МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій

Кафедра комп’ютерної інженерії та електромеханіки

**КВАЛІФІКАЦІЙНА РОБОТА**

на тему:

**Комп'ютерна система моніторингу та контролю рівня води на базі Arduino**

Рівень вищої освіти другий (магістерський)

Спеціальності 123 «Комп’ютерна інженерія»

Освітня програма «Комп’ютерна інженерія»

Виконав: студент групи МгКІ-23 .

Санніков Максим Русланович .

(прізвище та ініціали)

Керівник: к.т.н., доц. Стаценко Д.В.

Рецензент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(науковий ступінь, вчене звання, прізвище та ініціали)

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій .

Кафедра комп’ютерної інженерії та електромеханіки .

Спеціальність 123 «Комп’ютерна Інженерія».

Освітня програма «Комп’ютерна Інженерія»

**ЗАТВЕРДЖУЮ**

Завідувач кафедри КІЕМ

\_\_\_\_\_\_\_\_\_\_\_\_ Дмитро СТАЦЕНКО

“\_\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 року

**ЗАВДАННЯ**

**НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**

Саннікову Максиму Руслановичу.

(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи **\_розробка комп'ютерної системи моніторингу та контролю рівня води на базі Arduino**

Науковий керівник роботи \_Стаценко Дмитро Володимирович, к.т.н., доцент . (прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

.

затверджені наказом вищого навчального закладу від 03.09.2024 № 188-уч.

2. Вихідні дані до кваліфікаційної роботи: *комп’ютерна система моніторингу та контролю рівня води,* *характеристики резервуарів, датчиків, мікроконтролера, а також вимоги до точності, швидкості та умов експлуатації, навчальна та методична література.*

3. Зміст кваліфікаційної роботи (перелік питань, які потрібно опрацювати):

1.Провести аналіз існуючих систем, схем, методів для моніторингу рівня води. 2. Описати функціональність, переваги використання мікроконтролера Arduino для автоматизації системи моніторингу як зі створеним пристроєм чи завантаженим. 3. Опис порівняння існуючих систем моніторингу з розробленою.

4. Дата видачі завдання \_\_\_\_\_\_\_\_\_\_\_\_2024

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів кваліфікаційної роботи | Терміни виконання етапів | Примітка про виконання |
| 1 | Вступ | 09.09.2024 |  |
| 2 | РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ | 20.09.2024 |  |
| 3 | РОЗДІЛ 2. АНАЛІЗ СИСТЕМ МОНІТОРИНГУ | 05.10. 2024 |  |
| 4 | РОЗДІЛ 3. СТВОРЕННЯ СХЕМИ ТА ЇЇ СИМУЛЯЦІЯ | 25.10.2024 |  |
| 5 | Висновки | 28.10.2024 |  |
| 6 | Оформлення (чистовий варіант) | 31.10.2024 |  |
| 7 | Подача кваліфікаційної роботи науковому керівнику для відгуку | 01.11.2024 |  |
| 8 | Подача кваліфікаційної роботи для рецензування (за 14 днів до захисту) | 09.11.2024 |  |
| 9 | Перевірка кваліфікаційної роботи на наявність ознак плагіату та текстових співпадінь (за 10 днів до захисту) | 11.11.2024 |  |
| 10 | Подання кваліфікаційної роботи на завідувачу кафедри (за 7днів до захисту) | 18.11.2024 |  |

З завданням ознайомлений:

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_**  Максим САННІКОВ

( підпис ) (Власне ім’я та ПРІЗВИЩЕ)

Науковий керівник \_\_\_\_\_\_\_\_\_\_\_\_ \_\_Дмитро СТАЦЕНКО.

( підпис ) (Власне ім’я та ПРІЗВИЩЕ)

**АНОТАЦІЯ**

**Санніков М. Р. Комп'ютерна система моніторингу та контролю рівня води на базі Arduino – Рукопис.**

Дипломна кваліфікаційна робота за спеціальністю 123 Комп’ютерна інженерія, освітньою програмою «Комп’ютерні системи та мережі» – Київський національний університет технологій та дизайну, Київ, 2024 рік.

Дипломну кваліфікаційну роботу присвячено вирішенню проблеми – розробки комп'ютерна система моніторингу та контролю рівня води на базі Arduino шляхом використовування програмнних середовищ Arduino IDE та Proteus. Arduino IDE було використано для написання програмного коду на мові програмування C/C++, що дозволило реалізувати логіку роботи системи, включаючи моніторинг рівня води, керування насосами, а також обробку вхідних сигналів з датчиків.

У середовищі Proteus створена віртуальна модель системи, яка включає мікроконтролер Arduino, сенсори та виконавчі механізми. Це дозволило протестувати та симулювати роботу системи до її фізичної реалізації, перевіряючи правильність коду та підключень без необхідності збирання реального обладнання.

Така система може використовуватися в різноманітних сферах, зокрема для моніторингу та контролю рівня води в резервуарах або водоймах на промислових об'єктах, у житлових будинках чи сільськогосподарських системах. Вона є універсальною і може бути адаптована для вирішення широкого спектра задач, незалежно від того, чи йдеться про водопостачання, зрошення, чи про управління водними ресурсами в автоматизованих системах. *Ключові слова: Arduino IDE, Proteus, моніторингу та контролю, протестувати та симулювати*, *C/C++.*

**ABSTRACT**

**Sannikov M. R. Computer-Based Water Level Monitoring and Control System Based on Arduino – Manuscript.**

Qualification thesis in the specialty 123 Computer Engineering, within the educational program "Computer Systems and Networks" – Kyiv National University of Technologies and Design, Kyiv, 2024.

This qualification thesis is dedicated to solving the problem of developing a computer-based water level monitoring and control system based on Arduino using the Arduino IDE and Proteus software environments. The Arduino IDE was used to write the program code in the C/C++ programming language, allowing the implementation of system logic, including water level monitoring, pump control, and processing input signals from sensors.

In the Proteus environment, a virtual model of the system was created, which includes the Arduino microcontroller, sensors, and actuators. This allowed the system to be tested and simulated before its physical implementation, verifying the accuracy of the code and connections without the need to assemble actual hardware.

Such a system can be used in various fields, particularly for monitoring and controlling water levels in tanks or reservoirs at industrial sites, residential buildings, or agricultural systems. It is versatile and can be adapted to solve a wide range of tasks, whether related to water supply, irrigation, or water resource management in automated systems.

Keywords: Arduino IDE, Proteus, monitoring and control, testing and simulation, C/C++.

**ЗМІСТ**

[ВСТУП 7](#_Toc74526011)

[РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ 1](#_Toc74526012)0

[1.1. Arduino IDE 1](#_Toc74526013)2

[1.2. Мова C/C++ 14](#_Toc74526015)

[1.3. Proteus 15](#_Toc74526014)

[1.4. Інтеграція Arduino в Proteus 21](#_Toc74526016)

[1.5. Інтеграція водяних резервуарів 23](#_Toc74526018)

[1.6. Існуючих систем моніторингу води 26](#_Toc74526018)

[Висновки до розділу 1 27](#_Toc74526019)

[РОЗДІЛ 2. АНАЛІЗ СИСТЕМ МОНІТОРИНГУ 27](#_Toc74526020)

[2.1. Архітектура та функціональність систем моніорингу 27](#_Toc74526021)

[2.2. Комп’ютерна система моніторингу води на Arduino UNO 33](#_Toc74526022)

[2.3. Метод вимірювання та пристрої 35](#_Toc74526022)

[2.4. Порівняння систем з проектом 38](#_Toc74526022)

[Висновки до розділу 2 41](#_Toc74526023)

[РОЗДІЛ 3. СТВОРЕННЯ СХЕМИ ТА ЇЇ СИМУЛЯЦІЯ 42](#_Toc74526024)

[3.1. Створення пристрою або датчика 42](#_Toc74526025)

[3.2. Створення та принцип роботи схеми в Proteus 47](#_Toc74526026)

[3.2.1. Створення схеми з використанням власного пристрою 49](#_Toc74526027)

[3.2.2.](#_Toc74526027) [Створення схеми з використанням завантаженого пристрою 49](#_Toc74526027)

[3.3. Архітектура та опис коду 53](#_Toc74526028)

[3.3.1. Опис коду з власним пристроєм 53](#_Toc74526028)

[3.3.2. Опис коду з завантаженим пристроєм 60](#_Toc74526028)

[3.4. Функціонал та порівняння 64](#_Toc74526028)

[Висновки до розділу 3 68](#_Toc74526029)

ЗАГАЛЬНІ [ВИСНОВКИ 69](#_Toc74526030)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 71](#_Toc74526031)

[ДОДАТКИ 7](#_Toc74526032)3

**ВСТУП**

**Актуальність роботи.** На даний момент у сучасному світі управління водними ресурсами є важливим завданням, оскільки кількість населення зростає, а доступ до чистої води стає дедалі обмеженішим. Відсутність ефективних методів контролю водних ресурсів може призвести до втрат води та аварій у системах водопостачання та зрошення. Створення надійних, автоматизованих систем моніторингу рівня води є ключовим кроком у підвищенні ефективності використання водних ресурсів. Це дослідження присвячене розробці комп'ютерної системи моніторингу та контролю рівня води на базі Arduino, яка має широке застосування в побуті, сільському господарстві, промислових системах і водопостачанні.

Технології моніторингу води є важливими для підтримання стабільного рівня водопостачання та попередження аварійних ситуацій, пов'язаних із переповненням або недоліком води в резервуарах. Система моніторингу рівня води на базі Arduino пропонує доступне та надійне рішення, що забезпечує автоматизований контроль рівня води в різних об'єктах. Такі системи широко використовуються в промислових резервуарах, басейнах, водних баштах та інших критичних інфраструктурах, де необхідно підтримувати оптимальний рівень води.

Комп'ютерна система моніторингу та контролю рівня води, заснована на Arduino, дозволяє автоматизувати цей процес, забезпечуючи постійний контроль та можливість реагувати на зміни рівня води без необхідності людського втручання. Такі системи можуть бути використані в домашньому господарстві для автоматизації систем водопостачання або зрошення, а також у промислових установках для ефективного управління водними ресурсами.

Arduino в поєднанні з мобільними пристроями дозволяє контролювати рівень води на віддалених об'єктах через смартфони або інші засоби. Це дає можливість відстежувати рівень води у режимі реального часу та отримувати сповіщення про критичні зміни, що дозволяє вчасно реагувати на можливі проблеми.

Завдяки простоті використання та широкому вибору сенсорів, Arduino стає ідеальною платформою для створення систем моніторингу. Використання ультразвукових датчиків для визначення рівня води дозволяє точно вимірювати відстань до поверхні води в резервуарах або водоймах. Зібрані дані можуть передаватися на комп'ютер або дисплей, що дозволяє візуалізувати процес моніторингу та забезпечити гнучкість у керуванні системою. Тому така система має значний потенціал для використання у водонапірних баштах для стабільного водопостачання, а також у промислових резервуарах для автоматизації виробничих процесів.

**Мета роботи** **–** створення комп’ютерної системи моніторингу та контролю рівня води на базі мікроконтролера Arduino для автоматизованого управління рівнями води в системах водопостачання. Система повинна забезпечувати ефективний контроль за рівнем води, запобіганню надзвичайних ситуаціяй і сприяти раціональному використанню водних ресурсів.

**Об’єкт дослідження –** процес створення комп'ютерної системи моніторингу та контролю рівня води на базі Arduino.

**Предмет дослідження –** вдосконалення процесу моніторингу та автоматизованого контролю рівня води в резервуарах за допомогою мікроконтролера Arduino. Включаючи наступні аспекти:

Програмне забезпечення для збору та обробки даних з сенсорів;

Алгоритми управління наповненням та випорожненням резервуарів;

Інтеграцію системи з апаратними компонентами, такими як контролери, насоси, клапани та сенсори рівня води.

**Методи досліджень**. Базою дослідження є комп'ютерна система моніторингу та контролю рівня води, розроблена на основі платформи Arduino на мові C/C++, та змодельована у віртуальному середовищі Proteus.

**Інформаційною базою досліджень** є навчальна та методична література, інтернет-ресурси, наукові публікації, літературні джерела та державні стандарти, що регулюють питання автоматизації та контролю водопостачання.

Також були використані технічні документи щодо роботи з мікроконтролерами Arduino, середовищем програмування Arduino IDE, а також технічні рекомендації по застосуванню сенсорів для вимірювання рівня води та інших компонентів.

**Наукова новизна.**

Запропоновано підхід, який дозволяє автоматизувати процеси моніторингу та регулювання рівня води, що забезпечує більш точний контроль та зменшує людське втручання.

Також впроваджено моделювання роботи системи в середовищі Proteus для попереднього тестування перед її фізичною реалізацією, що мінімізує ризики помилок та підвищує надійність.

**Практичне значення отриманих результатів.** Створення комп'ютерної системи моніторингу та контролю рівня води на базі Arduino, що дозволяє забезпечити її ефективність, простоту в експлуатації та адаптацію до конкретних потреб користувача.

Основні практичні результати:

1. Автоматизація процесу моніторингу рівня води – зменшення людського фактора та мінімізація ризику переповнення або пересихання резервуарів.
2. Оптимізація водопостачання – забезпечення ефективного контролю за використанням водних ресурсів у житлових будинках чи промислових системах.
3. Можливість дистанційного керування та контролю – зручність у використанні, коли потрібно регулювати рівень води без фізичної присутності на об'єкті.

**Структура та обсяг роботи.** Дипломна кваліфікаційна робота складається зі вступу, 3 розділів та висновків по них, загальних висновків, списку використаних джерел та додатків. Основний текст роботи викладений на 69 сторінках, містить 32 рисунок, 1 таблицю, список джерел з 19 найменувань. Загальний обсяг роботи, враховуючи додаток, складає 83 аркуша.

**РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ**

**1.1.** **Arduino IDE**

**Arduino** — це популярна апаратна платформа з відкритим кодом, яка призначена для розробки електронних пристроїв та прототипів. Вона відзначається простотою використання та універсальністю, тому широко використовується як професійними інженерами, так і ентузіастами в області електроніки та автоматизації. Arduino надає зручне середовище для взаємодії з фізичними пристроями, дозволяючи користувачам легко підключати датчики, реле, двигуни, світлодіоди та інші електронні компоненти до мікроконтролера, і тим самим контролювати різні аспекти фізичного світу.

Платформа складається з двох основних частин: апаратної (мікроконтролери, різні модулі та плати розширення) та програмної (Arduino IDE). Різноманітність моделей Arduino дозволяє користувачам обирати пристрій відповідно до своїх потреб.

**Рис. 1.1 – лого Arduino**

Наприклад, Arduino Uno, мабуть, найпопулярніша модель, бо часто використовується у проектах початкового рівня. Водночас, є більш потужні моделі, такі як Arduino Mega або Arduino Due, які мають більше входів/виходів та підходять для складних завдань.

Arduino підтримує широкий спектр сенсорів та модулів, що робить її ідеальною для розробки різних проектів – від простих пристроїв для освітлення до комплексних систем для автоматизації будинків або промислових рішень. Завдяки її відкритій архітектурі, існує величезна спільнота розробників, які діляться своїми проектами, бібліотеками та ідеями, що значно полегшує процес створення нових систем та пристроїв.

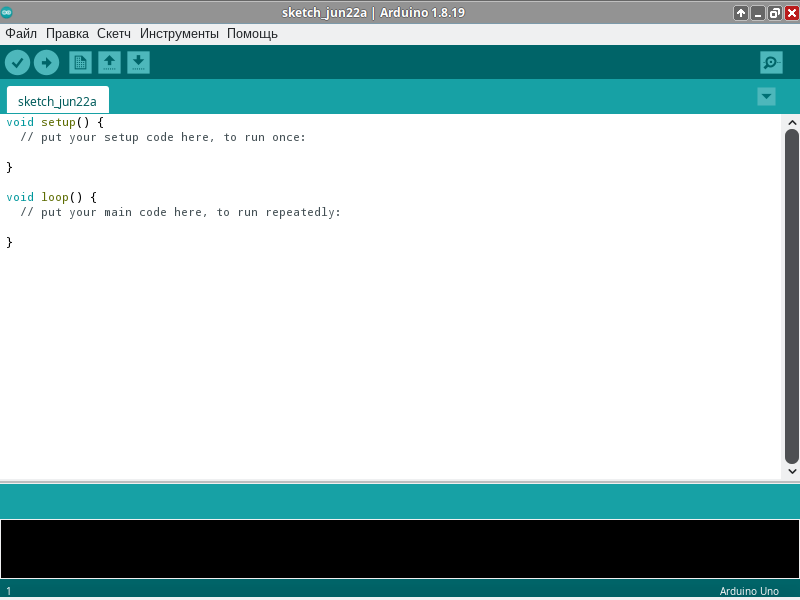
Однією з найважливіших складових Arduino є Arduino IDE (Integrated Development Environment) — це безкоштовна та зручна програма для розробки програмного забезпечення для платформ Arduino. Arduino IDE має простий інтерфейс, який дозволяє навіть новачкам у програмуванні швидко опанувати основи кодування та почати створювати свої проекти.

