МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп’ютерних систем та інфокомунікацій

Кафедра комп’ютерних систем, мереж і кібербезпеки

**Курсова робота**

|  |  |
| --- | --- |
| з | IOT |
|  | (назва дисципліни) |
| на тему | «Сканер цифрових сигналів на базі arduino» |

ХАІ.503.535СТ2.123.9631822 ПЗ

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Виконав: здобувач (ка) | | | *3* | курсу групи № | | | | *535ст2* |
| напряму підготовки (спеціальності) | | | | | | | | |
| *123 – комп’ютерна інженерія* | | | | | | | | |
| (шифр і назва напряму підготовки (спеціальності)) | | | | | | | | |
| *Саєнко Олександр Олександрович* | | | | | | | | |
| (прізвище й ініціали здобувача) | | | | | | | | |
| Керівник: | *доцент закладу вищої освіти, к.т.н. Куланов В.О.* | | | | | | | |
| (посада, науковий ступінь, прізвище й ініціали) | | | | | | | | |
| Національна шкала: | | | | |  | | | |
| Кількість балів: | | | | |  | | | |
| Члени комісії: | |  | | | |  |  | |
|  | | (підпис) | | | |  | (прізвище й ініціали) | |
|  | |  | | | |  |  | |
|  | | (підпис) | | | |  | (прізвище й ініціали) | |
|  | |  | | | |  |  | |
|  | | (підпис) | | | |  | (прізвище й ініціали) | |

Харків – 2023

# ЗМІСТ

[ЗМІСТ 2](#_Toc152620189)

[СПИСОК СКОРОЧЕНЬ 6](#_Toc152620190)

[1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 7](#_Toc152620191)

[1.1 Аналіз предметної області 7](#_Toc152620192)

[1.2 Постановка задачі: 8](#_Toc152620193)

[1.2.1 Мета проекту: 8](#_Toc152620194)

[1.2.2 Завдання проекту: 8](#_Toc152620195)

[1.2.3 Очікувані результати: 9](#_Toc152620196)

[2 ПРОЕКТУВАННЯ 9](#_Toc152620197)

[2.1 Апаратна частина 9](#_Toc152620198)

[2.1.1 ППВМ (персональний програмно-обчислювальний модуль): 10](#_Toc152620199)

[2.1.2 Arduino: 11](#_Toc152620200)

[2.1.3 Raspberry Pi: 12](#_Toc152620201)

[2.1.4 Вибір Arduino: 13](#_Toc152620202)

[2.2 Програмна частина: 14](#_Toc152620203)

[3 РОЗРОБЛЕННЯ 16](#_Toc152620204)

[3.1 Розробка поведінки 16](#_Toc152620205)

[3.2 Проектування коду Arduino 17](#_Toc152620206)

[3.3 Проектування додатку 19](#_Toc152620207)

[3.3.1 Проектування коду додатку 19](#_Toc152620208)

[3.3.2 Проектування інтерфейсу програми 20](#_Toc152620209)

[3.4 Проектування протоколу передачі інформації 22](#_Toc152620210)

[4 ВЕРИФІКАЦІЯ ТА ТЕСТУВАННЯ 24](#_Toc152620211)

[ВИСНОВКИ 27](#_Toc152620212)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 28](#_Toc152620213)

[ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ. 29](#_Toc152620214)

[4.1 А.1 Вступ 29](#_Toc152620215)

[4.2 А.2 Підстави для розробки 29](#_Toc152620216)

[4.3 А.3 Призначення розробки 29](#_Toc152620217)

[4.4 А.4 Вимоги до програми або програмного виробу 29](#_Toc152620218)

[А.4.1 Вимоги до функціональних характеристик: 29](#_Toc152620219)

[А.4.2 Вимоги до складу і параметрів технічних засобів: 30](#_Toc152620220)

[А.4.3 Вимоги до інформаційної та програмної сумісності: 30](#_Toc152620221)

[А.4.4 Спеціальні вимоги: 30](#_Toc152620222)

[4.5 А.5 Вимоги до програмної документації 31](#_Toc152620223)

[ДОДАТОК Б. ВИХІДНІ КОДИ. 32](#_Toc152620224)

[4.5.1 Код на Arduino 32](#_Toc152620225)

[4.5.2 Документація, з CoDeSys: 55](#_Toc152620226)

[ДОДАТОК В. НАСТАНОВА КОРИСТУВАЧУ. 56](#_Toc152620227)

[1 Відомості про призначення 56](#_Toc152620228)

[5 Умови, необхідні для виконання 57](#_Toc152620229)

[5.1 Апаратні засоби: 57](#_Toc152620230)

[5.2 Програмні засоби: 57](#_Toc152620231)

[5.3 Документація та ресурси: 57](#_Toc152620232)

[5.4 Знання та навички: 58](#_Toc152620233)

[6 Послідовність дій користувача 58](#_Toc152620234)

[6.1 Інсталяція CoDeSys 58](#_Toc152620235)

[6.2 Конфігурація з'єднання 59](#_Toc152620236)

[6.3 Інструкції щодо використання візуалізації 64](#_Toc152620237)

[7 Тексти повідомлень, опис їх змісту та відповідні дії користувача: 65](#_Toc152620238)

[7.1 Повідомлення: "Шина не підключена ". 65](#_Toc152620239)

[7.2 Повідомлення: "R:0 G:0 B:0". 65](#_Toc152620240)

[7.3 Повідомлення: "Немає зв'язку з пристроєм". 65](#_Toc152620241)

[ДОДАТОК Г. ПРЕЗЕНТАЦІЯ. 67](#_Toc152620242)

# СПИСОК СКОРОЧЕНЬ

КР – коротке замикання

ПК – персональний комп’ютер

РК-дисплей - Рідкокристалічний дисплей

ПЗ – програмне забезпечення

CAD - computer-aided design

I2C - Inter-Integrated Circuit

FBD – Functional Block Diagram

IDE - integrated development environment

SCADA - Supervisory Control And Data Acquisition

SCL – Serial clock

SDA – Serial data

# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## Аналіз предметної області

Поточний стан ринку сканерів логічних сигналів:

Відлагоджувальний процес електронних схем з використанням програмованих пристроїв вимагає ефективних інструментів, і сканери логічних сигналів стають важливим компонентом цього процесу. У цьому контексті сканери функціонують як доступніша альтернатива дорогим осцилографам, дозволяючи інженерам та ентузіастам проводити аналіз цифрових сигналів з більш обмеженими бюджетними ресурсами.

Однак, незважаючи на їхню важливість, сьогоднішній ринок логічних сканерів залишається відносно малодослідженим та обмеженим [1] у реченні. Відсутність різноманітності моделей і виробників призводить до того, що існуючі пристрої стають дорогими та не завжди доступними для широкого кола користувачів. Ця ситуація створює потребу у розробці більш доступних і функціональних альтернатив, які б задовольнити потреби як професійних інженерів, і студентів, котрі займаються електронікою.

При цьому, сканери логічних сигналів є важливим інструментом для аналізу цифрових сигналів у широкому спектрі додатків: від розробки та налагодження вбудованих систем до створення прототипів і навчальних проектів. Успішний розвиток доступних та ефективних логічних сканерів може суттєво покращити умови роботи інженерів та електронників, які зіткнулися з обмеженнями поточного ринку.

Вимоги до логічних сканерів:

У більшості сценаріїв застосування логічних сканерів важливими характеристиками є надійність, доступність та простота використання. Існуючі моделі не завжди задовольняють цим вимогам, особливо у контексті бюджетних проектів та лабораторних робіт.

## Постановка задачі:

### Мета проекту:

Метою даного курсового проекту є розробка доступного та функціонального логічного сканера сигналів на базі Arduino. Проект спрямований на усунення проблеми дорожнечі та обмеженої доступності існуючих рішень.

### Завдання проекту:

* Вибір платформи: Визначення сумісної з бюджетними вимогами платформи, що базується на Arduino, для реалізації логічного сканера.
* Розробка апаратної частини: Створення схеми та монтажу пристрою з урахуванням вимог до надійності та простоти використання.
* Програмна реалізація: Написання програмного коду для Arduino, що забезпечує збирання та передачу даних про логічні сигнали на комп'ютер.
* Інтерфейс користувача: Розробка інтерфейсу користувача на комп'ютері для зручного моніторингу та аналізу даних, що отримуються від логічного сканера.
* Документація: Підготовка докладної документації, включаючи схеми, код, інструкції зі складання та використання для забезпечення можливості відтворення проекту іншими користувачами.

### Очікувані результати:

Очікується, що розроблений логічний сканер на базі Arduino стане доступним та ефективним інструментом для інженерів, студентів та ентузіастів, що працюють із цифровими сигналами. Цей проект також надасть основу для подальших досліджень та покращень у галузі розробки бюджетних електронних пристроїв.

# ПРОЕКТУВАННЯ

## Апаратна частина

У стадії проектування логічного сканера було розглянуто кілька платформ: ППВМ (персональний програмно-обчислювальний модуль), Arduino і Raspberry Pi.

### ППВМ (персональний програмно-обчислювальний модуль):

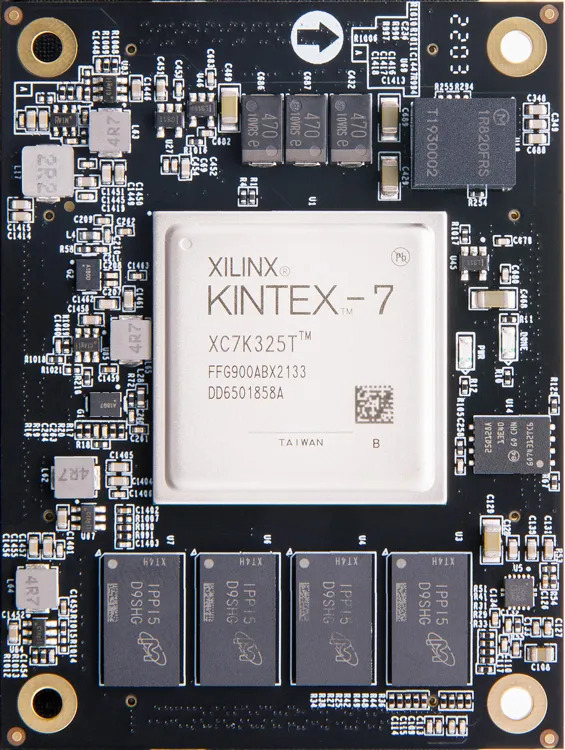


Рисунок 2.1 – ППВМ XILINX Kintex7 SoM FPGA Core Board XC7K325T

ППВМ [2] є універсальні обчислювальні пристрої з широким спектром функціональності. Проте, їхня висока вартість і складність інтеграції роблять їх менш переважними для цього проекту. Переваги обчислювальної потужності можуть бути надмірними для простих завдань логічного сканування.

Переваги:

* Висока обчислювальна потужність.
* Різноманітність функціональних можливостей.

Недоліки:

* Висока вартість.
* Складність інтеграції для найпростіших проектів.

### Arduino:

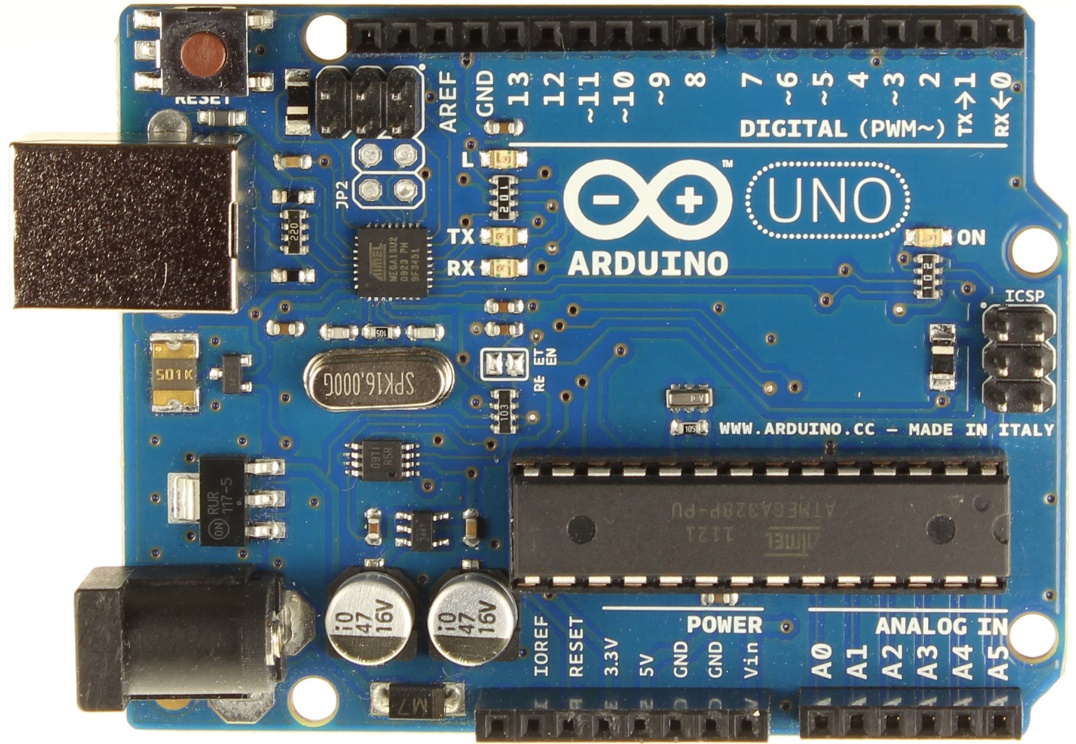


Рисунок 2.2 – Arduino uno

Arduino [3] є бюджетним і простим у використанні рішенням, що ідеально підходить для проектів з обмеженим бюджетом. Вони мають достатню обчислювальну потужність для завдань логічного сканування, при цьому надають зручні засоби розробки.

