Chapter 1: Introduction to ArduPilot

ArduPilot is a powerful and flexible open-source autopilot software that can control many types of unmanned vehicles: aerial drones (like quadcopters, hexacopters, and octocopters), ground vehicles (rovers), surface boats (USVs), and even underwater robots (ROVs). It runs on a flight controller, which is the main onboard computer that reads sensors and sends commands to motors or servos.

With more than ten years of development, ArduPilot is one of the most advanced autopilot systems available. It is supported by a large community and is used in research, education, and industry.

What ArduPilot Does

At its core, ArduPilot is responsible for:

- Reading data from sensors like GPS, compass, barometer, and gyroscopes
- Estimating position and orientation
- Controlling motors and servos to move the vehicle
- Running missions, avoiding obstacles, and returning home safely

It uses a modular software architecture that allows support for different vehicle types and hardware platforms.

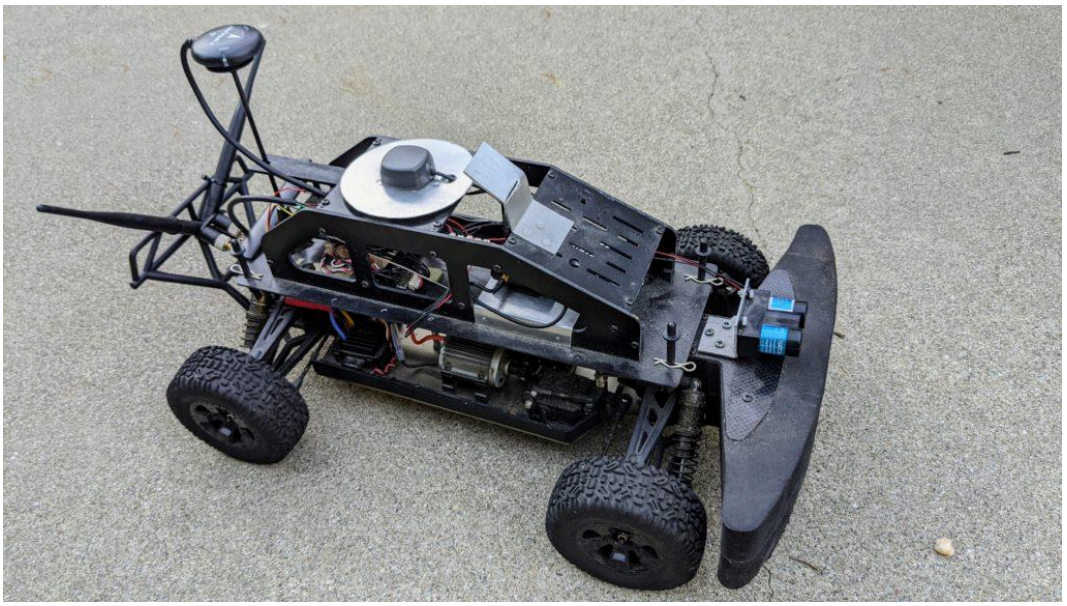
Supported Vehicle Types

ArduPilot works with multiple vehicle types by switching firmware and adjusting parameters:

- **ArduCopter:** Multirotor drones like quadcopters, hexacopters, helicopters, etc)

- **ArduPlane:** Fixed-wing aircraft and VTOL (Vertical Take-Off and Landing)

- **ArduRover:** Ground vehicles and surface boats (UGVs & USVs)



- **Sub:** Underwater robots (ROVs)

- **Other:** Blimps and airships

Each vehicle type has its own set of flight modes and features, but they all share the same core logic for sensors, control, and mission handling.

USVs in ArduPilot

When using ArduPilot's Rover firmware with `FRAME_CLASS = 2`, it activates USV-specific features such as:

Station-Keeping (Loiter Mode)

The USV can hold its position using GPS. It applies just enough thrust to stay within a small area. This is useful for sampling or waiting.

Differential Thrust Control

USVs can turn and steer using two motors spinning in opposite directions (skid steering). This method allows precise maneuvering without a rudder.

Marine Sensor Support

ArduPilot can connect to sensors like sonar, water speed sensors, and depth finders. These can be used for navigation, obstacle detection, or environmental monitoring.

ArduPilot Software Architecture

The software is structured into layers, each with a specific role:

1. Hardware Abstraction Layer (HAL)

This layer allows ArduPilot to work on different autopilot boards (Pixhawk, Cube, Matek, etc.). It translates the same flight code to different hardware. It handles:

- Reading from sensors (GPS, compass, IMU)
- Sending commands to motors/servos
- Communication through UART, I2C, CAN, or SPI

This design enables the same codebase to run on various autopilot boards, from STM32-based systems to Linux-based computers (such as Raspberry Pi or Jetson).

2. Shared Libraries

These are common software components used across all vehicles:

- **Sensor drivers** to read GPS, barometer, sonar, and more
- **EKF (Extended Kalman Filter):** Combines data from multiple sensors (e.g., GPS, IMU, compass) to estimate the accurate position, speed, and orientation of the vehicle, even in noisy or challenging environments
- **PID control algorithms:** It helps the vehicle stay stable by changing the motor or servo outputs. It uses three parts: proportional, integral, and derivative. We need to tune it correctly to get smooth and accurate movement
- **Navigation logic:** Responsible for path following, mission execution, obstacle avoidance, and safety behaviors

These libraries allow customization without needing to write new code from scratch.

3. Vehicle-Specific Code

Each vehicle type has a special layer with its unique logic:

- **Copter**: Flight modes and motor mixing
- **Plane**: Takeoff and landing logic
- **Rover**: Driving and turning logic
- **Sub**: Depth control and underwater behavior

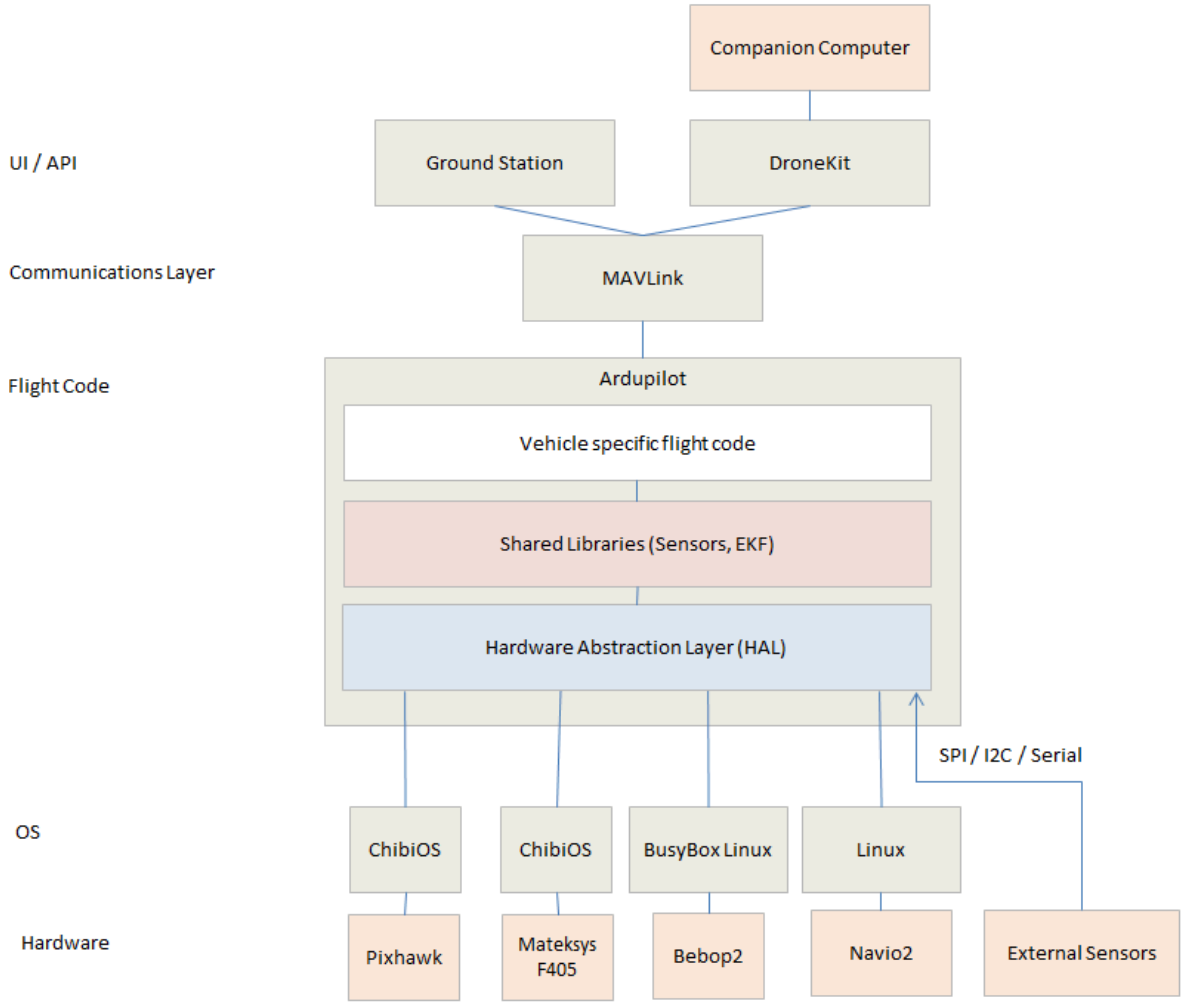
4. Scheduler (Task Manager)

The scheduler controls how fast each part of the software runs:

- IMU updates: 400 Hz
- Navigation logic: 50 Hz
- Output to motors: as needed by control loops

The scheduler makes sure that important tasks run on time.

