КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ
Кафедра радіотехніки та радіоелектронних систем

«На правах рукопису»

Робота допущена до захисту в ЕК
рішенням кафедри радіотехніки та радіоелектронних систем
від 22 травня 2024 року, протокол № ___.
Завідувач кафедри доктор фіз.-мат. наук, професор
_____ Ігор АНІСІМОВ

ДИПЛОМНА РОБОТА МАГІСТРА
на тему:
«СИСТЕМА З ІНДЕНТИФІКАЦІЄЮ ДАТЧИКА ЗА ДОПОМОГОЮ SVD АЛГОРИТМУ ДЛЯ ПРИСКОРЕННЯ ОТРИМАННЯ СИГНАЛУ ТРИВОГИ»

Виконав:
студент 2-го курсу магістратури
денної форми навчання
спеціальності 172 - Телекомунікації та радіотехніка
ОНП «Інформаційна безпека телекомунікаційних систем і мереж»
Дейкало Артем Юрійович _____

Науковий керівник:
канд. військ. наук, доцент
Довбня Сергій Якович _____

Рецензент:
док. техніч.наук, старший науковий співробітник
Гільгурт Сергій Якович _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____ Артем ДЕЙКАЛО

Київ 2024

РЕФЕРАТ

Магістерська робота: 48 с., 8 рис., 14 джерел.

SVD – МЕТОД, ОЦИФРОВКА ДАНИХ, ВІДНОВЛЕННЯ ДАНИХ, ІНВЕРСІЙНА ФІЛЬТРАЦІЯ.

Об’єктом даного дослідження є система з ідентифікацією датчика за допомогою SVD алгоритму з метою прискорення отримання сигналу тривоги. Метод SVD (Singular Value Decomposition) надає можливість зменшити об'єм інформації, зберігаючи при цьому суттєві характеристики сигналу. Це дозволяє здійснювати ефективний обмін даними, зменшуючи час виявлення та передачі тривоги.

Мета роботи полягає в розробці системи з ідентифікацією датчика за допомогою SVD алгоритму для зменшення об'єму даних що передаються, забезпечуючи при цьому необхідну інформацію для ідентифікації тривоги.

Розглянута задача по створенню системи з ідентифікацією датчика за допомогою SVD алгоритму для прискорення сигналу тривоги.

Створено програму обробки даних та відновлення на їх основі.

Створено віртуальну модель передачі даних між двома ESP 32.

За допомогою метода SVD проведена обробка даних отриманих від ESP32 що емітує датчик (давач).

Розроблений алгоритм ідентифікації датчика (давача) на основі SVD методу.

Створено система з ідентифікацією датчика за допомогою SVD алгоритму, що пришвидшує отримання сигналу тривоги завдяки зменшенню кількості даних необхідних до передачі.

ЗМІСТ

ВСТУП

У сучасному світі розвитку технологій та зростанні важливості інформації, забезпечення безпеки передачі даних стає актуальною проблемою великої кількості сфер, зокрема в радіозв’язку [8-12].

Забезпечення ефективності та швидкості обміну інформацією у системах безпеки, які використовують датчики, стає важливим завданням. Швидкість передачі тривоги важлива, щонайменше для того що б сигнал не встигли перехопити. Другим не менш важливим аспектом інформації що передається в охоронних системах є складність ідентифікації сигналу. Щоб попередити розшифровку сигналу і потенційну його підміну, а також, забезпечити точне виконання протоколу, або інструкцій – сигнал має мати достатній рівень захисту. Однак при цьому часто виникає конфлікт між швидкістю передачі інформації та кількістю даних, які потрібно передати.

Для забезпечення конфіденційності інформації системи охорони використовують шифрування що є необхідним, проте здебільшого збільшує кількість інформації що необхідно передати, це в свою чергу збільшує час доставки сигналу тривоги.

Насправді, у багатьох випадках основним завданням є максимально швидке виявлення та ідентифікація тривоги, а не повне збирання даних. У зв'язку з цим, виникає необхідність у розробці ефективних методів для прискорення отримання сигналу тривоги, не втрачаючи при цьому важливої інформації.

Об’єктом даного дослідження є система з ідентифікацією датчика за допомогою SVD алгоритму з метою прискорення отримання сигналу тривоги. Метод SVD (Singular Value Decomposition) надає можливість зменшити об'єм інформації, зберігаючи при цьому суттєві характеристики сигналу. Це дозволяє здійснювати ефективний обмін даними, зменшуючи час виявлення та передачі тривоги.

На даний час існують багато методів ідентифікації даних від датчика, але ця галузь постійно розвивається і потребує нових рішень. В цей час використання методу сингулярного розкладу SVD не було в повній мірі досліджено для задач передачі інформації.

Мета роботи полягає в розробці системи з ідентифікацією датчика за допомогою SVD алгоритму для зменшення об'єму даних що передаються, забезпечуючи при цьому необхідну інформацію для ідентифікації тривоги.

1. ОГЛЯД ЛІТЕРАТУРИ

На підставі проведеного аналізу [1-12] розглянуто та опановано методику SVD, обрано математичний апарат, і елементну базу побудови системи з ідентифікацією датчика.

1.1. Методика SVD

SVD – Singular Value Decomposition (SVD або сингулярне розкладання). Розклад за сингулярним значенням матриці у лінійній алгебрі є розкладанням цієї матриці на три матриці.

Для розкладання сингулярних значень береться прямокутна матриця даних визначена як A (де A матриця n x p). Теорема SVD стверджує [2]:

$$A_{nxp} = U_{nxn} S_{nxp} V^T_{pxp}, \tag{1.1}$$

де $U^TU=I_{nxn}$ та $V^TV=I_{pxp}$ (U та V - ортогональні), стовпці U є лівими сингулярними векторами, S (ті самі розміри, що і A) має одиничні значення і є діагональним (амплітуди мод), V^T включає праві сингулярні вектори (вектори рівня виразу).

SVD являє собою розширення вихідних даних у системі координат, де коваріаційна матриця є діагональною. Розрахунок за допомогою SVD проводиться знаходженням власних значень AA^T і власних векторів A^TA. Власні вектори A^TA складають стовпці V, власні вектори AA^T утворюють стовпці U. Крім того, сингулярні значення в S є квадратними коренями власних значень з AA^T або A^TA. Діагональні елементи матриці S являють собою сингулярні значення початкової матриці. Власні числа в діагональній матриці S розташовані в порядку спадання важливості значень.

Сингулярні значення завжди є дійсними числами. Якщо матриця A є дійсною матрицею, то U і V також дійсні [3].

SVD - розв'язується таким чином:

$$A = \begin{bmatrix} 3 & 5 \\ 2 & 4 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \tag{1.2}$$

У цьому прикладі матриця являє собою матрицю 4x2. Ми знаємо, що для *nxn* матриця W, ненульовий вектор $\mathbf{x}$ є власним вектором W, якщо: $W_x=|\mathbf{x}|$ для деякого скаляра l. Тоді скаляр l називають власним значенням A, а $\mathbf{x}$ називають власним вектором A, що відповідає l [3].

Отже, щоб знайти власні значення SVD-матриці, ми обчислюємо матриці AA^T і A^TA. Як було зазначено раніше, власні вектори AA^T складають стовпці U, тому ми можемо виконати наступний аналіз, щоб знайти U.

$$AA^T = \begin{bmatrix} 3 & 5 \\ 2 & 4 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 3 & 2 & 1 & 0 \\ 5 & 4 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 34 & 26 & 8 & 0 \\ 26 & 20 & 6 & 0 \\ 8 & 6 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = W \tag{1.3}$$

Тепер, коли у нас є матриця *nxn*, ми можемо визначити власні значення матриці W. Оскільки $W_x = |\mathbf{x}|$, то $(W-|I|)\mathbf{x} = 0$.

$$\begin{bmatrix} 34-\lambda & 26 & 8 & 0 \\ 26 & 20-\lambda & 6 & 0 \\ 8 & 6 & 2-\lambda & 0 \\ 0 & 0 & 0 & -\lambda \end{bmatrix} = (W - \lambda I)x = 0 \tag{1.4}$$

Для унікального набору власних значень, визначник матриці ($W-|I|$) має дорівнювати нулю. Таким чином з розв'язку характеристичного рівняння, $|W-|I||=0$ отримуємо: $\overline{\lambda_1 = 54{,}925, \lambda_2 = 1{,}075\ \lambda_3 = 0\ \lambda_4 = 0}$ (чотири власні значення, оскільки це поліном четвертого ступеня). Це значення можна використовувати для визначення власного вектора, який можна помістити в стовпці U. Таким чином, ми отримуємо таку систему рівнянь [2-3]:

$$\begin{cases} 54{,}925x_1 + 26x_2 + 8x_3 = 0 \\ 26x_1 + 20x_2 + 6x_3 = 0 \\ 8x_1 + 6x_2 = 0 \\ x_4 = 0 \end{cases} \tag{1.5}$$