Переваги:

* Доступна вартість.
* Простота у використанні та програмуванні.

Недоліки:

* Обмежена обчислювальна потужність для складних завдань.
* Обмежений обсяг оперативної пам'яті.

### Raspberry Pi:



Рисунок 2.3 - Raspberry Pi 1 Model B+

Raspberry Pi [4] пропонують більш високу обчислювальну потужність порівняно з Arduino та прийнятну вартість. Однак для простих завдань логічного сканування їх надмірна функціональність може зробити їх зайвими.

Переваги:

* Висока обчислювальна потужність.
* Широкий набір інтерфейсів.

Недоліки:

* Великі ціни в порівнянні з Arduino.
* Більш складне налаштування та використання.

### Вибір Arduino:

Після ретельного аналізу, для нашого логічного сканера було вирішено використовувати Arduino. Вона поєднує в собі прийнятну обчислювальну потужність для задач логічного сканування, доступну вартість та простоту використання. Технічні характеристики вибраної платформи Arduino Uno наведені у таблиці:

Таблица 2.1 -Хариктеристики arduino

|  |  |
| --- | --- |
| Характеристика | Значення |
| Мікроконтролер | ATmega328P |
| Робоча частота | 16 МГц |
| Число входів/виходів | 14 цифрових, 6 аналогових |
| Об'єм оперативної пам'яті | 2 КБ |
| Напруга | 5 В |

Вибір Arduino обумовлений її відповідним поєднанням доступності, простоти використання та достатньої обчислювальної потужності для завдань логічного сканування.

## Програмна частина:

Середою розробки коду у Arduino стала є ARDUINO IDE 2.0

ARDUINO IDE 2.0 - це інтегроване середовище розробки (IDE) для програмування платформи Arduino. Воно є одним з найпопулярніших інструментів для розробки програмного забезпечення для Arduino, завдяки своїй простоті використання та зручному інтерфейсу.

За допомогою ARDUINO IDE 2.0 можна створювати проекти для різних плат Arduino, включаючи Arduino Uno, Mega, Nano та інші. В програмі доступні різні інструменти для розробки, такі як текстовий редактор з підсвічуванням синтаксису, консоль для відлагодження коду та інші.

Більше того, ARDUINO IDE 2.0 містить велику кількість готових бібліотек для швидкої розробки програмного забезпечення, які можуть бути використані для роботи з різними датчиками, екранами та іншими пристроями. Це значно спрощує процес розробки та зменшує час, необхідний для написання коду.

Отже, в нашому проекті ми використовували ARDUINO IDE 2.0 для створення виконавчого коду для нашого Arduino, який був написаний за допомогою мови програмування С++.

Visual Studio з плагіном ZedGraph:

Для створення інтерфейсу користувача і візуалізації даних на ПК ми вибираємо Visual Studio, потужне інтегроване середовище розробки. Це зумовлено її багатим набором інструментів та підтримкою різних мов програмування.

Задіяння плагіна ZedGraph у Visual Studio значно полегшить побудову графіків та візуалізацію результатів сканування. ZedGraph пропонує гнучкі можливості для створення високоякісних графіків, що важливо для аналізу логічних сигналів.

Переваги Visual Studio з плагіном ZedGraph:

* Потужне середовище розробки: Visual Studio забезпечує широкий функціонал для створення різноманітних програм, включаючи графічний інтерфейс та обробку даних.
* Гнучкість візуалізації з ZedGraph: Плагін ZedGraph дозволяє легко вбудовувати барвисті та інформативні графіки в інтерфейс користувача, забезпечуючи наочне представлення даних.

Недоліки Visual Studio із плагіном ZedGraph:

* Обсяг ресурсів: Visual Studio може споживати значні ресурси системи, що може позначитися на продуктивності програми.
* Складність для новачків: Для тих, хто тільки-но починає освоювати розробку, Visual Studio може здатися складною через багатство функціоналу.

Вибір Visual Studio з плагіном ZedGraph обґрунтований бажанням використовувати потужні інструменти для розробки інтерфейсу користувача та ефективної візуалізації даних, що важливо для роботи з результатами логічного сканування.

# РОЗРОБЛЕННЯ

## Розробка поведінки

В ході виконання процесу логічного сканування Arduino проводить періодичне зчитування цифрових сигналів з мінімальним часовим інтервалом. Ці сигнали можуть бути даними про стан різних входів або іншу інформацію, що підлягає моніторингу.

Після успішного зчитування Arduino формує JSON-файл, структуруючи інформацію про сигнали. Формат JSON забезпечує зручний та стандартизований спосіб подання даних. Кожен JSON-файл включає тимчасові мітки та значення сигналів, що дозволяє подальшому аналізу та візуалізації.

Сформований JSON-файл відправляється Serial порту на персональний комп'ютер (ПК), де вже запущена спеціалізована програма, створена в середовищі розробки Visual Studio. Програма на ПК очікує на прихід JSON-файлів по Serial порту і здійснює їх прийом.

Після прийому JSON-файлу програма аналізує його вміст, витягує інформацію про сигнали і перетворює її на зручний формат для візуалізації. Тут важливу роль відіграє використання плагіна ZedGraph Visual Studio, який надає інструменти для побудови графіків.

Програма відображає дані у вигляді графіків, дозволяючи оператору або розробнику моніторити та аналізувати динаміку зміни логічних сигналів у реальному часі. Графіки можуть містити тимчасові осі, значення сигналів та інші параметри, забезпечуючи повну інформацію про стан системи.

Таким чином, весь процес – від зчитування сигналів на Arduino до візуалізації даних на ПК – є ефективною системою моніторингу та аналізу цифрових сигналів, забезпечуючи прозорість і зручність у роботі з даними.

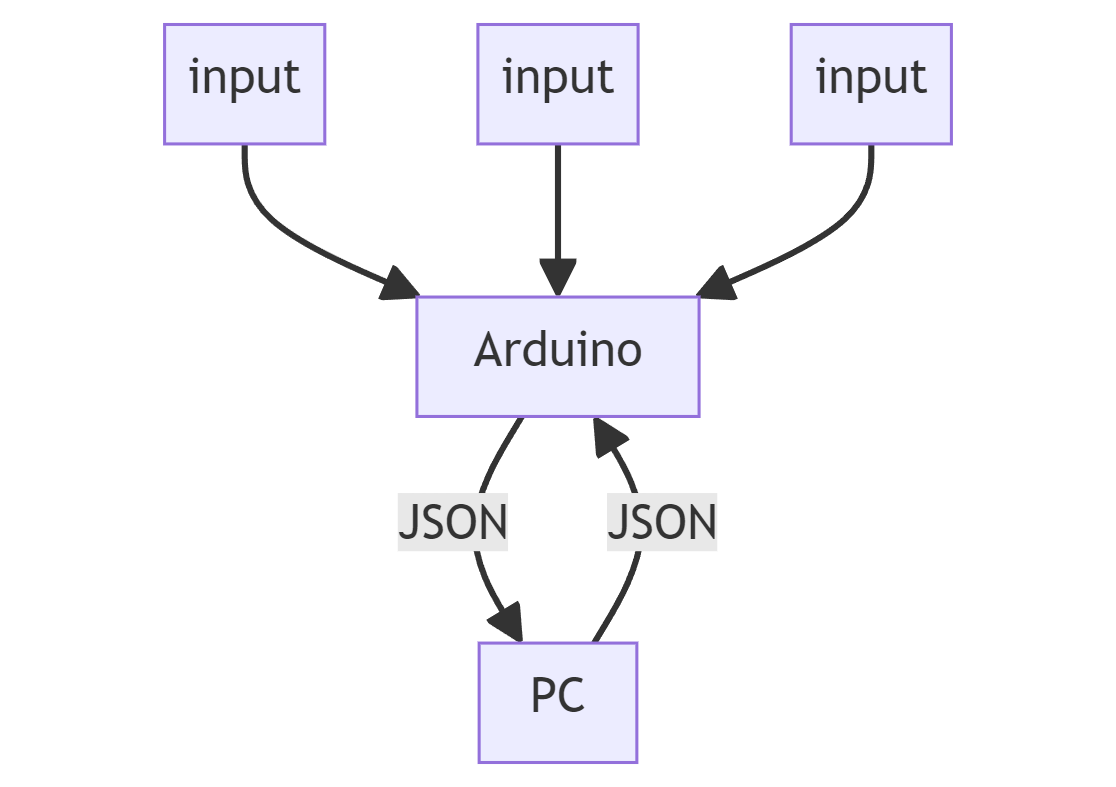


Рисунок 3.1 - Алгоритм роботи Програми.

## Проектування коду Arduino

На початку Arduino налаштовує кілька цифрових входів для зчитування сигналів. Потім, у нескінченному циклі, Arduino перевіряє наявність даних на Serial порту. При отриманні команди "check" Arduino відправляє підтвердження і переходить в режим очікування нової команди.

Коли на ПК приходить команда gv (запит на передачу значень сигналів), Arduino активує режим передачі даних. У цьому режимі лічені значення з цифрових входів формуються в JSON структуру. Кожне отримане значення разом з міткою часу додається масив, який потім серіалізується в JSON формат і відправляється на ПК через Serial порт.

Код також забезпечує обробку команд "ngv" (скасування передачі значень) та "check" для підтвердження готовності до нової команди. Це забезпечує ефективне управління процесом зчитування та передачі даних, забезпечуючи точність та своєчасність моніторингу логічних сигналів.

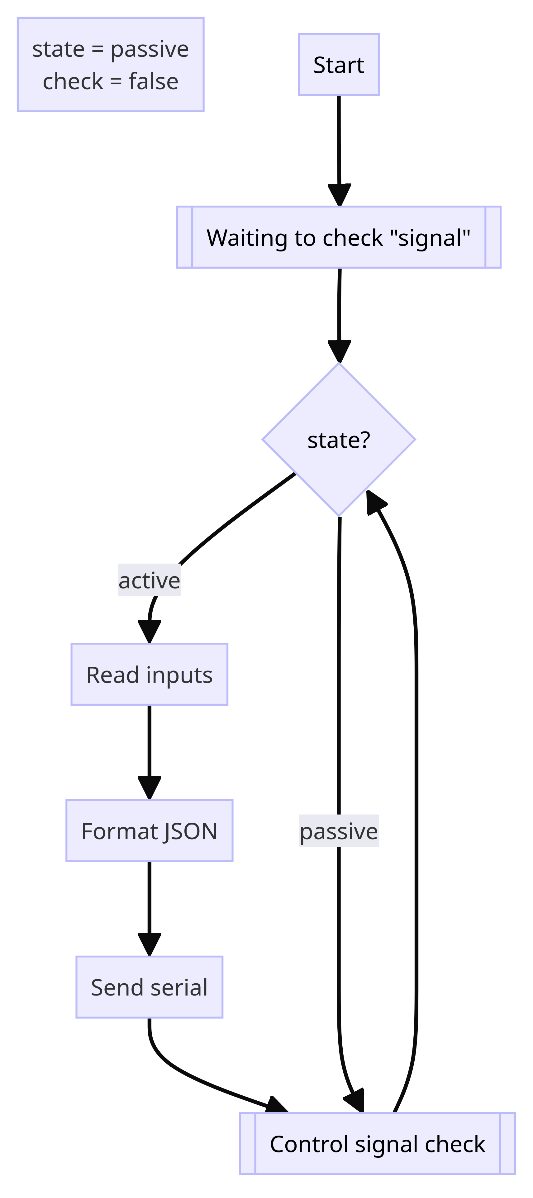


Рисунок 3.2 Алгоритм роботи arduino

## Проектування додатку

### Проектування коду додатку

В першу чергу, програма ініціалізує графічний інтерфейс, включаючи графік ZedGraph, який використовуватиметься для візуалізації даних. Також відбувається налаштування параметрів Serial порту та вибору COM-порту та швидкості передачі.

