Title: Controlling a Pixhawk 6X Drone Using MAVLink Over Ethernet with Raspberry Pi 5

1. Overview

This guide explains how to control a drone using MAVLink over an Ethernet connection between a Raspberry Pi 5 and a Pixhawk 6X flight controller. MAVLink (Micro Air Vehicle Link) is a communication protocol used for drone communication. The goal is to set up reliable communication over Ethernet to send commands, receive telemetry, and change flight modes using a Raspberry Pi.

1. Hardware Requirements

* Raspberry Pi 5 (with Ethernet or USB-Ethernet adapter)
* Pixhawk 6X flight controller (running ArduCopter firmware)
* Ethernet cable or router/switch (to connect Pi and Pixhawk)
* JST-GH to Ethernet adapter (for Pixhawk Ethernet port)
* MicroSD card with Raspberry Pi OS
* Power supply for both devices

Optional:

* USB-to-Serial adapter (for fallback communication)

1. Software Requirements

* Raspberry Pi OS (64-bit)
* MAVProxy (MAVLink ground control CLI)
* ArduCopter firmware (installed on Pixhawk)
* QGroundControl (optional, for GUI-based verification)

1. Network Configuration

Option A: Direct Ethernet (Static IP)

* Connect Raspberry Pi directly to Pixhawk via Ethernet
* Assign static IPs:
  + Raspberry Pi: 192.168.4.1
  + Pixhawk: 192.168.4.2

Option B: Via Router/Switch (DHCP or Static)

* Connect both Raspberry Pi and Pixhawk to a network
* Assign IPs via DHCP or static IP configuration

Pixhawk MAVLink Parameters:

* MAV\_1\_CONFIG = Ethernet
* MAV\_1\_MODE = Normal (1)
* MAV\_1\_REMOTE\_PRT = 14550
* MAV\_1\_REMOTE\_IP = [Raspberry Pi IP]

1. How Communication Works

The communication between the Raspberry Pi and Pixhawk via MAVLink over Ethernet follows a client-server UDP model:

* The Pixhawk acts as a MAVLink endpoint configured to send MAVLink messages over a UDP connection to a designated IP and port (typically the Raspberry Pi).
* The Raspberry Pi runs MAVProxy, which listens on a specified UDP port (e.g., 14550) for incoming MAVLink messages.
* MAVProxy can also send messages (e.g., commands, mode switches) back to the Pixhawk using a reciprocal UDP connection (Raspberry Pi as sender, Pixhawk as receiver).
* This bi-directional communication allows telemetry data (GPS, attitude, battery, etc.) to be received from the Pixhawk and control commands (e.g., takeoff, land, mode change) to be sent from the Raspberry Pi.

Key components:

* UDP Protocol: Connectionless, lightweight protocol ideal for real-time telemetry.
* MAVLink Messages: Structured data packets defined by the MAVLink protocol, serialized and transmitted over UDP.
* ArduPilot Firmware: Listens for MAVLink messages and reacts accordingly (e.g., arming the drone, executing waypoints).
* MAVProxy: Parses MAVLink packets, displays information, logs data, and allows operator control.

In effect, the Ethernet connection becomes a digital telemetry link, with Raspberry Pi acting as a network-based ground control station.

1. Installing and Configuring MAVProxy

Install MAVProxy on Raspberry Pi:

sudo apt update

sudo apt install python3-pip -y

pip3 install MAVProxy

Run MAVProxy:

mavproxy.py --master=udp:0.0.0.0:14550 --out=udp:192.168.4.2:14550

* This listens on all interfaces on port 14550
* Adjust the --out IP/port to match Pixhawk's expected address

1. Verifying Communication

When MAVProxy starts successfully, you should see output like:

Online system 1

Mode STABILIZE

Test commands:

mode GUIDED

arm throttle

takeoff 3

Use these commands to change flight modes and initiate takeoff (only on safe setups or simulation).

1. Sending Commands and Testing Modes

Switching modes:

mode STABILIZE

mode ACRO

mode AUTO

You can also use QGroundControl to monitor messages and switch modes for visual confirmation.

1. Measuring and Monitoring Latency

To measure latency, start by opening QGroundControl and navigating to the MAVLink Inspector (via the Widgets menu). This tool displays real-time MAVLink messages along with their timestamps and intervals. Observe the time between sent commands and corresponding telemetry updates to estimate communication delays.

Alternatively, with MAVProxy running, you can monitor the timestamped logs in the terminal or redirect logs to a file using:

mavproxy.py --master=udp:0.0.0.0:14550 --logfile=log.tlog

Then analyze the .tlog file using tools like Mission Planner or DroneKit.

For detailed packet-level inspection, install Wireshark on your Raspberry Pi or a connected computer. Add the MAVLink dissector plugin, then capture packets on the Ethernet interface. Filter by UDP port (e.g., udp.port == 14550) and examine timestamps between command and response packets to calculate round-trip latency.

1. Troubleshooting

* If no heartbeat: Verify IP settings and physical connections
* If no MAVProxy output: Check if UDP port is correctly set
* If commands fail: Confirm flight mode allows command (e.g., GUIDED required for takeoff)
* Use ping or arp-scan to ensure Pixhawk is reachable

1. References

* ArduPilot MAVLink: <https://ardupilot.org/dev/docs/mavlink.html>
* MAVProxy: <https://ardupilot.github.io/MAVProxy/html/index.html>
* QGroundControl: <https://docs.qgroundcontrol.com/en/>
* ArduCopter Flight Modes: <https://ardupilot.org/copter/docs/flight-modes.html>
* MAVLink UDP Setup: <https://ardupilot.org/dev/docs/mavlink-udp.html>