Програма підтримує написання коду на мові програмування C/C++, яка була спеціально адаптована для потреб користувачів Arduino.

Програмування в Arduino IDE здійснюється шляхом написання скетчів — невеликих програм, які завантажуються на мікроконтролер через USB-інтерфейс. Процес написання та завантаження коду в Arduino є дуже інтуїтивним: користувачі можуть створювати програми, використовуючи функції бібліотек для взаємодії з різними компонентами, наприклад, для роботи з датчиками температури, ультразвуковими сенсорами або двигунами.

Інтерфейс Arduino IDE має кілька важливих елементів:

* Головне вікно для написання коду, яке має звичний для програмістів вигляд.
* Кнопки для компіляції та завантаження коду на мікроконтролер, що робить процес швидким і зручним.
* Консоль для виведення повідомлень про помилки та інші важливі повідомлення під час компіляції.



**Рис. 1.2 – Вигляд середовища розробки Arduino IDE**

Для запуску коду, написаного користувачем, необхідні лише дві базові функції: одна для ініціалізації ескізу, а інша - для головного циклу програми. Після цього код компілюється і зв’язується з основною програмою main(), утворюючи виконуваний циклічний процес.

Вбудована програма avrdude відповідає за перетворення виконуваного коду у шістнадцятковий формат, який потім завантажується на плату Arduino за допомогою завантажувача, що знаходиться у внутрішньому програмному забезпеченні плати.

Arduino та Arduino IDE — це потужний інструментарій для розробки та створення інноваційних проектів в галузі електроніки та автоматизації. Легкість у використанні, безмежні можливості налаштувань роблять цю платформу ідеальним вибором як для новачків, так і для досвідчених інженерів.

**1.2.** **Мова** C/C++

Мова C була розроблена як інструмент для написання операційної системи UNIX. Її основною метою було забезпечення низькорівневого доступу до апаратного забезпечення при збереженні зручної для програмування абстракції. C швидко набула популярності завдяки своїй універсальності та продуктивності.

C++ була створена як розширення C і додала концепцію об'єктно-орієнтованого програмування (ООП). Це дозволило програмістам створювати більші та складніші системи, підтримуючи при цьому високу ефективність та контроль над ресурсами. У поєднанні, C і C++ стали основою для багатьох сучасних мов програмування та широко використовуються для розробки програмного забезпечення, операційних систем, ігрових движків, а також систем вбудованого програмування.

Мова програмування C/C++ — це потужний інструмент для розробки як високопродуктивних програм, так і програм для мікроконтролерів, таких як Arduino. Вона є однією з найпоширеніших мов програмування в світі завдяки своїй ефективності, гнучкості та широким можливостям для роботи з апаратним забезпеченням.

Коли Arduino була розроблена на початку 2000-х років, творці платформи обрали C/C++ як мову програмування через її вже доведену ефективність в управлінні апаратним забезпеченням та низьким рівнем доступу до ресурсів системи. Це стало логічним вибором, оскільки мова C/C++ вже мала міцну базу у сфері розробки для мікроконтролерів. Arduino використовує вбудовані бібліотеки та функції, написані на C/C++, що дозволяє легко взаємодіяти з його апаратними компонентами.

У контексті системи моніторингу та контролю рівня води, C/C++ дає можливість реалізувати складні алгоритми для управління насосами, датчиками та іншими пристроями. Наприклад, в рамках проекту можна:

1. Моніторити рівень води в резервуарах і автоматично вмикати або вимикати насоси в залежності від отриманих даних.
2. Обробляти вхідні сигнали від різних сенсорів і приймати рішення в реальному часі, що є критично важливим для управління водними ресурсами.
3. Оптимізувати використання енергії завдяки контролю роботи виконавчих механізмів, таких як клапани та насоси, що дозволяє зменшити споживання електроенергії.
4. Інтеграція з додатковими компонентами, такими як LCD дисплеї або контролери для ручного або дистанційного управління, що також легко реалізується на C/C++.

Отже, добре продуманій архітектурі C/C++ та високій продуктивності її розробки таких систем, як моніторинг та контроль рівня води, стає більш надійною та оптимізованою. Мова дозволяє обробляти складні завдання в реальному часі, що є ключовим для систем, які повинні працювати безперервно та точно. C/C++ як основні мови програмування в Arduino обумовлені не лише в системному програмуванні, але й її здатністю забезпечити гнучкість та контроль, які необхідні для успішного впровадження таких систем, як автоматизовані системи моніторингу та управління водою.

**1.3. Proteus**

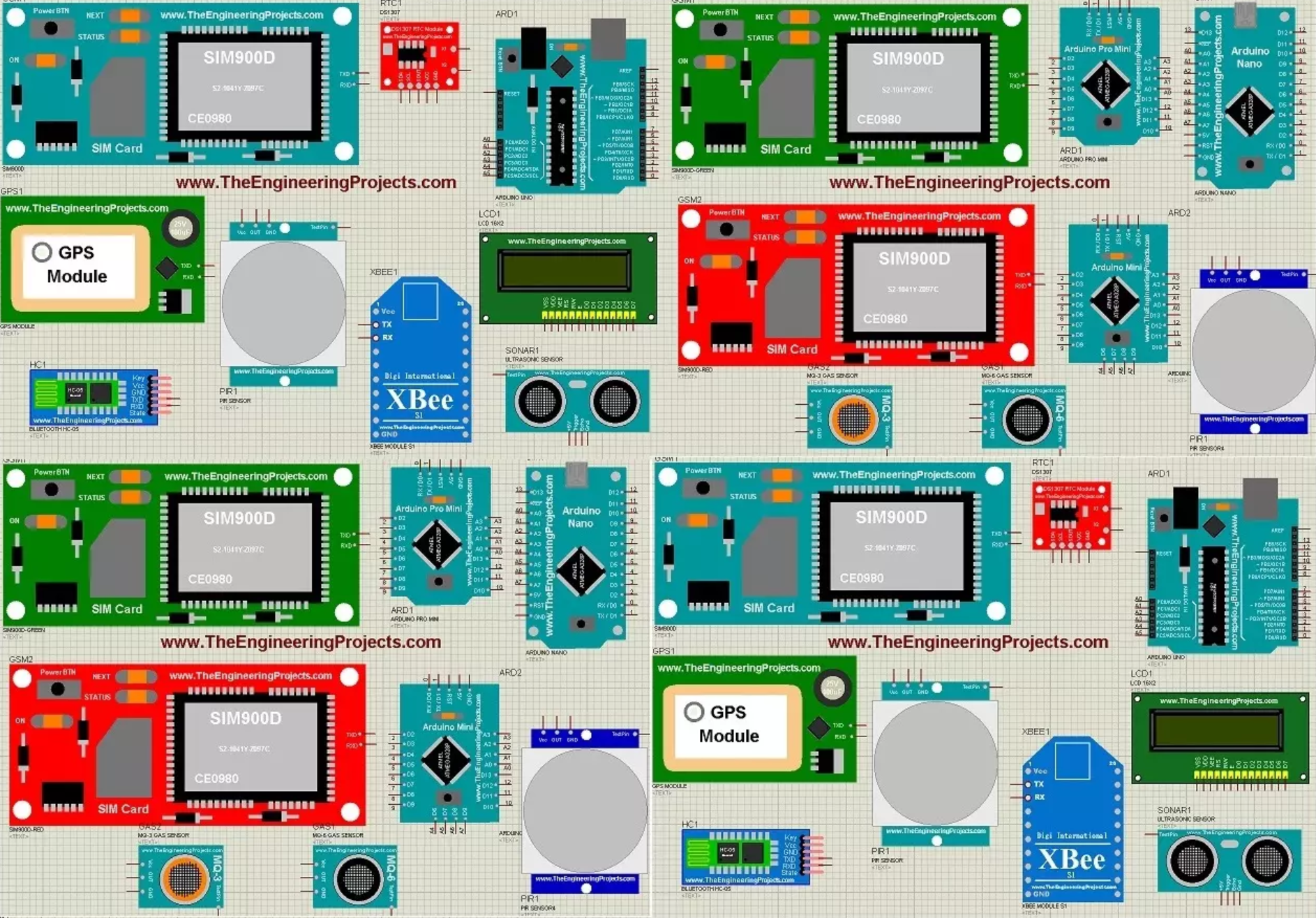
Proteus - це потужне програмне середовище для моделювання електронних схем, яке широко використовується інженерами та студентами для проектування та тестування апаратних систем.

В основі Proteus лежить можливість візуального створення електронних схем та їх симуляції. Завдяки своєму функціоналу, Proteus дозволяє моделювати роботу різних електронних компонентів, таких як резистори, конденсатори, мікроконтролери, дисплеї, датчики.



**Рис. 1.3. лого Proteus**

Proteus є програмою для проектування та симуляції електронних схем і друкованих плат (PCB). Це комплексне середовище для тестування як аналогових, так і цифрових схем. Одна з основних переваг Proteus полягає в тому, що користувачі можуть симулювати роботу електронних пристроїв, не маючи доступу до фізичного обладнання. Наприклад, створюючи схему з мікроконтролером Arduino, можна протестувати код і взаємодію компонентів без необхідності фізично збирати всі елементи.



**Рис. 1.4.** **Усі бібліотеки датчиків Proteus**

Proteus підтримує бібліотеки тисяч компонентів, включаючи мікроконтролери серій PIC, AVR, ARM і, звісно, Arduino. Програма дозволяє використовувати графічний інтерфейс для розміщення компонентів на віртуальному полі, підключення їх між собою та програмування вбудованих мікроконтролерів безпосередньо у середовищі.

Одна з найбільших переваг Proteus у поєднанні з Arduino полягає у можливості моделювати реальні проекти на основі цього мікроконтролера. Arduino IDE надає можливість написання коду, але тестування цього коду на фізичному обладнанні може вимагати додаткових ресурсів або часу. Завдяки підтримці мікроконтролерів Arduino у Proteus, користувачі можуть завантажувати програмний код у віртуальний мікроконтролер та спостерігати, як система буде працювати в реальних умовах.

У проекті комп'ютерна система моніторингу та контролю рівня води на базі Arduino, роль Proteus полягає у створенні віртуальної моделі системи. Основною перевагою тут є можливість тестування та симуляції всіх функцій системи до її фізичної реалізації. Це дозволяє побачити, як саме реагує Arduino на вхідні сигнали від датчиків рівня води, як контролюються виконавчі механізми, і як відображається інформація на дисплеї або іншому контролері. Proteus має важливе значення у завданні, а саме у:

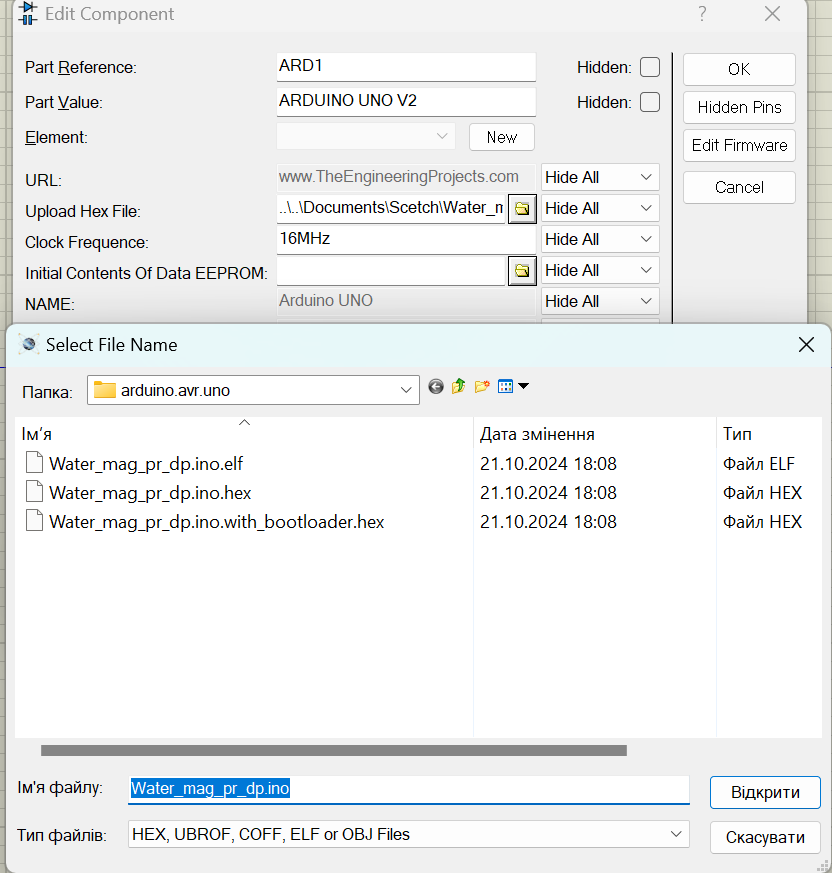
1. **Проектування електронних схем**: Proteus дозволяє розробляти та візуалізувати схеми підключення датчиків рівня води, насосів, дисплеїв, мікроконтролерів Arduino та інших компонентів, які використовуються для контролю і моніторингу рівня води.
2. **Моделювання системи**: За допомогою Proteus є можливість створити віртуальну модель системи моніторингу рівня води, де всі компоненти (датчики, реле, насоси, дисплеї) взаємодіють між собою. Це дає змогу перевірити правильність роботи системи та її функціональність перед впровадженням на практиці.
3. **Відлагодження програмного забезпечення:** Proteus дозволяє тестувати програмний код, що відповідає за зчитування даних з датчиків рівня води та керування насосом. Це допомагає знайти та виправити помилки в коді ще до фізичного тестування.
4. **Симуляція роботи контролера:** Proteus забезпечує точну симуляцію поведінки всіх компонентів системи контролю рівня води, таких як аналогові датчики та цифрові елементи керування. Це дає можливість побачити, як система реагуватиме на зміни рівня води і коли вмикатиме або вимикатиме насос.
5. **Мікроконтролерне моделювання:** За допомогою Proteus можна протестувати програмний код для Arduino в реальних умовах симуляції, що дозволить переконатися у правильності роботи всіх функцій перед завантаженням його на фізичну плату.
6. **Симуляція вбудованих систем:** Proteus дозволяє симулювати роботу системи моніторингу води як єдиної інтегрованої системи, що допомагає перевірити її стабільність та ефективність у різних умовах.
7. **Віртуальне відлагодження:** Proteus дозволяє тестувати функції системи та оперативно виявляти проблеми в схемах або коді, забезпечуючи швидке виправлення до того, як система буде зібрана фізично.
8. **Візуалізація результатів:** Програма надає можливість відслідковувати зміну рівня води та реакцію системи на ці зміни в реальному часі, що допомагає оцінити її ефективність і коректність роботи.

Щоб створити проект у Proteus, спочатку запускаємо програму та вибираємо опцію «New project» у головному меню або на панелі інструментів. Далі потрібно назвати свій проект і вибрати місце для збереження файлу. Після цього програма надасть кілька варіантів, таких як встановлення типу друкованої плати (PCB), середовища моделювання та інших параметрів. Якщо ваш проект не вимагає виготовлення друкованих плат (PCB).

Наступним кроком є додавання компонентів на робочу схему. Це робиться за допомогою вікна «Pick Devices» або «Component Mode» де користувач може знайти потрібні компоненти, такі як мікроконтролери, резистори, світлодіоди, реле та інші компоненти. Компоненти можна додати до схеми простим перетягуванням на робочу область.

Після розміщення компонентів важливо з'єднати їх проводами, щоб створити електричну схему. Це робиться за допомогою інструмента «Place Wire», який дозволяє з'єднувати виводи компонентів, формуючи логічні зв'язки.

Якщо в проекті використовується мікроконтролер, потрібно завантажити програмний код у вигляді .HEX або .INO файлу. Код у форматі .INO створюється і редагується в середовищі розробки Arduino IDE, що дозволяє програмувати мікроконтролери Arduino з використанням мови програмування C/C++. Після написання програми, вона компілюється у файл з розширенням .HEX, який містить машинний код для мікроконтролера. У Proteus, файл з розширенням .HEX може бути завантажений безпосередньо в мікроконтролер через опцію 'Program File' в його властивостях. Це дозволяє користувачу симулювати роботу мікроконтролера на етапі проектування, перевіряючи правильність виконання коду в умовах симуляції.

Крім того, такий підхід є особливо корисним, коли потрібно перевірити взаємодію програмного забезпечення з іншими компонентами системи, такими як датчики, приводи або дисплеї, в умовах, максимально наближених до реального середовища.

**Рис. 1.5. Завантаження .HEX файлу на плату**

Останній етап – це запуск симуляції. Користувач натискає на кнопку «Run Simulation», і Proteus починає симуляцію проекту. Під час цього процесу можна спостерігати за поведінкою компонентів і змінювати параметри схеми, що дозволяє протестувати її роботу ще до фізичної реалізації.

