**QUAYEWORKS**

**QW HAWK-H7 Drone Flight Controller**  
*User Manual (Beginner-Friendly Guide)*

**Introduction and Overview**

The **QW HAWK-H7** is a high-performance **drone flight controller** designed for hobbyists, students, and educators. It serves as the “brain” of a multicopter, managing the motors and sensors to keep your drone stable and responsive. Built around a powerful STM32 **H7-series MCU**, the HAWK-H7 can handle advanced flight algorithms and even basic onboard AI/vision processing. In fact, it’s described as a flight controller *and* an AI vision board suitable for drones from **quadcopters to octocopters**​

This means whether you’re building a small 4-motor quad or an 8-motor heavy lift drone, the HAWK-H7 has you covered. The goal of this manual is to help you get started with the HAWK-H7, from understanding its features to wiring it up, configuring the software, and troubleshooting common issues – all in a friendly, beginner-oriented way. Let’s get your drone project off the ground!

**Main Features and Technical Specifications**

**Key Features:** The QW HAWK-H7 comes packed with features to support robust flight control and easy integration of peripherals:

* **High-Performance MCU:** 32-bit STM32H743 microcontroller (Cortex-M7 at 480MHz) with 2MB Flash and 1MB RAM​

– plenty of processing power for complex tasks and future upgrades.

* **Onboard Sensors:** Integrated **6-axis IMU** (InvenSense MPU6050 accelerometer/gyroscope) for measuring drone orientation and movement​

a **barometric altimeter** (Bosch BMP388) for precise altitude readings​

and a **3-axis magnetometer** (QMC5883L compass) for heading information​

These sensors give the controller full awareness of orientation, acceleration, and altitude.

* **Wireless & Telemetry:** Built-in **Bluetooth module (HC-05)** for wireless telemetry and configuration – you can connect a phone or laptop to adjust settings or view data in real-time. (Bluetooth is tied to a UART port on the board for easy access.)
* **On-Screen Display (OSD) chip:** An AT7456E OSD is onboard, which allows you to overlay flight data onto an analog video feed (FPV camera input/output). This is great for First-Person View pilots – you can see telemetry (like altitude, speed, battery voltage) on your goggles or monitor while flying.
* **SD Card Logging:** A microSD card slot is provided for data logging (“black box” recording of flight data, or saving sensor readings). This helps with tuning and post-flight analysis.
* **Abundant I/O and Expansion:** Supports up to **8 motor outputs** (for quad, hexa, octocopter configurations) and additional **4 servo outputs** for accessories (gimbal control, control surfaces, etc.). Multiple **UART ports** for devices like GPS, telemetry radios, RC receivers, or additional sensors (e.g., UART6 is designated for GPS by default​​

An **I²C bus** is available for external sensors or compass, and an **SPI port** is used internally for the OSD chip​

There are also dedicated trigger pins for up to 3 **ultrasonic rangefinders** or similar sensors (for obstacle sensing or terrain following)​

* **Power Supply and Regulators:** Wide input voltage range, supporting roughly **7V–42V** (2~10s LiPo batteries)​

thanks to onboard regulators. The board provides a stable **5V output (BEC)** to power peripherals like RC receiver, GPS, or small servos, and even a **9V output** for devices such as a camera or video transmitter (ensuring video equipment can be powered from the flight controller)​

. Multiple regulator circuits (Buck converters) are present to share the load for different subsystems, ensuring stable power.

* **Compact and Lightweight:** Board dimensions are approximately 50mm x 50mm (fits easily on most drone frames) with mounting holes at standard spacings (e.g., 30.5mm square pattern with grommets). Weight is around 15–20 grams (just the board). *(These numbers may vary slightly; always double-check latest specs.)*
* **Integrated USB Connectivity:** The HAWK-H7 includes a USB-to-UART bridge (CH340C chip) for easy connection to a PC​

Simply plug in via USB to exchange data or flash firmware – no external adapter needed. Additionally, a secondary **STM32F103** chip on board acts as a debugger interface (similar to an ST-Link)​

which means you can program and debug the H7 MCU directly over USB through the ST-Link protocol. This makes firmware uploading and troubleshooting much easier for beginners.

* **LED Indicators:** Status LEDs for power and activity provide visual feedback (e.g., a steady power LED when the board is powered, and a blinking status LED during boot or when in bootloader mode). These help in diagnosing at a glance if the board is powered or if the firmware is running.

**Technical Specifications (Summary):**

* **Microcontroller Unit (MCU):** STM32H743IIT6 (Arm Cortex-M7 @ 480MHz, 2MB Flash, 1MB RAM)​
* **Co-Processor/Debugger:** STM32F103CBT6 (handles onboard debug interface and possibly I/O safety tasks)
* **Sensors:**
  + *IMU:* MPU6050 6-DoF (3-axis gyro + 3-axis accelerometer)​
  + *Barometer:* BMP388 digital barometric pressure sensor (24-bit, ~±0.5m altitude precision)​
  + *Magnetometer:* QMC5883L 3-axis compass for heading​
* **Connectivity:**
  + *USB:* Micro USB port (via CH340C UART bridge) for PC connection (virtual COM port)
  + *Bluetooth:* HC-05 module on UART3 for wireless telemetry/configuration
  + *UART Ports:* 6 dedicated UARTs (e.g., UART6 for GPS​

file-hijba5izgzc74pacixsnkc

, others free for telemetry radio, RC receiver, etc.)

* + *I²C Bus:* 1 I2C bus (used by internal sensors; can attach external I2C devices)
  + *SPI:* SPI4 used internally for OSD chip​

file-hijba5izgzc74pacixsnkc

(not typically for external use)

* + *CAN:* **(Not available on this board)** – No CAN bus interface on HAWK-H7 (or not enabled in current design).
* **Outputs:**
  + *Motor Outputs:* 8 PWM outputs for ESCs (supports up to 8 brushless motors; compatible with standard PWM or OneShot protocols).
  + *Servo Outputs:* 4 PWM outputs for servos (could be used for controlling gimbal or control surfaces if using in a plane/VTOL).
  + *Other I/O:* 3 dedicated PWM trigger outputs for ultrasonic sensors or general use; some spare GPIO pins may be available for custom use (advanced users).
* **Memory:** MicroSD card slot for log storage (supports high-capacity microSD up to 32GB or more for extensive logging).
* **Power Input:** 2S–10S LiPo battery (≈7.4V to 42V) input range​

Onboard switching regulators provide 5V (2A) and 9V (1.5A) outputs for powering peripherals​

* **Dimensions:** ~50 x 50 mm board size; mounting holes at 30.5 x 30.5 mm (M3 or M4 with grommets).
* **Weight:** ~20 g (board only, without any cables or accessories).

These specifications mean the QW HAWK-H7 is flexible and capable – it can run complex flight software, support a variety of sensors and add-ons, and fit into many drone builds. In the next sections, we’ll see how everything is laid out on the board and how to set it up step by step.

**Visual Component Map**

To help you get familiar with the hardware, the HAWK-H7 board has various components and connectors on its **top** and **bottom** sides. Below is a simplified map of the board labeling the key components and ports (imagine a diagram or photo of the board with labels for each item):

* **Top Side Components:** MCU (center of board), IMU chip, barometer, magnetometer, microSD card slot, USB port, status LEDs, Bluetooth module, and connectors for GPS, receiver, etc.
* **Bottom Side Components:** Additional connectors (for motors/servos, power input, camera in/out), the OSD chip, voltage regulators, and any debug pins (SWD interface pins).

