DRAWING NO.	REVISION	MODULE	LANGUAGE
BP-001	A	Microcontroller Fundamentals	English

Architecture of Autonomous Sensor Arrays

Wiring, powering, and coding a multi-sensor system on a single microcontroller — the foundational skill behind almost every hardware-based science fair project.

SKILL LEVEL: BEGINNER–INTERMEDIATE | BUILD TIME: 2–3 HOURS | EST. COST: $18–28

01 Objective

By the end of this blueprint, you will have a single Arduino Uno simultaneously reading three different sensor types — distance, temperature/humidity, and light — and printing structured, labeled readings to your computer. More importantly, you'll understand *why* the code is organized the way it is: every sensor gets its own read function, so the system can scale from 3 sensors to 30 without becoming unmanageable. This "one function per sensor" pattern is the actual architecture referenced in the title, and it's the same pattern used in real environmental monitoring and robotics systems.

02 Prerequisites

- Arduino IDE installed on your computer (free, from arduino.cc)
- Comfort with the idea of digital pins (on/off) vs. analog pins (a range of values) — if this is new, spend 15 minutes on Arduino's own "Blink" and "AnalogReadSerial" examples before starting
- A USB cable that fits your specific Arduino board (not all USB cables carry data — check yours works before you begin)

03 Bill of Materials

COMPONENT	QTY	APPROX. COST	NOTES
Arduino Uno R3 (or compatible clone)	1	$6–10	Clones work identically and are widely available through regional electronics retailers
Half-size breadboard	1	$2–3	
HC-SR04 ultrasonic distance sensor	1	$1–2	
DHT11 temperature & humidity sensor	1	$2–3	DHT22 also works with a one-line code change; DHT11 is cheaper and sufficient for this build
LDR (light-dependent resistor)	1	<$1	

10kΩ resistor	1	<$1	Pulls the LDR reading to a stable value
Male-to-male jumper wires	~15	$2–3	
USB-A to USB-B cable	1	$2–4	Confirm it's a data cable, not charge-only

04 Tools Required

- Computer with Arduino IDE installed
- No soldering required for this build

SAFETY NOTES

This build runs entirely on 5V USB power — there is no meaningful shock risk. Two habits still matter: (1) always disconnect the USB cable before changing any wiring, so you never short two pins together while the board is powered, and (2) double-check polarity (+ / –) on the breadboard rails before connecting a sensor, since reversed power can damage the sensor permanently even at low voltage.

05 System Architecture

All three sensors share the Arduino's 5V and GND rails on the breadboard, then each connects to its own dedicated pin, as shown below.

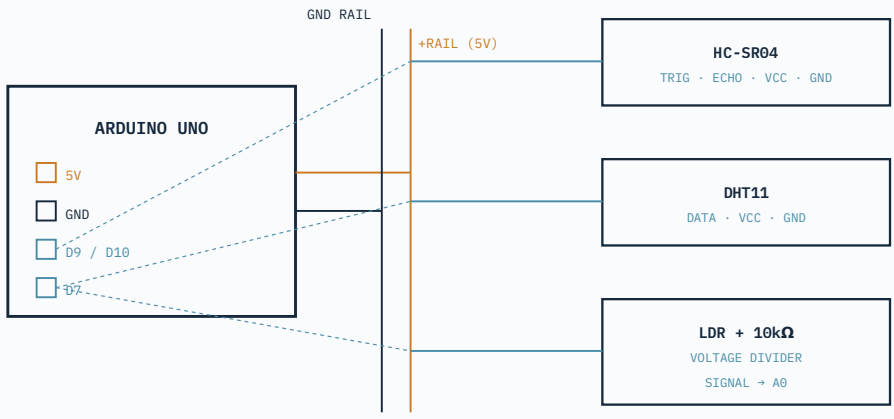

FIG. 1 — SENSOR ARRAY WIRING: PIN ASSIGNMENTS AND SHARED POWER RAIL

SENSOR	SIGNAL PIN(S)	POWER
HC-SR04 (Ultrasonic)	Trig → D9, Echo → D10	5V / GND
DHT11 (Temp/Humidity)	Data → D7	5V / GND
LDR (Light)	Signal → A0 (via voltage divider with 10kΩ resistor)	5V / GND