**1.4.** **Інтеграція Arduino в Proteus**

Інтеграція Arduino в Proteus стала значним кроком у розвитку середовищ для електронного проектування та симуляції. Історія цього процесу показує, як розробники прагнули об'єднати апаратні та програмні засоби для більш зручного та ефективного створення і тестування електронних систем. Ось основні причини, чому і як відбулася ця інтеграція:

1. **Популярність Arduino**: Arduino став дуже популярним завдяки простоті використання, доступності та великій кількості відкритих ресурсів. Це стимулювало розробників різних програмних середовищ, включаючи Proteus, до адаптації своїх інструментів для підтримки Arduino. Щоб задовольнити потреби зростаючої спільноти розробників, інтеграція Arduino в середовища для моделювання стала логічним кроком.
2. **Симуляція перед фізичною реалізацією**: До інтеграції Arduino в Proteus більшість тестувань відбувалася шляхом завантаження коду на фізичну плату Arduino і підключення всіх компонентів вручну. Це вимагало часу та ресурсів, і виникали ризики пошкодження обладнання або неправильного підключення. Включення Arduino в Proteus дозволило моделювати всю систему віртуально, що значно спростило процес тестування. Розробники могли налагоджувати і виправляти програмний код, перевіряти підключення компонентів і спостерігати за результатами без використання реальної плати.
3. **Розширення можливостей Proteus**: Proteus вже був потужним інструментом для моделювання аналогових і цифрових схем, але йому не вистачало інтеграції популярних мікроконтролерних платформ. Розробники Proteus додали підтримку Arduino як відповідь на попит спільноти. Це дозволило користувачам з легкістю додавати плати Arduino до своїх схем і тестувати їх з іншими компонентами, такими як датчики, мотори і дисплеї, без необхідності збирати фізичну плату.
4. **Інтеграція з Arduino IDE**: Щоб користувачі могли зручно писати код і тестувати його в симуляції, розробники Proteus додали можливість завантаження HEX-файлів (файли з компільованим кодом з Arduino IDE) безпосередньо у віртуальний мікроконтролер Arduino в Proteus. Це зробило процес програмування та тестування більш зручним, оскільки дозволило швидко переносити код з IDE у середовище симуляції.
5. **Розширення бібліотек та компонентів**: Для забезпечення підтримки Arduino в Proteus необхідно було створити відповідні моделі мікроконтролерів, а також додати бібліотеки компонентів. Це зробило можливим підключення різних сенсорів, дисплеїв та інших виконавчих пристроїв у симуляції. Таким чином, інтеграція Arduino перетворила Proteus на універсальне середовище, де можна тестувати як прості, так і складні системи.
6. **Попит від навчальних закладів**: Університети та школи по всьому світу використовують Arduino як навчальний інструмент для вивчення електроніки та програмування. Інтеграція Arduino у Proteus дозволила викладачам і студентам моделювати лабораторні роботи, створювати схеми і тестувати їх без необхідності фізичного обладнання. Це значно скоротило витрати на лабораторне обладнання і підвищило ефективність навчання.

Для інтеграції Arduino та Proteus важливо створити симуляцію, яка дозволяє протестувати апаратну частину та програмне забезпечення до фізичного збирання системи. Це інтегрування дає змогу перевірити правильність коду, підключень і налаштувань віртуально.

Початковий крок - додавання бібліотеки Arduino в Proteus. Існує кілька сторонніх бібліотек, які містять моделі плат Arduino, такі як Arduino Uno, Mega та інші. Після завантаження та встановлення цих бібліотек можна використовувати ці моделі в схемах. На схемі розміщують плату Arduino і з'єднують її з іншими компонентами, як-от сенсори, дисплеї, двигуни, світлодіоди тощо. Після цього програмний код створюється в середовищі Arduino IDE, де можна використовувати стандартні бібліотеки і функції C/C++. Завантаження коду в Proteus виконується шляхом збереження скомпільованого файлу з розширенням .hex в Arduino IDE.

Далі цей файл імпортується в модель Arduino в Proteus, що дозволяє протестувати роботу програми разом із підключеними компонентами. Така інтеграція забезпечує переваги, оскільки розробник може швидко виявляти та виправляти помилки в коді чи підключеннях без необхідності створення реальної схеми. Це особливо корисно для проєктів, де необхідно протестувати кілька компонентів або відлагодити складну логіку керування.

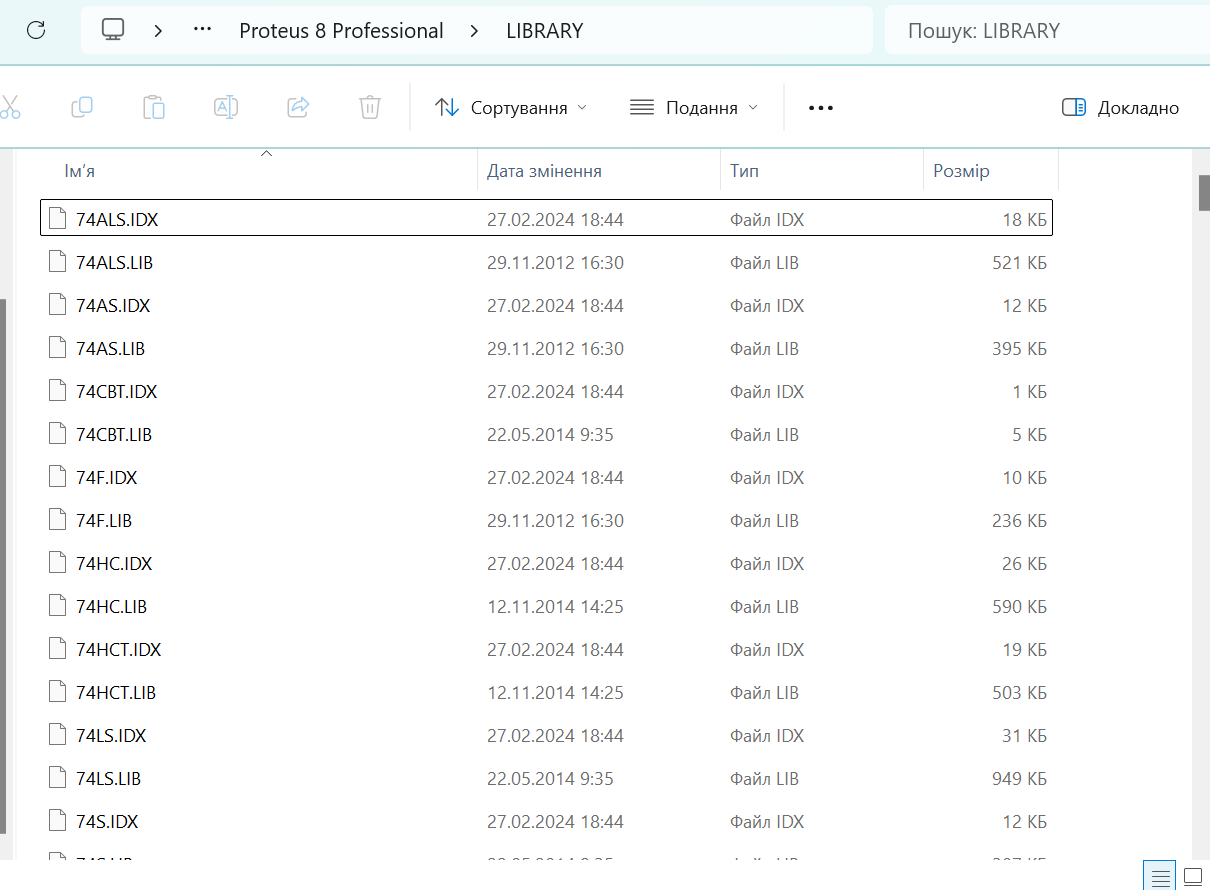
**1.5.** **Інтеграція водяних резервуарів**

В інтернеті існують численні онлайн-репозиторії, де користувачі діляться своїми бібліотеками компонентів для Proteus, включаючи моделі резервуарів для води. Ці бібліотеки зазвичай доступні у вигляді файлів, які можна легко імпортувати до середовища Proteus, використовуючи вбудовані інструменти управління бібліотеками.

Процес імпорту компонентів досить простий: достатньо завантажити бібліотеку, відкрити 'Library Manager' у Proteus, та імпортувати файл. Після цього новий компонент з'явиться у бібліотеці, і його можна буде використовувати у проекті.

**Пошук та імпорт бібліотек для резервуарів води:**

* **Онлайн-репозиторії**: Одне з перших місць, де можна знайти бібліотеку резервуару для води для Proteus — це спільноти в Інтернеті або спеціалізовані репозиторії компонентів Proteus. Ці бібліотеки часто надаються у вигляді файлів .IDX або .LIB, сумісних з Proteus. Сайти, такі як форум користувачів Labcenter Electronics, GitHub або спеціалізовані блоги з Proteus, є чудовими джерелами.
* **Процес імпорту**: Після завантаження необхідного файлу бібліотеки резервуару для води його можна імпортувати в Proteus, виконуючи такі дії:
* Відкрийте Proteus і перейдіть до меню 'Library'.
* Виберіть 'Library Manager' та натисніть 'Import'.
* Знайдіть завантажений файл бібліотеки та оберіть його для імпорту. Новий компонент резервуару для води стане доступним у бібліотеці компонентів у відповідній категорії.



**Рис. 1.6. Корнева папка бібліотек Proteus**

Після додавання необхідного компонента його можна інтегрувати в проект, підключивши до інших елементів, таких як мікроконтролери або датчики рівня води. Це дозволяє симулювати поведінку резервуару, тестувати роботу насосів і взаємодію з іншими елементами системи до її фізичної реалізації.

У випадках, коли потрібного компонента немає в доступних бібліотеках, Proteus також дозволяє створювати власні пристрої з нуля. Використовуючи вбудований редактор схем, можна розробити схему резервуару для води з урахуванням специфічних параметрів та вимог. У проекті можна реалізувати точки підключення для датчиків рівня води, входи та виходи для насосів або клапанів, а також інтегрувати інші елементи, необхідні для роботи резервуару.

Proteus також надає можливість створювати макети друкованих плат, якщо резервуар чи інший компонент потребує апаратного управління. Це може бути корисно для симуляції фізичної реалізації компонента в умовах більшого проекту. Програмування поведінки створеного компонента можна виконати за допомогою блок-схем або за допомогою коду на Arduino, що дозволяє максимально наблизити роботу системи до реальної поведінки обладнання.

Для резервуарів води та інших компонентів можна написати логіку, яка дозволяє управляти їх роботою, наприклад, автоматично заповнювати або зливати воду в залежності від сигналів датчиків рівня. Після написання коду та інтеграції компонентів у проект, користувачі можуть тестувати та калібрувати систему у віртуальному середовищі, щоб впевнитися в її коректній роботі перед фізичним збиранням.

Створення власних компонентів у Proteus має кілька значних переваг.

По-перше, це дає можливість створювати елементи, які повністю відповідають вимогам проекту.

По-друге, це дозволяє налаштувати та симулювати складні системи, такі як багаторівневі системи управління водними ресурсами з декількома резервуарами або промисловими установками. Це також відкриває можливість для масштабування проекту, додаючи додаткові резервуари або насоси без необхідності пошуку додаткових бібліотек.

Proteus дозволяє користувачам створювати власні компоненти за допомогою вбудованого редактора схем. Ця гнучкість є важливою для проектів, які вимагають унікальних специфікацій або якщо існуючий компонент потребує модифікації для відповідності проекту.

* **Проектування схеми**: Першим кроком є створення схеми резервуару, яка може включати основні функції, такі як точки входу та виходу води, з'єднання для датчиків рівня, а також зони для інтеграції насосів або клапанів. Власний компонент має імітувати поведінку реального резервуару настільки точно, наскільки це можливо, щоб забезпечити надійність моделювання.
* **Створення макету друкованої плати (опціонально)**: Якщо схема резервуару включає електронні контролери, створення макету друкованої плати за допомогою режиму 'PCB Layout' у Proteus забезпечує можливість фізичної інтеграції компонента у більш масштабний проект.
* **Програмування поведінки**: Для того щоб резервуар працював правильно в Proteus, його поведінку може бути запрограмовано за допомогою логіки блок-схем або пов'язано з кодом Arduino для симуляції таких операцій, як наповнення, злив або моніторинг рівня на основі сигналів з датчиків.

Таким чином, Proteus стає дуже потужним інструментом для моделювання складних систем, що дозволяє розробникам підходити до проектів з максимальним рівнем індивідуалізації. Це забезпечує більш реалістичне моделювання, зменшуючи ризик помилок під час фізичної реалізації. Використання власних пристроїв резервуарів води та бібліотек у Proteus підвищує гнучкість і дозволяє користувачам створювати індивідуальні рішення, які відповідають їхнім вимогам. Незалежно від того, чи це імпорт наявних бібліотек, чи створення компонентів з нуля, Proteus підтримує широкий спектр можливостей для реалізації складних систем.

**1.6.** **Огляд існуючих систем моніторингу води**

Загалом існує дуже велика кількість систем таких як моніторинг рівня рідини, води, оливи, та ін. Вони використовуються в різних сферах, таких як промисловість, сільське господарство та звичайно ж будинки. Ці системи базуються на різних принципах роботи та технологіях, починаючи від механічних і закінчуючи повністю цифровими та керованими програмним забезпеченням.

**Механічні системи** використовують прості принципи, такі як поплавці або противаги, які реагують на зміни рівня рідини в резервуарі та активують механічні важелі для запуску або зупинки насоса. Ці системи зазвичай не вимагають складного апаратного чи програмного забезпечення, але мають обмежену точність і функціональність.

**Електронні системи** на основі датчиків (ультразвукових, ємнісних, резистивних) забезпечують більшу точність. Вони використовують такі мікроконтролери, як Arduino або Raspberry Pi(маленький, недорогий одноплатний комп'ютер, розроблений для навчання), для обробки сигналів і контролю рівня рідини. Ці системи можуть підключатися до інших пристроїв через бездротові протоколи (Wi-Fi, Bluetooth) і передавати дані на смартфон або комп’ютер для моніторингу в реальному часі через спеціальні програми.

**Ультразвукові** датчики визначають рівень рідини, вимірюючи час, необхідний для відбиття звукових хвиль від поверхні рідини. **Ємнісні та резистивні системи** працюють шляхом зміни електричних сигналів при зміні рівня рідини.

Сучасні системи IoT дозволяють дистанційно контролювати рівень рідин за допомогою мобільних додатків. Такі системи можна інтегрувати з хмарними сервісами, дозволяючи користувачам отримувати доступ до даних з будь-якої точки світу. Наприклад, модуль Wi-Fi передає дані від датчика на сервер або смартфон через роутер.

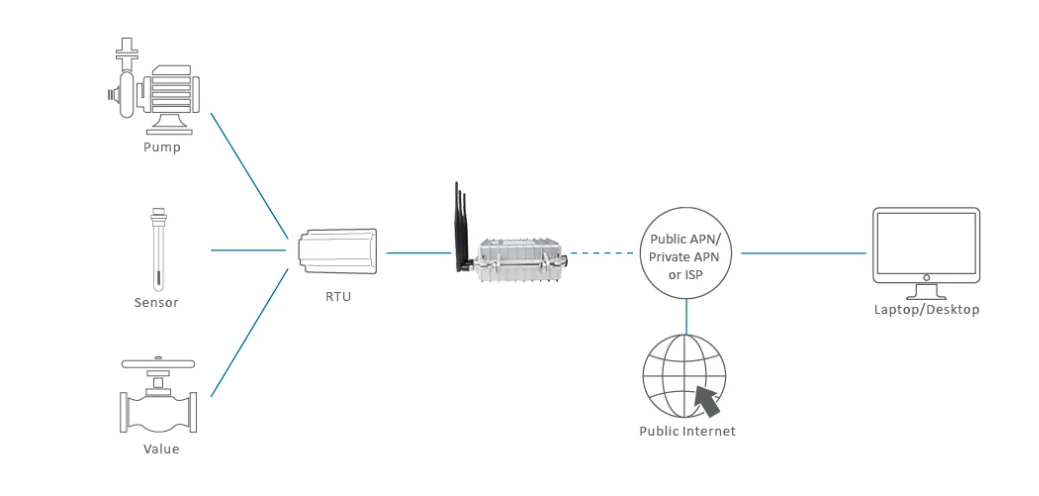
Самі системи можуть бути написані на різних мовах програмування, таких як C++, Python або спеціалізованих мовах для вбудованих систем. Під час використання додатків або платформ онлайн-моніторингу користувачам іноді потрібно завантажити додаткові програми для налаштування та моніторингу системи в режимі реального часу.