*Insert a labeled diagram here showing major components on the top and bottom side of the board, including the MCU, IMU, barometer, magnetometer, GPS UART port, motor output pins/connectors, camera input/output connectors (for the OSD), microSD slot, USB port, Bluetooth module, LED indicators, and any other notable components.*

*(Use the above diagram to locate the ports and components referenced throughout this manual. For example, find the “GPS” label on the diagram to see where to plug in your GPS module, etc.)*

**Getting Started**

Let’s power up your HAWK-H7 and perform the initial setup. This section will guide you through the basic steps from unboxing the board to getting it ready for software configuration.

**1. Safety Check and Setup Area:** Before handling the board, ensure you’re working on a clean, static-free surface. Avoid carpeting or other static-prone areas when handling electronics. Have a USB cable ready (Micro USB for the HAWK-H7) and, if possible, a LiPo battery (within the supported voltage range) for later testing of full system power. Keep propellers **off** the motors during initial setup for safety (you won’t need props until flight testing).

**2. Inspect the Board:** Take a moment to examine the HAWK-H7. Identify the connectors and labels (using the component map above as a reference). Notably, find the USB port, the main power/battery connector, and the pin headers or plugs for motors (often labeled M1, M2, etc.), servos, and the **GPS port**. The board might have small labels or silkscreen text next to connectors (e.g., “GPS”, “M1–M8”, “UART3/BT”, etc.). Familiarize yourself with these labels now – it will make the next steps easier.

**3. Powering the Board (USB and Battery):** For the initial setup, it’s usually best to start by powering the board via **USB**. Connect the HAWK-H7 to your computer using a USB cable. The board’s power LED should light up, indicating it’s receiving 5V from USB. (Note: USB power is enough to run the flight controller and its sensors, but it **will not** power any motors or servos – that requires a battery. So, while connected to USB, the board can initialize and communicate with your PC, but of course the motors won’t spin – that’s normal.)

* *Drivers:* On first connection, your PC may install drivers for the CH340 USB-UART chip. If it doesn’t automatically recognize the board, you might need to install the CH340 driver (commonly available from vendor websites). Once installed, the board will show up as a COM port on Windows or a tty device on Mac/Linux.
* *Battery Power:* You do not need to connect a LiPo battery yet for initial configuration. However, locate the main battery input. This might be a two-wire connector or solder pads labeled VIN/BAT. Ensure you **DO NOT** connect a battery backwards – double-check polarity (red wire to +, black wire to –) when the time comes. For now, leave the battery disconnected.

**4. Initial Boot and Indicators:** With USB power, the HAWK-H7 should boot up its firmware (if pre-loaded). You may notice a status LED blinking – typically, a slow blink might indicate the system is in standby or waiting for input. If there’s a problem (for example, if the firmware isn’t loaded yet), the status LED might not blink at all or might blink in an error pattern. For now, assume the board boots normally. If you have a terminal program or the QuayeWorks configuration software, you can already try to connect (we’ll cover software setup next).

**5. Optional – Mounting the Board:** If you’re integrating the HAWK-H7 into a drone frame from the start, you can mount it now or after configuring. Use the provided anti-vibration grommets or foam tape to mount the board securely at the drone’s center of gravity (usually near the middle of the frame). Ensure the board’s orientation is correct – typically, an arrow on the board or a marking indicates which end should face forward. (If in doubt, forward direction is usually the edge where the GPS port or similar connector is, but confirm in the diagram or documentation.) You can always recalibrate or set orientation in software if needed.

**6. Ready for Software Setup:** At this point, your board is powered via USB, recognized by the computer, and the basic hardware is set up. Next, we’ll walk through installing the necessary software tools and getting firmware onto your HAWK-H7, as well as connecting peripherals like motors, radio receiver, GPS, etc.

*(Troubleshooting tip: If the board doesn’t light up when connected to USB, try another cable or USB port. Some USB cables are charge-only and have no data lines – you need a full USB data cable. Also, check if any jumper on the board needs to be set for USB powering. The HAWK-H7 typically should power via USB without any jumpers.)*

**Software Setup (STM32CubeIDE & Firmware)**

The HAWK-H7 can run custom flight control firmware, which you can compile and upload using **STMicroelectronics’ STM32CubeIDE** – a free integrated development environment. In this section, we’ll set up the software toolchain, compile the firmware (or use a precompiled firmware), and flash it to the board. Don’t worry if you’re new to this; we’ll go step by step.

**1. Install STM32CubeIDE:** Download and install **STM32CubeIDE** from the STMicroelectronics website (available for Windows, Mac, and Linux)​

This is an all-in-one development tool that includes an editor, compiler, and debugger for STM32 microcontrollers. During installation, accept any prompts to install drivers (ST-Link drivers, etc.). After installation, launch STM32CubeIDE.

**2. Get the HAWK-H7 Firmware Project:** QuayeWorks provides the firmware source code and project files for the HAWK-H7 on their GitHub repository​

Visit the QuayeWorks GitHub page (see **Resources** section at the end of this manual for the link) and download the HAWK-H7 firmware project. This might be provided as a .zip archive or by cloning a Git repository. Once downloaded, you should have a project folder that can be opened in STM32CubeIDE.

* *If you are not comfortable compiling from source:* QuayeWorks may also provide pre-compiled firmware binaries (.hex or .bin files). In that case, you can skip the compilation step and directly flash the binary using STM32CubeProgrammer or the IDE. However, going through the compile process is educational and ensures you can modify the code later if needed.

**3. Open the Project in STM32CubeIDE:** In CubeIDE, go to **File → Open Projects from File System**, and navigate to the project folder you downloaded. Import the project. Once imported, you should see the project (with sources, configuration files, etc.) in the Project Explorer in CubeIDE. This project is pre-configured for the HAWK-H7’s STM32H743 MCU, including all the sensor drivers and peripheral setups needed.

**4. Review and Configure (if needed):** Before building, check a few things:

* In the project’s **README** or documentation (if provided), see if any user-specific configuration is needed (such as enabling/disabling certain features). For example, you might need to specify if you want to use the Bluetooth module or adjust some sensor settings.
* Open the **CubeMX .ioc file** (if provided) – this graphical configuration will show the pin assignments (you can see that UART6 is used for GPS, etc., matching what we listed earlier). You don’t necessarily need to change anything, but it’s a nice visual confirmation of how the hardware is set up.

**5. Build (Compile) the Firmware:** Click the **Build** button (hammer icon) or **Project → Build Project**. The IDE will compile the code. This may take some time the first run. Watch the Console for any errors. If the build finishes successfully with no errors (and no red text), you will have a firmware binary ready to flash. The output is usually in the project’s **/Debug** or **/Release** folder (e.g., a .elf or .bin file).

**6. Prepare the Board for Flashing:** Ensure your HAWK-H7 is connected to the PC via USB. Because the board has an integrated ST-Link debugger (through the F103 chip), STM32CubeIDE can likely detect it and flash directly. Make sure the board’s drivers are installed – it should appear as an STLink device or similar. If the board isn’t detected as a debugger, you may need to put it in bootloader mode for DFU flashing: typically, holding a **BOOT0** button or jumper while pressing reset will put the H7 into USB DFU mode. However, with the on-board ST-Link, this manual step might not be necessary.

**7. Flash the Firmware:** In CubeIDE, click the **Run/Debug** button. The first time, it will ask to configure a debug configuration. Use **ST-Link** as the debug method. Then start the process – the IDE will erase the chip and upload the new firmware. You should see progress logs indicating a successful flash. (If using DFU mode, alternatively you would use **STM32CubeProgrammer** tool: select the USB DFU connection and flash the .bin file to 0x08000000 address. But this is only if ST-Link method isn’t available.)

