**МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ**
**ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ УПРАВЛІННЯ, ПСИХОЛОГІЇ ТА БЕЗПЕКИ**

**Кафедра інформаційних технологій**

# РОЗРОБЛЕННЯ СИСТЕМИ ВИМІРЮВАННЯ ПОКАЗНИКА РН РОЗЧИНІВ ДЛЯ КРИМІНАЛІСТИЧНОЇ ЕКСПЕРТИЗИ ТА ЛАБОРАТОРНОГО АНАЛІЗУ В ПОЛІЦІЇ

**Кваліфікаційна робота**
здобувача вищої освіти
4 курсу денної форми навчання
**Ярослави ЖАБ'ЯК**

**Науковий керівник:**
доцент кафедри ІТ
**Ігор ФАРМАГА**

**Рецензент:**

_____-
вчене звання, науковий ступінь

_____-
(Ім'я ПРІЗВИЩЕ рецензента)

***Кваліфікаційна робота допущена до захисту***
«___» _____ 2026 р., протокол № ____

Завідувач кафедри інформаційних технологій
_____ **Олег ЗАЧЕК**_
(підпис)

Львів
2026

# СПИСОК УМОВНИХ СКОРОЧЕНЬ

ПК (Персональний комп’ютер) – обчислювальна машина для індивідуального користування.

МК (Мікроконтролер) – інтегральна схема, для керування електронними пристроями.

АЗ/ПЗ – відповідно, апаратна частина (технічне оснащення) та програмна складова системи.

САПР (Computer-Aided Design) – комплекс програм для автоматизації процесів проектування та конструювання.

Arduino Uno – модуль для розробки, побудований на базі мікроконтролера ATmega328.

Arduino IDE – програмне середовище, що використовується для написання, налагодження та завантаження прошивок у контролери сімейства Arduino.

AVR — архітектура 8-бітних мікроконтролерів, компанії Atmel.

РКД/LCD – дисплеї відображення інформації на основі рідких кристалів.

АЦП (Analog-to-Digital Converter) – перетворювач, що трансформує аналоговий сигнал у цифровий код.

ПЗП (Read-Only Memory) – енергонезалежний тип пам’яті, де зберігаються дані, що не потребують постійного оновлення.

Flash – напівпровідникова пам’ять, яка зберігає записану інформацію при відключенні живлення та підтримує багаторазове перезаписування.

EEPROM – постійна пам’ять з можливістю електричного стирання та повторного програмування окремих байтів.

USART – універсальний приймач та передавач для послідовної передачі даних як у синхронному, так і в асинхронному режимах.

$I^2C$ – двопровідна шина для зв’язку, що використовує лінії даних (SDA) та синхронізації (SCL).

ШІМ (Pulse-Width Modulation) – метод керування потужністю сигналу шляхом зміни тривалості імпульсів при незмінній частоті.

# АНОТАЦІЯ

**Жаб'як Я., Фармага І.** (керівник). Розроблення системи вимірювання показника ph розчинів для криміналістичної експертизи та лабораторного аналізу в поліції. Бакалаврська кваліфікаційна робота. – Львівський державний університет внутрішніх справ, Львів, 2026.

В дипломній роботі розроблено інформаційну систему вимірювання показника pH розчинів для криміналістичної експертизи та лабораторного аналізу в поліції. Система забезпечує вимірювання фізико-хімічних параметрів розчинів таких як: температура, водневий показник (pH). Виміряні значення виводяться на РКД та через послідовний порт в ПК.

Система контролює температуру розчину, справність сенсора температури і сигналізує про аварійні ситуації .

При розробці інформаційної системи вимірювання показника ph розчинів використано сучасну елементну базу. Основними її складовими є платформа Arduino Uno на мікроконтролері AVR ATmega328, цифровий давач температури DS18B20, сенсор кислотності pH, і алфавітно-цифровий рідкокристалічний дисплей 16x2.

Спроектовано електричну принципову схему й створено модель інформаційної системи вимірювання показника ph розчинів в Proteus VSM.

Розроблено її алгоритм роботи і програмне забезпечення для платформи Arduino Uno на мові С в середовищі Arduino IDE.

Створено програмне забезпечення для взаємодії з цифровим термодатчиком та сенсором pH, а також інтегровано підпрограму для відображення даних на рідкокристалічному дисплеї. Процес функціонування системи було протестовано у середовищі Proteus ISIS. Отримані дані підтвердили працездатність основного алгоритму та точність розроблених модулів для вимірювання кислотності розчинів.

**Ключові слова:** платформа Arduino Uno, інформаційна система, сенсор показника pH, цифровий давач DS18B20, САПР Proteus VSM, Arduino IDE, мова програмування С.

# ABSTRACT

**Zhabiak J., Farmaha I.** (Supervisor). Development of a pH measurement system for forensic examination and police laboratory analysis. Bachelor's thesis. – Lviv State University of Internal Affairs, Lviv, 2026.

This bachelor's thesis presents the development of an information system for measuring the pH levels of solutions, specifically designed for forensic examination and police laboratory analysis. The system provides measurement of physicochemical parameters, such as temperature and potential of hydrogen (pH). The measured values are displayed on an LCD and transmitted to a PC via a serial port.

The system monitors the temperature of the solution, ensures the proper functioning of the temperature sensor, and provides alerts in case of emergency situations.

The information system was developed using a modern electronic component base. Its core components include the Arduino Uno platform based on the AVR ATmega328 microcontroller, a DS18B20 digital temperature sensor, a pH acidity sensor, and a 16x2 alphanumeric liquid crystal display.

The electrical schematic diagram was designed, and a model of the pH measurement information system was created in Proteus VSM. The operational algorithm and software were developed for the Arduino Uno platform using the C programming language within the Arduino IDE environment.

Software modules were created to interact with the digital thermal sensor and pH sensor, along with an integrated subroutine for data visualization on the LCD. The system's operation was tested in the Proteus ISIS simulation environment. The obtained data confirmed the efficiency of the main algorithm and the accuracy of the developed modules for measuring the acidity of solutions.

**Keywords:** Arduino Uno platform, information system, pH sensor, DS18B20 digital sensor, Proteus VSM CAD, Arduino IDE, C programming language.

**ЗМІСТ**

# ВСТУП

Вимірювання кислотності (рівня pH) є критично важливим процесом у багатьох сферах, оскільки цей показник визначає хімічні властивості розчинів, швидкість реакцій та життєздатність біологічних організмів.

Ось основні причини, чому це необхідно: 1. Харчова промисловість: Контроль pH необхідний для виробництва молочних продуктів (сирів, йогуртів), напоїв (вина, пива) та консервації. Певний рівень кислотності запобігає розмноженню шкідливих бактерій. 2. Хімічне виробництво: Багато хімічних процесів, таких як синтез ліків або виробництво добрив, вимагають суворого дотримання діапазону pH для отримання потрібного продукту. 3. Захист обладнання: Занадто кислий або занадто лужний розчин може спричинити корозію труб та резервуарів на підприємствах. 4. Екологія та сільське господарство: Якість води: Рівень pH води у водоймах впливає на життя риб та рослин. Наприклад, підвищення кислотності води в річках може призвести до загибелі екосистеми. 5. Родючість ґрунту: Різні сільськогосподарські культури потребують специфічного рівня pH ґрунту для засвоєння мінеральних речовин. 6. Водоканали: Перед тим як вода потрапить у водогін, її pH коригують, щоб вона була безпечною для споживання та не руйнувала міські мережі. 7. Очищення води. Очищення стоків: На етапі фільтрації стічних вод зміна pH допомагає осаджувати важкі метали та інші шкідливі домішки. 8. Криміналістика та медицина. Експертиза доказів: У криміналістиці вимірювання pH допомагає ідентифікувати невідомі речовини, знайдені на місці події, або визначити склад хімічних розчинів, що могли бути використані у злочинних цілях. 9. Лабораторна діагностика: Контроль кислотності біологічних рідин (крові, сечі) є важливим маркером стану здоров’я людини.

Метою роботи є розробка апаратного і програмного забезпечення інформаційної системи вимірювання показника ph розчинів для криміналістичної експертизи та лабораторного аналізу в поліції. Система має

вимірювати такі фізико-хімічні параметри розчинів: температура, кислотність (pH).

Для побудови технічної бази системи контролю кислотності розчинів доцільно застосувати платформу Arduino Uno, серцем якої є мікроконтролер ATmega328 (AVR) виробництва Atmel, у поєднанні з цифровим термодатчиком DS18B20. Сучасне приладобудування важко уявити без використання мікроконтролерів, оскільки вони інтегровані у більшість інтелектуальних пристроїв.

Завдяки високому ступеню інтеграції периферійних модулів безпосередньо на кристалі, МК здатні керувати складними вузлами за мінімальної кількості зовнішніх компонентів. Такий підхід забезпечує компактність пристрою, високу енергоефективність та економічну вигідність проєкту.

Для реалізації поставленої мети визначено такі **завдання**:

- спроектувати інформаційну систему вимірювання показника pH та створити її модель в САПР Proteus VSM з використанням наступних електронних компонент: мікроконтролерна плата Arduino Uno (базована на МК ATmega328P), модуль символьного РК-індикатора формату 1602, термодатчик DS18B20 з цифровим інтерфейсом 1-Wire, аналоговий вимірювач pH серії SEN0161;
- розробити алгоритм роботи інформаційної системи вимірювання показника pH розчинів;
- розробити програмні функції для роботи з давачем температури та отримання даних з давачів;
- розробити основне вбудоване програмне забезпечення інформаційної системи вимірювання показника pH розчинів:
- дослідити роботу моделі інформаційної системи вимірювання показника pH та провести аналіз отриманих результатів в Proteus ISIS.

**Об’єкт дослідження** – процес вимірювання та контролю показника pH рідких середовищ (розчинів) в умовах криміналістичних експертиз та спеціальних лабораторних аналізів для потреб правоохоронних органів.

**Предмет дослідження** – методи, алгоритми, апаратно-програмні засоби системи вимірювання pH розчинів для криміналістичної експертизи та лабораторного аналізу.

**Методи досліджень.** Для досягнення поставленої мети використано комплекс теоретичних та експериментальних методів: методи системного аналізу та літературного пошуку, методи теорії електричних кіл та схемотехнічного проектування, методи алгоритмізації та об’єктно-орієнтованого програмування, методи цифрової фільтрації та статистичної обробки сигналів, методи комп’ютерного моделювання та аналітичної апроксимації, методи експериментальних досліджень та метрологічного градуювання (калібрування), методи метрологічного аналізу та оцінювання похибок.

**Структура роботи.** Кваліфікаційна робота складається із вступу, чотирьох розділів, висновків, списку використаних джерел, додатків. Обсяг основного тексту роботи складає 56 сторінок, 37 рисунків, 5 таблиць, 1 додаток і 12 бібліографічних джерела. Загальний обсяг роботи – 67 сторінок.

# РОЗДІЛ 1
# АНАЛІЗ ТА ДОСЛІДЖЕННЯ ПРОБЛЕМНОЇ ОБЛАСТІ

## 1.1. Основні показники розчинів та їх оцінка

Для аналізу стану розчинів у лабораторній або криміналістичній практиці використовують низку фізико-хімічних показників. Їхня оцінка дозволяє ідентифікувати речовину, визначити її концентрацію та придатність для використання. Ключові параметри та методи їхнього оцінювання:

1. Водневий показник (pH). Це головний індикатор кислотності або лужності середовища. Оцінка: pH<7: Кисле середовище (кислоти, соки, шлунковий сік). pH=7: Нейтральне (дистильована вода). pH>7: Лужне (мило, нашатирний спирт, відбілювач). Метод вимірювання: Використання електронних pH-метрів забезпечує точність до 0.01, тоді як лакмусовий папір дає лише приблизне значення.

2. Температура. Температура впливає на швидкість хімічних реакцій та точність вимірювання інших показників (наприклад, pH суттєво залежить від температури). Оцінка: Вимірюється в градусах Цельсія (°C).

Метод вимірювання: Терморезистори або цифрові датчики типу DS18B20, які дозволяють автоматизувати компенсацію похибок при вимірюванні pH.

3. Електропровідність (EC). Здатність розчину проводити електричний струм. Вона прямо залежить від кількості розчинених солей (іонів). Оцінка: Висока провідність вказує на високу мінералізацію або наявність забруднень.

Застосування: Оцінка чистоти питної води або контроль складу добрив у гідропоніці.

4. Концентрація розчинених часток (TDS). Загальна кількість розчинених твердих речовин (солі, мінерали, метали). Оцінка: Вимірюється в ppm (частин на мільйон). 0–50 ppm — ідеально чиста вода. 100–300 ppm — звичайна водопровідна вода. понад 500 ppm – технічна вода, не рекомендована для пиття.

5. Окислювально-відновний потенціал (ORP/Redox). Показує здатність розчину приєднувати або віддавати електрони. Оцінка: Вимірюється в мілівольтах (мВ). Застосування: Важливий для перевірки дезінфекції води (наприклад, у басейнах) та оцінки якості стічних вод.

Шкала pH. Для розуміння результатів вимірювань використовується стандартна шкала від 0 до 14: pH<7: Кисле середовище (наприклад, лимонний сік, електроліт). pH=7: Нейтральне середовище (дистильована вода). pH > 7: Лужне середовище (наприклад, мильний розчин, відбілювач).

Хімічні методи використовуються для визначення кислотності, лужності у розчині металів, солей, органічних та синтетичних речовин.

## 1.2. Огляд існуючих приладів контролю кислотності розчинів

Для контролю кислотності розчинів використовують різні прилади — від найпростіших хімічних тестів до складних цифрових аналізаторів. Вибір приладу залежить від необхідної точності, умов експлуатації та бюджету. Основні типи пристроїв:

1. Портативні pH-метри (“ручки”) – це компактні електронні пристрої, що поєднують в одному корпусі електрод, обчислювальний блок і дисплей.

Призначення: Швидкі вимірювання в польових умовах, побуті або невеликих лабораторіях. Переваги: Зручність, низька ціна, миттєвий результат. Недоліки: Потребують частого калібрування в буферних розчинах.

2. Стаціонарні (лабораторні) pH-метри – це високоточні прилади, які використовуються у професійних хімічних та криміналістичних лабораторіях. Призначення: Проведення складних аналізів, де важлива похибка до 0,1 pH. Особливості: мають виносні електроди (скляні або пластикові), функції автоматичної температурної компенсації (АТК) та можливість підключення до ПК для запису даних.

3. Промислові контролери – це стаціонарні системи для безперервного моніторингу в режимі реального часу. Призначення: очисні споруди, заводи, басейни. Особливості: Зазвичай монтуються в щити керування. Можуть

автоматично подавати сигнал на насоси для додавання реагентів, якщо рівень pH виходить за межі норми.

Системи на базі мікроконтролерів. Аналоговий pH-модуль (наприклад, серії SEN), скляний електрод, датчик температури та контролер. Переваги: можливість повної автоматизації, передачі даних через Wi-Fi/Bluetooth, низька собівартість та гнучкість налаштувань.

Візуальні методи (Індикатори): найстаріший і найпростіший спосіб, що базується на зміні кольору речовини-індикатора. Лакмусовий папір: змінює колір залежно від середовища. Оцінка проводиться шляхом порівняння зі шкалою-еталоном. Рідкі індикатори: краплі, що додаються в розчин (наприклад, фенолфталеїн). Точність: дуже низька (зазвичай крок становить 1 одиницю pH), що недостатньо для професійної експертизи. Прилади для контролю кислотності розчинів:


Рис. 1.1 Капельний (рідкий) тестер


Рис.1.2. Професійний ручний pH-метр BENETECH GM761 (ціна 1684 грн)

Шкала вимірювання pH 0..14

Електронний вимірювач pH PH600 зображено на Рис.1.3. Шкала вимірювання pH 0..14 , робоча температура до 80° C.

Рис. 1.3. Електронний вимірювач pH PH600 (ціна 115$)

Перелік приладів для вимірювання того чи іншого параметра розчинів можна продовжити, але всі вони або професійні і відносно дорогі. Для якісної оцінки pH-розчинів, є потреба у дешевих приладах та інформційно-вимірювальних системах, які б моніторили такі параметри розчину як кислотність та температура та у випадку перевищення гранично-допустимих норм давали відповідні повідомлення. Отже розробка подібних систем є актуальною.

В дипломній роботі запропоновано інформаційну систему вимірювання показника pH розчинів з використанням сенсорів температури, pH (кислотності). Інформаційна система може використовуватися в криміналістичної експертизі та лабораторному аналізі в поліції для контролю pH-розчину, має хороші можливості для обробки та представлення інформації, компактна, набагато дешевша існуючих пристроїв, які мають те саме призначення.

# РОЗДІЛ 2
# ЗАСОБИ РОЗРОБКИ АПАРАТНОГО І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВИМІРЮВАННЯ ПОКАЗНИКА РН РОЗЧИНІВ

## 2.1. Система автоматизованого проектування і моделювання програмованих пристроїв Proteus VSM

Система автоматизованого проектування (САПР) Proteus VSM від компанії Labcenter Electronics є потужним рішенням для комплексного проектування та моделювання електронних пристроїв, зокрема тих, що базуються на мікроконтролерах. В основі роботи програми лежить застосування точних математичних алгоритмів, що дозволяють достовірно імітувати поведінку широкого спектра електронних компонентів. Головною особливістю платформи є її здатність до реалістичної симуляції роботи мікроконтролерів, DSP-процесорів та інших інтелектуальних вузлів у режимі реального часу.

Функціонал програмного середовища охоплює повний цикл розробки: від створення принципової схеми та її верифікації за допомогою віртуальних приладів до відлагодження програмного забезпечення мікроконтролера.

Додатково пакет оснащений засобами для проектування друкованих плат, включаючи можливість тривимірного перегляду майбутнього виробу, що важливо для контролю фізичних габаритів.

Система підтримує значну кількість архітектур мікропроцесорів, зокрема ARM, AVR, PIC, 8051, Motorola та Basic Stamp. Бібліотека компонентів містить понад 6000 елементів, а гнучка інтеграція з різними компіляторами та асемблерами забезпечує можливість налагодження навіть складних систем, що складаються з кількох процесорів.

Програмний комплекс складається з двох взаємопов’язаних підсистем:

- ISIS: середовище для розробки принципових схем та проведення їхньої інтерактивної симуляції, де користувач може спостерігати за роботою пристрою в динаміці.
- ARES: спеціалізований інструмент для створення топології друкованих плат. Він дозволяє виконувати розведення провідників та розміщення компонентів на основі раніше спроектованої в ISIS схеми, використовуючи при цьому алгоритми автоматичного трасування.

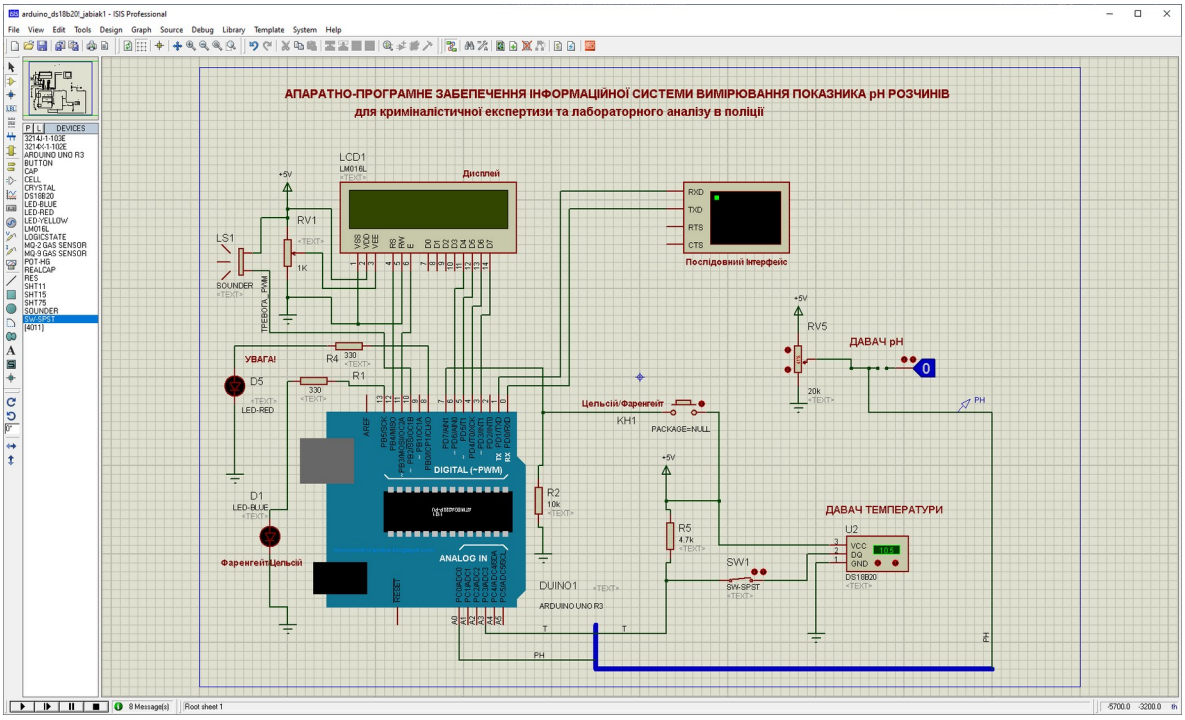



Рис. 2.1. Середовище Proteus ISIS

Для інтеграції симуляційної моделі з фізичними периферійними пристроями у середовищі Proteus реалізовано спеціальні програмні модулі. Зокрема, компонент COMPIM призначений для трансляції даних через апаратний COM-порт ПК, тоді як елемент USBCONN слугує містком для роботи з USB-інтерфейсом.

