Dial a Drone

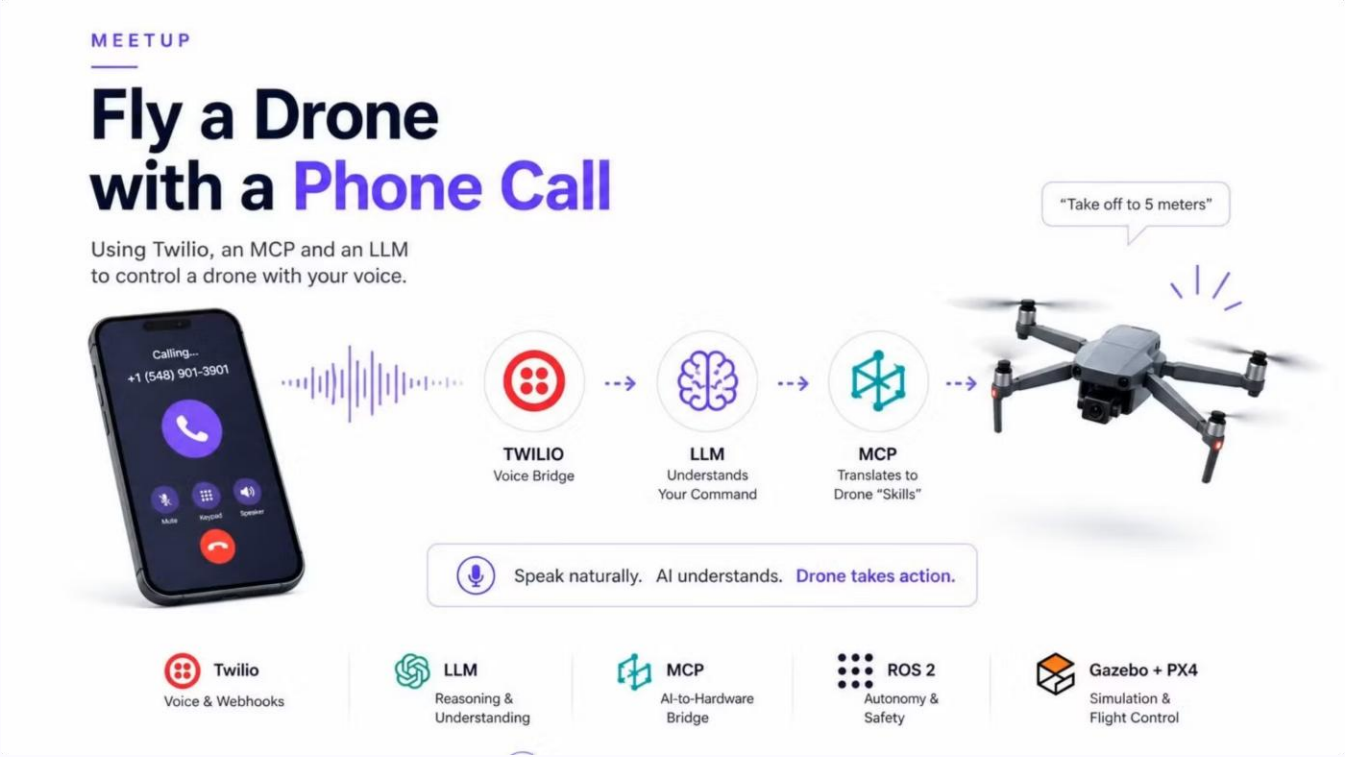
PX4 Developers Conference May ‘26

Godfrey Nolan

godfrey@riis.com

Mission Accomplished

Architecture



Use Cases

Search & Rescue

Deploying robots in dangerous or hard-to-reach environments to locate and assist survivors.

Inspection

Industrial and infrastructure inspection using autonomous robotic systems.

Military / Defense Simulations

Realistic defense training and simulation environments powered by robotics.

