🧪🔬 "Mission: Wi-Fiction – The Case of the Invisible Ruler"

Welcome, Agent Engineer.

You've been selected by the **Ministry of Wireless Affairs (MWA)** to solve a secret problem that's baffling scientists and spies alike: **how do you measure distance... without a ruler, tape, laser, camera, GPS, or even yelling "Marco!" and waiting for "Polo!"?**

The answer lies in something you can't see, can't touch, but can definitely abuse: **Wi-Fi signal strength (RSSI)**.

🎯 Your Top-Secret Mission

You're tasked with building a futuristic gadget that measures the distance between two ESP32 microcontrollers using only the signal strength of their Wi-Fi communication. But you're not allowed to use magic or guesswork. Oh no, the MWA insists you use **machine learning**. After all, AI is trending, and funding depends on buzzwords.

To complete this mission, you must build a **5-part system**. Each part is critical, like the limbs of a well-balanced robot octopus.

🛰️ 1. Transmitter ESP32 – The Talkative One

This ESP32 thinks it's a radio DJ. Its only job is to broadcast Wi-Fi UDP packets shouting **"Hello from the transmitter!"** into the wireless void. It does this over and over, every second, like an overly excited morning host. You'll write code to:

- **Turn the ESP32 into a Wi-Fi Access Point**
- **Broadcast messages via UDP on a specific port**

🛠️ File: `transmiter.cpp`
Role: **Be loud. Stay consistent. Never stop talking.**

📡 2. Receiver ESP32 – The Eavesdropper

This ESP32 is the nosy neighbor of the wireless world. It listens to everything the transmitter says and notes how strong the signal is. This strength, the RSSI, will help us understand how far it is from the source.

🛠 File: `reciver.cpp`
Role: **Listen, record RSSI, and maybe judge how weak the signal is.**

📜 3. Data Collection – The Scientist's Notebook

Now it's time to walk the receiver around your lab, room, or wherever you are. At different known distances (e.g., 0.5m, 1m, 2.5m), you collect RSSI values and log them along with the actual distance.

🛠 File: `data_collection.cpp`
Role: **Collect RSSI and distance data so your ML model has something to learn.**

💡 Note: Distance labels are added **manually** as comments in code or inputs during experiments.

🧠 4. Model Training – The Machine Whisperer

Once you've got a pile of numbers, head into Python land. Using TensorFlow and scikit-learn, you'll normalize the data, train a simple neural network to predict distance from RSSI, and export `model.tflite` model small enough to fit on your ESP32's digital brain.

🛠 File: `train_model.py`
Role: **Teach the machine to see the pattern: weaker signal = farther away.**

Bonus: You'll extract the **mean** and **standard deviation** of RSSI to be used later in inference.

🤖 5. On-Device Inference – The Smart ESP32

This is the final form of your creation. You flash the ESP32 with the model. Now, instead of printing raw RSSI, it will predict the actual distance — like a robot guessing how far its crush is without getting too close.

🛠 File: `infrence.cpp`
Role: **Run real-time inference, normalize RSSI, and predict distance. No magic, just math.**

🏆 Final Objective

At the end of this adventure, your ESP32s should:

- Broadcast and receive packets
- Log and analyze signal strengths
- Train a model on real-world data
- Deploy the model on a microcontroller
- Predict distance based on live Wi-Fi signals

Congratulations, Agent Engineer. If you complete this mission, you'll have built a low-cost, ML-powered distance estimation system — no GPS, no LIDAR, no drama.

But remember:
The real ruler... was Wi-Fi all along.