Програмний комплекс вирізняється глибокою інтеграцією з популярними середовищами розробки (IDE) та сторонніми компіляторами, що значно спрощує процес налагодження коду для різних архітектур:

Для платформи AVR: забезпечено повну сумісність із такими інструментами, як CodeVisionAVR, ICC та WinAVR (що використовує компілятор GCC).

Для сімейств 8051 та ARM: передбачена ефективна взаємодія з екосистемою Keil.

## 2.2. Інструментарій розробки ПЗ Arduino IDE для вбудованих систем на МК AVR

Програмний комплекс Arduino IDE є багатофункціональним середовищем розробки, яке надає фахівцям зручний інструментарій для швидкого написання та відлагодження коду. Ця платформа оптимізована для оперативного проектування інтелектуальних електронних модулів і систем різної складності.

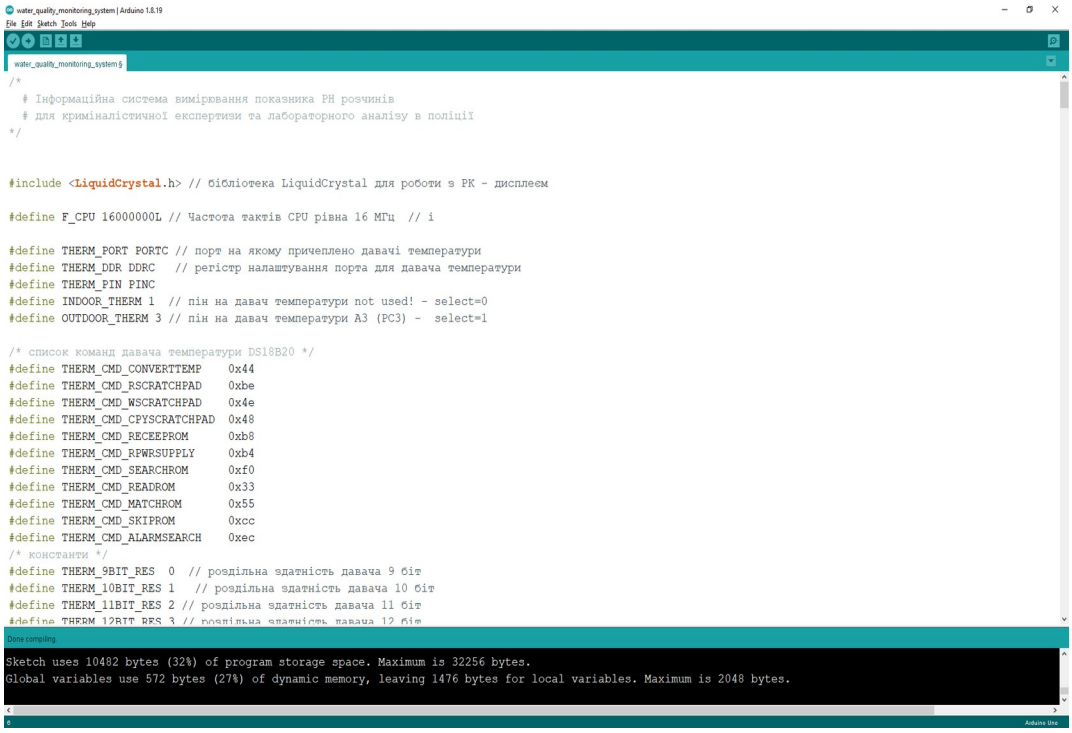



Рис. 2.1. Arduino IDE (Integrated Development Environment)

*Особливості середовища Arduino IDE*. Дане програмне забезпечення розроблене спеціально для взаємодії з однойменною апаратною платформою. Своєю популярністю Arduino IDE завдячує відкритому вихідному коду, лаконічному синтаксису, а також величезному масиву доступних драйверів і бібліотек. Суттєвою перевагою є підтримка прямого програмування мікроконтролерів через USB-порт, що дозволяє відмовитися від використання додаткових апаратних програматорів.

Робочий простір програми складається з кількох функціональних зон: текстового вікна для написання коду, області системних сповіщень (терміналу), консолі та панелі швидкого доступу до основних команд. Програмні файли, створені в цьому середовищі, отримали назву «скетчі». Технологічним підґрунтям для них слугують мови програмування C та C++.

Безпосередня робота над алгоритмами здійснюється у вбудованому редакторі, який оснащений усіма необхідними інструментами для роботи із текстом: від звичайного редагування до розширеного пошуку та заміни елементів. Усі технічні дані, включаючи детальні звіти про хід компіляції та виявлені помилки, виводяться у вікно консолі. Для зручного керування ключовими етапами розробки передбачена інтуїтивно зрозуміла панель інструментів.

*Функціональні можливості інтерфейсу Arduino IDE*. Для керування процесом розробки передбачено набір піктограм швидкого доступу:

- Verify/Compile (Перевірка): запускає процес синтаксичного аналізу програмного коду та його перетворення у двійковий формат.
- Stop (Зупинка): завершує активну сесію обміну даними через послідовний інтерфейс.
- New (Створити): відкриває вікно для розробки нового програмного модуля з чистого аркуша.
- Open (Відкрити): викликає провідник для вибору та завантаження раніше створених файлів.
- Save (Зберегти): записує актуальний стан проекту на накопичувач комп'ютера.
- Upload (Вивантаження): виконує фінальну компіляцію та записує отриману прошивку в енергонезалежну пам’ять мікроконтролера.
- Serial Monitor (Монітор порту): відкриває інтерактивне вікно для моніторингу даних, що надходять від плати в реальному часі.

*Структура головного меню*. Додаткові параметри та інструменти розподілені за тематичними категоріями: File (Файл), Edit (Правка), Sketch

(Скетч), Tools (Інструменти) та Help (Довідка). Вміст цих розділів може адаптуватися до поточного контексту роботи користувача.

*Можливості вкладки "Edit".* Окрім стандартних операцій із текстом, цей розділ пропонує специфічні функції експорту:

- Copy for Forum: формує копію коду з розміткою, придатною для коректного відображення у повідомленнях на спеціалізованих веб-ресурсах.
- Copy as HTML: перетворює текст скетчу на HTML-код, зберігаючи підсвічування синтаксису для публікації на веб-сторінках.

*Інструменти вкладки "Sketch".* Цей розділ фокусується на роботі зі структурою проекту та зовнішніми модулями:

- Include Library: забезпечує підключення сторонніх бібліотек, автоматично прописуючи відповідні інструкції #include у заголовку файлу.
- Show Sketch Folder: миттєво відкриває системну директорію, у якій розміщено вихідні файли поточного проекту.
- Add File...: інтегрує зовнішні документи до скетчу, створюючи для них окремі вкладки в редакторі.

*Функціональні можливості вкладки "Tools" (Інструменти).* Цей розділ меню містить сервісні засоби для налаштування апаратної частини та впорядкування програмного тексту:

- Auto Format (Автоформатування): покращує візуальну структуру скетчу, автоматично виправляючи відступи та перевіряючи цілісність логічних дужок, що полегшує аналіз коду.
- Board (Плата): інструмент для вибору конкретної модифікації контролера Arduino, під яку буде здійснюватися компіляція.
- Serial Port (Послідовний порт): надає доступ до переліку активних COM-портів. Система автоматично перевіряє наявність нових підключень щоразу, коли користувач відкриває цей підпункт.

- Burn Bootloader (Записати завантажувач): функція для прошивки системного завантажувача в пам'ять МК. Для успішного завершення операції необхідно заздалегідь сконфігурувати тип плати та вибрати відповідний тип програматора (наприклад, AVR ISP).

*Особливості розділу "File" (Файл).* Ключовим компонентом тут є Sketchbook, що виступає в ролі базової робочої папки для збереження проектів. Швидкий перехід до раніше створених робіт можливий безпосередньо через меню "Файл" або стандартне діалогове вікно відкриття.

*Організація роботи з документами.* Середовище розробки підтримує паралельну роботу з декількома проектами, кожен з яких відкривається в окремому вікні. Програма коректно взаємодіє з файлами різних форматів, що складають основу сучасної вбудованої розробки:

1. Скетчі: стандартні файли проектів Arduino з розширенням .ino (у застарілих версіях — без розширення).
2. Вихідний код: класичні файли на мовах C (.c) та C++ (.cpp).
3. Заголовні файли: допоміжні документи .h, що використовуються для підключення бібліотек та оголошення функцій.

*Алгоритм програмування контролера Arduino.* Перед початком завантаження програмного коду в апаратну частину необхідно задати параметри конфігурації. У меню Tools (Інструменти) слід послідовно визначити тип використовуваної платформи у розділі Board, а також вказати активний канал зв'язку в підпункті Serial Port.

В операційній системі Windows доступні інтерфейси відображаються як COM-порти. Це можуть бути як стандартні апаратні порти (COM1 чи COM2), так і віртуальні з'єднання, що виникають при підключенні через USB (наприклад, COM4 або порти з вищими порядковими номерами). Як тільки налаштування завершено, можна запускати процес прошивки, натиснувши відповідну кнопку на панелі швидкого доступу або вибравши команду Upload у вкладці Sketch.

Під час завантаження даних на плату (зокрема на модель Arduino Uno) про успішний обмін інформацією свідчить динамічна індикація світлодіодів RX та TX. Технологічну можливість такого спрощеного програмування забезпечує Bootloader (завантажувач). Це спеціальне малогабаритне програмне забезпечення, що зберігається в пам'яті мікроконтролера і дозволяє оновлювати прошивку безпосередньо через USB-інтерфейс, усуваючи необхідність у додатковому обладнанні. Завантажувач переходить у робочий режим під час кожного скидання (Reset) або на початку сесії програмування, що зазвичай супроводжується миготінням системного світлодіода.

*Робота з бібліотеками та інтерфейсами.* Для розширення базових можливостей платформи та спрощення взаємодії з периферією застосовуються бібліотеки. Вони дозволяють інтегрувати готові алгоритми обробки сигналів та складні протоколи керування обладнанням. Додати нові модулі можна через автоматизоване меню Include Library або шляхом копіювання вихідних файлів безпосередньо у папку libraries. Важливо пам’ятати, що кожна підключена бібліотека збільшує обсяг займаної Flash-пам’яті, тому варто уникати використання надлишкових компонентів.

Для налагодження та візуалізації роботи пристрою в реальному часі використовується Serial Monitor. Цей інструмент забезпечує двосторонній зв’язок: відображення текстової інформації від контролера та відправку керуючих команд через термінальний ввід користувача.

*Налаштування середовища:* Основні параметри Arduino IDE редагуються у вкладці Preferences. Якщо певну функцію неможливо змінити через графічний інтерфейс, це можна зробити у конфігураційному файлі, адреса якого вказана у нижній частині вікна налаштувань.

*Апаратні особливості та електричні характеристики.* Кожна плата Arduino має певну кількість універсальних виводів (пінів) для зв'язку з зовнішніми компонентами. Залежно від архітектури, вони можуть опрацьовувати як цифрові, так і неперервні (аналогові) сигнали за допомогою вбудованого АЦП.

*Керування портами:* Режим роботи кожного виводу – як входу для зчитування, так і виходу для керування – визначається за допомогою функції pinMode(). Варто зазначити, що цифрові входи можуть переходити у стан високого імпедансу, що робить їх чутливими до електромагнітних завад.

*Стабілізація вхідного потенціалу:* Для запобігання хибним спрацюванням на вільних входах, де спостерігається “плаваючий” рівень напруги, необхідно застосовувати підтягувальні резистори. Вони жорстко фіксують логічний рівень, з’єднуючи вхід із шиною живлення або масою (GND), що забезпечує стабільну роботу логіки.

*Експлуатаційні обмеження:* При проектуванні схем критично важливо враховувати граничні параметри струму. Максимальне навантаження на один вивід не повинно перевищувати 40 мА. Цієї потужності недостатньо для безпосереднього підключення силових агрегатів, таких як електродвигуни або реле, тому для них слід використовувати драйвери або транзисторні ключі. Перевищення допустимих значень струму або випадкове замикання може спричинити фізичне пошкодження кристала мікроконтролера.

*Застосування широтно-імпульсної модуляції (ШІМ).* Метод ШІМ базується на генерації цифрового сигналу у вигляді послідовності прямокутних імпульсів, де змінюється співвідношення між тривалістю логічної одиниці та періодом сигналу. Такий підхід дозволяє імітувати зміну амплітуди вихідної напруги від 0 до 5 В. Регулювання ефективного (середнього) значення напруги відбувається через корекцію ширини імпульсу при сталій частоті проходження сигналів. У програмному середовищі Arduino для реалізації цього механізму передбачена функція analogWrite(). Параметр інтенсивності в ній варіюється від 0 до 255 одиниць:

- analogWrite(255): забезпечує постійний рівень напруги 5 В
- analogWrite(127): створює сигнал із коефіцієнтом заповнення 50%, що відповідає середній напрузі 2,5 В.

*Робота з аналоговими інтерфейсами та архітектура ПЗ*. Мікроконтролери серії ATmega інтегрують у собі 6-канальний аналого-

цифровий перетворювач (АЦП). Завдяки 10-бітному розрізненню, система здатна масштабувати вхідну напругу в числову послідовність від 0 до 1023. Характерною рисою аналогових входів є їхня взаємозамінність: вони можуть бути сконфігуровані як звичайні цифрові порти (із відповідною адресацією від 14 до 19). Приклад переведення аналогового входу у цифровий режим роботи:

- pinMode(14, OUTPUT); // ініціалізація виводу як виходу
- digitalWrite(14, HIGH); // подача логічної одиниці

*Особливості програмної структури:* Перед безпосередньою компіляцією програмний “скетч” проходить етап автоматичного перетворення на стандартний C/C++ код, який згодом опрацьовується інструментарієм AVR-GCC.

Модельний ряд контролерів Arduino включає численні платформи, що різняться архітектурою інтегрованих мікросхем. Найбільш відомі версії, як-от Uno та Duemilanove, функціонують на базі мікроконтролера ATmega328, який забезпечує баланс між продуктивністю та енергоспоживанням.

Для реалізації складних інженерних рішень, що вимагають підвищених обчислювальних ресурсів, застосовується плата Arduino Mega2560. Вона базується на потужному чіпі ATmega2560, що дозволяє опрацьовувати більші масиви даних та керувати значною кількістю периферійних пристроїв.

Етап вибору апаратної бази є фундаментальним при розробці системи, оскільки характеристики обраної плати безпосередньо впливають на:

- продуктивність: тактову частоту ядра процесора;
- комунікацію: швидкість передачі даних через інтерфейси зв’язку;
- ресурсну місткість: об’єм Flash-пам’яті, доступної для завантаження програмних алгоритмів.

На Рис.2.2 представлена конструкція та компонування елементів найбільш популярної платформи — Arduino UNO.

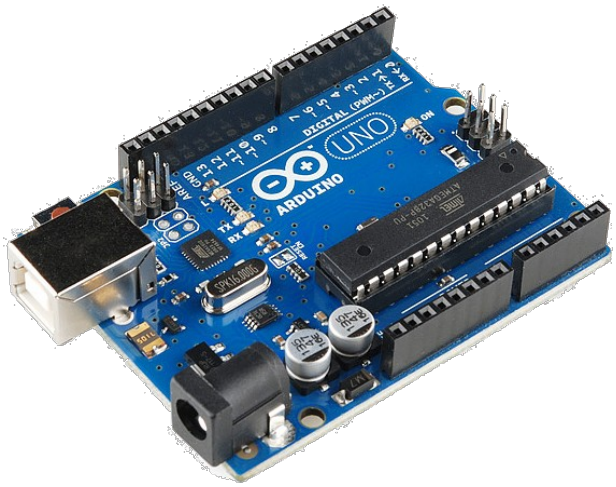


Рис. 2.2. Апаратна платформа Arduino UNO

Для під’єднання Arduino UNO до ПК потрібно вибрати платформу Arduino UNO (Tools>Board>Arduino> Genuino Uno), Рис. 2.3.

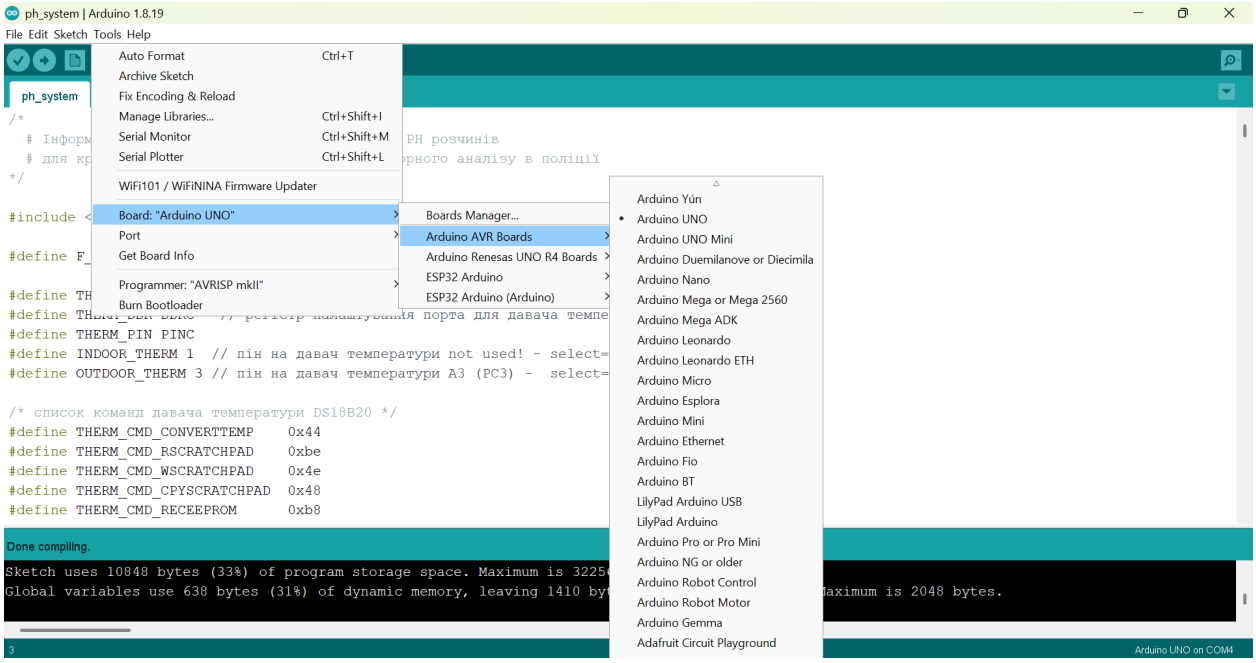



Рис. 2.3. Вибір платформи в меню Tools>Board>Arduino> Genuino Uno

Та вибрати порт через який буде здійснено підключення (Tools>Port).