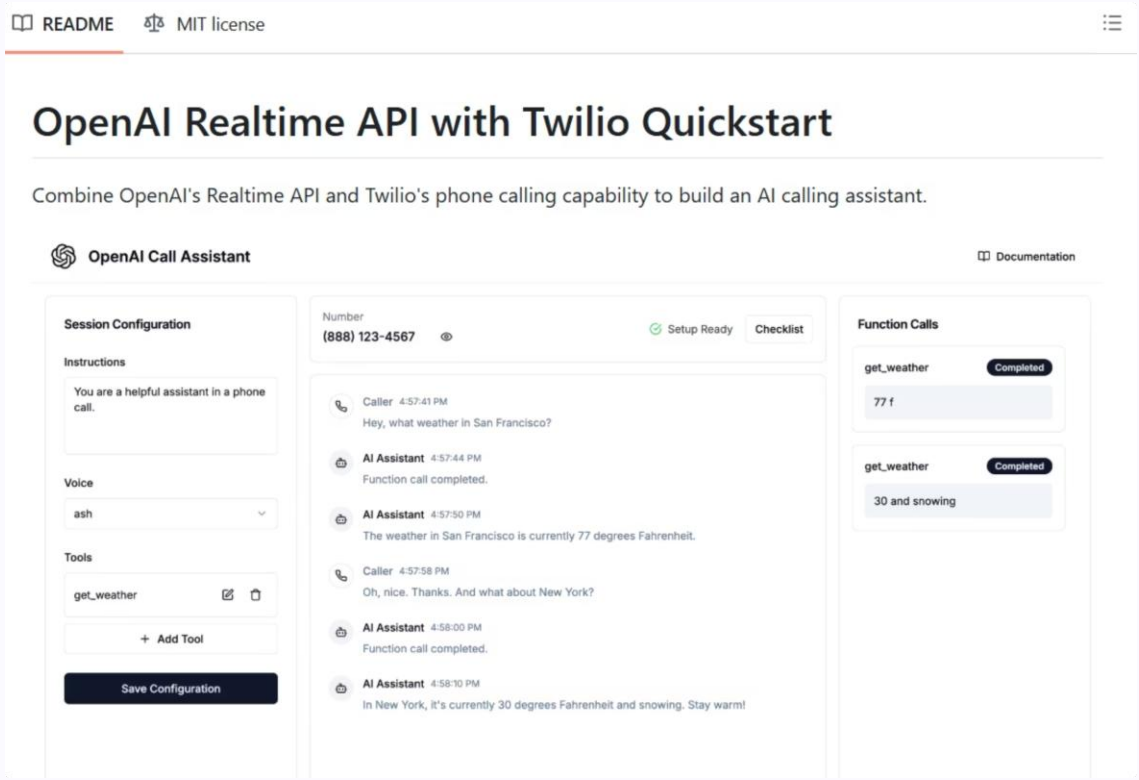
Education & Robotics Training

Hands-on learning experiences for students and professionals in robotics.

Accessibility

Voice-first robotics enabling greater independence for people with disabilities.

Architecture



Phone Call → Twilio → /media-stream (WebSocket)

