

Міністерство освіти і науки України
Державний заклад
«Луганський національний університет імені Тараса Шевченка»

Факультет технологій та інформаційних систем

Кафедра інформаційних технологій та систем

Мартинюк Ілля Богданович

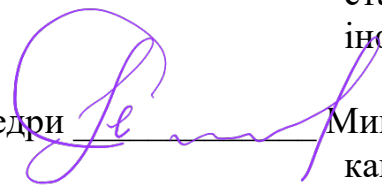
**РОЗРОБКА ЛАБОРАТОРНОГО СТЕНДУ ІоТ "РОЗУМНА ФЕРМА"
НА ESP32**

кваліфікаційна робота

**здобувача вищої освіти першого (бакалаврського) рівня
освітньої програми «Інженерія програмного забезпечення»
за спеціальністю 121 Інженерія програмного забезпечення**

Особистий підпис  Ілля МАРТИНЮК

Науковий керівник  Володимир ДОНЧЕНКО,
старший викладач кафедри
інформаційних технологій та систем

Завідувач кафедри  Микола СЕМЕНОВ,
кандидат педагогічних наук, доцент
кафедри інформаційних технологій
та систем

Лубни – 2026

Міністерство освіти і науки України
Державний заклад „Луганський національний університет
імені Тараса Шевченка”

Факультет

Факультет технологій та інформаційних
систем

Кафедра

Інформаційних технологій та систем

Рівень освіти

Перший (бакалаврський)

Спеціальність

121 «Інженерія програмного забезпечення»

(код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТС

М.А. Семенов

(підпис)

(ініціали, прізвище)

“ ” 2025 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Мартинюку Іллі Богдановичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) **Розробка лабораторного стенду IoT "Розумна ферма" на ESP32**

Керівник кваліфікаційної роботи

Донченко В.Ю.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету

Від“ ” 2025 року №

2. Строк подання студентом проекту (роботи)

3. Вихідні дані до роботи (проекту)

у результаті виконання роботи

розроблено лабораторний стенд IoT "Розумна ферма" на ESP32

(визначаються кількісні або (та) якісні показники, яким повинен відповідати об'єкт розробки)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) **АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕНЬ. ВИБІР КОМПОНЕНТІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЛАБОРАТОРНОГО СТЕНДУ "РОЗУМНА ФЕРМА" НА ESP32. ПРАКТИЧНА РЕАЛІЗАЦІЯ ЛАБОРАТОРНОГО СТЕНДУ IoT "РОЗУМНА ФЕРМА" НА ESP32**

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання „_____” _____ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	До 15 жовтня	
	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	Другий тиждень листопада (10 листопада)	
	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником.	До 15 грудня	
	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	До 28 січня	
	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	Перший тиждень березня	
	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	До 31 березня	
	Попередній захист роботи на кафедрі	квітень	
	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	За 10 днів до державної атестації	
	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	За 5 днів до державної атестації	

Здобувач освіти

підпис

Ілля МАРТИНЮК

(ініціали, прізвище)

Керівник проєкту (роботи)

підпис

Володимир ДОНЧЕНКО

АНОТАЦІЯ

Мартинюк І. Б.

Тема: Розробка лабораторного стенду IoT "Розумна ферма" на ESP32.

Спеціальність: 121 "Інженерія програмного забезпечення"

Установа: ЛНУ імені Тараса Шевченка, 2026 р.

Бакалаврська робота містить: 110 с., 81 рис., 8 табл., 2 додат., 35 джерел.

Об'єкт дослідження – процеси моніторингу та автоматизованого керування параметрами середовища в системах розумного фермерства за допомогою технологій Інтернету речей.

Предмет дослідження – архітектура, апаратні компоненти, алгоритмічне та програмне забезпечення лабораторного стенду IoT «Розумна ферма» на базі мікроконтролера ESP32 з можливістю веб-керування та мобільної інтеграції.

Мета роботи – розробка лабораторного стенду IoT «Розумна ферма» на базі ESP32, який забезпечує збір даних із датчиків, керування виконавчими пристроями, реалізацію автоматизованих сценаріїв роботи та віддалений доступ через вебінтерфейс і мобільний застосунок.

Результати роботи. У дипломному проєкті проведено комплексний аналіз сучасного стану розвитку IoT-технологій в аграрному секторі, за результатами якого визначено архітектуру та технічні вимоги до розробки багатофункціональної навчально-лабораторної системи. Обґрунтовано та сформовано апаратну структуру лабораторного стенду на основі двоядерного мікроконтролера KEYESTUDIO ESP32 PLUS. Підібрано та інтегровано комплекс сенсорів (температури, вологості, освітленості, руху й відстані) та виконавчих пристроїв (реле, сервопривод, зумер, світлодіоди). Програмно реалізовано алгоритми локальної автоматизації в середовищі Arduino IDE, які забезпечують:

- плавне ШІМ-керування яскравістю освітлення та обробку сигналів ручного керування із захистом від деренчання контактів кнопки;
- автоматичну активацію фітоламп при зниженні рівня природного світла (менше 800 одиниць АЦП);
- роботу охоронної сигналізації периметра на основі інфрачервоного датчика руху PIR та динамічної звукової сирени;
- інтелектуальне безконтактне годування тварин із відкриттям заслінки сервоприводом SG90 при наближенні об'єкта до датчика HC-SR04 в діапазоні 2–7 см;
- автоматичне зрошення залежно від сухості ґрунту із багаторівневим алгоритмом захисту водяного мікронасоса від «сухого ходу» при спустошенні резервуара.

Спроектовано та розгорнуто локальний веб-сервер на базі бібліотеки <WebServer.h>, що дозволяє через браузер будь-якого пристрою відстежувати телеметрію та надсилати текстові команди для керування актуаторами стенду. Налаштовано двосторонню мережеву взаємодію з кросплатформним мобільним APP-додатком за допомогою сокет-з'єднання. Додаток забезпечує віддалений моніторинг показників мікроклімату та безпосереднє дистанційне керування елементами ферми в реальному часі. Створено діючий фізичний прототип лабораторного стенду та локальну інформаційну систему індикації (на базі LCD 1602), які повністю готові до впровадження в освітній процес для практичної підготовки студентів у галузі програмування вбудованих систем та IoT.

Ключові слова. ІНТЕРНЕТ РЕЧЕЙ (IoT), МІКРОКОНТРОЛЕР ESP32, РОЗУМНА ФЕРМА, АВТОМАТИЗАЦІЯ МІКРОКЛІМАТУ, АВТОМАТИЧНЕ ЗРОШЕННЯ, ДАТЧИК РУХУ (PIR), УЛЬТРАЗВУКОВИЙ ДАЛЕКОМІР, ШИРОТНО-ІМПУЛЬСНА МОДУЛЯЦІЯ (ШИМ), ВЕБ-СЕРВЕР, МОБІЛЬНИЙ ДОДАТОК (APP), ARDUINO IDE.

ABSTRACT

Martyniuk Illia

Theme: Development of IoT laboratory stand "Smart Farm" on ESP32.

Speciality: 121 "Software Engineering"

Institution: Luhansk Taras Shevchenko National University (LTSNU), 2026.

Diploma work contains: 110 pages, 81 Fig., 8 Table, 2 adj., 35 source.

A research object is processes of monitoring and automated control of environmental parameters in smart farming systems using Internet of Things technologies.

The article of research is architecture, hardware components, algorithmic and software of the IoT laboratory stand "Smart Farm" based on the ESP32 microcontroller with the possibility of web management and mobile integration.

An aim of work is development of the IoT laboratory stand "Smart Farm" based on ESP32, which provides data collection from sensors, control of actuators, implementation of automated work scenarios, and remote access via a web interface and mobile application.

Job performances. The diploma project conducted a comprehensive analysis of the current state of development of IoT technologies in the agricultural sector, the results of which determined the architecture and technical requirements for the development of a multifunctional educational and laboratory system. The hardware structure of the laboratory stand based on the dual-core KEYESTUDIO ESP32 PLUS microcontroller was substantiated and formed. A set of sensors (temperature, humidity, illumination, motion and distance) and actuators (relay, servo, buzzer, LEDs) was selected and integrated. Local automation algorithms were implemented in the Arduino IDE environment, which provide:

- smooth PWM control of lighting brightness and processing of manual control signals with protection against rattling of button contacts;
- automatic activation of phytolamps when the level of natural light decreases (less than 800 ADC units);

- operation of perimeter security alarm based on an infrared PIR motion sensor and a dynamic sound siren;
- intelligent contactless feeding of animals with opening of the damper by the SG90 servo drive when the object approaches the HC-SR04 sensor in the range of 2–7 cm;
- automatic irrigation depending on the dryness of the soil with a multi-level algorithm for protecting the water micropump from “dry running” when the tank is empty.

A local web server based on the <WebServer.h> library has been designed and deployed, which allows you to monitor telemetry and send text commands to control the stand actuators via the browser of any device. Two-way network interaction with a cross-platform mobile APP application has been configured using a socket connection. The application provides remote monitoring of microclimate indicators and direct remote control of farm elements in real time. A working physical prototype of a laboratory stand and a local information display system (based on LCD 1602) have been created, which are fully ready for implementation in the educational process for practical training of students in the field of embedded systems programming and IoT.

Keywords. INTERNET OF THINGS (IoT), ESP32 MICROCONTROLLER, SMART FARM, MICROCLIMATE AUTOMATION, AUTOMATIC IRRIGATION, MOTION SENSOR (PIR), ULTRASOUND RANGE METER, PULSE WIDTH MODULATION (PWM), WEB SERVER, MOBILE APPLICATION (APP), ARDUINO IDE.

ІТС. ІПЗ4.0626-ВІІ
ВІДОМІСТЬ ПРОЄКТУ. РОЗРОБКА ЛАБОРАТОРНОГО СТЕНДУ ІоТ
"РОЗУМНА ФЕРМА" НА ESP32.

[illegible]

Міністерство освіти і науки України
Державний заклад «Луганський національний університет
імені Тараса Шевченка»

Факультет (інститут)

Факультет технологій та інформаційних
систем

(повна назва)

Кафедра

Кафедра інформаційних технологій та
систем

(повна назва)

(код, назва)

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання програмної розробки (ПР):

**"РОЗРОБКА ЛАБОРАТОРНОГО СТЕНДУ ІoT "РОЗУМНА ФЕРМА"
НА ESP32"**

ІТС. ІПЗ4.0626.02-ТЗ

ПОГОДЖЕНО

Керівник кваліфікаційної роботи

Донченко В.Ю.

“ _____ ” 2026р

ВИКОНАВЕЦЬ

Студент групи 4ІПЗ

Мартинюк І. Б.

“ _____ ” 2026р

Лубни – 2026

ЗМІСТ

1. ВСТУП	3
2. ПРИЗНАЧЕННЯ ПРОДУКЦІЇ	3
3. МЕТА СТВОРЕННЯ СИСТЕМИ.....	3
4. ХАРАКТЕРИСТИКА ОБ'ЄКТА.....	4
4.1. Основні задачі та функції об'єкта	4
5. ВИМОГИ ДО СИСТЕМИ	4
5.1. Вимоги до системи в цілому	4
5.1.1. Вимоги до функціональної структури.....	4
5.1.2. Вимоги до апаратного забезпечення.....	5
5.1.3. Вимоги до програмного та мережевого забезпечення.....	6
5.1.4. Вимоги до надійності та безпеки	6
6. ТЕХНІКО-ЕКОНОМІЧНІ ВИМОГИ ДО КІНЦЕВОГО ПРОДУКТУ	7
7. ВИМОГИ ДО МАТЕРІАЛІВ ТА КОМПЛЕКТУЮЧИХ.....	7
8. ЕТАПИ ВИКОНАННЯ ПР.....	7
9. ПРИЙМАННЯ	8
10. ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЕХНІЧНОГО ЗАВДАННЯ, ЩО	
ЗАТВЕРДЖЕНО.....	9

1. ВСТУП

1.1 Найменування: Розробка лабораторного стенду IoT "Розумна ферма" на ESP32.

1.2 Шифр ПР: АДР-9

1.3 Підстава до виконання ПР: Підставою для виконання даної розробки є завдання на дипломний проєкт.

1.4 Терміни розробки:

1.4.1 Початок 30 жовтня 2025р.

1.4.2 Закінчення 20 квітня 2026р.

1.5 Фінансується за рахунок коштів замовника.

2. ПРИЗНАЧЕННЯ ПРОДУКЦІЇ

2.1. Призначення:

Стенд призначений для використання у закладах вищої освіти як навчально-лабораторне обладнання для дослідження технологій Інтернету речей (IoT), програмування вбудованих систем, вивчення сучасних бездротових протоколів передачі даних та методів побудови кіберфізичних систем.

3. МЕТА СТВОРЕННЯ СИСТЕМИ

Метою проєкту є розробка лабораторного стенду IoT «Розумна ферма» на базі ESP32, який забезпечує збір даних із датчиків, керування виконавчими пристроями, реалізацію автоматизованих сценаріїв роботи та віддалений доступ через вебінтерфейс і мобільний застосунок.

Цілі створення:

- імітація та автоматизація основних технологічних процесів у сільському господарстві (контроль мікроклімату, полив, освітлення, безпека, годування).
- Забезпечення віддаленого моніторингу та керування виконавчими механізмами через веб-інтерфейс та мобільний додаток у реальному часі.

- Підвищення якості підготовки фахівців з інженерії програмного забезпечення завдяки наочній практичній базі.

4. ХАРАКТЕРИСТИКА ОБ'ЄКТА

Об'єктом автоматизації є модель замкнутої агроєкосистеми («Розумна ферма»), яка містить контури вимірювання фізичних параметрів навколишнього середовища (температура, вологість повітря, вологість ґрунту, рівень води в резервуарі, рівень природного освітлення, наявність руху в периметрі) та виконавчі пристрої для корекції цих параметрів.

4.1. Основні задачі та функції об'єкта

Система призначена для:

1. навчання студентів принципам побудови IoT-систем;
2. проведення лабораторних робіт з програмування ESP32;
3. дослідження алгоритмів автоматизації фермерських процесів;
4. демонстрації роботи сенсорів та виконавчих механізмів;
5. віддаленого моніторингу та керування елементами ферми.

5. ВИМОГИ ДО СИСТЕМИ

5.1. Вимоги до системи в цілому

Лабораторний стенд IoT «Розумна ферма» повинен функціонувати як комплексна кіберфізична система, що поєднує локальну автоматизацію (вбудовані алгоритми мікроконтролера) та дворівневий мережевий інтерфейс користувача (веб-інтерфейс та мобільний додаток).

5.1.1. Вимоги до функціональної структури

Система повинна забезпечувати виконання таких автономних та мережевих функцій:

- **Підсистема освітлення:** ручне увімкнення/вимкнення світла за допомогою механічної кнопки з програмним придушенням деренчання контактів, а також плавне регулювання яскравості за допомогою широтно-імпульсної модуляції (ШИМ).

- **Автоматичне доосвітлення:** зчитування рівня освітленості за допомогою фоторезистора та автоматична активація штучного світла у разі падіння показників нижче встановленого порогу (800 одиниць АЦП).
- **Охоронна сигналізація:** фіксація несанкціонованого руху за допомогою пасивного інфрачервоного датчика (PIR) з подальшим увімкненням звукового оповіщення (п'єзозумер) та світлової індикації.
- **Інтелектуальне годування:** виявлення наближення об'єкта ультразвуковим датчиком відстані (в діапазоні 2–7 см) та автоматичне відкриття заслінки корму за допомогою сервоприводу (поворот на кут від 160° до 80°).
- **Контур автоматичного поливу:** моніторинг вологості ґрунту та рівня води в баку за допомогою аналогових датчиків. Керування водяним насосом через реле.
- **Захист від «сухого ходу»:** негайне блокування роботи водяного насоса у разі критичного зниження рівня води в резервуарі для запобігання поломці обладнання.
- **Локальне відображення даних:** виведення поточної телеметрії та сервісних повідомлень (наприклад, попередження про відсутність води) на текстовий рідкокристалічний дисплей LCD 1602.

5.1.2. Вимоги до апаратного забезпечення

Для реалізації стенду повинен використовуватися набір компонентів на базі *Keyestudio ESP32 Smart Farm Starter Kit*, що включає:

- **Центральний контролер:** KEYESTUDIO ESP32 PLUS (мікропроцесор Xtensa LX6, 240 МГц, підтримка Wi-Fi 802.11 b/g/n).
- **Датчики:** цифровий датчик температури та вологості повітря DHT11, фоторезистор (датчик світла), аналоговий датчик

вологості ґрунту, датчик рівня води, ультразвуковий далекомір HC-SR04, ІЧ-датчик руху PIR.

- **Актuatorи:** модуль електромеханічного реле (5V), мікросервопривод Tower Pro SG90, пасивний зумер, світлодіодні модулі.
- **Індикація:** дисплей LCD 1602 з I2C-інтерфейсом.
- **Комунікація:** роз'єми типу RJ11 (EASY Plug) для надійного з'єднання модулів.

5.1.3. Вимоги до програмного та мережевого забезпечення

- **Середовище розробки:** Arduino IDE (мова програмування C++).
- **Мережевий режим:** ESP32 функціонує в режимі бездротової станції (STA) та підключається до локальної Wi-Fi мережі. Бібліотеки ESP32 Wi-Fi, Servo, LiquidCrystal, WebServer.
- **Веб-інтерфейс:** розгортання вбудованого HTTP-сервера на базі бібліотеки <WebServer.h>. Веб-сторінка повинна віддавати дані датчиків у текстовому/графічному форматі та містити елементи керування (кнопки) для перемикання станів реле, вентилятора, сервоприводу та зумера.
- **Мобільна інтеграція:** підтримка обміну даними через сокет-з'єднання з кросплатформним мобільним додатком (APP) для оперативного двостороннього зв'язку (дзеркалювання функцій керування та моніторингу).

5.1.4. Вимоги до надійності та безпеки

Програмне забезпечення мікроконтролера повинно коректно обробляти помилки зчитування з датчиків (наприклад, повернення значення NaN від DHT11) без зависання всієї системи.

Живлення стенду має здійснюватися від безпечної постійної напруги 5 В через порт USB або зовнішній блок живлення.

Схеми підключення силових елементів (насос, реле) повинні бути гальванічно ізольовані або працювати в межах слабкострумівих безпечних ліній.

6. ТЕХНІКО-ЕКОНОМІЧНІ ВИМОГИ ДО КІНЦЕВОГО ПРОДУКТУ

Вартість робіт по розробці даної ПР визначається згідно договору на розробку. Вартість запропонованих аналогів повинна забезпечити економічну доцільність їх застосування.

7. ВИМОГИ ДО МАТЕРІАЛІВ ТА КОМПЛЕКТУЮЧИХ

7.1. Вимоги до екологічної безпечності під час експлуатації.

Не пред'являються.

7.2. Спеціальні вимоги до кінцевого продукту.

Не пред'являються.

7.3. Вимоги до безпеки для населення під час експлуатації продукції.

Не пред'являються.

8. ЕТАПИ ВИКОНАННЯ ПР.

Етапи виконання ПР можуть уточнювати згідно календарного плану робіт по узгодженню між замовником та виконавцем:

- **Аналітичний етап:** Огляд літератури, аналіз технологій IoT та формування архітектури системи.
- **Схемотехнічний етап:** Проектування схем підключення датчиків та виконавчих пристроїв до розпіновки (pinout) ESP32.
- **Етап програмної розробки:**
 - Написання драйверів та функцій для локального зчитування даних і логіки вбудованих підсистем.
 - Розробка коду HTTP-веб-сервера та інтеграція HTML/CSS інтерфейсу.
 - Налаштування сокет-з'єднання для комунікації з мобільним додатком.

- **Тестування та налагодження:** Комплексна перевірка роботи стенду в різних режимах, симуляція аварійних ситуацій (відсутність води, рух уночі), виправлення помилок (debugging).

№	Етапи виконання роботи	Термін виконання та обсяг робіт	звітні матеріали
1	Аналіз розробки програмного комплексу та розробка першої версії. Аналіз вимог. Розробка структури. Попереднє тестування		Частковий програмний комплекс на ЕОМ замовника, що виконує всі основні функції та звітна документація п.8.2
2	Коректування структури. Розробка допоміжних функцій. Розробка остаточної версії програмного комплексу та його опрацювання. Тестування		Готовий програмний комплекс на ЕОМ замовника та звітна документація п.8.2
3	Доопрацювання окремих модулів та навчання користувачів. Розробка звітних матеріалів згідно п.8 цього ТЗ		звітні матеріали згідно пункту 8

9. ПРИЙМАННЯ

7.1. Необхідні вимоги для впровадження ПР та завершення робіт.

Оцінка результатів розробки і доцільність її продовження здійснюється замовником по представленню наступних матеріалів:

- встановлений програмний комплекс на ЕОМ замовника;
- перелік файлів на резервному носії;
- стислий опис роботи ПР та опис всіх файлів, які необхідні для роботи ПР.
- перелік документів
 - Технічне завдання
 - Пояснювальна записка

7.2. Перелік звіних документів, необхідних для прийняття етапів роботи:

- стислий опис результатів етапу у вигляді анотованого звіту(для 1 та 2 етапів);
- частковий програмний комплекс на ЕОМ замовника згідно календарного плану робіт;
- акт приймання продукції.

7.3. Загальний перелік до приймання звітних документів, макетів, експериментальних зразків.

До приймання пред'являються: акт здачі-приймання продукції, акт впровадження ПР.

7.4. Тестування ПР

Тестування виконується до "Програми та методики тестування", яка розробляється виконавцем та затверджується замовником

10. ПОРЯДОК ВНЕСЕННЯ ЗМІН ДО ТЕХНІЧНОГО ЗАВДАННЯ, ЩО ЗАТВЕРДЖЕНО.

Дане технічне завдання може уточнюватися в процесі розробки ПР при узгодженні сторін з оформленням доповнень до ТЗ.

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЗ «ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА»**

Факультет технологій та інформаційних систем

(назва факультету, інституту)

Інформаційних технологій та систем

(назва кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

БАКАЛАВРА

(освітньо-кваліфікаційний рівень)

**на тему: РОЗРОБКА ЛАБОРАТОРНОГО СТЕНДУ ІоТ "РОЗУМНА
ФЕРМА" НА ESP32**

Виконав: студент 4 курсу

**Спеціальності 121 «Інженерія програмного
забезпечення»**

(шифр і назва напрямку підготовки, спеціальності)

Мартинюк І. Б.

(прізвище та ініціали)

Керівник

Донченко В.Ю.

(прізвище та ініціали)

Рецензент

Козуб Ю.Г.

(прізвище та ініціали)

Лубни – 2026 року

ЗМІСТ

ВСТУП.....	5
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕНЬ.....	10
1.1. Огляд сучасних IoT технологій та їх використання в лабораторних стендах.....	10
1.2. Огляд існуючих рішень для моніторингу та управління умовами навколишнього середовища	12
1.2.1. Інтелектуальні системи моніторингу та управління для будівель і приміщень	12
1.2.2. Розумні фермерські системи	14
1.2.3. Освітні та лабораторні рішення для моніторингу та управління середовищем	16
1.2.4. Промислові IoT-рішення для контролю середовища.....	19
1.3. Формування технічних вимог та концептуальної архітектури розроблюваної системи «Розумна ферма»	21
1.3.1. Технічні вимоги до лабораторного стенду	21
1.3.2. Концептуальна архітектура системи	22
Висновки до розділу.....	25
РОЗДІЛ 2. ВИБІР КОМПОНЕНТІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЛАБОРАТОРНОГО СТЕНДУ "РОЗУМНА ФЕРМА" НА ESP32	28
2.1. Огляд основних компонентів системи	28
2.1.1. Плата розробника ESP32 PLUS на WROOM-32 WIFI+Bluetooth	28
2.1.2. Модуль цифрової кнопки EASY Plug RJ11	38
2.1.3. Модуль світлодіода Keystudio LED Module	39
2.1.4. Датчик руху PIR Motion Sensor Module	41
2.1.5. Пасивний дзвінок (Passive Buzzer Module).....	43
2.1.6. Серводвигун SG90.....	46
2.1.7. Ультразвуковий датчик Ultrasonic Sensor HC-SR04	49

2.1.8. Реле електромеханічне SRD-5VDC-SL-C	53
2.1.9. Дисплей LCD 1602	55
2.2. Вибір середовища програмної розробки.....	58
Висновки до розділу.....	63
РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ЛАБОРАТОРНОГО СТЕНДУ	
ІоТ "РОЗУМНА ФЕРМА" НА ESP32	66
3.1. Модуль системи освітлення	66
3.1.1. Засвітити світлодіод	66
3.1.2. Керування світлодіодом за допомогою ШИМ	67
3.1.3. Прочитати цифрове значення Button.....	68
3.1.4. Кнопка автоматичного блокування	70
3.1.5. Використання кнопки для керування світлодіодним модулем	71
3.2. Модуль системи керування світлом	72
3.2.1. Фотокомірковий сенсор	72
3.2.2. Система керування світлом	74
3.3. Модуль системи сигналізації	76
3.3.1. Датчик руху PIR.....	76
3.3.2. Пасивний дзвінок	77
3.3.3. Система сигналізації	79
3.4. Модуль розумної системи годування тварин	81
3.4.1. Двері живильної kabіни	81
3.4.2. Інтелектуальна система годування	83
3.5. Модуль системи автоматичного зрошення	85
3.5.1. Система відкачування води	85
3.5.2. Система автоматичного зрошення.....	87
3.6. Розробка веб-керуванням розумної ферми.....	90
3.6.1. Підключення плати ESP32 до мережі	90
3.6.2. Створення веб-сайту	92
3.7. Розробка APP додатку для керування розумної ферми	97
Висновки до розділу.....	100

ВИСНОВКИ	104
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	107
ДОДАТОК А.....	111
ДОДАТОК Б	118

ВСТУП

Стрімкий розвиток цифрових технологій, поширення концепції Індустрії 4.0 та зростання кількості підключених до мережі пристроїв зумовлюють підвищену увагу до систем Інтернету речей (Internet of Things, IoT). У сучасній освітній та інженерній практиці зростає інтерес до систем віддалених лабораторій (remote labs) із використанням технологій Internet of Things. Зокрема, такі системи дозволяють студентам працювати з реальними пристроями та середовищами, перебуваючи фізично поза лабораторією, що підвищує гнучкість навчання та розширює доступ до обладнання. Вивчення принципів побудови, функціонування та керування системами IoT є важливою складовою підготовки фахівців у галузі інформаційних технологій та автоматизації. Опанування IoT-технологіями дозволяє формувати у студентів компетенції, пов'язані з проєктуванням, налаштуванням та віддаленим керуванням інтелектуальними пристроями й мережею елементів IoT.

Особливого значення ці технології набувають в агропромисловому секторі (Smart Farming), де автоматизація контролю мікроклімату, раціональне управління водними ресурсами та автоматизація процесів життєзабезпечення тварин і рослин дозволяють суттєво підвищити врожайність, мінімізувати вплив людського фактора та оптимізувати операційні витрати.

Водночас важливим завданням закладів вищої освіти є підготовка фахівців, здатних проєктувати, налаштовувати та експлуатувати сучасні IoT-системи. Для формування практичних навичок студентів необхідні спеціалізовані лабораторні стенди, які дають можливість досліджувати принципи роботи датчиків, виконавчих механізмів, мережевих протоколів та засобів віддаленого керування.

В наш час активно досліджується забезпечення віддаленого доступу до спеціалізованого обладнання з Інтернету речей на базі мікроконтролерів. Проблематика розроблення віддалених лабораторій неодноразово розглядалася у наукових дослідженнях. Зокрема у статті Г. Джеваги та К. Яковлева [1] описано методику побудови такої лабораторії на основі

мікроконтролерів Arduino, програмного забезпечення AnyDesk і засобів візуального контролю через веб-камеру. Це дозволяє здобувачам освіти віддалено програмувати мікроконтролери, спостерігати за змінами станів сенсорів та актуаторів у реальному часі, а також набувати практичних навичок у роботі з реальним апаратним забезпеченням. У роботі [2] описано реалізацію лабораторії дистанційного навчання керуванню Інтернет речами, в якій використано мікроконтролери на базі мікросхем STM32, модуль ESP8266 і серверну архітектуру для дистанційного контролю студентських платформ. Використання таких лабораторій суттєво поліпшує виконання завдань студентами в порівнянні з традиційним online-навчанням. У дослідженні [3] зазначається, що сучасні віддалені лабораторії є важливою частиною цифровізації освітнього процесу для забезпечення доступу здобувачів до фізичних лабораторій з реальним обладнанням під час практичних online-занять. Запропонований підхід застосування Raspberry Pi та програмного забезпечення віддаленого робочого місця у праці [4] дає змогу студентам ефективно організувати віддалений доступ до лабораторного обладнання з інженерних дисциплін під час дистанційного навчання. Це рішення поєднує простоту реалізації, взаємодію з реальним фізичним обладнанням і низьку вартість порівняно з використанням промислових програмованих логічних контролерів. У дослідженні [5] наголошується, що для того, щоб надати учасникам дистанційного навчання реалістичний практичний досвід роботи в лабораторії, можна використовувати віддалений доступ до реальної робочої лабораторії. У роботі [6] розглядається сучасне цифрове середовище на базі штучного інтелекту для керування системою Інтернет речей. Оскільки одноплатні комп'ютери мають обмежені ресурси, тому перед їх впровадженням необхідно проводити бенчмаркінг пристроїв для оцінки їхньої придатності та продуктивності.

Актуальність роботи зумовлена потребою створення доступного навчального комплексу, який поєднує сучасні технології Інтернету речей, засоби автоматизації та веборієнтоване керування. Використання

мікроконтролера ESP32 дозволяє реалізувати бездротовий обмін даними, інтегрувати різноманітні датчики та виконавчі пристрої, а також забезпечити взаємодію користувача із системою через вебінтерфейс і мобільний додаток.

Об'єктом дослідження є процеси моніторингу та автоматизованого керування параметрами середовища в системах розумного фермерства за допомогою технологій Інтернету речей.

Предметом дослідження є архітектура, апаратні компоненти, алгоритмічне та програмне забезпечення лабораторного стенду IoT «Розумна ферма» на базі мікроконтролера ESP32 з можливістю веб-керування та мобільної інтеграції.

Метою роботи є розробка лабораторного стенду IoT «Розумна ферма» на базі ESP32, який забезпечує збір даних із датчиків, керування виконавчими пристроями, реалізацію автоматизованих сценаріїв роботи та віддалений доступ через вебінтерфейс і мобільний застосунок.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Проаналізувати сучасний стан розвитку IoT-технологій, існуючі промислові, освітні та комерційні рішення для моніторингу навколишнього середовища й автоматизації фермерських господарств.
2. Обґрунтувати вибір центрального обчислювального ядра (мікроконтролера) та сформулювати оптимальну специфікацію сенсорів і виконавчих пристроїв для побудови лабораторного стенду.
3. Розробити структурні та принципові схеми підключення периферійних модулів до обраної плати розробника.
4. Вибрати та налаштувати інтегроване середовище програмної розробки.
5. Програмно реалізувати та протестувати окремі функціональні модулі стенду: підсистему освітлення, контур автоматичного

регулювання світла, систему охоронної сигналізації, інтелектуальний вузол годування тварин та замкнуту систему автоматичного зрошення.

6. Розробити мережеве програмне забезпечення для розгортання локального HTTP-веб-сервера та налаштувати взаємодію з кросплатформним мобільним APP-додатком для віддаленого моніторингу телеметрії та двостороннього керування стендом.

Методи досліджень базуються на принципах системного аналізу, теорії автоматичного керування, методах схемотехнічного проєктування мікроконтролерних пристроїв, об'єктно-орієнтованому програмуванні мовою C++ (в середовищі Arduino IDE), а також на використанні мережевих протоколів передачі даних TCP/IP (HTTP, сокет-з'єднання).

Практичне значення отриманих результатів полягає у створенні діючого фізичного прототипу лабораторного стенду на основі комплекту Keyestudio ESP32 Smart Farm Starter Kit, який може бути безпосередньо впроваджений в освітній процес кафедри інформаційних технологій та систем для виконання практичних і лабораторних робіт з дисциплін, пов'язаних із проєктуванням IoT, вбудованих систем та автоматизації.

У першому розділі здійснено детальний аналіз предметної області та сучасного стану розвитку технологій Інтернету речей (IoT) в аграрному секторі (Smart Farming). Проведено критичний огляд існуючих комерційних, промислових та навчально-лабораторних рішень для моніторингу й управління параметрами навколишнього середовища. Обґрунтовано актуальність розробки та сформовано технічні вимоги до комплексної системи автоматизації мікроклімату й процесів життєзабезпечення розумної ферми.

У другому розділі проведено порівняльний аналіз поширених апаратних платформ і обґрунтовано вибір двоядерного мікроконтролера ESP32 (на базі плати розробника KEYESTUDIO ESP32 PLUS) як центрального обчислювального ядра системи. Сформовано повну специфікацію периферійного обладнання, датчиків (руху, відстані, вологості, освітленості)

та актуаторів (реле, сервоприводу, зумера, світлодіода), а також розроблено схеми їхнього підключення та взаємодії. Обґрунтовано вибір інтегрованого середовища розробки Arduino IDE та описано базову структуру побудови програмного коду.

У третьому розділі виконано практичну апаратну реалізацію та програмне забезпечення ключових функціональних модулів лабораторного стенду. Розроблено та протестовано алгоритми для підсистеми автоматичного ШІМ-керування освітленням, охоронної сигналізації, інтелектуального вузла годування тварин на основі ультразвукового далекоміра, а також замкнутого контуру автоматичного зрошення із вбудованим захистом водяного насоса від «сухого ходу». За допомогою вбудованого Wi-Fi модуля ESP32 розгорнуто локальний HTTP-веб-сервер та налаштовано двосторонню взаємодію з мобільним APP-додатком для віддаленого моніторингу телеметрії та дистанційного керування всіма виконавчими механізмами стенду в реальному часі.

РОЗДІЛ 1.

АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕНЬ

1.1. Огляд сучасних IoT технологій та їх використання в лабораторних стендах

Інтернет речей (IoT) являє собою інтеграцію обчислювальних пристроїв, сенсорів, програмного забезпечення і мережових технологій для збору та обміну даними між об'єктами через Інтернет. Це дозволяє створювати інтелектуальні системи, що автоматично адаптуються до зовнішніх умов і виконують функції збору, обробки та аналізу даних у реальному часі. IoT технології у лабораторних стендах дають змогу отримувати постійний доступ до даних експерименту, автоматизувати вимірювання та керувати процесами дистанційно, що є особливо корисним в умовах сучасних вимог до безпеки та гнучкості науково-дослідницької роботи [5].

Переваги IoT для лабораторних стендів:

1. Автоматизація процесів: за допомогою мікроконтролерів та сенсорів можливо реалізувати автоматичний збір даних, а також автономне управління лабораторними процесами (наприклад, підтримання температури чи рівня вологості).
2. Дистанційний контроль і управління: підключення до мережі Інтернет дозволяє керувати лабораторним обладнанням з будь-якого місця. Наприклад, дослідники можуть запускати або зупиняти експерименти, контролювати стан приладів і змінювати параметри системи віддалено.
3. Масштабованість: завдяки використанню модульних IoT-рішень лабораторний стенд можна доповнювати новими компонентами, сенсорами або модулями, що збільшує гнучкість і можливості системи.
4. Зберігання та аналіз даних у хмарних середовищах: результати експериментів можуть бути збережені в хмарних базах даних і легко доступні для аналізу, що дозволяє дослідникам отримувати нові інсайти.

Основні компоненти IoT-систем у лабораторних стендах:

1. Мікроконтролери та одноплатні комп'ютери;

IoT-системи для лабораторних стендів найчастіше базуються на мікроконтролерах, що забезпечують необхідну обчислювальну потужність і мають різні можливості підключення.

2. Сенсори для моніторингу параметрів середовища;

У лабораторних IoT-системах використовуються різні сенсори для збору даних про параметри навколишнього середовища. Це дає змогу забезпечити точний моніторинг умов проведення експерименту:

3. Протоколи передачі даних;

Для ефективної передачі даних між IoT-пристроями та серверами, а також для інтеграції пристроїв з різноманітними системами використовуються спеціалізовані протоколи. Вибір протоколу залежить від потреб у швидкості передачі, енергоспоживанні, ресурсах мережі та інших характеристиках.

