MEMORY-BASED FORENSIC FRAMEWORK FOR AUTOMATION

Presented by Jhil Patel

Mentored by : Prof. Tapas Kumar Maiti

THE CORE IDEA

“Turn a simple ESP32 + SD card logger into a legally reliable, tamper-evident forensic system using SHA-256 hash-chaining to ensure every recorded value is verifiable and protected under Indian IT Act standards”

PROBLEM

- Sensor logs on SD cards can be manually altered, overwritten, or replaced without leaving any visible trace of tampering.

- Traditional logging systems cannot differentiate genuine data from forged entries, making them unreliable for audits, automation, and legal evidence.

WHY IS THIS A SERIOUS PROBLEM?

- Data becomes inadmissible in court under Section 65B if integrity cannot be proven.
- Compromised logs break automation systems, audits, compliance, safety checks.
- IoT devices are typically cheap → no built-in forensic layer.
- Industries need trustworthy evidence logs.

POSSIBLE TAMPERING TECHNIQUES

- Manual File Editing
  - Remove SD card → modify CSV → reinsert unnoticed.
- SD Card Swapping / Cloning
  - Replace original with forged copy.
- Network-Based Modification
  - HTTP requests altering entries if device is exposed.
- Metadata Manipulation
  - Changing timestamps or deleting rows.

PREVALENT FORENSIC TECHNIQUES

- Chain of Custody documentation
- Write-blockers to prevent accidental modification
- Hash-based verification (MD5, SHA-1, SHA-256)
- Secure storage + audit logs
- Metadata preservation

But these methods are mostly offline & expensive → not suitable for embedded systems.

EXISTING TOOLS & INDUSTRY PRACTICES

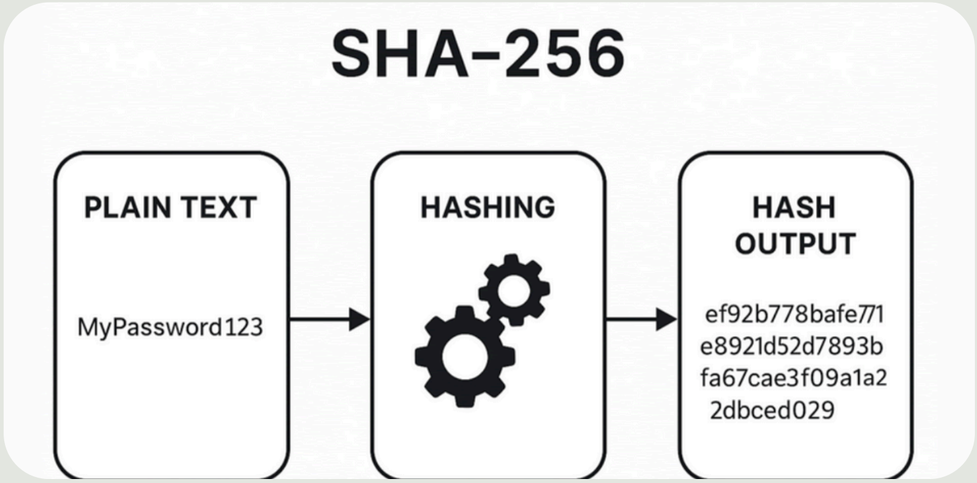
- Metasploit – simulates attacks, tests vulnerabilities in IoT systems.
- Falcon Neo2 – forensic-grade SD card imaging and evidence preservation.
- Wireshark – captures and analyses network tampering attempts.
- Software Hashing Tools – MD5, SHA-1/SHA-256 for digital integrity verification.

IMPLEMENTED STRATEGY

- Embed a lightweight forensic layer directly on the ESP32.
- Use SHA-256 hash-chaining to protect every log entry.
- Detect even a single-value modification instantly.
- Align logging integrity with IT Act & Section 65B requirements.

