DRAWING NO.	REVISION	MODULE	LANGUAGE
BP-002	A	Data Systems	English

Real-Time Environmental Data Logging & Logic Integration

Give your sensor array a memory and a brain: timestamp every reading, save it permanently to a file, and make the system respond automatically when conditions cross a threshold.

SKILL LEVEL: INTERMEDIATE | BUILD TIME: 2–3 HOURS | EST. COST: $10–16 (PLUS BP-001 HARDWARE)

Before you start: this blueprint builds directly on **BP-001 (Architecture of Autonomous Sensor Arrays)**. You'll reuse the same sensor wiring and code structure — if you haven't built BP-001 yet, start there first.

01 Objective

A sensor that only prints to a screen loses its data the moment you close the Serial Monitor. In this blueprint, you'll add two components to your BP-001 sensor array: a **Real-Time Clock (RTC)**, so every reading gets a real timestamp, and an **SD card module**, so every reading gets saved permanently as a `.csv` file you can open in Excel or Google Sheets. You'll also add basic decision logic — an LED that turns on automatically when a reading crosses a threshold you set. This is the difference between "a sensor that displays numbers" and "a system that collects evidence over time," which is exactly the language ISEF judging rewards.

02 Prerequisites

- A completed BP-001 build (sensor array wired and working)
- Comfort with the `DHT.h` library workflow from BP-001

03 Bill of Materials

COMPONENT	QTY	APPROX. COST	NOTES
MicroSD card module (SPI)	1	$2–3	Any generic breakout module works
MicroSD card (1–8 GB)	1	$3–5	Must be formatted FAT16 or FAT32
DS3231 Real-Time Clock module	1	$2–4	Keeps accurate time even when unpowered, via a small coin battery
LED (any color)	1	<$1	Acts as the threshold-alert indicator
220Ω resistor	1	<$1	Protects the LED
Additional jumper wires	~10	$1–2	

04 Tools Required

- Computer with Arduino IDE and your working BP-001 sketch
- A way to format the microSD card (built into Windows/macOS)

SAFETY NOTES

Same low-voltage 5V system as BP-001 — no meaningful shock risk. Always disconnect USB before wiring changes. The RTC module's coin battery is standard and safe to handle, but avoid puncturing or short-circuiting it directly.

05 System Architecture

The SD card module communicates over SPI (4 data lines plus power), and the RTC communicates over I2C (just 2 data lines plus power) — both run alongside your existing BP-001 sensor wiring without conflict, since they use different Arduino pins.

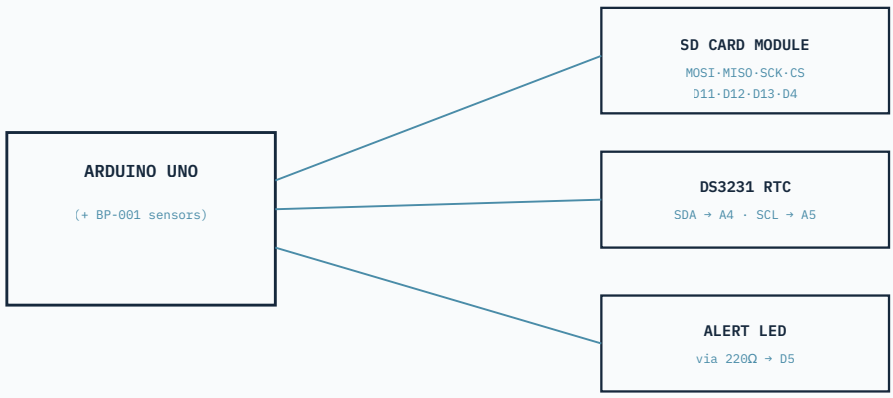

FIG. 1 — ADDING SD LOGGING (SPI) AND RTC TIMESTAMPS (I2C) ALONGSIDE BP-001'S SENSORS

MODULE	INTERFACE	PINS
SD Card Module	SPI	MOSI → D11, MISO → D12, SCK → D13, CS → D4
DS3231 RTC	I2C	SDA → A4, SCL → A5
Alert LED	Digital Out	Anode → D5 (via 220Ω), Cathode → GND