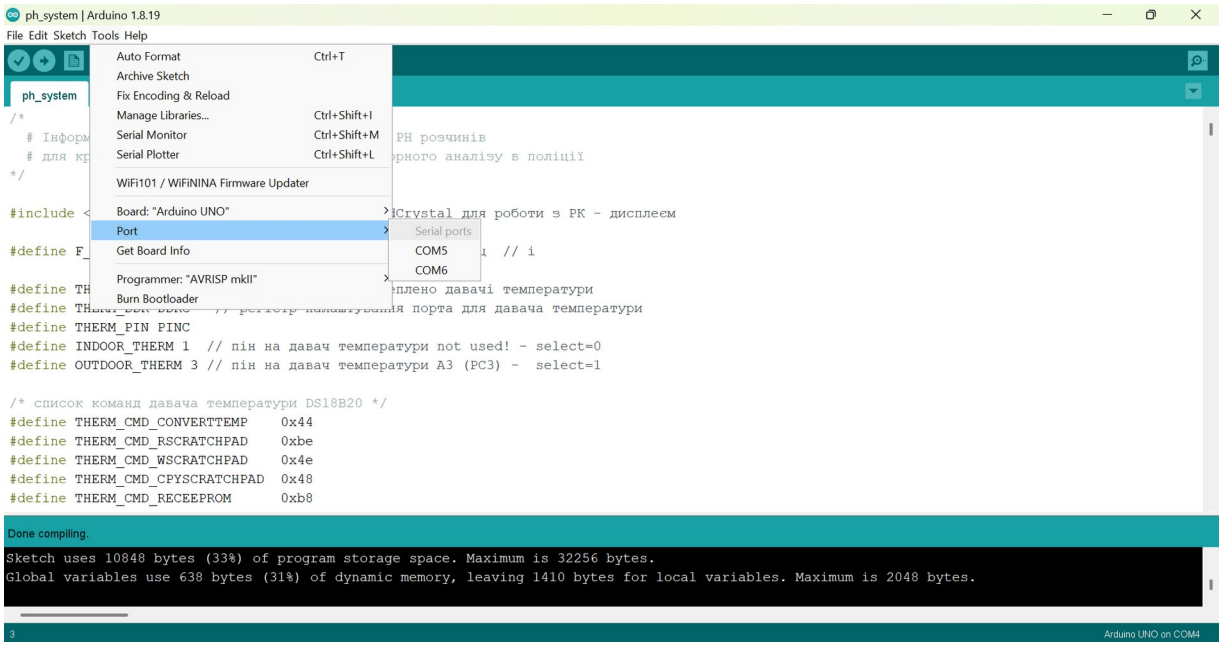



Рис. 2.4. Вибір порта підключення плати Arduino

Для запису розробленого програмного алгоритму в пам'ять мікроконтролера ATmega328p, встановленого на платформі Arduino Uno, необхідно активувати команду Upload у вкладці Sketch. Процес передачі даних супроводжується візуальною індикацією – миготінням вбудованого світлодіода на платі.

# РОЗДІЛ 3
# АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВИМІРЮВАННЯ ПОКАЗНИКА РН РОЗЧИНІВ ДЛЯ ЛАБОРАТОРНОГО АНАЛІЗУ В ПОЛІЦІЇ

## 3.1. Вибір електронних компонент для проектування інформаційної системи вимірювання показника pH розчинів

Процес розробки інформаційної системи для лабораторного визначення рівня pH передбачає використання такої елементної бази:

- мікроконтролерна плата Arduino Uno (базована на МК ATmega328P);
- модуль символьного РК-індикатора формату 1602;
- термодатчик DS18B20 з цифровим інтерфейсом;
- аналоговий вимірювач pH серії SEN0161.

Основні версії плат Arduino включають широкий спектр контролерів, адаптованих під специфічні інженерні завдання. Серед ключових модифікацій варто виділити:

- Uno: найпопулярніша версія базової платформи, що вважається стандартом.
- Due: Продуктивне рішення, побудоване на 32-розрядній архітектурі ARM Cortex-M3 (SAM3U4E).
- Leonardo: Виконана на базі чипа ATmega32U4, що дозволяє платі імітувати периферійні пристрої (клавіатуру чи мишу).
- Mega 2560 та Mega: Серія для масштабних проєктів, що використовує МК ATmega2560 (з USB-інтерфейсом на базі ATMega8U2) або ATmega1280 відповідно.
- Mega ADK: Спеціалізована версія з підтримкою USB-host для інтеграції з Android-пристроями.
- Nano: Малогабаритне рішення, оптимізоване для роботи з макетними платами через роз'єм Mini-B.

- Mini та Pro Mini: Ультракомпактні модулі для фінальної реалізації пристроїв, де критичними є вартість та розміри.
- Pro: Орієнтована на складні інженерні системи, що потребують гнучкої інтеграції.
- LilyPad: Унікальний форм-фактор для носимої електроніки, призначений для монтажу безпосередньо на текстильні вироби.
- BT (Bluetooth): Контролер із вбудованим радіомодулем для бездротового обміну даними та прошивки.
- Duemilanove та Diecimila: Попередні покоління базових плат на мікроконтролерах серії ATmega.

Технічні характеристики обраної плати безпосередньо визначають обчислювальну потужність, конфігурацію завантажувача (Bootloader) та параметри синхронізації при передачі скетчів.

Arduino Uno – плата Arduino Uno (Рис.3.1) побудована на мікроконтролері ATmega328.

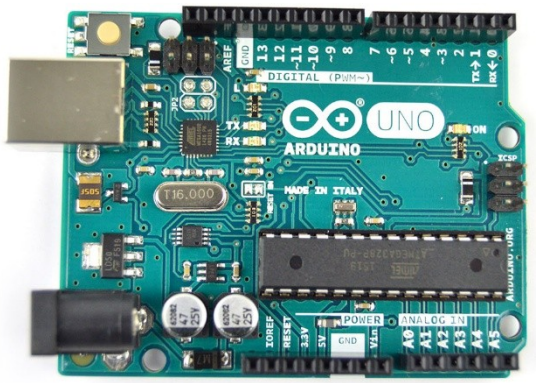



Рис. 3.1. Плата Arduino Uno

*Таблиця 3.1. Характеристики плати Arduino Uno*

| | |
|---|---|
| Мікроконтролер | ATmega328 |
| Робоча напруга | 5 В |
| Вхідна напруга | 7-12 В |
| Цифрові входи/виходи | 14 (6 з яких використовуються як виходи ШІМ) |
| Аналогові входи | 6 |
| Постійний струм через вхід/вихід | 40 мА |
| Флеш-пам'ять | 32 Кбайт, при цьому 0,5 Кбайт використовуються для завантажувача (bootloader). |
| ОЗП | 2 Кбайт |
| EEPROM | 1 Кбайт |
| Тактова частота | 16 МГц |

Arduino Pro Mini – ця плата (Рис.3.2) побудована на мікроконтролері ATmega168.

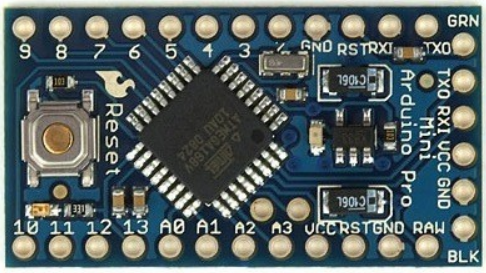



Рис. 3.2. Плата Arduino Pro Mini

Arduino Duemilanove – ця платформа використовує мікроконтролер ATmega168 або ATmega328.

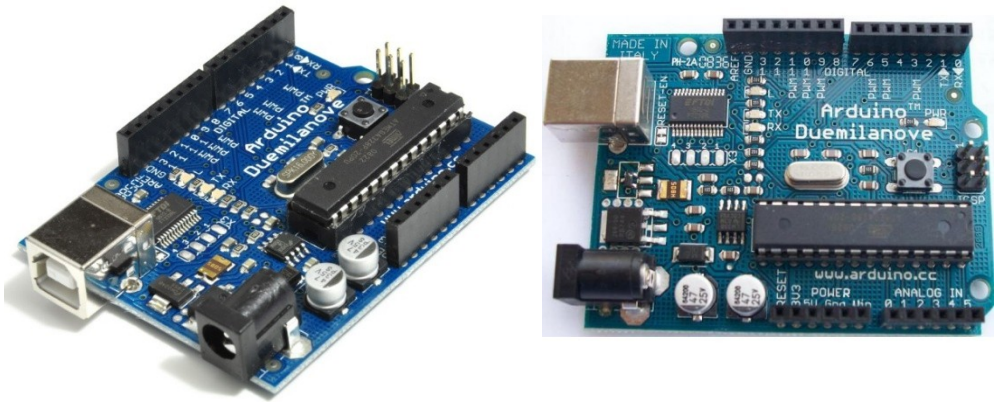



Рис. 3.3. Плата Arduino Duemilanove

Arduino Nano – плата Nano (Рис. 3.4), використовує МК ATmega328 (Arduino Nano 3.0) або ATmega168. (Arduino Nano 2.x), має малі розміри і може використовуватися в компактних виробах.

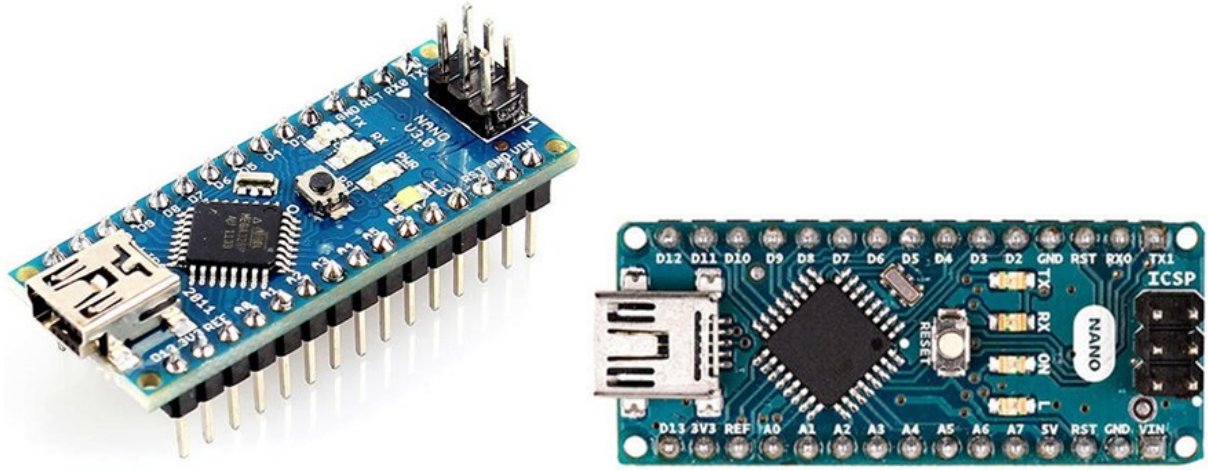



Рис. 3.4. Плата Arduino Nano

Arduino LilyPad – плата Arduino LilyPad (Рис. 3.5) розроблена для використання як елемент одягу. Її можна вшити в тканину з вбудованим джерелом живлення, сенсорами і приводами. Платформа має мікроконтролер ATmega168V.

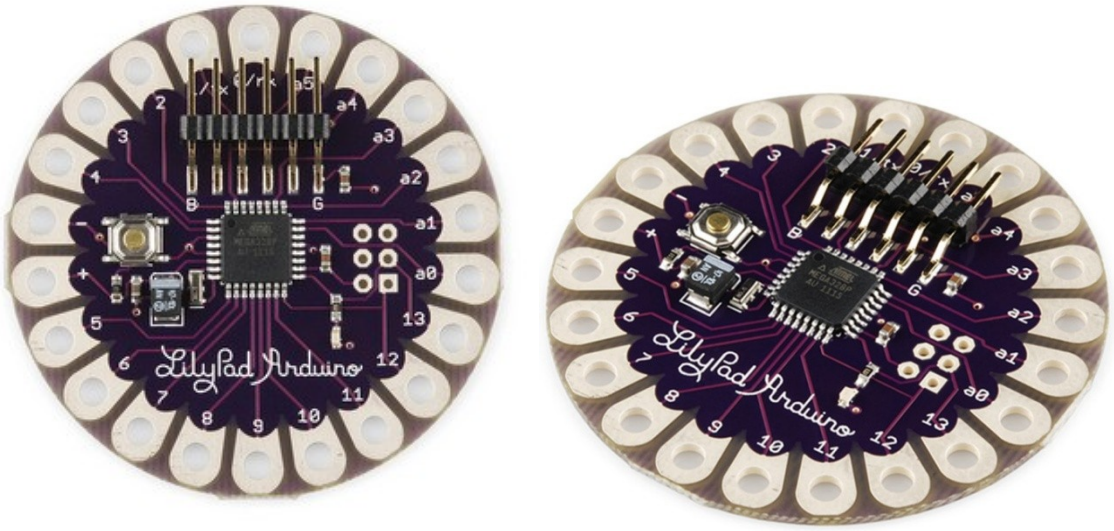



Рис. 3.5. Плата Arduino LilyPad

Arduino Due (рис. 3.6) - плата на базі процесора Atmel SAM3X8E ARM Cortex-M3. Це плата Arduino на основі 32-бітного МК з ARM-ядром. Arduino Due працює від 3,3 В. Максимальна напруга, яку подають на входи/виходи, становить 3,3 В.

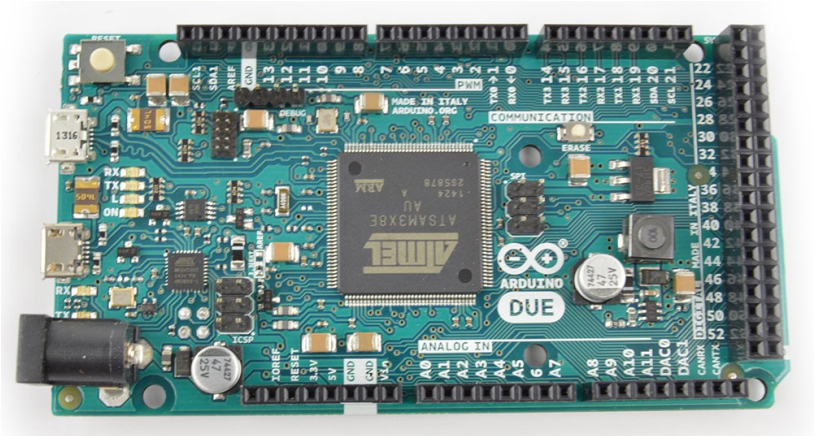



Рис. 3.6. Плата Arduino Due

Високий попит на платформу Arduino зумовлений не лише її бюджетністю та простотою освоєння середовища програмування. Ключовим фактором успіху є модульна архітектура, що дозволяє використовувати плати

розширення (шилди) для швидкого нарощування технічних можливостей системи.

*Конфігурування портів вводу-виводу Arduino.* Виводи МК платформ Arduino характеризуються універсальністю: вони можуть функціонувати у режимах як входу, так і виходу. Аналогові піни мікроконтролерів ATmega за необхідності можуть бути перепрограмовані для роботи як стандартні цифрові порти. Згідно з архітектурою, за замовчуванням усі виводи встановлені в режим входу, для якого характерним є стан високого імпедансу. Оскільки зміна логічного рівня у цьому стані потребує мінімального струму, не підключені (“плаваючі”) входи схильні до впливу електромагнітних завад, що призводить до отримання випадкових значень.

Для стабілізації роботи входів за відсутності зовнішнього сигналу застосовують метод фіксації потенціалу за допомогою резисторів на 10 кОм (підтяжка до живлення +5 В або до “землі”). Мікроконтролери сімейства ATmega мають інтегровані резистори номіналом 20 кОм, які можна активувати програмно. Програмна реалізація підтяжки виглядає наступним чином:

- pinMode(pin, INPUT); // встановлення режиму входу.
- digitalWrite(pin, HIGH); // активація внутрішнього підтягуючого резистора.

Виводи, налаштовані на вихід, переходять у низькоімпедансний стан і здатні забезпечувати вихідний струм до 40 мА. Слід враховувати, що такої потужності недостатньо для прямого керування силовими навантаженнями, як-от електродвигуни чи реле. Недотримання лімітів струму або коротке замикання на виходах може призвести до виходу з ладу транзисторів порту або повної поломки мікроконтролера.

*Функціонування аналогових входів.* Мікроконтролери сімейства ATmega, на базі яких побудовано платформи Arduino, оснащені вбудованим шестиканальним аналого-цифровим перетворювачем (АЦП). Даний модуль

має 10-бітну розрядність, що забезпечує дискретизацію вхідної напруги в діапазоні цифрових значень від 0 до 1023 одиниць.

Особливістю аналогових входів є їхня здатність працювати у режимі стандартних цифрових портів введення-виведення. У такій конфігурації вони отримують порядкову нумерацію від 14 до 19. Приклад програмного звернення до аналогового входу як до цифрового:

- pinMode(14, OUTPUT); // встановлення режиму виходу для піна A0.
- digitalWrite(14, HIGH); // подача високого логічного рівня.

Варто враховувати, що після використання піна як цифрового виходу, функція analogRead() може повертати помилкові дані. Для коректного зняття аналогових показників необхідно попередньо перевести порт у відповідний режим роботи.

*Функціонування широтно-імпульсної модуляції (ШІМ).* Широтно-імпульсна модуляція (ШІМ) є методом керування потужністю сигналу шляхом зміни тривалості імпульсів за умови незмінної частоти їх слідування. Завдяки чергуванню високого (5 В) та низького (0 В) логічних рівнів формується середнє значення вихідної напруги. Для імітації аналогового сигналу в Arduino використовується функція analogWrite(). Програмне значення в діапазоні від 0 до 255 визначає коефіцієнт заповнення (робочий цикл):

- 255 (100%): постійний високий рівень напруги (5 В);
- 127 (50%): генерація імпульсів, тривалість яких дорівнює паузі між ними, що еквівалентно середній напрузі 2,5 В;
- 0 (0%): відсутність сигналу (0 В).

*Архітектура пам’яті мікроконтролерів сімейства ATmega.* Апаратна архітектура чипів ATmega (моделі 168, 328, 1280 та 2560), що лежать в основі плат Arduino, базується на використанні трьох типів запам'ятовуючих пристроїв:

- Flash-пам’ять: Призначена для розміщення виконуваного коду програми (скетчу). Вона є енергонезалежною, тобто зберігає дані після вимкнення живлення.

- SRAM (Статична оперативна пам’ять): Використовується для тимчасового зберігання змінних та виконання обчислень під час роботи контролера. Це енергозалежний тип пам’яті.
- EEPROM: Спеціалізована енергонезалежна область пам’яті для тривалого зберігання констант або налаштувань, які мають бути доступні після перезавантаження системи.

*Таблиця 3.2. Порівняльні характеристики об’ємів пам’яті*

| Мікроконтролер | Flash-пам’ять (загальна) | SRAM (ОЗП) | EEPROM |
|---|---|---|---|
| ATmega328 | 32 КБ (з них 2 КБ під завантажувач) | 2 КБ | 1024 байт |
| ATmega1280 | 128 КБ (з них 2 КБ під завантажувач) | 8 КБ | 4096 байт |
| ATmega2560 | 256 КБ (з них 2 КБ під завантажувач) | 16 КБ | 8 КБ (8192 байт) |

*Опис мікроконтролера AVR ATmega328P*. МК ATmega328P – це високоефективне 8-бітне рішення на базі архітектури AVR, що поєднує низьке енергоспоживання з потужними обчислювальними можливостями. Завдяки впровадженню прогресивної RISC-архітектури, більшість із 131 інструкції виконується лише за один тактовий цикл. Основні обчислювальні характеристики:

- Продуктивність: Досягає 20 MIPS при робочій частоті 20 МГц.
- Наявність 32 робочих регістрів загального призначення підвищує швидкість обробки даних.
- Режим роботи: Повністю статична архітектура.