↓

VoiceSession bridges audio to OpenAI

↓

OpenAI decides → function_call

↓

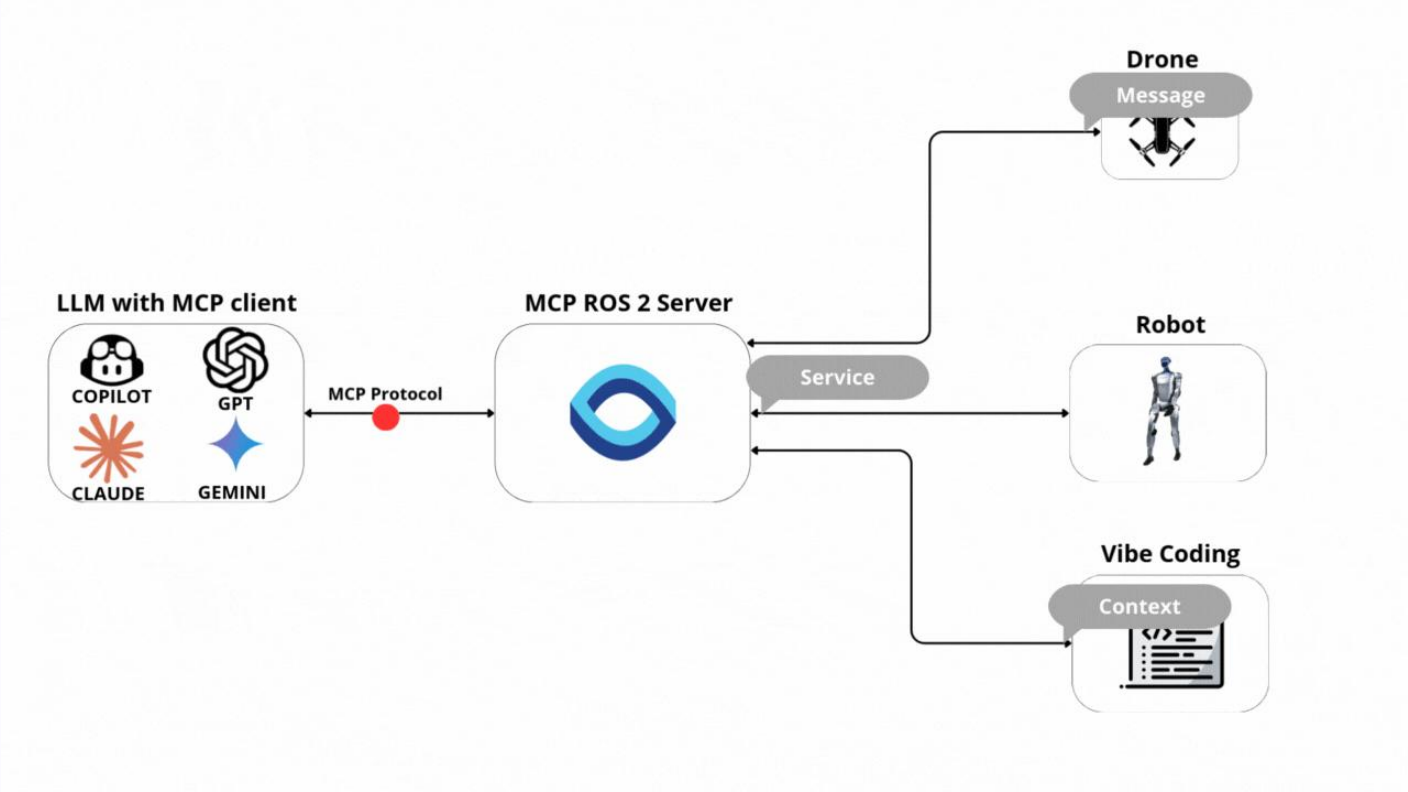
DockerMCPClient executes on ROS 2 drone

↓

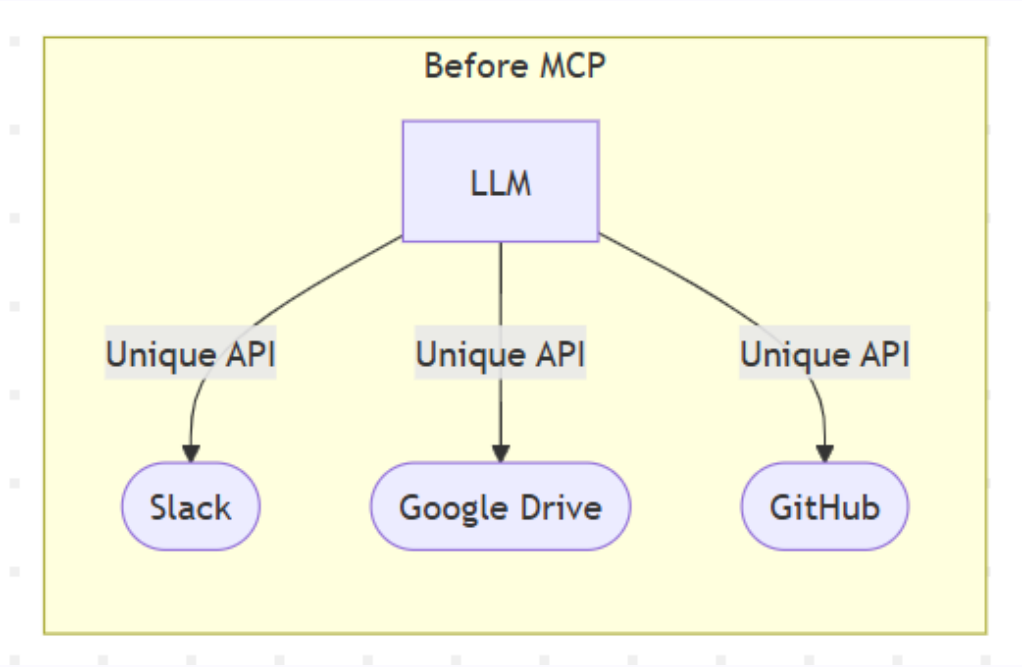
Result fed back to OpenAI → spoken response → Twilio → Caller

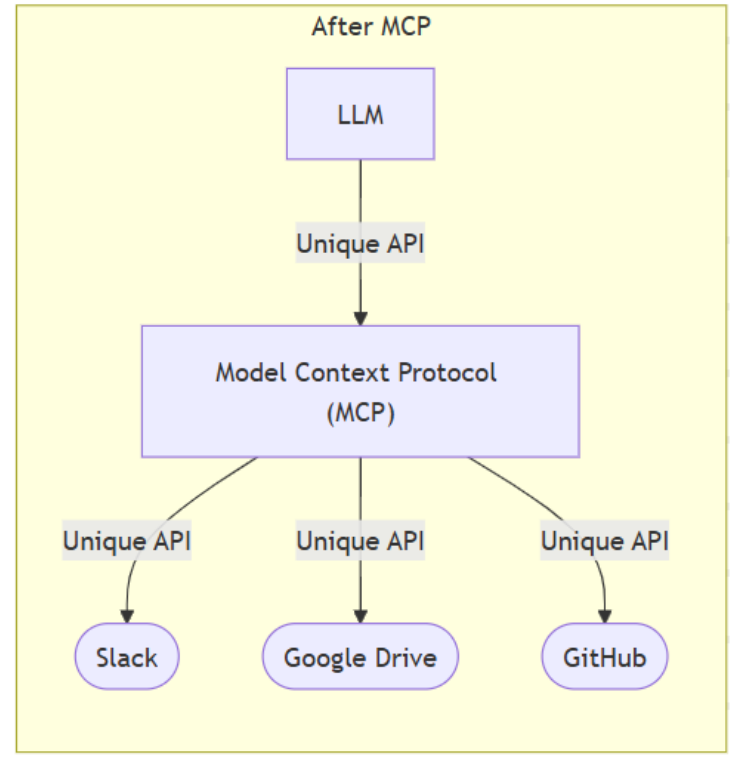
```
SYSTEM_PROMPT = """
You are an expert Drone Pilot.
Your goal is to fly the drone safely and accurately based on user voice commands.
IMPORTANT: Always respond in English only. Never use any other language.


You have access to a suite of "Smart Skills" that handle the complex flight logic for you.
Do NOT try to fly the drone manually (e.g. do not calculate vectors yourself).
ALWAYS use the provided high-level tools.

### Available Skills:
- **Pre-Flight:** `preflight_check()` — Run before takeoff to verify FCU, GPS, battery.
- **Status:** `get_mode()`, `get_heading()`
- **Takeoff/Land:** `perform_takeoff(altitude)`, `perform_landing()`, `return_to_launch()`
- **Movement:** `move_relative(direction, distance)`, `set_altitude(height)`, `hold_position()`
  - direction must be one of: 'forward', 'backward', 'left', 'right', 'up', 'down'
- **Turning (relative):** `turn(degrees)` — Rotate relative to current heading. Positive = right/clockwise, Negative = left/counter-clockwise.
  - Example: "Turn right 90" → `turn(90)`, "Turn around" → `turn(180)`, "Turn left 45" → `turn(-45)`
- **Heading (absolute):** `set_heading(angle)` — Face an exact compass bearing (0=N, 90=E, 180=S, 270=W).
  - Only use when the user specifies a compass direction (e.g. "face north", "heading 270").
- **Patterns:** `fly_pattern(pattern, radius)` - pattern is 'circle', 'square', or 'figure8'
- **Search:** `scan_area(width)`
- **Gimbal Control:** `set_gimbal(pitch, roll, yaw)` — Always set roll=0 unless explicitly asked.
  - Pitch: -90 (straight down) to 0 (horizon) to 30 (up)
  - Yaw: -180 to 180 (pan left/right relative to drone body)
  - Common positions: "nadir"/"look down" → `set_gimbal(-90, 0, 0)`, "look forward" → `set_gimbal(0, 0, 0)`, "pan left" → `set_gimbal(0, 0, -90)`, "pan right" →
- **Vision:** `take_photo()` (Returns an image)
```