Після спрощення перших двох рівнянь отримуємо відношення, яке пов'язує значення x_1 з x_2. Значення x_1 і x_2 вибираються таким чином, що елементи S є квадратними коренями з власних значень. Таким чином, розв'язок, що задовольняє наведену раніше систему рівнянь x_1 = -0,58 і x_2 = 0,82 і $x_3 = x_4 =$ 0 (це і є другий стовпець U-матриці). Підставляючи інше власне значення, отримуємо:

$$U = \begin{bmatrix} -0.6914 & 0.7133 & -0.1111 & 0 \\ -0.5194 & -0.2148 & 0.8264 & 0 \\ -0.2343 & -0.6665 & -0.7071 & 0 \\ 0.4131 & 0.0000 & 0.0000 & 1 \end{bmatrix} \tag{1.6}$$

Аналогічно A^TA складає стовпці V, тому ми можемо виконати подібний аналіз, щоб знайти значення V [2].

$$AA^T = \begin{bmatrix} 3 & 5 \\ 2 & 4 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 3 & 2 & 1 & 0 \\ 5 & 4 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 34 & 26 & 8 & 0 \\ 26 & 20 & 6 & 0 \\ 8 & 6 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{1.7}$$

і аналогічним чином отримуємо вираз:

$$V = \begin{bmatrix} -0.6007 & -0.7995 \\ -0.7995 & 0.6007 \end{bmatrix} \tag{1.8}$$

Нарешті, як згадувалося раніше, S є квадратним коренем з власних значень з AA^T або A^TA. Його можна отримати безпосередньо, надаючи нам:

$$S = \begin{bmatrix} 7.41 & 0 \\ 0 & 1.04 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{1.9}$$

1.2. Побудова квазірозв‘язку методом SVD

Для побудови квазірозв‘язку операторного рівняння вибирається модальний метод, у вигляді:(див. формулу (А.10) в Додатку А).

Для виконання умови обмеженості коефіцієнти повинні задовольняти нерівності:(див. формулу (А.11) в Додатку А). Ця нерівність визначає, наскільки великі можуть бути коефіцієнти, щоб ряд залишався збіжним.

Модальні методи є потужним інструментом аналізу лінійних систем. Якщо перетворення сигналу у лінійній системі описується ермітовим оператором, то за базис представлення вхідного та вихідного сигналів зручно обирати власні моди оператора системи. В такому представлені передавальна характеристика набуває вигляду діагональної матриці, де кількість ненульових діагональних елементів визначає кількість мод, що беруть участь у формуванні образу. При цьому реальні лінійні системи в основному описуються неермітовими операторами, діагоналізація яких незастосовна в дослідженнях обернених задач [4].

Розглядаючи випадок, коли оператор A є самоспряженим (ермітовим) оператором, тобто його матриця дорівнює своїй транспонованій комплексно-спряженій матриці. Та за припущенням, оператор є обмеженим і неперервним. Маємо систему власних функцій оператора, повну як в просторі визначення, так і в просторі значень. У цьому випадку ми маємо систему власних функцій оператора, яка є повною як у просторі визначення, так і в просторі значень (див. формулу (А.12) в Додатку А). Ці функції є розв‘язками інтегрального рівняння Фредгольма з відповідним ядром (див. формулу (А.13) в Додатку А).

З огляду на повноту системи власних функцій, ми можемо представити вхідний і вихідний сигнали у вигляді сум по власних функціях з відповідними коефіцієнтами (див. формулу (А.14) та (А.15) в Додатку А).

Звідки, підставляючи ці співвідношення у початкове рівняння (А.10) і враховуючи попередні результати (А.15), отримаємо зв‘язок між коефіцієнтами який визначає їх залежність від власних значень оператора (див. формулу (А.16) в Додатку А), що дозволяє з’ясувати, як змінюються коефіцієнти у розкладі при зміні власних значень [4].

Таким чином, формально ми отримаємо відновлену функцію вхідного сигналу представивши її як суму часток коефіцієнтів і власних функцій, поділених на відповідні власні значення (див. формулу (А.17) в Додатку А). З чого випливає, що кожний вклад у розклад сигналу залежить від відношення коефіцієнта до власного значення.

В загальному випадку дане рішення не є квазірозв‘язком. Оскільки оператор А – обмежений, а це значить, що з нерівності Шварца маємо випливає, що частки зменшуються швидше, ніж самі коефіцієнти (див. формулу (А.18) в Додатку А) [4].

З формули 1.18 слідує що функція z(s) представляється у вигляді ряду за власними функціями оператора (див. формулу (А.19) в Додатку А).

Оскільки $\|z\|^2 < \infty$ слідує, що власні числа λ_n швидко збігаються до нуля, так що ряд (1.17) може бути розбіжним з огляду на це, розв’язок може не бути стійким при великих відхиленнях [4].

При додаванні шуму (не коректних значень) до вихідного сигналу маємо відхилення, яке також можна представити як суму по власних функціях з відповідними коефіцієнтами, що дозволить врахувати вплив шуму на розклад сигналу (див. формулу (А.20) та (А.21) в Додатку А) , де γ_n – коефіцієнт розвинення реалізації шуму $\eta(x)$ по власним функціям оператора А.

Шум також можна подати у вигляді суми власних функцій з відповідними коефіцієнтами. Коефіцієнти шуму визначаються розкладом реалізації шуму за власними функціями оператора (див. формулу (А.22) в Додатку А). [4]

Якщо система описується неермітовим оператором, тобто оператором, який не є самоспряженим, тоді використовується інший підхід (див. формулу (А.23) в Додатку А) [4].

Припустимо, що існують дві системи функцій, які задовольняють певну ортонормованість у відповідних просторах, тобто кожна функція з однієї системи ортогональна до всіх функцій іншої системи і має норму одиницю. (див. формулу (А.24) та (А.25) в Додатку А).

Для знаходження таких систем функцій запишемо рівняння, спряжене до початкового, (див. формулу (А.26) в Додатку А) де A^+ – оператор, ермітово спряжений до A [5] дане рівняння дозволяє знайти власні функції і власні значення для неермітового оператора.

Тоді, з умови ортонормованості (А.25) матимемо (див. формулу (А.27) в Додатку А).

Розглянемо тепер внутрішній інтеграл для перевірки ортонормованості власних функцій (див. формулу (А.28) в Додатку А).

Тоді отримаємо рівняння з ермітовим ядром, яке визначає власні функції та власні значення ермітового оператора (див. формулу (А.29) в Додатку А).

Дане рівняння має ермітове ядро, тому відповідний оператор має власні функції, причому його власні значення дійсні. Щоб задовольнити рівність (А.29), необхідно розглянути власні функції ермітового оператора (див. формулу (А.30) в Додатку А), що є рівнянням на власні функції ермітового оператора (див. формулу (А.31) в Додатку А) [4].

1.3 Мікроконтролери

Мікроконтролер, або однокристальний мікрокомп'ютер – це спеціалізована система на базі мікропроцесора. Дана система виконується у вигляді мікросхеми, що включає мікропроцесор, модулі пам'яті для зберігання програм, інструкцій, та даних, інтерфейси вводу/виводу та компоненти зі спеціальними функціями (лічильники, компаратори, АЦП тощо)[5].

Мікроконтролери застосовуються для управління/керування електронними пристроями. В загальному це – однокристальний комп'ютер, призначений для управління функціями що не потребують використання складної зовнішньої операційної системи. Реалізація у вигляді унітарної мікросхеми дозволяє значно зменшити розміри, вартість та витрати енергії пристроїв реалізованих на базі мікроконтролерів [5-6].

Мікроконтролери застосовуються для реалізації багатьох сучасних пристроїв, включаючи телефони, охоронні системи, системи розумного будинку, тощо. Вони використовуються в керуванні двигунів і систем безпеки сучасних автомобілів, а також для створення систем управління та збору

інформації. Більшість процесорів, що виробляються у світі використовується у створенні саме мікроконтроллерів [6].

Мікроконтролери мають певний набір периферійних пристроїв таких як:

- інтерфейси вводу/виводу, такі як UART, I2C, SPI, CAN,USB;
- Аналого-цифрові та цифро-аналогові перетворювачі;
- Компаратори;
- Широтно-імпульсні модулятори;
- RTC (real-time clock) – годинник реального часу;

Інтерпретуючи дані які мікроконтролер отримує від периферійних пристроїв вводу/виводу за допомогою центрального процесора, він здатен керувати функціями пристрою в який вбудований [6].

Інформація що надходить мікроконтролеру зберігається в його пам’яті даних. Ця інформація використовується процесором який в свою чергу використовує інструкції, що зберігаються в пам’яті програм, що б розшифрувати і застосувати вхідні дані [5].

