ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО - КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: « Алгоритм управління роботизованими системами на мові JavaScript »

на здобуття освітнього ступені магістра

зі спеціальності 126 Інформаційні системи та технології

освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ _____*Влад Фадієнко*_____

(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. ІСДМ-61

_____Фадієнко Владислав_____

Ім'я, ПРІЗВИЩЕ

Керівник: _____Валентина Данильченко доктор філософії____

науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент: _____

науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційні системи та технології

Ступінь вищої освіти Магістр

Спеціальність 126 «Інформаційні системи та технології»

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою Каміла Сторчак
Ім'я, ПРІЗВИЩЕ
« » 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Фадієнко Владислав Сергійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: « Алгоритм управління роботизованими системами на мові JavaScript »
керівник кваліфікаційної роботи Валентина Данильченко доктор філософії,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024р. №320,
2. Строк подання кваліфікаційної роботи «29» грудня 2024р.
3. Вихідні дані до кваліфікаційної роботи: Визначити основні вимоги та завдання, які повинні бути вирішені під час виконання дипломної роботи.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Розробка теоретичної частини
2. Вивчення функціоналу вимоги до системи

5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «01» вересня 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	10.10-12.10.2024	Виконано
2	Обробка матеріалу	12.10-15.10.2024	Виконано
3	Розробка теоретичної частини	15.10-18.10.2024	Виконано
4	Вивчення функціональних вимог до системи:	18.10-22.10.2024	Виконано
5	Аналіз архітектури:клієнт-серверна взаємодія, основні компоненти	22.10-26.10.2024	Виконано
6	Висновки за результатами аналізу	30.10-31.10.2024	Виконано
7	Розробка презентації	31.10-01.11.2024	Виконано
8	Передзахист	09.12-11.12.2024	
9	Здача в деканат	20.12-29.12.2024	

Здобувач вищої освіти Владислав Фадієнко

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи Валентина Данильченко

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістр:

Мета роботи – розробити алгоритм управління роботизованими системами на основі мови програмування JavaScript.

Об’єкт дослідження – процес взаємодії програмного забезпечення з апаратними компонентами роботизованих систем, зокрема на базі мікроконтролерів Arduino.

Предмет дослідження – методи та інструменти для створення алгоритмів керування роботизованими системами з використанням бібліотек JavaScript, таких як Johnny-Five, Serialport і Babbler_h.

Короткий зміст роботи: У роботі досліджено розробку алгоритмів для управління роботизованими системами на платформі Arduino з використанням сучасних бібліотек JavaScript.

Проаналізовано архітектуру та функціональні можливості бібліотек Johnny-Five, Serialport і Babbler_h, їх переваги та обмеження. Продемонстровано алгоритми, що реалізують керування різними компонентами роботизованих систем, включаючи датчики, світлодіоди та серводвигуни. Розглянуто особливості інтеграції JavaScript з апаратним забезпеченням через протоколи, зокрема Firmata, та організацію передачі даних у форматі JSON. Увагу приділено оптимізації роботи пристроїв та забезпеченню стабільності зв’язку між програмним і апаратним компонентами.

КЛЮЧОВІ СЛОВА: РОБОТИЗОВАНІ СИСТЕМИ, JAVASCRIPT, ARDUINO, БІБЛІОТЕКА JOHNNY-FIVE, SERIALPORT, BABBLER_H, ПРОТОКОЛ FIRMATA, УПРАВЛІННЯ ДАТЧИКАМИ, СЕРВОДВИГУНИ, JSON, ІНТЕГРАЦІЯ АПАРАТНИХ КОМПОНЕНТІВ, АСИНХРОННА РОБОТА, СТАБІЛЬНІСТЬ З’ЄДНАННЯ, ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ, АЛГОРИТМИ КЕРУВАННЯ, ОБРОБКА ДАНИХ, ІНТЕРАКТИВНІ ПРОЄКТИ, ЕФЕКТИВНІСТЬ РОЗРОБКИ.

ABSTRACT

Text part of the qualification work for obtaining the degree of Master:

The purpose of the work is to develop an algorithm for controlling robotic systems based on the JavaScript programming language.

The object of the study is the process of interaction of software with hardware components of robotic systems, in particular based on Arduino microcontrollers.

The subject of the study is methods and tools for creating algorithms for controlling robotic systems using JavaScript libraries, such as Johnny-Five, Serialport and Babbler_h.

Summary of the work: The work investigates the development of algorithms for controlling robotic systems on the Arduino platform using modern JavaScript libraries.

The architecture and functionality of the Johnny-Five, Serialport and Babbler_h libraries, their advantages and limitations, are analyzed. Algorithms that implement the control of various components of robotic systems, including sensors, LEDs and servo motors, are demonstrated. The features of JavaScript integration with hardware via protocols, in particular Firmata, and the organization of data transfer in JSON format are considered. Attention is paid to optimizing the operation of devices and ensuring the stability of communication between software and hardware components.

KEYWORDS: ROBOTIC SYSTEMS, JAVASCRIPT, ARDUINO, JOHNNY-FIVE LIBRARY, SERIALPORT, BABBLER_H, FIRMATA PROTOCOL, SENSOR CONTROL, SERVO MOTORS, JSON, HARDWARE COMPONENT INTEGRATION, ASYNCHRONOUS OPERATION, CONNECTION STABILITY, MICROCONTROLLER PROGRAMMING, CONTROL ALGORITHMS, DATA PROCESSING, INTERACTIVE PROJECTS, DEVELOPMENT EFFICIENCY.

Вступ

У сучасному світі робототехніка та автоматизація стрімко розвиваються, проникаючи в різні сфери людської діяльності, такі як промисловість, медицина, сільське господарство та побут. З огляду на це, зростає потреба в створенні зручних, доступних і гнучких інструментів, які дозволяють ефективно розробляти електронні пристрої та системи. Одним із таких інструментів є Arduino — апаратно-програмна платформа, що здобула популярність завдяки простоті використання, доступній ціні та широкій спільноті розробників.

Arduino пропонує модульний підхід до створення електронних пристроїв, що дозволяє легко адаптувати платформи до різних завдань. Завдяки підтримці численних бібліотек і компонентів, ця платформа є ідеальним вибором як для початківців, так і для досвідчених розробників.

Поряд із цим, Node.js, як асинхронний серверний фреймворк, відкриває широкі можливості для програмування та взаємодії з апаратними засобами. Node.js надає розробникам інструменти для створення високопродуктивних додатків, які дозволяють взаємодіяти з мікроконтролерами, такими як Arduino. Завдяки численним бібліотекам, зокрема Babbler_h, Johnny-Five та іншим, розробники можуть реалізовувати управління робототехнічними пристроями, обробляти дані з датчиків, контролювати виконавчі механізми тощо.

Ця робота спрямована на аналіз використання платформи Node.js для розробки робототехнічних рішень на основі Arduino. У ході дослідження:

Розглянуто архітектуру та принципи роботи мікроконтролера Arduino, його основні характеристики та особливості використання.
Проведено аналіз бібліотек Node.js, що дозволяють управляти апаратними компонентами.
Наведено приклади реалізації практичних завдань, таких як керування світлодіодами, обробка команд та автоматичне перепідключення пристроїв.

Мета роботи полягає в демонстрації можливостей інтеграції Arduino з Node.js для створення функціональних і надійних робототехнічних систем. Практичні приклади та аналіз можливостей допоможуть визначити найефективніші підходи до реалізації проєктів, що використовують ці інструменти, а також нададуть цінні рекомендації для подальших розробок.

Це дослідження стане корисним для студентів, розробників, інженерів та ентузіастів, які цікавляться створенням сучасних електронних пристроїв і хочуть освоїти інтеграцію апаратних та програмних платформ.

1. АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ПРОГРАМУВАННЯ РОБОТА ARDUINO НА NODEJS

1.1 Теоретичні відомості про NodeJs

Node.js — це інноваційна програмна платформа, яка забезпечує виконання серверного коду на мові JavaScript, використовуючи високопродуктивний двигун Chrome V8. Її розробка орієнтована на створення додатків, які потребують високої продуктивності та ефективної роботи з ввід-вивід операціями. Node.js широко використовується для створення односторінкових додатків (SPA), стрімінгових платформ, чатів, API-серверів і навіть багатокористувацьких онлайн-ігор.

Основні переваги Node.js

Асинхронне програмування

Node.js реалізує неблокуючу модель виконання, що дозволяє операціям відбуватися паралельно без очікування завершення попередніх задач. Це забезпечує високу продуктивність навіть у випадках великої кількості одночасних запитів.

Однопоточність з подієвою моделлю

Node.js працює в одному потоці, що дозволяє уникнути накладних витрат на управління потоками. При цьому, завдяки подієвій моделі, платформа може обробляти тисячі одночасних з'єднань, забезпечуючи високу ефективність і масштабованість.

Ефективне використання ресурсів

Завдяки неблокуючим викликам, Node.js мінімізує споживання оперативної пам’яті, що особливо важливо для серверів з обмеженими ресурсами. Це також дозволяє зменшити затримки обробки запитів.

Цикл подій

Це ключовий механізм, що забезпечує обробку асинхронних викликів і подій. Він постійно перевіряє чергу задач і виконує їх у міру доступності, підтримуючи плавність і швидкість роботи програми.

Принцип роботи циклу подій

Очікування задач

Цикл подій знаходиться у стані очікування до тих пір, поки не виникне подія, як-от HTTP-запит, завершення таймера чи подія від користувача.

Обробка подій

При появі події двигун викликає відповідний обробник, який виконує поставлену задачу, наприклад, обробляє запит чи зчитує дані з файлової системи.

Повернення до очікування

Після завершення обробки задачі цикл подій повертається до стану очікування, готовий до нових викликів.

Застосування Node.js

Node.js ідеально підходить для:

Веб-серверів: Завдяки своїй продуктивності платформа використовується для створення високонавантажених серверних додатків.
Чатів і месенджерів: Асинхронна природа Node.js робить її ідеальною для програм реального часу.
Стрімінгових сервісів: Забезпечує високу ефективність при передачі великих обсягів даних.
Інтернету речей (IoT): Node.js часто використовується для обробки даних з численних сенсорів та взаємодії з пристроями.
API серверів: Полегшує створення масштабованих і легковагових API для роботи з базами даних та іншими службами.

Node.js поєднує простоту використання з високою продуктивністю, що робить її одним із найпопулярніших рішень для сучасних розробників. Завдяки активній спільноті та широкому спектру доступних бібліотек, платформа продовжує розширювати свої можливості та знаходить нові сфери застосування.