**Висновки до розділу 1**

1. У розділі було розглянуто основні компоненти та технології, що складають основу комп'ютерної системи моніторингу та контролю рівня води на базі Arduino.
2. Описано Arduino IDE як основне середовище розробки для програмування мікроконтролера, а також мову програмування C/C++, яка використовується для написання коду.
3. Досліджено можливості програмного забезпечення Proteus для моделювання та симуляції електронних схем, що дозволяє перевірити роботу системи без фізичного збирання прототипу.
4. Розглянуто інтеграцію Arduino в середовищі Proteus, що забезпечує можливість тестування програмного коду та системи в цілому.
5. Описано інтеграцію водяних резервуарів у Proteus, включаючи використання готових бібліотек компонентів та створення власних різноманітних пристроїв, що дозволяє адаптувати систему під специфічні вимоги проєкту.
6. Розглянуто існуючі системи моніторингу, які базуються на різних принципах роботи, включаючи механічні та електронні рішення. Були проаналізовані механічні системи, що використовують прості поплавкові механізми, а також сучасні електронні системи, оснащені датчиками (ультразвуковими, ємнісними, резистивними).**РОЗДІЛ 2.** **АНАЛІЗ СИСТЕМ МОНІТОРИНГУ**

**2.1. Архітектура та функціонельність систем моніторингу**

**1.** Smart Water Quality Management (SWQM) — це сучасне рішення Інтернету речей (IoT) для моніторингу якості води. Така система зазвичай використовує мережу датчиків для миттєвого вимірювання різних параметрів якості води, таких як pH(показник, який вказує на кислотність або лужність водного розчину), каламутність, розчинений кисень, температура та наявність забруднюючих речовин. Дані датчиків передаються в центральну систему та аналізуються за допомогою складних алгоритмів, включаючи моделі машинного навчання.



**Рис. 2.1. Система дистанційного керування та моніторингу водопостачання**

Насоси відповідають за перекачування води, а датчики контролюють важливі параметри, такі як рівень або тиск води. Клапан контролює потік води, регулюючи відкриття і закриття системи водопостачання.

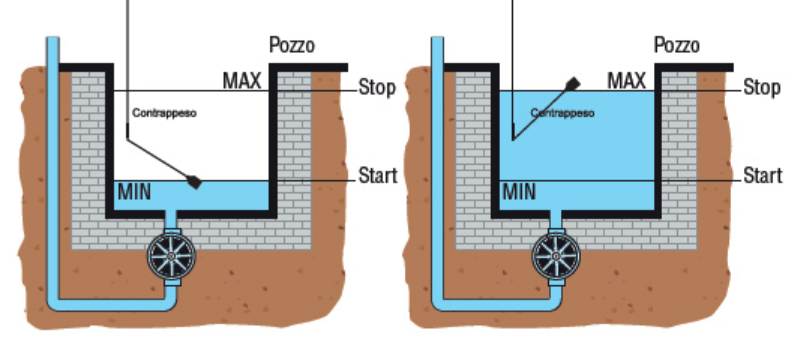
Основним компонентом є блок дистанційного керування (RTU), який збирає дані від датчиків і передає команди до насосів і клапанів для автоматизації процесів. RTU (Remote Terminal Unit) - це електронний пристрій, який використовується для дистанційного моніторингу та керування різними системами.

RTU забезпечує дистанційне керування та моніторинг системи шляхом підключення до мережі через загальнодоступну або приватну точку доступу (APN) або через Інтернет-провайдера. Оператори можуть отримувати доступ до даних і керувати всією системою через комп’ютер або ноутбук із підключенням до Інтернету, надаючи їм повний контроль над водопостачанням з будь-якого місця.

Такі системи широко використовуються в різних сферах, таких як міське водопостачання, очищення промислових стічних вод, сільське господарство та аквакультура. Вони забезпечують безпечну питну воду, оптимізують процеси очищення стічних вод, контролюють сільськогосподарське зрошення.

**2.** Механічні системи з поплавковими датчиками є простим і надійним рішенням для контролю рівня бака. Поплавкові датчики використовуються в різних галузях промисловості, таких як водопостачання, сільське господарство та промислові процеси.

Принцип роботи поплавкового датчика заснований на використанні поплавця, який плаває на поверхні рідини. Коли рівень рідини змінюється, поплавок піднімається або опускається, викликаючи спрацьовування перемикача або магнітного контакту.



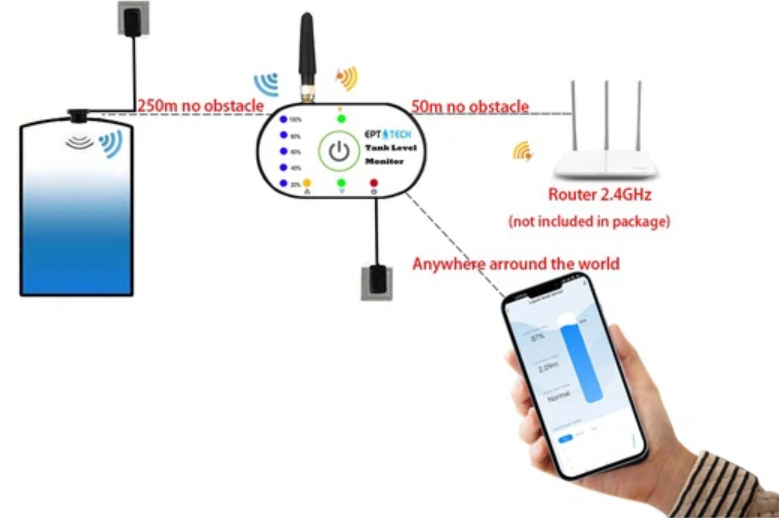
**Рис. 2.2. Робота механічної системи з поплавковим датчиком**

Коли поплавок досягає певного рівня води, наприклад високого або низького, він активує або вимикає електричні контакти, надсилаючи сигнали про стан рідини. Цей сигнал можна використовувати для ввімкнення або вимкнення насосів, клапанів або інших систем керування.

Варіантів даного типу датчиків багато, включаючи вертикальні і горизонтальні моделі. Вертикальний датчик встановлений у верхній частині резервуара і працює на основі зміни положення поплавця вздовж вертикальної осі, а горизонтальний датчик встановлений збоку від резервуара і спрацьовує, коли рідина збоку досягає певної позначки. рівень.

**3.** Ультразвукові датчики широко використовуються в системах контролю рівня води завдяки своїй точності та простоті використання. Ці системи працюють шляхом вимірювання часу, який потрібен звуковій хвилі, щоб досягти поверхні води та повернутися до приймача датчика.

У контексті мониторингу рівня води, ультразвукові датчики можна використовувати у великих промислових резервуарах для води та побутових системах водопостачання або для контролю рівня води в акваріумах.

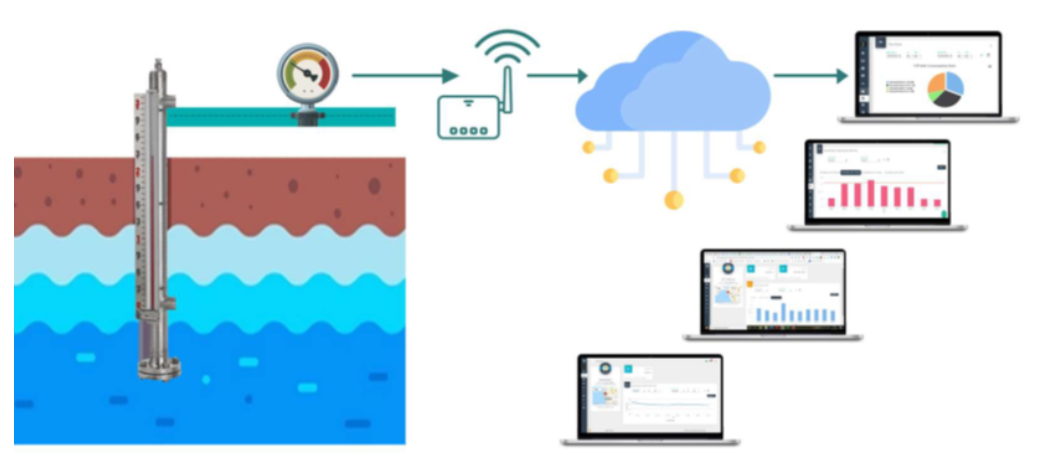
Їх можна підключити до мікроконтролера, наприклад Arduino, що дозволяє автоматизованим системам керувати насосами та іншим обладнанням на основі рівня води.

**Рис. 2.3. Система бездротового моніторингу рівня води на основі ультразвукових датчиків**

Як на прикладі малюнка, це бездротова система моніторингу рівня води на основі ультразвукових датчиків. Основні компоненти системи включають ультразвуковий датчик, який встановлено на верхній частині бака і використовує звукові хвилі для вимірювання відстані до води. Ці дані передаються на блок моніторингу рівня води, який має індикатор, який візуально відображає рівень води у відсотках від 100% до 20%.

Модуль підключається до мережі Wi-Fi через маршрутизатор, що дозволяє передавати дані на смартфон користувача. Додаток на смартфоні відображає поточний рівень води в резервуарі, що дозволяє стежити за ситуацією з будь-якої відстані.

4. Резистивна система, вона ж використовується для вимірювання рівня води, заснована на принципі зміни опору в залежності від рівня рідини в баку. У таких системах використовуються два або більше електродів, які занурені у воду на різну висоту. Коли вода досягає певного рівня, замикання ланцюга між електродами викликає зміну опору, за якою можна визначити кількість рідини.



**Рис. 2.4. Робота механічної системи з поплавковим датчиком**

На цьому рисунку показано систему моніторингу рівня води, яка використовує цифрові манометри, підключені до бездротового маршрутизатора, для передачі даних на хмарний сервер. Потім ці дані потрапляють на веб-інтерфейси на різних пристроях, таких як ноутбуки чи інші комп’ютери.

Отже, резистивні датчики працюють на основі провідності води. Вода є провідником електрики, і коли вона заповнює простір між електродами, струм починає проходити через воду, змінюючи резистивність системи. Відповідно, чим більше води між електродами, тим більша зміна опору фіксується, що дозволяє визначити рівень води.

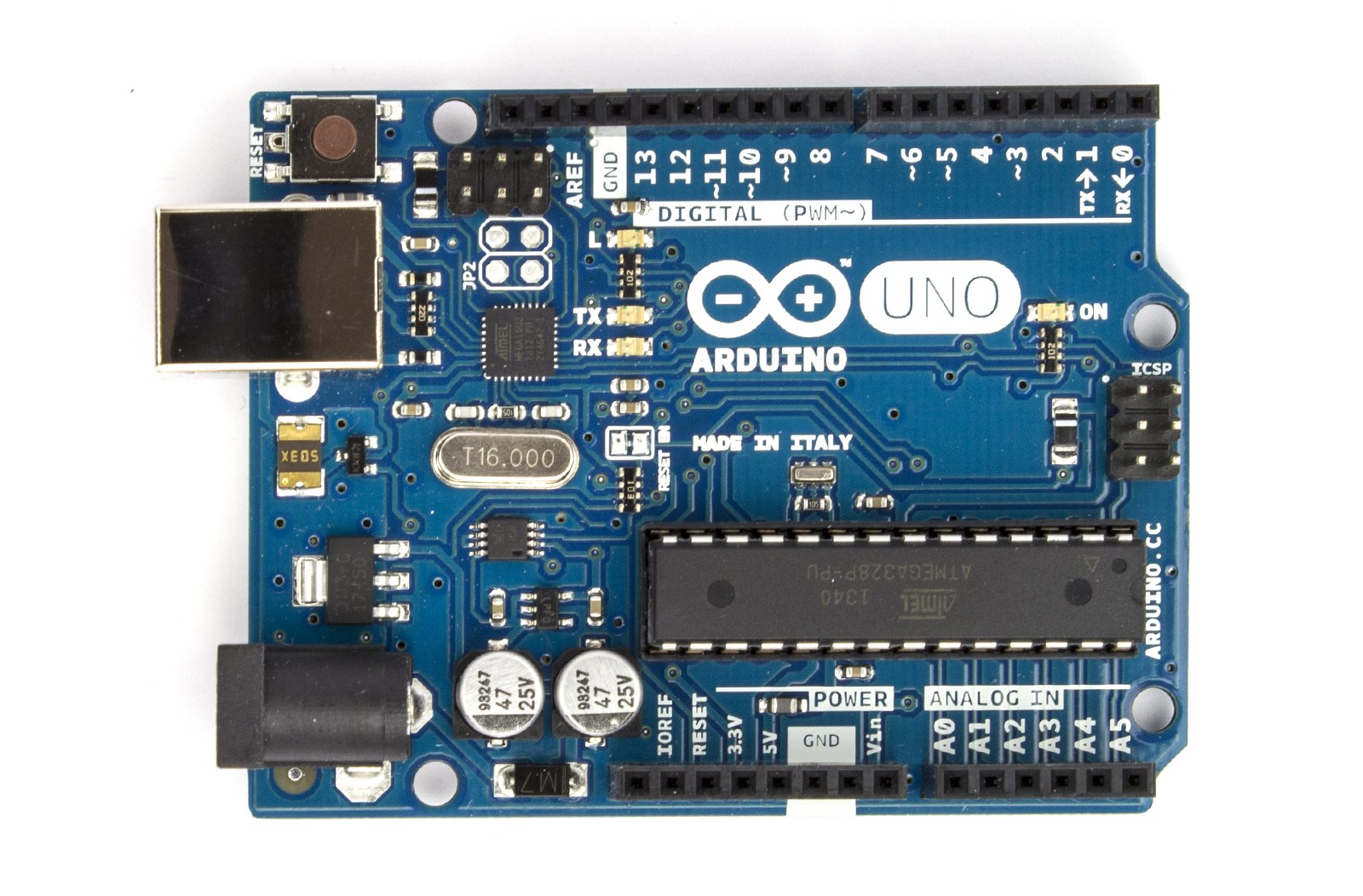
5. Ємнісні системи моніторингу рівня рідини використовують принцип зміни ємності між двома електродами в залежності від рівня рідини між ними. Основна ідея полягає в тому, що ємність конденсатора збільшується або зменшується в залежності від того, наскільки рідина заповнює простір між електродами.

Цей тип сенсорів особливо ефективний для моніторингу рівня води, оливи та інших рідин, оскільки рідини мають діелектричні властивості, які впливають на електричну ємність.

**2.2. Комп’ютерна система моніторингу води на Arduino UNO**

Вибір саме плати Arduino Uno для проекту моніторингу води базується на кількох важливих факторах, які забезпечують її ефективність і універсальність порівняно з іншими мікроконтролерами.

Основним фактором є його простота та використання, Arduino Uno має простий інтерфейс, а його конфігурація не вимагає спеціальних знань електроніки чи програмування. Це робить плату простою у використанні навіть для початківців, що особливо важливо для навчальних проектів або швидкого прототипування. Простота Arduino IDE також полегшує розробку коду та завантаження його на плату.



**Рис. 2.5. Arduino UNO**

Проект моніторингу водних об’єктів передбачає використання різноманітних компонентів, таких як датчики рівня води, реле керування водяними насосами, світлодіодні індикатори та інформаційні дисплеї віртуальних терміналів. Arduino Uno має достатню кількість аналогових і цифрових контактів для підключення всіх необхідних компонентів, що дозволяє ефективно керувати датчиками і виконавчими механізмами.

Характеристики плати:

1. 14 цифрових входів/виходів: використовуються для керування реле, LED-індикаторами, ультразвуковими датчиками і двигунами.
2. 6 аналогових входів: дозволяють вимірювати напругу, що подається з аналогових датчиків.
3. Вбудований USB-порт: забезпечує живлення та програмування плати за допомогою комп'ютера.
4. Живлення: плата може живитися від USB або зовнішнього джерела (7-12 В), що робить її універсальною для автономної роботи.

Arduino Uno є відносно доступною та малопотужною платою, що робить її чудовим вибором для проектів, орієнтованих на збереження ресурсів. Він забезпечує функціональність, необхідну для моніторингу води без надмірного споживання енергії та мінімальних фінансових витрат.

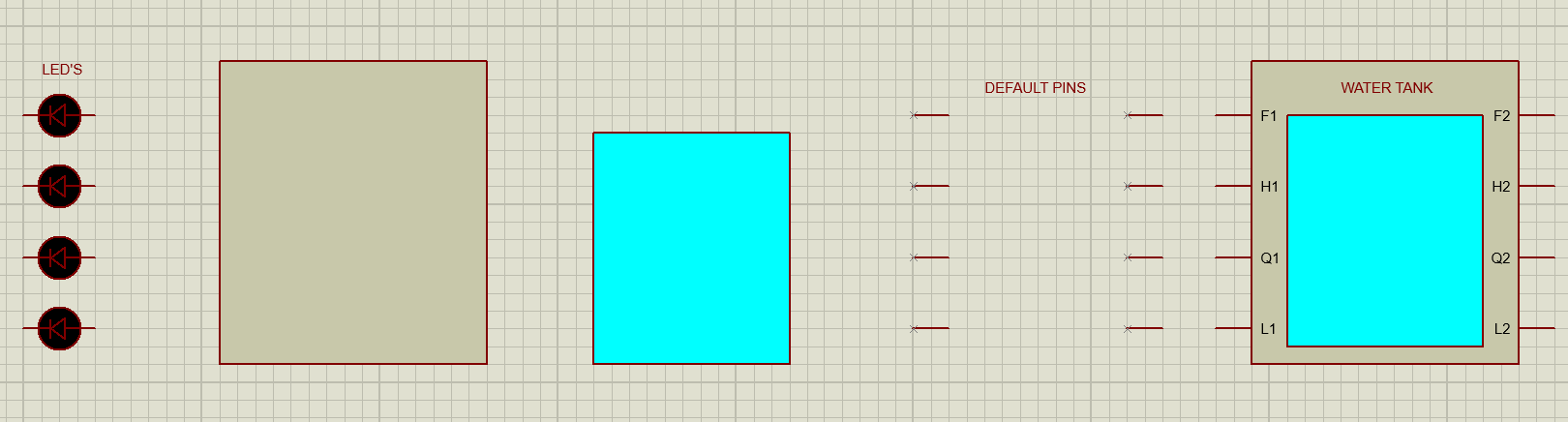
Arduino як програмне забезпечення було і є дуже популярною платформою, і існує велика спільнота користувачів, які створюють проекти, діляться кодом і надають технічну підтримку. Завдяки цьому можна знайти багато прикладів та інструкцій, які допомагають розібратися з нюансами реалізації проектів, схожих на систему моніторингу води. Така підтримка спільноти робить процес розробки більш ефективним та дуже економить час.