Конфігурація пам’яті:

- FLASH (пам’ять програм): 32 КБ із можливістю внутрішньосистемного перепрограмування (ресурс – до 10 000 циклів).
- EEPROM (енергонезалежна пам’ять): 1 КБ для зберігання даних із високою надійністю (до 100 000 циклів перезапису).
- SRAM (оперативна пам'ять): Внутрішній статичний ОЗП об’ємом 2 КБ.

Периферійні пристрої:

- Таймери та лічильники: Два 8-бітних та один 16-бітний модулі з функціями порівняння, захоплення та гнучким налаштуванням подільників частоти.
- ШІМ (PWM): Шість каналів для формування широтно-імпульсної модуляції.
- Аналогові модулі: 10-бітовий 8-канальний аналого-цифровий перетворювач (АЦП) та інтегрований аналоговий компаратор.
- Підтримка протоколів USART, SPI (режими Master/Slave) та I2C.

Системні функції та енергозбереження:

- Керування живленням: Шість інтелектуальних режимів енергозбереження (зокрема Idle, Power-save, Power-down).
- Програмований сторожовий таймер (Watchdog), моніторинг напруги живлення (Brown-out Detection).
- Вбудований RC-генератор із калібруванням та підтримка зовнішніх джерел переривань.
- Лінії вводу-виводу: До 32 програмованих портів I/O.
- Живлення: Робочий діапазон напруги від 2,7 В до 5,5 В.
- Постачається у форматах PDIP (28 виводів) або TQFP/QFN (32 виводи).

***Огляд цифрового термодатчика DS18B20.*** Сучасний ринок електронних компонентів насичений різноманітними рішеннями для моніторингу температури. Вибір конкретної моделі базується на таких критичних параметрах, як:

- точність вимірювання та робочий температурний діапазон.
- рівень напруги живлення, габаритні розміри та енергоспоживання.
- протоколи передачі даних для зв’язку з мікроконтролерами та методи перетворення сигналу.

Одним із найбільш затребуваних у цій галузі є модель DS18B20, розроблена компанією Dallas Semiconductor. Цей пристрій належить до класу прецизійних цифрових термометрів. Його ключовою особливістю є

можливість програмного налаштування роздільної здатності, що дозволяє гнучко балансувати між швидкістю обробки даних та точністю отриманих показників.

*Технічні параметри та функціональні можливості DS18B20.* Цифровий сенсор DS18B20 має наступні експлуатаційні властивості:

- Обмін даними з центральним контролером здійснюється через однопровідну шину 1-Wire, що мінімізує кількість необхідних портів введення-виведення.
- Кожен екземпляр має у ROM-пам’яті унікальний 64-бітний ідентифікатор. Це дозволяє створювати мережі з багатьох датчиків, підключених до однієї лінії, з можливістю звернення до кожного окремо.
- Пристрій працює в широкому діапазоні напруги (3,0 – 5,5 В), що забезпечує повну сумісність як зі стандартною 5-вольтовою логікою, так і з сучасними 3,3-вольтовими мікропроцесорними системами.
- Робочий температурний діапазон: від -55°C до +125°C.
- Похибка вимірювань: не перевищує ±0,5°C (у межах від -10°C до +85°C).
- Гнучкість налаштувань: користувач може самостійно обирати розрядність перетворення від 9 до 12 біт.
- Датчик має незалежну пам’ять для зберігання порогових значень температури (верхнього та нижнього рівнів). При виході за ці межі пристрій генерує сигнал тривоги, що важливо для систем терморегулювання.

Модель DS18B20 є програмно ідентичною до датчика DS1822, що спрощує перехід між компонентами. Завдяки своїй надійності, ці сенсори масово інтегруються в: промислові автоматизовані системи контролю, побутову техніку та розумні будинки, системи клімат-контролю та термостатичні вузли, обладнання для моніторингу навколишнього середовища.

Виробник пропонує давач DS18B20 у декількох варіантах виконання: компактних корпусах для поверхневого монтажу (SOIC та μSOP), а також у

класичному трьохвивідному корпусі TO-92, який нагадує звичайний транзистор (Рис.3.7). Ключові експлуатаційні переваги пристрою:

- Взаємодія з мікроконтролером реалізована через протокол 1-Wire. Унікальною рисою сенсора є підтримка режиму «паразитного живлення». Це дозволяє пристрою отримувати необхідну енергію безпосередньо з лінії передачі даних, фактично зводячи кількість необхідних дротів до мінімуму.
- Наявність індивідуального 64-бітного серійного номера у кожного датчика дозволяє об'єднувати велику кількість сенсорів у єдину мережу.
- Завдяки вищезгаданим особливостям, розгортання системи температурного моніторингу на локальному об'єкті вимагає використання лише одного сигнального провідника для опитування всіх вузлів.

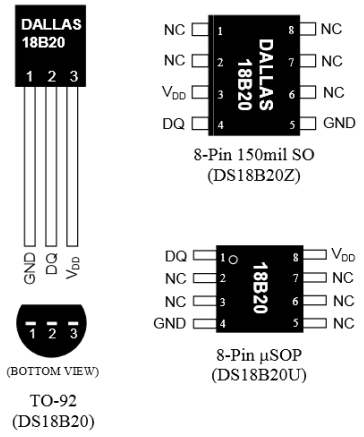



Рис. 3.7. Термодатчик DS18B20 у корпусах TO-92, SOIC iuSOP

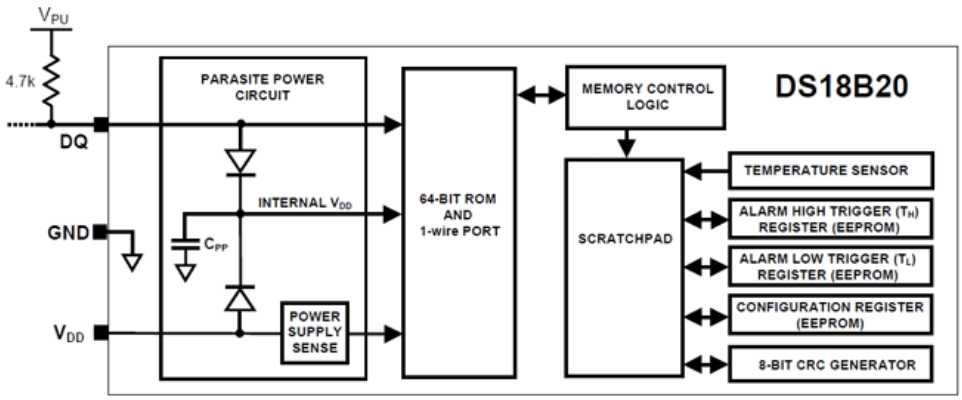



Рис. 3.8. Блок-схема термодатчика DS18B20

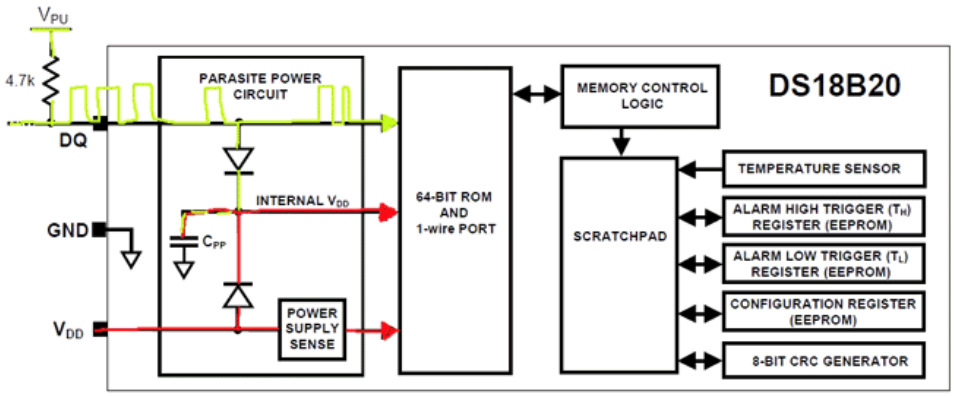



Рис. 3.9. Система живлення термодатчика DS18B20

Конструкція датчика дозволяє використовувати спрощену двопровідну схему підключення завдяки режиму “паразитного живлення”. Цей метод базується на накопиченні енергії безпосередньо з лінії передачі даних.

Коли на шині DQ утримується високий логічний рівень, струм проходить через підтягуючий резистор і внутрішній діод датчика, заряджаючи інтегрований конденсатор Cpp. У цей момент внутрішня шина живлення пристрою (Internal Vdd) отримує енергію від лінії даних. У моменти, коли лінія DQ переходить у низький стан (логічний нуль), датчик перемикається на споживання енергії, що була заздалегідь накопичена в конденсаторі.

Використання такого способу живлення накладає певні вимоги до часових інтервалів протоколу. Тривале перебування лінії DQ в низькому стані призводить до критичного розряду конденсатора та зупинки роботи сенсора. Тому в періоди бездіяльності на шині обов’язково має підтримуватися високий рівень.

Під час виконання найбільш енерговитратних операцій – таких як безпосереднє вимірювання температури або запис даних у незалежну пам’ять (EEPROM) – сила струму на внутрішній лінії Internal Vdd зростає до 1,5 мА. Потужності внутрішнього конденсатора недостатньо для таких пікових навантажень, а падіння напруги на стандартному підтягуючому резисторі може спричинити збої.

Для стабільної роботи в таких режимах необхідно впроваджувати схему “сильної підтяжки” (Strong Pullup). Вона передбачає використання

додаткового транзисторного ключа, який напряму з'єднує лінію DQ з джерелом живлення в моменти максимального навантаження, забезпечуючи необхідну потужність для коректного завершення операцій (Рис.3.10).

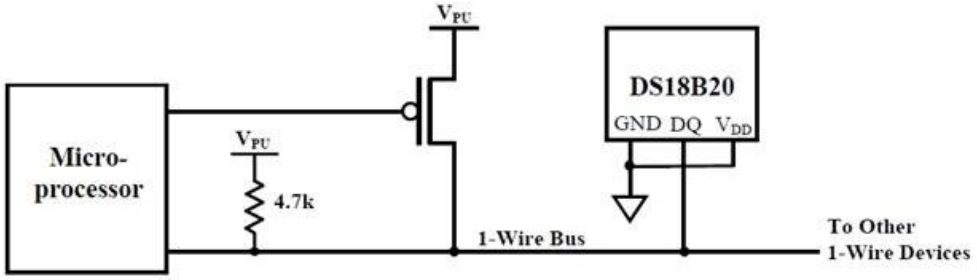



Рис. 3.10. Схема підтяжки для термодатчика DS18B20

У випадках, коли використовується режим паразитного живлення, алгоритм взаємодії з датчиком вимагає суворого дотримання часових інтервалів під час виконання команд Convert T [44h] (запуск вимірювання) та Copy Scratchpad [48h] (перезапис даних у пам'ять). Одразу після передачі коду команди (не пізніше ніж через 10 мкс), необхідно задіяти зовнішній MOSFET-транзистор. Це дозволить жорстко з'єднати лінію DQ з джерелом живлення, компенсуючи падіння напруги. Протягом усього періоду виконання операції – часу перетворення (T_{conv}) або циклу запису в EEPROM (близько 10 мс) – активний рівень підтяжки має залишатися незмінним. Важливо, щоб під час роботи потужного ключа на лінії DQ не відбувалося жодних інших маніпуляцій чи передачі сигналів.

Варто зауважити, що при використанні стандартної трьохпровідної схеми (з підключенням виводу VDD до окремого джерела живлення) дані маніпуляції з лінією DQ не є обов'язковими, що значно спрощує програмну реалізацію драйвера (Рис.3.11).

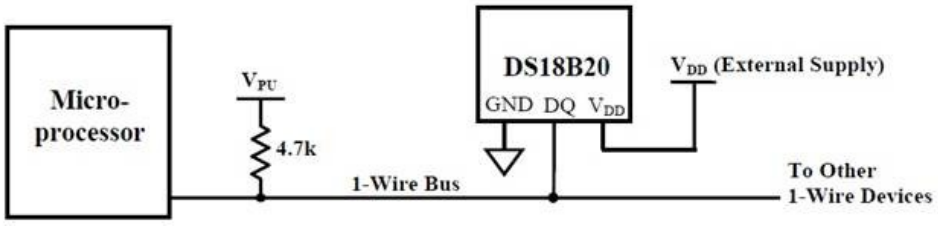



Рис. 3.11. Стандартне живлення термодатчика DS18B20

При проєктуванні системи важливо критично підійти до вибору способу живлення сенсора. Попри зручність двопровідної схеми, режим паразитного живлення має суттєве температурне обмеження.

Не варто застосовувати живлення через лінію даних, якщо передбачається експлуатація датчика в середовищі з температурою понад 100 °C. При досягненні високих температурних показників у напівпровідниковій структурі пристрою різко зростають струми витоку. Це може спричинити збої в роботі, помилки при зчитуванні даних або повну нестабільність системи. Для високотемпературних вимірювань розробникам рекомендується переходити на стандартну схему з виділеним зовнішнім джерелом живлення. Це гарантує стабільну роботу сенсора та точність отриманих результатів у всьому паспортному діапазоні. Функціональна схема термодатчика зображена на Рис.3.12.

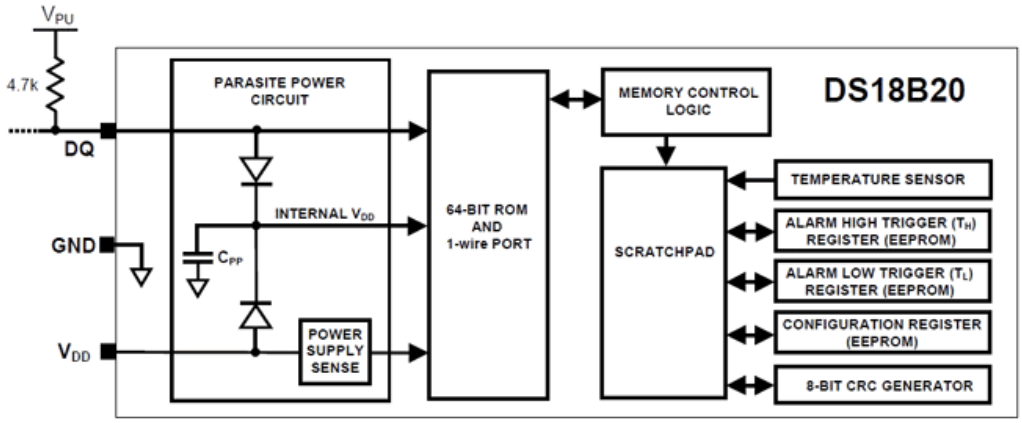



Рис. 3.12. Функціональна схема термодатчика DS18B20

*Алгоритм взаємодії та структура команд DS18B20.* Процес обміну даними з пристроєм базується на чіткій ієрархії сигналів. Будь-яка дія починається з циклу ініціалізації, після чого слідує етап адресації. Команди роботи з ROM повинні передаватися після сигналу скидання (Reset) та відповіді датчика (Presence pulse). Вони дозволяють системі ідентифікувати конкретний пристрій на шині або виконати пошук усіх доступних адрес. Кожна інструкція має фіксовану довжину – 8 біт (1 байт). Тільки після успішного вибору датчика (адресації) контролер може надсилати функціональні команди, безпосередньо пов’язані з вимірюванням

температури або зчитуванням внутрішньої пам’яті. Команди термодатчика наведено в Табл.3.3.

*Таблиця 3.3. Перелік та функціонал ROM-команд термодатчика DS18B20*

| Команда | HEX | Дія ведучого пристрою (МК) |
|---|---|---|
| Search ROM | F0h | Запуск ітераційного процесу визначення адрес пристроїв. |
| Read ROM | 33h | Прийом 8-ми байтів даних (Family Code, Serial Number, CRC). |
| Match ROM | 55h | Передача 8-ми байтів адреси цільового датчика. |
| Skip ROM | CCh | Перехід безпосередньо до передачі функціональної команди. |
| Alarm Search | ECh | Опитування шини для виявлення “тривожних” станів. |

*Функціональні команди термодатчика DS18B20.* Дана група команд призначена для безпосереднього керування робочими процесами датчика, такими як активація циклу температурного вимірювання або перенесення даних між реєстрами пам’яті. Специфікація пристрою передбачає шість основних функціональних команд. Кожна інструкція передається у вигляді 8-бітного коду (одного байта). За допомогою цих команд контролер ініціює перетворення аналогового сигналу в цифровий код, здійснює копіювання проміжного блокнота (Scratchpad) у незалежну пам’ять (EEPROM) та виконує інші операції з даними.

*Таблиця 3.4. Функціональні команди термодатчика DS18B20*

| Назва команди | Код (HEX) | Опис функціонального призначення |
|---|---|---|
| Convert T | 44h | Запуск одного циклу перетворення температури. Результат зберігається в оперативній пам’яті. |
| Write Scratchpad | 4Eh | Запис 3-х байтів даних в оперативну пам’ять (порогові значення $T_H$, $T_L$ та регістр конфігурації). |
| Read Scratchpad | BEh | Послідовне зчитування вмісту оперативної пам’яті (9 байтів, включаючи дані температури та CRC). |
| Copy Scratchpad | 48h | Копіювання значень $T_H$, $T_L$ та конфігурації з оперативної пам’яті в енергонезалежну пам'ять EEPROM. |
| Recall E2 | B8h | Відновлення значень $T_H$, $T_L$ та конфігурації з EEPROM в оперативну пам’ять (відбувається автоматично при ввімкненні). |
| Read Power Supply | B4h | Визначення типу живлення: від зовнішнього джерела (“1”) чи паразитного (“0”). |

*Підключення датчика DS18B20 з використанням порта 1-Wire.* Розроблений наприкінці 1990-х років компанією Dallas Semiconductor, однопровідний інтерфейс 1-Wire став універсальним стандартом передачі даних. Сьогодні асортимент Dallas Semiconductor включає величезну кількість однопровідних пристроїв для реалізації найрізноманітніших мережевих

архітектур. Популярність даного інтерфейсу обумовлена низкою технологічних переваг: простий і надійний спосіб ідентифікації кожного вузла в мережі; лаконічний алгоритм обміну даними, що не потребує великих обчислювальних потужностей; спрощена топологія лінії зв'язку, що значно полегшує монтаж; низький рівень споживання енергії підключеними компонентами; можливість легкого додавання або видалення пристроїв без переналаштування всієї мережі; здатність працювати на лініях зв'язку значної довжини.

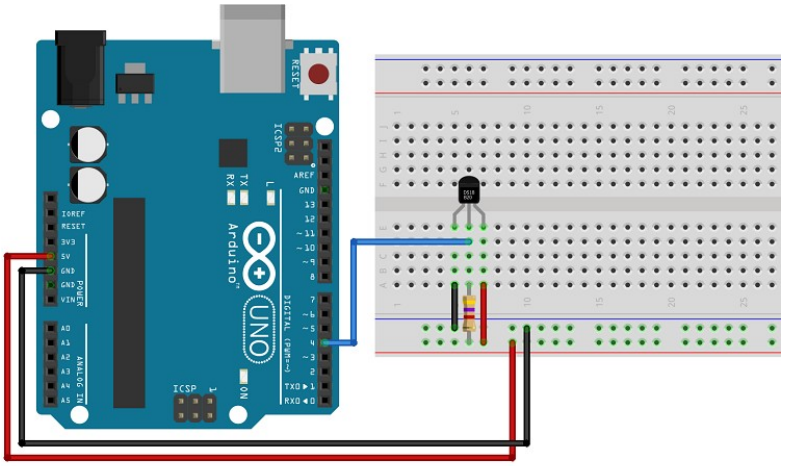


Рис. 3.13. Схема підключення термодавача DS18B20 до плати Arduino

***Опис аналогового сенсора pH SEN0161.*** Аналоговий сенсор pH серії SEN0161 від компанії DFRobot – це популярне та надійне інженерне рішення для вимірювання водневого показника (pH) середовища у проектах на базі мікроконтролерів. Межі вимірювання датчика pH SEN0161 від 0 до 14 де 7 – означає нейтральне середовище (не кислотне і не лужне ). Виміри pH – менші за 7 властиве кислому розчину, а більше 7 – лужному. Нормальним вважається розчин з pH  в межах 6,5-8,5;

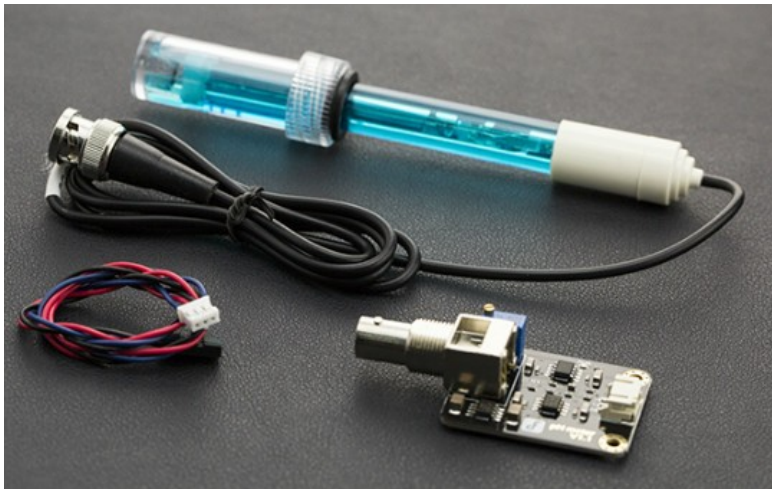


Рис. 3.14. Аналоговий датчик pH SEN0161

*Характеристики датчика pH SEN0161:* Для стабільної роботи пристрою необхідне джерело постійного струму з напругою 5 В (DC). Модуль здатен фіксувати рівень кислотності в межах від 0 до 14 pH. Похибка вимірювань становить не більше ±0,1 pH (за умови калібрування та експлуатації при температурі 25 °C). Пристрій коректно функціонує в температурному режимі від 0 до 60 °C. Час відгуку системи на зміну складу середовища не перевищує 1 секунди (1000 мс).

