

Алгоритми та методи отримання показів точного часу в електронних  
вбудованих системах

Шифр: LONGINES

## **Список скорочень**

PTP — Precision Time Protocol

RTC — Real-Time Clock

NTP — Network Time Protocol

ПТС — Пристрій тривалого спостереження

EtherCAT — Ethernet for Control Automation Technology

## Зміст

Вступ	4
1 Види часу та аналіз рішень для його синхронізації у вбудованих системах	6
1.1 Вбудовані системи та покази реального часу	6
1.2 Джерела точного часу. Атомні годинники	8
1.3 Служби контролю часу та точки відліку	10
1.4 Методи розповсюдження всесвітнього координаційного часу	10
1.5 Відмінності між локальною синхронізацією вбудованих систем та значенням реального часу	12
2 Прикладна частина	18
2.1 Опис приладу тривалого спостереження	18
2.2 Синхронізація часу між джерелом часу та ПТС	20
Висновки	25
Перелік посилань	26

## Вступ

**Наукова новизна** дослідження полягає в комплексному аналізі та практичній реалізації методів синхронізації часу в промислових та вбудованих системах із високою точністю, зокрема запропоновано використання програмного RTC як альтернативи апаратним годинникам у системах, де їх точність або наявність обмежена, реалізовано адаптацію протоколу RTP для мережових промислових транспортних систем. Синхронізація часу на рівні програмного забезпечення може ефективно компенсувати обмеження вбудованих систем і забезпечити точність, близьку до апаратних рішень, відкриваючи можливості для створення більш гнучких і економічних систем управління та моніторингу у мережах з різнорідними пристроями.

**Практичне значення роботи:** показано проблеми точної синхронізації часу в системах з апаратними обмеженнями та експериментально продемонструвало недоліки системи, яка працює без поступової синхронізації за алгоритмом майстер-слейв, що підтверджує необхідність використання програмних рішень і поступових методів узгодження часу для досягнення високої точності.

Людство з початку свого розвитку прагнуло до всебічної автоматизації процесів. Це підвищувало якість продукції, швидкість її виготовлення та повторюваність результатів.

Для автоматизації на сучасних підприємствах використовуються різні вбудовані системи, зокрема мікроконтролери, які характеризуються високою швидкодією та гнучкістю. Однак постає проблема синхронізації різних елементів системи.

Одним із рішень є використання провідної лінії між усіма пристроями. Лінія може мати довжину кілька кілометрів, і час реакції між першим та останнім елементом залежатиме від топології мережі та навантаження повідомленнями.

Для забезпечення точної синхронізації необхідно використовувати локальні годинники на кожному виконавчому вузлі, синхронізовані з опорним годинником перед початком роботи.

У цій роботі розглянуто стандарти синхронізації часу між вбудованими системами, особливості організації мереж та алгоритм синхронізації часу між модулями тривалого спостереження для коректного логування параметрів.

## **1 Види часу та аналіз рішень для його синхронізації у вбудованих системах**

### **1.1 Вбудовані системи та покази реального часу**

При побудові складної системи, що повинна працювати синхронно, спираючись на покази реального часу, то постає питання – на базі якої обчислювальної одиниці будувати ці системи.

При першому погляді може здатись, що такі функції можуть виконувати персональні комп'ютери загального призначення, сервери, або механічні системи – і так це можливо, але ефективність залишається під питанням.

Механічні системи можна одразу викреслювати зі списку, оскільки вони не мають адаптивного керування, побудова складної логіки вимагає великих геометричних розмірів цієї системи. В момент, коли потрібно буде змінити принцип обробки даних – доведеться будувати нову систему з початку.

Сервери використовуються для обробки великих обсягів даних, мають високу обчислювальну потужність, вартість, та споживають багато потужності. В будові систем реального часу їх недоцільно використовувати в якості основної обчислювальної ланки та арбітра виконавчих пристроїв, датчиків, тощо. Найкраще їх застосування, це обробка і зберігання даних, що поступають від основних обчислювальних елементів системи.

Персональні комп'ютери загального призначення мають ті самі вади, що і сервери. Висока обчислювальна потужність, великі розміри, вартість. Ще одним недоліком, як серверів, так і ПК є архітектура процесорів – CISC (Complex Instruction Set Computer). Дана архітектура має повний перелік команд, команди мають різну довжину, що негативно впливає на швидкість роботи.

Нішу основних обчислювальних одиниць в системах реального часу зайняли мікроконтролери, що є одними з варіантів вбудованих систем. Вони мають вищу швидкість роботи порівняно з процесорами загального призначення. Головним фактором тут є архітектура RISC (Reduced instruction set computer),

вона має скорочений список інструкцій, а самі інструкції мають однакову довжину. Геометричні розміри, як самих мікроконтролерів, так допоміжної обв'язки значно менші ніж у ПК загального призначення. Вбудовані системи також мають безліч інтерфейсів комунікації на вибір, що дозволяє побудувати систему не обмежуючись інтерфейсом Ethernet для ПК чи серверів. Також мікроконтролери мають високу адаптивність та гнучке налаштування.

Мікроконтролери використовуються не тільки в системах реального часу десь на підприємствах, а в оселях людей – це більшість побутових приладів, таких як пральні машини, телевізори, праски, тощо. Ці прилади мають лише внутрішню синхронізацію. Наприклад робота мікроконтролера всередині холодильника включає вимірювання температури всередині камери холодильника і регулюванні часу роботи компресора для підтримання цієї температури. Для коректної роботи необхідно лише налаштувати часові інтервали ввімкнення через вбудовані таймери МК.

Показання реального часу і синхронізація по ним важливі в промисловості для роботи обладнання, наприклад конвеєрних ліній, що повинні виконувати дії в певній послідовності, у відведенні часові проміжки, порушення яких призведе до виходу з ладу обладнання, травмам або загибелі людей.

Широке застосування вбудовані системи отримали в альтернативні енергетиці. В цій галузі важливо виконувати виміри багатьох параметрів, таких як вологість, температура, повітряний тиск. Вимір кожного з цих параметрів, може відбуватись не у рівномірно віддалені проміжки часу і при цьому бути прив'язаний до показів реального часу, що дозволить побудувати графіки залежності та провести поглиблений аналіз. Результатом аналізу можуть бути внесені корегувальні коефіцієнти в роботу виконавчих механізмів (поворотних механізмів сонячних панелей чи вітряків, кутів відкриття шлюзів ГЕС, тощо). Виконавчими механізмами керує також мікроконтролер.