1.2 Робота двигуна JavaScript та черг задач

JavaScript є однопотоковою мовою програмування, але завдяки своїй унікальній архітектурі, заснованій на асинхронній моделі виконання, він забезпечує високу ефективність навіть у багатозадачному середовищі. Основним компонентом цієї архітектури є двигун JavaScript, який працює в поєднанні з чергами задач. Це дозволяє виконувати завдання паралельно, не блокуючи головний потік, і раціонально використовувати ресурси системи.

Двигун JavaScript: Основи роботи

Двигун JavaScript активується тільки за наявності задач для виконання. У перервах між задачами він перебуває в стані очікування, що зменшує навантаження на центральний процесор. Ця модель роботи є основою для створення інтерактивних веб-додатків і продуктивних серверних програм.
Задачі, що надходять до двигуна, організовуються у спеціальні черги, які поділяються на два основні типи:
Макрозадачі
Мікрозадачі

Макрозадачі: Основний потік виконання

Макрозадачі — це задачі, які виконуються у черзі за принципом FIFO (First In, First Out). До цього типу належать:

Таймери (setTimeout, setInterval).
Системні виклики (запити до файлової системи чи мережі).
Події введення/виведення (кліки, натискання клавіш).

Після початку виконання макрозадачі двигун JavaScript блокується на її завершенні, перш ніж перейти до наступної. Наприклад, подія натискання кнопки додається до черги макрозадач і виконується в порядку її надходження.

Мікрозадачі: Пріоритетні задачі

Мікрозадачі виконуються з вищим пріоритетом, ніж макрозадачі. Вони створюються під час виконання коду і часто асоціюються з операціями, що потребують негайного завершення, такими як:

Обробка обіцянок (Promise).
Виклик process.nextTick у Node.js.

Мікрозадачі виконуються одразу після завершення поточної макрозадачі, але до переходу до наступної. Це забезпечує швидку обробку асинхронних подій та покращує реактивність додатків.

Порядок виконання задач

Двигун JavaScript дотримується чіткої послідовності у виконанні задач:

Виконання поточної макрозадачі.
Обробка всіх накопичених мікрозадач.
Перехід до наступної макрозадачі.

Наприклад, якщо виклик Promise генерує мікрозадачу під час виконання макрозадачі, ця мікрозадача буде оброблена до виконання наступної макрозадачі.

Node.js: Реалізація моделі

Node.js, як серверна платформа, активно використовує цю модель:

Макрозадачі обробляють запити до файлової системи, баз даних або мережі.
Мікрозадачі, як-от process.nextTick, дозволяють виконувати критично важливі операції без затримок.

Якщо немає задач для обробки, Node.js переходить у режим очікування, зменшуючи споживання ресурсів.

Пріоритетність задач і взаємодія з подіями

Мікрозадачі завжди виконуються раніше, ніж макрозадачі. Наприклад, подія кліку (макрозадача) може викликати Promise (мікрозадача), який буде виконаний ще до обробки наступної події кліку.

Ця модель дозволяє створювати високо реактивні додатки. Наприклад:

Подія кліку додається до черги макрозадач.
В обробнику кліку генерується мікрозадача (Promise), яка виконується негайно.

Ефективність у реальних додатках

Цикл подій і черги задач дозволяють Node.js обробляти тисячі одночасних підключень, не блокуючи основний потік. Така архітектура забезпечує:

Плавність роботи навіть при великому навантаженні.
Мінімальну затримку при виконанні задач.
Гнучкість у побудові складних асинхронних сценаріїв.

Модель роботи двигуна JavaScript із чергами задач є ключовим елементом успіху сучасних веб-додатків і серверних платформ. Вона дозволяє створювати швидкі, масштабовані та ефективні рішення, які відповідають вимогам користувачів у реальному часі.

1.3 Приклад обробки запиту у Node.js

Розглянемо сценарій, коли веб-додаток повинен прочитати файл на сервері й повернути його вміст клієнту. Цей процес демонструє ефективність асинхронної архітектури Node.js:

Запит клієнта

Клієнт надсилає HTTP-запит на сервер із вимогою отримати вміст певного файлу. Це може бути запит до статичного файлу (наприклад, зображення або текстового документа) або до динамічного ресурсу.

Обробка запиту сервером

Після отримання запиту сервер викликає функцію зворотного виклику (callback), яка починає пошук і читання потрібного файлу у файловій системі.

Асинхронна операція

У цей момент операція читання файлу передається до системи вводу/виводу (I/O), яка працює асинхронно. Сервер не чекає завершення цієї операції, а продовжує приймати й обробляти інші запити від клієнтів.

Завершення операції

Як тільки файл буде знайдено та прочитано, система I/O сигналізує серверу про завершення операції. Сервер викликає функцію зворотного виклику, яка формує відповідь і надсилає її клієнту.

Відправка відповіді клієнту

У відповідь клієнт отримує вміст файлу або повідомлення про помилку, якщо файл недоступний.

Переваги такого підходу:

Масштабованість: Сервер може обробляти тисячі запитів одночасно без блокування.
Ефективність: Ресурси серверу використовуються оптимально, оскільки він не простоює під час виконання тривалих операцій вводу/виводу.
Швидкість: Користувачі отримують швидку відповідь, навіть якщо сервер працює з великою кількістю запитів.

Приклад коду на Node.js:

```
const http = require('http');
const fs = require('fs');

const server = http.createServer((req, res) => {
  if (req.url === '/file') {
    fs.readFile('example.txt', 'utf8', (err, data) => {
      if (err) {
        res.writeHead(500, { 'Content-Type': 'text/plain' });
        res.end('Error reading file');
      } else {
        res.writeHead(200, { 'Content-Type': 'text/plain' });
        res.end(data);
      }
    });
  } else {
    res.writeHead(404, { 'Content-Type': 'text/plain' });
    res.end('Not Found');
  }
});

server.listen(3000, () => {
  console.log('Server running on http://localhost:3000/');
});
```

Додаткові можливості Node.js

Node.js є універсальною платформою для створення серверних рішень. Вона пропонує широкий спектр функцій:

Робота з файлами

Створення, читання, редагування, перейменування та видалення файлів. Маніпуляції з файлами реалізуються за допомогою модуля fs (file system).

Динамічна генерація контенту

Node.js дозволяє створювати сторінки з динамічним контентом, який генерується під час виконання запиту. Це корисно для створення персоналізованих сторінок або API.

Обробка даних із форм

Сервер може приймати дані, введені користувачем через веб-форми, обробляти їх і зберігати у базах даних або файлах.

Реалізація WebSocket-з’єднань

Node.js підтримує двосторонню комунікацію у реальному часі між клієнтом і сервером, що використовується для чату, ігор та онлайн-стрімінгу.

Створення API

Node.js спрощує створення RESTful API для взаємодії з базами даних або зовнішніми системами.

Інтеграція із сторонніми сервісами

За допомогою модулів, таких як axios або request, Node.js дозволяє легко надсилати HTTP-запити до сторонніх API і обробляти отримані дані.

Переваги використання Node.js:

Висока продуктивність: Завдяки неблокуючій моделі вводу/виводу.
Легкість масштабування: Підходить для обробки великої кількості одночасних з’єднань.
Розвинена екосистема: Бібліотека npm пропонує понад мільйон модулів для розширення функціональності.
Гнучкість: Node.js використовується для створення як простих веб-серверів, так і складних хмарних платформ.

Node.js залишається популярною платформою для розробки сучасних веб-додатків, поєднуючи продуктивність і зручність використання.

1.4 Запуск сервера на Node.js

Щоб запустити сервер на Node.js, необхідно попередньо встановити цю платформу на ваш комп'ютер. Нижче наведено покрокову інструкцію:

Встановлення Node.js

1. Завантажте інсталяційний файл з офіційного сайту NodeJS.org.
2. Запустіть завантажений .exe файл після чого зв’явиться встановлювач NodeJS(Рис1).
3. Після завершення встановлення рекомендується перезавантажити комп'ютер.
4. Для перевірки правильності встановлення відкрийте командний рядок (*Command Prompt*) і введіть команду:
5. node -v

   У результаті буде виведено версію Node.js, яка підтверджує успішну інсталяцію.

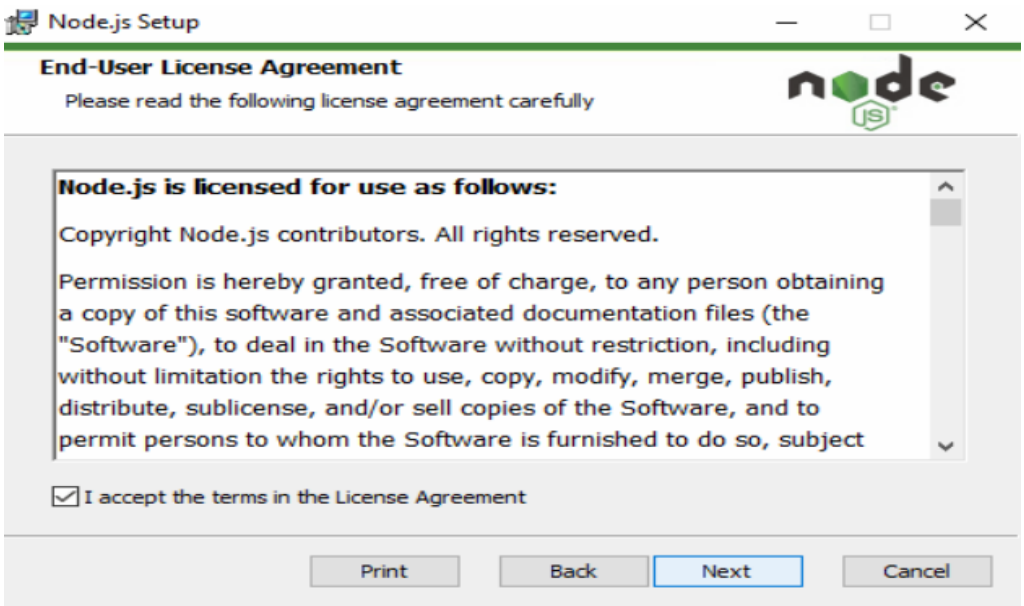


Рис 1. Встановлювач NodeJS.

Створення та запуск сервера[1]

1. Створіть новий файл із розширенням .js, наприклад server.js.
2. Введіть у файл наступний код для запуску базового сервера:
3. const http = require('http');
4.
5. const server = http.createServer((req, res) => {
6. res.statusCode = 200;
7. res.setHeader('Content-Type', 'text/plain');
8. res.end('Hello, World!\n');
9. });
10.
11. server.listen(3000, '127.0.0.1', () => {
12. console.log('Server running at http://127.0.0.1:3000/');
13. });
14. Збережіть файл.
15. У командному рядку перейдіть до каталогу, де знаходиться ваш файл, і запустіть сервер командою:
16. node server.js

    У результаті в консолі з'явиться повідомлення:

    Server running at http://127.0.0.1:3000/

Використання додаткових бібліотек

Після запуску сервера ви можете підключати різні бібліотеки для розширення функціоналу. Наприклад, для роботи з апаратними засобами, такими як Arduino, можна використовувати відповідні бібліотеки Node.js, наприклад, johnny-five чи інші.