4. Хмарні сервіси та платформи;

Хмарні платформи є важливим елементом IoT-систем, оскільки вони забезпечують зберігання, обробку та аналіз великих обсягів даних, що генеруються IoT-пристроями. Вони також надають інструменти для моніторингу, управління пристроями та створення складних аналітичних моделей.

IoT технології значно підвищують можливості лабораторних стендів, дозволяючи проводити експерименти у режимі реального часу, аналізувати дані та керувати системою дистанційно. Основні напрями використання IoT у лабораторних стендах:

1. Моніторинг та автоматизація експериментальних умов: Лабораторні IoT-системи здатні підтримувати параметри середовища (температура, вологість, освітлення) на постійному рівні, що важливо для експериментів з довготривалим спостереженням. Зібрані дані автоматично зберігаються на серверах або хмарних платформах для подальшого аналізу.

2. Віддалене управління і доступ: Інтеграція з Інтернетом дозволяє користувачам здійснювати контроль і управління стендом через веб-інтерфейс або мобільний додаток. Це особливо актуально для навчальних лабораторій, де студенти можуть проводити експерименти без фізичної присутності, або для віддалених досліджень.
3. Моделювання та симуляція умов: Лабораторні IoT-системи можуть бути налаштовані для створення штучних умов, що імітують різні ситуації, такі як зміна клімату чи коливання рівня освітленості. Це корисно для наукових експериментів, коли потрібно дослідити вплив різних параметрів на об'єкт.
4. Аналітика та збереження історичних даних: Системи з хмарним зберіганням даних надають можливість обробляти результати експериментів, будувати графіки змін та зберігати інформацію для подальших порівнянь. Це значно полегшує створення звітів і аналіз отриманих результатів.

1.2. Огляд існуючих рішень для моніторингу та управління умовами навколишнього середовища

Моніторинг і управління умовами навколишнього середовища стали однією з ключових сфер застосування технологій інтернету речей (IoT), оскільки дозволяють ефективно збирати дані про параметри довкілля, передавати їх для обробки і автоматично реагувати на зміни в реальному часі. Розглянемо детальніше типи існуючих рішень та їх особливості застосування в різних галузях.

1.2.1. Інтелектуальні системи моніторингу та управління для будівель і приміщень

Системи моніторингу для будівель здатні значно підвищити комфорт і знизити витрати на енергоносії, а також забезпечити безпеку і стабільність умов середовища, що є важливим для офісних приміщень, закладів охорони здоров'я, шкіл тощо. Основні функції таких рішень включають контроль за

температурою, вологістю, освітленням, рівнем CO₂, а також виявлення присутності людей для оптимізації ресурсів.

Nest Thermostat від Google. Це інтелектуальний термостат, який використовує технології машинного навчання для автоматичного регулювання температури в будинку. Nest вивчає поведінку мешканців, їхні звички та час перебування в приміщенні, а також враховує погодні умови, щоб оптимізувати температуру в будинку. Завдяки цій адаптивності термостат може значно знижувати витрати на енергію, зменшуючи опалення або охолодження, коли це не потрібно. Користувачі також можуть контролювати термостат за допомогою мобільного додатку.



Рис. 1.1. Nest Thermostat

Honeywell Home T9. Цей розумний термостат оснащений вбудованими сенсорами, які дозволяють відстежувати температуру та вологість в різних кімнатах, забезпечуючи більш точний контроль клімату в будинку. Завдяки можливості налаштовувати умови для різних зон, Honeywell T9 допомагає підтримувати комфортну температуру в кожній частині будинку, що дозволяє створювати індивідуальні кліматичні умови для спальні, вітальні або офісу. Термостат також можна налаштувати через мобільний додаток, а його функціонал включає використання геолокації для автоматичного включення

або вимикання кліматичних систем залежно від місця перебування користувача.



Рис. 1.2. Honeywell Home T9

Такі системи підтримують підключення до мобільних додатків, що дозволяє користувачам отримувати сповіщення про зміни в умовах та здійснювати управління дистанційно. Це робить їх надзвичайно корисними для будівель з великим потоком людей та змінними умовами середовища.

1.2.2. Розумні фермерські системи

Системи для сільського господарства активно використовуються для автоматизації процесів поливу, контролю рівня освітленості, внесення добрив та захисту врожаю від погодних умов. Багато сучасних фермерських рішень базуються на IoT-платформах, що надають можливість фермеру в реальному часі спостерігати за станом рослин та ґрунту і налаштовувати управління ресурсами відповідно до умов [6].

John Deere Field Connect. Це інноваційне рішення для сільського господарства, яке використовує IoT-технології для моніторингу та управління агрономічними процесами. Field Connect включає різноманітні сенсори для вимірювання вологості ґрунту, температури повітря, вологості та інших

важливих параметрів. Зібрані дані передаються на сервер, де фермери можуть переглядати їх у реальному часі.



Рис. 1.3. Портативний дисплей John Deere

Це дозволяє їм приймати обґрунтовані рішення щодо поливу, застосування добрив, обробки рослин та інших агрономічних заходів, що допомагає підвищити ефективність сільськогосподарського виробництва і оптимізувати витрати.

Netafim Precision Irrigation. Система точного поливу, що використовує IoT для збору даних про стан ґрунту, температуру, вологість і рівень сонячного випромінювання, забезпечує високоточне управління поливом. Система Netafim дозволяє регулювати полив для кожного окремого ряду рослин, що зменшує витрату води і запобігає її надмірному використанню. Це не лише допомагає зберігати ресурси, але й оптимізує умови для росту рослин, підвищуючи врожайність і знижуючи витрати на водні ресурси. Технологія також сприяє стійкості сільського господарства до змін клімату.



Рис. 1.4. Netafim Precision Irrigation

Завдяки таким рішенням фермери можуть оптимізувати витрати води та ресурсів, що особливо важливо в умовах глобального потепління і нестачі водних ресурсів. Водночас використання IoT-технологій дозволяє значно зменшити трудозатрати і підвищити ефективність управління фермою.

1.2.3. Освітні та лабораторні рішення для моніторингу та управління середовищем

Навчальні комплекти IoT для моніторингу середовища стають усе популярнішими в навчальних закладах, оскільки дозволяють студентам отримати практичні навички в роботі з IoT-пристроями, сенсорами та системами автоматизованого управління. Лабораторні комплекти зазвичай включають мікроконтролери, різноманітні сенсори та програмне забезпечення для створення власних IoT-рішень.

Keyestudio ESP32 Smart Farm Starter Kit. Основою стенда (рис. 1.5) є мікроконтролер ESP32, який підтримує бездротові протоколи зв'язку, такі як Wi-Fi та Bluetooth, що робить його зручним для інтеграції з хмарними сервісами та мобільними додатками [1]. Стенд забезпечує можливість роботи з різноманітними датчиками для вимірювання параметрів навколишнього середовища, зокрема вологості ґрунту, температури, рівня вологості повітря та освітленості. Завдяки цьому можна моделювати умови, максимально

наближені до реальних, що особливо актуально для вивчення впливу різних факторів на ріст рослин.

Автоматизація є ключовою функцією цього стенда. Наприклад, на основі отриманих даних система може автоматично вмикати чи вимикати полив, освітлення або вентиляцію. Інтелектуальні алгоритми прийняття рішень, запрограмовані на базі контролера, дають можливість підвищити ефективність використання ресурсів. Додатково користувач має можливість налаштування порогових значень параметрів, що дозволяє змінювати поведінку системи залежно від поставлених завдань.



Рис. 1.5. Keyestudio ESP32 Smart Farm Starter Kit

Завдяки модульній структурі стенд легко адаптується під різні дослідницькі задачі. Його конструкція дозволяє доповнювати систему новими датчиками або виконавчими механізмами, розширюючи функціонал без

необхідності значних технічних змін. Це робить стенд корисним для розробки прототипів реальних рішень у сфері сільського господарства, які можуть бути масштабовані для впровадження в комерційних системах.

Лабораторний стенд також сприяє інтерактивному навчанню. Студенти можуть отримати практичні навички роботи з апаратними та програмними компонентами, освоїти методи збору, обробки та аналізу даних. Інтеграція із сучасними програмними платформами дозволяє вивчати принципи розробки додатків для моніторингу та управління в реальному часі.

Загалом, стенд на базі Keyestudio ESP32 Smart Farm Starter Kit є ефективним інструментом для розвитку навичок і знань у сфері IoT, автоматизації та розумного сільського господарства. Його використання сприяє впровадженню інноваційних рішень, що відповідають викликам сучасного агропромислового сектору.

Grove IoT Starter Kit від Seeed Studio. Цей комплект на основі платформи Arduino та ESP32 включає різноманітні сенсори для вимірювання температури, вологості, атмосферного тиску, освітленості тощо. Grove IoT Starter Kit також містить модулі для підключення до Wi-Fi та інтеграції з хмарними платформами, що дозволяє створювати рішення для віддаленого моніторингу та керування пристроями. Цей набір ідеально підходить для студентів і початківців, які хочуть навчитися створювати IoT-проекти та інтегрувати свої пристрої з інтернетом для віддаленого моніторингу і автоматизації.

Освітні набори часто включають готові приклади та проекти, що дозволяють студентам швидко освоїти принципи роботи з IoT-системами, отримати навички програмування, а також розробити власні рішення для моніторингу та управління умовами навколишнього середовища.



Рис. 1.6. Grove IoT Starter Kit від Seeed Studio

1.2.4. Промислові IoT-рішення для контролю середовища

У промислових умовах системи моніторингу середовища забезпечують стабільність технологічних процесів, зберігання продукції та зменшення ризиків для безпеки персоналу. Промислові IoT-рішення зазвичай включають високоточні сенсори, стійкі до складних умов, та інтегровані з програмними платформами для обробки великих обсягів даних.

Schneider Electric EcoStruxure. Це потужна платформа для управління промисловими об'єктами, що дозволяє підприємствам контролювати ключові параметри навколишнього середовища, такі як температура, вологість, рівень CO₂, тиск та інші важливі показники. EcoStruxure інтегрує різноманітні сенсори і дає змогу здійснювати моніторинг в реальному часі, а також використовує аналітику даних для оптимізації виробничих процесів, зниження витрат на енергоспоживання та підвищення ефективності. Ця платформа дозволяє здійснювати віддалений моніторинг, автоматизувати управління обладнанням і покращувати операційну ефективність.



Рис. 1.7. Schneider Electric EcoStruxure

Siemens MindSphere. Хмарна платформа для промислових IoT-додатків, що об'єднує сенсори, технологічне обладнання та виробничі процеси для збору даних і їх візуалізації. MindSphere дозволяє підприємствам відстежувати і контролювати виробничі процеси в реальному часі, що забезпечує покращене прийняття рішень і оптимізацію виробництва. Платформа підтримує інтеграцію з різними пристроями та системами, використовуючи аналітичні інструменти для передбачення та оптимізації робочих процесів, а також забезпечує високий рівень безпеки та масштабованості для великих промислових застосувань.

Такі рішення дозволяють компаніям не лише збирати інформацію про навколишнє середовище, а й підвищувати ефективність виробничих процесів за рахунок оптимізації витрат ресурсів, автоматизації процесів і зменшення кількості відходів.



Рис. 1.8. Siemens MindSphere

1.3. Формування технічних вимог та концептуальної архітектури розроблюваної системи «Розумна ферма»

На основі проведеного аналізу сучасного стану розвитку технологій Інтернету речей та огляду існуючих комерційних і навчальних рішень, виникає потреба у проектуванні комплексної системи, яка б поєднувала в собі високу функціональність, низьку собівартість та наочність для освітнього процесу. Для успішної реалізації лабораторного стенду «Розумна ферма» необхідно чітко формалізувати технічні вимоги до системи та розробити її концептуальну архітектуру.

1.3.1. Технічні вимоги до лабораторного стенду

Технічні вимоги до розроблюваної системи поділяються на функціональні (визначають перелік завдань, які виконує система) та нефункціональні (визначають експлуатаційні обмеження, вимоги до надійності та інтерфейсів).

Функціональні вимоги:

1. *Моніторинг параметрів середовища:* забезпечення безперервного збору даних про температуру та вологість повітря, рівень освітленості, вологість ґрунту та рівень води в резервуарі.
2. *Автоматизація освітлення:* наявність можливості ручного керування освітленням за допомогою механічної кнопки (із програмним захистом від деренчання контактів) та автоматичного увімкнення доосвітлення при падінні рівня природного світла нижче критичного порогу.
3. *Охоронна функція:* виявлення руху в зоні контролю за допомогою інфрачервоного сенсора з миттєвою активацією світло-звукової сигналізації.
4. *Інтелектуальне годування:* безконтактне визначення наближення об'єкта до годівниці та автоматичне відкриття заслінки за допомогою приводу.

5. *Автоматичне зрошення та захист обладнання*: керування водяним насосом залежно від рівня сухості ґрунту, а також реалізація алгоритму аварійного блокування насоса у разі відсутності води в баку для запобігання «сухому ходу».
6. *Мережева взаємодія*: розгортання вбудованого веб-сервера для доступу через HTTP-протокол та підтримка обміну даними з мобільним додатком.

Нефункціональні вимоги:

1. *Надійність*: стійкість програмного забезпечення мікроконтролера до помилок зчитування даних (обробка некоректних значень сенсорів без зависання системи).
2. *Ергономічність та наочність*: наявність локального текстового дисплея для відображення критичних параметрів безпосередньо на стенді та інтуїтивно зрозумілого графічного інтерфейсу користувача у мобільному додатку та на веб-сторінці.
3. *Безпека*: використання низьковольтного живлення (5 В), що є критично важливим для безпечної експлуатації стенду студентами в лабораторних умовах.

1.3.2. Концептуальна архітектура системи

Для забезпечення гнучкості, масштабованості та модульності системи «Розумна ферма» її концептуальну архітектуру доцільно побудувати за трирівневим принципом (рис. 1.X). Така модель дозволяє чітко розмежувати апаратне забезпечення, логіку обробки даних та інтерфейси взаємодії з користувачем.

1. **Фізичний рівень (Рівень сприйняття та виконання / Perception Layer)**: Це нижній рівень архітектури, який безпосередньо взаємодіє з фізичним середовищем «Розумної ферми». До його складу входять дві основні групи елементів:
 - *Сенсори (датчики)*: DHT11 (температура/вологість), фоторезистор (освітленість), датчик вологості ґрунту,

датчик рівня води, ультразвуковий далекомір HC-SR04 та ІЧ-датчик руху PIR. Вони трансформують фізичні величини в електричні сигнали (аналогові або цифрові).

- *Актuatorи (виконавчі механізми):* модуль електромеханічного реле (для керування насосом/вентилятором), мікросервопривод SG90 (керування годівницею), пасивний п'єзозвуковий зумер та світлодіоди сигналізації. Вони виконують команди, що надходять від керуючого ядра.

2. Процесорно-мережевий рівень (Рівень керування / Control Layer): Центральною ланкою цього рівня є мікроконтролер ESP32. На цьому рівні реалізується вся логіка роботи системи:

- *Програмна обробка:* опитування датчиків з певною періодичністю, фільтрація шумів, переведення аналогових значень АЦП у відсотки або фізичні одиниці, виконання алгоритмів зворотного зв'язку (наприклад, включення реле поливу при низькій вологості).
- *Комунікаційний модуль:* використання вбудованого апаратного Wi-Fi інтернет-модуля для забезпечення бездротового зв'язку. Мікроконтролер конфігурується в режимі станції (STA) для інтеграції в локальну мережу кафедри чи лабораторії.

3. Прикладний рівень (Рівень користувача / Application Layer): Верхній рівень архітектури, що відповідає за представлення даних користувачеві (досліднику, студенту) та передачу його керуючих команд назад до мікроконтролера. Він реалізований двома паралельними інтерфейсами:

- *Локальний інтерфейс:* рідкокристалічний текстовий дисплей LCD 1602 (підключений через шину I2C) для оперативного моніторингу на місці.

–Віддалений інтерфейс (Веб-сервер та Мобільний додаток): адаптивна HTML-сторінка, що генерується безпосередньо ESP32 за HTTP-запитом, та кросплатформний додаток (APP). Вони забезпечують графічну візуалізацію телеметрії у реальному часі та містять інтерактивні елементи (кнопки, перемикачі) для дистанційного керування фермою.

Розроблена концептуальна архітектура забезпечує повну автономність локальних захисних алгоритмів стенду (наприклад, відключення насоса при відсутності води відбудеться навіть за умови втрати Wi-Fi з'єднання) і одночасно надає гнучкі інструменти для віддаленого керування кіберфізичною системою через мережу Інтернет. Це повністю відповідає сучасним тенденціям побудови систем класу Індустрія 4.0 та Інтернет речей.

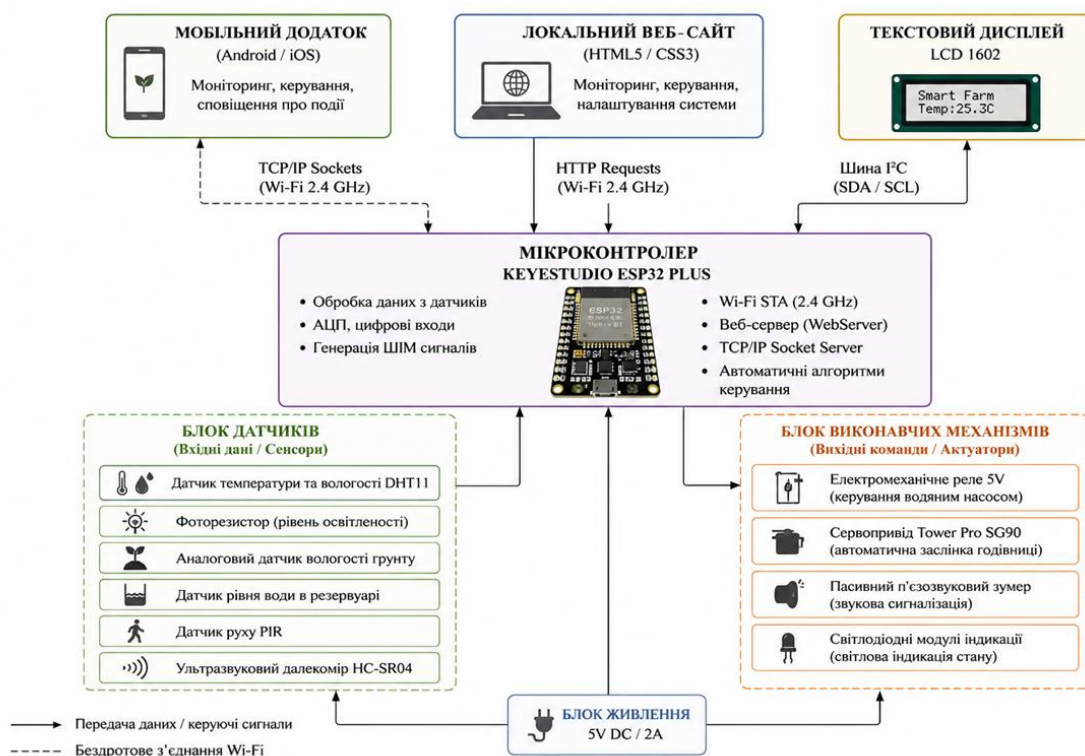


Рис. 1.9. Структурна схема лабораторного стенду IoT "Розумна ферма" на ESP32

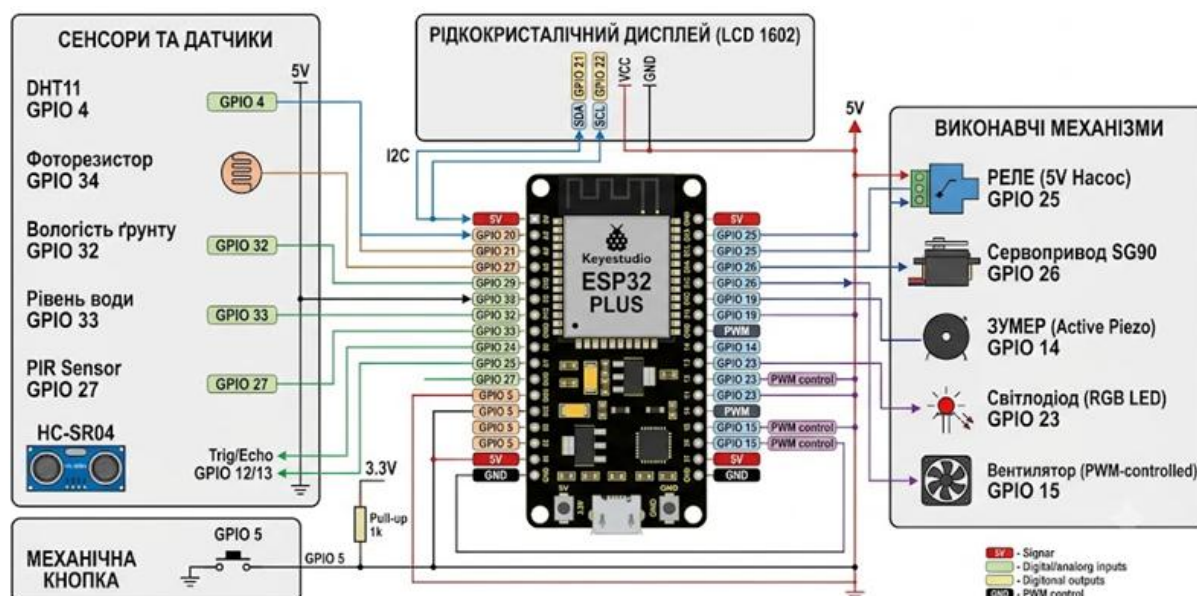


Рис. 1.10. Апаратна архітектура лабораторного стенду IoT "Розумна ферма" на ESP32

Висновки до розділу

У першому розділі було проведено комплексний аналіз предметної області, досліджено сучасний стан розвитку технологій Інтернету речей (IoT) та сформовано теоретико-алгоритмічний базис для створення лабораторного стенду «Розумна ферма».

За результатами виконання завдань першого розділу отримано такі результати:

1. **Досліджено особливості застосування IoT-технологій** у сучасному освітньому процесі. Визначено, що інтеграція мікроконтролерних систем у лабораторну базу дозволяє автоматизувати збір телеметрії, забезпечити наочність фізичних процесів та реалізувати дистанційний доступ до обладнання, що є критично важливим в умовах змішаного навчання.
2. **Виконано порівняльний аналіз існуючих рішень** для моніторингу та управління екосистемами (як промислових платформ від Siemens та Schneider Electric, так і готових навчальних наборів). На основі аналізу зроблено висновок, що готові комерційні системи є дорогими та закритими для

програмної модифікації, а базові конструктори потребують проектування індивідуальної архітектури та написання власного програмного забезпечення, що й обґрунтовує доцільність виконання даного дипломного проекту.

3. **Сформульовано та систематизовано технічні вимоги** до проектуваного стенду. Визначено перелік обов'язкових контурів автоматизації (моніторинг температури/вологості, освітлення, полив, безпека та годування). Особливу увагу приділено нефункціональним вимогам, зокрема забезпеченню безпечної експлуатації стенду в лабораторії за рахунок використання низьковольтного живлення (5 В).
4. **Розроблено концептуальну трирівневу архітектуру системи**, яка дозволила чітко розмежувати апаратну частину та програмні інтерфейси:
 - *Фізичний рівень* — об'єднує масив обраних сенсорів (DHT11, PIR, фоторезистор, датчики вологості ґрунту та рівня води, далекомір HC-SR04) та актуаторів (реле, сервопривод SG90, зумер, LED).
 - *Процесорно-керуючий рівень* — базується на мікроконтролері ESP32, який виконує роль керуючого ядра, обробляє сигнали АЦП/ШІМ та забезпечує бездротовий зв'язок через Wi-Fi.
 - *Прикладний рівень* — включає локальний дисплей LCD 1602 та віддалені інтерфейси (вбудований HTTP веб-сервер та мобільний додаток на сокетах).
5. **Спроектовано загальний алгоритм функціонування комплексної IoT-системи**, який описує логіку паралельного опитування датчиків, обробки екстрених подій (наприклад, проникнення в периметр або критичне зниження рівня води) та передачі даних на прикладний рівень. Мною було передбачено

логіку захисту водяного насоса від «сухого ходу», що підвищує надійність системи.

Таким чином, сформовані технічні вимоги, розроблена архітектура та узагальнений алгоритм повністю виконують завдання етапу проєктування і є готовою основою для переходу до другого розділу — вибору елементної бази, розрахунку схем підключення та практичної програмно-апаратної реалізації стенду.

РОЗДІЛ 2.

ВИБІР КОМПОНЕНТІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЛАБОРАТОРНОГО СТЕНДУ "РОЗУМНА ФЕРМА" НА ESP32

2.1. Огляд основних компонентів системи

2.1.1. Плата розробника ESP32 PLUS на WROOM-32 WIFI+Bluetooth

При проєктуванні автоматизованої системи "РОЗУМНА ФЕРМА" одним із ключових аспектів є обґрунтований вибір апаратних компонентів, оскільки від цього безпосередньо залежить функціональність, ефективність та надійність системи. Для забезпечення комплексного контролю параметрів середовища необхідно створити платформу, здатну до стабільного збору, обробки та передачі даних у режимі реального часу.

Апаратна архітектура системи будується на базі взаємопов'язаних компонентів, які забезпечують точність вимірювань, надійність передачі даних та автономність роботи. Основні критерії при виборі складових включають енергоефективність, стійкість до різноманітних умов експлуатації та адаптивність до різних сценаріїв використання. До складу системи входять: мікроконтролер або одноплатний комп'ютер для обробки даних, набір сенсорів для фіксації ключових параметрів стану людини, модулі бездротового зв'язку та джерела живлення, оптимізовані для тривалої автономної роботи. Кожен компонент підбирається таким чином, щоб забезпечити безперебійну роботу системи, мінімізувати втручання людини та максимізувати корисність зібраної інформації для оперативного прийняття рішень.

Особливу увагу слід приділити вибору апаратної платформи, яка виконуватиме роль ядра системи моніторингу. Ефективність збору, обробки та передачі даних значною мірою визначається характеристиками обраного контролера чи комп'ютера. Для IoT-рішень зазвичай застосовують мікроконтролери або одноплатні комп'ютери, тому критеріями їх порівняння є продуктивність, енергоспоживання, надійність та можливість інтеграції з іншими компонентами системи. У результаті аналізу доступних апаратних

платформ обирається оптимальна конфігурація, що забезпечує реалізацію високоефективної системи моніторингу стану людини.

Arduino є однією з найпопулярніших платформ для розробки електронних проєктів та прототипів. Вона характеризується широким набором доступних бібліотек і модулів для роботи з різними сенсорами, що спрощує інтеграцію периферійних пристроїв.

Однак Arduino не має вбудованих можливостей для бездротового зв'язку, тому для підключення до мереж Wi-Fi або Bluetooth необхідно використовувати додаткові модулі, такі як ESP8266 або HC-05. Це ускладнює апаратну архітектуру системи, знижує її компактність та енергоефективність порівняно з іншими платформами.

Raspberry Pi – це одноплатний комп'ютер із високою обчислювальною потужністю, що дозволяє запускати повноцінні операційні системи (наприклад, Linux) та реалізовувати складніші функціональні можливості, включно з обробкою й зберіганням великих обсягів даних. Незважаючи на це, його значні розміри, висока вартість та підвищене енергоспоживання обмежують застосування в автономних системах моніторингу на віддалених пасіках без постійного джерела живлення.

ESP8266 є економічним та менш потужним мікроконтролером у порівнянні з ESP32, однак володіє вбудованим модулем Wi-Fi, що робить його зручним для простих IoT-проєктів. Основними перевагами є доступність та легкість інтеграції. Водночас відсутність Bluetooth та обмежена обчислювальна потужність знижують ефективність ESP8266 у складних системах, що потребують одночасної обробки даних із численних сенсорів.

ESP32 є однією з найпопулярніших платформ для розробки IoT-систем завдяки двоядерному процесору, що забезпечує високопродуктивну паралельну обробку даних. Вбудовані модулі Wi-Fi та Bluetooth дозволяють організувати бездротову передачу даних без необхідності додаткових компонентів, забезпечуючи інтеграцію з віддаленими серверами та мобільними додатками. Низьке енергоспоживання робить ESP32 оптимальним

для автономних систем, що працюють від акумуляторів. Крім того, наявність різноманітних периферійних інтерфейсів (I2C, SPI, UART) дозволяє підключати широкий спектр сенсорів для моніторингу температури, вологості, ваги та інших параметрів бджолиних вуликів.

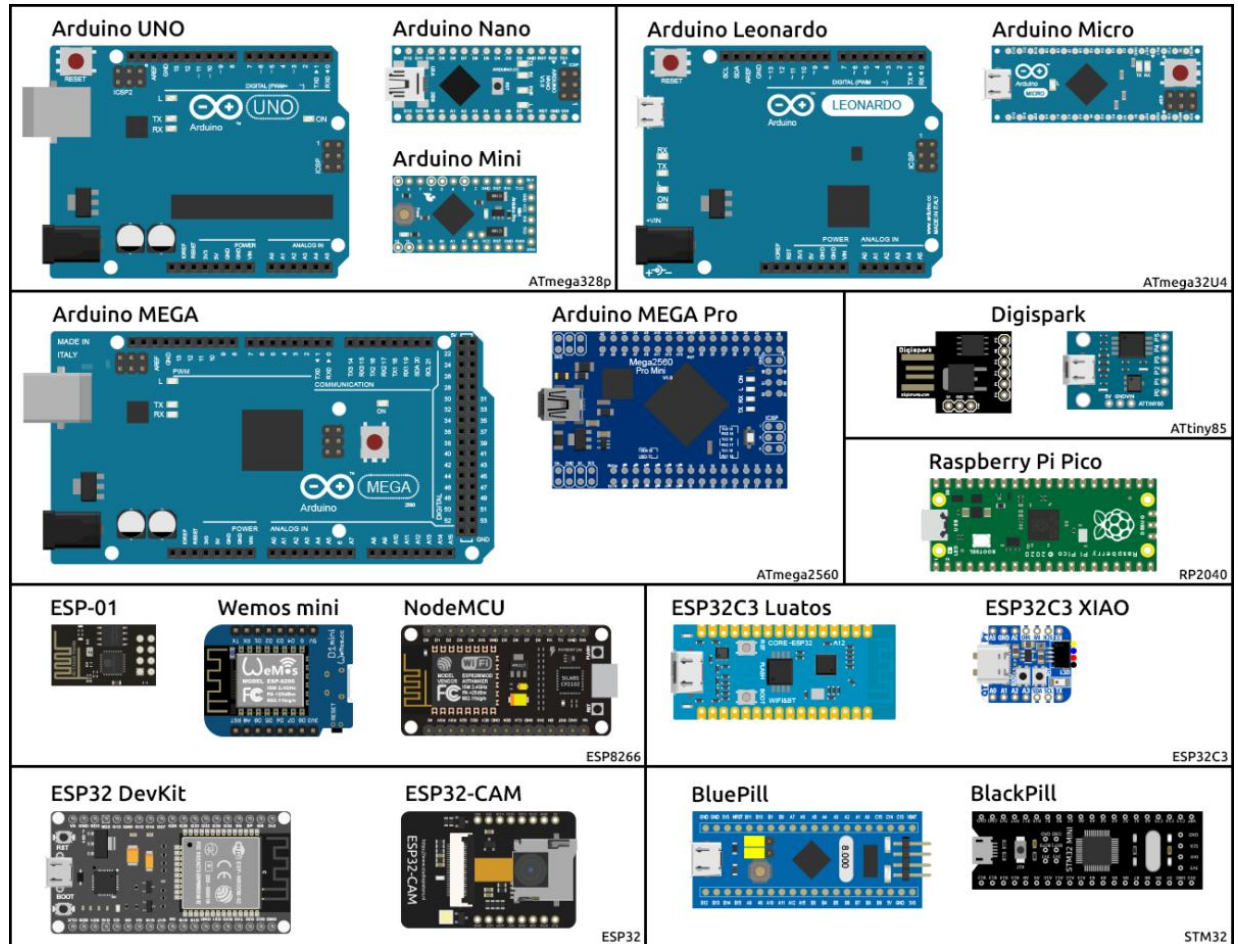


Рис. 2.1. Платформи для розробки електронних проєктів

За результатами порівняльного аналізу наявних на ринку мікроконтролерних модулів для реалізації проєкту "Розумна ферма" на ESP32 в рамках даної роботи було обрано плату розробника ESP32 PLUS (зазвичай це покращена версія класичної NodeMCU на базі модуля ESP-WROOM-32) , зовнішній вигляд якої представлено на рисунку 2.2. та пояснюється її численними перевагами:

- висока продуктивність і багатоядерна архітектура;
- велика кількість цифрових та аналогових входів/виходів;
- підтримка апаратних протоколів, таких як I2C, UART, SPI;
- можливість живлення через USB або VIN;

- широке використання в середовищі Arduino IDE та наявність великої кількості бібліотек;
- вбудовані таймери, що забезпечують точний контроль часу реального часу та формування ШІМ-сигналів.

Плата KEYESTUDIO ESP32 PLUS — універсальна розробка WIFI та Bluetooth плата на основі ESP32, інтегрована з модулем ESP32-WROOM-32 та сумісний з Arduino.

Він оснащений датчиком Холла, високошвидкісним SDIO/SPI, UART, I2S та I2C. Крім того, оснащена операційною системою freeRTOS, яка досить підходить для Інтернету речей і розумного дому.

Завдяки поєднанню потужності, бездротових інтерфейсів та енергоефективності, вона закриває 95% потреб такого проєкту.



Рис. 2.2. Загальний вигляд плати KEYESTUDIO ESP32 PLUS

Таблиця 2.1

Характеристики плати ESP32 PLUS на WROOM-32

Характеристика	Значення	Перевага для “Розумної ферми”
Мікроконтролер	ESP32-WROOM-32	Потужна IoT-платформа
Процесор	Двоядерний Xtensa LX6	Багатозадачність
Тактова частота	До 240 МГц	Швидка обробка даних
Оперативна пам’ять	520 КБ SRAM	Підтримка складних програм
Flash-пам’ять	4 МБ	Зберігання прошивки та вебінтерфейсу

Характеристика	Значення	Перевага для “Розумної ферми”
USB-UART конвертер	CH340	Просте програмування через USB
Напруга живлення	5В	Зручне підключення живлення
Живлення мікромодуля	2.2 ~ 3.6В	Енергоефективність
Максимальний струм стабілізатора	800 мА	Живлення датчиків і модулів
Робочий струм (середній)	~80 мА	Низьке споживання
Робочий струм (піковий)	До 500 мА	Стабільна робота Wi-Fi
Wi-Fi стандарти	FCC/CE/IC/TELEC/KCC/SRRC/NC C	Сумісність з міжнародними стандартами
Wi-Fi протоколи	802.11 b/g/n/d/e/i/k/r	Надійний бездротовий зв’язок
Швидкість Wi-Fi	До 150 Мбіт/с	Швидка передача даних
Частотний діапазон	2.4 ~ 2.5 ГГц	Робота з більшістю роутерів
Підтримка A-MPDU/A-MSDU	Так	Оптимізація передачі пакетів
Bluetooth	v4.2 BR/EDR + BLE	Підключення BLE-датчиків
Радіо приймач	NZIF, чутливість -98 dBm	Стабільний зв’язок
Передавач	Class-1/2/3 AFH	Гнучке налаштування потужності
Аудіо кодеки	CVSD, SBC	Робота з аудіо-модулями
GPIO	До 32	Підключення багатьох сенсорів
ADC	12-бітний	Аналогові датчики
DAC	2 канали	Аналоговий вихід
PWM	LED PWM, Motor PWM	Керування насосами та моторами

Характеристика	Значення	Перевага для “Розумної ферми”
Інтерфейси	SD, UART, SPI, SDIO, I ² C, I ² S, IR	Підключення периферії
Сенсорні входи	Touch Sensor	Сенсорні кнопки
Підсилювач	LNA	Покращення радіозв’язку
Датчики на борту	Hall sensor, температурний датчик	Додатковий моніторинг
Генератори	26 МГц та 32 кГц	Стабільність роботи
Deep Sleep	Підтримується	Автономна робота
Робоча температура	–40°C ~ +85°C	Робота в теплиці та на вулиці
Відстань між пінами	25.5 мм	Сумісність із макетними платами
Розміри плати	27 × 55 мм	Компактність
Режими Wi-Fi	Station / softAP / AP+Station / P2P	Гнучка мережева робота
Захист Wi-Fi	WPA/WPA2/WPA2-Enterprise/WPS	Безпечне підключення
Шифрування	AES / RSA / ECC / SHA	Захист IoT-даних
ОТА оновлення	Підтримується	Віддалене оновлення
Завантаження прошивки	UART / OTA / Host Download	Гнучка розробка
Мережеві протоколи	IPv4, IPv6, TCP, UDP, HTTP, FTP, MQTT, SSL	IoT-інтеграція
Програмування	Arduino IDE, SDK, Cloud Server	Простота розробки
Налаштування	AT-команди, Android/iOS App	Зручне керування
Підтримка хмарних сервісів	Так	Віддалений моніторинг ферми

Центральним елементом апаратної частини є мікроконтролер ESP32 — високопродуктивна платформа з інтегрованими модулями Wi-Fi та Bluetooth, що широко використовується в системах Інтернету речей. ESP32 оснащений двоядерним процесором Tensilica LX6 з тактовою частотою до 240 МГц, що забезпечує достатню обчислювальну потужність для паралельної обробки даних, керування периферійними пристроями та організації мережевих з’єднань. Мікроконтролер має до 34 доступних GPIO-виводів, однак частина з

них зарезервована для внутрішніх функцій, що необхідно враховувати під час проєктування апаратної схеми.