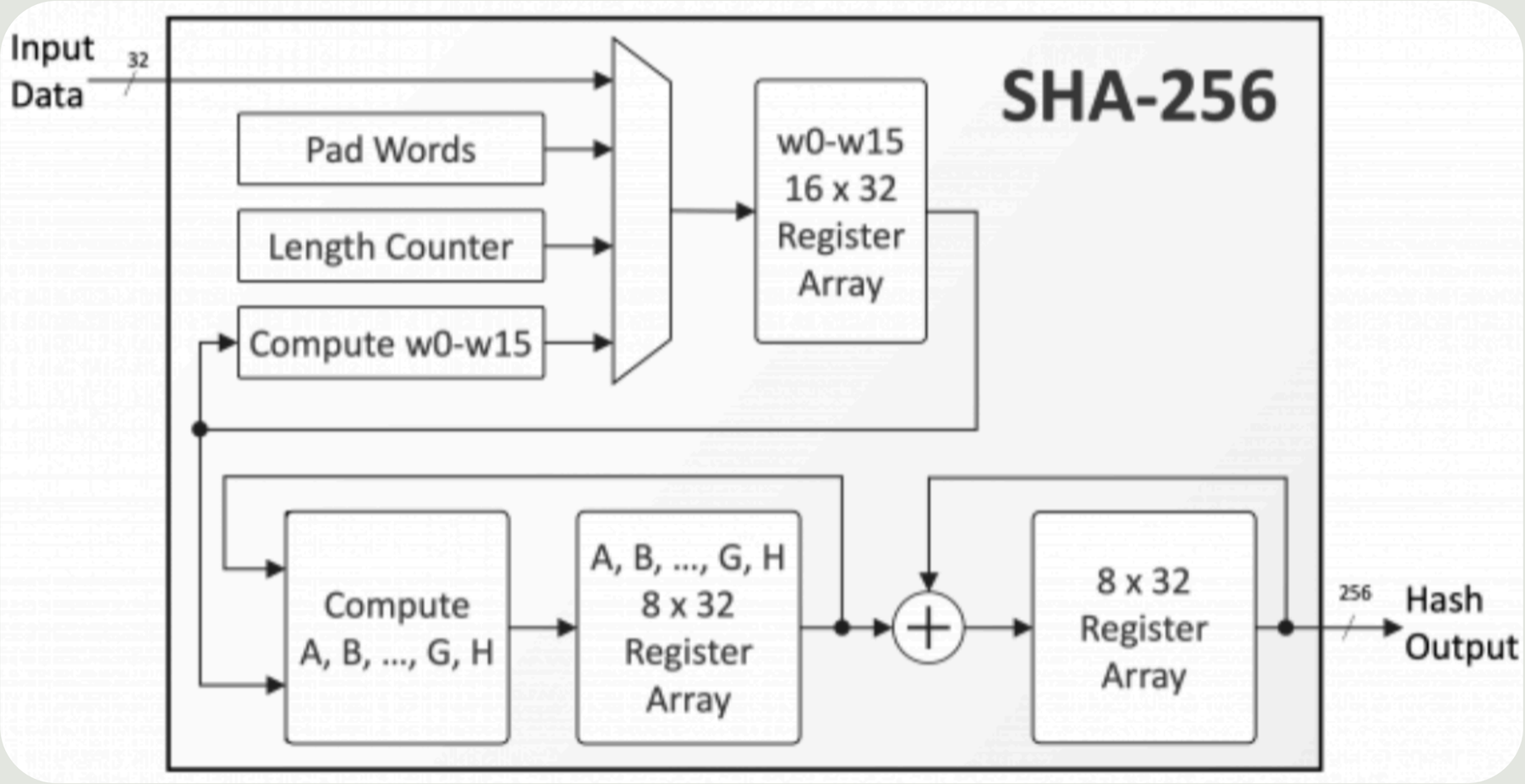
WHAT IS SHA256?

- SHA (Secure Hash Algorithm) is a family of cryptographic hash functions used to generate a fixed-length digital fingerprint of data.



- It converts any input — small or large — into a unique 256-bit value that cannot be reversed or forged. Even a one-character change in the input produces a completely different hash, making SHA ideal for tamper detection, integrity checking, and digital forensics.

WHAT IS SHA256?



Source : https://www.cast-inc.com/security/crypto-hash-functions/sha-256

HASH-CHAINING LOGIC USED

```
entry_hash = SHA256(previous_hash + (timestamp + value))
```

Thus:

- Each line depends on the previous line
- Editing ANY record breaks the chain
- Final hash stored in NVS works as a “trusted root”

This mimics how blockchains detect tampering.

