SD Card Forensics on ESP32

RAS Minor Mini-Project - 1 PC408

Phase 2 : Hardware Implementation without forensic protection

Mentored by : Dr. Tapas Kumar Maiti

Jhil Patel

06.12.2025
SID : 202301090

INTRODUCTION

Phase 2 focuses on implementing a basic hardware-level data logging system using the ESP32 microcontroller and an SD card module. At this stage, no forensic protection mechanisms (such as hashing, integrity verification, tamper detection, or secure storage) are applied. The goal is to demonstrate baseline data acquisition, time stamping, storage, and user interaction before adding forensic mechanisms in later phases. This phase establishes the functional system that future forensic protections will enhance.

OBJECTIVES

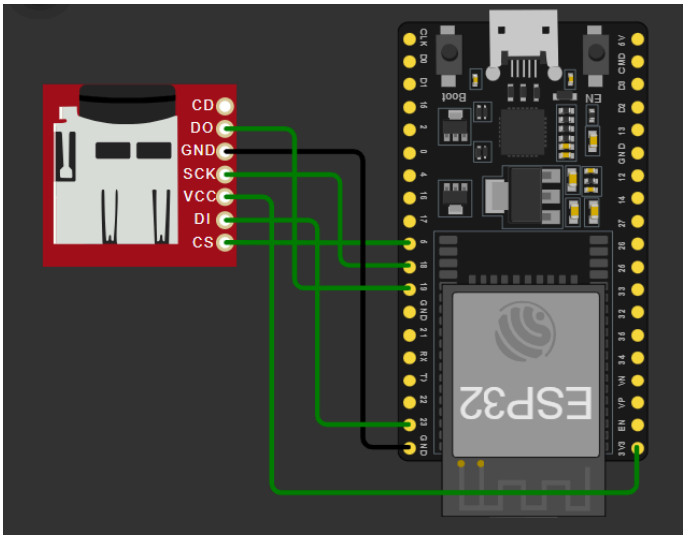
- Implement a standalone ESP32 data logger capable of:
- Recording temperature values at fixed intervals
- Capturing accurate date and time stamps via NTP
- Storing data locally on an SD card in CSV format
- Enable user interaction through serial commands:
    - show → Display logged data
    - append → Start logging and append new measurements
    - append <value> → Manually add custom data
    - overwrite → Clear file and restart logging
    - stop → Stop logging and return to initial menu
- View the file
- Simulate tempering values manually.
- Verify system status (without forensic checks)
- Establish a stable hardware + firmware base for Phase 3 (tamper-proof logging).

HARDWARE COMPONENTS USED

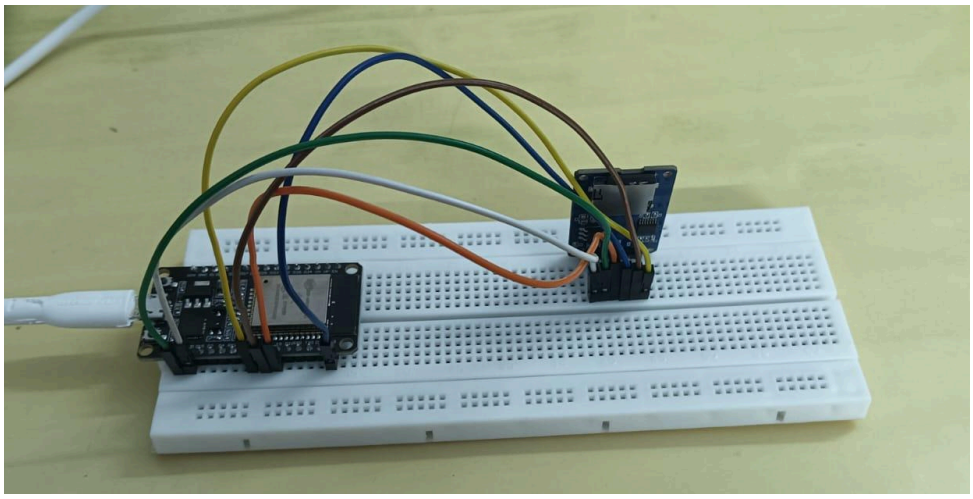
Component	Description
ESP32 Dev Board	Microcontroller handling logging, networking & SD interface
MicroSD Card & Module	Local storage for recorded sensor/entered values