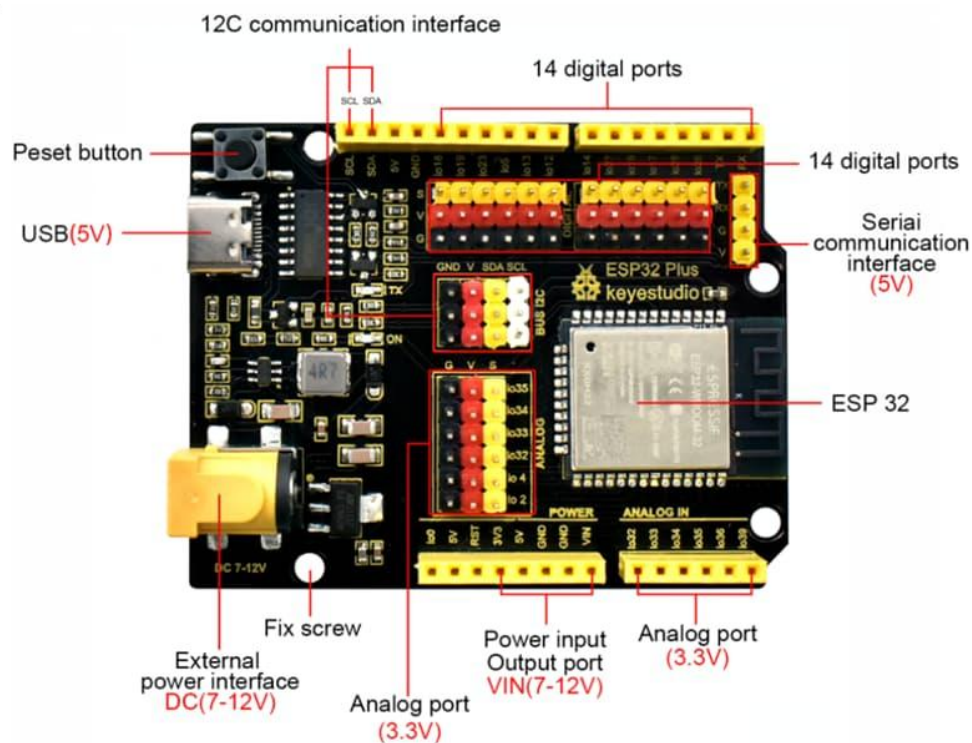
Плата розробника версії PLUS пропонує зручний доступ до виводів (GPIO) мікроконтролера, що дозволяє легко інтегрувати її в макет за допомогою безпаячних макетних плат або індивідуальних друкованих плат.

Для підключення периферійних пристроїв «Розумної ферми» використовуються такі апаратні ресурси ESP32:

- **Аналогові входи (ADC):** використовуються для зчитування показників з аналогових датчиків вологості ґрунту та датчиків рівня води. Завдяки 12-бітній роздільній здатності забезпечується висока точність вимірювання (значення від 0 до 4095).
- **Інтерфейс I2C:** застосовується для підключення цифрових датчиків (наприклад, датчика освітленості BH1750 або дисплея LCD/OLED). Використання шини I2C дозволяє підключати кілька пристроїв до двох ліній зв'язку (SDA та SCL), заощаджуючи вільні виводи.
- **Цифрові GPIO з підтримкою ШІМ (PWM):** необхідні для плавного керування швидкістю вентиляторів витяжки, сервоприводами кватирок теплиці або яскравістю фітоламп системи доосвітлення.

Формування широтно-імпульсних сигналів (PWM) у мікроконтролері ESP32 може бути реалізоване практично на будь-якому GPIO-виводі, що надає значну гнучкість під час керування виконавчими пристроями, такими як світлодіоди, сервоприводи, двигуни постійного струму або інші елементи керування.

Принципова схема плати показана на рисунку 2.5.



KS5016 Keyestudio	
TOP/ Left side header	Bottom/ Right Side Header
RX	io39
TX	io36
IO26	io34
io25	io33
io17	io32
io16	
io27	VIN
io14	GND
	GND
io12	5V
io13	3V3
io5	RST
io23	5V
io19	io0
io18	
GND	
5V	
SDA	
SCL	

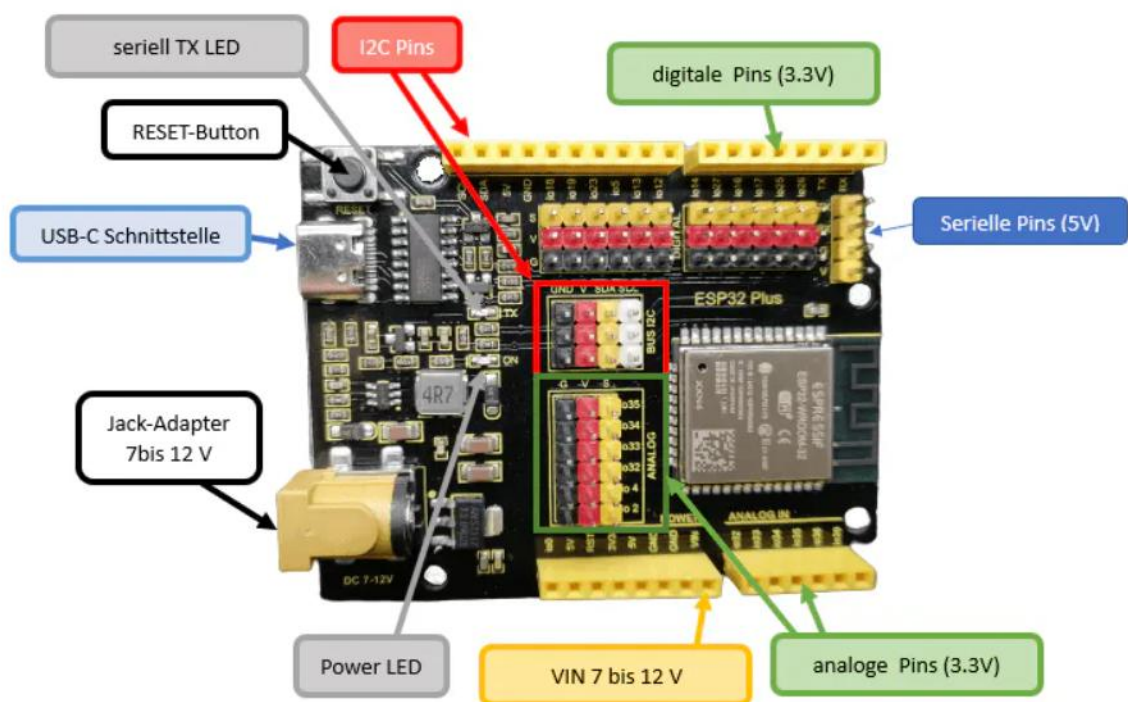


Рис. 2.3. Структура виводів плати KEYESTUDIO ESP32 PLUS

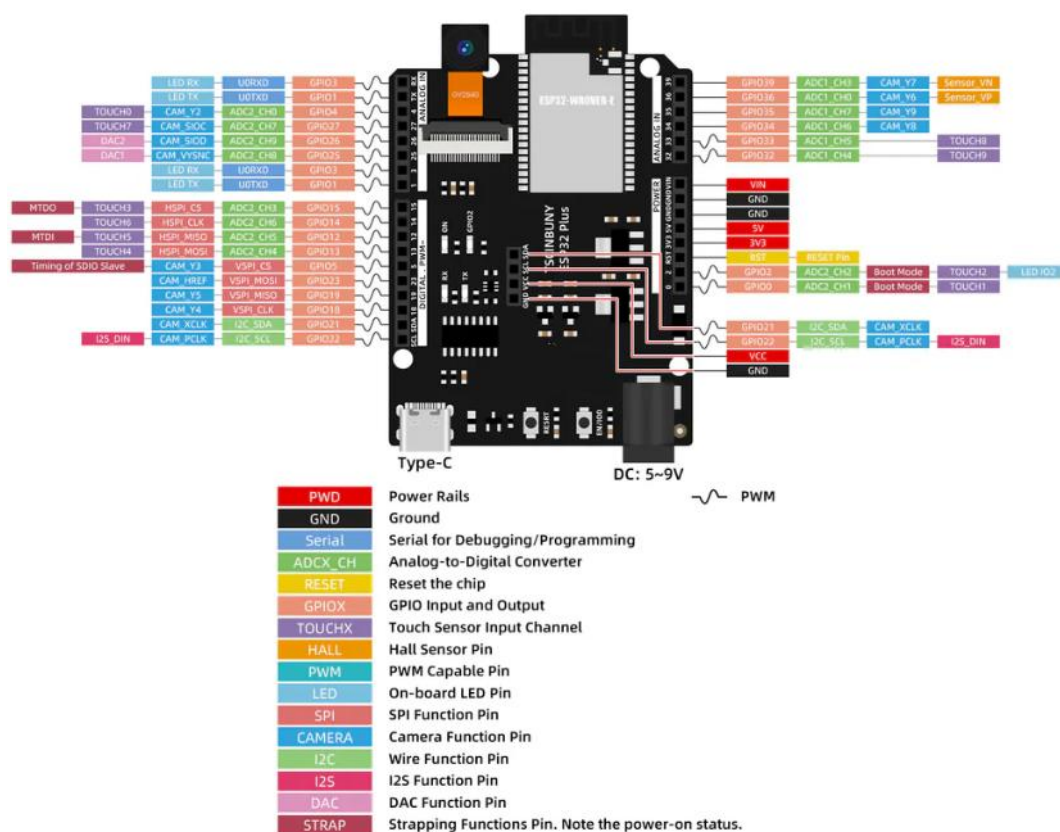


Рис. 2.4. Распиновка платы KEYESTUDIO ESP32 PLUS

Мікроконтролер ESP32 підтримує широкий спектр комунікаційних протоколів, що суттєво розширює можливості інтеграції зовнішніх модулів. Інтерфейс I2C забезпечує підключення кількох пристроїв через дві сигнальні лінії — SDA (GPIO21) та SCL (GPIO22), що дозволяє мінімізувати кількість задіяних виводів.

Для підключення високошвидкісних периферійних пристроїв використовується інтерфейс SPI, який реалізується через виводи GPIO19 (MISO), GPIO23 (MOSI), GPIO18 (SCK) та GPIO5 (CS). Даний інтерфейс застосовується для обміну даними з такими компонентами, як модуль бездротового зв'язку LoRa RA-01 SX1278, SD-карти пам'яті або TFT-дисплеї. Окрім цього, ESP32 оснащений трьома апаратними інтерфейсами UART (UART0, UART1, UART2), що забезпечує можливість організації послідовного зв'язку з GSM-модулями, GPS-приймачами, RFID-зчитувачами та іншими зовнішніми пристроями.

Вбудовані мережеві інтерфейси є критично важливими для концепції Smart Farming:

- **Wi-Fi модуль** працює на частоті 2.4 ГГц і підтримує режими Station (STA), Access Point (AP) та їх комбінацію. У проекті режим STA використовується для підключення макету до локальної роутерної мережі з подальшим виходом в Інтернет для надсилання телеметрії на IoT-платформу (Blynk/ThingsBoard).
- **Bluetooth Low Energy (BLE)** забезпечує можливість енергоефективної локальної конфігурації макету. За допомогою мобільного додатка через BLE користувач може первинно налаштувати параметри Wi-Fi мережі (SSID та пароль) без необхідності перепрошивки плати.

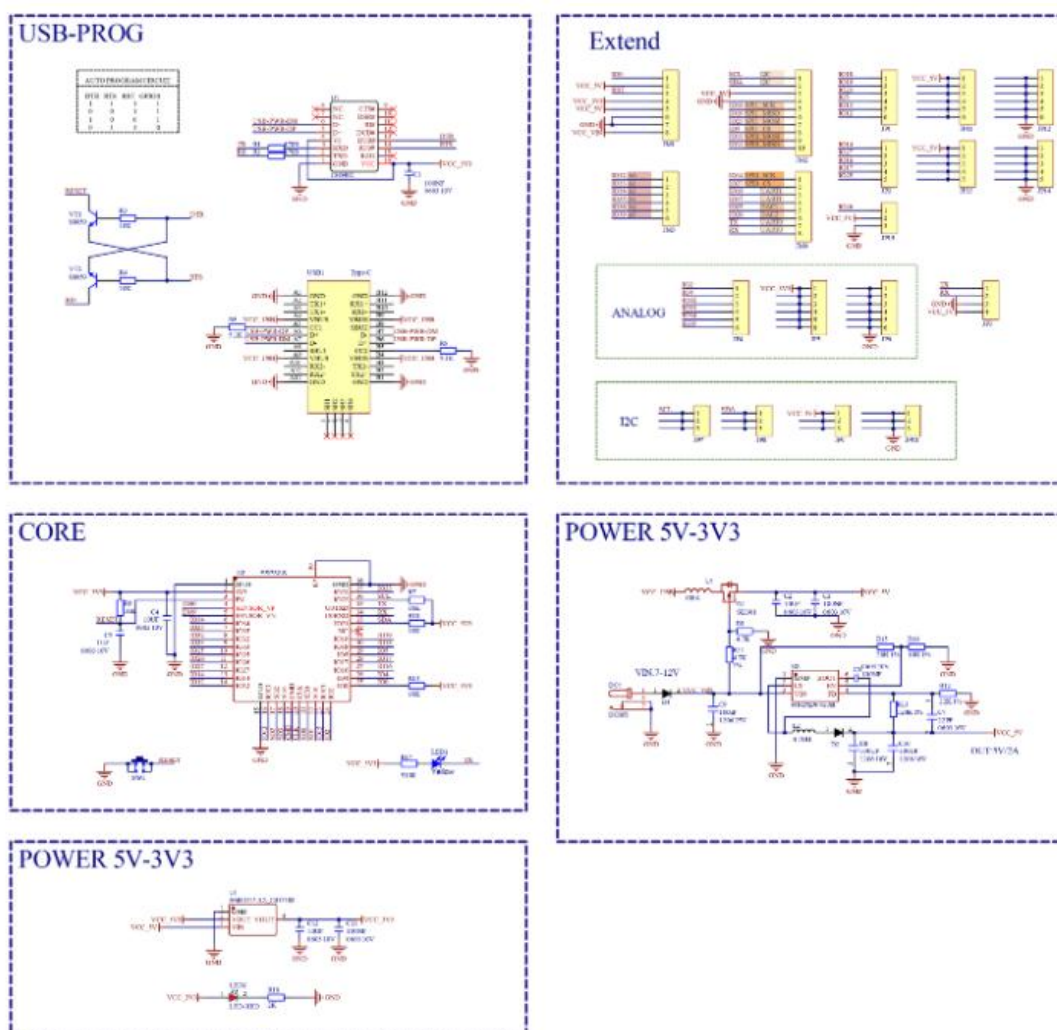


Рис. 2.5. Принципова схема плати KEYESTUDIO ESP32 PLUS

2.1.2. Модуль цифрової кнопки EASY Plug RJ11

Модуль тактильної кнопки EASY Plug Push-button являє собою компактне апаратне рішення, призначене для комутації сигналів постійного струму в системах автоматизації. Цей базовий елемент інтерфейсу користувача сумісний із популярними мікроконтролерними платформами, такими як Arduino та Raspberry Pi. Функціонування модуля базується на зміні логічних рівнів: при замиканні контактів (натисканні) на цифровому виході формується сигнал високого рівня (HIGH), а при розмиканні (відпусканні) — низького рівня (LOW). Завдяки простоті підключення до ліній введення-виведення (GPIO), модуль є оптимальним для первинного тестування портів мікроконтролера та освоєння базових алгоритмів обробки зовнішніх переривань.

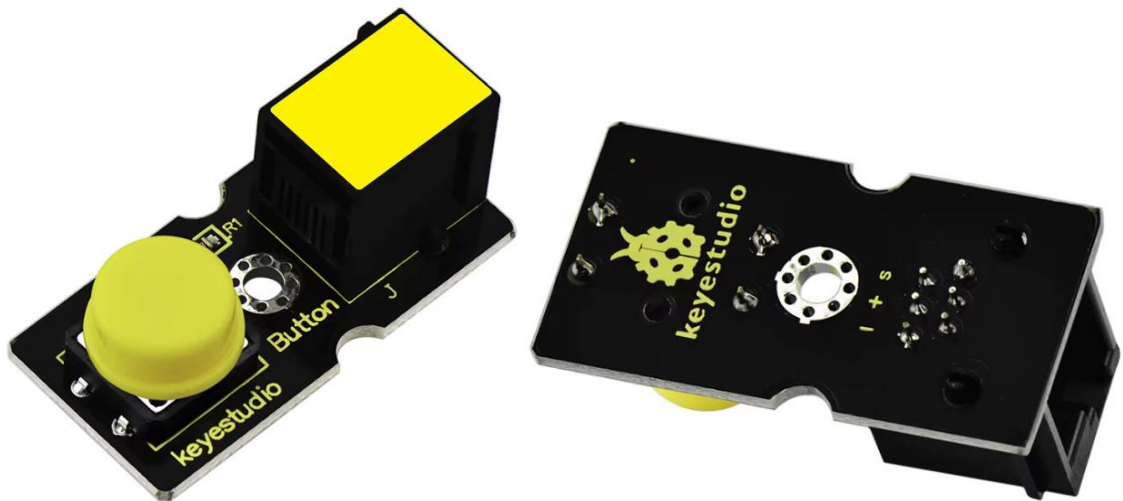


Рис. 2.6. Загальний вигляд модуля цифрової кнопки EASY Plug RJ11

Обраний кнопковий модуль має такі технічні та конструктивні параметри. Пристрій належить до класу цифрових датчиків та формує на виході дискретний сигнал. Діапазон напруги живлення становить від 3.3 до 5 В, що забезпечує повну сумісність як з 5-вольтними логічними рівнями плат Arduino, так і з 3.3-вольтними рівнями мікроконтролера ESP32. Для швидкого та надійного підключення до макету в модулі передбачено інтерфейсний роз'єм RJ11. Конструктивно елемент оснащений ергономічною тактильною кнопкою збільшеної площі із захисним ковпачком підвищеної зносостійкості.

Для роботи треба з'єднати червоний з V, чорний з G, жовтий з S.

Підключення до кнопкового модуля

Модульний Pin	Колір дроту	ESP32 Pin плати
V	ЧЕРВОНИЙ	V
G	ЧОРНИЙ	G
S	ЖОВТИЙ	io5

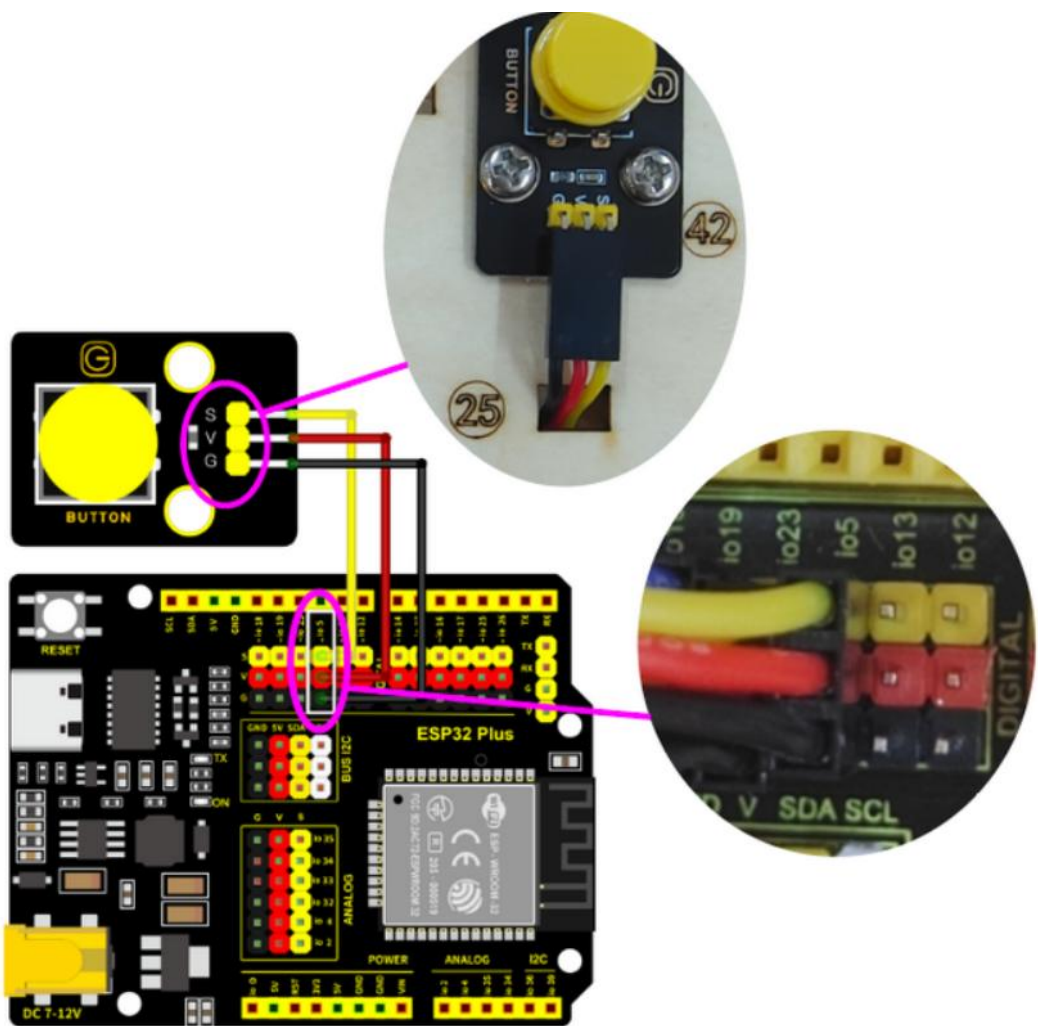


Рис. 2.7. Підключення кнопки EASY Plug RJ11 до плати KEYESTUDIO ESP32 PLUS

2.1.3. Модуль світлодіода Keyestudio LED Module

Модуль світлодіода Keyestudio LED Module призначений для візуальної індикації стану роботи системи. Він містить світлодіод та необхідні елементи для безпечного підключення до мікроконтролера, зокрема плати KEYESTUDIO ESP32 PLUS.

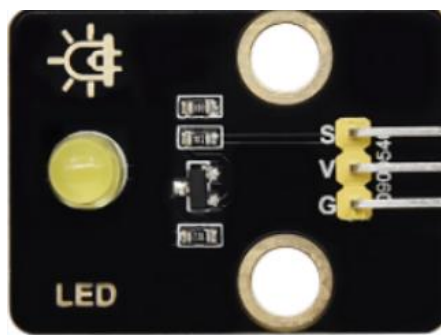


Рис. 2.8. Модуль світлодіода Keyestudio LED Module

Основні характеристики

- Робоча напруга: 3.3–5 В.
- Інтерфейс підключення: S (Signal), V (VCC), G (GND).
- Низьке енергоспоживання.
- Простота інтеграції з платформами ESP32 та Arduino.

У проєкті «Розумна ферма» світлодіодний модуль використовується для світлової сигналізації різних станів системи:

- індикація увімкнення живлення;
- сигналізація про недостатню вологість ґрунту;
- повідомлення про запуск системи поливу;
- індикація аварійних ситуацій або помилок датчиків;
- відображення режиму роботи ферми.

Мікроконтролер ESP32 подає цифровий сигнал на контакт S. При подачі логічної одиниці (HIGH) світлодіод загоряється, а при логічному нулі (LOW) — вимикається. Таким чином користувач може швидко оцінити поточний стан системи без підключення до програмного інтерфейсу.

Таблиця 2.3

Підключення до KEYESTUDIO ESP32 PLUS

Модульний Pin	Колір дроту	ESP32 Pin плати
V	ЧЕРВОНИЙ	V
G	ЧОРНИЙ	G
S	ЖОВТИЙ	io5

Переваги використання

- наочна індикація роботи системи;
- швидке виявлення несправностей;
- простота монтажу та програмування;
- сумісність із навчальними та промисловими IoT-проектами.

У системі «Розумна ферма» модуль світлодіода виконує роль простого та надійного засобу оповіщення користувача про стан датчиків, процес поливу та роботу контролера.

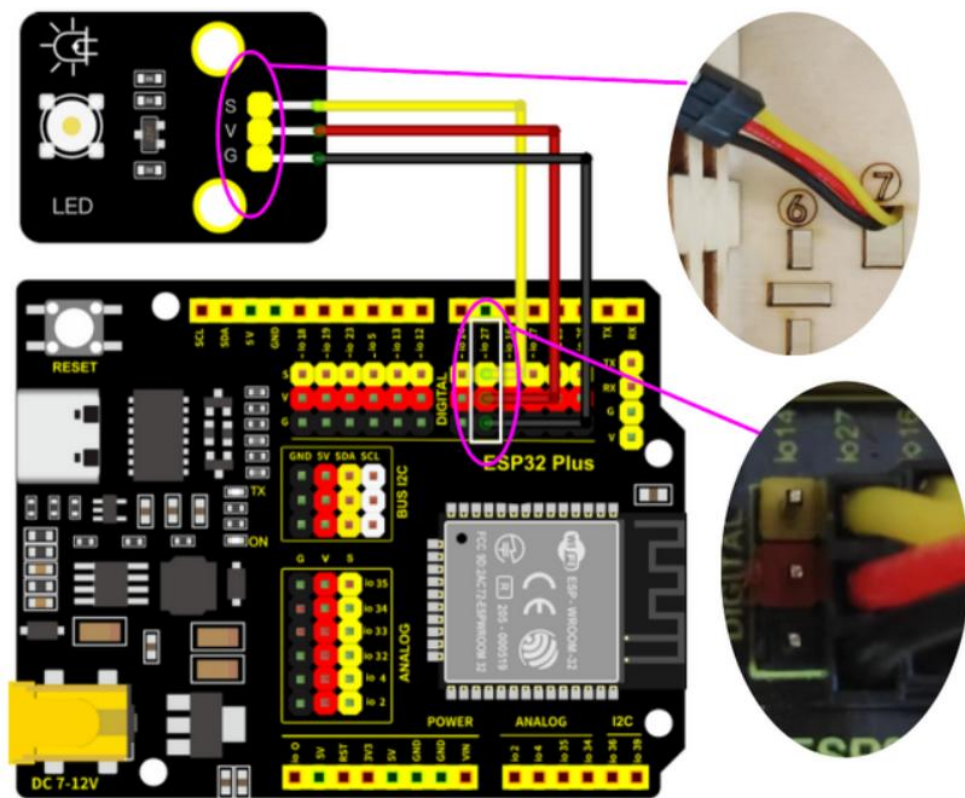


Рис. 2.9. Підключення до KEYESTUDIO ESP32 PLUS

2.1.4. Датчик руху PIR Motion Sensor Module

Датчик руху PIR (Passive Infrared Sensor) є пасивним інфрачервоним сенсором, який реагує на зміну теплового випромінювання об'єктів. Коли людина або тварина потрапляє в зону дії датчика, він формує цифровий сигнал, який надходить на мікроконтролер ESP32.

У системі «Розумна ферма» датчик використовується для автоматичного виявлення присутності людей, тварин або сторонніх об'єктів на контрольованій території.

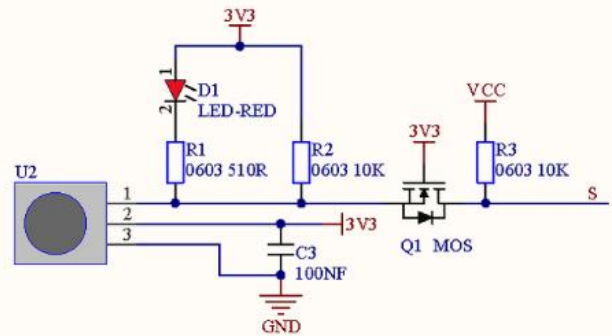
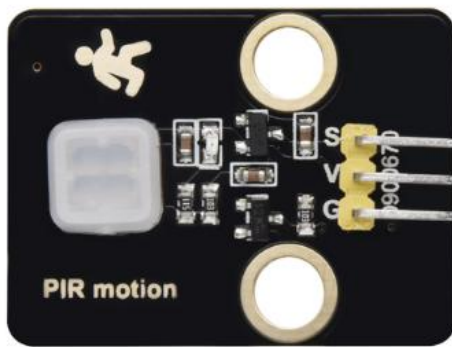


Рис. 2.10. Датчик руху PIR Motion Sensor Module

Основні характеристики

- Тип датчика: пасивний інфрачервоний (PIR).
- Робоча напруга: 3.3–5 В.
- Цифровий вихід.
- Низьке енергоспоживання.
- Простота підключення до ESP32.
- Дальність виявлення: до 3–5 м (залежно від моделі).
- Кут огляду: приблизно 100–120°.

Датчик постійно контролює рівень інфрачервоного випромінювання в зоні спостереження. За відсутності руху на виході **S** підтримується низький логічний рівень (LOW). Після виявлення руху датчик формує високий рівень (HIGH), який зчитується мікроконтролером ESP32.

Використання в системі «Розумна ферма»

У проєкті Smart Farming датчик PIR може виконувати такі функції:

- контроль доступу до теплиці;
- виявлення появи тварин або птахів;
- автоматичне ввімкнення освітлення;
- активація системи відеоспостереження;
- надсилання повідомлень користувачу через Wi-Fi;
- охорона сільськогосподарських об'єктів.

Підключення PIR Motion Sensor Module до KEYESTUDIO ESP32 PLUS

Модульний Pin	Колір дроту	ESP32 Pin плати
V	ЧЕРВОНИЙ	V
G	ЧОРНИЙ	G
S	ЖОВТИЙ	io5

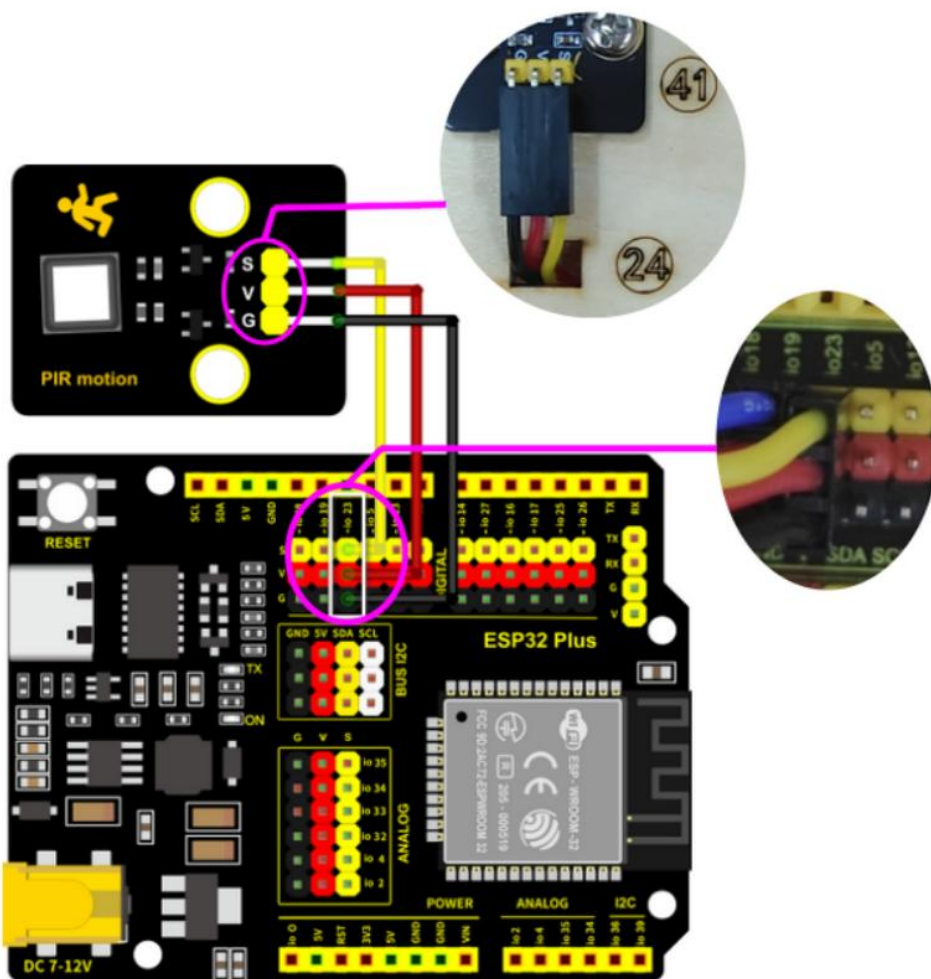


Рис. 2.11. Підключення PIR Motion Sensor Module до KEYESTUDIO ESP32 PLUS

2.1.5. Пасивний дзвінок (Passive Buzzer Module)

Пасивний дзвінок (Passive Buzzer Module) — це звуковий модуль, який використовується для генерації звукових сигналів різної частоти. На відміну від активного дзвінка, пасивний не має вбудованого генератора, тому

для створення звуку необхідно подавати на нього імпульсний сигнал певної частоти від мікроконтролера.

Основні характеристики

- Тип: пасивний п'єзоелектричний дзвінок.
- Робоча напруга: 3.3–5 В.
- Можливість відтворення різних тонів і мелодій.
- Низьке енергоспоживання.
- Сумісність з ESP32 та Arduino.

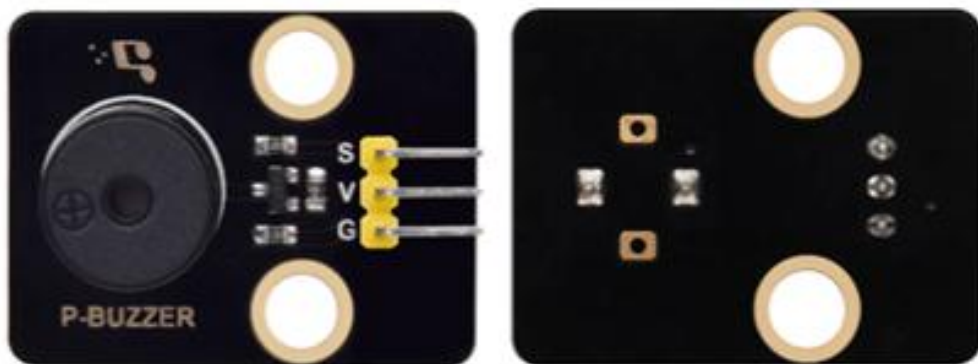


Рис. 2.12. Passive Buzzer Module

Пасивний дзвінок не може вібрувати, випромінюючи сам звук, якщо не подається сигнал квадратної хвилі з певною частотою. Крім того, випромінюючий звук змінюється залежно від частоти квадратної хвилі, тому пасивний дзвінок може імітувати мелодії.

Аналогову хвилю сквайра можна генерувати, змінюючи рівень потужності на контактах. Наприклад, після того, як високий рівень, що триває 500 мс, він змінюється на низький ще на 500 мс, а потім знову на високий.

Ми приводимо сигнал через хвилю сквайра в межах 200~5000 Гц і можемо обчислити частоту(f): $f=1/T$, T — це період (загальний час високого та низького рівнів).

