EMBEDDED

л.8
Особливості реального світу

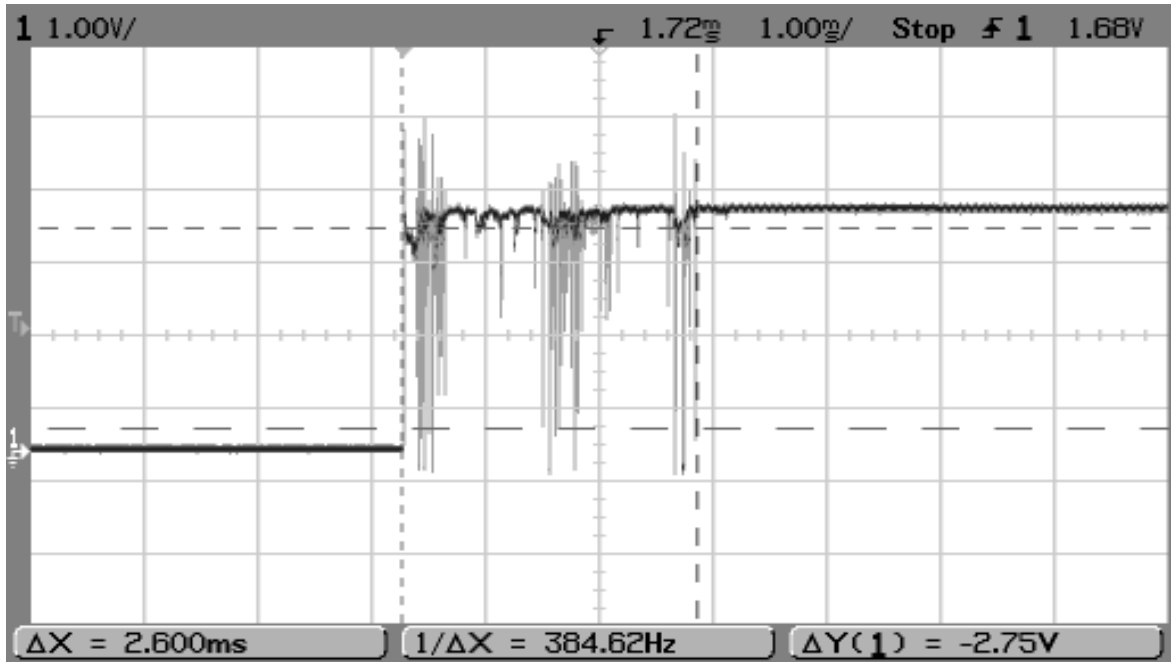
Палій Святослав

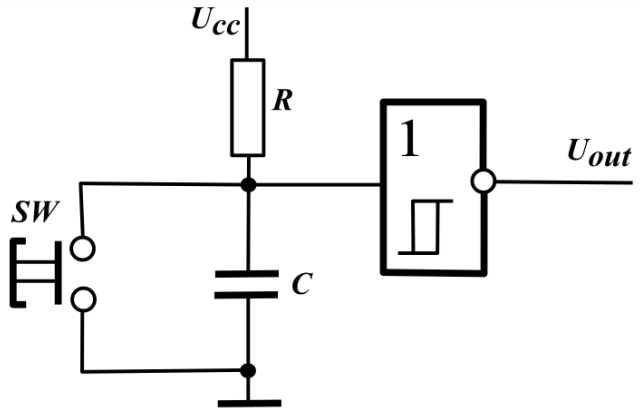
Український Католицький Університет

APPLIED SCIENCES FACULTY

Цифровий сигнал

Навіть цифровий сигнал котрий мав би бути еталоном стабільності не завжди саме таким є



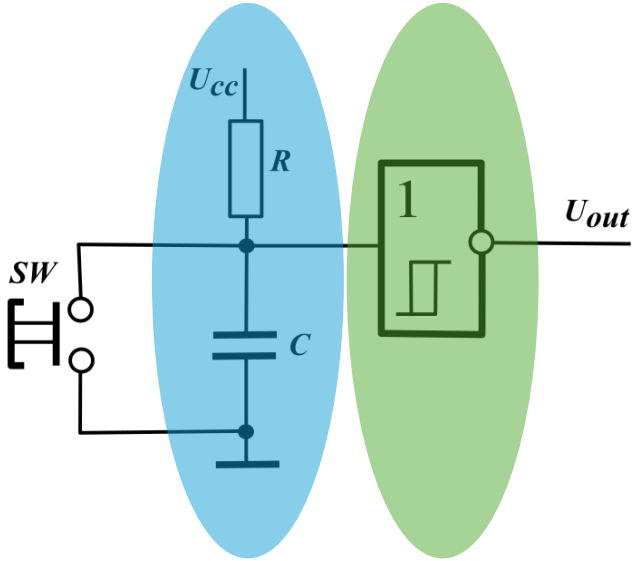


На екрані осцилографа показано брязкіт кнопки.

