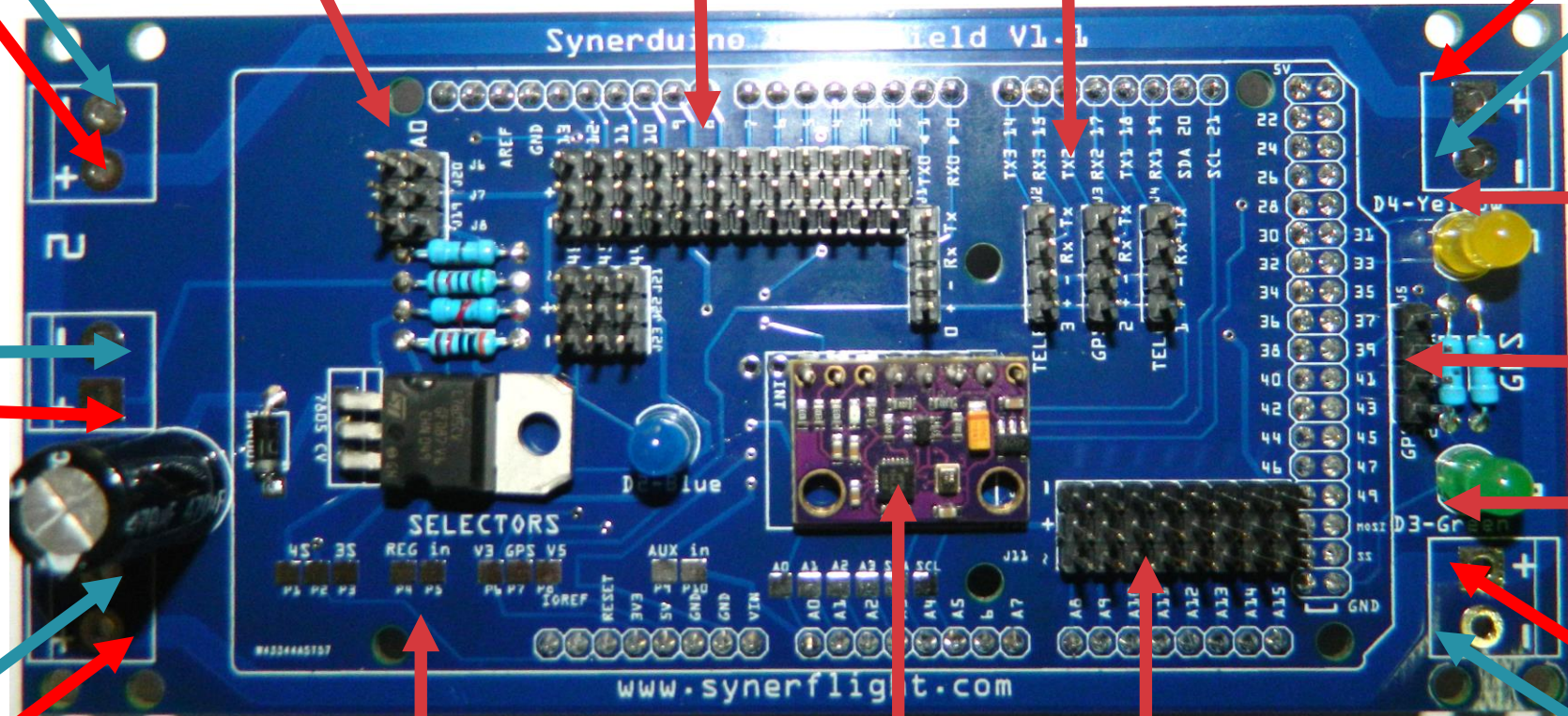
Jumper Pads Selector Zone

RC PWM in

ESC is
Solder on
Top side
only

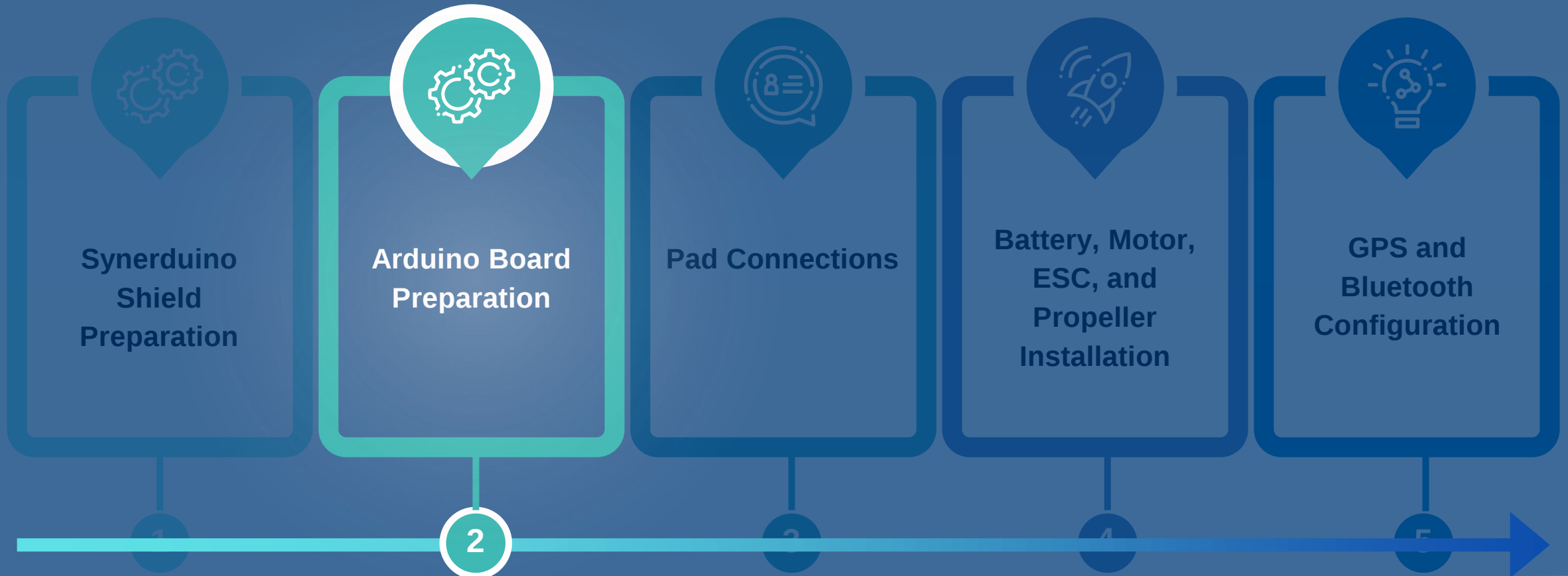
For improve performance IMU must be protected from the Environment

IMU : MPU-9250 Mag HMC/QMC5883 & BMP280



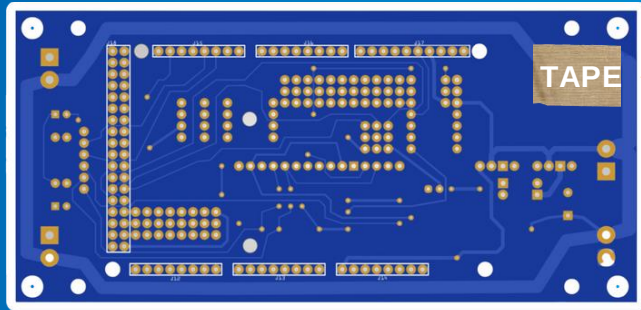
ASSEMBLING PROCESS

This section outlines the essential steps for assembling your Synerduino Drone Kit. Follow these steps carefully to ensure a successful assembly and get your drone ready for flight!



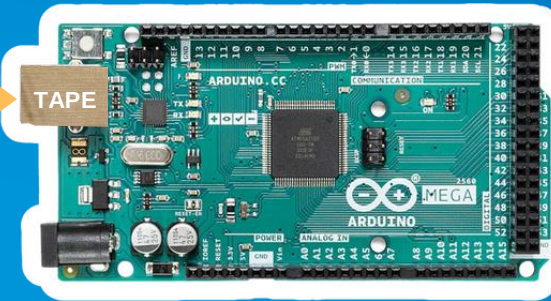
BOARD PREPARATION

Step 1: Add tape to these areas to ensure insulation from the Arduino board.



Add tape to the top right side corner at the back of the Synerduino board.

2560 MEGA



UNO 328



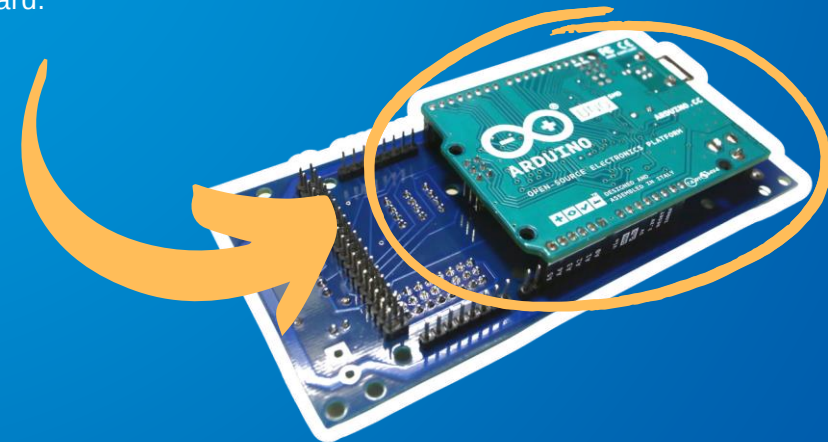
Add tape to the top-left side of the Arduino 2560 MEGA/UNO 328 board to cover the metal part.

Note: The exposed metal areas may come into contact with the Synerduino kit components, potentially causing a short circuit.

Step 2: Make sure to seal the cover onto the sensor using PVA glue, and allow it to fully dry before proceeding.

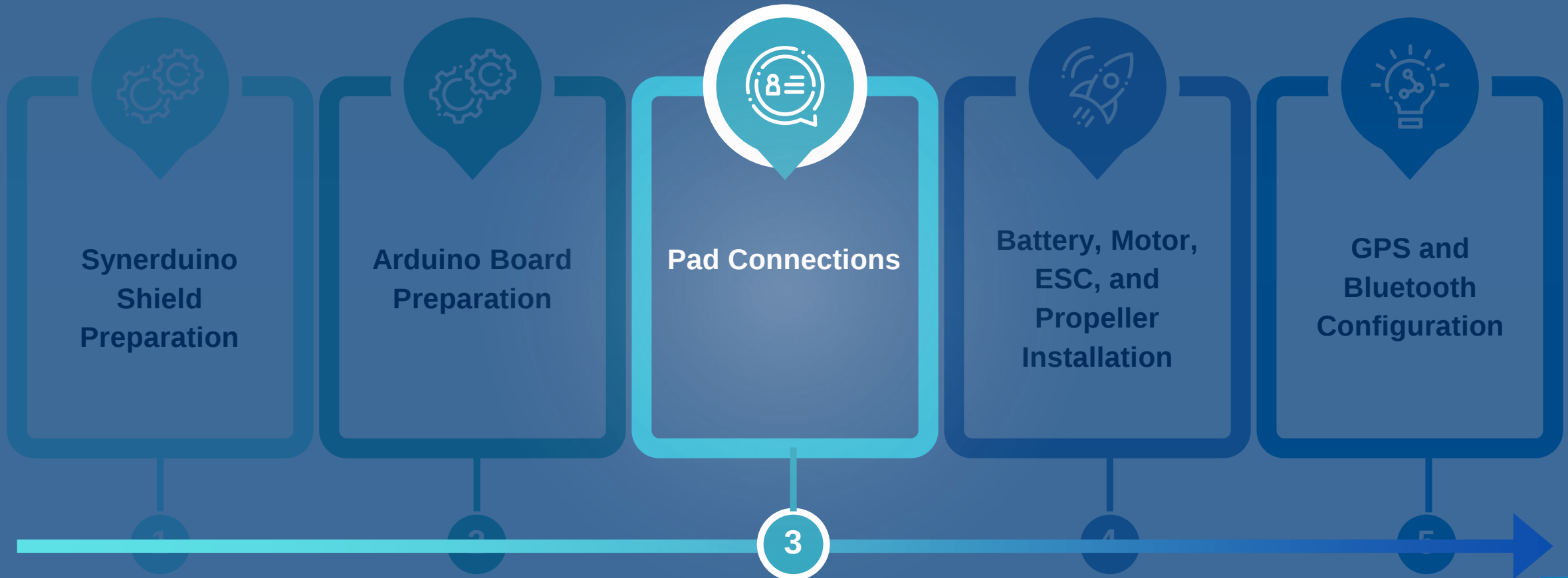


Step 3: Now, connect the Arduino Uno Shield to the back of the Synerduino board.



ASSEMBLING PROCESS

This section outlines the essential steps for assembling your Synerduino Drone Kit. Follow these steps carefully to ensure a successful assembly and get your drone ready for flight!



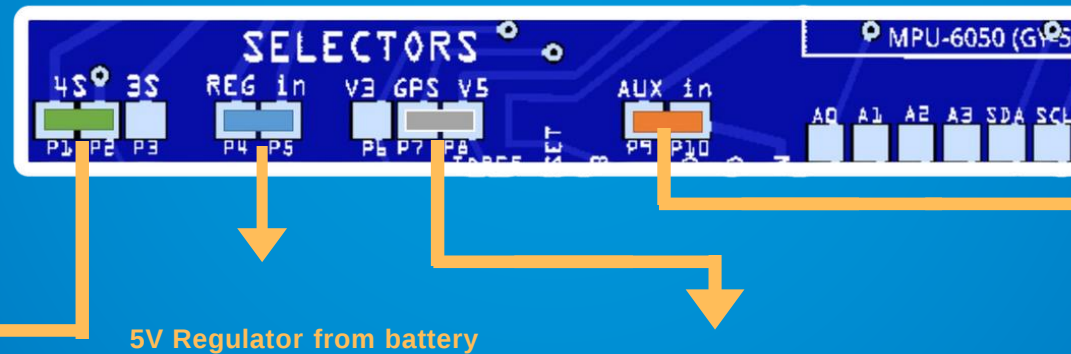
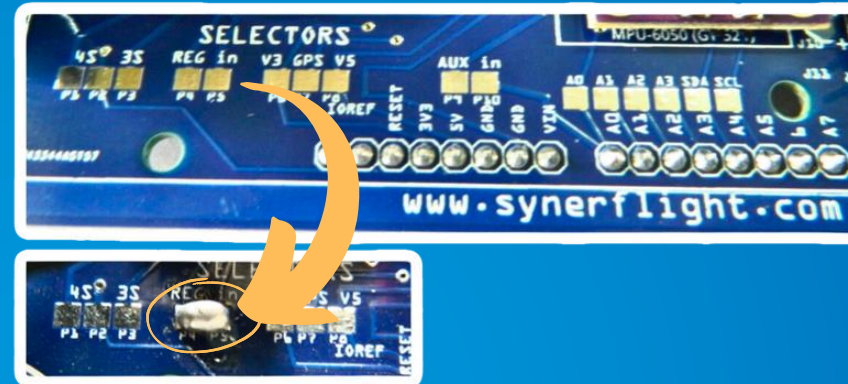
PAD CONNECTIONS

SYNERDUINO KWAD SHIELD V1 BOARD

Step 4: Power Selector Jumper Pads are directly added to the main board, enabling users to choose the desired power source through a simple soldering step:

Apply a small blob of solder to bridge the specific pads corresponding to your preferred power option.

This approach provides a secure connection, giving you the flexibility to configure power supply without additional components, all in a compact and reliable manner.



Battery cell monitoring 4s or 3s

To use the onboard battery monitoring with Aux In:

- Set to 3S if you're using a 1S-3S battery.
- Set to 4S if you're using a 4S battery.
- Leave it open when using Aux In as external sensors or when using 5S-6S batteries.

5V Regulator from battery

- For Reg In, short the pads to use the regulator to power both the Synerduino and Arduino board, along with the built-in power distributor.

GPS Pins V+ voltage in front of the board

The 2nd GPS pin comes with a voltage selector:

- Set to 5V for a regular GPS.
- Set to 3V for an external I2C sensor, such as a magnetometer.

External I2C A4 and A4

For External Sensors not supported by the Board

Analog 0 pin Auxin / Battery monitor

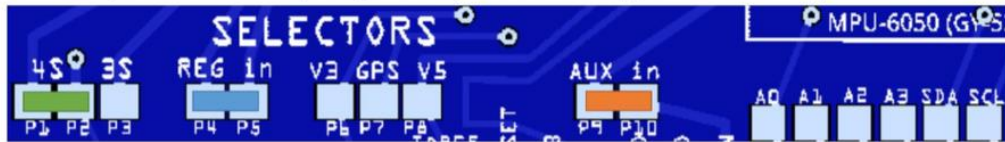
For Aux In:

- Leave it open to utilize the A0 pins for external ADC sensors.
- Short the pads to use the built-in battery monitoring. Ensure the Cell Selector is set to 3S or 4S, depending on the battery configuration.

PAD CONNECTIONS

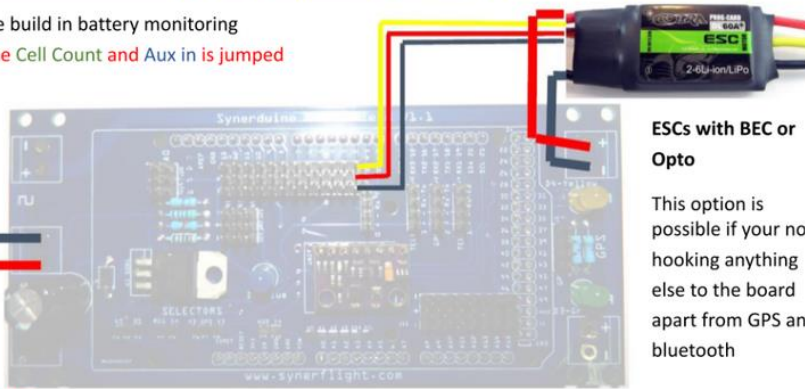
SYNERDUINO KWAD SHIELD V1 BOARD

Reg & Vbat - A0 as Battery voltage monitor , ESC BEC or OPTO applied to the 5V PWM pins



For those who would use the build in battery monitoring circuit upto 4s lipo ensure the Cell Count and Aux in is jumped before powering up

2s to 4s Lipo -
Build in Power
distribution +

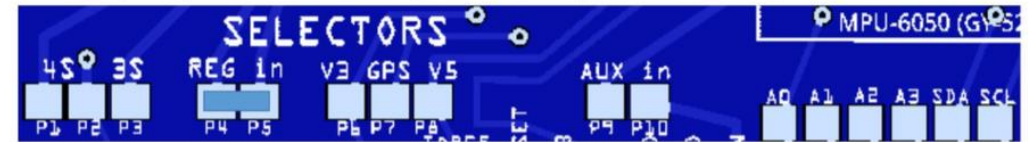


ESCs with BEC or Opto

This option is possible if your not hooking anything else to the board apart from GPS and bluetooth

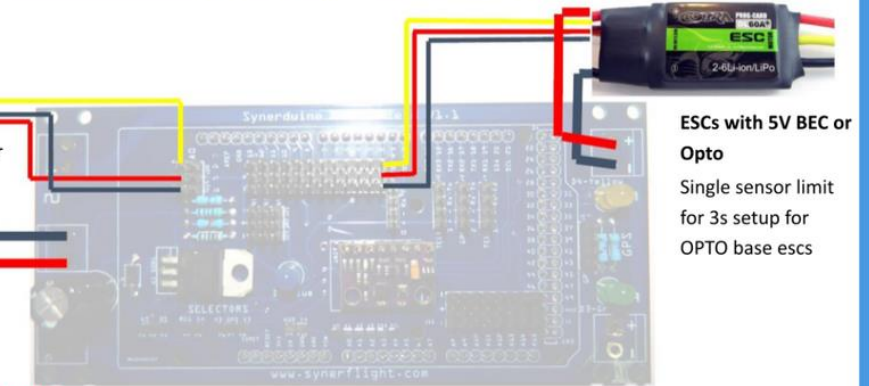
Recommended setup for beginner

Reg in only - A0 External ADC sensor , ESC BEC or OPTO applied to the 5V PWM pins



2s to 3s Lipo -
Build in Power
distribution +

A0
External ADC Sensor



ESCs with 5V BEC or Opto
Single sensor limit for 3s setup for OPTO base escs

Recommended setup for beginner

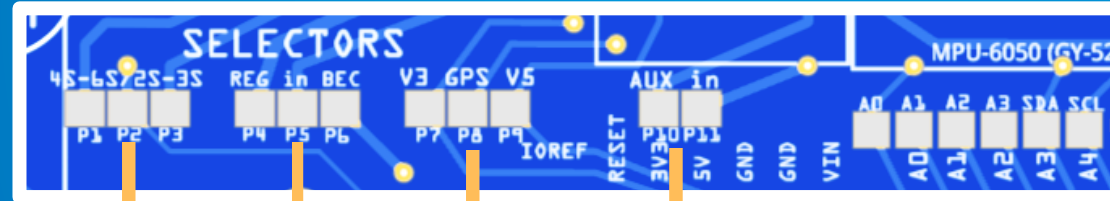
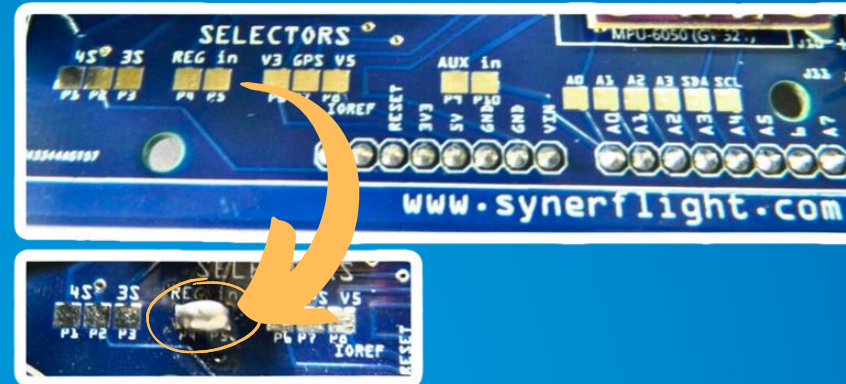
PAD CONNECTIONS

SYNERDUINO ArDUINO V2 BOARD

Step 4: Power Selector Jumper Pads are directly added to the main board, enabling users to choose the desired power source through a simple soldering step:

Apply a small blob of solder to bridge the specific pads corresponding to your preferred power option.

This approach provides a secure connection, giving you the flexibility to configure power supply without additional components, all in a compact and reliable manner.



Battery cell monitoring 4s or 3s

To use the onboard battery monitoring with Aux In:

- Set to 3S if you're using a 1S-3S battery.
- Set to 4S if you're using a 4S battery.
- Leave it open when using Aux In as external sensors or when using 5S-6S batteries.

5V Regulator from battery

- For Reg In, short the pads to use the regulator to power both the Synerduino and Arduino board, along with the built-in power distributor.
- Bec to Used ESC's BeC to power the Synerduino shield and Arduino Board

GPS Pins V+ voltage in front of the board

The 2nd GPS pin comes with a voltage selector:

- Set to 5V for a regular GPS.
- Set to 3V for an external I2C sensor, such as a magnetometer.

Analog 0 pin Auxin / Battery monitor

For Aux In:

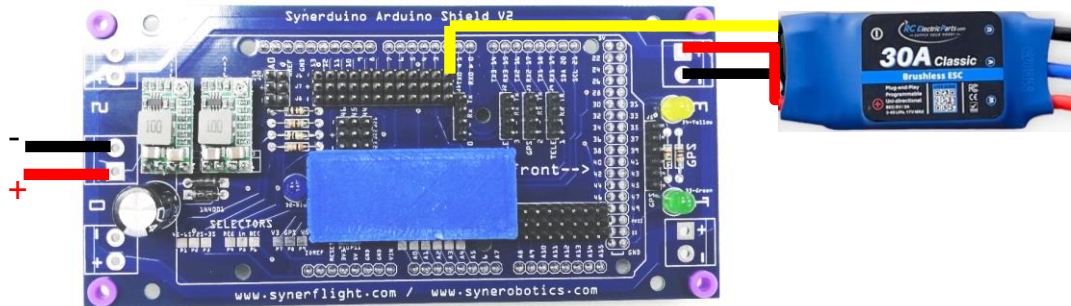
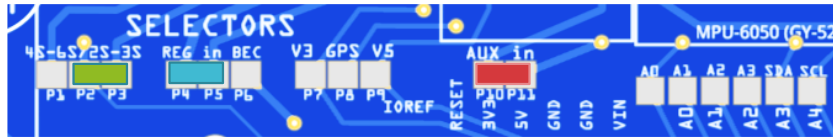
- Leave it open to utilize the A0 pins for external ADC sensors.
- Short the pads to use the built-in battery monitoring. Ensure the Cell Selector is set to 3S or 4S, depending on the battery configuration.

External i2C A4 and A4

For External Sensors not supported by the Board

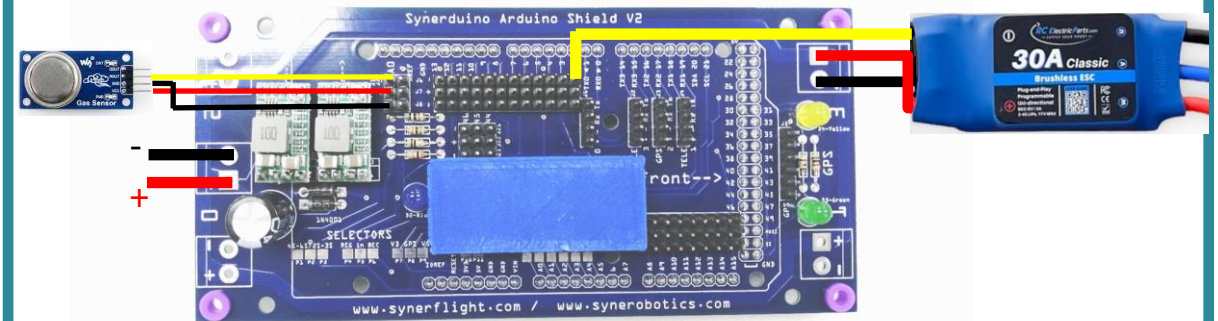
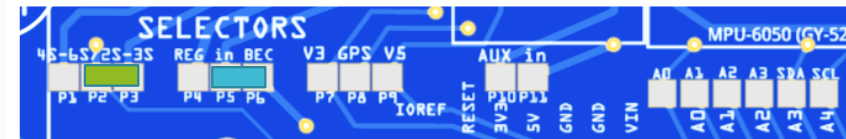
PAD CONNECTIONS

SYNERDUINO ARDUINO SHIELD V2 BOARD



Reg in – to use the onboard power supply to run the board up to 3A

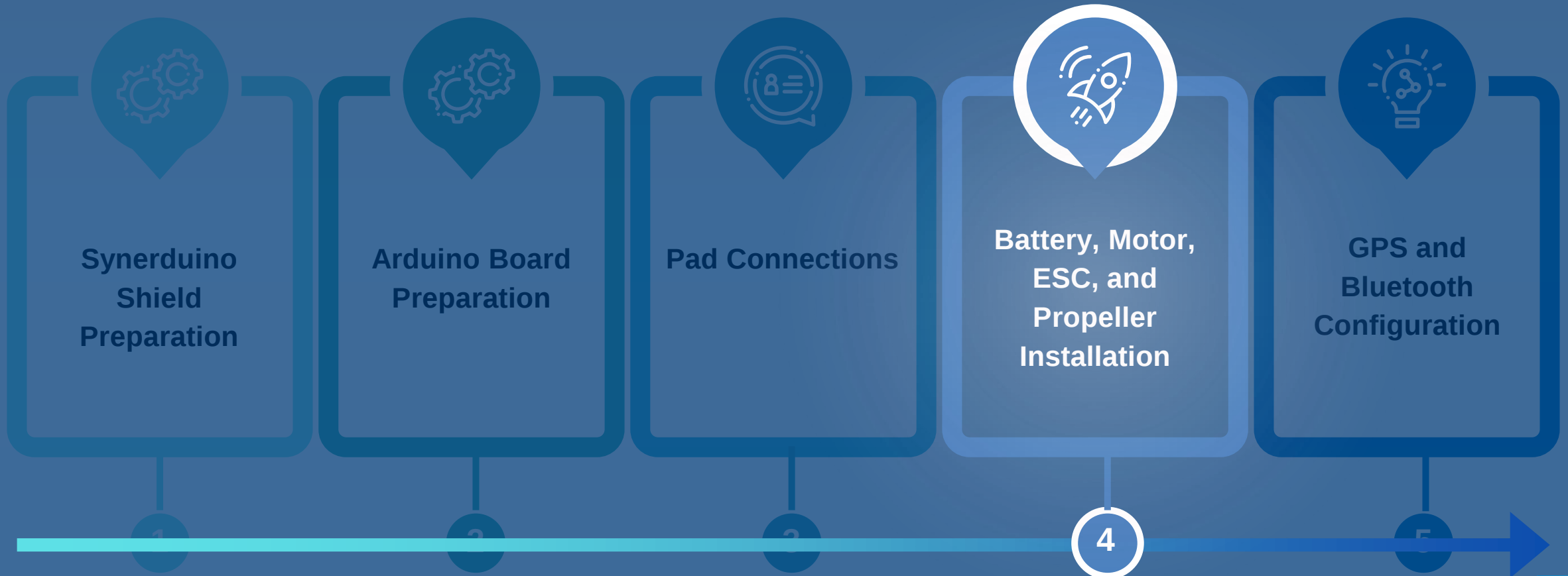
Aux in – to Read the Battert Voltage assign by the 4s-6s or 2s-3s pads



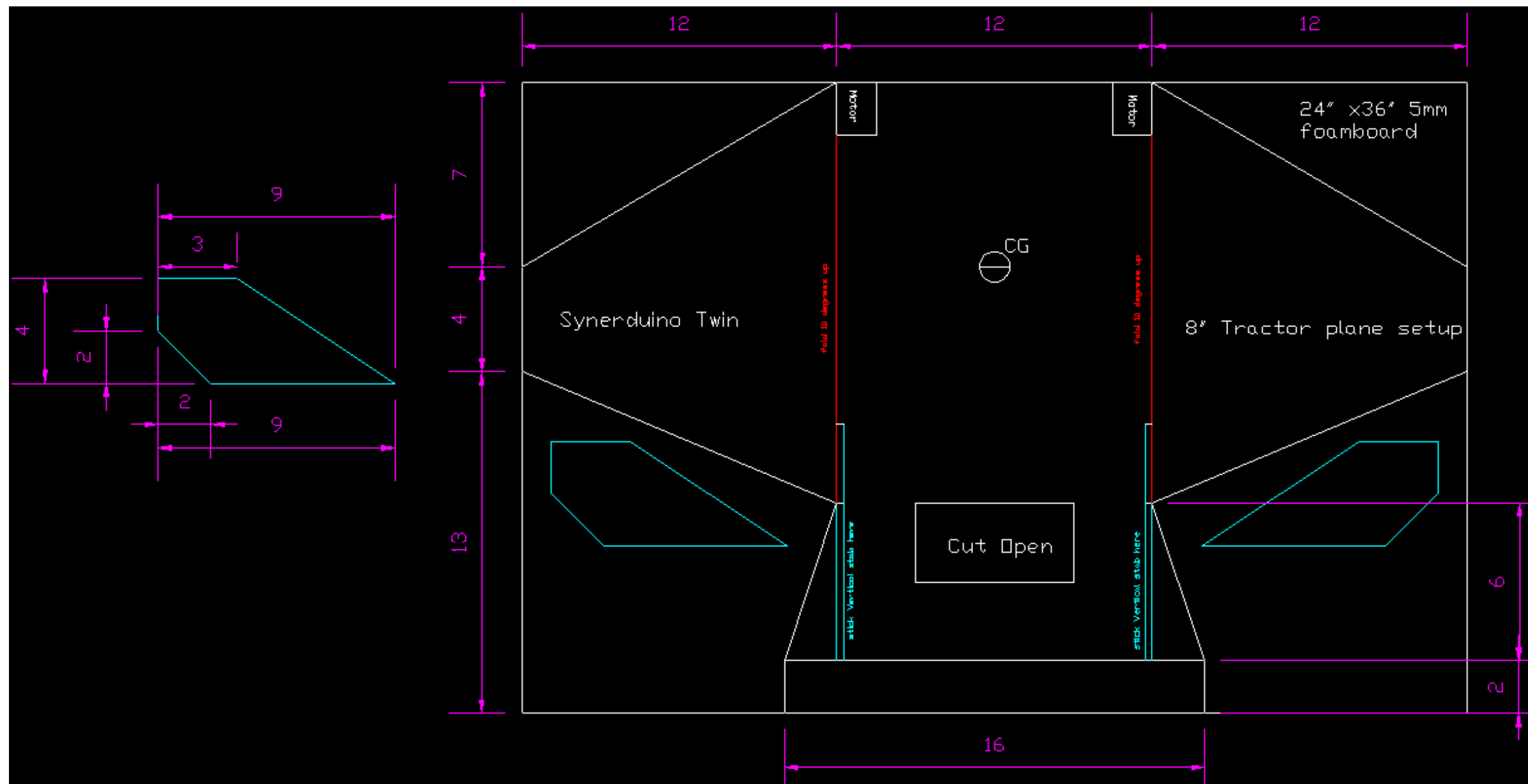
BEC in – if your using External BEC or ESC Bec to provide much needed Power to run Servos and External sensors and Companion Hardware

ASSEMBLING PROCESS

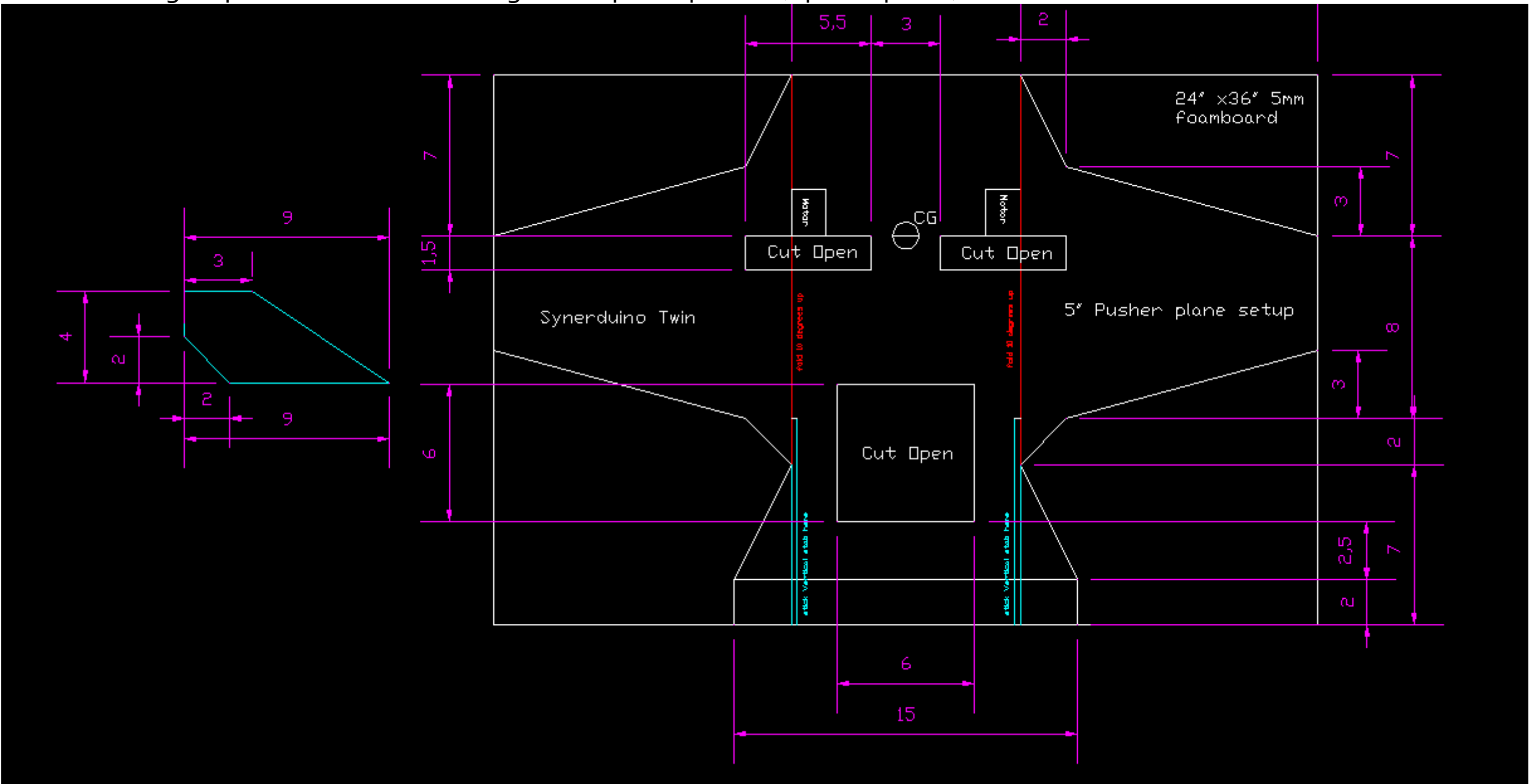
This section outlines the essential steps for assembling your Synerduino Drone Kit. Follow these steps carefully to ensure a successful assembly and get your drone ready for flight!