06 Step-by-Step Build

1. **Place the Arduino and breadboard side by side.** Connect a jumper from Arduino 5V to the breadboard's red (+) rail, and from Arduino GND to the blue (–) rail. Every sensor will draw power from these two rails.
2. **Wire the HC-SR04.** VCC → red rail, GND → blue rail, Trig → Arduino pin D9, Echo → Arduino pin D10.
3. **Wire the DHT11.** VCC → red rail, GND → blue rail, Data → Arduino pin D7. If your DHT11 module doesn't have a built-in pull-up resistor, add a 10kΩ resistor between Data and VCC.
4. **Wire the LDR as a voltage divider.** Connect one leg of the LDR to the red (+) rail. Connect the other leg to both Arduino pin A0 *and* to one leg of the 10kΩ resistor. Connect the resistor's other leg to the blue (–) rail. This divider is what lets the LDR's resistance change translate into a readable voltage.
5. **Double-check all connections** against the wiring table in Section 5 before connecting USB power.
6. **Connect USB and upload the code** from Section 7. Open the Serial Monitor (Tools → Serial Monitor, set to 9600 baud) to see live readings.

07 Code Walkthrough

The code below follows the "one function per sensor" architecture — notice that `loop()` itself is short and readable, because each sensor's complexity is contained in its own function. This is the pattern to keep as you add more sensors in future projects.

```
// MMI Blueprint BP-001 — Autonomous Sensor Array
// Requires: DHT sensor library (install via Library Manager)
#include <DHT.h>

#define TRIG_PIN 9
#define ECHO_PIN 10
#define DHT_PIN 7
#define LDR_PIN A0

DHT dht(DHT_PIN, DHT11);

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  dht.begin();
}

// -- One function per sensor: this is the core architecture pattern --
float readDistanceCM() {
  digitalWrite(TRIG_PIN, LOW); delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  return duration * 0.0343 / 2;  // speed of sound conversion
}

float readLightPercent() {
  int raw = analogRead(LDR_PIN);       // 0–1023
```

```
  return (raw / 1023.0) * 100.0;
}

void loop() {
  float distance = readDistanceCM();
  float humidity = dht.readHumidity();
  float temp = dht.readTemperature();
  float light = readLightPercent();

  Serial.print("Distance: "); Serial.print(distance); Serial.println(" cm");
  Serial.print("Temp: "); Serial.print(temp); Serial.println(" C");
  Serial.print("Humidity: "); Serial.print(humidity); Serial.println(" %");
  Serial.print("Light: "); Serial.print(light); Serial.println(" %");
  Serial.println("---");

  delay(1000);  // read once per second
}
```

Note: `dht.readHumidity()` and `dht.readTemperature()` occasionally return `nan` (not-a-number) if the timing is off — this is normal for the DHT11 and is addressed in the troubleshooting table below.

08 Testing & Troubleshooting

SYMPTOM	LIKELY CAUSE	FIX
Serial Monitor shows nothing	Wrong baud rate, or wrong board/port selected	Set Serial Monitor to 9600 baud; check Tools → Port matches your Arduino
Distance always reads 0 or a huge number	Trig/Echo pins swapped, or loose jumper	Re-check wiring against Section 5's pin table
Temperature/Humidity shows "nan"	DHT11 needs ~2 seconds between reads; reading too fast	Increase the delay() at the end of loop() to at least 2000ms while testing
Light reading stuck at 0 or 100%	Voltage divider wired backward	Confirm the LDR and resistor order matches Step 4 exactly
Arduino not recognized by computer	Charge-only USB cable, or missing drivers	Try a different cable first — this is the most common cause

09 Where This Goes Next

From this blueprint to an ISEF-worthy project

A working sensor array is a component, not yet a project. The step that turns it into an ISEF-level submission is choosing a real problem it can measure — for example: monitoring greenhouse conditions for a local farm, tracking air quality near a school, or building an early-warning system for a specific hazard in your community. Pair this blueprint with **BP-002 (Real-Time Data Logging)** to start recording that data over time, which is what turns "I built a sensor" into "I collected evidence for a hypothesis" — the actual language ISEF judging rewards. See *The ISEF & University Playbook* module for how to frame this formally as a research abstract.