Ця інструкція дозволяє легко створити і запустити базовий сервер на Node.js, а також розширити його можливості за допомогою додаткових бібліотек.

1.5 Бібліотека Serialport: огляд, переваги та недоліки

Опис бібліотеки

Serialport — JavaScript-бібліотека, яка надає розробникам інструменти для взаємодії з послідовними портами. Вона підтримує роботу у середовищах Node.js та Electron і є універсальним рішенням для створення програм, які потребують зв’язку з мікроконтролерами, такими як Arduino. Serialport забезпечує простий і ефективний спосіб обміну даними між комп’ютером і зовнішніми пристроями.

Переваги Serialport

Зручність роботи з Arduino
Serialport пропонує простий спосіб встановлення з'єднання між комп'ютером і Arduino через послідовний порт. Завдяки інтуїтивно зрозумілому API розробники можуть легко реалізувати передачу та отримання даних.
Доступна документація
Бібліотека має добре організовану документацію з прикладами використання та описом основних методів, що значно полегшує її вивчення.
Широкий вибір ресурсів для навчання
У мережі доступно багато навчальних матеріалів, таких як відеоуроки, статті та готові приклади, які допомагають швидко освоїти роботу з бібліотекою.
Кросплатформеність
Бібліотека підтримує роботу на Windows, macOS і Linux, що робить її універсальною для різних середовищ розробки.
Гнучкі налаштування
Можливість налаштування параметрів послідовного з'єднання, таких як швидкість передачі даних, кількість бітів даних, парність і стоп-біти, забезпечує сумісність з широким спектром пристроїв.
Підтримка асинхронної роботи
Serialport підтримує асинхронну обробку даних, що дозволяє програмам працювати ефективніше без блокування головного потоку виконання.

Недоліки Serialport

Обмеженість у виконанні складних обчислень

Serialport зосереджена на обміні даними через послідовний порт і не може використовуватися для ресурсомістких завдань, які вимагають інтенсивних обчислень.
Необхідність знання C++ для програмування Arduino
Для створення коду для Arduino потрібно володіти мовою C++, що може ускладнити роботу для тих, хто знайомий лише з JavaScript.
Проблеми сумісності версій
Іноді виникають труднощі з узгодженням версій Serialport із Node.js або іншими бібліотеками, що може вимагати додаткового часу для налагодження.
Необхідність встановлення додаткових залежностей
Для роботи бібліотеки потрібні додаткові модулі та залежності, що може створювати труднощі в обмежених середовищах.
Складність інтеграції з нестандартними пристроями
Якщо пристрій використовує нестандартний протокол обміну даними через послідовний порт, його налаштування може вимагати значних зусиль.
Чутливість до апаратних помилок
Serialport не має вбудованих механізмів для діагностики апаратних несправностей, таких як порушення з'єднання чи нестабільна передача даних, що може ускладнити налагодження.

Застосування Serialport

Serialport широко використовується для створення програм, які взаємодіють із фізичними пристроями. Її основні сценарії застосування включають:
Читання та запис даних із сенсорів, підключених до Arduino.
Управління зовнішніми пристроями, такими як мотори або дисплеї.
Розробку IoT-додатків для моніторингу та контролю параметрів у реальному часі.
Проведення тестування та діагностики обладнання через послідовні порти.
Автоматизацію процесів у промислових і побутових пристроях.

1.6 Бібліотека Babbler_h: опис, переваги та недоліки

Опис бібліотеки

Babbler_h — це клієнтська бібліотека для комунікації з пристроями на базі Arduino, яка спрощує процес обміну даними та підвищує стабільність зв’язку між програмою та мікроконтролером. Завдяки вбудованій логіці роботи вона дозволяє легко інтегрувати пристрої в програмне забезпечення, використовуючи різні канали зв’язку, такі як послідовний порт, Wi-Fi чи Bluetooth.

Переваги Babbler_h

Інкапсульована логіка підключення

Бібліотека автоматично відслідковує статус підключення, виявляє розриви зв’язку та повторно встановлює з’єднання. Це зменшує необхідність ручного керування з’єднанням.
Інформує про зміни статусу з'єднання, забезпечуючи зручність для користувача.

Асинхронна передача даних

Команди передаються у фоновому режимі, не блокуючи основний потік виконання програми. Це дозволяє розробляти більш продуктивні програми.

Умови стабільного підключення

Зв'язок вважається встановленим тільки за дотримання двох умов:
Відкритий канал зв’язку.
Пристрій відповідає командою "OK" на запит ping.

Обробка помилок

Бібліотека забезпечує обробку некоректних відповідей або відсутності відповіді від пристрою.
У разі виникнення помилок повідомляє користувача для своєчасного реагування.

Повідомлення про завершення команд

Після виконання кожної команди бібліотека надсилає підтвердження, що дозволяє ефективно відстежувати прогрес виконання.

Підтримка різних каналів зв'язку

Можливість використання послідовного порту, Wi-Fi чи Bluetooth для передачі даних, що робить бібліотеку універсальною.

Недоліки Babbler_h

Формат передачі даних

Усі команди та відповіді повинні передаватися виключно у форматі JSON. Це може обмежити використання бібліотеки у випадках, де інші формати є бажаними.

Часовий ліміт на виконання команд

Пристрій повинен відповідати на кожну команду протягом 5 секунд. У разі перевищення цього часу виникає помилка, що може стати проблемою при роботі з повільними процесами.

Обов’язковість команди ping

Для роботи бібліотеки пристрій повинен підтримувати команду ping, що вимагає відповідного налаштування прошивки.

Архітектура бібліотеки

Модуль каналів зв’язку

Відповідає за встановлення, підтримку та обробку з’єднань через послідовний порт, Wi-Fi чи Bluetooth.
Забезпечує передачу даних і обробку відповідей від пристрою.

Модуль команд

Дозволяє реєструвати функції як команди для виконання на пристрої.
Відповідає за пошук потрібної команди за назвою і передачу її для виконання.

Допоміжні протоколи

Формати для обробки команд і відповідей із використанням JSON.
Забезпечують упаковку та розпаковку даних для передачі через канал зв'язку.
Формат взаємодії

Обмін даними між пристроєм та програмою відбувається у вигляді JSON-рядків. Кожен рядок є пакетом даних, який може містити:

Команду: текстовий рядок, який ініціює дію на пристрої.
Відповідь: текстовий рядок, який підтверджує виконання команди або містить результат її виконання.

Приклад пакету JSON

```
{
 "command": "getTemperature",
 "response": {
  "status": "OK",
  "data": {
   "temperature": 23.5
  }
 }
}
```

Застосування Babbler_h

Бібліотека Babbler_h ідеально підходить для:

Моніторингу стану датчиків у реальному часі.
Керування виконавчими механізмами, такими як реле чи двигуни.
Розробки інтелектуальних пристроїв для IoT-систем.
Автоматизації процесів у домашніх чи промислових умовах.

1.7 Бібліотека Johnny-Five

Бібліотека Johnny-Five є потужною платформою для розробки робототехнічних проектів за допомогою JavaScript. Вона дозволяє взаємодіяти з мікроконтролерами, такими як Arduino, та іншими пристроями через порти вводу-виводу (I/O). Ця бібліотека особливо популярна серед розробників, оскільки вона надає простий, доступний та ефективний спосіб програмувати фізичні пристрої за допомогою JavaScript, що робить її зручною для людей без глибоких технічних знань.

Переваги використання Johnny-Five:

1. Підтримка різних платформ:

Johnny-Five працює не лише з Arduino, але й з іншими популярними платформами, такими як Raspberry Pi, BeagleBone та іншими пристроями, що підтримують порти вводу-виводу. Це дозволяє створювати гнучкі і масштабовані рішення, незалежно від конкретної платформи.

2. Простота використання:

Бібліотека має зрозумілий та інтуїтивно зрозумілий API, що дозволяє навіть новачкам у робототехніці швидко розпочати роботу з мікроконтролерами. Завдяки чітко структурованій документації, прикладам і простоті налаштування, Johnny-Five є ідеальним вибором для початківців і досвідчених розробників.

3. Широка спільнота та активна підтримка:

Johnny-Five має велику спільноту розробників, що робить її підтримку дуже зручною. Величезна кількість статей, відеоуроків, посібників і прикладів коду дозволяють швидко знаходити рішення на будь-які проблеми, що виникають під час розробки.

4. Гнучкість та можливості розширення:

Бібліотека підтримує роботу з численними пристроями, що дозволяє використовувати Johnny-Five для різноманітних проектів — від простих світлодіодів до складних роботизованих систем із датчиками і сервоприводами.

Інтеграція з Node.js:

Оскільки Johnny-Five працює на платформі Node.js, можна скористатися всіма перевагами асинхронного виконання та високопродуктивного середовища для створення інтерактивних проектів та робототехнічних систем, де програмування та взаємодія з фізичними пристроями стають простими та швидкими.

Недоліки Johnny-Five:

1. Додаткові вимоги до налаштувань:

Для того, щоб використовувати бібліотеку, необхідно встановити протокол Firmata на мікроконтролер, що додає певний крок в процес налаштування. Це може бути ускладненням для новачків, які не знайомі з процесом прошивки.

2. Залежність від протоколу Firmata:

Усі операції через Johnny-Five відбуваються через Firmata, що вимагає попереднього завантаження прошивки на мікроконтролер (наприклад, Arduino). Без цього Johnny-Five не зможе працювати з пристроєм.

3. Підвищені вимоги до обробки складних проектів:

Хоча бібліотека підтримує велику кількість датчиків і компонентів, для більш складних проектів може знадобитися глибше розуміння протоколів зв’язку та оптимізації коду.

Налаштування роботи з Johnny-Five:

1. Підключення Arduino до комп’ютера:

Використовуйте USB-кабель для з’єднання Arduino з комп’ютером, що дозволяє встановити фізичне з’єднання між пристроєм і комп’ютером.

2. Інсталяція Arduino IDE:

Завантажте та встановіть Arduino IDE для налаштування мікроконтролера. Ідея Arduino IDE полягає в тому, щоб надати користувачам інтуїтивно зрозуміле середовище для написання, компіляції та завантаження коду на Arduino.

Перевірка підключення:

Після підключення Arduino до комп’ютера відкрийте Arduino IDE, перейдіть до меню Інструменти > Порт та переконайтеся, що плата Arduino підключена до потрібного порту.