Цифровий сигнал

В додачу до програмних методів боротьби можна розглядати і апаратні засоби

- RC-фільтр
- Тригер Шмідта



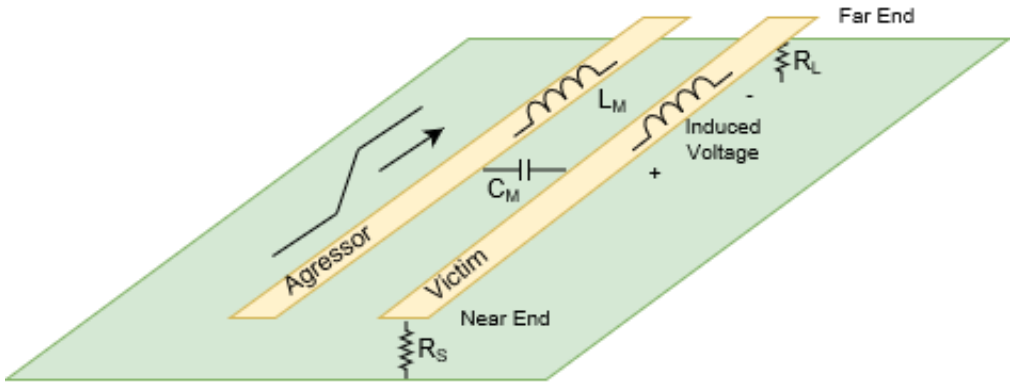
Три́гер Шмі́тта - електронна модель двопозиційного релейного елемента, статична характеристика якого має зону неоднозначності завдяки петлі гістерезиса.

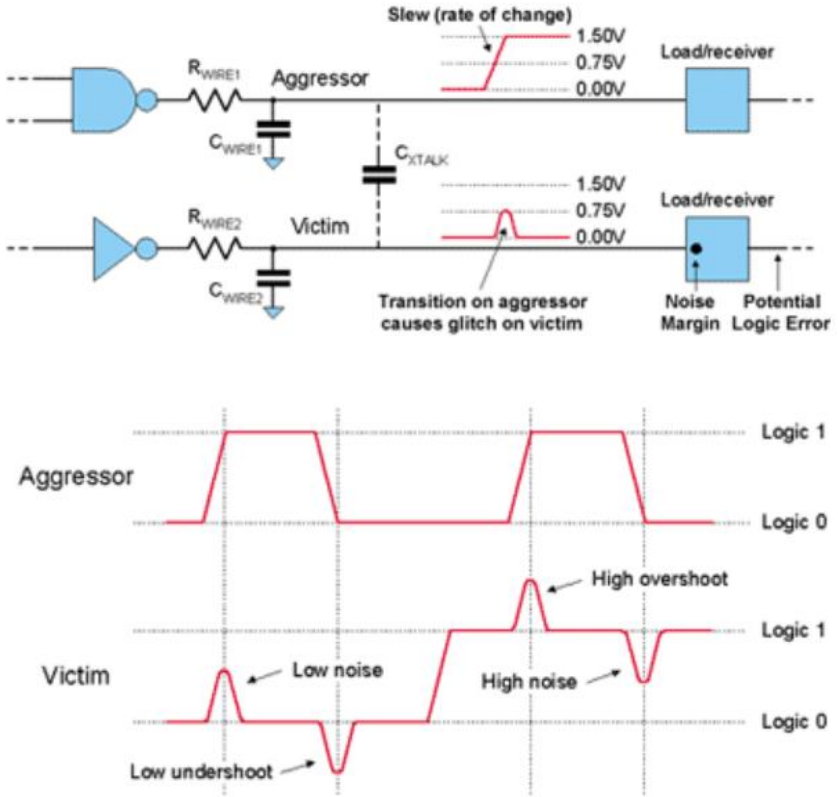
На тригерах Шмітта побудований інтегральний таймер NE555, який вперше випущений в 1971 році і виготовляється донині

* В більшості випадкі зовнішні компоненти здорожчують систему та збільшують її габарити, тому таке вирішення не є поширеним

Ідеалізований vs. реальний сигнал

Перехресні завади (crosstalk) — сигнали завади, що виникають у лінії зв'язку через наявність сигналу у сусідніх лініях передачі.
Завади виникають при зміні стану сигналу на одній з ліній.
Виникнення перехресної завади зв'язане з наявністю ємнісних та індуктивних паразитних зв'язків між лініями передачі даних.
Паразитний зв'язок зменшується при збільшенні відстані між лініями передачі.

