USC University of Southern California

EE 105: Introduction to Electrical Engineering

Lab 1.1

Asymmetric touch panel

Yasser Khan

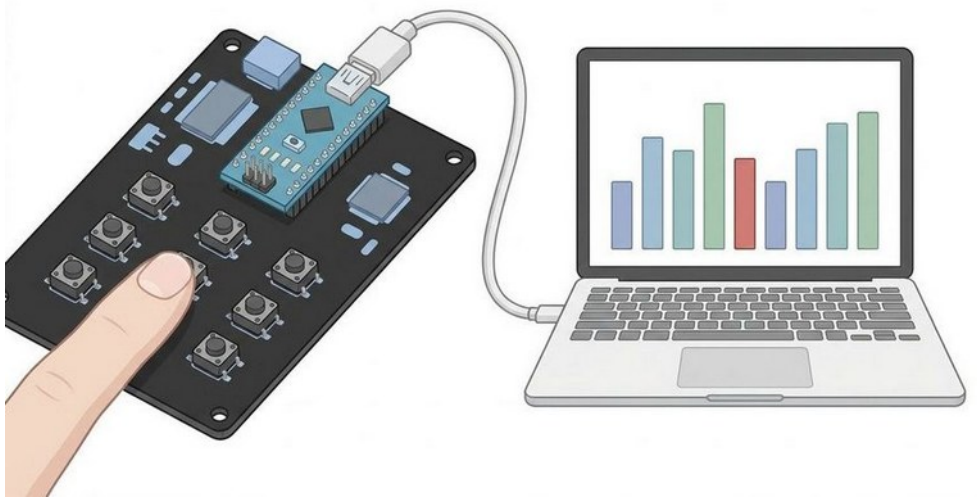
Rehan Kapadia

Today's mission

Turn one voltage into a button name: measure nine buttons, check them with KCL, build a decoder.

By the end of this lab you can:

- **Measure** the nine node voltages of the touch panel
- **Check** them against nodal analysis and KCL
- **Decode** a voltage into a button name, live
- Tell a **floating input** from a real press



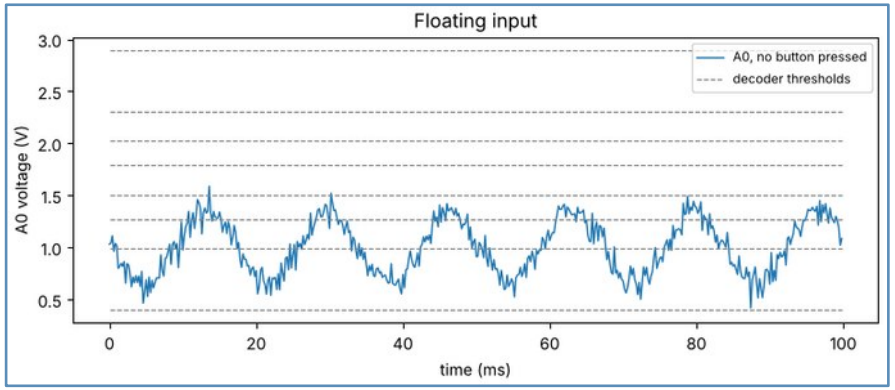
Press a button and its grid node connects to A0.

A	B	C	D	E
Connect **10 min**	Nine buttons **20 min**	Nodal analysis **25 min**	Decoder **25 min**	Floating input **20 min**

Key takeaways

What to remember from this lab, with the numbers that matter.