If one part, like a sensor, is slow or not working well, the main system still works. This helps the boat stay safe and under control, even if something goes wrong.



ArduPilot Ecosystem

ArduPilot works with many tools and components that support mission planning, communication, and advanced features.

Autopilot Hardware (Flight Controller)

Examples: Pixhawk 2.4.8, Cube Orange+, Matek H7 series, etc

- Runs the ArduPilot firmware
- Connects to all sensors and outputs
- Controls the vehicle in real-time
- Should be mounted near the vehicle's center of gravity, away from high-current power wires, ESCs, or sources of electromagnetic interference

Ground Control Station (GCS)

Examples: Mission Planner (Windows), QGroundControl (all platforms)

- Used to plan missions and upload them to the vehicle
- Shows live data, logs, and status information
- Allows mode switching, waypoint updates, and parameter tuning
- On Android-based remote controllers, apps like QGroundControl can run directly on the screen
- If no onboard Android system is present, GCS can run on a separate laptop or tablet, connected via telemetry (wireless) or USB (wired)

MAVLink Protocol

MAVLink is a lightweight, open communication protocol used by ArduPilot. It enables real-time interaction between the autopilot and external systems:

- Sends telemetry data (position, battery, sensor values) to the GCS
- Receives commands like flight mode changes, waypoint uploads, or parameter updates
- Allows communication with companion computers for AI, vision, and mission logic
- Used for diagnostics, scripting, and system monitoring

MAVLink serves as the primary interface between ArduPilot and the external world, making it a key element of autonomy and system integration.

Companion Computers

Examples: Raspberry Pi, Jetson Nano

- Used for advanced tasks: AI, computer vision, ROS integration
- Communicate with the flight controller via MAVLink
- They can make important decisions or help understand data from the environment.

Summary

- ArduPilot is an open-source system that supports many vehicle types
- The architecture includes HAL, shared libraries, vehicle-specific logic, and a real-time scheduler
- USVs are supported through the Rover firmware (`FRAME_CLASS = 2`)
- The system uses GPS, motors, ESCs, and other sensors to control the vehicle
- The ecosystem includes autopilot hardware, GCS (local or remote), telemetry, and optional companion computers
- MAVLink provides the essential link between ArduPilot and external systems

Chapter 2: Requirements Definition

This chapter presents the hardware requirements and basic design principles for building an Unmanned Surface Vehicle (USV) with ArduPilot. It focuses on the components used in the Summer School USV kit, explains differential thrust as the primary steering method, provides guidance for choosing compatible hardware, and details how to verify proper integration of all parts.

Communication Interfaces in ArduPilot Systems

All ArduPilot vehicles use different kinds of communication signals to connect the autopilot with other parts. Each signal has its own job, like moving motors, reading sensors, or sending data to the ground. Some common types of signals are:

- **PWM** – used to control ESCs or servos via individual signal wires.
- **UART** – used for devices like GPS modules, telemetry radios, or companion computers.
- **I2C** – used for digital sensors such as compasses or barometers.
- **CAN** – a high-speed, robust protocol found in modern GPS or smart ESCs.
- **PPM/SBUS** – used for RC input, allowing multiple channels over a single wire.

Each type of signal works in a different way and has good and bad sides. PWM is easy to use but needs more wires. PPM and SBUS use fewer wires but can be a bit slower. UART works with many devices but there are only a few ports. CAN is fast and strong but needs special hardware.

It is important to check what protocol each device supports and make sure the autopilot can handle it. The safest way is to open the datasheets of your components and confirm:

- The type of signal (PWM, UART, etc.)
- The voltage requirements
- The correct port or pin assignment

By matching these details correctly, you reduce the risk of communication errors and make the integration process smoother and safer.

Core Components of an ArduPilot-Based USV

To build a working USV, several components must be integrated:

The **autopilot** is the central unit running ArduPilot firmware. It connects to all sensors and actuators, reads their data, and sends commands in real-time. A stable power source is required, usually through a power module.

A **GPS module** and **compass** provide global position and heading. These are often combined in one device. While most GPS modules use UART for data and I2C for the compass, some advanced options use CAN, which supports faster and more reliable communication.

Motors provide propulsion. **Electronic Speed Controllers (ESCs)** drive the motors and must be reversible for full maneuverability. In ArduPilot, motors are controlled using PWM, DShot, or CAN protocols depending on ESC type.

A **telemetry module** enables real-time communication with a Ground Control Station (GCS). Some modern RC receivers include telemetry features, reducing the need for a separate telemetry module.

An **RC system** allows manual override and safety control. It must be electrically and protocol-compatible with the flight controller.

Each component must match others in terms of voltage, port type, and supported protocol.

Propulsion and Steering: Differential Thrust

The standard method used in this Summer School is **differential thrust**, also known as skid steering. Two motors are installed—one on each side of the vehicle—and their relative speeds are adjusted to control both forward motion and turns.

This setup does not require a rudder or steering servo. To turn left, the right motor spins faster than the left; to pivot in place, the motors spin in opposite directions. This makes differential thrust ideal for high-maneuverability boats like catamarans, especially in complex environments.

How to Choose Compatible Hardware

Choosing the right parts means thinking about both technical needs and real-world limits. It's good to follow two steps: first, define the mission and physical constraints of the vehicle. Then, check if the components are compatible by reading their technical specifications

1. Define operational needs

- **Mission profile**: Required speed, duration
- **Environment**: Saltwater or freshwater?
- **Physical limits**: Space in the hull, waterproofing, weight limits
- **Power system**: Match battery voltage and capacity with motor and ESC needs
- **Sensor needs**: GPS, compass, sonar, temperature, or other environmental data

2. Verify compatibility through specifications

To confirm compatibility:

- Read the **datasheets** of each device
- Check required voltage ranges and signal types (e.g., PWM, CAN)
- Count available ports on the autopilot (UART, I2C, CAN)
- Look for firmware support in the ArduPilot documentation

Tip: Drawing a system map with all connections, voltages, and protocols before wiring helps prevent mistakes.

Real-World Configuration: Summer School USV Kit

In this section, we analyze the compatibility of the USV kit used in the Summer School. The goal is to show how each component matches with the others, based on voltage, ports, and protocols. This kind of analysis helps verify that the full system will operate safely and correctly.

Autopilot: Pixhawk 2.4.8

- **Input Voltage:** 3.3V to 6.6V (from power module)
- **Communication Interfaces:** UART (for GPS, telemetry), I2C (for compass), RC IN (PPM/SBUS)
- **Output Signals:** PWM (for ESCs and servos)

Specifications

- **Processor**
  - 32-bit ARM Cortex M4 core with FPU
  - 168 Mhz/256 KB RAM/2 MB Flash
  - 32-bit failsafe co-processor
- **Sensors**
  - MPU6000 as main accel and gyro
  - ST Micro 16-bit gyroscope
  - ST Micro 14-bit accelerometer/compass (magnetometer)
  - MEAS barometer
- **Power**
  - Ideal diode controller with automatic failover
  - Servo rail high-power (7 V) and high-current ready
  - All peripheral outputs over-current protected, all inputs ESD protected
- **Interfaces**
  - 5x UART serial ports, 1 high-power capable, 2 with HW flow control
  - Spektrum DSM/DSM2/DSM-X Satellite input
  - Futaba S.BUS input (output not yet implemented)
  - PPM sum signal
  - RSSI (PWM or voltage) input
  - I2C, SPI, 2x CAN, USB
  - 3.3V and 6.6V ADC inputs
- **Dimensions**
  - Weight 38 g (1.3 oz)
  - Width 50 mm (2.0")
  - Height 15.5 mm (.6")
  - Length 81.5 mm (3.2")

GPS + Compass: NEO-M8N Combo

- **GPS Interface:** UART (TX/RX), typically connected to TELEM1 or GPS port on the autopilot
- **Compass Interface:** I2C, used by the onboard compass chip (e.g., HMC5883L or IST8310)
- **Operating Voltage:** 1.65V to 3.6V; some modules include onboard regulators and accept 5V input.

Product Attributes

TYPE	DESCRIPTION
Category	RF and Wireless RF Receivers
Manufacturer	u-blox
Series	NEO-M8
Packaging	Tape & Reel (TR) Cut Tape (CT) Digi-Reel®
Part Status	Active
Frequency	1.575GHz
Sensitivity	-167dBm
Data Rate (Max)	-
Modulation or Protocol	BeiDou, Galileo, GLONASS, GNSS, GPS
Applications	General Purpose
Current - Receiving	21mA
Data Interface	I2C, SPI, UART, USB
Memory Size	-
Antenna Connector	-
Features	-
Voltage - Supply	1.65V ~ 3.6V
Operating Temperature	-40°C ~ 85°C
Mounting Type	Surface Mount
Package / Case	24-SMD Module
Supplier Device Package	-
Base Product Number	NEO-M8

The module connects to the dedicated GPS + I2C port of the Pixhawk. All electrical levels and port types are compatible.