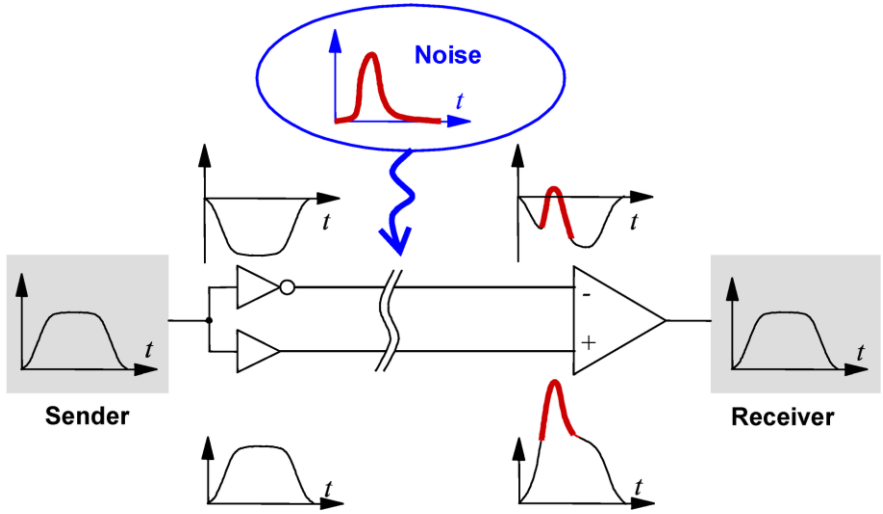


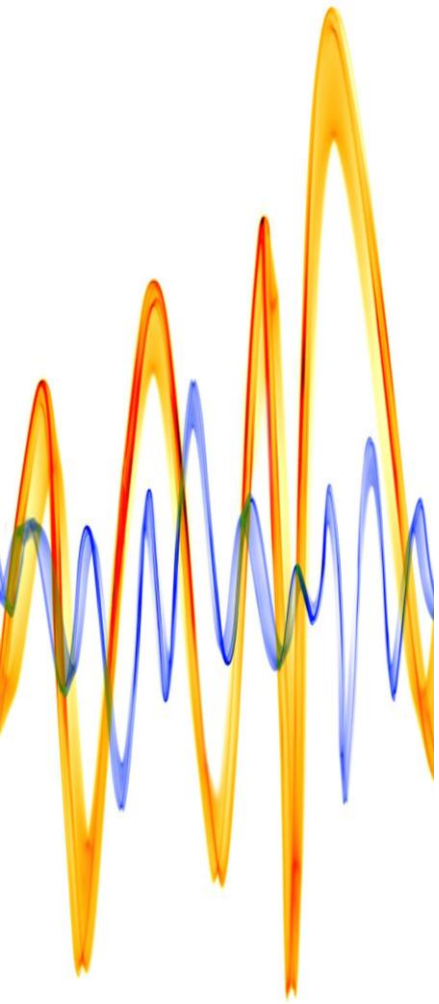


Джерело EETimes:
https://www.eetimes.com/signal-integrity-sign-off-verification/

Подолання:

- Сповільнення фронтів сигналів
- Екранування ліній одна від одної
- Використання диференційної передачі даних - передачі із використанням двох протифазних сигналів, які передаються по двох спарених провідниках, що називаються диференційною парою. Оскільки зовнішні завади, як правило, впливають на обидва дроти однаково а інформація міститься в різниці між ними, техніка покращує стійкість до наведених завад.





Критерії вибору комунікаційного інтерфей

- **I2C (Inter-Integrated Circuit)**
- **Пропускна здатність:** Підходить для менш швидкісних додатків (до 3.4 Мбіт/с).
- **Кількість дротів:** Використовує лише два дроти (SDA та SCL).
- **Кількість пристроїв:** Може підключати велику кількість пристроїв через одну шину.
- **Відстань передачі:** Обмежена, зазвичай в рамках друкованої плати.

- **SPI (Serial Peripheral Interface)**
- **Пропускна здатність:** Забезпечує високошвидкісну передачу даних (до 60 Мбіт/с).
- **Кількість дротів:** Потрібно щонайменше чотири дроти (MISO, MOSI, SCLK, CS).
- **Кількість пристроїв:** Може підключати обмежену кількість пристроїв через індивідуальні лінії CS для кожного з них.
- **Відстань передачі:** Обмежена, зазвичай в рамках друкованої плати.

- **UART (Universal Asynchronous Receiver-Transmitter)**
- **Пропускна здатність:** Невисока швидкість передачі даних (зазвичай до 1 Мбіт/с).
- **Кількість дротів:** Використовує два дроти (TX та RX), а також GND для спільного заземлення.
- **Кількість пристроїв:** Зазвичай використовується для з'єднання двох пристроїв.
- **Відстань передачі:** Може передавати на більші відстані (до 100 метрів з диференційними драйверами).