Wi-Fi Network	Required for NTP time synchronization
Jumper Wires	Connections between SD module and ESP32

WOKWI SIMULATION CONNECTION REFERENCE

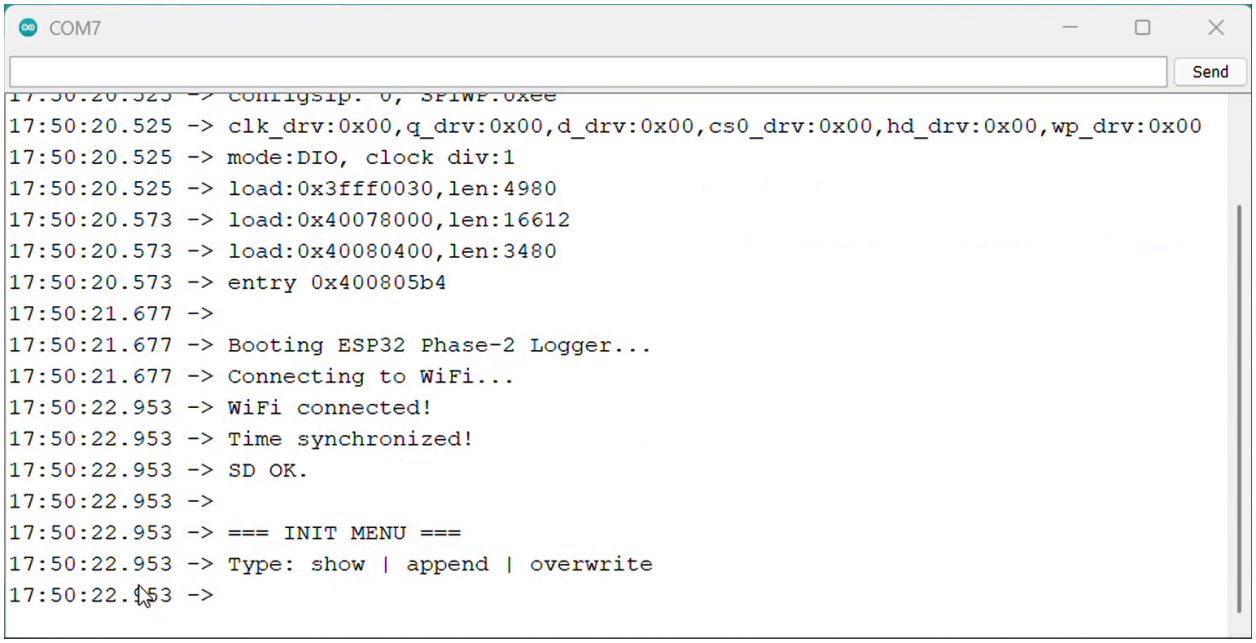


ACTUAL CONNECTIONS

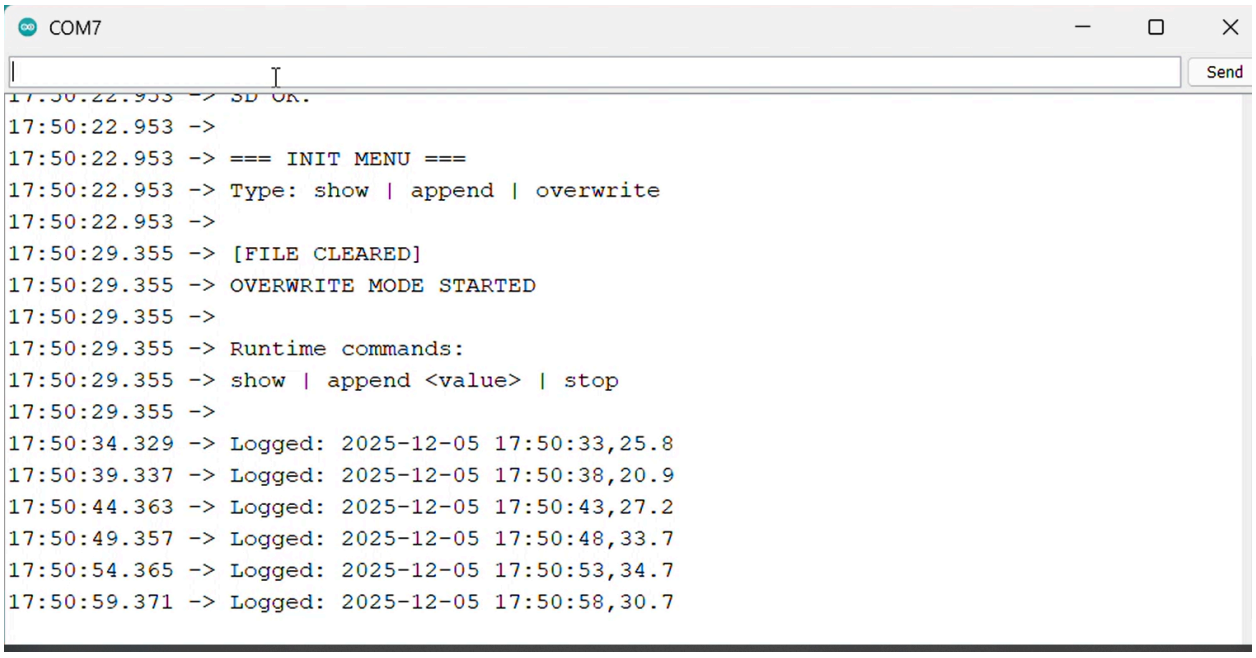


DRY RUN RESULTS

In normal flow the temperature sensor data is logged every 5 seconds for dry run