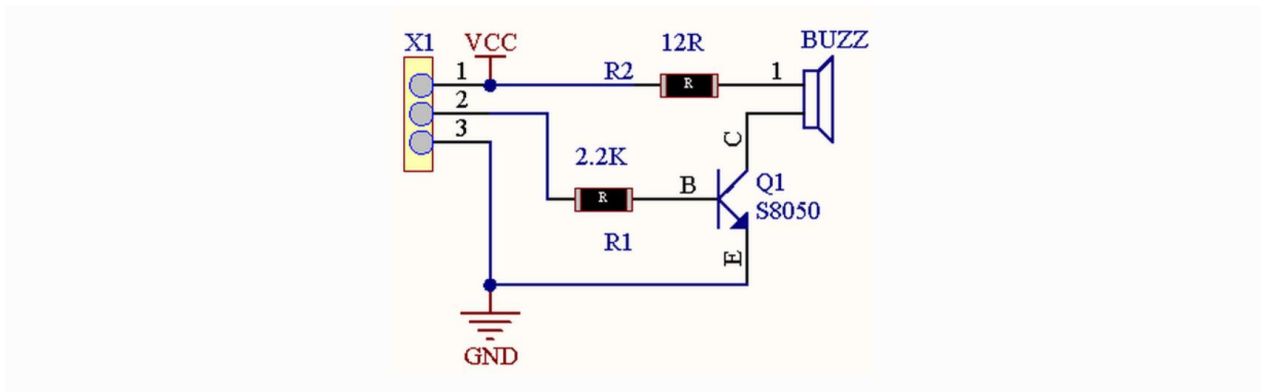


Рис. 2.13. Схема Passive Buzzer Module

Таблиця 2.5

Підключення Passive Buzzer Module до KEYESTUDIO ESP32 PLUS

Модульний Pin	Колір дроту	ESP32 Pin плати
V	ЧЕРВОНИЙ	V
G	ЧОРНИЙ	G
S	ЖОВТИЙ	io5

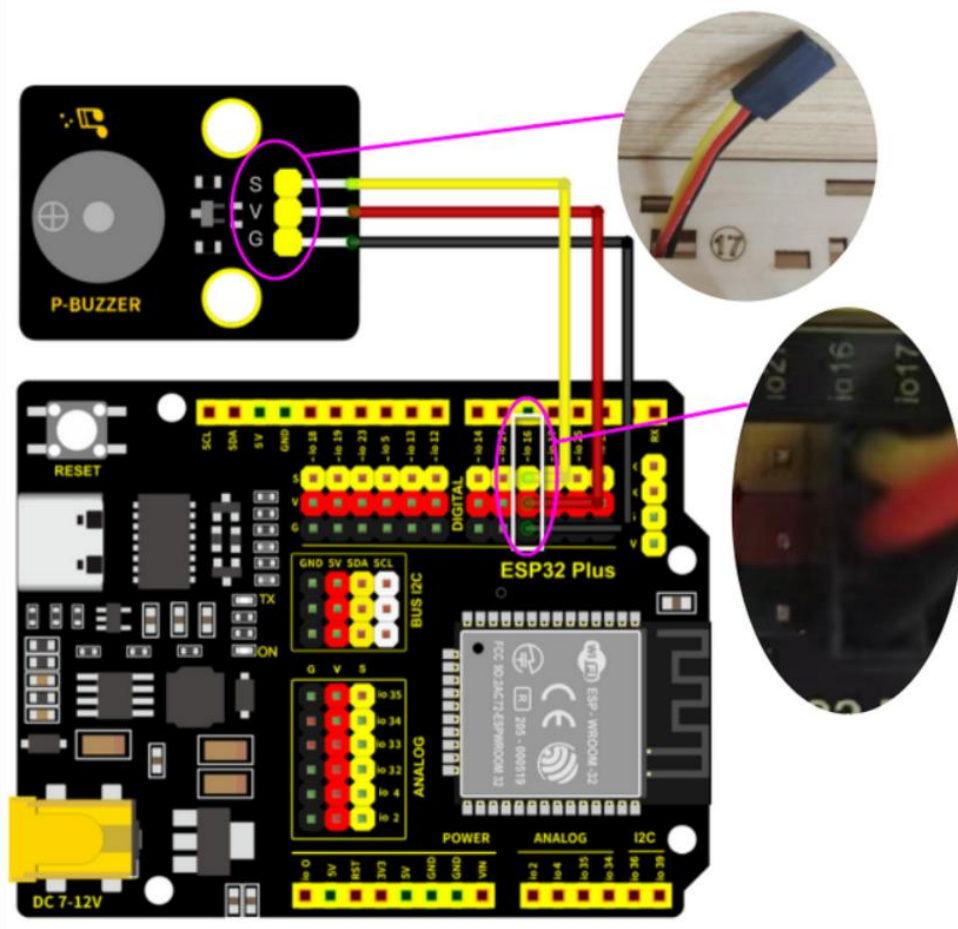


Рис. 2.14. Підключення Passive Buzzer Module до KEYESTUDIO ESP32 PLUS

2.1.6. Серводвигун SG90

Для забезпечення рухомості двері було обрано сервопривід SG-90 (рис. 2.15), який відповідає вимогам за габаритами та вантажопідйомністю, а також відрізняється доступною вартістю.

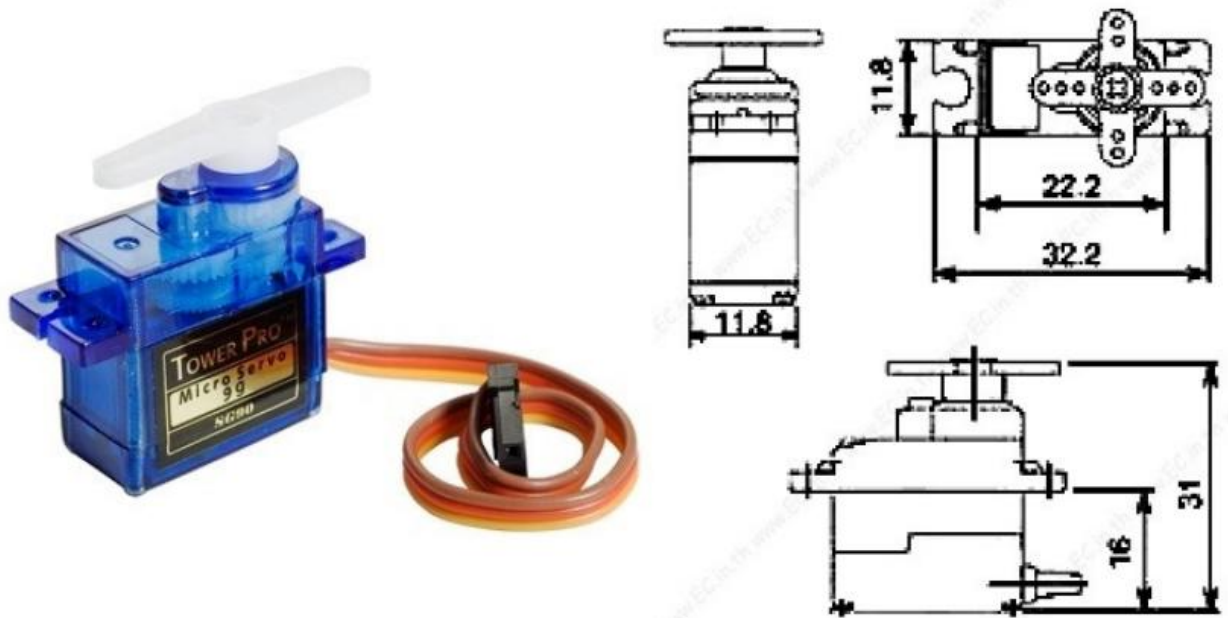


Рис. 2.15. Мікро сервопривід Tower Pro SG90

Сервоприводи є електромеханічними пристроями, що поєднують у собі датчик для вимірювання заданих параметрів і блок керування, який автоматично підтримує ці параметри в установлених межах. Перетворення електричної енергії на механічний рух здійснюється за допомогою електродвигуна.

Приводний механізм сервопривода складається з електродвигуна та редуктора. Редуктор призначений для зменшення швидкості обертання двигуна, оскільки його початкова частота обертів зазвичай є надто високою для практичного застосування. Він містить корпус із системою валів і шестерень, які забезпечують передачу та перетворення крутного моменту.

Окрім електродвигуна, редуктора та потенціометра, сервопривід оснащений електронною платою керування. Вона обробляє сигнал, що надходить від потенціометра, порівнює його із заданим значенням і на основі результатів цього порівняння регулює роботу електродвигуна, здійснюючи його увімкнення або вимкнення.

Всередині сервопривода Tower Pro SG90 міститься електронний модуль керування, який регулює роботу електродвигуна на основі вхідного сигналу. Цей сигнал містить інформацію про цільову позицію валу. Щоб контролювати його поточний стан, редуктор підключений до вбудованого змінного резистора (потенціометра). Електроніка пристрою постійно порівнює фактичне положення редуктора з бажаним і, орієнтуючись на опір резистора, подає на двигун напругу потрібної полярності, щоб повернути вал у задану точку.

Команда про необхідний кут повороту кодується тривалістю імпульсів у керуючому сигналі (ШІМ). Цей сигнал повинен мати стабільну частоту 50 Гц. Хоча ключовим параметром є шпаровість (співвідношення тривалості імпульсу до його періоду), на практиці найчастіше оцінюють саме довжину самого імпульсу. Для генерації такого керуючого сигналу зазвичай застосовують мікроконтролери з підтримкою широтно-імпульсної модуляції.

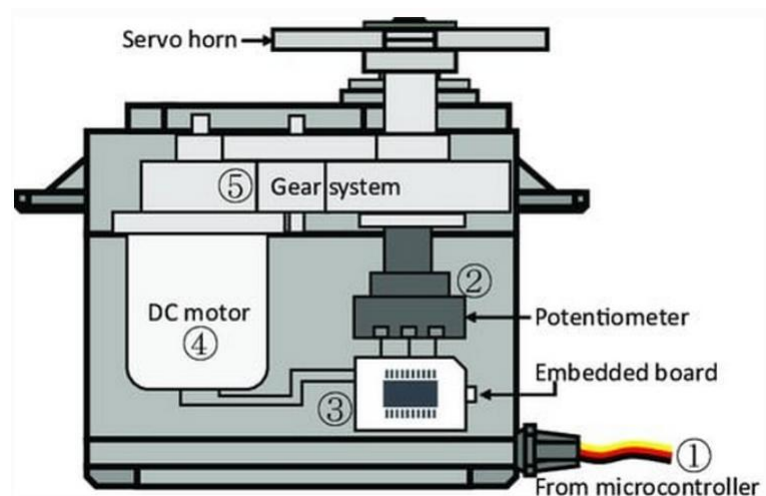


Рис. 2.16. Внутрішня структура сервопривіда Tower Pro SG90

(1) Сигнал(и): Приймає керуючий сигнал від мікроконтролера.

(2) Потенціометр: зворотний зв'язок у Servo. Він вимірює положення вихідного валу.

(3) Вбудована плата (внутрішній контролер): ядро Servo. Він обробляє зовнішній керуючий сигнал і зворотний зв'язок положення і керує Servo.

(4) Двигун постійного струму: частина виконання. Він показує швидкість, крутний момент і положення.

(5) Система передач: масштабує вихідні потужності від двигуна до кінцевого виходу, кутом узгоджується з певним передавальним числом.

Дана модель сервоприводу є якісним і компактним мікросервоприводом, що відрізняється доступною вартістю та малою масою. Вага редуктора може бути віднесена до ультралегкої, оскільки становить лише 9 г.

Таблиця 2.6

Технічні характеристики мікро сервоприводу Tower Pro SG90

Характеристика	Показник
Вага, г	14,7
Частота, мкс	1520
Тип двигуна	Колекторний
Габаритні розміри, мм	23 x 12 x 28,5
Матеріал корпусу та редуктора	Пластик
Швидкість, сек/60°	0,1 – 0,12
Живлення, В	4,8 – 6
Зусилля, кг/см	1,2 – 2,5

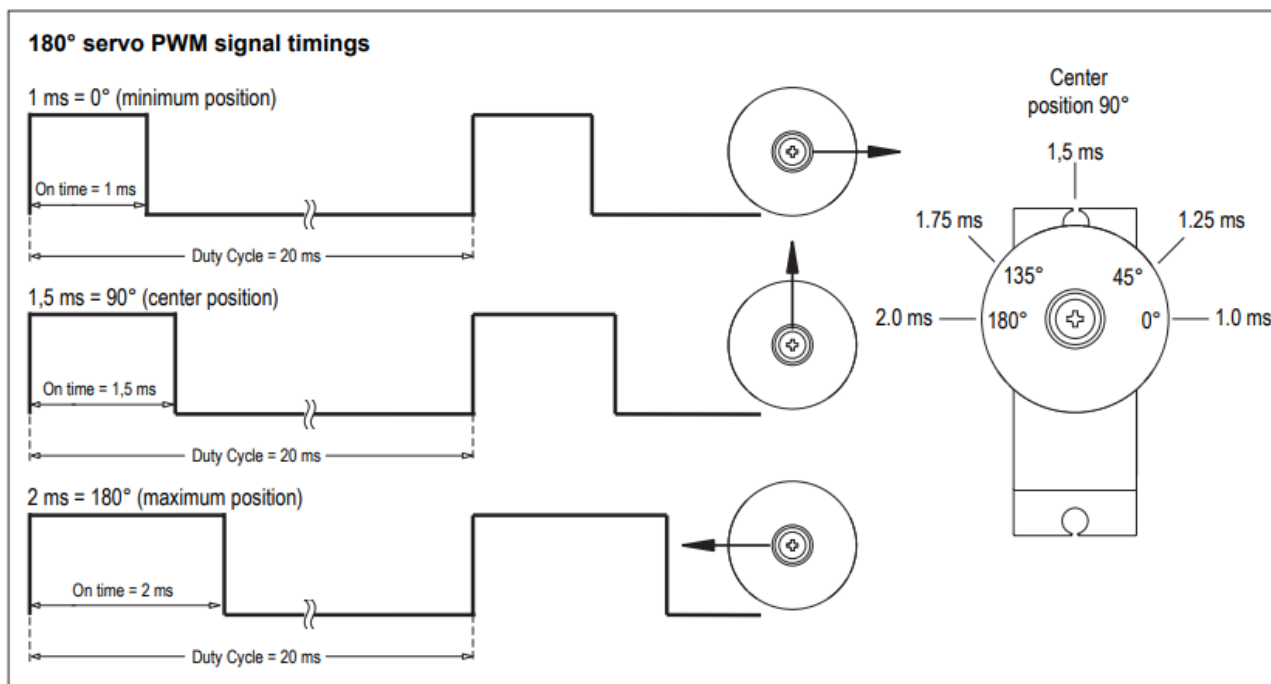


Рис. 2.17. Схема сервоприводу Tower Pro SG90

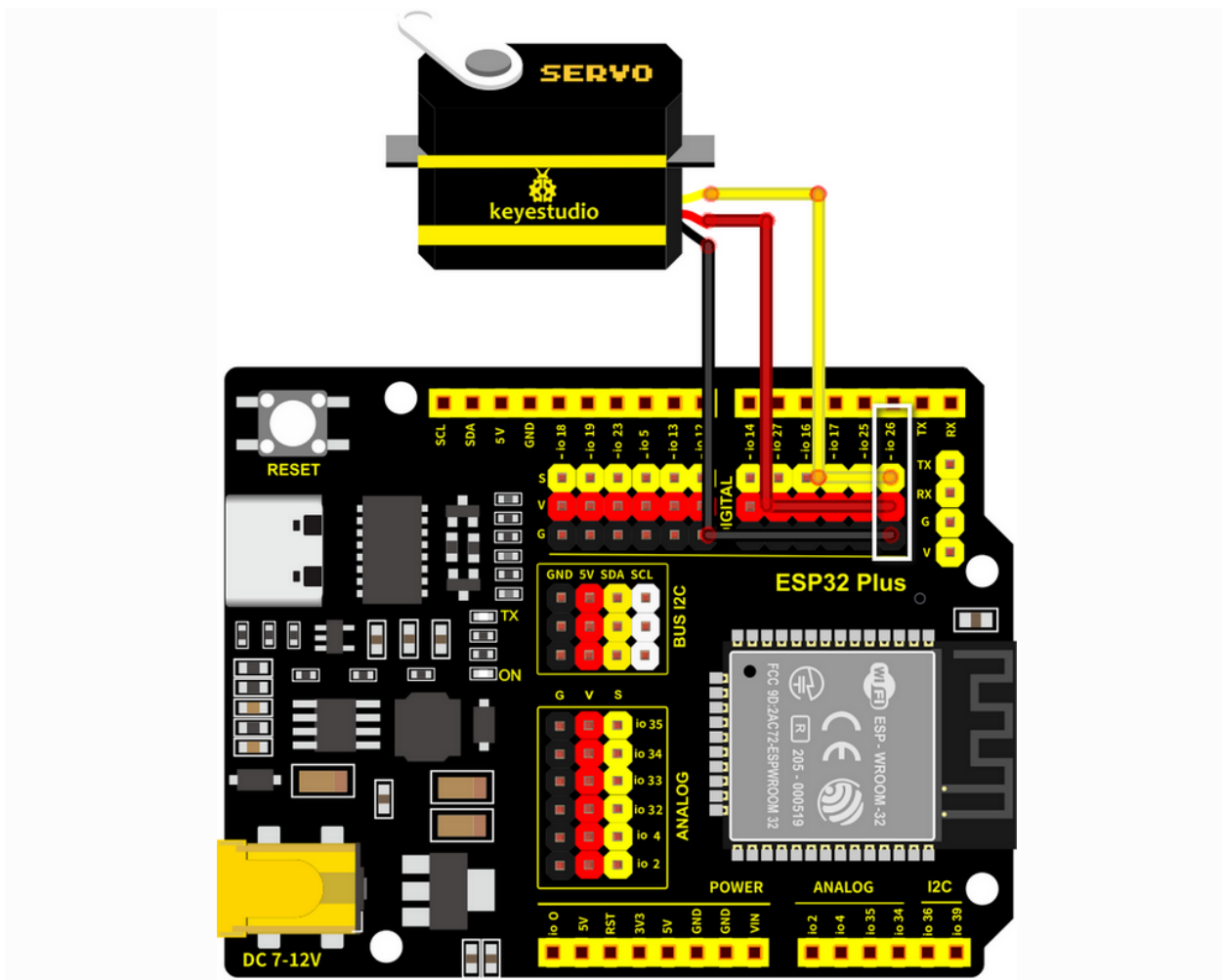


Рис. 2.18. Схема підключення серво до KEYESTUDIO ESP32 PLUS

2.1.7. Ультразвуковий датчик Ultrasonic Sensor HC-SR04

Одним із найбільш розповсюджених і надійних ультразвукових далекомірів, який часто застосовується в системах супроводу людей з вадами зору, є HC-SR04 Ultrasonic Sensor (рисунок 2.19). Цей модуль вирізняється високою популярністю та широкою доступністю на ринку, забезпечуючи точні вимірювання відстані за допомогою ультразвукових хвиль.



Рис. 2.19. Ultrasonic Sensor HC-SR04 [12]

Датчик HC-SR04 функціонує за принципом випромінювання ультразвукових хвиль і подальшого вимірювання часу, за який відбитий сигнал повертається після відбиття від об'єкта. Аналізуючи часову затримку між передачею та прийомом сигналу, модуль визначає відстань до перешкоди.

Алгоритм роботи датчика:

1. **Формування тригерного сигналу.** На вхід **Trig** подається керуючий імпульс високого рівня тривалістю близько 10 мкс.
2. **Генерація ультразвуку.** Після отримання тригерного імпульсу датчик випромінює пакет з 8 ультразвукових коливань частотою 40 кГц.
3. **Приймання відбитого сигналу.** У разі наявності об'єкта на шляху поширення хвилі ультразвук відбивається та фіксується приймальним елементом.
4. **Вимірювання часової затримки.** Контакт **Echo** утримується у високому логічному рівні протягом часу, необхідного для проходження ультразвукової хвилі до об'єкта та назад.
5. **Обчислення відстані.** Відстань розраховується за формулою:

$$Distance = \frac{Time \times Speed\ of\ Sound}{2}$$

де швидкість звуку в повітрі становить приблизно **343 м/с**.

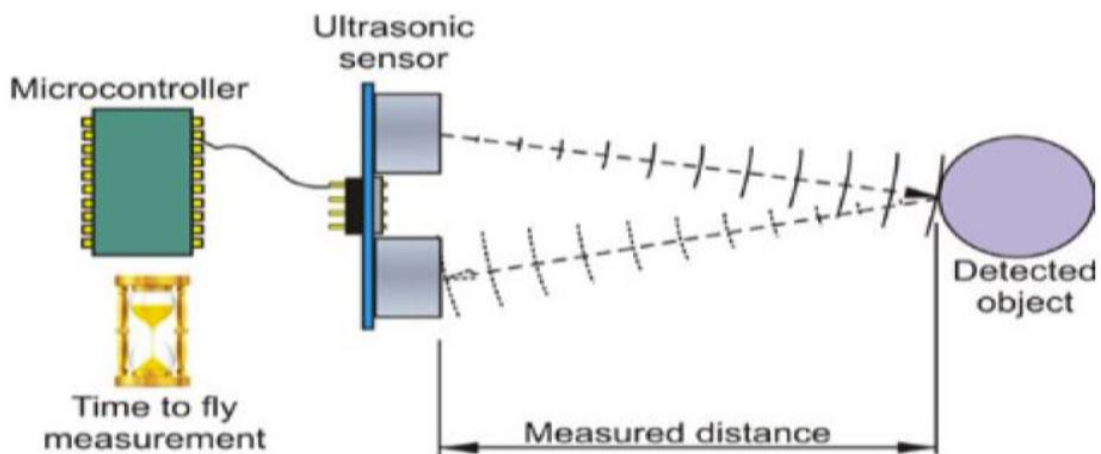


Рис. 2.20. HC-SR04 працює: вид зверху

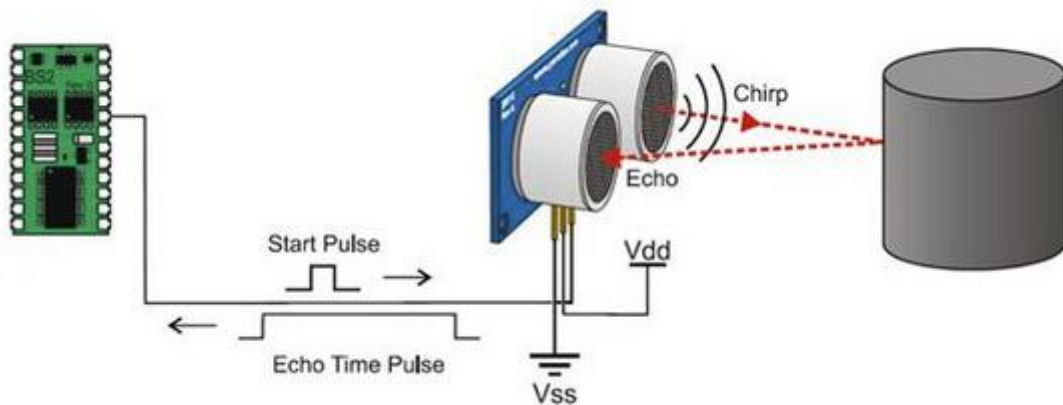


Рис. 2.21. HC-SR04 принцип роботи

Саме цей ультразвуковий далекомір було обрано завдяки його здатності виконувати вимірювання відстані до об'єктів у діапазоні від 2 до 400 см з високою роздільною здатністю та точністю до міліметра. Модуль HC-SR04 відзначається простотою підключення та експлуатації, оскільки має лише чотири основні виводи:

- **VCC (Voltage Common Collector)** – контакт живлення датчика, який підключається до джерела постійної напруги для забезпечення його роботи;
- **GND (Ground)** – заземлювальний вивід, що слугує опорним нульовим потенціалом та забезпечує замикання електричного кола;
- **Trig (Trigger)** – вхідний контакт запуску, на який подається короткий імпульс для ініціювання випромінювання ультразвукової хвилі;
- **Echo** – вихідний контакт, що формує сигнал, тривалість якого відповідає часу проходження ультразвукової хвилі до об'єкта та назад, на основі чого визначається відстань.

Ультразвуковий датчик HC-SR04 тривалий час представлений на ринку та широко доступний у магазинах електроніки й на онлайн-платформах. Його невисока вартість у поєднанні з надійністю та достатньою точністю вимірювань робить цей модуль доцільним вибором для реалізації дипломного проєкту.

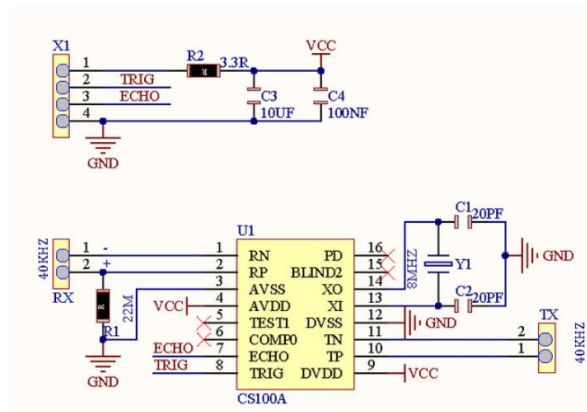


Рис. 2.22. Схема Ultrasonic Sensor HC-SR04

Таблиця 2.6

Підключення Ultrasonic Sensor HC-SR04 до KEYESTUDIO ESP32 PLUS

Модульний Pin	Колір дроту	ESP32 Pin плати
V	ЧЕРВОНИЙ	V (io12)
G	ЧОРНИЙ	G (io12)
ECHO	ЗЕЛЕНИЙ	io13
TRIG	СИНІЙ	io12

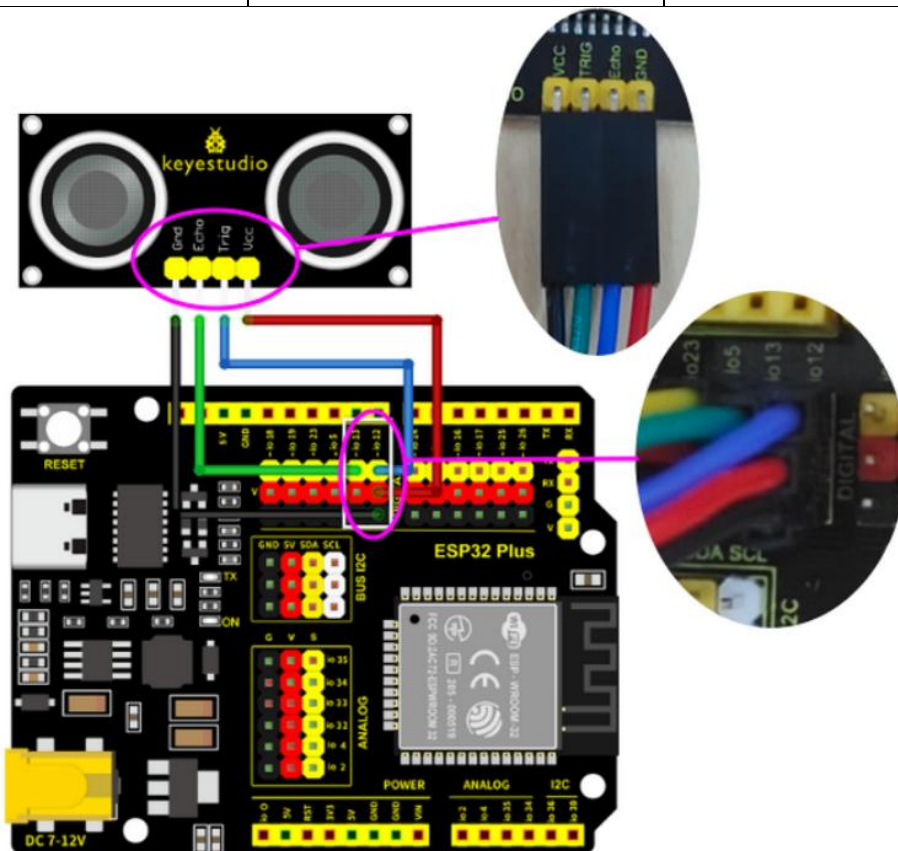


Рис. 2.23. Схема підключення Ultrasonic Sensor HC-SR04 до KEYESTUDIO ESP32 PLUS

2.1.8. Реле електромеханічне SRD-5VDC-SL-C

Ми використовуємо плату ESP32 для вмикання/вимикання водяного насоса за допомогою релейного модуля. Насос піднімає воду та транспортує рідини, зазвичай у використанні поєднується з релейним модулем.

У використанні його часто застосовують для управління високою напругою та струмом навантаження, наприклад, двигунами, датчиками високого струму та високопотужним освітленням.

Характеристики:

- максимальне навантаження: 10A 250V змінного струму
- номінальна напруга на котушці: 5V постійного струму
- кількість контактів перемикавання: 2 (NO і NC)
- температура навколишнього середовища: -25 ... + 70 град. Цельсія
- опір ізоляції: $\geq 100 \text{ МОм}$
- розміри: 19x15x15 мм.



Рис. 2.24. Реле електромеханічне SRD-5VDC-SL-C

Зазвичай відкриті (NO): Цей контакт зазвичай відкритий, якщо сигнал не приймається сигнальним контактом реле. Тому звичайні контакти від'єднуються через NC-контакт і підключені через NO-контакт.

Спільний контакт (COM): Цей контакт підключається до інших модулів, наприклад, водяного насоса.

Зазвичай закриті (NC): Контакт NC-контакту з'єднаний із COM-контактом, утворюючи замкнене коло. Він використовує плату ESP32 для керування закриттям і відключенням релейного модуля.

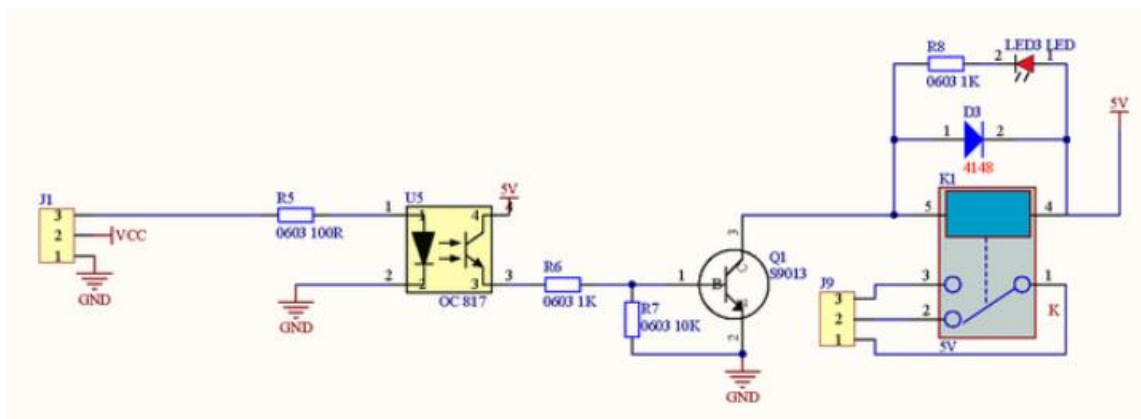


Рис. 2.25. Схема SRD-5VDC-SL-C

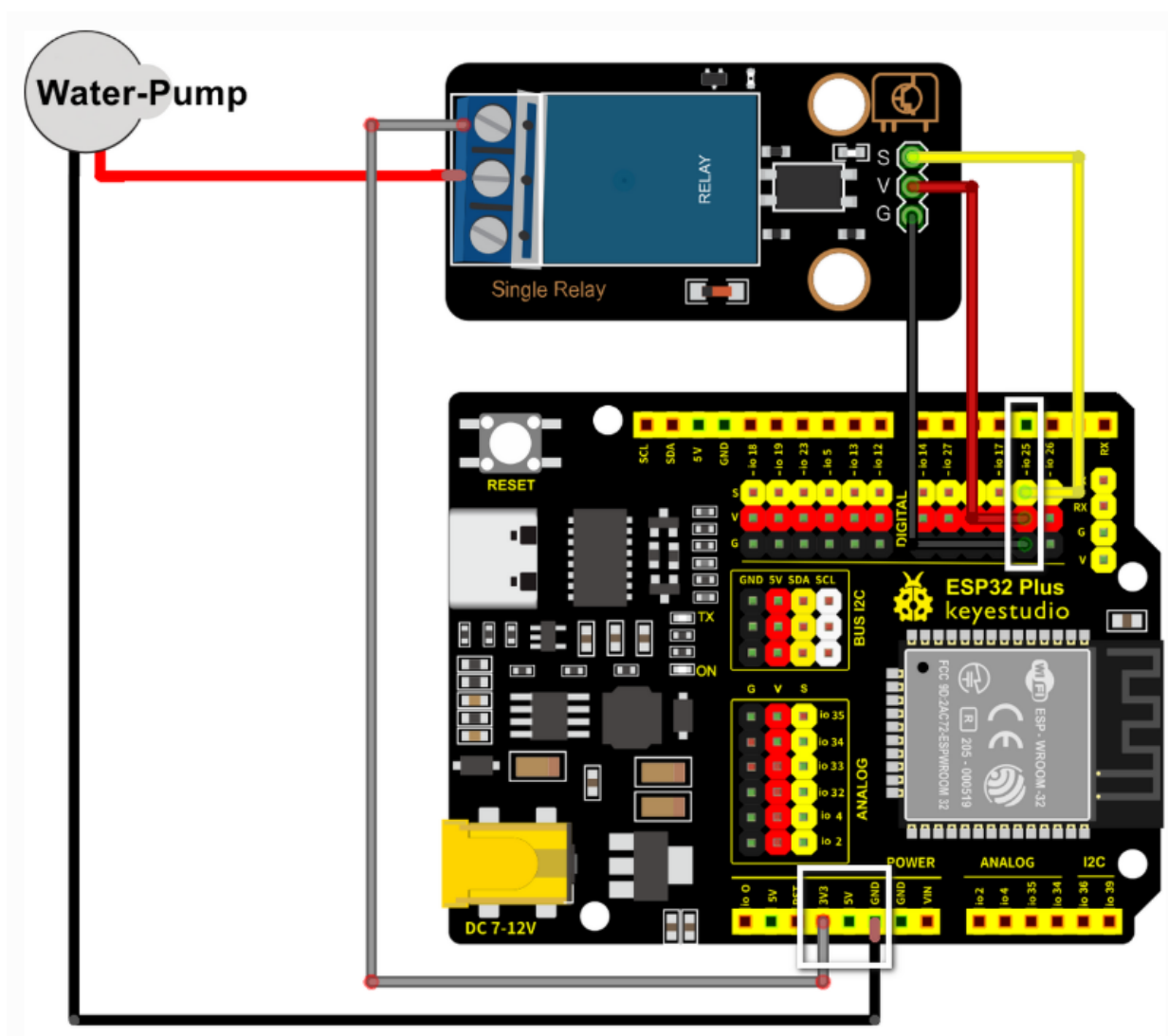


Рис. 2.26. Схема підключення SRD-5VDC-SL-C до KEYESTUDIO ESP32 PLUS

Червоний дріт водяного насоса підключений до середньої клеми реле, а чорний дріт підключений до GND Плата ESP32.

Крім того, для підключення лівої клеми потрібно використовувати дріт Dupont ретрансляційного модуля до 3,3V ESP32.

Таблиця 2.7

Підключення SRD-5VDC-SL-C до KEYESTUDIO ESP32 PLUS

Модульний Pin	Колір дроту	ESP32 Pin плати
V	ЧЕРВОНИЙ	V
G	ЧОРНИЙ	G
S	ЖОВТИЙ	Io25

2.1.9. Дисплей LCD 1602

Рідкокристалічні дисплеї LCD 1602 є поширеним вибором для виведення текстової інформації у багатьох електронних проєктах завдяки своїй доступності та широкому спектру модифікацій [27]. Класична схема підключення LCD 1602 вимагає використання мінімум 6 цифрових виводів мікроконтролера, що може бути значним обмеженням для проєктів з обмеженою кількістю доступних портів. Це пов'язано з тим, що для керування кожним сегментом символів на дисплеї потрібен окремий сигнал керування.