Основна логіка роботи програми полягає у обробці команд, отриманих від Arduino. Програма відправляє команду "check" для перевірки з'єднання, після чого активує режим прийому даних. Отримані JSON-файли десеріалізуються, витягуючи значення сигналів та їх тимчасові мітки. Ці дані використовуються для оновлення графіка ZedGraph.

Інтерфейс користувача надає кнопки для управління процесом передачі даних: запуск, зупинка, інверсія сигналів. Таймер оновлення графіка забезпечує плавне оновлення відображення даних у реальному часі. Таким чином, додаток забезпечує зручну взаємодію з даними, отриманими від Arduino, та їхнє наочне уявлення у вигляді графіка.

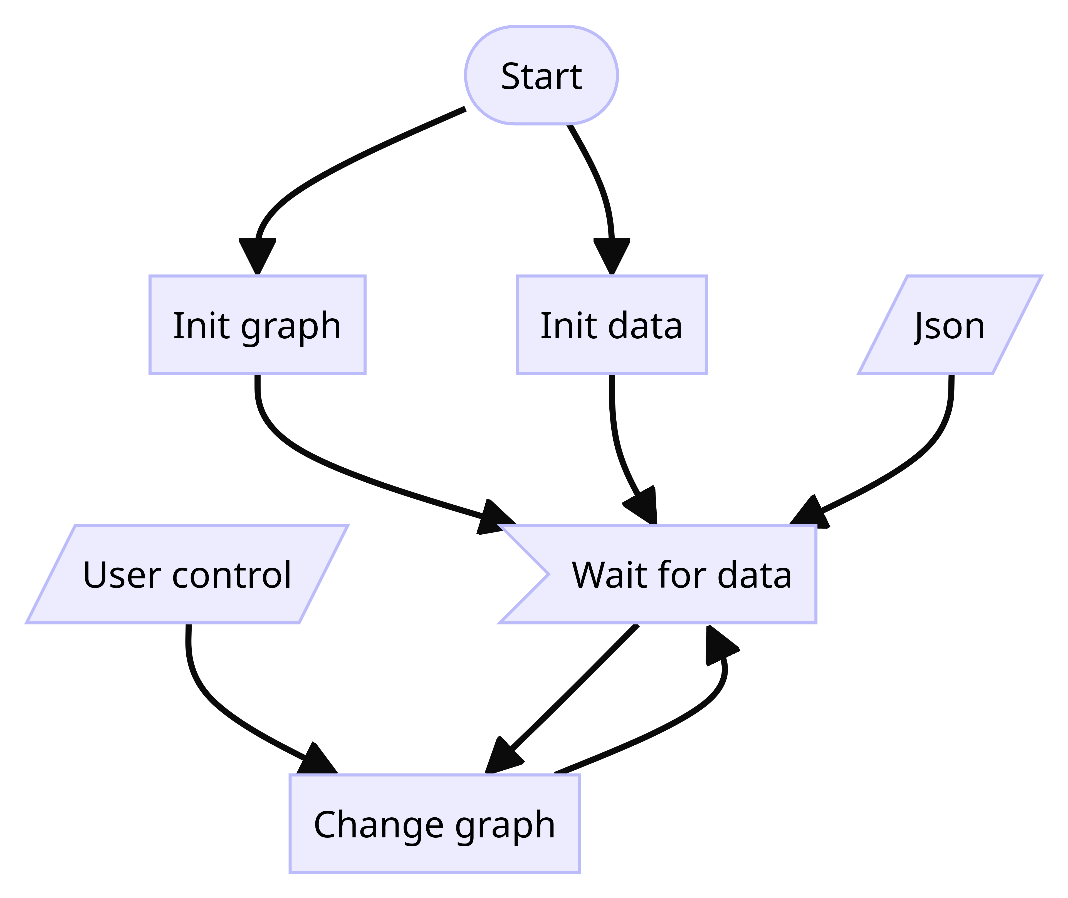


Рисунок 3.3 Алгоритм роботи додатку

### Проектування інтерфейсу програми

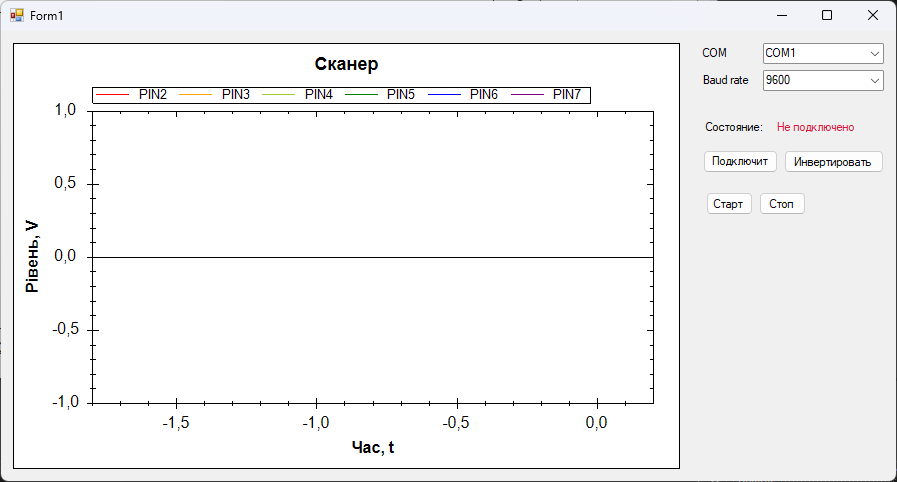


Рисунок 3.4 Загальний вигляд вікна

#### Графік ZedGraph:

- Призначення: Подання візуалізації логічних сигналів як графіка.

- Опис: Графік відображає динаміку зміни станів цифрових сигналів із часом. Кожна лінія графіка відповідає одному цифровому входу, а осі позначають час та величину сигналу.

#### Порт для підключення:

- Призначення: Вибір COM-порту для підключення Arduino.

- Опис: Дозволяє вибрати порт, через який здійснюється зв'язок з Arduino.

#### Швидкість передачі:

- Призначення: Вибір швидкості передачі для взаємодії з пристроєм.

- Опис: Дозволяє налаштувати швидкість передачі даних між програмою та Arduino.

#### Статус підключення:

- Призначення: відображення стану підключення.

- Опис: Індикатор стану підключення: "Підключено" (зелений текст) при успішному підключенні та "Не підключено" (червоний текст) за відсутності підключення.

#### Ініціювати підключення:

- Призначення: Запуск процедури підключення до Arduino.

- Опис: При натисканні кнопки відбувається встановлення зв'язку з Arduino, оновлення графіка та відображення статусу підключення.

#### Почати передачу сигналів та Зупинити передачу сигналів:

- Призначення: Управління передачею даних Arduino на ПК.

- Опис: Натискання "Старт" запускає передачу даних, "Стоп" зупиняє передачу. Стан кнопок відповідає поточному режиму програми.

#### Інвертувати сигнали:

- Призначення: Зміна інверсії сигналів.

- Опис: При натисканні кнопки відбувається інверсія сигналів, що корисно для адаптації під різні логічні пристрої.

#### Таймер оновлення графіка:

- Призначення: періодичне оновлення графіка.

- Опис: Таймер забезпечує плавне оновлення графіка із заданою періодичністю, що забезпечує наочне відстеження змін сигналів у реальному часі.

## Проектування протоколу передачі інформації

Таблица 3.1 Структура JSON файлу з Arduino

|  |  |  |
| --- | --- | --- |
| **Tag** | **Type** | **Desc** |
| PIN2 | Digital | Состояние цифрового входа 2 |
| PIN3 | Digital | Состояние цифрового входа 3 |
| PIN4 | Digital | Состояние цифрового входа 4 |
| PIN5 | Digital | Состояние цифрового входа 5 |
| PIN6 | Digital | Состояние цифрового входа 6 |
| PIN7 | Digital | Состояние цифрового входа 7 |
| Time | Integer | Метка времени в миллисекундах |

Приклад JSON файлу з Arduino:

{

"v": [

{"PIN2": 1},

{"PIN3": 0},

{"PIN4": 1},

{"PIN5": 0},

{"PIN6": 1},

{"PIN7": 0}

],

"t": {"Time": 1578901234567}

}

# ВЕРИФІКАЦІЯ ТА ТЕСТУВАННЯ

Для перевірки функціональності сканера цифрових сигналів на базі Arduino було проведено низку тестів, у ході яких використовувалася друга Arduino для емуляції сигналів.

Тест 1: Підключення та ініціалізація

Опис: Підключення Arduino до ПК та ініціалізація програми.

Кроки:

* Запустити програму на ПК.
* Вибрати відповідний COM-порт та швидкість передачі даних.
* Натиснути кнопку "Підключити".

Очікуваний результат: статус підключення на екрані повинен змінитися на "Підключено", графік повинен бути ініціалізований.

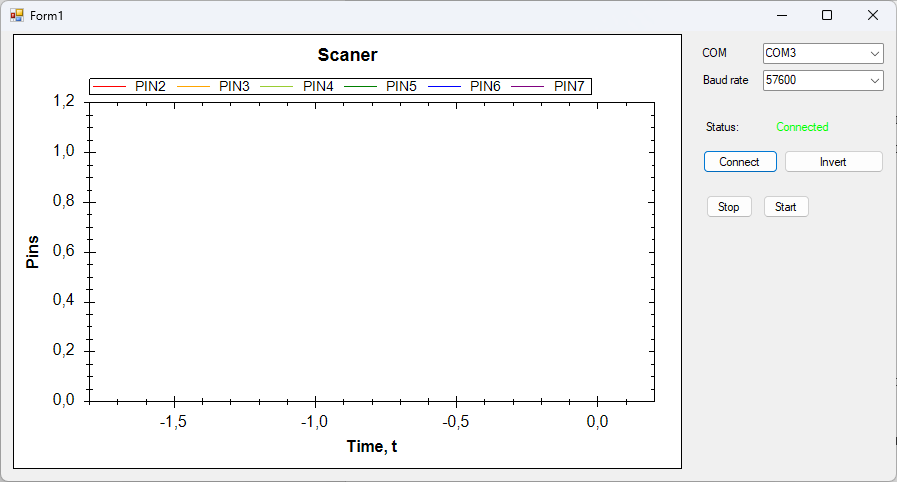


Рисунок 4.1 – Результат 1го тесту

Тест 2: Передача та візуалізація сигналів

Опис: Передача сигналів з другого Arduino та їх візуалізація на графіку.

Кроки:

* Запустити програму на другий Arduino, що емулює сигнали на пінах 2-7.
* Натиснути кнопку "Start" на ПК.

Очікуваний результат: Графік повинен відображати зміни стану сигналів у реальному часі.

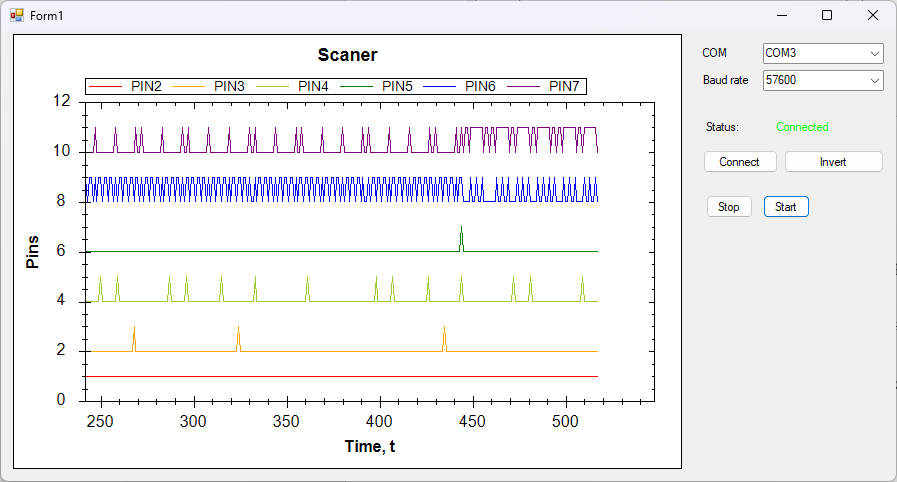


Рисунок 4.2 - Результат 2го тесту

Тест 3: Зупинення передачі сигналів

Опис: Зупинка передачі сигналів та перевірка графіка.

