Multi-Range Conductivity Meter for Fruit and Vegetable Tissue

.

Anatolie Jentimir Umass Lowell / ML Courses

October 23, 2025

Abstract

This document describes a complete, reproducible build of a low-cost, conductivity-based measurement system for fresh produce. The instrument uses two stainless electrodes inserted into fruit tissue and a multi-range resistive voltage divider driven from the Arduino GIGA R1 at 3.3 V. The system auto-selects one of four reference resistors (1 kΩ, 10 kΩ, 400 kΩ, 2 MΩ) to maximize ADC resolution, computes the sample resistance R_s, and reports conductivity σ in S/cm using a geometry factor (cell constant) K. The project includes theory, hardware, firmware, calibration, data logging, and a scientific-style template suitable for course labs and machine learning projects. All firmware and diagrams are provided.

1 General Idea

Electrical conductivity (EC) in aqueous media correlates with the concentration of dissolved ions. Fresh produce contains ionic species (e.g., K^+, Na^+, Cl^-, NO_3^-). By inserting two electrodes into the fruit and applying a known excitation through a reference resistor, we can infer the fruit tissue's bulk resistance and estimate an *apparent* conductivity. While this is not a selective nitrate measurement, changes in EC can serve as a proxy for overall ionic strength and are useful for comparative studies across samples, varieties, and storage conditions.

2 Theory and Formulas

2.1 Voltage Divider

The circuit is a simple divider where V_s is measured at the midpoint:

$$V_s = V_{\text{ref}} \cdot \frac{R_s}{R_s + R_{\text{ref}}}. \tag{1}$$

Solving for the unknown sample resistance R_s,

$$R_s = R_{\text{ref}} \cdot \frac{V_s}{V_{\text{ref}} - V_s}. \tag{2}$$

2.2 Conductance and Conductivity

Conductance G (in siemens) is the inverse of resistance: $G = 1/R_s$. To convert to a geometry-normalized conductivity (apparent) in S/cm, we introduce the cell constant K in cm^{-1}:

$$\sigma\ [\text{S/cm}] = K \cdot G = \frac{K}{R_s}. \tag{3}$$

The cell constant K depends on electrode spacing and effective area along the current path. For needle-type probes in fruit, K is best determined empirically by calibration.

2.3 Auto-Ranging Heuristic

To maximize ADC resolution, we select the range where V_s is closest to mid-scale ($V_{\text{ref}}/2 \approx 1.65\,\text{V}$ for 3.3 V systems). We define the score

$$\text{score} = \left| V_s - \frac{V_{\text{ref}}}{2} \right| \tag{4}$$

and choose the range with minimum score (rejecting open/short detections).

3 Hardware

3.1 Bill of Materials

- Arduino GIGA R1 (3.3 V analog domain).
- Electrodes: two stainless needles or food-safe metal probes, fixed in a spacer jig (constant spacing & depth).
- Resistors: $1\,\mathrm{k\Omega}$, $10\,\mathrm{k\Omega}$, $400\,\mathrm{k\Omega}$, $2\,\mathrm{M\Omega}$ (1/4 W or lower).
- Capacitor: 10 nF (code "103") from A0 to GND (stabilizes high-R readings).

3.2 Electrical Diagram

Figure 1 shows the simplified schematic. Only one digital pin (D2–D5) is driven HIGH at a time; the others remain as inputs (high-Z). The active pin sources 3.3 V through its reference resistor to the common A0 node and electrode.

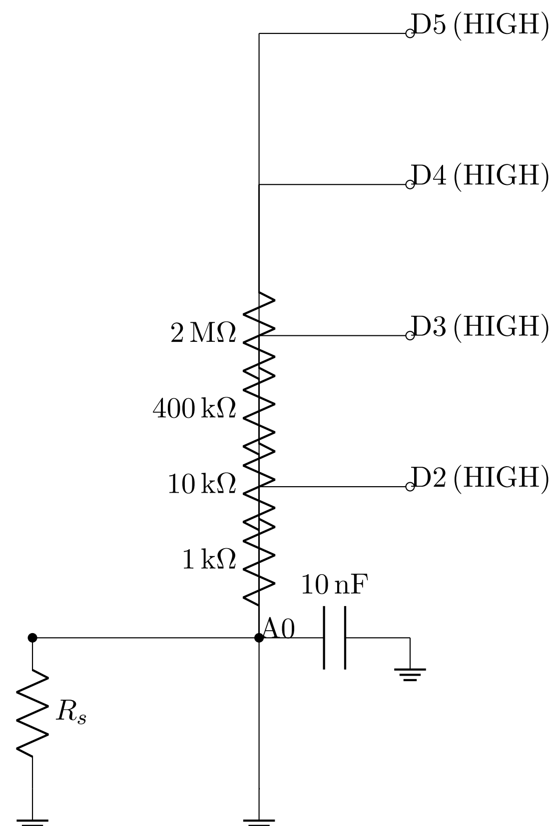

Figure 1: Simplified measurement cell and auto-ranging network. Only one of D2–D5 is driven HIGH at any time; others are inputs (high-Z). The second electrode is tied to GND.