Для подолання цього обмеження часто використовують LCD модулі з інтерфейсом I2C. Цей інтерфейс дозволяє підключити дисплей до мікроконтролера лише за допомогою двох проводів (SCL і SDA), суттєво зменшуючи кількість зайнятих портів. Це робить LCD дисплеї з інтерфейсом I2C більш універсальними та зручними у використанні з платформами ESP 32.

Характеристики:

- Роз'єм: RJ11
- Сумісність із бібліотекою Arduino LiquidCrystal
- Білий текст на блакитному тлі
- Ширина 16 символів, 2 рядки
- Адреса I2C: 0x27
- Підсвічування: синє
- Колір тексту: Білий

- Напруга живлення: 5 В
- Регулювання контрастності: потенціометр
- Розміри: 99мм * 37мм * 21мм
- Вага: 37.4 г

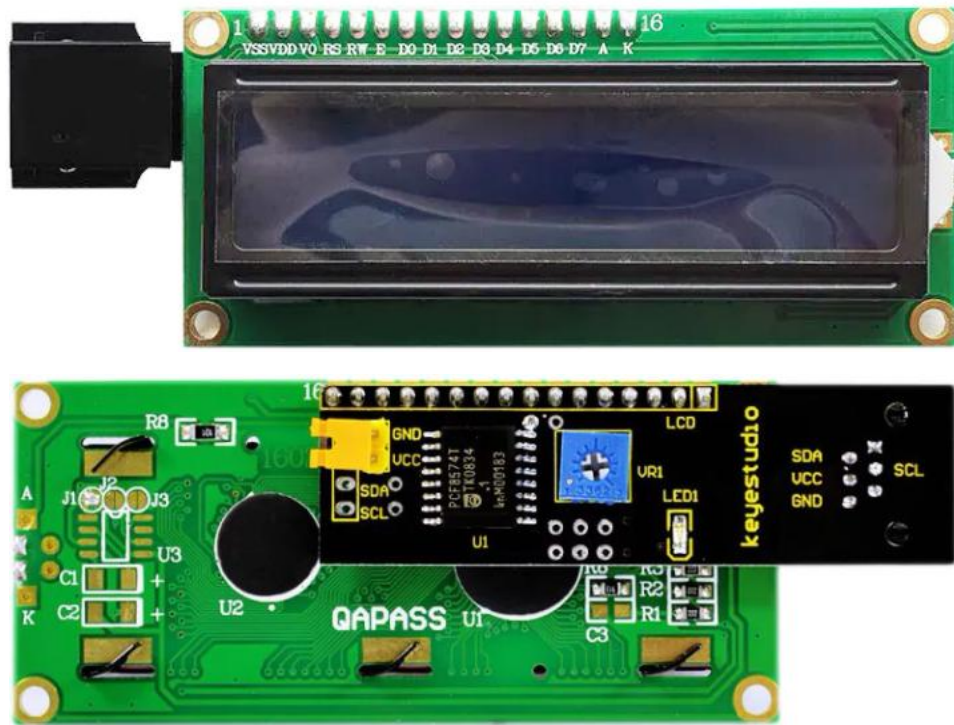


Рис. 2.27. Дисплей LCD 1602

Кожен з виводів рідкокристалічного дисплея відіграє критичну роль у його функціонуванні:

- **GND (земля):** слугує точкою відліку для електричних потенціалів у схемі.
- **5 В (живлення):** забезпечує необхідну напругу для роботи електроніки дисплея.
- **Контраст:** дозволяє регулювати яскравість зображення шляхом зміни напруги на цьому виводі.
- **RS (Register Select):** визначає, чи буде наступна передана інформація інтерпретована як команда (наприклад, встановити курсор) чи як дані для відображення на екрані.
- **RW (Read/Write):** використовувався в старих моделях дисплеїв для читання даних з дисплея. В сучасних моделях зазвичай заземлений.

- **Enable (E):** Сигнал, який інформує дисплей про те, що дані на лініях даних готові для прийому.
- **DB0-DB7 (Data Bits):** група з 8 ліній, за якими передаються біти даних, що визначають, які сегменти рідких кристалів мають бути увімкнені для формування зображення.
- **A (анод) та K (катод):** виводи для підключення зовнішнього джерела живлення для підсвічування дисплея.

Рідкокристалічний дисплей (LCD) має наступні технічні характеристики:

- Тип відображення: символів, з можливістю завантаження користувацьких символів.
- Підсвічування: світлодіодне, що забезпечує яскраве та рівномірне підсвічування символів.
- Контролер: HD44780, що відповідає за керування роботою дисплея.
- Живлення: 5 вольт постійного струму.
- Розмір дисплея: 16 символів по ширині та 2 рядки по висоті.
- Робоча температура: від -20°C до +70°C.
- Температура зберігання: від -30°C до +80°C.
- Кут огляду: 180 градусів, що забезпечує гарну видимість з різних кутів.

Стандартна схема приєднання монітора безпосередньо до KEYESTUDIO ESP32 PLUS зображена на рисунку 2.28.

Підключіть червоний до V, чорний до G, синій з SDA, зелений до SCL.

Таблиця 2.8

Підключення LCD 1602 до KEYESTUDIO ESP32 PLUS

Модульний Pin	Колір дроту	ESP32 Pin плати
V	ЧЕРВОНИЙ	V
G	ЧОРНИЙ	G
SCL	ЗЕЛЕНИЙ	SCL
SDA	СИНІЙ	SDA

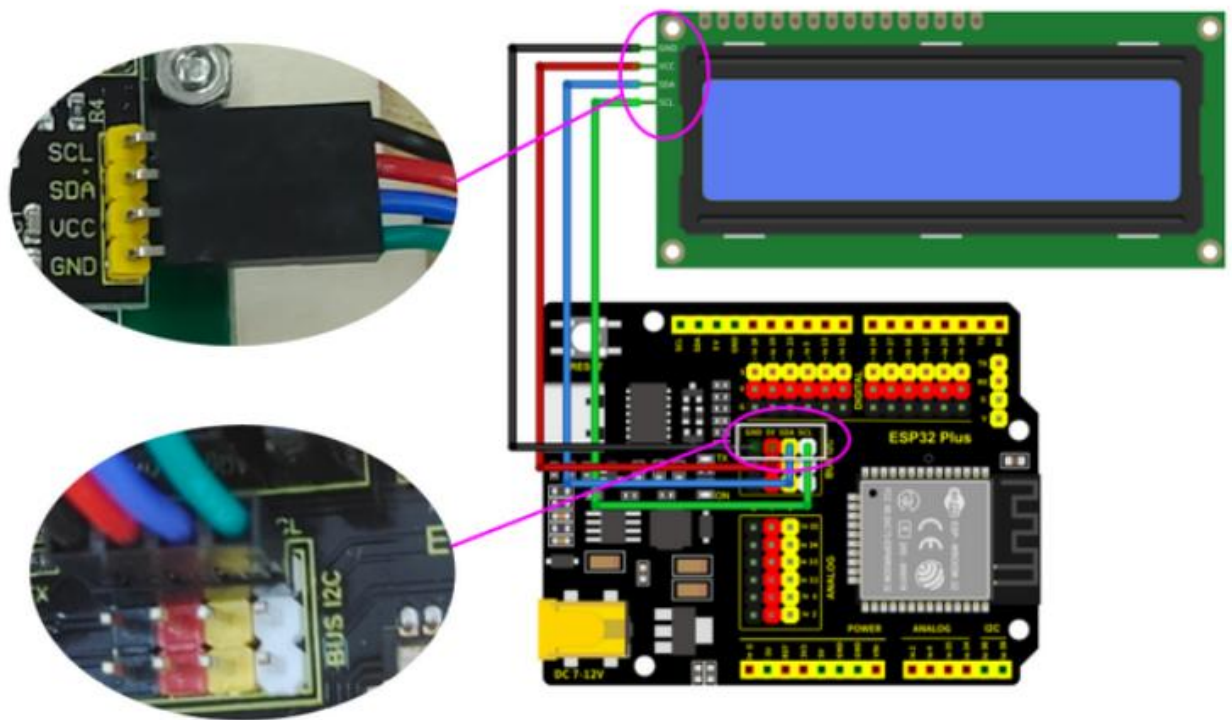


Рис. 2.28. Схема підключення LCD 1602 до KEYESTUDIO ESP32 PLUS

2.2. Вибір середовища програмної розробки

Arduino IDE - це інтегроване середовище розробки, призначене для створення програмного забезпечення для платформи ESP32. Воно надає зручний інтерфейс для написання коду, його компіляції та завантаження в мікроконтролер. Середовище підтримує широкий спектр платформ і містить бібліотеки функцій для роботи з різноманітними датчиками та виконавчими пристроями.

Особливості Arduino IDE:

1. Середовище розробки Arduino IDE має лаконічний дизайн, що дозволяє швидко освоїти його навіть новачкам в програмуванні.
2. Arduino IDE використовує спрощену версію мови C++, що робить процес написання коду більш доступним для широкого кола користувачів.
3. Arduino IDE включає в себе велику кількість стандартних бібліотек, які спрощують роботу з різноманітними сенсорами, актуаторами та іншими периферійними пристроями.

4. Arduino IDE підтримує широкий спектр платформи ESP32, що дозволяє використовувати єдине середовище розробки для різних проєктів.
5. Цей інструмент дозволяє відстежувати дані, що передаються між комп'ютером і платою ESP32, що значно спрощує процес налагодження програм.
6. Arduino IDE є проєктом з відкритим кодом, що дозволяє користувачам вносити зміни до його функціональності та розширювати його можливості.

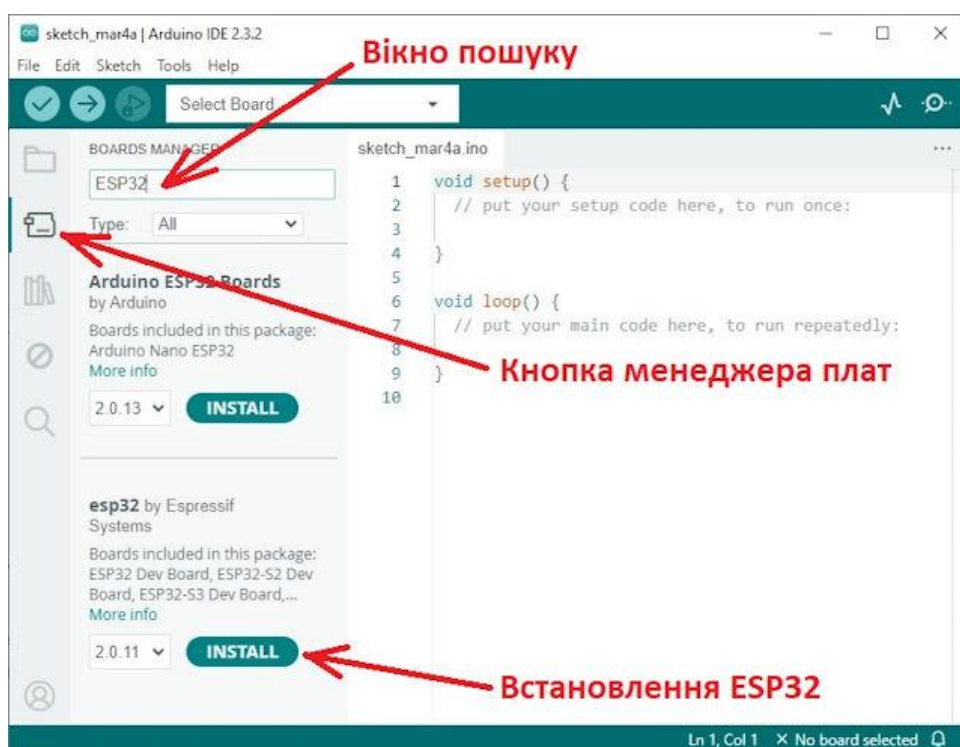


Рис. 2.29. Інтерфейс Arduino IDE

Скетчі, написані в середовищі розробки Arduino IDE, зберігаються у файлах з розширенням .ino. Редактор коду в IDE забезпечує стандартний набір функцій для роботи з текстом: копіювання, вставка, пошук і заміна. Інформація про хід виконання програми та можливі помилки відображається в області повідомлень. Вибір моделі плати ESP32 та послідовного порту здійснюється в нижній частині вікна програми. Панель інструментів надає швидкий доступ до основних функцій середовища розробки. На панелі інструментів знаходиться 6 кнопок (рис. 2.30):



Рис. 2.30. Панель інструментів

Основні функції Arduino IDE

- **Перевірити.** Дана опція дозволяє перевірити синтаксис написаного коду. У разі виявлення помилок вони відображаються в області повідомлень.
- **Завантажити.** Використовується для компіляції програми та її завантаження у мікроконтролер ESP32.
- **Створити.** Дозволяє створити новий файл для написання скетчу.
- **Відкрити.** Призначена для відкриття існуючого файлу зі скетчем.
- **Зберегти.** Дозволяє зберегти поточний скетч у файл.
- **Монітор послідовного порту.** Використовується для відкриття програми Serial Monitor, яка відображає дані, що надходять від ESP32 на комп'ютер через послідовний інтерфейс. Монітор підтримує роботу як із USB-варіантами плати, так і зі звичайними версіями ESP32. Для передачі даних на зовнішній пристрій у вікні монітора вводиться текст, після чого натискається кнопка "Відправити" або клавіша Enter. Швидкість передачі даних потрібно налаштувати відповідно до параметрів, вказаних у функції `Serial.begin()` у вашому коді. Функція `Serial.print()` дозволяє виводити текст у вікно монітора.

Бібліотеки значно розширюють функціональність програм, надаючи додаткові можливості, такі як робота з апаратними засобами або обробка даних. Для додавання бібліотеки слід перейти до меню **Sketch**, вибрати опцію **Include Library** і обрати необхідну бібліотеку. Після цього в програму додається оператор `#include`, а бібліотека компілюється разом зі скетчем. Важливо зазначити, що кожна підключена бібліотека використовує частину пам'яті мікроконтролера.

Більшість бібліотек уже попередньо встановлені разом із програмним забезпеченням ESP32, однак додаткові бібліотеки можна завантажити з зовнішніх джерел.

Для завантаження скетчу необхідно вибрати відповідну плату та порт (див. рис.2.31), які використовуються для вашої операційної системи. Це здійснюється через меню Tools, де у розділі Board вибирається модель плати, а в розділі портів — відповідний COM-порт (наприклад, COM1, COM2, COM4, COM5, COM7 тощо) або USB-з'єднання.

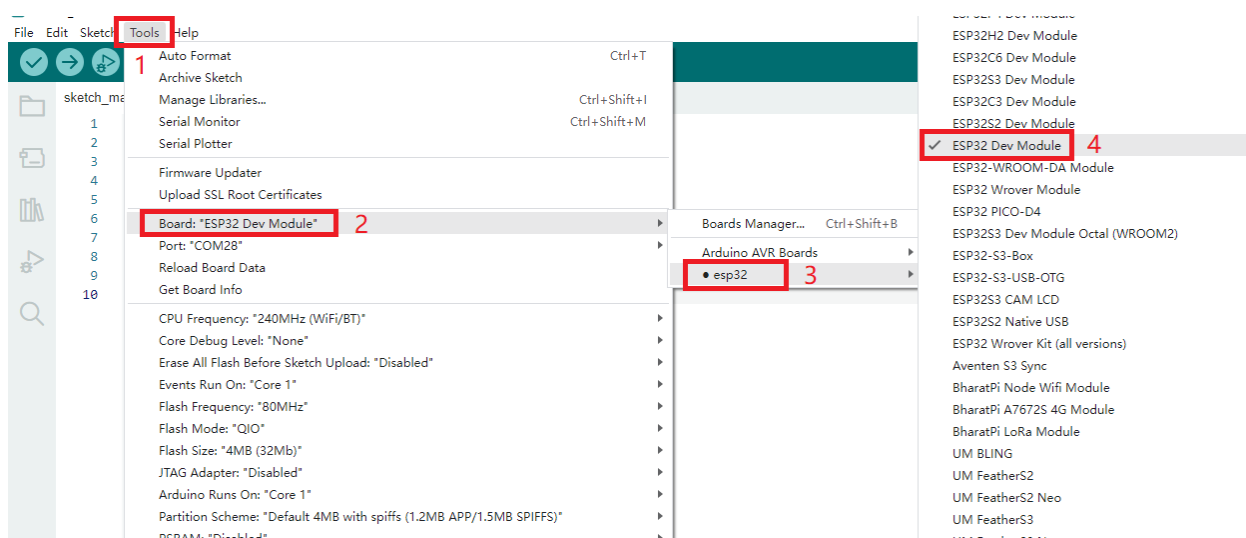


Рис. 2.31. Зображення вибору плати

Для завантаження програми в плату ESP32 необхідно встановити з'єднання між комп'ютером і платою за допомогою USB-кабелю. В середовищі розробки Arduino IDE слід вибрати відповідну модель плати та послідовний порт, до якого підключено Arduino. Ця інформація необхідна для того, щоб комп'ютер міг правильно спілкуватися з платою.

Після вибору потрібних параметрів можна приступати безпосередньо до процесу прошивки.

Для цього необхідно скомпілювати написаний код. Компілятор перетворить ваш код, написаний на спрощеній версії C++, в машинний код, який розуміє мікроконтролер Arduino. Скомпільований код передається в буфер завантажувача - спеціальної програми, яка зашита в пам'ять мікроконтролера. Завантажувач, у свою чергу, переписує отриманий код в основну пам'ять мікроконтролера і запускає його на виконання.

Щоб завантажити код програми на плату ESP32, нам знадобиться адаптер USB-послідовний (адаптер FTDI) або адаптер програматора ESP32.

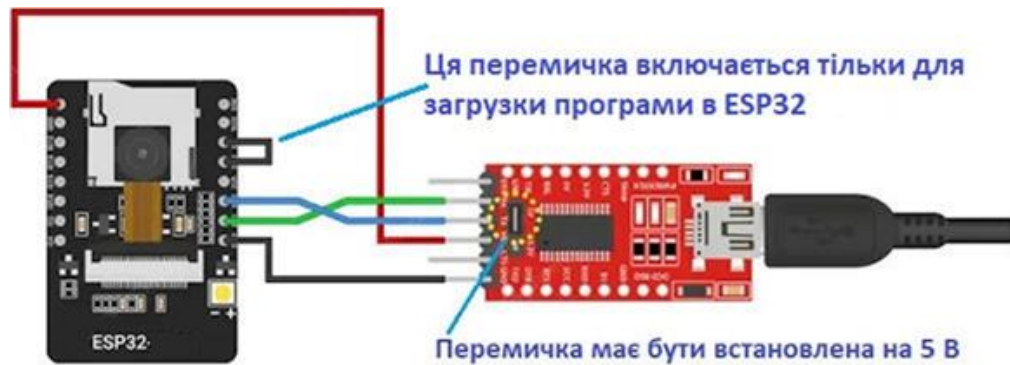


Рис. 2.32. Програмування та прошивка плати

Процес прошивки супроводжується скиданням мікроконтролера та миганням світлодіодів, що індикують активність передачі даних. Для забезпечення зручності користувача середовище розробки Arduino IDE надає інтуїтивно зрозумілий інтерфейс та автоматизує більшість рутинних операцій.

Функції `setup()` та `loop()` є основою будь-якого скетчу для Arduino.

- `setup()` виконується один раз на початку програми і використовується для ініціалізації портів введення-виведення, встановлення початкових значень змінних та підключення бібліотек.
- `loop()` виконується циклічно і містить основний алгоритм програми. Все, що має повторюватися, розміщується саме тут.

Основні функції для роботи з портами введення-виведення:

- Цифрові порти: `pinMode()`, `digitalWrite()`, `digitalRead()`. Ці функції дозволяють керувати цифровими виходами (наприклад, світлодіодами) та зчитувати значення з цифрових входів (наприклад, кнопок).
- Аналогові порти: `analogWrite()`, `analogRead()`, `analogReference()`. Ці функції використовуються для роботи з аналоговими сигналами, такими як напруга.
- Спеціальні функції: `tone()`, `noTone()` для генерації звукових сигналів, `millis()` для вимірювання часу.

Легкість тестування та відлагодження програмного забезпечення в Arduino IDE значно спрощується завдяки вбудованим інструментам, таким як серійний монітор і серійний лог, які дозволяють у режимі реального часу контролювати роботу системи, оперативно відслідковувати сигнали від датчиків і виявляти потенційні помилки в алгоритмах обробки даних. Крім того, Arduino IDE підтримує інтеграцію з різноманітними відладчиками та емуляторами, що забезпечує додатковий рівень перевірки коду ще на етапі прототипування.

Важливою складовою ефективності розробки є глобальна спільнота Arduino, яка надає великий обсяг готових бібліотек, прикладів коду, форумів і документації. Це не лише скорочує час на розробку, але й дозволяє обмінюватися досвідом, отримувати рекомендації щодо оптимізації роботи з датчиками та комунікаційними модулями, а також швидко вирішувати технічні проблеми.

Завдяки поєднанню інтуїтивно зрозумілого інтерфейсу, широких можливостей для відлагодження та доступу до ресурсів спільноти, Arduino IDE забезпечує надійну основу для розробки програмного забезпечення для нашої системи моніторингу вуликів. Використання цього середовища дозволяє не лише швидко налаштовувати і тестувати систему в польових умовах, але й підтримувати високу надійність роботи, гнучко адаптувати код до змін у конфігурації датчиків і масштабувати систему під різні умови експлуатації. Такий підхід забезпечує ефективну інтеграцію кіберфізичних компонентів та надійну обробку великих обсягів даних у режимі реального часу, що критично для сучасних IoT-рішень у бджільництві.

Висновки до розділу

У результаті проведеного аналізу сучасних апаратних платформ обґрунтовано вибір мікроконтролера для автоматизованої системи «Розумна ферма». Порівняння популярних рішень (Arduino, Raspberry Pi, ESP8266, ESP32) показало, що оптимальним ядром для побудови IoT-системи є двоядерний мікроконтролер ESP32. Він поєднує високу обчислювальну

потужність (процесор Xtensa LX6 до 240 МГц), енергоефективність (підтримка режиму Deep Sleep), наявність вбудованих бездротових інтерфейсів Wi-Fi та Bluetooth, а також велику кількість портів GPIO з підтримкою апаратних протоколів (I2C, SPI, UART) та 12-бітного ADC. Для практичної реалізації лабораторного стенду обрано покращену плату розробника KEYESTUDIO ESP32 PLUS, яка повністю покриває технічні потреби проєкту.

Сформовано та детально описано специфікацію периферійного обладнання й датчиків лабораторного стенду, а також схеми їх підключення до базової плати:

- Для побудови інтерфейсу користувача та локального керування обрано цифровий модуль тактильної кнопки EASY Plug RJ11.
- Для світлового та звукового сповіщення про поточний стан системи, запуск поливу чи аварійні ситуації інтегровано світлодіодний модуль Keyestudio LED Module та модуль пасивного п'єзоелектричного дзвінка (Passive Buzzer Module).
- Для автоматизації просторових дій та механічних елементів (наприклад, відкривання дверей чи кватирок) обрано легкий і компактний мікросервопривід Tower Pro SG90, який керується за допомогою ШІМ-сигналу.
- Безпечне керування виконавчими пристроями з високим струмом навантаження (водяним насосом системи поливу) реалізовано шляхом підключення електромеханічного релейного модуля SRD-5VDC-SL-C.
- Моніторинг просторових параметрів (виявлення руху та вимірювання відстаней) покладено на інфрачервоний датчик руху PIR Motion Sensor та високоточний ультразвуковий далекомір Ultrasonic Sensor HC-SR04.
- Візуалізацію текстових даних телеметрії в локальному режимі забезпечено за допомогою рідкокристалічного дисплея LCD 1602, підключеного через енергозберігаючу шину I2C.

Обґрунтовано вибір програмного забезпечення для розробки прошивки мікроконтролера. Як базове інтегроване середовище обрано кросплатформову систему Arduino IDE, яка підтримує архітектуру ESP32 та використовує спрощену мову C++. Вибір зумовлений наявністю великої кількості відкритих бібліотек для роботи з периферією, зручними вбудованими інструментами відлагодження й моніторингу в реальному часі (Serial Monitor), а також циклічною структурою побудови коду на основі функцій `setup()` та `loop()`. Це дозволяє забезпечити швидке прототипування, надійну обробку даних, гнучкість конфігурації та можливість подальшого масштабування IoT-системи.

РОЗДІЛ 3.

ПРАКТИЧНА РЕАЛІЗАЦІЯ ЛАБОРАТОРНОГО СТЕНДУ IoT

"РОЗУМНА ФЕРМА" НА ESP32

3.1. Модуль системи освітлення

3.1.1. Засвітити світлодіод

Пишемо код за допомогою Arduino IDE.

```
#define LED_BUILTIN 27 //LED pins

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                     // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                     // wait for a second
}
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюмо код.

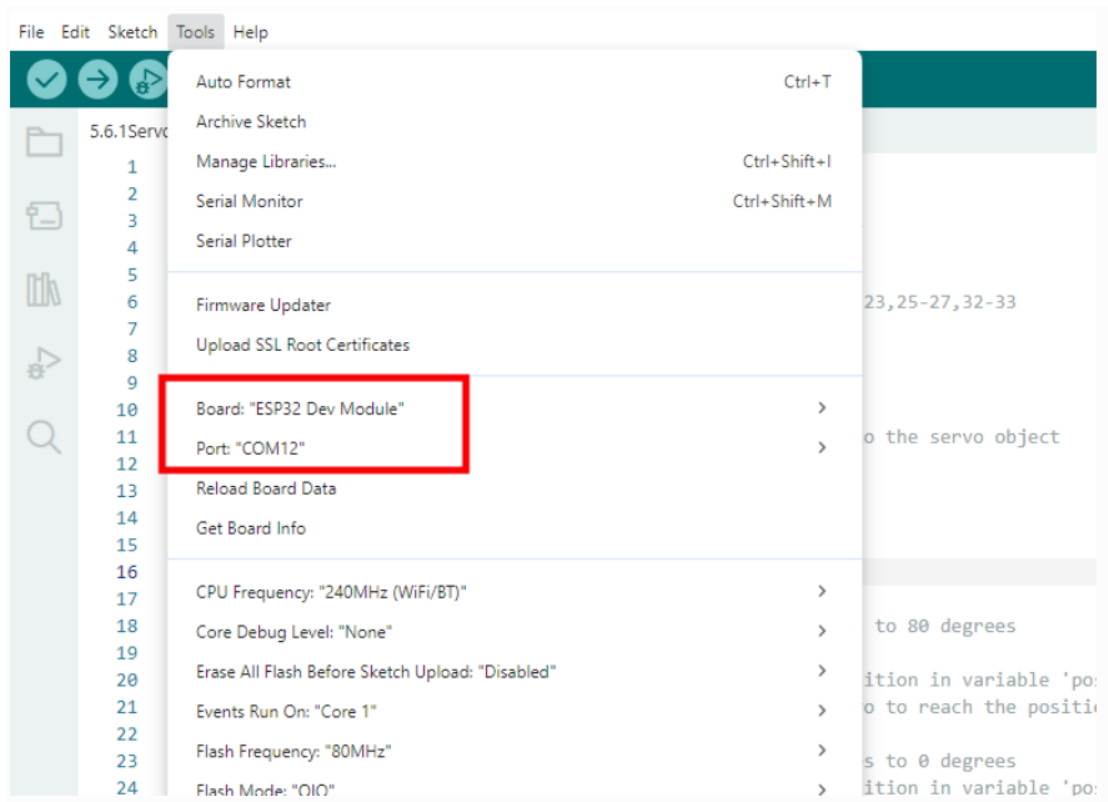


Рис. 3.1. Завантаження коду

Результат тесту:

LED миготить на секунду, оскільки іо27 на платі ESP32 видає по черзі високий і низький рівень кожну секунду.

РІВЕНЬ ПОТУЖНОСТІ	РЕЗУЛЬТАТ
ВИСОКИЙ	LED УВИМКНЕНО
НИЗЬКИЙ	LED ВИМКНЕНО

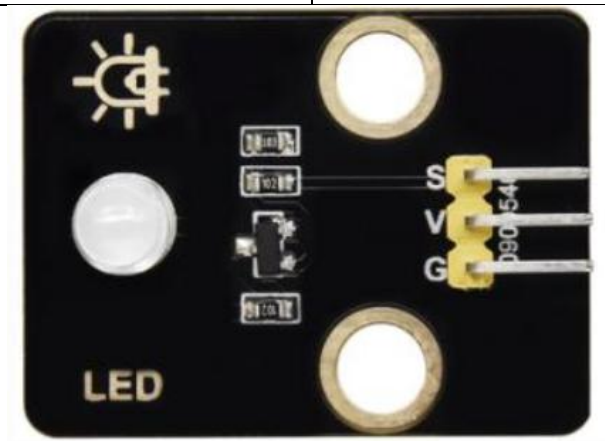


Рис. 3.2. LED миготіння

3.1.2. Керування світлодіодом за допомогою ШИМ

Пишемо код за допомогою Arduino IDE.

```
#define led 27    //Define LED pin

void setup(){
  pinMode(led, OUTPUT); //Set pin to output mode
}

void loop(){
  for(int i=0; i<255; i++) //for Loop statement. Constantly increase variable i till 255, exit the loop
  {
    analogWrite(led, i); //PWM output, used to control the brightness of LED
    delay(3);
  }
  for(int i=255; i>0; i--) //for Loop statement. Constantly decrease variable i till 0, exit the loop
  {
    analogWrite(led, i);
    delay(3);
  }
}
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.

Board: "ESP32 Dev Module" >

Port: "COM12" >

Результат тесту:

На відповідній частоті сигналу PWM змінює ефективну вихідну напругу, змінюючи робочий цикл за один період. Простими словами, у визначений час, чим вищий рівень виведення порту виводу, тим більший PWM-показник і світліший світлодіод.

LED-модуль повільно світиться з темного на яскраве, а потім від темного до яскравого.

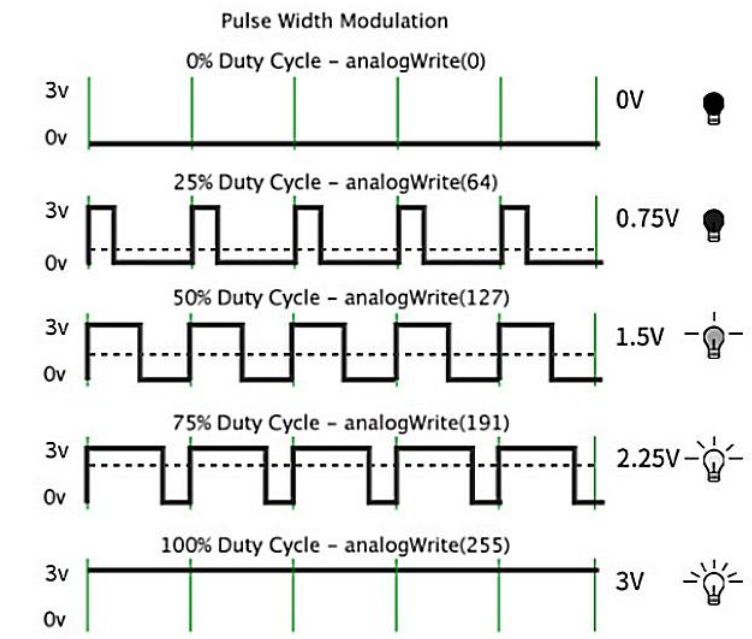


Рис. 3.3. Результати тесту

3.1.3. Прочитати цифрове значення Button

Пишемо код за допомогою Arduino IDE.

```
#define ButtonPin 5 //Define the button pin to 5

void setup() {
  //initialize serial port and set baud rate to 9600
  Serial.begin(9600);
  //Set pin to input mode
  pinMode(ButtonPin,INPUT);
}

void loop() {
  //Define a value as the read button value
  int ReadValue = digitalRead(ButtonPin);
  //Serial port prints the defined value
  Serial.print("The current status of the button is : ");
  Serial.println(ReadValue);
  delay(500);
}
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуємо код.



Результат тесту:

Відкриваємо послідовний монітор і встановимо швидкість передачі на 9600.

Коли кнопка відпускається, значення дорівнює 1; Якщо натиснути кнопку, результат стає 0.

Принцип кнопкового модуля — це схема, керована цією кнопкою.

Коли натискають кнопку, схема замикається, і струм проходить через кнопку до GND, що змушує цифровий вхідний контакт виявляти низький рівень.

Коли кнопка відпускається, схема розривається, і рівень штифтів підвищується завдяки підтягуючому резистору, який дозволяє цифровому контакту виявляти високого рівня.

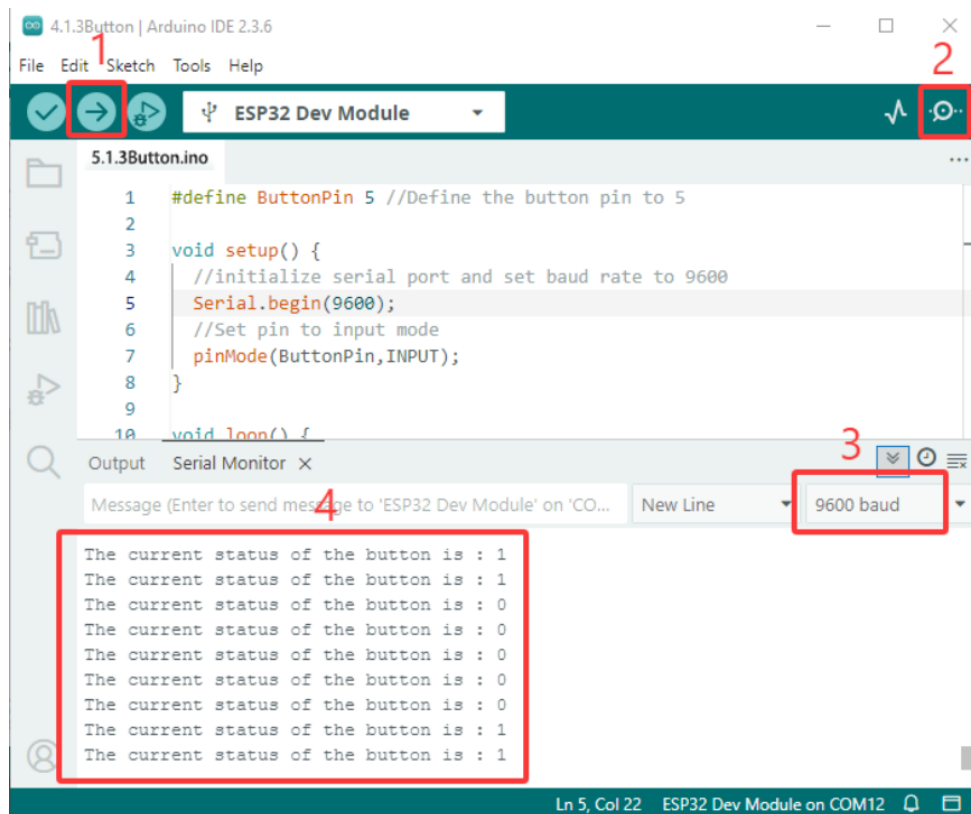


Рис. 3.4. Робота коду

3.1.4. Кнопка автоматичного блокування

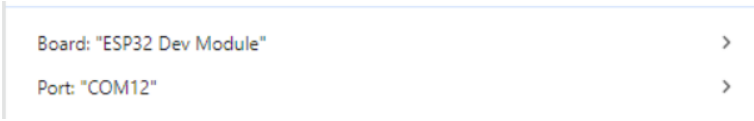
Пишемо код за допомогою Arduino IDE.

```
#define ButtonPin 5 //Define the button pin
int value = 0;      //Define a value to determine the status of button

void setup() {
    //Initialize the serial port and set baud rate to 9600
    Serial.begin(9600);
    //Set the pin to input mode
    pinMode(ButtonPin, INPUT);
}

void loop() {
    //Define a value as the read button value
    int ReadValue = digitalRead(ButtonPin);
    //Detect whether the button is pressed
    if (ReadValue == 0) {
        //Eliminate the button shake
        delay(10);
        if (ReadValue == 0) {
            value = !value;
            Serial.print("The current status of the button is : ");
            Serial.println(value);
        }
        //Detect again whether the button is still pressed
        //Pressed: execute the loop; Released: exit the loop to next step
        while (digitalRead(ButtonPin) == 0);
    }
}
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.