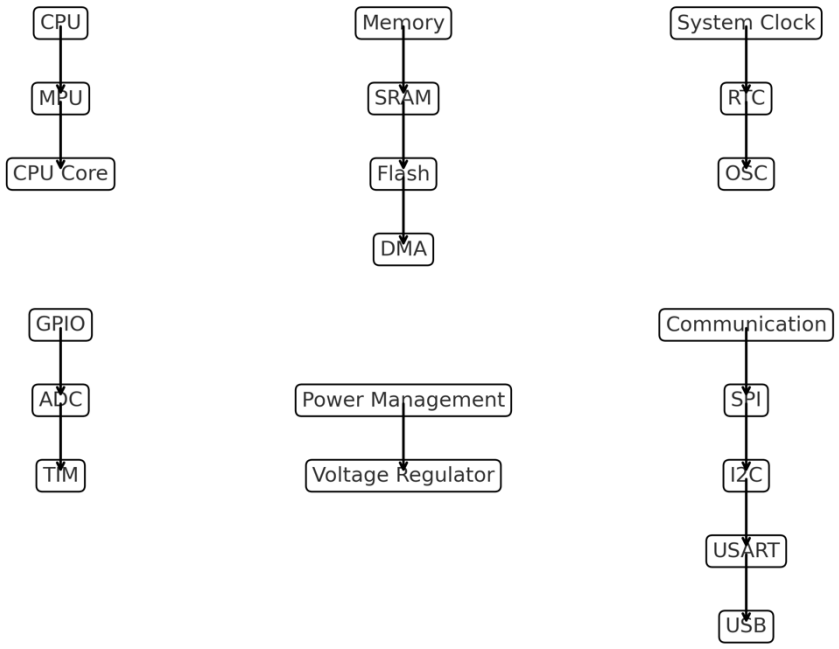

Рис. 1.1. Периферійні пристрої мікроконтролерів [5].

Для розуміння побудови системи варто розібрати головні складові мікроконтролерів, що б використовувати їх можливості.

Для подальшого побудови системи ідентифікації датчика було обрано мікроконтролер з наступною конфігурацією [6]:

- Процесор (CPU) – даний елемент обробляє і реагує на різного роду інструкції, які керують роботою мікроконтролера. Це передбачає виконання арифметичних, логічних операцій та операцій вводу або виводу.
- Пам’ять мікроконтролера використовується для зберігання даних, які процесор отримує і використовує для виконання запрограмованих інструкцій. Пам’ять мікроконтролера поділяється на два основних типи:

1. Пам’ять програм – зберігає довгострокову інформацію про інструкції, які використовує процесор. Пам’ять програм, завдяки енергонезалежності зберігає інформацію протягом значного періоду часу, не потребуючи джерела живлення, що дозволяє в свою чергу записати (вшити) програму виконання інструкцій в мікропроцесор одного разу та використовувати його як самостійний обчислювальний апарат.
2. Пам’ять даних – енергозалежний тип пам’яті, що слугує для тимчасового зберігання даних. Даний тип пам’яті потребує живлення для збереження даних, тому вони залишаються тимчасовими та зберігаються лише при наявності джерела живлення.

- Периферійні пристрої вводу/ виводу є інтерфейсом для зв’язку процесора з іншими пристроями та зовнішнім світом. Порти вводу використовуються для отримання інформації і надсилання її до процесора у вигляді бінарних даних. Отримавши дані процесор надсилає відповідні команди на зовнішні пристрої виводу, які використовуються для виконання поза межами мікроконтролеру, до прикладу іншим мікроконтролером.
- Периферійні пристрої першочергово це допоміжні компоненти які взаємодіють з пам’яттю та процесором. Головним же типом є периферійні пристрої вводу/виводу, інші типи були перераховані вище.

- Послідовний порт дозволяє мікроконтролеру підключатися до зовнішніх компонентів. За допомогою послідовного порту мікроконтролер можна під’єднати до комп’ютера та мати змогу виводити інформацію за допомогою комп’ютера та середовища розробки [5-6].

Обрана конфігурація мікроконтролера наведена на рисунку 1.2.

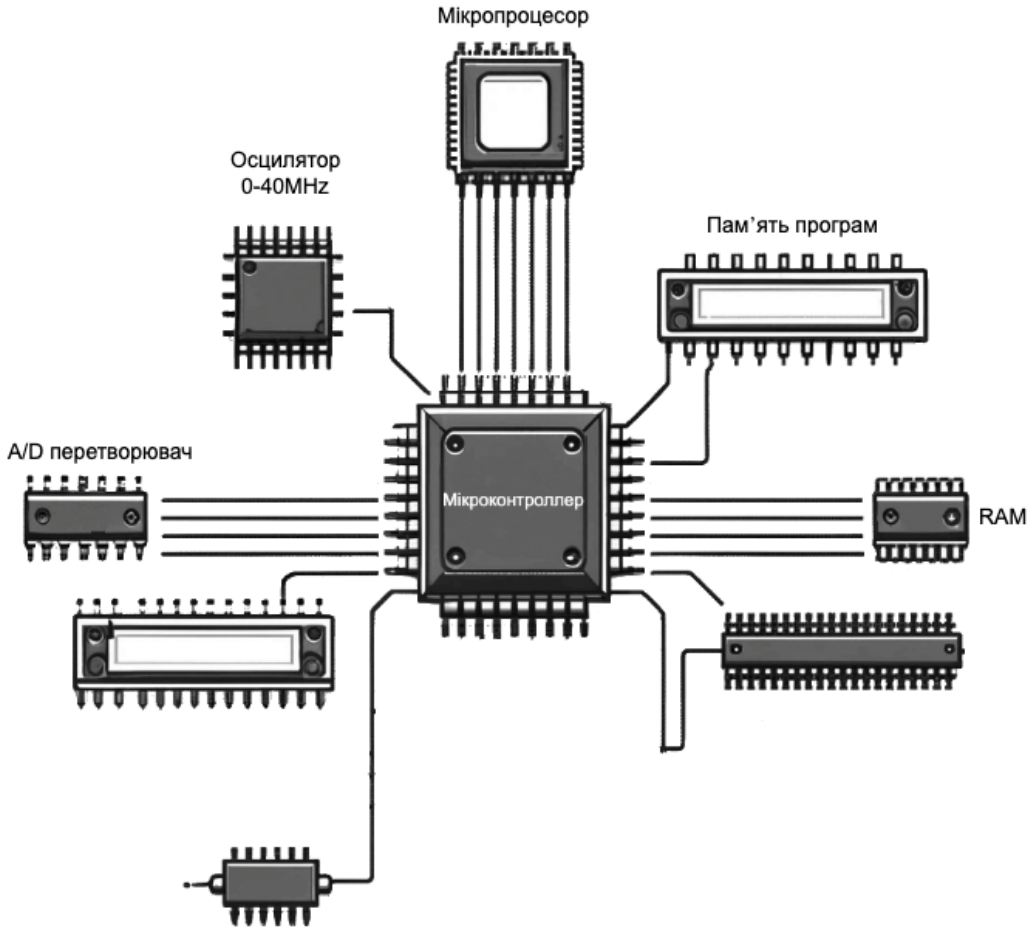

Рис 1.2. Схема внутрішніх з’єднань мікроконтролера

1.4 ESP 32

Для виконання завдань розроблення системи з ідентифікацією датчика за допомогою SVD процедури для пришвидшення отримання сигналу тривоги, обрано серію мікроконтролерів ESP32.

Лінійка ESP32 – це серія мікроконтролерів з низьким енергоспоживанням, що представляють собою єдину систему на кристалі з інтегрованими Wi-Fi 802.11 та дворежимним Bluetooth версії 4.2, контролерами та антенами [7].

У серії ESP32 використовуються ядро процесора Tensilica Xtensa LX6 в двох і одноядерному виконанні. Двоядерні моделі забезпечують вищу продуктивність для більш складних обчислень, тоді як одноядерні варіанти можуть бути використані в менш вимогливих системах, при цьому будучи більш енергоефективними. Система включає в себе вбудований радіочастотний тракт, що складається з балансного перетворювача (симетричний трансформатор), інтегрованих антенних перемикачів, компонентів радіочастотного діапазону, LNA (low-noise amplifier) підсилювача, блоку живлення, фільтрів та модулів керування. Вбудований радіочастотний тракт дозволяє ESP32 забезпечувати стабільний бездротовий зв’язок, що є важливим критерієм для багатьох застосувань в області IoT (Internet of things).

Китайська компанія Espressif Systems є розробником серії мікроконтролерів ESP 32, виробляються ж вони на потужностях компанії TSMC, яка є провідним виробником напівпровідних пристроїв в світі [7].

Технічні характеристики ESP32:

Мікроконтролер, має двоядерний 32-розрядний мікропроцесор, і здатний виконувати до 600 мільйонів інструкцій на секунду. Щодо пам'яті, ESP32 має статичну оперативну пам'ять, що дозволяє зберігати тимчасові дані для виконання програм.