RC Receiver: FlySky FS-iA6B

- **Output Protocol:** PPM or PWM (using adapter cable)
- **Voltage Range:** 4.0V to 8.4V

Product Model: FS-iA6B

PWM channel: 6

Wireless frequency: 2.4GHz

Wireless protocol: AFHDS 2A

Range: 500 ~ 1500m (in the air)

Antenna type: Dual copper tube antenna (150mm * 2)

Power: 4.0-8.4V

RSSI: Supported

Data port: PWM / PPM / i.bus / s.bus

Temperature range: -10 ℃ — + 60 ℃

Humidity range: 20% -95%

Online Update: Yes

Dimensions: 47 * 26.2 * 15 mm

Weight: 10g

Certification: CE, RCM, FCC ID: N4ZFLYSKYIA10

ESCs: Shark G2 30A (x2)

- **Input Voltage:** 2S–4S LiPo (7.4V to 16.8V)
- **Signal Input:** PWM
- **Max Current:** 40A

Specification:	
Model	Shark G2 30A 2-4S
Continuous Current(A)	30 Amps
Burst Current(A)	40 Amps
Voltage	2-4S Lipo/5-12NC
Power Wire	16 Gauge 70mm-Red/70mm-Black
Motor Wire	16 Gauge 70mm-Black
BEC Output	5V/6V Adjustable 4A
Size(not including wires)	60mm*25mm*10mm(L*W*H)
Weight	25g

These ESCs accept PWM from the Pixhawk and operate within the voltage range of the 3S battery.

Motors (Brushless): 300W Underwater Thrusters (x2)

- **Voltage Range:** ~12V to 16V
- **Current Draw:** ~20A-30A max
- **Use Environtment:** fresh water, sea water

Voltage: 12V-16V
Set propeller current: 20-30A
Motor KV value: 500KV
Maximum power of set propeller: 600W
Propeller set sustained power: 300W
Total thrust Maximum thrust: 3.4Kg(total thrust of two thrusters)
Ship speed: 2-4km/h
Removable weight: 0-250kg
Use environment: fresh water, sea water

These thrusters are compatible with the Shark ESCs in both voltage and current rating. The ESCs can safely drive the motors.

Battery: Gens ace 8000mAh 3S

- **Voltage:** 11.1V nominal (3S)

Description

Gens ace G-Tech 8000mAh 11.1V 100C 3S1P Lipo Battery Pack with EC5-Bashing Series

Specs:

- Product Type: Hard Case Lipo Battery Pack-Bashing Series
- Capacity: 8000mAh
- Voltage: 11.1V
- Net Weight: 510g
- Dimensions: 158*47*34mm
- Discharge Plug: EC5

The battery provides enough power for both ESCs and the Pixhawk (via the power module). It fits within the voltage limits of all devices.

Telemetry: 915 MHz or 2.4 GHz Radio

- **Interface:** UART (TX/RX)
- **Voltage:** 5V (powered via TELEM1/2 port or USB)

This module connects to the TELEM1 or TELEM2 port of the Pixhawk. It must be configured to the correct baud rate and use a legal frequency for the operating region (e.g., 915 MHz is not allowed in the EU).

This type of compatibility check is an essential step before wiring or flashing firmware.

Summary

- ArduPilot-based USVs require carefully selected hardware that fits mission needs and technical specs
- Differential thrust is used for its simplicity and precision, with no need for a rudder
- Compatibility should be confirmed using datasheets, port availability, and protocol support
- Drawing a system map before assembly helps catch issues early

Chapter 3: Basic Setup & USV Activation

This chapter explains how to bring a USV to life after selecting and wiring all the components. The goal is to install the ArduPilot firmware, configure essential parameters, and calibrate sensors and controls so that the boat can be safely tested and manually driven. This chapter assumes all hardware is physically connected and focuses on software configuration in Mission Planner.

It is divided into two parts: configuration before installing the components into the boat, and configuration after the full setup is wired and connected.

Part A – Pre-installation Setup (Flight Controller Only)

Before mounting anything inside the boat, we perform the initial setup with just the Pixhawk connected. This allows us to work comfortably, access USB, and complete important steps without physical constraints.

1. Installing the Firmware

Use **Mission Planner** to install the correct firmware:

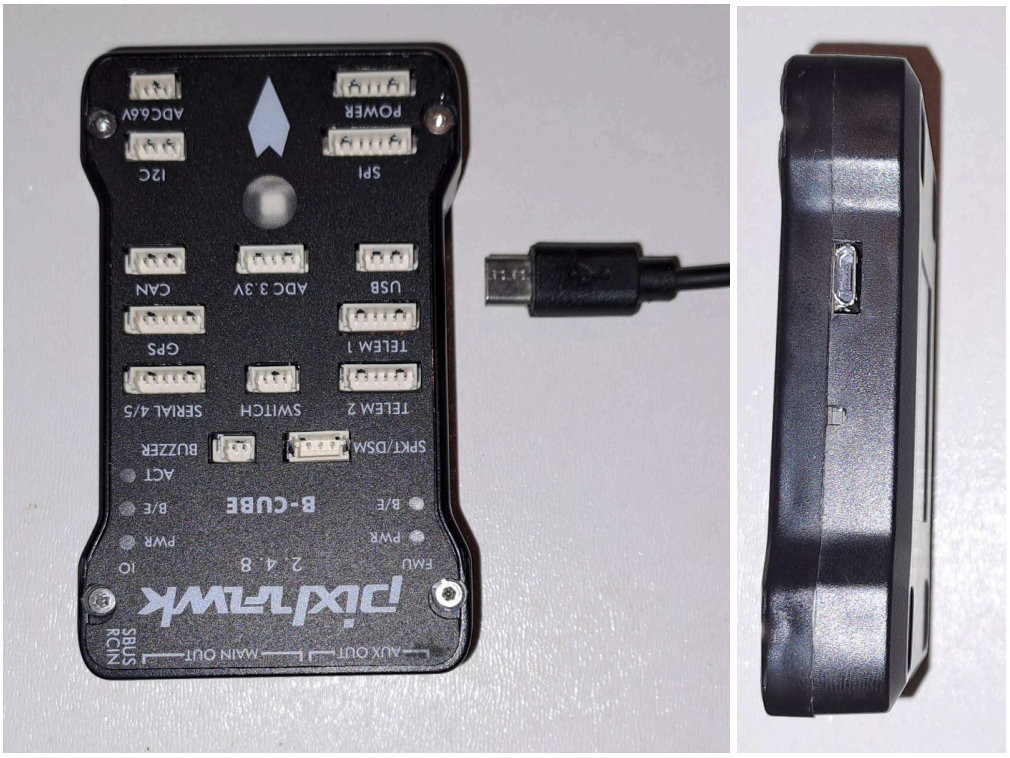
1. Download and Install **Mission Planner**

– Windows: Installing Mission Planner -- Windows

– Linux: Installing Mission Planner -- Linux

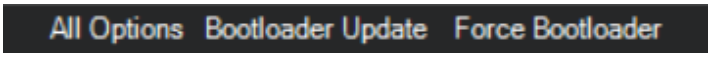
2. Open **Mission Planner**

3. Connect the USV to your PC via **USB cable**



4. Open **Setup > Install Firmware**

5. Go to **All**



6. **Choose Platform**

   Start by identifying the autopilot board you are using. For most **Pixhawk 2.4.8** boards select: **fmuv3**.

7. **Choose Version**

   Pick the latest available version. For example: `4.6.0`

Using the latest stable ensures that your system is up to date with bug fixes and performance improvements.

8. **Choose Version Type**

   Select a firmware version type that is reliable. **Recommended:** `Stable`

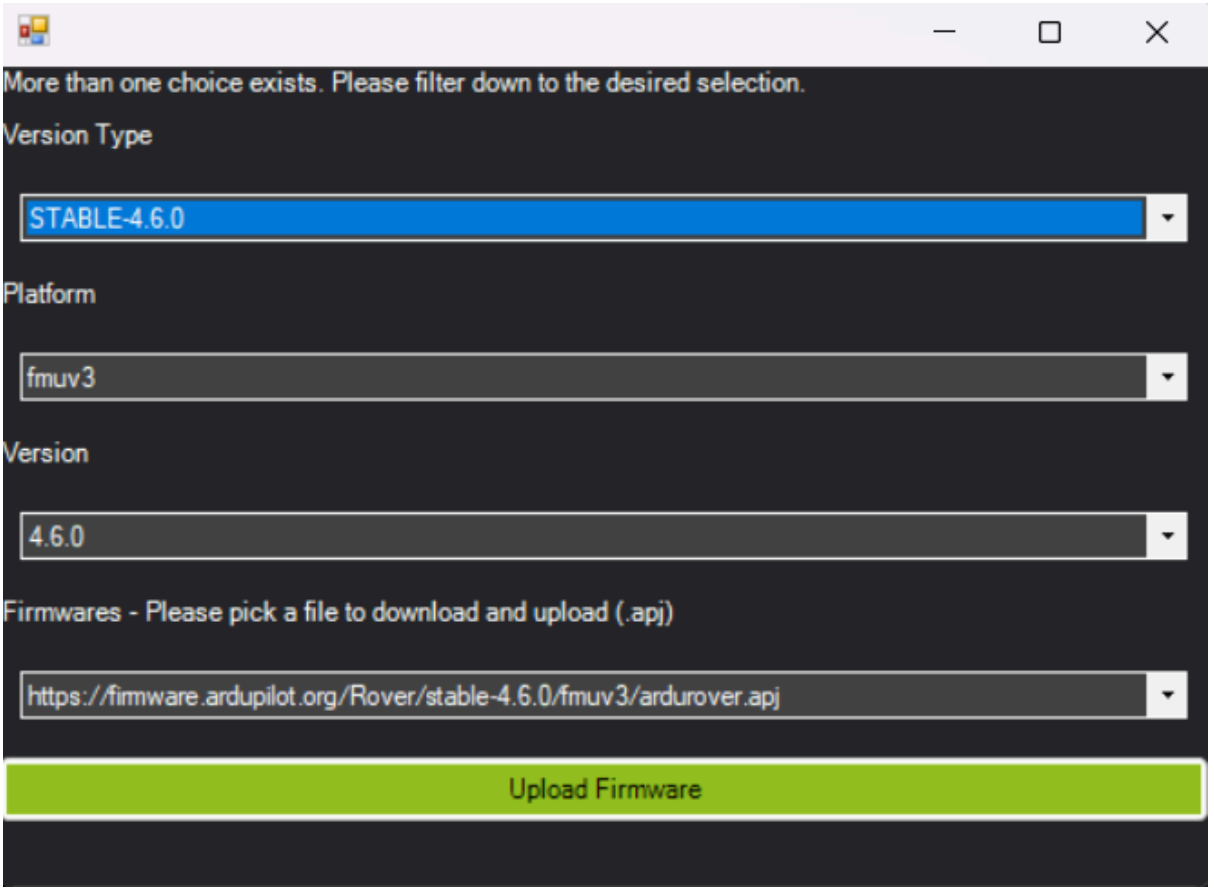
   Stable versions are well-tested and suitable for educational and field use. **Avoid BETA or DEV versions**.