* After flashing, the board will reset and start running the new firmware. Watch the LED behavior – often flight controller firmware will have a distinct LED pattern (for example, blinking in a heartbeat pattern).

**8. Verify Communication:** Once firmware is loaded, verify that you can communicate with the board. Depending on the firmware, you might use a Ground Control Station (GCS) software (if it’s ArduPilot or PX4 based firmware) or a simple serial terminal. For instance, if it’s running a custom QuayeWorks firmware, they might provide a GUI or you could open a serial terminal (115200 baud on the COM port of the CH340) and see debug messages. This step confirms the board is alive with your new code.

**9. Sensor Calibration and Basic Settings:** After flashing, you may need to calibrate sensors and configure the flight control parameters:

* *Accelerometer/Gyro calibration:* Most flight software will require the IMU to be calibrated (placing the board level and running a routine to zero out biases). This might be initiated via software (some firmware start with calibration mode on first boot). Follow the firmware’s instructions – for example, in ArduPilot you would connect with Mission Planner and do “Calibrate Accel/Compass”. In a custom setup, there may be a command or code routine.
* *Magnetometer calibration:* If you plan to use the compass (QMC5883L), you’ll usually perform a compass calibration (moving the board or drone in various orientations). This yields more accurate heading information.
* *RC calibration:* If using a radio transmitter/receiver, calibrate the sticks (ensure the controller knows the min/max PWM or SBUS values for throttle, etc.). More on connecting the receiver in the wiring section below.

**10. Saving and Testing:** Save any configuration changes. At this stage, your HAWK-H7 should have the firmware running and be basically configured. We haven’t attached any peripherals yet (motors, GPS, etc.) – that’s next. But from a software perspective, you’ve set up the development environment and gotten the firmware onto the board successfully – congratulations!

*(If you encounter issues during this process, check the Troubleshooting section. For example, if the IDE can’t find the board to flash, you may need to install the ST-Link driver or use the DFU method. If the code doesn’t compile, ensure you have the correct STM32 toolchain installed. The QuayeWorks GitHub README may list specific versions of CubeIDE or additional libraries needed.)*

**Wiring and Peripheral Connections**

With the board powered and firmware ready, it’s time to connect all the external components that make your drone work: motors (via ESCs), servos (if any), GPS module, radio receiver, telemetry/Bluetooth, camera, etc. The HAWK-H7’s design simplifies this with clearly labeled ports. We’ll go through each major connection type. Refer to the **Visual Component Map** (and any labels on the board) to locate the right connectors. Always **disconnect power** before making or changing connections to avoid shorts or damage.

* **Motors and ESCs:** The HAWK-H7 supports up to 8 motor outputs. These are typically arranged either as through-hole pins or an 8-pin connector. They may be labeled **M1** through **M8** on the board. Connect each ESC’s signal wire to the corresponding motor output pin. For example, on a quadcopter, you’ll use M1, M2, M3, M4 for the four ESCs; on a hexacopter, use M1–M6; on an octocopter, M1–M8. The signal ground or common ground of the ESCs should be tied to the board’s ground. If you are using a 4-in-1 ESC (common in smaller drones), the board likely has a single **8-pin JST connector** that can plug directly to the 4-in-1 ESC’s harness (carrying 4 motor signals plus power/ground and possibly a telemetry wire). In summary, plug in all motor controllers to their respective outputs. Do **not** connect the ESC’s BEC 5V output to the flight controller (if using opto or no-BEC ESCs, this isn’t an issue; if ESC has a 5V BEC and you’re using it to power servos or the FC, consult the power section). Usually the FC is powered separately and ESC BECs are not needed except to power servos on some setups.
* **Servos (for Gimbal or Control Surfaces):** In addition to motors, the board provides 4 servo output pins (labeled for example as **Servo1–4** or by function). In the technical pinout, they were named “Arm, Roll, Pitch, Yaw” servos​

file-hijba5izgzc74pacixsnkc

– but essentially they are just four PWM outputs similar to motor outputs. You might use these for a camera gimbal (to stabilize a camera in pitch/yaw), or in a fixed-wing aircraft for controlling ailerons, etc., or even for a parachute or payload release mechanism. To connect a servo, plug its control wire into the servo pin, and its power leads to 5V and GND. **Important:** The HAWK-H7’s 5V regulator can supply some current to servos, but high-power or many servos might require an external BEC. If your servo draws a lot of current (e.g., over 2A), consider powering it from a separate BEC (and still connect the grounds together). For small servos (like micro servos on a gimbal), the onboard 5V should be fine.

* **GPS Module:** The GPS module (and compass, if the GPS has one) typically connects via a UART port. The HAWK-H7 reserves **UART6** for GPS​

file-hijba5izgzc74pacixsnkc

. Look for a 6-pin JST-GH connector labeled “GPS”. Connect your GPS module’s cable here. The typical wiring is: 5V (to power the GPS), GND, TX, RX (for GPS data), and possibly SDA/SCL if the GPS unit has an integrated compass (many GPS modules have an I2C compass, which might use those extra pins to connect to the board’s I2C). On the HAWK-H7, the onboard compass QMC5883L might be used instead of an external one, but if your GPS has a better compass (e.g., HMC5883L), you could disable the internal and use the external via I2C. In summary: plug the GPS unit into the GPS port. Make sure the **TX from GPS goes to RX on the board and vice versa** (the cable should handle this by how it’s pinned, so just plug in). After powering, your GPS should get power (some have an LED) – it may take a few minutes to lock satellites, which is normal.

* **RC Receiver (Radio Control):** To manually control your drone, you’ll connect an RC receiver. The HAWK-H7 supports popular receiver protocols. If your receiver outputs **PPM** or a single **SBUS** signal (as many modern receivers do), you can use one of the UART inputs or a dedicated RC input pin. Check the board diagram: some flight controllers label an “RC” or “SBUS” pin. If not, you can use (for example) UART8 RX for SBUS input (since UART8 was tied to the USB serial, you might repurpose if not using USB at the same time). Another approach is connecting a PPM receiver to a timer input pin – but to keep it simple, let’s assume SBUS (which is common for hobby radios like FrSky). Connect the receiver’s SBUS output to the **RX pin of a spare UART** (e.g., UART1 RX) and in your firmware configure that UART for SBUS input. Also connect the receiver’s 5V (or 4.5V) and GND to the flight controller’s 5V/GND. If using IBUS (FlySky) or other serial protocols, the connection is the same (just a UART RX). For traditional PWM receivers (individual channel outputs), you would need a PPM encoder or use multiple inputs – since the HAWK-H7 doesn’t have 8 separate channel inputs, using SBUS/PPM is strongly recommended.
  + *Binding and setup:* Make sure your receiver is bound to your transmitter. Many modern receivers have a bind button – bind it according to its manual before connecting to the flight controller. Once connected, you’ll later verify in software that stick inputs are being received (e.g., you should see channels move in the configurator or through serial data when you move your transmitter sticks).
* **Telemetry & Bluetooth:** The board has a built-in **Bluetooth HC-05** module on UART3​

file-hijba5izgzc74pacixsnkc

. This means you can use a phone app or PC Bluetooth to connect to the flight controller for telemetry data or configuration (within Bluetooth range ~10m). By default, the HC-05 on HAWK-H7 likely is set up to output basic telemetry. To use it, pair your device with “HC-05” (PIN is usually 1234 or 0000 by default), then you can connect via a terminal or a GCS app that supports Bluetooth (some apps like MultiWii or Betaflight Configurator might read telemetry via BT). If not using the built-in BT, you can instead connect an external **telemetry radio** (like an HC-12, XBee, or 3DR telemetry module) to a free UART. For example, if UART2 or UART8 is free, connect the radio’s TX->RX, RX->TX, 5V, GND accordingly. This would give you long-range telemetry. *Note:* If you use the built-in Bluetooth, it might already occupy one UART, limiting how many free ports remain. Check firmware settings to ensure you know which UART is allocated where.