3.3 RC Settling Considerations

The A0 node sees R_{ref} in series with a sample path and a shunt capacitor $C = 10\,\mathrm{nF}$. Approximate settling time constant $\tau \approx R_{\mathrm{ref}}C$. For 2 MΩ, $\tau \approx 20\,\mathrm{ms}$; five time constants ($\approx 100\,\mathrm{ms}$) produce $< 1\%$ residual error. The firmware adds extra delay on high-R ranges and averages multiple samples after one throwaway reading.

4 Calibration

4.1 Cell Constant (K)

With the fixed-spacing probe, we choose K such that a typical apple reads $\sigma \approx 1.0\,\text{mS/cm} = 0.010\,\text{S/cm}$. For a representative $R_s \approx 97\,\text{k}\Omega$:

$$K = \sigma \cdot R_s \approx 0.010 \times 97{,}000 \approx 970\ \text{cm}^{-1}. \tag{5}$$

This is a pragmatic calibration suitable for comparative studies. For higher rigor, measure a standard solution (e.g., 0.9 % saline $\approx 0.015\,\text{S/cm}$ at room temperature) using the same probe geometry, then set $K = \sigma_{\text{ref}} R_{\text{ref}}$.

4.2 Micro-Tuning Rule

If measured σ is σ_m but the target is σ_t, update:

$$K_{\text{new}} = K_{\text{old}} \cdot \frac{\sigma_t}{\sigma_m}. \tag{6}$$

5 Indicative Conductivity Table for Fruits (Educational)

These values are indicative for classroom use, not lab-certified; real readings vary by variety, ripeness, temperature, and probe geometry.

Fruit (room temp)	Typical R_s with wide spacing (kΩ)	Apparent σ (mS/cm)
Apple	80–120	0.8–1.2
Banana	40–90	1.0–2.0
Cucumber	10–40	2.0–6.0
Tomato	15–50	1.5–5.0
Pear	70–130	0.7–1.3
Grape	20–60	1.5–4.0
Potato	50–200	0.5–2.0

Table 1: Indicative classroom ranges for apparent conductivity using the described probe geometry. Use for relative comparisons; re-calibrate K if geometry changes.

6 Firmware Logic

6.1 Core Steps

1. For each range (D2..D5): set only that pin HIGH; others INPUT (high-Z).
2. Discard the first ADC reading; then average N samples with min/max rejection.
3. Compute V_s, test for open/short guards, compute R_s via Eq. 2.
4. Score each range by proximity to mid-scale ($\approx 1.65\,\text{V}$).
5. Choose the best range; compute G and $\sigma = K/R_s$.
6. Present data on the LCD; print to Serial.
7. Optionally output numeric-only CSV series compatible with Arduino Serial Plotter.

6.2 Safety and Limits

Operate strictly at 3.3 V. Do not ingest sampled fruit portions. Clean probes between measurements. EC is non-specific to nitrate; use only as an indirect, comparative indicator.

7 Complete Firmware (Arduino GIGA R1)

The following sketch integrates auto-ranging, display, and a Serial Plotter mode toggled by pressing ‘p‘.