9. **Confirm Firmware File**

   Check the automatically generated URL for the `.apj` file. For example:

   `https://firmware.ardupilot.org/Rover/stable-4.6.0/fmuv3/ardurover.apj`

   This confirms you are installing **ArduRover** firmware for a compatible Pixhawk board.

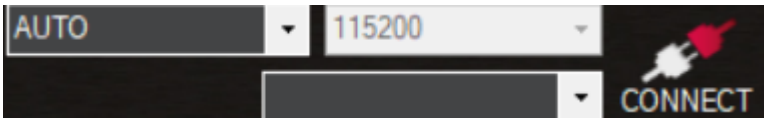


10. **Upload Firmware**

    Click **Upload Firmware** to begin the installation. Wait until the process completes.

    A progress bar will appear — **do not disconnect** the device or close Mission Planner during this process.

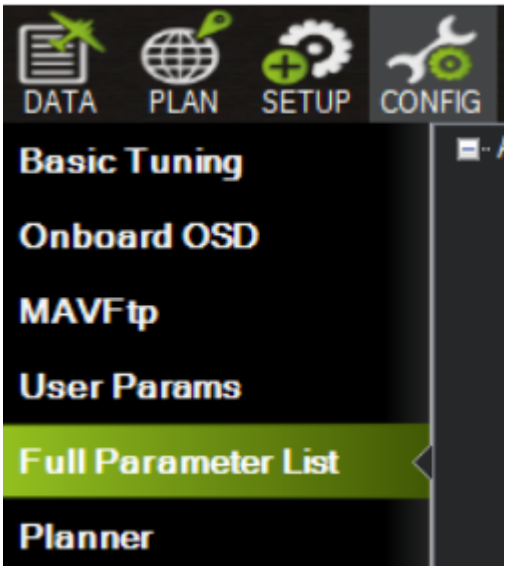
11. Choose **AUTO** and press **Connect**



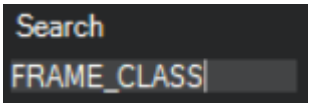
12. You can verify a successful upload when Mission Planner reconnects and identifies the board as running **Rover firmware**.

2. Set Frame Class and Skid Steering

1. Go to **Config > Full Parameter List**

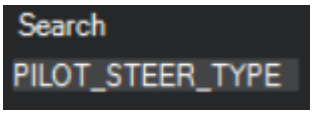


2. Set:
    - `FRAME_CLASS = 2` (this defines the vehicle as a **boat**)



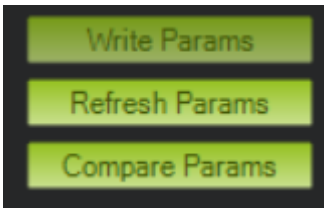
Name	Value	Default	Units
FRAME_CLASS	2	2	

    - `PILOT_STEER_TYPE = 3` (activates **skid steering** for 2 thrusters)



Name	Value	Default	Units
PILOT_STEER_TYPE	3	0	

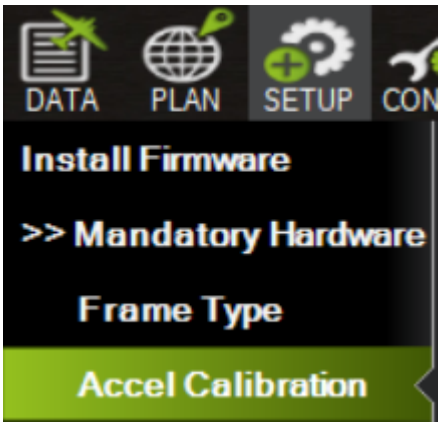
3. Click **Write Params**



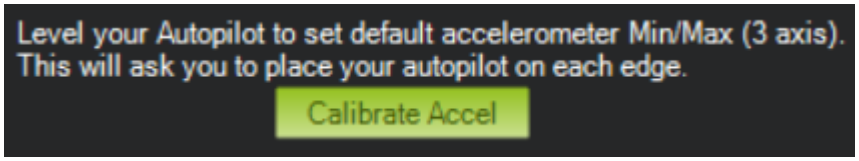
4. **Refresh Params**

3. Accelerometer Calibration

1. Go to **Mandatory Hardware > Accel Calibration**

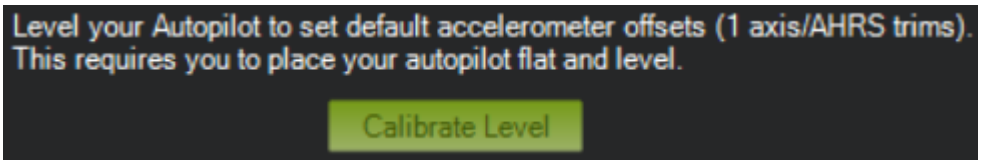


2. Click **Calibrate Accel** and follow the six-position prompts:
   - **Level** (top up)
   - **Left side**
   - **Right side**
   - **Nose down**
   - **Tail down**
   - **Upside down**



   Keep the Pixhawk still during each step.

3. Afterwards, click **Calibrate Level** to save the upright "zero" orientation.



Part B – Full Setup with Components Connected

Once the basic setup is done, connect all components: **GPS, compass, RC receiver, power module, ESCs, and motors**.

4. Set Orientation (Offsets)

If the Pixhawk is not mounted perfectly flat and straight inside the boat:

- Go to `AHRS_ORIENTATION` in **Config > Full Parameter List**
- Set its actual mounting direction (e.g., **Yaw 180** if rotated backward)

Important: If you decide to change the `AHRS_ORIENTATION` to a value other than the default (which assumes the arrow points forward and top faces up), then you **must re-do the accelerometer calibration afterwards**. Calibration results are only valid for the orientation that was active during the process. For this reason, it is recommended to always install the autopilot **near the center of gravity (CG)** and as level and aligned as possible with the boat's heading, so you can safely keep the default orientation and avoid unnecessary recalibrations.

5. IMU Position Offsets

Depending on your flight controller model (e.g., Pixhawk 2.4.8), it may contain **two or three IMUs**. These IMUs are all placed internally in the same board housing.

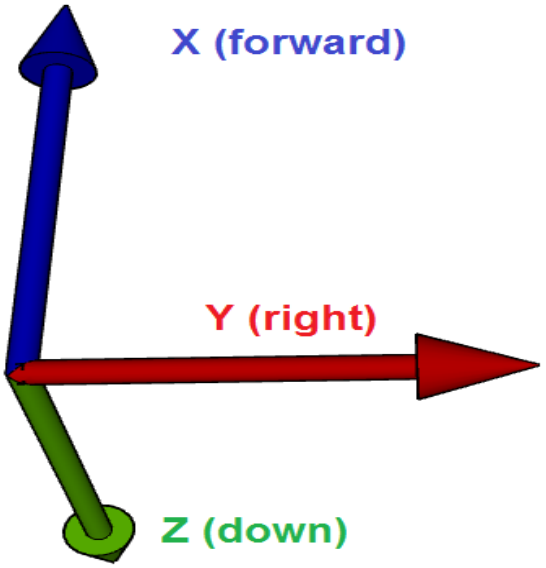
To define their physical offset from the boat's center of gravity, set the following parameters:

- `INS_POS1_X`, `INS_POS1_Y`, `INS_POS1_Z`
- `INS_POS2_X`, `INS_POS2_Y`, `INS_POS2_Z`
- `INS_POS3_X`, `INS_POS3_Y`, `INS_POS3_Z`

Use the same values for all of them.