Synerduino Dart Design is base of the FT Flyer require 5mm 24"x36" foam board
 Wings have a dihedral of 15-20 degrees at most (Special plane)



Hole in slot Wings is possible with such design as required pusher (Special plane)



[\(Arduino 1.8.18\)SynerduinoPlane4-GY91GY801-1.8.18Download](#)

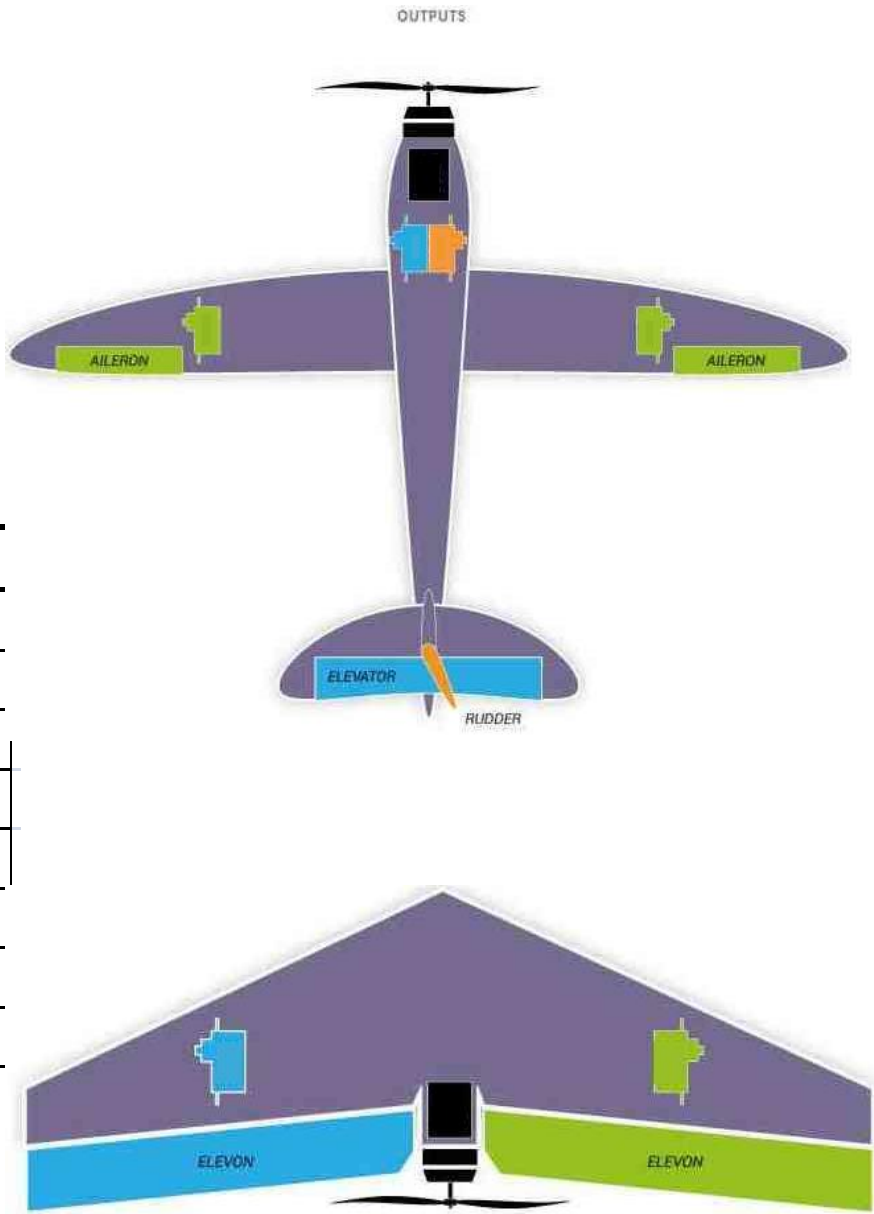
[\(Arduino 1.8.5\) SynerduinoPlanes4-GY91-GY801Download](#)

[\(PatrikE codes with updated sensors\) MultiWii_FW_synerduinoDownload](#)

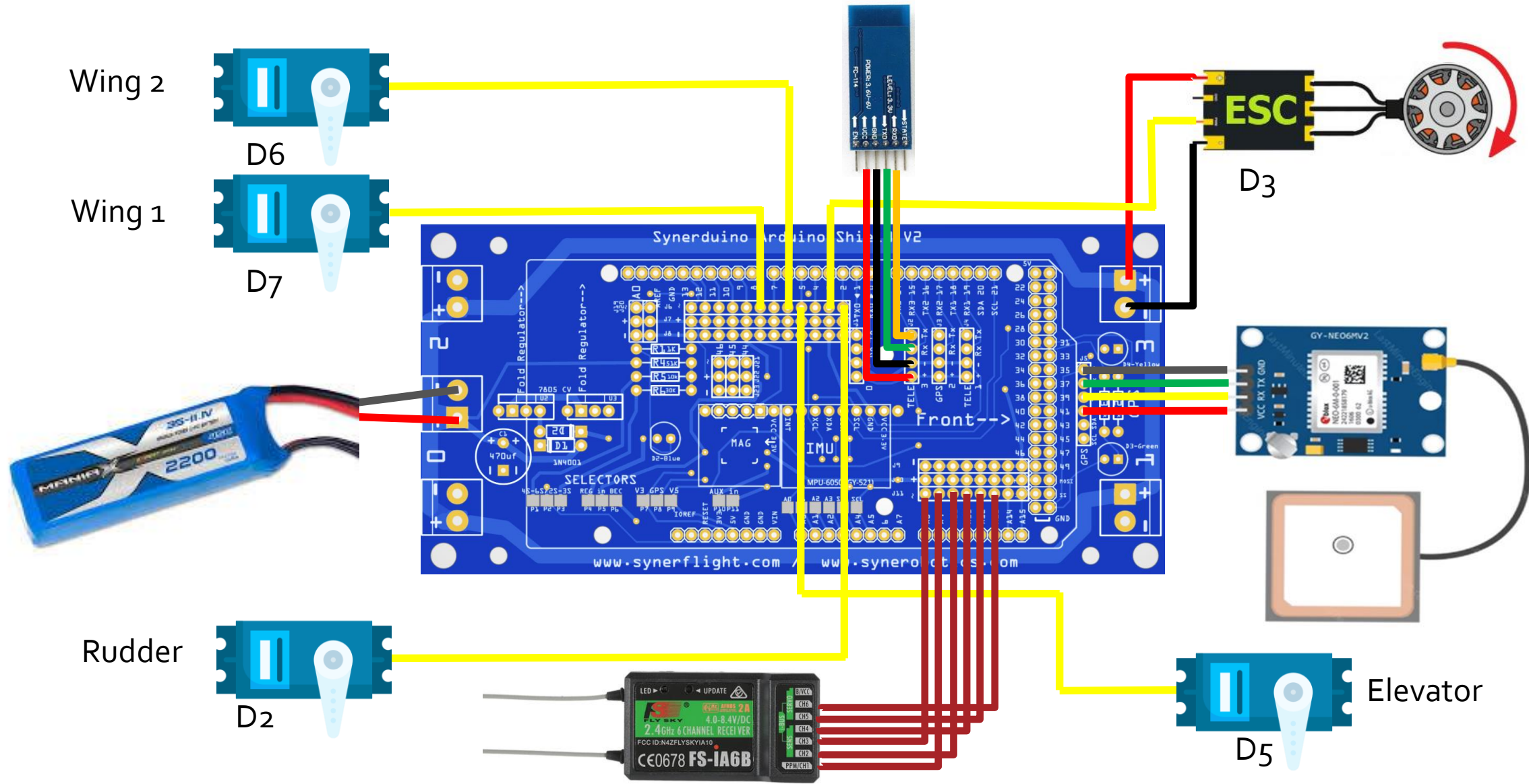
	MINI	MEGA	AirPlane	Heli-90	Heli-120	Engine nr:
servo[0] =	A0	D34/44	-	-	-	
servo[1] =	A1	D35/45	-	-	-	
servo[2] =	A2	D33/46	-	-	-	
servo[3] =	D12	D7/D37	Wing:1	Nckservo	Nckservo	
servo[4] =	D11	D6	Wing:2	Roll	Left	motor[3]
servo[5] =	D3	D2	Rudder	Tail	Tail	motor[2]
servo[6] =	D10	D5	Elev	Coll	Right	motor[1]
servo[7] =	D9	D3	Engine	Engine	Engine	motor[0]

Note: Servo 3 can also serves as conventional Flaps

See the Aircraft Settings Config.h



BATTERY, MOTOR, ESC, & PROPELLER INSTALLATION



PWM Pins arrangement PWM Output use special airplane ino file for this setup utilizing Bi copter special Plane types

Differential Thrust Wing

Arduino Mega Diff Thrust Plane

Elevator	Rudder	Aileron	Throttle
D6	D2	D7	D3 Left motor D5 Right motor

Arduino Uno Diff Thrust Plane

Elevator	Rudder	Aileron	Throttle
D11	D3	D12	D9 Left motor D10 Right motor

Arduino Mega Wing

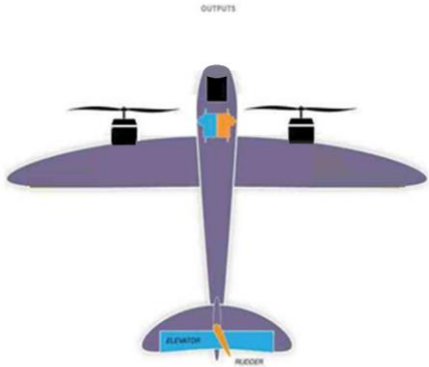
Elevon	Elevon	Rudder	Throttle
D6 L	D2 R	D7	D3 Left motor D5 Right motor

Arduino Uno Wing

Elevon	Elevon	Rudder	Throttle
D11 L	D3 R	D12	D9 Left motor D10 Right motor

Special Airplane Firmware .ino (RET ,4Ch , Differential Thrust)

[Synerduino_Special_Airplane-2-GY91-1.8.16Download](#)
[Synerduino special airplane3 GY91-1.8.16Download](#)



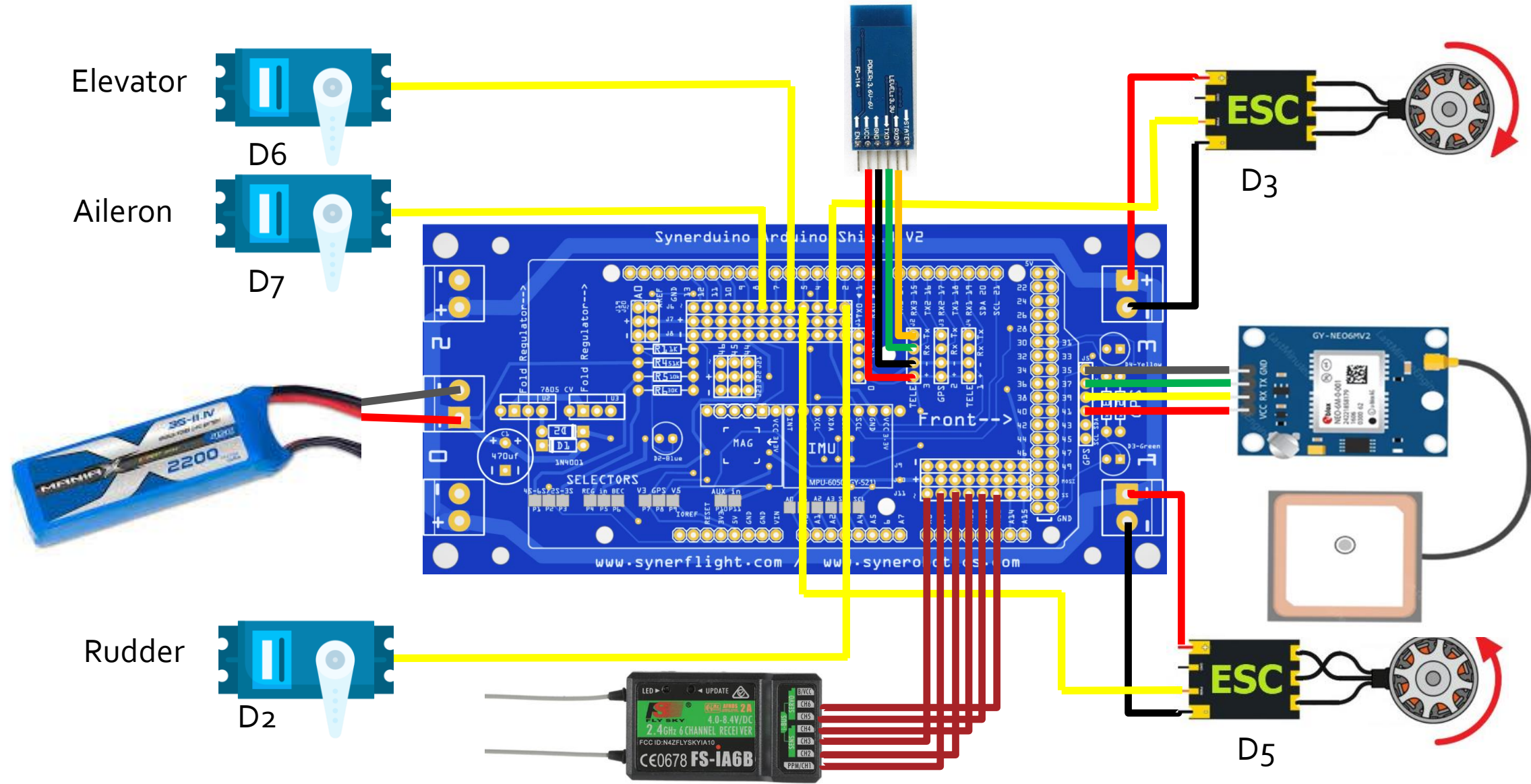
Synerduino Firmware .ino (Flying wings)

[SynerduinoWingPlane3-GY801Download](#)
[Synerduino-Firmware-AirplaneDownload](#)



Note: check if your RC servo needs reversing on FC config or on your Transmitter to ensure out put of differential respond correctly
Some models of transmitter may need RC output reversing to operate correctly

BATTERY, MOTOR, ESC, & PROPELLER INSTALLATION



PWM Pins arrangement PWM Output use special airplane ino file for this setup utilizing Bi copter special Plane types

Arduino Mega Airplane

Elevator	Rudder	Aileron	Throttle
D6	D2	D7	D3 D5 motor

Arduino Uno Airplane

Elevator	Rudder	Aileron	Throttle
D11	D3	D12	D9 D10 motor

Arduino Mega Wing

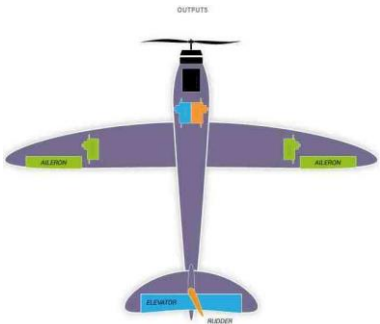
Elevon	Elevon	Aileron	Throttle
D6 L	D2 R	D7	D3 D5 motor

Arduino Uno Wing

Elevon	Elevon	Rudder	Throttle
D11 L	D3 R	D12	D9 D10 motor

Special Airplane Firmware .ino (RET ,4Ch , Differential Thrust)

[Synerduino_Special_Airplane-2-GY91-1.8.16Download](#)
[Synerduino special airplane3 GY91-1.8.16Download](#)



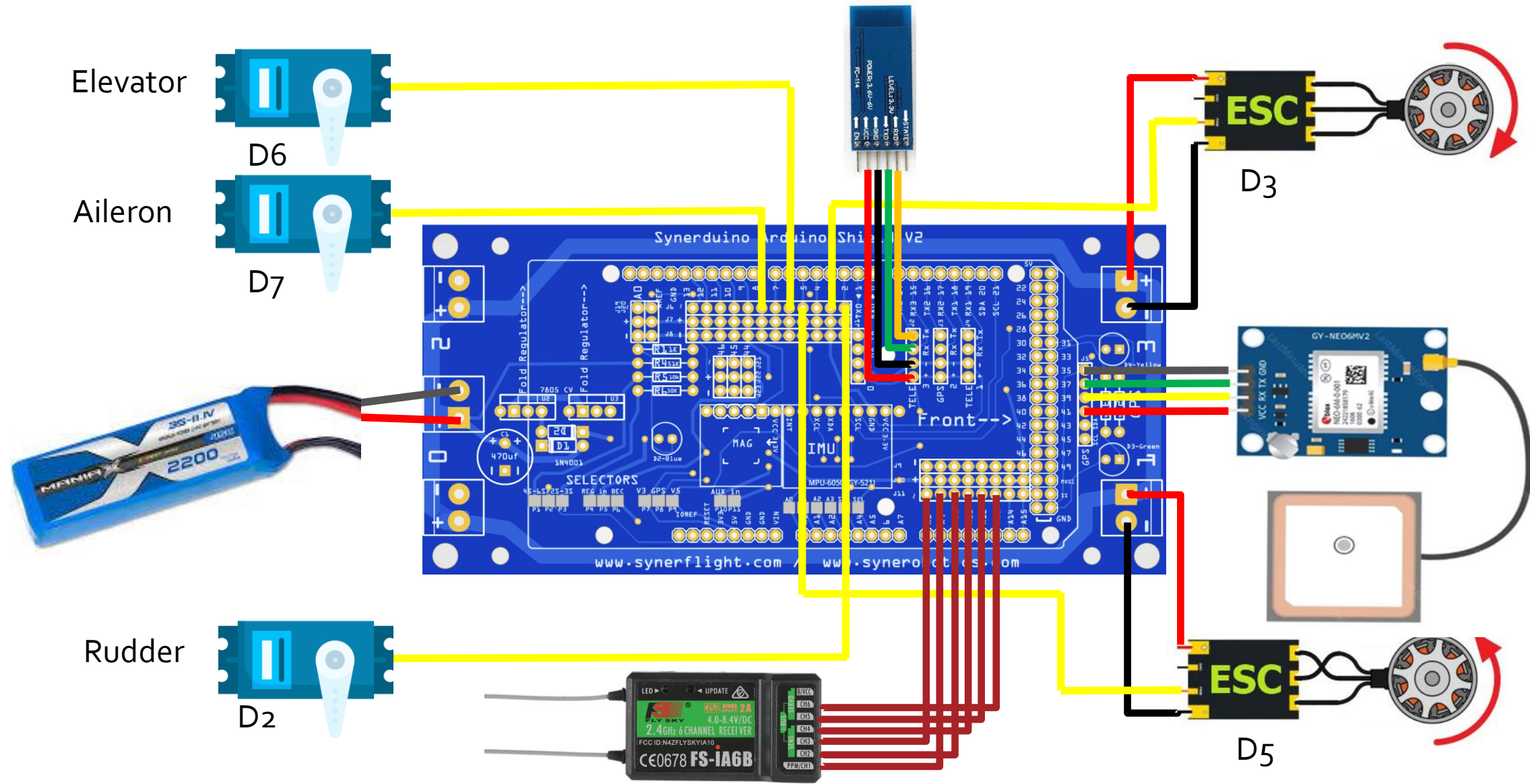
Synerduino Firmware .ino (Flying wings)

[SynerduinoWingPlane3-GY801Download](#)
[Synerduino-Firmware-AirplaneDownload](#)



Note: check if your RC servo needs reversing on FC config or on your Transmitter to ensure out put of differential respond correctly
Some models of transmitter may need RC output reversing to operate correctly

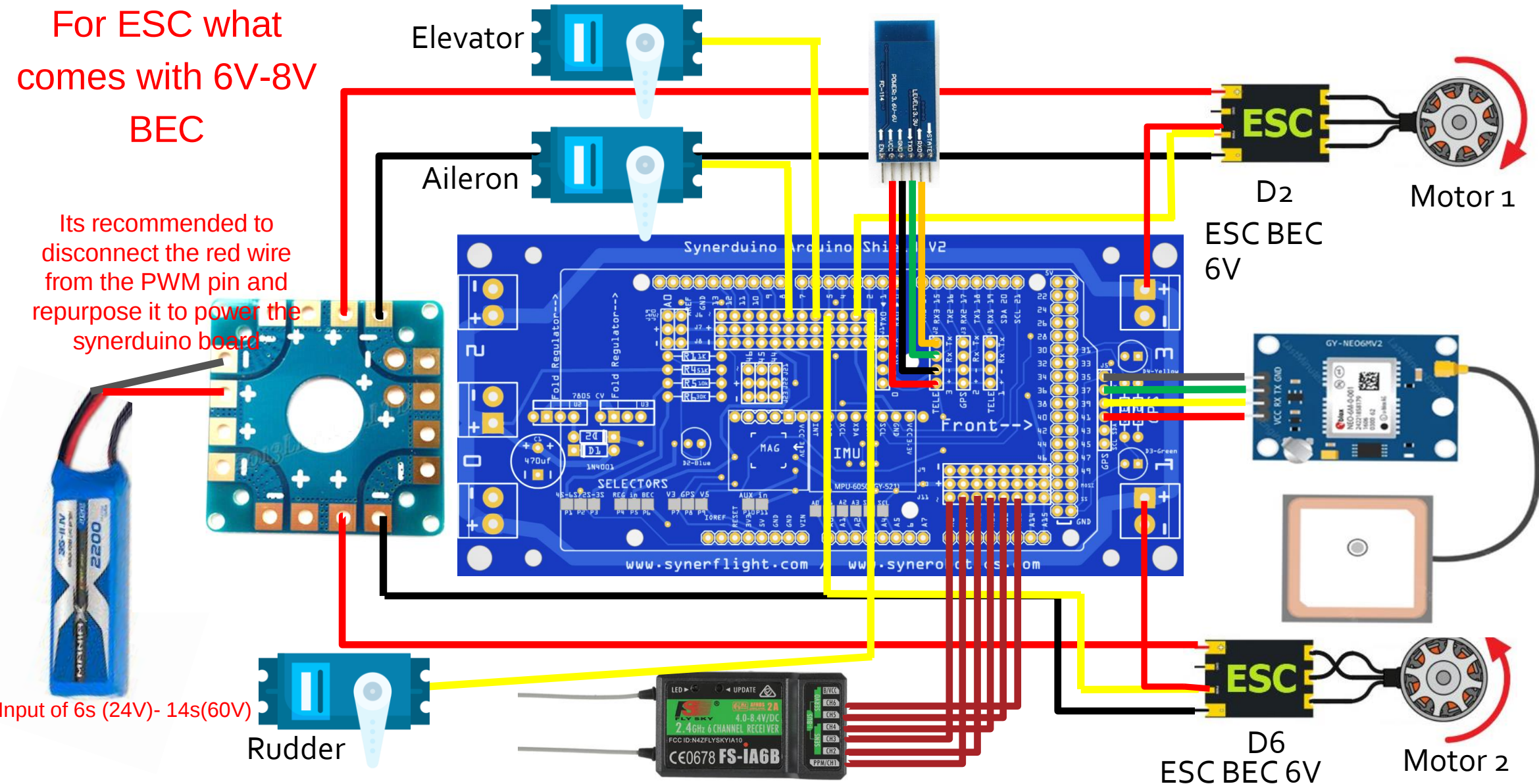
BATTERY, MOTOR, ESC, & PROPELLER INSTALLATION



High Voltage BEC (5V- 12V) and High Current (40A-300A) DIFFERENTIAL

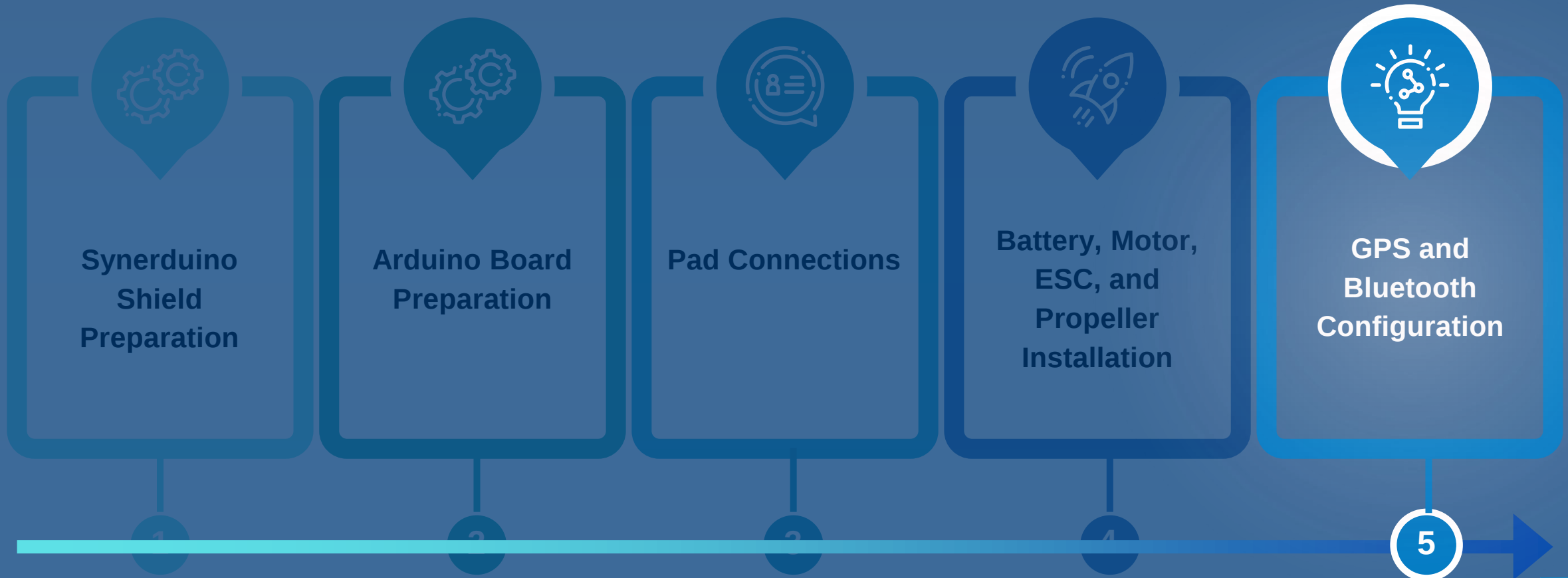
For ESC what comes with 6V-8V BEC

Its recommended to disconnect the red wire from the PWM pin and repurpose it to power the synerduino board



ASSEMBLING PROCESS

This section outlines the essential steps for assembling your Synerduino Drone Kit. Follow these steps carefully to ensure a successful assembly and get your drone ready for flight!



TELEMETRY



38400 OR 57600 FOR SIK RADIO
DEPENDENT IF USES 433MHZ OR
900MHZ (63kbps)



38400 FOR XBEE RADIO



115200 FOR BLUETOOTH HC-05

STANDARD FOR ALL DRONES TO USE SERIAL LINK AS TELEMETRY MAINLY ON YOUR TX RX SERIAL PORTS , NOTE: THE LOWER THE FREQUENCY OF THE RADIO THE LOWER THE BAUD IS NEEDED ,

MOST DRONES REQUIRE MINIMUM 63kbps AIRSPEED TO COMMUNICATE PROPERLY
PROTOCOL IS MSP RAW OR MAVLINK

BLUETOOTH CONFIGURATION

TO CONNECT THE **BLUETOOTH HC-05 MODULE** TO THE BOARD, ENSURE THE HEADERS ARE ARRANGED CORRECTLY. FOLLOW THIS WIRING CONFIGURATION:

VCC (Bluetooth) connects to **+** (Board)

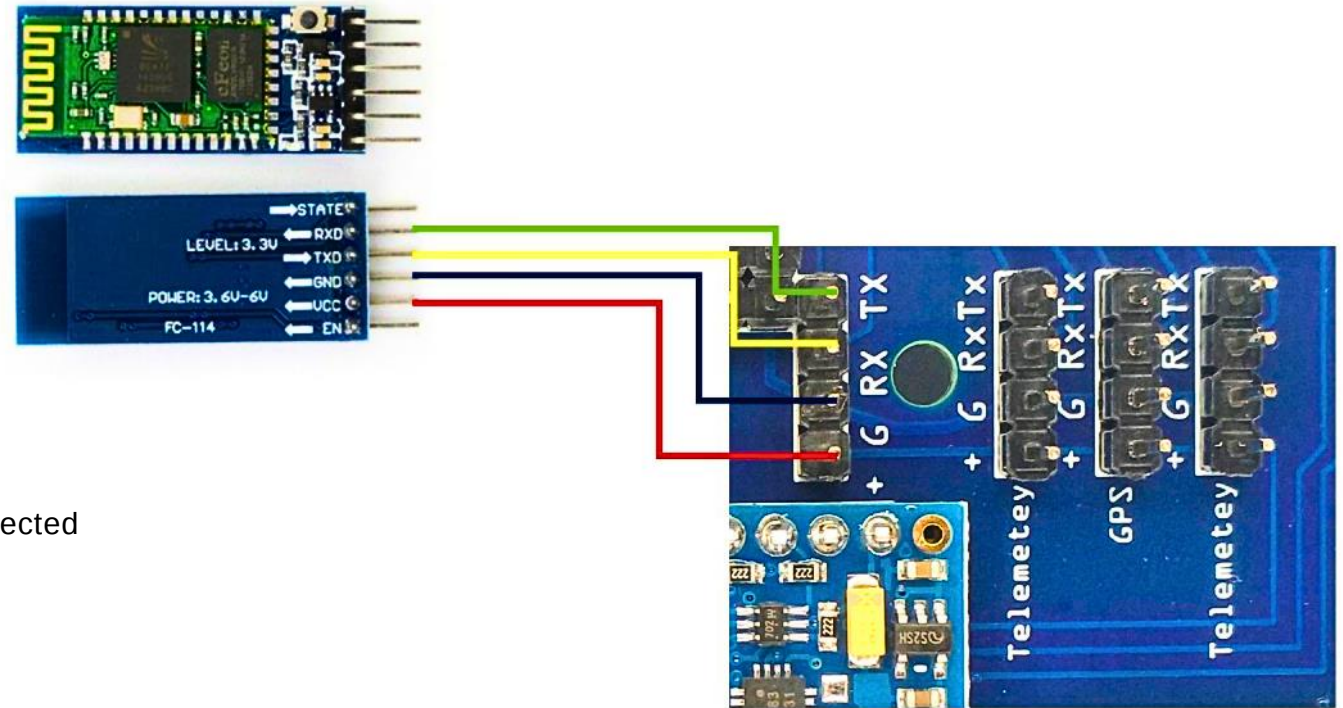
GND (Bluetooth) connects to **G** (Board)

TX (Bluetooth) connects to **RX** (Board)

RX (Bluetooth) connects to **TX** (Board)

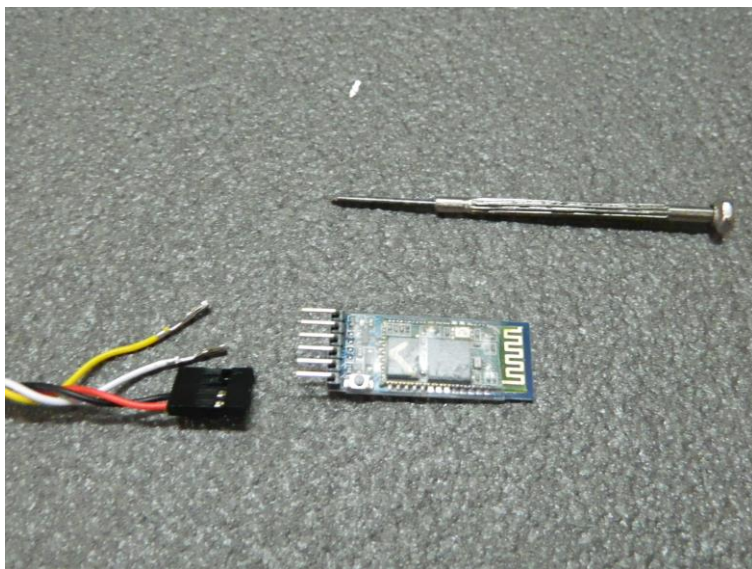
Any Serial Radio can be configured to run on
Serial 0, 1, or Serial 3. with the matching Baud

Serial 0 can be used for telemetry only if the USB is disconnected



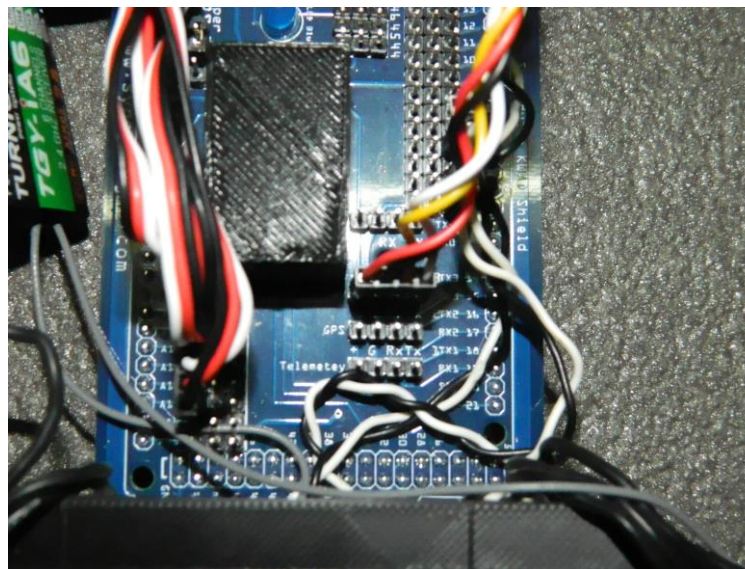
NOTE: DOUBLE-CHECK THAT THE WIRE COLORS MATCH THESE MARKINGS. INCORRECT INSTALLATION OR POLARITY MAY DAMAGE THE ARDUINO BOARD. THE BLUETOOTH MODULE IS PRESET TO A BAUD RATE OF 115200 FOR YOUR CONVENIENCE, BUT YOU MAY CHANGE THE SETTINGS IF NEEDED.

Bluetooth



BLUETOOTH PLUG INTO SERIAL 1 OR
SERIAL 3

115200 FOR BLUETOOTH HC-05



ATTENTION:

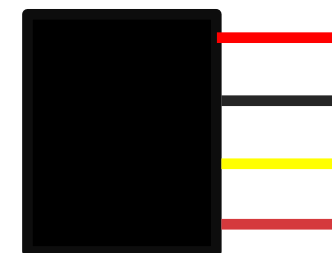
YOU MAY NEED TO REARRANGE THE HEADERS
TO CONNECT THE BLUETOOTH MODULE TO THE
SHIELD BOARD ACCORDINGLY

VCC >> +

GND >> G

TX >> RX

RX >> TX

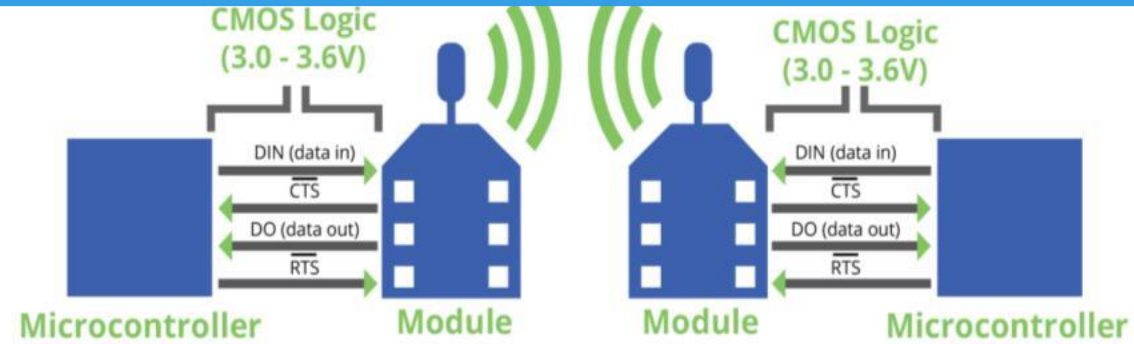


SEE TO IT THE WIRES COLOR CODE MATCHES THE
MARKINGS

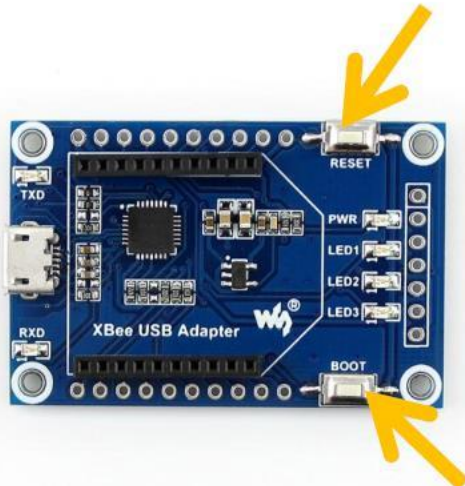
IMPROPER INSTALLATION MAY CAUSE DAMAGE
TO THE ARDUINO BOARD AND SHIELD DUE TO
REVERSE POLARITY

NOTE: WE PRESET THE BLUETOOTH FOR YOUR
CONVENIENCE TO THE PROPER SETUP BUT
SHOULD YOU WISH TO CHANGE THE SETTING ON
YOUR DIGRESSION

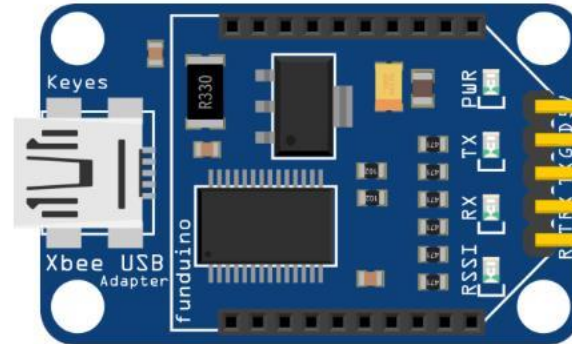
SERIAL RADIO CONFIGURATION



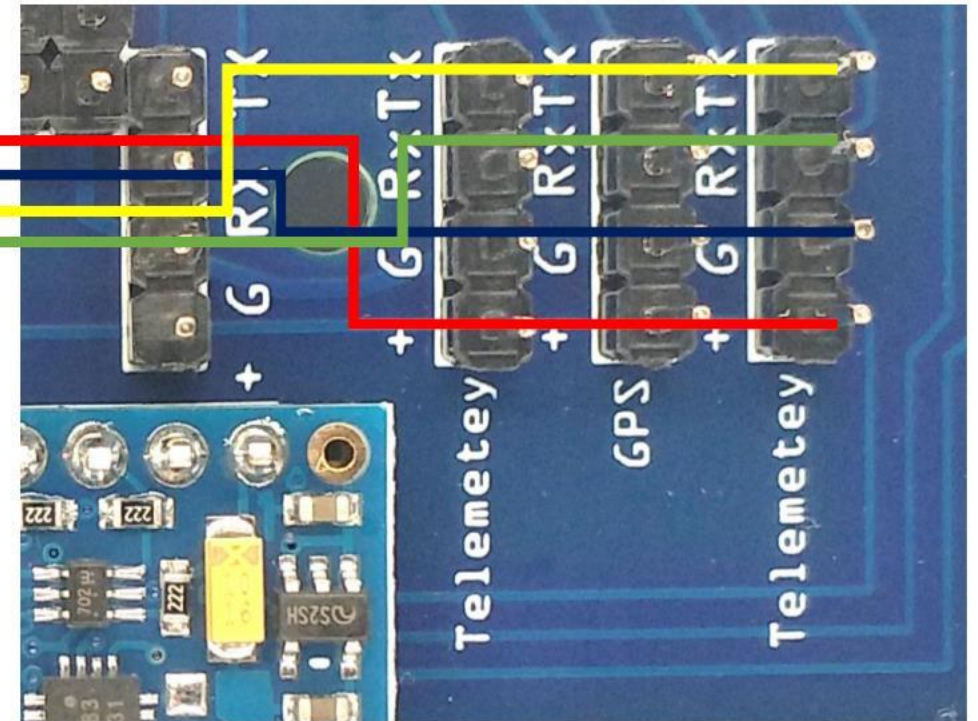
38400 FOR XBEE RADIO



GET THE USB MODULE WITH
BOOT AND RESET BUTTON AS
YOU MAY NEED TO RESET THE
XBEE WHEN UPDATING
FIRMWARE



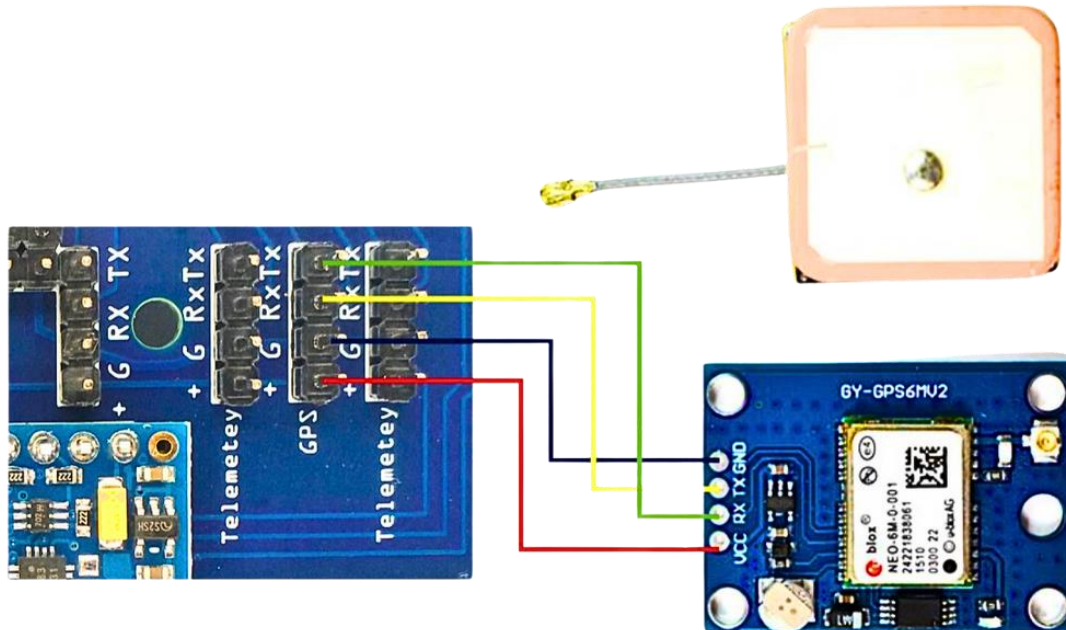
VCC >> +
GND >> G
TX >> TX
RX >> RX



GPS CONFIGURATION

THE **TELEMETRY MODULE** INCLUDES PINS LABELED RX, TX, GND, AND VCC, WHICH CORRESPOND TO THE GPS MODULE'S MATCHING PINS.