Мікроконтролер ESP32 був обраний з наступною конфігурацією периферійних інтерфейсів, що дозволяють взаємодіяти з різноманітними зовнішніми пристроями і сенсорами. Наприклад, він має аналого-цифровий перетворювач (АЦП). Він може перетворювати аналогові сигнали (наприклад, від сенсорів) на цифрові дані з високою точністю. Також ESP 32 маж цифро-аналогові перетворювачів (ЦАПів), які використовуються для перетворення цифрових даних на аналогові сигнали, що корисно для керування різними

пристроями. Можливість підключення до інших пристроїв для обміну даними, наприклад, датчиків, дисплеїв та зберігачів інформації. Завдяки такому набору інтерфейсів ESP32 є універсальним рішенням для інтеграції в різі типи систем, дозволяючи підключати різноманітні датчики, дисплеї, накопичувачі та інші периферійні пристрої, з підтримкою даних видів підключення.

Для виконання завдання передачі даних в системах сигналізації використовується бездротовий зв'язок, ESP32 підтримує Wi-Fi та Bluetooth. Також підтримується зв'язок з іншими бездротовими пристроями для передачі інформації на невеликі відстані. ESP 32 з позначенням обраної конфігурації відповідних портів зображена на рисунку 1.3.

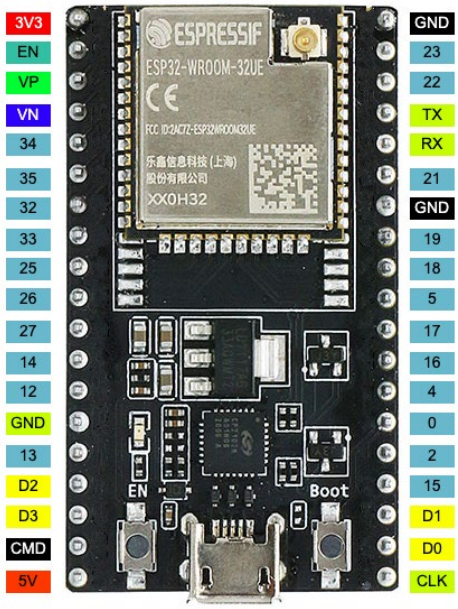

Рис. 1.3. – Позначення портів на платі ESP32 Espressif ESP-WROOM-32 [7].

Мікроконтролер ESP32, поставляється в різних версіях, таких як комплекти розробки з додатковими можливостями, наприклад, слотами для карт пам'яті та дисплеями. Є також версії з вбудованими датчиками для інтернету речей [7].

Крім оригінального виробника, є інші компанії, що випускають сумісні чіпи, які мають схожі характеристики. [7-8].

Деякі модулі ESP32 мають додаткову пам’ять для зберігання даних. Залежно від версії, пристрій може підтримувати різні способи підключення для взаємодії з іншими пристроями. Він також має вбудовані датчики, такі як датчик магнітного поля та температури [7]. Крім того, доступні додаткові набори датчиків для розробників. ESP32 також має апаратне прискорення для криптографічних алгоритмів, а також вбудований генератор випадкових чисел. [7].

Перевага платформи ESP32, є можливість підтримки основними операційними системами такими як: Windows, Linux або MacOS. Цього можна досягти за допомогою різних середовищ розробки, таких як додаткове середовище ESP32 для Arduino, Espressif IoT Development Framework (ESP-IDF) або розробки на основі Python з використанням рушія Micropython [7-8].

Обрана модель ESP32 представлена на рисунку 1.4.

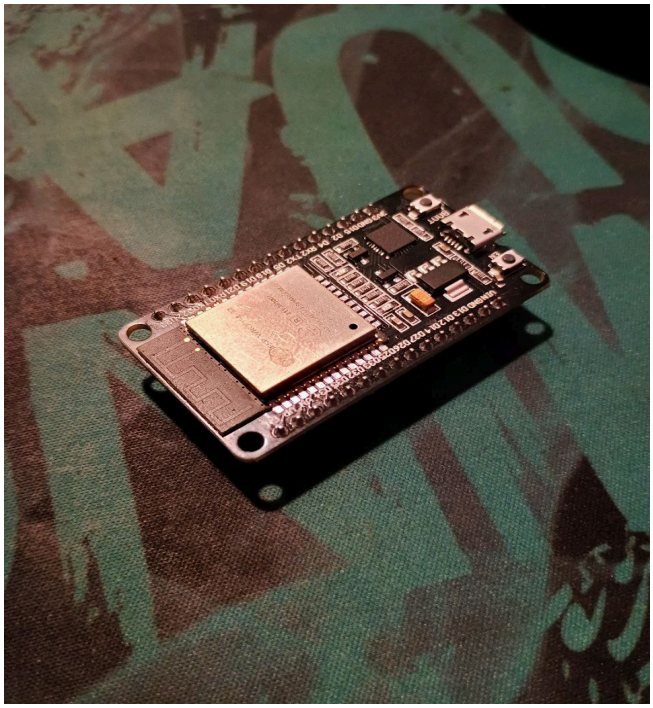

Рис. 1.4. Зовнішній вигляд ESP32

ESP32 була обрана за наступними параметрами:

- Робочий діапазон температур: мікроконтролер ESP32 має необхідний діапазон робочих температур від -40 градусів Цельсія до 125 градусів Цельсія. Так як в більшості вимог до систем сигналізації необхідний стандартний діапазон температур, -10 градусів Цельсію - +40 Градусів Цельсію [8].
- Низьке енергоспоживання: згідно вимог автономності роботи стандартів, що застосовуються до систем сигналізації, пристрій повинен працювати від 3 років. Енергоспоживання ESP32 в режимі очікування становить 150 mkA/год, що дозволяє дотриматись вимог автономності за рахунок компактного джерела автономного живлення ємністю в 1314 mA.
- Мікросхема Wi-Fi і Bluetooth: ESP32 має можливість функціонувати як автономна система, завдяки з'єднанню безпосередньо за допомогою дротових інтерфейсів, та працювати з іншими системами, використовуючи для з'єднання Bluetooth і WiFi. [7]

Для програмування ESP32 доступні різні мови, платформи та середовища, їх перелік та опис можна подивитись в додатку Б таблиця 1.

1.5 Системи безпеки та сигналізації

Дана робота спрямована на розробку системи з аутентифікацією датчика за допомогою SVD алгоритму, для пришвидшення отримання сигналу тривоги. Системи ідентифікації тривоги найчастіше використовуються в системах безпеки а саме системах сигналізації (системи виявлення вторгнення)[8].

Системи сигналізації охоплюють широкий спектр технологій і пристроїв, призначених для виявлення, сигналізації та запобігання несанкціонованому доступу, вторгненню або порушенню безпеки в різних сферах. Системи сигналізації можуть включати датчики виявлення вторгнення по периметру, датчики руху, датчики розбиття скла та дверні/віконні контакти, серед іншого, налаштовані на активацію звукових сирен, стробоскопів або сповіщень для

визначених зацікавлених сторін при виявленні підозрілої діяльності. Забезпечуючи раннє попередження і стримуючий ефект, системи сигналізації посилюють заходи безпеки, відлякують зловмисників і сприяють формуванню культури пильності та реагування [8-9].

Основними параметрами системи сигналізації обрано:

- Швидкість передачі: Системи сигналізації повинні демонструвати швидку передачу тривожних сигналів в діапазоні від 0.1с.-3с для зменшення ризиків і запобігання подальшій ескалації загроз безпеці [9].
- Доступність: 24/7 - система залишатиметься функціональною навіть за несприятливих умов або технічних збоїв, таким чином підтримуючи пильність безпеки в будь-який час [9].
- Аутентифікація: Відновлення за допомогою оберненого методу SVD. Ідентифікуються джерела тривожних сигналів, гарантуючи, що тривоги будуть викликані лише справжніми інцидентами безпеки, а не помилковими або зловмисними діями [9].
- Конфіденційність: Забезпечена передаючою сингулярних чисел початкової матриці значень, що не дає змогу відновити дані без необхідних масивів розкладу [9].

Для визначення необхідних алгоритмів спрацювання системи сигналізації в яку він інтегрований, централь (приймач інформації) має отримати певну інформацію під час передачі тривоги датчиком (давачем).

В сигналі тривоги присутні наступні типи кодифікації:

- Тип датчика, що передав інформацію: Слугує для визначення конкретного датчика, який спричинив активацію тривоги.
- Тип тривоги: Вказує на характер події, що спричинила активацію сигналу тривоги.
- Номер пакета: Унікальний ідентифікатор пакета даних, що дозволяє відрізняти та відстежувати кожен окремий пакет у процесі передачі. Також за

часту використовується для додаткового підтвердження валідності даних прийнятих централлю (приймачем).

- Значення сенсора: Числове значення, що вимірюється датчиком в момент активації тривоги. Значення сенсора дозволяє визначити критичність тривоги.
- Внутрішній час відправки: Час, коли сигнал тривоги був згенерований та відправлений, синхронізований у шістнадцятковій системі.

1.6 Шифрування

Використовується метод скремблювання з кінцевою кількістю перетворень. Здійснюється перетворення матриці початкових даних з додаванням хибних значень. [10-11].