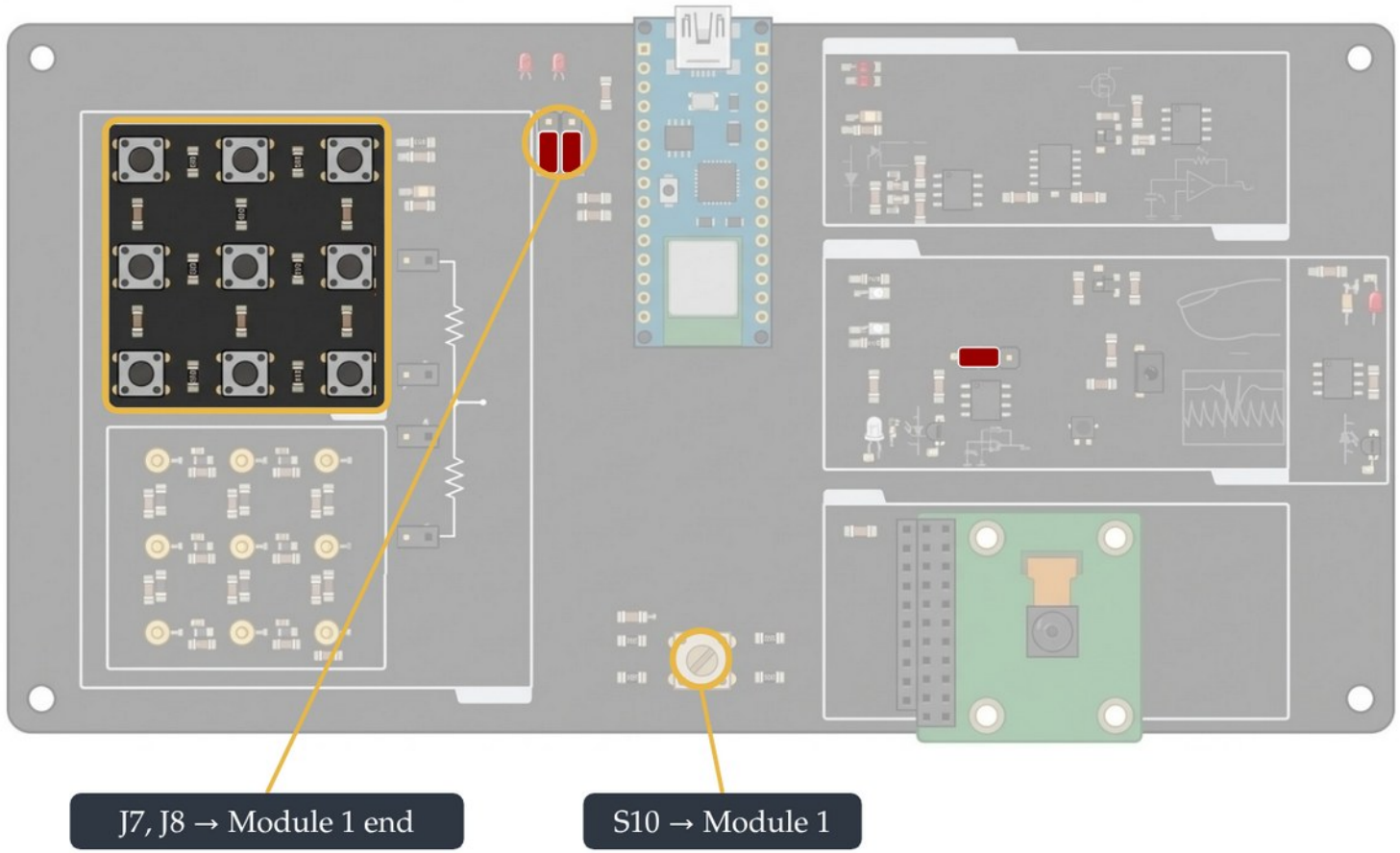
- **Unequal resistors make nine voltages.** 100 Ω down, 50 Ω across; an all-equal grid gives five levels
- **One ADC count is 3.22 mV** (3.3 V / 1024). Most buttons read within about 10 counts of nodal analysis
- **KCL: each node is a weighted average** of its neighbors; measured nodes obey it within about 10 mV
- **Thresholds go halfway between levels.** Closest pair 169 mV (52 counts) apart: 85 mV of margin
- **A floating A0 swings at 60 Hz** by hundreds of mV; a held button stays within about 20 mV



Expected result of E1 on the simulated board: A0 floats and swings at 60 Hz.