TO CONNECT THESE COMPONENTS, COLOR-CODED WIRES CLARIFY THESE CONNECTIONS, WITH **YELLOW** INDICATING **RX TO TX**, **GREEN** FOR **TX TO RX**, **BLACK** FOR **GROUND (GND)**, AND **RED** FOR **POWER (VCC)**.



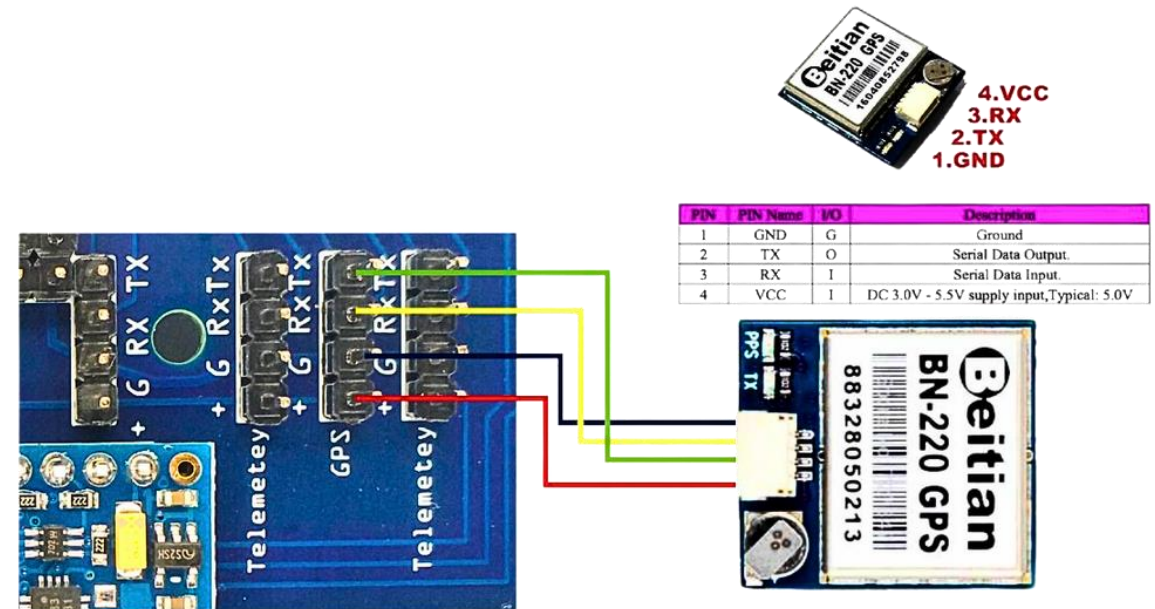
THE **BEITIAN BN-220 GPS MODULE** SHOWS A SPECIFIC PINOUT TABLE THAT DESCRIBES EACH PIN'S FUNCTION:

PIN 1 IS GND (GROUND)

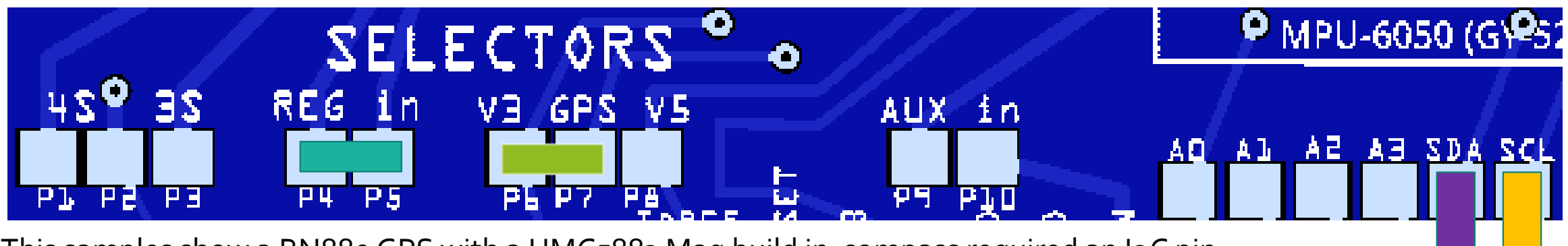
PIN 2 IS TX (DATA OUTPUT)

PIN 3 IS RX (DATA INPUT)

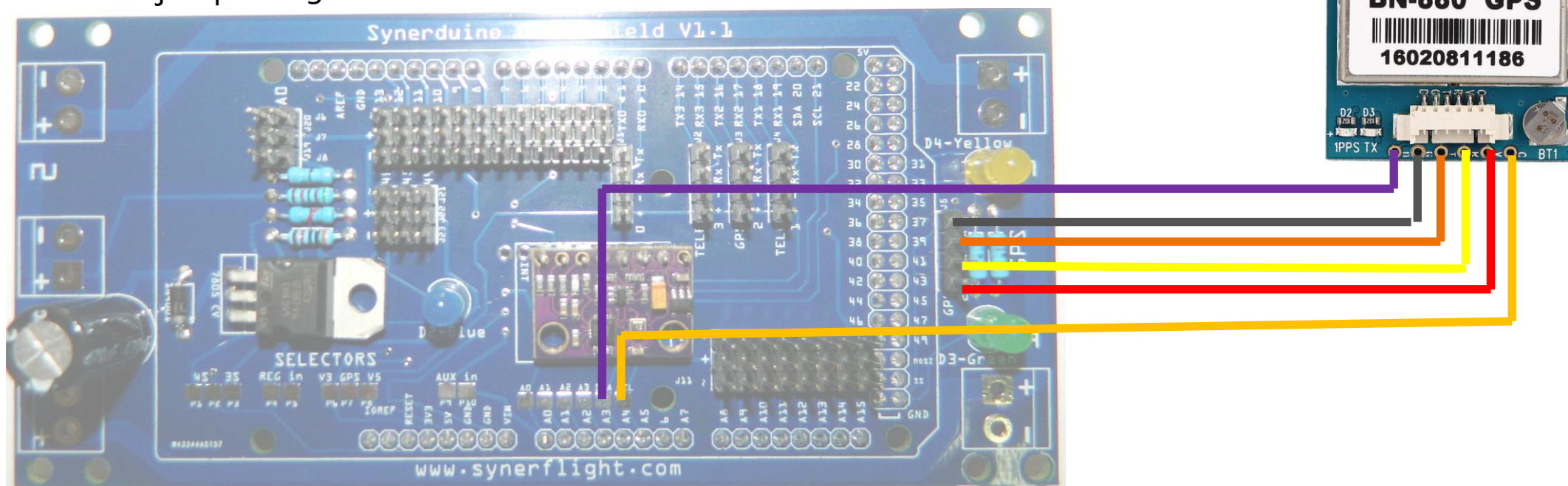
PIN 4 IS VCC (POWER SUPPLY RANGES FROM 3.3V TO 5.0V)



External Sensors



This samples show a BN880 GPS with a HMC5883 Mag build in compass required an I2C pin connection this works of all other I2C sensors (pls ensure the address doesn't conflict with the IMU as found in Sensors.cpp) Note: other than the GPS build in sensors might require 3V you may need to set jumper to 3V



RECEIVER TYPES PROTOCOL



PPM RECEIVER



SERIALRECEIVER



PWM RECEIVER

RECEIVER TYPE CONFIGURATIONS

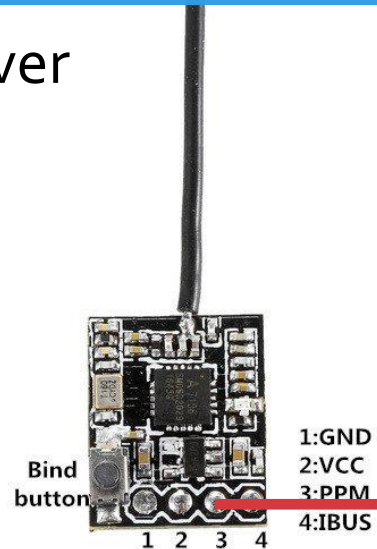
RX > SBUS INPUT	FUTABA FORMAT	JR FORMAT	WALKERA FORMAT	GRAUPNER FORMAT	MEGA 2560
	AETR	TAER	EATR	ERTA	INPUT
					
THROTTLE	CH3	CH1	CH3	CH3	A8
AILERON	CH1	CH2	CH2	CH4	A9
ELEVATOR	CH2	CH3	CH1	CH1	A10
RUDDER	CH4	CH4	CH4	CH2	A11
AUX 1	CH5	CH5	CH5	CH5	A12
AUX 2	CH6	CH6	CH6	CH6	A13
AUX 3	CH7	CH7	CH7	CH7	A14
AUX 4	CH8	CH8	CH8	CH8	A15

RECEIVER TYPE CONFIGURATIONS

RX > SBUS INPUT	FUTABA FORMAT	JR FORMAT	WALKERA FORMAT	GRAUPNER FORMAT	UNO
	AETR	TAER	EATR	ERTA	INPUT
					
THROTTLE	CH3	CH1	CH3	CH3	D2
AILERON	CH1	CH2	CH2	CH4	D4
ELEVATOR	CH2	CH3	CH1	CH1	D5
RUDDER	CH4	CH4	CH4	CH2	D6
AUX 1	CH5	CH5	CH5	CH5	D7
AUX 2	CH6	CH6	CH6	CH6	D8
AUX 3	CH7	CH7	CH7	CH7	N/A
AUX 4	CH8	CH8	CH8	CH8	N/A

OTHER RECEIVER TYPE

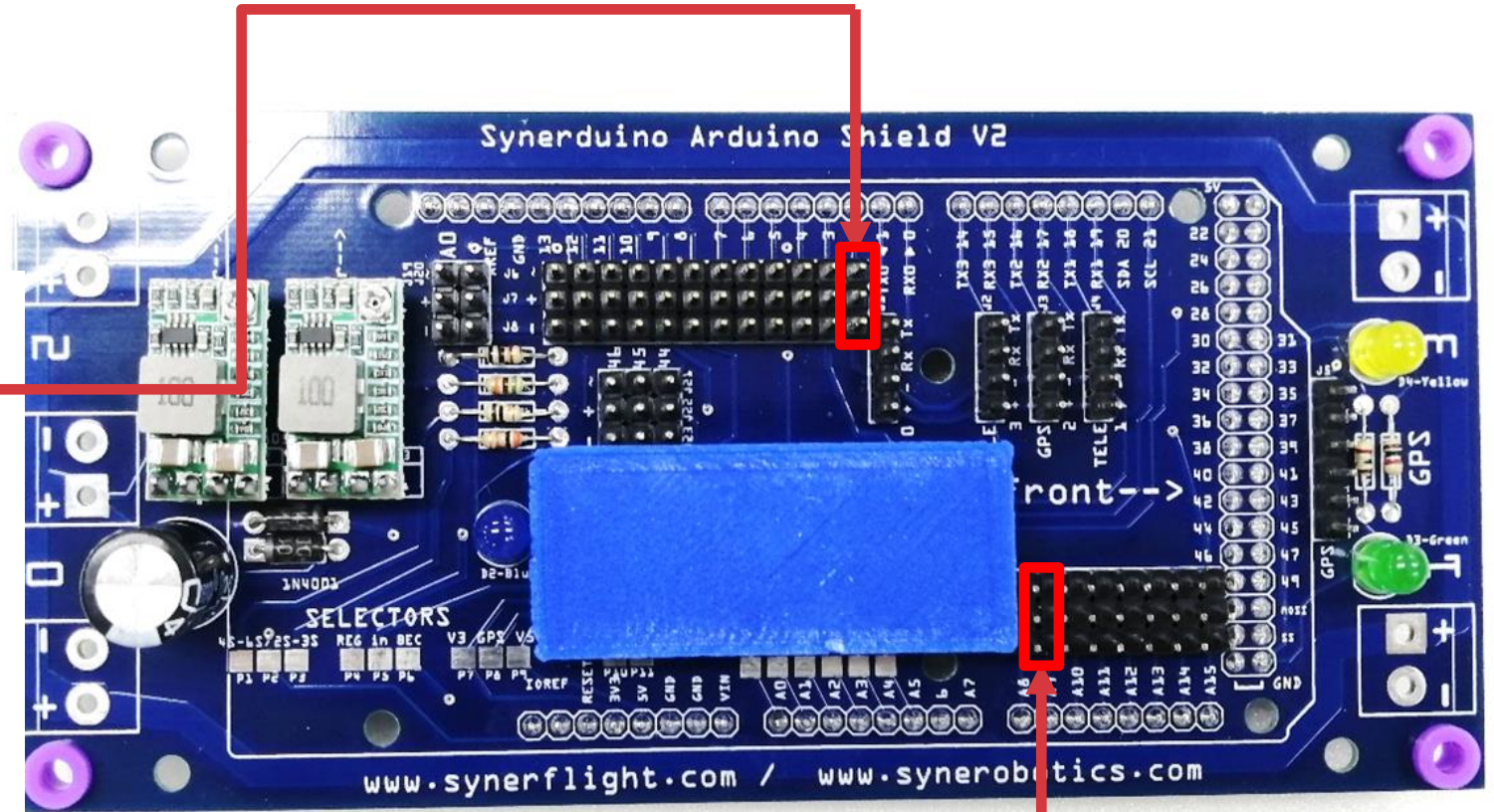
PPM Receiver



Pin D2 for Uno



Pin A8 for Mega

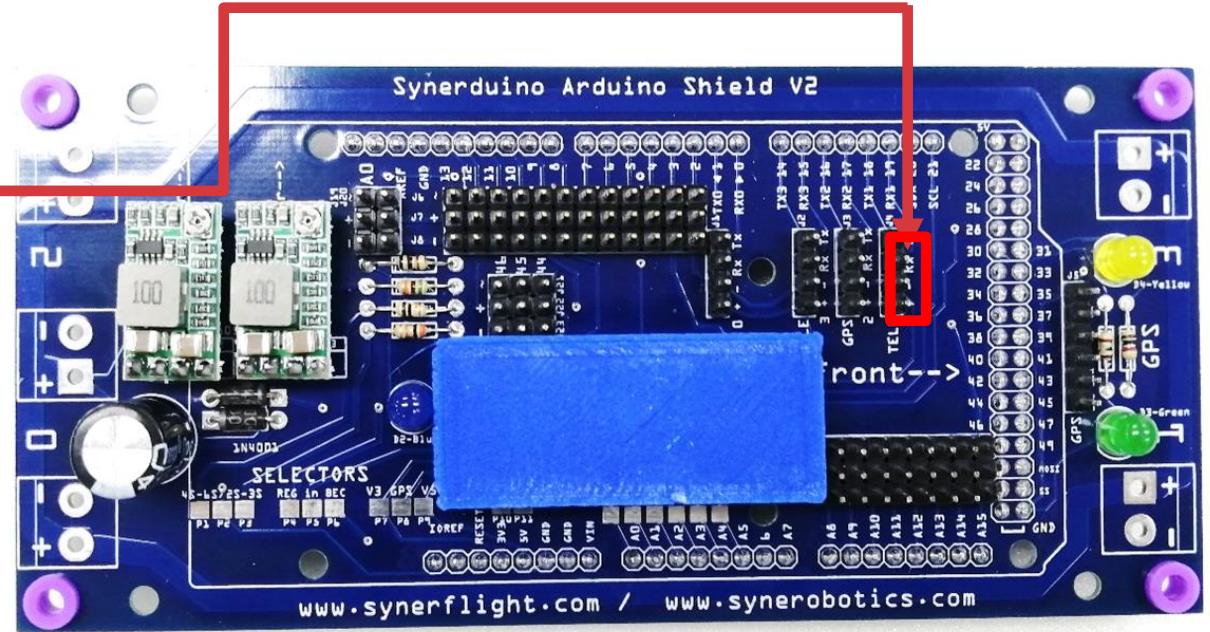


OTHER RECEIVER TYPE

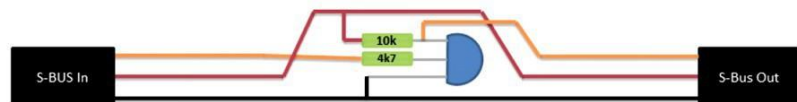
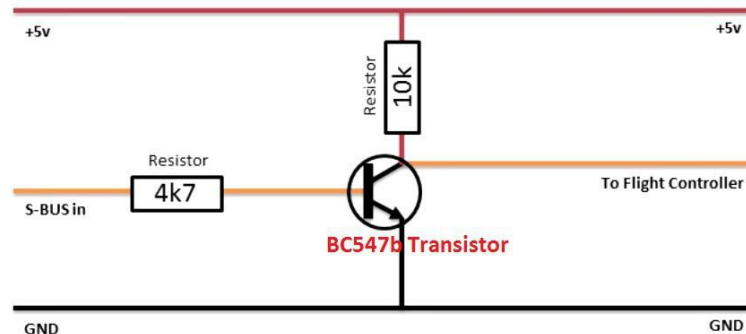
SBUS Receiver



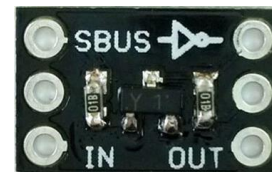
RX 1 Telemetry



The SBUS system uses Futaba protocol and should be compatible with Most SBUS Receivers



blog.OscarLiang.net



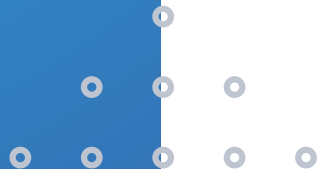
SBUS Inverter

Should there be issues in Signal Inversion an SBUS inverter may be use

You can tell if the RC control is not being read

Most modern Receivers now comes with Serial Protocol as they are faster than the old PWM or PPM standard and its now the Modern defacto for Receiver to Flight Control Board communication

SOFTWARE SETUP



AIRCRAFT SELECTION FOR FIXWINGS

Synerflight has 2 Selection for Fixwings on the Download tab

Legacy – support single motor Fixwings (Airplane and Flying Wing)

- Legacy Plane (Single Motor Classic FixWing)
- Synerduino Firmware .ino (Flying wings)

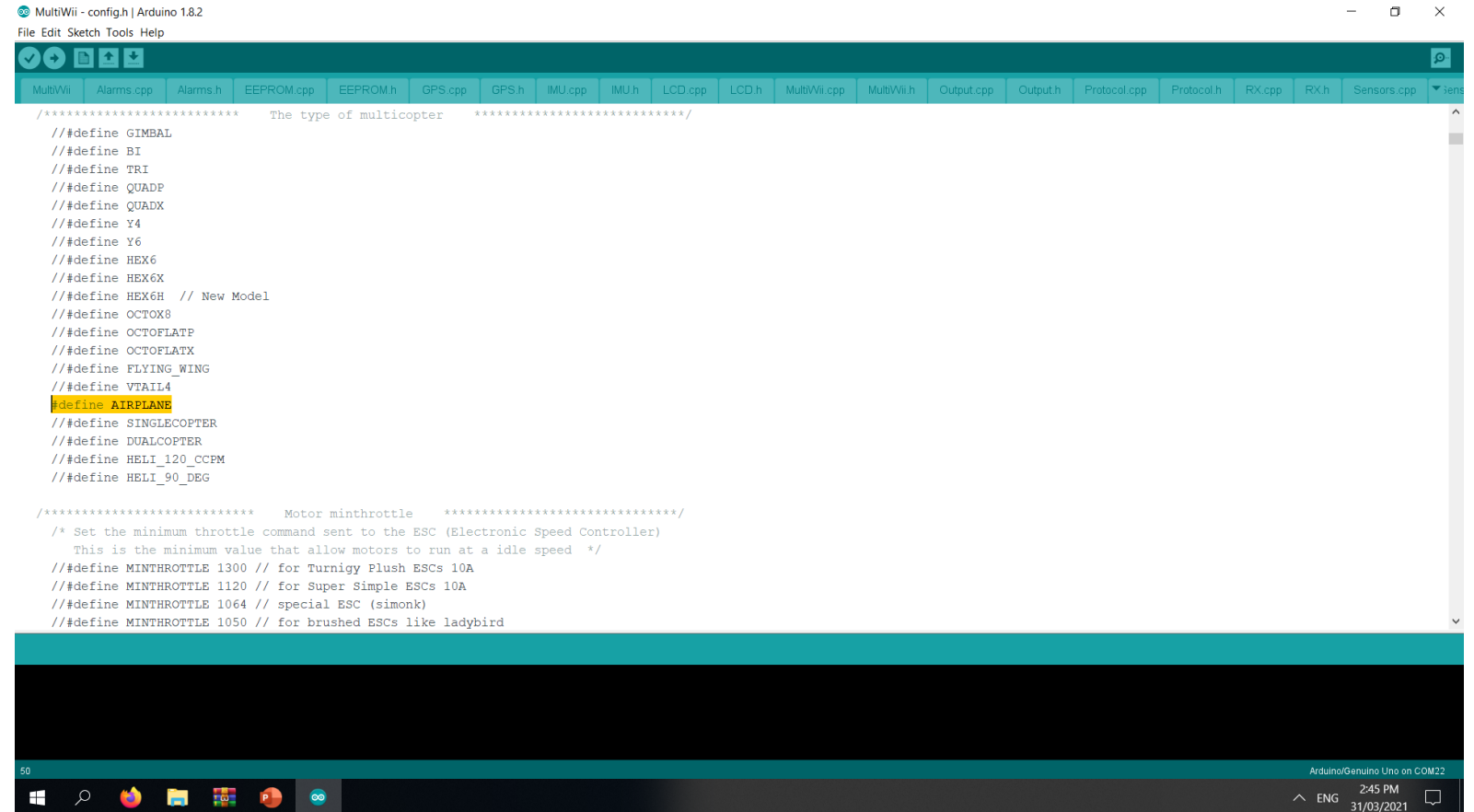
Special – Support Diff thrust and Special Mode Fixwings

- Special Airplane (Support M9 – M10 NMEA , M7-M8 UBX.Home Reset) (2024)
- Legacy Special Airplane (Support M5 – M6 NMEA , M7-M8 UBX) (2023)

AIRCRAFT SELECTION FOR FIXWINGS

Legacy Airplane

- Uncomment #Define AIRPLANE
- This puts the vehicle into
- Conventional Airplane mode
- Uncomment #Define FLYINGWING
- This puts the vehicle into
- Conventional FlyingWing mode



```
MultiWii - config.h | Arduino 1.8.2
File Edit Sketch Tools Help

MultiWii Alarms.cpp Alarms.h EEPROM.cpp EEPROM.h GPS.cpp GPS.h IMU.cpp IMU.h LCD.cpp LCD.h MultiWii.cpp MultiWii.h Output.cpp Output.h Protocol.cpp Protocol.h RX.cpp RX.h Sensors.cpp Sensors.h

/***** The type of multicopter *****/
// #define GIMBAL
// #define BI
// #define TRI
// #define QUADP
// #define QUADX
// #define Y4
// #define Y6
// #define HEX6
// #define HEX6X
// #define HEX6H // New Model
// #define OCTOX8
// #define OCTOFLATP
// #define OCTOFLATX
// #define FLYING_WING
// #define VTAIL4
// #define AIRPLANE
// #define SINGLECOPTER
// #define DUALCOPTER
// #define HELI_120_CCPM
// #define HELI_90_DEG

/***** Motor minthrottle *****/
/* Set the minimum throttle command sent to the ESC (Electronic Speed Controller)
   This is the minimum value that allow motors to run at a idle speed */
// #define MINTHROTTLE 1300 // for Turnigy Plush ESCs 10A
// #define MINTHROTTLE 1120 // for Super Simple ESCs 10A
// #define MINTHROTTLE 1064 // special ESC (simonk)
// #define MINTHROTTLE 1050 // for brushed ESCs like ladybird

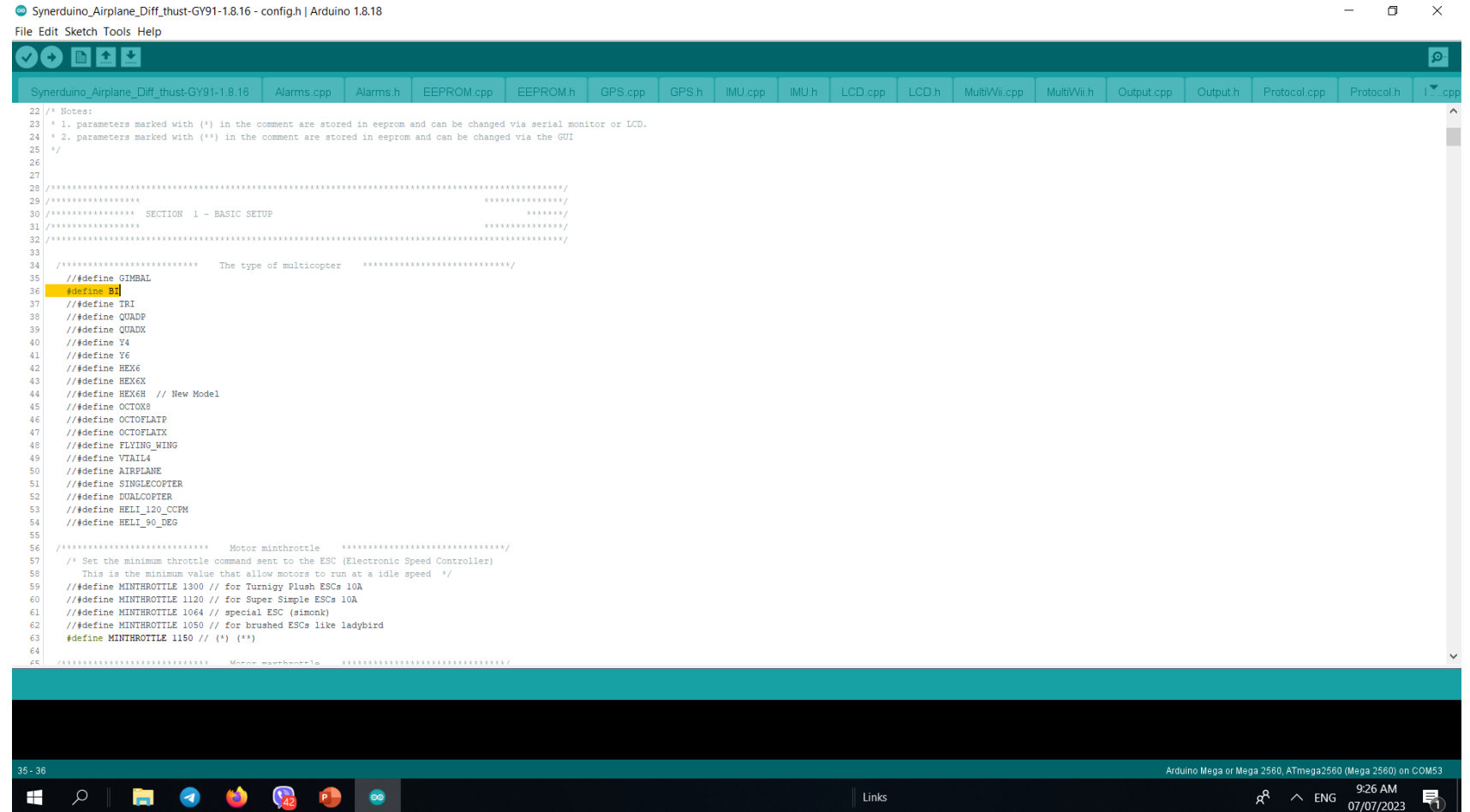
50
Arduino/Genuino Uno on COM22
ENG 2:45 PM 31/03/2021
```

AIRCRAFT SELECTION FOR FIXWINGS

Special Airplane

Config.h

- Uncomment Define Bi
- This puts the vehicle into
- Bi diff thrust mode Spacial plane mode
- Only applicable on special Airplane ino skech



```
Synerduino_Airplane_Diff_thrust-GY91-1.8.16 - config.h | Arduino 1.8.18
File Edit Sketch Tools Help

22 /* Notes:
23  * 1. parameters marked with (*) in the comment are stored in eeprom and can be changed via serial monitor or LCD.
24  * 2. parameters marked with (**) in the comment are stored in eeprom and can be changed via the GUI
25  */
26
27
28 /***** SECTION 1 - BASIC SETUP *****/
29
30 /***** SECTION 1 - BASIC SETUP *****/
31
32 /***** SECTION 1 - BASIC SETUP *****/
33
34 /***** The type of multicopter *****/
35
36 #define Bi
37 // #define TRI
38 // #define QUADP
39 // #define QUADX
40 // #define Y4
41 // #define Y6
42 // #define HEX6
43 // #define HEX6X
44 // #define HEX6H // New Model
45 // #define OCTOX8
46 // #define OCTOFLATP
47 // #define OCTOFLATX
48 // #define FLYING_WING
49 // #define VTAIL4
50 // #define AIRPLANE
51 // #define SINGLECOPTER
52 // #define DUALCOPTER
53 // #define HELI_120_OCRM
54 // #define HELI_90_DEG
55
56 /***** Motor minthrottle *****/
57 /* Set the minimum throttle command sent to the ESC (Electronic Speed Controller)
58  This is the minimum value that allow motors to run at a idle speed */
59 // #define MINTHROTTLE 1300 // for Turnigy Plush ESCs 10A
60 // #define MINTHROTTLE 1120 // for Super Simple ESCs 10A
61 // #define MINTHROTTLE 1064 // special ESC (simonk)
62 // #define MINTHROTTLE 1050 // for brushed ESCs like ladybird
63 #define MINTHROTTLE 1150 // (*) (**)
64
65 /***** Motor throttle *****/
```

AIRCRAFT SELECTION FOR FIXWINGS

CONFIG.H

IF FIXWING USES FLAPPERON OR CONVENTIONAL FLAPS

Synerduino_4CHAirplane-2-GY91-1.8.16 - config.h | Arduino 1.8.18

File Edit Sketch Tools Help

```
281 /***** Cam Stabilisation *****/
282 /* The following lines apply only for a pitch/roll tilt stabilization system. Uncomment the first or second line to activate it */
283 // #define SERVO_MIX_TILT
284 #define SERVO_TILT
285
286 /* camera trigger function : activated via Rc Options in the GUI, servo output=A2 on promini */
287 // trigger interval can be changed via (*GUI*) or via AUX channel
288 #define CAMTRIG
289 #define CAM_TIME_HIGH 1000 // the duration of HIGH state servo expressed in ms
290
291 /***** Airplane *****/
292 // #define USE_THROTTLESERVO // For use of standard 50Hz servo on throttle.
293
294 // #define FLAPPERONS AUX4 // Mix Flaps with Ailerons.
295 #define FLAPPERON_EP { 1500, 1700 } // Endpoints for flaps on a 2 way switch else set {1020,2000} and program in radio.
296 #define FLAPPERON_INVERT { -1, 1 } // Change direction on flapperons { Wing1, Wing2 }
297
298 // #define FLAPS // Traditional Flaps on SERVO3.
299 // #define FLAPSPEED 3 // Make flaps move slowm Higher value is Higher Speed.
300
301 /***** Common for Heli & Airplane *****/
302
303 /* Governor: attempts to maintain rpm through pitch and voltage changes
304 * predictive approach: observe input signals and voltage and guess appropriate corrections.
305 * (the throttle curve must leave room for the governor, so 0-50-75-80-80 is ok, 0-50-95-100-100 is _not_ ok.
306 * Can be toggled via aux switch.
307 */
308 // #define GOVERNOR_P 7 // (*) proportional factor. Higher value -> higher throttle increase. Must be >=1; 0 = turn off
309 // #define GOVERNOR_D 4 // (*) decay timing. Higher value -> takes longer to return throttle to normal. Must be >=1;
310
311 /* tail precomp from collective */
312 #define YAW_COLL_PRECOMP 10 // (*) proportional factor in 0.1. Higher value -> higher precomp effect. value of 10 equals no
313 #define YAW_COLL_PRECOMP_DEADBAND 120 // (*) deadband for collective pitch input signal around 0-pitch input value
314
315 // #define VOLTAGEDROP_COMPENSATION // voltage impact correction
```

Done uploading.
avrdude done. Thank you.

291 - 300

Windows taskbar: File Explorer, Firefox, VS Code, PowerPoint, etc.

MOTOR MIN THROTTLE

CONFIG.H



MultiWii - config.h | Arduino 1.8.2

File Edit Sketch Tools Help

```
//#define HELI_90_DEG

/***** Motor minthrottle *****/
/* Set the minimum throttle command sent to the ESC (Electronic Speed Controller)
   This is the minimum value that allow motors to run at a idle speed */
//#define MINTHROTTLE 1300 // for Turnigy Plush ESCs 10A
//#define MINTHROTTLE 1120 // for Super Simple ESCs 10A
//#define MINTHROTTLE 1064 // special ESC (simonk)
//#define MINTHROTTLE 1050 // for brushed ESCs like ladybird
#define MINTHROTTLE 1150 // (*) (*)

/***** Motor maxthrottle *****/
/* this is the maximum value for the ESCs at full power, this value can be increased up to 2000 */
#define MAXTHROTTLE 1850

/***** Mincommand *****/
/* this is the value for the ESCs when they are not armed
   in some cases, this value must be lowered down to 900 for some specific ESCs, otherwise they failed to initiate */
#define MINCOMMAND 1000

/***** I2C speed for old WMP config (useless config for other sensors) *****/
#define I2C_SPEED 100000L //100kHz normal mode, this value must be used for a genuine WMP
//#define I2C_SPEED 400000L //400kHz fast mode, it works only with some WMP clones

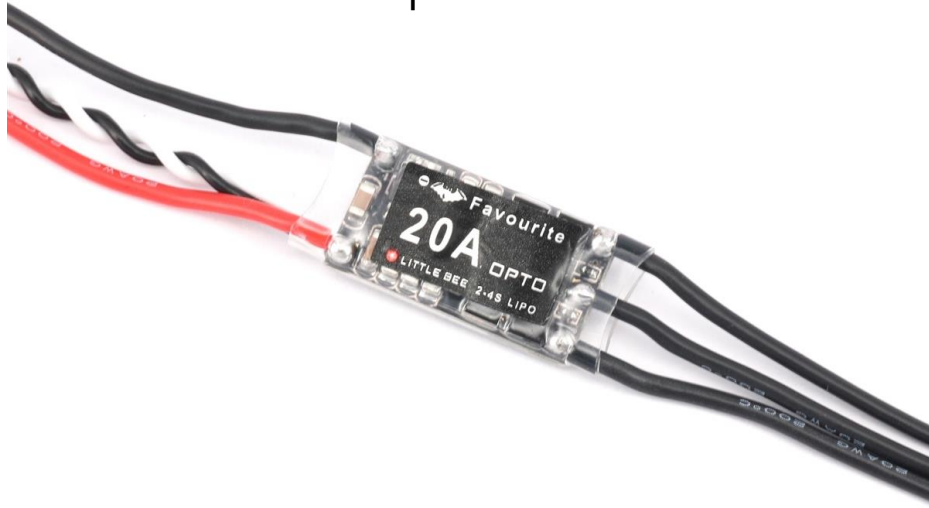
/***** Internal i2c Pullups *****/
/* enable internal I2C pull ups (in most cases it is better to use external pullups) */
//#define INTERNAL_I2C_PULLUPS

/***** constant loop time *****/
//#define LOOP_TIME 2000
```

#define MAXTHROTTLE 2000
Because you want Full Power sometimes!