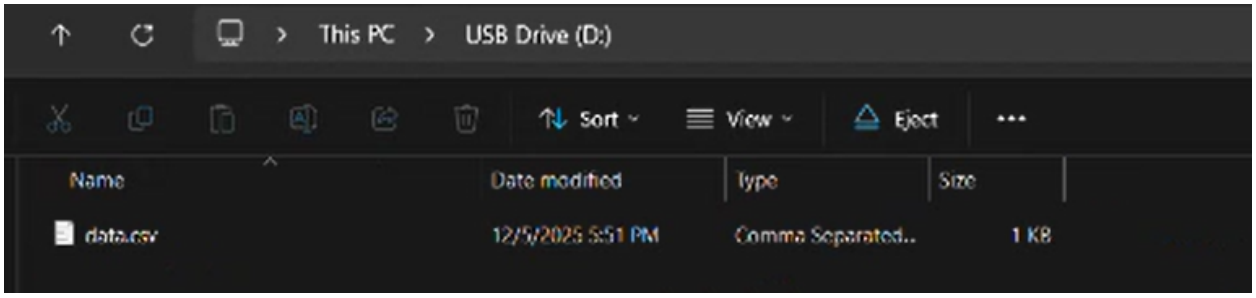
In normal flow the temperature sensor data is logged every 5 sec with Date-Time stamp.

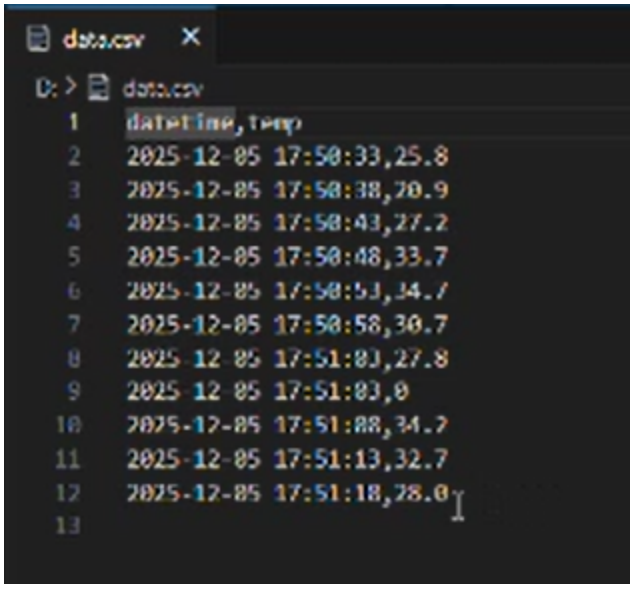


Manually appending a value to simulate an external data injection, for example an out-of-range temperature of 0°C.