> Example: If the Pixhawk is mounted **10 cm behind** the CG:
>
> - `INS_POS1_X = -0.1`
> - `INS_POS2_X = -0.1`
> - `INS_POS3_X = -0.1`
>
> The offset values are in **meters** and use the following convention:
>
> - **X**: forward (+), backward (−)
> - **Y**: right (+), left (−)
> - **Z**: up (+), down (−)

In most builds where the Pixhawk is close to the CG, these values can safely be left at `0.`



6. GPS and Compass Setup

Installing the GPS and Compass:

1. Connect the GPS's 6-pin connector to the slot labelled "GPS"
2. Connect the GPS's 2-pin connector to the slot labelled "I2C"

Setup of the GPS and Compass

To activate the GPS and compass correctly, we first need to ensure the proper serial ports are enabled and configured in the autopilot.

Enable GPS and Compass Ports

Go to **Config > Full Parameter List** and set:

- `SERIAL3_PROTOCOL = 5` → This enables GPS communication on UART3.
- `SERIAL3_BAUD = 230` → Set to 230400 baud if using a high-speed GPS like the NEO-M8N.
- `GPS_TYPE = 1` → Activates the primary GPS module.
- `COMPASS_ENABLE = 1` → Enables compass support.
- `COMPASS_EXTERNAL = 1` → Tells the system to use the external compass
- `GPS_POS1_X = 0`, `GPS_POS1_Y = 0`, `GPS_POS1_Z = 0` → These define the position of the GPS relative to the boat's center of gravity. Default values assume it is located exactly at the CG. Adjust if needed (units in meters).

Once these are set and saved, reboot the autopilot.

Check GPS and Compass Detection

After rebooting:

- Look at the **bottom status bar** of Mission Planner.
- Confirm that **Sats** (satellites) is **6 or higher** and **HDOP** (horizontal dilution of precision) is **below 2.0**.
  This means the GPS is active and has a good signal.

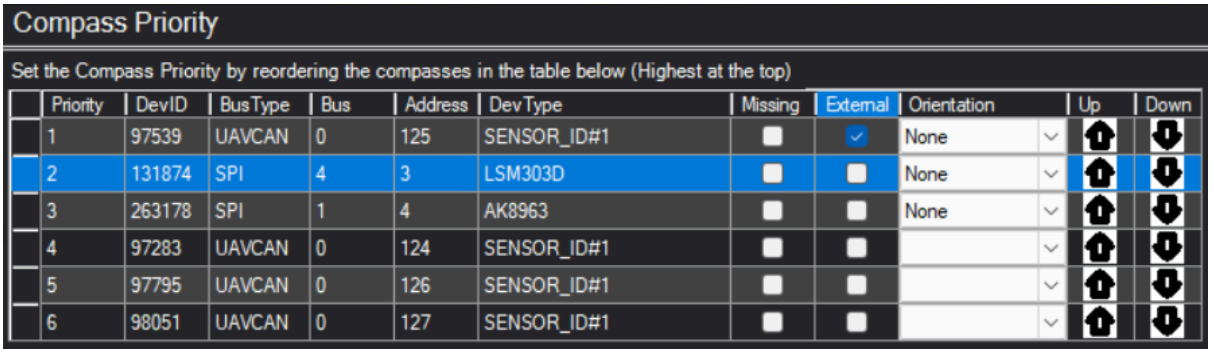
Note: You will see the number above only if you are outside (in an open area). If you are indoors, it's a good practice to stay near windows.

If detection is successful, continue with compass configuration.

Set Compass Priority

Go to **Setup > Mandatory Hardware > Compass**:

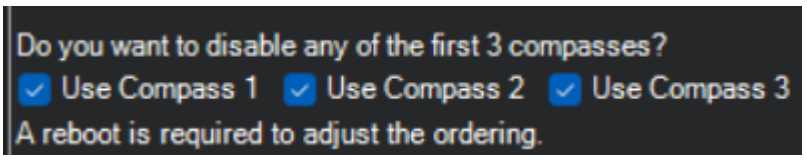
- You will see a list of detected compasses with their IDs.



- Identify the **external compass** by its **device ID** (usually not ending in "SPI"). This is the compass integrated into the GPS module. Set it as **Compass 2** and give it the **highest priority** (move to the top using the arrows).

- The **internal compass** is typically listed as **Compass 1** and shows "SPI" in the description. This is located on the flight controller board and should be set to the **lowest priority**.

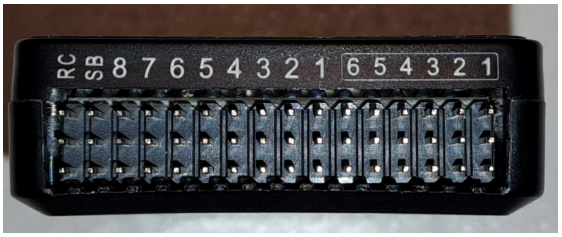
- Use the **Check boxes** for `Use Compass` to enable or disable each one.



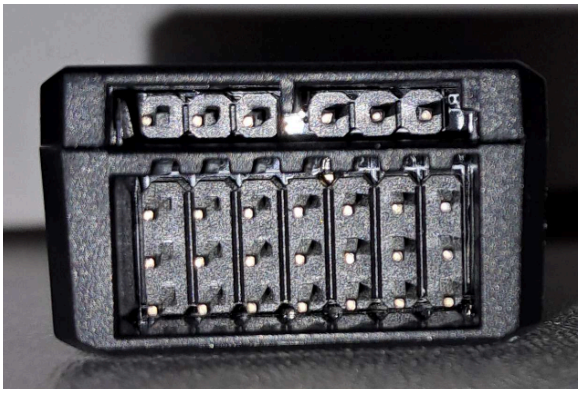
7. Radio Setup (Transmitter & Receiver)

Installation of radio receiver:

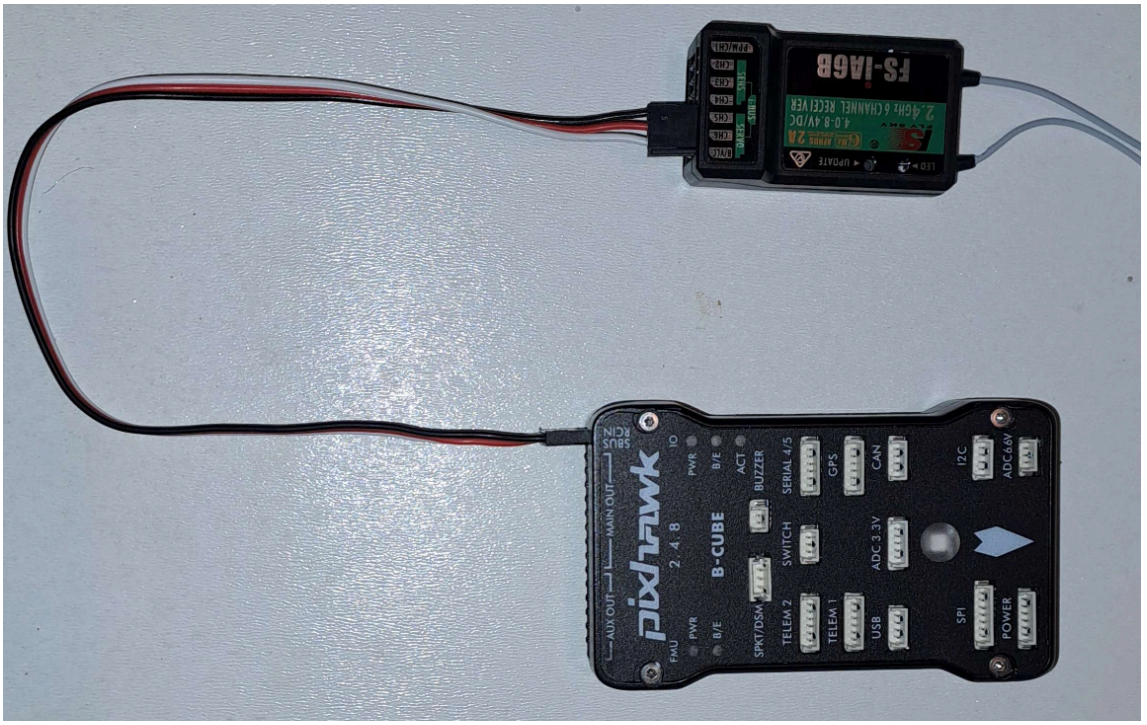
a. Connect the triple wire (black-red-white) to the Pixhawk on the RC pins, with the orientation shown below (black wire is on the top pin row):



b. Connect the other end of the triple wire to the RX receiver on the pins labelled: CH5, CH6, B/VCC, with the orientation shown below (black wire is on CH5 pin):



c. The end result should look like this:



Bind Transmitter and Receiver

Before proceeding with any calibration, you must first pair (bind) the FlySky transmitter (e.g., FS-i6X) with the FS-iA6B receiver:

1. Connect the B/VCC S and G pins with the provided wire.
2. Power up the receiver.
3. On the transmitter, enter **Setup > Rx Setup > Bind**, and confirm.
4. The LED on the receiver should turn solid — binding is complete.

Watch this video for the full bind process:
[How to bind FlySky FS-i6X to FS-iA6B Receiver]

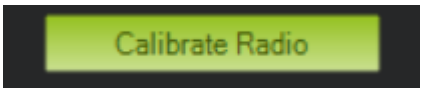
Radio Calibration

If your transmitter requires internal stick calibration (e.g., FS-i6X), perform that first from the transmitter's own menu to ensure correct channel limits.