* **Camera and Video Transmitter:** One of the cool features of the HAWK-H7 is the OSD chip for video overlay. To use this:
  + **Analog FPV Camera:** Connect your analog camera’s video output to the flight controller’s **camera input** pad or connector. Also connect the camera’s power (some flight controllers provide a 5V or 9V output specifically for camera/VTX – the HAWK-H7 has a 9V 1.5A regulator ideal for video gear​

So you might have a connector labeled “Video In” or “Cam”. That likely has pins for Video signal, 9V out, and GND. Plug the camera there (ensuring correct polarity).

* + **Video Transmitter (VTX):** Similarly, connect the VTX’s video input to the flight controller’s **video output** pad/connector (often labeled “Vout” or “Vid Out”), and power the VTX from the provided 9V or 5V as appropriate (many video transmitters run on 7-24V, so 9V is perfect). Ensure common ground with the board.
  + With this setup, the analog video signal flows from the camera, into the FC’s OSD chip (which overlays text data), then out to the VTX, which broadcasts it to your goggles. The firmware (e.g., Betaflight OSD or ArduPilot MAVLink OSD) will automatically send data to the OSD chip to display. You can typically configure what to display via your GCS or RC transmitter (things like altitude, speed, battery, etc.).
  + If you are using a **digital FPV system** (like DJI/Caddx Vista or Walksnail Avatar HD), the analog OSD won’t directly work. However, many digital systems support an emulated OSD via MSP protocol on a UART. For instance, if you had a DJI FPV unit, you’d connect it to a UART and turn on MSP for that port in firmware, so it can receive OSD data digitally. This is more advanced; if you go digital, consult the digital system’s integration guide. For now, with analog, just plug camera and VTX as described.
* **SD Card:** Simply insert a **microSD card** into the slot (usually on the top or bottom side). Make sure it clicks in fully. Use a quality card (Class 10 or UHS-I) for reliable logging. The firmware will log data such as sensor readings, GPS tracks, etc., to the card. After flights, you can remove the card and analyze logs on a computer (this is extremely useful for troubleshooting issues like vibration or tuning problems). Ensure the card is formatted as FAT32 for best compatibility.
* **Additional Sensors (optional):** If you plan to expand with extra sensors, the HAWK-H7 offers ways to do so:
  + *I2C expansion:* You can connect I2C sensors (like a lidar for distance measurement, or an external magnetometer if needed) to the I2C bus. The board’s I2C bus is shared with internal sensors, so be mindful of address conflicts. For example, many I2C lidars have configurable addresses – set them so they don’t clash with 0x68 (MPU6050) or others. Use the SDA/SCL pads or port (if available).
  + *UART sensors:* Some sensors or devices use serial (GPS we did, but others could be an airspeed sensor like the digital ones, or an Arduino-based sensor). Use a free UART accordingly.
  + *Ultrasonic Rangefinders:* The board provided 3 dedicated trigger pins (and likely echo pins would go to some inputs). If you have ultrasonic modules (HC-SR04 style), typically one pin triggers, another listens. The HAWK-H7 technical info shows PF6, PF7, PF8 as triggers​

file-hijba5izgzc74pacixsnkc

. The echo might be captured via interrupts on other pins. This is an advanced use – you would need to write code to handle these or ensure your firmware supports rangefinders. If you’re a beginner, you can skip this until you’re comfortable.

* **Connecting the Battery:** Once all components are wired, the final connection is the main battery. Double-check every connection **before** plugging in the LiPo! Verify polarity on the battery leads. It’s wise to use a **smoke stopper** (a resettable fuse device) the first time you power the system with a battery – this can prevent damage in case of a wiring mistake. When ready, connect the LiPo to the board’s power input. The board should power on (LEDs on), your ESCs will chime (if they have startup tones), and connected peripherals like receiver and VTX should power up. At this point, the full system is alive. Keep fingers and tools away from the motors – if anything were mis-configured and a motor spins, you don’t want anything getting caught.

Now all hardware should be connected properly. The next steps (not in this section) would be configuring your radio in the software, testing motor directions, calibrating ESCs, etc., which are part of configuration profiles and tuning. We will cover basic configuration next.

*(Tip: Use zip ties or heat shrink to secure wiring so that nothing can snag in the spinning parts. Also ensure that the GPS module is placed away from high-interference areas – usually on a mast or at least away from the power wires – for best satellite reception and compass accuracy. For the first setup, it’s okay to have it loose, but eventually, mount it properly and perform compass calibration.)*

**Safety and Power Guidelines**

Working with drones involves high-speed propellers, powerful batteries, and sensitive electronics. This section outlines important **safety precautions** and power management tips to keep you and your equipment safe. Always take safety seriously – a beginner-friendly approach is to develop good habits early!

* **Battery Safety (LiPo care):** Use only the recommended battery types (Lithium Polymer packs, 2S–10S as per specs). **Never short-circuit** the battery wires. Always double-check polarities when connecting – a reverse connection can permanently damage the HAWK-H7 and other electronics. When charging LiPo batteries, follow proper procedures: *Always charge LiPos in a safe area (preferably on a non-flammable surface or inside a fireproof bag) and never leave them unattended while charging​*

Use a quality LiPo balance charger, set to the correct cell count and current for your battery. Overcharging or improper charging can cause fire or explosion. Likewise, do not over-discharge your batteries – most flight controllers or ESCs have a low-voltage cutoff; ensure it’s configured (e.g., ~3.5V per cell under load as a safe minimum). If a battery is damaged, puffed, or past its safe cycle life, dispose of it properly (in saltwater or per local guidelines​

Treat LiPo batteries with respect and they’ll safely power your drone adventures.

* **Propeller Safety:** *Always remove propellers or keep them off until you have fully configured and tested the motors on the bench.* Inadvertent motor spin-ups can cause serious injury. Even small drone propellers can cut skin badly. Only install props when you are ready to test fly in a proper area. When working on your drone with the battery plugged in, ensure the drone cannot arm (many flight softwares have safety switches or won’t arm via USB by default). Be mindful of where your hands are relative to motors when powered.
* **Common Ground and Wiring:** Ensure a **common ground** connection between all components – the flight controller, ESCs, BECs, servos, etc., should all share a ground reference​

file-hijba5izgzc74pacixsnkc

. This prevents unpredictable behavior (for example, a missing ground between the flight controller and an ESC can cause the ESC signal reference to float, leading to erratic motor outputs). Basically, always connect the ground wires along with the signal wires for every device. Use appropriately thick wires for battery and ESC connections to handle high current, and ensure they are soldered or crimped securely – a loose power connection can lead to voltage spikes or loss of power in flight.