Модуль підтримує два типи вихідних сигналів для зручної інтеграції з мікроконтролерами:

1. Аналоговий: Формує напругу в діапазоні 0 – 4,5 В, що пропорційна рівню pH.
2. Цифровий: Генерує логічні рівні (високий/низький). Поріг спрацювання дискретного сигналу регулюється за допомогою вбудованого прецизійного потенціометра.

Найвища точність результатів досягається при температурі аналізованого розчину близько 25 °C. Оскільки результати вимірювань термічно залежні, для отримання прецизійних даних у широкому діапазоні температур рекомендується використовувати алгоритми програмної термокомпенсації, або підтримувати температуру розчину в межах 25 °C.

Для калібрування сенсорів кислотності, є тестові набори розчинів Рис.3.15. на яких є позначки кислотності (наприклад pH=7,00, або pH=4,00).


Рис. 3.15. Тестовий набір розчинів для калібрування сенсора pH SEN0161

Для отримання достовірних вимірювань сенсор pH SEN0161 треба колібрувати не рідше одного разу на шість місяців. Електрод pH сенсора має при температурі (25 °C) характеристики наведені у Табл.3.5.

*Таблиця 3.5. Залежність потенціалу електрода та вихідної напруги модуля SEN0161 від pH (при 25°C)*

| Значення pH | Характер середовища | EPC електрода (мВ)(на роз'ємі BNC) | Вихідна напруга модуля ($V_{out}$, В)(на АЦП Arduino) | Значення АЦП Arduino(10-біт, 0−1023) |
|---|---|---|---|---|
| 0 | Сильнокисле | +414,12 | 4,14 | 848 |
| 2 | Кисле | +295,80 | 3,68 | 753 |
| 4 | Слабокисле | +177,48 | 3,22 | 659 |
| 5 | Слабокисле | +118,32 | 2,98 | 611 |
| 6 | Слабокисле | +59,16 | 2,74 | 562 |
| 7 | Нейтральне | 0,00 | 2,50 | 512 |
| 8 | Слаболужне | -59,16 | 2,26 | 463 |
| 9 | Слаболужне | -118,32 | 2,02 | 414 |
| 10 | Слаболужне | -177,48 | 1,78 | 365 |
| 12 | Лужне | -295,80 | 1,32 | 270 |
| 14 | Сильнолужне | -414,12 | 0,86 | 176 |

*Порядок підключення та калібрування модуля pH з сенсором SEN0161.* Монтаж системи виконується згідно зі схемою підключення (Рис.3.16). Процес включає такі кроки:

1. Електрод підключається до вимірювальної плати за допомогою стандартного роз'єму BNC.
2. Вихідний інтерфейс плати pH-метра з'єднується з одним із аналогових входів (ADC) платформи Arduino.

3. У пам’ять мікроконтролера завантажується відповідний скетч для зчитування даних.

Для отримання достовірних результатів сенсор потребує обов’язкового налаштування перед кожним робочим циклом із використанням еталонних буферних розчинів.

Етап 1: Програмне корегування зміщення (Offset)

1. Занурюємо електрод у розчин із нейтральним показником pH 7,00.
2. За допомогою монітора послідовного порту (Serial Monitor) в Arduino IDE фіксуємо поточні вимірювання.
3. При виявленні відхилення (наприклад, замість 7,00 пристрій видає 6,88) розраховуємо різницю (7,00-6,88=0,12).
4. Вносимо отримане значення (0,12) у відповідну змінну (калібрувальну константу) у вихідному коді програми.

Етап 2: Налаштування підсилення для кислотних середовищ

1. Занурюємо сенсор у розчин із рівнем pH 4,00.
2. Витримуємо паузу близько 60 секунд для стабілізації показників.
3. Обертанням підстроювального потенціометра на платі добиваємося відображення значення 4,00 у вікні РКД.

Етап 3: Робота з лужними розчинами. Завдяки лінійності характеристик pH-електрода, після налаштування в кислому середовищі можна переходити до лужного калібрування:

1. Використовуємо еталон із показником pH 9,18.
2. Аналогічно коригуємо підсилення потенціометром до моменту стабілізації значень на рівні 9,18.

Критично важливою умовою точної роботи є чистота сенсора. Перед кожною зміною досліджуваного розчину або переходом до іншого еталона електрод необхідно ретельно очищувати, промиваючи його в дистильованій воді.

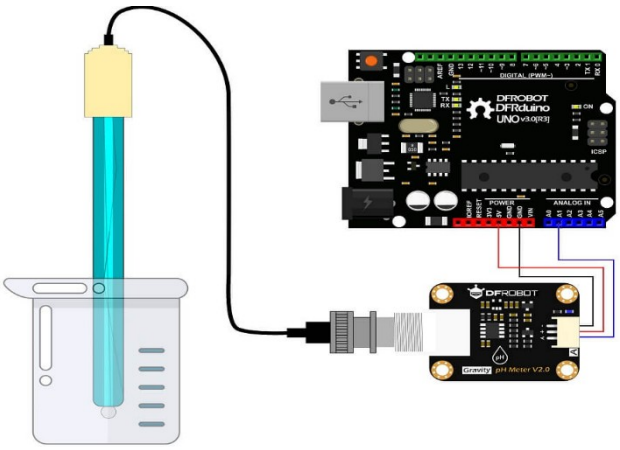


Рис. 3.16. Схема підключення датчика pH до плати Arduino

**Опис та вивід інформації на рідкокристалічний дисплей РКД (LCD).** Для виведення значного масиву даних у системах із жорсткими вимогами до енергоефективності оптимальним вибором є рідкокристалічні дисплеї (РКД). Це економічно вигідне рішення, що поєднує функціональність та низьке споживання струму. Типовий пристрій складається з керуючого контролера та рідкокристалічної матриці. Більшість сучасних моделей оснащені інтегрованою LED-підсвіткою, що робить можливим зчитування інформації навіть у повній темряві. Завдяки широкому діапазону робочих температур (від -20 °C до +70 °C), такі екрани стабільно працюють у складі портативних приладів, польового обладнання та бортових систем. Модуль дозволяє відображати текстову інформацію у два рядки, по 16 знаків у кожному. Формування кожного символу відбувається в межах окремої матриці розміром 5×8 пікселів. Для забезпечення чіткості та кращої читабельності тексту між сусідніми символами передбачено технологічні проміжки шириною в одну точку (Рис.3.17.).

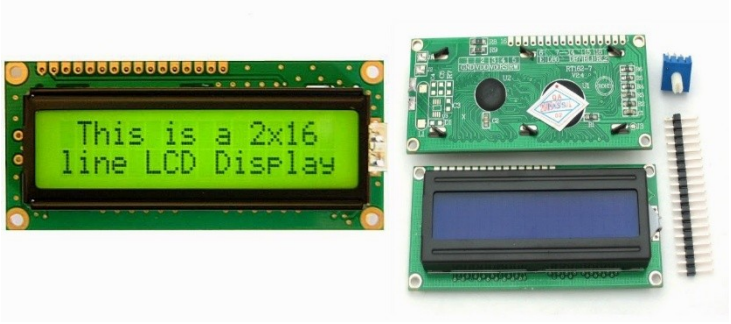



Рис. 3.17. Модуль РКД 2×16

*Архітектура пам'яті та функціональні можливості РКД-модуля.* Принцип роботи дисплея базується на відповідності між кодом символу та його відображенням, що реалізується через внутрішню оперативну пам'ять (ОЗП). Архітектура модуля включає логічні блоки керування панеллю та два типи пам'яті: генератор стандартних знаків та область пам'яті для символів, визначених користувачем.

*Технічні та експлуатаційні характеристики.* Вбудований знакогенератор має дві програмно доступні сторінки, що дозволяє працювати з кириличним та латинським алфавітами. Залежно від налаштувань під час ініціалізації, обмін даними може здійснюватися через 4-бітну або 8-бітну шину. Модуль підтримує повний цикл операцій із шиною даних:

- прийом та виконання команд керування;
- двобічний обмін даними (запис та зчитування інформації з внутрішньої ОЗП).

Система дозволяє створювати та зберігати в пам'яті до 8 власних унікальних символів або графічних зображень. Можливість налаштування курсору (доступно два типи), з опцією активації режиму миготіння.

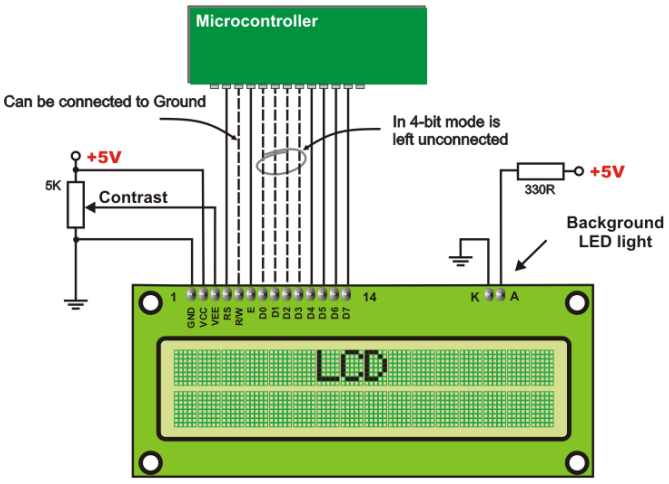



Рис. 3.18. Схема вмикання РКД 16×2 до МК

## 3.2. Проектування інформаційної системи вимірювання показника pH розчинів засобами САПР Proteus

Інформаційна система вимірювання показника pH повинна моніторити такі фізико-хімічні параметри розчинів як: температура, водневий показник (pH). На Рис.3.19 зображено запропоновану структуру інформаційної системи вимірювання показника ph розчинів.

Апаратне забезпечення системи складається з плати Arduino Uno на МК ATmega328P, давачів для вимірювання фізико-хімічних параметрів розчинів, зумера для подачі тривожних сигналів, алфавітно-цифрового рідкокристалічного дисплею для виводу останніх виміряних фізико-хімічних параметрів та повідомлень, виконавчих пристроїв для нагрівання та охолодження розчинів. Давачі утворюють систему збору фізико-хімічних параметрів розчину, які поступають до МК плати Arduino для подальшої обробки та передачі по послідовному інтерфейсі в ПК.

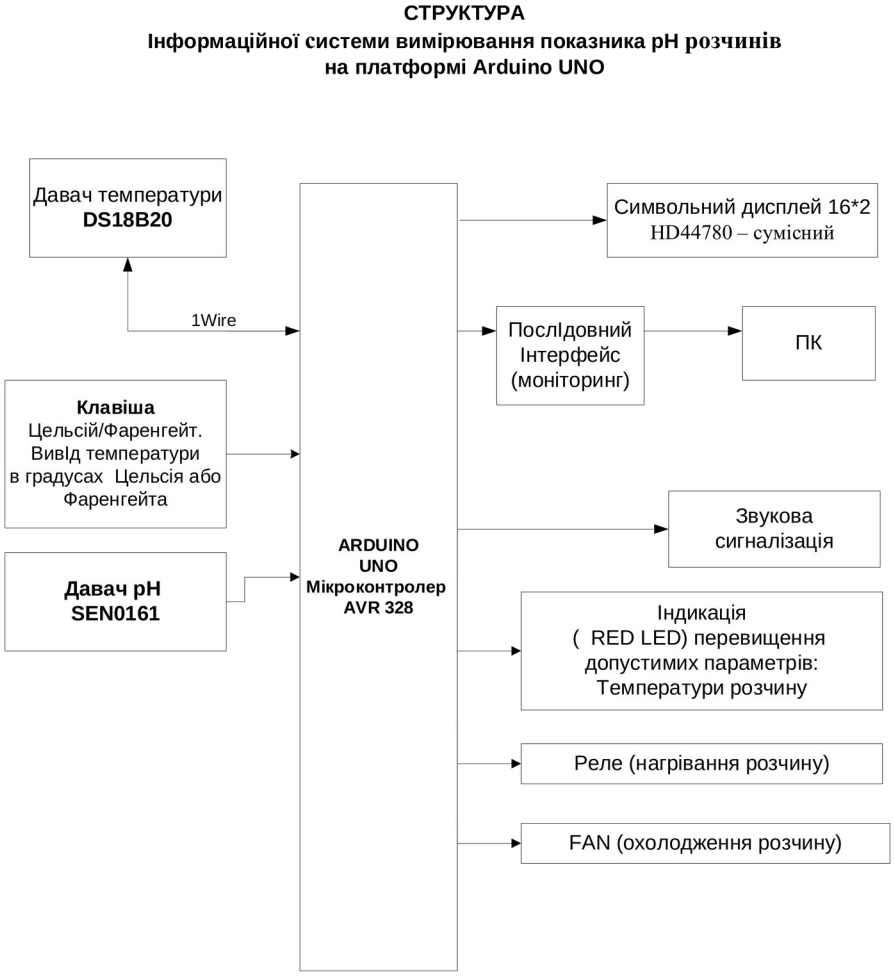



Рис. 3.19. Загальна структура інформаційної системи вимірювання показника pH розчинів

На Рис. 3.20 зображено спроектовану апаратну частину інформаційної системи вимірювання показника pH розчинів в системі автоматизованого проектування електронних схем та програмованих пристроїв Proteus. Давач pH змодельовано як потенціометр pH-(RV5). Вихід датчика pH підключено до входу A0 АЦП. На електричній схемі вивід даних DQ цифрового датчика температури DS18B20 підєднано до піна A5. РКД підключено до виводів 12 (RS), 11 (E), 5 (D4), 4 (D5), 3 (D6), 2 (D7). Клавіша зміни одиниць виводу параметрів температури “Цельсій/Фаренгейт” підключено до піна PD7. Пищик звукової сигналізації підключено до піна(10) PB2/SS/OC1B вихід ШІМ-генератора. Синій LED “Фаренгейт/Цельсій” підєднано до 13 піна (PB5) а червоний LED “Тривога” підєднано до піна 8 PB0 через обмежуючі резистори R1, R4. Для обміну по послідовному інтерфейсу використано піни 1 і 0 (Arduino UNO).

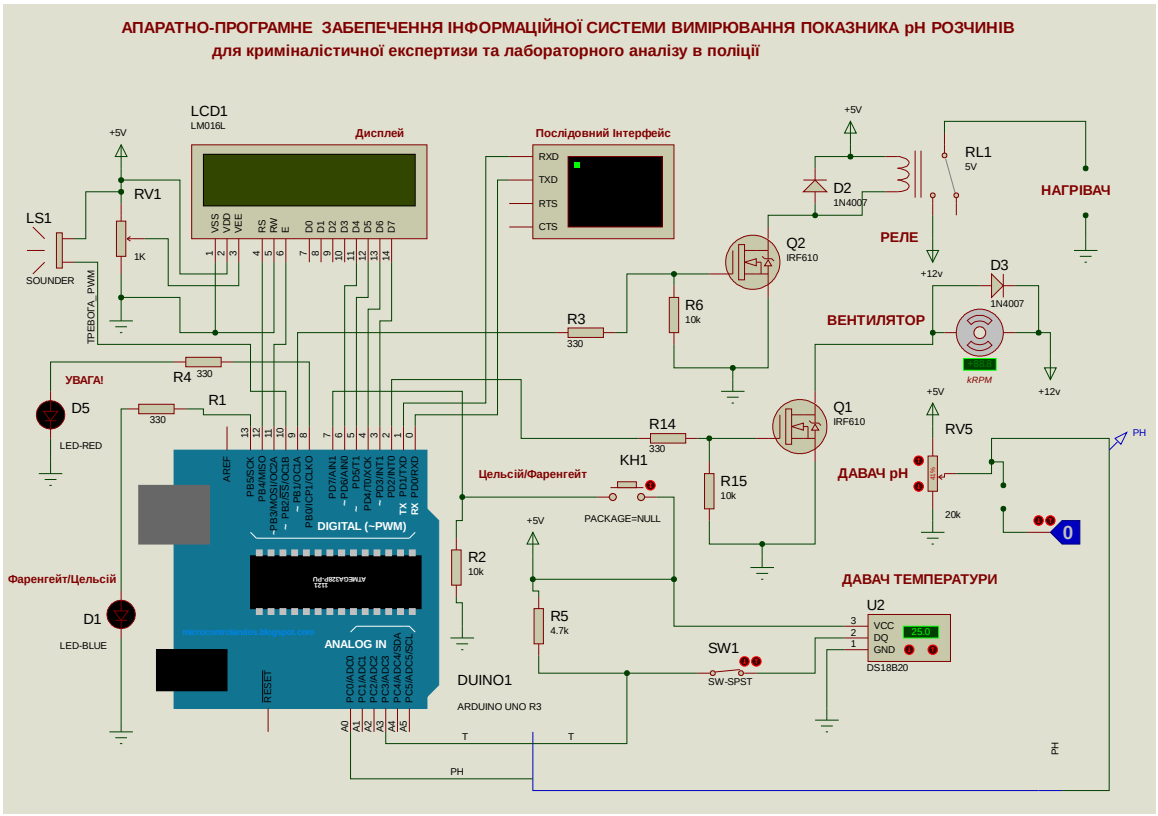



Рис. 3.20 Апаратне забезпечення інформаційної системи вимірювання показника ph розчинів спроектоване в САПР Proteus VSM

Виконавчі пристрої для підтримання температури розчину в межах 25°C±1°C підєднано до виводів 2 (охолодження розчину) і 9 (нагрівання).

# РОЗДІЛ 4
# ПРОГРАМНО-АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВИМІРЮВАННЯ ПОКАЗНИКА РН РОЗЧИНІВ

## 4.1. Алгоритм роботи інформаційної системи вимірювання показника pH розчинів

На Рис.4.1, 4.2 зображено блок-схему алгоритму роботи інформаційної системи вимірювання показника pH розчинів.