Board: "ESP32 Dev Module" >
Port: "COM12" >

Результат тесту:

Відкриваємо послідовний монітор і встановлюємо швидкість передачі на 9600.

Якщо натиснути кнопку один раз, з'явиться 1. Якщо натискати кнопку вдруге, значення стає 0. Тепер поширена кнопка має функцію автоматичного блокування.



Рис. 3.5. Робота коду

3.1.5. Використання кнопки для керування світлодіодним модулем

Пишемо код за допомогою Arduino IDE.

```
#define ButtonPin 5    //Define a button pin
#define LED         27 //Define LED pin
int value = 0;        //Define a value to detect button status

void setup() {
    //initialize serial port and set baud rate to 9600
    Serial.begin(9600);
    //Set pin to input mode
    pinMode(ButtonPin,INPUT);
    //Set pin to output mode
    pinMode(LED,OUTPUT);
}

void loop() {
    //Define a value as the read button value
    int ReadValue = digitalRead(ButtonPin);
    //Detect whether the button is pressed
    if (ReadValue == 0) {
        //Eliminate the button shake
        delay(10);
        if (ReadValue == 0) {
            value = !value;
            //Detect the button status, press once to light up LED, press again to turn off LED, in a loop
            if(value) {
                digitalWrite(LED,HIGH);
            }else{
                digitalWrite(LED,LOW);
            }
        }
        //Detect the button status again
        //Pressed: execute the loop; Released: exit the loop to next step
        while (digitalRead(ButtonPin) == 0);
    }
}
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.

Board: "ESP32 Dev Module" >

Port: "COM12" >

Результат тесту:

Якщо натиснути кнопку один раз, світлодіод загоряється; Якщо натиснути ще раз, світлодіод вимикається. Ця операція є петлею, що відповідає принципу освітлення в реальності.

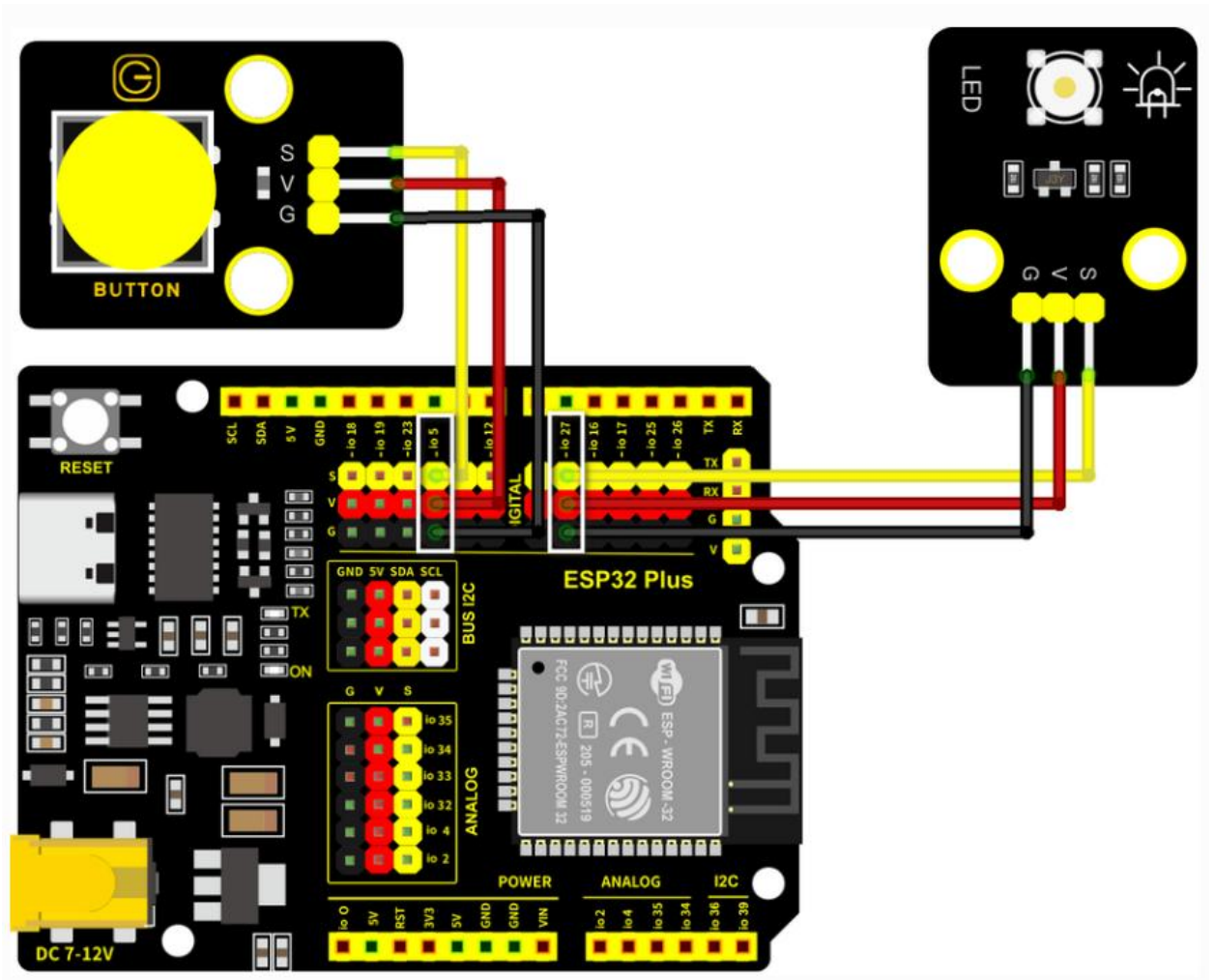


Рис. 3.6. Схема підключення до KEYESTUDIO ESP32 PLUS

3.2. Модуль системи керування світлом

3.2.1. Фотокомірковий сенсор

Пишемо код за допомогою Arduino IDE.

```

#define PhotocecllPin 34 //Define the photoresistor pin

void setup() {
  //Initialize the serial port
  Serial.begin(9600);
  //Set the pin to input mode
  pinMode(PhotocecllPin, INPUT);
}

void loop() {
  //Read the value of photoresistor
  int ReadValue = analogRead(PhotocecllPin);
  //Print the value. NOTE: ESP32 board is 12-bit ADC, whose detection value range is within 0~4095.
  Serial.print("Photocecll value: ");
  Serial.println(ReadValue);
  delay(500);
}

```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.

Board: "ESP32 Dev Module" >

Port: "COM12" >

Результат тесту:

Відкрий послідовний монітор.

Чим яскравіше світло, яке фіксує фоторезистор, тим більшим буде значення.

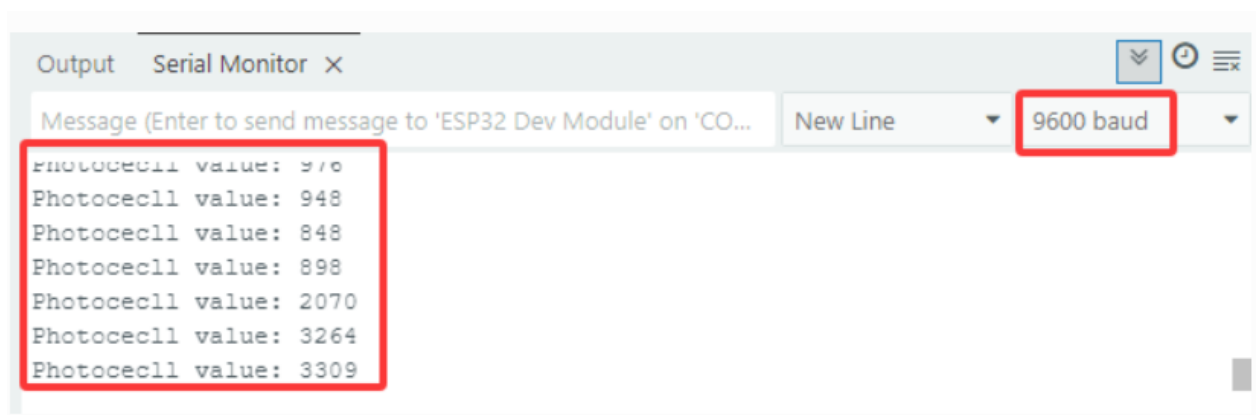


Рис. 3.7. Результати тесту

Фоторезисторний модуль перетворює світловий сигнал на електричний (напруга, струм і резистор). Коли світло потрапляє на фоторезистор, чим сильніше світло, тим менший опір, тому тим більша напруга VCC проходить через фоторезистор.

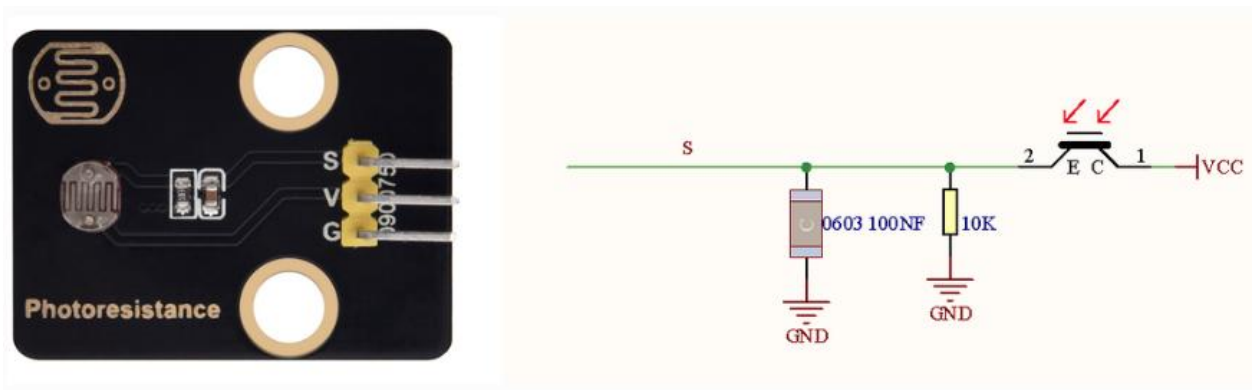


Рис. 3.8. Фоторезисторний модуль

3.2.2. Система керування світлом

Пишемо код за допомогою Arduino IDE.

```
#define PhotocecllPin 34 //Define the photoresistor pin
#define LED          27 //Define LED pin

void setup() {
    //Initialize serial port
    Serial.begin(9600);
    //Set the photoresistor pin to input mode
    pinMode(PhotocecllPin,INPUT);
    //Set the LED pin to output mode
    pinMode(LED,OUTPUT);
}

void loop() {
    //Read the value of the photoresistor
    int ReadValue = analogRead(PhotocecllPin);
    //Print the value. NOTE: ESP32 board is 12-bit ADC, whose detection value range is within 0~4095.
    Serial.print("Photocecll value: ");
    Serial.println(ReadValue);
    //Determine:
    //The value of the photoresistor >= 800, LED turns off
    //The value of the photoresistor <= 800, LED turns on
    if(ReadValue >= 800) {
        digitalWrite(LED,LOW);
        Serial.println("LED OFF");
    }
    else{
        digitalWrite(LED,HIGH);
        Serial.println("LED ON");
    }
    delay(100);
}
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.

Board: "ESP32 Dev Module"	>
Port: "COM12"	>

Результат тесту:

Коли значення фоторезистора перевищує 800 (вдень), світлодіод вимикається. Однак, якщо значення менше 800, світлодіод автоматично загоряється.

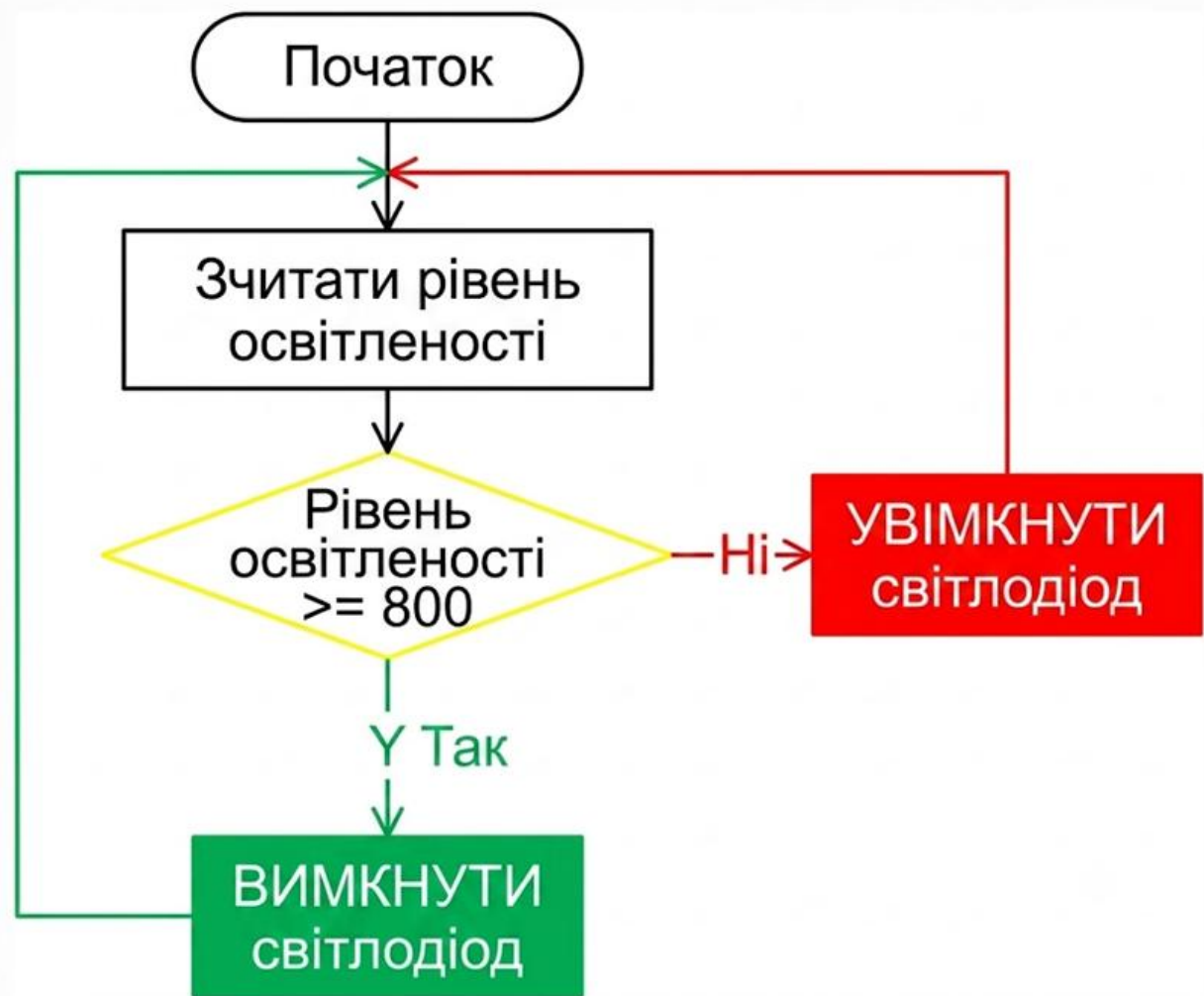


Рис. 3.9. Алгоритм роботи

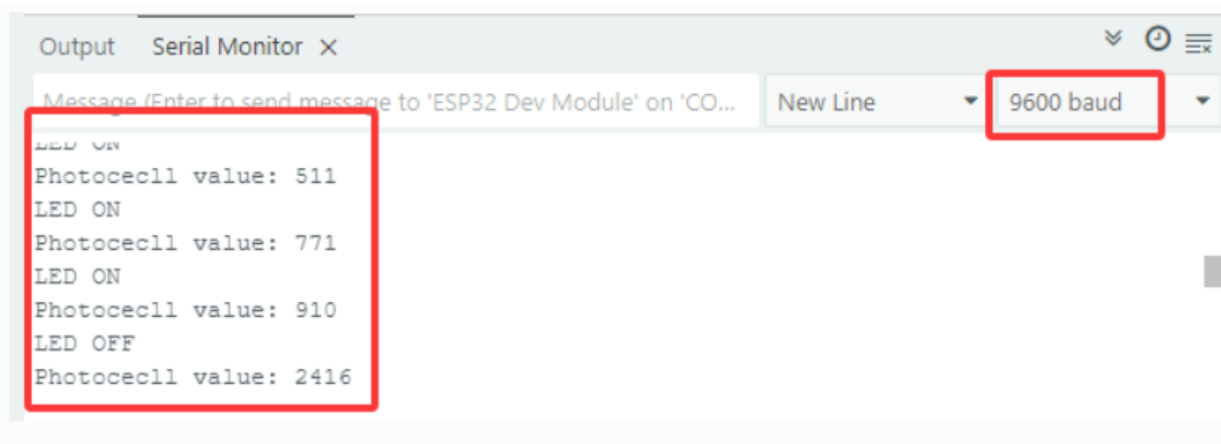


Рис. 3.10. Результати тесту

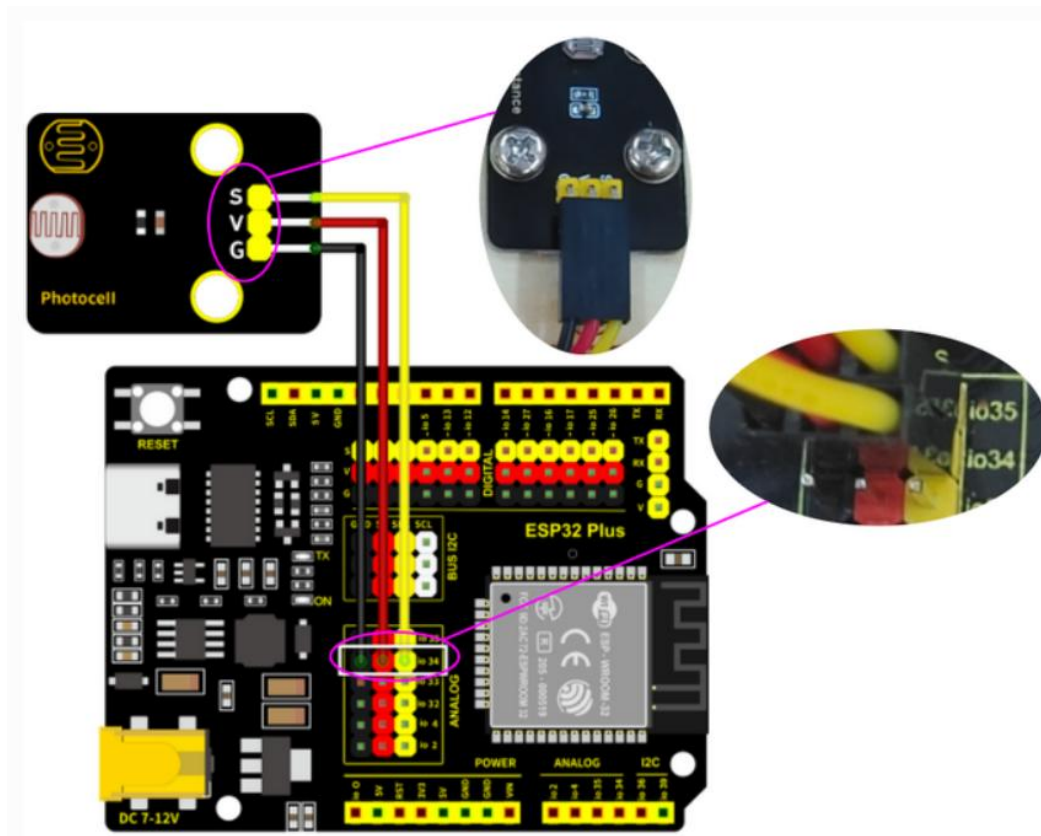


Рис. 3.11. Схема підключення фоторезистора до KEYESTUDIO ESP32 PLUS

3.3. Модуль системи сигналізації

3.3.1. Датчик руху PIR

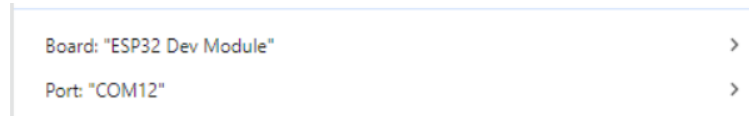
Пишемо код за допомогою Arduino IDE.

```
#define PyroelectricPIN 23

void setup() {
  Serial.begin(9600);
  pinMode(PyroelectricPIN, INPUT);
}

void loop() {
  //Read the value of PIR motion sensor
  int ReadValue = digitalRead(PyroelectricPIN);
  if(ReadValue){
    Serial.println("Someone");
  }
  else{
    Serial.println("No one");
  }
  delay(100);
}
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.



Результат тесту:

Коли хтось знаходиться в районі, на моніторі відображається «Хтось», і червоний світлодіод на сенсорі вмикається. Однак, якщо нікого немає, ніхто не буде надрукований, і світлодіод на сенсорі завжди буде увімкненим.

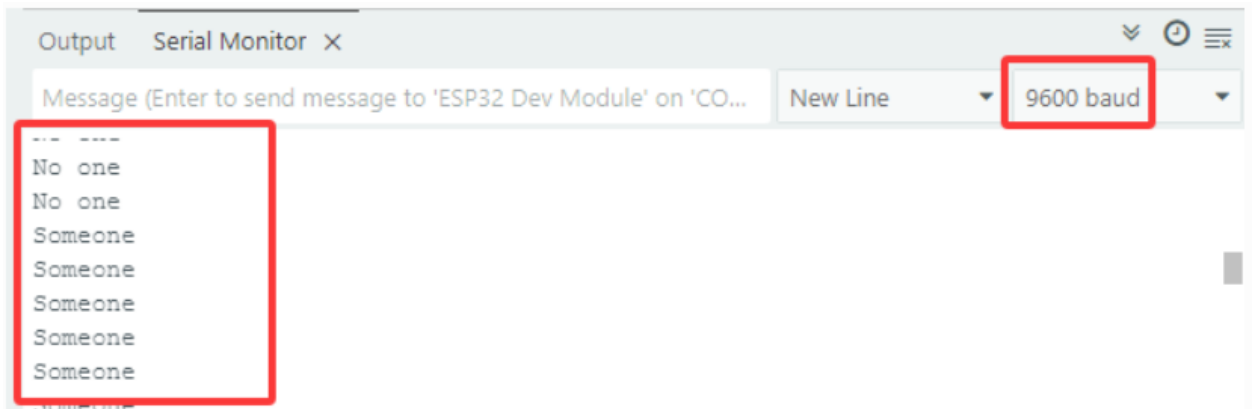


Рис. 3.12. Результати тесту

3.3.2. Пасивний дзвінок

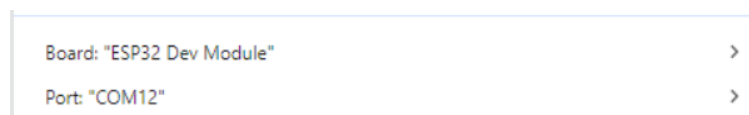
Пишемо код за допомогою Arduino IDE.

```
#define BuzzerPin 16 //Define the buzzer pin

void setup() {
  //Set the pin to output mode
  pinMode(BuzzerPin,OUTPUT);
}

void loop() {
  digitalWrite(BuzzerPin,HIGH);
  delayMicroseconds(500);//DeLay 500us
  digitalWrite(BuzzerPin,LOW);
  delayMicroseconds(500);//DeLay 500us
}
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.



Результат тесту:

Пасивний Buzzer постійно видає звук.

Звук дзвінка

Пишемо код за допомогою Arduino IDE.

```
const int buzzerPin = 16;    //Set buzzer pin to 16
void setup() {
    ledcAttachChannel(buzzerPin,1000,8,4);
}
void loop() {
    ledcWriteTone(buzzerPin,532);           //duo --C2
    delay(100);
    ledcWriteTone(buzzerPin,587);           //re --D3
    delay(100);
    ledcWriteTone(buzzerPin,659);           //mi --E3
    delay(100);
    //Alarm
    for(int i = 200; i<=1000; i+=10){
        ledcWriteTone(buzzerPin,i);
        delay(10);
    }
    //Alarm
    for(int i = 1000; i>=200; i-=10){
        ledcWriteTone(buzzerPin,i);
        delay(10);
    }
    ledcWriteTone(buzzerPin,0);
}
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.

Board: "ESP32 Dev Module" >

Port: "COM12" >

Результат тесту:

Сигналізація дзвінка через функцію. `ledcWriteTone()`

`ledcWriteTone()` генерує PWM-сигнал з певною частотою, щоб приводити вібрацію дзвінка, а тривалість і тон регулюються відповідними параметрами.

Функцію потрібно використовувати разом із функцією `ledcWriteTone()``ledcAttachChannel()`

ledcAttachChannel

Ця функція використовується для встановлення обов'язків для каналу LEDC.

```
bool ledcWriteChannel(uint8_t channel, uint32_t duty);
```

- `channel` виберіть канал LEDC.
- `duty` вибрати обов'язок для вибраного каналу LEDC.

Ця функція повернеться, якщо завдання встановлення буде успішним. Якщо повернено, виникає помилка, і мито не встановлено. `true``false`

ledcWriteTone

Ця функція використовується для налаштування контакту LEDC на 50 % PWM тону на вибраній частоті.

```
uint32_t ledcWriteTone(uint8_t pin, uint32_t freq);
```

- `pin` Виберіть LEDC PIN.
- `freq` Виберіть частоту PWM-сигналу. Якщо частота — , duty буде

встановлено на 0. `0`

Ця функція поверне set для LEDC pin. Якщо повернено, виникає помилка, і контакт LEDC не налаштований. `frequency``0`

3.3.3. Система сигналізації

Пишемо код за допомогою Arduino IDE.


```

#define BuzzerPin 16          //Set buzzer pin to 16
#define PyroelectricPIN 23    //Set PIR motion sensor to 23
#define Led 27                //Set led pin to 27

void setup() {
  Serial.begin(9600);
  //Set the pins modes
  pinMode(PyroelectricPIN, INPUT);
  pinMode(Led, OUTPUT);

  ledcAttachChannel(BuzzerPin, 1000, 8, 4);
}
void loop() {
  //Read the value of PIR motion sensor
  int ReadValue = digitalRead(PyroelectricPIN);
  if(ReadValue){
    Serial.println("Someone");
    digitalWrite(Led, HIGH);
    //Alarm
    for(int i = 200; i <= 1000; i += 10){
      ledcWriteTone(BuzzerPin, i);
      delay(10);
    }
    digitalWrite(Led, LOW);
    //Alarm
    for(int i = 1000; i >= 200; i -= 10){
      ledcWriteTone(BuzzerPin, i);
      delay(10);
    }
  }
  //Stop alarming
  ledcWriteTone(BuzzerPin, 0);
  Serial.println("No one");
}

```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.

Board: "ESP32 Dev Module" >

Port: "COM12" >

Результат тесту:

Коли датчик виявляє рух, дзвінок видає звук, а світлодіод миготить, нагадуючи про вторгнення.

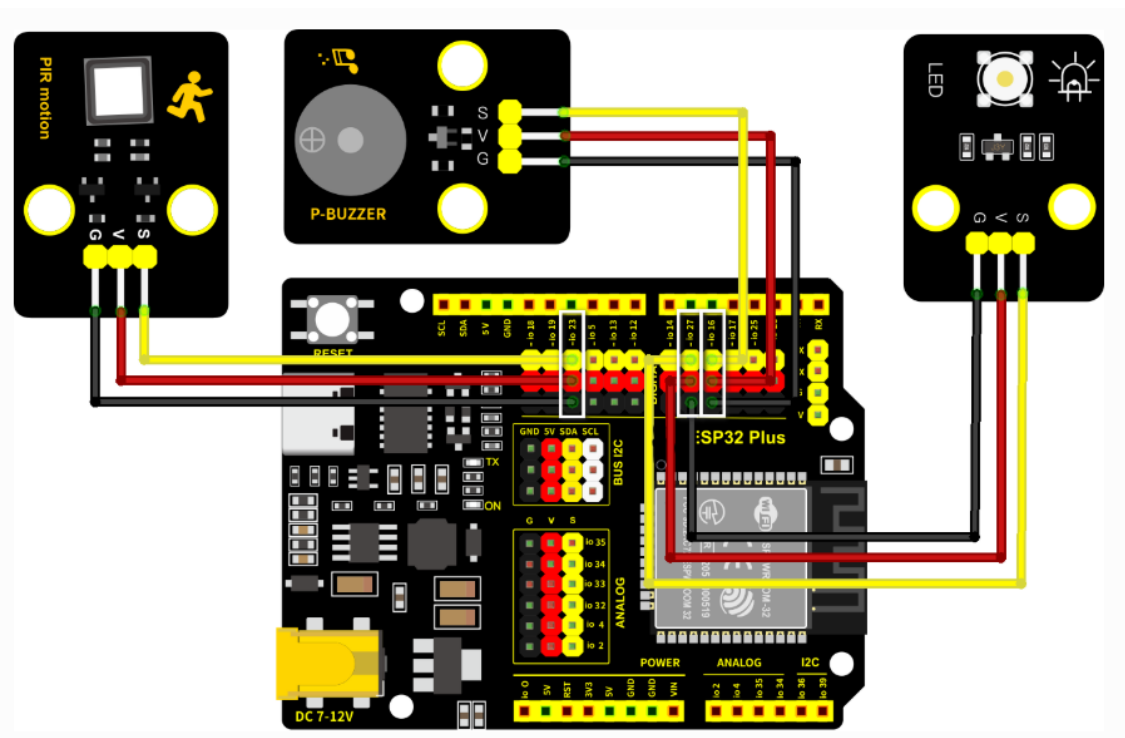


Рис. 3.13. Схема модулю сигналізації

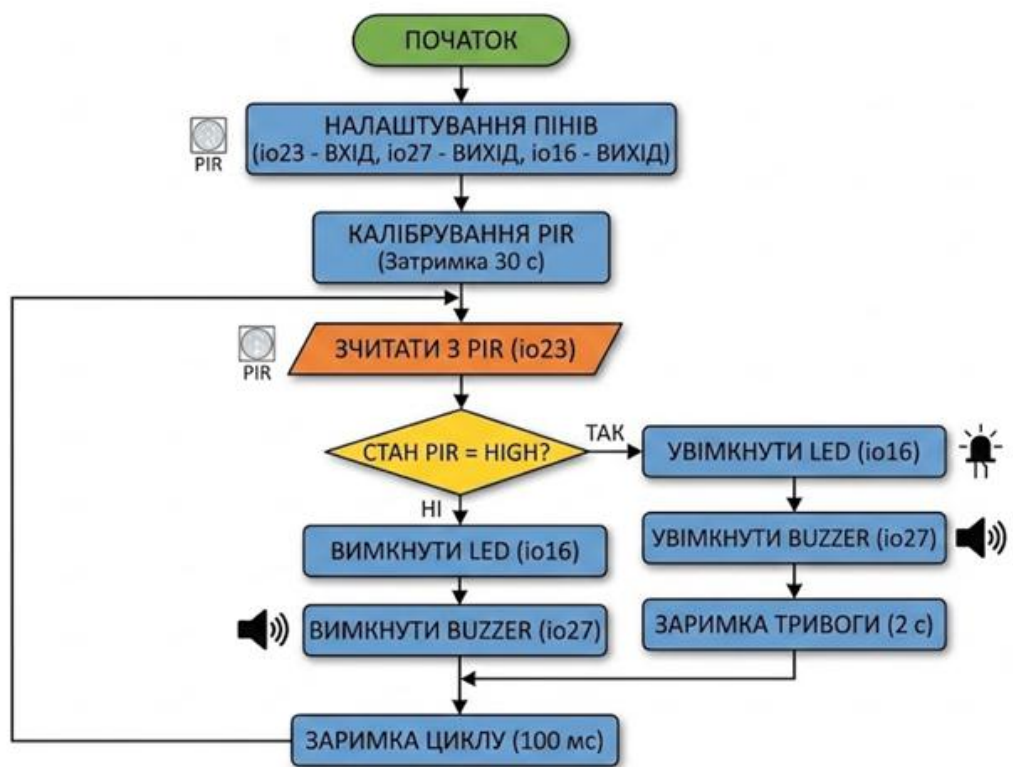


Рис. 3.14. Блок-схема алгоритму роботи модулю сигналізації

3.4. Модуль розумної системи годування тварин

3.4.1. Двері живильної kabini

Пишемо код за допомогою Arduino IDE.

```

#include <ESP32Servo.h> //Import the library of servo
Servo myservo; // create servo object to control a servo
                // 16 servo objects can be created on the ESP32

int pos = 0;    // variable to store the servo position
// Recommended PWM GPIO pins on the ESP32 include 2,4,12-19,21-23,25-27,32-33
int servoPin = 26;

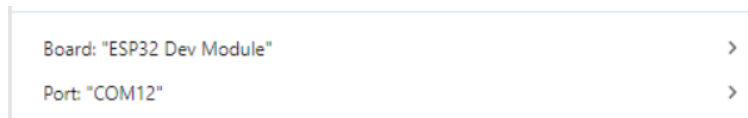
void setup() {
  Serial.begin(9600);
  myservo.attach(servoPin); // attaches the servo on pin 26 to the servo object
  myservo.write(180);
  delay(2000);
}

void loop() {

  for (pos = 80; pos <= 179; pos += 1) { // goes from 0 degrees to 80 degrees
    // in steps of 1 degree
    myservo.write(pos);                // tell servo to go to position in variable 'pos'
    delay(15);                          // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 81; pos -= 1) { // goes from 80 degrees to 0 degrees
    myservo.write(pos);                // tell servo to go to position in variable 'pos'
    delay(15);                          // waits 15ms for the servo to reach the position
  }
}

```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.



Підключимо Servo до контакту IO26 плати ESP32. Підключіть жовтий до S, червоний до V, чорний до G.

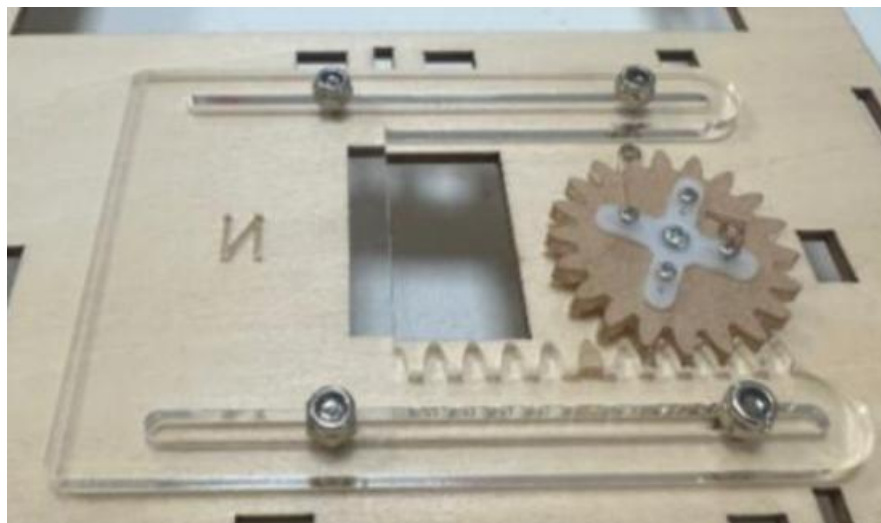


Рис. 3.15. Принцип роботи дверей

3.4.2. Інтелектуальна система годування

Розумна система годування інтелектуально годує домашніх птахів за допомогою ультразвукового модуля та сервоприводу. Перший визначає відстань до тварин, а другий керує відкриттям або закриттям ящика для годування. Коли тварину помічають біля коробки, серво відкриває її, щоб поїсти.

Пишемо код за допомогою Arduino IDE.

```
#include <ESP32Servo.h> //Import the library of servo on ESP32 board
Servo myservo; // create servo object to control a servo
                // 16 servo objects can be created on the ESP32

#define TrigPin 12 //connect trig to D12
#define EchoPin 13 //connect echo to D13
#define ServoPin 26
int duration,distance;

void setup(){

  Serial.begin(9600); //Set the baud rate to 9600
  pinMode(TrigPin,OUTPUT); //set trig pin to output mode
  pinMode(EchoPin,INPUT); //Set echo pin to input mode
  myservo.attach(ServoPin); // attaches the servo on pin 26 to the servo object
}
void loop(){
  Serial.println(getDistance());
  //When the distance is detected within 2~7cm, open the feeding box. Or else, close.
  if (getDistance() >= 2 && 7 >= getDistance()) {
    //Servo rotates to 80° to open the box
    myservo.write(80);
    delay(500);
  }
  else{
    myservo.write(180);
    delay(500);
  }
}

//Put the gotten distance in a function
float getDistance() {

  digitalWrite(TrigPin,LOW);
  delayMicroseconds(2);
  digitalWrite(TrigPin,HIGH);
  delayMicroseconds(10); //Trigger the trig pin via a high level lasting at least 10us
  digitalWrite(TrigPin,LOW);
  duration = pulseIn(EchoPin,HIGH); //the time of high level at echo pin
  distance = duration/58; //convert into distance(cm)
  delay(50);

  return distance;
}
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуємо код.