* **Power Distribution and BECs:** The HAWK-H7 has onboard regulators (5V and 9V) that can power your receiver, small servos, and FPV gear. However, be mindful of their current limits (2A continuous at 5V, etc.). If you connect many devices that draw a lot of current (say LED strips, high-torque servos, etc.), you might exceed what the onboard BEC can supply, causing voltage to drop or the regulator to overheat. A sign of this is the flight controller browning out (resetting) when loads like servos move. If in doubt, use a dedicated external BEC for those high loads. For most basic setups (receiver + GPS + maybe a small gimbal servo), the onboard 5V is fine. The 9V output is typically for a video transmitter; ensure your VTX current draw is within 1.5A (most are). Avoid powering heavy components from the flight controller directly if they can be powered off a PDB (Power Distribution Board) or battery with their own regulator. Also, if using multiple BECs, **do not tie multiple 5V BEC outputs together**; use one as the source and disconnect others (to prevent them from back-feeding each other).
* **ESC and Motor Safety:** Modern ESCs usually have safety features, but it’s important to calibrate your ESC throttle range (if needed) according to the firmware instructions so that motors don’t spin unexpectedly. When you first test motors (with props off!), do so at a low throttle and ensure they respond correctly. If a motor does not spin or the ESC beeps indicating an error (like lost signal), double-check the wiring and configuration. Keep loose wires away from spinning motor bells – secure all cables.
* **Physical Handling:** The HAWK-H7 has delicate sensors (IMU, barometer). Avoid dropping the board or subjecting it to extreme shocks. The barometer in particular can be affected by wind – when you mount the board in a drone, it’s good to cover the barometer with a small piece of open-cell foam to shield it from direct propeller wind, which can otherwise cause erratic altitude readings. The IMU is vibration-sensitive: use the provided soft mount grommets for the controller to reduce high-frequency frame vibrations (this will improve flight performance by giving cleaner sensor data).
* **Environmental Considerations:** Keep the board dry – electronics and water don’t mix. If you plan to fly in damp conditions, consider a conformal coating on the board (advanced). Avoid operating in extreme temperatures outside the range (-20°C to 80°C typically) – sensor readings might drift and battery performance drops in cold weather.
* **Pre-flight Checklist:** Develop a habit of doing a brief check before every flight: Confirm battery is fully charged and securely strapped in, all connectors (to motors, etc.) are tight, perform a radio range check, ensure GPS has lock (for GPS-enabled flight modes), calibrate compass if you moved to a new location far away, and arm the drone only when clear to take off. It’s better to catch an issue on the ground than in the air.

By following these safety and power guidelines, you will minimize the risk of accidents and component damage. Drones are tons of fun, but they do carry potential dangers if not handled carefully. Always err on the side of caution, and happy (and safe) flying!

**Configuration Profiles (Examples)**

Every drone setup is a little different. In this section, we provide example configuration profiles to help you set up your HAWK-H7 for a **basic drone**. These examples illustrate typical configurations and parameter settings for common drone types. Use these as a starting point and adjust as necessary for your specific build.

**Example 1: Quadcopter (X Configuration) – Basic Setup**

Imagine you’ve built a standard quadcopter with 4 motors, running the HAWK-H7 as the flight controller. Here’s how you’d configure it:

* **Frame Type:** In your flight control software (firmware), select **Quadcopter X** as the frame type. This indicates four motors in an “X” layout (front-left, front-right, rear-left, rear-right). The firmware will assign each motor output to a position (usually Motor 1 = front-right, 2 = rear-left, 3 = front-left, 4 = rear-right for an X quad, though this can vary – refer to the firmware’s documentation for motor ordering). Ensure your wiring of motors to M1–M4 corresponds to the positions expected by the software. If not, you can either remap outputs in software or simply swap motor plugs to match.
* **Motor Orientation and Props:** Configure the motor spin direction pattern according to the quad X norms. Typically, for a quad-X: Motor 1 (front-right) spins clockwise, M2 (rear-left) clockwise, M3 (front-left) counter-clockwise, M4 (rear-right) counter-clockwise – this is an example, some use the opposite. The important part is that diagonally opposite motors spin the same direction. Set your props accordingly (cw props on cw motors, ccw on ccw motors). In the software, there may be a motor test function – use it (with props off) to verify each motor spins the correct direction and is in the correct order. If a motor is spinning the wrong way, simply swap any two of its three wires at the ESC to reverse it (or use ESC config if supported).
* **Radio Control Settings:** After wiring your SBUS/PPM receiver as described, go into the firmware’s radio/RC configuration. Select the input type (e.g., SBUS serial on UARTX). Calibrate the channels by moving your sticks through full range so the software learns the min, max, and mid values. Verify that the correct channel is assigned to the correct function (Throttle, Yaw, Pitch, Roll) and that they move in the correct direction (e.g., when you push the pitch stick forward, the pitch channel value increases and the drone would interpret that as a command to tilt forward). If any channel is reversed, you can reverse it in your transmitter or in the flight controller software. Also set up an **arm/disarm switch** on your transmitter and configure it on a channel (commonly channel 5 or a dedicated arming function). This switch will be used to arm the motors.
* **Flight Modes:** For a beginner, it’s good to start with a stabilized mode (often called “Angle” or “Horizon” mode in many firmware) where the drone self-levels. Configure a switch on your transmitter to toggle flight modes. For example, position 0: Stabilized (self-level), position 1: Altitude Hold (if you have a barometer, the HAWK-H7 does, so you can use AltHold mode to maintain altitude), position 2: perhaps GPS Loiter (if GPS is attached and configured). The exact modes depend on your firmware (Betaflight, iNav, ArduPilot, etc., each has their own set). Set the mode channel ranges accordingly so the FC knows which mode to be in.
* **PID and Tuning:** The firmware will have default PID values for a generic quadcopter. These are usually a good starting point. Given the HAWK-H7 has the MPU6050 (an older but reliable IMU) and likely good noise handling, the defaults might work fine. For a medium-size quad (450mm frame with 10-inch props, for instance), default PIDs of something like P=40, I=30, D=20 (just illustrative) might be given. If you find the drone oscillates or is sloppy, you can adjust these later. As a beginner, stick to defaults or use an auto-tune feature if available (ArduPilot has AutoTune mode, for instance, which flies the drone through twitches to tune PIDs).
* **Auxiliary Functions:** Configure any aux channels if needed: e.g., a buzzer (if attached to a pin, assign a switch to trigger “beeper”), LED strip (program colors if using addressable LEDs on an output), or a battery failsafe (set the voltage thresholds so that at, say, 14.0V on a 4S battery the system triggers a low-battery alarm or auto-landing). The HAWK-H7 monitors voltage (likely via an analog pin or INA219 sensor onboard) so ensure the voltage scaling is correct in software (usually entering the measured battery voltage to calibrate).
* **Compass & GPS:** For a basic flying in stabilized mode, you don’t need GPS. But if you want to use features like Loiter (position hold) or Return-to-Home, configure the GPS and magnetometer. The GPS should be detected on UART6 as configured; set the protocol (e.g., UBX or NMEA, most autopilot firmware auto-detects). Verify you’re getting a GPS lock (satellite count increases, home position set after a few minutes). For compass, decide whether to use internal (onboard QMC5883L) or an external (if your GPS module has one). If using the internal, do a compass calibration dance (rotate the quad around all axes as prompted by the GCS until calibration completes). If using an external, tell the software which one is primary and calibrate similarly. Set the orientation if the compass is not mounted facing forward (some GPS modules need you to specify an offset if rotated). A successful compass setup means the heading shown matches the actual drone heading (check with a compass or by pointing the drone north and seeing if the GCS shows north).
* **Arming and Test Hover:** Once everything is configured, it’s time for a test hover. Find a wide open area. Do a final pre-flight: props on tight, battery fully charged, no loose wires, transmitters on and bound, verify mode switch is in the correct start position (usually stabilized mode), and the board is armed (some systems require a stick combo or just the switch to arm – ensure it’s armed and the safety checks passed, like GPS lock if needed or calibration done). Increase throttle slowly until the drone lifts off. If it tends to drift, correct with the stick or land and adjust trims (though in modern FC, you usually don’t trim on transmitter; you adjust the accelerometer calibration or use auto-trim function). A stable hover in angle mode indicates a successful basic setup! From here, you can try other modes or do further tuning as needed.