SETUP

Set the board before you plug it in: one selector, two jumper caps, an empty header.



1. **S10 → Module 1**
   Turn the selector to Module 1, the circuits block.

2. **J7, J8 → Module 1 end**
   Both caps on pins 2-3, just left of the Arduino. A0 then reads the touch panel.

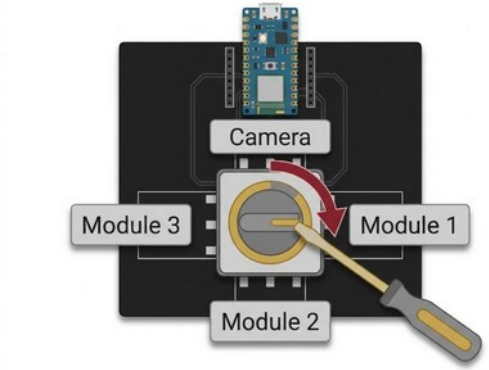
3. **Plugged in: nothing**
   Leave the divider header J1 to J4 empty.

4. **USB data cable**
   Use the kit cable. A charge-only cable cannot connect.

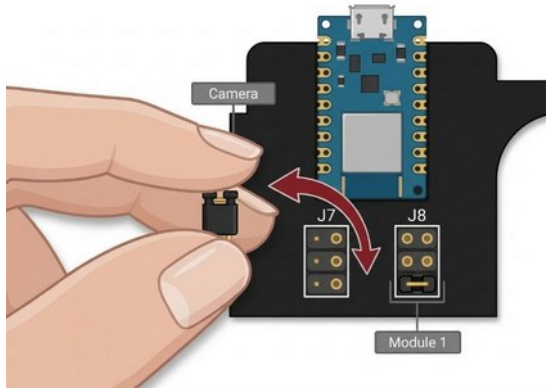
Part A: Connect

Set the board, plug it in, run the first cell.

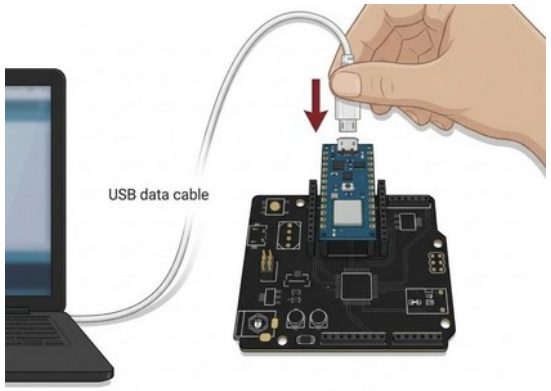
PART A **Goal: A1 · DO** connect to the board and check the selector.



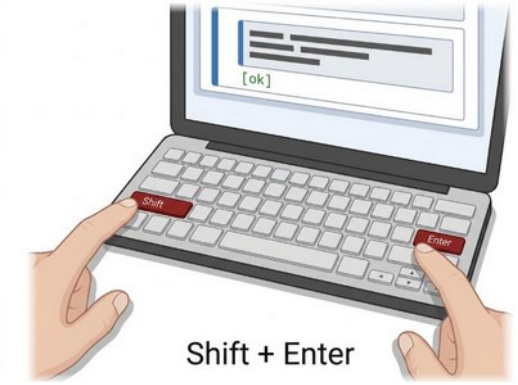
1 Turn S10 to Module 1



2 J7, J8 (3-pin headers): caps on pins 2-3, the Module 1 end



3 Plug the USB data cable into the Nano



4 Run the first cell with Shift+Enter

You should see: both lines start with **[ok]**. A **[!!]** line about the selector is only a warning: if S10 points at Module 1, carry on.