Так як розроблена система має інтегруватись в систему сигналізації, одним з головних критеріїв якої повинна бути конфіденційність інформації, яка забезпечується завдяки різного виду шифруванням, було обрано спосіб шифрування, який полягає в заповненні матриці даних тривоги хибними значеннями [12].

2. ОСНОВНА ЧАСТИНА

2.1 Віртуальна модель

В даній роботі як об’єкт дослідження виступає система передачі даних між двома мікроконтролерами ESP32. Ця система має на меті забезпечити безперебійний зв’язок між двома мікроконтролерами. Де один мікроконтролер ESP32 виступає в якості передавача тривоги (датчика) а інший в якості приймача тривоги (централі). Мікроконтролер, що емітує роботу датчика відповідає за генерацію та відправку даних тривоги, до мікроконтролера, що виступає в якості централі.

Для отримання експериментальних даних про швидкість та якість передачі даних та подальшій їх використанні в модернізації системи, першим етапом було обрано електронний сервіс віртуального моделювання електроніки Wokwi.com обрано та налаштовано симуляції ESP32 (Рис. 2.1).

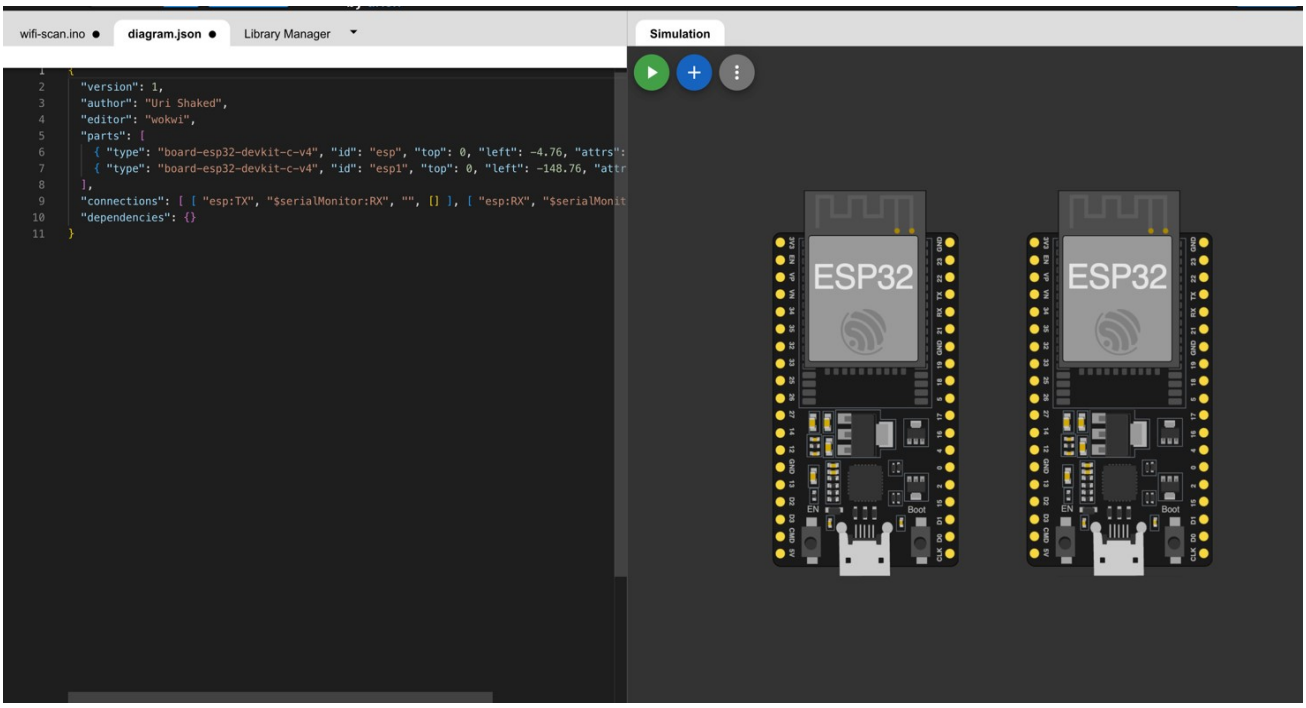

Рис. 2.1. Налаштована симуляція двох ESP32 в сервісі віртуального моделювання електроніки Wokwi.com.

Другим етапом налаштування ESP32 мікроконтролерів є реалізація віртуальної моделі. Для реалізації віртуальної моделі, обидва мікроконтролера ESP32 були сконфігуровані для одночасної роботи в режимах точки доступу (AP) та станції (STA) Wi-Fi. Це дозволяє одному ESP32 виступати в ролі точки доступу, тоді як інший може підключатися до цієї точки доступу в рамках локальної мережі, проте є і можливість налаштування глобальної мережі для даної системи. Цей підхід створює можливість обом мікроконтролерам працювати в одній безпровідній мережі та обмінюватися даними.

Третій етап: за допомогою бібліотеки esp_now.h, на кожному ESP32 був налаштований веб-сервер, що дозволяє обмінюватися даними через HTTP запити. Це відкриває можливість для взаємодії між мікроконтролерами за допомогою веб-інтерфейсу, спрощуючи обмін даними.

HTTP-запити дозволяють мікроконтролерам здійснювати обмін даними через бездротову мережу Wi-Fi. Це створює ефективний спосіб комунікації між мікроконтролерами. Використання HTPP реалізує взаємодію між різними пристроями у мережі, створюючи умови для передачі інформації між вебсерверами. Даний підхід є корисним в практичних застосуваннях, оскільки дозволяє легко інтегрувати різні системи та пристрої, забезпечуючи гнучкість та розширюваність мережі.

Наступним кроком стало налаштування партнера для ESP-NOW. ESP-NOW використовує концепцію партнерів, тобто пристроїв, які можуть обмінюватися даними. У нашому випадку один ESP32 налаштовується як партнер для іншого. Аналогічно, другий налаштовується як партнер для першого.

Обмін даними через ESP-NOW:

- ESP-NOW дозволяє відправляти та отримувати дані за допомогою функцій esp_now_send() та esp_now_recv().
- Створений код з’єднання та передачі даних між двома ESP32 представлений в додатку.

2.2 Збір даних від датчика і їх обробка

Для обробки тривоги було згенеровано п’ять байтів що відповідають п’яти числам записаним в шістнадцятковій системі. Дані описують основні типи інформації а саме: перший байт – тип датчика, другий байт – тип тривоги, третій байт – номер пакету, четвертий байт – значення сенсору, внутрішній час відправки. Результатом генерації стала матриця розміру п’ять на один.

Рис. 2.2. Розподілення даних згенерованої тривоги.

2.3 Використання SVD алгоритму для виділення основних компонентів сигналу

Використовуючи метод наповнення матриці хибними значеннями, було сформовану матрицю даних двадцять на двадцять значень. Де обов’язковими елементами матриці по діагоналі є значення з Рис. 2.2, а всі інші створені випадковим чином в діапазоні від одного до тридцяти. Результатом цього стала матриця розміром двадцять на двадцять Рис. 2.3, яка записана до документу формату .xlsx за допомогою пакету Microsoft Office Excel. Червоним кольором позначені початкові значення тривоги.

14	D	C	F	13	9	11	16	12	17	15	7	C	12	12	18	16	1E	13	1B
10	3C	4	4	1B	2	B	11	11	1B	13	1D	16	9	3	F	16	F	1	2
10	10	11	1C	15	1E	4	F	3	A	18	1D	16	15	A	12	12	8	17	18
13	1D	12	E	13	3	3	9	1E	17	1D	8	10	19	C	1C	17	1E	15	2
1D	F	5	46	B	E	1E	12	1D	1A	B	1C	4	E	E	15	E	12	C	6
15	11	A	9	5	1A	B	7	15	1E	6	B	15	E	10	18	4	7	5	1B
19	18	14	1	1A	F	E	B	E	15	8	1B	1D	17	12	1D	15	4	1D	4
E	12	15	16	11	51	9	3	1B	E	8	3	1E	13	6	6	5	7	5	E
C	6	9	1C	4	16	4	19	1A	2	14	7	12	D	12	13	11	3	5	1A
1D	15	2	16	19	16	16	E	10	1C	8	1C	14	A	2	11	16	6	7	B
15	1A	10	8	1C	16	F	E	18	1D	13	9	1C	9	7	12	6	14	13	17
F	A	1	10	15	11	14	1	3	8	4	D	F	C	19	9	4	E	5	1C
8	E	10	12	2	2	3	10	C	14	8	C	6	16	7	14	1C	4	1D	1C
6	5	18	A	8	A	C	A	1E	F	17	C	4	A	B	18	6	13	13	11
2	9	B	1B	4	B	13	C	3	1C	63	A	7	18	1B	4	1B	F	3	1
A	7	16	18	8	1B	4	7	3	19	7	12	17	E	1A	11	5	13	4	10
18	D	11	C	4	F	8	1B	19	12	19	11	16	18	A	9	F	10	15	12
2	17	17	9	13	A	1C	1B	10	12	13	13	F	18	19	14	5C	14	1A	8
A	7	18	11	C	19	1B	17	5	4	1D	1A	7	1B	2	12	1B	1A	D	17
E	E	12	16	D	14	D	D	2	3	C	18	E	A	1A	3	11	C	F	1D

Рис. 2.3. Матриця значень тривоги наповнена хибними значеннями.

