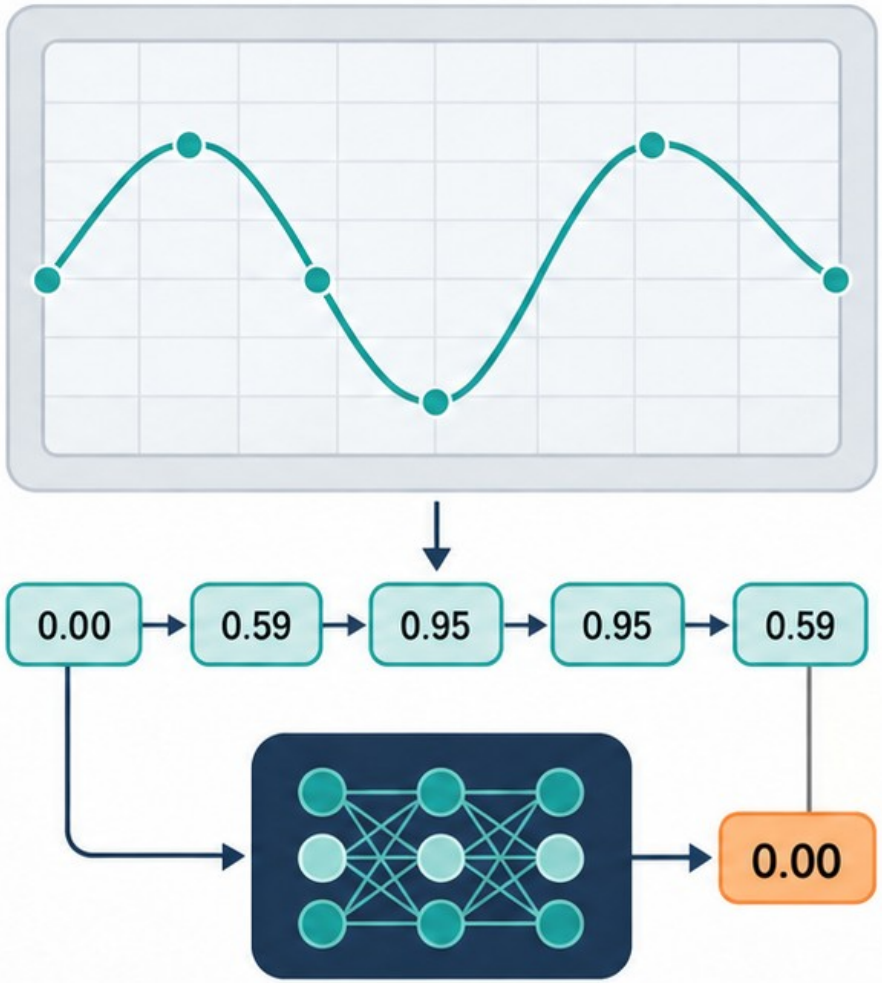
Large Language Models for Measurement & Automation

How an LLM sees a sine wave, and how to put one to work in the lab



Arash Ahmadi
Ph.D. Student, Electrical and Computer Engineering
INQUIRE Lab, The University of Oklahoma

Guest lecture | ECE 4433/5433 Measurement and Automation | Dr. Sarah Sharif

Today's roadmap

1

What an LLM computes

2

The math inside

3

How it sees a measurement

4

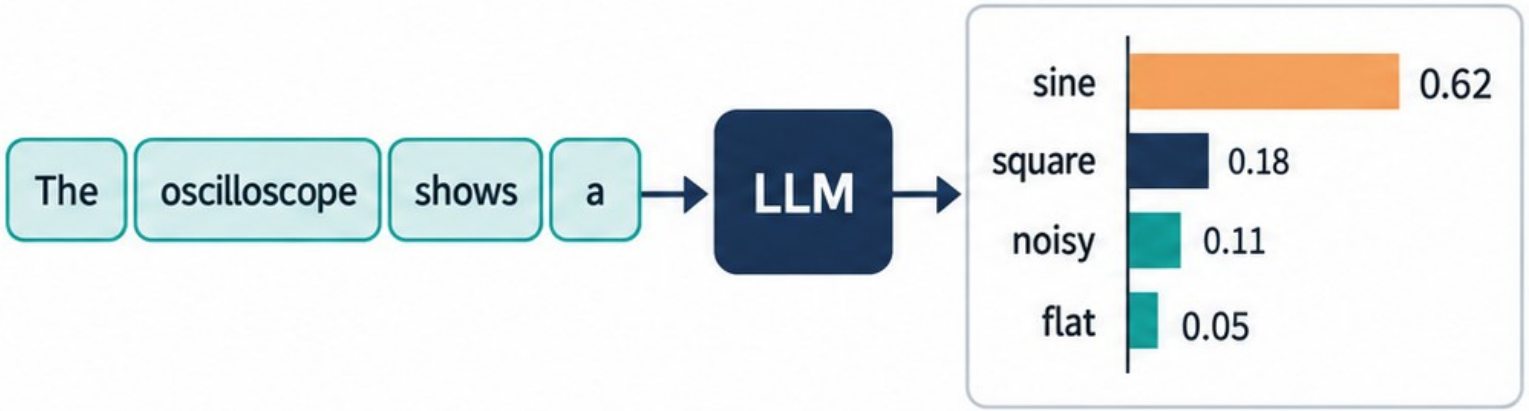
LLMs that run instruments

5

Pitfalls and best practices

Goal: know when to trust an LLM with your data, and how to wire one into a measurement system safely.