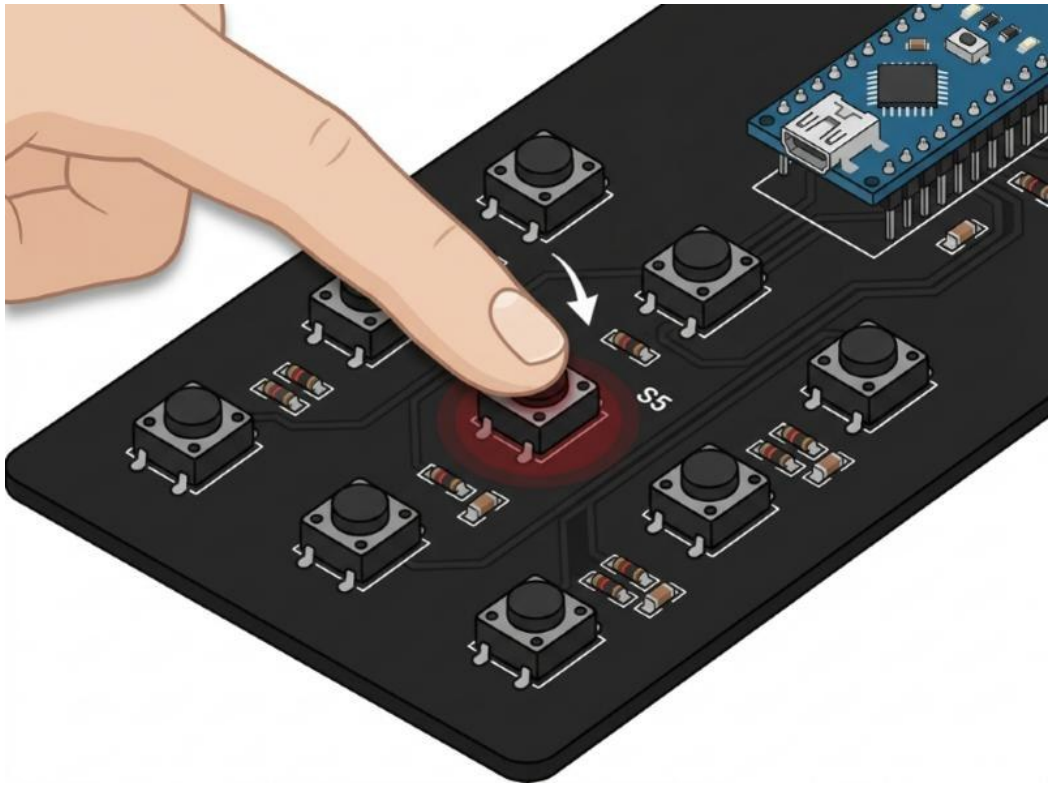
Part B: Measure the nine buttons

Hold a button, press Enter, keep holding: the cell averages 200 readings of A0 (40 ms).

PART B

Goal: measure the voltage of each of the nine buttons.

- Predict first: which button reads 3.3 V, which 0 V, which half of 3.3 V?
- **B1 · DO Hold the button down, then press Enter** in the box
- Keep holding until its voltage prints, then go on to the next button
- **B2 · DO** Only if needed: redo one button with **measure("S4")**



Hold the button until its voltage prints.

You should see: S1 about **3.30 V**, S9 about **0 V**, S5 about **1.65 V**; the others **0.80 to 2.50 V**, each one different.

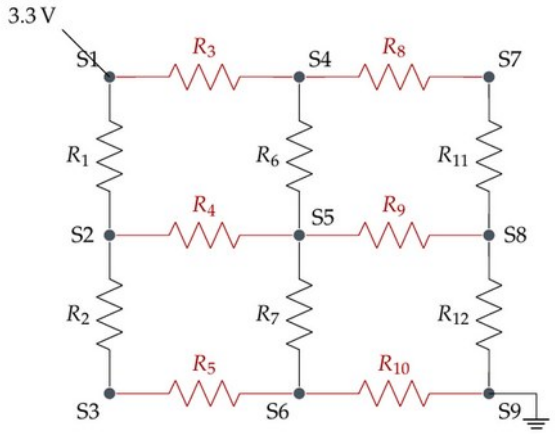
Part C: Compare with nodal analysis

KCL makes every inner node voltage a conductance-weighted average of its neighbors.