Кроки:

* Натисніть кнопку "Stop" на ПК.
* Очікуваний результат: Передача сигналів повинна припинитися, графік повинен бути видимим.

Результати тестування збіглися з очікуваними.

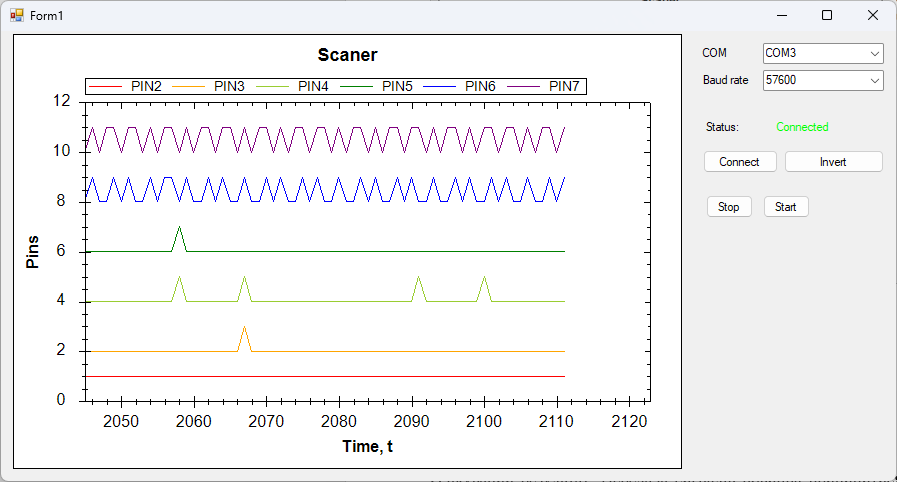


Рисунок 4.3 - Результат 3го тесту

# ВИСНОВКИ

У результаті розробки сканера цифрових сигналів з урахуванням Arduino досягнуто такі результати.

Ефективність використання Arduino: Вибір Arduino як платформа для сканера обумовлений доступністю та широкою підтримкою спільноти. Результати тестування підтвердили надійну роботу пристрою та його здатність ефективно зчитувати та передавати цифрові сигнали.

Доступність та простота використання: Реалізація сканера за допомогою Arduino дозволила створити простий у використанні пристрій. Інтерфейс користувача інтуїтивно зрозумілий, що робить сканер доступним широкого кола користувачів.

Вартісна ефективність: Проект виправдав свою доцільність, забезпечуючи дешевизну та доступність у порівнянні з ринковими аналогами. Використання недорогих компонентів, таких як Arduino знизило витрати на створення пристрою.

Надійність та стабільність: Результати тестування підтвердили стабільну роботу сканера. Відсутність збоїв у процесі передачі та візуалізації сигналів свідчить про надійність розробленого пристрою.

Подальші перспективи: У майбутньому можна розширити функціональність сканера, додавши можливості аналізу та обробки сигналів. Також можливе використання додаткових інтерфейсів зв'язку для ширшого спектра застосувань.

В цілому, розроблений сканер цифрових сигналів на базі Arduino є ефективним, доступним і стабільним пристроєм для аналізу та контролю цифрових сигналів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

|  |  |
| --- | --- |
| [1] | «saleae-logic-pro-16,» Saleae, Inc, 2023. [Онлайновий]. Available: https://usd.saleae.com/products/saleae-logic-pro-16. |
| [2] | «XILINX Kintex7 SoM FPGA Core Board XC7K325T,» Alinx Electronic Limited 沪ICP备13046728号, 2023. [Онлайновий]. Available: https://alinx.com/. |
| [3] | «Arduino Uno Rev3,» Arduino S.r.l. - Partita IVA 09755110963, 2021. [Онлайновий]. Available: https://store.arduino.cc/products/arduino-uno-rev3. |
| [4] | «Raspberry Pi 1 Model B+,» Raspberry Pi, [Онлайновий]. Available: www.raspberrypi.com. |
| [5] | CODESYS GROUP, «CODESYS ENGINEERING AND VISUALISATION,» CODESYS GmbH, 2022. [Онлайновий]. Available: https://www.codesys.com/. |
| [6] | «ChatGPT 3.5,» OpenAI, [Онлайновий]. Available: https://chat.openai.com/. [Дата звернення: 03-05 2023]. |
| [7] | IArduino, «Автоматический полив растений,» [Онлайновий]. Available: https://lesson.iarduino.ru/page/urok-30-avtomaticheskiy-poliv-rasteniy/. |
| [8] | Ф. Б.Р., «Автоматизація обладнання для підтримання мікроклімату в складі для зберігання овочевої продукції,» Сумський державний університет, Суми, 2020. |

# ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ.

## А.1 Вступ

На підставі доручення замовника необхідно розробити програму візуалізації даних з Arduino за допомогою CoDeSys.

## А.2 Підстави для розробки

* Замовлення від замовника на розробку системи візуалізації даних з Arduino.
* Технічне завдання від замовника.

## А.3 Призначення розробки

Призначенням розробки є створення системи, яка здатна збирати дані з датчиків, підключених до Arduino, та відображати їх на графічному інтерфейсі, розробленому в середовищі CoDeSys.

## А.4 Вимоги до програми або програмного виробу

### А.4.1 Вимоги до функціональних характеристик:

* Система повинна забезпечувати збір даних з датчиків, підключених до Arduino через вхідні порти та аналого-цифрові перетворювачі.
* Має бути можливість передачі даних з Arduino до ПК за допомогою з'єднання USB.
* Розроблене програмне забезпечення на Arduino повинно забезпечувати збір даних з датчиків і передачу їх на ПК.
* Розроблене програмне забезпечення на CoDeSys повинно забезпечувати візуалізацію отриманих даних з датчиків.
* Система повинна мати графічний інтерфейс користувача для відображення даних на ПК.

### А.4.2 Вимоги до складу і параметрів технічних засобів:

* Система повинна працювати з Arduino та його вхідними портами та аналого-цифровими перетворювачами.
* З'єднання між Arduino та ПК має бути реалізоване за допомогою USB-з'єднання.

### А.4.3 Вимоги до інформаційної та програмної сумісності:

* Програмне забезпечення на Arduino повинно бути сумісне з Arduino-платформою та використовувати відповідні бібліотеки для збору даних з датчиків.
* Програмне забезпечення на CoDeSys повинно бути сумісне з використовуваною версією CoDeSys та підтримувати відповідний протокол передачі даних.

### А.4.4 Спеціальні вимоги:

* Розробка програмного забезпечення повинна відповідати принципам структурованого програмування та кращим практикам програмування.
* Графічний інтерфейс користувача повинен бути зручним та інтуїтивно зрозумілим для користувача.
* Система повинна бути стабільною, надійною та ефективною у роботі з датчиками та відображенням даних

## А.5 Вимоги до програмної документації

Розробник повинен підготувати наступну документацію:

* Технічне завдання.
* Керівництво користувача з інструкціями щодо встановлення, налаштування та використання програмного забезпечення.
* Опис програмного коду з коментарями.
* Звіт про тестування, включаючи результати та висновки.

Результатом проекту має стати робоча система, яка здатна збирати та відображати дані з датчиків, підключених до Arduino, на графічному інтерфейсі в середовищі CoDeSys.

# ДОДАТОК Б. ВИХІДНІ КОДИ.

### Код на Arduino

Скетч  
#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C \_lcd1(0x3f, 16, 2);

int \_dispTempLength1=0;

boolean \_isNeedClearDisp1;

struct UB\_224761930

{

int ubo\_93944566 = 0;

int ubo\_233985835 = 0;

int ubo\_185383222 = 0;

int \_swi1 = 0;

int \_mux1 = 0;

int \_mux2 = 0;

int \_mux3 = 0;

}

;

UB\_224761930 UB\_224761930\_Instance1;

int UB\_224761930\_ubi\_145854433 = 0;

int UB\_224761930\_ubi\_50719639 = 0;

int UB\_224761930\_ubi\_152830701 = 0;

bool UB\_224761930\_ubi\_208329377 = 0;

struct UB\_216958797

{

String ubo\_181543441 = "";

}

;

UB\_216958797 UB\_216958797\_Instance1;

int UB\_216958797\_ubi\_248398040 = 0;

int UB\_216958797\_ubi\_201969404 = 0;

int UB\_216958797\_ubi\_244852152 = 0;

bool \_modbusSlaveDataTable\_0[4];

int \_modbusSlaveAddresTable\_0[4] = {0, 1, 2, 3};

int \_modbusSlaveDataTable\_4[3];

int \_modbusSlaveAddresTable\_4[3] = {0, 1, 2};

byte \_modbusSlaveBufferSize = 0;

byte \_modbusSlaveLastRec = 0;

long \_modbusSlaveTime;

byte \_modbusSlaveBuffer[64];

const unsigned char \_modbusSlave\_fctsupported[] = {1, 5, 15, 3, 6, 16};

int \_PWDC = 0;

bool \_trgrt2 = 0;

bool \_trgrt2I = 0;

bool \_trgt1 = 0;

bool \_trgt1I = 0;

bool \_trgrt1 = 0;

bool \_trgrt1I = 0;

int \_disp1oldLength = 0;

bool \_bounseInputD7S = 0;

bool \_bounseInputD7O = 0;

unsigned long \_bounseInputD7P = 0UL;

bool \_tempVariable\_bool;

int \_tempVariable\_int;

void setup()

{

pinMode(7, INPUT\_PULLUP);

pinMode(13, OUTPUT);

digitalWrite(13, 0);

Wire.begin();

delay(10);

Serial.begin(9600, SERIAL\_8N1);

TCCR2A = 0x00;

TCCR2B = 0x07;

TIMSK2=0x01;

TCNT2=100;

\_lcd1.init();

\_lcd1.backlight();

\_bounseInputD7O = digitalRead(7);

}

void loop()

{

if (\_isNeedClearDisp1)

{

\_lcd1.clear();

\_isNeedClearDisp1= 0;

}

\_modbusSlavePoll();

\_PWDC = 0;

bool \_bounceInputTmpD7 = (digitalRead (7));

if (\_bounseInputD7S)

{

if (millis() >= (\_bounseInputD7P + 40))

{

\_bounseInputD7O= \_bounceInputTmpD7;

\_bounseInputD7S=0;

}

}

else

{

if (\_bounceInputTmpD7 != \_bounseInputD7O)

{

\_bounseInputD7S=1;

\_bounseInputD7P = millis();

}

}

//Плата:1

if ((\_modbusSlaveDataTable\_0[1]))

{

if (\_trgrt2I)

{

\_trgrt2 = 0;

}

else

{

\_trgrt2 = 1;

\_trgrt2I = 1;

}

}

else

{

\_trgrt2 = 0;

\_trgrt2I = 0;

}

;

if (\_bounseInputD7O)

{

if (\_trgrt1I)

{

\_trgrt1 = 0;

}

else

{

\_trgrt1 = 1;

\_trgrt1I = 1;

}

}

else

{

\_trgrt1 = 0;

\_trgrt1I = 0;

}

;

bool \_tmp1 = ((\_trgrt2) || (\_trgrt1));

if (\_tmp1)

{

if (! \_trgt1I) \_trgt1 = ! \_trgt1;

}

\_trgt1I = \_tmp1;

\_tempVariable\_bool = \_trgt1;

\_modbusSlaveDataTable\_0[0] = \_tempVariable\_bool;

UB\_224761930\_ubi\_145854433 = (map(((analogRead (0))), (0), (1023), (0), (255)));

UB\_224761930\_ubi\_50719639 = (map(((analogRead (1))), (0), (1023), (0), (255)));

UB\_224761930\_ubi\_152830701 = (map(((analogRead (2))), (0), (1023), (0), (255)));

UB\_224761930\_ubi\_208329377 = \_trgt1;

\_func\_UB\_224761930(&UB\_224761930\_Instance1, UB\_224761930\_ubi\_145854433, UB\_224761930\_ubi\_50719639, UB\_224761930\_ubi\_152830701, UB\_224761930\_ubi\_208329377);

\_tempVariable\_int = UB\_224761930\_Instance1.ubo\_185383222;

\_modbusSlaveDataTable\_4[2] = \_tempVariable\_int;

\_tempVariable\_int = UB\_224761930\_Instance1.ubo\_233985835;