MOTOR MIN THROTTLE

Electronic Speed Controller



Brushless Motor



Idle Speed
MINTHROTTLE 1150

MINCOMMAND 800 - 1100

PWM Frequency

MAXTHROTTLE 1850

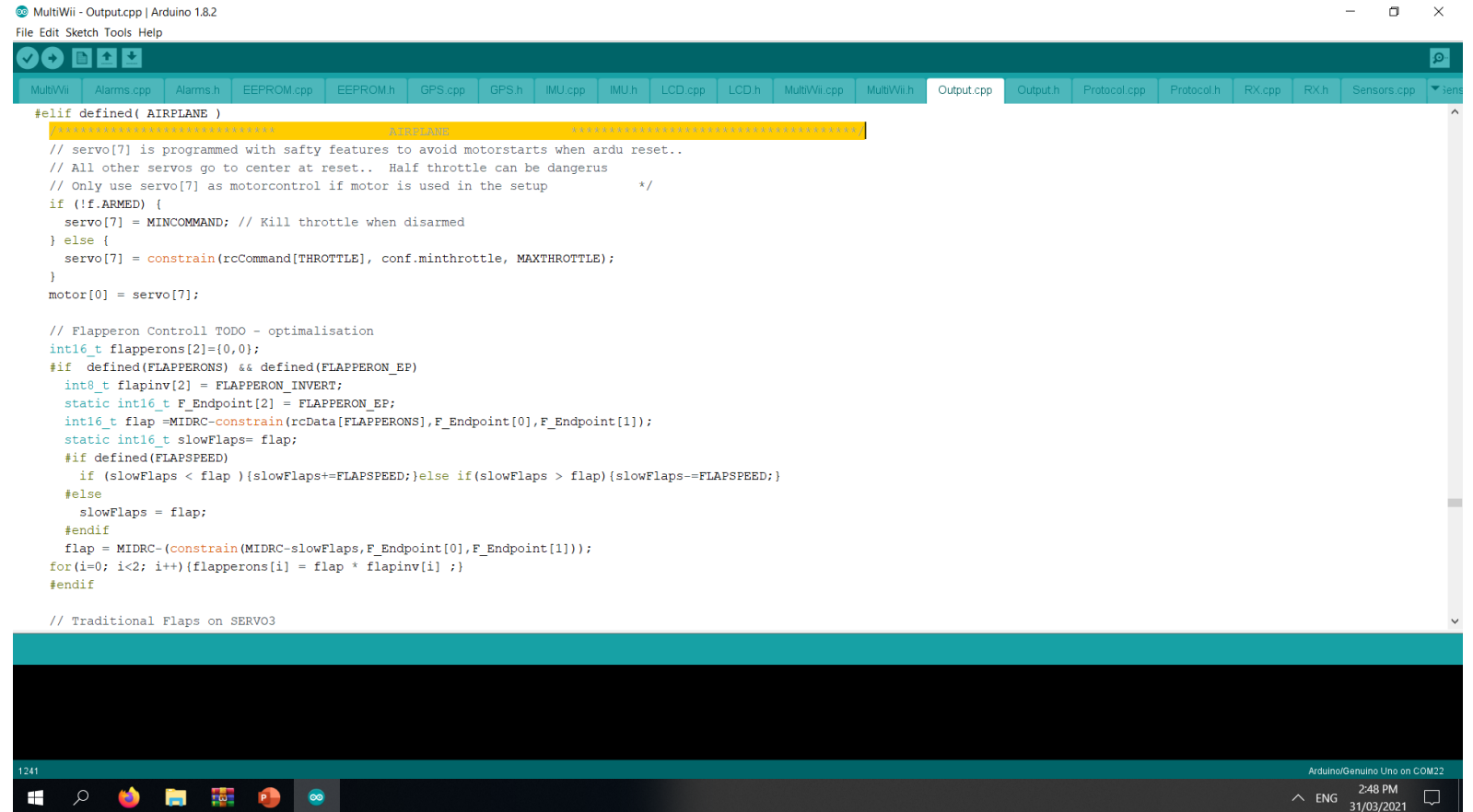
1900 - 2000

1500

LEGACY AIRPLANE

Legacy Airplane

- Servo reversing is done thru here apart those on the flywii GUI
- This also covers flaps and flapperons
- Flying wing / Vtail modes are also set in this area



```
#elif defined( AIRPLANE )
// servos are programmed with safty features to avoid motorstarts when ardu reset..
// All other servos go to center at reset.. Half throttle can be dangerous
// Only use servo[7] as motorcontrol if motor is used in the setup      */
if (!f.ARMED) {
    servo[7] = MINCOMMAND; // Kill throttle when disarmed
} else {
    servo[7] = constrain(rcCommand[THROTTLE], conf.minthrottle, MAXTHROTTLE);
}
motor[0] = servo[7];

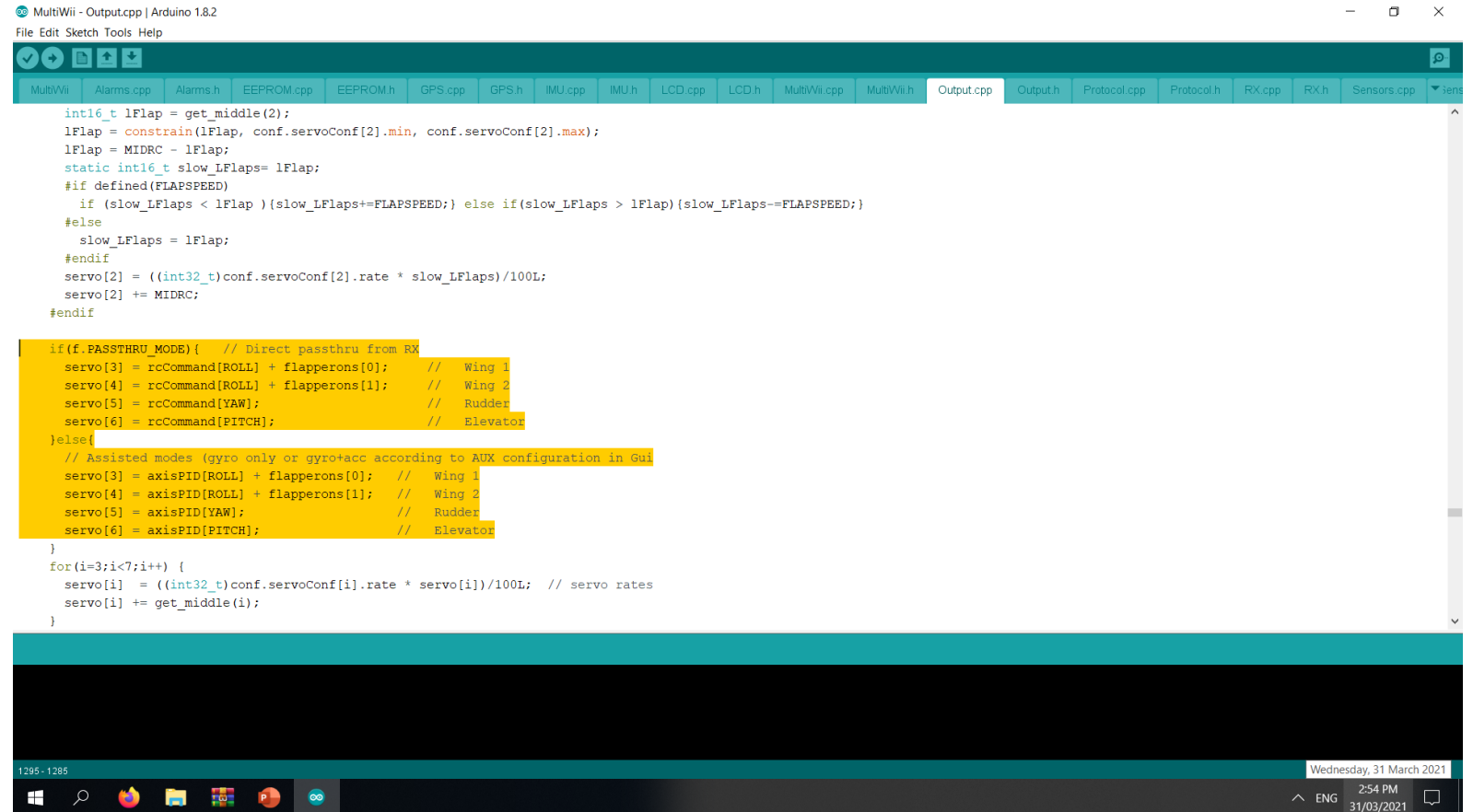
// Flapperon Controll TODO - optimisation
int16_t flapperons[2]={0,0};
#if defined(FLAPPERONS) && defined(FLAPPERON_EP)
    int8_t flapinv[2] = FLAPPERON_INVERT;
    static int16_t F_Endpoint[2] = FLAPPERON_EP;
    int16_t flap =MIDRC-constrain(rcData[FLAPPERONS],F_Endpoint[0],F_Endpoint[1]);
    static int16_t slowFlaps= flap;
    #if defined(FLAPSPEED)
        if (slowFlaps < flap ){slowFlaps+=FLAPSPEED;}else if(slowFlaps > flap){slowFlaps-=FLAPSPEED;}
    #else
        slowFlaps = flap;
    #endif
    flap = MIDRC-(constrain(MIDRC-slowFlaps,F_Endpoint[0],F_Endpoint[1]));
    for(i=0; i<2; i++){flapperons[i] = flap * flapinv[i] ;}
#endif

// Traditional Flaps on SERVO3
```

LEGACY AIRPLANE

Legacy Airplane

- Servo reversing is done thru here apart those on the flywii GUI
- This also covers flaps and flapperons
- Flying wing / Vtail modes are also set in this area



```
MultiWii - Output.cpp | Arduino 1.8.2
File Edit Sketch Tools Help

MultiWii Alarms.cpp Alarms.h EEPROM.cpp EEPROM.h GPS.cpp GPS.h IMU.cpp IMU.h LCD.cpp LCD.h MultiWii.cpp MultiWii.h Output.cpp Output.h Protocol.cpp Protocol.h RX.cpp RX.h Sensors.cpp Sensors.h

int16_t lFlap = get_middle(2);
lFlap = constrain(lFlap, conf.servoConf[2].min, conf.servoConf[2].max);
lFlap = MIDRC - lFlap;
static int16_t slow_LFlaps= lFlap;
#if defined (FLAPSPEED)
    if (slow_LFlaps < lFlap ) {slow_LFlaps+=FLAPSPEED;} else if (slow_LFlaps > lFlap) {slow_LFlaps-=FLAPSPEED;}
#else
    slow_LFlaps = lFlap;
#endif
servo[2] = ((int32_t)conf.servoConf[2].rate * slow_LFlaps)/100L;
servo[2] += MIDRC;
#endif

if(f.PASSTHRU_MODE){ // Direct passthru from RX
    servo[3] = rcCommand[ROLL] + flapperons[0]; // Wing 1
    servo[4] = rcCommand[ROLL] + flapperons[1]; // Wing 2
    servo[5] = rcCommand[YAW]; // Rudder
    servo[6] = rcCommand[PITCH]; // Elevator
}else{
    // Assisted modes (gyro only or gyro+acc according to AUX configuration in Gui)
    servo[3] = axisPID[ROLL] + flapperons[0]; // Wing 1
    servo[4] = axisPID[ROLL] + flapperons[1]; // Wing 2
    servo[5] = axisPID[YAW]; // Rudder
    servo[6] = axisPID[PITCH]; // Elevator
}
for(i=3;i<7;i++){
    servo[i] = ((int32_t)conf.servoConf[i].rate * servo[i])/100L; // servo rates
    servo[i] += get_middle(i);
}
```

Main Mix table should you need to custom mix your plane (Define {Bi})

Special Airplane

```
Synerduino_Airplane_Diff_thust-GY91-1.8.16 - Output.cpp | Arduino 1.8.18
File Edit Sketch Tools Help

1136 #if defined( MY_PRIVATE_MIXING )
1137 #include MY_PRIVATE_MIXING
1138 #elif defined( BI )
1139 //motor[0] = PIDMIX(+1, 0, 0): //LEFT
1140 //motor[1] = PIDMIX(-1, 0, 0): //RIGHT
1141 //servo[4] = (SERVODIR(4,2) * axisPID[YAW]) + (SERVODIR(4,1) * axisPID[PITCH]) + get_middle(4): //LEFT
1142 //servo[5] = (SERVODIR(5,2) * axisPID[YAW]) + (SERVODIR(5,1) * axisPID[PITCH]) + get_middle(5): //RIGHT
1143 //***** BI Airplane Elevator Aileron *****
1144 motor[0] = PIDMIX(0, 0, +1): //LEFT
1145 motor[1] = PIDMIX(0, 0, -1): //RIGHT
1146 servo[4] = (SERVODIR(4,2) * axisPID[PITCH]) + get_middle(4): //ELEVATOR
1147 servo[5] = (SERVODIR(5,2) * axisPID[ROLL]) + get_middle(5): //AILERON
1148 //***** BI Wing Elevon *****
1149 //motor[0] = PIDMIX(0, 0, +1): //LEFT
1150 //motor[1] = PIDMIX(0, 0, -1): //RIGHT
1151 //servo[4] = (SERVODIR(4,2) * axisPID[ROLL]) + (SERVODIR(4,1) * axisPID[PITCH]) + get_middle(4): //LEFT
1152 //servo[5] = (SERVODIR(5,2) * axisPID[ROLL]) + (SERVODIR(5,1) * axisPID[PITCH]) + get_middle(5): //RIGHT
1153
1154 #elif defined( TRI )
1155 motor[0] = PIDMIX( 0,+4/3, 0): //REAR
1156 motor[1] = PIDMIX(-1,-2/3, 0): //RIGHT
1157 motor[2] = PIDMIX(+1,-2/3, 0): //LEFT
1158 servo[5] = (SERVODIR(5, 1) * axisPID[YAW]) + get_middle(5): //REAR
1159 #elif defined( QUADP )
1160 motor[0] = PIDMIX( 0,+1,-1): //REAR
1161 motor[1] = PIDMIX(-1, 0,+1): //RIGHT
1162 motor[2] = PIDMIX(+1, 0,+1): //LEFT
1163 motor[3] = PIDMIX( 0,-1,-1): //FRONT
1164 #elif defined( QUADX )
1165 motor[0] = PIDMIX(-1,+1,-1): //REAR_R
1166 motor[1] = PIDMIX(-1,-1,+1): //FRONT_R
1167 motor[2] = PIDMIX(+1,+1,+1): //REAR_L
1168 motor[3] = PIDMIX(+1,-1,-1): //FRONT_L
1169 #elif defined( Y4 )
1170 motor[0] = PIDMIX(+0,+1,-1): //REAR_1_CW
1171 motor[1] = PIDMIX(-1,-1, 0): //FRONT_R_CCW
1172 motor[2] = PIDMIX(+0,+1,+1): //REAR_2_CCW
1173 motor[3] = PIDMIX(+1,-1, 0): //FRONT_L_CW
1174 #elif defined( Y6 )
1175 motor[0] = PIDMIX(+0,+4/3,+1): //REAR
1176 motor[1] = PIDMIX(-1,-2/3,-1): //RIGHT
1177 motor[2] = PIDMIX(+1,-2/3,-1): //LEFT
1178 motor[3] = PIDMIX(+0,+4/3,-1): //UNDER_REAR
1179 motor[4] = PIDMIX(-1 -2/3 -1) - //UNDER_RIGHT
1180
1181 #endif
```

Main Mix table should you need to custom mix your plane (Define {Bi})

Special Airplane

```
✓ → 📄 ⬆ ⬇
Synerduino_3CHAirplane-GY91-1.8.16  Alarms.cpp  Alarms.h  EEPROM.cpp  EEPROM.h  GPS.cpp  GPS.h  IMU.cpp  IMU.h  LCD.cpp  LCD.h  MultiWii.cpp  MultiWii.h  Output.cpp
1129 #if defined(DYNBALANCE)
1130     return;
1131 #endif
1132 #define PIDMIX(X,Y,Z) rcCommand[THROTTLE] + axisPID[ROLL]*X + axisPID[PITCH]*Y + YAW_DIRECTION * axisPID[YAW]*Z
1133 #define SERVODIR(n,b) ((conf.servoConf[n].rate & b) ? -1 : 1)
1134
1135 /***** main Mix Table *****/
1136 #if defined(MY_PRIVATE_MIXING)
1137 #include MY_PRIVATE_MIXING
1138 #elif defined(BI)
1139 //motor[0] = PIDMIX(+1, 0, 0); //LEFT
1140 //motor[1] = PIDMIX(-1, 0, 0); //RIGHT
1141 //servo[4] = (SERVODIR(4,2) * axisPID[YAW]) + (SERVODIR(4,1) * axisPID[PITCH]) + get_middle(4); //LEFT
1142 //servo[5] = (SERVODIR(5,2) * axisPID[YAW]) + (SERVODIR(5,1) * axisPID[PITCH]) + get_middle(5); //RIGHT
1143 /***** 3ch Airplane Throttle Elevator Rudder *****/
1144 motor[0] = PIDMIX(0, 0, 0); //LEFT
1145 motor[1] = PIDMIX(0, 0, 0); //RIGHT
1146 servo[4] = (SERVODIR(4, 1) * axisPID[YAW]) + get_middle(4); //RUDDER
1147 servo[5] = (SERVODIR(5, 1) * axisPID[PITCH]) + get_middle(5); //ELEVATOR
1148 /***** BI Airplane Elevator Ailron *****/
1149 //motor[0] = PIDMIX(0, 0, +1); //LEFT
1150 //motor[1] = PIDMIX(0, 0, -1); //RIGHT
1151 //servo[4] = (SERVODIR(4,1) * axisPID[ROLL]) + get_middle(4); //AILERON
1152 //servo[5] = (SERVODIR(5,1) * axisPID[PITCH]) + get_middle(5); //ELEVATOR
1153 /***** BI Wing Elevon *****/
1154 //motor[0] = PIDMIX(0, 0, +1); //LEFT
1155 //motor[1] = PIDMIX(0, 0, -1); //RIGHT
1156 //servo[4] = (SERVODIR(4,2) * axisPID[ROLL]) + (SERVODIR(4,1) * axisPID[PITCH]) + get_middle(4); //LEFT
1157 //servo[5] = (SERVODIR(5,2) * axisPID[ROLL]) + (SERVODIR(5,1) * axisPID[PITCH]) + get_middle(5); //RIGHT
-----
```

Special Airplane

Define custom selection of servo timers

/****** all the Mega types *****/

```

Synerduino_4CHAirplane-2-GY91-1.8.18  Alarms.cpp  Alarms.h  EEPROM.cpp  EEPROM.h  GPS.cpp  GPS.h  IMU.cpp  IMU.h  LCD.cpp  LCD.h  MultiWii.cpp  MultiWii.h  Output.cpp  Output.h  Protocol.cpp  Protocol.h  RX.cpp
605 #define AUX2PIN 5 //PIN 67 = PIN A13
606 #define AUX3PIN 6 //PIN 68 = PIN A14
607 #define AUX4PIN 7 //PIN 69 = PIN A15
608 #define V_BATPIN A0 // Analog PIN 0
609 #define PSensorPIN A2 // Analog PIN 2
610 #define PCINT_PIN_COUNT 8
611 #define PCINT_RX_BITS (1<<2), (1<<4), (1<<5), (1<<6), (1<<7), (1<<0), (1<<1), (1<<3)
612 #define PCINT_RX_PORT PORTK
613 #define PCINT_RX_MASK PCMSK2
614 #define PCIR_PORT_BIT (1<<2)
615 #define RX_PC_INTERRUPT PCINT2_vect
616 #define RX_PCINT_PIN_PORT PINK
617
618 #define SERVO_1_PINMODE pinMode(34,OUTPUT);pinMode(44,OUTPUT); // TILT_PITCH - WING left
619 #define SERVO_1_PIN_HIGH PORTC |= 1<<3;PORTL |= 1<<5;
620 #define SERVO_1_PIN_LOW PORTC &= ~(1<<3);PORTL &= ~(1<<5);
621 #define SERVO_2_PINMODE pinMode(35,OUTPUT);pinMode(45,OUTPUT); // TILT_ROLL - WING right
622 #define SERVO_2_PIN_HIGH PORTC |= 1<<2;PORTL |= 1<<4;
623 #define SERVO_2_PIN_LOW PORTC &= ~(1<<2);PORTL &= ~(1<<4);
624 #define SERVO_3_PINMODE pinMode(33,OUTPUT); pinMode(46,OUTPUT); // CAM TRIG - alt TILT_PITCH
625 #define SERVO_3_PIN_HIGH PORTC |= 1<<4;PORTL |= 1<<3;
626 #define SERVO_3_PIN_LOW PORTC &= ~(1<<4);PORTL &= ~(1<<3);
627 #define SERVO_4_PINMODE pinMode(37, OUTPUT);pinMode(7,OUTPUT); // // Spare output - alt TILT_ROLL
628 #define SERVO_4_PIN_HIGH PORTC |= 1<<0; PORTH |= 1<<4;
629 #define SERVO_4_PIN_LOW PORTC &= ~(1<<0);PORTH &= ~(1<<4);
630
631 #define SERVO_5_PINMODE pinMode(6,OUTPUT); // BI LEFT
632 #define SERVO_5_PIN_HIGH PORTH |= 1<<3;
633 #define SERVO_5_PIN_LOW PORTH &= ~(1<<3);
634 #define SERVO_6_PINMODE pinMode(2,OUTPUT); // TRI REAR - BI RIGHT
635 #define SERVO_6_PIN_HIGH PORTE |= 1<<4;
636 #define SERVO_6_PIN_LOW PORTE &= ~(1<<4);
637 #define SERVO_7_PINMODE pinMode(5,OUTPUT); // new
638 #define SERVO_7_PIN_HIGH PORTE |= 1<<3;

```

Arduino Version: 1.8.18

INERTIAL MEASURING UNIT MEASURING UNIT

Pls see the Board Specs
Data sheets for the installed
IMUs onboard

This is the heart of every flight controller AKA the
Main 4 ,

Gyro – stabilization on Roll Pitch Yaw Axis

Acc - Horizontal and Vertical stabilization XYZ

Baro – Altitude hold control

Mag – Heading and Compass

Each sensor has a corresponding address registry set
by manufacturer

You can find it on sensors.ccp tab

**Sensors work best if mounted as close to CG
as possible.**



Magnetometer



Barometer



Accelerometer



Gyroscope

INERTIAL MEASURING UNIT MEASURING UNIT

Config.h

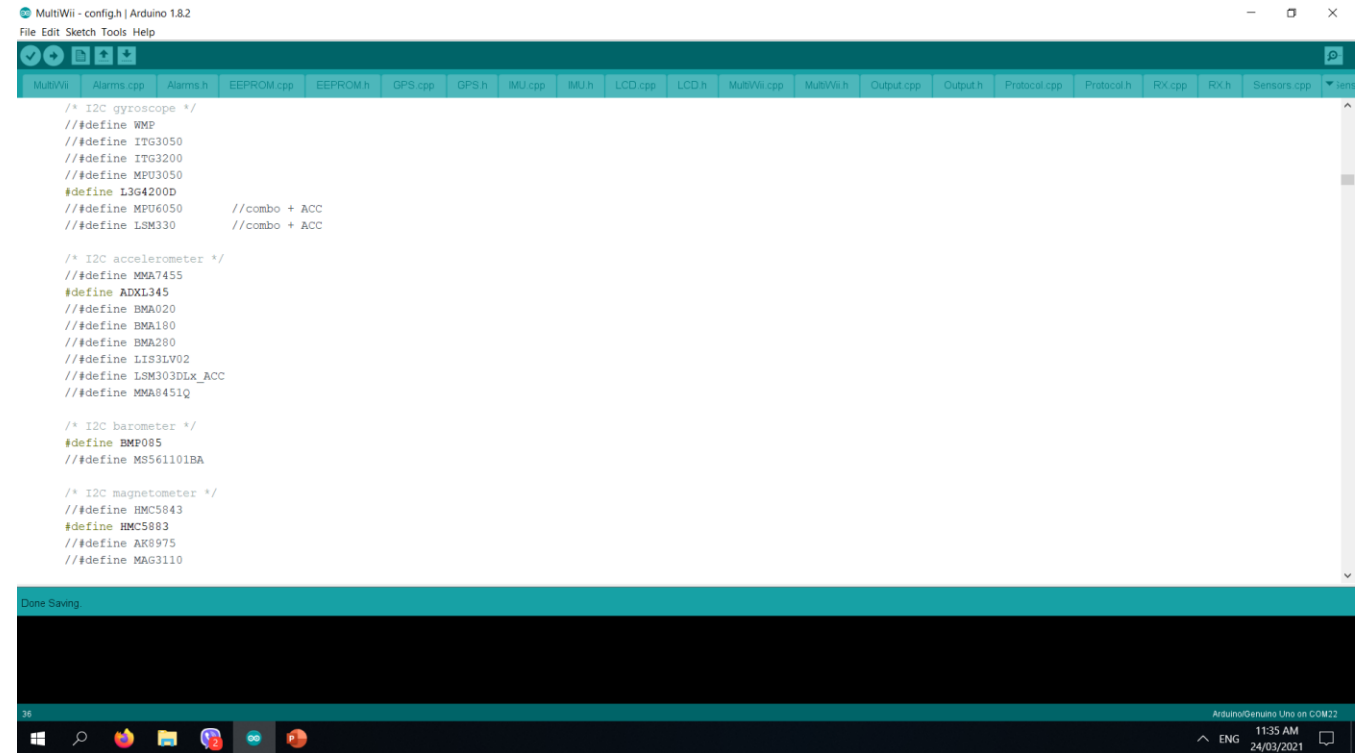
Sensors

#define L3G4200D

#define ADXL345

#define BMP085

#define MMC5883



```
MultiWii - config.h | Arduino 1.8.2
File Edit Sketch Tools Help

MultiWii Alarms.cpp Alarms.h EEPROM.cpp EEPROM.h GPS.cpp GPS.h IMU.cpp IMU.h LCD.cpp LCD.h MultiWii.cpp MultiWii.h Output.cpp Output.h Protocol.cpp Protocol.h RX.cpp RX.h Sensors.cpp Sensors.h

/* I2C gyroscope */
#define WMP
#define ITG3050
#define ITG3200
#define MPU3050
#define L3G4200D
#define MPU6050 //combo + ACC
#define LSM330 //combo + ACC

/* I2C accelerometer */
#define MMA7455
#define ADXL345
#define BMA020
#define BMA180
#define BMA280
#define LIS3LV02
#define LSM303DLX_ACC
#define MMA8451Q

/* I2C barometer */
#define BMP085
#define MS561101BA

/* I2C magnetometer */
#define HMC5843
#define MMC5883
#define AK8975
#define MAG3110
```

CONFIG.H



INERTIAL MEASURING UNIT MEASURING UNIT

MultiWii - config.h | Arduino 1.8.2

File Edit Sketch Tools Help

MultiWii Alarms.cpp Alarms.h EEPROM.cpp EEPROM.h GPS.cpp GPS.h IMU.cpp IMU.h LCD.cpp LCD.h MultiWii.cpp MultiWii.h Output.cpp Output.h Protocol.cpp Protocol.h RX.cpp RX.h Sensors.cpp

```

/***** Independent sensor *****/
/* leave it commented if you already checked a specific board above */
/* I2C gyroscope */
// #define WMP
// #define ITG3050
// #define ITG3200
// #define MPU3050
#define L3G4200D
// #define MPU6050 // combo + ACC
// #define LSM330 // combo + ACC

/* I2C accelerometer */
// #define MMA7455
#define ADXL345
// #define BMA020
// #define BMA180
// #define BMA280
// #define LIS3LV02
// #define LSM303DLx_ACC
// #define MMA8451Q

/* I2C barometer */
#define BMP085
// #define MS561101BA

/* I2C magnetometer */
// #define HMC5843
#define HMC5883

```

163

Arduino/Genuino Uno on COM41

9:28 PM 13/02/2020

Depending on the Version of ic2 sensors installed on the board
you got select appropriately for it to work

SENSORS.CCP



INERTIAL MEASURING UNIT MEASURING UNIT

```
MultiWii - Sensors.cpp | Arduino 1.8.2
File Edit Sketch Tools Help

MultiWii Alarms.cpp Alarms.h EEPROM.cpp EEPROM.h GPS.cpp GPS.h IMU.cpp IMU.h LCD.cpp LCD.h MultiWii.cpp MultiWii.h Output.cpp Output.h Protocol.cpp Protocol.h RX.cpp RX.h Sensors.cpp

#endif

// *****
// I2C Gyroscope L3G4200D
// *****
#ifdef L3G4200D
#define L3G4200D_ADDRESS 0x69
void Gyro_init() {
    delay(100);
    i2c_writeReg(L3G4200D_ADDRESS, 0x20, 0x8F); // CTRL_REG1 400Hz ODR, 20hz filter, run!
    delay(5);
    i2c_writeReg(L3G4200D_ADDRESS, 0x24, 0x02); // CTRL_REG5 low pass filter enable
    delay(5);
    i2c_writeReg(L3G4200D_ADDRESS, 0x23, 0x30); // CTRL_REG4 Select 2000dps
}

void Gyro_getADC () {
    i2c_getSixRawADC(L3G4200D_ADDRESS, 0x80|0x28);

    GYRO_ORIENTATION( ((rawADC[1]<<8) | rawADC[0])>>2 ,
                      ((rawADC[3]<<8) | rawADC[2])>>2 ,
                      ((rawADC[5]<<8) | rawADC[4])>>2 );

    GYRO_Common();
}
#endif

// *****
// I2C Gyroscope ITG3200 / ITG3205 / ITG3050 / MPU3050
```

851 Arduino/Genuino Uno on COM41 9:37 PM 13/02/2020

SENSORS.CCP



INERTIAL MEASURING UNIT MEASURING UNIT

MultiWii - Sensors.cpp | Arduino 1.8.2

File Edit Sketch Tools Help

```
// I2C Accelerometer ADXL345
// *****
// I2C address: 0x3A (8bit) 0x1D (7bit)
// Resolution: 10bit (Full range - 14bit, but this is autoscaling 10bit ADC to the range +/- 16g)
// principle:
// 1) CS PIN must be linked to VCC to select the I2C mode
// 2) SD0 PIN must be linked to VCC to select the right I2C address
// 3) bit b00000100 must be set on register 0x2D to read data (only once at the initialization)
// 4) bits b00001011 must be set on register 0x31 to select the data format (only once at the initialization)
// *****
#ifdef ADXL345
#ifndef ADXL345_ADDRESS
#define ADXL345_ADDRESS 0x1D
#define ADXL345_ADDRESS 0x53 //WARNING: Conflicts with a Wii Motion plus!
#endif

void ACC_init () {
    delay(10);
    i2c_writeReg(ADXL345_ADDRESS, 0x2D, 1<<3); // register: Power CTRL -- value: Set measure bit 3 on
    i2c_writeReg(ADXL345_ADDRESS, 0x31, 0x0B); // register: DATA_FORMAT -- value: Set bits 3(full range) and 1 0 on (+/- 16g-range)
    i2c_writeReg(ADXL345_ADDRESS, 0x2C, 0x09); // register: BW_RATE -- value: rate=50hz, bw=20hz
}

void ACC_getADC () {
    i2c_getSixRawADC(ADXL345_ADDRESS, 0x32);

    ACC_ORIENTATION( ((rawADC[1]<<8) | rawADC[0]) ,
                     ((rawADC[3]<<8) | rawADC[2]) ,
```

642

Arduino/Genuino Uno on COM41

SENSORS.CCP



INERTIAL MEASURING UNIT MEASURING UNIT

MultiWii - Sensors.cpp | Arduino 1.8.2

File Edit Sketch Tools Help

```
// *****
// I2C Barometer BOSCH BMP085
// *****
// I2C adress: 0x77 (7bit)
// principle:
// 1) read the calibration register (only once at the initialization)
// 2) read uncompensated temperature (not mandatory at every cycle)
// 3) read uncompensated pressure
// 4) raw temp + raw pressure => calculation of the adjusted pressure
// the following code uses the maximum precision setting (oversampling setting 3)
// *****

#ifdef BMP085
#define BMP085_ADDRESS 0x77

static struct {
    // sensor registers from the BOSCH BMP085 datasheet
    int16_t ac1, ac2, ac3;
    uint16_t ac4, ac5, ac6;
    int16_t b1, b2, mb, mc, md;
    union {uint16_t val; uint8_t raw[2]; } ut; //uncompensated T
    union {uint32_t val; uint8_t raw[4]; } up; //uncompensated P
    uint8_t state;
    uint32_t deadline;
} bmp085_ctx;
#define OSS 3

/* transform a series of bytes from big endian to little
```

331

Arduino/Genuino Uno on COM41

SENSORS.CPP



INERTIAL MEASURING UNIT MEASURING UNIT

MultiWii - Sensors.cpp | Arduino 1.8.2

File Edit Sketch Tools Help

```
MultiWii | Alarms.cpp | Alarms.h | EEPROM.cpp | EEPROM.h | GPS.cpp | GPS.h | IMU.cpp | IMU.h | LCD.cpp | LCD.h | MultiWii.cpp | MultiWii.h | Output.cpp | Output.h | Protocol.cpp | Protocol.h | RX.cpp | RX.h | Sensors.cpp | Sensors.h

// I2C Compass MMC5883
// *****
// I2C address: 0x30 (8bit) 0x00 (7bit)
// *****

#if defined(MMC5883)

#define MAG_ADDRESS 0x30
#define MAG_DATA_REGISTER 0x00 //Read register address
//REG CONTROL
#define MMC5883MA_OUT 0x00
#define MMC5883MA_XOUT 0x00
#define MMC5883MA_XOUT_LOW 0x00
#define MMC5883MA_XOUT_HIGH 0x01
#define MMC5883MA_YOUT 0x02
#define MMC5883MA_YOUT_LOW 0x02
#define MMC5883MA_YOUT_HIGH 0x03
#define MMC5883MA_ZOUT 0x04
#define MMC5883MA_ZOUT_LOW 0x04
#define MMC5883MA_ZOUT_HIGH 0x05
#define MMC5883MA_TEMPERATURE 0x06
#define MMC5883MA_STATUS 0x07
#define MMC5883MA_INTERNAL_CONTROL_0 0x08
#define MMC5883MA_INTERNAL_CONTROL_1 0x09
#define MMC5883MA_INTERNAL_CONTROL_2 0x0A
#define MMC5883MA_X_THRESHOLD 0x0B
#define MMC5883MA_Y_THRESHOLD 0x0C
#define MMC5883MA_Z_THRESHOLD 0x0D

1264 - 1265 Arduino/Genuino Uno on COM22
ENG 3:22 PM 31/03/2021
```

SENSOR ORIENTATION

CONFIG.H

[illegible]

SENSORS IMU ORIENTATION IS IMPORTANT SEE TO IT THE ACC GYRO AND MAG ALL COMPLIMENT EACH OTHER

RECEIVER TYPES PROTOCOL

CONFIG.H



For SBUS Receiver

Channel Mapping

Uncomment SBUS
on RX Serial Port 1
(Telemetry 1)

You may need to
uncomment and
change the ordering
of your channel
depending on your
Transmitter's model
and specification

```
SynerduinoKwad3-GY91-1.8.16-sbus - config.h | Arduino 1.8.18
File Edit Sketch Tools Help

SynerduinoKwad3-GY91-1.8.16-sbus | Alarms.cpp | Alarms.h | EEPROM.cpp | EEPROM.h | GPS.cpp | GPS.h | IMU.cpp | IMU.h | LCD.cpp | LCD.h | MultiWii.cpp | MultiWii.h | Output.cpp | Output.h | Protocol.cpp | Protocol.h | RX.cpp | RX.h

392 //*****
393 // Defines that allow a "Bind" of a Spektrum or Compatible Remote Receiver (aka Satellite) via Configuration GUI.
394 // Bind mode will be same as declared above, if your TX is capable.
395 // Ground, Power, and Signal must come from three adjacent pins.
396 // By default, these are Ground=4, Power=5, Signal=6. These pins are in a row on most MultiWii shield boards. Pins can be overridden below.
397 // Normally use 3.3V regulator is needed on the power pin!! If your satellite hangs during bind (blinks, but won't complete bind with a solid light), go direct 5V on all pins.
398 //*****
399 // For Pro Mini, the connector for the Satellite that resides on the FTDI can be unplugged and moved to these three adjacent pins.
400 //define SPEK_BIND //Un-Comment for Spektrum Satellite Bind Support. Code is ~420 bytes smaller without it.
401 //define SPEK_BIND_GROUND 4
402 //define SPEK_BIND_POWER 5
403 //define SPEK_BIND_DATA 6
404
405 //***** SBUS RECIVER *****/
406 /* The following line apply only for Futaba S-Bus Receiver on MEGA boards or PROMICRO boards.
407 You have to invert the S-Bus-Serial Signal e.g. with a Hex-Inverter like IC SN74 LS 04 */
408 //define SBUS PITCH,YAW,THROTTLE,ROLL,AUX1,AUX2,AUX3,AUX4,8,9,10,11,12,13,14,15,16,17 // dsm2 orangerx
409 //define SBUS ROLL,PITCH,THROTTLE,YAW,AUX1,AUX2,AUX3,AUX4,8,9,10,11,12,13,14,15,16,17 // T148G
410 //define RX_SERIAL_PORT 1
411 //define SBUS_MID_OFFSET 988 //SBUS Mid-Point at 1500
412
413 //***** HOTT RECIVER *****/
414 /* Graupner Hott HD */
415 //define SUMD PITCH,YAW,THROTTLE,ROLL,AUX1,AUX2,AUX3,AUX4
416 //define RX_SERIAL_PORT 1
417
418 //*****
419 //*****
420 //***** SECTION 4 - ALTERNATE CPUs & BOARDS *****
421 //*****
422 //*****
423
424
```

Done Saving.

avrdude done. Thank you.

410 - 409

Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM17

Links ENG 8:48 PM 24/07/2022



AUX PIN

For UNO you need to
define your PPM Aux 2
Pin

```
MultiWii - config.h | Arduino 1.8.2
File Edit Sketch Tools Help

MultiWii Alarms.cpp Alarms.h EEPROM.cpp EEPROM.h GPS.cpp GPS.h IMU.cpp IMU.h LCD.cpp LCD.h MultiWii.cpp MultiWii.h Output.cpp Output.h Protocol.cpp Protocol.h RX.cpp RX.h Sensors.cpp Sensors.h

/***** Promini Specific Settings *****/
/***** Hexa Motor 5 & 6 Pins *****/
/* PIN A0 and A1 instead of PIN D5 & D6 for 6 motors config and promini config
   This mod allow the use of a standard receiver on a pro mini
   (no need to use a PPM sum receiver) */
#define A0_A1_PIN_HEX

/***** AUX 2 Pin *****/
/* possibility to use PIN8 or PIN12 as the AUX2 RC input (only one, not both)
   it deactivates in this case the POWER PIN (pin 12) or the BUZZER PIN (pin 8) */
#define RCAUXPIN8
#define RCAUXPIN12

/***** Teensy 2.0 Support *****/
/* uncomment this if you use a teensy 2.0 with teensyduino
   it needs to run at 16MHz */
#define TEENSY20

/***** Settings for ProMicro, Leonardo and other Atmega32u4 Boards *****/
```

427 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM30 11:12 AM 11/01/2021

TELEMETRY COM SPEED

CONFIG.H

SERIAL BAUD RATE

WILL DEPEND ON WHAT BAUD YOUR TELEMETRY MODULE ARE SET INTO YOU CAN CHANGE IT TO SUITE THE PORT YOUR DEVICE IS CONNECTED TO.