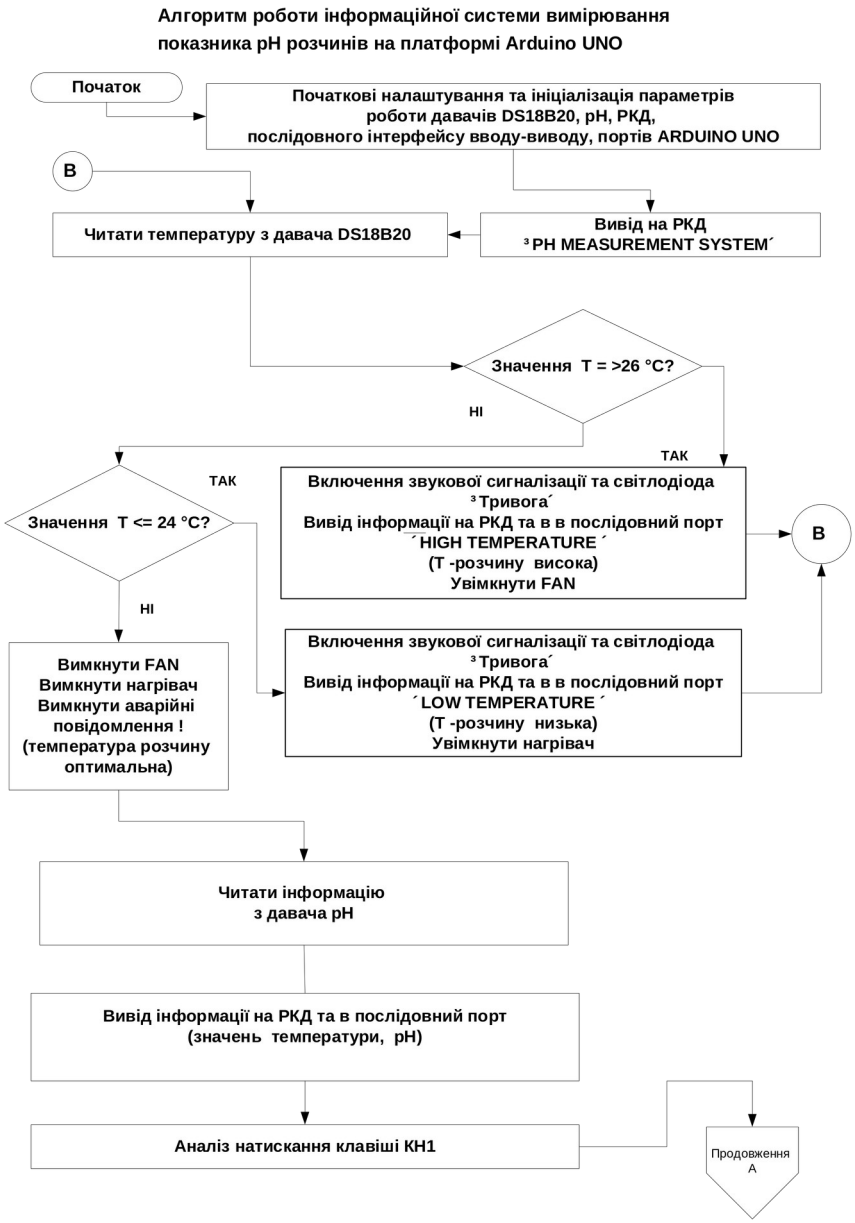
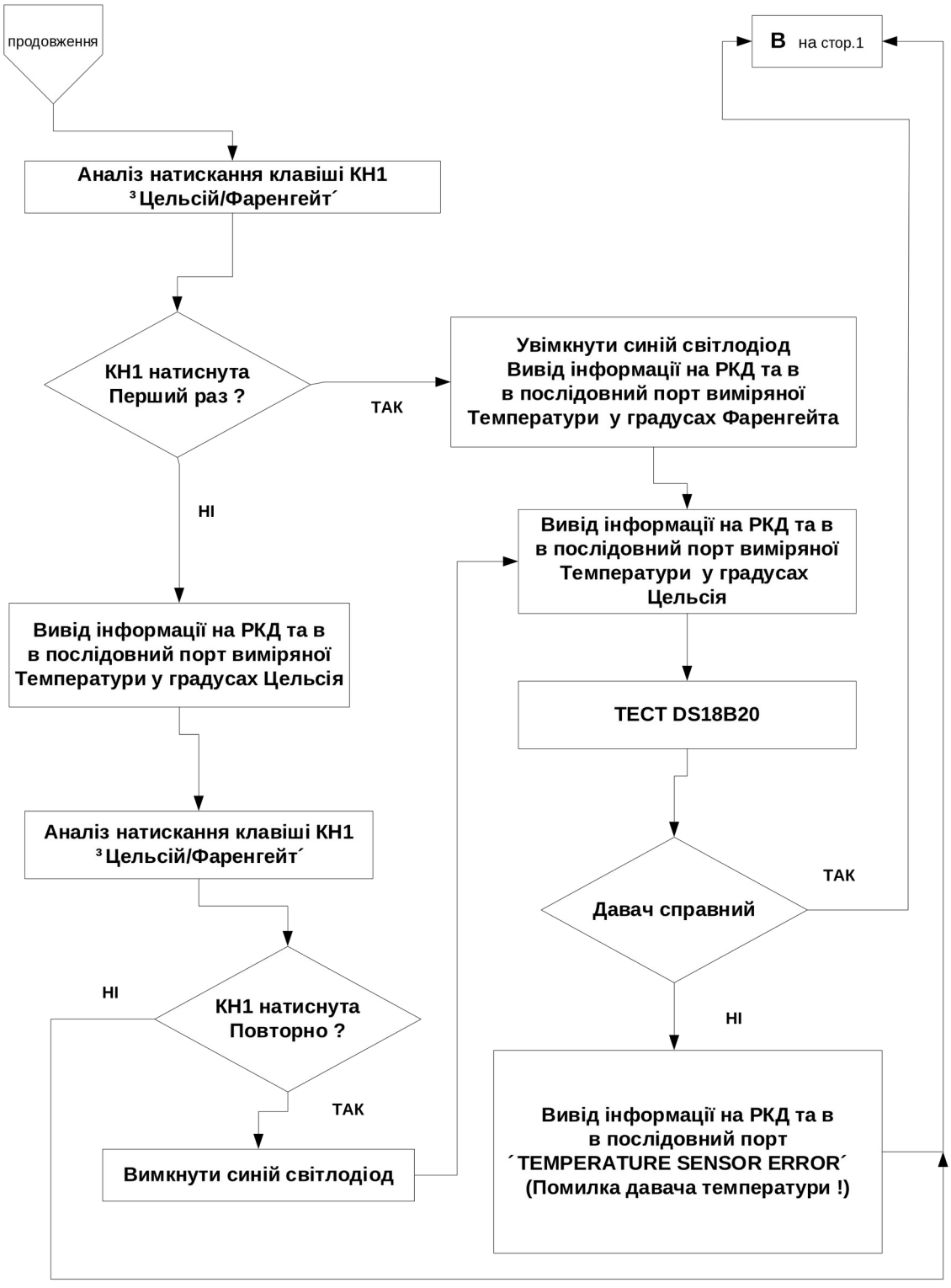



Рис. 4.1. Алгоритм роботи інформаційної системи вимірювання показника pH розчинів (початок)

**Алгоритм роботи інформаційної системи вимірювання показника pH розчинів на платформі Arduino UNO (продовження)**




Р

ис. 4.2. Алгоритм роботи інформаційної системи вимірювання показника pH розчинів

## 4.2. Розроблення програмних функцій для роботи з давачем температури та отримання даних з давачів

Для вимірювання рівня водневого показника pH розроблено наступні функції:

```
/* Функція вимірювання водневого показника pH */
float pH_measurement()
{
 int buff[10], temp;
float pHSensorVolt, pHSensorVal;
 unsigned long int pHSensorAvgValue;

 for (int i = 0; i < 10; i++)
 {
  buff[i] = analogRead(pHSensorAnalogInPin);
  delay(10);
 }
 for (int i = 0; i < 9; i++)
  for (int j = i+1; j < 10; j++)
   if (buff[i] > buff[j])
   {
    temp = buff[i];
    buff[i] = buff[j];
    buff[j] = temp;
   }
 pHSensorAvgValue = 0;
 for (int i = 2; i < 8; i++)
 pHSensorAvgValue += buff[i];
 pHSensorVolt = (float) pHSensorAvgValue * 5.0 / 1024 / 6;
 pHSensorVal = -5.70 * pHSensorVolt + 21.34;
 Serial.print("pH sensor voltage: ");
 Serial.print(pHSensorVolt, 2);
 Serial.println(" V");
 Serial.print("pH value: ");
 Serial.println(pHSensorVal, 2);
 return pHSensorVal;
}
```

Функція ініціалізація давача температури

```
  pinMode (A3, OUTPUT); digitalWrite (A3, HIGH);
  if (therm_init(1, -55, 125, THERM_9BIT_RES))
  {
    lcd.clear();
    lcd.print("T: error_");
    digitalWrite(RED, HIGH);
    pinMode(SPEAKER, OUTPUT);   // sets the digital pin as output
    iz=127; // duty= 50%
    analogWrite(SPEAKER, iz);   // speaker on,  speaker- pin.
     delay(10);
```

```
    while(1);
  }

/* Функція зчитування температури з давача */
uint8_t therm_read_temperature(uint8_t sensor_id, float *temp)
{
  uint8_t digit, decimal, resolution, sign;
  uint16_t meas, bit_mask[4]={0x0008, 0x000c, 0x000e, 0x000f};

  if (sensor_id) therm_dq=OUTDOOR_THERM;
    else therm_dq=INDOOR_THERM;
    if (therm_reset()) return 1;
  therm_write_byte(THERM_CMD_SKIPROM);
  therm_write_byte(THERM_CMD_CONVERTTEMP);
  while(!therm_read_bit());
  therm_reset();
  therm_write_byte(THERM_CMD_SKIPROM);
  therm_write_byte(THERM_CMD_RSCRATCHPAD);
  if (therm_read_spd()) return 1;
  therm_reset();
  resolution=(__ds18b20_scratch_pad.conf_register>>5) & 3;
  meas=__ds18b20_scratch_pad.temp_lsb;  // LSB
  meas|=((uint16_t)__ds18b20_scratch_pad.temp_msb) << 8; // MSB
  if (meas & 0x8000)
  {
    sign=1;  //відмічаємо мінусову температуру
    meas^=0xffff;  //перетворюємо в плюсову
    meas++;
  }
  else sign=0;
  //зберігаємо цілу і дробову частини температури
  digit=(uint8_t)(meas >> 4); //зберігаємо цілу частину
  decimal=(uint8_t)(meas & bit_mask[resolution]); //отримуємо дробову частину
  *temp=digit+decimal*0.0625;
  if (sign) *temp=-(*temp); //ставемо знак мінус, якщо мінусова температура
  return 0;
}

/* Функція генерування звуку */
void zvyk(void)
  {
    pinMode(SPEAKER, OUTPUT);   // sets the digital pin as output
    iz=127; // duty= 50%
    analogWrite(SPEAKER, iz);   // speaker on,  speaker- pin.
     delay(1000);
    pinMode(SPEAKER, INPUT);
  }
```

## 4.3. Програмна реалізація алгоритму роботи інформаційної системи вимірювання показника pH розчинів

Програма реалізує алгоритм роботи інформаційної системи вимірювання показника pH розчинів згідно Рис. 4.1. Рис. 4.2 приведена у додатку.

## 4.4. Моделювання роботи інформаційної системи вимірювання показника pH розчинів в Proteus ISIS

Скомпільована програма є hex-файлом прошивки для МК плати Arduino Uno. Створену модель системи і роботу прошивки перевіряємо в симуляторі Proteus ISIS. Proteus – це потужне ПЗ для моделюванняі та відлагодження складних пристроїв. PROTEUS VSM має можливість моделювання роботи програмованих пристроїв: апаратних платформ Arduino UNO мікроконтролерів, мікропроцесорів, DSP і інш. Пакет Proteus складається з двох частин: ISIS- програма синтезу та моделювання електронних схем і ARES- програма розробки друкованих плат.

На Рис. 4.3–4.5 зображено моделювання роботи спроектованої інформаційної системи вимірювання показника ph розчинів.

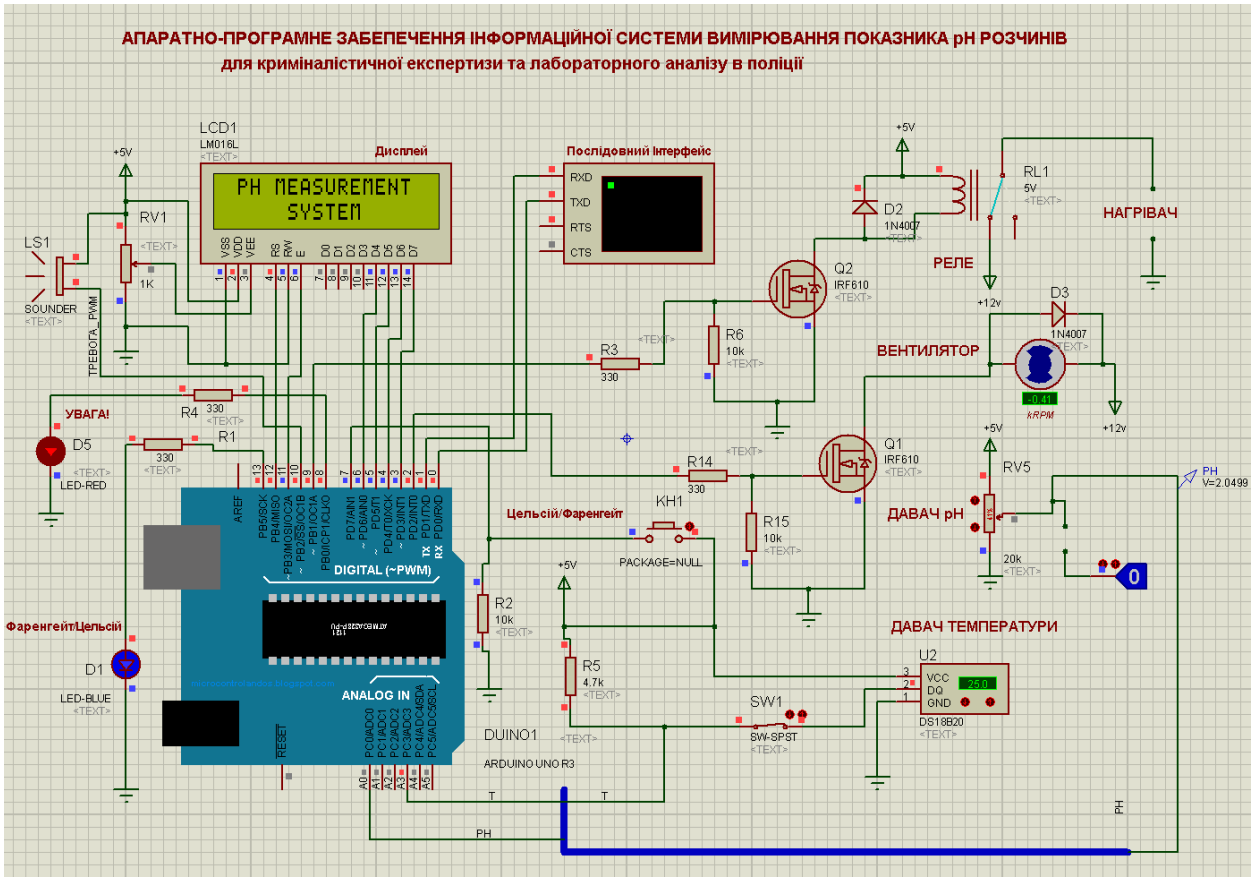


Рис. 4.3. Моделювання роботи інформаційної системи системи вимірювання показника pH розчинів в Proteus ISIS – ініціалізація

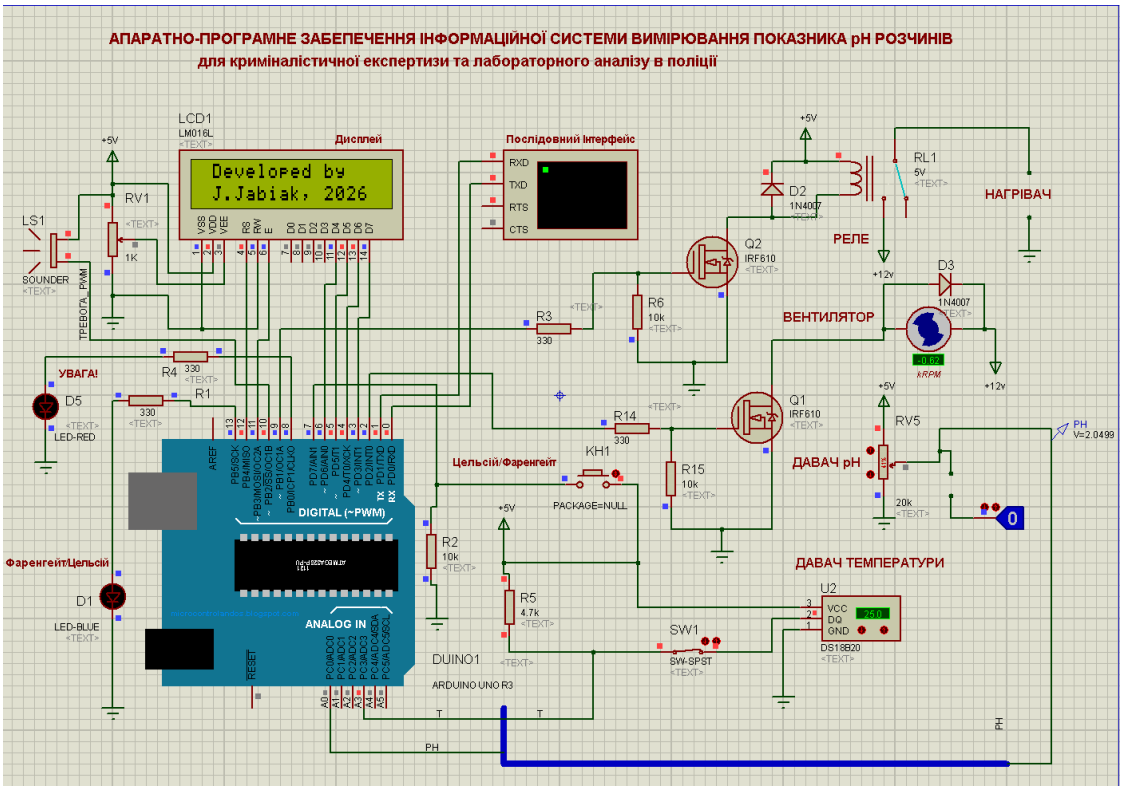



Рис. 4.4. Моделювання роботи системи вимірювання показника pH розчинів в Proteus ISIS – вивід інформації про розробника

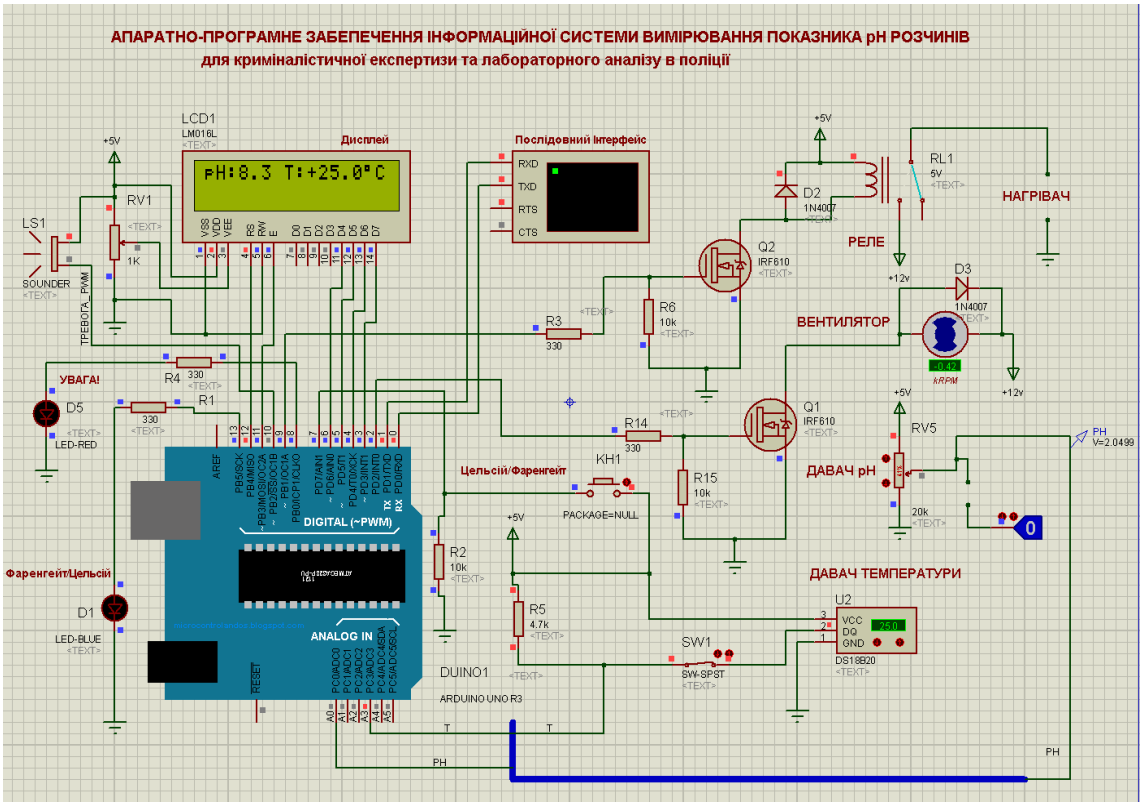


Рис. 4.5. Моделювання роботи системи вимірювання показника pH розчинів в Proteus ISIS – робочий режим

В робочому режимі Рис.4.5 на дисплеї відображається інформація про pH, та температуру розчину. При індикації температури в градусах Фаренгейта, вімкнений світлодіод D1(синій) “Фаренгейт/Цельсій” при вимкненому D1 вивід температури в °C. Одночасно з виводом на РКД інформація виводиться через послідовний інтерфейс в ПК (Рис.4.6.). Інформація на дисплеї поновлюється постійно щосекундно.