Architecture



What is MCP?


Before MCP
LLM
Unique API
Unique API
Unique API
Slack
Google Drive
GitHub


After MCP
LLM
Unique API
Model Context Protocol
(MCP)
Unique API
Unique API
Unique API
Slack
Google Drive
GitHub

Why MCP Matters

Beyond Chatbots

Autonomous execution of complex tasks

Self-correcting behavior

Multi-step reasoning

Persistent Reasoning

Maintains context across sessions

Learns from previous interactions

Builds knowledge base over time

MCP Server Backend

State Storage

Maintains model context between interactions

Tool Orchestration

Manages plugin execution and integration

Workflow Management

Coordinates multi-step autonomous processes

Prompt Routing

Directs inputs to appropriate handlers

Build Your Own MCP Server

Choose Your Stack

Select frameworks and languages that fit your needs

Implement Context Storage

Build persistent memory with vector and relational systems

Create Tool Interfaces

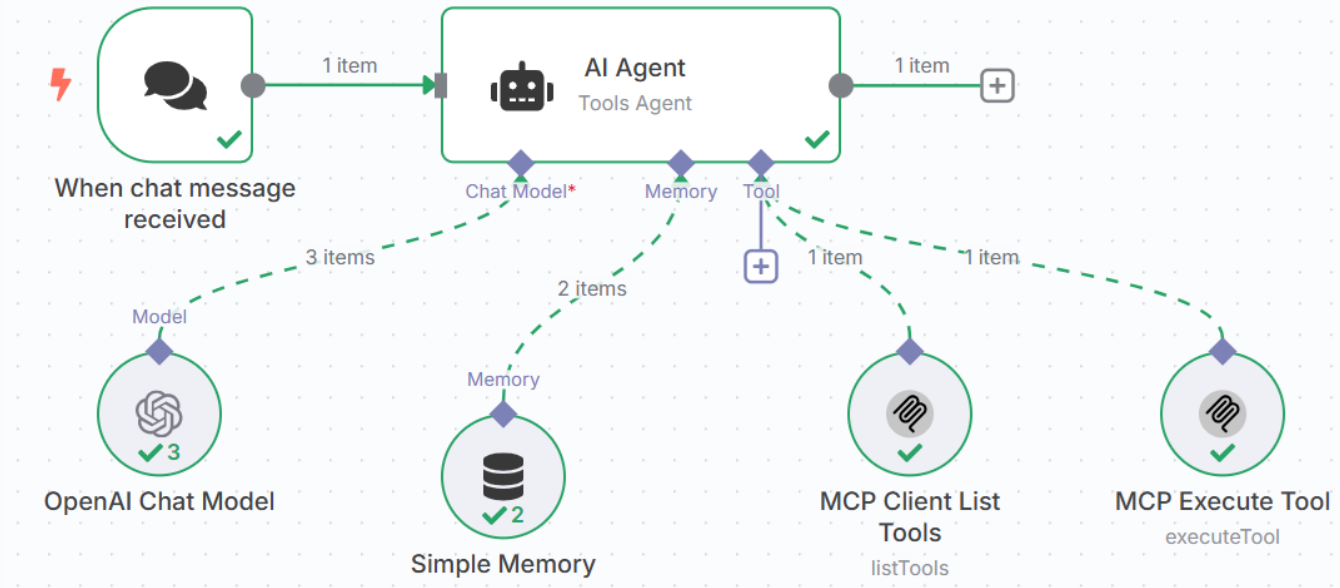
Design flexible APIs for external tool access