**Example 2: Octocopter Configuration (Advanced)**

*(Just as a brief example to show the HAWK-H7’s capability – beginners can skip this until ready.)* Suppose you want to build an 8-motor X8 octocopter for heavy lifting (two motors on each arm, or 8 single motors on 8 arms). The HAWK-H7 can control 8 motors. You’d select an **Octocopter X** frame in the software. Wire motors M1–M8 to the 8 ESCs. The software will have a predefined motor order for octo (e.g., M1 front, then around clockwise M2, M3... etc.). You would ensure all motors spin correct directions (lots of pairs to check!). You might also configure a **motor redundancy** feature if the firmware supports (so it can handle one motor failure gracefully). All other setup (GPS, receiver, etc.) is similar to the quad. Octocopters draw more power, so pay extra attention to power distribution and possibly use multiple batteries or a high-C rated battery. Also, the default PID may need adjusting for the heavier platform. Typically, higher weight and more motors might require lower P gains to avoid oscillations. It’s advisable to test in stabilized mode and possibly use AutoTune if available, due to the complexity of 8 motors.

**Example 3: Fixed-Wing (Airplane) Configuration *(Optional)***

*(For completeness, if someone wants to use HAWK-H7 in a plane or rover, it’s capable, but this manual focuses on multicopters primarily.)* The HAWK-H7 can also act as an autopilot for a plane. In that case, you’d use 3 of the servo outputs for control surfaces (e.g., aileron, elevator, rudder) and one motor output for the throttle (ESC). You’d enable a fixed-wing mode in the firmware, calibrate control surface neutrals, and set up stabilization (so the IMU can level the plane). The GPS can be used for navigation (waypoints, return to launch), and the magnetometer for heading. Planes also use an airspeed sensor often, which you could connect via I2C (e.g., an MS4525DO sensor). The configuration would include tuning PID for pitch/roll hold. This is quite advanced compared to a quadcopter, so ensure you have some experience (or a buddy) before trying an autonomous plane.

These profiles hopefully give you an idea of how to set up various drones with the HAWK-H7. The **Quadcopter** example is the most common for beginners – start there. As you gain experience, you can explore more complex configurations. Remember that every time you change the hardware (different frame, prop size, etc.), you may need to revisit the configuration and tuning. The HAWK-H7’s powerful hardware and flexible firmware support means you have a lot of room to grow and experiment with different UAV projects. Enjoy the process, take it one step at a time, and soon you’ll be comfortable tweaking settings to get that perfect flight performance!

**Troubleshooting Guide**

Even with careful setup, you might encounter issues. Don’t worry – this section lists common problems and their solutions. If your HAWK-H7 isn’t behaving as expected, look for the symptoms below and try the suggested fixes. Most issues can be resolved with a bit of patience and debugging.

* **No Power / No Boot (Board seems dead):** If you plug in USB or the battery and no LED lights up at all, the board isn’t powering on. First, check the power source: Is the USB cable good and fully plugged in? If on battery, is the battery charged and connected to the correct input? Use a multimeter to see if 5V is reaching the board’s 5V rail. If not, there could be a blown fuse or regulator. Check for any visible damage on the board (burn marks or a hot component). If you suspect the onboard 5V regulator is not working (for instance, if you accidentally shorted something earlier), you can try powering the board via USB (which typically bypasses some regulators) or via an external 5V to the 5V pins to see if it lights up. Also, ensure any jumper that might enable/disable USB power is set correctly (some boards have a VIN/VUSB select jumper). If none of these work, the board might be defective – contact QuayeWorks support for guidance.
* **Fails to Connect to PC (USB Issues):** You plugged in the USB but the device doesn’t show up on your computer. If using Windows, check Device Manager – do you see an “Unknown device” or “USB-Serial CH340” or “STMicroelectronics STLink” device? If it’s unknown, install the appropriate drivers (CH340 driver for the serial port, ST-Link driver if using that interface). If nothing shows up at all, try a different USB port or cable. Also ensure the board’s USB connector isn’t damaged (wiggle gently to see if the power LED flickers – if so, the connector solder might be broken). Another thing: if the board is stuck in **bootloader mode** (BOOT0 jumper set), it might not run the USB interface firmware. Ensure BOOT0 is not unintentionally pulled up. In a pinch, you can also connect via the UART pins using an external USB-serial adapter if the on-board USB won’t work. For example, use an FTDI cable to connect to the UART8 (CH340) pins directly – this can be a workaround to re-flash firmware if USB is problematic.
* **Cannot Flash Firmware (CubeIDE errors or no ST-Link found):** If STM32CubeIDE says it cannot find the target or ST-Link, make sure the board is in the correct mode. If relying on the on-board ST-Link (F103 chip), verify that your PC recognized it (it should appear as “STLink dongle” or similar under USB devices). If not, reinstall the ST-Link USB driver from ST’s website. If CubeIDE still fails, try using the standalone **STM32CubeProgrammer**: connect via UART (the CH340 COM port) and use the **UART bootloader** method. To do this, you’ll need to put the MCU in bootloader mode: set BOOT0 to 1 (if there’s a button or jumper labeled BOOT0, hold/enable it), reset the board, then in CubeProgrammer choose “UART” and the COM port and 115200 baud, and try connecting. The STM32H7 has a built-in bootloader that listens on UART and USB DFU. If UART doesn’t work, try **USB DFU mode**: BOOT0 = 1 and connect USB, then in CubeProgrammer choose USB. The board should enumerate as “STM Device in DFU mode” if successful. Then erase or flash the firmware. Don’t forget to set BOOT0 back to 0 (off) after flashing and reset, so the MCU boots the new firmware.
* **Firmware Flashed but Board Not Responding:** Suppose the flashing went fine, but now the board seems unresponsive (no telemetry, no LED blink etc.). It could be that the firmware you loaded isn’t running or got stuck. One possibility is wrong firmware build (for example, a firmware for a different board or wrong MCU). Ensure you compiled for STM32H743 and the correct memory addresses. You might also have an issue with clock setup (HSE oscillator config) causing a hang – check the project settings if you made changes. Try flashing a known-good firmware (maybe an earlier backup or a simple test firmware that just blinks an LED) to confirm the board itself is okay. If that runs, then the issue is with the firmware configuration. Use the debugger in CubeIDE: set breakpoints or pause to see where it might be stuck. If you’re not familiar with debugging, as a quick check, add a line in code to toggle an LED in a loop early in main(), and see if it blinks – if not, the code might not even be reaching main (which could mean bad vector table or clock init). Recheck the startup files and linker script. These are advanced steps – alternatively, reach out on forums or to QuayeWorks for support with the firmware.
* **No Sensor Data / “Sensor not responding”:** If the flight controller boots but reports that, say, the **IMU or barometer is not found**, it usually means a wiring or initialization issue on the I²C bus (since MPU6050 and BMP388 in this design are on I2C​

file-hijba5izgzc74pacixsnkc

). Possible causes: the sensors might be unpowered (check that the 3.3V regulator on the board is functioning – many sensors run on 3.3V; if a 3.3V line is down, none of the I2C sensors will respond). Or the sensors could be in reset/standby – recall the “Sensor Enable Pins” listed (PA15 for MPU6050, etc.​

file-hijba5izgzc74pacixsnkc

). The firmware needs to toggle those pins to turn on the sensors. Ensure your firmware code sets those pins high (or low, depending on design) to enable power to sensors. If you missed that, the sensors stay off. Also ensure you’ve configured the correct I2C address in the driver – e.g., some chips have alternative addresses if AD0 pin is tied differently. The MPU6050 typically at 0x68, BMP388 at 0x77 or 0x76. Use an I2C scanner (some firmware have a CLI “i2c scan” or you can write a quick sketch) to see if the addresses appear. If nothing on I2C, likely power or wiring. If one sensor works and another doesn’t, possibly that sensor is damaged or its enable pin logic is wrong. Check the board schematic if available to confirm how sensors are connected.