An LLM is a next-token predictor



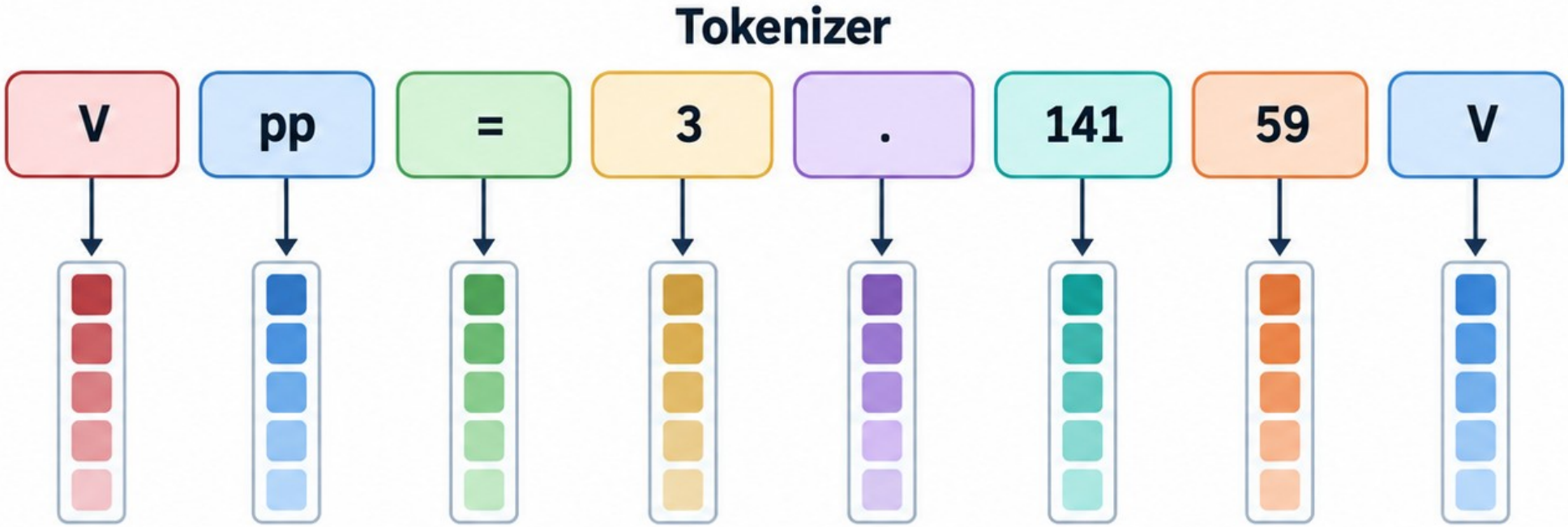
$$P(x_1, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_{<t})$$

$$x_t \sim \text{softmax}(z_t / \tau)$$

One token at a time. Temperature τ controls randomness.

Everything an LLM does, from chat to writing lab code, is this loop repeated.

Step 1: Text becomes tokens, tokens become vectors



Embedding $e_i = W_E[token_i] \in \mathbb{R}^d$

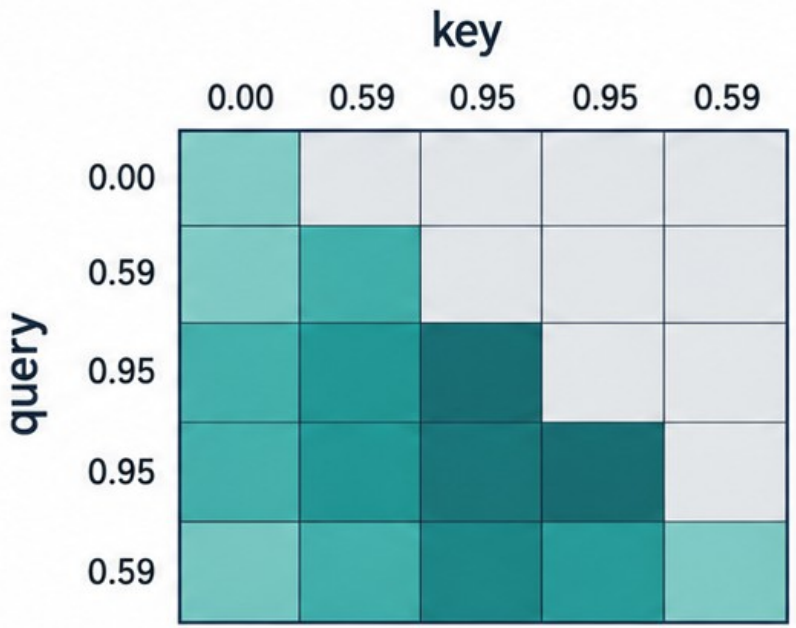
Numbers are split into arbitrary pieces: 3.14159 → '3' '.' '141' '59'. The model never sees the value, only the pieces.