3. Інсталяція протоколу Firmata:

Для роботи з Johnny-Five потрібно завантажити прошивку Firmata на Arduino. Для цього в Arduino IDE відкрийте Файл > Приклади > Firmata > StandardFirmata і натисніть Upload. Після цього прошивка буде завантажена на мікроконтролер, і ви зможете використовувати його для роботи з Johnny-Five.

4. Використання бібліотеки у Node.js:

Після налаштування пристрою можна використовувати Johnny-Five у Node.js для контролю різних датчиків, світлодіодів, моторів та інших пристроїв. Завдяки асинхронному підходу в Node.js, Johnny-Five забезпечує високу швидкість обробки даних і можливість взаємодії з кількома компонентами одночасно.

Взаємодія з Johnny-Five:

1. Протокол Firmata:

Johnny-Five використовує Firmata як протокол для обміну даними між комп’ютером і мікроконтролером. Це дозволяє безпосередньо програмувати Arduino через Node.js і керувати різноманітними компонентами.

2. Програмування пристроїв:

Ви можете програмувати такі компоненти, як світлодіоди, датчики температури, серводвигуни, мотори і багато інших, використовуючи прості JavaScript-команди. Взаємодія з пристроями через Johnny-Five стає доступною навіть для новачків у робототехніці.

3. Виконання складних завдань:

За допомогою Johnny-Five можна створювати складні роботизовані системи, що взаємодіють з навколишнім середовищем через датчики, камери або навіть приймають команду через інтерфейси користувача, побудовані на Node.js. В результаті, Johnny-Five відкриває широкі можливості для розробки інтерактивних та інтелектуальних проектів, дозволяючи створювати потужні

роботизовані системи з використанням JavaScript, що є зручним і ефективним інструментом для робототехніки.

Висновки до розділу 1

У цьому розділі детально розглянуто аспекти застосування фреймворку Node.js для роботи з мікроконтролерами Arduino, що дозволяє створювати ефективні рішення для робототехнічних, автоматизованих та IoT-систем. Node.js став популярним інструментом для розробників завдяки своїй здатності обробляти великий потік даних та запитів одночасно, що робить його ідеальним для роботи з пристроями, що потребують реального часу.[2]

Області застосування Node.js. Один із найбільших аспектів використання Node.js для мікроконтролерів Arduino — це можливість інтеграції з різноманітними галузями, такими як робототехніка, автоматизація та Інтернет речей (IoT). Наприклад, Node.js дозволяє створювати сервери для управління роботами через веб, що робить систему доступною для керування з будь-якого пристрою, підключеного до Інтернету. Також Node.js ефективно використовуються для збору та обробки даних від сенсорів у реальному часі, що важливо в автоматизованих системах.

Переваги Node.js. Однією з найбільших переваг Node.js є його асинхронна модель роботи, яка дозволяє обробляти запити без блокування потоку виконання. Це особливо важливо для систем, що працюють у реальному часі, таких як робототехніка, де необхідно швидко обробляти дані від сенсорів і контролювати виконавчі механізми. Завдяки цьому, Node.js забезпечує високу продуктивність і ефективне управління ресурсами навіть при великих навантаженнях. Іншими словами, використання Node.js дозволяє досягти оптимальної роботи систем, не перевантажуючи процесор та інші ресурси мікроконтролера.

Встановлення та запуск. Процес встановлення Node.js для роботи з Arduino досить простий. Потрібно лише встановити сам фреймворк на комп'ютер, після чого можна підключати мікроконтролер за допомогою USB і почати налаштовувати сервер для обміну даними між Arduino та комп'ютером. Завдяки підтримці багатьох платформ, Node.js може бути використаний на різних операційних системах, таких як Windows, Linux або macOS, що робить його універсальним для використання в різних умовах.

Бібліотеки для контролю Arduino. Існує кілька бібліотек, які дозволяють розробникам ефективно взаємодіяти з мікроконтролером Arduino через

Node.js. Однією з найпопулярніших є Johnny-Five, що надає високий рівень абстракції для роботи з різними апаратними компонентами, такими як сенсори, мотори та світлодіоди. Вона дозволяє легко підключати Arduino до комп'ютера та програмувати взаємодію з різними периферійними пристроями за допомогою простого API.

Особливості бібліотек. Кожна бібліотека має свої специфічні особливості, які можуть бути корисні в залежності від типу проекту. Наприклад, Johnny-Five підтримує багато різних платформ і модулів, що дає розробникам великий вибір інструментів для створення роботизованих систем. Однак деякі бібліотеки можуть мати обмеження по сумісності з певними версіями Arduino або потребувати додаткового налаштування. Важливою перевагою є те, що ці бібліотеки мають активну спільноту і добру документацію, що допомагає уникнути проблем при налаштуванні і використанні.

Аналіз бібліотек для програмування робота на Arduino дозволив виявити основні вимоги та особливості їх використання, а також продемонструвати, як правильне дотримання рекомендацій щодо налаштування і вибору бібліотек може забезпечити стабільне та надійне управління роботами. Зокрема, вибір правильної бібліотеки залежить від конкретних задач проекту: деякі бібліотеки більше підходять для простих роботів, інші ж — для складних і багатофункціональних систем, які потребують високої точності і швидкості реакцій.

2. КОНСТРУКЦІЯ ПРИЛАДУ ARDUINO

2.1. Конструкція приладу Arduino

Arduino — це популярна апаратно-програмна платформа з відкритим вихідним кодом, яка використовується для створення електронних пристроїв і прототипів. Вона складається з мікроконтролера, який виконує роль "мозку" пристрою, і різноманітних компонентів, що дозволяють здійснювати широкий спектр завдань. Це може включати прості функції, такі як вмикання та вимикання світлодіодів, а також більш складні проекти, наприклад, створення роботизованих систем, автоматизованих пристроїв або інтеграція з іншими платформами в IoT-системах.[3]

Arduino працює за принципом відкритого коду, що дозволяє розробникам створювати та модифікувати програмне забезпечення, підлаштовуючи його під свої потреби. Це робить платформу доступною для широкого кола користувачів, від початківців до професіоналів. Вона широко застосовується в освіті, хобі-проектах, а також в індустрії для створення прототипів і тестування ідей.

Arduino підтримує безліч різноманітних компонентів, таких як сенсори, датчики, мотори, дисплеї, і світлодіоди, що дозволяє розробникам реалізовувати складні функціональні системи. Інтерфейс для програмування платформи є дуже простим і зрозумілим, а також існує велика кількість бібліотек і спільноти, що полегшує процес розробки. Завдяки цьому, Arduino став одним із найпоширеніших інструментів для створення електронних пристроїв і навчання робототехніці.

Основні компоненти та принцип роботи Arduino

1. Мікроконтролер

Мікроконтролер є серцем плати Arduino. Це програмоване пристрій, який відповідає за виконання програмного коду та обробку вхідних і вихідних сигналів. Наприклад, на популярній платі Arduino Uno використовується мікроконтролер ATmega328P. Цей мікроконтролер може зчитувати сигнали з датчиків та передавати їх на вивідні пристрої (світлодіоди, двигуни, дисплеї тощо).

2. Система живлення

Arduino може живитися від різних джерел. Найпоширенішим є USB-підключення до комп'ютера, яке забезпечує як живлення, так і з'єднання для передачі даних. Окрім того, плата підтримує живлення від батарей або

зовнішнього адаптера постійного струму в діапазоні 7-12 В. Це дає можливість використовувати Arduino в автономних системах.

3. Порти вводу-виводу (I/O)

Arduino має різноманітні порти для підключення зовнішніх пристроїв. Ці порти можуть бути цифровими або аналоговими. Цифрові порти використовуються для простих вмикань і вимикань пристроїв (наприклад, світлодіодів), тоді як аналогові порти дозволяють зчитувати змінні значення, такі як температура або рівень освітленості, що подаються від датчиків.

4. Пам'ять

У платах Arduino є три види пам'яті: Flash-пам'ять, яка зберігає програмний код; SRAM, що використовується для зберігання змінних під час виконання програми; та EEPROM, яка дозволяє зберігати дані навіть після вимкнення живлення. Така структура пам'яті дає можливість ефективно зберігати і обробляти інформацію на різних етапах роботи пристрою.

5. Роз'єм USB та кнопка Reset

USB-роз'єм забезпечує зв'язок між Arduino та комп'ютером для завантаження програми, а також може служити джерелом живлення. Кнопка Reset дозволяє перезавантажити програму на мікроконтролері без необхідності відключати живлення.

6. Можливості розширення

Для розширення функціональних можливостей плати використовуються шилди — додаткові плати, які підключаються до Arduino. Вони можуть надавати підтримку таких технологій, як Wi-Fi, Bluetooth, а також додавати можливості управління моторними приводами, підключенням сенсорів, дисплеїв і багато іншого.

2.2 Резистор

Резистор (Рис.2) — це електронний компонент, який обмежує потік електричного струму в колі. Його основна функція полягає в захисті чутливих елементів схеми, наприклад, світлодіодів, від надмірного струму. Значення опору резистора вимірюється в Омах (Ω), а вибір резистора залежить від необхідного струму та напруги в колі. У схемах з RGB-світлодіодами резистори часто використовуються для кожного кольорового каналу, щоб забезпечити їхню стабільну роботу.