Отже, вбудовані системи – це основні обчислювальні пристрої в системах реального часу. Основними перевагами є достатня обчислювальна потужність,

адаптивність, низька вартість та малі геометричні розміри. Вони здатні до високого рівня синхронізації, розподілення та формування чіткої послідовності дій.

## **1.2 Джерела точного часу. Атомні годинники**

Людство з давніх часів намагалось визначати час. Спочатку це було визначення за положенням сонця, а точніше тінями, що кидали предмети під його впливом. Місяцем та зорями. Але всі ці способи об'єднувало одне – періодичність. Вони всі відбувались з певною частотою і по ним було можливо визначати час, не точний, але достатній на той час.

З переходом до напівпровідників, в якості джерела тактування почали використовувати кварцові резонатори. Їх перевагами були більша температурна стабільність ніж у RC та LC кіл. Вони мають досить широкий діапазон частот. Але вони не настільки точні, як атомні годинники.

Атомарні годинники – прилади, що використовують коливання атомів, для визначення точних періодів часу. Коливанням атома вважають перехід від одного енергетичного рівня до іншого. Частоту коливань атома визначають за формулою:

$$\nu = \frac{E_2 - E_1}{h}$$

де:  $\nu$  – частота, Гц;

$E_1, E_2$  – енергетичні рівні;

$h$  – стала Планка.

Перші атомні годинники (рис. 1.1) використовувати цезій-133 в якості атома. Одна секунда дорівнювала 9 192 631 770 коливань атома.



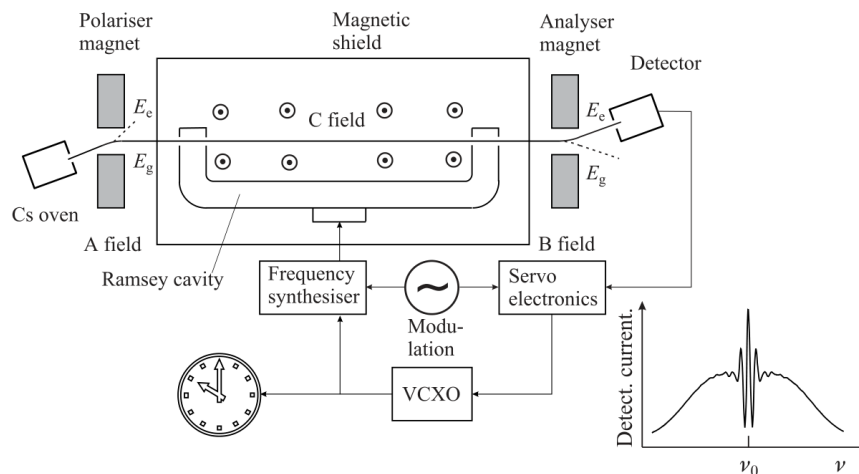


Рисунок 1.1 – Атомний годинник на основі цезію[4]

Цезій підігрівали в печі. Атоми мали змогу вилітати через вузький отвір на виході з печі і потрапляли на поляризаційний магніт, що відсіював атоми заряджені до вищого енергетичного рівня. Далі атоми проходили першу пластину детектору і вмикали вимірювання. Рух через камеру, зменшував енергію атомів і при проходженні вихідного магніту вони вимикали вимірювання. Цим самим створюючи точний відлік часу. Далі залишалось лише обробити дані та відкоригувати похибку.

Сучасним комерційним стандартом в сфері атомних годинників є рубідієвий газовий резонатор. Він має невелику потужність споживання, високу точність. Даний тип годинників широко використовується в лабораторіях, телефонній комунікації, високоточних лічильниках частоти. [4]

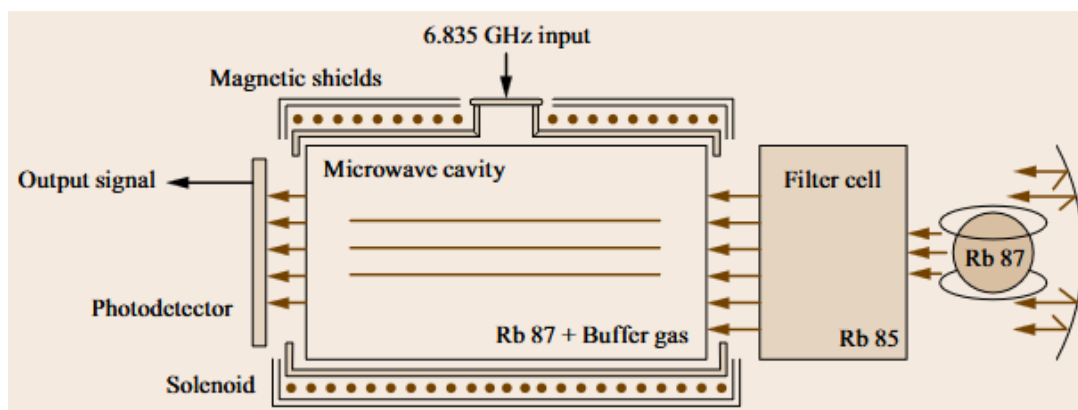


Рисунок 1.2 – Рубідієвий газовий резонатор[4]

### **1.3 Служби контролю часу та точки відліку**

Міжнародне бюро часу (МБЧ) – організація утворена Міжнародним астрономічним союзом для координації національних служб часу та обчислення всесвітнього часу. Бюро розташовувалось в Парижі.

Основним завданням МБЧ було узгодження і обчислення всесвітнього часу за астрономічними спостереженнями різних обсерваторій та поширення даних часу у світі. В 1988 році МБЧ було реорганізоване в Міжнародне бюро мір і ваг.

Всесвітній час (UT – universal time) – шкала сонячного часу, що вимірюється за кутом середнього Сонця відносно гринвіцького меридіана.

Всесвітній координаційний час (UTC – coordinated universal time) – стандарт, за яким суспільство визначає час. Він синхронізується за UT1, що враховує рух полюсів землі. UTC не може відрізнитись від UT1 більше ніж на 0,8 с.

Абсолютною величиною часу є секунда, що визначається за допомогою Всесвітнього атомарного часу (TAI – Temps Atomique International) і атомних годинників. Точку відліку атомарного часу обрано так, щоб він збігався з всесвітнім часом UT2 від дати 1 січня 1958 року.