Також для роботи з різними датчиками та компонентами існує велика кількість готових бібліотек в Arduino IDE, що значно спрощує їх інтеграцію в проект. Зазвичай це ультразвукові сенсори або модулі для відображення інформації на LCD екрані.

Arduino IDE використовує спрощену версію мови C/C++, що дозволяє швидко створювати код без надмірних складнощів, властивих традиційному програмуванню мікроконтролерів. Прості команди, такі як *digitalWrite()* для керування цифровими виходами, *analogRead()* для зчитування аналогових значень, або *Serial.print()* для виводу інформації на віртуальний термінал, роблять процес написання коду швидким та інтуїтивним.

**2.3** **Метод вимірювання та пристрої**

Методом вимірювання води зазвичай використовують **ультразвукові датчики** (наприклад, **HC-SR04**), які працюють за принципом ехолокації. Датчик випромінює ультразвуковий сигнал, який відбивається від поверхні води та повертається до датчика. За часом, який потрібен для повернення сигналу, обчислюється відстань до поверхні води, а звідси - і її рівень, але система моніторингу води базується на створенному пристрої та завантаженим аналогом, метод вимірювання рівня води дещо відрізняється від стандартних підходів, як-от використання ультразвукових чи аналогових датчиків.



**Рис. 2.6. Компоненти пристрою**

**Опис компонентів**

1. Світлодіоди (LED's): Чотири світлодіоди, розташовані зліва, використовуються для індикації рівня заповненості цистерни. Кожен світлодіод відповідає за певний рівень:

FULL: Позначає повний рівень води у цистерні.

1/4: Позначає, що вода досягла четвертої частини цистерни.

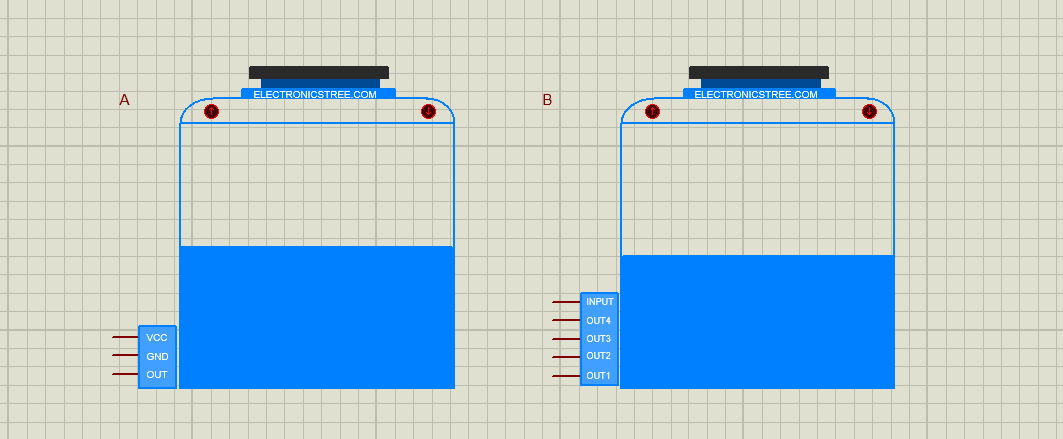
HALF: Позначає, що цистерна заповнена наполовину.

LOW: Позначає, що вода на низькому рівні.

1. Бежевий квадрат — це компонент, який представляє конструкцію самої цистерни для води. Він служить "контейнером" для води і має контактні точки для підключення до електричних елементів схеми. Через ці контакти можна зчитувати різні рівні заповненості, зокрема низький, середній і високий рівні, а також повний обсяг цистерни.
2. Блакитний квадрат — це індикатор рівня води в цистерні. Він змінюється відповідно до кількості води в системі. Коли вода додається або зменшується, блакитний квадрат збільшується або зменшується, візуально показуючи поточний обсяг води. Це дозволяє спостерігати за рівнем заповненості цистерни у реальному часі.
3. Цистерна для води (Water Tank): Великий прямокутник праворуч з мітками для вимірювання рівня води. Вона має кілька контактів (позначені як F1, F2, H1, H2, Q1, Q2, L1, L2), які відповідають за зчитування рівнів води через сенсорні точки або підключення до інших елементів системи.
4. Піни (Default Pins): Ряд стандартних контактів для з'єднання з іншими елементами схеми, такими як контролер або додаткові сенсори.

Однак, у стандартній бібліотеці Proteus відсутні спеціалізовані моделі для сенсорів або датчиків рівня води, що обмежує можливості симуляції подібних систем. Щоб вирішити цю проблему, були створені нові моделі резервуарів з датчиками рівня, які спеціально розроблені для моніторингу рівня води. Ці моделі можуть використовуватися не тільки для контролю води, але й для інших проєктів, де потрібно відстежувати рівень речовин.

У запропонованій бібліотеці від **ElectronicsTree** представлено дві моделі резервуарів: одну цифрову та одну аналогову. Обидві моделі легко взаємодіють із будь-яким керуючим обладнанням у середовищі Proteus. Завдяки своїй простоті вони можуть використовуватися без підключення до Arduino, що робить їх універсальними та легкими для інтеграції у різноманітні проєкти. Ці резервуари можна застосовувати як датчики рівня води, що робить їх ефективними для контролю рівня рідини у більшості домашніх та навчальних проєктів.



**Рис. 2.7. Моделі резервуарів А та Б**

**Опис компонентів**

Цистерна **A:**

Контактні пін-контакти: Містить три пін-контакти: VCC, GND і OUT.

VCC (живлення) використовується для подачі напруги.

GND підключається до землі.

OUT використовується для зчитування даних про рівень води.

Світлодіоди: Вгорі цистерни розміщені два червоні світлодіоди, що можуть використовуватись для індикації статусу (наприклад, повного або мінімального рівня води).

Рівень води: Блакитний рівень води на схемі показує поточний стан заповнення. Цей рівень можна регулювати, і він відображає, скільки води в цистерні.

Цистерна **B:**

Більша кількість виходів: Має п'ять пін-контактів: один для вводу (INPUT) і чотири для виходу (OUT1, OUT2, OUT3, OUT4).

INPUT використовується для подачі сигналу керування.

OUT1-OUT4 можуть використовуватися для детектування рівня води на різних етапах заповнення (від найнижчого до найвищого рівня), таким чином, надаючи більш точну інформацію про заповненість.

Світлодіоди: Подібно до першої цистерни, B варіант також має два червоні світлодіоди, які сигналізують про рівень води або інший важливий статус.

Рівень води: На схемі вказаний поточний рівень заповнення цистерни з боку користувача.

**2.4** **Порівняння систем з проектом**

Сучасні системи моніторингу рівня та якості води використовують різні технології, включаючи механічні системи з поплавковими датчиками, ультразвукові датчики, резистивні та ємнісні системи. Кожен метод має свої переваги та обмеження залежно від умов використання та ступеня автоматизації та точності.

1. Розумне управління якістю води (SWQM)

Системи Smart Water Quality Management (SWQM) є однією з передових технологій, яка використовує датчики для вимірювання різних показників якості води: рН, вміст розчиненого кисню, електропровідність тощо. SWQM є високотехнологічним і може дистанційно контролювати параметри води в режимі реального часу, що дуже важливо для міських систем водопостачання, а також моніторингу навколишнього середовища. Головною перевагою такої системи є можливість зберігати та аналізувати дані через інтернет та інтегрувати їх в інші інформаційні системи. Однак висока вартість і складність реалізації можуть бути шкідливими для невеликих проектів.

У порівнянні з моделлю Arduino, яка зосереджена на простому моніторингу рівня води, SWQM пропонує набагато більше функцій. Система Arduino менш складна та економічна, що робить її більш придатною для невеликих та особистих проектів.

2. Механічна система з датчиком поплавка

Механічні системи з поплавковими датчиками є одними з найстаріших і найпростіших методів контролю рівня води. У системі цього типу поплавок піднімається або опускається разом із рівнем води, активуючи механічний контакт, який сигналізує про досягнення певного рівня води. Вони широко використовуються в резервуарах для води, колодязях та інших простих системах, що не вимагають високої точності.

Основними перевагами поплавкових датчиків є простота і дешевизна, але точність таких систем залишає бажати кращого. Крім того, механічні частини з часом зношуються. Системи Arduino, змодельовані в Proteus, більш точні та не мають рухомих механічних частин, що робить їх надійнішими та довговічнішими, ніж механічні системи.

3. Ультразвуковий датчик

Ультразвукові датчики безконтактно вимірюють рівень рідини, випромінюючи ультразвукові хвилі та вимірюючи час, потрібний для повернення ультразвукових хвиль після відбиття від поверхні рідини. Ці датчики є високоточними та підходять для промислових середовищ, де необхідний безконтактний моніторинг. Основними перевагами ультразвукових датчиків є точність і можливість використання в жорстких умовах. Однак їх висока вартість і складність налаштування можуть бути перешкодою для домашніх проектів або невеликих систем.

Порівняно з ультразвуковими датчиками систему Arduino простіше проектувати, оскільки вона використовує аналогові та цифрові датчики для контролю рівня, що призводить до нижчої вартості та легшого налаштування. Однак ультразвукові датчики більш точні, особливо у великих резервуарах.

4. Резистивна система

Резистивні системи використовують кілька металевих зондів для вимірювання рівня води. Ці зонди занурюють у воду та вимірюють зміну опору як функцію рівня рідини. Резистивні системи зазвичай використовуються в домашніх резервуарах через їх низьку вартість і простоту використання, але вони можуть мати недоліки в суворих умовах, які можуть призвести до корозії зонда.

Перевага системи Arduino для імітації контролю рівня води в Proteus полягає в тому, що її можна налаштувати для роботи з різними датчиками, включаючи резистивні. Arduino також пропонує гнучкість програмного забезпечення, що дозволяє адаптуватися до різних умов і уникнути проблем з корозією шляхом вибору матеріалів або додавання додаткових функцій захисту.

5. Ємнісна система моніторингу

Ємнісні системи використовують зміни в електричному полі для визначення рівня рідини. Коли рівень води змінюється, ємність між датчиком і рідиною також змінюється, що дозволяє безконтактно вимірювати рівень води. Такі системи є високоточними та надійними, особливо в суворих умовах, де потрібна герметизація.

Система Arduino є більш економічною (бюджетною) альтернативою і може використовувати ємнісні датчики, якщо це необхідно, але головними перевагами є інтеграція з іншими датчиками та гнучкість налаштування параметрів відповідно до вимог проекту. На відміну від спеціалізованих ємнісних систем, наша система на основі Arduino є універсальною платформою, яка дозволяє користувачеві адаптувати систему до різних умов використання.

**Висновки до розділу 2**

1. Розглянута архітектура систем моніторингу води та їхні основні компоненти показали, що успішна система моніторингу повинна поєднувати датчики, контролери, програмне забезпечення та інтерфейси для надання точних даних користувачеві.
2. Комп'ютерна система моніторингу води, розроблена на базі Arduino UNO, була обрана як оптимальна платформа завдяки своїй простоті, гнучкості, доступності та можливості підключення широкого спектру сенсорів для вимірювання рівня води. Arduino IDE спрощує процес програмування та налагодження, а модель в Proteus дозволяє симулювати роботу системи перед її фізичним створенням.
3. Розглянуто різні методи вимірювання рівня води, такі як ультразвукові, резистивні, ємнісні та механічні системи з поплавковими датчиками. Кожен метод має свої особливості, проте вибір був зроблений на користь Arduino з використанням цифрових і аналогових сенсорів, оскільки цей метод забезпечує достатню точність і є економічно вигідним.
4. Порівняння показало, що система на Arduino є оптимальною для побутових і малих комерційних проєктів завдяки низькій вартості та простоті у використанні. Вона поступається промисловим системам за точністю та витривалістю, але водночас є більш доступною для персональних проєктів і легко адаптується під різні вимоги.

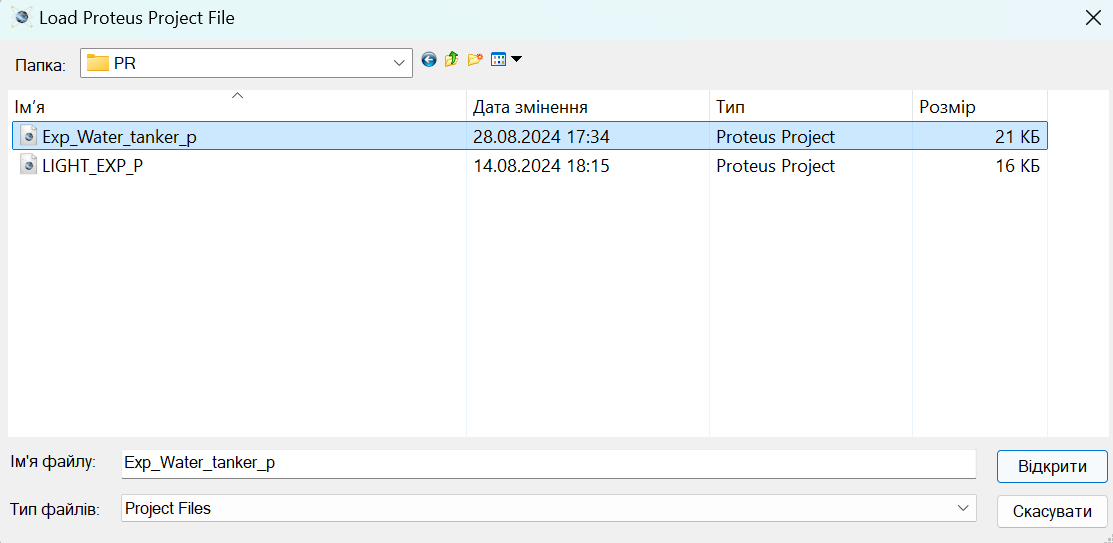
**РОЗДІЛ 3. СТВОРЕННЯ СХЕМИ ТА ЇЇ СИМУЛЯЦІЯ**

**3.1.** **Створення пристрою або датчика**

Створення пристрою або датчика в Proteus - це складний, але цікавий процес, що включає кілька основних кроків, від налаштування компонентів до їх інтеграції та програмування.

Докладний опис створення дивайсу "water\_tank".

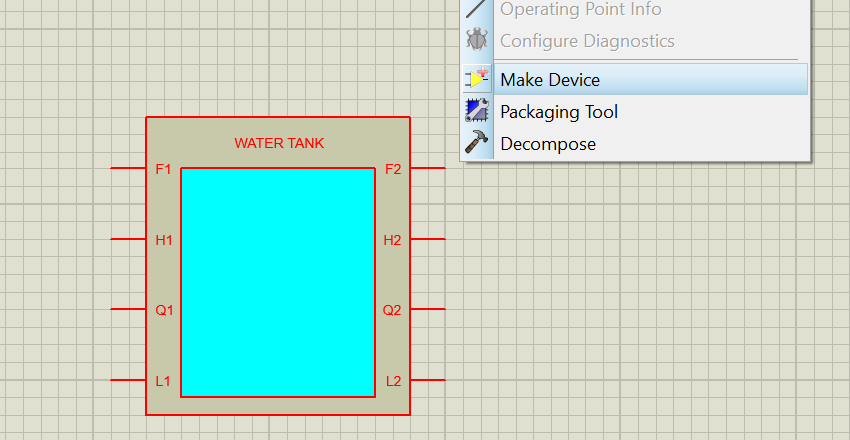
Перш за все, потрібно відкрити програму Proteus та створити новий проект або відкрити вже існуючий.



