**Spike Report — Platform for Prismatica**

**Owner: Matthew  
Timebox: 5 days  
Options:  
A) Android handset + Java/Kotlin + SQLite  
B) Raspberry Pi + Python + MongoDB**

**Result Summary (1–3 sentences)**

**Android + SQLite is stable for mobile CRUD, but it’s too rigid for Prismatica’s evolving, sensor-heavy use case. Standard Android devices can’t directly interface with arbitrary sensor buses (GPIO/I²C/SPI) without special hardware, and SQLite is awkward for high-frequency, schema-evolving time-series. Overall, this stack lacks the flexibility Prismatica needs.**

**Recommendation: Do not adopt (Option A). Proceed with Option B spike.**

**1) Findings (data)**

**Technical gotchas**

* **Android handsets have limited direct hardware I/O; custom sensors typically require BLE/USB bridges or special builds.**
* **SQLite is robust for local state, but not ideal for large, changing time-series (no built-in TTL, partitioning, or flexible schemas).**
* **Android lifecycle is manageable, but it doesn’t add value for headless/edge sensing.**

**DX / complexity notes**

* **Android app DX: good (tooling, testing, packaging).**
* **Sensor adaptability: limited on phones; broad on Raspberry Pi (native GPIO/I²C/SPI, easy driver ecosystem).**
* **Data model agility: easier with document/time-series stores than with SQLite migrations.**

**2) Decision Matrix**

**Weights: Performance (p95)=3, Dev speed=2, Testability=2, Long-term maintenance=3, Ecosystem/docs=1.  
Ratings: 1 (poor) → 5 (excellent).**

| **Criterion** | **Weight** | **Option A: Android + SQLite** | **Option B: Pi + Python + Mongo** |
| --- | --- | --- | --- |
| **Performance (p95)** | **3** | **1** | **4** |
| **Dev speed/boilerplate** | **2** | **2** | **4** |
| **Testability** | **2** | **4** | **4** |
| **Long-term maintenance** | **3** | **4** | **4** |
| **Ecosystem/docs** | **1** | **4** | **4** |
| **Weighted score** |  | **31** | **44** |

**Conclusion: Option B wins on total score, primarily due to sensor flexibility and data-model agility.**

**3) Costs & Risks**

* **Tech debt / divergence: Prismatica’s analytics roots (Python, pandas, PyTorch) fit poorly with a Java/SQLite mobile core; adds integration friction.**
* **Hardware fit: Phones constrain sensor interfaces; Pi aligns with diverse sensors and edge deployments.**
* **Data evolution: SQLite migrations become brittle as schemas evolve; document DB or time-series DB better matches changing sensor payloads.**

**4) Next Steps**

1. **Spike Option B: Raspberry Pi + Python + MongoDB**
   * **Minimal vertical slice: capture 2–3 sensor streams (mock OK), persist to Mongo, query/aggregate, expose REST/WebSocket.**
   * **Measure: write throughput (events/sec), p95 read latency, CPU/RAM on Pi, schema change effort (add a field).**
2. **Gateway pattern: define a thin HTTP/MQTT API so Android (or any client) can view/control without owning the sensor layer.**
3. **Re-evaluate storage (if needed): compare Mongo vs. Timeseries options (MongoTS, TimescaleDB) using the same slice.**
4. **Decision: adopt B if metrics and DX meet targets; otherwise adjust store or transport.**