USB (Universal Serial Bus)
•**Швидкість передачі даних:** Дуже висока (до 20 Гбіт/с у USB 3.2).
•**Відстань передачі:** до 5 метрів для USB 3.x.
•**Застосування:** Підключення периферійних пристроїв, зарядка, передача даних.

Ethernet
•**Швидкість передачі даних:** Висока (до 1 Гбіт/с або більше).
•**Відстань передачі:** До 100 метрів для стандартного кабелю; кілометри для оптоволоконних кабелів.
•**Застосування:** Локальні мережі, стабільне та швидке з'єднання.

Bluetooth
•**Швидкість передачі даних:** Низька до середньої (до 3 Мбіт/с).
•**Відстань передачі:** До 100 метрів для Bluetooth 5.0. Низьке споживання енергії.
•**Застосування:** Підключення малопотужних пристроїв, аудіопристрої, передачі даних на короткі відстані.

WiFi
•**Швидкість передачі даних:** Висока (до 9.6 Гбіт/с для WiFi 6).
•**Відстань передачі:** До 100 метрів. Відносно високе енергоспоживання.
•**Застосування:** Бездротовий доступ до мережі.

Zigbee
•**Швидкість передачі даних:** Низька (до 250 кбіт/с).
•**Відстань передачі:** До 100 метрів, з можливістю розширення через мережу з багатьма вузлами.
•**Споживання енергії:** Дуже низьке, що робить його придатним для пристроїв з живленням від батарейок.
•**Застосування:** Інтернет речей (IoT), автоматизація будівель, датчики, розумні домівки.
•**Особливості:** Підтримує мережі з багатьма вузлами, самовідновлювальна мережа, висока надійність.

•Якщо потрібне стабільне і швидке з'єднання для локальної мережі, вибирайте Ethernet.
•Якщо потрібна передача даних між периферійними пристроями, USB буде найкращим варіантом.
•Для передачі даних на короткі відстані з низьким споживанням енергії Bluetooth є добрим вибором.
•Якщо потрібне бездротове з'єднання для доступу до мереж, вибирайте WiFi.
•IOT: Zigbee або подібні

Аналоговий сигнал

Аналоговий сигнал більш схильний до спотворень при передачі та обробці в порівнянні з цифровим.

Найбільш критичним є утримання співвідношення сигнал-шум (SNR) в допустимих рамках.

Останнім часом є тенденція перетворити аналоговий сигнал в цифрову форму якомога ближче до джерела аналогового сигналу

Цифрова та аналогова фільтрація.

Застосування гістерезису

Застосування технік обмеження спектру сигналу

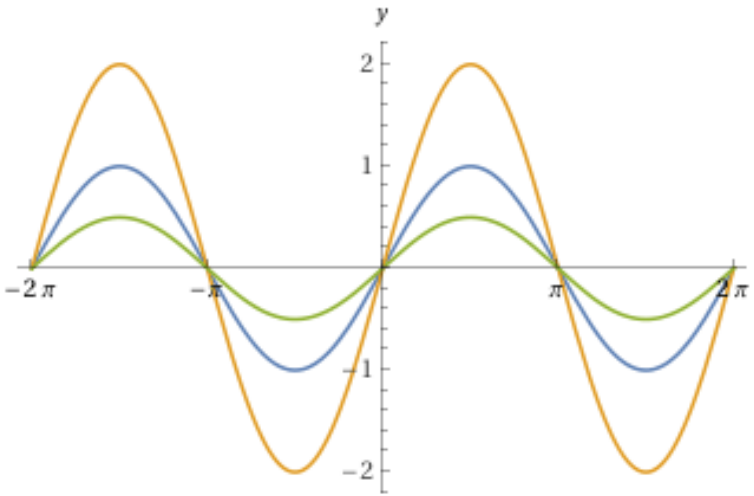
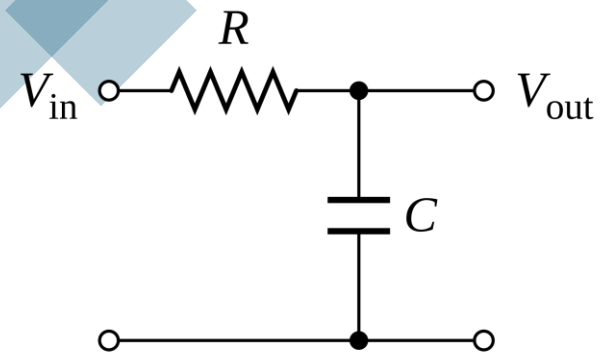
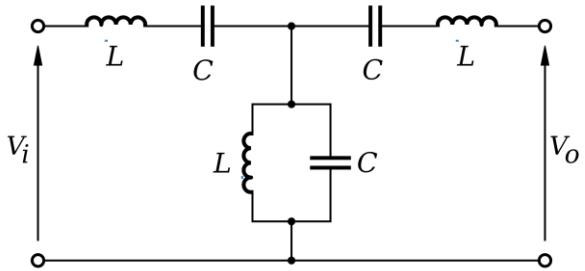
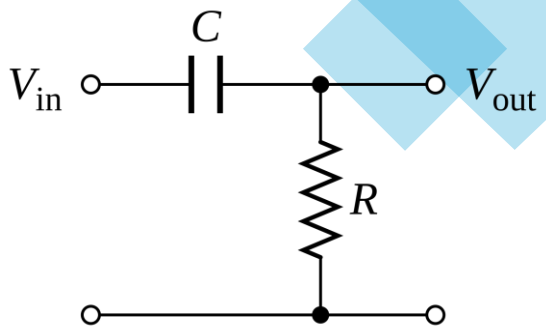