METHODOLOGY

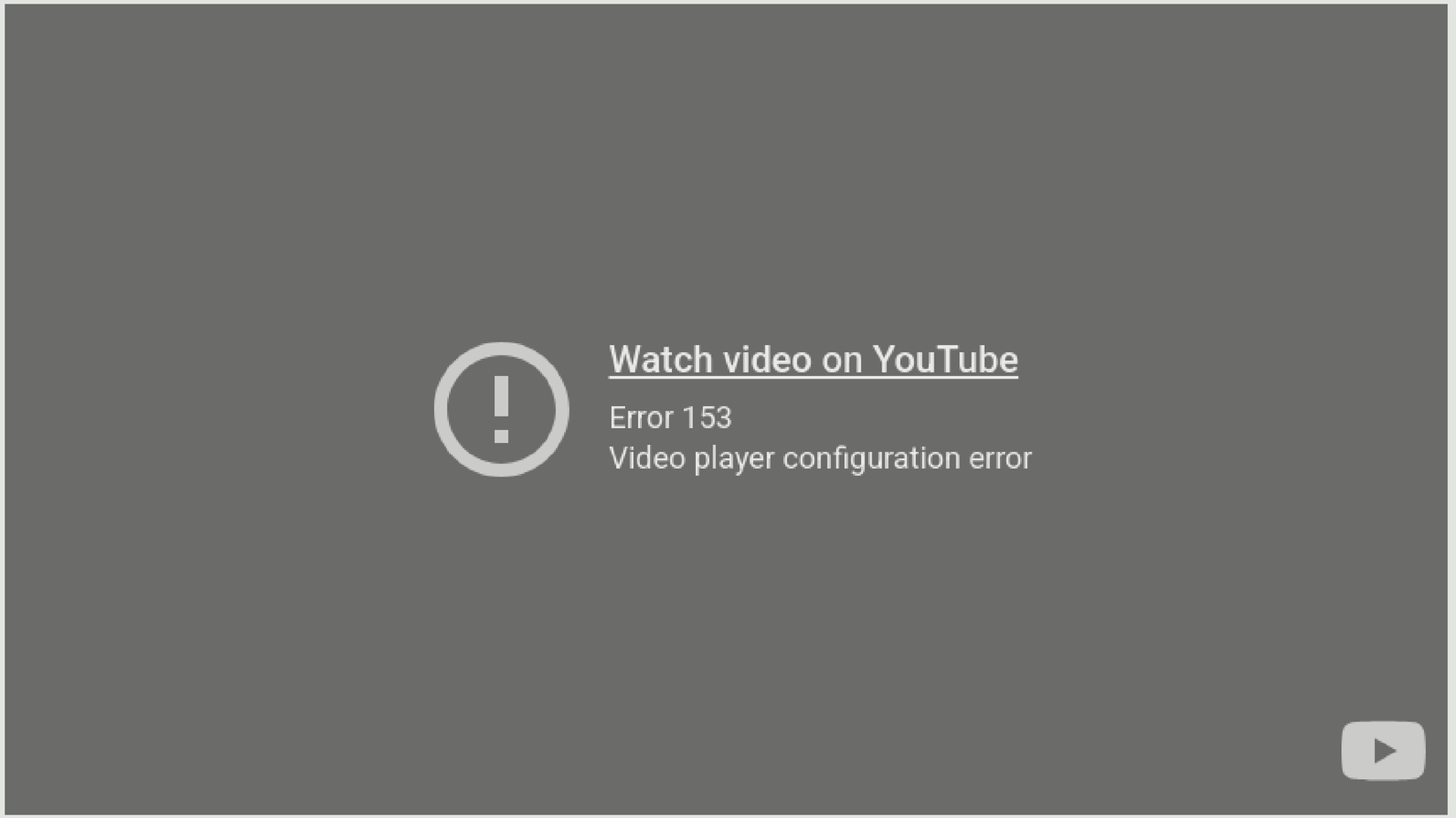
- ESP32 reads sensor value → generates timestamp via NTP
- Get prev_hash from last SD entry
- Compute current_hash = SHA256(prev_hash + new_entry)
- Save line:
- datetime, temp, prev_hash, entry_hash
- Store entry_hash in NVS
- Verification: recompute chain → compare with NVS final hash.

```
22:21:10.261 -> === INIT MENU ===
22:21:10.261 -> show | append | overwrite | reset
23:01:29.768 ->
23:01:29.768 -> ----- FILE CONTENT -----
23:01:29.768 -> datetime,temp,prev_hash,entry_hash
23:01:29.768 -> 2025-12-11 20:37:21,29.0,0000000000000000000000000000000000000000000000000
```