В подальшому, при написанні цієї роботи під поняттям час буде вважатись саме UTC оскільки він є загальноприйнятим у повсякденному використанні.

### **1.4 Методи розповсюдження всесвітнього координаційного часу**

Служби, що надають час, також займаються його розповсюдженням по світу. На початку свого існування, вони робили це за допомогою радіо, з мінімальним врахуванням затримок. На сьогоднішній день, синхронізація часу

відбувається через різні мережі, за допомогою спеціалізованих сервісів і протоколів.

Основним протоколом для передачі часу є NTP (Network Time Protocol). Його позиціонують, як стійкий транспортний механізм. Побудований над протоколом UDP (User Datagram Protocol). Протокол визначає формат повідомлень та розрахунок похибки часу при транспортуванні.

Робота протоколу виглядає наступним чином. Клієнт надсилає запит на час у сервера і в цьому запиті залишає свій часовий штамп моменту відправлення. Сервер відправляє відповідь додаючи свій часовий штамп. При отриманні клієнт записує час отримання і за чотирма часовими відліками розраховує часову затримку передачі пакетів. Точність протоколу досягає декількох наносекунд.

NTP сервери – це джерела часу, при запитах на які можна отримати всесвітній координатний час. Вони мають декілька рівнів точності, що мають назви Stratum1, Stratum2 і т.д.. Перший рівень – сервери, що синхронізуються з еталонними годинниками. Другий і Третій рівні синхронізуються ієрархічно один з одним.

На території України діють наступні онлайн сервери, що побудовані на основі NTP:

- time.in.ua – сервер, що надає перший та другий рівень точності;
- ntp.org.ua – надавав послуги з синхронізації годинників, але на даний момент більшість серверів недоступні;
- ua.pool.ntp.org – надає пул з чотирьох серверів на вибір, для синхронізації. Вказівки про рівень точності відсутні.

Сервер time.in.ua поки єдиний в Україні, що надає синхронізацію з сервера першого рівня точності – еталонного годинника. В якості джерела використовують супутники з технологією GPS.

Гіршою альтернативою до NTP є HTTP запити. Робота побудована по типу клієнт-сервер. Недоліками є обрахунок затримок, час самих затримок більший.

## **1.5 Відмінності між локальною синхронізацією вбудованих систем та значенням реального часу**

Вбудована система, як мікроконтролер, має в своєму складі ядро, периферійні пристрої та джерело тактування для синхронізації роботи всередині себе. Найчастіше – це кварцовий резонатор. Таким чином досягається потрібна тактовність роботи на рівні одного пристрою.

На виробництвах таких пристроїв десятки, сотні чи тисячі. І всі вони повинні мати синхронізацію не тільки всередині себе, а всією системою.

Наприклад, на фабриці з виробництва газет встановлені різні станки, що підключені до однієї мережі. Якщо робота всіх станків не буде синхронізована, то партії товару буде зіпсовано.

Система може бути синхронізована по еталонному годиннику. Кожен з пристроїв відправить запит в мережу і отримує час. Цей час буде відрізнятись на затримку отримання інформації (непередбачувану величину).

Тому, доцільно використовувати синхронізацію не всієї системи одночасно, а поступову синхронізацію всіх її елементів, що враховуватиме персональні затримки.

Precision Time Protocol (PTP) IEEE-1588 – протокол для синхронізації пристроїв між собою. Він досягає точності менше однієї мікросекунди, чого достатньо для систем керування.

Топологія (рис. 1.3) взаємодії побудована за типом Master-Slave. Головний годинники надсилає сигнали синхронізації до прикордонних (Boundary) годинників, а вони в свою чергу відсилають до підпорядкованих годинників. Таким чином синхронізація відбувається поступово, з усіма елементами мережі.

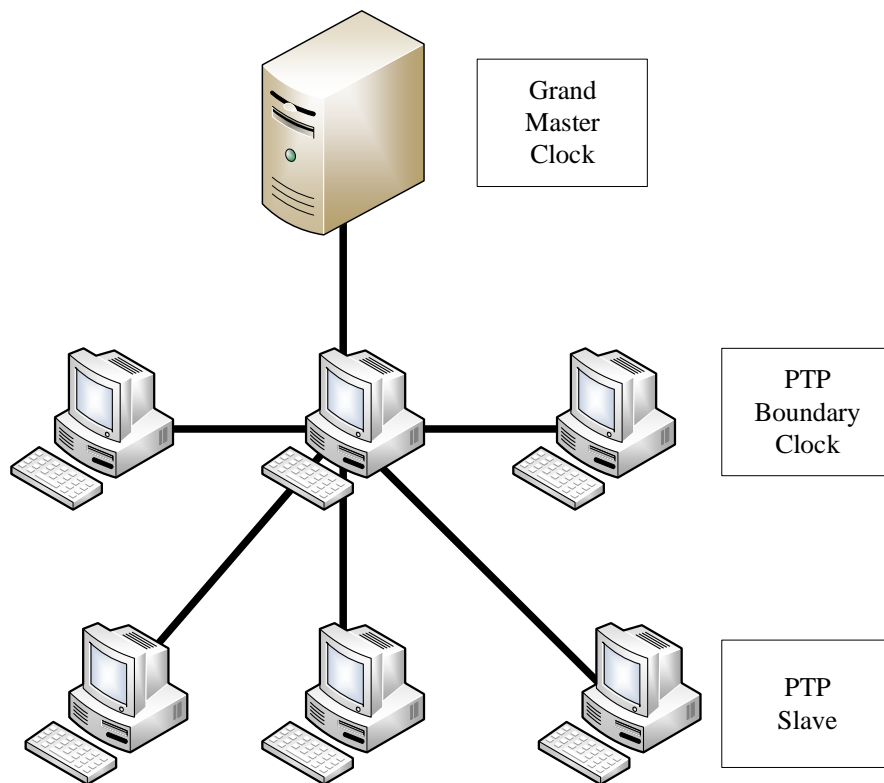


Рисунок 1.3 – Топологія взаємодії по протоколу РТР

Порядок проведення синхронізації:

1. Майстер запит на синхронізацію і час свого опорного годинника( $t_1$ );
2. Слейв отримує повідомлення на синхронізацію та фіксує час отримання по своєму годиннику ( $t_2$ );
3. Слейв відправляє відповідь з затримкою( $t_3$ );
4. Майстер відправляє відповідь з часом отримання повідомлення від слейва( $t_4$ );
5. Слейв розраховує зміщення на час затримки.

