LANGUAGE	MODULE	REVISION	DRAWING NO.
العربية	أنظمة البيانات	A	BP-002

تسجيل البيانات البيئية اللحظي ودمج المنطق

أعطِ مصفوفة المستشعرات ذاكرة وعقلًا: وثّق كل قراءة بتوقيت زمني، احفظها بشكل دائم في ملف، واجعل النظام يستجيب تلقائيًا عند تجاوز القراءات لحد معين.

المستوى: متوسط | مدة التنفيذ: ٢–٣ ساعات | التكلفة التقريبية: 10–16$ (بالإضافة إلى عتاد BP-001)

قبل البدء: يبني هذا المخطط مباشرة على **BP-001 (بنية مصفوفات الاستشعار المستقلة)**. ستُعيد استخدام نفس توصيلات المستشعرات وبنية الكود — إذا لم تكن قد نفّذت BP-001 بعد، ابدأ به أولًا.

01 الهدف

المستشعر الذي يطبع القراءات على الشاشة فقط يفقد بياناته بمجرد إغلاق `Serial Monitor`. في هذا المخطط، ستضيف مكوّنين إلى مصفوفة مستشعرات BP-001: **ساعة زمنية حقيقية (RTC)** لتوثيق كل قراءة بتوقيت حقيقي، و**وحدة بطاقة SD** لحفظ كل قراءة بشكل دائم في ملف `.csv`. يمكن فتحه في `Excel` أو `Google Sheets`. ستضيف أيضًا منطق قرار بسيط — مصباح LED يضيء تلقائيًا عند تجاوز القراءة لحد معين تحدده أنت. هذا هو الفرق بين "مستشعر يعرض أرقامًا" و"نظام يجمع أدلة عبر الزمن" — وهي بالضبط اللغة التي يكافئها تحكيم ISEF.

02 المتطلبات الأساسية

- إتمام بناء BP-001 (مصفوفة المستشعرات موصولة وتعمل)
- الإلمام بطريقة عمل مكتبة `DHT.h` من BP-001

03 قائمة المكونات

المكوّن	العدد	التكلفة التقريبية	ملاحظات
وحدة بطاقة microSD (SPI)	١	2–3$	أي وحدة عامة تعمل بشكل جيد
بطاقة microSD (١–٨ جيجابايت)	١	3–5$	يجب تهيئتها بنظام FAT16 أو FAT32
وحدة الساعة الزمنية DS3231	١	2–4$	تحافظ على الوقت الدقيق حتى دون تغذية، عبر بطارية صغيرة
مصباح LED (أي لون)	١	<1$	يعمل كمؤشر تنبيه عند تجاوز الحد

مقاومة 220Ω	١	<$1
أسلاك توصيل إضافية	١٠~	$1–2

تحمي مصباح LED

04 الأدوات المطلوبة

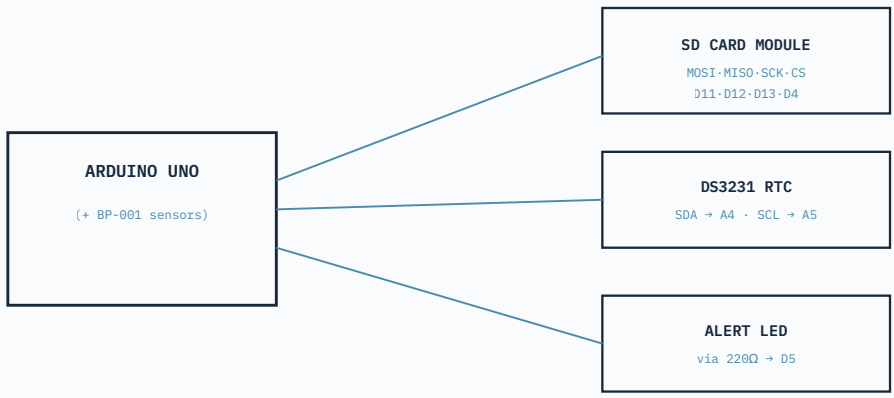
- حاسوب مثبت عليه Arduino IDE وكود BP-001 العامل
- وسيلة لتهيئة بطاقة microSD (مدمجة في Windows/macOS)

ملاحظات السلامة

نفس نظام الجهد المنخفض ٥ فولت كما في BP-001 — لا يوجد خطر صعق حقيقي. افصل USB دائمًا قبل تعديل التوصيلات. بطارية وحدة RTC قياسية وآمنة عند التعامل معها، لكن تجنّب ثقبها أو التسبب بقصر كهربائي فيها مباشرة.