SERIAL 0 CAN BE USE FOR TELEMETRY GIVEN NOTHING IS CONNECTED TO THE USB AT THIS POINT. AND FIRMWARE MUST BE FLISH PRIOR TO HOOKING UP ANYTHING TO THIS PINS

SERIAL 1 , 3 IS RESERVE FOR TELEMETRY

115200 FOR BLUETOOTH HC-05

38400 FOR XBEE RADIO

57600 for SIK RADIO

SERIAL 2 IS RESERVE FOR GPS

NMEA Baud 57600

```
SynerduinoKwad3-GY801-InvertedMag-1.8.16-M9-10 - config.h | Arduino 1.8.18
File Edit Sketch Tools Help

SynerduinoKwad3-GY801-InvertedMag-1.8.16-M9-10  Alarms.cpp  Alarms.h  EEPROM.cpp  EEPROM.h  GPS.cpp  GPS.h  IMU.cpp  IMU.h  LCD.cpp  LCD.h  MultiWii.cpp  MultiWii.h  Output.cpp  Output.h  Protocol.cpp  Proto...

509 /***** SECTION 5 - ALTERNATE SETUP *****/
510 /*****
511 /*****
512
513 /**** Serial com speed *****/
514 /* This is the speed of the serial interfaces */
515 /*Serial 1 Bluetooth 115200 */
516 /*Serial 2 GPS */
517 /*Serial 3 SIK/Xbee 38400*/
518 /* This is the speed of the serial interfaces */
519 #define SERIAL0_COM_SPEED 115200
520 // #define SERIAL3_COM_SPEED 115200
521 #define SERIAL2_COM_SPEED 115200
522 // #define SERIAL1_COM_SPEED 115200
523
524 /*Serial 1 Bluetooth Serial 2 GPS Serial 3 Telemetry*/
525 #define SERIAL3_COM_SPEED 38400
526 #define SERIAL1_COM_SPEED 57600
527
528
529 /* when there is an error on I2C bus, we neutralize the values during a short time. expressed in microseconds
530 it is relevent only for a conf with at least a WMP */
531 #define NEUTRALIZE_DELAY 100000
532
533 /*****
534 /***** Gyro filters *****/
535 /*****
536
537 /***** Lowpass filter for some gyros *****/
538 /* ITG3200 & ITG3205 Low pass filter setting. In case you cannot eliminate all vibrations to the Gyro, you can try
539 to decrease the LPF frequency, only one step per try. As soon as twitching gone, stick with that setting.
540 It will not help on feedback wobbles, so change only when copter is randomly twitching and all dampening and
```



MultiWii - config.h | Arduino 1.8.2
File Edit Sketch Tools Help

MultiWii Alarms.cpp Alarms.h EEPROM.cpp EEPROM.h GPS.cpp GPS.h IMU.cpp IMU.h LCD.cpp LCD.h MultiWii.cpp MultiWii.h Output.cpp Output.h Protocol.cpp Protocol.h RX.cpp RX.h Sensors.cpp

```
/* introduce a deadband around the stick center
Must be greater than zero, comment if you dont want a deadband on roll, pitch and yaw */
#define DEADBAND 6

GPS

/* Enable this for using GPS simulator (NMEA only)*/
#define GPS_SIMULATOR

/* GPS using a SERIAL port
if enabled, define here the Arduino Serial port number and the UART speed
note: only the RX PIN is used in case of NMEA mode, the GPS is not configured by multiwii
in NMEA mode the GPS must be configured to output GGA and RMC NMEA sentences (which is generally the default conf for most GPS devices)
at least 5Hz update rate. uncomment the first line to select the GPS serial port of the arduino */

#define GPS_SERIAL 2 // should be 2 for flyduino v2. It's the serial port number on arduino MEGA
// must be 0 for PRO_MINI (ex GPS_PRO_MINI)
// note: Now a GPS can share MSP on the same port. The only constrain is to not use it simultaneously, and use the same port speed.

// avoid using 115200 baud because with 16MHz arduino the 115200 baudrate have more than 2% speed error (57600 have 0.8% error)
#define GPS_BAUD 38400 // ublox 8 new standard
#define GPS_BAUD 9600
#define GPS_BAUD 57600 // GPS_BAUD will override SERIALx_COM_SPEED for the selected port (my ublox 6)
#define GPS_BAUD 115200
```

Done Saving

676 Arduino/Genuine Uno on COM41

Windows taskbar: 4:05 PM, 20/02/2020

**Set GPS BAUD to 57600
Or its matching baud
as configured to the
GPS module**



SynerduinoKwad3-GY91-1.8.16-M9-10 - config.h | Arduino 1.8.18

File Edit Sketch Tools Help

SynerduinoKwad3-GY91-1.8.16-M9-10 Alarms.cpp Alarms.h EEPROM.cpp EEPROM.h GPS.cpp GPS.h IMU.cpp IMU.h LCD.cpp LCD.h MultiWii.cpp MultiWii.h Output.cpp Output.h Protocol.cpp Protocol.h RX.cpp RX.h

```

692 // must be 0 for PRO_MINI / UNO (ex GPS_PRO_MINI) - GPS must be disconnected when using USB Serial
693 // note: Now a GPS can share MSP on the same port. The only constrain is to not use it simultaneously with the MSP
694
695 // avoid using 115200 baud because with 16MHz arduino the 115200 baudrate have more than 2% speed error (57600 have 0.8% error)
696 // #define GPS_BAUD 9600 // GPS default
697 // #define GPS_BAUD 38400
698 #define GPS_BAUD 57600 // GPS_BAUD will override SERIALx_COM_SPEED for the selected port
699 // #define GPS_BAUD 115200
700
701 /* GPS protocol
702  NMEA - Standard NMEA protocol GGA, GSA and RMC sentences are needed
703  UBLOX - U-Blox binary protocol, use the ublox config file (u-blox-config.ublox.txt) from the source tree
704  MTK_BINARY16 and MTK_BINARY19 - MTK3329 chipset based GPS with DIYDrones binary firmware (v1.6 or v1.9)
705  With UBLOX and MTK_BINARY you don't have to use GPS_FILTERING in multiwii code !!!
706
707  SEE UCENTER https://www.u-blox.com/en/product/u-center */
708
709
710 // #define NMEA //for Ublox NMEA GPS - NMEA Protocol Pls. sea GPS.h M5-M6 Protocol M8-M10 Protocol uncomment
711 #define UBLOX //for Beitian GPS - UBLX Protocol
712 // #define MTK_BINARY16
713 // #define MTK_BINARY19
714 // #define INIT_MTK_GPS // initialize MTK GPS for using selected speed, 5Hz update rate and GGA & RMC sentence of binary protocols
715
716
717 /* I2C GPS device made with an independant arduino + GPS device
718  including some navigation functions
719  contribution from EOSBandi http://code.google.com/p/i2c-gps-nav/
720  You have to use at least I2CGpsNav code r33 */
721 /* all fonctionnalities allowed by SERIAL_GPS are now available for I2C_GPS: all relevant navigation computations are gathered in the main FC */
722
723 // #define I2C_GPS

```

711

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM83

Depending on the GPS Version

Older formats uses NMEA protocol

Newer GPS uses UBLOX UBX Protocol

Support:
M6-M8 UBLOX
M8-M10 NMEA (Flash Required)

BATTERY MONITORING

CONFIG.H



```
MultiWii - config.h | Arduino 1.8.2
File Edit Sketch Tools Help

MultiWii | Alarms.cpp | Alarms.h | EEPROM.cpp | EEPROM.h | GPS.cpp | GPS.h | IMU.cpp | IMU.h | LCD.cpp | LCD.h | MultiWii.cpp | MultiWii.h | Output.cpp | Output.h | Protocol.cpp | Protocol.h | RX.cpp | RX.h | Sensors.cpp | Sensors.h

/***** battery voltage monitoring *****/
/* for V BAT monitoring
after the resistor divisor we should get [0V;5V]->[0;1023] on analog V_BATPIN
with R1=33k and R2=51k
vbat = [0;1023]*16/VBATSCALE
must be associated with #define BUZZER ! */
//#define VBAT // uncomment this line to activate the vbat code
#define VBATSCALE 131 // (*) (**) change this value if readed Battery voltage is different than real voltage
#define VBATNOMINAL 126 // 12,6V full battery nominal voltage - only used for lcd.telemetry
#define VBATLEVEL_WARN1 107 // (*) (**) 10,7V
#define VBATLEVEL_WARN2 99 // (*) (**) 9.9V
#define VBATLEVEL_CRIT 93 // (*) (**) 9.3V - critical condition: if vbat ever goes below this value, permanent alarm is triggered
#define NO_VBAT 16 // Avoid beeping without any battery
#define VBAT_OFFSET 18 // offset in 0.1Volts, gets added to voltage value - useful for zener diodes

/* for V BAT monitoring of individual cells
* enable both VBAT and VBAT_CELLS
*/
//#define VBAT_CELLS
#define VBAT_CELLS_NUM 0 // set this to the number of cells you monitor via analog pins
#define VBAT_CELLS_PINS {A0, A1, A2, A3, A4, A5 } // set this to the sequence of analog pins
#define VBAT_CELLS_OFFSETS {0, 50, 83, 121, 149, 177 } // in 0.1 volts, gets added to voltage value - useful for zener diodes
#define VBAT_CELLS_DIVS { 75, 122, 98, 18, 30, 37 } // divisor for proportional part according to resistors - larger value here gives smaller voltage

/***** powermeter (battery capacity monitoring) *****/

892 Arduino/Genuino Uno on COM41
Windows taskbar: 10:24 PM 13/02/2020
```

PIN ASSIGNMENTS

CONFIG.H



```
MultiWii - config.h | Arduino 1.8.2
File Edit Sketch Tools Help

MultiWii Alarms.cpp Alarms.h EEPROM.cpp EEPROM.h GPS.cpp GPS.h IMU.cpp IMU.h LCD.cpp LCD.h MultiWii.cpp MultiWii.h Output.cpp Output.h Protocol.cpp Protocol.h RX.cpp RX.h Sensors.cpp Sens

/***** Buzzer Pin *****/
/* this moves the Buzzer pin from TX0 to D8 for use with ppm sum or spectrum sat. RX (not needed if A32U4ALLPINS is active) */
// #define D8BUZZER

/***** Promicro version related *****/
/* Inverted status LED for Promicro ver 10 */
// #define PROMICRO10

/***** override default pin assignments *****/

/* only enable any of this if you must change the default pin assignment, e.g. your board does not have a specific pin */
/* you may need to change PINx and PORTx plus #shift according to the desired pin! */
#define OVERRIDE_V_BATPIN A0 // instead of A3 // Analog PIN 3

// #define OVERRIDE_PSENSORPIN A1 // instead of A2 // Analog PIN 2

// #define OVERRIDE_LEDPIN_PINMODE pinMode (A1, OUTPUT); // use A1 instead of d13
// #define OVERRIDE_LEDPIN_TOGGLE PINC |= 1<<1; // PINB |= 1<<5; //switch LEDPIN state (digital PIN 13)
// #define OVERRIDE_LEDPIN_OFF PORTC &= ~(1<<1); // PORTB &= ~(1<<5);
// #define OVERRIDE_LEDPIN_ON PORTC |= 1<<1; // was PORTB |= (1<<5);

// #define OVERRIDE_BUZZERPIN_PINMODE pinMode (A2, OUTPUT); // use A2 instead of d8
// #define OVERRIDE_BUZZERPIN_ON PORTC |= 1<<2 //PORTB |= 1;
// #define OVERRIDE_BUZZERPIN_OFF PORTC &= ~(1<<2); //PORTB &= ~1;

/*****

Done Saving.

479 Arduino/Genuino Uno on COM41
9:30 AM 21/02/2020
```



CONFIG.H

Arm/DisArm

Option for combination stick command to start and stop the drone

Note: Combination stick only works on some vehicles configuration . Others uses Aux Arm switch

Arm Only When Flat

Is a safety option not to arm when the drone is not level prevent starting up when on a slope or when moving

MultiWii - config.h | Arduino 1.8.2

File Edit Sketch Tools Help

MultiWii Alarms.cpp Alarms.h EEPROM.cpp EEPROM.h GPS.cpp GPS.h IMU.cpp IMU.h LCD.cpp LCD.h MultiWii.cpp MultiWii.h Output.cpp Output.h Protocol.cpp Protocol.h RX.cpp RX.h Sensors.cpp Sensors.h

```
/* NEW: not used anymore for servo coptertypes <== NEEDS FIXING - MOVE TO WIKI */
#define YAW_DIRECTION 1
//#define YAW_DIRECTION -1 // if you want to reverse the yaw correction direction

#define ONLYARMWHENFLAT //prevent the copter from arming when the copter is tilted

/***** ARM/DISARM *****/
/* optionally disable stick combinations to arm/disarm the motors.
 * In most cases one of the two options to arm/disarm via TX stick is sufficient */
#define ALLOW_ARM_DISARM_VIA_TX_YAW
//#define ALLOW_ARM_DISARM_VIA_TX_ROLL

/***** SERVO *****/
/* info on which servos connect where and how to setup can be found here
 * http://www.mutiwii.com/wiki/index.php?title=Config.h#Servos\_configuration
 */

/* Do not move servos if copter is unarmed
 * It is a quick hack to overcome feedback tail wigglight when copter has a flexible
 * landing gear
 */
//#define DISABLE_SERVOS_WHEN_UNARMED

/* if you want to preset min/middle/max values for servos right after flashing, because of limited physical
 * room for servo travel, then you must enable and set all three following options */
//#define SERVO_MIN {1020, 1020, 1020, 1020, 1020, 1020, 1020, 1020, 1020}
//#define SERVO_MAX {2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000}
//#define SERVO_MTD {1500, 1500, 1500, 1500, 1500, 1500, 1500, 1500, 1500} // (*)
```

234

Generic ESP8266 Module, 80 MHz, Flash, ck, 26 MHz, 40MHz, QIO, 512K (no SPIFFS), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM24

8:32 PM 12/08/2020

Works for Multirotor Example : Rudder Stick Left to Rudder Arm Stick Right to Disarm
Note: motors will spool up to Idle Speed

CONFIG.H



```
SynerduinoKwad - config.h | Arduino 1.8.2
File Edit Sketch Tools Help

SynerduinoKwad | Alarms.cpp | Alarms.h | EEPROM.cpp | EEPROM.h | GPS.cpp | GPS.h | IMU.cpp | IMU.h | LCD.cpp | LCD.h | MultiWii.cpp | MultiWii.h | Output.cpp | Output.h | Protocol.cpp | Protocol.h | RX.cpp | RX.h | Senc...cpp

A Value on 200 will give a very distinct transfer */
// #define ACROTRAINER_MODE 200 // http://www.multiwii.com/forum/viewtopic.php?f=16&t=1944#p17437

/*****                               *****/
/* Failsafe check pulses on four main control channels CH1-CH4. If the pulse is missing or bellow 985us (on any of these four channels)
the failsafe procedure is initiated. After FAILSAFE_DELAY time from failsafe detection, the level mode is on (if ACC is available),
PITCH, ROLL and YAW is centered and THROTTLE is set to FAILSAFE_THROTTLE value. You must set this value to descending about 1m/s or so
for best results. This value is depended from your configuration, AUW and some other params. Next, after FAILSAFE_OFF_DELAY the copter is disarmed,
and motors is stopped. If RC pulse coming back before reached FAILSAFE_OFF_DELAY time, after the small quard time the RC control is returned to normal. */
#define FAILSAFE // uncomment to activate the failsafe function
#define FAILSAFE_DELAY 10 // Guard time for failsafe activation after signal lost. 1 step = 0.1sec - 1sec in example
#define FAILSAFE_OFF_DELAY 200 // Time for Landing before motors stop in 0.1sec. 1 step = 0.1sec - 20sec in example
#define FAILSAFE_THROTTLE (MINTHROTTLE + 200) // (*) Throttle level used for landing - may be relative to MINTHROTTLE - as in this case

#define FAILSAFE_DETECT_TRESHOLD 985

/*****                               *****/
/* I2C DFRobot LED RING communication */
// #define LED_RING

/*****                               *****/
// #define LED_FLASHER
// #define LED_FLASHER_DDR DDRB
// #define LED_FLASHER_PORT PORTB
// #define LED_FLASHER_BIT PORTB4
// #define LED_FLASHER_INVERT
// #define LED_FLASHER_SEQUENCE 0b00000000 // leds OFF

609 Arduino/Genuino Uno on COM8
ENG 9:54 AM 22/07/2021
```

THROTTLE FAILSAFE –WHAT THE THROTTLE INPUT WOULD BE IF SIGNAL IS LOST BETWEEN THE SYNERDUINO BOARD , RECEIVER AND TRANSMITTER

ESC CALIBRATION

To ensure a
proper tune
stable flight

All ESCs must
be Calibrated

SynerduinoKwad3-GY91-1.8.16-M9-10 - config.h | Arduino 1.8.18

File Edit Sketch Tools Help

```
1163
1164
1165 /*****
1166 /****      ESCs calibration      ****/
1167 /****
1168
1169 /* to calibrate all ESCs connected to MWii at the same time (useful to avoid unplugging/re-plugging each ESC)
1170 Warning: this creates a special version of MultiWii Code
1171 You cannot fly with this special version. It is only to be used for calibrating ESCs
1172 Read How To at http://code.google.com/p/multiwii/wiki/ESCsCalibration */
1173 #define ESC_CALIB_LOW MINCOMMAND
1174 #define ESC_CALIB_HIGH 2000
1175 // #define ESC_CALIB_CANNOT_FLY // uncomment to activate
1176
1177 /****      internal frequencies      ****/
1178 /* frequenies for rare cyclic actions in the main loop, depend on cycle time
1179 time base is main loop cycle time - a value of 6 means to trigger the action every 6th run through the main loop
1180 example: with cycle time of approx 3ms, do action every 6*3ms=18ms
1181 value must be [1; 65535] */
1182 #define LCD_TELEMETRY_FREQ 23 // to send telemetry data over serial 23 <=> 60ms <=> 16Hz (only sending interlaced, so 8Hz update rate)
1183 #define LCD_TELEMETRY_AUTO_FREQ 967 // to step to next telemetry page 967 <=> 3s
1184 #define PSENSOR_SMOOTH 16 // len of averaging vector for smoothing the PSENSOR readings; should be power of 2; set to 1 to disable
1185 #define VBAT_SMOOTH 16 // len of averaging vector for smoothing the VBAT readings; should be power of 2; set to 1 to disable
1186 #define RSSI_SMOOTH 16 // len of averaging vector for smoothing the RSSI readings; should be power of 2; set to 1 to disable
1187
1188 /*****
1189 /****      Dynamic Motor/Prop Balancing      ****/
1190 /****
1191 /*      !!! No Fly Mode !!!      */
1192
1193 // #define DYNBALANCE // (**) Dynamic balancing controlled from Gui
1194
```

1175 - 1176

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM83

STE

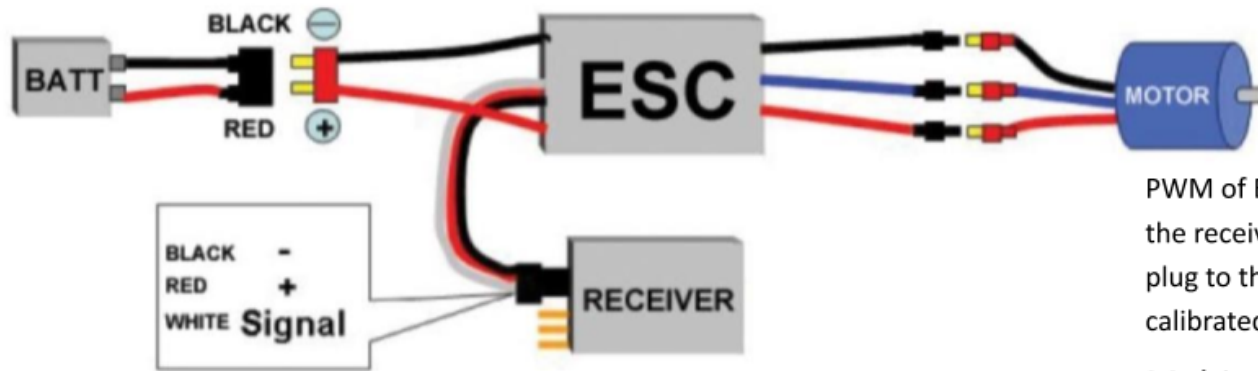
XLoader

Download

Electronic Speed Controller CALIBRATION



Propellers are removed during this process



ESC calibration will vary based on what brand of ESC you are using, so always refer to the documentation for the brand of ESC you are using for specific information (such as tones). “All at once” calibration works well for most ESCs, so it is good idea to attempt it first and if that fails try the “Manual ESC-by-ESC” method.

If your ESC happens to be an OPTO. The Synerduino board can provide as power supply for both RC Receiver and ESCs when soldered in . Get the PWM Pin of the ESC you want to calibrate and plug it into the Throttle Channel of your Receiver

PWM of ESC is directly hook up to the receiver Throttle pin . Ensure the receiver is getting power thru the Aux PWM pin which remains plug to the synerduino board (process is repeated till all ESCs are calibrated)

Multicopters must have all ESCs calibrated similarly to ensure reliable operations

Motor must be plug in at this point w/o the propeller. As it will serve several purpose

- An Speaker to listen to calibration tone of the ESC
- Identify motor rotation should it needs to be corrected
- Test full speed range

STE

Synerduino Multiwii ESC calibration method , for All ESC at Once

2

Throttle Pin is Connected to the Throttle Channel of the Receiver

All other Channels as is

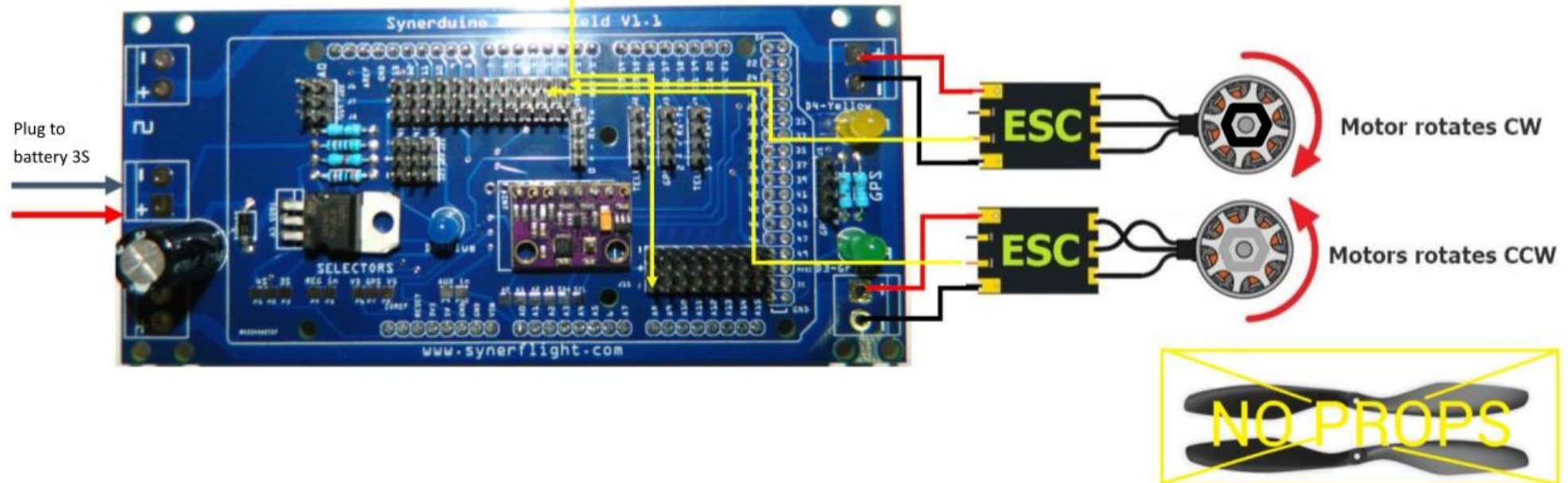


to calibrate all ESCs connected to MWii at the same time (useful to avoid unplugging/re-plugging each ESC)

Warning: this creates a special version of MultiWii Code

You cannot fly with this special version. It is only to be used for calibrating ESCs

This is applicable to those who have PPM and SBUS Receivers



STE

4

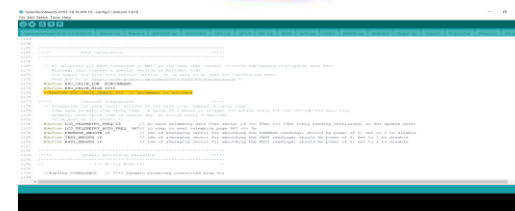
- Remove props or tie the copter down.
- Plug in USB.
- Uncomment `#define ESC_CALIB_CANNOT_FLY` // uncomment to activate
- Flash firmware to Synerduino
- Disconnect the USB.
- Plug in the battery. The copter will not fly and will use ESC Tone/LEDs to indicate finished calibration (after approx. 5-10 seconds).
- Disconnect the battery.
- Plug in USB and Comment `#define ESC_CALIB_CANNOT_FLY` // uncomment to activate
- You can test carefully with your ESCs calibrated.



Connect battery to power module.



Disconnect battery.



Connect battery to power module.

FLYWIIGUI INSTALLATION

STE

1

Download the FlyWiiGUI groundstation and open FlywiiGUI.exe

Name	Date modified	Type	Size
210130-0301	30/04/2021 3:01 PM	File	3 KB
210814-0408	14/09/2021 4:08 PM	File	3 KB
212812-0428	12/06/2021 4:28 PM	File	3 KB
214012-0340	12/06/2021 3:40 PM	File	3 KB
AForge.Controls.dll	25/01/2015 1:15 PM	Application extens...	44 KB
AForge.dll	25/01/2015 1:15 PM	Application extens...	17 KB
AForge.Imaging.dll	25/01/2015 1:15 PM	Application extens...	248 KB
AForge.Math.dll	25/01/2015 1:15 PM	Application extens...	67 KB
AForge.Video.DirectShow.dll	25/01/2015 1:15 PM	Application extens...	52 KB
AForge.Video.dll	25/01/2015 1:15 PM	Application extens...	19 KB
AForge.Video.FFMPEG.dll	25/01/2015 1:15 PM	Application extens...	60 KB
avcodec-53.dll	25/01/2015 1:15 PM	Application extens...	13,181 KB
avdevice-53.dll	25/01/2015 1:15 PM	Application extens...	342 KB
avfilter-2.dll	25/01/2015 1:15 PM	Application extens...	870 KB
avformat-53.dll	25/01/2015 1:15 PM	Application extens...	2,405 KB
avutil-51.dll	25/01/2015 1:15 PM	Application extens...	135 KB
FlyWiiGULexe	30/10/2021 11:41 ...	Application	6,945 KB
FlyWiiGULexe.config	28/02/2017 5:31 PM	CONFIG File	1 KB
FlyWiiGULexe.manifest	30/10/2021 11:41 ...	MANIFEST File	30 KB

The FlyWii GUI is a free updated version of the [MultiWii WinGUI](#). It serves as the ground control station for the MultiWii 2.4 controller software.

FlyWii GUI is currently only supported for Windows 7/8/10.



Download

Latest Release

FlwiiGUI Ground Station Software .EXE*

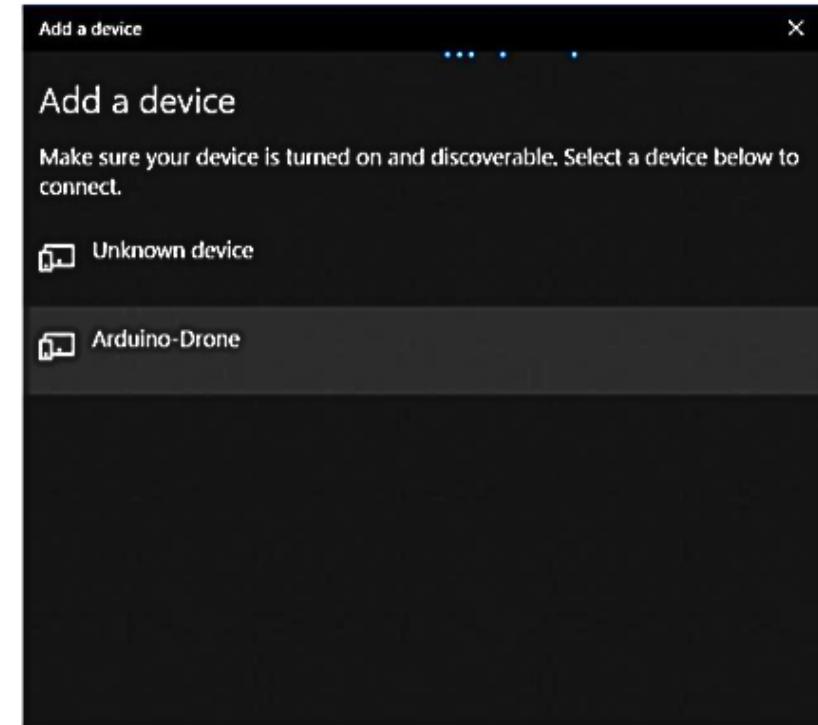
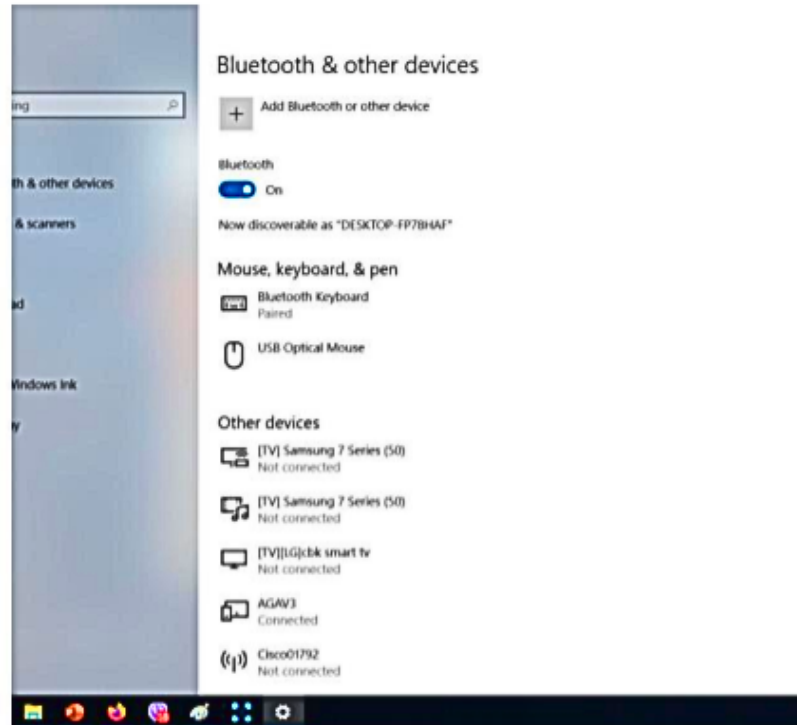
FlyWiiGUI20

Download

FLYWIIGUI INSTALLATION

STE

2



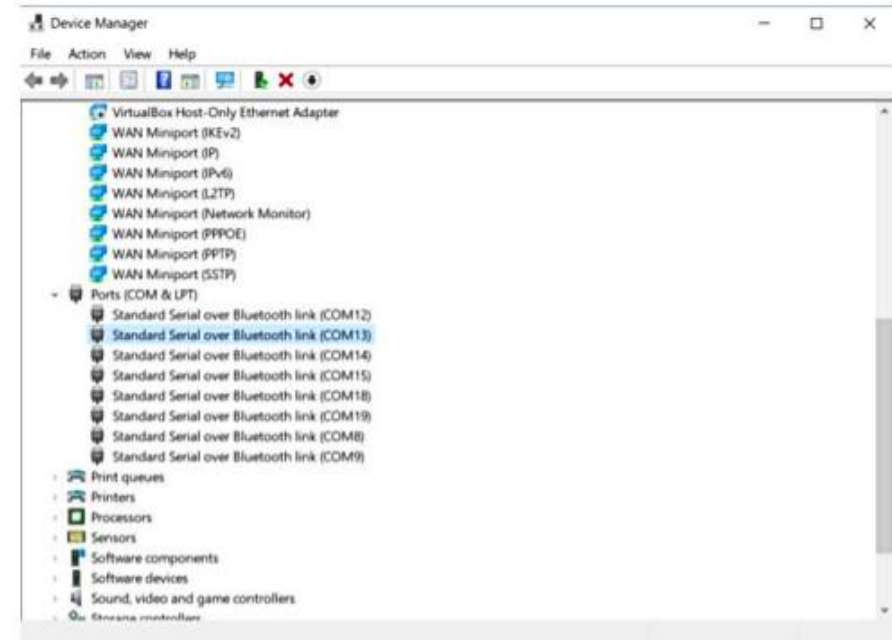
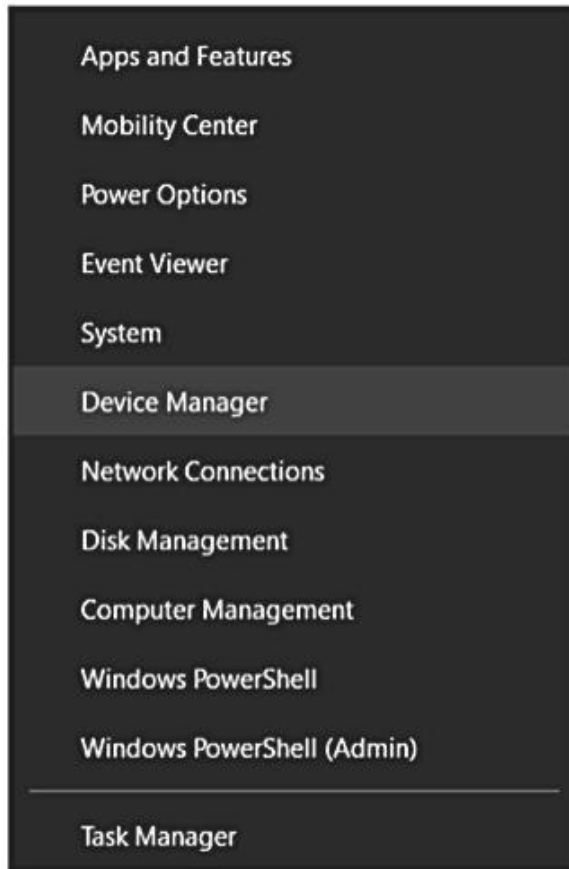
Adding Bluetooth on Windows Device Manager look for Arduino-Drone BT device

Take note on which Serial Com port its added to in Device Manager

FLYWIIGUI INSTALLATION

STE

3



In Device Manager Located in COM & LPT

FLYWIIGUI INSTALLATION

STE

4

Select the com port your Bluetooth is connected to .

At this point Disconnect your Physical USB and your drone should be running on batteries using only the Bluetooth to communicate

Connect to the Drone with the associated COM port and Baud as found in your device manager



FLYWIIGUI INSTALLATION

Calibration Acc – Drone must be on level surface

Write settings after changes made in any of the parameters

Serial Com

Altitude

Heading

Calibration Mag/Compass

Flight Log

RC PWM

Refresh rate

Attitude

GPS

Serial Data

Vertical Speed

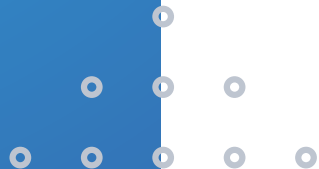
Analog sensor / Battery

Sensor Status

Frame type / Motor PWM

The screenshot shows the FlyWiigUI software interface. At the top, there is a menu bar with options: Connect, Read Settings, Write Settings, Load Defaults, Load from File, Save to File, Start Log, Start GPS log, Log Browser, and About. Below the menu bar is a toolbar with buttons: Refresh Rate, Pause, Calibrate ACC, Calibrate Mag, and Bind Spektrum. The main display area contains several gauges and data fields. On the left, there are three circular gauges: Attitude (showing roll and pitch), Altitude (showing height in meters), and Heading (showing direction with a compass rose). Below these are three more gauges: GPS (showing distance to home), Vertical Speed (showing up/down speed), and an Analog sensor / Battery gauge (showing 0.0v). On the right, there is a section for RC PWM with a diagram of a quadcopter and a table of PWM values for Thr, Pitch, Roll, Yaw, and Aux channels. Below this is a section for Active Sensors with buttons for ACC, GPS, BADO, MAG, OPTIC, and SONAR. At the bottom, there is a section for Serial Data with fields for Packet's sent, received, and CRC error, and a section for Telemetry link info with fields for RSSI and Noise. The bottom right corner shows battery status information: PC Error count, Battery voltage, and Power Sum.