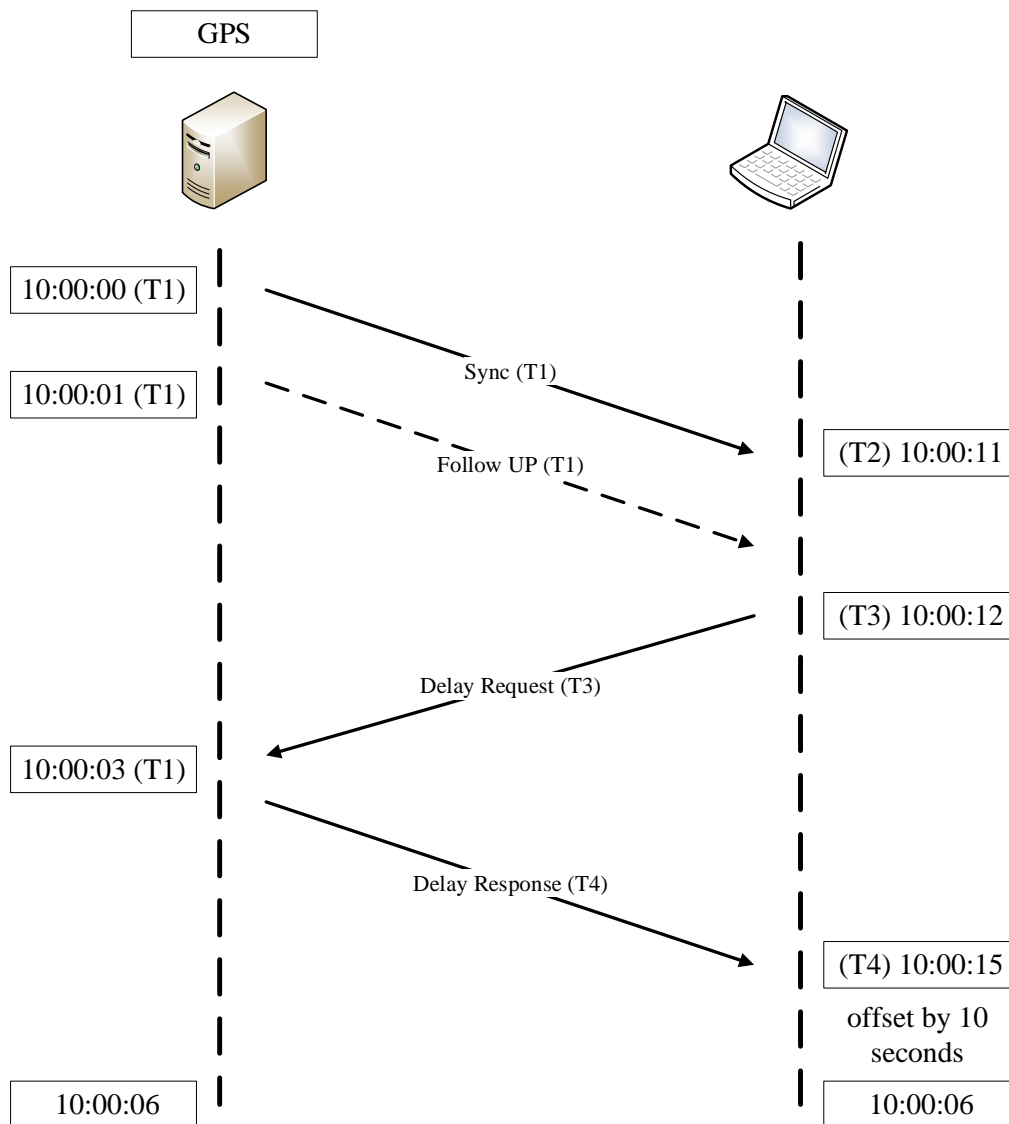


Рисунок 1.4 – Схема синхронізації

Протокол має досить простий алгоритм для синхронізації та високу точність. Може працювати в фоновому режимі. Але постає наступна проблема – топологія мережі і пріоритет повідомлень.

Наприклад, якщо використовувати Ethernet, то всі дані мають в ньому однаковий рівень пріоритету, і проходять однакову кількість рівнів мережевої моделі. В домашньому використанні – це не проблема. Оскільки неважливо прийшли дані о 10:00:01 чи 10:00:10, а в системах реального часу – це великий термін.

В системах реального часу використання низько ефективних інтерфейсів може призводити до втрати синхронізації. Для вирішення цієї проблеми було створено EtherCAT – Ethernet for Control Automation Technology. На рисунку 1.5 зображено загальну мережеву модель та мережеву модель EtherCAT. Як можна побачити в EtherCAT йде розділення на типи даних, що дозволяє швидше опрацьовувати інформацію з вищим пріоритетом – інформацію реального часу. Інші дані циркулюють по скороченій мережевій моделі, пропускаючи представницький та сеансовий рівні.[2]