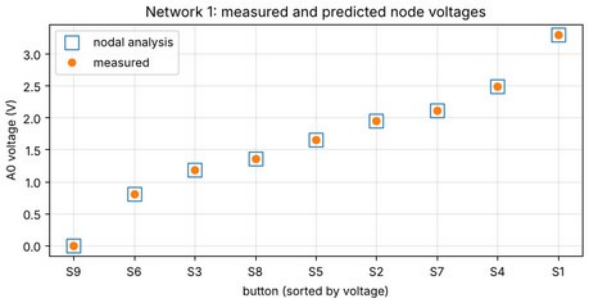
PART C

Goal: check your measurements with KCL, by hand and in code.

- **C1 · ANSWER** KCL at S2 with S1 = 3.3 V, S3 = 1.185 V and S5 = 1.650 V
- **C2 · EDIT kcl_voltage(node, V)** from the resistor dictionary **R**
- **C3 · RUN** The table, saved as **touch_panel_table.csv**



Vertical 100 Ω, horizontal (red) 50 Ω



Expected C3: each dot sits on its square

You should see: C1 gives **S2 = 1.946 V**. KCL from your measured neighbors lands within about **10 mV** of each measured node.

Check (Checkpoint 1): every button within a few tens of mV of nodal analysis, and the C2 predictions close to your measurements.

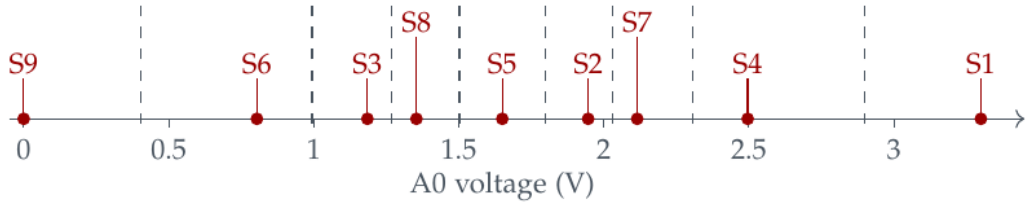
Part D: A button decoder

Pick the nearest level: a threshold halfway between each pair of neighboring levels.

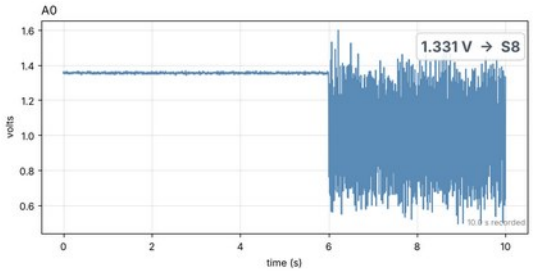
PART D

Goal: turn a voltage into a button name and run it live.

- **D1 · EDIT** Eight thresholds halfway between neighbors; **decode(v)** with **np.searchsorted**
- **D2 · RUN** Draw your levels and thresholds
- **D3 · DO** Live decoder, 10 s: press buttons, watch the title **after you let go**
- **D4 · DO** Optional: the firmware's own decoder, **circuits**



Levels and thresholds from the textbook values; yours come from your measurements.



Expected D3: flat while held; after you let go, A0 swings and the title still names a button.

You should see: thresholds near **0.40, 0.99, 1.27, 1.50, 1.80, 2.03, 2.31, 2.90 V**. S3 and S8 are only **169 mV** apart: a margin of about **85 mV**.

Check (Checkpoint 2): the live decoder names each of the nine buttons correctly.