FLIGHT TUNING





FLIGHT DECK – IF THIS DOESN'T LOOK RIGHT CHECK YOUR SENSORS ORIENTATION AGAIN USING THE SENSOR GRAPH

TELEMETRY CONNECTION SEE YOUR CHECK YOUR BLUETOOTH RADIO OR USB ON WHERE IS THE VIRTUAL COM PORT IS

COMPASS (MAG AND GYRO)

PWM OUTPUT INDICATOR

PWM INPUT INDICATOR

SAVE CONFIG

FLIGHT & GPS LOGS

ALTITUDE (BARO)

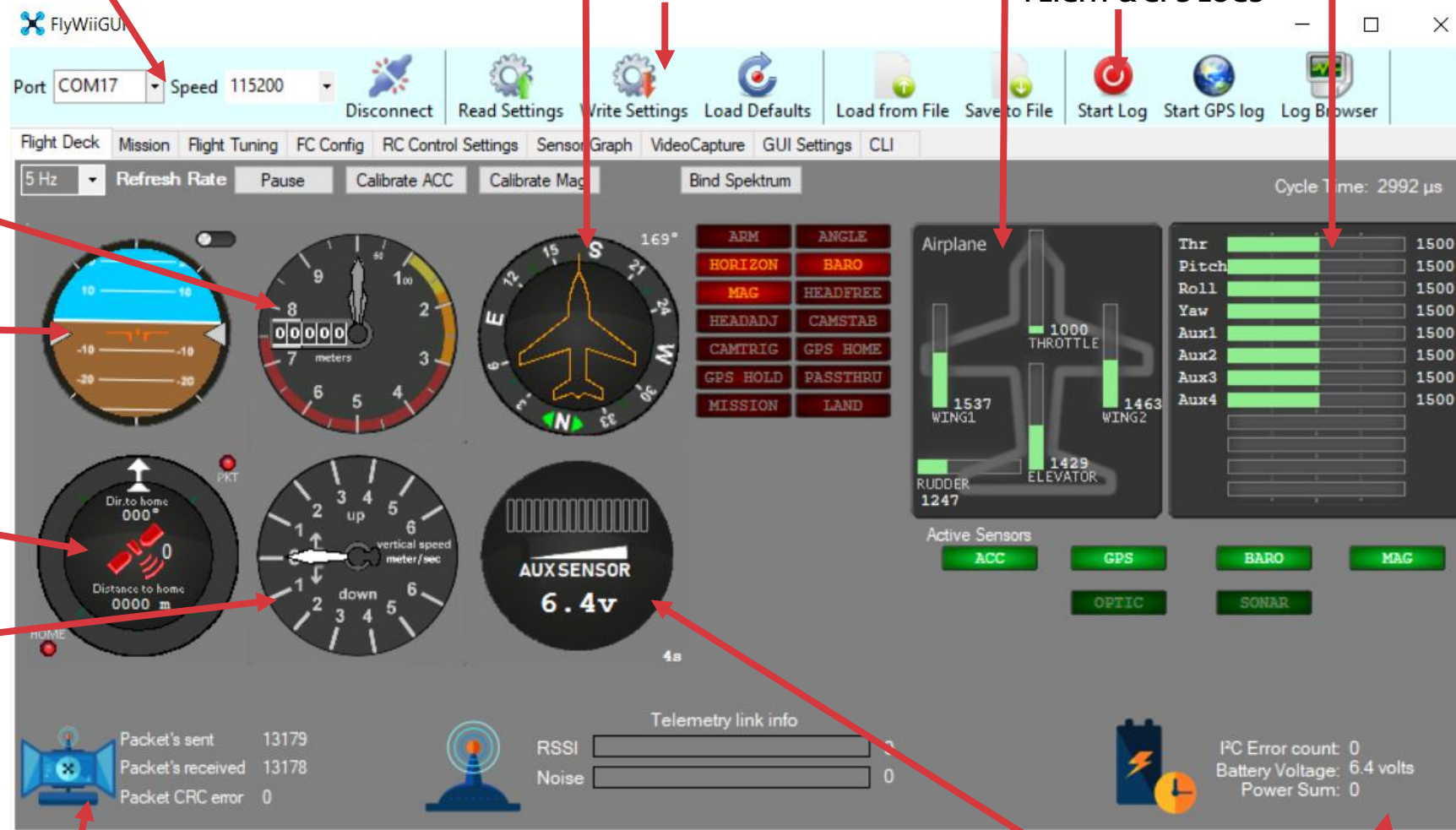
ATTITUDE (ARTIFICIAL HORIZON)
(GYRO XYZ AND ACC XYZ)

GPS SATELLITE COUNT
(4 SATS FOR 3D FIX – IDEAL 7 SATS)

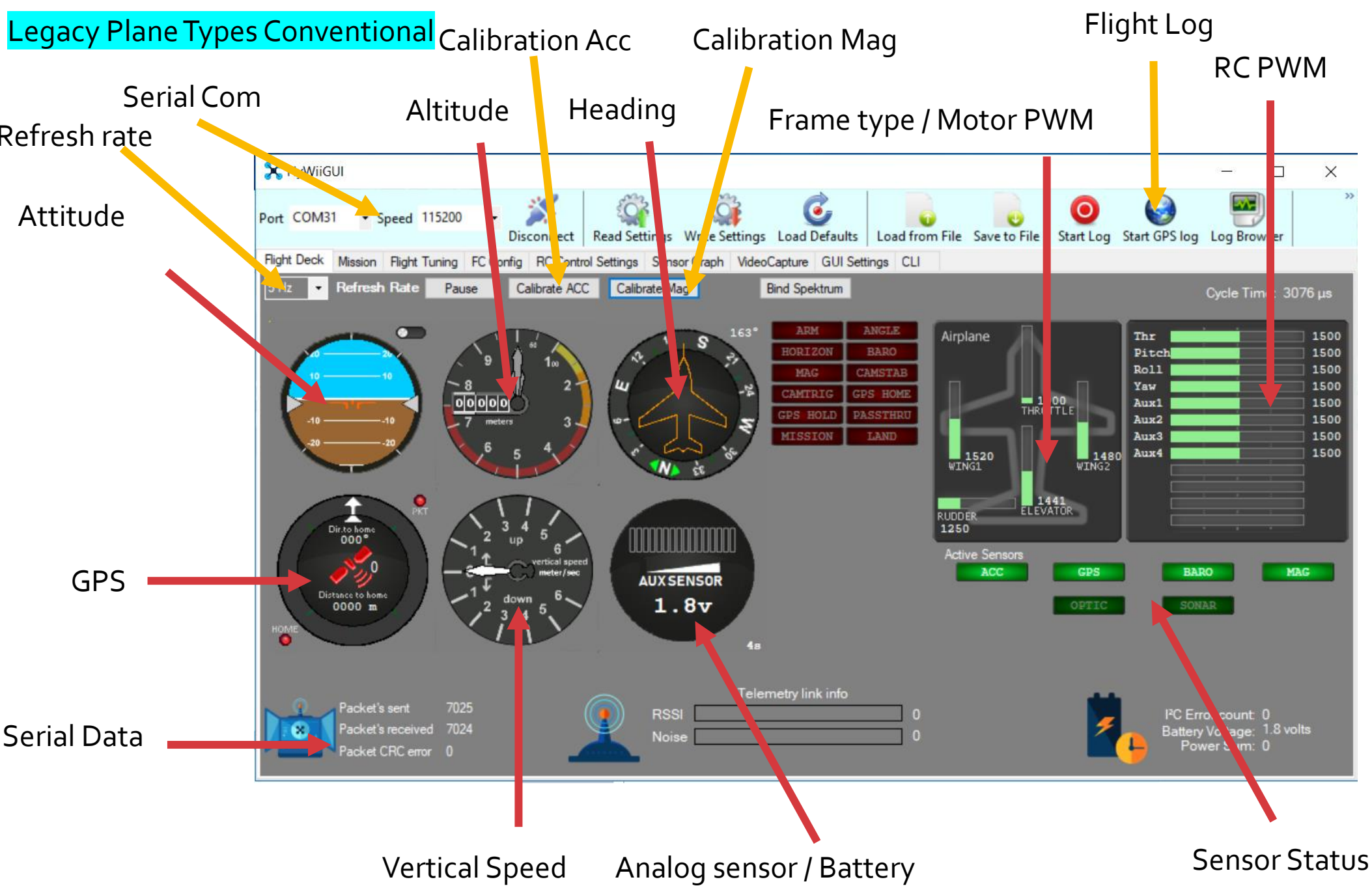
VERTICAL SPEED INDICATOR
(ACC Z AXIS)

PACKETS STATUS
(IF THE ERROR NUMBERS ARE HIGH PLS
CHECK YOUR TELEMETRY CONFIG)

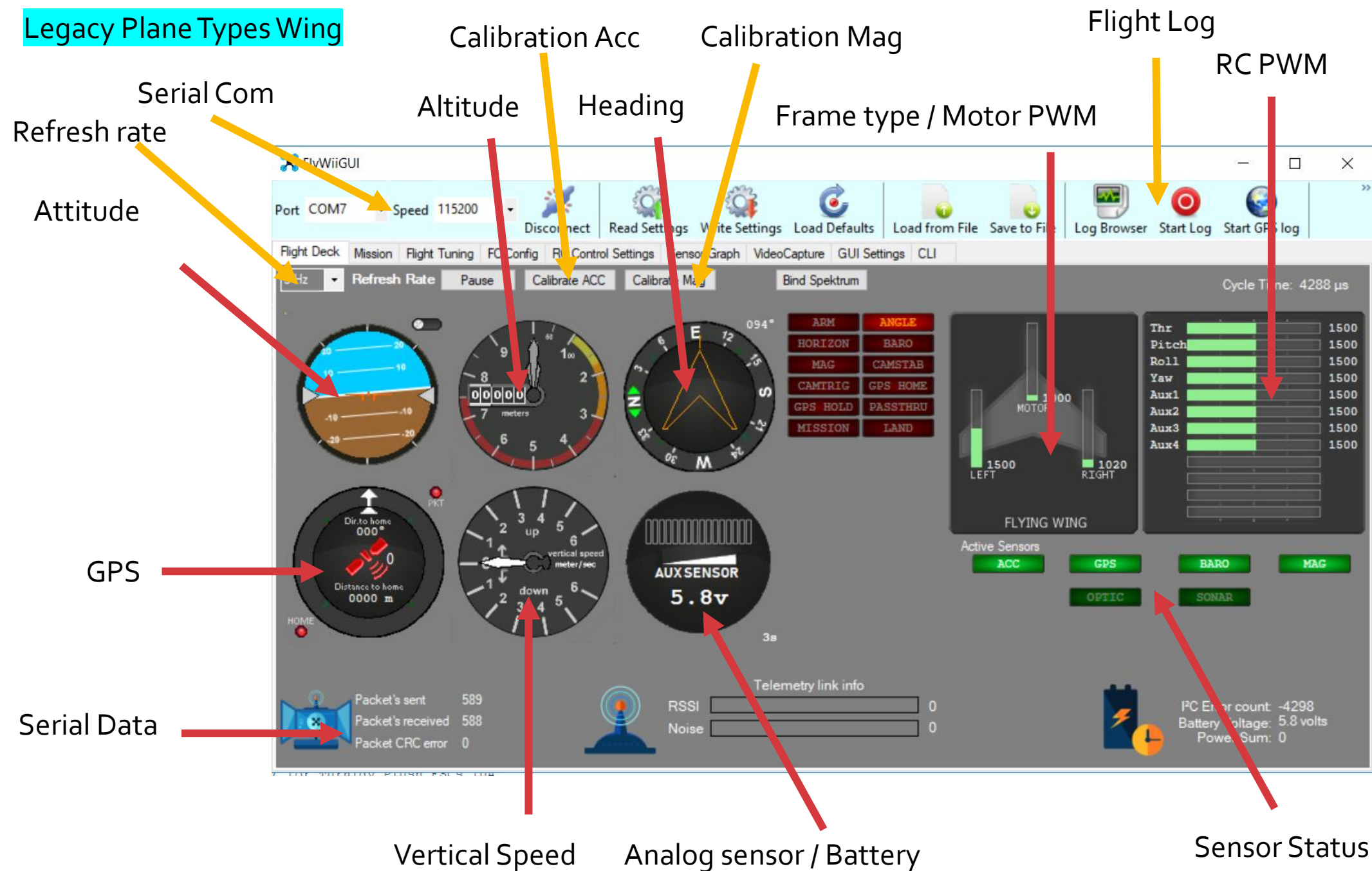
POWER STATUS / AUX SENSOR
ALSO KNOWN AS FUEL GAUGE
(VBAT)



Legacy Plane Types Conventional



Legacy Plane Types Wing



Special Plane Types Differential

Calibration Acc

Calibration Mag

Flight Log

RC PWM

Serial Com

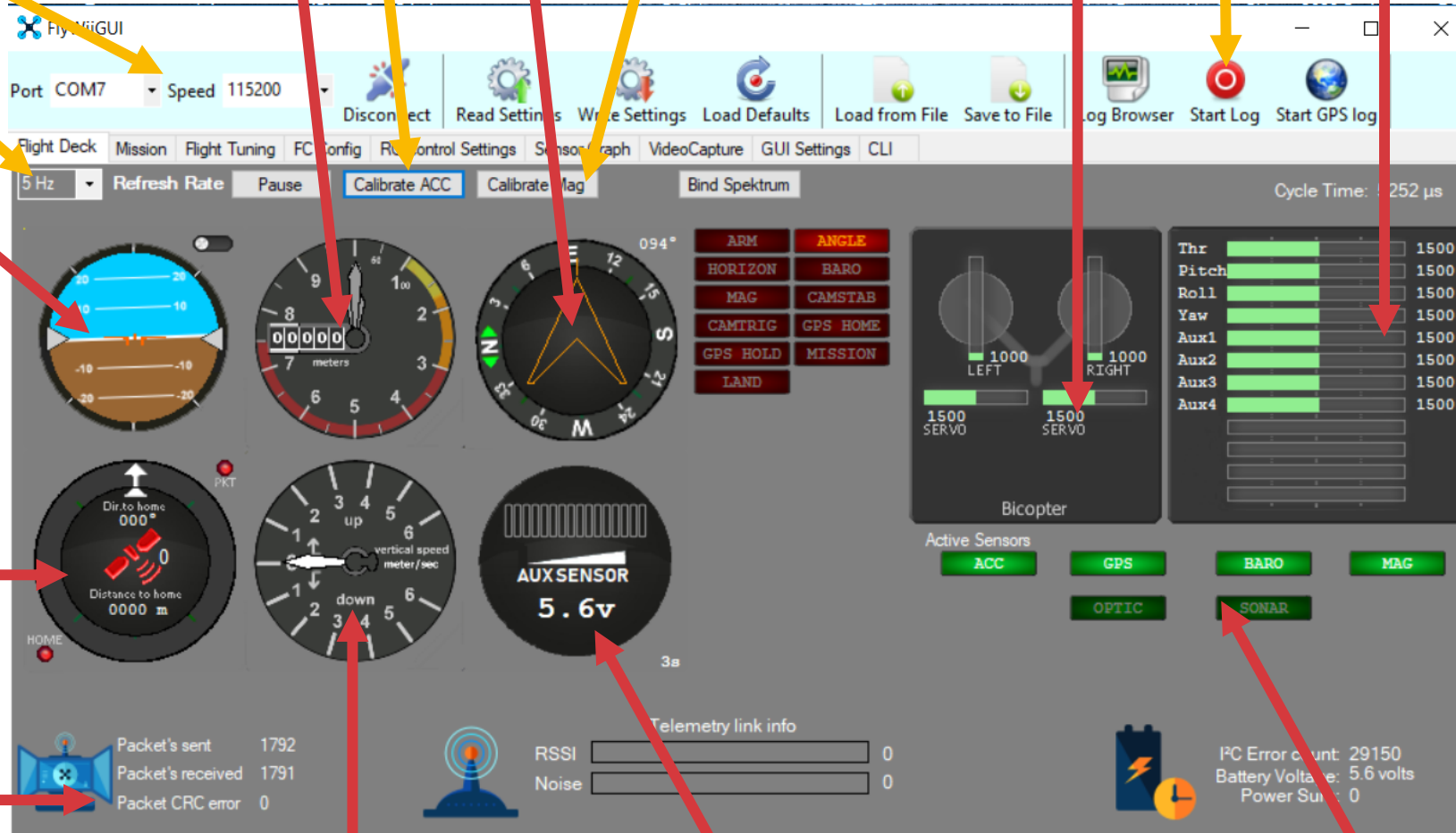
Altitude

Heading

Frame type / Motor PWM

Refresh rate

Attitude



GPS

Serial Data

Vertical Speed

Analog sensor / Battery

Sensor Status

Calibration Mag



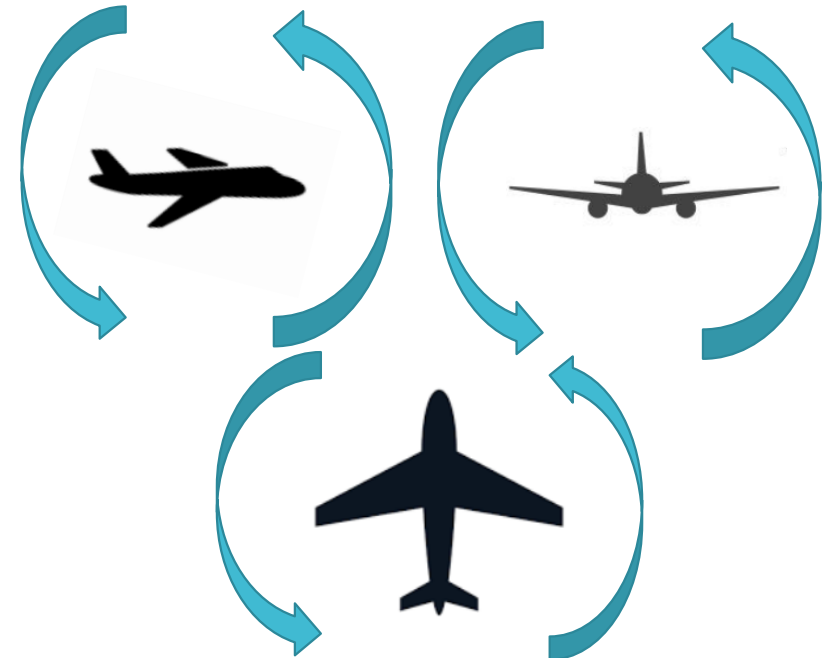
Refresh Rate . Telemetry update speed

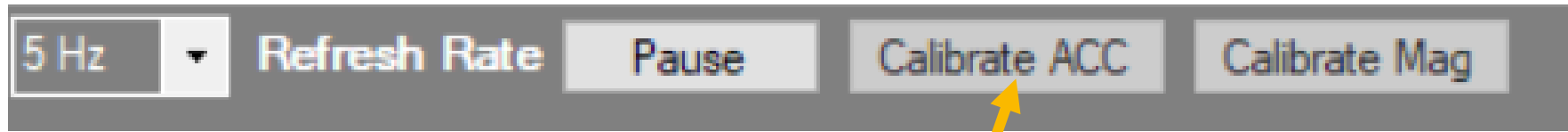
Acc Calibration . Set the Plane down on a level surface . Away from any metal objects for 10 secs.

Mag Calibration . rotate the aircraft 360 degrees in all axis within 1 min. while the blue Led flashes

Mag Calibration must be perform when launching your drone in a new location for the first time. Pls verified the Compass if the drone heading matches your compass app in your phone.

These Calibration must be perform after Parameter updates after Flashing the firmware
Blue LED would flash during these calibration processes

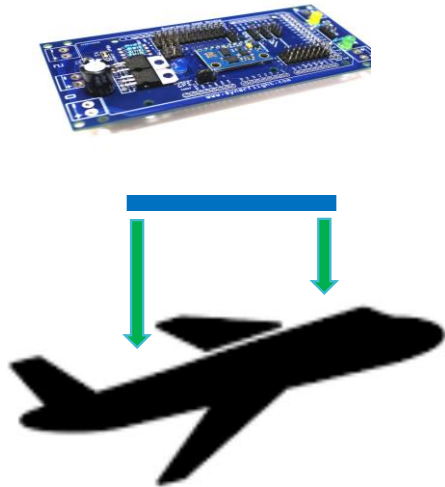




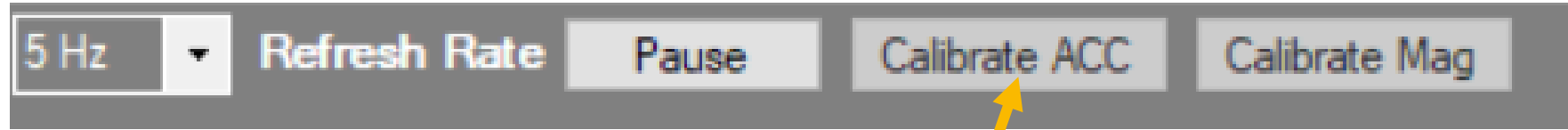
Flight controller Mounting in relation to Angle of Attack

In Neutral AOA – the Flight controller (Synerduino) can be set into a Neutral AOA to the airframe as being Level

This Hold true for any Fixwing that uses Flat plate airfoil wings to compensate for the lack of neutral lift unlike NACA or Cambered Airfoil



Calibration ACC



Due to the Nature of Flat wing Aircraft like the Synerduino Dart / FT Flyer an **Neutral Angle of Attack** must be identify and establish
The Synerduino Dart/FT flyer is set as 45 degrees AOA



Acc Calibration ,set the Aircraft on its neutral Angle of attack for 10 secs.



Max Speed AOA – this is when the aircraft is in its maximum Waypoint cruise speed as set by **Max Nav Banking** angle on Flight tuning tab Navigation settings



Neutral AOA – this is where the Aircraft AOA is settle on Horizon mode



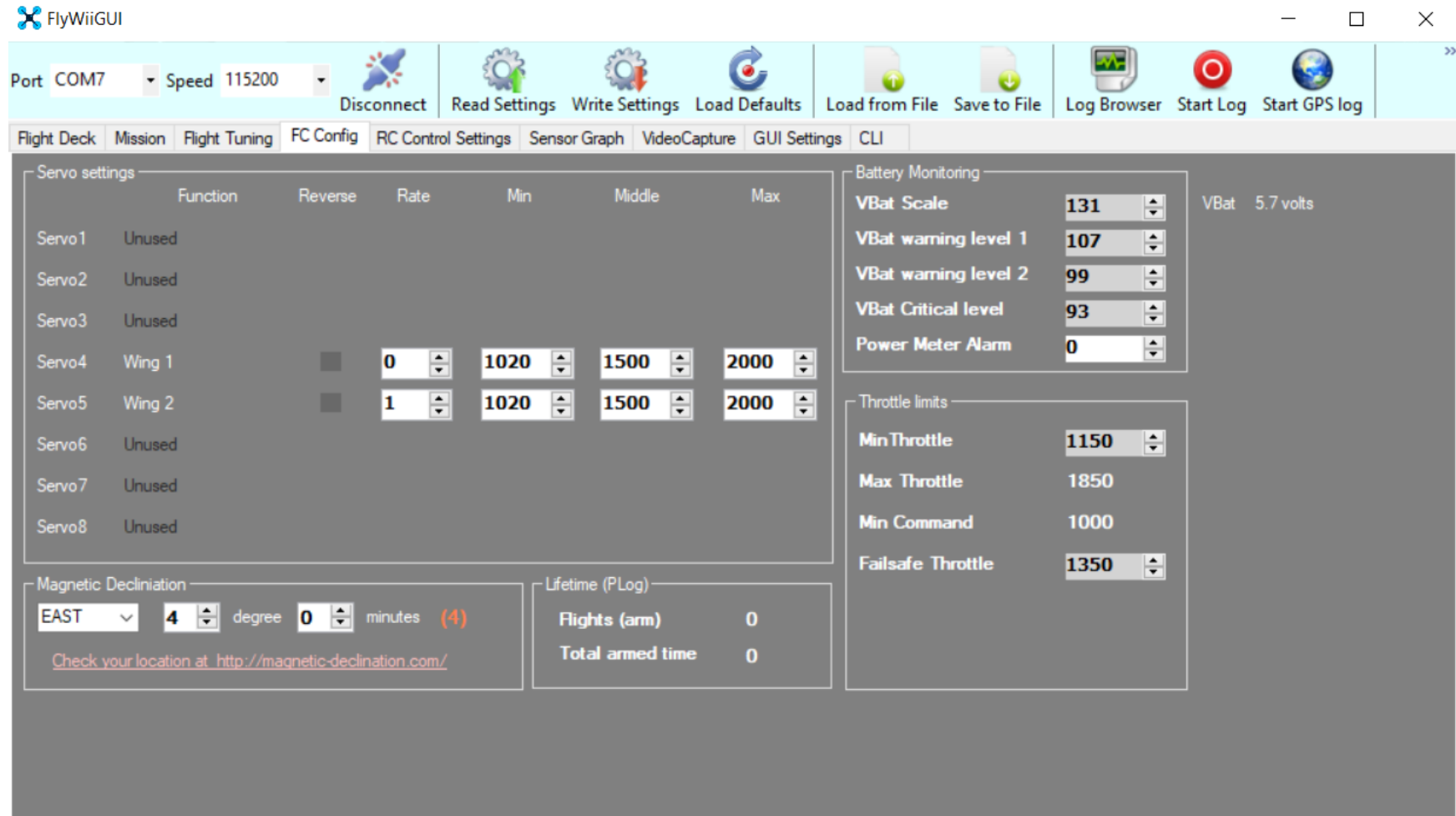
Stall AOA – this is when the aircraft is slowing down also set by **Max Nav Banking** angle on **Flight tuning tab Navigation settings** . Happends when the mission is set to hold the aircraft tries to slow down or when on landing

Legacy Plane Types Wing

This is for control Surface Augmentation for the 2 servos
Wing1 elevon & Wing 2 Elevon

- **FC Config**
 - **Gyro or Acc assisted Mode.**
 - Check if Gyro move servos in right directions.
 - Lift a wingtip and Aileron goes up.
 - Lift the tail and Elevator goes up.
 - Rudder moves in same direction as the tail.
 -
- Use the Servo Tab in Gui to Change Servo directions.

- Rate 1-norm or 2-rev

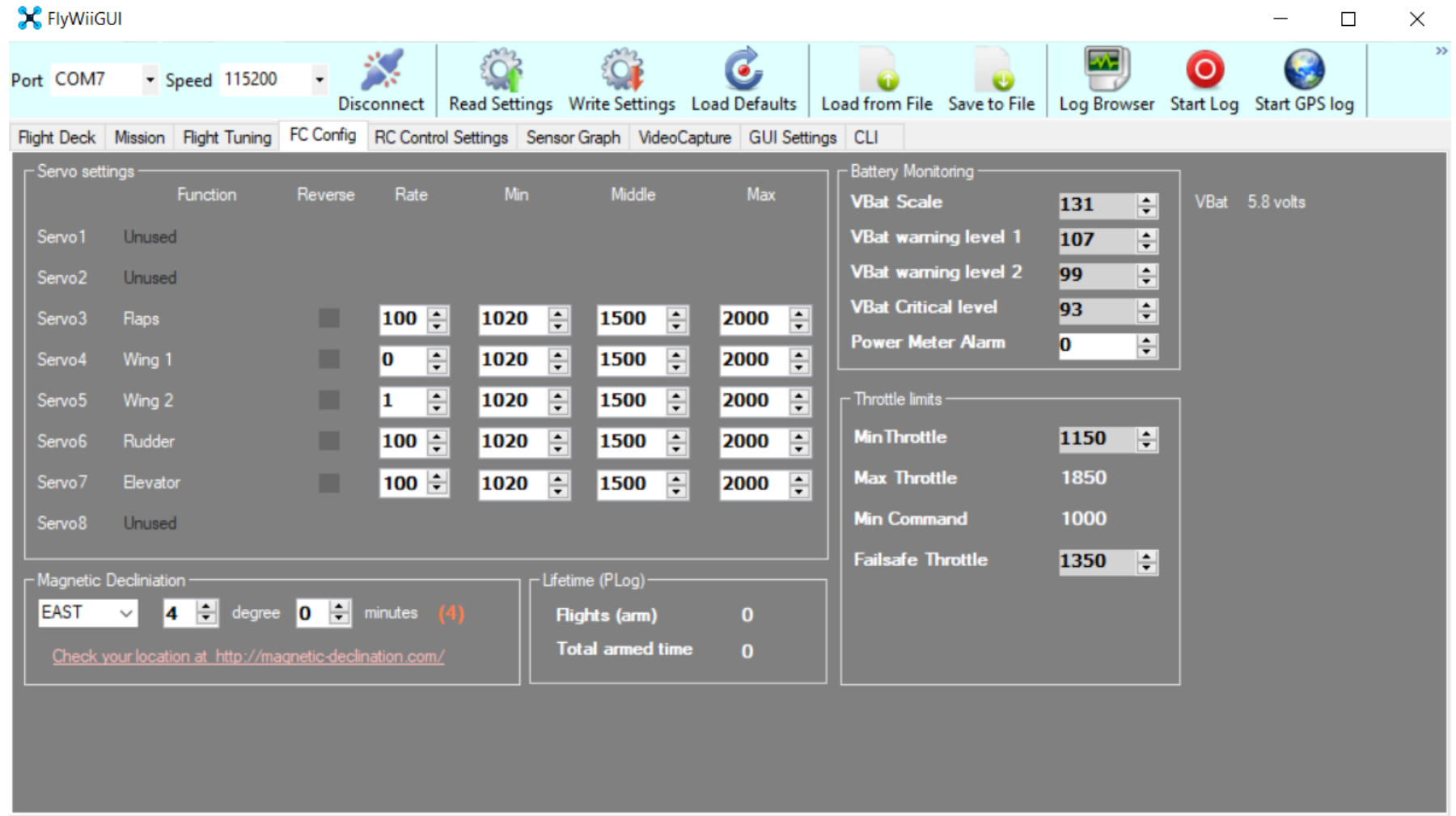


Legacy Plane Types Conventional

This is for control Surface Augmentation for Surface types shows the 5 servos Elevator ,Rudder ,Flaps and Aileron

- **FC Config**
 - **Gyro or Acc assisted Mode.**
 - Check if Gyro move servos in right directions.
 - Lift a wingtip and Aileron goes up.
 - Lift the tail and Elevator goes up.
 - Rudder moves in same direction as the tail.
 -
- Use the Servo Tab in Gui to Change Servo directions.

- Rate 1-norm or 2-rev



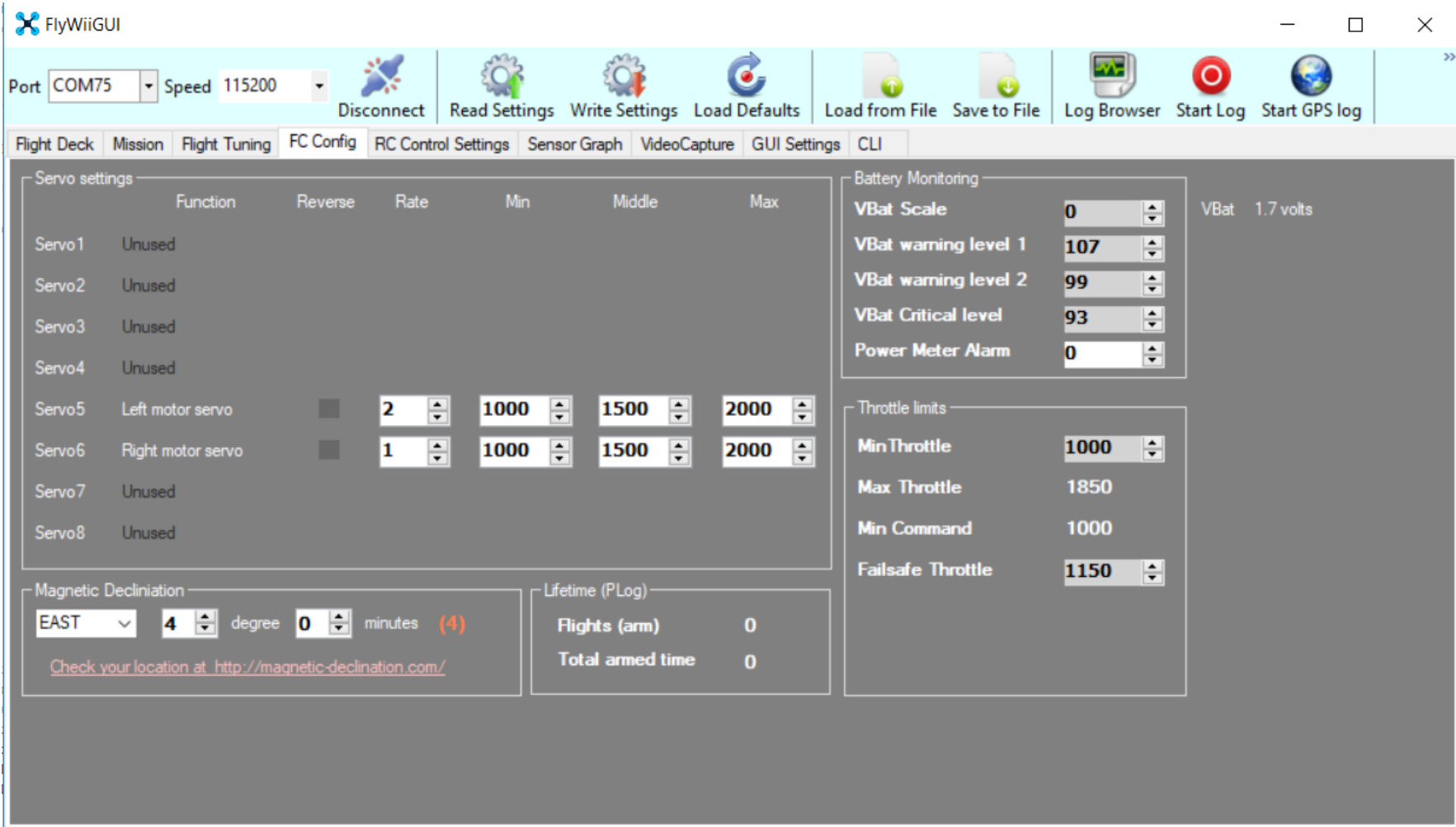
Special Plane Types Diff thrust or Special

This is for control Surface Augmentation for shows the 2 servos Elevator and Aileron on Special Types

- *FC Config*
- *Gyro or Acc assisted Mode.*
- Check if Gyro move servos in right directions.
- Lift a wingtip and Aileron goes up.
- Lift the tail and Elevator goes up.
- Rudder moves in same direction as the tail.
-

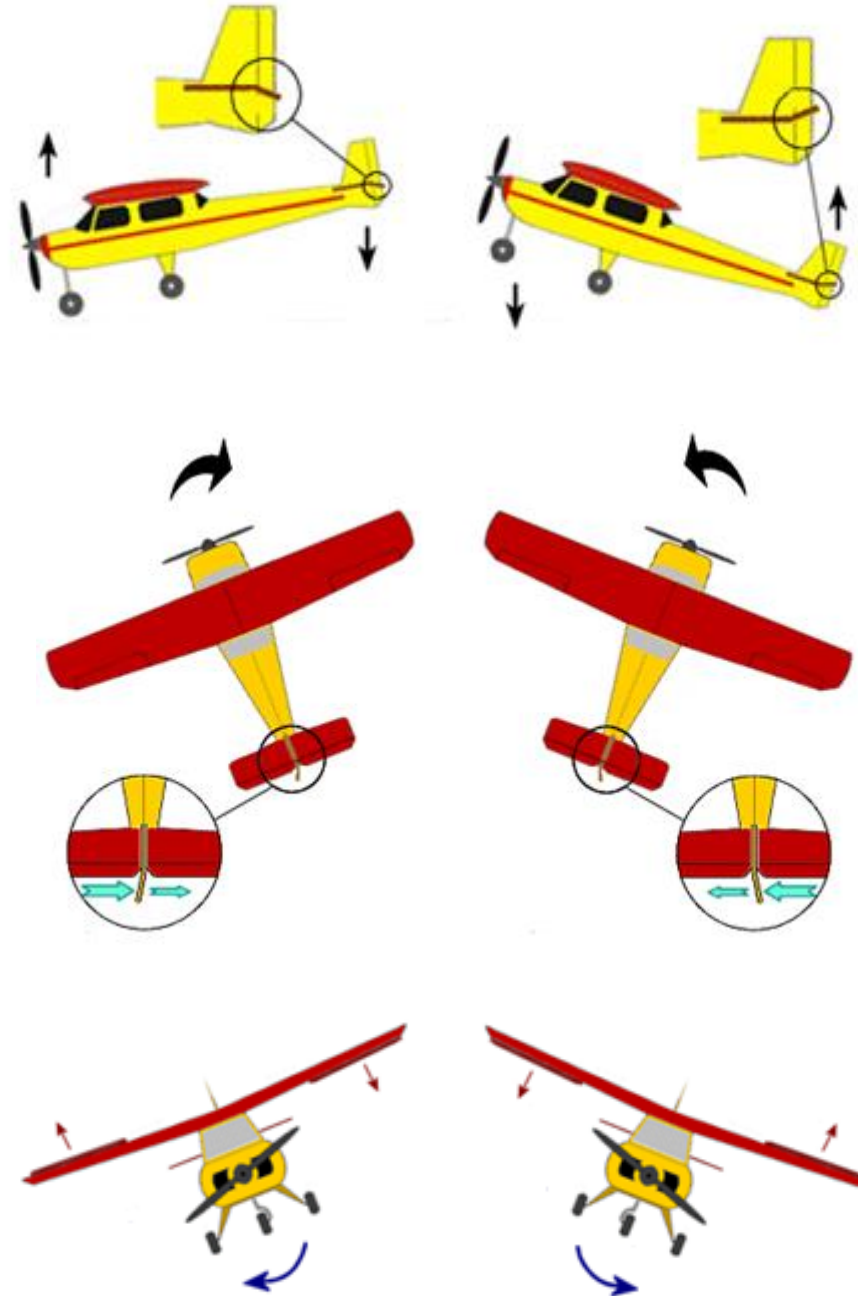
Use the Servo Tab in Gui to Change Servo directions.

- Rate 1-norm or 2-rev



This is for surface control Augmentation

- **Servos Settings**
- **Gyro or Acc assisted Mode.**
- Check if Gyro move servos in right directions.
- Lift a Right wingtip and Right Aileron goes up.
- Lift the tail and Elevator goes up.
- Rudder moves in same direction as the tail.
- Use the Servo Tab in Gui to Change Servo directions.
- The Aircraft should give the correct proportion for control attitude correction
- Control surfaces should oppose the disturbance applied to the aircraft

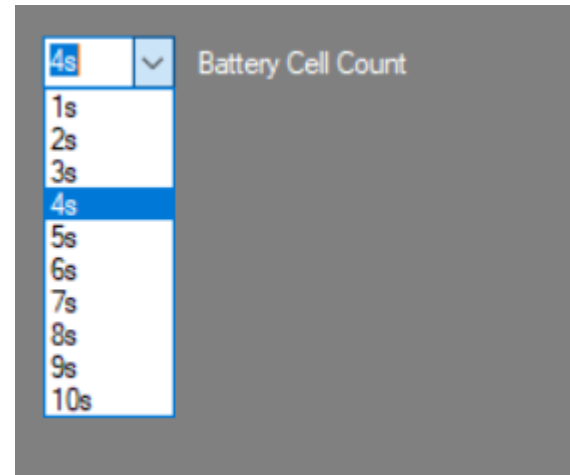
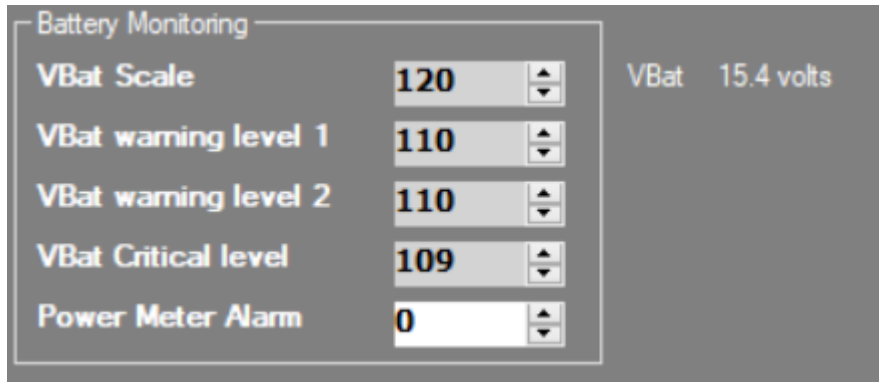


The arrow represent an external force acting on the aircraft opposite to it the control surface counter it with the opposite force

The Gyro and Accelerometer would always counter the force opposite to it .