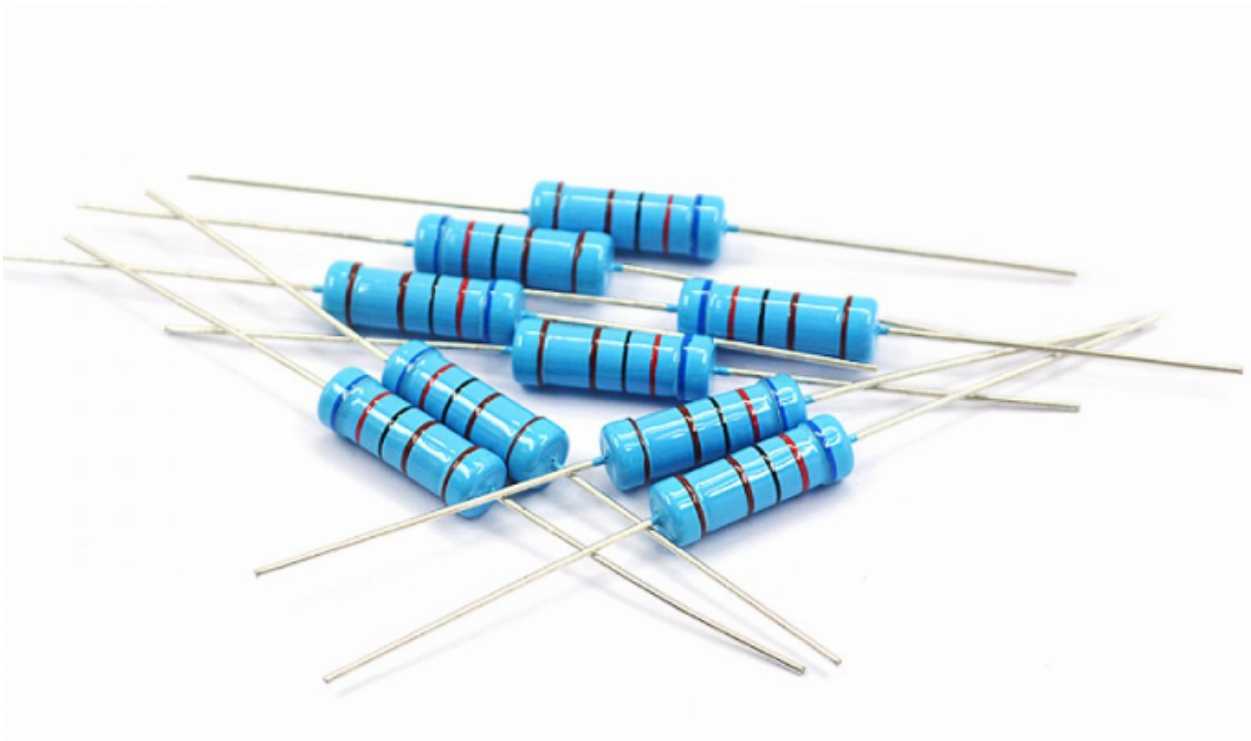


Рис. 2 Резистор

2.3RGB-світлодіод

RGB-світлодіод (Рис.3) — це пристрій, який може відображати широкий спектр кольорів за рахунок комбінування трьох базових кольорів: червоного (Red), зеленого (Green) і синього (Blue). Кожен з цих кольорів управляється окремим анодом або катодом (залежно від типу світлодіода), що дозволяє створювати різні відтінки шляхом зміни інтенсивності кожного кольору. RGB-світлодіоди широко використовуються для декоративного освітлення, індикаторів стану та створення динамічних світлових ефектів.

Рис.3 RGB- світлодіод

2.3 Типи плат Arduino

Arduino представлено кількома різними моделями, кожна з яких має свої характеристики, що підходять для різних завдань:
Arduino Uno (Рис.4) — найбільш популярна плата , яка підходить для багатьох базових завдань.
Це технічні характеристики мікроконтролера ATmega328, який є основою для популярних плат на кшталт Arduino Uno. Ось короткий опис його характеристик:

Мікроконтролер: ATmega328
Робоче напруга: 5 В
Рекомендуване напруга живлення: 7-12 В
Граничне напруга живлення: 6-20 В
Цифрові входи/виходи: 14 (6 з них підтримують ШІМ - широтно-імпульсну модуляцію)
Аналогові входи: 6
Максимальний струм одного виходу: 40 мА
Максимальний вихідний струм на виході 3.3V: 50 мА
Flash-пам'ять: 32 КБ (з яких 0.5 КБ використовуються для завантажувача)
SRAM: 2 КБ
EEPROM: 1 КБ
Тактова частота: 16 МГц

Ці характеристики дозволяють використовувати ATmega328 для багатьох проектів, що вимагають управління через цифрові або аналогові сигнали, а також для базових систем обробки даних.

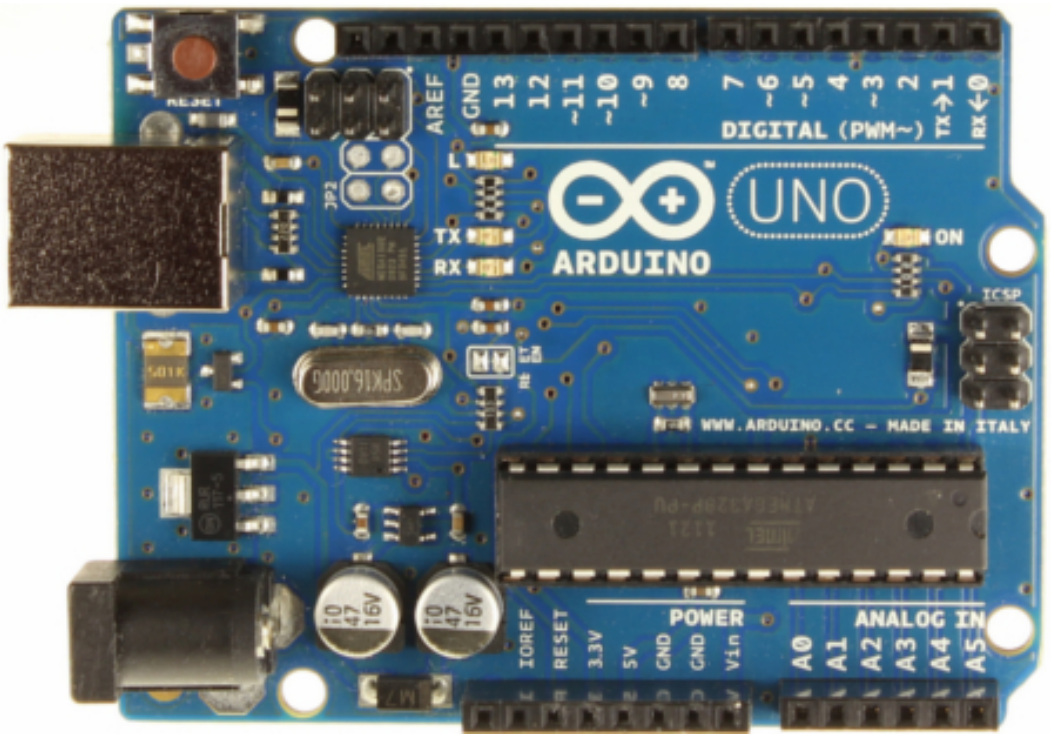


Рис.4 Arduino Uno

Arduino Mega (Рис.5) — має значно більше портів вводу-виводу і підходить для складніших проектів.

Ось характеристики мікроконтролера ATmega2560, який має набагато більше ресурсів порівняно з ATmega328 і використовується в більш складних проектах:

-Мікроконтролер: ATmega2560

-Робоче напруга: 5 В

-Рекомендуване напруга живлення: 7-12 В

-Граничне напруга живлення: 6-20 В

-Цифрові входи/виходи: 54 (з яких 15 можуть бути використані як ШІМ - широтно-імпульсну модуляцію)

-Аналогові входи: 16

-Максимальний струм одного виходу: 40 мА

-Максимальний вихідний струм на виході 3.3V: 50 мА

-Flash-пам'ять: 256 КБ (з яких 8 КБ використовуються для завантажувача)

-SRAM: 8 КБ

-EEPROM: 4 КБ

-Тактова частота: 16 МГц

Цей мікроконтролер значно потужніший, ніж ATmega328, завдяки більшій кількості цифрових і аналогових входів/виходів, а також більшій пам'яті (як

Flash, так і SRAM). Це робить його ідеальним для більш складних або великих проектів, де потрібно більше пінів або обробки даних.

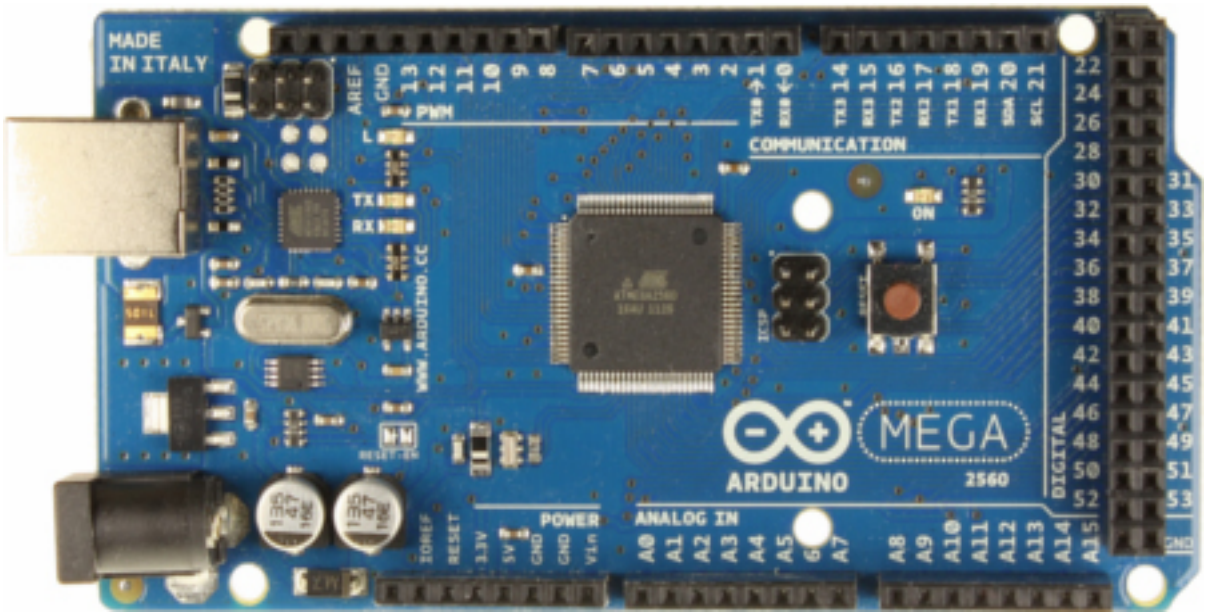


Рис.5 Arduino Mega

Arduino Nano (Рис.6) — компактна плата для вбудованих систем, де важлива економія простору.

Це характеристики мікроконтролера Atmel ATmega168 або ATmega328, який часто використовується в таких популярних платах, як Arduino, і має дуже схожі характеристики, з невеликими відмінностями залежно від моделі.

Ось детальніше:

-Мікроконтролер: ATmega168 або ATmega328
-Робоче напруга (логічний рівень): 5 В
-Рекомендуване напруга живлення: 7-12 В
-Граничне напруга живлення: 6-20 В
-Цифрові входи/виходи: 14 (з яких 6 можуть бути використані як широтно-імпульсну модуляцію)
-Аналогові входи: 8
-Максимальний струм одного виходу: 40 мА
-Flash-пам'ять: 16 КБ (для ATmega168) або 32 КБ (для ATmega328) (з яких 2 КБ використовуються для завантажувача)
-SRAM: 1 КБ (для ATmega168) або 2 КБ (для ATmega328)
-EEPROM: 512 байт (для ATmega168) або 1 КБ (для ATmega328)
-Тактова частота: 16 МГц

-Розміри плати: 1.85 см x 4.3 см

Цей мікроконтролер є відмінним вибором для проектів, де потрібно помірно потужне обчислення та управління з великою кількістю цифрових і аналогових входів, при цьому плата займає мінімум місця, що робить її зручною для компактних застосувань.

