Sid Pasumarthi, et.al, International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 15, Issue 7, July 2025, pp 84-87

RESEARCH ARTICLE OPEN ACCESSC

Time Synchronization in Automotive Systems: A Conceptual Review and Prototype-Based Demonstration

Sid Pasumarthi*, Sivani Manisha Vankayala**
**(Automotive Professional and Independent Researcher, Michigan, USA, Email: sidpasumarthi@gmail.com)*
*** (Automotive Professional and Independent Researcher, Michigan, USA, Email: sivani.manisha@gmail.com)*

ABSTRACT
Modern automotives have numerous Electronic Control Units (ECU) that operate in distributed networking systems using protocols such as CAN, Ethernet, LIN, Flex-Ray. Among the networking protocols CAN remains to be widely used in automotive systems due to its reliability, cost effectiveness, real-time communication and prioritization. One of the limitations of CAN would be lack of native support for precise time synchronization which would be a critical requirement for ensuring data consistency and event coordination for distributed networking systems. Existing protocols such as CANopen, Time Triggered Communication (TTC) provide synchronized solutions but involve hardware-specific configurations or lack two-way time validation mechanisms. The paper presents a low cost and novel time synchronization solution based on the concepts of IEEE 1588 Precision Time Protocol (PTP) and CANopen Protocol. The solution is demonstrated over the Arduino Uno R4 connected over CAN Bus, the solution incorporates periodic time broadcast, slave clock correction, and acknowledgement based synchronization. The work includes a conceptual foundation, practical implementation details, results, limitations, and potential real-world applications. This accessible methodology provides a stepping stone toward cost-effective synchronization solutions in educational, prototyping, and low-end automotive environments.
Keywords: ECU, CAN, PTP, TTC, CANopen

Date of Submission: 15-07-2025 Date of acceptance: 30-07-2025

I.INTRODUCTION

Time synchronization errors may happen between ECUs with inconsistent timestamps causing potential incorrect ordering of events.

IEEE 1588 Precision Time Protocol (PTP), is a time synchronization protocol that can enable incredibly precise synchronization of distributed clocks in Ethernet-based systems. PTP is a widely adopted protocol, especially in industries where deterministic behavior is necessary, like telecommunications, industrial automation, and power grid systems.

PTP works by establishing a master-slave clock relationship. The master sends synchronization messages with timestamps to slave devices.

The protocol is usually expressed in terms of a four-message exchange which can be analyzed as follows:

Sync message – The Master transmits the timestamp.

Follow_Up message – The Master sends the exact transmission time of the Sync message.

Delay_Req message – The Slave sends the message to Master to understand the time associated with its propagation delay.

Delay_Resp message – The Master sent to Slave to allow the determination of offset and delay.

PTP can provide time synchronization accurately, and often to the sub-microsecond level. This precision however makes the standard difficult to implement on low-cost, resource constrained microcontrollers (like an Arduino Uno), which do not have the Ethernet hardware necessary to conform to the standard and additional hardware must be included to add the timestamping capability that a low-cost embedded system does not provide.
CANopen, is a higher-layer protocol based on CAN for communication. CANopen offers synchronization by using two types of messages.

SYNC Object message – Broadcast message that the Master sends to inform all Slave ECUs to be synchronized in their operations

TIME Object message – The Master transmits the message to Slave to provide absolute time

CANopen has challenges of synchronization operations such as two-way delay compensation and clock adjustment to account for drift, which leads to a lower accuracy of synchronization. However, it has relatively low overhead and is directly compatible with CAN, which makes it attractive for lightweight systems.
This paper outlines a low-cost, low complexity time synchronization methodology based on principles found in both the IEEE 1588 PTP standard and the CANopen Protocol, and implemented over a standard CAN Protocol.

II.PROPOSED METHODOLOGY

The proposed methodology utilizes a simple time synchronization scheme over the CAN using inexpensive Arduino Uno microcontrollers. Following an industrial protocol like the IEEE 1588 Precision Time Protocol (PTP) and the CANopen SYNC, this solution outlines a reliable periodic synchronization scheme, without requiring high-end hardware.

The system consists of two nodes, a Master ECU and a Slave ECU, with each node being an Arduino Uno with a CAN transceiver (MCP2515).