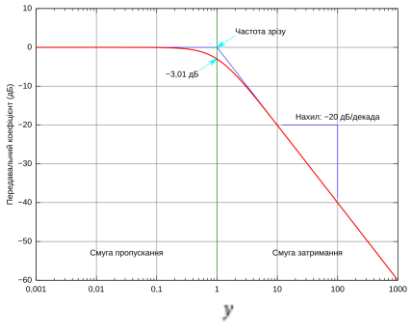
Рисунок з Wikipedia

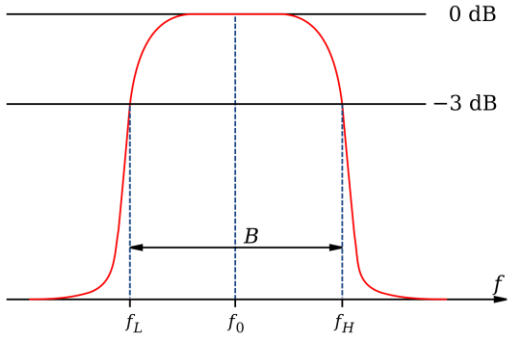
Найпростіші аналогові фільтри

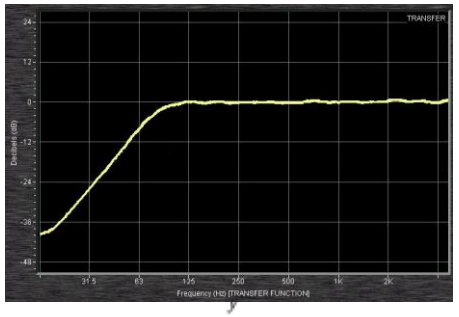


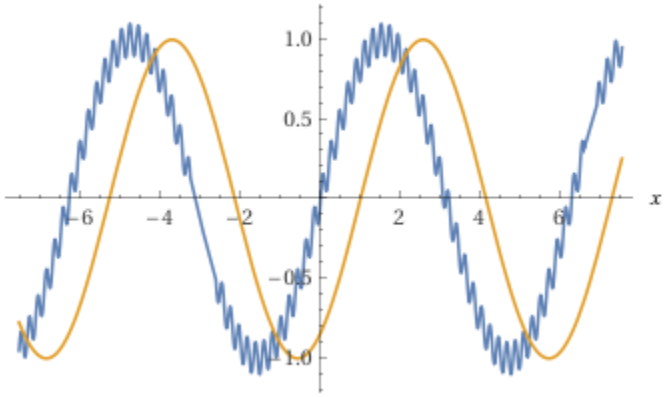


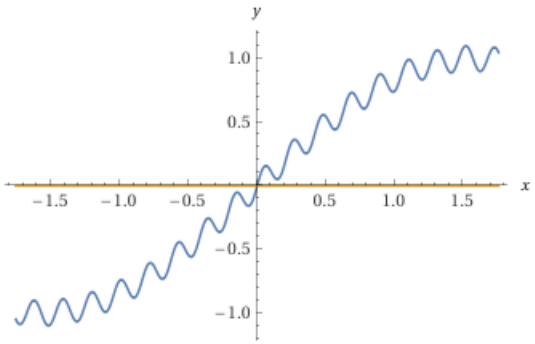


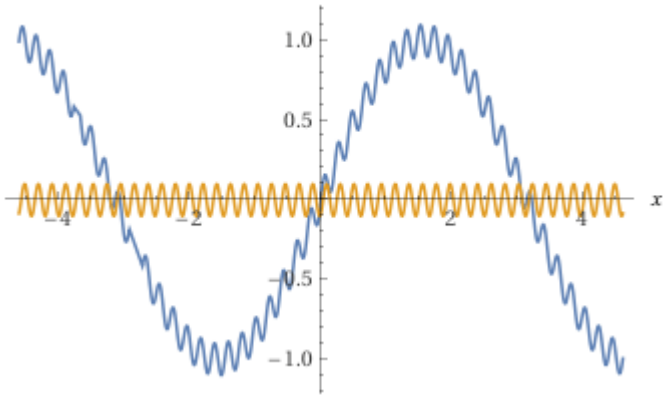












[Дуже просто але трохи детальніше + online калькулятор](https://myproject.com.ua/kalkuliator-reaktyvnoho-oporu-kondensatoriv-ta-kotushok.html)