* + *Compass not responding:* The QMC5883L compass is also I2C. If it’s not detected, similar steps as above. Additionally, compasses often need initialization sequence – make sure your firmware has driver support for QMC5883L (some older software expects HMC5883; the QMC is a different chip requiring different handling or at least a config to treat it as HMC clone). ArduPilot and iNav support QMC, just ensure it’s turned on. If using an external compass and not the internal, disable the internal one to avoid conflict.
* **GPS not detected:** If you see “No GPS” in your OSD or GCS, check the GPS wiring. Is it on the configured UART (default UART6 on this board)? Did you set the correct baud rate in software? Many GPS modules default to 9600 or 38400 or 57600 baud depending on maker. Autodetect might fail, so set it manually (e.g., 115200 is common for UBlox GPS after they’re configured). Also verify the GPS has power (led on it blinking means it’s searching for sats, which implies power is fine). If the GPS uses UART but you accidentally plugged into the wrong port, obviously it won’t be found – double-check it’s indeed on UART6 pins. Another nuance: some flight controllers require you to set the protocol (UBlox, NMEA, etc.). Ensure that matches your GPS – most modern ones speak UBlox binary if it’s a Ublox chip. You can test the GPS by unplugging it from FC and connecting to a FTDI/USB adapter direct to PC running u-center software to see if it outputs data. This can confirm the GPS is functional. If it is, then the issue is likely a config in FC software. Fix the port settings and try again.
* **No Motor Output or Motors Won’t Arm:** If you can’t get the motors to spin at all, even though the flight controller seems to be on, a few things to check:
  + **Arming sequence:** Most flight controllers require an arm command for safety. For example, in Betaflight you might need to have throttle at minimum and yaw to the right for a few seconds, or in ArduPilot you might need to toggle an arm switch or click in the GCS to arm. If you skip this, the FC will never send signals to the ESCs. So ensure you are actually arming the system. An indicator of armed state is often a change in LED (maybe from blinking to solid) or a message on OSD (“Armed”).
  + **Pre-arm checks:** If arming fails, many firmwares won’t allow arming if certain conditions aren’t met (e.g., compass calibration incomplete, or GPS required for certain mode and no lock, or RC signal not present). The controller might be refusing to arm. Connect to the GCS or serial console to see if there’s an error message (like “PreArm: Compass not calibrated” or “GPS Lock required”). If so, address that issue (calibrate compass, or disable the requirement by changing flight mode, etc.). For initial testing, you can often use a manual mode that doesn’t require GPS, to get things going.
  + **ESC Calibration:** If using ESCs that require throttle range calibration (mostly older or hobby ESCs, not so much modern ones with auto-calibration), then you need to do that: basically send full throttle while powering on ESC, then low throttle – typically done via receiver directly. But many flight controllers have a pass-through or a routine to do it. If motors twitch but don’t run uniformly, wrong calibration could be a reason.
  + **Output signal type:** Some ESCs (especially modern BLHeli\_s or BLHeli\_32) can accept various signal types: PWM, OneShot, DShot (digital). If your firmware is set to output DShot and you have old analog ESCs, they won’t understand that. Or vice versa, if set to standard PWM 50Hz and you have fancy ESCs, they’ll still work because they all accept PWM, but at low rate. Check the configuration: set the output protocol appropriately (if in doubt, start with PWM or Oneshot125 which almost all ESCs support). If using DShot, ensure the wiring is correct (some boards require removing a resistor to use pure digital signals). The HAWK-H7’s design seems geared for analog signals, so PWM/Oneshot is default.
  + **Wiring issue:** If a particular motor output isn’t working, could be a bad solder joint or pin. Test with a servo connected to that output to see if it moves when you try to arm and throttle up (with props off, a servo will show you movement if it’s getting a signal above its minimum). If one channel is dead, the board could have a hardware fault on that pin (not common, but possible). You could remap another free output to act as that motor in software as a workaround if hardware issue is suspected.
* **Unstable Flight or Drifting:** (This is post-takeoff, assuming you got flying but it’s not stable.) This can be due to *many* factors, but common ones:
  + *Compass interference:* If in GPS mode the drone toilet-bowls (circles), that’s usually compass issues – perform calibration again and keep compass away from power wires.
  + *Vibrations:* If the quad oscillates or the altitude holds poorly, excessive vibration could be feeding into the gyro or baro. Ensure the FC is soft-mounted and props are balanced. Check log vibration metrics if available. Adding a bit more foam to baro or better prop balance can help.
  + *PID tuning:* Defaults not suitable – if it oscillates rapidly (buzzing sound), P gains too high; if it feels sluggish or drifts a lot, P too low or I too low (for drift). Tuning PIDs or using auto-tune can fix this.
  + *Weight distribution:* If the drone is very tail-heavy or one motor arm heavier, it might drift or motors on one side work harder (you’ll see difference in motor outputs in logs). Try to balance the weight.
  + *Wind:* In a stable (angle) mode, the drone will drift with wind since you’re just self-leveling. That’s normal – you need to actively correct with input or use a GPS position hold mode. In GPS hold, drifting could mean poor GPS fix or compass issue. Check number of satellites and HDOP value; wait for a good lock (HDOP < 1.5 ideally) before takeoff.

If none of the above covers your problem, consider seeking help on online forums or the QuayeWorks support channels. When asking for help, provide as much detail as possible: firmware used and version, a wiring diagram or photos, and the exact symptoms. Often, community members can assist in diagnosing unusual issues. The HAWK-H7 is designed to be user-friendly, but as with any complex system, there can be hiccups. The good news is that once you iron out the initial wrinkles, the system tends to be reliable, and you’ll enjoy many flights. Keep learning and don’t get discouraged by setbacks – troubleshooting is a great way to deepen your understanding of how your drone works!

**Glossary**

**Below are definitions for various technical terms and acronyms** you’ll encounter while using the QW HAWK-H7 and drones in general. This glossary is written in beginner-friendly language to help you grasp the concepts quickly:

* **BEC (Battery Eliminator Circuit):** A voltage regulator that provides a constant voltage (like 5V) to power electronics from a higher battery voltage. In RC models, a BEC “eliminates” the need for a separate receiver battery by dropping the main battery voltage to e.g. 5V for the flight controller and servos​

(The HAWK-H7 has BECs on board to supply 5V and 9V.)