\_modbusSlaveDataTable\_4[1] = \_tempVariable\_int;

UB\_216958797\_ubi\_248398040 = UB\_224761930\_Instance1.ubo\_93944566;

UB\_216958797\_ubi\_201969404 = UB\_224761930\_Instance1.ubo\_233985835;

UB\_216958797\_ubi\_244852152 = UB\_224761930\_Instance1.ubo\_185383222;

\_func\_UB\_216958797(&UB\_216958797\_Instance1, UB\_216958797\_ubi\_248398040, UB\_216958797\_ubi\_201969404, UB\_216958797\_ubi\_244852152);

if ((\_modbusSlaveDataTable\_0[3]))

{

\_dispTempLength1 = ((UB\_216958797\_Instance1.ubo\_181543441)).length();

if (\_disp1oldLength > \_dispTempLength1)

{

\_isNeedClearDisp1 = 1;

}

\_disp1oldLength = \_dispTempLength1;

\_lcd1.setCursor(int((16 - \_dispTempLength1)/2), 0);

\_lcd1.print((UB\_216958797\_Instance1.ubo\_181543441));

}

else

{

if (\_disp1oldLength > 0)

{

\_isNeedClearDisp1 = 1;

\_disp1oldLength = 0;

}

}

\_tempVariable\_int = UB\_224761930\_Instance1.ubo\_93944566;

\_modbusSlaveDataTable\_4[0] = \_tempVariable\_int;

digitalWrite(13, (\_modbusSlaveDataTable\_0[2]));

}

bool \_isTimer(unsigned long startTime, unsigned long period)

{

unsigned long currentTime;

currentTime = millis();

if (currentTime>= startTime)

{

return (currentTime>=(startTime + period));

}

else

{

return (currentTime >=(4294967295-startTime+period));

}

}

ISR(TIMER2\_OVF\_vect)

{

TCNT2=100;

if(\_PWDC >= 1000)

{

asm volatile ("jmp 0x0000");

}

else

{

\_PWDC = \_PWDC+1;

}

;

}

int modbusCalcCRC(byte length, byte bufferArray[])

{

unsigned int temp, temp2, flag;

temp = 0xFFFF;

for (unsigned char i = 0; i < length; i++)

{

temp = temp ^ bufferArray[i];

for (unsigned char j = 1; j <= 8; j++)

{

flag = temp & 0x0001;

temp >>= 1;

if (flag) temp ^= 0xA001;

}

}

temp2 = temp >> 8;

temp = (temp << 8) | temp2;

temp &= 0xFFFF;

return temp;

}

byte \_modbusSlavePoll()

{

byte avalibleBytes = Serial.available();

if (avalibleBytes == 0) return 0;

if (avalibleBytes != \_modbusSlaveLastRec)

{

\_modbusSlaveLastRec = avalibleBytes;

\_modbusSlaveTime = millis();

return 0;

}

if (!(\_isTimer(\_modbusSlaveTime, 5))) return 0;

\_modbusSlaveLastRec = 0;

byte state = \_modbusGetSlaveRxBuffer();

if (state < 7)

{

return state;

}

if ((\_modbusSlaveBuffer[0] != 1) && (\_modbusSlaveBuffer[0] != 0)) return 0;

byte exception = \_modbusValidateRequest();

if (exception > 0)

{

if (exception != 255)

{

\_modbusSlaveBuildException(exception);

\_modbusSlaveSendTxBuffer();

}

return exception;

}

switch (\_modbusSlaveBuffer[1])

{

case 1 :

return process\_modbus\_FC1(0);

break;

case 5:

return process\_modbus\_FC5();

break;

case 15:

return process\_modbus\_FC15();

break;

case 3 :

return process\_modbus\_FC3(4);

break;

case 6 :

return process\_modbus\_FC6();

break;

case 16 :

return process\_modbus\_FC16();

break;

default:

break;

}

return 25;

}

byte \_modbusValidateRequest()

{

int msgCRC =

((\_modbusSlaveBuffer[\_modbusSlaveBufferSize - 2] << 8) | \_modbusSlaveBuffer[\_modbusSlaveBufferSize - 1]);

if (modbusCalcCRC(\_modbusSlaveBufferSize - 2, \_modbusSlaveBuffer) != msgCRC)

{

return 255;

}

boolean isSupported = false;

for (uint8\_t i = 0; i < sizeof(\_modbusSlave\_fctsupported); i++)

{

if (\_modbusSlave\_fctsupported[i] == \_modbusSlaveBuffer[1])

{

isSupported = 1;

break;

}

}

if (!isSupported)

{

return 1;

}

int intRegs = 0;

byte byteRegs;

switch (\_modbusSlaveBuffer[1])

{

case 1:

if(!(checkModbusRange(word(\_modbusSlaveBuffer[2], \_modbusSlaveBuffer[3]) , word(\_modbusSlaveBuffer[4], \_modbusSlaveBuffer[5]) ,0)))

{

return 2;

}

break;

case 5:

if(!(checkModbusAddres(word(\_modbusSlaveBuffer[2], \_modbusSlaveBuffer[3]),0)))

{

return 2;

}

break;

case 15 :

if(!(checkModbusRange(word(\_modbusSlaveBuffer[2], \_modbusSlaveBuffer[3]) , word(\_modbusSlaveBuffer[4], \_modbusSlaveBuffer[5]) ,0)))

{

return 2;

}

break;

case 6 :

if(!(checkModbusAddres((word(\_modbusSlaveBuffer[2], \_modbusSlaveBuffer[3])),4)))

{

return 2;

}

break;

case 3 :

case 16 :

if(!(checkModbusRange((word(\_modbusSlaveBuffer[2], \_modbusSlaveBuffer[3])), (word(\_modbusSlaveBuffer[4], \_modbusSlaveBuffer[5])),4)))

{

return 2;

}

break;

}

return 0; // OK, no exception code thrown

}

bool checkModbusAddres(int addr, byte table)

{

return (!((modbusSlaveIndexForAddres(addr,table)) == -1));

}

int modbusSlaveIndexForAddres(int addr, byte table)

{

int tableSize = 0;

switch (table)

{

case 0:

tableSize = 4;

break;

case 4:

tableSize = 3;

break;

}

for (byte i = 0; i < tableSize; i++)

{

if((modbusSlaveAddresFromIndex(i,table)) == addr)

{

return i;

}

}

return -1;

}

int modbusSlaveAddresFromIndex(byte index, byte table)

{

switch (table)

{

case 0:

return \_modbusSlaveAddresTable\_0[index];

break;

case 4:

return \_modbusSlaveAddresTable\_4[index];

break;

}

return -1;

}

bool checkModbusRange(int startAddr, int addrNumber, byte table)

{

for (int i=0; i < addrNumber; i++)

{

if(!(checkModbusAddres((startAddr+i),table)))

{

return false;

}

}

return true;

}

void \_modbusSlaveBuildException(byte exception)

{

byte func = \_modbusSlaveBuffer[1];

\_modbusSlaveBuffer[0] = 1;

\_modbusSlaveBuffer[1] = func + 0x80;

\_modbusSlaveBuffer[ 2 ] = exception;

\_modbusSlaveBufferSize = 3;

}

void \_modbusSlaveSendTxBuffer()

{

if(\_modbusSlaveBuffer[0] == 0)

{

return;

}

byte i = 0;

int crc = modbusCalcCRC(\_modbusSlaveBufferSize,\_modbusSlaveBuffer);

\_modbusSlaveBuffer[ \_modbusSlaveBufferSize ] = crc >> 8;

\_modbusSlaveBufferSize++;

\_modbusSlaveBuffer[ \_modbusSlaveBufferSize ] = crc & 0x00ff;

\_modbusSlaveBufferSize++;

Serial.write(\_modbusSlaveBuffer, \_modbusSlaveBufferSize);

delay (5);

Serial.flush();

\_modbusSlaveBufferSize = 0;

}

byte \_modbusGetSlaveRxBuffer()

{

boolean bBuffOverflow = false;

\_modbusSlaveBufferSize = 0;

while (Serial.available())

{

\_modbusSlaveBuffer[ \_modbusSlaveBufferSize ] = Serial.read();

\_modbusSlaveBufferSize ++;

if (\_modbusSlaveBufferSize >= 64) bBuffOverflow = true;

}

if (bBuffOverflow)

{

return -3;

}

return \_modbusSlaveBufferSize;

}

byte process\_modbus\_FC1(byte table)

{

byte bytesNo, bitsNo;

int currentCoil, coil;

bool value;

byte index;

int startCoil = word(\_modbusSlaveBuffer[2], \_modbusSlaveBuffer[ 3]);

int coilNo = word(\_modbusSlaveBuffer[4], \_modbusSlaveBuffer[5]);

bytesNo = byte(coilNo / 8);

if (coilNo % 8 != 0) bytesNo ++;

\_modbusSlaveBuffer[2] = bytesNo;

\_modbusSlaveBufferSize = 3;

bitsNo = 0;

for (currentCoil = 0; currentCoil < coilNo; currentCoil++)

{

coil = startCoil + currentCoil;

index = modbusSlaveIndexForAddres(coil, table);

if (table == 0)

{

value = \_modbusSlaveDataTable\_0[index];

}

bitWrite(\_modbusSlaveBuffer[ \_modbusSlaveBufferSize ], bitsNo, value);

bitsNo ++;

if (bitsNo > 7)

{

bitsNo = 0;

\_modbusSlaveBufferSize++;

}

}

if (coilNo % 8 != 0) \_modbusSlaveBufferSize ++;

\_modbusSlaveSendTxBuffer();

return \_modbusSlaveBufferSize + 2;

}

byte process\_modbus\_FC3(byte table)

{

int startAddr = word(\_modbusSlaveBuffer[2], \_modbusSlaveBuffer[3]);

int byteRegsno = word(\_modbusSlaveBuffer[4], \_modbusSlaveBuffer[5]);

int i;

int value;

byte index;

\_modbusSlaveBuffer[ 2 ] = byteRegsno \* 2;

\_modbusSlaveBufferSize = 3;

for (i = startAddr; i < startAddr + byteRegsno; i++)

{

index = modbusSlaveIndexForAddres(i, table);

if (table == 4)

{

value = \_modbusSlaveDataTable\_4[index];

}

\_modbusSlaveBuffer[ \_modbusSlaveBufferSize ] = highByte(value);

\_modbusSlaveBufferSize++;

\_modbusSlaveBuffer[ \_modbusSlaveBufferSize ] = lowByte(value);

\_modbusSlaveBufferSize++;

}

\_modbusSlaveSendTxBuffer();

return \_modbusSlaveBufferSize + 2;

}

byte process\_modbus\_FC5()

{

byte index;

int coil = word(\_modbusSlaveBuffer[2], \_modbusSlaveBuffer[3]);

index = modbusSlaveIndexForAddres(coil, 0);

\_modbusSlaveDataTable\_0[index] = (\_modbusSlaveBuffer[4] == 0xff);

\_modbusSlaveBufferSize = 6;

\_modbusSlaveSendTxBuffer();

return \_modbusSlaveBufferSize + 2;

}

byte process\_modbus\_FC15()

{

byte frameByte, bitsNo;

byte currentCoil, coil;

byte index;

int startCoil = word(\_modbusSlaveBuffer[2], \_modbusSlaveBuffer[3]);

int coilNo = word(\_modbusSlaveBuffer[4], \_modbusSlaveBuffer[5]);

bitsNo = 0;

frameByte = 7;

for (currentCoil = 0; currentCoil < coilNo; currentCoil++)

{

coil = startCoil + currentCoil;

index = modbusSlaveIndexForAddres(coil, 0);

\_modbusSlaveDataTable\_0[index] = bitRead(\_modbusSlaveBuffer[ frameByte ], bitsNo);

bitsNo ++;

if (bitsNo > 7)

{

bitsNo = 0;

frameByte++;

}

}

\_modbusSlaveBufferSize = 6;

\_modbusSlaveSendTxBuffer();

return \_modbusSlaveBufferSize + 2;

}

byte process\_modbus\_FC6()

{

int addres = word(\_modbusSlaveBuffer[2], \_modbusSlaveBuffer[3]);

int index;

index = modbusSlaveIndexForAddres(addres, 4);

\_modbusSlaveDataTable\_4[index] =word(\_modbusSlaveBuffer[4], \_modbusSlaveBuffer[5]);

\_modbusSlaveBufferSize = 6;

\_modbusSlaveSendTxBuffer();

return \_modbusSlaveBufferSize + 2;

}

byte process\_modbus\_FC16()