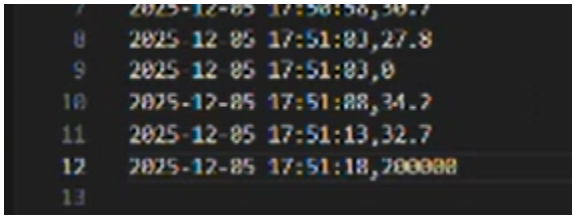
```
stop
17:50:29.355 -> [FILE CLEARED]
17:50:29.355 -> OVERWRITE MODE STARTED
17:50:29.355 ->
17:50:29.355 -> Runtime commands:
17:50:29.355 -> show | append <value> | stop
17:50:29.355 ->
17:50:34.329 -> Logged: 2025-12-05 17:50:33,25.8
17:50:39.337 -> Logged: 2025-12-05 17:50:38,20.9
17:50:44.363 -> Logged: 2025-12-05 17:50:43,27.2
17:50:49.357 -> Logged: 2025-12-05 17:50:48,33.7
17:50:54.365 -> Logged: 2025-12-05 17:50:53,34.7
17:50:59.371 -> Logged: 2025-12-05 17:50:58,30.7
17:51:04.332 -> Logged: 2025-12-05 17:51:03,27.8
17:51:04.523 -> Logged: 2025-12-05 17:51:03,0
17:51:09.374 -> Logged: 2025-12-05 17:51:08,34.2
17:51:14.351 -> Logged: 2025-12-05 17:51:13,32.7
17:51:19.340 -> Logged: 2025-12-05 17:51:18,28.0
```

Viewing logged data file “data.csv”

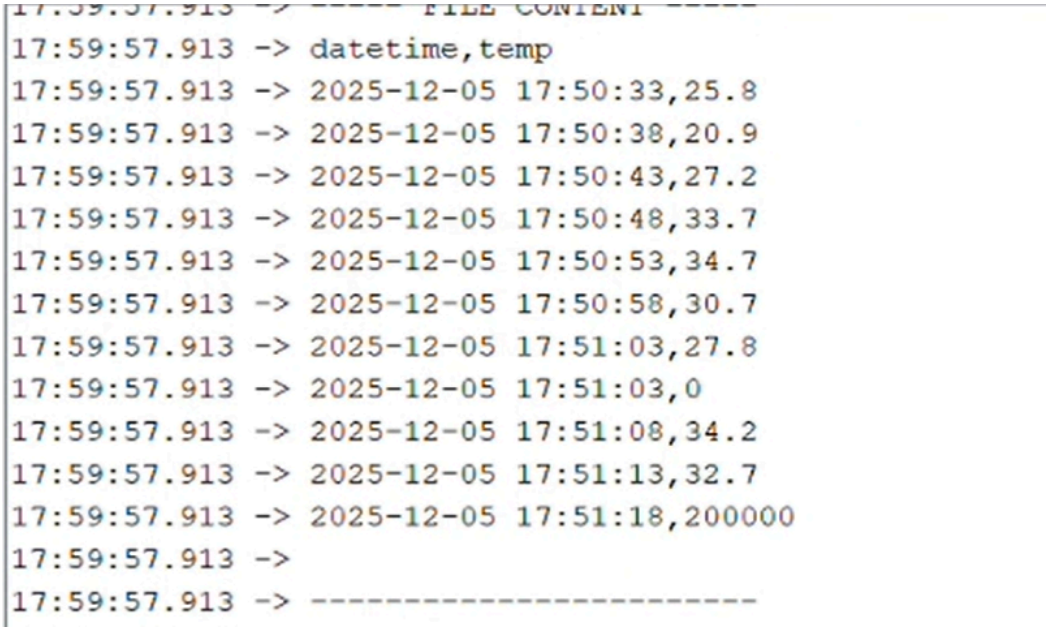




Manually tempering the last data value as 200000.



Viewing the data again in Serial Monitor



In the absence of a security layer, manual tampering and legitimate ESP32 logged data are treated identically by the system. As a result, malicious or accidental data injections may remain untracked, since there is no visible way to distinguish authentic records from tampered ones.

DEMO VIDEO LINK

https://youtu.be/yDZdFGvUID0

CONCLUSION

In Phase 2, a complete functional hardware-based logging system was implemented using ESP32 and SD card storage.
The system successfully achieves:

- Time-stamped data logging
- Stable long-term storage
- User interaction through serial monitor.
- Basic environmental monitoring workflow

This serves as the foundation for **Phase 3**, where forensic security features (hash chain, tamper detection, clone detection, integrity verification) are added to transform this simple logger into a **forensically sound IoT evidence recorder**.