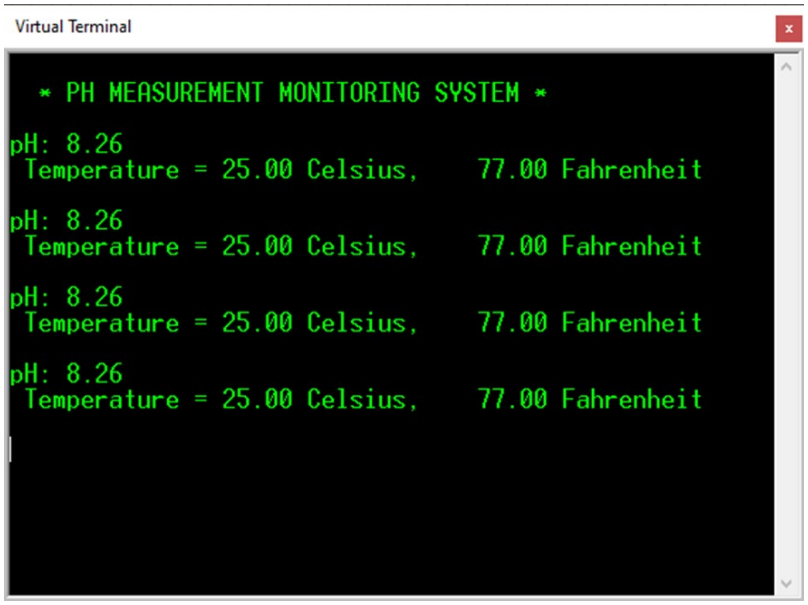



Рис. 4.6. Вивід інформації по послідовному інтерфейсу в ПК

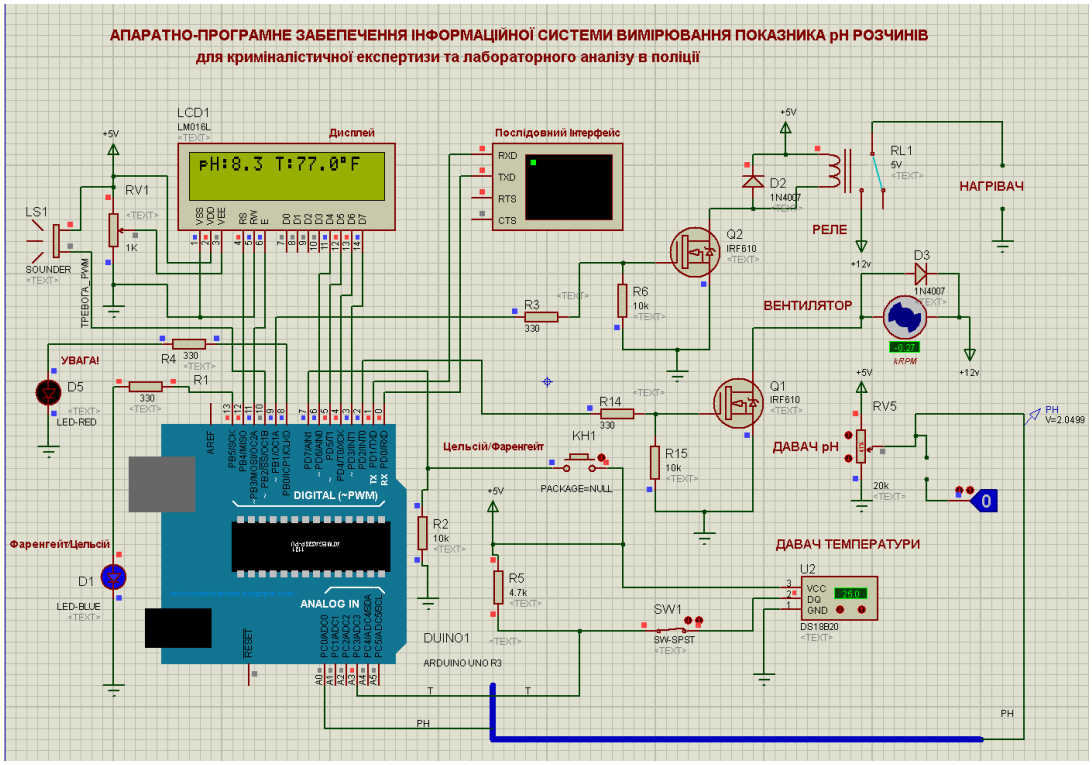



Рис. 4.7. Моделювання роботи в Proteus ISIS – робочий режим

Вивід температури у Фаренгейтах – синій LED=ON. При відхиленні температури розчину від норми спрацьовує світлова (червоний світлодіод – “Тривога”) та звукова сигналізація. В послідовний порт та на екран виводяться відповідні повідомлення:

- HIGH TEMPERATURE – температура розчину висока.
- LOW TEMPERATURE – температура розчину низька

При високій температурі розчину вмикаєть вентилятор охолодження. При низькій температурі розчину вмикаєть Реле для комутації нагрівального пристрою. В обох випадках вмикається тривожна сигалізація (світлодіод “УВАГА” і звучить пищик) (Рис.4.8, 4.9). Якщо температура розчину оптимальна (25°C), сигнали “Тривоги” – вимкнені. Робота давача pH симулюється потеціометром RV5.

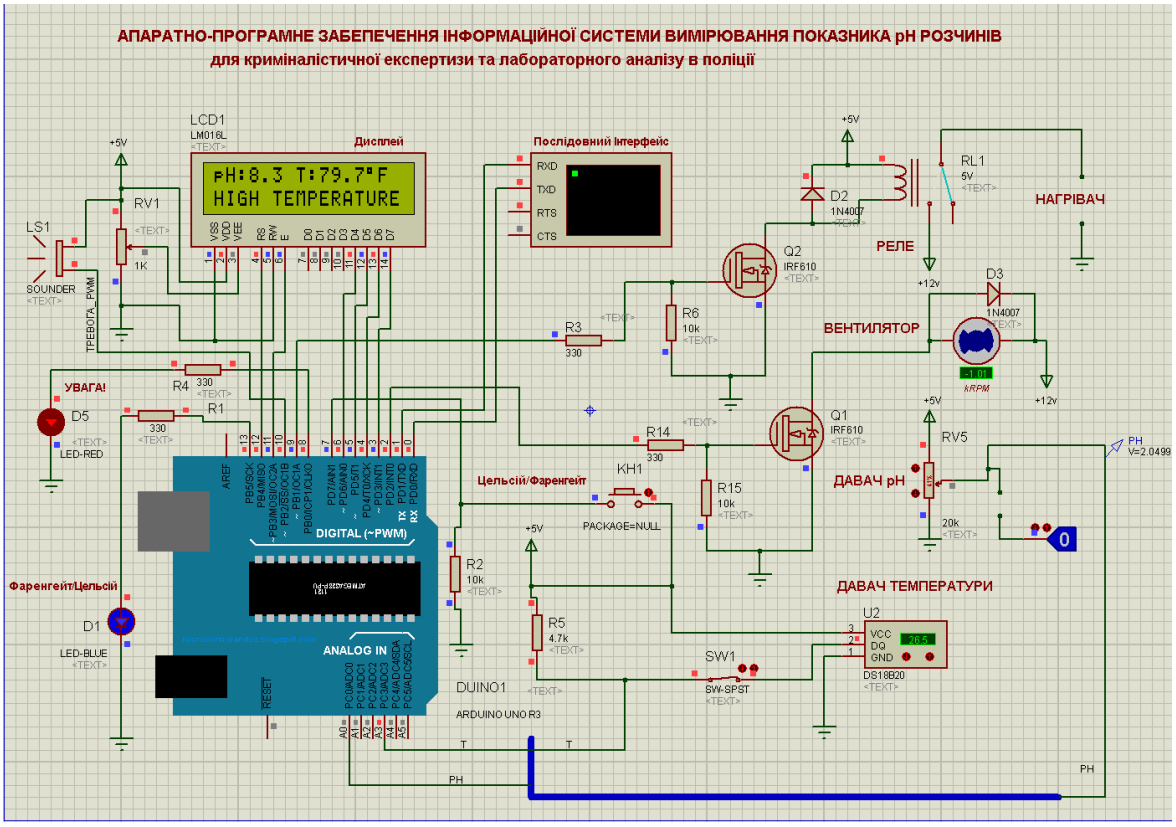



Рис. 4.8. Моделювання роботи в Proteus ISIS перевищення оптимальної температури розчину

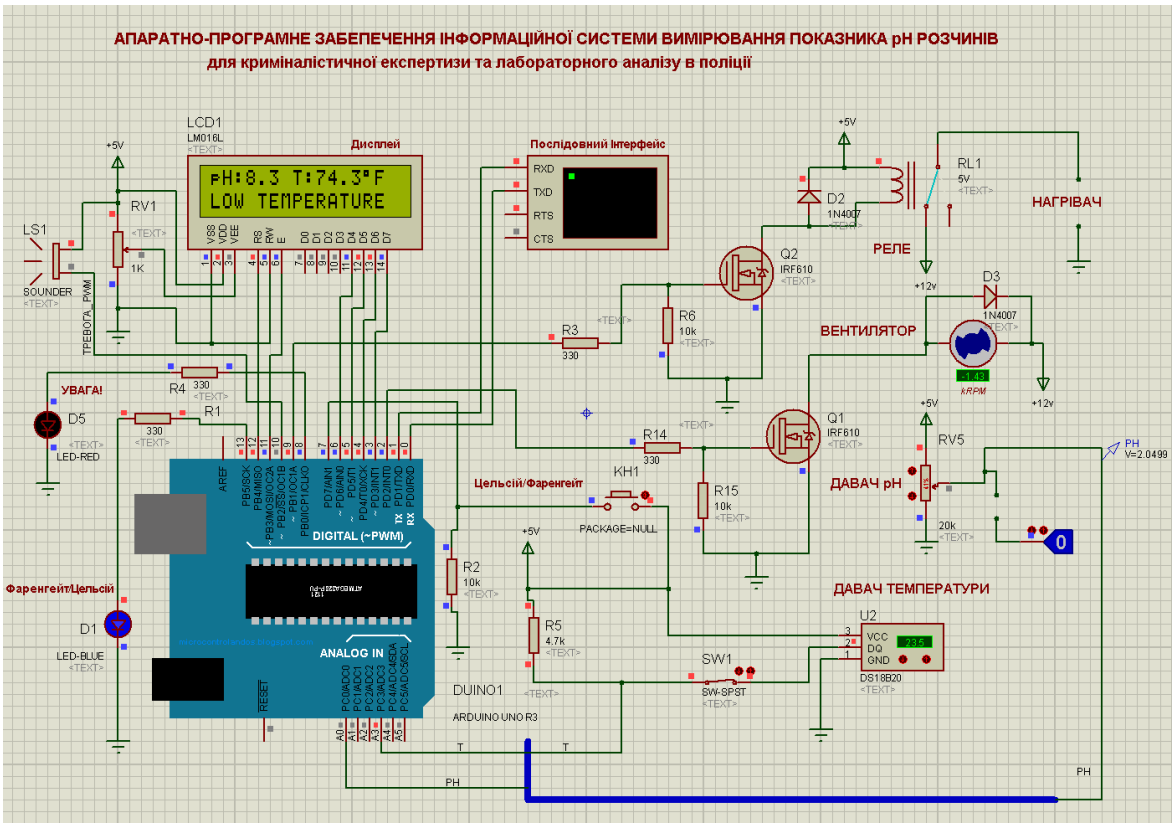



Рис. 4.9. Моделювання роботи в Proteus ISIS пониження оптимальної температури розчину

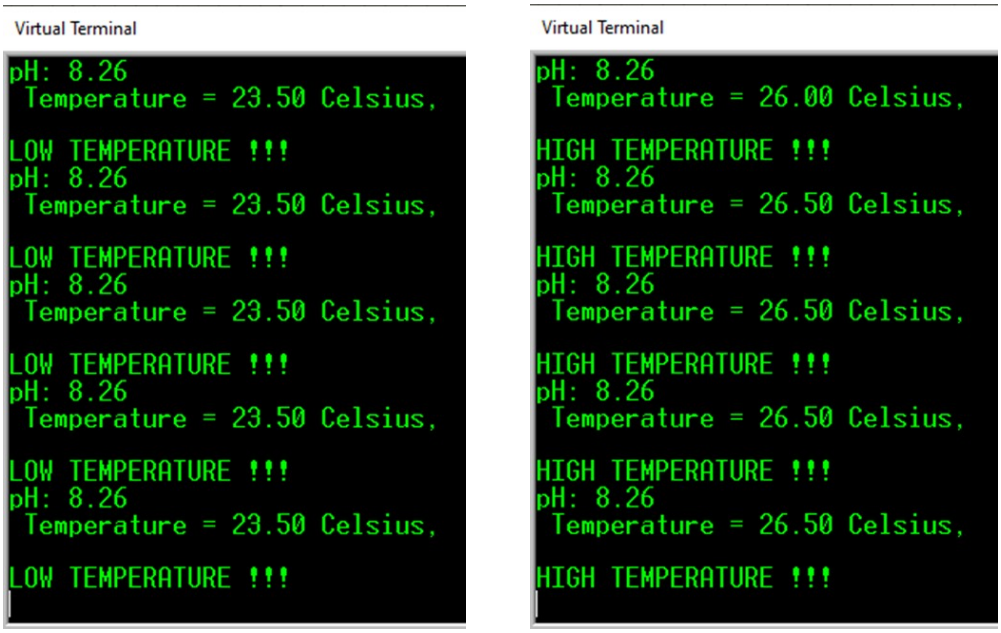



Рис. 4.10. Вивід тривожних повідомлень в ПК по послідовному інтерфейсу

Успішне тестування в емуляторі Proteus ISIS підтвердило стабільність роботи основного алгоритму та точність обчислень інформаційної система та розроблених програмних модулів.

# ВИСНОВКИ

В дипломній роботі було реалізовано апаратне та програмне забезпечення інформаційної система вимірювання показника pH розчинів. Система вимірює такі фізико-хімічні параметри розчинів як: температура та водневий показник (pH) а також автоматичне підтримання заданого температурного режиму для стабілізації параметрів розчину. Отримані дані виводяться на РКД та через послідовний інтерфейс в ПК.

При розробці інформаційної система вимірювання показника pH розчинів використано сучасну елементну базу. Основними її елементами є платформа Arduino Uno на мікроконтролері AVR ATmeg328, цифровий термодатчик DS18B20, а для виводу результатів – LCD-модуль формату 16×2.

Спроектовано електричну принципову схему та створено імітаційну модель інформаційної система вимірювання показника pH розчинів в Proteus VSM. Розроблено її алгоритм роботи і програмне забезпечення для МК ATmega328 платформи Arduino Uno на мові C в середовищі Arduino IDE.

Створено програмні модулі для роботи з цифровим давачем температури, давачем кислотності та використано програмний модуль виводу інформації на РКД. Проведено моделювання роботи інформаційної системи в емуляторі Proteus ISIS. Успішне тестування в емуляторі Proteus ISIS підтвердило стабільність роботи основного алгоритму та точність обчислень інформаційної система та розроблених програмних модулів.

Відмінною особливісю розробленої системи вимірювання показника pH розчинів є його мала собівартість і багатофункціональність. Система вирізняється високою надійністю завдяки впровадженню механізмів реагування на критичні зміни температури, що безпосередньо впливає на точність вимірювання кислотності.

В процесі реалізації проєкту здобуто нові теоретичні знання та прикладні навички у сфері проектування і програмування вбудованих систем.

## СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ


1. Електронний ресурс. Режим доступу: https://www.arduino.cc/
2. Середовище розробки Arduino IDE. Електронний ресурс: https://www.arduino.cc/en/Main/Software
3. Brian Huang and Derek Runberg. The Arduino Inventor's Guide. – SparkFun Electronics, 2017. – p. 472.
4. Електронний ресурс. Режим доступу: https://learn.adafruit.com/category/learn-arduino.
5. Jonathan Oxer and Hugh Blemings. Practical Arduino: Cool Projects for Open Source Hardware. - Springer-Verlag New York , 2009 - P. 445.
6. Massimo Banzi. Getting Started with Arduino. – 2nd Edition. O'Really Media Inc., 2011 – p. 130.
7. Emily Gertz & Patrick Di Justo. Environmental Monitoring with Arduino. Building Simple Devices to Collect Data About the World Around Us. – Published by O'Really Media Inc., 2012 – p. 96.
8. Kimmo and Tero Karvinen. Make: Arduino Bots and Gadgets. – Published by O'Really Media Inc., 2011 – p. 297.
9. Brian Evans. Beginning Arduino Programming. Writing Code for the Most Popular Microcontroller Board in the World. – Springer Science + Business Media, 2011 – p. 271.
10. Michael Margolis. Arduino Cookbook, Second Edition. – Published by O'Reilly Media Inc., 2012. – p. 724.
11. Enrique Ramos Melgar and Ciriaco Castro Diez with Przemek Jawosrki. Arduino and Kinect Projects. Design, Build, Blow Their Minds. – Springer Sceince+Business Media New York, 2012. – p. 411.
12. Електронний ресурс. Режим доступу: https://www.arduino.cc/en/-Tutorial/HomePage.