Position is added so order matters:
$h_i = e_i + p_i$

Step 2: Attention, the core equation

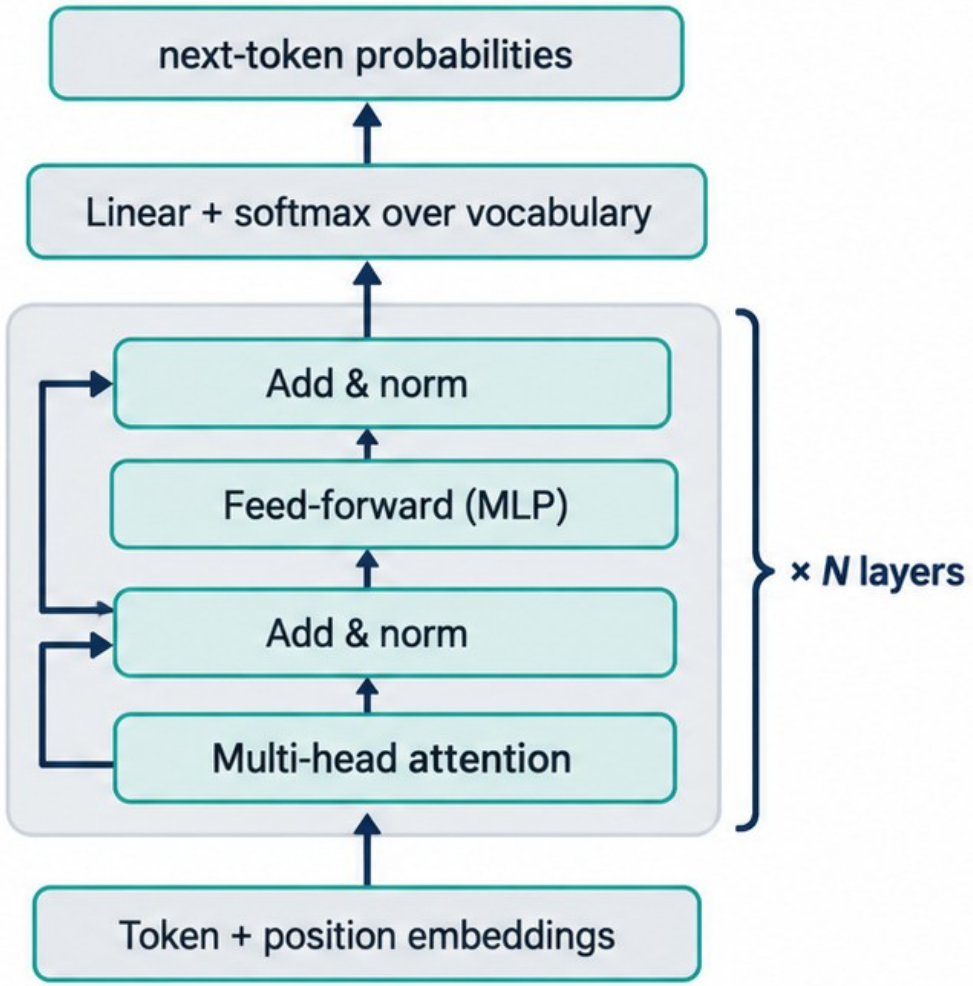
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V$$

$Q = HW_Q$ (what I am looking for) $K = HW_K$ (what I contain) $V = HW_V$ (what I pass on)



Each new sample looks back at earlier samples. The causal mask hides the future.

Step 3: Stack it, then learn it



Training objective

$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log P_\theta(x_t \mid x_{<t})$$

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$$

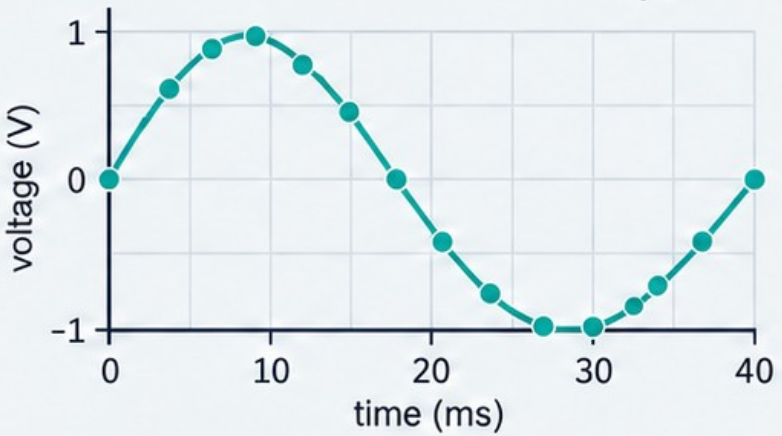
- Cross-entropy on trillions of tokens
- Billions of parameters θ
- Same math whether the text is prose, code, or numbers

How an LLM sees a measurement: the sine wave

What the scope sees

$$v(t) = A\sin(2\pi ft + \phi)$$

sampled at f_s: $v[n] = v\left(\frac{n}{f_s}\right)$



What we send

```
0.00, 0.59, 0.95, 0.95, 0.59,
0.00, -0.59, -0.95, -0.95,
-0.59, 0.00, ...
```