{

byte func = \_modbusSlaveBuffer[1];

int startAddr = \_modbusSlaveBuffer[2] << 8 | \_modbusSlaveBuffer[3];

int byteRegsno = \_modbusSlaveBuffer[4] << 8 | \_modbusSlaveBuffer[5];

int i;

int index;

\_modbusSlaveBuffer[4] = 0;

\_modbusSlaveBuffer[5] = byteRegsno;

\_modbusSlaveBufferSize = 6;

for (i = 0; i < byteRegsno; i++)

{

index = modbusSlaveIndexForAddres((startAddr+i), 4);

\_modbusSlaveDataTable\_4[index] =word(\_modbusSlaveBuffer[ 7 + i \* 2 ], \_modbusSlaveBuffer[8 + i \* 2 ]);

}

\_modbusSlaveSendTxBuffer();

return \_modbusSlaveBufferSize +2;

}

void \_func\_UB\_224761930(struct UB\_224761930 \*\_ubInstans, int ubi\_145854433, int ubi\_50719639, int ubi\_152830701, bool ubi\_208329377)

{

int ubo\_93944566 = \_ubInstans->ubo\_93944566;

int ubo\_233985835 = \_ubInstans->ubo\_233985835;

int ubo\_185383222 = \_ubInstans->ubo\_185383222;

int \_swi1 = \_ubInstans->\_swi1;

int \_mux1 = \_ubInstans->\_mux1;

int \_mux2 = \_ubInstans->\_mux2;

int \_mux3 = \_ubInstans->\_mux3;

if(ubi\_208329377)

{

\_swi1=1;

}

else

{

\_swi1=0;

}

if((\_swi1) == 0)

{

\_mux1 = ubi\_152830701;

}

if((\_swi1) == 1)

{

\_mux1 = (255)-(ubi\_152830701);

}

ubo\_185383222 = \_mux1;

if((\_swi1) == 0)

{

\_mux2 = ubi\_50719639;

}

if((\_swi1) == 1)

{

\_mux2 = (255)-(ubi\_50719639);

}

ubo\_233985835 = \_mux2;

if((\_swi1) == 0)

{

\_mux3 = ubi\_145854433;

}

if((\_swi1) == 1)

{

\_mux3 = (255)-(ubi\_145854433);

}

ubo\_93944566 = \_mux3;

\_ubInstans->ubo\_93944566 = ubo\_93944566;

\_ubInstans->ubo\_233985835 = ubo\_233985835;

\_ubInstans->ubo\_185383222 = ubo\_185383222;

\_ubInstans->\_swi1 = \_swi1;

\_ubInstans->\_mux1 = \_mux1;

\_ubInstans->\_mux2 = \_mux2;

\_ubInstans->\_mux3 = \_mux3;

}

void \_func\_UB\_216958797(struct UB\_216958797 \*\_ubInstans, int ubi\_248398040, int ubi\_201969404, int ubi\_244852152)

{

String ubo\_181543441 = \_ubInstans->ubo\_181543441;

ubo\_181543441 = ((String("R:")) + ((String(ubi\_248398040, DEC))) + (String(" G:")) + ((String(ubi\_201969404, DEC))) + (String(" B:")) + ((String(ubi\_244852152, DEC))));

\_ubInstans->ubo\_181543441 = ubo\_181543441;

}

### Документація, з CoDeSys:

# ДОДАТОК В. НАСТАНОВА КОРИСТУВАЧУ.

# Відомості про призначення

Ця програма візуалізації даних з Arduino створена з метою збору, передачі та відображення даних з підключених до Arduino датчиків. Головна функція програми полягає в забезпеченні зручного та інтуїтивно зрозумілого інтерфейсу, що дозволяє користувачу відслідковувати реальний стан датчиків та отримувати поточні дані.

Ця програма дозволяє збирати дані з датчиків, підключених до Arduino, шляхом використання вхідних портів та аналого-цифрових перетворювачів. Зібрані дані можуть бути передані на ПК за допомогою з'єднання USB. Користувач має змогу відображати та моніторити отримані дані на графічному інтерфейсі програми, який реалізований в середовищі CoDeSys.

Основні функції програми включають збір та передачу даних з Arduino, візуалізацію цих даних на графічному інтерфейсі, а також забезпечення зручних інструментів для користувача, що дозволяють контролювати та маніпулювати отриманими даними.

Ця програма призначена для широкого кола користувачів, включаючи інженерів, розробників, дослідників та будь-яких інших осіб, які потребують збору, моніторингу та візуалізації даних з датчиків, підключених до Arduino. Простий інтерфейс та інтуїтивно зрозумілі функції роблять програму доступною для використання навіть тим, хто не має глибоких знань у програмуванні або автоматизації.

# Умови, необхідні для виконання

## Апаратні засоби:

* Комп'ютер з операційною системою, сумісною з вимогами CoDeSys.
* Arduino-плата або Arduino-сумісний мікроконтролер з відповідними вхідними портами та можливістю підключення датчиків.
* Датчики, які необхідно підключити до Arduino для збору даних.
* USB-кабель для з'єднання Arduino з комп'ютером.
* Додаткові Arduino-пристрої (якщо планується розширення системи через I2C з'єднання).

## Програмні засоби:

* Остання версія CoDeSys, яку необхідно завантажити та встановити на комп'ютер.
* Arduino IDE (Integrated Development Environment) для розробки програмного забезпечення для Arduino.

## Документація та ресурси:

* Документація, що пояснює використання CoDeSys та Arduino, включаючи посібники, інструкції та приклади.
* Ресурси Інтернету для отримання додаткової інформації, підтримки та відповідей на можливі питання.

## Знання та навички:

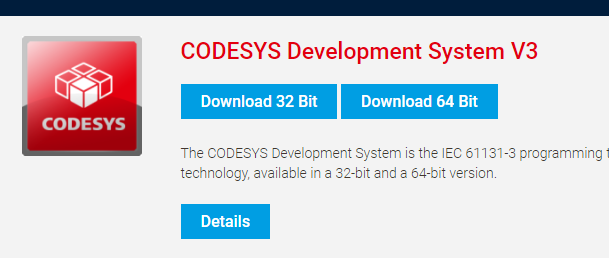
* Базові знання програмування мікроконтролерів та мови програмування, яку підтримує Arduino.
* Розуміння принципів роботи CoDeSys та можливостей візуалізації даних.
* Розуміння основних протоколів зв'язку, зокрема Modbus.
* Знання електроніки та здатність підключати та налаштовувати датчики до Arduino.

# Послідовність дій користувача

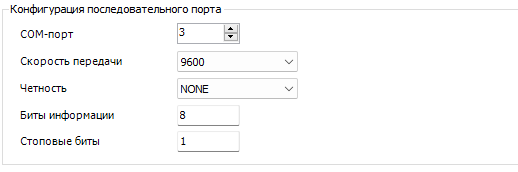
## Інсталяція CoDeSys

Інструкції щодо встановлення та налаштування CoDeSys:

1. Завантажте останню версію CoDeSys з офіційного веб-сайту виробника [5].



1. Встановіть CoDeSys на ваш комп'ютер.
2. Налаштуйте параметри підключення до контролера, такі як IP-адреса, номер порту та тип підключення.



1. Перевірте підключення до контролера та його доступність в CoDeSys.

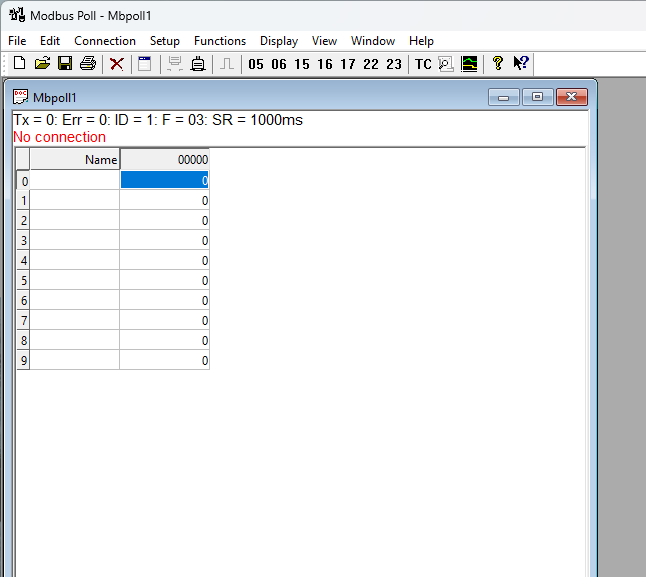


Після цих кроків ви зможете приступити до програмування та налагодження контролера з використанням CoDeSys.

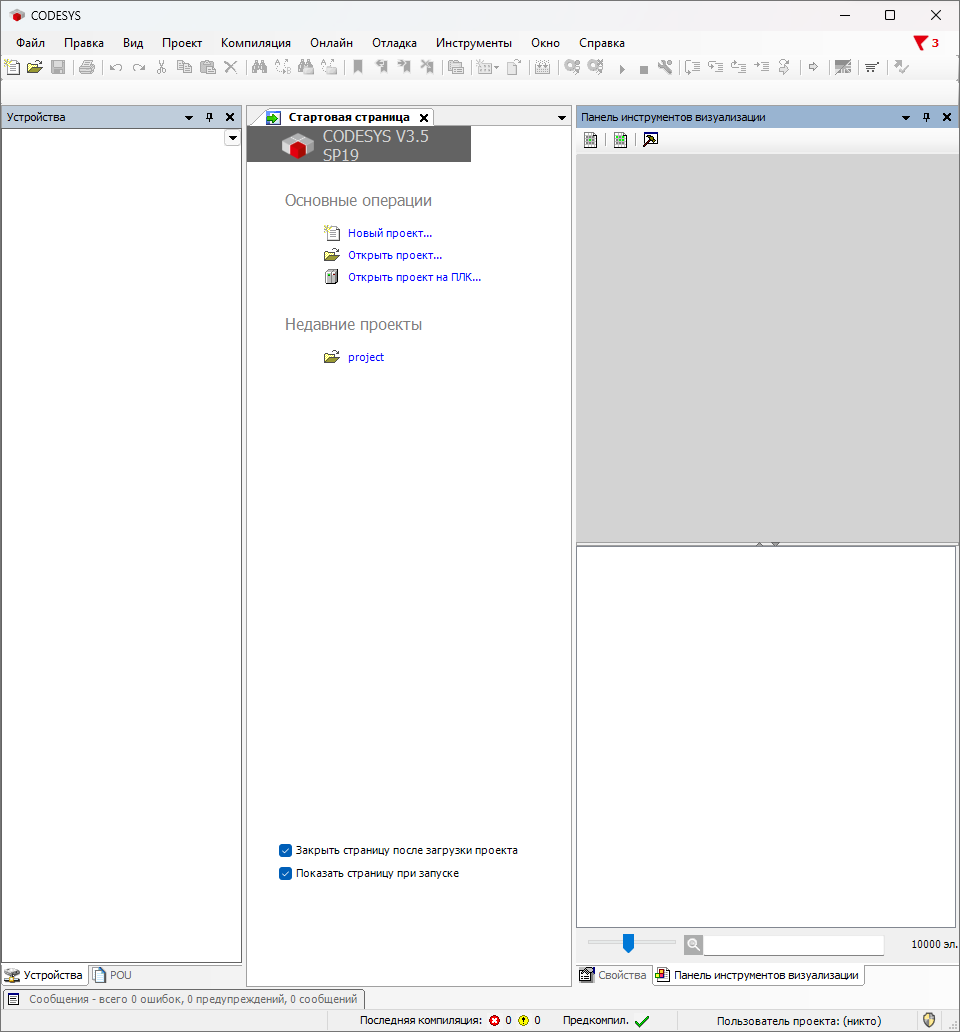
## Конфігурація з'єднання

Інструкції щодо використання модуля комунікації Modbus для підключення до контролера:

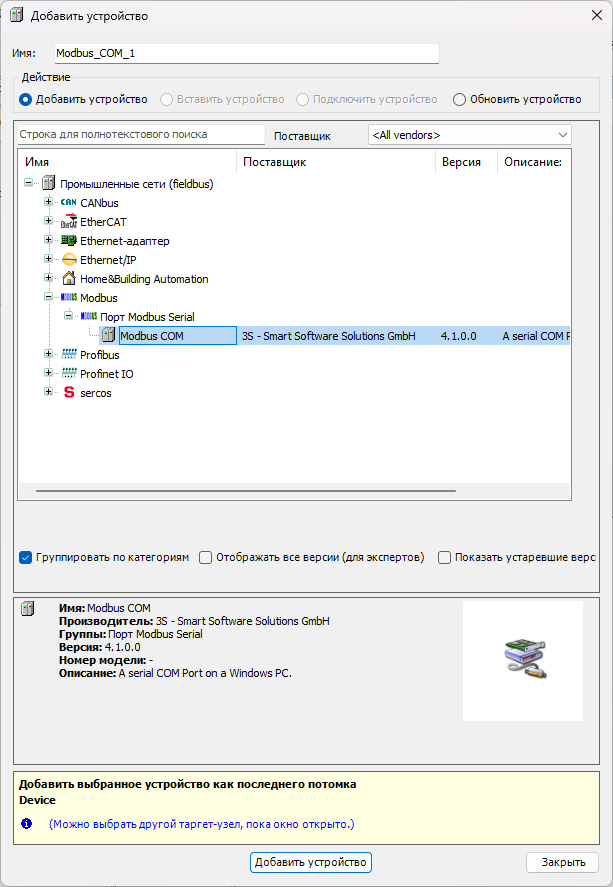
1. Переконайтеся, що ваш контролер підтримує протокол Modbus і має необхідні комунікаційні порти.
2. Підключіть ваш контролер до комп'ютера за допомогою відповідного з'єднання, наприклад, RS-485 або Ethernet.
3. Встановіть необхідне програмне забезпечення для забезпечення комунікації між вашим комп'ютером та контролером. Найчастіше використовується CoDeSys, але також можуть використовуватися інші програми, наприклад, ModScan або QModMaster.