Board: "ESP32 Dev Module"

Port: "COM12"

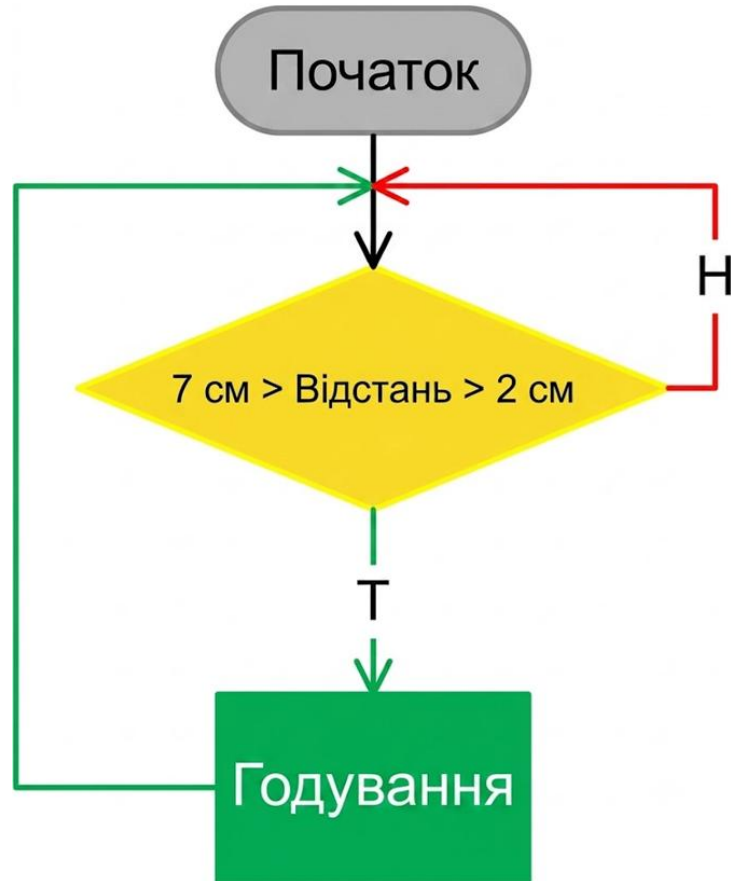


Рис. 3.16. Блок схема роботи інтелектуальної системи годування

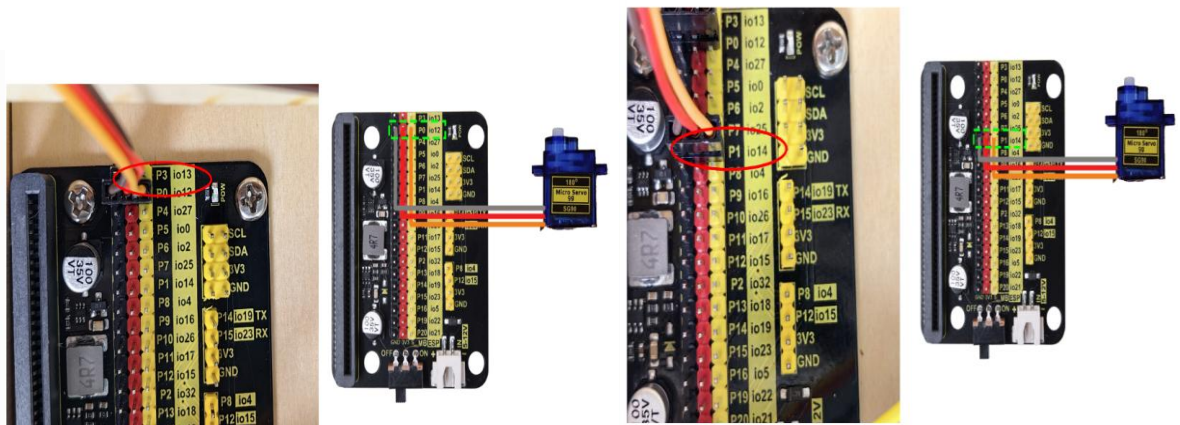


Рис. 3.17. Схема підключення серво до KEYESTUDIO ESP32 PLUS

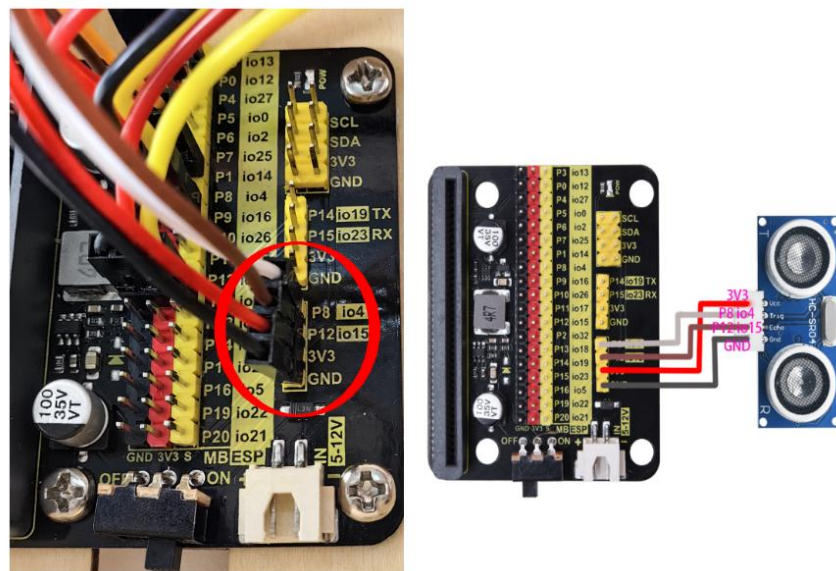


Рис. 3.18. Підключення Ultrasonic-Sensor

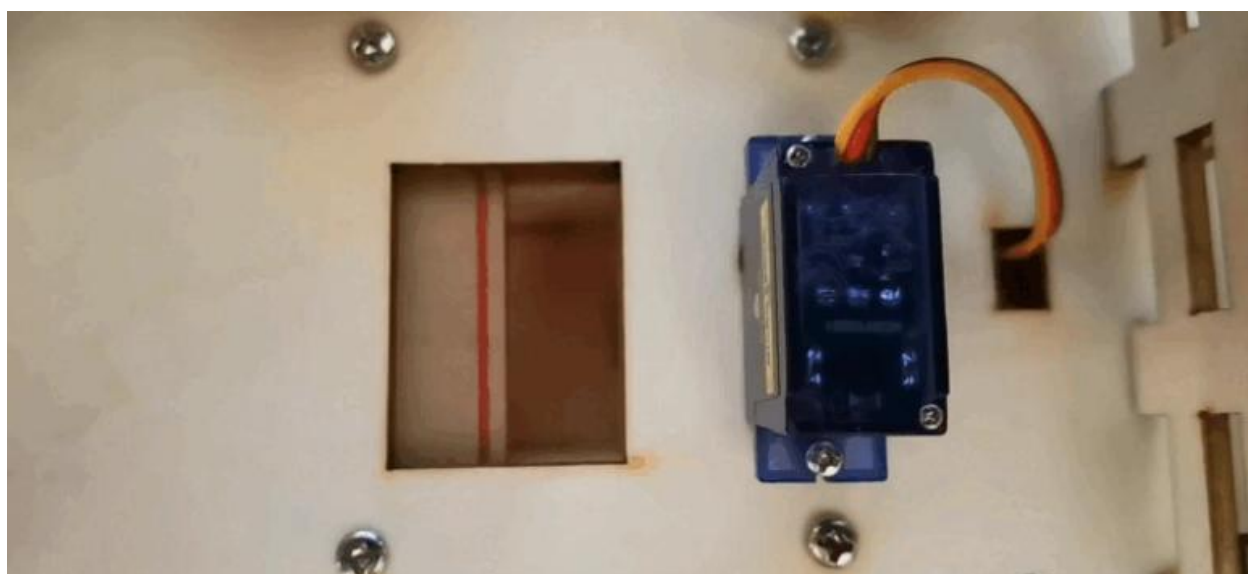


Рис. 3.19. Робота модулю інтелектуальної системи годування

3.5. Модуль системи автоматичного зрошення

3.5.1. Система відкачування води

У цьому експерименті ми використовуємо плату ESP32 для вмикання/вимикання водяного насоса за допомогою релейного модуля. Насос піднімає воду та транспортує рідини, зазвичай у використанні поєднується з релейним модулем.

Пишемо код за допомогою Arduino IDE.

```

#define RelayPin 25
char content; //Define a character string as the received value from serial port

void setup() {
    Serial.begin(9600);
    pinMode(RelayPin,OUTPUT);
}

void loop() {
    //Serial.read() receives one byte once. For example, when input "aaa", it receives one "a" at a time
    if(Serial.available() > 0) {
        if (Serial.read() == 'a') //When the input value equals to "a", irrigation begins.
        {
            digitalWrite(RelayPin,HIGH);
            delay(400); //irrigation delay
            digitalWrite(RelayPin,LOW);
            delay(700);
        }
    }
}

```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.

Board: "ESP32 Dev Module" >

Port: "COM12" >

Результат тесту:

Відкрийте послідовний монітор і введіть «а», накачайте воду один раз.

Зверніть увагу: не переливайте воду з пластикових басейнів під час експериментів. Розлив води на інші датчики може спричинити не лише коротке замикання або вимкнення модулів, а й генерацію тепла та навіть вибух.

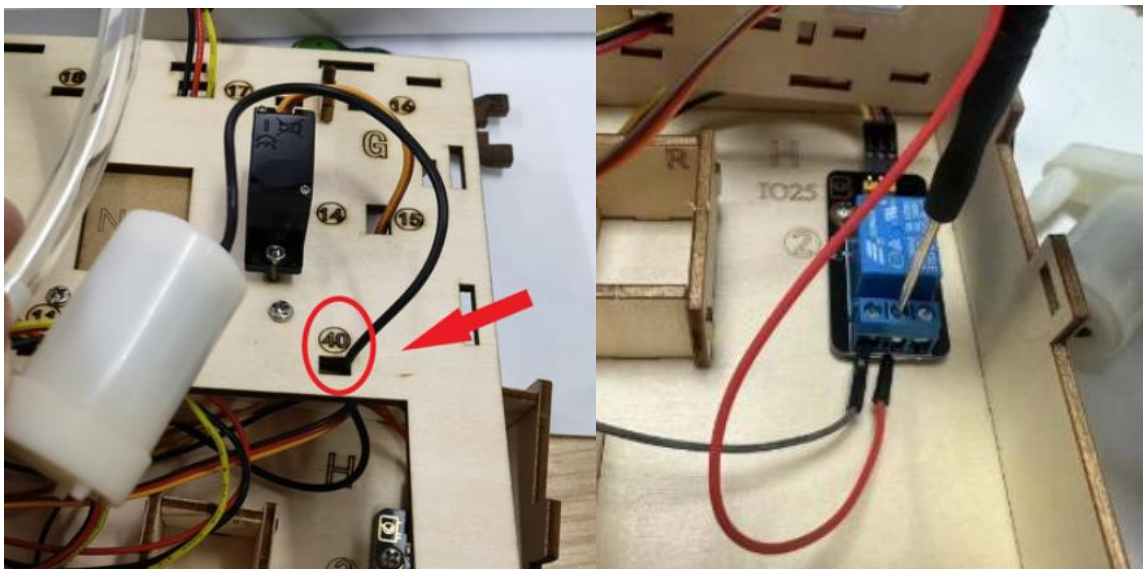


Рис. 3.20. З'єднання модулю відкачування води

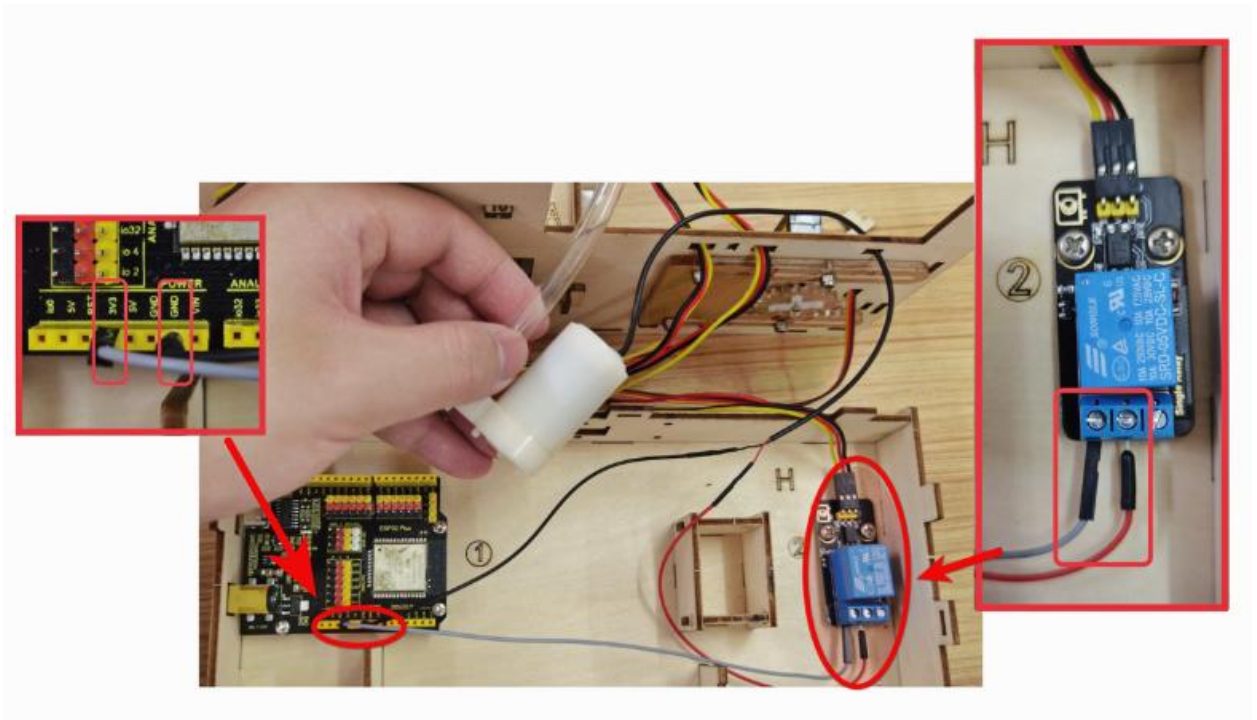


Рис. 3.21. З'єднання модулю відкачування води

3.5.2. Система автоматичного зрошення

У цьому експерименті ми підключаємо два датчики до плати ESP32 і програмуємо, щоб зчитувати їхні вихідні значення для керування реле та водяним насосом.

Якщо ґрунт дуже сухий, реле вмикається, щоб керувати водяним насосом для зрошення рослин; А якщо рівень води занадто низький, водяний насос не зможе працювати, і дзвінок спрацює.

Пишемо код за допомогою Arduino IDE.

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.

Board: "ESP32 Dev Module" >

Port: "COM12" >


```

#include <LiquidCrystal_I2C.h>

#define BuzzerPin 16
#define SoilHumidityPin 32
#define WaterLevelPin 33
#define RelayPin 25
#define ButtonPin 5 //Define a button pin
int value = 0;      //Set an initial button value

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
    //Set the pins mode
    pinMode(SoilHumidityPin, INPUT);
    pinMode(WaterLevelPin, INPUT);
    pinMode(RelayPin, OUTPUT);
    pinMode(ButtonPin, INPUT);

    //Initialize LCD
    lcd.init();
    //Turn on LCD backlight
    lcd.backlight();
    //Clear LCD displays
    lcd.clear();

    ledcAttachChannel(BuzzerPin, 1000, 8, 4);
}

void loop() {
    //define variables as the read values of water level, humidity and button state
    int shvalue = analogRead(SoilHumidityPin);
    int wlvalue = analogRead(WaterLevelPin);
    int ReadValue = digitalRead(ButtonPin);

    //Set the display position of cursor
    lcd.setCursor(0, 0);
    //Set the display position of character strings
    lcd.print("SoilHum:");
    lcd.setCursor(9, 0);
    lcd.print(shvalue);
    lcd.setCursor(0, 1);
    lcd.print("WaterLevel:");
    lcd.setCursor(11, 1);
    lcd.print(wlvalue);

```

```

//Determine whether the button is pressed
if (ReadValue == 0) {
    //Eliminate the button shake
    delay(10);
    if (ReadValue == 0) {
        value = !value;
        Serial.print("The current status of the button is : ");
        Serial.println(value);
    }
    //Again, determine whether the button is still pressed
    //Pressed: execute the Loop; Released: exit the Loop to next execution
    while (digitalRead(ButtonPin) == 0)
        ;
}
//When the detected humidity is lower than the set threshold, the buzzer starts to alarm. Press
if (500 >= shvalue && value == 0) {
    ledcWriteTone(BuzzerPin, 532);
    delay(100);
    ledcWriteTone(BuzzerPin, 532);
    delay(100);
    ledcWriteTone(BuzzerPin, 659);
    delay(100);
    ledcWriteTone(BuzzerPin, 0); //Stop alarming
}
//When the detected water level is Lower than the set threshold, the buzzer starts to alarm. Pre:
if (500 >= wlvalue && value == 0) {
    ledcWriteTone(BuzzerPin, 411);
    delay(100);
    ledcWriteTone(BuzzerPin, 639);
    delay(100);
    ledcWriteTone(BuzzerPin, 411);
    delay(100);
    ledcWriteTone(BuzzerPin, 0); //Stop alarming
}
//When the detected humidity is lower than the set threshold, and the water is sufficient in the
if (500 >= shvalue && wlvalue >= 1000) {
    digitalWrite(RelayPin, HIGH);
    delay(400); //Irrigation delay.
    digitalWrite(RelayPin, LOW);
    delay(700);
}
delay(500);
//Clear displays
lcd.clear();
}

```

Результат тесту:

LCD 1602 відображатиме поточні значення вологості ґрунту та рівня води.

Коли виявлене значення вологості ґрунту нижче 500, дзвінок сповіщає, що ґрунт стає сухим. Якщо рівень води перевищує 1000, зрошення починається автоматично.

Коли виявлений рівень води нижчий за 500, система відкачування води не працює, і сигналізація дзвінка повідомляє, що води недостатньо. Натисніть кнопку, щоб перестати тривожити.



Рис. 3.22. Результати тесту на екрані

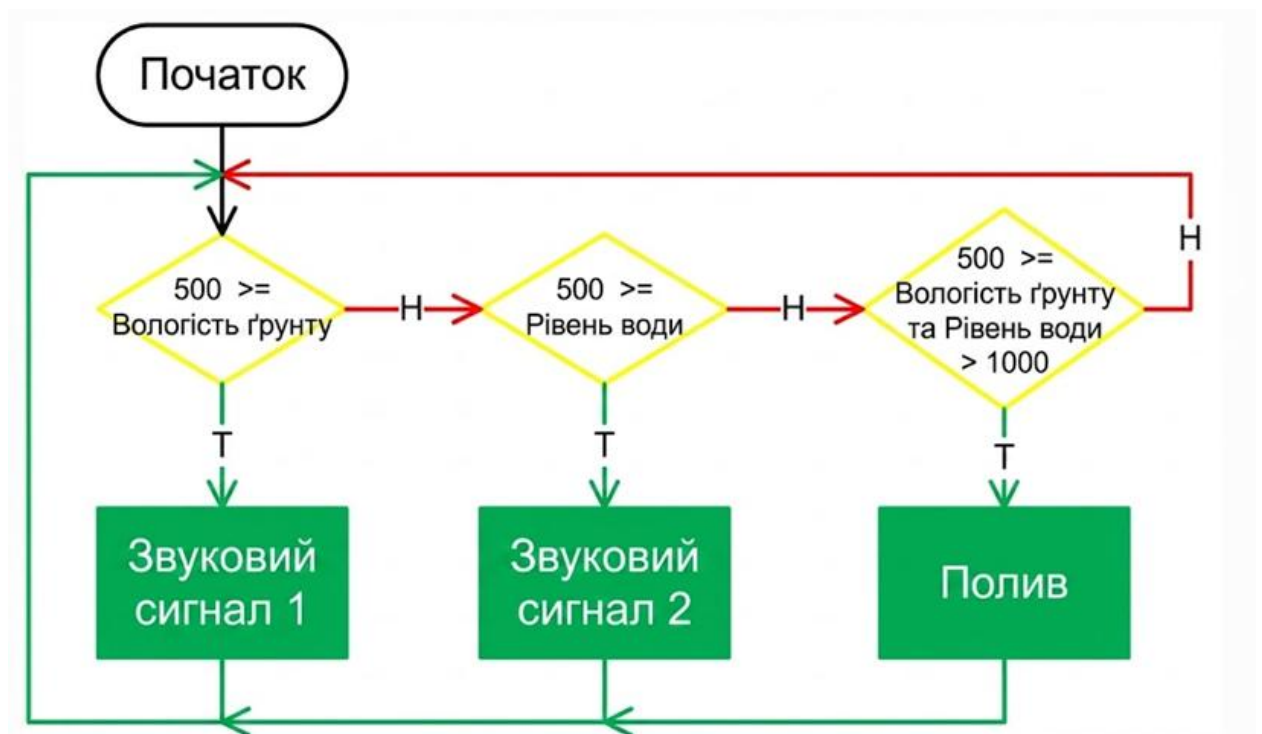


Рис. 3.23. Алгоритм роботи модуля

3.6. Розробка веб-керуванням розумної ферми

3.6.1. Підключення плати ESP32 до мережі

Плата ESP32 оснащена Wi-Fi (2.4G) та Bluetooth (4.2), що дозволяє їй легко підключатися до WiFi та спілкуватися з іншими пристроями в мережі.

Що потрібно підготувати:

- WiFi 2.4 ГГц (це може бути мобільний хотспот або роутер)

-Ім'я та пароль WIFI

-Телефон/iPad/комп'ютер, який може підключатися до того самого Wi-Fi.

Arduino IDE надає вам бібліотечний файл <WiFi.h>, який підтримує Конфігурації Wi-Fi та моніторинг мережі ESP32.

Режим базової станції (STA або Wi-Fi клієнтський режим): у цьому режимі ESP32 підключається до Wi-Fi точки доступу (AP).

Режим точки доступу (Soft-AP або Wi-Fi хотспот): У цьому режимі інші Wi-Fi пристрої підключаються до ESP32.

AP-STA режим: У цьому режимі ESP32 є Wi-Fi точкою доступу, а також Wi-Fi пристроєм, підключеним до іншої Wi-Fi точки доступу.

Ці режими сумісні з кількома безпечними режимами, такими як WPA, WPA2 та WEP. Він може сканувати Wi-Fi точки доступу, включно з активним і пасивним скануванням. Підтримує режим розпусності для моніторингу Wi-Fi пакетів IEEE802.11.

Пишемо код за допомогою Arduino IDE.

```
#include <WiFi.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

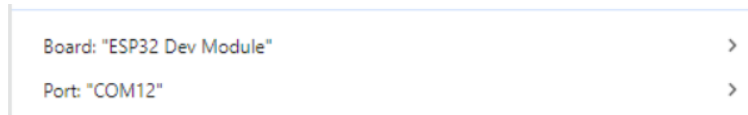
void setup() {
  Serial.begin(9600);
  //Initialize Wifi
  WiFi.begin(ssid, password);
  //Scan for wifi. If connection fails, stay in connecting, and execute "while" loop
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  //Connected. Print the IP address
  Serial.println("Connected to WiFi");
  Serial.println(WiFi.localIP());
}

void loop() {
}
```

Змініть код на назву вашого Wi-Fi і на пароль від Wi-Fi: `your_SSIDyour_PASSWORD`

```
const char* ssid = "your_SSID";  
const char* password = "your_PASSWORD";
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюмо код.



Результат тесту:

Завантажте код, і плата підключиться до Wi-Fi мережі та надрукує IP-адресу на послідовному моніторі.



Рис. 3.24. Результати тесту

3.6.2. Створення веб-сайту

Поки є підключення до Wi-Fi, бібліотека веб-серверів ESP32 може надавати веб-сторінки. У наступному прикладі коду ми створили простий вебсайт, щоб показати «Hello, World!».

Пишемо код за допомогою Arduino IDE.

```

#include <WiFi.h>
#include <WebServer.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

WebServer server(80); //Set the server port to 80. Enter the website by IP address rather than the port

//Initialize the website
void handleRoot() {
    //Used to send HTTP to the client-side for response, sending 200 means success.
    server.send(200, "text/html", "<h1>Hello, World!</h1>");
}

void setup() {
    Serial.begin(9600);
    //Initialize wifi
    WiFi.begin(ssid, password);
    //Scan for wifi. If connection fails, stay in connecting, and execute "while" loop
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi...");
    }

    //Connected. Print the IP address
    Serial.println("Connected to WiFi");
    Serial.println(WiFi.localIP());

    server.on("/", handleRoot);
    //Start server
    server.begin();
    Serial.println("Web server started");
}

void loop() {
    server.handleClient();
}

```

Змініть код на назву вашого Wi-Fi і на пароль від Wi-Fi. Потім завантажте код. `your_SSIDyour_PASSWORD`

```

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.

```

Board: "ESP32 Dev Module"
Port: "COM12"

```

Результат тесту:

У цьому прикладі коду ми створюємо бібліотеку Web server by WebServer на ESP32. Функція handleRoot() запитує обробку в кореневому

шляху та надсилає HTML-відповідь «Привіт, світ!» на сторону клієнта. Потім `setup()` встановлює `rootroute`, а `server.begin()` запускає веб-сервер.

Натисніть на послідовний монітор, щоб переглянути IP-адресу:

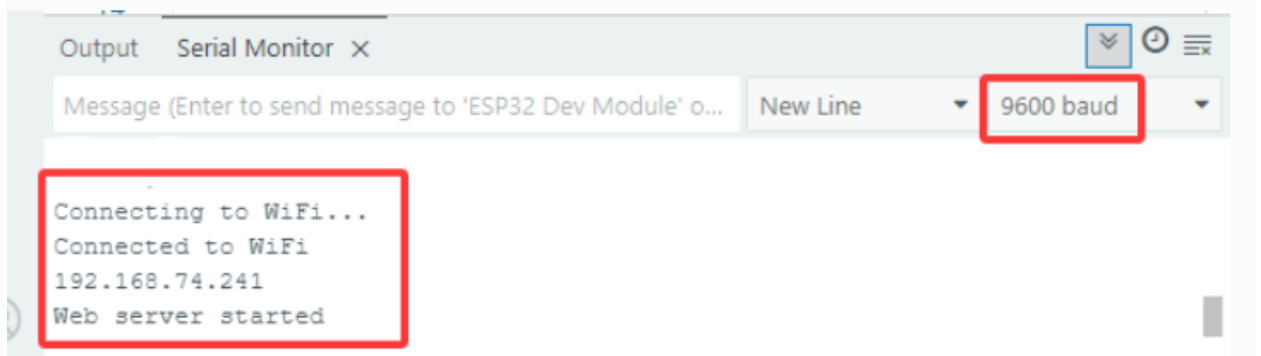
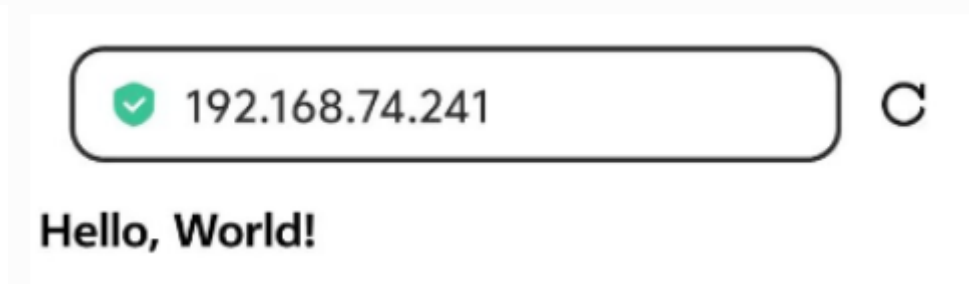


Рис. 3.25. Результати тесту

Коли ПК, мобільні телефони та плата ESP32 підключені до одного Wi-Fi, ви можете одночасно відвідати цей вебсайт як на ПК, так і на телефонах.

Доступ до IP у браузері ПК або телефоні:



Потрібен 2.4 GHz Wi-Fi, а не 5G. ПК або мобільний телефон, що отримує доступ до IP-адреси, має бути підключений до того ж Wi-Fi, що й плата ESP32.

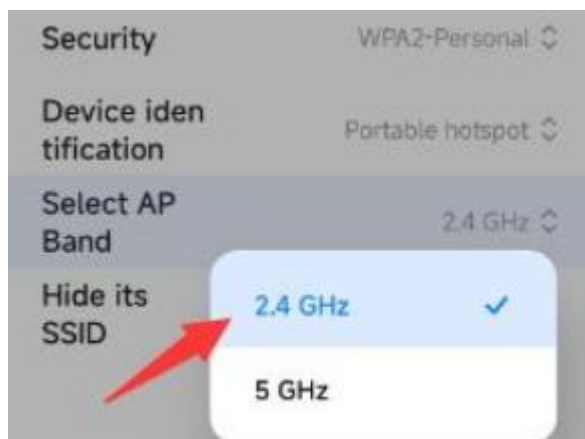


Рис. 3.26. Налаштування

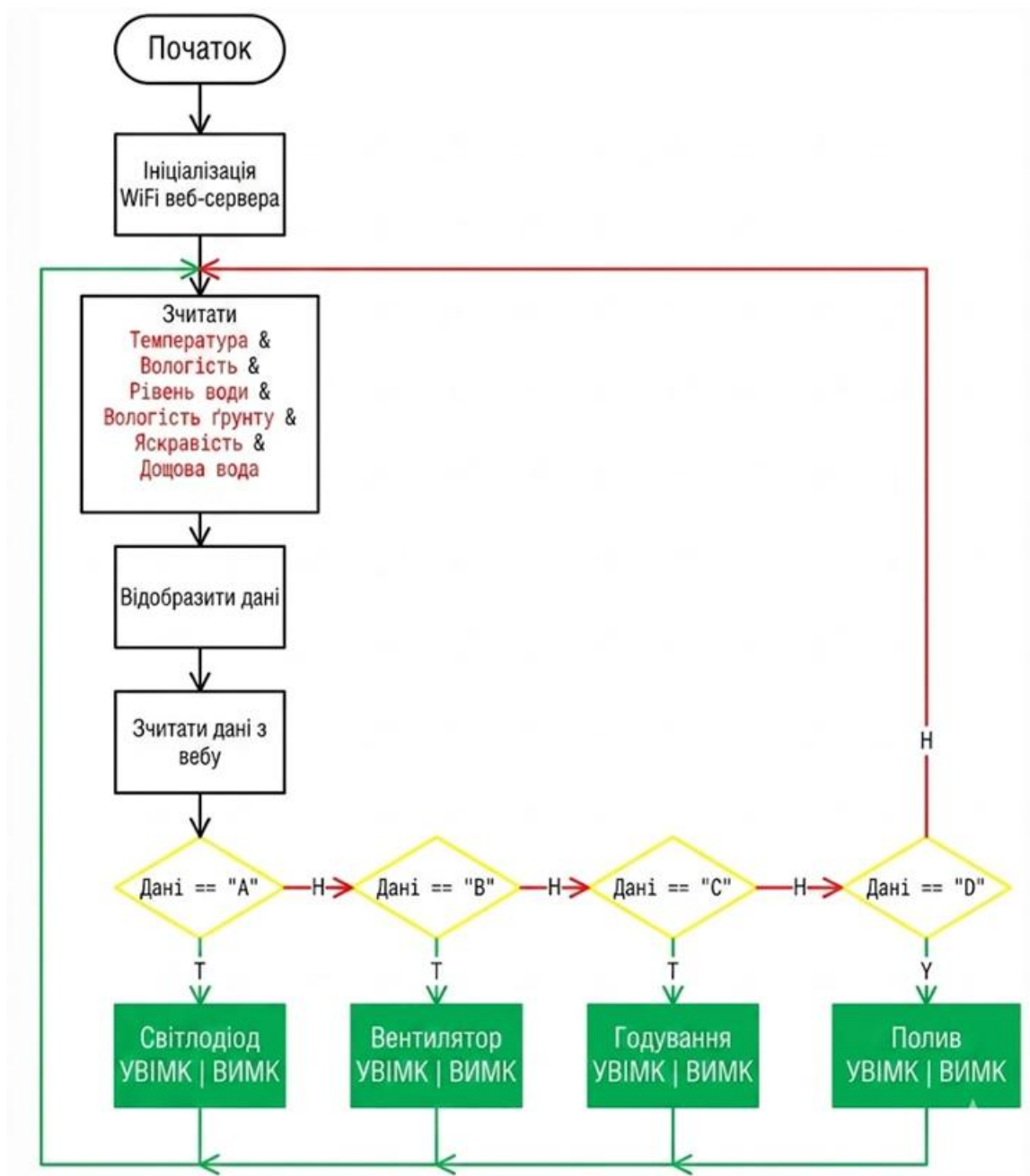


Рис. 3.27. Алгоритм роботи web-додатку

Пишемо код за допомогою Arduino IDE який наведено в додатку А.