# ДОДАТКИ

# Додаток 1

## Головний програмний модуль інформаційної системи вимірювання показника pH розчинів

```
/*
  # Інформаційна система вимірювання показника pH розчинів
  # для криміналістичної експертизи та лабораторного аналізу в поліції
*/

#include <LiquidCrystal.h> // бібліотека LiquidCrystal для роботи з РК - дисплеєм

#define F_CPU 16000000L   // Частота тактів CPU рівна 16 МГц

#define THERM_PORT PORTC // порт на якому причеплено давачі температури
#define THERM_DDR DDRC   // регістр налаштування порта для давача температури
#define THERM_PIN PINC
#define INDOOR_THERM 1   // пін на давач температури not used! - select=0
#define OUTDOOR_THERM 3  // пін на давач температури A3 (PC3) -  select=1

/* список команд давача температури DS18B20 */
#define THERM_CMD_CONVERTTEMP    0x44
#define THERM_CMD_RSCRATCHPAD    0xbe
#define THERM_CMD_WSCRATCHPAD    0x4e
#define THERM_CMD_CPYSCRATCHPAD  0x48
#define THERM_CMD_RECEEPROM      0xb8
#define THERM_CMD_RPWRSUPPLY     0xb4
#define THERM_CMD_SEARCHROM      0xf0
#define THERM_CMD_READROM        0x33
#define THERM_CMD_MATCHROM       0x55
#define THERM_CMD_SKIPROM        0xcc
#define THERM_CMD_ALARMSEARCH    0xec
/* константи */
#define THERM_9BIT_RES  0  // роздільна здатність давача 9 біт
#define THERM_10BIT_RES 1   // роздільна здатність давача 10 біт
#define THERM_11BIT_RES 2 // роздільна здатність давача 11 біт
#define THERM_12BIT_RES 3 // роздільна здатність давача 12 біт

typedef unsigned char uint8_t;  typedef unsigned int uint16_t;

char  buffer[33];  // буфер для LCD
float  toutf=0.0f;
volatile  uint8_t  meas_res_tout;
volatile char  fsmasw [6] ;   //  робочий масив float to string
float tempInCelsius, tempInFahrenheit; // температура

// pH
#define SensorPin A0          // pH meter Analog output to Arduino Analog Input 0
#define Offset 0.00           // deviation compensate
#define LED 13
#define samplingInterval 20
#define printInterval 800
```

```
#define ArrayLenth  40    // times of collection
int pHArray[ArrayLenth];   // Store the average value of the sensor feedback
int pHArrayIndex = 0;

// ініціалізація бібліотеки LiquidCrystal номера пінів Arduino до яких підключено LCD
LiquidCrystal lcd(12, 11,6, 5, 4, 3);

const int btnPin = 7; // номер піна до якого підключена кнопка "Цельсій/Фаренгейт"
const int ledPin =  13; // номер піна до якого підключений синій LED

const int RED = 8;        // LED "тревога" під'єднано на pin 8
const char SPEAKER = 10;  // пищик  під'єднано на pin 10

const char FAN = 2;      //  охолодження – вентилятор
const char HOT = 9;      //  підігрів - нагрівач

const float TMAX = 26;   // максимально-допустима температура розчину (26)
const float TMIN =24;    // мінімально-допустима температура розчину (24)

const float PHMAX = 8.5; //  максимально-допустима кислотність
const float PHMIN = 6.5; //  мінімально-допустима кислотність

int btnState = 0; // змінна для зчитування статусу кнопки

const int pHSensorAnalogInPin = A0; // пін вхід АЦП до якого підключено давач pH
float pHSensorValue = 0;  // ph

int iz=0;           // duty для пищика

float tempSensorValue[2] = {0, 0};  // масив для температури Цельсій Фаренгейт

String tempUnitSign[2];
int decPlaces = 1;
int unit = 0; // початкові одиниці виводу температури в градусах Цельсія

void zvyk(void)   // функція генерування звуку
 {
   pinMode(SPEAKER, OUTPUT);   // sets the digital pin as output
   iz=127; // duty= 50%
   analogWrite(SPEAKER, iz);   // speaker on,  speaker- pin.
    delay(1000);
   pinMode(SPEAKER, INPUT);
 }


// Давач pH. Визначення рівня pH розчину.
float pH_measurement()
{
 int buff[10], temp;
 unsigned long int pHSensorAvgValue;
 float pHSensorVolt, pHSensorVal;
```

```
 for (int i = 0; i < 10; i++)
 {
  buff[i] = analogRead(pHSensorAnalogInPin);
  delay(10);
 }
 for (int i = 0; i < 9; i++)
  for (int j = i + 1; j < 10; j++)
   if (buff[i] > buff[j])
   {
    temp = buff[i];
    buff[i] = buff[j];
    buff[j] = temp;
   }
 pHSensorAvgValue = 0;
 for (int i = 2; i < 8; i++)
  pHSensorAvgValue += buff[i];
 pHSensorVolt = (float) pHSensorAvgValue * 5.0 / 1024 / 6;

  pHSensorVal = -2.80 * pHSensorVolt + 14;

 Serial.print("pH: ");
 Serial.println(pHSensorVal, 2);
 Serial.print(" ");
 return pHSensorVal;
}


void floa_to_str(float  outf ) // переводить дробове число в текст (char fsmasw [6] )
{
 int  digitx = 0, n, n0 ;
 char sd ;                              //  float to string
 unsigned char  nk=0 , q=0;          // кількість знаків в числі  float to string
 byte k1 = 1, k = 0, x=0, y=0 ;          //  float to string

delete []fsmasw ;
digitx = (int) (10 * toutf) ;  //переводимо дробове в ціле число (множимо на 10)
 if (digitx < 0)              // аналізуємо знак числа (додатнє, чи від'ємне)
 {
  fsmasw [0]   =  '-' ;      // робимо число додатнім
  digitx = (digitx * (-1));
 }
 else    { fsmasw [0]   =  '+' ; } ; // пишемо знак в масив
 n = digitx;
 while (n > 0)
 { n = n / 10;   k++;  }
if (k == 0) {   k = 2;  };              // якщо цифра є 0!
if (digitx < 10) {   k = 2;  };
 nk  = k;                          //  кількість цифр в числі
 // розбивка числа(digitx) на символи цифр і запис їх в робочий масив
 n0 = digitx;
 for (x = 0; x < nk; x++)       // nk– кількість цифр в числі digitx
 {
```

```
   k1 = n0 - (n0 / 10) * 10;   // sd – символ цифри для запису в масив  fsmasw [6]
   switch (k1)
   {
    case 0:   sd = '0';  break;  case 1:   sd = '1';  break;   case 2:   sd = '2';  break; case 3:   sd = '3';  break;
    case 4:   sd = '4';  break;  case 5:   sd = '5';  break;   case 6:   sd = '6';  break;  case 7:   sd = '7';  break;
    case 8:   sd = '8';  break;  case 9:   sd = '9';  break;      default:  break;
   };
   y = (nk - x);
   fsmasw[y] = sd;  // запис в масив цифр числа, де fsmasw [1]- старша цифра числа…
   n0 = n0 / 10;
  };
  q = nk + 1;   fsmasw[q] = fsmasw[nk];
  fsmasw[nk] = '.';  q = nk + 2;  fsmasw[q] = '\0';     //  вставити “кому”
}

// Функції давача температури DS18B20
/* структура для збереження RAM */
struct __ds18b20_scratch_pad_struct
{
  uint8_t  temp_lsb, temp_msb, tem_high, temp_low, conf_register,
         res1, res2, res3, crc;
};

/* функції для внутрішнього використання */
uint8_t therm_reset(void);
void therm_write_bit(uint8_t bit);
uint8_t therm_read_bit(void);
uint8_t therm_read_byte(void);
void therm_write_byte(uint8_t byte);

/* функція обчислення контрольної суми */
uint8_t therm_crc8(unsigned char *data, unsigned char num_bytes);

/* функція ініціалізація давачів температури */
uint8_t therm_init(uint8_t sensor_id, char temp_low, char temp_high, uint8_t resolution);

/* функція зчитування температури з давача */
uint8_t therm_read_temperature(uint8_t sensor_id, float *temp);

struct __ds18b20_scratch_pad_struct __ds18b20_scratch_pad;
uint8_t therm_dq;

void therm_input_mode(void)
{  THERM_DDR&=~(1<<therm_dq);}

void therm_output_mode(void)
{ THERM_DDR|=(1<<therm_dq);}

void therm_low(void)
{ THERM_PORT&=~(1<<therm_dq); }

uint8_t therm_reset()
```

```
{
  uint8_t i;
  therm_low(); therm_output_mode(); delayMicroseconds(480);   //delay_us(480);
  therm_input_mode();  delayMicroseconds(60); //delay_us(60);
   //зберігаємо значення на шині і чекаємо завершення 480 мкс періода
  i=(THERM_PIN & (1<<therm_dq));  delayMicroseconds(420); // delay_us(420);
  if ((THERM_PIN & (1<<therm_dq))==i) return 1;//повертаємо результат(0=OK, 1=WRONG)
  return 0;
}

void therm_write_bit(uint8_t _bit)
{
  therm_low();  therm_output_mode(); delayMicroseconds(1);   //delay_us(1);
  if (_bit) therm_input_mode();  delayMicroseconds(60);     //delay_us(60);
  therm_input_mode();
}

uint8_t therm_read_bit(void)
{
  uint8_t _bit=0;
  therm_low();  therm_output_mode();  delayMicroseconds(1);       //delay_us(1);
  therm_input_mode();  delayMicroseconds(14);                     //delay_us(14);
  if (THERM_PIN&(1<<therm_dq)) _bit=1; delayMicroseconds(45);     //delay_us(45);
  return _bit;
}

uint8_t therm_read_byte(void)
{
  uint8_t i=8, n=0;
  while (i--)
  { n>>=1; n|=(therm_read_bit()<<7); }
  return n;
}

void therm_write_byte(uint8_t byte)
{
  uint8_t i=8;
  while (i--)
  { therm_write_bit(byte&1); byte>>=1;  }
}

uint8_t therm_crc8(uint8_t *data, uint8_t num_bytes)
{
  uint8_t byte_ctr, cur_byte, bit_ctr, crc=0;
  for (byte_ctr=0; byte_ctr<num_bytes; byte_ctr++)
  {
    cur_byte=data[byte_ctr];
    for (bit_ctr=0; bit_ctr<8; cur_byte>>=1, bit_ctr++)
    if ((cur_byte ^ crc) & 1) crc = ((crc ^ 0x18) >> 1) | 0x80;
    else crc>>=1;
  }
  return crc;
```

```
}

 uint8_t therm_init(uint8_t sensor_id, char temp_low, char temp_high, uint8_t resolution)
{
 resolution=(resolution<<5)|0x1f;
 if (sensor_id) therm_dq=OUTDOOR_THERM; // ініціалізуємо давач sensor_id
 else therm_dq=INDOOR_THERM;
 if (therm_reset()) return 1;
 therm_write_byte(THERM_CMD_SKIPROM);
 therm_write_byte(THERM_CMD_WSCRATCHPAD);
 therm_write_byte(temp_high); therm_write_byte(temp_low);
 therm_write_byte(resolution);
 therm_reset();
 therm_write_byte(THERM_CMD_SKIPROM);
 therm_write_byte(THERM_CMD_CPYSCRATCHPAD); delay(15);   //delay_ms(15);
 return 0;
}

uint8_t therm_read_spd(void)
{
 uint8_t i=0, *p;
 p = (uint8_t*) &__ds18b20_scratch_pad;
 do
   *(p++)=therm_read_byte();
 while(++i<9);
 if (therm_crc8((uint8_t*)&__ds18b20_scratch_pad,8)!=__ds18b20_scratch_pad.crc)
 return 1;
 return 0;
}


uint8_t therm_read_temperature(uint8_t sensor_id, float *temp)
{
 uint8_t digit, decimal, resolution, sign;
 uint16_t meas, bit_mask[4]={0x0008, 0x000c, 0x000e, 0x000f};

 if (sensor_id) therm_dq=OUTDOOR_THERM;
   else therm_dq=INDOOR_THERM;
   if (therm_reset()) return 1;
 therm_write_byte(THERM_CMD_SKIPROM);
 therm_write_byte(THERM_CMD_CONVERTTEMP);
 while(!therm_read_bit());
 therm_reset();
 therm_write_byte(THERM_CMD_SKIPROM);
 therm_write_byte(THERM_CMD_RSCRATCHPAD);
 if (therm_read_spd()) return 1;
 therm_reset();
 resolution=(__ds18b20_scratch_pad.conf_register>>5) & 3;
 meas=__ds18b20_scratch_pad.temp_lsb;  // LSB
 meas|=((uint16_t)__ds18b20_scratch_pad.temp_msb) << 8; // MSB
 if (meas & 0x8000)
 {
```

```
    sign=1;  //відмічаємо мінусову температуру
    meas^=0xffff;  //перетворюємо в плюсову
    meas++;
  }
  else sign=0;
  //зберігаємо цілу і дробову частини температури
  digit=(uint8_t)(meas >> 4); //зберігаємо цілу частину
  decimal=(uint8_t)(meas & bit_mask[resolution]); //отримуємо дробову частину
  *temp=digit+decimal*0.0625;
  if (sign) *temp=-(*temp); //ставемо знак мінус, якщо мінусова температура
  return 0;
}

void setup()
{
   DDRD = 0X78; DDRB = 0X18;
  Serial.begin(9600); // ініціалізація послідовного інтерфйесу, швидкість 9600 бод
  // ініціалізуємо LED пін як вихід:
  pinMode(ledPin, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(RED, OUTPUT);   // sets the digital pin as output- alarm

  pinMode(FAN, OUTPUT);          // sets the digital pin 2 as output
  pinMode(HOT, OUTPUT);          // sets the digital pin 9 as output

 //ініціалізація давачів температури
  pinMode (A3, OUTPUT); digitalWrite (A3, HIGH);
  if (therm_init(1, -55, 125, THERM_9BIT_RES))
  {
    lcd.clear();
    lcd.print("T: error_");
    digitalWrite(RED, HIGH);    // вставити LED red і звук!!!!!!!!!
    pinMode(SPEAKER, OUTPUT);   // sets the digital pin as output
    iz=127; // duty= 50%
    analogWrite(SPEAKER, iz);   // speaker on,  speaker- pin.
     delay(10);
     while(1);
  }

  // ініціалізуємо пін кнопки як вхід:
  pinMode(btnPin, INPUT);
  // ініціалізуємо LCD
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" PH MEASUREMENT");
  lcd.setCursor(0, 1);
  lcd.print("     SYSTEM  ");
  Serial.println("   ");
  Serial.println("  * PH MEASUREMENT MONITORING SYSTEM * ");
  Serial.println("   ");
// тест світлодіодів
```

```
   digitalWrite(RED, HIGH);     // червоний
   digitalWrite(ledPin, HIGH);  // синій
   digitalWrite(FAN, HIGH);     // охолодження
   digitalWrite(HOT, HIGH);     // нагрівання

  zvyk();
  delay(900); delay(900);delay(900);
  digitalWrite(RED, LOW);    // вимкнути
  digitalWrite(ledPin, LOW); // вимкнути

  digitalWrite(FAN, LOW);  // охолодження вимкнути
  digitalWrite(HOT, LOW);  // нагрівання  вимкнути

 delay(500);  delay(900);  delay(900);
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print(" Developed by");
 lcd.setCursor(0, 1);
 lcd.print(" J.Jabiak, 2026");  //
 delay(500);  delay(900);  delay(900);
 lcd.clear();

 //tempUnitSign[0] = char(223);
 tempUnitSign[0] += "C ";
 // tempUnitSign[1] = char(223);
 tempUnitSign[1] += "F ";

}

void loop()
{
 // вимірювання кислотності
 pHSensorValue = pH_measurement();
 //lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("pH:");
 lcd.print(pHSensorValue, 1); // вивід значення ph на LCD - float

//міряємо температуру з давачів
  meas_res_tout=therm_read_temperature(1,&toutf);  //sensor - PC3
   delay(10);
   if ( meas_res_tout)
   {
     lcd.clear();
        lcd.print("T: error_");
       digitalWrite(RED, HIGH);    // вставити LED red і звук!!!!!!!!!!
       pinMode(SPEAKER, OUTPUT);   // sets the digital pin as output
       iz=127; // duty= 50%
       analogWrite(SPEAKER, iz);   // speaker on,  speaker- pin.
       delay(1000);  delay(1000);
       pinMode(SPEAKER, INPUT);
       Serial.println(" ");
```

```
        Serial.println("Temperature sensor ERROR ! ");
        Serial.println(" ");
        // помилка DS18B20

    }
      else
    {
      digitalWrite(RED, LOW);
      tempInCelsius=toutf;
      tempInFahrenheit = tempInCelsius * 1.8 + 32.0;
      Serial.print("Temperature = ");
      Serial.print(tempInCelsius);      // градуси Цельсія
      Serial.print(" Celsius,    ");
      Serial.print(tempInFahrenheit);  // по Фаренгейту
      Serial.println(" Fahrenheit");
      Serial.println(" ");

   if (unit==0)
   {
     floa_to_str( toutf );   // конвертація дробового числа toutf в текст
      sprintf(buffer, "%s%cC   ", fsmasw, 0xdf);
      lcd.setCursor(6, 0);
      lcd.print(" T:");
      lcd.print("      ");
      lcd.setCursor(9, 0);
      lcd.print(buffer);
      delay(1000);
    // Serial.println("T Celsius,    ");
   }
   else {

        lcd.setCursor(6, 0);
        lcd.print(" T: ");
        lcd.print("      ");
        lcd.setCursor(9, 0);
        lcd.print(tempInFahrenheit,1);
        lcd.write(0xDF);  // друк символа градус
        lcd.print("F");

       };

      delay(1000);
    };
if(tempInCelsius>=TMAX)  //TMAX = 26;
{
  lcd.setCursor(0, 1);
  lcd.print("HIGH TEMPERATURE");
  zvyk();
  digitalWrite(RED, HIGH);   // turns the RED  LED - on ТРЕВОГА
  digitalWrite(FAN, HIGH);   // увімкнути охолодження

  Serial.println("HIGH TEMPERATURE !!!");
```

```
  delay(1000);
   digitalWrite(RED, LOW);   // turns the RED  LED - off ТРЕВОГА
  pinMode(SPEAKER, INPUT);   // sets the digital pin as input
}
else
{
  lcd.setCursor(0, 1);
  lcd.print("               "); // Print a message to the LCD.
  digitalWrite(FAN, LOW);       // вимкнути охолодження
};
if(tempInCelsius<=TMIN)  //TMIN = 24;
{
  lcd.setCursor(0, 1);
  lcd.print("LOW TEMPERATURE");
  zvyk();
  digitalWrite(RED, HIGH);   // turns the RED  LED - on ТРИВОГА
  digitalWrite(HOT, HIGH);   // увімкнути нагрів

  Serial.println("LOW TEMPERATURE !!!");
  delay(1000);
   digitalWrite(RED, LOW);   // turns the RED  LED - off ТРИВОГА
  pinMode(SPEAKER, INPUT);   // sets the digital pin as input
}
else
{
  lcd.setCursor(0, 1);
  lcd.print("               "); // Print a message to the LCD.
  digitalWrite(HOT, LOW);       // вимкнути нагрів

};

  // зчитуємо статус кнопки
  btnState = digitalRead(btnPin);
  // перевіряємо чи натиснута кнопка. Якщо так, btnState є високим (HIGH):
  if (btnState == HIGH) {
    if (unit)
    {
      unit = 0;
      // погасити синій світлодіод LED:
      digitalWrite(ledPin, LOW);
       delay(1000);
    }
    else
    {
      unit = 1;
      // засвітити синій світлодіод LED:
      digitalWrite(ledPin, HIGH);
    }
  }
   delay(10);
}
```

```
void pH_measurement2(void)
{
  static unsigned long samplingTime = millis();
  static unsigned long printTime = millis();
  static float pHValue, voltage;
  if (millis() - samplingTime > samplingInterval)
  {
    pHArray[pHArrayIndex++] = analogRead(SensorPin);
    if (pHArrayIndex == ArrayLenth) pHArrayIndex = 0;
    voltage = avergearray(pHArray, ArrayLenth) * 5.0 / 1024;
    pHValue = 3.5 * voltage + Offset;
    samplingTime = millis();
  }
  if (millis() - printTime > printInterval)  //
  {
    Serial.print("Voltage:");
    Serial.print(voltage, 2);
    Serial.print("    pH value: ");
    Serial.println(pHValue, 2);
    digitalWrite(LED, digitalRead(LED) ^ 1);
    printTime = millis();
  }
}
double avergearray(int* arr, int number) {
  int i;
  int max, min;
  double avg;
  long amount = 0;
  if (number <= 0) {
    Serial.println("Error number for the array to avraging!/n");
    return 0;
  }
  if (number < 5) { //less than 5, calculated directly statistics
    for (i = 0; i < number; i++) {
      amount += arr[i];
    }
    avg = amount / number;
    return avg;
  } else {
    if (arr[0] < arr[1]) {
      min = arr[0];
      max = arr[1];
    }
    else {
      min = arr[1]; max = arr[0];
    }
    for (i = 2; i < number; i++) {
      if (arr[i] < min) {
        amount += min; // arr < min
        min = arr[i];
      } else {
        if (arr[i] > max) {
```

```
        amount += max; // arr > max
        max = arr[i];
       } else {
        amount += arr[i]; // min <= arr <= max
       }
     }
    }
    avg = (double)amount / (number - 2);
  }
  return avg;
}
```