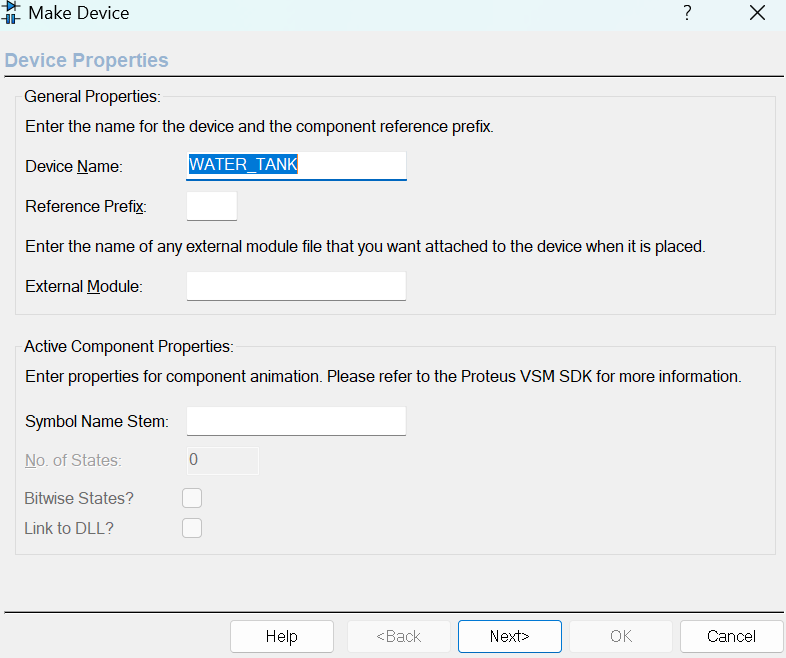
**Рис. 3.1. Завантаження створенного проекту**

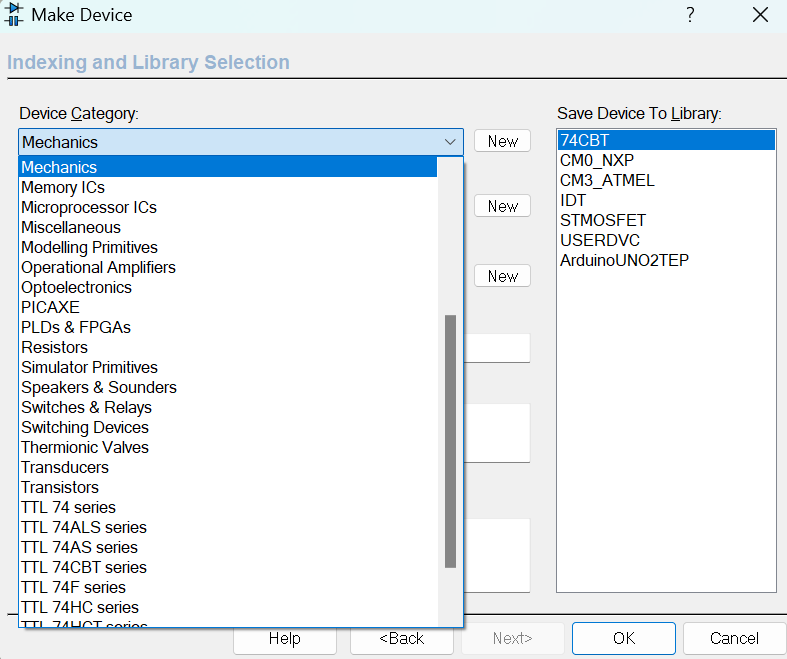
Наступний крок - зайти у вкладку "Make Device", що дозволяє створити новий пристрій або компонент. У вікні "Make Device" необхідно визначити основні параметри майбутнього пристрою, такі як ім'я, тип пристрою (наприклад, датчик або виконавчий механізм), а також вибрати основні елементи, що будуть входити до його складу.

Після визначення параметрів слід розмістити необхідні компоненти на схемі. У випадку з "water\_tank" це можуть бути контакти, що позначають рівень води, або електричні з'єднання для живлення. Важливо забезпечити правильне розміщення та підключення всіх компонентів, оскільки це вплине на функціональність пристрою.



**Рис. 3.2. Створення пристрою за допомогою Make Device**

******Рис. 3.3. Налаштування властивостей пристрою**

****

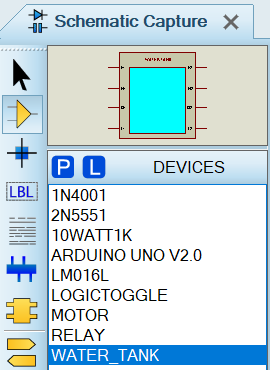
**Рис. 3.4. Збереження пристрою до бібліотеки**

Вибір параметра "mechanics" під час створення пристрою в Proteus означає, що пристрій буде належати до категорії механічних компонентів. Цей параметр використовується для компонентів, які представляють собою механічні або фізичні частини системи, що можуть включати резервуари, клапани, насоси тощо. Це дозволяє коректно відобразити властивості та поведінку пристрою у схемі, щоб він був інтегрований як частина системи, яка взаємодіє з іншими компонентами (наприклад, електронними датчиками чи контролерами).

Потім необхідно перейти до налаштування властивостей компонентів та їх програмування. Тут потрібно використовувати дочірні вікна для кожного з компонентів, що входять до складу пристрою. У цих вікнах можна задавати значення параметрів, таких як рівень напруги або сила струму, а також програмувати поведінку пристрою за допомогою скриптів або використовуючи спеціалізовані налаштування.

Далі потрібно зібрати всі компоненти в єдиний пристрій. У "Make Device" є можливість створити бібліотеку з користувацьким пристроєм, де можна об'єднати всі налаштовані компоненти в один. Це робиться за допомогою функції "Create Child Device", що дозволяє інтегрувати підкомпоненти пристрою у зручну і компактну структуру.

Після того як пристрій зібраний і запрограмований, його необхідно протестувати. Для цього можна використовувати середовище симуляції в Proteus, що дозволяє перевірити роботу пристрою без фізичної реалізації.



**Рис. 3.5. Поява створеного пристрою на панелі**

Коли симуляція завершена, пристрій можна зберегти як окремий елемент бібліотеки Proteus, і він буде доступний для подальшого використання у будь-яких проектах.

Альтернативно, замість створення датчика або пристрою (з нуля), можна скористатися готовими бібліотеками та моделями, доступними в Інтернеті. Багато сайтів та ресурсів надають бібліотеки у форматах, сумісних із Proteus, вони ж, .lib, .idx, або файли прошивки у форматі .hex.

Завантаживши потрібний файл бібліотеки або модель (Формату .LML) пристрою, його можна легко додати до бібліотеки Proteus.



**Рис. 3.6. Поява встановленого пристрою в Proteus**

Для цього потрібно скопіювати файли до каталогу бібліотек, який зазвичай розташований у директорії встановлення програми Proteus (C:\Program Files\Labcenter Electronics\Proteus 8 Professional\LIBRARY та C:\Program Files\Labcenter Electronics\Proteus 8 Professional\MODELS). Після цього новий пристрій стане доступним у списку компонентів, і його можна буде використовувати в будь-якому проекті, що значно полегшує процес інтеграції стандартних або популярних компонентів, таких як водяні датчики, реле, або інші елементи.

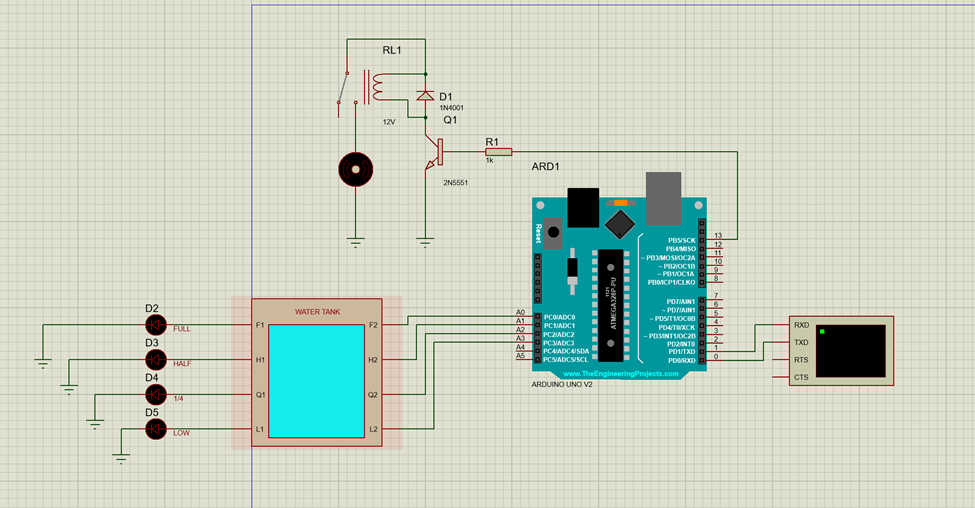
**3.2. Створення та принцип роботи схеми в Proteus**

Було розроблено дві схеми: для першої був створений власний пристрій, який дозволив глибоко вивчити принципи роботи електронних компонентів та реалізувати оригінальні алгоритми управління, та схему на основі готового рішення, що демонструє можливості швидкого прототипування та використання комерційних модулів. Для моделювання та симуляції схем було використано середовище Proteus, що забезпечило віртуальне тестування систем перед їх фізичною реалізацією.

У першій схемі детально описаний власний пристрій та компоненти, включаючи вибір оптимальних датчиків рівня води, підбір елементів.

У Другій схемі були використані готові компоненти з бібліотеки Electronicstree.

**3.2.1.** **Створення схеми з використанням власного пристрою**

****

**Рис. 3.7. Варіант схеми зі свтореним пристроєм**

Цей варіант проекту базується на використанні власного пристрою-датчика, створеного спеціально для моніторингу рівня води в резервуарі. Даний пристрій здатний точно визначати різні рівні заповненості резервуару, а також контролювати роботу ротора. У схемі використовується Arduino UNO як головний контролер, до якого підключено кілька елементів, серед яких світлодіоди, реле, транзистори, та власноруч створений датчик води, що представляє собою повністю інтегрований компонент системи.

Основне призначення цього датчика полягає в тому, щоб забезпечувати точне визначення рівня води та відповідним чином реагувати на зміни рівня. Датчик, в залежності від рівня води в резервуарі, посилає сигнал на контролер, який керує увімкненням або вимкненням ротора (насоса), щоб автоматично наповнювати або спустошувати резервуар. Крім того, за допомогою віртуального терміналу користувач може слідкувати за статусом системи та керувати нею.

Опис датчика:

1. Датчик LOW (Низький рівень води) – цей датчик відповідає за визначення мінімального рівня води в резервуарі. Коли вода досягає цього рівня, сигнал з датчика передається на Arduino, і система реагує на критичне зниження рівня води. За його допомогою контролюється робота насоса, щоб уникнути сухого ходу, коли резервуар майже порожній.
2. Датчик 1/4 (Чверть резервуару) – цей датчик вказує на рівень, коли резервуар заповнений на 25%. Він використовується для більш точного моніторингу рівня води, щоб вчасно активувати або деактивувати насос. При досягненні цього рівня система може забезпечувати автоматичне регулювання подачі води.
3. Датчик HALF (Половина резервуару) – відповідає за відстеження середнього рівня води. Датчик реагує на рівень, коли резервуар заповнений на 50%, що дозволяє зменшити кількість циклів увімкнення та вимкнення насоса.
4. Датчик FULL (Повний резервуар) – цей датчик сигналізує про те, що резервуар повністю заповнений. Він необхідний для автоматичного відключення насоса, коли рівень води досягає максимального значення. Така функціональність запобігає переповненню резервуару і можливому витоку води.

Кожен із цих датчиків підключений до відповідного світлодіода (D2 для FULL, D3 для HALF, D4 для 1/4, D5 для LOW), який візуально відображає рівень води. Це надає оператору простий і зрозумілий інтерфейс для контролю за станом системи, хоча дізнатися стан можна ще у віртуальному терміналі, якщо світлодіоди вийдуть з ладу або просто не працюють.

Загальний опис схеми:

1. Реле (RL1) – електромеханічний перемикач, який використовується для автоматичного включення та вимкнення ротора (або насоса). У проекті реле підключено до транзистора та контролюється сигналом від Arduino. Це реле забезпечує можливість комутації більшої потужності (в нашому випадку ротор на 12В), чим та, яку може надати сам Arduino.
2. Транзистор Q1 (2N5551) – використовується як електронний перемикач для керування реле. Транзистор підсилює сигнал з Arduino, що дозволяє живити реле необхідним струмом і вмикати насос, коли це потрібно. Підключений до реле, транзистор забезпечує управління безпосередньо через виходи мікроконтролера.
3. Діод D1 (1N4001) – захисний елемент, який використовується для захисту транзистора та інших елементів схеми від індуктивного напругового стрибка, що виникає при розриві кола реле. Це важливо для захисту схеми від пошкоджень.
4. Ротор (насос) – у схемі це механічний елемент, який відповідає за перекачування води між резервуарами або підтримку певного рівня води. Ротор підключений через реле, тому його робота керується Arduino на основі даних від датчиків рівня води.
5. Світлодіоди (D2, D3, D4, D5) – кожен світлодіод відповідає за індикацію певного рівня води в резервуарі. Світлодіоди підключені до контролера, і в залежності від рівня води вони вмикаються, щоб наочно продемонструвати поточний стан резервуару: від "низького" до "повного".
6. Віртуальний термінал – у проекті це інтерфейс, за допомогою якого користувач може спостерігати та керувати роботою системи. Термінал дозволяє переглядати інформацію про стан резервуару (наприклад, рівень води) та керувати насосом, ввімкнувши або вимкнувши його вручну.
7. Arduino UNO (ARD1) – мікроконтролер, який виконує функцію головного мозку системи. Він отримує сигнали з датчиків рівня води, обробляє їх і керує роботою насоса через реле та транзистор. Arduino також відповідає за передачу інформації на віртуальний термінал.
8. Резистор (R1) – обмежує струм у колі транзистора, щоб уникнути пошкодження компонентів і забезпечити правильну роботу схеми.

**3.2.2. Створення схеми з використанням завантаженого пристрою**

****

**Рис. 3.8. Варіант схеми зі завантаженим пристроєм**

Цей варіант схеми реалізує систему моніторингу рівня води в трьох окремих резервуарах (TL1, TL2, TL3), підключених до плати Arduino Uno. Система також має мотор, який живить систему і підключений до піна 13.

Загальний опис схеми:

1. Кожен резервуар (TL1, TL2, TL3) має аналоговий датчик рівня води, який підключений до відповідного аналогового входу на платі Arduino (A0, A1, A2). Ці датчики вимірюють рівень води в резервуарах і передають значення до Arduino, яке обробляється в коді, щоб визначити поточний рівень води в кожному резервуарі.
2. Arduino Uno є центральним контролером системи. Він зчитує показники рівня води від датчиків, обробляє команди користувача з терміналу та керує станом мотора. Піни A0, A1, і A2 зчитують аналогові значення рівнів води, а цифровий пін 13 використовується для керування мотором.
3. Мотор підключений до контакту 13 Arduino і живиться від резистора 220 Ом, який допомагає обмежити струм, щоб уникнути перевантаження. Цей двигун керує потоком води, запускаючи або зупиняючи на основі команд користувача. Коли мотор вимкнений, система блокує інші функції, щоб запобігти перевантаженню.
4. Віртуальний термінал використовується для взаємодії з користувачем, надаючи інформацію про рівень води в резервуарах, статус мотора(вмикати/вимикати мотор) , та меню вибору дій.

**3.3. Архітектура та опис коду**

**3.3.1. Опис коду з власним пристроєм**

*#include <LiquidCrystal.h>*

*const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;*

*LiquidCrystal led(rs, en, d4, d5, d6, d7);*

Ініціалізація дисплея, підключеного до пінів на Arduino, щоб показувати повідомлення.

*int lowTank = A3;*

*int quTank = A2;*

*int halfTank = A1;*

*int fullTank = A0;*

*int motor = 13;*

Ці змінні представляють різні рівні води (lowTank, quTank, halfTank, fullTank) та мотор, підключений до цифрового піна 13.

*void setup() {*

*led.begin(16, 2);*

*Serial.begin(9600);*

*led.print("Tank Status..........");*

*pinMode(lowTank, OUTPUT);*

*pinMode(quTank, OUTPUT);*

*pinMode(halfTank, OUTPUT);*

*pinMode(fullTank, OUTPUT);*

*pinMode(motor, OUTPUT);*

*}*

У *setup()* налаштовується LCD, послідовна комунікація та конфігуруються пінни рівнів цистерн та мотора як виходи. На екрані відображається повідомлення "Tank Status".

*void loop() {*

*Serial.write(27);*

*Serial.print("[][][][]");*

*Serial.write(27);*

*Serial.print("[][][][]");*

*delay(500);*

*Serial.println("Terminal console");*

*Serial.println("what should have been done:");*

*Serial.println("1. ON/OFF rotor");*

*Serial.println("2. Fill the tank on Low");*

*Serial.println("3. Fill the tank on 1/4");*

*Serial.println("4. Fill the tank on HALF");*

*Serial.println("5. Fill the tank on FULL");*

*Serial.println("6. Status");*

*while (Serial.available() == 0) {}*

*char command = Serial.read();*

*switch (command) {*

*case '1':*

*toggleMotor();*

*break;*

*case '2':*

*fillTank(0);*

*break;*

*case '3':*

*fillTank(1);*

*break;*

*case '4':*

*fillTank(2);*

*break;*

*case '5':*

*fillTank(3);*

*break;*

*case '6':*

*checkStatus();*

*break;*

*default:*

*Serial.println("Wrong command!");*

*break;*

*}*

*delay(1000);*

*}*

*loop() -* відображає меню в послідовному моніторі з опціями для керування мотором і рівнями наповнення цистерни. Чекає введення команди користувача і викликає відповідні функції. Якщо введено невірну команду, відображається помилка.