Once the transmitter and receiver are paired and internal calibration (if needed) is complete:

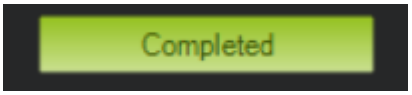
1. Open **Setup > Mandatory Hardware > Radio Calibration**

2. Click **Calibrate Radio**



3. Move all sticks and switches on the transmitter through their full range
4. Mission Planner will display the current PWM values for each channel in real-time
5. Confirm that:
    - Throttle, roll, pitch, and yaw respond correctly
    - Switches turn on and off in a clear and stable way
6. Click **Click when done** to complete the process

7. Wait for the confirmation message: **Complete**

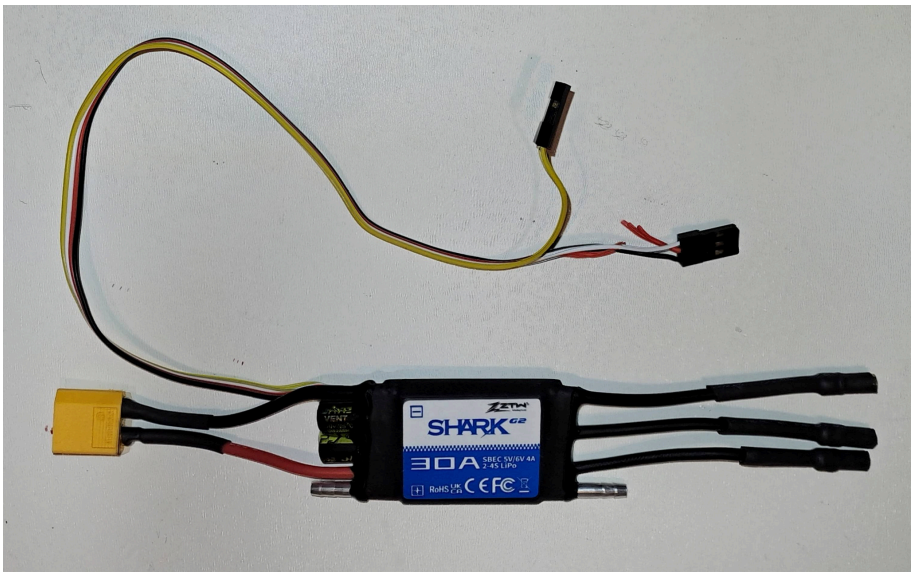


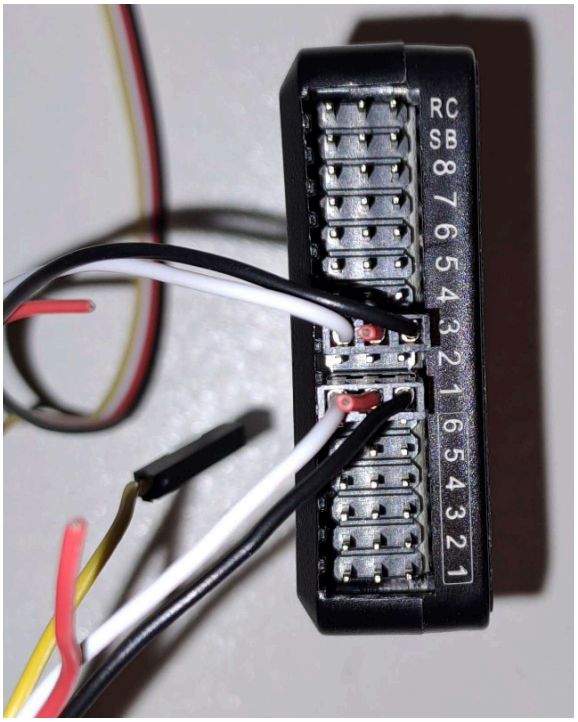
Tip: If any channels appear reversed or stuck, adjust them using the transmitter's **reverse settings**.

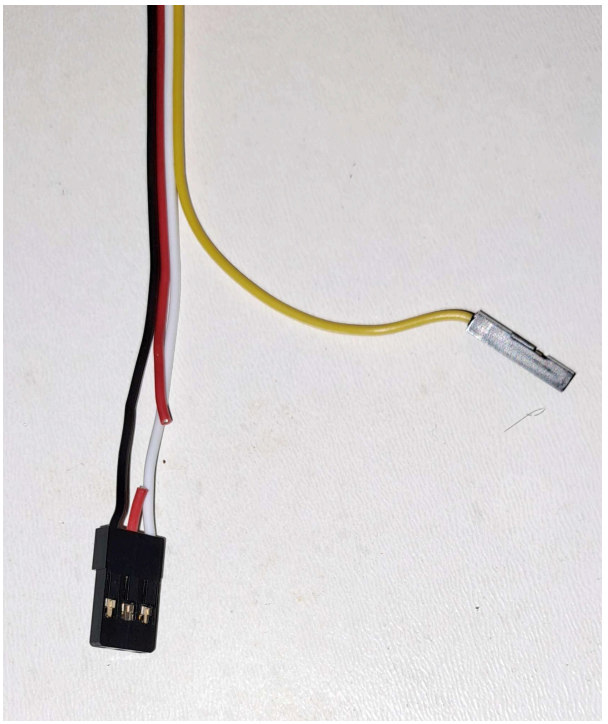
8. Servo Output Setup

Installation of ESCs:

1. Connect the ESCs 3-pin connectors to MAIN OUT slots 1 and 3.







NOTE: For our system layout, the red wire **MUST** be disabled.

To control the thrusters of your USV, you must correctly assign the output functions of the autopilot. The **Servo Output** menu in Mission Planner allows you to:

- Assign a function to each output pin (e.g., motor control, rudder, gimbal)
- Set reversal (normal/reversed direction)
- Define output limits (min, max, trim)
- Test individual outputs (e.g., spin motors or move servos)

You can assign output functions in two ways:

1. **Using the Servo Output tab:** Go to **Setup > Mandatory Hardware > Servo Output**, click the arrow next to each output, and select the desired function from the dropdown list.

2. **Using Full Parameter List:** Go to **Config > Full Parameter List** and manually set the values.

For a basic skid steering configuration (2 thrusters: left and right), you will need to define which output pins control which motor. The Summer School's USVs follows a standard configuration:

Assign Output Functions

Set the following parameters:

- `SERVO1_FUNCTION = 73` (ThrottleLeft)
- `SERVO3_FUNCTION = 74` (ThrottleRight)

These values tell the system to send left and right motor signals to outputs 1 and 3.. This mapping assumes that the thrusters are connected to **MAIN OUT 1** and **MAIN OUT 3** of the Pixhawk 2.4.8.

Check Output Direction

Once functions are assigned, you may need to reverse one or both outputs depending on motor direction. This can also be done in two ways:

1. From the Servo Output tab, toggle the **Reversed** checkbox next to the channel.
2. Or set the parameter manually:
    - `SERVO1_REVERSED = 1` (to reverse ThrottleLeft)
    - `SERVO3_REVERSED = 1` (to reverse ThrottleRight)

Set the value to `0` if the motor already spins in the correct direction.

Reversing can also be tested later during **Motor Test**.

9. ESC Setup

The Shark G2 30A ESCs used in this platform work over standard PWM. No changes to the protocol are required — they are plug & play.

These ESCs can go forward only, or both forward and backward. By default, they work in both directions, which is good for boats that turn using two motors.

To calibrate the ESCs, follow the audio-guided method shown in this video:
[Shark G2 ESC Calibration Tutorial]

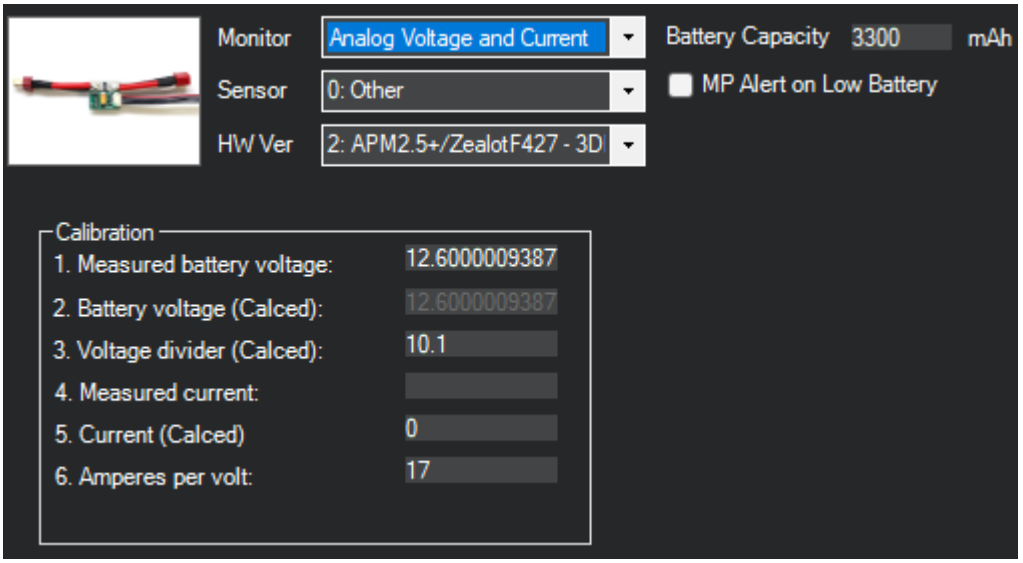
Note: The calibration is optional if the ESCs are already functioning correctly. However, it's a good idea to recalibrate them after initial setup or if throttle behavior is abnormal.