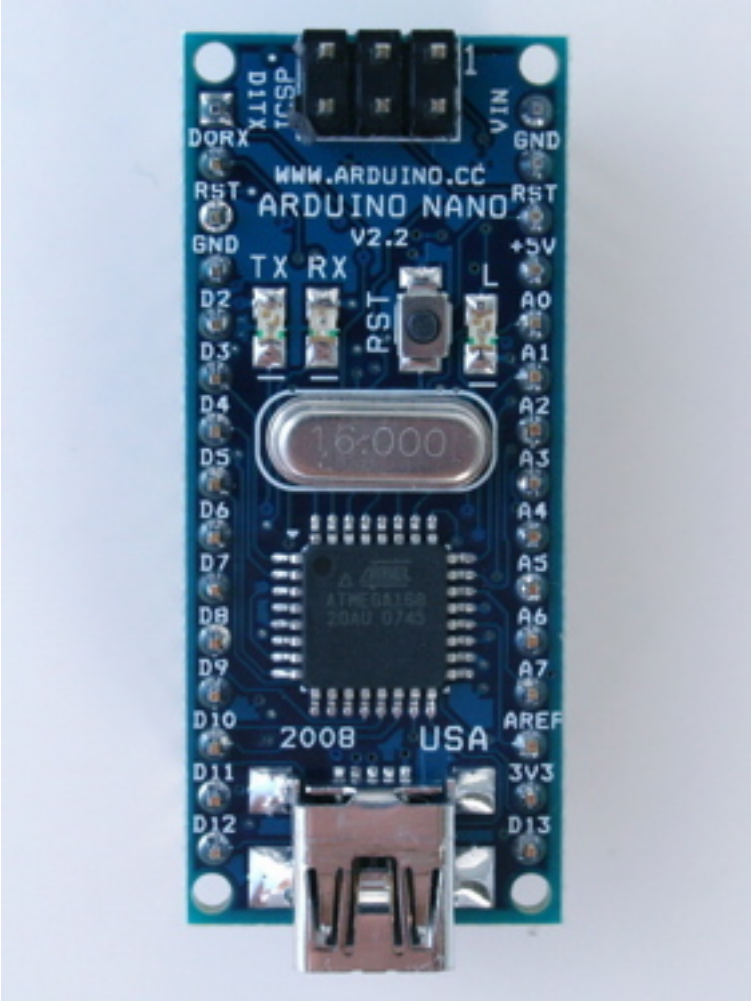


Рис.6 Arduino Nano

Arduino Leonardo (Рис.7) — ця плата має можливість емуляції клавіатури або миші через USB, що робить її корисною для створення специфічних пристроїв введення.

Ось характеристики мікроконтролера ATmega32u4, який є потужнішим варіантом у порівнянні з іншими чіпами, завдяки вбудованому USB-контролеру і більшим можливостям для обробки сигналів:

-Мікроконтролер: ATmega32u4
-Робоче напруга: 5 В
-Рекомендуване напруга живлення: 7-12 В
-Граничне напруга живлення: 6-20 В
-Цифрові входи/виходи: 20
-Канали ШІМ: 7
-Аналогові входи: 12
-Максимальний струм одного виходу: 40 мА
-Максимальний вихідний струм на виході 3.3V: 50 мА
-Flash-пам'ять: 32 КБ (з яких 4 КБ використовуються для завантажувача)
-SRAM: 2.5 КБ
-EEPROM: 1 КБ
-Тактова частота: 16 МГц

Цей мікроконтролер має більше аналогових входів та цифрових пінів у порівнянні з іншими чіпами, і також підтримує 7 каналів ШІМ, що дає більше можливостей для управління різними пристроями (наприклад, для керування моторами, яскравістю світлодіодів тощо). Вбудований USB-контролер дозволяє підключати мікроконтролер безпосередньо до комп'ютера або інших USB-пристроїв без додаткових компонентів.

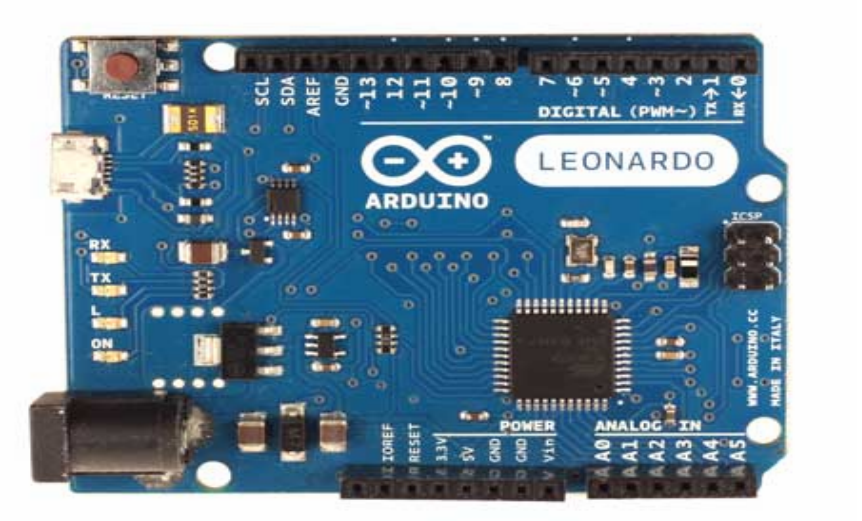


Рис.7 Arduino Leonardo

2.4 Принцип роботи Arduino

Принцип роботи Arduino полягає в тому, що він функціонує як центральний мікроконтролер, який зчитує дані з підключених пристроїв (датчиків) і приймає рішення для виконання завдань на основі вбудованого програмного коду. Ось як це працює:

1. Зчитування даних з датчиків:

Arduino отримує сигнали від різних датчиків, таких як температурні, вологомірні, освітлювальні або руху. Ці сигнали можуть бути аналоговими або цифровими і передаються на пін мікроконтролера.

2. Обробка даних:

Після того як дані надходять в Arduino, мікроконтролер починає їх обробку. За допомогою програмного коду (зазвичай на мові C++ або схожій), Arduino аналізує ці дані. Наприклад, якщо температура в приміщенні перевищує певний поріг, Arduino може прийняти рішення, щоб включити охолоджувальний вентилятор.

3. Виконання команд:

Після аналізу вхідних даних Arduino виконує визначені дії. Це можуть бути різні команди:

Включення або вимкнення світлодіодів (LED);

Управління мотором або сервоприводом (наприклад, для відкриття дверей або руху робота);

Передача даних на комп'ютер або інші пристрої (наприклад, через послідовний порт або Wi-Fi).

Це дозволяє Arduino інтегруватися в різноманітні електронні проекти, здійснювати автоматизацію і створювати інтерактивні системи, які реагують на зміни навколишнього середовища.

Висновки до розділу 2

У цьому розділі розглядається ефективне управління мікроконтролером Arduino за допомогою базових компонентів і матеріалів, що дозволяють створювати різноманітні електронні проекти. Аналіз технічних характеристик і конструктивних особливостей Arduino демонструє, чому ця платформа є такою популярною серед розробників і ентузіастів електроніки. Одним із ключових аспектів є важливість правильного підключення компонентів до мікроконтролера, що забезпечує стабільну роботу системи. Важливу роль відіграє також належне живлення, оскільки забезпечення необхідного напруги і струму для різних компонентів критично важливе для безперебійної роботи пристрою.

Особлива увага приділяється портам вводу-виводу Arduino, їх призначенню та правильному використанню при з'єднанні з різними пристроями та датчиками. Порти можуть бути налаштовані для роботи як на вхід, так і на вихід, що дозволяє підключати різноманітні компоненти, такі як світлодіоди, датчики температури, датчики руху і багато інших. Це робить Arduino надзвичайно універсальним інструментом для створення прототипів і автоматизації процесів.[4]

Порядок підключення основних елементів, таких як світлодіоди, датчики і інші пристрої, є критично важливим для побудови функціонального прототипу. Роз'яснення, як правильно з'єднувати ці елементи з платою Arduino, дає змогу уникнути помилок і забезпечити коректну роботу системи. Важливим етапом є також підключення Arduino до комп'ютера через USB-кабель, що дозволяє швидко налаштувати пристрій для програмування та інтеграції в більш складні проекти. Завдяки цьому, користувачі можуть легко відтворювати і тестувати свої ідеї, що значно прискорює процес розробки.

Отже, Arduino є потужним і доступним інструментом для розробки електронних пристроїв, і його простота у підключенні та налаштуванні робить його ідеальним вибором для новачків та професіоналів, які прагнуть швидко створювати інноваційні прототипи.

3. ПРИКЛАДИ ВИКОРИСТАННЯ БІБЛІОТЕК ДЛЯ КОНТРОЛЮВАННЯ РОБОТА НА ARDUINO

3.1. Приклад використання бібліотеки Serialport

Serialport — це потужна бібліотека для JavaScript, що забезпечує зручну взаємодію з послідовними портами. Вона дозволяє легко підключати та взаємодіяти з різними пристроями, такими як Arduino, через серійні порти, надаючи простий і інтуїтивно зрозумілий API для відправки і отримання даних. Завдяки Serialport, розробники можуть швидко налаштувати двосторонню комунікацію між комп'ютером та мікроконтролерами, що є важливим елементом для багатьох проектів у галузі робототехніки, автоматизації та IoT.

Однією з основних переваг Serialport є її здатність працювати на різних операційних системах, таких як Windows, Linux та macOS, що робить її універсальним інструментом для розробки багатоплатформних рішень. Бібліотека підтримує низку функцій для роботи з послідовними портами, включаючи налаштування параметрів зв'язку (швидкість, парність, стопові біти тощо), а також надає можливості для асинхронного зчитування та запису даних.

Завдяки цьому, Serialport є ідеальним інструментом для проектів, які потребують швидкої та ефективної передачі даних між комп'ютером і мікроконтролерами, зокрема для керування Arduino або іншими апаратними платформами. Вона підтримує передачу даних у реальному часі, що робить її чудовим вибором для систем, де важлива безперервна комунікація з пристроями, зокрема в автоматизації, сенсорних мережах та робототехніці.

Встановлення бібліотеки

Для роботи з бібліотекою Serialport необхідно встановити її локально за допомогою команди:

```
npm install serialport
```

Приклад коду для запису в послідовний порт

На стороні Arduino можна використовувати наступний код для передачі повідомлення "Hello world!" через послідовний порт:

```
void setup() {
```

```
  Serial.begin(9600); // Встановлення швидкості передачі даних
}

void loop() {
  Serial.println("Hello world!"); // Надсилання повідомлення
  delay(1000); // Затримка 1 секунда
}
```

Важливі моменти, які слід враховувати:

Визначення порту: На комп’ютері необхідно знайти назву послідовного порту. Це можна зробити через Arduino IDE у меню Інструменти > Порти.