*void toggleMotor() {*

*static bool motorState = false;*

*motorState = !motorState;*

*digitalWrite(motor, motorState ? HIGH : LOW);*

*Serial.println(motorState ? "Rotor on" : "Rotor off");*

*if (!motorState) {*

*digitalWrite(lowTank, LOW);*

*digitalWrite(quTank, LOW);*

*digitalWrite(halfTank, LOW);*

*digitalWrite(fullTank, LOW);*

*Serial.println("All LEDs turned off as the rotor is off");*

*}*

*}*

Функція *toggleMotor()* перемикає стан мотора (вкл/викл) щоразу при виклику. Коли мотор вимкнено, всі світлодіоди рівнів цистерн також вимикаються, і виводиться повідомлення що мотор вимкнений, поки мотор не буде знову включений, наступні дії заблоковані.

*void fillTank(int level) {*

*digitalWrite(lowTank, LOW);*

*digitalWrite(quTank, LOW);*

*digitalWrite(halfTank, LOW);*

*digitalWrite(fullTank, LOW);*

*switch (level) {*

*case 0:*

*Serial.println("Fill the tank on Low");*

*digitalWrite(lowTank, HIGH);*

*break;*

*case 1:*

*Serial.println("Fill the tank on 1/4");*

*digitalWrite(quTank, HIGH);*

*break;*

*case 2:*

*Serial.println("Fill the tank on HALF");*

*digitalWrite(halfTank, HIGH);*

*break;*

*case 3:*

*Serial.println("Fill the tank on FULL");*

*digitalWrite(fullTank, HIGH);*

*break;*

*}*

*Serial.print("You have filled the tank to level: ");*

*Serial.println(level == 0 ? "LOW" : level == 1 ? "1/4" : level == 2 ? "HALF" : "FULL");*

*delay(1000);*

*}*

Функція *fillTank(int level)* наповнює цистерну до заданого рівня, увімкнувши відповідний світлодіод, і вимикає інші. Показує повідомлення з поточним рівнем цистерни.

*void checkStatus() {*

*int i = digitalRead(fullTank);*

*int j = digitalRead(halfTank);*

*int k = digitalRead(quTank);*

*int l = digitalRead(lowTank);*

*Serial.print("FULL pin state: ");*

*Serial.println(i);*

*Serial.print("HALF pin state: ");*

*Serial.println(j);*

*Serial.print("1/4 pin state: ");*

*Serial.println(k);*

*Serial.print("LOW pin state: ");*

*Serial.println(l);*

*if (i == HIGH) {*

*Serial.println("Status: FULL");*

*} else if (j == HIGH) {*

*Serial.println("Status: HALF");*

*} else if (k == HIGH) {*

*Serial.println("Status: 1/4");*

*} else if (l == HIGH) {*

*Serial.println("Status: LOW");*

*} else {*

*Serial.println("Status: EMPTY");*

*}*

*}*

Функція *checkStatus()* перевіряє поточний стан кожного піну рівня цистерни та виводить відповідне повідомлення про рівень води в цистерні. В залежності від активного світлодіода, система визначає статус як FULL, HALF, 1/4, LOW або EMPTY (якщо всі вимкнені).

**3.3.2.** **Опис коду з завантаженим пристроєм**

*#define SENSOR\_PIN1 A0*

*#define SENSOR\_PIN2 A1*

*#define SENSOR\_PIN3 A2*

*#define MOTOR\_PIN 13*

Визначено константи для трьох сенсорів рівня води та піну для мотора. *SENSOR\_PIN1, SENSOR\_PIN2, і SENSOR\_PIN3* використовуються для зчитування рівня води, а *MOTOR\_PIN* контролює мотор.

*bool motorStatus = false;*

*int userCommand = 0;*

Змінні *motorStatus* та *userCommand* - *motorStatus* відстежує, чи увімкнений мотор (початково вимкнений), а *userCommand* зберігає введену команду користувача.

*void setup() {*

*pinMode(MOTOR\_PIN, OUTPUT);*

*digitalWrite(MOTOR\_PIN, LOW);*

*Serial.begin(9600);*

*displayMotorStatus();*

*}*

У функції *setup()* налаштовується мотор як вихід та відразу вимикається. Початковий статус мотора відображається на екрані.

*void loop() {*

*if (Serial.available() > 0) {*

*userCommand = Serial.parseInt();*

*if (userCommand == 3) {*

*motorStatus = !motorStatus;*

*digitalWrite(MOTOR\_PIN, motorStatus ? HIGH : LOW);*

*if (motorStatus) {*

*Serial.println("The motor is now on. You can proceed with this action.");*

*displayMenu();*

*} else {*

*Serial.println("The motor is off. You cannot perform other actions.");*

*}*

*} else if (motorStatus) {*

*if (userCommand == 1) {*

*displayTankLevels();*

*} else if (userCommand == 2) {*

*Serial.println("Command 2 is executed");*

*} else {*

*Serial.println("Unknown command. Please enter 1 to check water level, 2 for another function, or 3 to turn the motor on/off.");*

*}*

*displayMenu();*

*} else {*

*Serial.println("The motor is off. Please turn it on (enter 3) to perform other actions.");*

*}*

*}*

*}*

*loop()* перевіряє наявність введення в послідовному моніторі. Якщо отримано команду 3, стан мотора перемикається. Якщо мотор увімкнений, дозволяється виконувати інші дії, такі як перевірка рівня води. Якщо ж мотор вимкнений, інші команди заборонені.

*void displayMotorStatus() {*

*if (motorStatus) {*

*Serial.println("The motor is on. You can proceed with actions.");*

*displayMenu();*

*} else {*

*Serial.println("The motor is off. Please turn it on (enter 3) to perform other actions.");*

*}*

*}*

Функція *displayMotorStatus()* показує поточний статус мотора при запуску або зміні його стану. Відображає меню, якщо мотор увімкнений.

*void displayMenu() {*

*Serial.println("=== Menu ===");*

*Serial.println("Press 1 to check the water level.");*

*Serial.println("Press 2 for another function.");*

*Serial.println("Press 3 to turn the motor on/off.");*

*Serial.println("=============");*

*}*

Функція *displayMenu()* відображає меню, яке дозволяє користувачу вибирати дії: перевірити рівень води, виконати додаткову команду або увімкнути/вимкнути мотор.

*void displayTankLevels() {*

*int sensorValue1 = analogRead(SENSOR\_PIN1);*

*int sensorValue2 = analogRead(SENSOR\_PIN2);*

*int sensorValue3 = analogRead(SENSOR\_PIN3);*

*float tankLevel1 = map(sensorValue1, 0, 1023, 0, 100);*

*float tankLevel2 = map(sensorValue2, 0, 1023, 0, 100);*

*float tankLevel3 = map(sensorValue3, 0, 1023, 0, 100);*

*Serial.println("=== Water Levels in Tanks ===");*

*Serial.print("Tank 1: ");*

*Serial.print(tankLevel1);*

*Serial.println("%");*

*Serial.print("Tank 2: ");*

*Serial.print(tankLevel2);*

*Serial.println("%");*

*Serial.print("Tank 3: ");*

*Serial.print(tankLevel3);*

*Serial.println("%");*

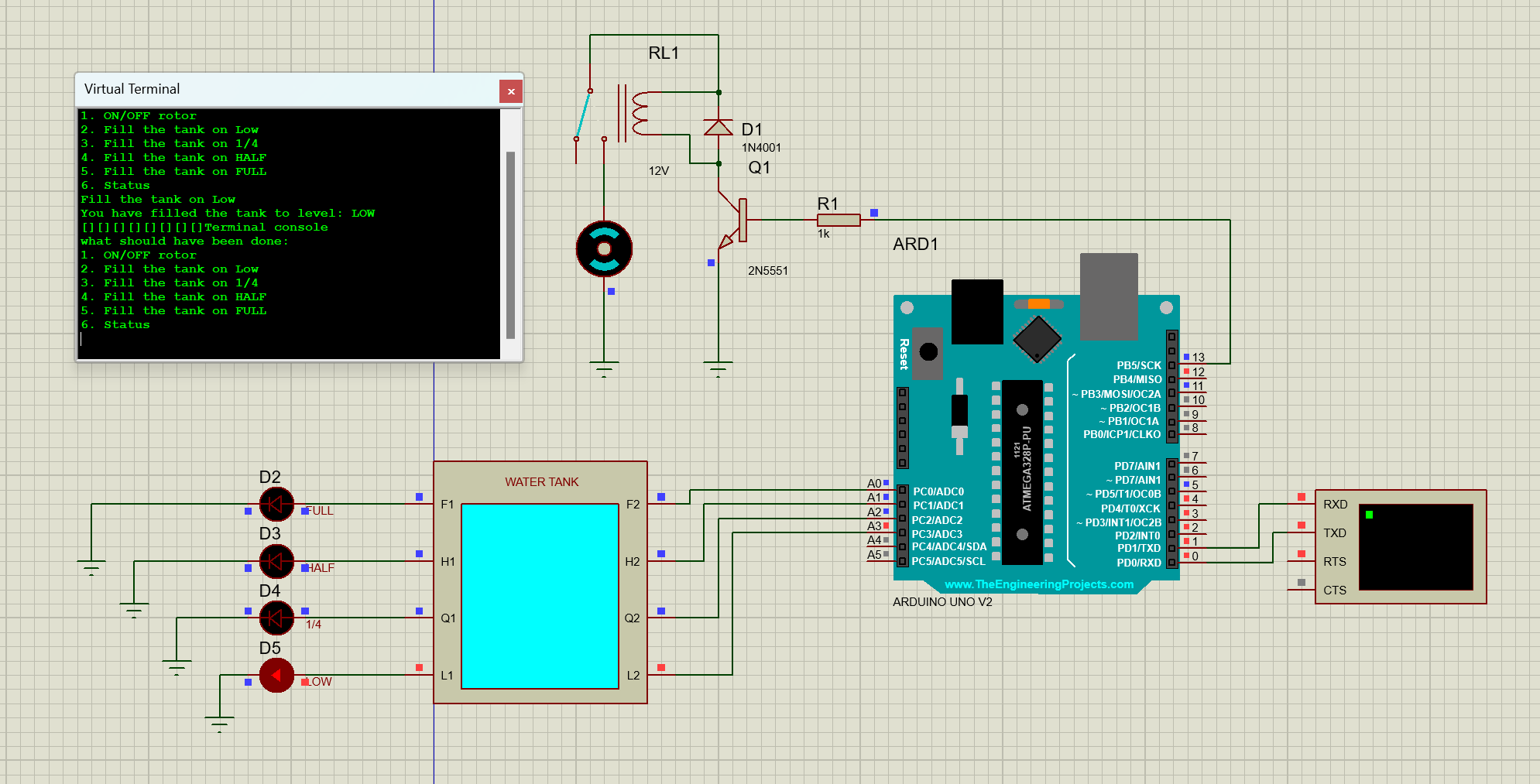
*Serial.println("=============================");*

*Serial.println();*

*}*

Функція *displayTankLevels()* зчитує аналогові значення з сенсорів, які потім масштабуються до відсотків, від 0 до 100, і показуються в послідовному моніторі.

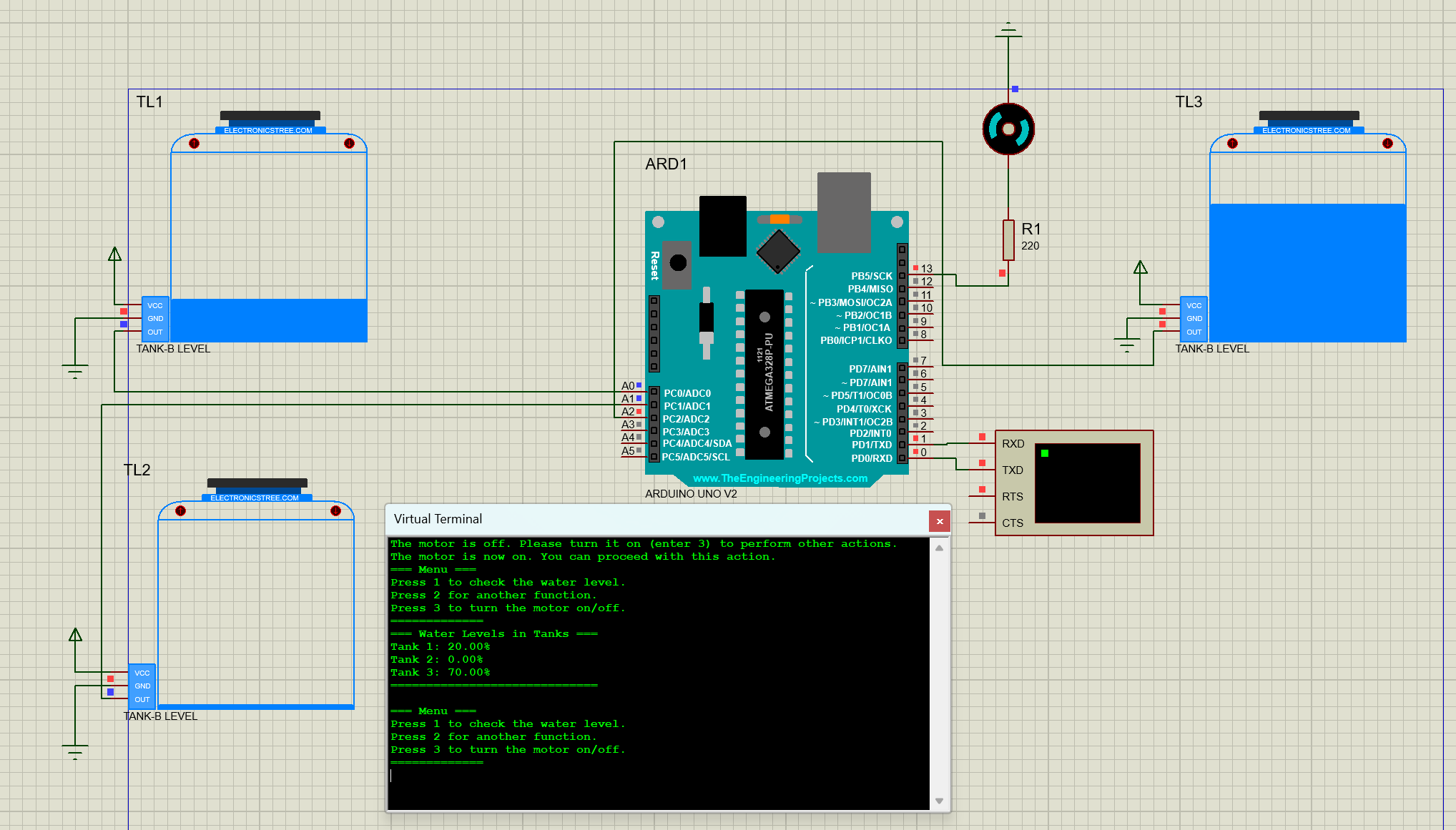
**3.4.** **Функціонал та порівняння**



**Рис. 3.9. Демонстрація роботи схеми зі створеним пристроєм**

На цій схемі показана базова система моніторингу рівня води, де є лише один резервуар для води та кілька індикаторів рівня води, що діють як центральний контролер системи. Він зчитує сигнали від різних датчиків рівня в баку та обробляє команди користувача через монітор послідовного порту. У резервуарі для води встановлено кілька датчиків, які визначають рівень води на різних позначках: низький, чверть, половина та повний. Кожен датчик відповідає певному рівню води та дає актуальну інформацію про рівень води. Світлодіодні індикатори підключені до кожного датчика рівня резервуара. Вони загоряються відповідно до рівня води, який визначає датчик. Червоні світлодіоди позначають кожен рівень води (низький, 1/4, 1/2, повний), і вони активуються, коли резервуар досягає цього рівня. Мотор контролюється через пін 13 на Arduino. Коли мотор увімкнено, він дозволяє воді надходити в резервуар. Система забороняє виконувати інші дії, поки мотор активний, щоб уникнути перевантаження.

Серійний монітор використовується як інтерфейс користувача для взаємодії з системою. Користувачі можуть вводити команди для ввімкнення/вимкнення мотора, заповнення резервуара до певного рівня або перевірки статусу резервуара. На серійному моніторі відображаються рівні резервуара та повідомлення системи, які допомагають зрозуміти поточний стан резервуара та мотора.



**Рис. 3.10. Демонстрація роботи схеми зі завантаженим пристроєм**

На цій схемі показана більш досконала система з інтегрованим релейним керуванням трьома резервуарами та мотором. Arduino Uno, як і в першій схемі, виступає основним процесором і контролером системи. Він обробляє вхідні сигнали від датчиків для трьох окремих резервуарів та здійснює контроль мотора на основі команд користувача.