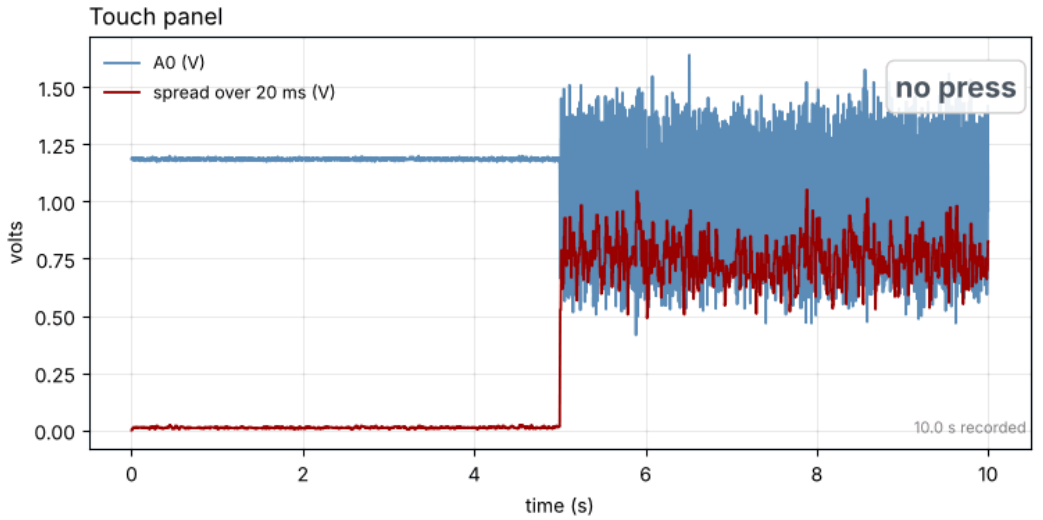
Part E: The floating input

With no button pressed, A0 floats and picks up 60 Hz from the room wiring.

PART E

Goal: report "no press" when the last 20 ms of readings spread too far.

- **E1 · RUN** Hands off: 0.1 s of A0; what does decode say?
- **E2 · DO** The 20 ms spread with S5 held, then with nothing pressed
- **E3 · EDIT SPREAD_LIMIT** between them; **decode_steady**
- **E4 · DO** The improved decoder live; press and release buttons



Expected E4: the 20 ms spread (red) jumps when you let go; the title says no press.

You should see: held: a spread **under about 20 mV**; floating: **hundreds of mV**. The title says **no press** whenever your hands are off.

Check (Checkpoint 3): the improved decoder names every button and says no press when nothing is pressed.

Stuck?

Check these first. If the fix does not help, raise your hand.

A [!!] line about the selector

Only a warning. Turn S10 to Module 1 and carry on.

Every button reads a random value

Caps of J7 and J8 on the Module 1 end (pins 2-3), S10 on Module 1.

The cell waits and nothing happens

It waits for you: hold the button, click the box under the cell, press Enter.

One button reads too high or low

You let go too early. Redo it in B2 with measure("S4") and its name.

The decoder names a button nobody pressed

That is the floating input. Part E fixes it with the 20 ms spread.

Two buttons read like S5

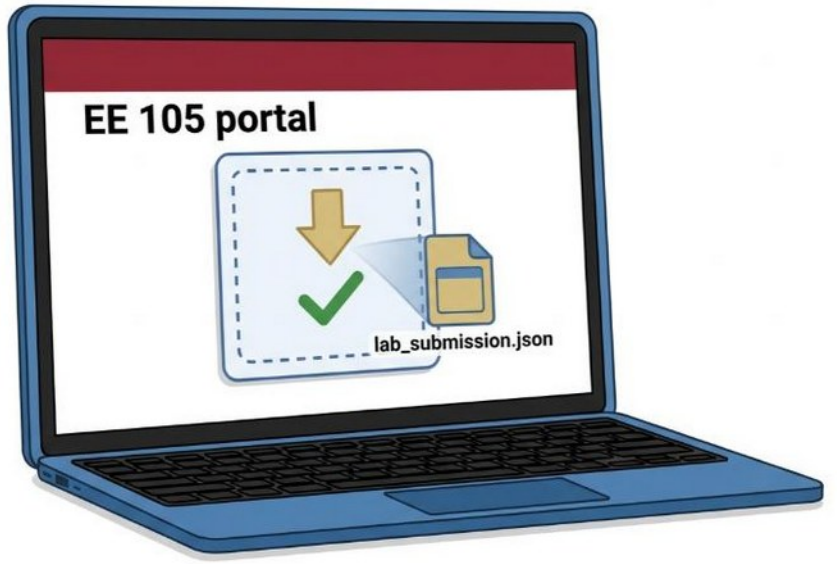
S2 and S8 together give 1.65 V by symmetry. Press one at a time.