https://myproject.com.ua/kalkuliator-reaktyvnoho-oporu-kondensatoriv-ta-kotushok.html

```c
#include <stdio.h>

#define ALPHA 0.1  // Коефіцієнт згладжування (0 < ALPHA <= 1.0)

// Функція для застосування фільтра низьких частот
void low_pass_filter(const double* input, double* output, int length) {
    output[0] = input[0];  // Початкове значення вихідного сигналу

    for (int i = 1; i < length; i++) {
        output[i] = ALPHA * input[i] + (1 - ALPHA) * output[i - 1];
    }
}

int main() {
    // Приклад вхідних даних
    double input[] = {1.0, 2.0, 3.0, 4.0, 10.0, 2.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0};
    int length = sizeof(input) / sizeof(input[0]);
    double output[length];

    // Застосування фільтра низьких частот
    low_pass_filter(input, output, length);

    // Виведення результатів
    printf("Filtered Output:\n");
    for (int i = 0; i < length; i++) {
        printf("%f\n", output[i]);
    }

    return 0;
}
```

Наприклад сутінковий давач

```c
#include <stdio.h>

#define ALPHA 0.9  // Коефіцієнт фільтрації (0 < ALPHA <= 1.0)

// Функція для застосування фільтра високих частот
void high_pass_filter(const double* input, double* output, int length) {
    output[0] = input[0];  // Початкове значення вихідного сигналу

    for (int i = 1; i < length; i++) {
        output[i] = ALPHA * (output[i - 1] + input[i] - input[i - 1]);
    }
}

int main() {
    // Приклад вхідних даних
    double input[] = {1.0, 2.0, 3.0, 4.0, 10.0, 2.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0};
    int length = sizeof(input) / sizeof(input[0]);
    double output[length];

    // Застосування фільтра високих частот
    high_pass_filter(input, output, length);

    // Виведення результатів
    printf("Filtered Output:\n");
    for (int i = 0; i < length; i++) {
        printf("%f\n", output[i]);
    }

    return 0;
}
```

Наприклад давач руху

Медіанний фільтр

Щоб продемонструвати, нелінійний віконний фільтр ми використаємо медіанний фільтр з розміром вікна = 3

x = (2, 3, 80, 6, 2, 3). Цей сигнал має переважно невеликі значення, за винятком одного елемента, який є незвично високим і вважається шумовим сплеском, і мета полягає в його усуненні. Отже, медіанний відфільтрований вихідний сигнал y буде:

y0 = med(0, 2, 3) = 2, (граничне значення приймається як 0)

y1 = med(2, 3, 80) = 3, (вже 2, 3 і 80 знаходяться в порядку зростання, тому немає необхідності їх упорядковувати)

y2 = med(3, 80, 6) = med(3, 6, 80) = 6, (3, 80 і 6 упорядковуються для знаходження медіани)

y3 = med(80, 6, 2) = med(2, 6, 80) = 6,

y4 = med(6, 2, 3) = med(2, 3, 6) = 3,

y5 = med(2, 3, 0) = med(0, 2, 3) = 2,

тобто,

- y = (2, 3, 6, 6, 3, 2).

Вимоги до дискретизації
Критерій Найквіста (теорема Котельнікова)

Теорема Найквіста-Шеннона

Якщо неперервний сигнал має спектр, обмежений частотою F_{MAX}, то його можна однозначно і без втрат відновити за дискретними відліками, узятими з частотою $f_{ДИСКР}=2Fmax$

Або обернено (критерій):

Для того, щоб відновити сигнал за його відліками без втрат, необхідно, щоб частота дискретизації була хоча б удвічі більшою за максимальну частоту первинного неперервного сигналу.

$$F_{ДИСКР} \geq 2F_{MAX}$$

Будь-який аналоговий сигнал можна відновити з якою завгодно точністю за його дискретними відліками, взятими з частотою щонайменше вдвічі більшою за максимальну частота, якою обмежений спектр реального сигналу.

Якщо максимальна частота в сигналі перевищує половину частоти дискретизації, то способу відновити сигнал з дискретного в аналоговий без спотворення не існує.

Теорему сформулював Гаррі Найквіст 1928 року. 1933 року подібні дані опублікував В. О. Котельников

Час: Ритмічність даних

- Навіть якщо на перший погляд не важливо зчитування даних з певною визначеною частотою (наприклад просте зчитування положення потенціометра) цифрові методи обробки сигналу передбачають що сигнал дискретизований ритмічно.