FLYWII GUI



(FC CONFIG TAB)

BATTERY MONITORING

VBAT SCALE - ADJUST THIS TO MATCH THE BATTERY VOLTAGE OUTPUT USING THE VOLTAGE ALARM INDICATOR

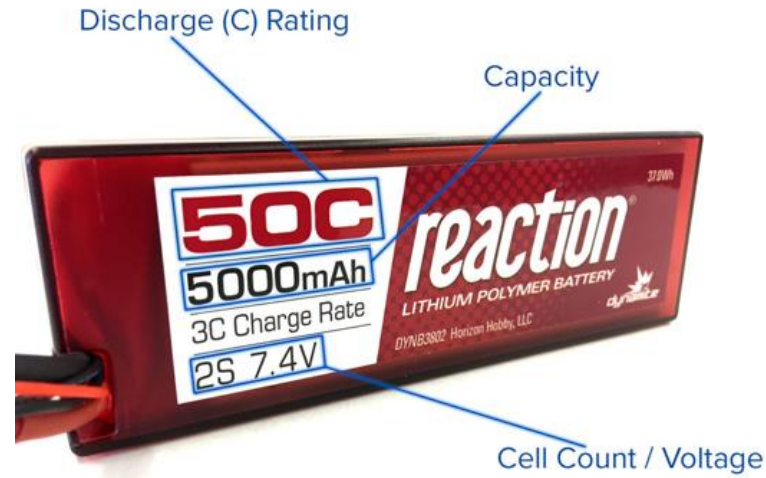
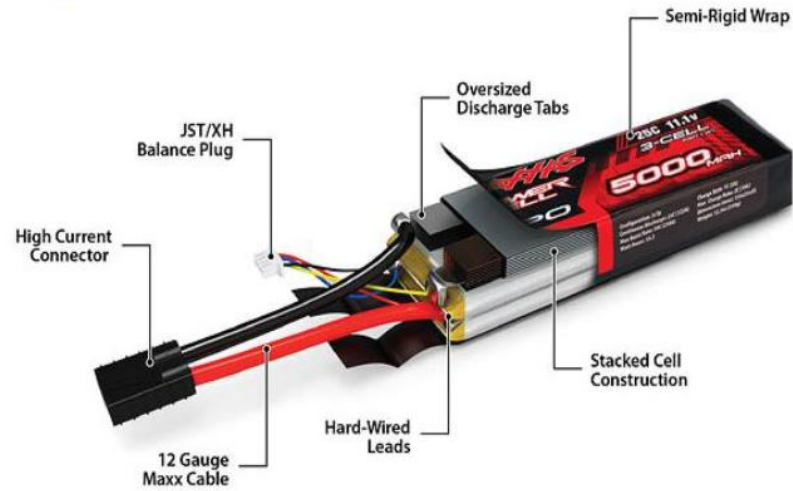
VBAT WARNING LEVEL – IDENTIFY THE NOTICE WHEN THE BATTERY DROPS TO THIS VOLTAGE

(GUI SETTINGS TAB)

BATTERY CELL COUNT- ADJUST THIS DEPENDING ON THE NUMBER OF CELLS

THIS BOARD SUPPORTS 2S-4S BATTERY

BATTERY MONITORING



BATTERY CARE – ONLY CHARGE AT 1C OR THE RECOMMENDED CHARGE RATE ON THE LABEL

USE BALANCE CHARGER TO SET THE CURRENT IN THE SAMPLE IS 5A

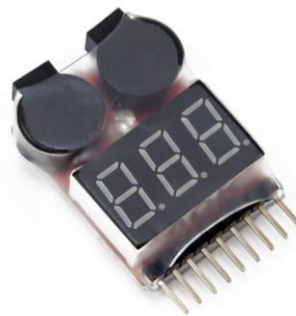
BATTERY STORAGE MODE IS 3.8V PER CELL

BATTERY DISCHARGE IS 3.6V PER CELL

BATTERY FULL CHARGE IS 4.2V PER CELL

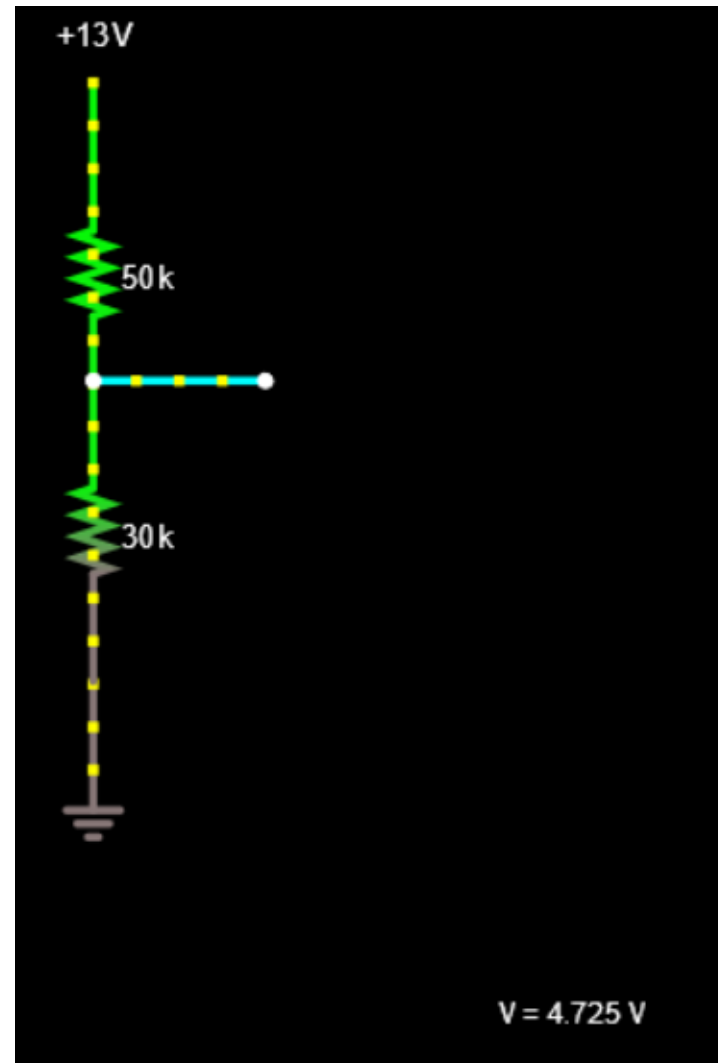
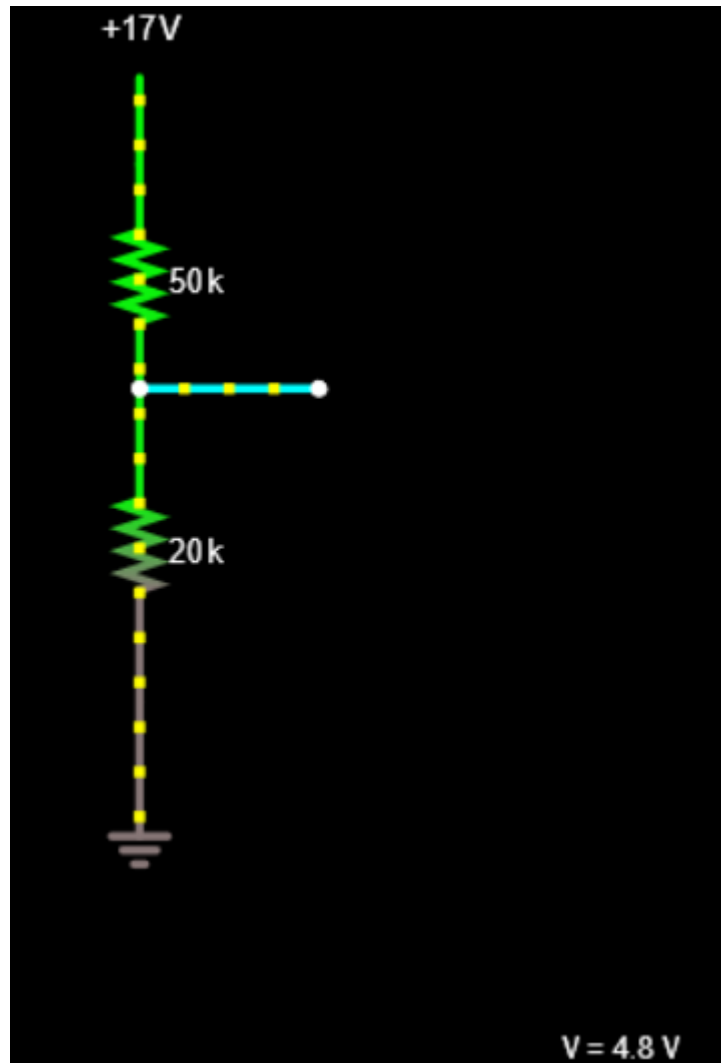
IT'S RECOMMENDED TO USE THE VOLTAGE ALARM TO MONITOR THE BATTERY VOLTAGE WHILE IN USE

BATTERY



4S 16.8V

3S 12.6V



VOLTAGE READING

HOW MUCH POWER IN YOUR BATTERY

VOLTAGE DIVIDER

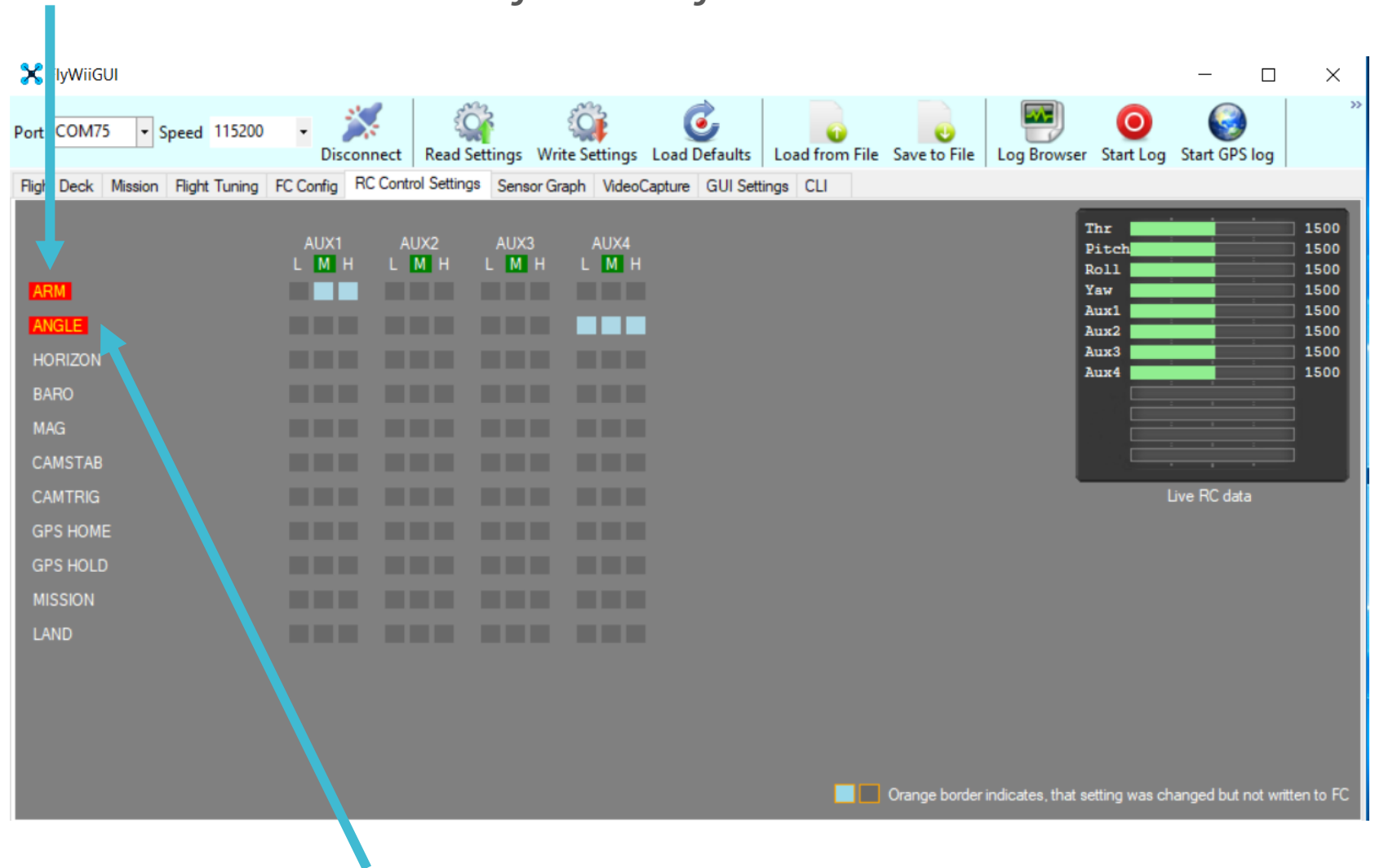
THIS ALLOWS THE 3V-5V TO BE INPUTTED TO THE A₀ ANALOG PIN OF THE ARDUINO TO READ THE BATTERY VOLTAGE

SWITCH THE VBAT JUMPER ACCORDINGLY TO THE BATTERY CELLS YOU'RE USING

3S OR 4S

ARM – Arms the main motor of the aircraft

- The Diff thrust plane of Synerduino is rather unique
- It behaves like a Bi-copter as also a plane
- ARM — to activate the Aircraft
- ANGLE - this is mostly use to keep the plane Level
- HORIZON - this also may work for certain Leveling application
- BARO — Altitude hold by throttle control
- MAG — Heading hold
- CAM TRIG — Payload Trigger
- GPS HOME — RTH
- MISSION — Perform Waypoint
- LAND — this allows the Aircraft to Land after RTH, must use in conjunction to GPS Home



Use Angle mode for stabilization

- **Acc Calibration**

- Place the plane on a stable surface.
- Wings Level with nose in expected Attack angle for level flight.
- Normally a few degrees up.
- Memorize the planes attitude in flight this should be Level for Acc.
-

- ***Passthru***

- Sends Rc commands direct to servos.
No influence from sensors.

- ***Gyro Mode (Acro)***
- This is "Normal" mode when nothing else is selected.
- The plane should compensate for movements. (Wind Gusts etc)
- The plane feels stable and locked in but still able to loop & roll.
- Stall speed is lower and it can be necessary to "Push" it down in landings.

- ***Stable Modes***

- With the sticks centered the plane will self stabilize.
Returning to level flight from almost any situation.
 - Provided there's enough Altitude for recovery.
 -
-
- **Horizon Mode** Allows rolls and loops. Levels with centered sticks
This is a comfortable flight mode for FPV.
-
- **Angle Mode** also limits how much the plane can tilt.
Gives a Stiff feeling and is only recommended for beginners.



Graphs and Sensors

Upload the sketch to the Arduino attach to the drone shield and open the FlywiiGUI sensor Graphs tab and hit connect to the appropriate COM your drone is connected to



the correct orientation

Roll Right + no#

Pitch nose down + No#

Z up + No#

Roll Right + no#

Pitch nose down + No#

Yaw Right +No#

Mag & HEAD degrees
corresponds to the compass

(0 degrees = North)

Alt up +no#

Example : if roll the drone to the right the Accelerometer and Gyroscope graphs would show positive numbers and to the Left Negative numbers

If Lift the drone up Vertically the accelerometer Z axis should shows positive numbers and altitude should show a climb in meters

Other Navigation Functions

Navigation settings

☒ Enable GPS filtering

☒ Enable GPS forward prediction filter

☐ Don't reset home position at arm

☒ Nav controls heading

☐ Fly with tail first

☐ Turn to takeoff heading at home

☐ Wait for reach RTH alt.

☐ Enable slow navigation

☒ Ignore throttle during Nav and RTH ☒ Takeover BARO mode

WP Radius (cm) **100**

RTH Altitude (m) **15**

Crosstrack gain **0.40**

Max Nav speed (cm/s) **200**

Min Nav speed (cm/s) **100**

Max Nav banking (degree) **30**

Land Speed **100**

Safe WP distance (m) **500**

Max Nav Altitude (m) **100**

Fence radius (m) **600**



WP Radius – the radius of the area the Pos PID with trigger it has reach the waypoint

Max Nav Speed – Maximum speed the drone travel between waypoints (too fast and you likely over shoot your target) *for first mission flight test Nav speed of 100cm/s with ("Enable Slow Navigation "Active)*

Min Nav Speed – the speed the drone travel when with in the WP Radius 200cm/s for starters

RTH Altitude – Altitude the drone will climb to when its below the altitude in relation to its home point when the RTH is trigger set this to 0 to RTH at current altitude

Max Nav Banking – the max allowable pitch and roll the drone will be set too while traveling between waypoints (tune this along with Max Nav Speed to take account with Environment conditions)

Max Nav Altitude – Max altitude the drone is cap to fly at

Land Speed – speed of descending for Landing cm/s

Safe WP Distance – max distance between waypoint before its null out

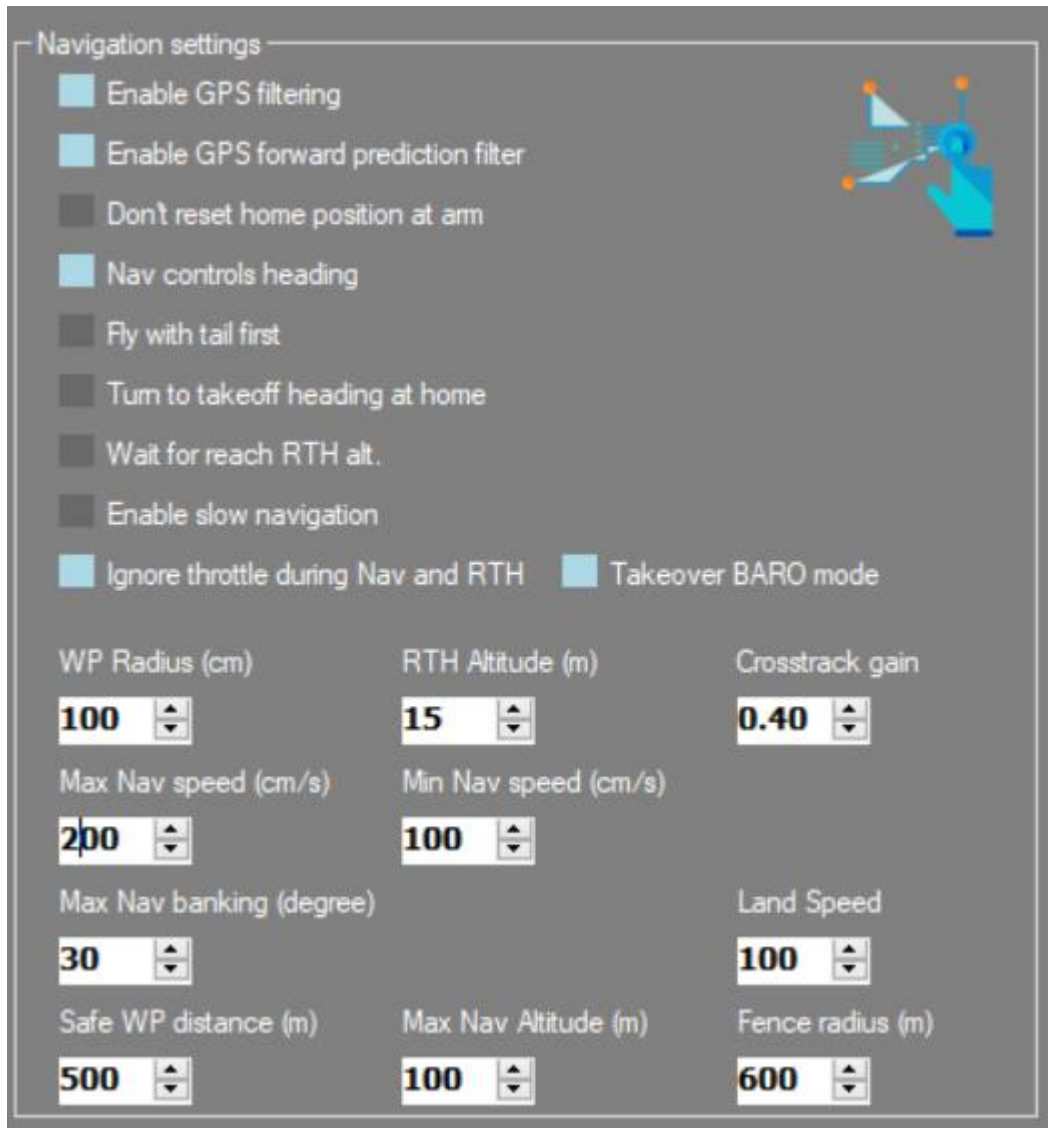
Fence Radius – Geo Fence to keep the drone with in the perimeter in relation to home position

CrossTrack gain - this tune the GPS and Nav sensitivity

GPS Filtering – use to enhance GPS accuracy

GPS Forward Prediction Filter – predicting the drones location and to compensate for lag . (optional) – not necessary for most application

Other Navigation Functions



Don't Reset Home position at Arm – this retains the home position where you first plug power on your drone

Nav Controls Heading – this points the drone to its next waypoint Important to fixwings

Fly tail first – Not applicable in fixwings

Turn take off heading at Home – when drone arrives at home position it orientates to its heading right after arming

Wait to reach RTH - this works with RTH altitude command which the drone would climb to the said altitude before initiating the flight to home position

Enable slow navigation – this works with keeping the drone to its **Min Nav speed**

Ignore throttle and Take over Baro – as the name suggest disable throttle stick command from the controller when the drone is on mission mode

Important Navigation Settings for Fixwing

- **Enable GPS Filtering ON**
- **Nav Control Heading ON**
- **Flywith Tail First OFF**
- **Turn Takeoff Heading OFF**
- **Wait for Reach RTH alt OFF**
- **Enable slow navigation OFF**

- Flight Tuning

FlyWiiGUI

Port: COM75 Speed: 115200

Disconnect Read Settings Write Settings Load Defaults Load from File Save to File Log Browser Start Log Start GPS log

Flight Deck Mission Flight Tuning FC Config RC Control Settings Sensor Graph VideoCapture GUI Settings CLI

Roll
P 3.3 I 0.030 D 23
Pitch
P 3.8 I 0.030 D 23
Yaw
P 7.0 I 0.045 D 0
Altitude
P 6.4 I 0.025 D 24
PosHold
P 0.18 I 0.00
PosHoldRate
P 6.5 I 0.14 D 0.053
Navigation Rate
P 15.0 I 0.33 D 0.083
Level
P 14.0 I 0.010 D 100
Mag
P 6.0

RC Expo 0.65
RC Rate 0.90
Thr. MID 0.50
Thr. EXPO 0.50

Navigation settings

- ☐ Enable GPS filtering
- ☐ Enable GPS forward prediction filter
- ☐ Don't reset home position at arm
- ☐ Nav controls heading
- ☐ Fly with tail first
- ☐ Turn to takeoff heading at home
- ☐ Wait for reach RTH alt.
- ☐ Enable slow navigation
- ☐ Ignore throttle during Nav and RTH
- ☐ Takeover BARO mode

WP Radius (cm) 100 RTH Altitude (m) 15 Crosstrack gain 0.40

Max Nav speed (cm/s) 200 Min Nav speed (cm/s) 100

Max Nav banking (degree) 30 Land Speed 100

Safe WP distance (m) 500 Max Nav Altitude (m) 100 Fence radius (m) 600

Rates/Expo

Roll/Pitch RATE 0.00
Yaw RATE 0.00
Throttle PID attenuation 0.00

Stability	Roll,Pitch,Yaw Gyro	PID : Level of your Gyro
Horizon / Level Mode	X,Y Accelerometer	PID : X ,Y Axis Level of your Accelerometer
Heading Lock	Compass/Mag	PID : heading of your magnetometer Calibration
Altitude Hold	Barometer / Z	PID : Barometer and Z accelerometer
Position Hold	GPS Pos	PID : sensitivity of GPS position reaction
Navigation Rate	GPS Nav	

Understanding impact of P, I and D

P : this is the amount of corrective force applied to return the Aircraft back to its initial position

The amount of force is proportional to a combination of the the deviation from initial position minus any command to change direction from the controller input. A higher P value will create a stronger force to resist any attempts to change it's position. If the P value is too high, on the return to initial position, it will overshoot and then opposite force is needed to compensate. This creates an oscillating effect until stability is eventually reached or in severe cases becomes completely destabilized.

I : this is the time period for which the angular change is sampled and averaged

The amount of force applied to return to initial position is increased by the I factor the longer the deviation exists until a maximum force value is reached. A higher I will increase the angular hold capability.

D : this is the speed at which the Aircraft is returned to its original position

Increasing value for D: Improves the speed at which deviations are recovered

With fast recovery speed comes a higher probability of overshooting and oscillations
Will also increase the effect of P

P – proportional

P provides a proportional amount of corrective force based upon the angle of error from desired position. The larger the deviation, the larger the corrective force.

A higher P value will create a stronger force to return to desired position. If the P value is too high, on the return to initial position, it will overshoot and then opposite force is needed to compensate. This creates an oscillating effect until stability is eventually reached or in severe cases, the overshoot becomes amplified and the multi-rotor becomes completely destabilized.

Increasing value for P :It will become more solid/stable until P is too high where it starts to oscillate and lose control. You will notice a very strong resistive force to any attempts to move the Multi-Rotor

Decreasing value for P: It will start to drift in control until P is too low when it becomes very unstable. Will be less resistive to any attempts to change orientation

Aerobatic flight: Requires a slightly higher P

Gentle smooth flight: Requires a slightly lower P

I – Integral

“I” gain provides a variable amount of corrective force based upon the angle of error from desired position.

The larger the deviation and / or the longer the deviation exists, the larger the corrective force. It is limited to prevent becoming excessively high.

A higher I will increase the heading hold capability

Increasing value for I: Increase the ability to hold overall position, reduce drift due to unbalanced frames etc

Decreasing value for I: Will improve reaction to changes, but increase drift and reduce ability to hold position

D- Divide / Derivative

This moderates the speed at which the Aircraft is returned to its original position.

A lower D will mean the Multi-Rotor will snap back to its initial position very quickly

Increasing value for D: Dampens changes. Slower to react to fast changes

Decreasing value for D: Less dampening to changes. Reacts faster to changes

Aerobatic flight: Lower D

Gentle smooth flight: Increase D



Basic PID Tuning – on the ground

Roll	P 1.2	I 0.044	D 16	RC Expo 0.65	Rate: 0.90 Expo: 0.65
Pitch	P 1.2	I 0.044	D 16	RC Rate 0.90	
Yaw	P 3.0	I 0.035	D 0		

Click on Write settings
after changes made in
any of the parameters to
save

- 1 - Set PID to the designers default recommended settings.
- 2 - Hold the Aircraft securely and safely in the air.
- 3 - Increase throttle to the hover point where it starts to feel light.
- 4 - Try to lean the Aircraft down onto each motor axis. You should feel a reaction against your pressure for each axis.
- 5 - Change P until it is difficult to move against the reaction. Without stabilization you will feel it allow you to move over a period of time. That is OK
- 6 - Now try rocking the aircraft. Increase P until it starts to oscillate and then reduce a touch.
- 7 - Repeat for Yaw Axis. Your settings should now be suitable for flight tuning.

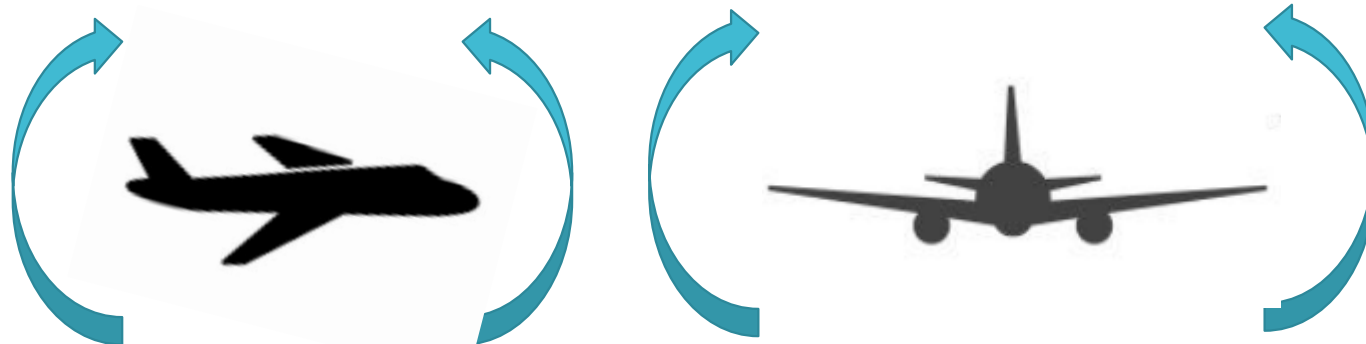


You will have to accept a compromise of optimal settings for stable hover and your typical mode of flying.

Obviously factor it towards your most common style.

Other factors affecting PID Taking known good PID values from an identical configuration will get you close, but bear in mind no two Aircraft will have the same flying characteristics and the following items will have an impact on actual PID values:

- 1 - Frame weight /size / material / stiffness
- 2 - Motors - power / torque /momentum
- 3 - Position - Motor-->motor distance (I.E. frame size)
- 4 -ESC / TX - power curves
- 5 - Prop - diameter / pitch / material
- 6 - BALANCING
- 7 - Pilot skills





Advanced Tuning - practical implementation

For Aerobatic flying: Increase value for P until oscillations start, then back of slightly

Change value for I until wobble is unacceptable, then decrease slightly

Decrease value for D until recovery from dramatic control changes results in unacceptable recovery oscillations, then increase D slightly Repeat above steps

For stable flying (RC): Increase value for P until oscillations start, then back of quite a bit

Decrease value for I until it feels too loose /unstable then increase slightly
Increase value for D



Click on Write settings after changes made in any of the parameters to save

PID : Level

Level		
P	8.0	I
	0.002	D
	80	

This will influence the flight characteristic with an accelerometer : this is Level Mode

P is the dominant part of autolevel mode.

I will tell how much force must be applied when the measured angle error persists

D is used to clamp the maximum correction for autolevel mode

Increase value for P will make the autolevel mode stronger

For smooth operation the sum of P axis + P level should stay near the default value : if you decrease P for Roll and Pitch axis you can increase P Level





ALT Hold

Altitude

P	3.4	I	0.080	D	45
---	-----	---	-------	---	----

Click on Write settings after changes made in any of the parameters to save

The Barometer sensor is used to detect the altitude of your aircraft and is used for altitude hold mode. As the barometer sensor is not very precise and is quite noisy, detection of small up and down movements is impossible.

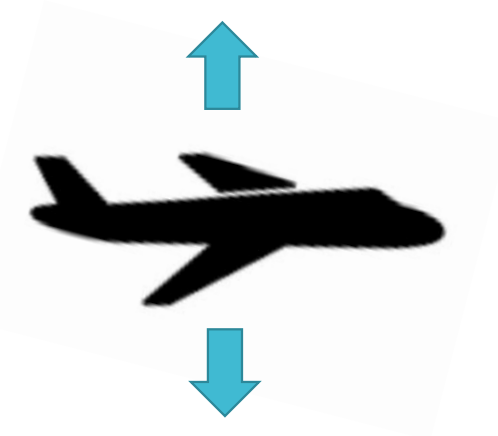
So small up and down movements are detected by the accelerometer Z axis. Combination of these two sensors gives good altitude hold.

PID settings for ALT works like this:

P - Is how much the Aircraft should rely on the barometer sensor. The higher the value is the stronger the multirotor relies on the Barometer reading.

I - Is used to compensate for drift caused by battery voltage drop during the flight. The higher the value is more the multirotor will react to voltage drops (or other varying factors over time).

D - Is how strong the Aircraft should react to data from the accelerometer Z axis. It is used to react to small up and down movements that the barometer cannot accurately sense. The higher the value is the stronger the Aircraft will react to small altitude changes.



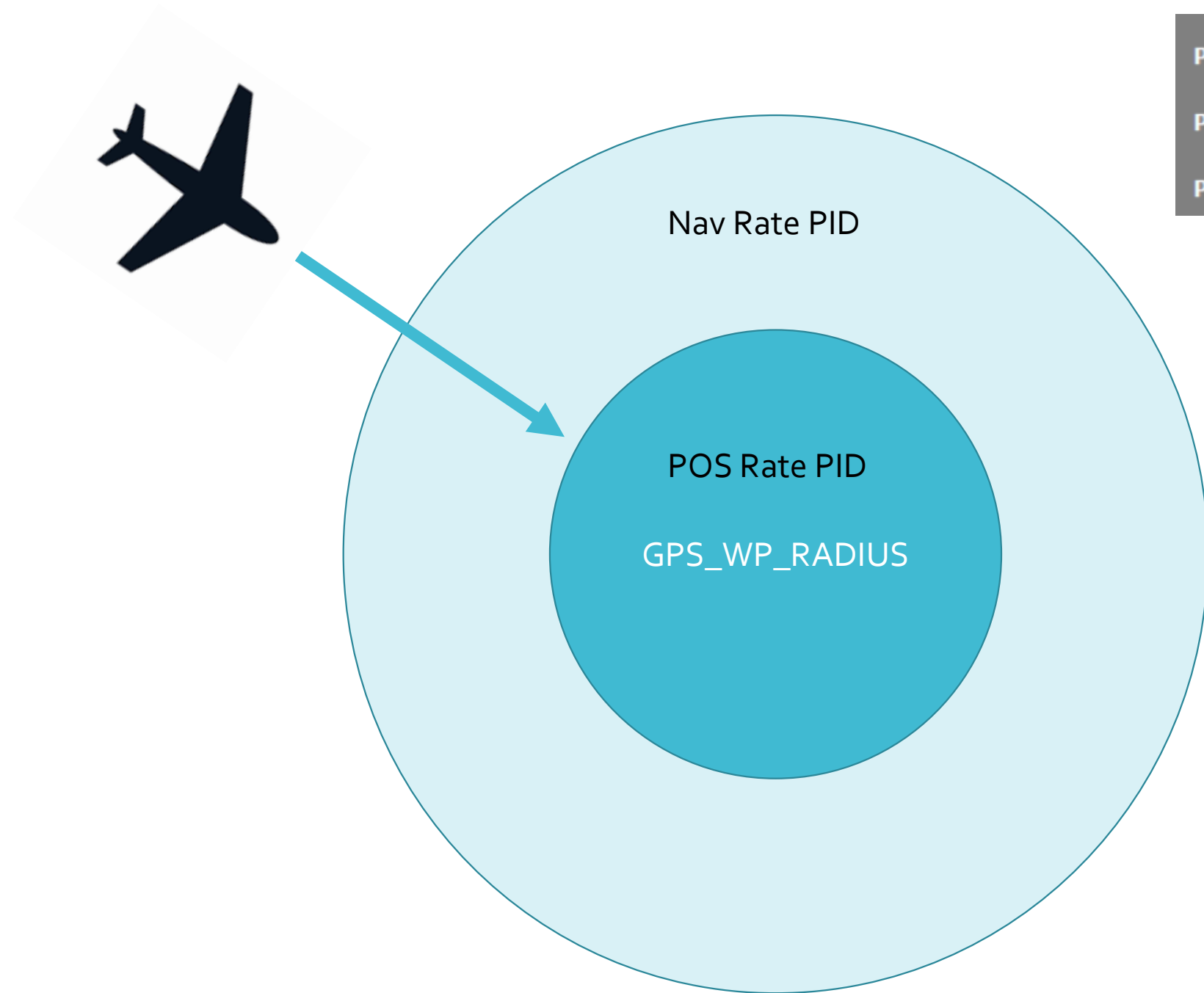
1. So set the P and I to 0
2. Start to play with D value only. Too high D may cause yoyo effect (up and down oscillations). With too low D the copter will be not able to react strong/fast enough to hold altitude. Your goal here is to set D to the value when copter doesn't oscillate up and down and also holds altitude quite well for a not very long period of time. Aircraft will not hold altitude perfectly at this point during long periods. It will slowly drift up or down, but altitude should be quite stable in short periods.
3. Start to increase P to the point where Aircraft holds altitude over long time period. If the value is too small the copter will drift slowly up and down. If the value is too high yoyo effect may appear. Goal here is to set it to the point where copter holds altitude for quite some time. Aircraft will still go slowly down due to battery voltage drop over time.
4. "I" is used to compensate the voltage drop. So start to increase the "I" value slowly until you get a perfect position hold during a very long time.
Now your altitude hold should be good enough.

- **For Mega 2560 + GPS Pos Rate PID controller & Pos Rate PID Tuning**
- Pos Rate PID controller
- Pos Rate PID Tuning
- The Pos Rate PID controller takes the commanded speed output from the Pos PI controller and commands an attitude in order to maintain the position hold location. This PID controller should be tuned before adjusting the Pos PI controller.
- The Pos Rate PID settings control how the attitude of the Vehicle is changed in order to move towards the desired hold location.
- The speed of movement is controlled by the Pos PI controller, while the attitude of the **Aircraft** is controlled by Pos Rate.
- When the **Aircraft** is within the defined distance of the hold location or waypoint (set by GPS_WP_RADIUS in config.h) the Pos Rate PID is used, when further away from the location the Nav Rate PID is used to return to within the defined waypoint radius.