Submit your results

The last cell saves your results in one file. Upload it on the course portal.

`lab1.1_submission.json`

1. **Save · RUN** writes **lab1.1_submission.json** next to the notebook
2. It holds your measurements, a sample of the raw readings and your decoder settings
3. Log in at **ee105.org/portal** with your student ID and portal PIN; open Lab 1.1
4. Upload the file. It is graded automatically; you can upload again after a fix



ee105.org/portal

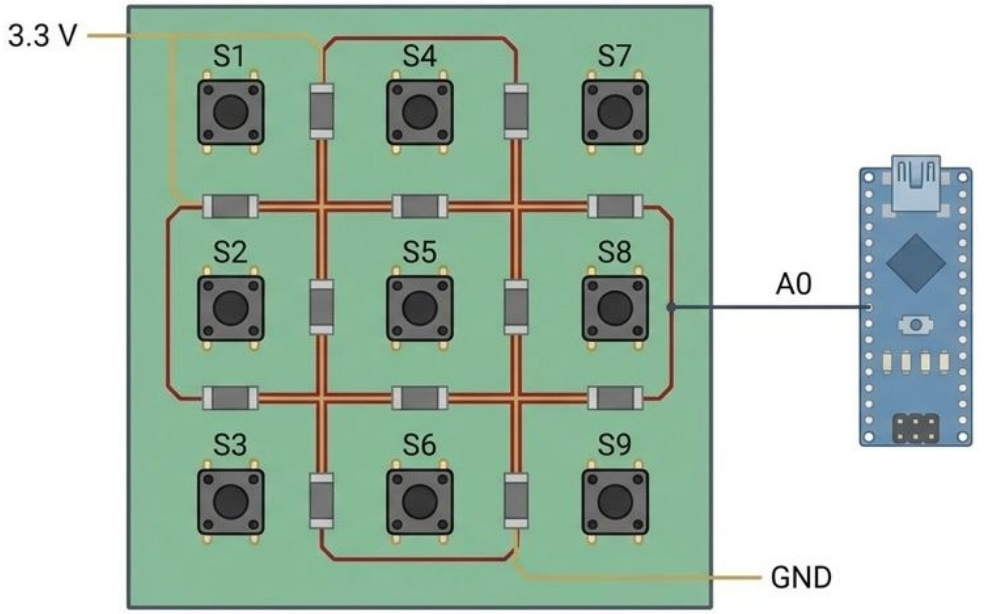
Check: the portal shows your grade. Redid a measurement? Run Save again and upload again. A simulated run earns no credit.

Lab complete

You turned a grid of resistors into a working touch panel controller.

You built:

- Nine measured node voltages that match nodal analysis
- A KCL check of your own measurements
- A nearest-level button decoder, running live
- A spread test that ignores a floating input



Network 1: nine buttons, twelve resistors, one analog pin.

Did you know?

A resistive panel cannot report two touches: S2 and S8 together read like S5. That is one reason phones use capacitive screens (Chapter 1).

Next

Lab 1.2, circuit analysis with Python: solve this grid exactly with SymPy and NumPy, then check it with a multimeter.