06 Step-by-Step Build

1. **Format the microSD card** as FAT32 (or FAT16 for cards under 2GB) using your computer's built-in formatter.
2. **Wire the SD card module.** VCC → 5V rail, GND → GND rail, then MOSI → D11, MISO → D12, SCK → D13, CS → D4.
3. **Wire the DS3231 RTC.** VCC → 5V rail, GND → GND rail, SDA → A4, SCL → A5. These are the same two pins regardless of how many I2C devices you add later — I2C devices share a bus.
4. **Wire the alert LED.** Long leg (anode) → 220Ω resistor → Arduino D5. Short leg (cathode) → GND rail.
5. **Install two libraries** via Arduino IDE's Library Manager: `RTClib` (by Adafruit) and `SD` (usually pre-installed).
6. **Upload the code below**, then open Serial Monitor at 9600 baud to confirm logging is working before disconnecting.

07 Code Walkthrough

This builds directly on BP-001's sensor functions. Three things are new: getting a timestamp from the RTC, writing a CSV line to the SD card, and a simple `if` statement that checks a threshold and controls the LED — the "logic integration" in this blueprint's title.

```
// MMI Blueprint BP-002 — Data Logging & Logic Integration
// Requires: DHT, RTClib, SD libraries
#include <DHT.h>
#include <Wire.h>
#include <RTClib.h>
#include <SD.h>

#define TRIG_PIN 9
#define ECHO_PIN 10
#define DHT_PIN 7
#define LDR_PIN A0
#define LED_PIN 5
#define SD_CS 4
#define TEMP_THRESHOLD 30.0  // alert if temperature exceeds this (°C)

DHT dht(DHT_PIN, DHT11);
RTC_DS3231 rtc;

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

float readDistanceCM() {
  digitalWrite(TRIG_PIN, LOW); delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH); delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  return pulseIn(ECHO_PIN, HIGH) * 0.0343 / 2;
}
```

```
void loop() {
  DateTime now = rtc.now();
  float distance = readDistanceCM();
  float temp = dht.readTemperature();
  float humidity = dht.readHumidity();
  int light = analogRead(LDR_PIN);

  // -- Logic integration: threshold check drives the LED --
  if (temp > TEMP_THRESHOLD) {
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }

  // -- Build one CSV row and append it to the log file --
  File dataFile = SD.open("log.csv", FILE_WRITE);
  if (dataFile) {
    dataFile.print(now.timestamp()); dataFile.print(",");
    dataFile.print(distance); dataFile.print(",");
    dataFile.print(temp); dataFile.print(",");
    dataFile.print(humidity); dataFile.print(",");
    dataFile.println(light);
    dataFile.close();
  }

  delay(2000);  // log once every 2 seconds
}
```

Note: the first time you run this, manually create `log.csv` with a header row (`timestamp,distance_cm,temp_c,humidity,light`) so the file opens cleanly in a spreadsheet later.

08 Testing & Troubleshooting

SYMPTOM	LIKELY CAUSE	FIX
SD.begin() fails / card not detected	Wrong CS pin, unformatted card, or bad connection	Confirm CS matches your code (D4), reformat card as FAT32
log.csv never appears	SD card write-protected, or full	Check the physical lock switch on the card adapter; try a freshly formatted card
RTC always shows the same old time	Coin battery dead or missing	Insert/replace the CR2032 battery on the RTC module
LED never turns on	Threshold set too high, or wiring reversed	Temporarily lower TEMP_THRESHOLD to test; check LED polarity

09 Where This Goes Next

From logged data to a research abstract

You now have a file full of timestamped, real-world measurements — this is the dataset an ISEF abstract is built around. Open `log.csv` in Google Sheets, graph a variable over time, and you have your first result section. Pair this with **BP-003 (Logic Gate Arrays)** to understand decision-making circuits at the hardware level, without a microcontroller at all — useful context for judges who ask "could this be built without code?"