* **Bluetooth:** A short-range wireless communication protocol. The HAWK-H7 uses a Bluetooth module (HC-05) to allow wireless data transfer, for example, connecting to a phone or laptop for telemetry or setup. Bluetooth typically works up to ~10 meters range.
* **ESC (Electronic Speed Controller):** An electronic device that takes the flight controller’s throttle signals and drives the brushless motors accordingly. It regulates motor speed by rapidly switching the motor coils on and off (using PWM techniques). In a drone, each motor has its own ESC. The ESC also often has a built-in BEC to power the receiver/FC (though many drone ESCs are “OPTO” with no BEC). An ESC receives control signals (PWM or digital) and outputs 3-phase AC to spin the motor​
* **Firmware:** The low-level software running on the flight controller’s MCU. It contains the flight control algorithms, reads sensor data, and outputs commands to motors. Examples of firmware are Betaflight, ArduPilot, PX4, iNav, or a custom QuayeWorks program. Firmware is typically uploaded (flashed) into the MCU’s memory and runs every time the board is powered on.
* **FPV (First Person View):** A flying method where the pilot uses a live video feed from the drone’s camera, as if they are “on board” the drone. The HAWK-H7’s OSD feature is for analog FPV systems, overlaying telemetry onto the camera feed. FPV can be analog (with a video transmitter and goggles receiving static-filled but low-latency video) or digital HD (clear image, often lower latency systems nowadays).
* **GPS (Global Positioning System):** A satellite-based navigation system. A GPS receiver on the drone listens to signals from satellites to determine the drone’s latitude, longitude, and altitude. With a GPS module, your flight controller can know its position and enable features like return-to-home or loiter (position hold). Often GPS modules also include a magnetometer (compass) and a status LED. The HAWK-H7 supports GPS via a UART connection.
* **IMU (Inertial Measurement Unit):** A set of sensors (usually a 3-axis gyroscope and 3-axis accelerometer, and sometimes a magnetometer) that measure the drone’s motion and orientation​

The gyroscope measures rotation rates (deg/sec) around the axes (roll, pitch, yaw). The accelerometer measures linear acceleration (including gravity). The flight controller integrates this data to know the drone’s attitude (orientation) and to detect changes in motion. The HAWK-H7’s primary IMU is the MPU6050, providing 6-DoF (Degrees of Freedom) data.

* **LiPo (Lithium Polymer) Battery:** The type of rechargeable battery commonly used to power drones. LiPo batteries have high discharge rates to provide the current drones need. They are described by cell count (S) and capacity (mAh). For example, a 4S 1500mAh LiPo has 4 cells (~14.8V nominal) and can supply 1.5A for one hour theoretically (or proportionally more current for shorter time). LiPos require careful handling: don’t overcharge (>4.2V per cell) or over-discharge (<3.3V per cell load) and follow safety guidelines (see Safety section).
* **MCU (Microcontroller Unit):** Essentially the “brain” chip of the flight controller. It’s a compact integrated circuit that contains a processor, memory, and input/output peripherals​

. The MCU runs the firmware. On HAWK-H7, the main MCU is an STM32H743, featuring a powerful Cortex-M7 CPU core. It reads sensor data and calculates motor outputs hundreds of times per second.

* **OSD (On-Screen Display):** A system that overlays text/graphics onto a video feed. In drones, OSD is used to display telemetry (battery voltage, altitude, speed, etc.) onto the FPV video seen by the pilot. The HAWK-H7 has an AT7456E OSD chip (similar to the classic MAX7456) which the firmware uses to draw text on the analog video signal. This way, you can see crucial info in your goggles without looking at a ground station.
* **PWM (Pulse Width Modulation):** A method to encode a value in the width of a pulse in a digital signal. Traditional RC receivers output PWM signals for each channel (with pulse width ~1000us for low, ~2000us for high). Flight controllers output PWM to ESCs to command motor speed. A modern ESC might accept higher frequency PWM or digital protocols, but the principle is the same: the length of the high signal dictates the commanded value​

. PWM is also used for servo control (50 Hz, 1-2ms pulses). The HAWK-H7 outputs PWM signals on its motor and servo pins.

* **SBUS (Serial BUS):** A digital serial protocol for RC receivers, developed by Futaba. It carries all channel data over one line (inverted serial at 100000 baud). Many receivers (FrSky, etc.) use SBUS. The flight controller must have an inverter (STM32H7 UARTs can handle inverted signals in hardware) or use a dedicated SBUS input. SBUS typically carries up to 16 channels. It’s fast and efficient compared to PWM wires for each channel. In this manual, we connect SBUS to a UART to get RC inputs.
* **Telemetry:** In drone context, this refers to the data transmitted from the drone back to the pilot or a ground station in real time. Telemetry can include position, speed, battery status, attitude, etc. The HAWK-H7 can send telemetry via multiple means: Bluetooth link to a phone, a 915MHz radio modem to a laptop, or even the FPV OSD (which is a form of one-way telemetry on screen). Having telemetry allows you to monitor your drone’s status during flight and is essential for long range or autonomous missions.
* **UART (Universal Asynchronous Receiver/Transmitter):** A hardware serial communication interface used for asynchronous serial communication. In simpler terms, a UART is a serial port that can send/receive bytes of data without a clock signal (using agreed baud rate). The HAWK-H7 has multiple UARTs which it uses to communicate with devices like GPS (NMEA or UBlox sentences via serial), the Bluetooth module, telemetry radios, SBUS receiver (through UART in mode), etc. Each UART is identified by a number (UART1, UART3, UART6, etc.)​

file-hijba5izgzc74pacixsnkc

and typically corresponds to specific pins on the MCU. For example, UART6 TX/RX might go to the GPS connector. When configuring, you’ll enable or disable functions on certain UARTs depending on what’s connected.

This glossary isn’t exhaustive, but it covers the main terms you’ll see as you work with your flight controller and drone. If you come across any other unfamiliar term, be sure to look it up – understanding the terminology will greatly enhance your ability to troubleshoot and make the most of your HAWK-H7. Happy learning!

**Resources**

For further information, technical details, and downloads, refer to the following resources related to the QW HAWK-H7 and its components. These links will provide more in-depth knowledge and support as you work with the flight controller:

* **QuayeWorks Official GitHub Repository** – Source code, firmware releases, and hardware design files for the HAWK-H7 (and other QuayeWorks projects). You can find the latest firmware, report issues, and see example code.

(Visit the URL: *github.com/QuayeWorks* for the project list).

* **QuayeWorks HAWK-H7 Hardware Project Page** – The open-hardware design page with schematics and PCB layout (OSHWLab). This includes the board’s circuit design under an open hardware license. *(For advanced users interested in the electrical design or making modifications.)*​
* **STM32H743IIT6 Microcontroller Datasheet** – Detailed specifications of the H7-series MCU used on the HAWK-H7. This STMicroelectronics datasheet covers the MCU’s capabilities (CPU, memory, peripherals)​

Helpful if you want to understand the controller’s performance or use specific MCU features in custom code. *(STMicroelectronics website – search for STM32H743IIT6)*.

* **MPU-6050 IMU Datasheet and Info** – Information on the 6-axis motion sensor. InvenSense (TDK) provides the product specification detailing its registers and performance. This is useful for understanding IMU data outputs or debugging IMU issues​

*(Search “MPU6050 datasheet” or see TDK InvenSense site.)*

* **BMP388 Barometric Sensor Info** – Bosch Sensortec’s page on the BMP388 barometer, including datasheet and application notes​

If you want to learn how the altitude sensing works or how to interpret pressure readings, this is the place.

* **STM32CubeIDE Download and Documentation** – STMicroelectronics’ official page for STM32CubeIDE​

Contains the installer, user manual, and tutorials. Refer to this if you encounter issues with the IDE or want to learn about its advanced debugging features.

* **Flight Controller Communities and Forums:** For general help beyond the HAWK-H7, you can visit forums like **RCGroups**, **ArduPilot Discuss**, or **IntoFPV**. There, many experienced hobbyists can offer advice. While not specific to QuayeWorks, questions about STM32-based flight controllers or Betaflight/ArduPilot setup often find answers in these communities.
* **QuayeWorks Documentation Site:** *(if available)* Check if QuayeWorks maintains an official documentation or wiki for their products. This might include step-by-step guides, videos, or FAQs specific to HAWK-H7. (As of writing, the GitHub and OSHWLab are primary sources; an official docs site may be in the works.)