- **For Mega 2560 + GPS Pos Rate PID controller & Pos Rate PID Tuning**
- The Pos Rate PID controller takes the commanded speed output from the Pos PI controller and commands an attitude in order to maintain the position hold location. This PID controller should be tuned before adjusting the Pos PI controller.
- The Pos Rate PID settings control how the attitude of the Vehicle is changed in order to move towards the desired hold location.
- The speed of movement is controlled by the Pos PI controller, while the attitude of the **Aircraft** is controlled by Pos Rate.
- When the Aircraft is within the defined distance of the hold location or waypoint (set by GPS_WP_RADIUS in config.h) the Pos Rate PID is used, when further away from the location the Nav Rate PID is used to return to within the defined waypoint radius.
- To tune the Pos Rate PID, initially set P, I and D values to 0.
- Gradually increase P until the Aircraft begins to position hold with some drift.
- Gradually increase D until the Aircraft responds more quickly to undesired changes in attitude caused by the wind. If this value is set too high you will see oscillations or sudden jerking in pitch and roll motion.
- If needed, gradually increase I value to allow the PID controller to compensate for long lasting error, ie if it is being blown by the wind.

- To tune the Pos Rate PID, initially set P, I and D values to 0.
- Gradually increase P until the Aircraft begins to position hold with some drift.
- Gradually increase D until the aircraft responds more quickly to undesired changes in attitude caused by the wind. If this value is set too high you will see oscillations or sudden jerking in pitch and roll motion.
- If needed, gradually increase I value to allow the PID controller to compensate for long lasting error, ie if it is being blown by the wind.

PosHold				
P	0.15	I	0.00	
PosHoldRate				
P	3.4	I	0.14	D 0.053
Navigation Rate				
P	2.5	I	0.33	D 0.083

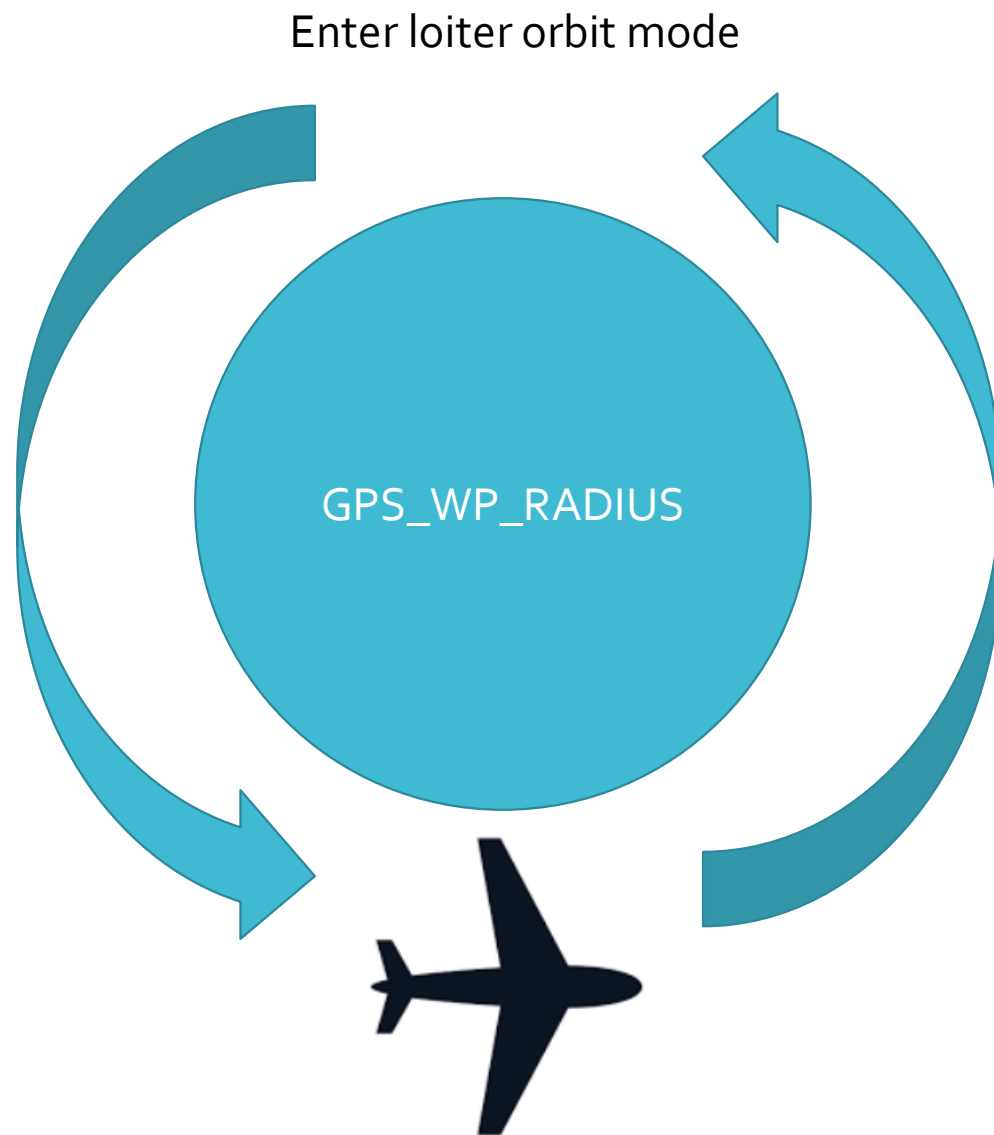


The Aircraft suppose to heading towards the waypoint

When the aircraft goes pass the waypoint

Mission – fly to next waypoint

GPS Hold – Aircraft would orbit the Waypoint



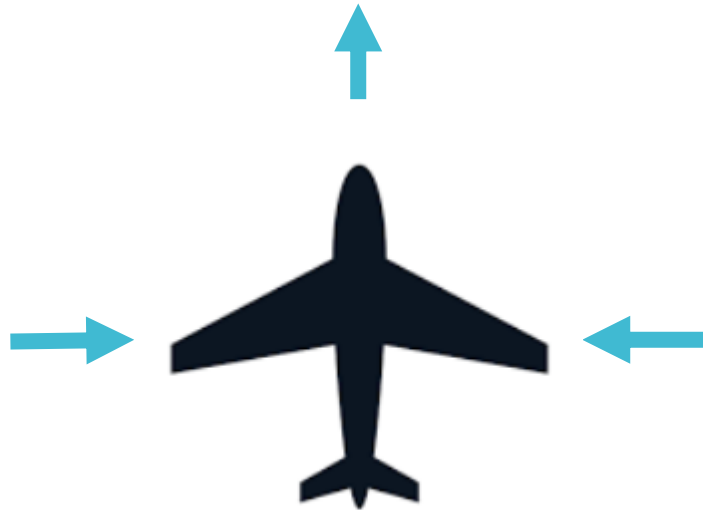
PosHold				
P	0.15	I	0.00	
PosHoldRate				
P	3.4	I	0.14	D 0.053
Navigation Rate				
P	2.5	I	0.33	D 0.083

To tune the Pos Rate PID, initially set P, I and D values to 0.

Gradually increase P until the multirotor begins to position hold with some drift.

Gradually increase D until the Aircraft responds more quickly to undesired changes in attitude caused by the wind. If this value is set too high you will see oscillations or sudden jerking in pitch and roll motion.

If needed, gradually increase I value to allow the PID controller to compensate for long lasting error, ie if it is being blown by the wind.





Port **COM17** Speed **115200**

Disconnect

Read Settings

Write Settings

Load Defaults

Load from File

Save to File

Start Log

Start GPS log

Log Browser

Flight Deck Mission Flight Tuning FC Config **RC Control Settings** Sensor Graph VideoCapture GUI Settings CLI

	AUX1			AUX2			AUX3			AUX4		
	L	M	H	L	M	H	L	M	H	L	M	H
ARM												
ANGLE												
HORIZON												
BARO												
MAG												
HEADFREE												
HEADADJ												
CAMSTAB												
CAMTRIG												
GPS HOME												
GPS HOLD												
PASSTHRU												
MISSION												
LAND												

RC Control Settings

Use Aux switch to setup flight modes and Navigation functions

HORIZON- keeps the aircraft Level

ARM – this is option should you decided to use a Aux switch oppose to the Combination Stick input to Arm/Disarm Drone. Some Airframes require Arm switch

BARO – Altitude Hold
MAG –Heading Hold

GPS Home – Return to Home (aircraft will climb to RTH altitude then Fly to Launch point)

GPS Hold – Hold Position , Orbit behavior or Stall behavior for fixwings with STOL capabilities

MISSION – fly a mission save from the mission tab

PASSTHRU- Disable all gyro Functions Manual

Thr 1500

Roll 1500

Yaw 1500

Aux1 1500

Aux2 1500

Live RC data

Orange border indicates, that setting was changed but not written to FC

MODES and AUX Setting

- Preferably to reserve 1 Aux channel as Arm switch as combination stick could make the vehicle move prematurely
- Then the alternative Aux for the Mission or RTH mode

MODES and AUX Setting

Passthru

Sends Rc commands direct to servos.
No influence from sensors.

Gyro Mode (Acro)

This is "Normal" mode when nothing else is selected.
The plane should compensate for movements. (Wind Gusts etc)
The plane feels stable and locked in but still able to loop & roll.
Stall speed is lower and it can be necessary to "Push" it down in landings.

Stable Modes

With the sticks centered the plane will self stabilize.
Returning to level flight from almost any situation.
Provided there's enough Altitude for recovery.

Horizon Mode Allows rolls and loops. Levels with centered sticks
This is a comfortable flight mode for FPV.

Angle Mode also limits how much the plane can tilt.
Gives a Stiff feeling and is only recommended for beginners.

MODES and AUX Setting

GPS Flight Modes

Missions

Puts the aircraft into waypoint mode set Aux switch to Mission & Horizon or Angle mode

GPS-Hold

Can only be enabled With AUX switch.

Check both Angle mode and GPS Hold in Gui.

When GPS-Hold is activated The position is saved in a 3D Waypoint.

The plane will Navigate and try to "hit" the WP continuously and maintaining altitude.

No pattern is programmed and the plane fly the shortest way back.

Often in a circle or figure eight.

MODES and AUX Setting

RTH (Return to home)

Can be enabled With AUX switch or by Failsafe.

Check both Angle mode and RTH in Gui.

When RTH is activated the plane will start Climb to reach safe Altitude.

If RTH is enabled Higher than set altitude it will start navigation and keep that altitude.

If altitude is safe the plane will start to Navigate to home Position.

Only use Angle/Horizon + Gps Home/Hold together.

Do **NOT** Activate BARO Or MAG for navigation.

It will interfere with the navigation code.

When the plane reaches **SAFE_DECSCEND_ZONE** the plane will begin descending to correct altitude.

The plane will keep flying in hold mode and continuously pass home.

If Failsafe is active at return The plane will Disarm motor and descend to a "Landing"

PID Settings

PID settings is made in Gui.
ALT & NavR.

Return Altitude can be set with
RTH Altitude
Scale is in Meters
15m is set as default.

WP Radius (cm)	RTH Altitude (m)	Crosstrack gain
400	0	0.40
Max Nav speed (cm/s)	Min Nav speed (cm/s)	
400	100	
Max Nav banking (degree)		Land Speed
20		100
Safe WP distance (m)	Max Nav Altitude (m)	Fence radius (m)
500	100	600



Missions

Note: Only functional for Mega 2560 Boards with GPS

Waypoint – the drone will travel between those **points**

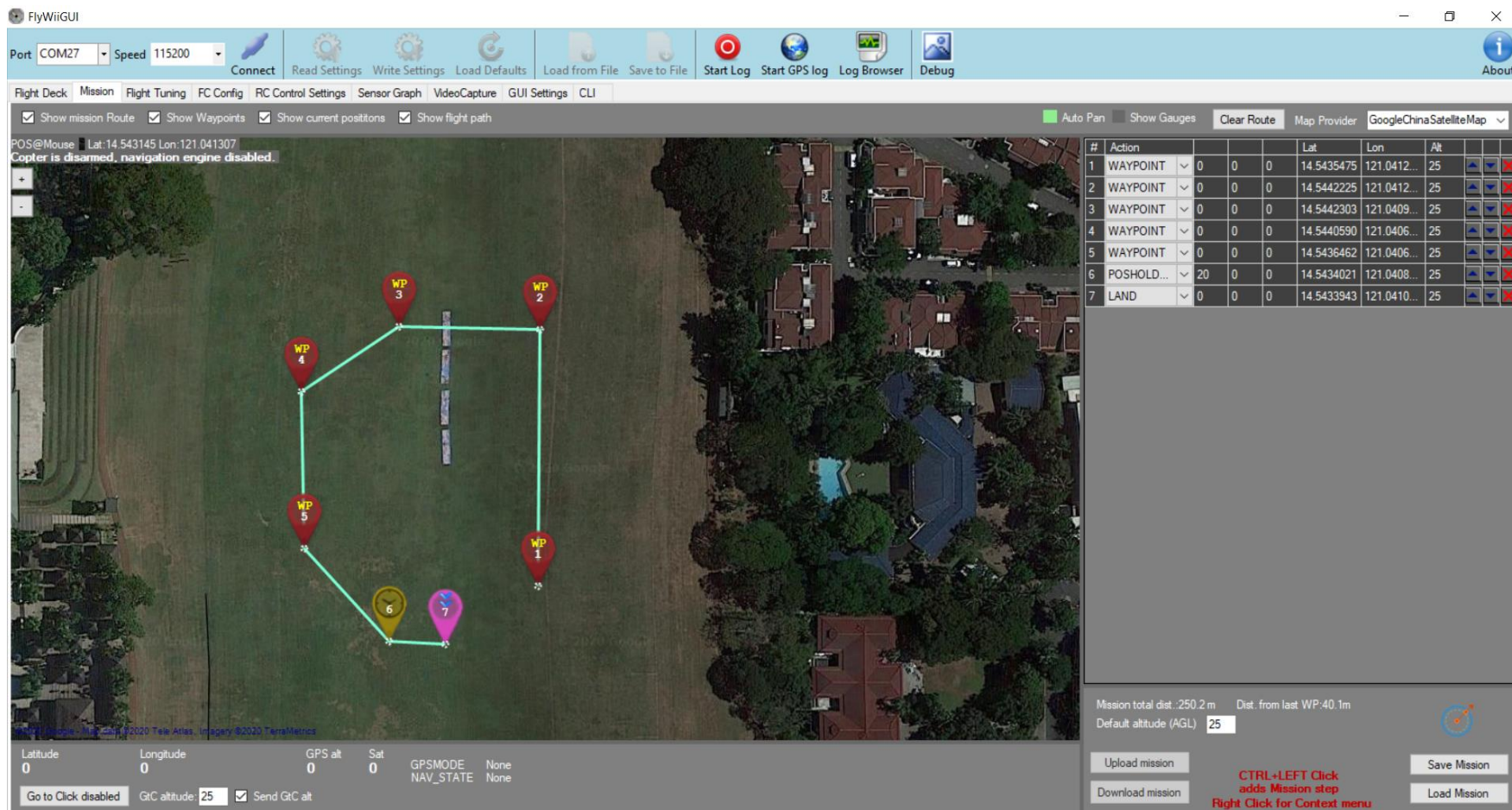
Time PosHold – Drone will wait X number of 00:00:00 then move to the next waypoint (limited for fixwing)

Unlimited PosHold – once the plane reach this point it will attempt to STOL Wait till you switch out of Mission mode

Land – the drone will land once it has reach this point (**Must be place at the end of the mission**)

RTN – the Drone will fly back to home position (**Must be place at the end of the mission**)

Default Alt – Altitude in meters (**for first Mission test waypoint with altitude 2m-3m Above Ground Level**) And set missions with 2m-3m altitude with Nav speed of 100cm/s .



RC Control Setting Tab – activate Baro , Mag , Mission

To start mission takeoff aircraft in stabilize mode up to 1-2meter altitude then switch the aux switch to mission mode .

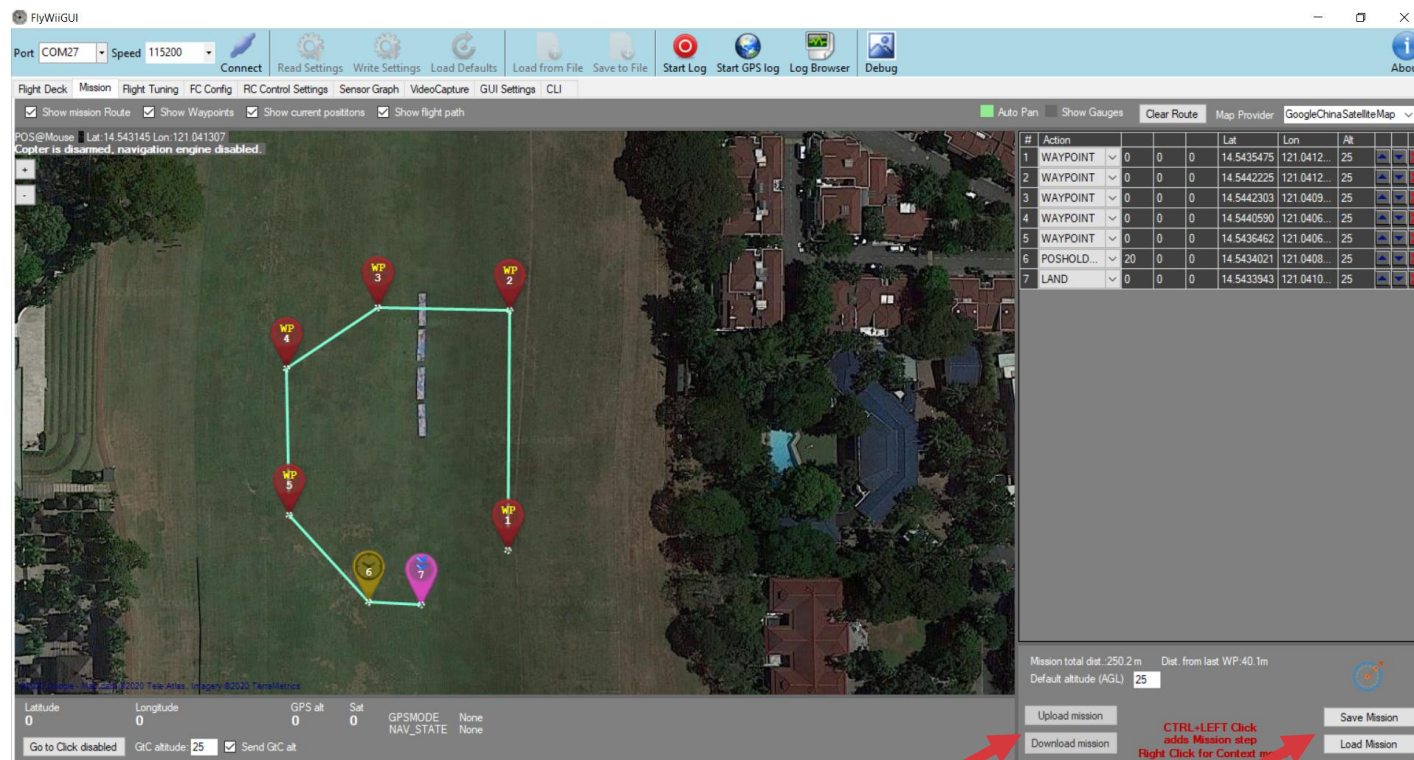
Any time you can switch out of it on hold or stabilize mode



Missions

Prerequisite and process for a good mission , Points to test before performing a mission

1. Plane is flying stable in horizon and Alt hold mode , holding altitude consistently less than 1m variation over 1 minute period . Tune PID and altitude PID when necessary. Return to passthru for manual control
2. RTH – set RTH Altitude to 0 , Max Nav speed to 100cm/s , set aux switch to RTH ,Horizon/Angle and write settings ,Fly the drone 5 Meters away from the Launch site and activate the RTH Aux switch ,see if the drone returns back to home position and holds position when arrive . Tune Navigation Rate PID when necessary



Mission upload to /download from plane

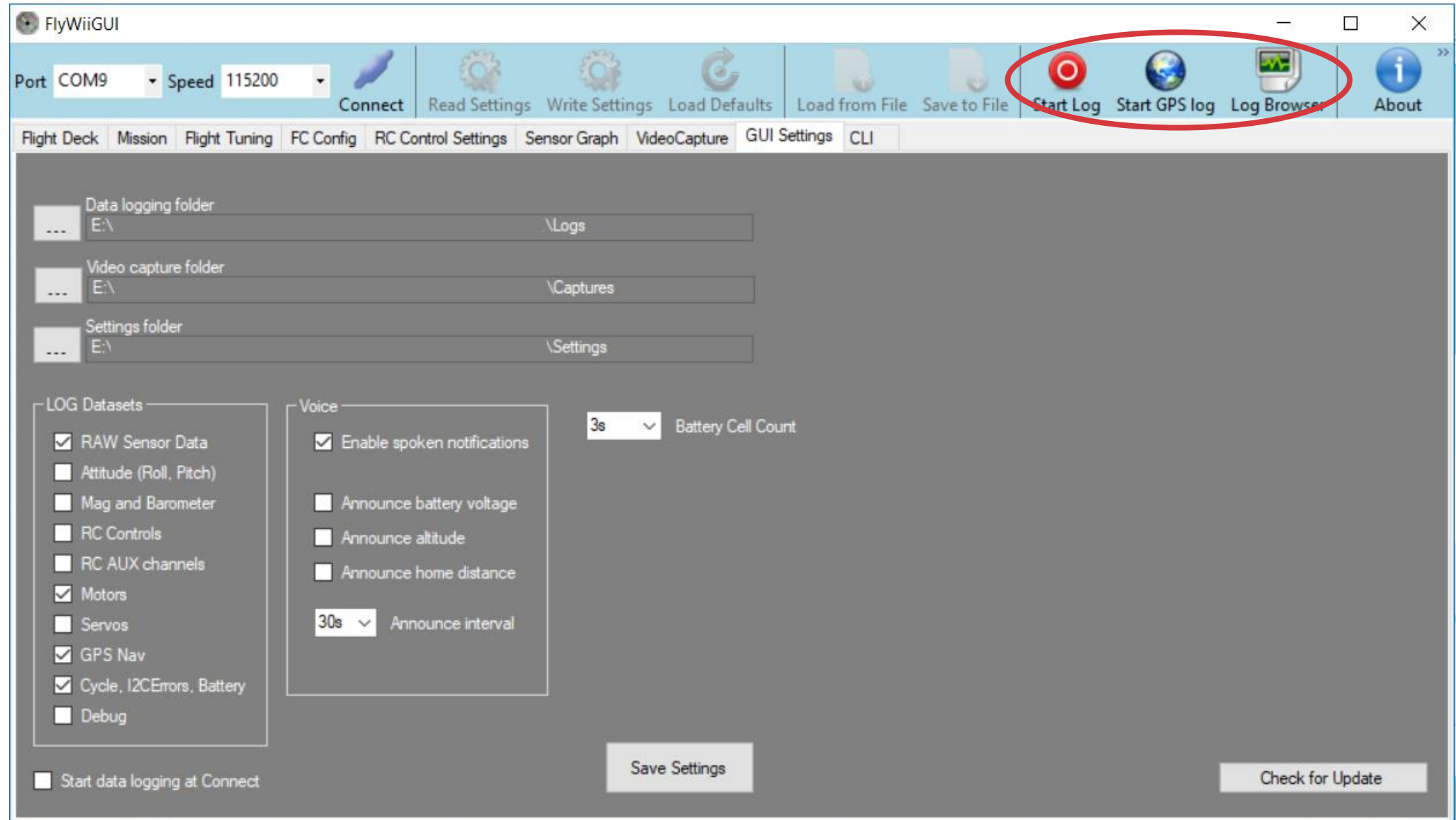
Mission Save to /Open from File



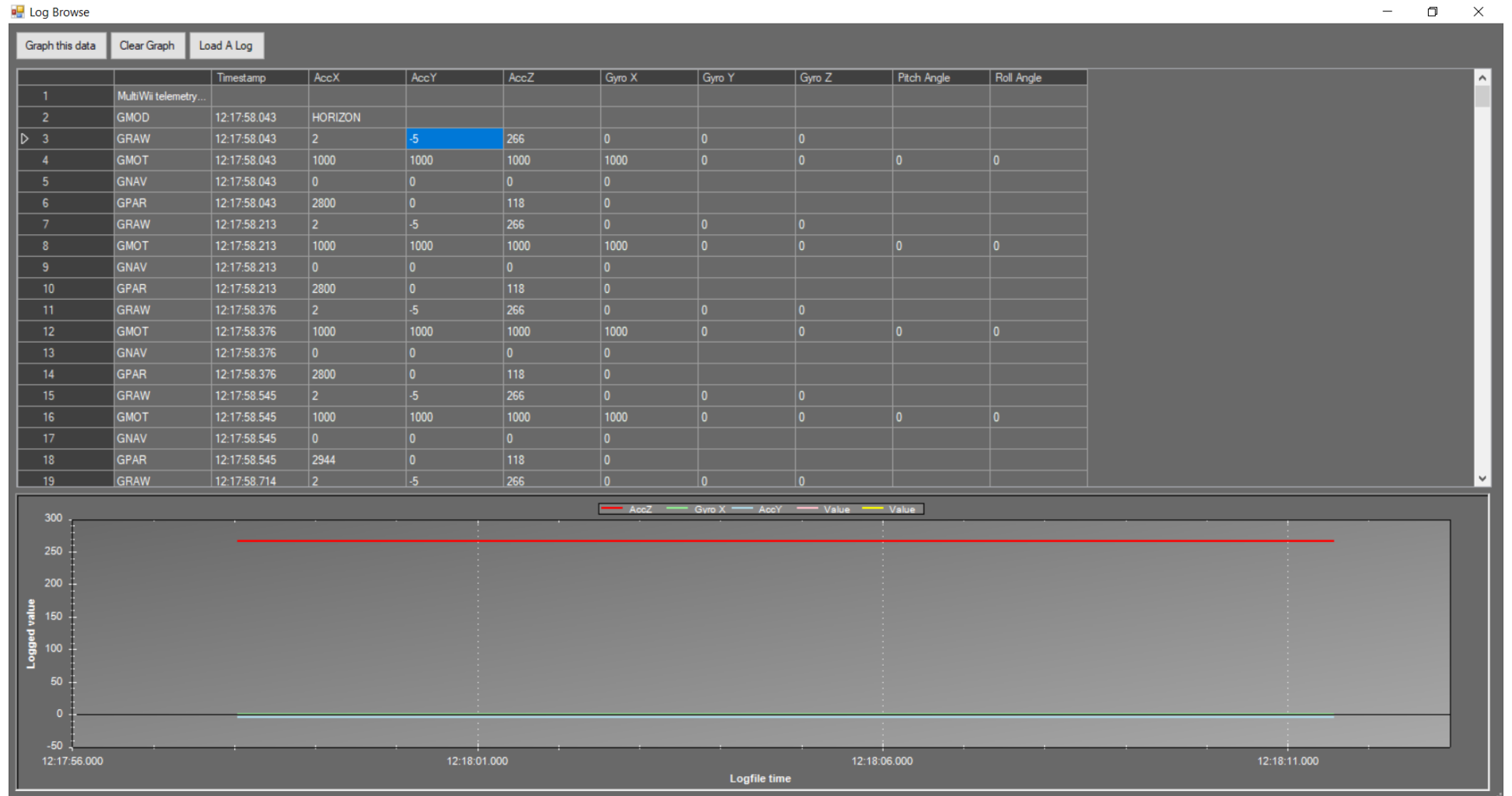
Graphs and Data Logging



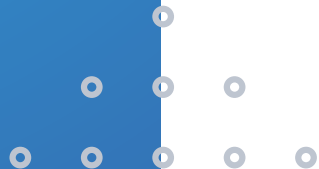
GUI Settings (where you save your PID ,Flight Logs and Video Logs)



Log Browser



PRE-FLIGHT



Preflight setup

After you have changed the servo Rates
you can set Dual rates and Expos in the Transmitter.
Engine must be *Armed* to prevent motor start by accident..
It can be Armed from AUX-channel if it's setup in the gui . (recommended)
Or with stick combination min throttle & max rudder.

First Flight.

Take of in Passthru.
Switch mode on safe height.
Activate Assisted modes (Horizon /Angle) and feel the difference.

Level-P value will Reduce the maximum throw in Level-Mode.

And your much Done on your setup

Cannot Arm Motors

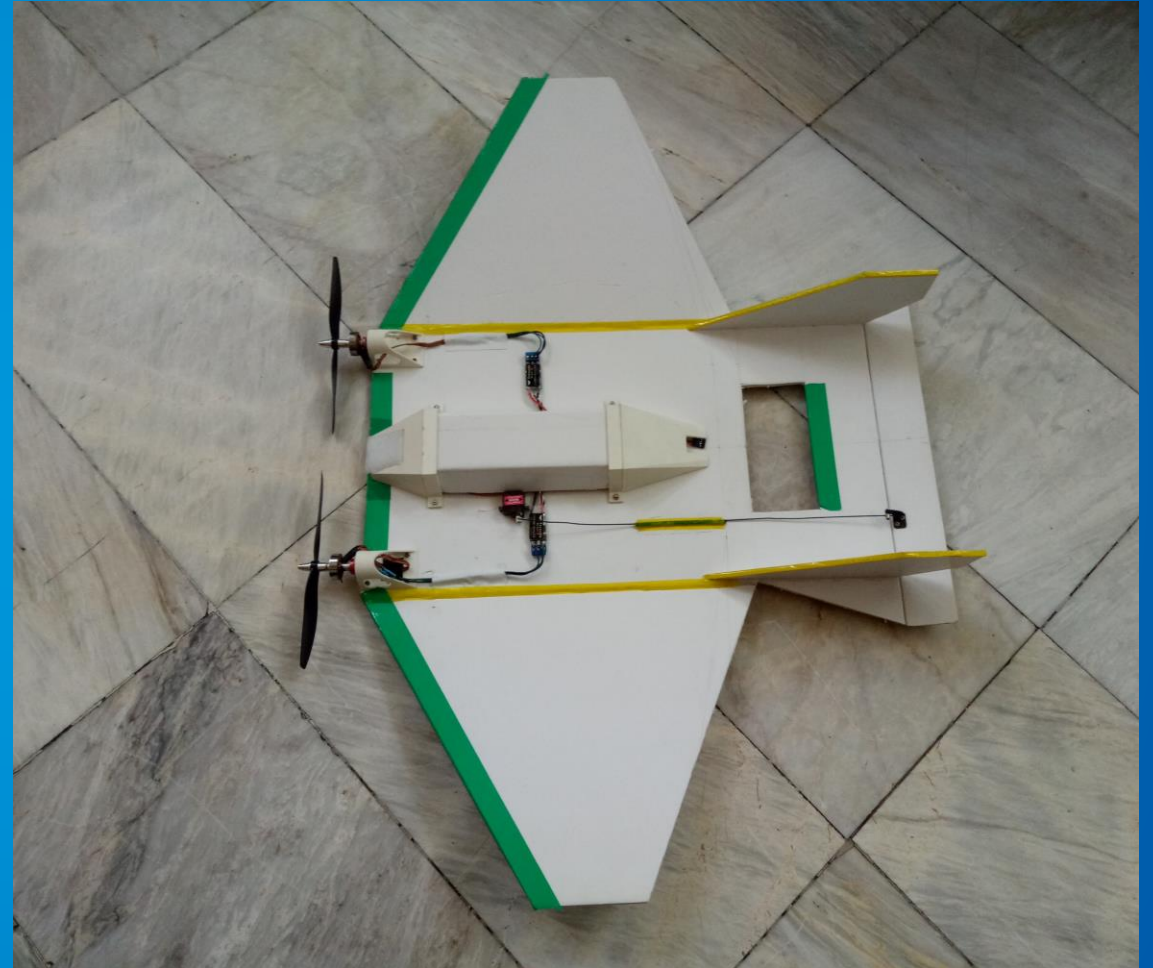
when on GPS Home , GPS Hold , Mission Flight modes & when USB is plugged in . (pls use Bluetooth telemetry)

Tests motor with Props off

Baro and Mag preferably switch off when Arming

Pls calibrate ACC and Mag in the FlyWii GUI Dashboard

Ensure the compass is facing the correct orientation



Preflight setup

After you have changed the servo Rates you can set Dual rates and Expos in the Transmitter.

Engine must be *Armed* to prevent motorstart by accident..

It can be Armed from AXU-channel if it's setup in the gui . (recommended)

Or with stick combination min throttle & max rudder.

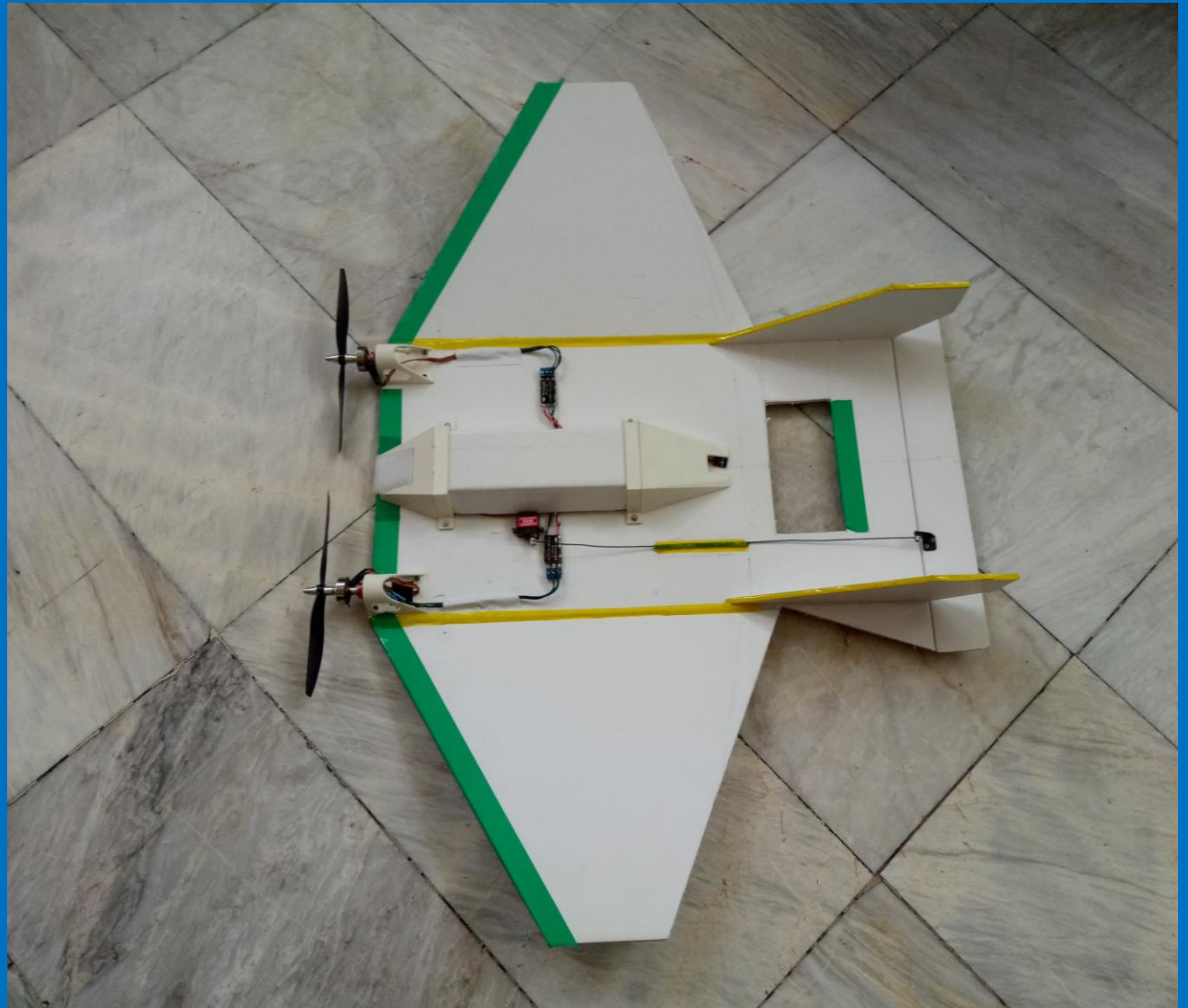
In RC control Setting set Aux 1 as Arm

First Flight.

Take of in Passthr.

Switch mode on safe height.

Activate Assisted modes and feel the difference.



LED INDICATOR

GPS LED

D31

Status LED

D30

No GPS



Motors Disarm



4 Sats



Not Level



5 Sats



Motors Arm



6 Sats



7+ Sats



indicate a valid GPS fix by flashing the LED

- led work as sat number indicator
- No GPS FIX -> LED blinks constant speed
- Fix and sat no. below 5 -> LED off
- Fix and sat no. ≥ 5 -> LED blinks, one blink for 5 sat, two blinks for 6 sat, three for 7 +



SYNERFLIGHT

SYNERDUINO

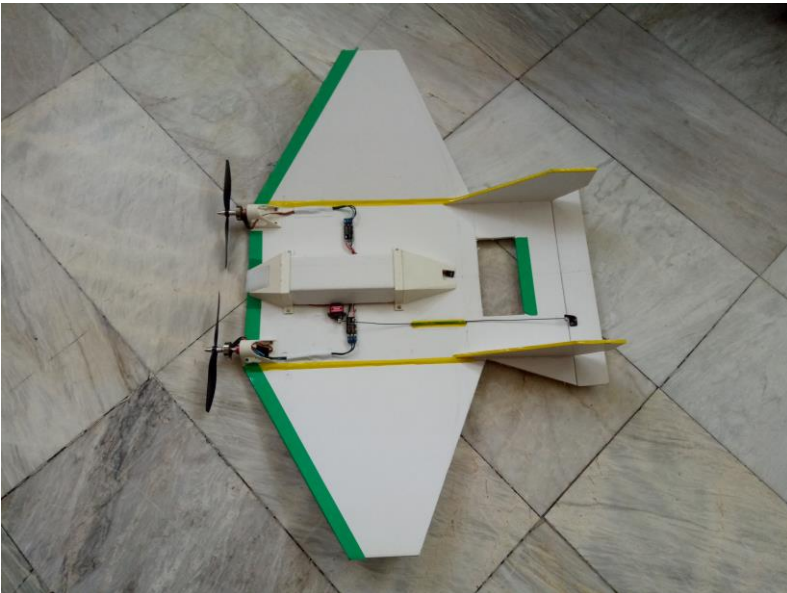
ARDU 2560



Conventional 4Ch AETR



Conventional 3Ch RET



Differential Thrust 3Ch RET



Flying Wing 3Ch

First timer for Autonomous RC Aircraft

We recommend that you choose a sizeable slow flying plane. Your comfortable flying

Either build from scratch or an available kit

Large enough to fit all your equipment and slow enough

To fly comfortably

This way its easy to tune and flown manually should needed