📄 8.3 – End-to-End System Test (Cloud Logging Path)

Goal

Demonstrate a complete flow:

`CAN data → Predictive rules → Event → Internet → Cloud log → Human observer`

Zero servers, zero backend maintenance.

Implementation Summary

Component	Role
UNO Q Linux	Rule execution + event generation
HTTPS POST	Transport
Google Apps Script	Cloud webhook
Google Sheets	Event log & dashboard

Execution Flow

1. **CAN frames generated**
2. **Values decoded using DBC**
3. **Predictive logic evaluates drift/fault rules**
4. **Event emitted on trigger**
5. **Event encoded as JSON**
6. **Event POSTed to Apps Script**

7. **Apps Script appends row to Sheet**
8. **Sheet visible to remote observer**

Demo Results

A real degradation sequence produced:

```
EARLY   HARNESS_A_DRIFT
ALERT   HARNESS_A_LOW_VS_DCDC
ALERT   HARNESS_B_LOW_VS_DCDC
ALERT   HARNESS_C_BOTH_LOW_VS_DCDC
```

Sheet captured ordered event stream as rows with voltages + metadata.

Success Criteria Check

Requirement	Result
On-device predictive logic	✔
Harness root cause classification	✔
Cloud event visibility	✔
Time-ordered	✔
Severity encoded	✔
Data structured	✔
Zero downtime	✔
Demo-friendly	✔

Phase 8 Key Takeaways

- Local analytics can classify harness failures
- EARLY phase detection provides predictive value
- Cloud logging proves the event pipeline
- No backend infrastructure required
- Spreadsheet is adequate for early demo + debugging
- System now resembles actual telematics stack:

```
Sensor → Edge Compute → Event → Cloud → Observer
```

Phase 8 Completes the Loop

Phase 1–7 answered:
Can we detect failures locally? → Yes

Phase 8 answers:
Can we show them to humans remotely? → Yes