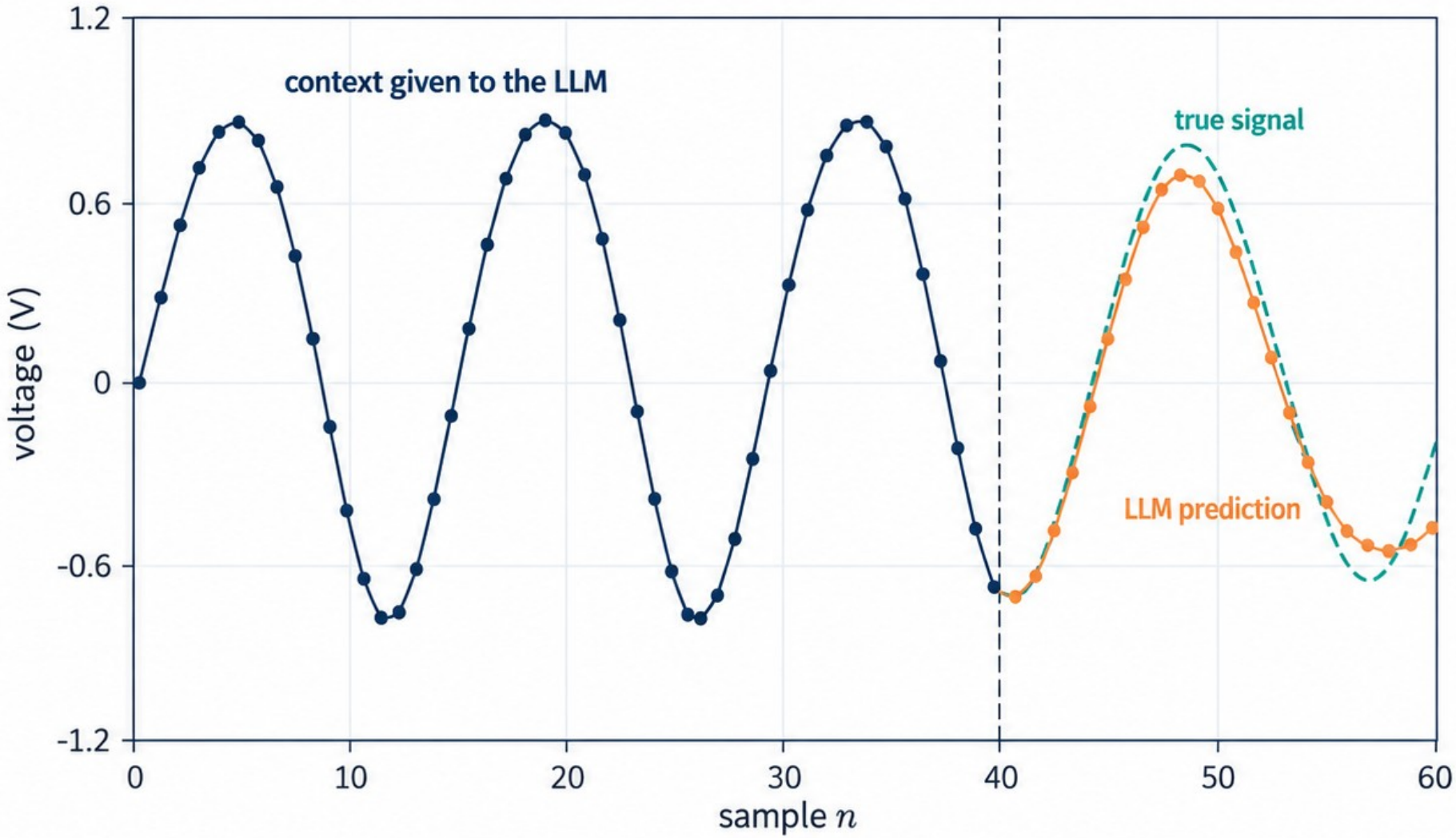
What the LLM actually gets

0 . 00 , 0 . 59 ,
0 . 95 , 0 . 95 ,
0 . 59 , 0 . 00 ,
– 0 . 59 , – 0 . 95 ,
– 0 . 95 , – 0 . 59 ,
– 0 . 00 , 0 . 59 ,
0 . 95 , 0 . 95 , 0 . 59 ,
0 . 00 , – 0 . 59 ,
– 0 . 95 , – 0 . 95 ,
– 0 . 59 , 0 . 00 , ...

a string of tokens, not a waveform

The model has no concept of volts, time, or frequency. It only sees a pattern in text.

In-context learning: the LLM continues the wave



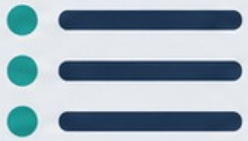
Works for clean, periodic data: the pattern is in the text.

Errors grow with horizon: small phase and amplitude drift.

Noise, drift and long records quickly exceed what text can carry.

Idea from Gruver et al., 'Large Language Models Are Zero-Shot Time Series Forecasters', NeurIPS 2023. Illustration.

