Time Synchronization Between Two ESP32 Devices Over Wi-Fi Using a PTP-Like Protocol

Tuan Khang Nguyen
Matrikelnummer: 03788196
Course: IN2060
GitHub Repository

Abstract

This project presents the design and implementation of a time synchronization system between two ESP32 microcontrollers over a Wi-Fi network. The system is inspired by the IEEE 1588 Precision Time Protocol (PTP) and uses a simplified message exchange mechanism to estimate clock offset and network delay. Communication between devices is performed using JSON-formatted messages. A Python-based graphical user interface (GUI) application is developed to log synchronization events and diagnostic information in real time.

1 Introduction

Time synchronization is a fundamental requirement in distributed embedded systems such as wireless sensor networks, robotics, and industrial automation. Accurate synchronization enables coordinated actions, precise timestamping, and reliable data fusion.

Although standard protocols such as Network Time Protocol (NTP) and IEEE 1588 Precision Time Protocol (PTP) provide accurate synchronization, their complexity and overhead may be unsuitable for lightweight embedded systems. This project implements a simplified PTP-inspired protocol for ESP32 devices communicating over Wi-Fi.

2 System Architecture

2.1 Hardware Components

- Two ESP32 development boards
- Wi-Fi network (Computer and two ESP32 are in the same network)
- Host computer running a Python GUI application

2.2 Software Components

- ESP32 firmware written in C++
- Wi-Fi communication using UDP sockets

- JSON-based message encoding
- Python GUI application for logging and monitoring

3 Precision Time Protocol (PTP)

3.1 Overview

The IEEE 1588 Precision Time Protocol (PTP) is designed to synchronize clocks in a network with high accuracy, often achieving sub-microsecond precision. PTP operates using a master-slave architecture, where a master clock distributes time information to one or more slave clocks.

The primary goals of PTP are:

- Estimation of clock offset between master and slave
- Estimation of network propagation delay

3.2 PTP Message Exchange

A typical two-step PTP synchronization cycle uses four messages:

1. **Sync**: Sent by the master to mark the start of synchronization.
2. **Follow_Up**: Sent by the master with the exact timestamp of the Sync transmission.
3. **Delay_Req**: Sent by the slave to measure path delay.
4. **Delay_Resp**: Sent by the master containing the receive timestamp of Delay_Req.

Using these messages, the slave can compute both the clock offset and the network delay.

4 Custom PTP-Like Protocol Design

4.1 Message Types

The protocol defines the following message types:

- **Sync (1)**: Initiates synchronization
- **Follow_Up (2)**: Contains precise Sync timestamp
- **Delay_Req (3)**: Delay measurement request
- **Delay_Resp (4)**: Delay measurement response
- **Log_Msg (5)**: Diagnostic and logging message
- **Start_PTP (6)**: Command to start synchronization

4.2 Message Structure

Listing 1: PTP Message Structure

```
enum class PTP_MSG_TYPE : uint8_t
{
  Sync = 1,
  Follow_Up = 2,
  Delay_Req = 3,
  Delay_Resp = 4,
  Log_Msg = 5,
  Start_PTP = 6,
};

struct PtpMsg
{
  PTP_MSG_TYPE type;
  int64_t time;
  String message;
};
```

Messages are serialized into JSON format to ensure readability and cross-platform compatibility.

5 Time Synchronization Algorithm

The synchronization process follows a simplified two-step PTP mechanism:

1. Master sends Sync message
2. Master sends Follow_Up with transmission timestamp t_1
3. Slave sends Delay_Req at timestamp t_3
4. Master responds with Delay_Resp containing timestamp t_4
5. Slave computes offset and delay

The clock offset θ and network delay d are computed as:

$$\theta = \frac{(t_2 - t_1) + (t_3 - t_4)}{2} \tag{1}$$

$$d = (t_4 - t_1) - (t_3 - t_2) \tag{2}$$

6 Python GUI Logging Application

The Python GUI application listens for `Log_Msg` packets over Wi-Fi and displays synchronization events in real time. The GUI provides timestamped logs, visual status indicators, and optional log file storage.

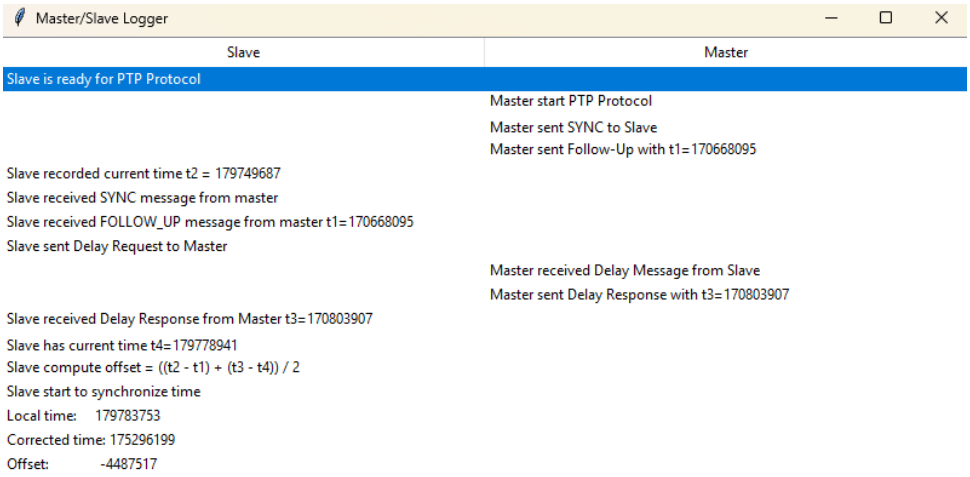

Figure 1: PTP message exchange timing diagram

7 Results and Discussion

Experimental results demonstrate millisecond to sub-millisecond synchronization accuracy under typical Wi-Fi conditions. While Wi-Fi introduces latency and jitter, the PTP-inspired approach provides sufficient accuracy for many embedded applications.

8 Limitations and Future Work

Limitations include sensitivity to Wi-Fi jitter, lack of authentication, and support for a limited number of nodes. Future work may include clock drift compensation, security mechanisms, and binary message encoding.

9 Conclusion

This project demonstrates a practical and lightweight implementation of a PTP-like time synchronization protocol for ESP32 devices over Wi-Fi. The combination of JSON messaging and a Python GUI enhances transparency and extensibility, making the system suitable for prototyping and educational use.

Project Repository

The source code and documentation for this project are available on GitHub:

`https://github.com/tuankhang2003/time-synchronization.git`

References

[1] IEEE Instrumentation and Measurement Society, *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std 1588-2008.

[2] Espressif Systems, *ESP32 Technical Reference Manual*, 2023.