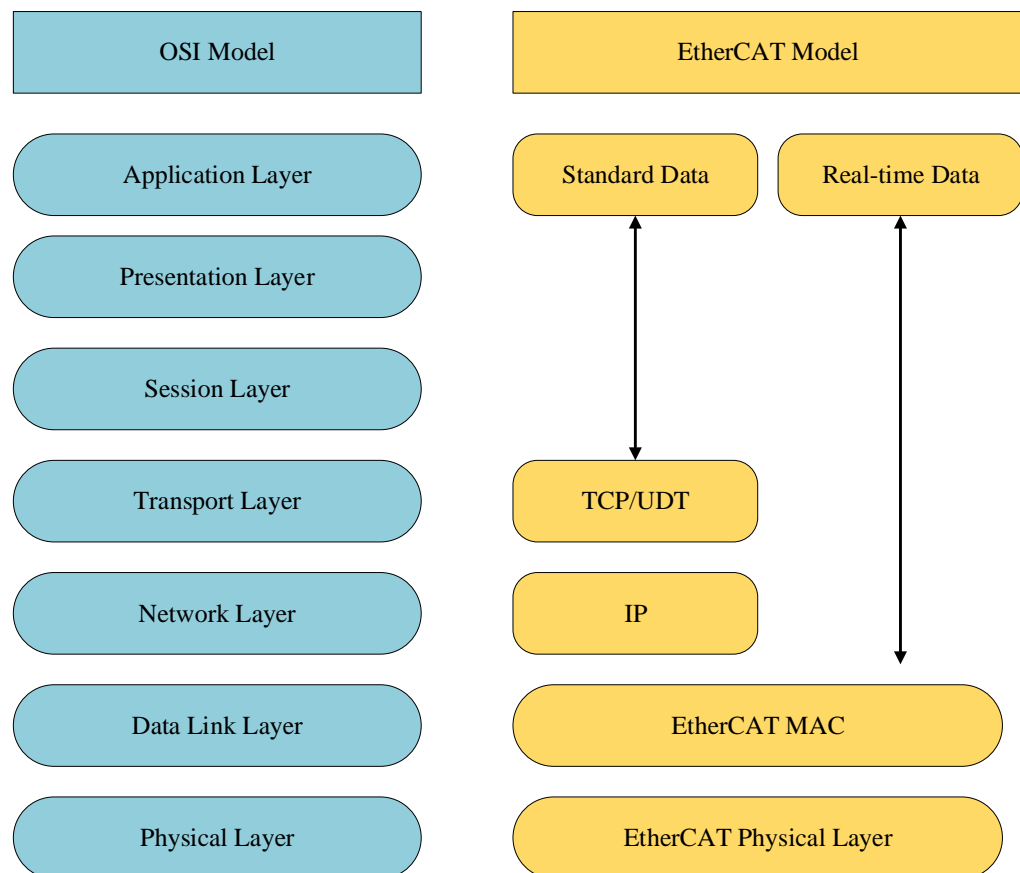


Рисунок 1.5 – Мережеві моделі OSI та EtherCAT

Другою перевагою EtherCAT є взаємодія між елементами мережі. Він використовує принцип Token ring. Початкового Token ring – це мережа з кільцевою топологією і детермінованим методом доступу, що заснований на передачі маркерного повідомлення – токена. Але EtherCAT підтримує всі

топології, що підтримує Ethernet і не прив'язаний до кільцевої. Основний принцип побудований на взаємодії майстер-підлеглий. [2]

Майстер виконує роль арбітра. Він ініціює передачу даних шляхом відправки маркерного пакету. Підлеглий, якому призначений цей пакет, модифікує його і відправляє далі, допоки він не прийде за адресом де буде модифікований і відправлений далі по мережі. Останній підлеглий поверне пакет до майстра.

Особливостями є те, що пакет може бути модифікований не одним підлеглим за одне коло. Тобто, пакет тримає дані від декількох до декількох підлеглих. Також підлегли пристрої не виконують операції читання та запису даних одна за одною, а роблять це одночасно.

Розподілені обов'язки, детермінований обмін даними між всіма елементами системи робить її передбачуваною, та дозволяє більш чітко визначати часові інтервали всередині системи.

Третя перевага – це стійкість до пошкоджень лінії. Якщо існує хоча б один шлях між усіма елементами мережі – то повідомлення прийде крізь мережу.

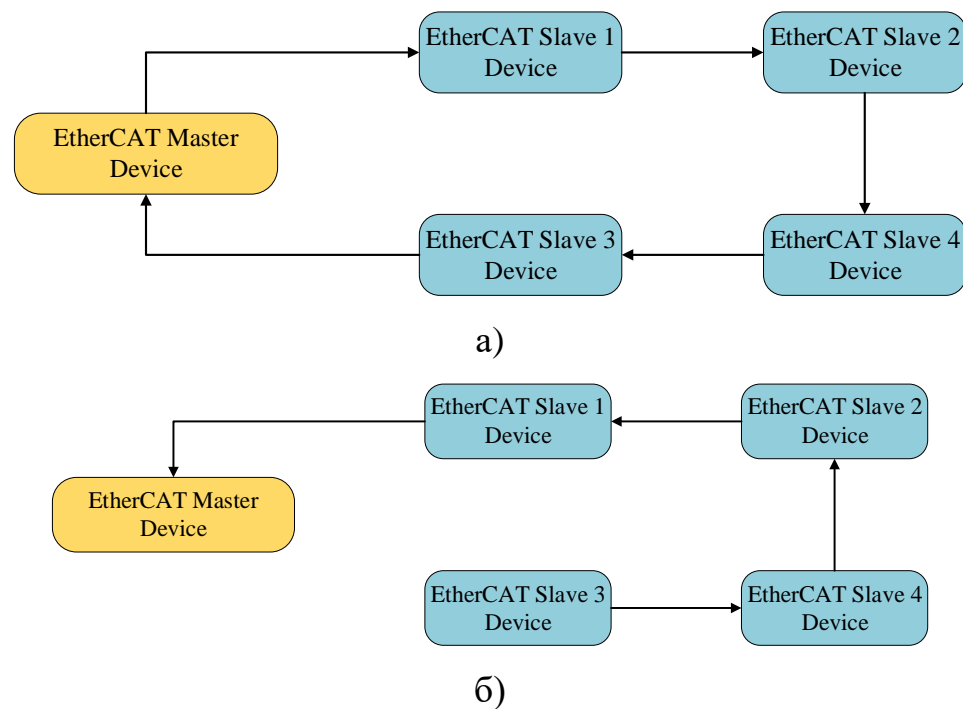


Рисунок 1.6 – Приклад обміну в кільцевій топології EtherCAT: а) стандартна шлях; б) шлях повернення токена при пошкодженні лінії



Для синхронізації часу в мережі EtherCAT використовують опорний годинник. Його значенням може бути локальний час сервера, головного ПК, тощо.

Перед початком роботи системи майстер зчитує час опорного годинника, та відправляє ширококомвні пакети по всій мережі для синхронізації часу між всіма підлеглими. Комунікація і розрахунок затримок відбувається окремо між майстром і кожним з підлеглих.

Впродовж роботи системи майстер відстежує час кожного підлеглого, локальний час отримання і відправки токена. Якщо джитер шум виходить за допустимі значення, майстер може ініціювати синхронізацію часу повторно.

## 2 Прикладна частина

### 2.1 Опис приладу тривалого спостереження

Для тестування та побудови алгоритму локальної синхронізації було зібрано тестовий стенд, пристрій тривалого спостереження (рис. 2.1).