05 البنية العامة للنظام

تتواصل وحدة بطاقة SD عبر بروتوكول SPI (٤ خطوط بيانات بالإضافة للتغذية)، بينما تتواصل وحدة RTC عبر بروتوكول I2C (خطّا بيانات فقط بالإضافة للتغذية) — يعمل كلاهما جنبًا إلى جنب مع توصيلات مستشعرات BP-001 دون أي تعارض، لأنهما يستخدمان منافذ مختلفة على الأردوينو.



شكل ١ — إضافة تسجيل SD (عبر SPI) وتوقيت RTC (عبر I2C) جنبًا إلى جنب مع مستشعرات BP-001

الوحدة	الواجهة	المنافذ
وحدة بطاقة SD	SPI	MOSI → D11, MISO → D12, SCK → D13, CS → D4

DS3231 RTC	I2C	SDA → A4, SCL → A5	
مصباح التنبيه	خرج رقمي		Anode → D5 (via 220Ω), Cathode → GND

06 خطوات التركيب

1. **هيّئ بطاقة microSD** بنظام FAT32 (أو FAT16 للبطاقات أقل من ٢ جيجابايت) باستخدام أداة التهيئة المدمجة في حاسوبك.
2. **وصّل وحدة بطاقة SD.** VCC → خط ٥ فولت، GND → خط الأرضي، ثم MOSI → D11، MISO → D12، SCK → D13، CS → D4.
3. **وصّل وحدة RTC DS3231.** VCC → خط ٥ فولت، GND → خط الأرضي، SCL → A5، SDA → A4. هذان المنفذان ثابتان بغض النظر عن عدد أجهزة I2C التي تضيفها لاحقًا — فأجهزة I2C تتشارك نفس الخط.
4. **وصّل مصباح التنبيه.** الطرف الطويل (موجب) → مقاومة 220Ω → منفذ D5. الطرف القصير (سالب) → خط الأرضي.
5. **ثبّت مكتبتين** عبر إدارة المكتبات في Arduino IDE: RTClib (من Adafruit) و SD (عادة مثبتة مسبقًا).
6. **ارفع الكود أدناه**، ثم افتح Serial Monitor بمعدل 9600 baud للتأكد من عمل التسجيل قبل الفصل.

07 شرح الكود

يبني هذا الكود مباشرة على دوال مستشعرات BP-001. هناك ثلاثة أمور جديدة: الحصول على توقيت من RTC، كتابة سطر CSV إلى بطاقة SD، وجملة شرطية if بسيطة تتحقق من تجاوز حد معين وتتحكم بمصباح LED — وهذا هو "دمج المنطق" المقصود في عنوان هذا المخطط.

```
// 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

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);
  }

  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);
}
```

ملاحظة: في المرة الأولى، أنشئ ملف `log.csv` يدويًا وأضف سطر عناوين (`timestamp,distance_cm,temp_c,humidity,light`) ليُفتح الملف بشكل منظم لاحقًا في برنامج جداول البيانات.

08 الاختبار واستكشاف الأخطاء

العارض	السبب المحتمل	الحل
فشل `SD.begin()` / البطاقة غير مكتشفة	منفذ CS خاطئ، بطاقة غير مهيأة، أو توصيلة سيئة	تأكد أن CS يطابق الكود (D4)، أعد تهيئة البطاقة بنظام FAT32
ملف `log.csv` لا يظهر أبدًا	البطاقة محمية ضد الكتابة، أو ممتلئة	تحقق من مفتاح القفل الفيزيائي على محول البطاقة؛ جرّب بطاقة مهيأة حديثًا
RTC يعرض نفس الوقت القديم دائمًا	البطارية فارغة أو مفقودة	أدخل/استبدل بطارية CR2032 في وحدة RTC
مصباح LED لا يضيء أبدًا	الحد مضبوط مرتفعًا جدًا، أو التوصيل معكوس	اخفض `TEMP_THRESHOLD` مؤقتًا للاختبار؛ تحقق من قطبية LED

09 الخطوة التالية

من البيانات المسجلة إلى ملخص بحثي

أصبح لديك الآن ملف مليء بقياسات حقيقية وموثّقة بالتوقيت — هذه هي مجموعة البيانات التي يُبنى عليها ملخص ISEF. افتح `log.csv` في Google Sheets، وارسم متغيرًا عبر الزمن، وستحصل على أول قسم

نتائج لمشروعك. اجمع هذا مع **BP-003 (مصفوفات البوابات المنطقية)** لفهم دوائر اتخاذ القرار على مستوى العتاد، دون أي متحكم دقيق على الإطلاق — سياق مفيد عندما يسأل المحكّمون "هل يمكن بناء هذا دون برمجة؟"