BaudRate: Швидкість передачі даних (baudRate) в коді Node.js повинна відповідати швидкості, встановленій у коді Arduino (Serial.begin(9600)).

Використання пакету @serialport/parser-readline

Для спрощення обробки даних можна скористатися пакетом @serialport/parser-readline. Цей пакет дозволяє розділяти повідомлення за допомогою заданого роздільника (delimiter), наприклад, нової строки.

Ось приклад коду на Node.js для зчитування даних з послідовного порту:

```
const SerialPort = require('serialport');
const Readline = require('@serialport/parser-readline');

// Встановлення послідовного порту
const port = new SerialPort('COM3', { baudRate: 9600 });

// Підключення парсера для розділення повідомлень
const parser = port.pipe(new Readline({ delimiter: '\r\n' }));

// Обробка подій
port.on('open', () => {
  console.log('Serial port opened');
});

parser.on('data', (data) => {
  console.log(`Got message from Arduino: ${data}`);
});
```

Пояснення параметрів:

path: Системний шлях до послідовного порту (наприклад, COM3 у Windows).

options: Додаткові налаштування порту, включаючи швидкість передачі (baudRate).

pipe: Функція для передачі даних через канал із застосуванням додаткових парсерів.

Приклад використання Serialport для управління світлодіодом

Для того щоб світлодіод на Arduino мигав із затримкою 1 секунду, необхідно написати код, який надсилає відповідну команду через послідовний порт. Код на стороні Node.js може виглядати так:

```
const SerialPort = require('serialport');
const port = new SerialPort('COM3', { baudRate: 9600 });

port.write('LED_ON', (err) => {
   if (err) {
      return console.log('Error on write:', err.message);
   }
   console.log('Command sent: LED_ON');
});
```

Код на Arduino відповідатиме отриманню та виконанню цієї команди:

```
void setup() {
   Serial.begin(9600);
   pinMode(13, OUTPUT); // Світлодіод на піні 13
}

void loop() {
   if (Serial.available()) {
      String command = Serial.readString();
      if (command == "LED_ON") {
         digitalWrite(13, HIGH); // Увімкнути світлодіод
         delay(1000); // Затримка 1 секунда
         digitalWrite(13, LOW); // Вимкнути світлодіод
      }
   }
}
```

Архітектура бібліотеки Serialport

Бібліотека Serialport для Node.js забезпечує можливість взаємодії з послідовними портами, що дозволяє здійснювати обмін даними між комп'ютером і різними пристроями, зокрема з мікроконтролерами, такими як

Arduino, через серійні порти. Архітектура бібліотеки побудована на кількох ключових компонентах, що забезпечують ефективну роботу з послідовними з'єднаннями.

Основні компоненти архітектури

1. Серійний порт:

Це основний об'єкт, який відповідає за підключення та роботу з фізичним серійним портом. Бібліотека Serialport створює абстракцію, яка дозволяє програмісту працювати з різними портами (наприклад, /dev/ttyUSB0 на Linux або COM3 на Windows), не залежно від специфіки кожної операційної системи.

Порт визначається через його унікальне ім'я, і на основі цього імені бібліотека забезпечує доступ до з'єднання.

2. Потоки для вводу/виводу:

Бібліотека Serialport реалізує механізм роботи з потоками для асинхронної передачі даних через серійний порт. Всі операції, такі як читання і запис, виконуються через потоки — стандартний механізм роботи з великими об'ємами даних в Node.js.

Потоки розділені на два типи: потік вводу (для отримання даних з пристрою) і потік виведення (для надсилання даних до пристрою).

3. Обробка подій:

Події — основний спосіб управління з'єднанням і передачею даних. Наприклад, події спрацьовують при відкритті порту, отриманні даних, виявленні помилок або закритті з'єднання.

Завдяки використанню асинхронних подій бібліотека дозволяє одночасно виконувати кілька операцій, не блокуючи основний потік виконання програми.

4. Налаштування параметрів порту:

Serialport дозволяє налаштувати такі параметри, як швидкість передачі даних (baud rate), кількість бітів даних, парність, стоп-біти та інші налаштування, що важливо для коректної взаємодії з пристроями.

Ці параметри можуть бути вказані при ініціалізації порту або змінені в процесі роботи.

5. Обробка помилок:

Для забезпечення надійної роботи бібліотека обробляє всі можливі помилки, що можуть виникнути при з'єднанні або передачі даних. Помилки обробляються через події, що дозволяє розробникам швидко реагувати на проблеми (наприклад, якщо порт не знайдений або пристрій не відповідає).

6. Буферизація даних:

Для ефективної роботи з великими об'ємами даних Serialport використовує механізм буферизації. Даний підхід дозволяє зберігати отримані дані в

буфері до того, як вони будуть повністю оброблені програмою або передані в інший процес.

Модулі бібліотеки Serialport

1. SerialPort:

Основний модуль для роботи з серійним портом. Він надає API для відкриття порту, налаштування параметрів і взаємодії з пристроєм через серійний порт.

2. Parser:

Модуль для обробки даних, що надходять через серійний порт. Він може розпізнавати різні формати даних, наприклад, байтові потоки або текстові рядки, і перетворювати їх в зручний для подальшої обробки формат.

Список доступних портів:

Модуль для отримання інформації про всі доступні серійні порти на комп'ютері. Це дозволяє програмно вибирати порт для підключення до пристрою.

3. Керування потоком:

Бібліотека підтримує кілька методів управління потоком, таких як XON/XOFF і RTS/CTS. Це важливо для забезпечення стабільної передачі даних між пристроєм і комп'ютером.

Структура взаємодії

1. Ініціалізація порту:

Для початку роботи з серійним портом бібліотека відкриває відповідний порт, використовуючи його унікальне ім'я. Після цього встановлюються необхідні параметри передачі даних, такі як baud rate, біт парності та інші налаштування.

2. Читання і запис даних:

Читання даних здійснюється за допомогою обробки події data, що активується кожного разу, коли нові дані надходять на порт.

Запис даних на пристрій виконується через потік виведення, який записує байти в порт.

3. Обробка помилок:

У разі виникнення помилок (наприклад, якщо порт недоступний або відсутня відповідь від пристрою), бібліотека генерує події помилок, які можна обробити для належного реагування.

3.2. Приклад використання бібліотеки Babbler_h

Babbler_h — це клієнтська бібліотека, створена для полегшення комунікації з пристроями Arduino, яка дозволяє спростити взаємодію завдяки інкапсуляції логіки з'єднання. Бібліотека значно знижує складність роботи з послідовними портами, автоматизуючи підключення і забезпечуючи стабільний зв'язок між комп'ютером і мікроконтролером. Однією з основних переваг Babbler_h є її здатність автоматично відновлювати з'єднання у випадку його розриву, що робить її ідеальним інструментом для проектів, де надійність зв'язку є критично важливою.

Ця бібліотека особливо корисна для розробників, які працюють в умовах, де можливі втрати зв'язку або коли з'єднання повинно бути стабільним протягом тривалого часу. Вона значно полегшує роботу з Arduino в таких складних умовах, де важливо мінімізувати можливі збої у зв'язку. Крім того, Babbler_h дозволяє зменшити потребу в ручному втручанні, оскільки автоматично керує перепідключенням, що знижує час простою системи та підвищує її надійність.

Інтерфейс бібліотеки спрощує налаштування та інтеграцію з іншими компонентами проекту, що робить її доступною навіть для початківців. Її використання не вимагає глибоких знань про роботу з послідовними портами, оскільки всі складні процеси взаємодії з апаратними компонентами вже інкапсульовані в самій бібліотеці. Це дозволяє розробникам зосередитись на основних функціональних аспектах проекту, не турбуючись про стабільність з'єднання.

Встановлення бібліотеки

Щоб розпочати роботу з Babbler_h, виконайте наступні кроки:

1. Встановіть бібліотеку через менеджер пакетів npm:
2. npm install babbler-js
3. Завантажте всі необхідні залежності:
4. npm install

Основні можливості бібліотеки

Автоматичне перепідключення: Бібліотека автоматично обробляє розриви з'єднання, сповіщає про зміни статусу і перепідключається, що забезпечує стабільну роботу навіть за умов нестабільного з'єднання.

Обробка команд у черзі: Команди надсилаються одна за одною, що дозволяє уникнути блокування виконання основного коду та забезпечує ефективну обробку запитів.
JSON-формат: Всі команди та відповіді між пристроєм і бібліотекою передаються у форматі JSON, що спрощує інтеграцію та роботу з даними.

Умови для успішного з'єднання

1. Відкритий канал зв'язку: Порт повинен бути налаштований і доступний для обміну даними. Це забезпечує надійний зв'язок між комп'ютером і пристроєм.
2. Перевірка статусу пристрою: Пристрій повинен відповідати на команду ping зі статусом OK. Це підтверджує, що пристрій правильно підключений і готовий до взаємодії.

Приклад коду для Node.js

Наведений нижче приклад демонструє, як встановити з'єднання та обмінюватися командами з Arduino за допомогою бібліотеки Babbler_h.

```
const Babbler = require('babbler-js');
const SerialPort = require('serialport');

// Встановлення послідовного порту
const port = new SerialPort('COM3', { baudRate: 9600 });
const babbler = new Babbler(port);

// Обробка подій
babbler.on('connected', () => {
   console.log('Babbler connected to Arduino');
   // Надсилання команди ping
   babbler.send('ping', (response) => {
      console.log(`Device response: ${response}`);
   });
});

babbler.on('disconnected', () => {
   console.log('Babbler disconnected');
});

// Початок роботи
port.on('open', () => {
   console.log('Serial port opened');
   babbler.connect();
});
```

Архітектура бібліотеки Babbler_h

Бібліотека Babbler_h побудована з метою забезпечення максимальної зручності в комунікації між комп’ютером та пристроями на базі Arduino. Її архітектура складається з трьох взаємопов’язаних компонентів, які спільно гарантують стабільність і функціональність роботи.

Основним елементом цієї бібліотеки є модуль для встановлення каналів зв’язку. Він відповідає за управління підключеннями, забезпечуючи підтримку різних інтерфейсів, таких як послідовний порт, Wi-Fi або Bluetooth. Це дозволяє використовувати бібліотеку для різних типів проєктів: від простих локальних застосувань до складних систем з бездротовою передачею даних. Бібліотека автоматично відстежує статус підключення та перепідключається у разі втрати зв’язку.

Другим важливим компонентом є модуль обробки команд. Ця частина бібліотеки реалізує інтелектуальну систему реєстрації команд, пошуку за назвою та їх обробки. Вона дозволяє розробникам додавати власні команди, адаптуючи пристрій під конкретні потреби проєкту. Завдяки цьому можливості бібліотеки значно розширюються, адже кожен пристрій може виконувати унікальні функції, залежно від вимог користувача.[5]

Не менш важливим є і модуль допоміжних протоколів, що відповідає за стандартизацію передачі даних. Формат JSON обрано як основний, оскільки він є універсальним і легко інтегрується з іншими системами. Команди та відповіді в цьому форматі дозволяють зменшити ризик помилок у передачі даних, а також спрощують діагностику і налагодження під час розробки.