10. Battery Monitor Setup & Calibration

Battery monitoring allows ArduPilot to read voltage and current from your battery, estimate remaining power, and trigger safety actions if power levels drop too low.

Where to Set It

1. Open **Mission Planner**
2. Go to: **Setup > Optional Hardware > Battery Monitor**



Basic Configuration (for Pixhawk 2.4.8 + 3S Battery)

Set the following fields:

- **Monitor**: `Analog Voltage and Current`
- **Sensor**: `Other`
- **HW Ver**: `APM 2.5+/ZealotF427 - 3D`

- **Battery Capacity**: `8000` (for Gens ace 3S 8000mAh)

These values match the power modules and batteries used on the USVs of the Summer School setup.

Calibration Methods

There are two supported ways to calibrate voltage and current readings:

Method 1: Manual Voltage Measurement

1. Disconnect the battery and measure its voltage with a multimeter.
2. Enter that value into the field: **1. Measured battery voltage**
3. Then, reconnect the battery.
4. Mission Planner will automatically calculate the **Voltage divider (Calced)** value.

Method 2: Manufacturer Values

- If the power module or distribution board includes calibration specs:
    - Enter the **Voltage divider (Calced)** value manually
    - Also enter **Amperes per volt**

Use whichever method is available and gives reliable results.

Where to View Battery Status

After configuration, battery status will appear in the **Mission Planner main screen**, near the virtual horizon (gyroscope view). It will display real-time voltage and current readings.

If your USV uses a second power system (e.g., for sensors or payloads), you can configure a second monitor:

- This is done via parameters (`BATT2_MONITOR`) or “Battery Monitor 2” tab
- Not required unless using split power systems

Why It Matters

Accurate battery monitoring helps ArduPilot:

- Estimate remaining mission time
- Trigger battery failsafes

Failsafe configuration will be covered in the next chapter.

11. Motor Test

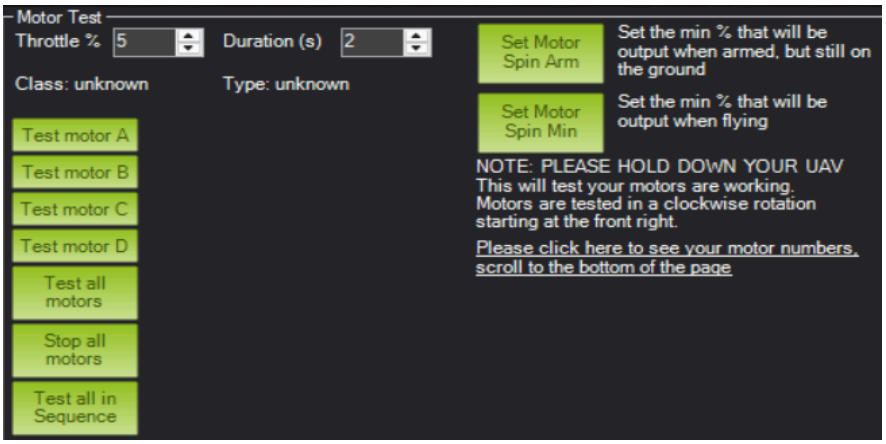
The motor test allows you to verify that the thrusters are connected correctly and spin in the correct direction. It is an important safety check before operating the USV in water. This test should **always be done out of the water** (this is depending on the specific thrusters used, always check the thruster datasheet/manual on safe operations), while the boat is safe and not touching the ground.

Before You Start

- Make sure the thrusters are free.
- Disconnect the propellers if the motors have visible blades.
- Ensure the vehicle is stable on a test stand.

How to Run the Motor Test

1. Open **Mission Planner** on your PC.
2. Connect the USV via USB or telemetry.
3. Go to: **Setup > Optional Hardware > Motor Test**



You will see a screen with throttle test options for outputs A, B, C, D.

Step-by-Step Testing

1. **Set Throttle (%)**: Choose a safe starting value, such as `15%`.
2. **Set Duration (s)**: Example: `10` seconds.
3. **Connect the main battery** to power the ESCs and thrusters.
4. **Test Motors One by One**:
    - Click the button labeled **Test motor A**, **B**, etc.
    - The corresponding thruster should spin.

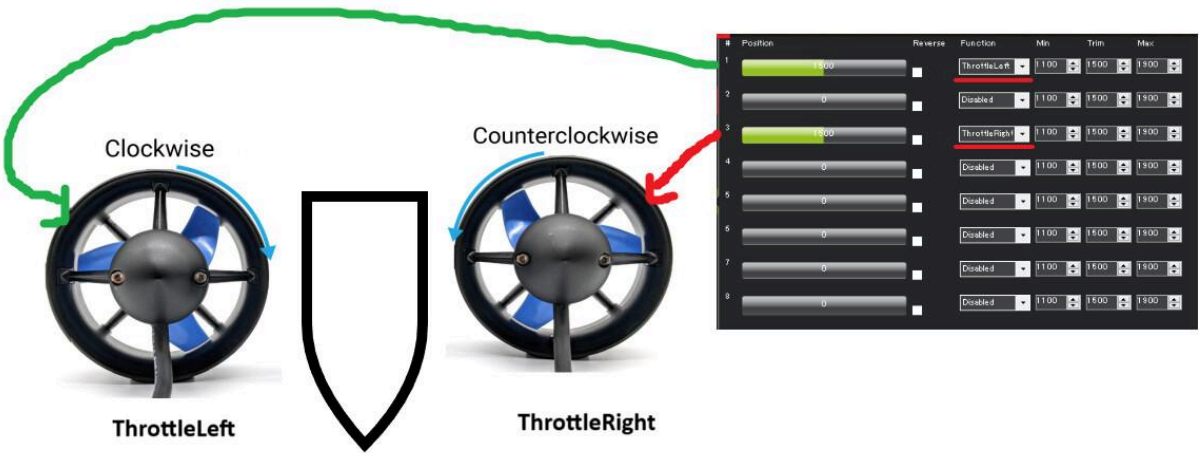
You can also use:

- **All**: Spins all motors at the same time.
- **All in sequence**: Spins motors one after another.
- **Stop**: Stops any spin right away.

Check Motor Direction

Make sure each motor spins in the correct direction:

- Left motor (ThrottleLeft) should spin **clockwise**.
- Right motor (ThrottleRight) should spin **counterclockwise**.



If a motor spins in the wrong direction, swap **two of three of the motor wires** coming from the ESC or check **"Reverse"** (Servo Output tab).

Note: The test is used only to verify connections and direction — it does **not** test full control response.

Performing this test ensures your USV moves and turns as expected when you switch to Manual or Auto modes.

12. Compass Calibration

Compass calibration is the process of teaching ArduPilot how to understand the magnetic environment of your USV. This helps the system find the correct heading while navigating.

When to Perform This Step

- After mounting all components in their final position
- Before field testing
- After moving the external compass to a new place.

Calibration Options

ArduPilot provides two main methods for compass calibration:

1. **Standard Calibration** (Recommended for small USVs)
2. **Large Vehicle Calibration** (Useful when the USV is too big to move around)

1. Standard Compass Calibration

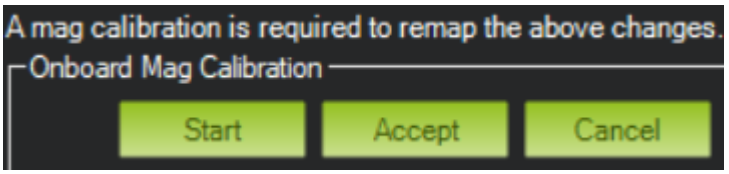
This method requires moving the USV in various orientations while ArduPilot records magnetic field readings.

Before You Start

- It's best to have two people: one holding the USV and one managing the calibration on the computer.
- To get the best results, do the calibration far from big metal things, buildings, or anything that has magnets (like watches or tools).

Steps

1. Open **Mission Planner**
2. Connect to the USV via USB or telemetry
3. Go to: **Setup > Mandatory Hardware > Compass**
4. Click **Start**



5. Slowly rotate the USV in all directions to expose it to magnetic fields from all angles
6. Keep following the steps until you finish all directions.
7. Wait for calibration to finish

8. Click **Accept**
9. **Reboot** the flight controller

After Calibration

- ArduPilot updates the COMPASS_OFSx values (offsets)
- Check for heading accuracy in the *Flight Data* view
- If heading seems unstable or wrong, repeat the calibration

2. Large Vehicle Calibration

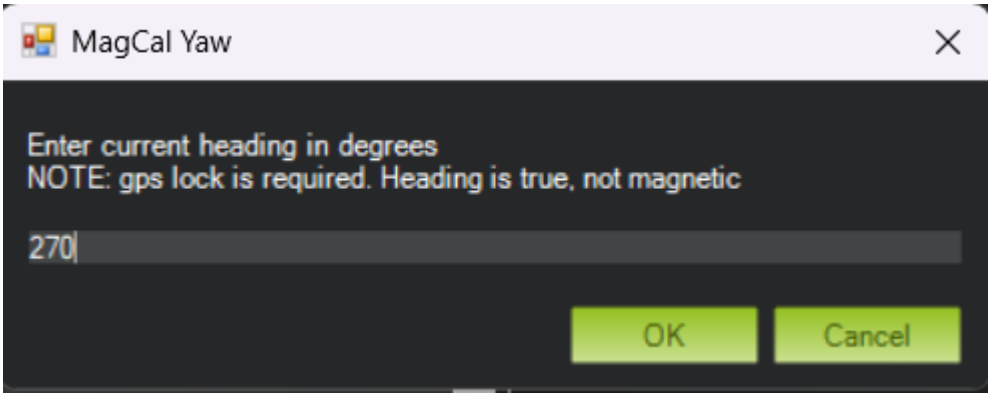
This method lets you manually measure the orientation of the compass and input values directly.