Four ways to give measurements to a model



1 Serialize as text

Fixed precision, scaled values, units and sample rate stated.
Simple, but long records get expensive.

2 Summarize first

Send RMS, peak, frequency from an FFT, not raw samples.
Short, reliable, loses detail.

best default for lab work

3 Let it call tools

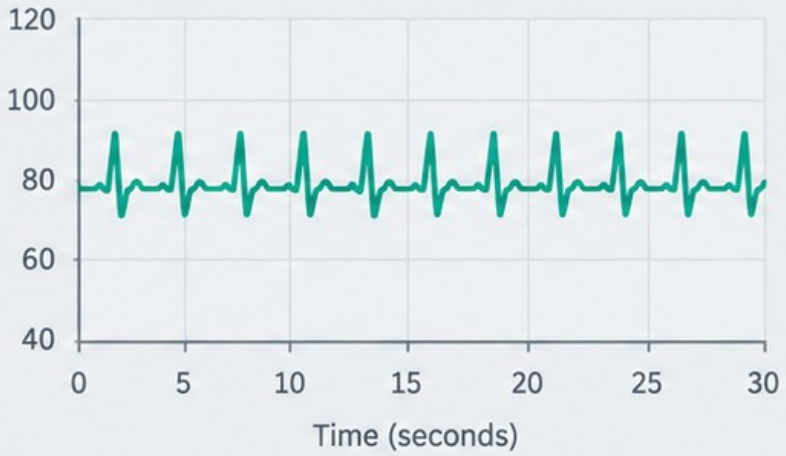
The LLM writes and runs code (NumPy, FFT, curve fit).
Numbers stay exact; the LLM plans and explains.

4 Time-series foundation models

Values are scaled and quantized into bins, then modeled like tokens.
Examples: Chronos (Amazon), TimesFM (Google).

Case study: when the numbers in the prompt disagree

Sensor stream



Heart rate: 80 BPM

LLM

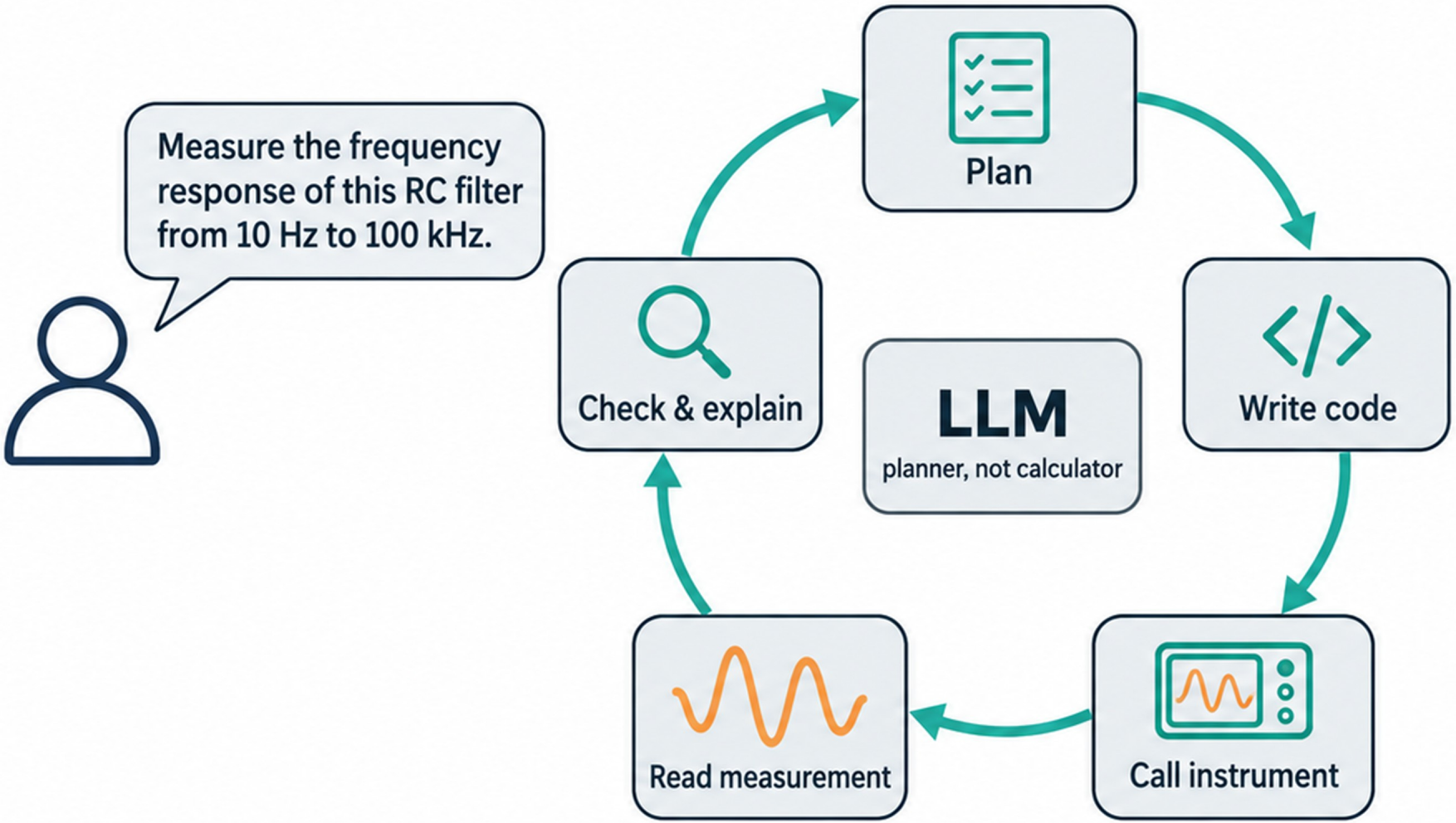
The patient's heart rate is 108 BPM.