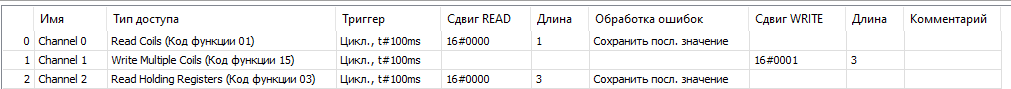
1. Відкрийте програму CoDeSys та створіть новий проект.



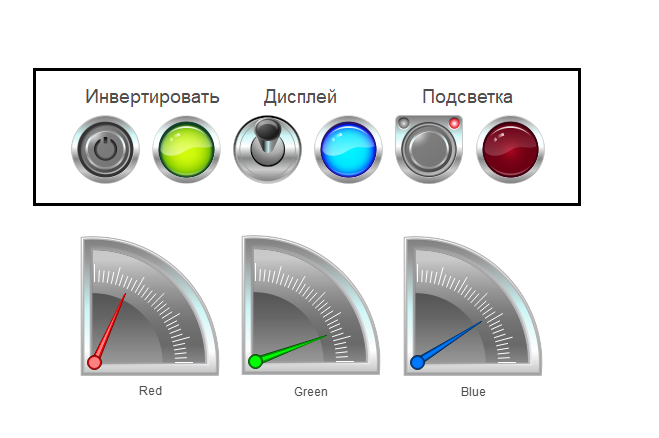
1. Додайте до вашого проекту модуль комунікації Modbus і налаштуйте параметри підключення до вашого контролера, такі як швидкість передачі даних та адреса.



1. Додайте необхідні керуючі та інформаційні сигнали до вашого проекту та виконайте їх програмування.



1. Запустіть ваш проект на виконання та перевірте, чи успішно відбувається обмін даними між вашим контролером та комп'ютером.
2. Використовуйте візуалізацію та елементи керування, щоб зручно контролювати та відслідковувати роботу вашого контролера.



1. Переконайтеся, що ви виконуєте всі необхідні кроки з налагодження з'єднання та програмування вашого контролера. Якщо виникають проблеми, зверніться до документації вашого контролера або до фахівців з технічної підтримки.

## Інструкції щодо використання візуалізації

У вікні візуалізації можливо побачити поточний стан контролера та отримувати інформацію з датчиків. Для керування необхідно використовувати відповідні елементи керування на інтерфейсі, наприклад кнопки, перемикачі, регулятори тощо. Перед використанням елементів керування, слід переконатися, що відповідні канали протоколу modbus правильно налаштовані та працюють коректно. Зверніть увагу на можливість введення неправильних параметрів, що може призвести до неочікуваної поведінки системи. Також, перед використанням системи варто переконатися у правильності з'єднання всіх компонентів та налагодити відповідну інфраструктуру збереження даних.

1. Відкрийте проект в CoDeSys, в якому була створена візуалізація.
2. Запустіть проект на виконання, натиснувши на кнопку "Run" у верхньому меню.
3. Зачекайте, поки проект завантажиться на пристрій.
4. Перейдіть до вікна візуалізації, натиснувши на кнопку "Visualization" у верхньому меню.
5. Використовуйте різні елементи керування, щоб керувати пристроєм. Наприклад, якщо є кнопка, використовуйте її для включення або виключення певного обладнання.
6. Перегляньте різні параметри, які можуть бути відображені в візуалізації, наприклад температуру, швидкість, оберти тощо.
7. Щоб закрити вікно візуалізації, натисніть на кнопку "Close" у верхньому меню.

Зверніть увагу, що доступні елементи керування та параметри візуалізації можуть відрізнятися в залежності від конкретного проекту та пристрою.

# Тексти повідомлень, опис їх змісту та відповідні дії користувача:

## Повідомлення: "Шина не підключена ".

Зміст: Це повідомлення вказує на те, що виникла помилка передачі даних або неправильне підключення шини зв'язку.

Дії користувача: Перевірте правильність підключення шини та впевніться, що передача даних працює належним чином. Перевірте наявність замикань або перевірте підключення до модуля зв'язку.

## Повідомлення: "R:0 G:0 B:0".

Зміст: Це повідомлення вказує на те, що відсутні дані з датчиків R (червоний), G (зелений) та B (синій).

Дії користувача: Перевірте підключення датчиків та їх правильну роботу. Впевніться, що датчики налаштовані та працюють належним чином. При необхідності, виконайте повторний запуск програми.

## Повідомлення: "Немає зв'язку з пристроєм".

Зміст: Це повідомлення вказує на те, що відсутній зв'язок з пристроєм через відсутність включеного модуля Gateway CoDeSys.

Дії користувача: Перевірте, чи модуль Gateway CoDeSys включений та працює належним чином. Переконайтеся, що з'єднання між пристроями встановлено правильно. При необхідності, виконайте повторний запуск програми та перевірте налаштування з'єднання.

Ці повідомлення допомагають користувачу зрозуміти стан системи та вжити відповідні дії для усунення проблеми, якщо вона виникла. При збоях, користувач може спробувати повторно запустити програму після вирішення проблеми або звернутися до технічної підтримки для отримання додаткової допомоги.