Listing 1: Arduino firmware for the conductivity meter (GIGA R1).

```


// ==== Globals ====
const int    AIN       = A0;
const float  VREF      = 3.3f;
const float  ADC_MAX   = 4095.0f;
const float  CAP_F     = 10e-9f;       // 10 nF
float        K_cell    = 970.0f;       // apple calibration

RangeCfg ranges[] = {
  {2,         1000.0f, "LOW",     2},
  {3,        10000.0f, "MID",     3},
  {4,       400000.0f, "HIGH",    5},
  {5,      2000000.0f, "ULTRA", 30}
};


int readADC_avg(int N = 20) {
  (void)analogRead(AIN);
  long acc = 0; int mn = 4095, mx = 0;
  for (int i = 0; i < N; ++i) {
    int v = analogRead(AIN);
    acc += v;
    if (v < mn) mn = v;
    if (v > mx) mx = v;
    delay(2);
  }
  acc -= mn; acc -= mx;
  return (int)(acc / (float)(N - 2));
}

RangeResult measure(const RangeCfg& rg) {
  enableOnly(rg);
  RangeResult out;
  out.raw = readADC_avg();
  out.Vs  = (out.raw / ADC_MAX) * VREF;

  const float OPEN_GUARD  = 0.005f;
  const float SHORT_GUARD = 0.005f;

  if (out.Vs >= (VREF - OPEN_GUARD)) {
    out.Rs = INFINITY;  out.ok = false;
```

```
  } else if (out.Vs <= SHORT_GUARD) {
    out.Rs = 0.0f;       out.ok = true;
  } else {
    out.Rs = rg.Rref * (out.Vs / (VREF - out.Vs));
    out.ok = true;
  }
  out.score = out.ok ? fabsf(out.Vs - 1.65f) : 1e6f;
  return out;
}


  snprintf(buf, sizeof(buf), "Time: %lu ms", t_ms);
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "Mode: %s", plotterMode ? "PLOTTER" : "TEXT");
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "Range: %s", range);
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "RAW: %d", raw);
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "Vs: %.5f V", Vs_V);
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "Rref: %.0f ohm", Rref_ohm);
  display.setCursor(x,y); display.print(buf); y+=dy;

  if (Rs_ohm >= 1e9) snprintf(buf, sizeof(buf), "Rs: OPEN");
  else               snprintf(buf, sizeof(buf), "Rs: %.2f ohm", Rs_ohm);
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "G: %.6f mS", G_mS);
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "Sigma: %.6f S/cm", sigma_S_per_cm);
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "K_cell: %.1f", K_cell_in);
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "Score: %.6f", score);
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "Open:%d Short:%d", open_flag ? 1:0, short_flag ? 1:0);
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "Cap_F: %.0f nF", cap_F * 1e9);
  display.setCursor(x,y); display.print(buf); y+=dy;

  snprintf(buf, sizeof(buf), "Settle: %d ms", extraSettle_ms);
```

```
  display.setCursor(x,y); display.print(buf);
}



  // Measure all ranges and pick the best
  RangeResult bestRes;
  const RangeCfg* bestCfg = NULL;
  float bestScore = 1e9f;

  for (size_t i = 0; i < sizeof(ranges)/sizeof(ranges[0]); ++i) {
    RangeResult rr = measure(ranges[i]);
    if (rr.score < bestScore) {
      bestScore = rr.score;
      bestRes = rr;
      bestCfg = &ranges[i];
    }
  }

  // Derived values
  float G_mS  = isinf(bestRes.Rs) ? 0.0f : (1000.0f / fmax(bestRes.Rs, 1e-12f));
  float sigma = isinf(bestRes.Rs) ? 0.0f : (K_cell   / fmax(bestRes.Rs, 1e-12f));
  bool openFlag  = (!bestRes.ok && isinf(bestRes.Rs));
  bool shortFlag = ( bestRes.ok && (bestRes.Rs == 0.0f));

  if (plotterMode) {
    printPlotterHeaderIfNeeded();
    printPlotterRow(bestRes.Vs, bestRes.Rs, G_mS, sigma);
  } else {
    Serial.print("["); Serial.print(bestCfg ? bestCfg->name : "NA"); Serial.print("]  ");
    Serial.print("Vs="); Serial.print(bestRes.Vs,4); Serial.print(" V  ");
    Serial.print("R=");  if (isinf(bestRes.Rs)) Serial.print("OPEN");
    else                 Serial.print(bestRes.Rs,1);
    Serial.print(" ohm  G="); Serial.print(G_mS,4); Serial.print(" mS  ");
    Serial.print("sigma="); Serial.println(sigma,6);

    Serial.print("CSV,");
    Serial.print(bestRes.Vs,4); Serial.print(",");
    Serial.print(bestCfg ? bestCfg->name : "NA"); Serial.print(",");
    if (isinf(bestRes.Rs)) Serial.print(1e7); else Serial.print(bestRes.Rs,1);
    Serial.print(",");
    Serial.println(sigma,6);
  }

  delay(160);
}
```

8 Data Analysis Notes for ML Courses

- Log the numeric CSV for each sample site; compute medians to suppress outliers.
- Potential features: σ, R_s, V_s, active range, ambient temperature, fruit variety, position index.
- Targets (if available): reference conductivity (calibration solution), or lab nitrate values.
- Suggested models: linear regression (with log-transforms), ridge/LASSO, tree-based regressors; or clustering for unlabeled exploration.

9 Safety

Operate at low voltage only. Clean and dry probes between measurements. Do not eat tested portions. Store the device away from moisture when not in use.

10 Future Improvements

1. 3D-printed probe with fixed spacing to reduce error in L/A.
2. Temperature compensation and display logging.
3. Wireless data transfer via Wi-Fi to a web dashboard.
4. On-device regression using TinyML.
5. Multi-frequency excitation for impedance spectroscopy.

11 References

- Arduino GIGA R1 WiFi documentation – `https://docs.arduino.cc/hardware/giga-r1-wifi`
- GIGA Display Shield – `https://docs.arduino.cc/hardware/giga-display-shield`
- A. Jentimir et al., "Bio-Conductivity Analysis of Fruit Tissue for Machine Learning," Umass Lowell notes, 2025.