TESTING METHODS USED

1. Manual Tampering
    - Edited CSV through PC → detected instantly.
2. HTTP-Based Attacks
    - Used /attack_edit & /attack_append endpoints.
    - Wireshark captured all malicious packets.
3. Complete SD Replacement
    - Stored NVS hash mismatch → detected SD swap.

DEMO VIDEO

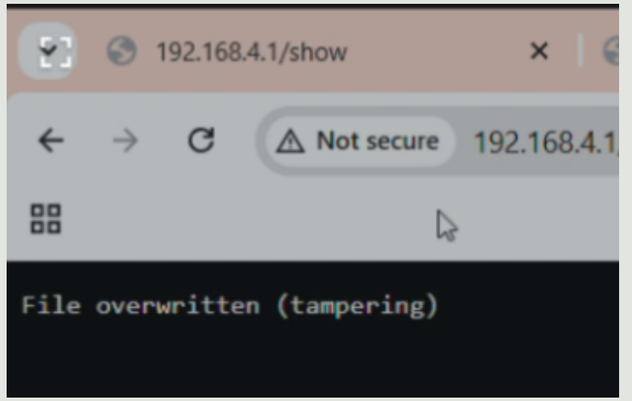

Watch video on YouTube
Error 153
Video player configuration error

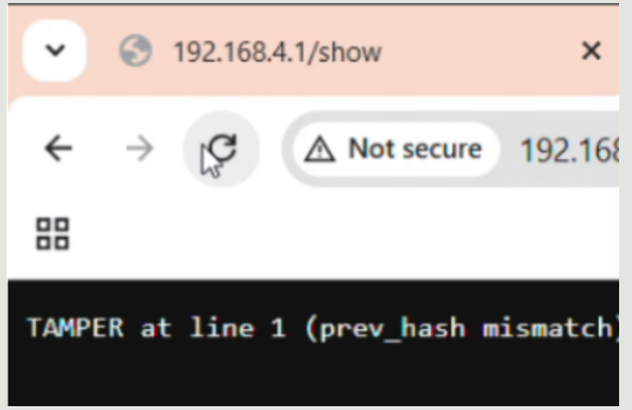
SCREENSHOTS

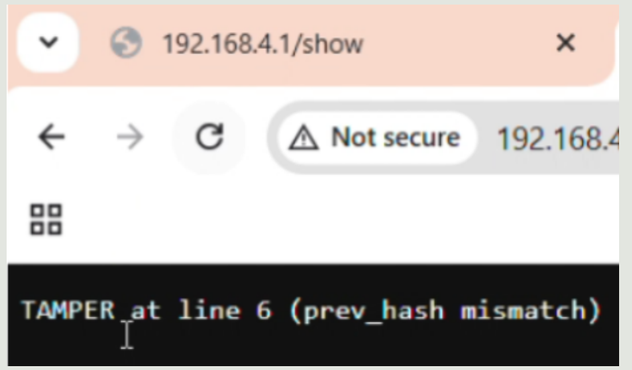
```
22:21:21.695 -> ----- FILE CONTENT -----
22:21:21.695 -> datetime,temp,prev_hash,entry_hash
22:21:21.695 -> 2025-12-09 22:03:40,32.1,0000000000000000000000000000000000000000000000000000000000000000
22:21:21.695 -> 2025-12-09 22:03:45,32.4,f2dfb79b8471121cc092fbb46f42d0d169888ae848123925
22:21:21.743 -> 2025-12-09 22:03:50,5000000,ed8cd42b38972aa19c3251b368c0ed5045a03b0889a9
22:21:21.743 ->
22:21:21.743 -> ----------------------
22:21:21.743 ->
22:21:21.743 -> TAMPERED at line 3 (hash mismatch)
```

```
:18.421 -> 2025-12-09 22:23:41,24.9,15422b6ebeb54580e7a6fa979c5b
:18.470 -> 2025-12-09 22:23:51,22.4,09e892a5cd3e41b82f6ba8aef1e7
:18.470 ->
:18.470 -> ----------------------
:18.470 ->
:18.470 -> TAMPERED at line 3 (prev_hash mismatch)
```

```
22:35:19.974 -> 2025-12-09 22:31:22,21.5,074b09f54f6f2458199fafa440fb
22:35:19.974 -> 2025-12-09 22:31:27,21.8,ba04c507e428204f23f406f6dd7e
22:35:20.023 -> 2025-12-09 22:31:32,31.4,ce0fecf3a9374349a8f9d394423f
22:35:20.023 -> 2025-12-09 22:31:37,24.4,a512f02ab5a6620a12d94bb78c79
22:35:20.023 -> 2025-12-09 22:31:42,100000,a512f02ab5a6620a12d94bb78c
22:35:20.023 -> ----------------------
22:35:20.023 ->
22:35:20.057 -> TAMPERED at line 6 (prev_hash mismatch)
```