Змініть код на назву вашого Wi-Fi і на пароль від Wi-Fi. Потім завантажте код. `your_SSIDyour_PASSWORD`

```
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.

Board: "ESP32 Dev Module" >

Port: "COM12" >

Результат тесту:

Переглянути IP-адресу з LCD1602:



Рис. 3.28. Результати тесту

Введіть IP-адресу в браузерах мобільних телефонів або ПК, ви можете побачити значення сенсора у верхній частині сторінки і керувати світлодіодом, вентилятором, подавальною кабіною та водяним насосом за допомогою кнопок нижче.

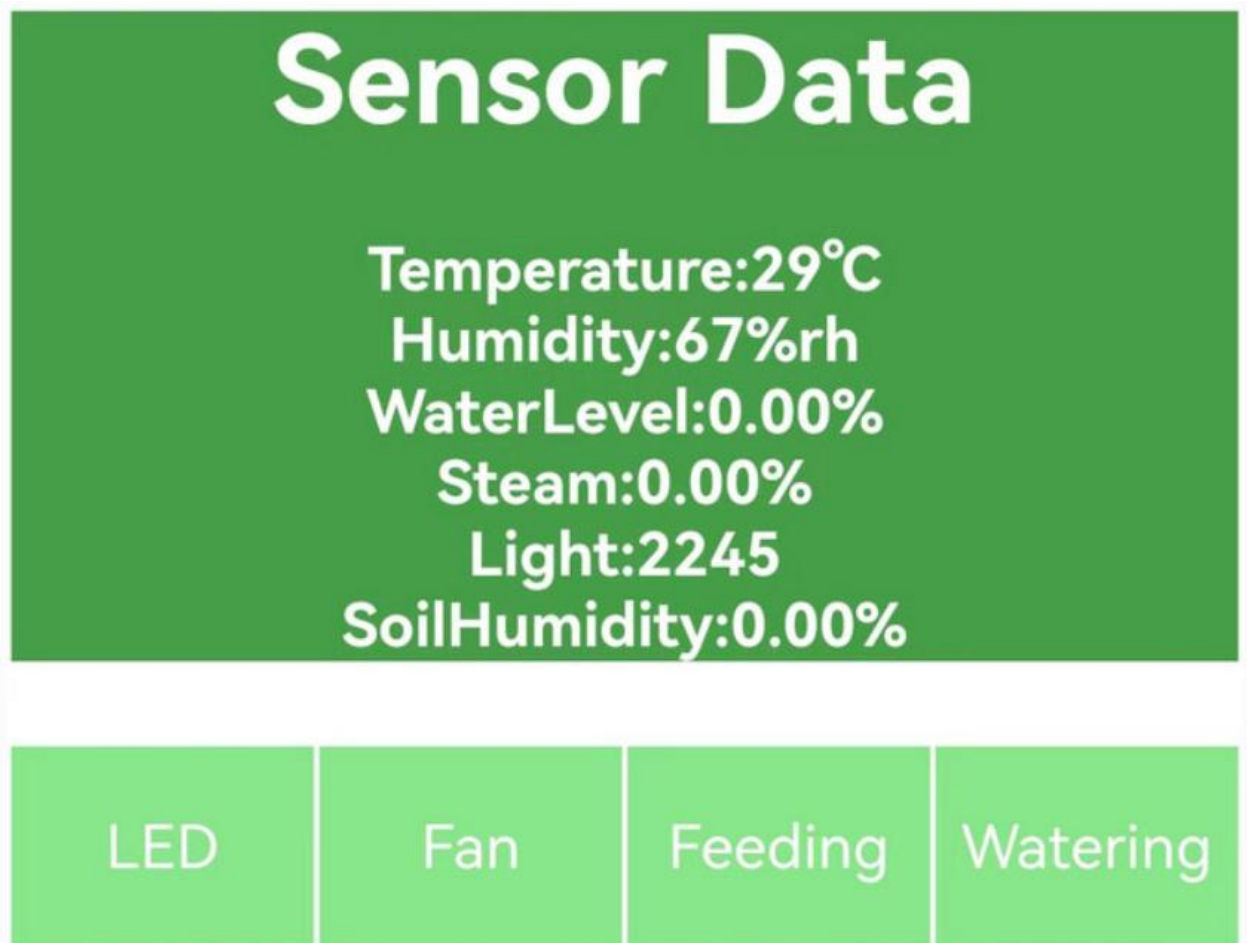


Рис. 3.29. Відображення веб-сторінки

3.7. Розробка APP додатку для керування розумної ферми

Пишемо код за допомогою Arduino IDE який наведено в додатку Б.

Змініть код на назву вашого Wi-Fi і на пароль від Wi-Fi. Потім завантажте код. `your_SSIDyour_PASSWORD`

```
const char* ssid = "your_SSID";  
const char* password = "your_PASSWORD";
```

Обираємо плату **ESP32 Dev Module** і **COM-порт**, і завантажуюємо код.

Board: "ESP32 Dev Module" >
Port: "COM12" >



Рис. 3.30. Головна сторінка додатку

Опис функції APP

Після завантаження коду підключіть телефон до того ж Wi-Fi, що й ESP32, для підключення достатньо ввести IP-адресу у верхньому правому контакті.



Рис. 3.31. Відображає IP-адресу

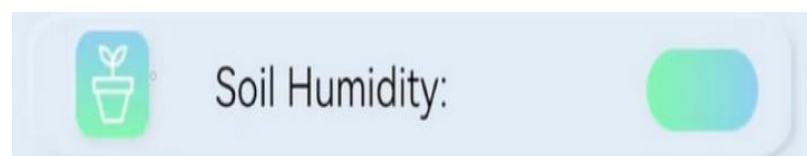


Рис. 3.32. Відображає значення вологості ґрунту на фермі в режимі реального часу

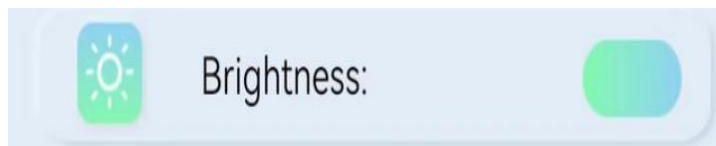


Рис. 3.33. Відображає значення сонячної яскравості ферми в реальному часі

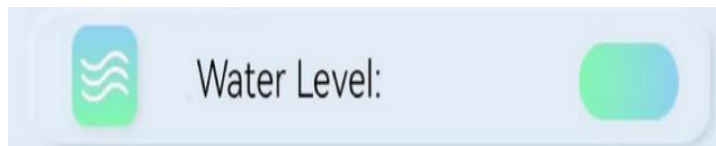


Рис. 3.34. Відображає рівень води на фермі в реальному часі

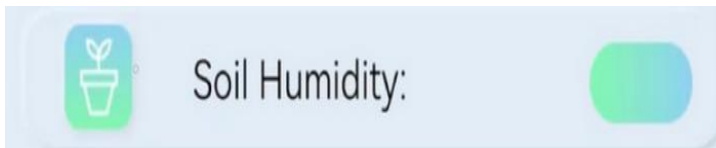


Рис. 3.35. Відображає значення вологості ґрунту на фермі в режимі реального часу



Рис. 3.36. Світлодіод керування

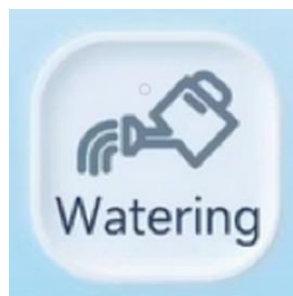


Рис. 3.37. Контролюйте зрошення за допомогою водяного насоса

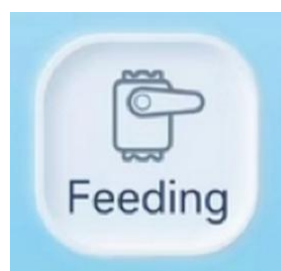


Рис. 3.38. Керуйте сервоприводом для відкриття або закриття коробки для годування

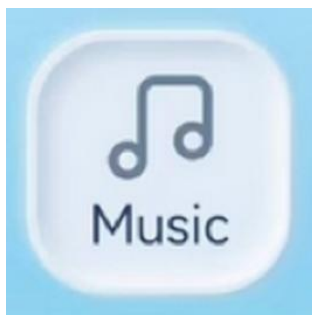


Рис. 3.39. Контролюй звук дзвінка

Висновки до розділу

У третьому розділі виконано практичну реалізацію, налаштування та комплексне тестування лабораторного стенду IoT «Розумна ферма» на базі сучасного та енергоефективного мікроконтролера KEYESTUDIO ESP32 PLUS. У ході розробки було успішно реалізовано та перевірено роботу окремих виконавчих підсистем, модулів автоматизації, а також засобів дистанційного моніторингу та керування.

За результатами виконання практичної частини роботи сформовано такі основні висновки:

1. Модуль системи освітлення та оптимізації енергоспоживання:

Досліджено базові принципи цифрового та аналогового керування світлодіодним модулем. Завдяки застосуванню широтно-імпульсної модуляції (ШИМ) за допомогою вбудованого периферійного контролера LEDC, реалізовано функцію плавного регулювання яскравості світловипромінювальних елементів. Розроблено та протестовано алгоритми антидеребзу («антишейк») для механічних кнопок із функцією фіксації стану (автоматичного блокування), що підвищило стабільність обробки сигналів ручного керування освітленням у реальних умовах.

2. Автоматизована система керування світлом:

На основі інтеграції фоторезистивного сенсора (аналоговий вхід ІО34) та 12-бітного аналого-цифрового перетворювача (АЦП) плати ESP32 з діапазоном значень від 0 до 4095, створено інтелектуальний контур керування освітленням. Розроблений програмний алгоритм забезпечує

автоматичне увімкнення освітлювального модуля у разі зниження інтенсивності природного світла нижче встановленого порогу (800 одиниць АЦП), що оптимізує енерговитрати фермерського господарства.

3. Модуль підсистеми безпеки та сигналізації:

Реалізовано систему захисту периметра ферми від небажаного вторгнення. На основі обробки цифрових сигналів з піроелектричного (PIR) датчика руху (IO23), пасивного п'єзозвукового випромінювача (зумера) та світлодіода створено ефективний модуль тривожного сповіщення. Застосування функцій `ledcAttachChannel()` та `ledcWriteTone()` дозволило гнучко керувати частотою звукового сигналу та тривалістю сирени, створюючи виразний звуковий малюнок під час фіксації руху.

4. Інтелектуальна система годування тварин:

Розроблено та впроваджено механізований вузол автоматичного подавання корму на основі взаємодії ультразвукового далекоміра HC-SR04 (Trig — IO5, Echo — IO18) та мікросервоприводу (IO26). Написано алгоритм розрахунку відстані на основі часу проходження ультразвукової хвилі. Забезпечено точне позиціонування заслінки кабіни годування (поворот сервоприводу на кут 80° – 90° для відкриття дверцят) виключно у визначеному безпечному діапазоні наближення птахів чи тварин — від 2 до 7 см.

5. Система автоматичного зрошення та контролю життєдіяльності рослин:

Спроектовано замкнутий контур поливу, який базується на одночасному моніторингу вологості ґрунту та критичного рівня води в резервуарі за допомогою аналогових сенсорів. Використання релейного модуля дозволило платі ESP32 безпечно комутувати живлення водяного мікронасоса. Програмно реалізовано багаторівневу логіку захисту: система автоматично активує полив тривалістю 400 мс при сухості

грунту (значення < 500), блокує роботу насоса у разі критичного спустошення резервуара для запобігання його «сухому ходу» та вмикає звукове сповіщення, яке оператор може вимкнути апаратною кнопкою. Інформативність системи підвищено за рахунок інтеграції дисплея LCD 1602 (інтерфейс I2C).

6. Мережева інтеграція та IoT-сервіси керування:

Використовуючи вбудований Wi-Fi модуль мікроконтролера ESP32 та стандартну бібліотеку `<WiFi.h>` у режимі базової станції (STA), забезпечено підключення лабораторного стенду до локальної бездротової мережі на частоті 2.4 ГГц із динамічним отриманням IP-адреси.

- У межах веб-орієнтованого підходу за допомогою бібліотеки `<WebServer.h>` розгорнуто локальний HTTP-сервер. Створено адаптивну веб-сторінку, яка дозволяє через звичайний браузер будь-якого підключеного пристрою (ПК, смартфона) в реальному часі моніторити показники мікроклімату (температуру, вологість повітря та ґрунту, освітленість, рівень води) та відправляти керуючі літери-команди ("A", "B", "C", "D") для активації виконавчих пристроїв.
- У межах мобільного підходу налаштовано взаємодію стенду із кросплатформним APP-додатком через сокет-з'єднання за визначеною IP-адресою. Графічний інтерфейс програми забезпечує підвищений комфорт користувача завдяки візуальним слайдерам моніторингу та інтерактивним кнопкам безпосереднього дистанційного керування освітленням (LED), зрошенням (Watering), вентиляцією (Fan), годівницею (Feeding) та звуковими ефектами (Music).

Таким чином, розроблений та практично реалізований третій розділ підтвердив повну працездатність усіх елементів, високу швидкість реагування

сенсорів та надійність архітектури ESP32 як центрального ядра сучасної кіберфізичної системи класу "Smart Farm".

ВИСНОВКИ

У дипломному проєкті вирішено актуальне науково-практичне завдання з проєктування, реалізації та комплексного дослідження функціонування лабораторного стенду Інтернету речей (IoT) «Розумна ферма» на базі мікроконтролера ESP32. Результати виконаної роботи дозволяють зробити такі висновки:

1. На основі аналізу літературних джерел та сучасного ринку IoT-рішень підтверджено доцільність впровадження кіберфізичних систем в аграрний сектор та навчальний процес. Розглянуто існуючі комерційні та промислові рішення (Nest, Honeywell, John Deere, Schneider Electric EcoStruxure, Siemens MindSphere), а також навчальні набори, що дозволило визначити необхідну функціональну структуру для імітації реальних технологічних процесів ферми на лабораторному стенді.
2. Проведено порівняльний аналіз поширених апаратних платформ (Arduino, Raspberry Pi, ESP8266, ESP32). За критеріями обчислювальної потужності, енергоефективності, вартості та наявності вбудованих бездротових інтерфейсів обґрунтовано вибір двоядерного мікроконтролера *Xtensa LX6* (плата розробника *KEYESTUDIO ESP32 PLUS*). Наявність інтегрованих модулів Wi-Fi/Bluetooth, 12-бітного АЦП та підтримки апаратних інтерфейсів (I2C, SPI, UART) робить його оптимальним вибором для побудови гетерогенних IoT-систем.
3. Зпроектовано та детально описано специфікацію периферійного обладнання стенду. Сформовано електричні схеми взаємодії та таблиці комутації для кнопочового модуля (EASY Plug RJ11), світлодіодного індикатора, пасивного п'єзоелектричного зумера, мікросервоприводу Tower Pro SG90, ультразвукового далекоміра HC-SR04, інфрачервоного датчика руху PIR, електромеханічного реле SRD-5VDC-SL-C та рідкокристалічного дисплея LCD 1602.

4. Обґрунтовано вибір інтегрованого середовища розробки Arduino IDE, яке завдяки використанню спрощеної мови C++, наявності розвиненої екосистеми відкритих бібліотек та вбудованих засобів налагодження в реальному часі (Serial Monitor) забезпечило швидке і надійне кросплатформове прототипування прошивки мікроконтролера.
5. Практично реалізовано та протестовано локальні алгоритми автоматизації вбудованих підсистем:
 - *Контур освітлення* отримав функцію плавного ШІМ-регулювання яскравості за допомогою периферійного контролера LEDC та програмний алгоритм захисту від деренчання механічних контактів кнопки («антишейк»).
 - *Система керування світлом* на основі фоторезистора автоматично активує штучне доосвітлення у разі падіння показників АЦП нижче порогу в 800 одиниць.
 - *Модуль безпеки* успішно фіксує порушення периметра за допомогою датчика PIR та запускає динамічну звукову сирену (через функції ledcWriteTone), дубльовану світловим миготінням.
 - *Інтелектуальна годівниця* забезпечує автоматизоване відкривання заслінки кабіни за допомогою сервоприводу SG90 лише при наближенні об'єкта (імітація тварини) до ультразвукового сенсора HC-SR04 в безпечному діапазоні від 2 до 7 см.
 - *Контур автоматичного зрошення* на основі зчитування показників вологості ґрунту та рівня води в резервуарі керує водяним мікронасосом через реле. Реалізовано логіку аварійного блокування двигуна при критичному спустошенні бака для унеможливлення «сухого ходу» з

одночасним виведенням телеметрії на локальний LCD-екран.

6. Успішно розв'язано завдання мережевої інтеграції стенду за допомогою вбудованого Wi-Fi модуля в режимі базової станції (STA). Розгорнутий на ESP32 локальний HTTP-веб-сервер (<WebServer.h>) надає адаптивну HTML-сторінку, яка відображає поточні дані датчиків (температуру, вологість, освітленість тощо) та дозволяє здійснювати релейне і серво-керування через звичайний браузер.
7. Розроблено та протестовано програмне забезпечення взаємодії стенду із кросплатформним мобільним APP-додатком через сокет-з'єднання. Dodatok забезпечує високий рівень ергономіки, виводячи графічні слайдери моніторингу телеметрії в реальному часі та інтерактивні кнопки безпосереднього дистанційного керування поливом, вентиляцією, світлом, годівницею та звуковою сигналізацією.

Створений лабораторний стенд продемонстрував повну працездатність, високу швидкість реакції на зміну параметрів навколишнього середовища та стабільність мережевого з'єднання, що повністю підтверджує надійність обраної архітектури та успішне досягнення мети дипломного проєкту.

Розроблений стенд може бути надалі модернізований шляхом підключення додаткових датчиків, інтеграції з хмарними платформами та впровадження алгоритмів інтелектуального аналізу даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Антонова М. В., Сидоренко А. О. Застосування технологій Інтернету речей в автоматизації процесів точного землеробства. *Технічна інженерія*. 2024. № 2 (94). С. 45–52.
2. Василенко О. М. Використання хмарних платформ для візуалізації телеметричних даних у реальному часі. *Вісник ЖДТУ. Серія: Технічні науки*. 2023. № 1 (91). С. 141–148.
3. Дерев'янка В. В., Савченко О. А. Використання IoT-технологій у сільському господарстві. *Технічні науки : зб. наук. пр.* 2021. № 3. С. 45–50.
4. Джевага Г., Яковлев К. Створення віддаленої лабораторії для виконання практичних робіт з робототехніки, автоматизації виробництва та IoT. *Вісник Чернігівського національного педагогічного університету. Серія: Педагогічні науки*. 2025. Вип. 33 (189). С. 33–40. DOI: 10.58407/visnik.2533052.
5. Довженко Н., Мазур Н., Костюк Ю., Рзаєва С. Інтеграція IoT та штучного інтелекту в інтелектуальні транспортні системи. *Кібербезпека: освіта, наука, техніка*. 2024. № 2 (26). С. 430–444. DOI: 10.28925/2663-4023.2024.26.708.
6. Дурняк Б. В., Шевчук В. Б. Основи алгоритмізації та програмування мікроконтролерних систем автоматизації. Львів : УАД, 2022. 188 с.
7. Інтернет речей (IoT) : підручник / В. М. Оліферчук та ін. ; за ред. В. М. Оліферчука. Київ : Сталь, 2022. 340 с.
8. Кліматичний моніторинг у сучасних теплицях: апаратні та програмні рішення / Д. І. Шевченко та ін. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. 2023. Вип. 50. С. 112–119.
9. Козак Р. М. Проектування інтелектуальних систем керування мікрокліматом для закритих екосистем. *Автоматика. Автоматизація. Електротехнічні комплекси та системи*. 2023. № 2. С. 63–70.

10. Кравченко Ю. В., Бойко О. В. Бездротові сенсорні мережі на основі протоколів Wi-Fi та BLE: архітектура та безпека. *Радіoeлектроніка, інформатика, управління*. 2024. № 3. С. 89–97.
11. Микитин Г. В. Порівняльний аналіз енергоефективності мікроконтролерів ESP8266 та ESP32 в автономних системах IoT. *Сучасні інформаційні технології та системи керування*. 2025. Т. 12, № 1. С. 28–35.
12. Посашков О., Цимбал О. Розроблення архітектури системи віддаленого доступу до навчального лабораторного обладнання з використанням автоматизованих рішень. *Сучасний стан наукових досліджень та технологій в промисловості*. 2024. № 3 (29). С. 64–75. DOI: 10.30837/2522-9818.2024.3.064.
13. Проектування систем Інтернету речей на базі мікроконтролерів : навч. посіб. / О. П. Ковалюк, І. С. Петренко. Львів : Новий Світ-2000, 2023. 215 с.
14. Ткаченко А. Ю., Іваненко П. В. Автоматизація аграрних процесів на базі мікроконтролерів. *Вісник інженерних технологій*. 2022. Т. 32, № 2. С. 85–90.
15. Ткаченко С. І. Методи калібрування ємнісних та резистивних датчиків вологості ґрунту в автоматизованих системах поливу. *Вимірювальна техніка та метрологія*. 2024. Т. 85, вип. 4. С. 54–61.
16. Al-Fuqaha A., Guizani M., Mohammadi M., Aledhari M., Ayyash M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*. 2015. Vol. 17, No. 4. P. 2347–2376.
17. Arduino IDE: інтегроване середовище розробки : вебсайт. URL: <https://www.arduino.cc/en/software> (дата звернення: 01.06.2026).
18. Bahatskyi O., Bahatskyi V., Sokolov V. Smart Home Subsystem for Calculating the Quality of Public Utilities. *Workshop on Cybersecurity*

- Providing in Information and Telecommunication Systems*. 2023. Vol. 3421. P. 168–173.
19. Blynk IoT Platform Documentation. URL: <https://docs.blynk.io/> (дата звернення: 20.04.2026).
 20. DHT11/DHT22 Humidity & Temperature Sensors Guide. *Adafruit Learning System*. URL: <https://learn.adafruit.com/dht> (дата звернення: 05.02.2026).
 21. ESP32 Technical Reference Manual. *Espressif Systems*. URL: <https://www.espressif.com/en/support/documents/technical-documents> (дата звернення: 01.06.2026).
 22. Espressif IoT Development Framework (ESP-IDF): official documentation. URL: <https://docs.espressif.com/projects/esp-idf/> (дата звернення: 01.06.2026).
 23. Huang C., Zhang J. Application of IoT Technology in Precision Agriculture. *Proceedings of the International Conference on IoT Applications*. 2019. P. 77–82.
 24. Introduction to ESP32 Platform. *Espressif Systems*. URL: <https://www.espressif.com/en/products/socs/esp32> (дата звернення: 12.03.2026).
 25. Jacko P. et al. Remote IoT Education Laboratory for Microcontrollers Based on the STM32 Chips. *Sensors*. 2022. Vol. 22, No. 4. P. 1440. DOI: 10.3390/s22041440.
 26. Kalashnikov A. et al. Remote Laboratory: Using Internet-of-Things (IoT) for E-learning. *Springer*. 2017. P. 43–46. URL: <http://aist.sumdu.edu.ua/main/> (дата звернення: 01.06.2026).
 27. Keyestudio ESP32 Smart Farm Starter Kit : technical documentation. URL: <https://wiki.keyestudio.com> (дата звернення: 01.06.2026).
 28. MQTT Essentials: A Guide to IoT Communication Protocols / HiveMQ Team. URL: <https://www.hivemq.com/mqtt-essentials/> (дата звернення: 18.04.2026).

29. Rejón C., Martín S., Robles-Gómez A. Easy Development of Industry 4.0 Remote Labs. *Electronics*. 2024. Vol. 13, No. 8. P. 1508. DOI: 10.3390/electronics13081508.
30. Sánchez Sánchez P. M. et al. LwHBench: A Low-Level Hardware Component Benchmark and Dataset for Single Board Computers. *Internet of Things*. 2023. DOI: 10.1016/j.iot.2023.100764.
31. Santos R., Santos S. Developing IoT Projects with ESP32: Cloud Integration and Data Visualization. 2nd ed. New York : O'Reilly Media, 2023. 310 p.
32. Wokwi IoT Simulator: online simulator for IoT projects. URL: <https://wokwi.com/> (дата звернення: 01.06.2026).
33. Yadav A., Singh R. Smart Agriculture Systems Using IoT and Cloud Computing. *International Journal of Advanced Research in Computer Science*. 2020. Vol. 11, No. 6. P. 123–130.
34. Zhebka V. et al. Methodology for Predicting Failures in a Smart Home Based on Machine Learning Methods. *Workshop on Cybersecurity Providing in Information and Telecommunication Systems (CPITS)*. 2024. Vol. 3654. P. 322–332.
35. Zhebka V. et al. Methods for Predicting Failures in a Smart Home. *Digital Economy Concepts and Technologies Workshop*. 2024. Vol. 3665. P. 70–78.

ДОДАТОК А

Лістинг коду

```
#include <Arduino.h>
#include <WiFi.h>
#include <WebServer.h>
#include <LiquidCrystal_I2C.h>
#include <dht11.h>
#include <ESP32Servo.h>
// Pin Definitions
#define DHT11PIN    17 // Temperature and humidity sensor pin
#define LEDPIN      27 // LED pin
#define SERVOPIN    26 // Servo pin
#define FANPIN1     19 // Fan IN+ pin
#define FANPIN2     18 // Fan IN- pin
#define STEAMPIN    35 // Steam sensor pin
#define LIGHTPIN    34 // Photoresistor pin
#define SOILHUMIDITYPIN 32 // Soil humidity sensor pin
#define WATERLEVELPIN 33 // Water level sensor pin
#define RELAYPIN    25 // Relay pin
// Initialize sensors and components
dht11 DHT11;
LiquidCrystal_I2C lcd(0x27, 16, 2);
Servo myservo; // Servo object to control the servo
// WiFi credentials
const char *SSID = "your_SSID";
const char *PASS = "your_PASSWORD";
// Create WebServer object
WebServer server(80);
// Variables for controlling states
static int A = 0;
static int B = 0;
static int C = 0;
// HTML Web page content
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML>
<html>
<title>TEST HTML ESP32</title>
<head>
  <meta charset="utf-8">
  <style>
    html, body {
      margin: 0;
      width: 100%;
      height: 100%;
```



```

display: flex;
justify-content: center;
align-items: center;
flex-direction: column;
background-color: #f0f0f0;
}
/* The main button container */
.btn {
display: flex;
justify-content: center; /* Center the buttons */
gap: 10px; /* Add space between buttons */
width: 320px; /* Set width to ensure buttons are tightly packed */
flex-wrap: wrap; /* Allow buttons to wrap to new lines if needed */
margin-bottom: 20px; /* Space between buttons and data display */
}
/* Button style */
.btn button {
width: 70px; /* Set width for buttons */
height: 70px; /* Set height for buttons */
border: none;
font-size: 16px;
color: #fff;
background-color: #89e689;
cursor: pointer;
}
.btn button:active {
top: 2px;
}
/* The data display area */
#dht {
text-align: center; /* Center the text */
width: 320px; /* Same width as the button container */
color: #fff;
background-color: #47a047;
font-size: 18px; /* Adjust font size for readability */
padding: 10px;
border-radius: 10px; /* Rounded corners */
box-sizing: border-box;
margin-bottom: 10px; /* Add space between the data display and buttons */
}
</style>
</head>
<body>
<!-- Display area for sensor data -->
<div id="dht"></div>

```

```

<!-- Button row -->
<div class="btn">
  <button id="btn-led" onclick="setLED()">LED</button>
  <button id="btn-fan" onclick="setFan()">Fan</button>
  <button id="btn-feeding" onclick="setFeeding()">Feeding</button>
  <button id="btn-watering" onclick="setWatering()">Watering</button>
</div>
<script>
function setLED() {
  var payload = "A";
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/set?value=" + payload, true);
  xhr.send();
}
function setFan() {
  var payload = "B";
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/set?value=" + payload, true);
  xhr.send();
}
function setFeeding() {
  var payload = "C";
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/set?value=" + payload, true);
  xhr.send();
}
function setWatering() {
  var payload = "D";
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/set?value=" + payload, true);
  xhr.send();
}
setInterval(function () {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("dht").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/dht", true);
  xhttp.send();
}, 1000)
</script>
</body>

```

```

</html>
)rawliteral";
// Function to merge sensor data into HTML format
String Merge_Data(void) {
    String dataBuffer;
    String Humidity;
    String Temperature;
    String Steam;
    String Light;
    String SoilHumidity;
    String WaterLevel;
    // Read DHT11 sensor
    int chk = DHT11.read(DHT11PIN);
    // Read other sensors
    Steam = String(analogRead(STEAMPIN) / 4095.0 * 100);
    Light = String(analogRead(LIGHTPIN));
    int shvalue = analogRead(SOILHUMIDITYPIN) / 4095.0 * 100 * 2.3;
    shvalue = shvalue > 100 ? 100 : shvalue;
    SoilHumidity = String(shvalue);
    int wlvalue = analogRead(WATERLEVELPIN) / 4095.0 * 100 * 2.5;
    wlvalue = wlvalue > 100 ? 100 : wlvalue;
    WaterLevel = String(wlvalue);
    Temperature = String(DHT11.temperature);
    Humidity = String(DHT11.humidity);
    // Construct HTML content
    dataBuffer += "<p>";
    dataBuffer += "<h1>Sensor Data</h1>";
    dataBuffer += "<b>Temperature:</b><b>" + Temperature +
    "</b><b>°C</b><br/>";
    dataBuffer += "<b>Humidity:</b><b>" + Humidity + "</b><b>%rh</b><br/>";
    dataBuffer += "<b>WaterLevel:</b><b>" + WaterLevel +
    "</b><b>%</b><br/>";
    dataBuffer += "<b>Steam:</b><b>" + Steam + "</b><b>%</b><br/>";
    dataBuffer += "<b>Light:</b><b>" + Light + "</b><br/>";
    dataBuffer += "<b>SoilHumidity:</b><b>" + SoilHumidity +
    "</b><b>%</b><br/>";
    dataBuffer += "</p>";
    return dataBuffer;
}

// Configure actions based on received HTTP requests
void Config_Callback() {
    if (server.hasArg("value")) {
        String HTTP_Payload = server.arg("value");
        Serial.printf("[%lu]%s\r\n", millis(), HTTP_Payload.c_str());
    }
}

```

```

if (HTTP_Payload == "A") {
  if (A) {
    digitalWrite(LEDPIN, LOW);
    A = 0;
  } else {
    digitalWrite(LEDPIN, HIGH);
    A = 1;
  }
}
if (HTTP_Payload == "B") {
  if (B) {
    digitalWrite(FANPIN1, LOW);
    digitalWrite(FANPIN2, LOW);
    B = 0;
  } else {
    delay(500);
    digitalWrite(FANPIN1, HIGH);
    digitalWrite(FANPIN2, LOW);
    delay(500);
    B = 1;
  }
}
if (HTTP_Payload == "C") {
  if (C) {
    myservo.write(80);
    delay(500);
    C = 0;
  } else {
    C = 1;
    myservo.write(180);
    delay(500);
  }
}
if (HTTP_Payload == "D") {
  digitalWrite(RELAYPIN, HIGH);
  delay(400);
  digitalWrite(RELAYPIN, LOW);
  delay(650);
}
}
server.send(200, "text/plain", "OK");
}
// Handle invalid URL access
void notFound() {
  server.send(404, "text/plain", "Not found");
}

```

```

}
void setup() {
  Serial.begin(9600);

  // Connect to WiFi
  WiFi.begin(SSID, PASS);
  while (!WiFi.isConnected()) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  // Set up pins
  pinMode(LEDPIN, OUTPUT);
  pinMode(STEAMPIN, INPUT);
  pinMode(LIGHTPIN, INPUT);
  pinMode(SOILHUMIDITYPIN, INPUT);
  pinMode(WATERLEVELPIN, INPUT);
  pinMode(RELAYPIN, OUTPUT);
  pinMode(FANPIN1, OUTPUT);
  pinMode(FANPIN2, OUTPUT);
  delay(1000);
  // Attach servo to the pin
  myservo.attach(SERVOPIN);
  myservo.write(180);
  // Initialize LCD
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("IP:");
  lcd.setCursor(0, 1);
  lcd.print(WiFi.localIP());
  // Set up server routes
  server.on("/", HTTP_GET, []() {
    server.send(200, "text/html", index_html);
  });
  server.on("/dht", HTTP_GET, []() {
    server.send(200, "text/plain", Merge_Data().c_str());
  });
  server.on("/set", HTTP_GET, Config_Callback);
  server.onNotFound(notFound);
  // Start the server
  server.begin();
}

```

```
}  
void loop() {  
  server.handleClient();  
}
```

ДОДАТОК Б

```
#include <Arduino.h>
#ifdef ESP32
#include <WiFi.h>
#elif defined(ESP8266)
#include <ESP8266WiFi.h>
#endif
#include <dht11.h>
#include <ESP32Servo.h>
#include <LiquidCrystal_I2C.h>
//To be displayed
#define DHT11PIN 17      //Temperature and humidity sensor pin
#define RAINWATERPIN 35  //Steam sensor pin
#define LIGHTPIN 34      //Photoresistor pin
#define WATERLEVELPIN 33 //Water level sensor pin
#define SOILHUMIDITYPIN 32 //Soil humidity sensor pin
//To be controlled
#define LEDPIN 27        //LED pin
#define RELAYPIN 25      //Relay pin (to control water pump)
#define SERVOPIN 26      //Servo pin
#define FANPIN1 19       //Fan IN+ pin
#define FANPIN2 18       //Fan IN- pin
#define BUZZERPIN 16     //Buzzer pin
const char* ssid = "your_SSID";
const char* pwd = "your_PASSWORD";
//Initialize LCD1602, 0x27 is I2C address
LiquidCrystal_I2C lcd(0x27, 16, 2);
WiFiServer server(80); //Initialize wifi server
dht11 DHT11;           //Initialize temperature and humidity sensor
Servo myservo;          // create servo object to control a servo
                        // 16 servo objects can be created on the ESP32
//Define variable as detected values
String request;
String dataBuffer;
int Temperature; //Temperature
int Humidity;    //Humidity
int SoilHumidity; //Soil humidity
int Light;       //Brightness
int WaterLevel;  //Water level
int Rainwater;   //Rainfall
void setup() {
  Serial.begin(9600);
  //Connect to wifi
  WiFi.begin(ssid, pwd);
```

```

//Determine whether connected
Serial.println("Connecting to WiFi...");
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
}
delay(1000);
//Serial monitor prints wifi name and IP address
Serial.println("Connected to WiFi");
Serial.print("WiFi NAME:");
Serial.println(ssid);
Serial.print("IP:");
Serial.println(WiFi.localIP());
//Initialize LCD
lcd.init();
// Turn the (optional) backlight off/on
lcd.backlight();
//lcd.noBacklight();
lcd.clear();
//Set the position of cursor
lcd.setCursor(0, 0);
//LCD prints
lcd.print("IP:");
//Set the position of cursor
lcd.setCursor(0, 1);
//LCD prints
lcd.print(WiFi.localIP());
//set pins mode
pinMode(LEDPIN, OUTPUT);
pinMode(RAINWATERPIN, INPUT);
pinMode(LIGHTPIN, INPUT);
pinMode(SOILHUMIDITYPIN, INPUT);
pinMode(WATERLEVELPIN, INPUT);
pinMode(RELAYPIN, OUTPUT);
pinMode(FANPIN1, OUTPUT);
pinMode(FANPIN2, OUTPUT);
pinMode(BUZZERPIN, OUTPUT);
delay(1000);
// attaches the servo on pin 26 to the servo object
myservo.attach(SERVOPIN);
myservo.write(160);
//Start server
server.begin();
// Configure LEDC channel
ledcAttachChannel(BUZZERPIN, 1000, 8, 4);

```



```

}
void loop() {
  //Check whether a client is connected to the web server
  //When the client is connected to server, "server.available()" returns a WiFiClient
  object for communication at client-side.
  WiFiClient client = server.available();
  if (client) {
    Serial.println("New client connected");
    while (client.connected()) {
      //Determine whether the server sends data
      if (client.available()) {
        request = client.readStringUntil('s');
        Serial.print("Received message: ");
        Serial.println(request);
      }
      //Acquire all sensor data
      getSensorsData();
      //put all data into "dataBuffer"
      dataBuffer = "";
      dataBuffer += String(Temperature, HEX);
      dataBuffer += String(Humidity, HEX);
      dataBuffer += dataHandle(SoilHumidity);
      dataBuffer += dataHandle(Light);
      dataBuffer += dataHandle(WaterLevel);
      dataBuffer += dataHandle(Rainwater);
      //Send data to server, transmit to APP
      client.print(dataBuffer);
      delay(500);
      //LED
      if (request == "a") {
        digitalWrite(LEDPIN, HIGH);
      } else if (request == "A") {
        digitalWrite(LEDPIN, LOW);
      }
      //Irrigation
      else if (request == "b") {
        digitalWrite(RELAYPIN, HIGH);
        delay(400); //Irrigation delay
        digitalWrite(RELAYPIN, LOW);
        delay(650);
      }
      //Fan
      else if (request == "c") {
        delay(800);
        digitalWrite(FANPIN1, HIGH);
      }
    }
  }
}

```

```

        digitalWrite(FANPIN2, LOW);
        delay(200);
    } else if (request == "C") {
        digitalWrite(FANPIN1, LOW);
        digitalWrite(FANPIN2, LOW);
    }
    //Feeding box
    else if (request == "d") {
        //Servo rotates to 80°, open feeding box
        myservo.write(80);
        delay(500);
    } else if (request == "D") {
        //Servo rotates to 160°, close feeding box
        myservo.write(160);
    }
    //Buzzer
    else if (request == "e") {
        ledcWriteTone(BUZZERPIN, 262);
        delay(800);
        ledcWriteTone(BUZZERPIN, 0);
        delay(100);
    }
    request = "";
}
Serial.println("Client disconnected");
}
}

void getSensorsData() {
    //Acquire data
    int chk = DHT11.read(DHT11PIN);
    //Steam sensor
    Rainwater = analogRead(RAINWATERPIN);
    //Photoresistor
    Light = analogRead(LIGHTPIN);
    //Soil humidity sensor
    SoilHumidity = analogRead(SOILHUMIDITYPIN) * 1.8;
    //Water level sensor
    WaterLevel = analogRead(WATERLEVELPIN) * 1.8;
    //Temperature
    Temperature = DHT11.temperature;
    //Humidity
    Humidity = DHT11.humidity;
}
//Convert data into percentage
String dataHandle(int data) {

```

```

// Convert analog values into percentage
int percentage = (data / 4095.0) * 100;
// If the converted percentage is greater than 100, output 100.
percentage = percentage > 100 ? 100 : percentage;
// Six characters store hexadecimal strings, one character is as terminators
char hexString[3];
// Convert hexadecimal values to 6-digit hexadecimal strings, add leading zeros: 0
is 00, 1 is 01...
sprintf(hexString, "%02X", percentage);
return hexString;
}

```