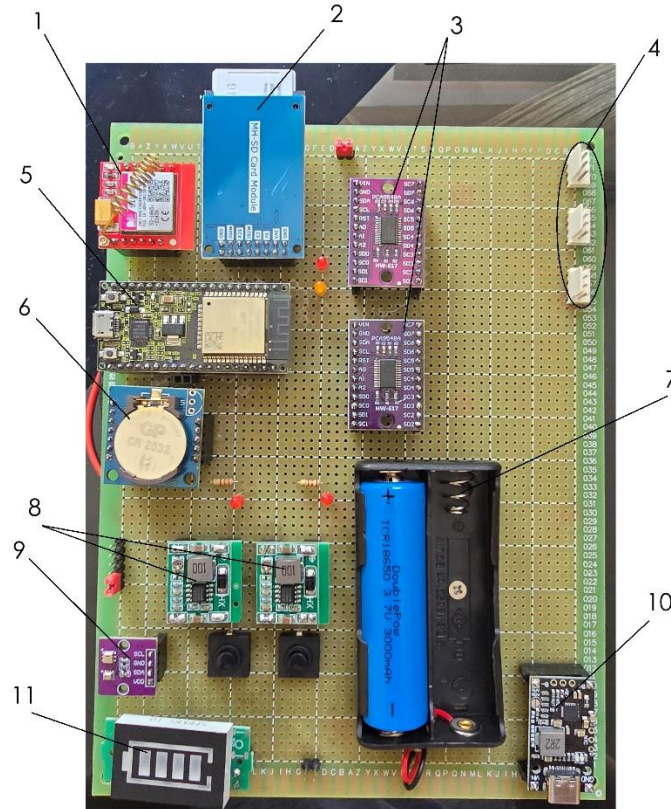


Рисунок 2.1 – Пристрій тривалого спостереження: 1 – GSM модуль; 2 – SD-Card холдер; 3 – I2C мультиплексори; 4 – Роз’єми для підключення зовнішніх сенсорів; 5 – Мікроконтролер; 6 – зовнішній RTC; 7 – відсік для акумуляторів; 8 – DC-DC перетворювачі; 9 – Внутрішній датчик температури та вологості; 10 – зарядний пристрій для акумуляторів; 11 – індикатор заряду

В якості головної обчислювальної одиниці був використаний мікроконтролер ESP-32, через наступні переваги: низьке споживання в режимі глибокого сну, наявність режиму глибокого сну, підтримка безпроводних інтерфейсів (Wi-Fi, BLE).

Подібні пристрої використовуються для тривалого спостереження за кліматичними, сейсмічними, оптичними параметрами. В залежності від встановлених сенсорів. Для коректного документування подій, всі пристрої повинні бути синхронізовані між собою та опорним годинником.

Пристрій представлений в єдиному екземплярі, тому для побудови алгоритму синхронізації будемо використовувати локальний годинник на ESP32 та одне доступних джерел часу.

В таблиці 2.1 наведено таблицю логування часових штампів з різних джерел: NTP – це прямий доступ з ntp сервера; RTC – годинник реального часу, що встановлений на зовнішній друкованій платі; GSM – дані отримані з NTP сервера при підключенні через мобільного оператора.

По-перше, в таблиці можна побачити, що всі джерела розсинхронені між собою, бо час доступу до кожного з джерел різний. Найшвидший доступ до NTP сервера при підключенні по локальній мережі. Найдовший час доступу до часу через мобільного оператора.

По-друге, логування велось з затримкою 10 с між записами. Без врахування затримок доступу до джерел часу. Тому проявилась проблема затримки розповсюдження: інтервал між записами був не 10с., а з додаванням затримок доступу до кожного з джерел, що склала близько 17 мс. та часу запису на карту пам'яті.

Таблиця 2.1 – Часові відліки без синхронізації джерел

NTP Date	NTP Time	NTP Delay	RTC Date	RTC Time	RTC Delay	GSM Date	GSM Time	GSM Delay
2025.6.12	9.55.42	626	2025.6.12	9.56.56	1165	2025.6.12	9.55.32	14678
2025.6.12	9.55.53	676	2025.6.12	9.57.6	1163	2025.6.12	9.55.42	14699
2025.6.12	9.56.3	626	2025.6.12	9.57.16	1165	2025.6.12	9.55.52	14685
2025.6.12	9.56.13	624	2025.6.12	9.57.26	1163	2025.6.12	9.56.2	14704
2025.6.12	9.56.23	626	2025.6.12	9.57.36	1165	2025.6.12	9.56.12	14682
2025.6.12	9.56.33	624	2025.6.12	9.57.46	1163	2025.6.12	9.56.22	14810

2025.6.12	9.56.43	626	2025.6.12	9.57.56	1165	2025.6.12	9.56.32	14681
-	-	-	-	-	-	-	-	-
2025.6.12	9.59.4	626	2025.6.12	10.0.16	1165	2025.6.12	9.58.53	14686
2025.6.12	9.59.14	674	2025.6.12	10.0.26	1163	2025.6.12	9.59.3	14689
2025.6.12	9.59.24	626	2025.6.12	10.0.36	1165	2025.6.12	9.59.13	14681
2025.6.12	9.59.34	624	2025.6.12	10.0.47	1163	2025.6.12	9.59.23	14741
2025.6.12	9.59.44	674	2025.6.12	10.0.57	1165	2025.6.12	9.59.33	14707
2025.6.12	9.59.54	624	2025.6.12	10.1.7	1163	2025.6.12	9.59.43	14742

## 2.2 Синхронізація часу між джерелом часу та ПТС

З синхронізацією часу між джерелом часу та ПТС є проблема відсутності апаратної підтримки RTP протоколу з боку ПТС та з боку мережі до якої підключений пристрій. Оскільки для отримання даних часу використовується підключення по інтерфейсу Wi-Fi до локальної мережі і NTP сервер, або через мережу мобільного оператора теж до NTP сервера, що не підтримують на апаратному рівні роботу через RTP протокол. Також ESP-32 не має MAC адреса, і всі запити повинні відбуватись через UDP. В теорії, якби ESP-32 була підключена до мережі з підтримкою RTP, то при зверненні на порти цього протоколу можна було отримати точні часові штампи. Але опрацювати їх з точністю, що дає цей протокол, з боку ПТС не можливо.

