Complete PX4 Autonomous Mission Guide (Without QGroundControl)

This is a comprehensive guide that walks through every step taken to set up Ubuntu (via WSL2), install PX4 Autopilot, configure jMAVSim simulation, enable MAVLink broadcasting, install MAVSDK, write a Python script, and run an autonomous mission — all without using QGroundControl.

Step 1: Install Ubuntu via WSL2 on Windows

1. Open PowerShell as Administrator and run:

    ```
    wsl --install
    ```

2. Restart your PC when prompted.
3. Launch Microsoft Store and install **Ubuntu 22.04 LTS**.
4. Open Ubuntu from Start Menu and set up your username and password.
5. Update Ubuntu:

    ```
    sudo apt update && sudo apt upgrade -y
    ```

Step 2: Install PX4 Autopilot in Ubuntu

1. Install system dependencies:

```
sudo apt install git wget python3-pip python3-jinja2 ninja-build exiftool zip
unzip protobuf-compiler -y
sudo apt install openjdk-17-jdk ant -y
```

1. Clone PX4-Autopilot repository:

```
git clone https://github.com/PX4/PX4-Autopilot.git --recursive
cd PX4-Autopilot
```

1. Run PX4 setup script:

```
bash ./Tools/setup/ubuntu.sh
```

This sets up the required toolchains and development tools.

Step 3: Build and Launch PX4 with jMAVSim

1. Build and launch simulation:

```
make px4_sitl_default jmavsim
```

1. PX4 will launch with jMAVSim GUI and show log messages in the terminal.

2. Keep this terminal open and running — it hosts the simulation.

Step 4: Enable MAVLink Broadcast

By default, PX4 only communicates on `localhost`. To allow MAVSDK (or QGC) to connect:

1. In PX4 terminal, enter:

   ```
   param set MAV_0_BROADCAST 1
   ```

2. This tells PX4 to send MAVLink messages over the network.

Step 5: Download and Install QGroundControl (Optional)

1. Go to: https://docs.qgroundcontrol.com/master/en/getting_started/download_and_install.html
2. Download the **AppImage for Linux**.
3. In Ubuntu, make it executable:

   ```
   chmod +x QGroundControl.AppImage
   ./QGroundControl.AppImage
   ```

You can use QGC to verify connectivity and visually monitor missions, though this guide does not require it.

Step 6: Install MAVSDK for Python

1. Install Python package:

```
pip install mavsdk
```

This installs the Python SDK to control PX4 over MAVLink.

Step 7: Create Python Script for Autonomous Mission

1. Create a new file called `mission.py`:

```
import asyncio
from mavsdk import System
from mavsdk.mission import MissionItem, MissionPlan, MissionItem

async def run():
    # Step 1: Connect to PX4 SITL
    drone = System()
    await drone.connect(system_address="udp://:14540")
    print("Waiting for drone to connect...")
    async for state in drone.core.connection_state():
        if state.is_connected:
            print(f"Drone discovered!")
            break

    # Step 2: Wait for Global Position Estimate
    print("Waiting for global position estimate...")
    async for health in drone.telemetry.health():
        if health.is_global_position_ok and health.is_home_position_ok:
            print("Global position estimate OK")
            break

    # Step 3: Define Mission Items
    mission_items = [
        MissionItem(
            47.398039859999997,   # Latitude
            8.5455725400000002,   # Longitude
            10,                   # Relative Altitude (meters)
            5,                    # Speed to reach waypoint (m/s)
            True,                 # Fly through this waypoint
            float('nan'),         # Gimbal pitch
            float('nan'),         # Gimbal yaw
            MissionItem.CameraAction.NONE,  # Camera action
            float('nan'),         # Loiter time at waypoint
            float('nan'),         # Photo interval
            float('nan'),         # Acceptance radius
            float('nan'),         # Yaw
            float('nan'),         # Camera photo distance
            MissionItem.VehicleAction.NONE  # Vehicle action
        )
    ]

    # Step 4: Upload the mission
    mission_plan = MissionPlan(mission_items)
    await drone.mission.set_return_to_launch_after_mission(True)
    print("Uploading mission...")
    await drone.mission.upload_mission(mission_plan)
    print("Mission uploaded.")

    # Step 5: Arm and Start the Mission
    print("Arming...")
    await drone.action.arm()
    print("Starting mission...")
    await drone.mission.start_mission()

    # Step 6: Monitor Mission Progress
    async for mission_progress in drone.mission.mission_progress():
        print(f"Mission progress: {mission_progress.current}/
{mission_progress.total}")
        if mission_progress.current == mission_progress.total:
            print("Mission completed!")
            break

    # Step 7: Wait until landed
    async for in_air in drone.telemetry.in_air():
        if not in_air:
            print("Landed!")
            break

    print("Disarming...")
    await drone.action.disarm()
    print("Mission finished.")

if __name__ == "__main__":
    asyncio.run(run())
```

1. Save the file in your home directory.

Step 8: Launch PX4 SITL in First Terminal

1. Open Terminal #1:

```
cd PX4-Autopilot
make px4_sitl_default jmavsim
```

1. This launches the drone simulation and PX4 firmware.

2. Keep it open — do **not** close this terminal.

Step 9: Run Autonomous Mission Script in Second Terminal

1. Open a **new terminal** (Terminal #2).

2. Run the script:

```
python3 mission.py
```

1. You should see:

2. Drone discovered

3. Global position OK
4. Drone armed
5. Mission started

The drone will take off and follow the waypoint(s) in your script.

Notes:

- If the script fails due to `MissionItem` argument errors, double-check you are passing all required parameters.
- You can add more waypoints by appending to `mission_items` list.
- PX4 SITL communicates on UDP port `14540` by default.
- Always ensure PX4 terminal is running before starting the mission script.

Troubleshooting:

- **Drone not discovered**: PX4 may not be broadcasting MAVLink → `param set MAV_0_BROADCAST 1`
- **Mission not starting**: Check for global position and connection state
- **Takeoff not happening**: Ensure drone is armed and mission uploaded correctly

---

You have now completed an end-to-end autonomous mission setup using PX4 SITL, MAVSDK Python, and jMAVSim — all without needing QGroundControl.

Let me know if you want to add:

- Live telemetry visualization
- Manual control override
- Camera/photo triggers
- Return-to-launch scenarios
- Offboard control via keyboard/joystick