The Master ECU is the time authority by periodically broadcasting time messages over the CAN bus. Each time message contains the Master ECU's internal timestamp based on the Master ECU's own millisecond counter which counts the number of milliseconds since boot.

The Slave ECU of the system reads the time message, and compares it with its own internal timer and computes its offset from the Master ECU's time.

The Master ECU has a predefined tolerance of time offsets to account for message delay. If the offset is greater than this threshold, the Slave ECU will correct its internal time to match that of the Master ECU. After performing the correction, the Slave ECU will also send an ACK (acknowledge) message back to the master confirming the synchronization.

The synchronization status which would be "Time Synchronized" or "Not Synchronized" is displayed on the Master ECU LCD screen. Both Master ECU and Slave ECU also log their active time and status of time synchronization to the Serial Monitor, for debugging and performance measurement.
It is assumed that there is minimal network delay in this method and therefore does not need any delay compensation that would be required in PTP, thereby making it suitable for education and low-cost applications.
The method operates in real-time with a fixed cycle period, (with an example cycle period of 2 seconds), where each node is still active during the entire cycle, and maintains temporal coherence.

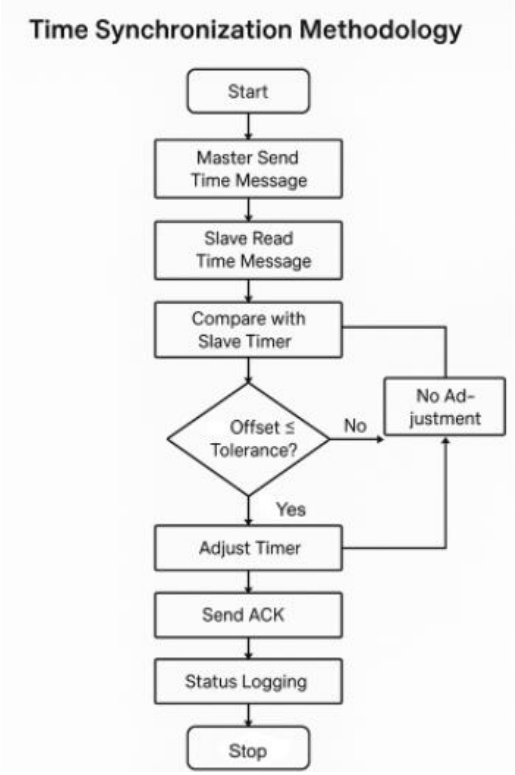

Fig 1: Flow chart of Proposed Methodology

The proposed method used a combination of the SYNC and TIME objects in CANopen and master-slave time authority in PTP to allow an effective time sync mechanism for embedded prototypes, limited automotive systems, or distributed automotive systems.

III.PROTOTYPE SYSTEM OVERVIEW

The prototype is aimed to create a straightforward time synchronization system over CAN with Arduino Uno's using a master-slave communication system with feedback acknowledgement and a status display using an LCD.

Hardware:
Master ECU: Arduino Uno + MCP2515 CAN module + LCD1602 (I2C)

Slave ECU: Arduino Uno + MCP2515 CAN module

CAN Bus: 2-wire CAN cable with 120Ω termination.

Communication Design:
The master sends its local time (HH:MM:SS) every 2 seconds as a CAN message with CAN ID 0x100.

The slave receives this and updates its internal clock with offset logic using millis().

The slave responds to the master with an ACK message with CAN ID 0x101 and includes the corrected time.

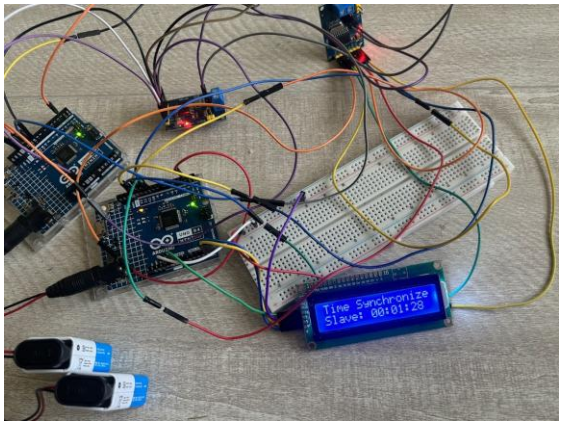

Fig 2: LCD Display, when the Time Synchronization is successful

The master receives the ACK message and uses it to update the LCD with the result of the synchronization.

