# Quick‑Start Guide: Fully‑GUIDED Mission with SITL + MAVProxy + DroneKit

This doc bundles **everything** you need to replicate the demo on Windows (or Linux/macOS) using three terminals:

* **Terminal 1** – start the *Software‑In‑the‑Loop* (SITL) simulator.
* **Terminal 2** – run **MAVProxy** to fan‑out data to Mission Planner *and* your script.
* **Terminal 3** – run auto\_guided.py, which stays in GUIDED from take‑off through yaw, forward flight, and forced disarm.

## 0. Prerequisites

| Component | Install cmd (PowerShell / CMD) |
| --- | --- |
| Python ≥ 3.10 | python --version (from **python.org**) |
| Virtual‑env | python -m venv .venv && .venv\Scripts\activate |
| DroneKit + SITL | pip install dronekit dronekit-sitl |
| MAVProxy | pip install MAVProxy |
| Mission Planner | Download & install (MissionPlanner\_<version>.msi) |

*Note*: On Python ≥3.10, DroneKit needs the **MutableMapping** hot‑patch (already in auto\_guided.py).

## 1. Launch the stack

### Terminal 1 — SITL

python -m dronekit\_sitl copter --model quad --home=47.398,8.545,10,0

* Opens **TCP 5760** internally.
* Leave this window running.

### Terminal 2 — MAVProxy two‑port bridge

python .venv\Scripts\mavproxy.py --master tcp:127.0.0.1:5760 --out udp:127.0.0.1:14550  
 --out udp:127.0.0.1:14551 # DroneKit script

Expected MAVProxy banner:

Master: tcp:127.0.0.1:5760  
output add 127.0.0.1:14550  
output add 127.0.0.1:14551

### (Optional) Mission Planner HUD

1. Click **CONNECT** → UDP.
2. Local Port = **14550** → **Start**.
3. Watch the HUD but keep it read‑only.

## 2. Run the fully‑GUIDED mission

### Terminal 3 — Python code

python auto\_guided.py

The script will:

1. **Connect** over UDP 14551.
2. **Disable arming checks** (SITL convenience).
3. **Take‑off** to 10 m.
4. simple\_goto() **5 m north**.
5. **Yaw east** using CONDITION\_YAW.
6. **Descend** slowly to ground.
7. **Force‑disarm** via MAV\_CMD\_COMPONENT\_ARM\_DISARM.

Console output ends with:

🏁 Completed guided flight and safe landing

## 3. Source code (auto\_guided.py)

#!/usr/bin/env python3  
"""  
auto\_guided.py — Fully GUIDED‑mode demo over UDP 14551  
  
Workflow  
========  
1. \*\*Connect\*\* to MAVProxy‑forwarded UDP 14551.  
2. \*\*Arm & take off\*\* to 10 m in \*\*GUIDED\*\*.  
3. \*\*Fly forward ≈5 m north\*\* with `simple\_goto()` (still in GUIDED).  
4. \*\*Yaw\*\* to face east as an example (optional).  
5. \*\*Land\*\* — all without ever switching out of GUIDED.  
  
Prerequisites  
-------------  
```bash  
# Terminal 1 – SITL  
python -m dronekit\_sitl copter --model quad --home=47.398,8.545,10,0  
  
# Terminal 2 – MAVProxy bridge  
python .venv/Scripts/mavproxy.py \  
 --master tcp:127.0.0.1:5760 \  
 --out udp:127.0.0.1:14550 \  
 --out udp:127.0.0.1:14551  
  
# Mission Planner → UDP 14550 (optional)  
# Terminal 3 → python auto\_guided.py  
```  
"""  
  
import math, time, collections, collections.abc  
collections.MutableMapping = collections.abc.MutableMapping # Python ≥3.10 fix  
  
from dronekit import connect, VehicleMode, LocationGlobalRelative  
from pymavlink import mavutil  
  
CONN\_STR = "udp:127.0.0.1:14551" # keep original UDP port  
TARGET\_ALT = 10 # take‑off altitude (m)  
D\_NORTH = 5 # metres to travel north  
  
# ———————————————————————————————————————————————————————————  
# Geo helpers  
# ———————————————————————————————————————————————————————————  
EARTH\_RADIUS = 6378137.0  
  
def get\_location\_metres(original, dNorth, dEast):  
 """Return LocationGlobalRelative offset by dNorth/dEast metres."""  
 dLat = dNorth / EARTH\_RADIUS  
 dLon = dEast / (EARTH\_RADIUS \* math.cos(math.pi \* original.lat / 180))  
 newlat = original.lat + (dLat \* 180 / math.pi)  
 newlon = original.lon + (dLon \* 180 / math.pi)  
 return LocationGlobalRelative(newlat, newlon, original.alt)  
  
  
def distance\_metres(a, b):  
 dlat = b.lat - a.lat  
 dlon = b.lon - a.lon  
 return math.sqrt((dlat \* dlat) + (dlon \* dlon)) \* 1.113195e5  
  