- В галузі електроніки **джитер** (*jitter — тремтіння*) це небажані фазові та/або частотні випадкові спотворення сигналу. Виникають внаслідок нестабільної періодичності оцифрування або навпаки перетворення в аналогову форму сигналу.
- Джитер є однією з основних проблем під час проєктування пристроїв цифрової електроніки, зокрема, цифрових інтерфейсів. Недостатньо акуратний розрахунок джитера може привести до його накопичення під час проходження цифрового сигналу по тракту і, зрештою, до непрацездатності пристрою.

Відлагодження систем реального часу

- **Як правило техніка покрокового виконання коду не є ефективною а часом і неможливою. Це відбувається через те, що поки ми заповільнюємо систему для відлагодження, час не змінює свого плину.**

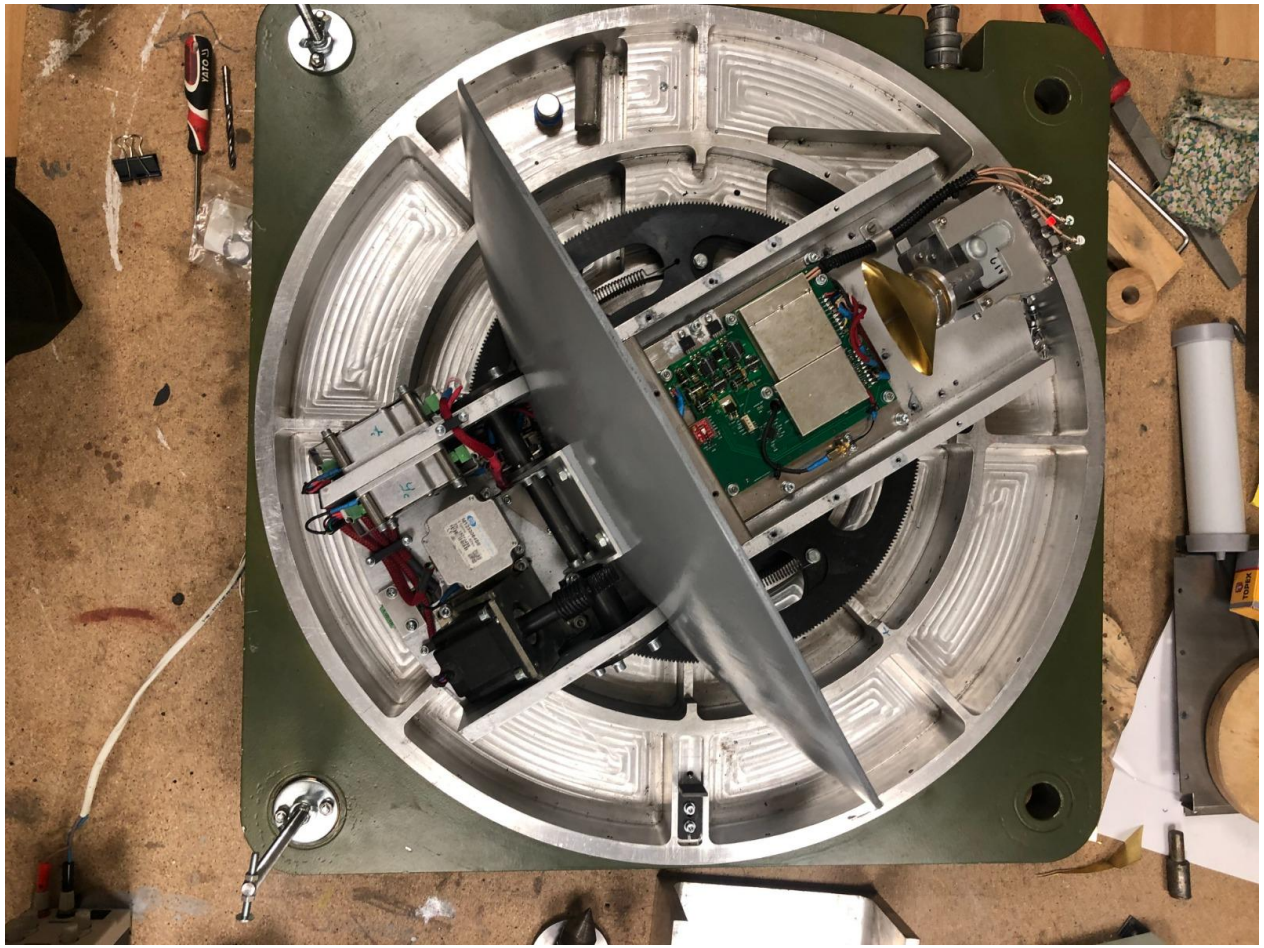
- **Для відлагодження взаємодії з часом можна використовувати**
  - **Вивід інформації додатковим каналом, при потребі перетворюючи дані в графічну форму.**
  - **Пастки для даних. При настанні певних умов дані починають виводитися додатковим каналом або навпаки логуються (записуються в системі)**
  - **Популярним способом є запис даних в ситуації, котра потребує відлагодження і подальший аналіз та відтворення в тому числі з використанням мов високого рівня.**