Define Agent Architecture

Establish roles, communication patterns, and logic flows

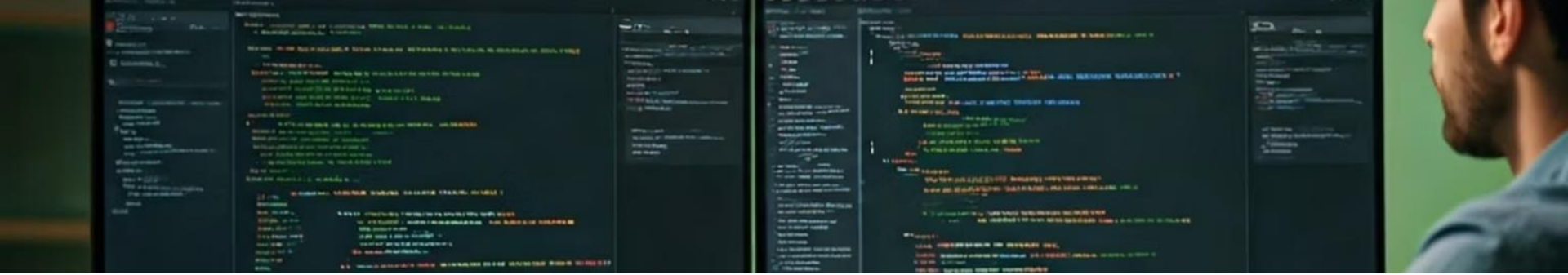
Integrate Model Backend

Connect to LLM providers with prompting


1 item
AI Agent
Tools Agent
1 item
When chat message received
Chat Model*
Memory
Tool
3 items
2 items
1 item
1 item
Model
3
OpenAI Chat Model
Memory
2
Simple Memory
MCP Client List Tools
listTools
MCP Execute Tool
executeTool

```
async with MCPServerStdio(
    params={
        "command": "npx",
        "args": ["-y", "@modelcontextprotocol/server-filesystem", samples_dir],
    }
) as server:
    tools = await server.list_tools()
```

```
async def main():
    async with MCPServerSse(
        name="SSE Python Server",
        params={
            "url": "http://localhost:8000/sse",
        },
    ) as server:
        trace_id = gen_trace_id()
        with trace(workflow_name="SSE Example", trace_id=trace_id):
            print(f"View trace: https://platform.openai.com/traces/trace?trace_id={trace_id}\n")
            await run(server)
```



Technical Implementation

API Endpoints

WebSockets for real-time or HTTP for RESTful

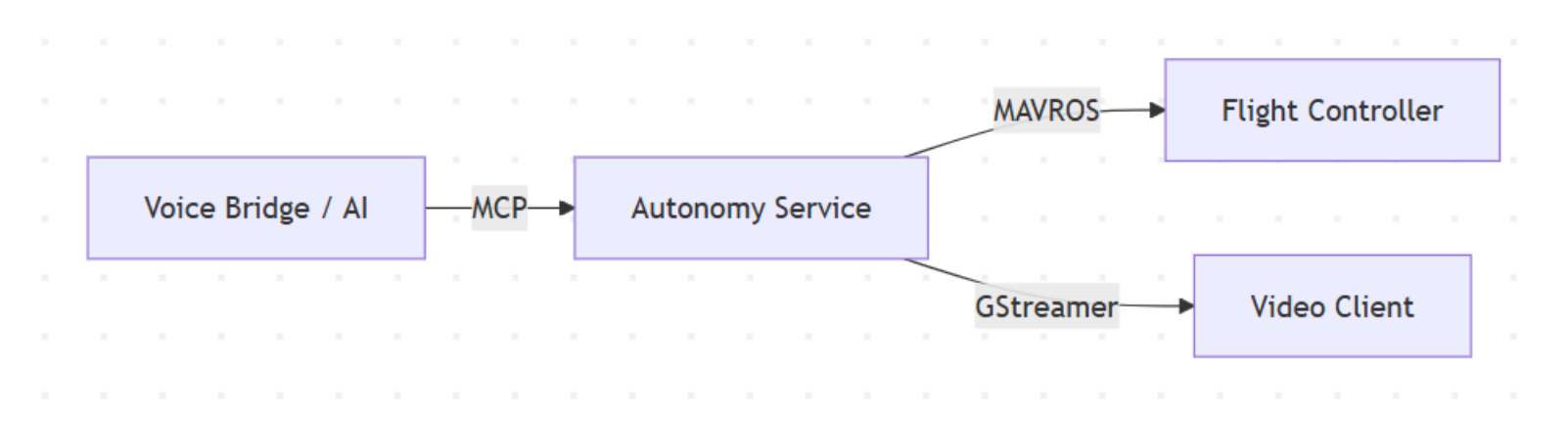
Vector Stores

Efficient semantic search capabilities

Persistent Storage

Database-backed memory with query optimization

Build Your Own MCP Server



Skills.py

Flight Skills / Capabilities

Skill	Description
`preflight_check`	Validates FCU connection, GPS fix, and battery before flight
`perform_takeoff`	Streams setpoints → switches to OFFBOARD → arms → ascends
`perform_landing` / `return_to_launch`	Switches to `AUTO.LAND` or `AUTO.RTL`
`hold_position`	Re-engages OFFBOARD at current position
`set_altitude` / `set_heading`	Smooth altitude/yaw changes
`move_relative`	Body-frame relative movement (forward/back/left/right/up/down) with heading-aware coordinate rotation
`fly_pattern`	Executes circle, square, or figure-8 flight patterns
`scan_area`	Lawnmower search pattern over a square area
`take_photo`	Captures and returns a base64-encoded JPEG from the camera feed
`set_gimbal`	Controls gimbal pitch/roll/yaw via MAVROS `MountControl`