За допомогою бібліотеки openpyxl для python було імпортовано дані та записані до двовимірного масиву *x* - апаратна матриця , для виконання методу SVD. Перед початком обробки з даних матриці x розрахунком був створений масив *Y* , який являє собою ядро матричного перетворення.

Сам метод обробки був проведений над даним масивом значень за допомогою бібліотеки numpy та функції np.linalg.svd. Результатом цієї функції є сингулярне розкладення методом SVD масиву x що записується в три матриці *U V D* відповідно. Для масиву значень *D* елементи якої по діагоналі являються власними числами оброблюваної матриці за допомогою функції LA.eigh із бібліотеки numpy знайдено сингулярні числа, що записані в матриці wD. При цьому стовпці матриці *U* складаються з лівих сингулярних векторів, а масив *V* в свою чергу з правих сингулярних векторів. Як результат маємо двадцять сингулярних чисел та два масиви такої ж кількості сингулярних векторів, що в свою чергу складаються із двадцяти стовпців.

Для перевірки правильності отриманих даних дві сусідні власні функції в матриці vU були перемножені між собою, отриманий результат рівний нулю, що свідчить про правильність розрахунків.

2.4. Розробка алгоритму ідентифікації на основі отриманих даних

Алгоритм ідентифікації спрямований на визначення достовірності даних отриманих централлю від датчика. Сам алгоритм ідентифікації у вигляді коду обробки реалізований на стороні централі. Загальний алгоритм передачі інформації зображено на Рис. 2.4.

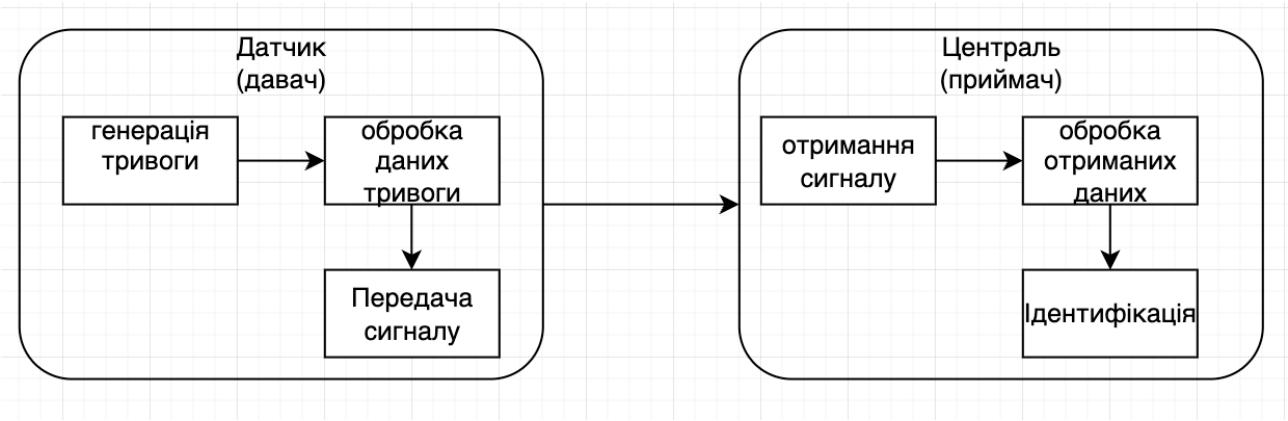

Рис. 2.4. Алгоритм передачі інформації між датчиком та централлю

Після проведення обробки сигналу тривоги за допомогою SVD алгоритму отримано три матриці U, V та D.

Оброблені дані можна використовувати як патерн інформації. А саме: матриці U та V що складаються з лівих сингулярних векторів (U), та з правих сингулярних векторів (V) початкової матриці значень тривоги наповненої хибними значеннями, записуються в пам’ять централі. Дані матриці (масиви), будуть використані для обробки даних що надійшли від датчика.

Підготовка даних для передачі від датчика до централі була реалізована наступним чином. З матриці D були виділені власні числа матриці даних

тривоги наповненої хибними значеннями. Для передачі даних, які є власними числами, на централь, використано функцію esp_now_send() з бібліотеки esp_now.h.

Отримання та ідентифікація даних отриманих від датчика була реалізована наступним чином. Для ідентифікації датчика був використаний зворотній метод SVD. Першим кроком отримані власні числа були використані для відновлення діагональної матриці D де по діагоналі розташовані отримані дані, а всі інші значення заповнені нулями.

Наступним кроком перемноживши матриці U та V що містяться в пам’яті централі з матрицею D яка отримана після обробки початкових даних отримуємо як результат відновлену початкову матрицю J тривоги з хибними значеннями.

Для проведення процесу ініціалізації замінюємо всі значення що лежать поза діапазоном від одного до тридцяти на нулі, та записуємо ці значення в матрицю K1. Наступним кроком порівнюємо отриману матрицю K1 та матрицю K. Якщо результат порівняння видає хибу значить дані були підмінені, та централь ігнорує їх. В нашому випадку результат порівняння повне співпадіння значень матриць, а отже дані валідні.

Виокремивши всі дані з матриці що лежать поза діапазоном від одного до тридцяти, як результат отримуємо початкові дані тривоги, що можна використати для виконання інструкцій безпеки.

2.5 Оцінка ефективності системи

Одним з головних критеріїв оцінки ефективності системи є швидкість передачі та отримання даних тривоги. Для оцінки швидкості передачі даних в порівнянні з прикладом стандартної системи, порівняємо об’єми переданих даних.

Стандартна система передає матрицю даних тривоги наповнену хибними значеннями. Дана матриця являє собою масив двадцять на двадцять, кожен

елемент якої є символом із шістнадцятковим значенням, яке можна розглядати як один байт (вісім біт). Прораховано об’єм необхідних даних для передачі (2.1).

Розмір необробленої матриці = 20 × 20 × 1 байт = 400 байт. (2.1)

Отже необроблена матриця для передачі потребує чотириста байт. В нашій системі для отримання значень тривоги від датчика достатньо передати власні числа. Так як результатом обробки початкової матриці наповненої хибними значеннями можуть бути не цілі числа то сингулярні значення представляються як числа з плаваючою комою. Стандартний розмір числа з плаваючою комою у мові програмування Python становить вісім байт (шістдесят чотири біти).

Самих власних чисел для матриці двадцять на двадцять буде відповідно двадцять. Прораховано об’єм необхідних даних для передачі (2.2).

Розмір сингулярних значень = 20 сингулярних значень × 8 байт = 160 байт (2.2)

В нашій системі ESP32 передає дані через Wi-Fi. Швидкість передачі даних може змінюватися залежно від таких факторів, як рівень сигналу і перевантаження мережі, для порівняння достатньо використати типові значення швидкості передачі даних.

Для ESP32 типова швидкість передачі даних за допомогою WiFi за нормальних умов може становити близько 10 Мбіт/с (1.25 Мбайт/с).

Обрахунок швидкості передачі для необробленої матриці (стандартної системи):

(2.3)

та сингулярних чисел (запропонована система):

(2.4)

Передача сингулярних значень відбувається значно швидше, ніж передача всієї необробленої матриці. Зокрема, передача сингулярних значень займає приблизно 40% часу, необхідного для передачі необробленої матриці. Це демонструє ефективність використання запропонованої системи в передачі даних тривоги.

Аналізуючи кількість переданої інформації можна зробити висновок що завдяки зменшенню кількості даних що необхідно передати під час тривоги зменшується і час її отримання. При цьому дані що передаються а не несуть в собі повної інформації про тривогу що без внутрішньої обробки не можуть бути використані, для отримання даних, а отже забезпечується конфіденційність даних. Алгоритм ідентифікації, успішно ідентифікував датчик та прийняв від нього значення. В роботі використовувався приклад шифрування, що використовується в системах сигналізації, запропонований алгоритм ідентифікації має гнучкість до інтеграції в різного виду систем сигналізації і його імплементація в існуючі системи може показати ще більшу ефективність проте в нашій роботі це не досліджувалось.

ВИСНОВКИ

1. Створена віртуальна модель системи передачі інформації між двома мікроконтролерами ESP 32, отримана модель використана в подальшому для розробки алгоритму індентифікації та симуляції роботи централь (приймач) та датчик (давач)

2. Згенеровано дані тривоги від датчика (давача) та за допомогою методу наповнення хибними значеннями змодельований метод шифрування в системах сигналізації. Проведена SVD-обробка даної матриці та знайдені сингулярні числа та матриці власних векторів.