# ———————————————————————————————————————————————————————————  
# Vehicle helpers  
# ———————————————————————————————————————————————————————————  
  
def set\_mode(v, mode, timeout=10):  
 v.mode = VehicleMode(mode)  
 start = time.time()  
 while time.time() - start < timeout:  
 if v.mode.name == mode:  
 print(f"✔ Mode → {mode}")  
 return  
 time.sleep(0.3)  
 raise RuntimeError(f"Mode change to {mode} failed")  
  
  
def arm\_and\_takeoff(v, alt):  
 print("⏳ Waiting for vehicle to be armable …")  
 while not v.is\_armable:  
 time.sleep(1)  
  
 print("⚙ Disabling arming checks (SITL demo)")  
 v.parameters["ARMING\_CHECK"] = 0 # demo only!  
  
 set\_mode(v, "GUIDED")  
 print("⚡ Arming motors …")  
 v.armed = True  
 while not v.armed:  
 time.sleep(0.5)  
  
 print(f"🚀 Taking off to {alt} m")  
 v.simple\_takeoff(alt)  
 while True:  
 current\_alt = v.location.global\_relative\_frame.alt  
 print(f" Altitude: {current\_alt:.1f} m")  
 if current\_alt >= alt \* 0.95:  
 print("✔ Target altitude reached")  
 break  
 time.sleep(0.5)  
  
  
def condition\_yaw(v, heading, relative=False):  
 msg = v.message\_factory.command\_long\_encode(  
 0, 0,  
 mavutil.mavlink.MAV\_CMD\_CONDITION\_YAW,  
 0,  
 heading, 0, 1, 1 if relative else 0,  
 0, 0, 0)  
 v.send\_mavlink(msg)  
  
# ———————————————————————————————————————————————————————————  
# Main  
# ———————————————————————————————————————————————————————————  
if \_\_name\_\_ == "\_\_main\_\_":  
 print(f"🔌 Connecting → {CONN\_STR}")  
 vehicle = connect(CONN\_STR, wait\_ready=True)  
  
 # 1) Take‑off  
 arm\_and\_takeoff(vehicle, TARGET\_ALT)  
  
 # 2) Compute target location 5 m north of current pos (same altitude)  
 home = vehicle.location.global\_relative\_frame  
 target = get\_location\_metres(home, D\_NORTH, 0)  
 print(f"🧭 simple\_goto → {D\_NORTH} m north (GUIDED)")  
 vehicle.simple\_goto(target, groundspeed=1) # m/s  
  
 # Wait until within ~0.5 m  
 while distance\_metres(vehicle.location.global\_relative\_frame, target) > 0.5:  
 print(" … travelling")  
 time.sleep(1)  
 print("✔ Reached target point")  
  
 # 3) Yaw east (optional demo of CONDITION\_YAW)  
 print("↻ Yawing to face east (90°)")  
 condition\_yaw(vehicle, 90)  
 time.sleep(2)  
  
 # 4) Controlled descent and disarm  
 print("⬇ Descending … (still in GUIDED)")  
 landing\_target = LocationGlobalRelative(  
 vehicle.location.global\_relative\_frame.lat,  
 vehicle.location.global\_relative\_frame.lon,  
 0.0,  
 )  
 vehicle.simple\_goto(landing\_target, groundspeed=0.5) # slow descent  
  
 # Wait until we're <0.5 m above ground  
 while vehicle.location.global\_relative\_frame.alt > 0.5:  
 print(f" Alt: {vehicle.location.global\_relative\_frame.alt:.1f} m → descending")  
 time.sleep(1)  
  
 # Ensure autopilot detects a landing state  
 print("⚡ Initiating disarm … (force command)")  
 # MAV\_CMD\_COMPONENT\_ARM\_DISARM, param1=0 (disarm), param2=21196 (force)  
 msg = vehicle.message\_factory.command\_long\_encode(  
 0, 0,  
 mavutil.mavlink.MAV\_CMD\_COMPONENT\_ARM\_DISARM,  
 0,  
 0, # param1 = 0 → disarm  
 21196, # param2 = magic number to force disarm  
 0, 0, 0, 0, 0)  
 vehicle.send\_mavlink(msg)  
  
 # Wait until the vehicle reports disarmed  
 while vehicle.armed:  
 time.sleep(0.5)  
  
 vehicle.close()  
 print("🏁 Completed guided flight and safe landing")()  
 print("🏁 Completed guided flight and safe landing")

*The latest version lives in Canvas →* ***Auto Mission***\* code tab.\*

## 4. Common pitfalls

| Symptom | Fix |
| --- | --- |
| Mode change to GUIDED failed | Ensure MAVProxy running; patch ARMING\_CHECK = 0. |
| Script hangs at connect() | Wrong port – use **14551** not 14550. |
| Mission Planner steals connection | Keep MP on 14550; script on 14551. |
| Drone won’t disarm | Script now sends **force‑disarm** magic 21196 param. |

Happy flying! Feel free to duplicate the simple\_goto() leg, change D\_NORTH / D\_EAST, or add set\_roi() for camera pointing.