What fixed it

- Show the exact prompt the model receives
- Send values as labeled fields with units: heart_rate_bpm: 80
- Verify every number the model states against the source data

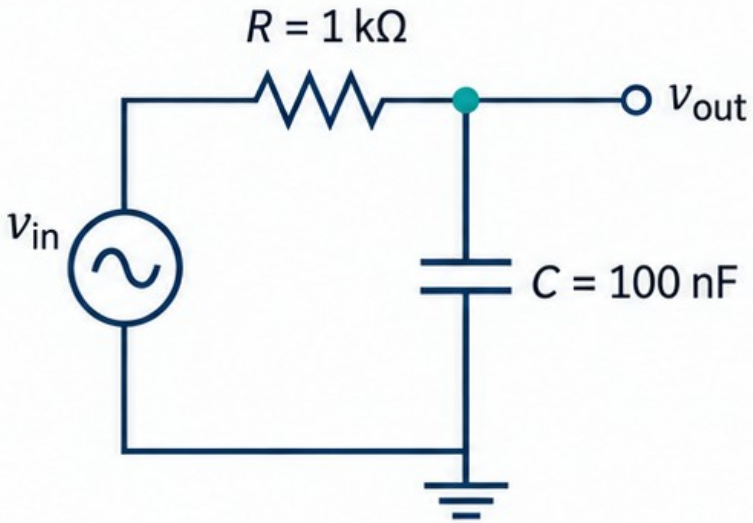
From an edge clinical-assistant prototype in our lab: the model read the sensor data but reported a different value.

LLMs that run instruments: the agent loop



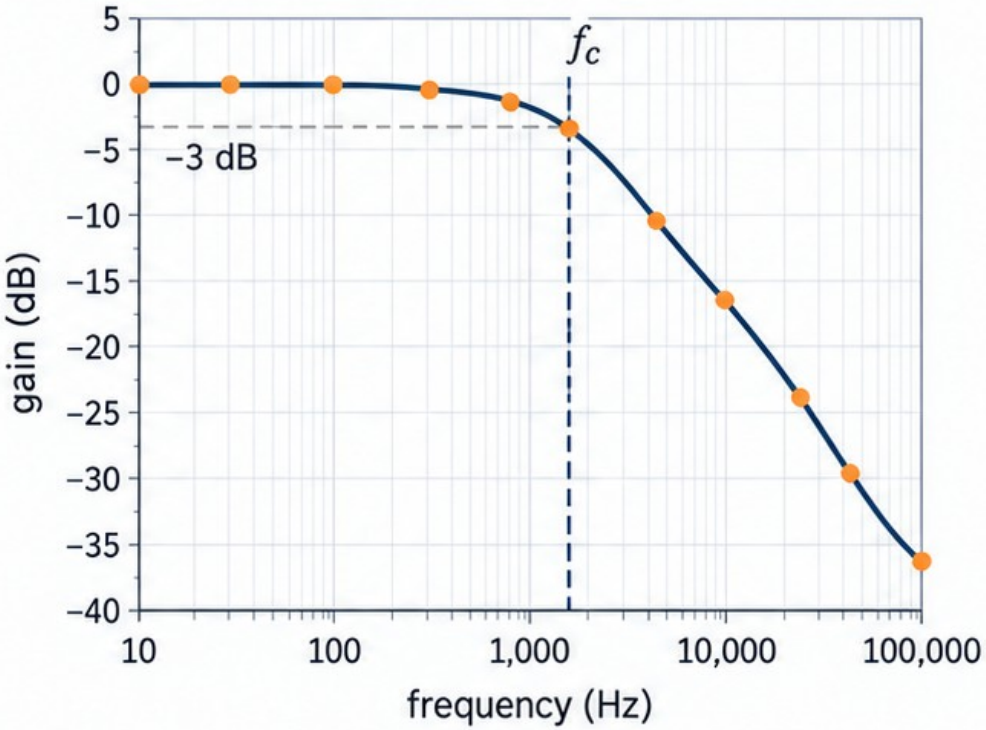

```
import pyvisa
rm = pyvisa.ResourceManager()
fg = rm.open_resource(FG_ADDR)
scope = rm.open_resource(SCOPE_ADDR)
fg.write('FREQ 1000')
vpp = scope.query('MEAS:VPP? CH2')
```