Основні компоненти системи включають датчики рівня води для трьох резервуарів (підключені до аналогових входів A0, A1, A2 на Arduino), мотор, керований через цифровий контакт 13, і віртуальний термінал для відображення інформації. Резистор 220 Ом забезпечує захист двигуна від перевантаження по струму. Управління мотором реалізовано так, що користувач може вмикати та вимикати його за допомогою команди, після чого стають доступними інші функції, такі як перевірка рівня води, чи додаткові функції зарезервовані для майбутніх потреб, таких як автоматичне заповнення резервуара або запуск насоса.

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Перша схема** | **Друга схема** |
| Кількість резервуарів | Один резервуар | Три незалежні резервуари |
| Рівні води | Фіксовані рівні (LOW, 1/4, HALF, FULL) | Відсоткові значення рівня води для кожного резервуара |
| Керування мотором | Команда для ввімкнення/вимкнення мотора | Команда для ввімкнення/вимкнення мотора |
| Датчики рівня води | Дискретні датчики для кожного рівня | Аналогові датчики, що зчитують відсотковий рівень води |
| Виведення інформації | Віртуальний термінал, що відображає рівень води та статус мотора | Віртуальний термінал, що показує точний рівень води у відсотках |
| Тип індикації рівня води | Лампи для кожного рівня | Відображення даних у віртуальному терміналі |
| Гнучкість у використанні | Обмежена, підходить для простого моніторингу одного резервуара | Більш універсальна, підходить для моніторингу декількох резервуарів |
| Захист мотора | 220 Ом резистор у колі мотора | 220 Ом резистор у колі мотора |
| Можливість розширення | Обмежена, складно додати більше резервуарів | Легка інтеграція додаткових резервуарів і функцій |
| Основна мета використання | Простий моніторинг контролю рівня води в одному резервуарі | Моніторинг та контроль рівня води у кількох резервуарах |

Загалом, обидві схеми призначені для моніторингу рівня води в резервуарах. Перша схема підходить для простих завдань контролю рівня води в одному резервуарі, використовуючи чітко визначені рівні та індикацію за допомогою світлодіодів. Це рішення може бути корисним у випадках, де важливо мати конкретні рівні заповнення, наприклад, низький або повний рівень. Друга схема, навпаки, є більш гнучкою та функціональною, оскільки підтримує моніторинг декількох резервуарів з точними відсотковими значеннями рівня води для кожного з них. Це робить її придатною для більш складних систем, де потрібен детальний контроль об’єму води в кожному резервуарі. Друга схема також полегшує можливість розширення системи, що дозволяє додавати нові функції та резервуари.

**Висновки до розділу 3**

1. Описано процес створення пристрою з основними датчиками, необхідними для моніторингу рівня води, а також компоненти для керування насосом, що забезпечують стабільне і ефективне функціонування системи.
2. Створення схеми в програмі Proteus, зокрема деталі налаштування елементів, таких як реле, транзистори та інші компоненти, і принцип їхньої роботи для точного контролю рівня води та запуску/зупинки мотора.
3. Розглянута архітектура коду, який дозволяє пристрою обробляти команди користувача для перемикання моторів і отримання даних з датчиків. Також пояснюються функції для керування станом системи, надання зворотного зв'язку користувачу через серійний монітор.
4. Аналіз функціональних можливостей обох схем показав, що кожна з них має свої сильні сторони, відповідні для різних сценаріїв використання. Схема зі створеним датчиком значно перевершує пристрій із завантаженим пристроєм від Electronicstree завдяки своїй функціональності та точності. Вона забезпечує контроль рівня води на основі датчиків, що дозволяє відстежувати стан цистерн із більшою деталізацією. Це робить систему придатною для великих об’єктів і складних мереж водопостачання, де потрібен гнучкий вибір джерела води. На відміну від простої базової конструкції завантаженого пристрою, створений датчик забезпечує режим реального часу, кращу інтеграцію з іншими компонентами та стабільність роботи в умовах змінного навантаження.

**ЗАГАЛЬНІ ВИСНОВКИ**

У межах дипломної роботи було створено комп’ютеризовану систему моніторингу води. Під час реалізації цього проєкту використовувалися інструменти та технології, зокрема Arduino IDE для написання коду, Proteus для моделювання схем і мова програмування C/C++ для забезпечення ефективності та гнучкості системи.

Основні етапи були присвячені дослідженню та розробці архітектури системи, включно з вибором датчиків, що забезпечують точні показники рівня води, та аналізом методів їх інтеграції в систему для досягнення максимальної ефективності. Проведені експерименти з різними конфігураціями допомогли розробити стабільну та надійну модель пристрою. Проєкт продемонстрував можливості платформи Arduino у створенні інтегрованої комп'ютерної системи моніторингу з простою, але ефективною архітектурою, яка може використовуватися як у побутових, так і промислових умовах. Поєднання доступних інструментів і методів дозволило створити універсальну систему, здатну адаптуватися до різних потреб користувача.

У процесі побудови схеми використовувалися як індивідуально розроблені, так і готові компоненти, що допомогло точно відтворити основні принципи роботи системи. Створений програмний код забезпечує функціональність системи, дозволяючи не лише автоматично контролювати рівень води, а й захищати пристрій від перевантаження, блокуючи сторонні команди під час роботи мотора, що мінімізує ризики збоїв і подовжує термін експлуатації системи. Крім того, реалізована інтерактивна взаємодія з користувачем, яка інформує про поточний стан резервуарів і пропонує можливі дії для покращення функціонування.

Симуляції в Proteus продемонстрували стабільну роботу системи та підтвердили її надійність. У процесі роботи були розроблені дві різні схеми з різними перевагами. Перша схема, яка використовує базові датчики для контролю рівня води в одному резервуарі, залишається ефективною для простих систем водопостачання завдяки своїй простій конструкції та мінімальним вимогам до обслуговування. Однак друга схема, розроблена на основі створеного пристрою, демонструє переваги завдяки більшій кількості датчиків і можливості одночасного контролю декількох резервуарів. Це дозволяє не лише точно відстежувати рівень води в кожному резервуарі окремо, а й реалізувати інтеграцію в більш складні системи з можливістю адаптації до конкретних умов. Схема зі створеним пристроєм також забезпечує вищу точність вимірювань, підвищену гнучкість у налаштуванні та можливість швидкого реагування на зміни, що робить її більш надійною та універсальною для промислових і побутових потреб.

Результати підтвердили, що створена система моніторингу відповідає початковим вимогам, демонструє високу точність вимірювань рівня води та може бути застосована для різноманітних побутових і промислових потреб. Цей проєкт продемонстрував ефективність використання доступних інструментів для створення багатофункціональної системи, яка здатна до адаптації та вдосконалення в майбутньому.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. КНУТД.ua // [Електронний ресурс] - Режим доступу: <https://knutd.edu.ua/university/institutes/nniiit/tosuems/>

2. IoT Tech Expo // [Електронний ресурс] - Режим доступу: <https://www.iottechexpo.com/>

3. IoT Design // [Електронний ресурс] - Режим доступу: <https://www.iotdesignpro.com/>

4. Arduino // [Електронний ресурс] - Режим доступу: <https://www.arduino.cc/en/software>

5. Proteus // [Електронний ресурс] - Режим доступу: <https://proteus.informer.com/>

6. Water Sensor Library For Proteus // [Електронний ресурс] - Режим доступу: <https://www.theengineeringprojects.com/2020/07/water-sensor-library-for-proteus.html>

7. Arduino Triggers // [Електронний ресурс] - Режим доступу: <https://docs.arduino.cc/arduino-cloud/cloud-interface/triggers/>

9. Labcenter electronics // [Електронний ресурс] - Режим доступу: <https://web.archive.org/web/20140531025852/http://www.labcenter.com/index.cfm>

10. Proteus cxem // [Електронний ресурс] - Режим доступу: <https://cxem.net/software/proteus.php>

11. Labcenter libraries // [Електронний ресурс] - Режим доступу: <https://www.labcenter.com/libraries/>

12. Proteus Libraries of Embedded Sensors // [Електронний ресурс] - Режим доступу:

<https://www.theengineeringprojects.com/2020/07/proteus-libraries-of-embedded-sensors.html>

13. Tank Level Sensor - Proteus 8 Library// [Електронний ресурс] - Режим доступу: <https://electronicstree.com/tank-level-sensor/>

14. Віджаякумар, М., і Аравіндхан, К. (2016). Автоматичний регулятор рівня води та пусковий пристрій мотопомпи через GSM. Міжнародний журнал досліджень електротехніки та електроніки, 4(2), 43-47.

15. Гайквад, М. А. та Вайшнаві, В. Д. (2017). Система моніторингу якості води на основі IOT, 10(5), 1107-1116.

16. Амріт Кумар Паніграгі, Чандан Кумар Сінгх, Дівеш Кумар, Неміша Хота: „Індикатор і контролер рівня води в резервуарі з використанням Arduino”, Міжнародний журнал передових досліджень в електротехніці, електроніці та інженерії вимірювальних приладів. ISSN (друк): 2320 – 3765, ISSN (онлайн): 2278 – 8875, Т. 6, Вип. 3, березень 2017, с. 1348-1354.

17. Пані Пуджа К, пані Кусумаваті, пані Павітра, пані Нішміта, професор Айшварія Д Шетті: «АВТОМАТИЧНИЙ ІНДИКАТОР РІВНЯ ВОДИ ТА КОНТРОЛЕР ЗА ВИКОРИСТАННЯМ ARDUINO», Міжнародний дослідницький журнал інженерії та технологій (IRJET) e-ISSN: 2395- 0056, Том: 07 Випуск: 05 | Травень 2020 р., стор. 5342-5346

18. Дипломне проєктування: методичні вказівки до виконання випускної кваліфікаційної роботи бакалавра для студентів усіх форм навчання спеціальності 123 – Комп’ютерна інженерія освітньої програми «Комп‟ютерні системи та мережі» / упор.: В.В. Осипенко, Б.М. Злотенко, Д.В. Стаценко – К. : КНУТД, 2022. – 37 с.

19. ПОЛОЖЕННЯ про кваліфікаційну роботу освітнього ступеня бакалавра Київського національного університету технологій та дизайну. УХВАЛЕНО Вченою радою Київського національного університету технологій та дизайну протокол від « 24 » 02 2021 р. № 7.

Д О Д А Т К И

ДОДАТОК А

*#include <LiquidCrystal.h>*

*const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;*

*LiquidCrystal led(rs, en, d4, d5, d6, d7);*

*int lowTank = A3;*

*int quTank = A2;*

*int halfTank = A1;*

*int fullTank = A0;*

*int motor = 13;*

*void setup() {*

*led.begin(16, 2);*

*Serial.begin(9600);*

*led.print("Tank Status..........");*

*pinMode(lowTank, OUTPUT);*

*pinMode(quTank, OUTPUT);*

*pinMode(halfTank, OUTPUT);*

*pinMode(fullTank, OUTPUT);*

*pinMode(motor, OUTPUT);*

*}*

*void loop() {*

*Serial.write(27);*

*Serial.print("[][][][]");*

*Serial.write(27);*

*Serial.print("[][][][]");*

*delay(500);*

*Serial.println("Terminal console");*

*Serial.println("what should have been done:");*

*Serial.println("1. ON/OFF rotor");*

*Serial.println("2. Fill the tank on Low");*

*Serial.println("3. Fill the tank on 1/4");*

*Serial.println("4. Fill the tank on HALF");*

*Serial.println("5. Fill the tank on FULL");*

*Serial.println("6. Status");*

*while (Serial.available() == 0) {}*

*char command = Serial.read();*

*switch (command) {*

*case '1':*

*toggleMotor();*

*break;*

*case '2':*

*fillTank(0);*

*break;*

*case '3':*

*fillTank(1);*

*break;*

*case '4':*

*fillTank(2);*

*break;*

*case '5':*

*fillTank(3);*

*break;*

*case '6':*

*checkStatus();*

*break;*

*default:*

*Serial.println("Wrong command!");*

*break;*

*}*

*delay(1000);*

*}*

*void toggleMotor() {*

*static bool motorState = false;*

*motorState = !motorState;*

*digitalWrite(motor, motorState ? HIGH : LOW);*

*Serial.println(motorState ? "Rotor on" : "Rotor off");*

*if (!motorState) {*

*digitalWrite(lowTank, LOW);*

*digitalWrite(quTank, LOW);*

*digitalWrite(halfTank, LOW);*

*digitalWrite(fullTank, LOW);*

*Serial.println("All LEDs turned off as the rotor is off");*

*}*

*}*

*void fillTank(int level) {*

*digitalWrite(lowTank, LOW);*

*digitalWrite(quTank, LOW);*

*digitalWrite(halfTank, LOW);*

*digitalWrite(fullTank, LOW);*

*switch (level) {*

*case 0:*

*Serial.println("Fill the tank on Low");*

*digitalWrite(lowTank, HIGH);*

*break;*

*case 1:*

*Serial.println("Fill the tank on 1/4");*

*digitalWrite(quTank, HIGH);*

*break;*

*case 2:*

*Serial.println("Fill the tank on HALF");*

*digitalWrite(halfTank, HIGH);*

*break;*

*case 3:*

*Serial.println("Fill the tank on FULL");*

*digitalWrite(fullTank, HIGH);*

*break;*

*}*

*Serial.print("You have filled the tank to level: ");*

*Serial.println(level == 0 ? "LOW" : level == 1 ? "1/4" : level == 2 ? "HALF" : "FULL");*

*delay(1000);*

*}*

*void checkStatus() {*

*int i = digitalRead(fullTank);*

*int j = digitalRead(halfTank);*

*int k = digitalRead(quTank);*

*int l = digitalRead(lowTank);*

*Serial.print("FULL pin state: ");*

*Serial.println(i);*

*Serial.print("HALF pin state: ");*

*Serial.println(j);*

*Serial.print("1/4 pin state: ");*

*Serial.println(k);*

*Serial.print("LOW pin state: ");*

*Serial.println(l);*

*if (i == HIGH) {*

*Serial.println("Status: FULL");*

*} else if (j == HIGH) {*

*Serial.println("Status: HALF");*

*} else if (k == HIGH) {*

*Serial.println("Status: 1/4");*

*} else if (l == HIGH) {*

*Serial.println("Status: LOW");*

*} else {*

*Serial.println("Status: EMPTY");*

*}*

*}*

ДОДАТОК Б

*#define SENSOR\_PIN1 A0*

*#define SENSOR\_PIN2 A1*

*#define SENSOR\_PIN3 A2*

*#define MOTOR\_PIN 13*

*bool motorStatus = false;*

*int userCommand = 0;*

*void setup() {*

*pinMode(MOTOR\_PIN, OUTPUT);*

*digitalWrite(MOTOR\_PIN, LOW);*

*Serial.begin(9600);*

*displayMotorStatus();*

*}*

*void loop() {*

*if (Serial.available() > 0) {*

*userCommand = Serial.parseInt();*

*if (userCommand == 3) {*

*motorStatus = !motorStatus;*

*digitalWrite(MOTOR\_PIN, motorStatus ? HIGH : LOW);*

*if (motorStatus) {*

*Serial.println("The motor is now on. You can proceed with this action.");*

*displayMenu();*

*} else {*

*Serial.println("The motor is off. You cannot perform other actions.");*

*}*

*} else if (motorStatus) {*

*if (userCommand == 1) {*

*displayTankLevels();*

*} else if (userCommand == 2) {*

*Serial.println("Command 2 is executed. (This could be another function)");*

*} else {*

*Serial.println("Unknown command. Please enter 1 to check water level, 2 for another function, or 3 to turn the motor on/off.");*

*}*

*displayMenu();*

*} else {*

*Serial.println("The motor is off. Please turn it on (enter 3) to perform other actions.");*

*}*

*}*

*}*

*void displayMotorStatus() {*

*if (motorStatus) {*

*Serial.println("The motor is on. You can proceed with actions.");*

*displayMenu();*

*} else {*

*Serial.println("The motor is off. Please turn it on (enter 3) to perform other actions.");*

*}*

*}*

*void displayMenu() {*

*Serial.println("=== Menu ===");*

*Serial.println("Press 1 to check the water level.");*

*Serial.println("Press 2 for another function.");*

*Serial.println("Press 3 to turn the motor on/off.");*

*Serial.println("=============");*

*}*

*void displayTankLevels() {*

*int sensorValue1 = analogRead(SENSOR\_PIN1);*

*int sensorValue2 = analogRead(SENSOR\_PIN2);*

*int sensorValue3 = analogRead(SENSOR\_PIN3);*

*float tankLevel1 = map(sensorValue1, 0, 1023, 0, 100);*

*float tankLevel2 = map(sensorValue2, 0, 1023, 0, 100);*

*float tankLevel3 = map(sensorValue3, 0, 1023, 0, 100);*

*Serial.println("=== Water Levels in Tanks ===");*

*Serial.print("Tank 1: ");*

*Serial.print(tankLevel1);*

*Serial.println("%");*

*Serial.print("Tank 2: ");*

*Serial.print(tankLevel2);*

*Serial.println("%");*

*Serial.print("Tank 3: ");*

*Serial.print(tankLevel3);*

*Serial.println("%");*

*Serial.println("=============================");*

*Serial.println();*

*}*