Useful when the USV is too large or heavy to move

Steps

1. Use a phone app to measure heading (degrees)
2. Go to: **Setup > Mandatory Hardware > Compass**
3. Click **Large Vehicle MagCal**

4. Write the number you find
5. Click **OK**



6. **Reboot** the flight controller

Verifying Results

After any calibration, test the compass:

- Place the USV facing north and verify heading in Mission Planner
- Rotate slowly and confirm the heading changes correctly
- If you have a small compass or a mobile phone, use it to check the direction

Example Video

You can watch this video to see how the Compass Calibration works:
Compass Calibration

The video shows a plane, but the process is exactly the same for USVs.
It helps you understand what to expect during the test.

13. Compass-Motor Calibration

Compass-Motor calibration helps ArduPilot detect and reduce magnetic interference caused by motor currents. This improves the compass accuracy during missions, especially when using high throttle.

When to Perform This Step

- Only after you have finished all other setup steps.
- Perform this as one of the last procedures before field testing.
- The external compass must already be calibrated.
- A current sensor must be installed and functional.

How It Works

When the motors spin, they create electromagnetic noise. This noise can affect compass readings and lead to heading errors. ArduPilot can measure this noise and apply corrections by calculating new offsets.

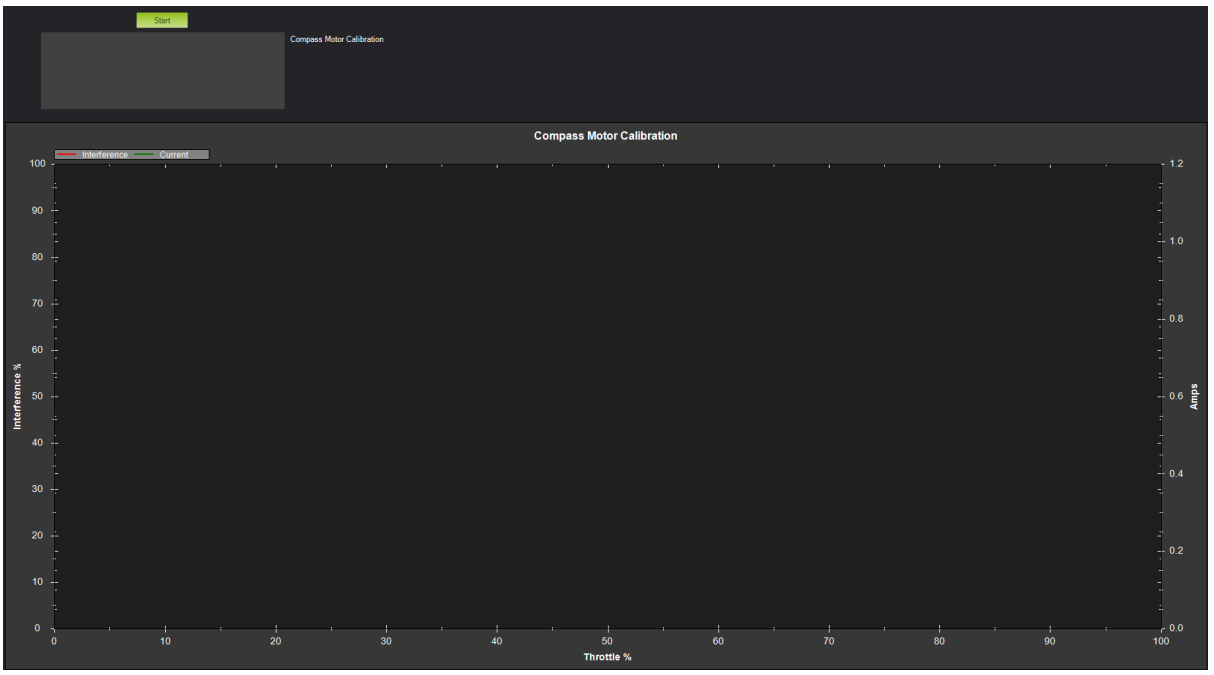
Before You Start

- Place the USV on a stable stand, away from the ground.
- Make sure the propellers are free and nothing is close to them.
- Perform the test out of the water.
- It’s best to have two people: one to hold the RC controller and one to manage the calibration on the computer.

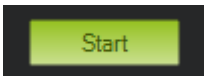
- **For best results, do the calibration far from big metal things, buildings with metal inside, or anything that makes magnetic noise (like tools).** Even if the USV is not moving, these things can change the compass readings and cause wrong values.

Performing the Calibration

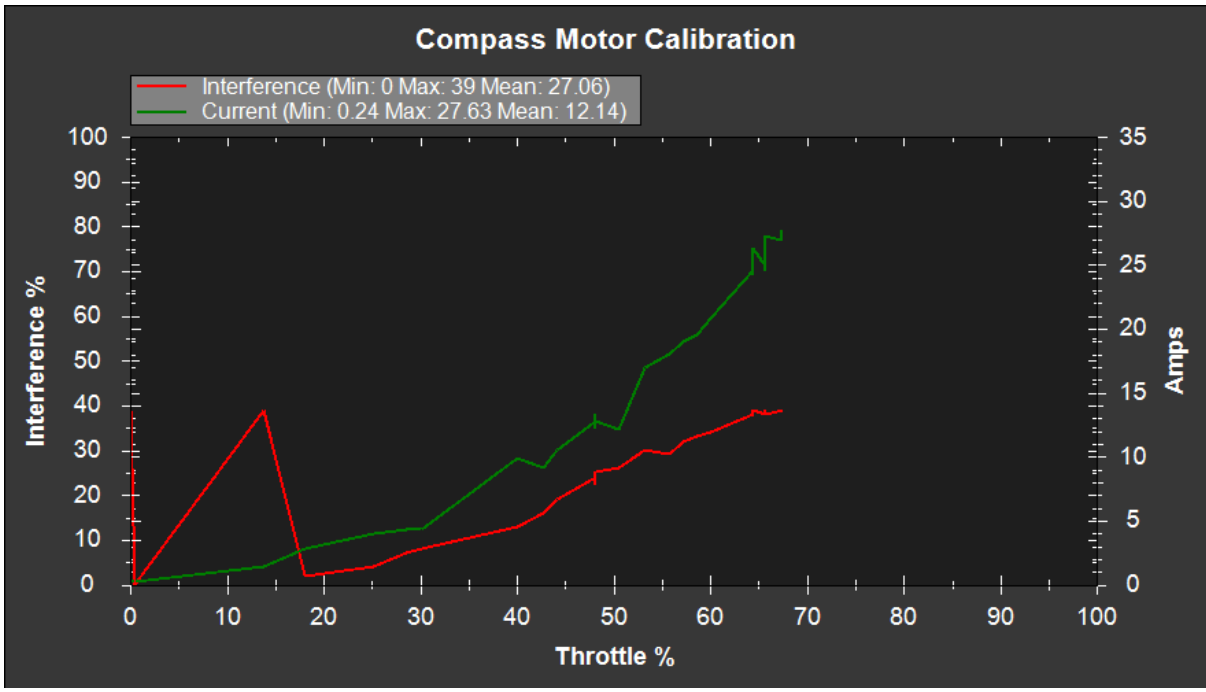
1. Open **Mission Planner**
2. Connect to the USV via USB or telemetry
3. Go to: **Setup > Optional Hardware > Compass/Motor Calibration**



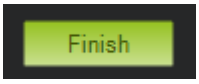
4. Hold the RC throttle stick all the way down
5. Click **Start**



6. Slowly increase the throttle
    - Watch the **% Throttle** graph
    - When it reaches 70–75%, hold the throttle steady for 5 seconds

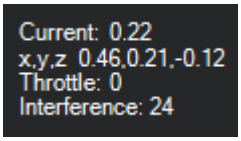


7. Lower the throttle quickly to zero
8. Click **Finish**



Reading the Results

After the calibration ends, you will see a value that shows the total magnetic interference:



- **Less than 30%**: Good — no action needed.
- **30–60%**: Borderline — consider moving the compass.
- **More than 60%**: Too much interference — compass readings may be unreliable.

What ArduPilot Adjusts

ArduPilot calculates new values for the `COMPASS_MOTx_X`, `COMPASS_MOTx_Y`, and `COMPASS_MOTx_Z` parameters. These offsets help reduce heading errors during throttle changes.

Note on Compass Placement

Magnetic interference depends on where the compass is placed. If interference is too high, try moving the compass away from strong current sources like power cables or ESCs. A common solution is to install the compass on a raised mount.

Example Video

You can watch this video to see how the Compass-Motor Calibration works:
Compass-Motor Calibration

The video shows a multicopter, but the process is exactly the same for USVs.
It helps you understand what to expect during the test.

Summary

After completing this chapter:

- Firmware is **installed**
- All **sensors and radio inputs** are calibrated
- **Servo outputs** are assigned and tested
- **ESCs and motors** behave correctly
- The USV is ready to be tested in **Manual mode**