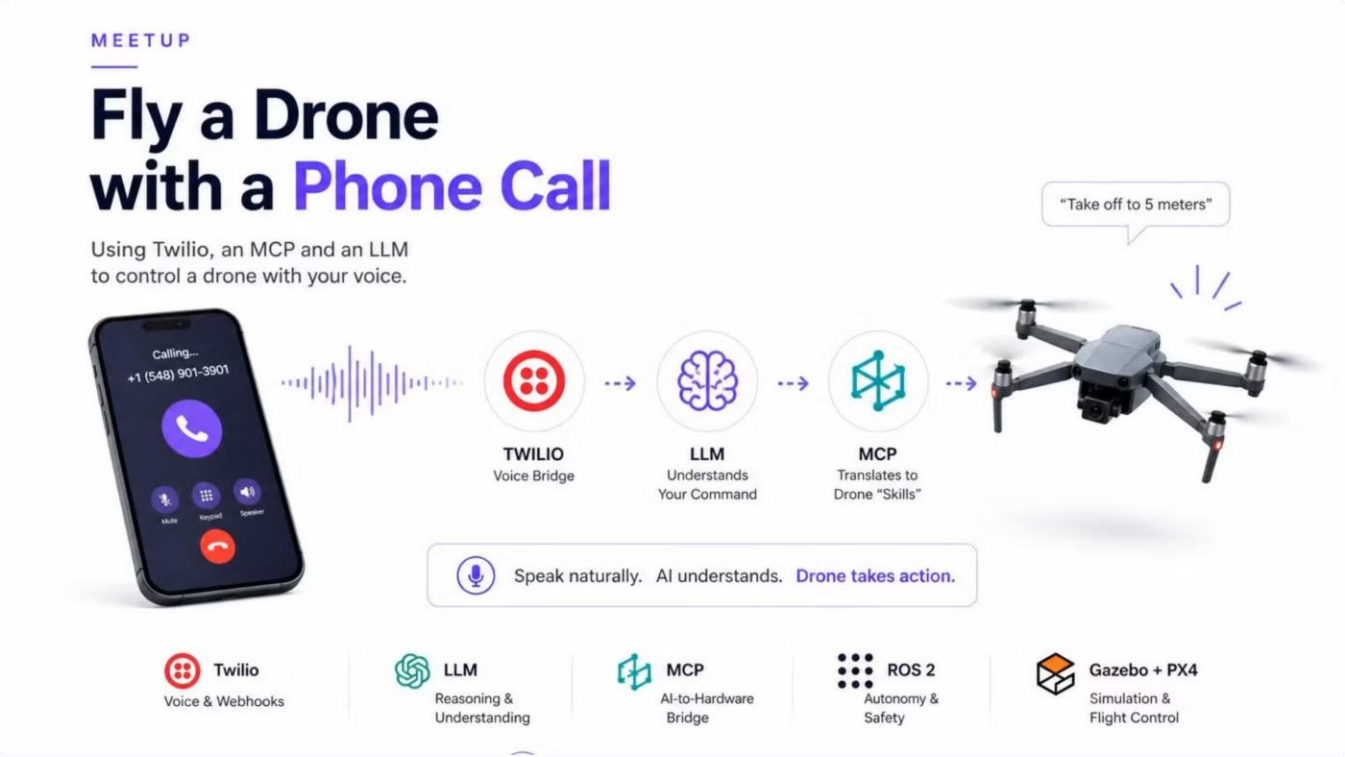
Server.py

Category	Detail
Framework	`FastMCP` — initializes a server named `"SmartDroneAgent"`
Core Dependency	`DroneSkills` — global instance that handles all actual drone logic
System Resource	`Status://system` — reports the drone's current operational status
Telemetry Tools	`get_mode`, `get_heading`, `get_telemetry`
Pre-flight Tools	`preflight_check`
Flight Control Tools	`perform_takeoff`, `perform_landing`, `return_to_launch`, `hold_position`
Movement Tools	`set_altitude`, `set_heading`, `turn`, `move_relative`
Pattern Tools	`fly_pattern` (circle, figure-8, square), `scan_area` (lawnmower search)
Camera Tools	`take_photo`, `set_gimbal`
Guard Pattern	Every tool checks `if not skills` before executing, returning a safe error string if uninitialized
Logic	Server is a thin wrapper — delegates almost all logic to `DroneSkills`, except `turn()` which computes a new absolute heading locally
Transport	Runs over `stdio`, making it suitable as a subprocess for an LLM host
Entry Point	`main()` instantiates `DroneSkills` and starts the MCP server