Тому, синхронізація між ПТС та джерелом часу буде відбуватись при підключенні до NTP сервера. NTP підтримує синхронізацію часу з урахуванням затримок. Точність гарантується на рівні 1 мс. [8] для локальних мереж. Для синхронізації опорного годинника ПТС цього буде достатньо.

Для синхронізації на NTP сервер будуть відправлені запити синхронізації. В результаті комунікації отримаємо чотири часові штампи і проведемо розрахунок за нижче наведеною формулою.

Формула для розрахунку затримки:

$$offset = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \quad (2.1)$$

де: t1, t2, t3, t4 – це часові штампи.

Таблиця 2.2 – Дані для прикладу розрахунку

Часовий штамп	Опорний годинник	Прилад тривалого спостереження
t1	19:55:10	-
t2	-	00:00:30
t3	-	00:00:31
t4	19:55:11	-

$$offset = \frac{(00:00:30 - 19:55:10) - (19:55:11 - 00:00:31)}{2} = -19:55:10$$

Значення Offset потрібно відняти від часу слейва, і встановити час опорного годинника.

### 2.3 Синхронізація часу між зовнішнім RTC та іншими ПТС

Для зберігання значень часу необхідно або використовувати RTC внутрішній або зовнішній.

В якості зовнішнього RTC використано DS1307 – простий годинник реального часу, що не має складної логіки комунікації. Мікросхема має внутрішній календар та декілька регістрів, що доступні для читання та запису. Мінімальна одиниця часу – секунда. Не підтримує протоколи синхронізації часу. Він не може самостійно вирахувати затримку комунікації та встановити час. Це необхідно виконати з боку майстра, що ініціює комунікацію.

Тому доцільно використовувати внутрішній RTC, якщо він доступний, або програмно реалізувати годинник.

З ESP-32 саме такий випадок. МК не має внутрішнього таймера, тому необхідно програмно реалізувати таймер реального часу.

Приклад простої програмної реалізації таймеру реального часу:

```
typedef struct
{
    uint8_t sec;
    uint8_t min;
    uint8_t hour;
    uint8_t day;
    uint8_t week;
    uint8_t year;
} soft_rtc_t;

hw_timer_t *rtcTimer = NULL;
soft_rtc_t soft_rtc;

void soft_rtc_init ()
{
    rtcTimer = timerBegin(1, 80, true); // запуск таймера
    timerAttachInterrupt(rtcTimer, &onRtcTick, true); // визначення обробника
переривання
    timerAlarmWrite(rtcTimer, 1000000, true); // час виклику обробника переривання
    timerAlarmEnable(rtcTimer); //дозвіл переривання
}

void IRAM_ATTR onRtcTick() //обробник переривання
{
    portENTER_CRITICAL_ISR(&timerMux);
    soft_rtc.sec++;

    if (soft_rtc.sec >= 60)
    {
        soft_rtc.sec = 0;
        soft_rtc.min++;
        if (soft_rtc.min >= 60)
        {
            soft_rtc.min = 0;
            soft_rtc.hour++;
            if (soft_rtc.hour >= 24)
            {
                soft_rtc.hour = 0;
                soft_rtc.day++;
            }
        }
    }
    portEXIT_CRITICAL_ISR(&timerMux);
}
```

Основною перевагою програмної реалізації є гнучкість налаштування.

Синхронізація часу між декількома ПТС відбувається на основі NTP алгоритму з чотирма часовими штампами. Основною відмінністю є інтерфейс комунікації. Всі ПТС можуть бути об'єднані між собою через будь-який

доступний інтерфейс: SPI, UART, I2C. Все залежить від особливостей середовища використання і відстані. Наприклад RS485 через UART може передавати дані на відстань до 1200 м і забезпечує достатню завадостійкість.

### 2.3 Результати після впровадження синхронізації часу

Після впровадження алгоритму синхронізації часу було зняти лог з часовими даними (табл. 2.3).

Час з NTP сервера та час програмного RTC збігається, але потрібно провести довгий період вимірювань додатково. А час з зовнішнього RTC відрізняється, оскільки при встановленні не відбувається синхронізація. Зовнішній модуль не підтримує алгоритм синхронізації.

Таблиця 2.3 – Часові відліки з алгоритмом синхронізації часу

NTP Date	NTP Time	NTP Delay	RTC Date	RTC Time	RTC Delay	Soft_RTC Time
9/14/2025	17:42:00	637	9/14/2025	17:41:59	1182	17:42:00
9/14/2025	17:42:10	623	9/14/2025	17:42:09	1182	17:42:10
9/14/2025	17:42:20	635	9/14/2025	17:42:19	1181	17:42:20
9/14/2025	17:42:30	630	9/14/2025	17:42:29	1180	17:42:30
9/14/2025	17:42:40	637	9/14/2025	17:42:39	1180	17:42:40
9/14/2025	17:42:50	624	9/14/2025	17:42:49	1189	17:42:50
9/14/2025	17:43:00	637	9/14/2025	17:42:59	1182	17:43:00
9/14/2025	17:43:10	624	9/14/2025	17:43:09	1179	17:43:10
9/14/2025	17:43:20	637	9/14/2025	17:43:19	1182	17:43:20
9/14/2025	17:43:30	623	9/14/2025	17:43:29	1181	17:43:30
9/14/2025	17:43:40	644	9/14/2025	17:43:39	1192	17:43:40
9/14/2025	17:43:50	632	9/14/2025	17:43:49	1194	17:43:50
9/14/2025	17:44:00	637	9/14/2025	17:43:59	1182	17:44:00
9/14/2025	17:44:10	625	9/14/2025	17:44:09	1179	17:44:10
9/14/2025	17:44:20	637	9/14/2025	17:44:19	1182	17:44:20
9/14/2025	17:44:30	623	9/14/2025	17:44:29	1182	17:44:30
9/14/2025	17:44:40	637	9/14/2025	17:44:39	1182	17:44:40
9/14/2025	17:44:50	625	9/14/2025	17:44:49	1216	17:44:50
9/14/2025	17:45:00	637	9/14/2025	17:44:59	1181	17:45:00
9/14/2025	17:45:10	624	9/14/2025	17:45:09	1183	17:45:10
9/14/2025	17:45:20	636	9/14/2025	17:45:19	1181	17:45:20
9/14/2025	17:45:30	625	9/14/2025	17:45:29	1185	17:45:30