Example: an RC low-pass filter, measured by an agent



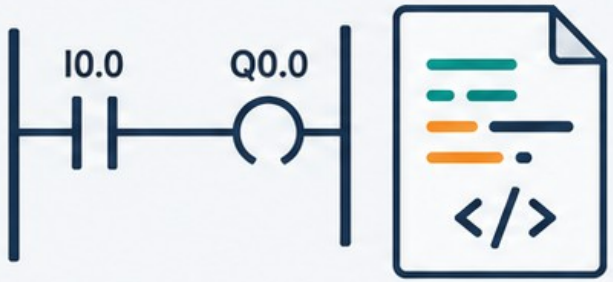
$$H(j\omega) = \frac{1}{1 + j\omega RC}$$

$$f_c = \frac{1}{2\pi RC} \approx 1.59\ \text{kHz}$$



The agent sweeps the generator, reads the scope, fits f_c with code, then explains: **'Measured f_c = 1.6 kHz, within 1% of theory.'**

Where this fits with PLCs, LabVIEW and your bench



Write the automation

- Draft PLC logic from a written sequence of operations, then verify it in simulation
- Explain an existing LabVIEW VI or PLC program rung by rung
- Turn a test plan into a PyVISA or SCPI script

Understand the data

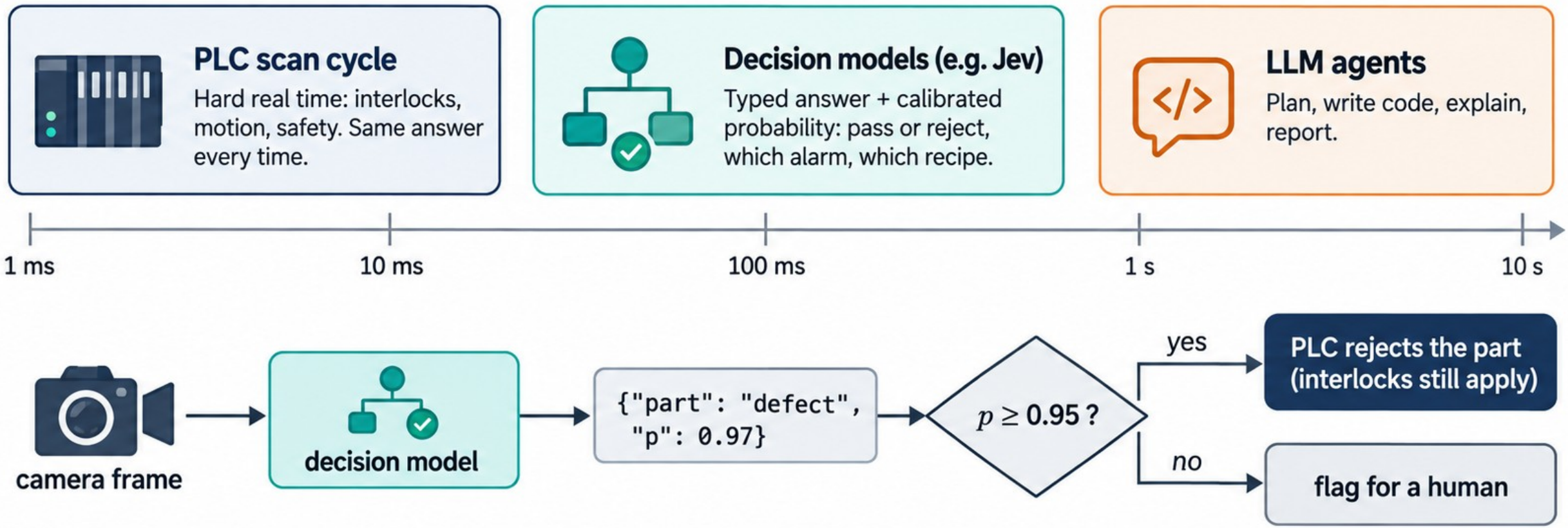
- Summarize long logs and flag anomalies
- Suggest likely causes: grounding, aliasing, loading, a stuck sensor
- Turn results into a lab report draft

Keep control deterministic

- The PLC or LabVIEW loop runs real-time control, never the LLM
- Interlocks and limits live in the controller, not in a prompt
- A human approves any change before it reaches hardware

The LLM writes and explains. The PLC and LabVIEW measure and control.

The middle ground: fast decision models



Treat the model's answer like one more sensor: the PLC still has the final say.

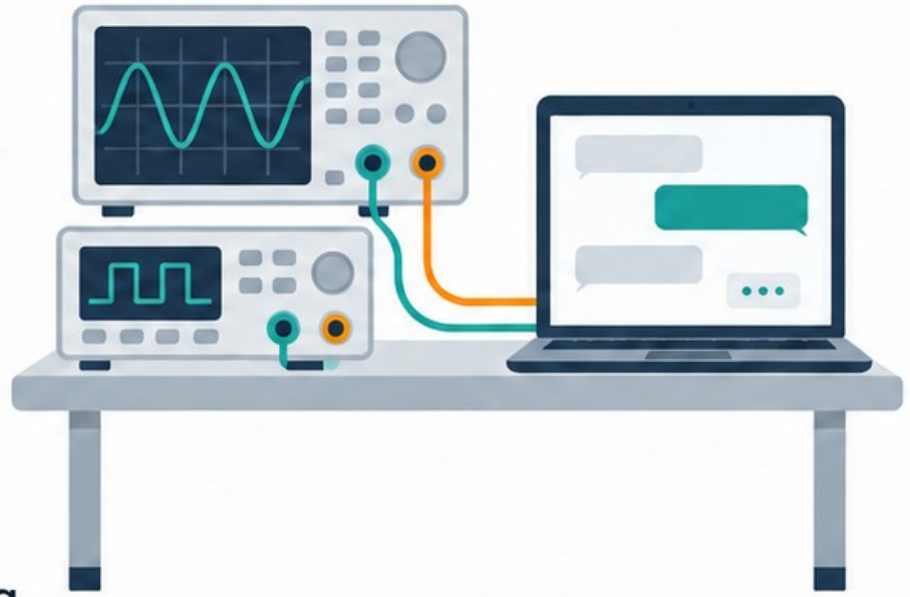
Jev: TypeSafe AI, early access Sept 2026. Speed as reported by the company.