Video_streamer.py

Responsibility	Details
ROS 2 Subscription	Subscribes to `/camera/image_raw` using a sensor-data QoS profile (best-effort), compatible with Gazebo/simulation environments
Image Conversion	Uses `cv_bridge` to convert incoming ROS `Image` messages to OpenCV (`bgr8`) frames
GStreamer Streaming	Encodes frames via H.264 (`x264enc`) with zero-latency tuning and streams over UDP via RTP (`rtph264pay` → `udpsink`)
Lazy Initialization	The `cv2.VideoWriter` is initialized on the first frame so resolution is inferred automatically at runtime
Configurable Target	Stream host and port are ROS 2 parameters (defaulting to `127.0.0.1:5600`), making it easy to retarget without code changes
Clean Shutdown	`destroy_node()` releases the `VideoWriter` before calling the parent shutdown, preventing resource leaks

Architecture



Setup

- git clone https://github.com/godfreynolan/voice-bridge
- git clone https://github.com/godfreynolan/autonomy-service
- git clone https://github.com/PX4/PX4-Autopilot --recursive

Voice Configuration

- `OPENAI_API_KEY=""`
- `DRONE_CONTAINER_NAME=autonomy-service`
- `PORT=8080`
- `TWILIO_ACCOUNT_SID=""`
- `TWILIO_AUTH_TOKEN=""`
- `TWILIO_PHONE_NUMBER=""`
- `NGROK_AUTH_TOKEN=""`

Autonomy Configuration

- `SIMULATION_MODE=true`
- `MAVROS_FCU_URL=udp://:14540@host.docker.internal:14580`
- `ENABLE_VIDEO=true`
- `PX4_GZ_WORLD=default`
- `GZ_MODEL_NAME=x500_gimbal_0`
- `ROS_DOMAIN_ID=0`
- `VIDEO_HOST=host.docker.internal`
- `VIDEO_PORT=5600`

Setup and Configuration

Clone Repos

```
mkdir drone-project
cd drone-project
# Get core services
git clone .../autonomy-service
git clone .../voice-bridge
# Recursive PX4 clone
git clone https://github.com/PX4/PX4-
Autopilot.git --recursive
cd PX4-Autopilot
```

Configure Keys

a) Edit voice-bridge/.env:

```
OPENAI_API_KEY=sk-...
DRONE_CONTAINER_NAME=autonomy-service
PORT=8080
TWILIO_ACCOUNT_SID=AC...
TWILIO_AUTH_TOKEN=...
TWILIO_PHONE_NUMBER=...
NGROK_AUTH_TOKEN=...
```

b) Edit autonomy-service/.env

```
SIMULATION_MODE=true
MAVROS_FCU_URL=udp://:14540@host.docker.i
nternal:14580
ENABLE_VIDEO=true
PX4_GZ_WORLD=default
GZ_MODEL_NAME=x500_gimbal_0
ROS_DOMAIN_ID=0
VIDEO_HOST=host.docker.internal
VIDEO_PORT=5600
```

Install Voice Bridge

```
cd voice-bridge
# Create env
python3 -m venv .venv
.\myenv\Scripts\activate
# Install dependencies
pip install -r requirements.txt
```

System Launch

Autonomy Service

```
cd autonomy-service
docker compose up
```

Simulator

```
cd PX4-Autopilot
make px4_sitl gz_x500_gimbal
```

⚠ Open QGroundControl

If additional background needed in simulation run:

```
make px4_sitl gz_x500_baylands
```