9/14/2025	17:45:40	662	9/14/2025	17:45:39	1192	17:45:40
9/14/2025	17:45:50	624	9/14/2025	17:45:49	1183	17:45:50
9/14/2025	17:46:00	637	9/14/2025	17:45:59	1180	17:46:00
9/14/2025	17:46:10	624	9/14/2025	17:46:09	1184	17:46:10
9/14/2025	17:46:20	637	9/14/2025	17:46:19	1181	17:46:20
9/14/2025	17:46:30	624	9/14/2025	17:46:29	1186	17:46:30
9/14/2025	17:46:40	637	9/14/2025	17:46:39	1180	17:46:40
9/14/2025	17:46:50	631	9/14/2025	17:46:49	1179	17:46:50
9/14/2025	17:47:00	637	9/14/2025	17:46:59	1181	17:47:00
9/14/2025	17:47:10	623	9/14/2025	17:47:09	1179	17:47:10
9/14/2025	17:47:20	637	9/14/2025	17:47:19	1180	17:47:20
9/14/2025	17:47:30	624	9/14/2025	17:47:29	1182	17:47:30
9/14/2025	17:47:40	637	9/14/2025	17:47:39	1184	17:47:40
9/14/2025	17:47:50	626	9/14/2025	17:47:49	1179	17:47:50
9/14/2025	17:47:59	637	9/14/2025	17:47:59	1182	17:48:00
9/14/2025	17:48:00	624	9/14/2025	17:48:00	1183	17:48:00
9/14/2025	17:48:11	636	9/14/2025	17:48:10	1183	17:48:10
9/14/2025	17:48:20	624	9/14/2025	17:48:19	1189	17:48:20
9/14/2025	17:48:30	637	9/14/2025	17:48:29	1181	17:48:30
9/14/2025	17:48:40	625	9/14/2025	17:48:39	1205	17:48:40
9/14/2025	17:48:50	637	9/14/2025	17:48:49	1183	17:48:50
9/14/2025	17:49:00	624	9/14/2025	17:48:59	1181	17:49:00
9/14/2025	17:49:10	637	9/14/2025	17:49:09	1181	17:49:10
9/14/2025	17:49:20	630	9/14/2025	17:49:19	1178	17:49:20
9/14/2025	17:49:30	637	9/14/2025	17:49:29	1181	17:49:30
9/14/2025	17:49:40	658	9/14/2025	17:49:39	1188	17:49:40
9/14/2025	17:49:50	637	9/14/2025	17:49:49	1181	17:49:50
9/14/2025	17:50:00	624	9/14/2025	17:49:59	1189	17:50:00
9/14/2025	17:50:10	638	9/14/2025	17:50:09	1181	17:50:10
9/14/2025	17:50:20	631	9/14/2025	17:50:19	1179	17:50:20
9/14/2025	17:50:30	637	9/14/2025	17:50:29	1181	17:50:30
9/14/2025	17:50:40	649	9/14/2025	17:50:39	1183	17:50:40
9/14/2025	17:50:50	645	9/14/2025	17:50:49	1192	17:50:50
9/14/2025	17:51:00	622	9/14/2025	17:50:59	1187	17:51:00
9/14/2025	17:51:10	637	9/14/2025	17:51:09	1181	17:51:10
9/14/2025	17:51:20	624	9/14/2025	17:51:19	1183	17:51:20
9/14/2025	17:51:30	637	9/14/2025	17:51:29	1182	17:51:30
9/14/2025	17:51:40	624	9/14/2025	17:51:39	1183	17:51:40
9/14/2025	17:51:50	645	9/14/2025	17:51:49	1192	17:51:50
9/14/2025	17:52:00	624	9/14/2025	17:51:59	1179	17:52:00



## **Висновки**

На основі проведених експериментів та аналізу матеріалу можна зробити висновок, що для забезпечення високоточних вимірювань часу між джерелом часу та опорним годинником, відповідно до точності протоколу РТР, необхідна програмна підтримка протоколу на обох кінцях комунікаційного каналу. Тільки при двосторонній реалізації протоколу можлива компенсація затримок та дрейфу годинника, що забезпечує точну синхронізацію в реальному часі.

Важливим аспектом є вибір місця для зберігання актуального часу. Не всі вбудовані системи оснащені апаратним RTC, а доступні зовнішні RTC можуть не забезпечувати необхідну точність для конкретних завдань. У таких випадках доцільним рішенням є використання програмного RTC, який дозволяє підтримувати та коригувати час з достатньою точністю, спираючись на синхронізацію через мережеві протоколи.

Для багатопристроєвих систем ефективним підходом є організація обміну часовими даними за моделлю «майстер–підлеглий». Така архітектура дозволяє поступово коригувати час на всіх вузлах, забезпечуючи узгодженість і точність синхронізації без потреби у високопродуктивних апаратних рішеннях на кожному вузлі.

Реалізація описаних підходів дозволяє створювати високоточні вбудовані та мережеві системи, де синхронізація часу є критичною, наприклад, у телекомунікаційних мережах, системах збору даних, автоматизації та промислових контролерах. Комбінування програмного забезпечення для синхронізації з вибором оптимального способу зберігання часу і правильною архітектурою мережі забезпечує стабільну та точну роботу систем у реальному часі.

## Перелік посилань

1. Stankovic J. A. *Real-Time and Embedded Systems*. — ACM Digital Library, 2000. — Режим доступу: <https://dl.acm.org/doi/pdf/10.1145/234313.234400>
2. Dewesoft. *What is EtherCAT Protocol*. — Режим доступу: <https://dewesoft.com/blog/what-is-ethercat-protocol>
3. Beckhoff Automation. *EtherCAT System Overview*. — Режим доступу: <https://infosys.beckhoff.com/english.php?content=../content/1033/ethercatsystem/2469118347.html&id=>
4. Springer. *Springer Handbook of Global Navigation Satellite Systems*. — Springer, 2017.
5. IPCisco. *Precision Time Protocol (PTP)*. — Режим доступу: <https://ipcisco.com/lesson/precision-time-protocol-ntp/>
6. Network Lessons. *Introduction to Precision Time Protocol (PTP)*. — Режим доступу: <https://networklessons.com/ip-services/introduction-to-precision-time-protocol-ntp>
7. *Renewable Energy Management Using Embedded Smart Systems*. — Elsevier, 2021.
8. TimeTools Ltd. *NTP Timing Accuracy*. — Режим доступу: <https://timetoolsltd.com/ntp/ntp-timing-accuracy/>