# ДОДАТОК Г. ПРЕЗЕНТАЦІЯ.

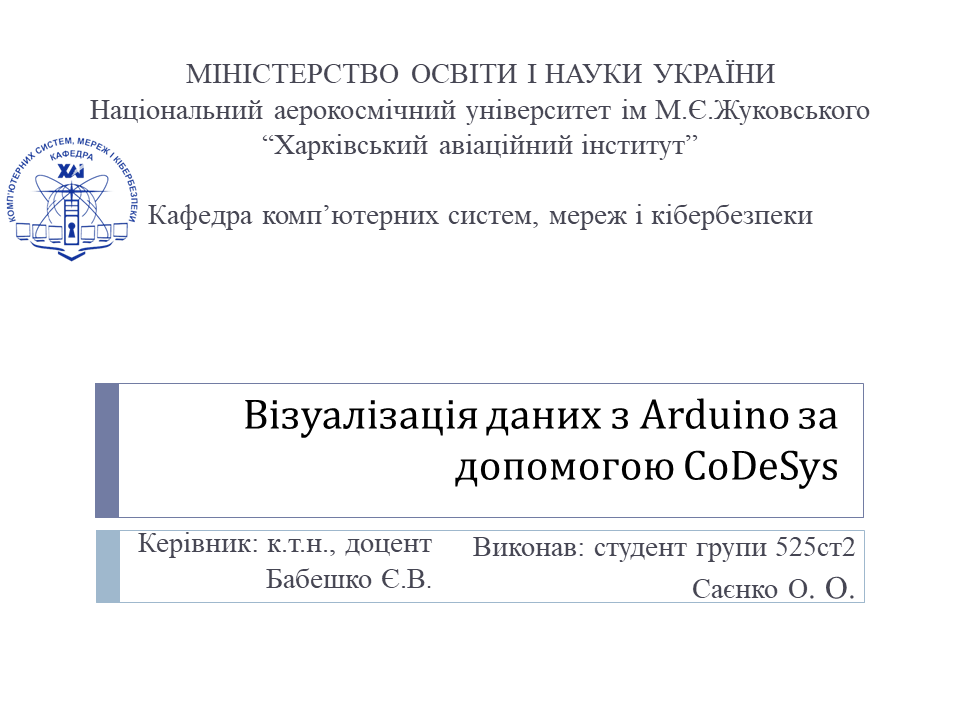


Рисунок Г.0.1 - Слайд 1

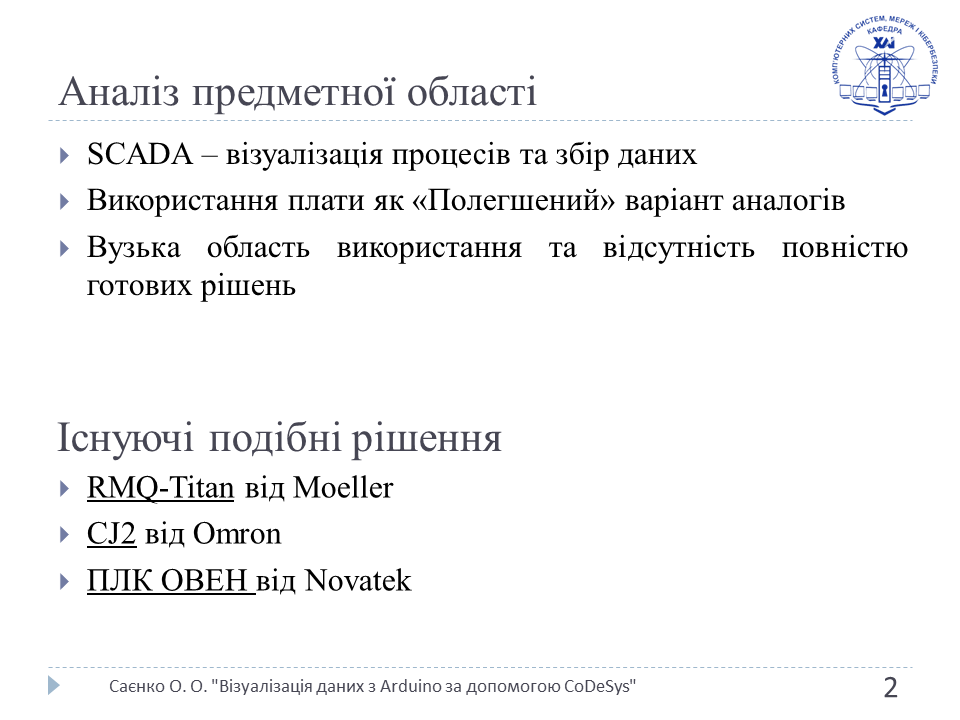


Рисунок Г.0.2 - Слайд 2

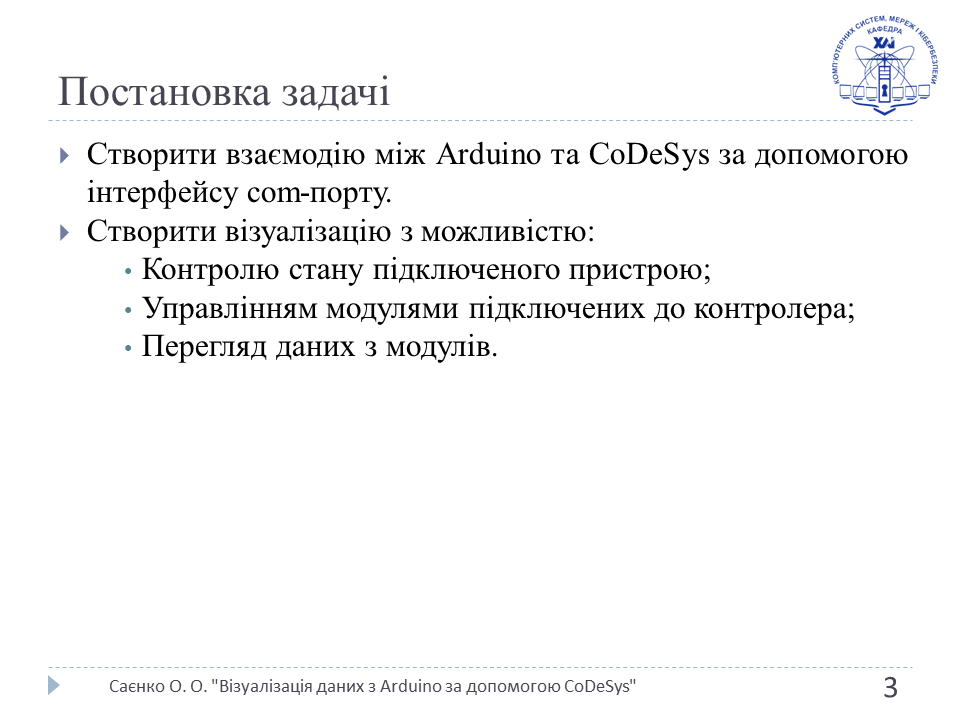


Рисунок Г.0.3 - Слайд 3

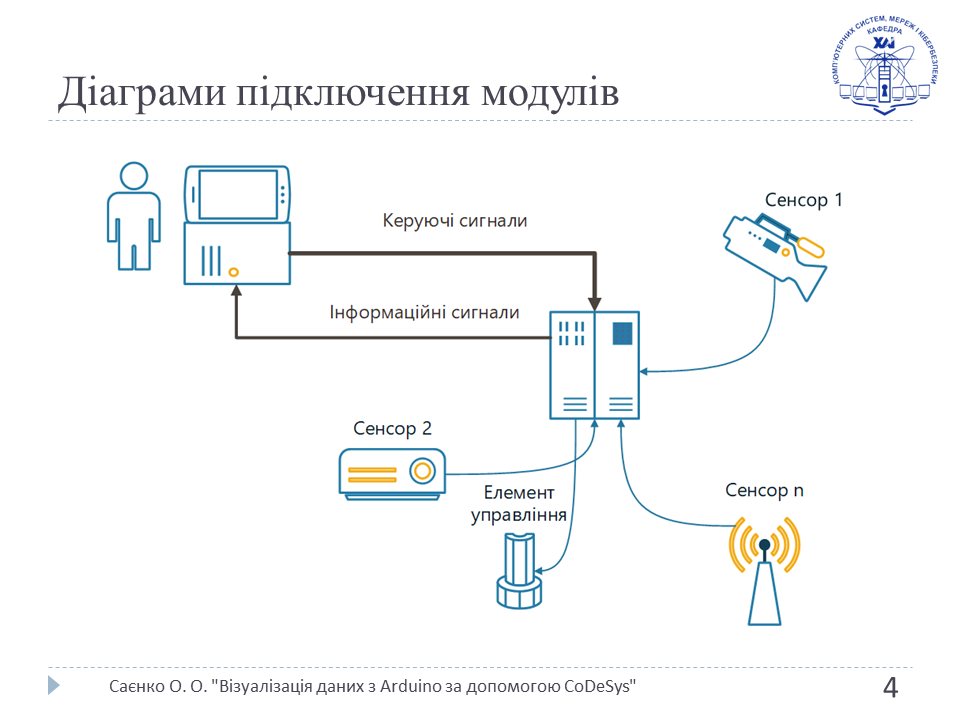


Рисунок Г.0.4 - Слайд 4

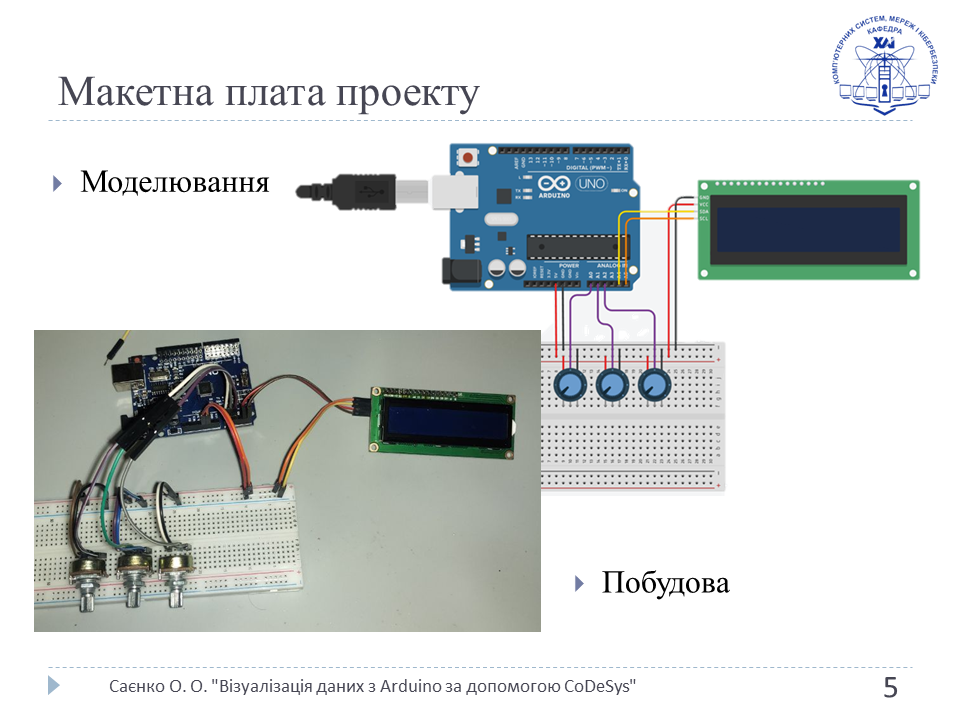


Рисунок Г.0.5 - Слайд 5

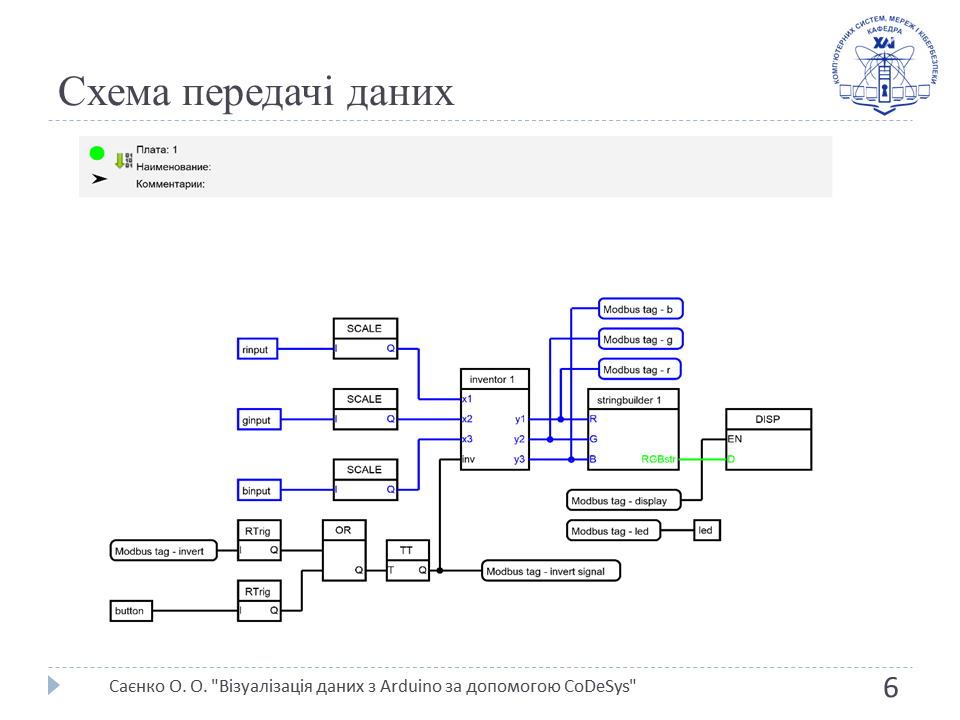


Рисунок Г.0.6 - Слайд 6



Рисунок Г.0.7 - Слайд 7



Рисунок Г.0.8 - Слайд 8



Рисунок Г.0.9 - Слайд 9

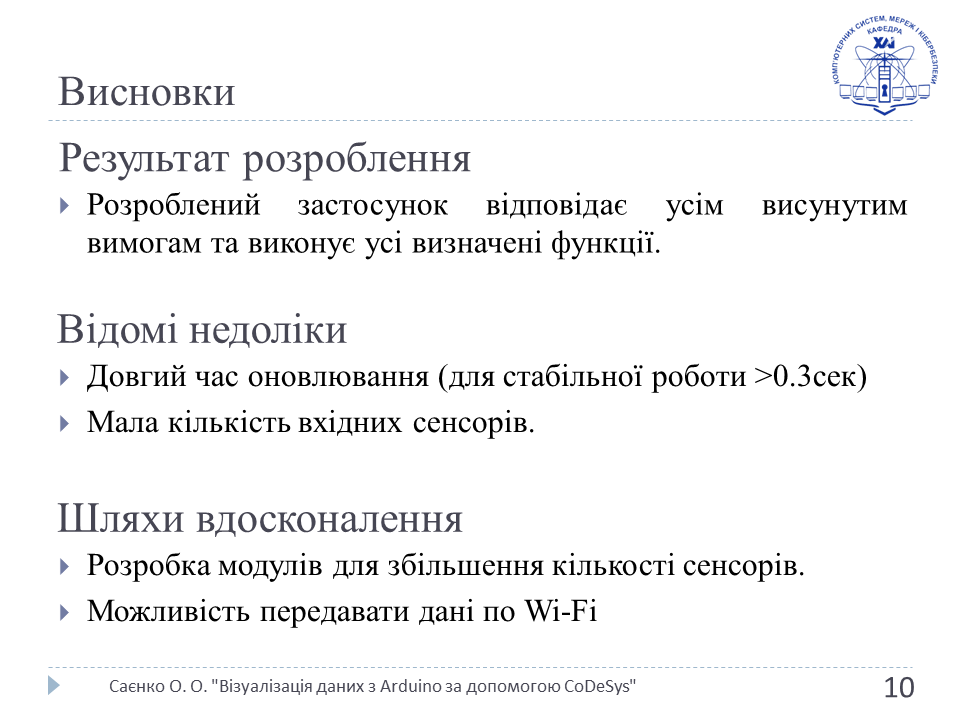


Рисунок Г.0.10 - Слайд 10

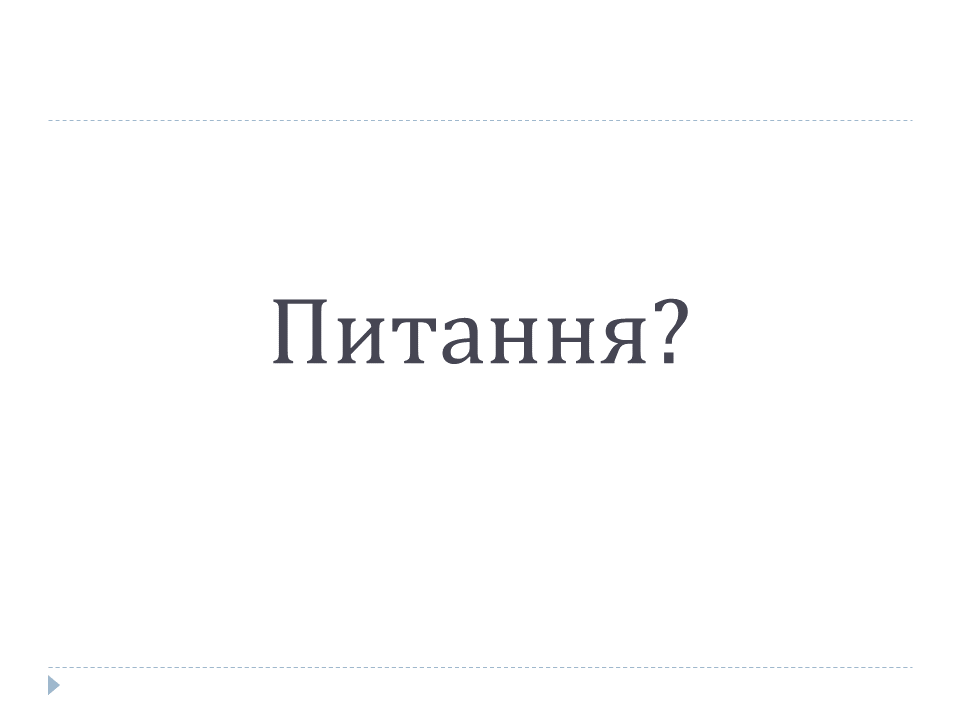


Рисунок Г.0.11 - Слайд 11