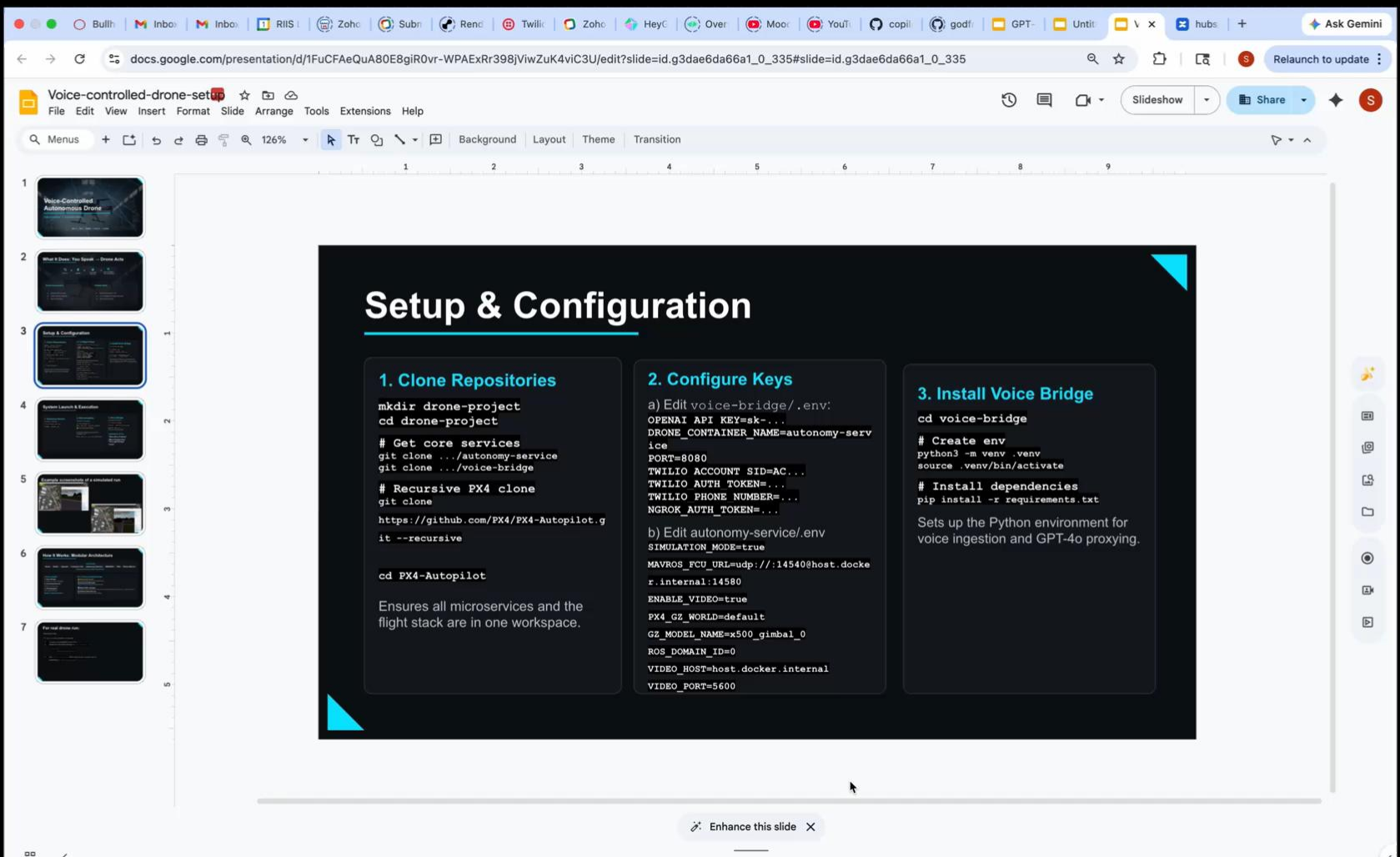
Run Voice Bridge

```
cd voice-bridge
source .venv/bin/activate
python main.py
```

Call your Twilio number

Commands to Try:

- "Take off to 5 meters"
- "Move forward 10m"
- "Turn right 90 deg"
- "Land"


docs.google.com/presentation/d/1FuCFAeQuA80E8giR0vr-WPAExRr398jViwZuK4viC3U/edit?slide=id.g3dae6da66a1_0_335#slide=id.g3dae6da66a1_0_335
Voice-controlled-drone-setup
File Edit View Insert Format Slide Arrange Tools Extensions Help
Slideshow
Share
Menus
126%
Background
Layout
Theme
Transition
Setup & Configuration
1. Clone Repositories
mkdir drone-project
cd drone-project
Get core services
git clone .../autonomy-service
git clone .../voice-bridge
Recursive PX4 clone
git clone
https://github.com/PX4/PX4-Autopilot.git --recursive
cd PX4-Autopilot
Ensures all microservices and the flight stack are in one workspace.
2. Configure Keys
a) Edit voice-bridge/.env:
OPENAI_API_KEY=sk-...
DRONE_CONTAINER_NAME=autonomy-service
PORT=8080
TWILIO_ACCOUNT_SID=AC...
TWILIO_AUTH_TOKEN=...
TWILIO_PHONE_NUMBER=...
NGROK_AUTH_TOKEN=...
b) Edit autonomy-service/.env
SIMULATION_MODE=true
MAVROS_FCU_URL=udp://:14540@host.docker.internal:14580
ENABLE_VIDEO=true
PX4_GZ_WORLD=default
GZ_MODEL_NAME=x500_gimbal_0
ROS_DOMAIN_ID=0
VIDEO_HOST=host.docker.internal
VIDEO_PORT=5600
3. Install Voice Bridge
cd voice-bridge
Create env
python3 -m venv .venv
source .venv/bin/activate
Install dependencies
pip install -r requirements.txt
Sets up the Python environment for voice ingestion and GPT-4o proxying.
Enhance this slide

Troubleshooting

⚠️ Drone won't arm?

Open QGroundControl first to establish link.

⚙️ Slow first PX4 build

This is normal; subsequent builds are fast.

🌐 Ngrok URL changes

Add a permanent auth token to config. If auto tunneling fails, in terminal run - `ngrok http 8000`

🐳 Docker slow first run

Expected behavior while pulling images.

Safety Considerations

- Add command validation layer
- Enforce geofencing
- Require confirmations for risky actions
- Implement emergency stop

Challenges / Limitations

- Latency (voice → AI → execution)
- Reliability of speech recognition
- Safety constraints (AI hallucination risk)
- Network dependency (Twilio + ngrok)

Future Improvements

- → Multi-drone coordination
- → Visual feedback loop (camera + AI)
- → Edge AI (remove cloud dependency)
- → Mobile app / Landline
- → Autonomous mission planning

Resources

- https://cookbook.openai.com/examples/agents_sdk/app_assistant_voice_agents
- https://github.com/openai/openai-realtime-twilio-demo
- https://github.com/wise-vision/ros2_mcp
- https://github.com/godfreynolan/voice-bridge
- https://github.com/godfreynolan/autonomy-service
- https://platform.openai.com/playground
- https://meetup.com/drone-software-meetup-group