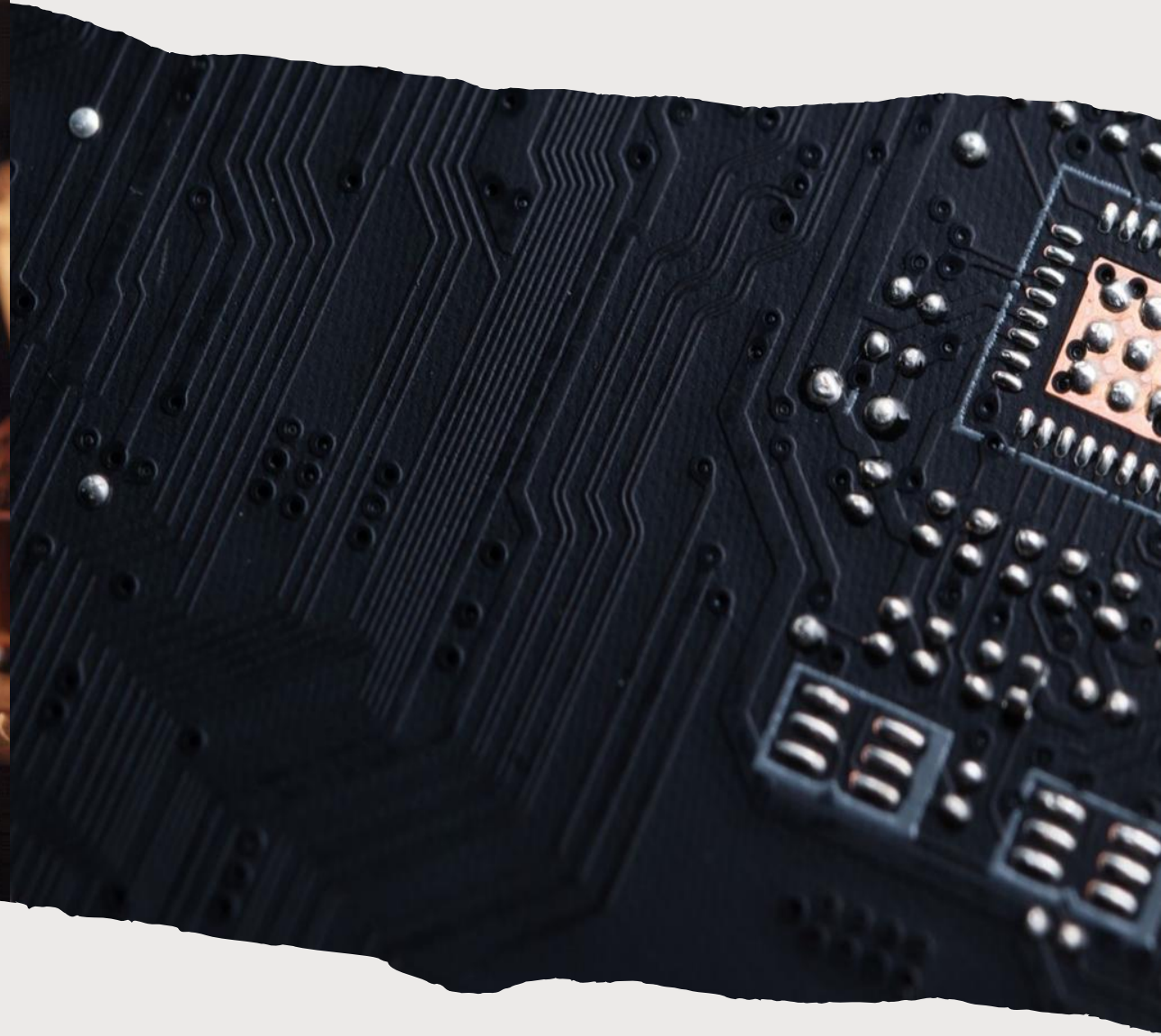
  - **Також можлива зворотна ситуація: пристрій тестується не в реальних умовах а за допомогою наперед записаного очікуваного (або реального з попереднього разу, коли була нештатна поведінка) сигналу.**
  - **Емуляція (відтворення з запису, тощо) зовнішнього світу повертає можливість покрокового виконання ☺**

- **Що ще стосовно часу**

- В системах де використовуються переривання, обов'язково має бути оцінено та виміряно скільки триває обробка переривань і які найгірші очікування виклику обробника переривання на подію. Це саме стосується критичних секцій, в яких обробка переривань заборонена.

Приклад запису та фільтра





Щиро дякую!