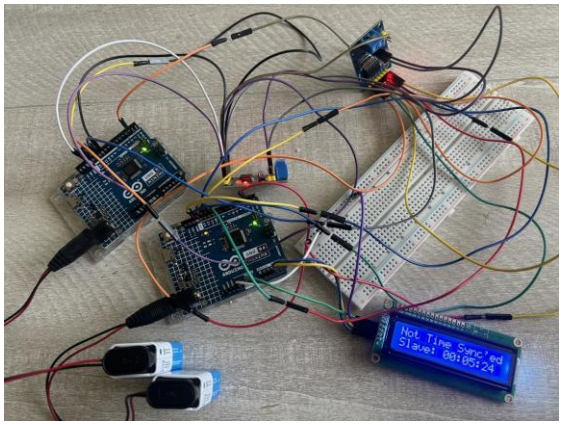

Fig 3: LCD Display, when the Time Synchronization is Failed

Functionality Behavior:
Periodic Sync: The Master sends sync messages every 2 seconds

Clock Correction:The Slave compares the Master and adjusts its internal timer

Acknowledgment Protocol: The Slave sends an acknowledgement message to the Master to confirm a sync was successful

LCD Feedback (Master): Displayed though the LCD, "ACK Received" or "SYNC FAILED"

Fig 4: Snippet of Master and Slave Module Time Synchronization Code

IV.Limitation and design constraints

Software based timers: Proposal is based on the on millis(), therefore, can only be accurate to milliseconds

No delay compensatory: Proposal doesn't account for CAN transmission delay, which is the opposite from PTP of which would add delay compensation

No hardware timestamps: Proposal doesn't account for gated jitter or accurate message timing

One-to-one system: Proposal is designed to be connected between One master node and One slave node

Volatile time base: Proposal doesn't have RTC, resets the time lose upon power

V.REAL WORLD APPLICATIONS

Low-Cost Fleet Monitoring: Commercial vehicles or non-critical systems, rough time alignment of a few milliseconds is frequently considered to be "good

enough" for remote health monitoring and behavior monitoring.

Sensor Fusion: Data allocation management from multiple types of sensors, which require some response time from the system to sync data across networked nodes

Educational and Demonstration Platforms:
A useful platform for teaching about CAN, time synchronization and embedded networking in academic labs.

VI.COMPARISON OF PROPOSED METHODOLOGY

Time Synchronization Feature Comparison

Feature	IEEE 1588 PTP	CANopen SYNC	This Prototype
Time Broadcast	✅ Sync Msg	✅ SYNC Object	✅ Time sent via CAN
Clock Correction	✅ Yes	⚠️ Optional	✅ Delta applied via offset
Delay Measurement	✅ Yes	❌ Not supported	❌ Not implemented
ACK Feedback	⚠️ Indirect	❌ None	✅ ACK via CAN ID
Hardware Timestamps	✅ Required	❌ Not supported	❌ Not supported
Microcontroller Usability	❌ High-end only	✅ Yes	✅ Yes (Arduino-compatible)

Fig 5: Comparison between PTP, CANopen and Proposed methodology Time Synchronization

VII.CONCLUSION

We have demonstrated that a lightweight, Arduino-based time synchronization system over CAN could work and be constructed. While it does not have the same scale of precision or complexity of IEEE 1588, or even the entire CANopen stack, it accomplishes the task of making the basic principles available to Academic prototypers, and embedded developers on a small scale.

With further development work—RTC integration, multi-node capacity, or delay compensation—it could become a more sophisticated option for semi-critical and testing deployments.

REFERENCES

[1]. IEEE 1588-2019 Standard: Precision Clock Synchronization
[2]. CAN in Automation (CiA) 301 – Application Layer and Communication Profile
[3]. ISO 11898-4 – Time-Triggered Communication on CAN (TTCAN)
[4]. Microchip AN2280 – Using CAN for Time Synchronization
[5]. NXP Semiconductors – CANopen Synchronization Application Notes