3. Задача передачі даних реалізована за допомогою передачі сингулярних чисел, що зменшило кількість необхідної інформації для передавання. Аналіз показав, що запропонована система вирішує поставлену задачу, є досить ефективною та має потенціал до впровадження в існуючи системи сигналізації.

4. Досягнуто показників швидкості передачі тривоги в 0.128 мс , що пришвидшило систему в 2.5 рази.

5. Покращити результати може розробка дворівневого механізму обробки за допомогою SVD процедури, а також оптимізація виконання обчислень використовуючи апаратне обчислення а не програмне.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Jeol. (n.d.). 分光分析(EELS、EDS、电子构造). Retrieved February 7, 2024, from http://surl.li/cdcen

2. GeeksforGeeks. (n.d.). Singular Value Decomposition(SVD). Retrieved March 1, 2024, from https://www.geeksforgeeks.org/singular-value-decomposition-svd/

3. Wikipedia. (n.d.). Сингулярний розклад матриці. Retrieved April 11, 2024, from https://uk.wikipedia.org/wiki/Сингулярний_розклад_матриці

4. ScienceDirect. (n.d.). Singular Value Decomposition. Retrieved April 12, 2024, from https://www.sciencedirect.com/topics/engineering/singular-value-decomposition

5. alsaibie. (n.d.). Part I L4 – Microcontroller Peripherals. Retrieved March 11, 2024, from https://alsaibie.github.io/me319/part_i/lecture4/#part_i_l4_-_microcontroller_peripherals

6. arrow. (n.d.). Introduction to Microcontrollers. Retrieved April 15, 2024, from https://www.arrow.com/en/research-and-events/articles/engineering-basics-what-is-a-microcontroller

7. Babiuch, M., Foltýnek, P., & Smutny, P. (2019). Using the ESP32 Microcontroller for Data Processing. 2019 20th International Carpathian Control Conference (ICCC), 1-6.

8. Liao, H. J., Lin, C. H. R., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications, 36(1), 16-24.

9. Bace, R. G., & Mell, P. (2001). Intrusion detection systems.

10. Anitha Ruth, J., Sirmathi, H., & Meenakshi, A. (2019). Secure data storage and intrusion detection in the cloud using MANN and dual encryption through various attacks. IET Information Security, 13(4), 321-329.

11. Cavusoglu, H., Mishra, B., & Raghunathan, S. (2005). The value of intrusion detection systems in information technology security architecture. Information Systems Research, 16(1), 28-46.

12. Bangali, J., & Shaligram, A. (2013). Design and Implementation of Security Systems for Smart Home based on GSM technology. International Journal of Smart Home, 7(6), 201-208.

13. Czech Republic. Government Decree No. 616/2006 Coll. on technical requirements for products in terms of electromagnetic compatibility.

14. Dautray, R., & Lions, J. L. (2012). Mathematical analysis and numerical methods for science and technology: volume 1 physical origins and classical methods. Springer Science & Business Media – 718 p.

ДОДАТОК А

Математична база квазірозв'язку

$$z(s)=\sum_{n=1}^{\infty} b_n \varphi_n(s) \tag{A.10}$$

$$\sum_{n=1}^{\infty} |b_n|^2 \le R^2 \tag{A.11}$$

$$A\varphi_n = \lambda_n \varphi_n \tag{A.12}$$

$$A\varphi_n = \int_S K(x,s)\varphi_n(s)ds = \lambda_n \varphi_n(x) \tag{A.13}$$

$$z(s)=\sum_{n=1}^{\infty} b_n \varphi_n(s) \tag{A.14}$$

$$u(x)=\sum_{n=1}^{\infty} c_n \varphi_n(x) \tag{A.15}$$

$$b_n = \frac{c_n}{\lambda_n} \tag{A.16}$$

$$\tilde{z}(s)=\sum_{n=1}^{\infty} \frac{b_n}{\lambda_n}\varphi_n(s) \tag{A.17}$$

$$\|Az\|^2 \le \|A\|^2 \|z\|^2 = \sum_{n=1}^{\infty} \lambda_n^2 \cdot \|z\|^2 < \infty \tag{A.18}$$

$$\sum_{n=1}^{\infty} \lambda_n^2 < \infty \quad , \tag{A.19}$$

$$Az = \int_S K(x,s)z(s)ds + \eta(x) = u(x) + \eta(x) = \tilde{u}(x) \tag{A.20}$$

$$\tilde{z}(s)=\sum_{n=1}^{\infty}\left(\frac{b_n}{\lambda_n}\right)\varphi_n(s)+\sum_{n=1}^{\infty}\left(\frac{\gamma_n}{\lambda_n}\right)\varphi_n(s) \tag{A.21}$$

$$\eta(x)=\sum_{n=1}^{\infty}\gamma_n\varphi_n(x) \tag{A.22}$$

$$Az(s)=\int_S K(x,s)z(s)ds=u(x) \tag{A.23}$$

$$\int_S K(x,s)\varphi_n(s)ds=\lambda_n\psi_n(x) \tag{A.24}$$

,

$$\int_S \varphi_n^*(s)\varphi_m(s)ds=\delta_{mn} \qquad \int_X \psi_n^*(x)\psi_m(x)dx=\delta_{mn} \tag{A.25}$$

,

$$A^+z(s)=\int_S K^*(s,x)z^*(s)ds=u^*(x) \tag{A.26}$$

$$\int_X \psi_n^*(x)\psi_m(x)dx=\delta_{mn}=\frac{1}{\lambda_n\lambda_m}\int_X dx\int_S ds\int_S ds'K^*(s',x)K(x,s)\varphi_n^*(s')\varphi_m(s)=$$

$$=\frac{1}{\lambda_n\lambda_m}\int_S ds\int_S ds'\left[\int_X dxK^*(s',x)K(x,s)\right]\varphi_n^*(s')\varphi_m(s) \tag{A.27}$$

$$\left[\int_X dxK^*(s',x)K(x,s)\right]=R(s',s) \tag{A.28}$$

$$\delta_{mn}=\frac{1}{\lambda_n\lambda_m}\int_S ds\int_S ds'R(s',s)\varphi_n^*(s')\varphi_m(s) \tag{A.29}$$

$$\int_S R(s',s)\varphi_n(s)ds=\lambda_n^2\varphi_n(x) \tag{A.30}$$

$$\hat{\mathbf{R}}=\int_S R(s',s)...ds \tag{A.31}$$

ДОДАТОК Б

Технічні характеристики ESP32

1. Процесор: 32-розрядний двоядерний Xtensa LX6
2. Тактова частота: 160 або 240 МГц
3. Продуктивність: До 600 DMIPS (мільйонів інструкцій на секунду)
4. Оперативна пам’ять (RAM): 520 КБ SRAM
5. АЦП (ADC): 12-розрядний АЦП з підтримкою до 18 каналів
6. ЦАП (DAC): 38 8-розрядних ЦАПів
7. Інтерфейси: SPI, I2C, UART, I2S, SDIO/SPI, CAN, Ethernet MAC, IR
8. Бездротовий зв’язок: Wi-Fi стандарту 802.11 b/g/n, Bluetooth версії 4.2 BR/EDR
9. Набори для розробки: ESP32 Developer Kit, ESP32 Wrover Kit, ESP32 Azure IoT Kit
10. Сумісні чіпи: ESP32 Thing DB (SparkFun), TTGO, D1, Lolin32, Lolin D32 (WeMoS)
11. Зовнішня пам’ять: SPI флеш-пам’ять, PSRAM
12. Вбудовані сенсори: Датчик ефекту Холла, Температурний сенсор, Сенсорні датчики
13. Апаратне прискорення криптографічних алгоритмів: AES, SHA-2, RSA, ECC, Генератор випадкових чисел
14. Операційні системи: Windows, Linux, MacOS
15. Середовища розробки: ESP32 для Arduino, ESP-IDF, Micropython
16. Діапазон робочих температур: Від -40°C до 125°C
17. Інтегровані компоненти: RF балун, Малошумний підсилювач, перемикачі антен, Модулі керування живленням, Підсилювач потужності.

Таблиця 1 Мови програмування, середовища розробки для ESP32

Інструмент або платформа	Опис
Arduino IDE з ESP32 Arduino Core	Розширює функціональність Arduino IDE для розробки програм для ESP32.
Espressif IoT Development Framework	Офіційний набір інструментів для розробки програмного забезпечення для ESP32, надає розширені можливості для створення IoT-проектів.
Espruino	SDK на JavaScript для програмування ESP32, пропонує простий спосіб створення програм за допомогою мови JavaScript.
Lua RTOS для ESP32	Операційна система для ESP32 на мові програмування Lua, надає розширені можливості для створення вбудованих систем.
Mongoose OS	Операційна система для підключених пристроїв на мікроконтролерах, забезпечує широкі можливості для розробки IoT-проектів на ESP32.
PlatformIO Ecosystem та IDE	Платформа розробки, яка підтримує ESP32 та інші мікроконтролери, надає розширені інструменти для розробки та налагодження програм.
Pymakr IDE	Середовище розробки для спрощення створення програмного забезпечення для ESP32, використовується з пристроями Pycom.
Simba Embedded Programming Platform	Платформа програмування вбудованих систем, яка підтримує ESP32 та інші мікроконтролери.
Whitecat Ecosystem Blockly	Середовище програмування для створення програм для ESP32 за допомогою блоків.
MicroPython	Версія мови програмування Python для вбудованих систем, підтримує ESP32 та інші мікроконтролери.
Zerynth	Платформа програмування на мові Python для IoT та мікроконтролерів, включаючи ESP32, надає зручні інструменти для розробки програмного забезпечення для IoT-проектів.