Pitfalls you will actually hit

Hallucinated numbers	States a value that is not in the data. Always verify against the source.
Unit and scale errors	mV vs V, ms vs s, dB vs linear. State units in every field.
Tokenization of digits	3.14159 is split into pieces, so arithmetic in text is unreliable.
Context limits	A 1 MS/s capture is far too long to paste. Summarize or use tools.
Unsafe commands	A wrong setpoint can damage hardware. Allowlists and hard limits in code.

A checklist for using LLMs with measurements

- ☑ Give units, sample rate and precision explicitly
- ☑ Let code do the math: FFT, fits, statistics
- ☑ Ask for code, then run and read it yourself
- ☑ Keep raw data out of the prompt; send summaries
- ☑ Log the exact prompt and the model's answer
- ☑ Cross-check every number against the instrument
- ☑ Enforce instrument limits in software
- ☑ A human approves anything that touches hardware

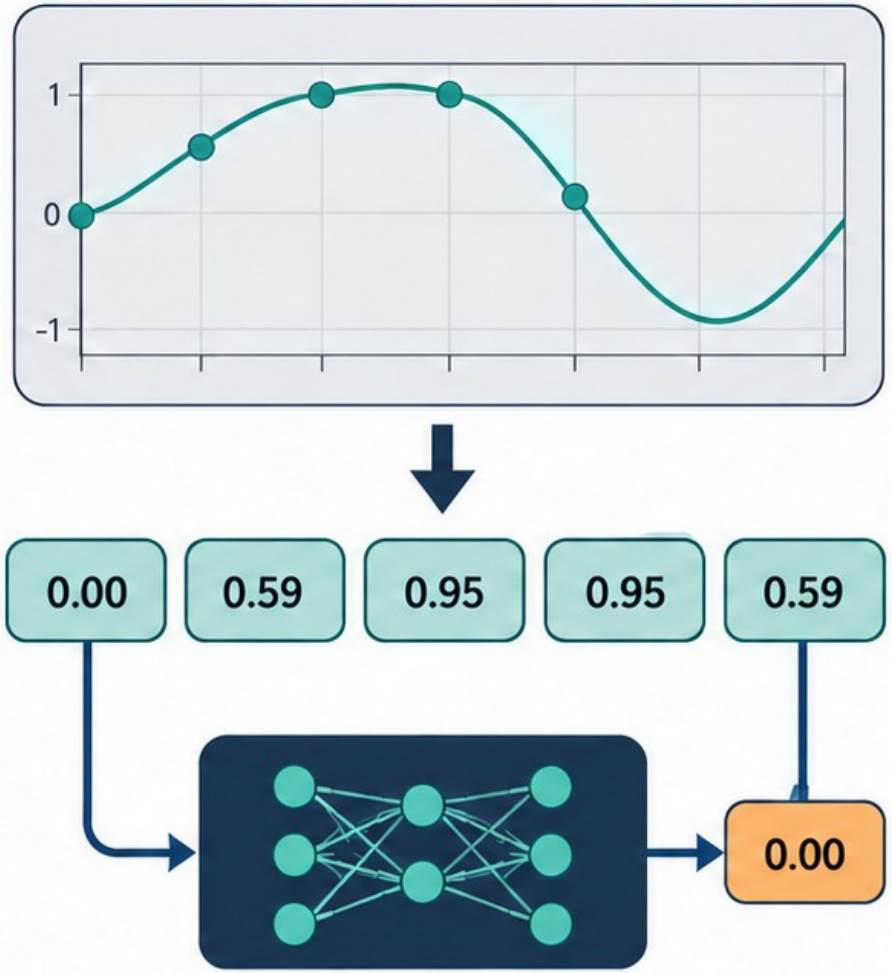


Summary

1 An LLM predicts the next token: $P(x_t \mid x_{<t})$

2 It sees your sine wave as text, not as volts and seconds

3 Best use in the lab: plan, write code, call instruments, explain. Let the tools do the measuring.



Questions?

Arash Ahmadi | arash.ahmadi@ou.edu

INQUIRE Lab, The University of Oklahoma

Join the LLM Engineering Club

Scan to join: inquirelab.ai/llm-club

Hands-on workshops: train and fine-tune LLMs on real GPUs
Free, open to every major, no AI background needed