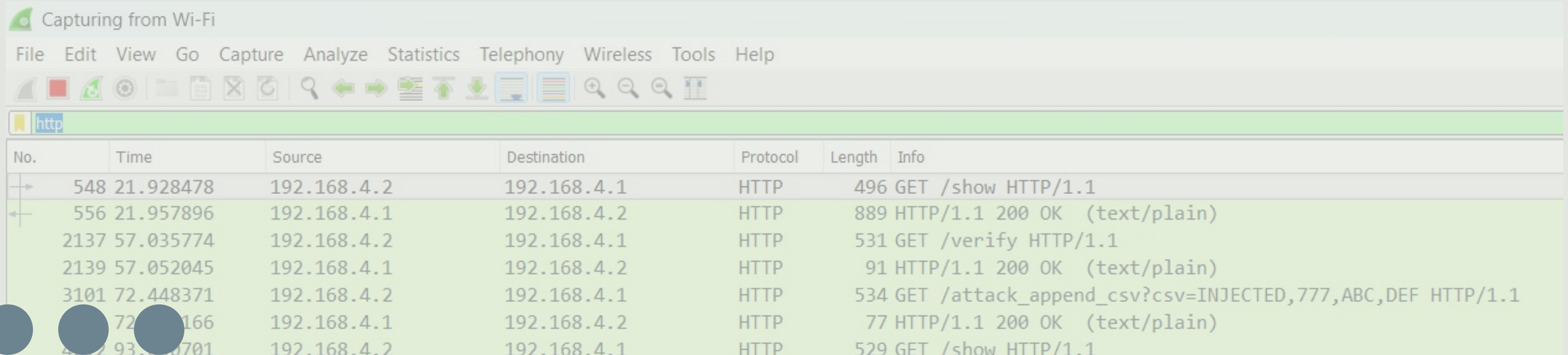


WHAT THE SYSTEM CAN DETECT

- ✓ Manual CSV editing
- ✓ Row deletion / row insertion
- ✓ SD card swapping
- ✓ Cloned SD card
- ✓ HTTP-based tampering (via simulation test)

WIRESHARK TRAFFIC CAPTURE

- Captured HTTP packets sent to ESP32.
- Shows clear-text tampering attempts.
- Demonstrates how network attackers manipulate logs.
- Highlights the need for cryptographic verification.



WHERE THE SYSTEM STILL LACKS

- ❌ No encryption → attacker can still read data
- ❌ HTTP interface not password protected
- ❌ Does not prevent tampering — only detects it
- ❌ No secure boot / flash integrity check

FUTURE SCOPE

- Add SHA-512 or HMAC for stronger trust
- Add password / token protected HTTP endpoints
- Encrypt logs using AES-256
- Add Secure Boot + Flash Integrity
- Upload hashed logs to cloud server for redundancy
- Automate tamper alerts via SMS/email.

LEARNINGS

- Understood real-world forensic principles used in cybersecurity.
- Implemented cryptographic hashing inside a microcontroller.
- Learned about legal implications (IT Act 2000, Section 65B).
- Explored attack surfaces in embedded/IoT systems.
- Understood how to design tamper-evident logging solutions.

CONCLUSION

The system proves that low-cost IoT hardware can achieve reliable, legally-aligned tamper detection using SHA-256 hash-chaining, making sensor data forensically trustworthy.

Thank You