ДОДАТОК В

1) Код віртуальної моделі

```
// ESP32 nodeMCU OWN MAC-Adress is {0x24, 0x0A, 0xC4, 0xEE, 0xD8, 0x08 };
// ESP32-D1-Mini OWN MAC-Adress is {0x08, 0x3A, 0xF2, 0x52, 0x09, 0x6C };
#include <esp_now.h>
#include <WiFi.h>

typedef struct test_struct {
  int arr[5]; } test_struct;

//Create a struct_message called myData test_struct myData;

//callback function that will be executed when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.print("Bytes received: ");
  Serial.println(len);   Serial.print("rearray=: ");   for (int i = 0; i < 5; i++) {
Serial.print(myData.arr[i]);

  }
  Serial.println();
}

void setup() {
  //Initialize Serial Monitor
```

```
        Serial.begin(115200);

        //Set device as a Wi-Fi Station
        WiFi.mode(WIFI_STA);

        if (esp_now_init() != ESP_OK) {
          Serial.println("Error initializing ESP-NOW");     return;
        }



        Serial.print("right befire executing
esp_now_register_recv_cb(OnDataRecv);");
        Serial.print("FINISHED executing
esp_now_register_recv_cb(OnDataRecv);");
        Serial.print("rearray AFTER esp fun=: ");
        Serial.print("myData is empty because right now ");
        Serial.print("not a SINGLE receiving data has happened");    for (int i = 0; i <
5; i++) {     Serial.print(myData.arr[i]);
        }
        Serial.print("as you cna see all data is zero");

      boolean TimePeriodIsOver (unsigned long &startOfPeriod, unsigned long
TimePeriod) {
        unsigned long currentMillis  = millis();
        if ( currentMillis - startOfPeriod >= TimePeriod ) {

          startOfPeriod = currentMillis;
           return true;
        }
        else return false;
```

```
}

unsigned long myNonBlockingTimer;

void myFunction1() {
  Serial.print("myFunction1() prints the content of myData ");    for (int i = 0; i
< 5; i++) {    Serial.print(myData.arr[i]);
  }
}

void myFunction2() {
  Serial.println("myFunction2() does calculations with the myData");    for (int
i = 0; i < 5; i++) {
   myData.arr[i] = myData.arr[i] * 10;
   Serial.print("myData.arr[");
   Serial.print(i);
   Serial.print("]=");
   Serial.println(myData.arr[i]);
  }

  Serial.print("myFunction2() prints myData after processing the data");
  }
}

void loop() {
  if ( TimePeriodIsOver(myNonBlockingTimer,500) ) {
   myFunction1();     myFunction2();
  }
}
```

2) Програма обробки методом SVD

```
import openpyxl
import numpy as np
import matplotlib.pyplot as plt
import pylab
from openpyxl import load_workbook
np.set_printoptions(precision=2, suppress=True)
from numpy.linalg import eigh, norm
from numpy import linalg as LA
wb = openpyxl.reader.excel.load_workbook(filename="pik.xlsx")
print(wb.sheetnames)
wb.active = 0

sheet = wb.active

x = np.zeros((20, 20))
y = np.zeros((20, 20))

for i in range(20):
 x[i, 0] = sheet['J' +str((i+1)*3)].value
for i in range(20):
 y[i, 0] = sheet['A' +str((i+1)*3)].value
f = 230

Y = np.zeros((20, 20))

for i in range (20):
    f=f-20
    print('f')
    print(f)
```

```
    for j in range(20):
        if f <= 20:
            Y[i, j] = x[f, 0]
        if  0>f or f>20:
            Y[i, j] = 0

        f= f+1

print("masuv20x20")
print(Y)

U, s, W = np.linalg.svd(Y)
V = W.T
D = np.zeros_like(Y, dtype=float)
D[np.diag_indices(min(Y.shape))] = s
print('U')
print(U)
print('V')
print(V)
print('D')
print(D)
print("perevirka")
print(np.dot(np.dot(U, D), V.T))

wD, vD = LA.eigh(D)
print('wD')
print(wD.shape)

h = np.zeros((20, 20))
h1 = np.zeros((20, 20))
```

```
for i in range(20):
    h[i, 0] = wD[20 - i]

for i in range(20):
    h1[i, 0] = i
print("wU")
for i in range(20):
    print(wD[20 - i])
print("end")

sU = np.dot(U, vD) - wD*vD
print('sU')
print(sU)

ur = np.zeros((20, 20))
for i in range(20):
    for j in range(20):
        ur[j, i] = U[j, i]
print("ur")
print(ur)

yr = np.zeros((20, 20))
for i in range(20):
 yr[i, 0] = x[i, 0]
print("yr")
print(yr)
C = np.zeros((20, 20))
C = ur.T.dot(yr)
print("C")
for i in range(20):
```

```
    print(C[i, 0])
L = np.zeros((20,20))
for i in range(20):
    L[i, 0] = wD[20 - i]
print("5wD")
print(L)
M = np.zeros((20,20))
for i in range(20):
    M[i, 0] = C[i, 0]/L[i, 0]

print("C/L")
for i in range(20):
    print(M[i, 0])

yrr = np.zeros((20,20))
vr = np.zeros((20, 20))

for i in range(20):
    for j in range(20):
        vr[j, i] = V[j, i]
for i in range(5):
    for j in range(20):
        yrr[j, i] = M[i, 0] * vr[j, i]

print("yrr")
print(yrr.shape)
yrr2 = np.zeros((20,20))

for j in range(20):
    yrr2[j, 0] = yrr[j, 0]+yrr[j, 1]
```

```
print("yrr2 shape")
print(yrr2.shape)

print("yrr2 ")
for i in range(20):
    print(yrr2[i, 0])

xr = np.zeros((20, 20))
for i in range(20):
 xr[i, 0] = sheet['P' + str((i+1))].value

plt.plot(y, yrr2)
#pylab.semilogy(h1, h)
plt.show()



fn = 'Result.XLSX'
wb = load_workbook(fn)
ws = wb['data']

for row in range(20, 20):
    for col in range(20, 20):
        cell = ws.cell(row=row, column=col)
        cell.value = U[(row-1),(col-1)]




wb.save(fn)
wb.close()
```

3) Код датчика (передавача)

```
import machine
import network
import espnow
import math
import urandom
def generate_matrix():
    J = [[0 for _ in range(20)] for _ in range(20)]
    special_values = ["3C", "46", "51", "63", "5C"]

    for i in range(20):
        for j in range(20):
            if i == j:
                J[i][j] = int(special_values[i % 5], 16)
            else:
                J[i][j] = urandom.randint(1, 30)
    return J

def svd_decomposition(J):
    U, D, V = machine.linalg.svd(J)
    return U, D, V

def correct_eigenvalues(D):
    D1 = [0 for _ in range(len(D))]
    for i in range(len(D)):
        D1[i] = abs(D[i])
    return D1

def send_matrix_D1(D1):
```

```
    mac = b'\x12\x34\x56\x78\x90\xab'
    espnow.init()
    peer = espnow.add_peer(mac)
    espnow.send(peer, D1)
def main():

    J = generate_matrix()
    U, D, V = svd_decomposition(J)
    D1 = correct_eigenvalues(D)
    send_matrix_D1(D1)

if __name__ == "__main__":
    main()
```

4) Код централі (приймача)

```
import machine
import espnow
def receive_data():
    K = []
    for i in range(5):
        row = []
        for j in range(4):
            received_number = esp_now.recv_data

            row.append(received_number)
        K.append(row)
    return K
def main():
    K = receive_data()
```

```
def compare_matrices(K, K1):
    G = []

    for i in range(len(K)):
        row = []

        for j in range(len(K[i])):
            if K[i][j] != K1[i][j]:
                return None
            if K1[i][j] < 1 or K1[i][j] > 30:
                row.append(K1[i][j])
            else:
                row.append(0)  # Інакше, додати 0
        G.append(row)
    return G

    K1 = []
    for i in range(len(J)):
        row = []

        for j in range(len(J[i])):
            if J[i][j] < 1 or J[i][j] > 30:
                row.append(J[i][j])
            else:
                row.append(0)

        K1.append(row)
        G = compare_matrices(K, K1)
        if G is not None:
        print("Matrix G:")
```

```
        for row in G:
            print(row)
    else:
        print("Matrices K and K1 do not match!")

if __name__ == "__main__":
    main()
```