Обмеження бібліотеки

Як і будь-яке програмне забезпечення, бібліотека Babbler_h має свої обмеження. В першу чергу, це стосується використання формату JSON. Усі команди, які передає або отримує пристрій, повинні відповідати цьому формату. Хоча це спрощує роботу з даними, воно вимагає відповідної підготовки пристрою, щоб підтримувати цей стандарт.

Ще одним важливим аспектом є часові обмеження. Пристрій має надати відповідь на команду протягом 5 секунд. У разі, якщо цей час перевищено, програма повідомить про помилку виконання. Це обмеження створене для забезпечення стабільності роботи системи, оскільки затримки у відповідях можуть вплинути на послідовність виконання інших команд.

Особливої уваги заслуговує вимога підтримки команди ping. Ця команда є базовою і дозволяє перевірити стан пристрою. Вона забезпечує швидку діагностику з'єднання та допомагає виявити можливі проблеми.

Особливості роботи

Бібліотека Babbler_h орієнтована на розробників, які прагнуть створювати стабільні й функціональні рішення на базі Arduino. Її основна перевага полягає в автоматизації багатьох рутинних процесів. Наприклад, у випадку розриву з'єднання бібліотека не лише повідомляє користувача, але й самостійно намагається перепідключитися до пристрою.

Додатковою перевагою є асинхронна обробка команд. Це дозволяє уникати блокування основного потоку виконання програми та зберігати високу продуктивність навіть за умов інтенсивної взаємодії з пристроєм. Завдяки цьому бібліотека є ідеальним вибором для проєктів, де необхідна висока швидкість виконання та відгуку системи.
Розробники можуть використовувати Babbler_h як базовий інструмент для реалізації складних інтеграційних проєктів, що включають підключення Arduino до веб-додатків, серверів чи інших програмних платформ.

Приклад використання бібліотеки Babbler_h

Бібліотека Babbler_h є потужним інструментом для ефективної клієнтської комунікації з пристроями на базі Arduino, що значно полегшує процес роботи з послідовними портами. Вона забезпечує простоту налаштування та використання, дозволяючи розробникам швидко інтегрувати зв'язок між комп'ютером та мікроконтролером. Однією з ключових особливостей цієї бібліотеки є автоматичне перепідключення пристроїв у разі втрати з'єднання, що робить її надзвичайно корисною в проектах, де важлива стабільність і надійність зв'язку між пристроями.

Ця особливість дає значну перевагу при розробці проектів, які передбачають постійне з'єднання між мікроконтролерами та комп'ютерними системами, а також у випадках, коли необхідно здійснювати тривалі операції з передачі даних, де навіть короткочасна втрата зв'язку може призвести до серйозних проблем. Використання бібліотеки Babbler_h дозволяє уникнути таких ситуацій, оскільки вона автоматично спробує відновити з'єднання, забезпечуючи безперервну і стабільну роботу.

Ще однією важливою перевагою цієї бібліотеки є її простота у використанні. Встановлення та налаштування Babbler_h не вимагає глибоких знань у роботі з послідовними портами, що робить її доступною для широкого кола користувачів, у тому числі для початківців. Бібліотека має чітку документацію та зрозумілий інтерфейс, що дозволяє легко інтегрувати її в проекти без необхідності вивчати складні аспекти роботи з низькорівневими компонентами, такими як налаштування послідовних з'єднань.

Завдяки своїй зручності і надійності Babbler_h ідеально підходить для різноманітних застосувань, включаючи автоматизацію, моніторинг та управління, а також для систем, де важлива безперервна комунікація між пристроями. Це робить бібліотеку важливим інструментом для тих, хто працює з Arduino та іншими мікроконтролерами, адже вона знижує складність розробки та покращує стабільність проектів.

Підготовка до роботи з бібліотекою

Перед тим, як розпочати роботу з бібліотекою Babbler_h, необхідно виконати кілька простих кроків для налаштування. Перш за все, потрібно встановити саму бібліотеку через менеджер пакетів npm. Це робиться за допомогою наступної команди:

1. Встановіть бібліотеку за допомогою команди:
2. npm install babbler-js

Після цього необхідно завантажити всі залежності для роботи програми, виконавши команду:

2. Завантажте всі залежності:
3. npm install

Цей крок дозволить завантажити всі необхідні пакети для правильної роботи бібліотеки.[6]

Реалізація базового підключення

Після того як бібліотека встановлена, можна переходити до створення базового підключення між програмою та пристроєм Arduino. Для цього створіть файл, наприклад, babbler.js, і додайте в нього наступний код:

```
const Babbler = require('babbler-js');
const SerialPort = require('serialport');

const portPath = 'COM3'; // Вкажіть правильний порт
const babbler = new Babbler(portPath);

// Події
babbler.on('connected', () => {
   console.log('Connected to the device');
   babbler.send('ping', (response) => {
      console.log(`Device response: ${response}`);
   });
});

babbler.on('disconnected', () => {
   console.log('Device disconnected');
});

// Підключення
babbler.connect();
```

Код забезпечує підключення до пристрою через вказаний послідовний порт. Після підключення до пристрою відправляється команда ping, а відповідь від Arduino виводиться на екран.

Для запуску скрипта використовуйте команду:

node babbler.js

Якщо підключення відбудеться успішно, у консолі буде виведено повідомлення:

Connected to the device
Device response: OK

Якщо з'єднання буде розірвано, програма повідомить про це та завершить роботу.

Приклад використання для керування світлодіодами

Один з можливих варіантів застосування бібліотеки Babbler_h — це керування світлодіодами. У цьому прикладі світлодіод буде вмикатися та вимикатися кожні 2 секунди. Крім того, якщо USB-кабель буде від'єднано, програма намагатиметься перепідключити пристрій через кожні 3 секунди. Це забезпечує безперервну роботу програми навіть при можливих фізичних відключеннях пристрою.

Ось код для реалізації такого функціоналу:

```
const Babbler = require('babbler-js');
const SerialPort = require('serialport');

const portPath = 'COM3'; // Вкажіть правильний порт
const babbler = new Babbler(portPath);

let ledState = false;

babbler.on('connected', () => {
   console.log('Connected to the device');
   setInterval(() => {
      ledState = !ledState;
      const command = ledState ? 'led_on' : 'led_off';
      babbler.send(command, (response) => {
         console.log(`Command: ${command}, Response: ${response}`);
      });
   }, 2000);
});
```

```
babbler.on('disconnected', () => {
   console.log('Device disconnected. Reconnecting...');
   setTimeout(() => babbler.connect(), 3000);
});

// Підключення
babbler.connect();
```

У цьому прикладі кожні 2 секунди змінюється стан світлодіода. Програма автоматично відправляє команду для ввімкнення або вимкнення світлодіода (led_on / led_off), в залежності від поточного стану. Водночас, якщо з'єднання з пристроєм розривається, програма буде чекати 3 секунди та знову намагатиметься підключитися до пристрою.

Запуск програми

Після того як ви налаштуєте програму, її можна запустити за допомогою команди:

node babbler-basic-blink.js

Поведінка програми

Програма демонструє наступні особливості під час виконання:

- У консолі буде видно стан світлодіода (увімкнено/вимкнено).
- У разі відключення USB-кабеля програма спробує автоматично перепідключити пристрій.
- Після успішного перепідключення світлодіод продовжить мигати, що є свідченням відновлення роботи.

Цей підхід дозволяє створювати більш стійкі і надійні програми, здатні працювати навіть за умов нестабільного з'єднання.[7]

3.3. Приклад використання бібліотеки Johnny-Five

Johnny-Five — це одна з найпопулярніших JavaScript-бібліотек для робототехніки, що надає можливість керувати мікроконтролерами Arduino через фреймворк Node.js. Завдяки простоті налаштування та інтеграції з різними апаратними платформами, ця бібліотека відкриває широкі можливості для розробників, які бажають створювати роботизовані системи, автоматизовані пристрої та IoT-рішення. Johnny-Five дозволяє працювати з такими компонентами, як сенсори, мотори, світлодіоди, дисплеї та інші елементи апаратного забезпечення.

Основною перевагою Johnny-Five є її простота у використанні, що дозволяє розробникам швидко і без особливих зусиль інтегрувати Arduino в різні проекти. Бібліотека абстрагує складність низькорівневого програмування, надаючи зручний API для взаємодії з апаратними пристроями. Це дозволяє зосередитись на розробці логіки та функціональності, замість того, щоб витрачати час на налаштування специфічних компонентів.

Johnny-Five підтримує різноманітні платформи, такі як Arduino Uno, Raspberry Pi, Intel Edison, BeagleBone, що дозволяє використовувати бібліотеку для різноманітних проектів. Вона забезпечує підтримку для підключення до різних сенсорів і виконавчих механізмів, а також дозволяє реалізувати складні алгоритми керування з використанням JavaScript.[8]

Крім того, Johnny-Five забезпечує чудову документацію, активну спільноту та велику кількість прикладів коду, що допомагає розробникам швидко освоїти бібліотеку та використовувати її для створення різноманітних проектів у галузі робототехніки, автоматизації та IoT.

Початок роботи

1. Встановлення бібліотеки

Щоб почати роботу з бібліотекою Johnny-Five, виконайте команду:

npm install johnny-five

2. Необхідні компоненти

Для прикладу роботи з бібліотекою вам знадобляться такі компоненти:

Arduino Uno (або сумісна плата).

Світлодіод (LED).

Прототипна плата.

Резистор (200–330 Ом).

Перемички (червона та чорна).[9]

3. Підключення апаратури
З’єднайте компоненти відповідно до схеми:

Підключіть анод світлодіода до цифрового порту 13 через резистор.

Катод підключіть до GND на платі.

Реалізація коду
Створіть файл johnny-five.js і напишіть наступний код:

```
const five = require("johnny-five");
const board = new five.Board();

board.on("ready", () => {
   const led = new five.Led(13);
   led.blink(500); // Світлодіод блимає кожні 500 мс
   console.log("LED blinking every 500 ms");
});
```

Запустіть програму командою:

```
node johnny-five.js
```

У результаті світлодіод на 13-му піні почне блимати з інтервалом у 500 мілісекунд.

Створення веб-інтерфейсу для керування світлодіодом
Далі ми розглянемо приклад створення веб-інтерфейсу для регулювання кольору RGB-світлодіода за допомогою повзунків.

Необхідні компоненти
RGB-світлодіод із загальним катодом.

-Резистори: 220 Ом (x2), 330 Ом (x1).
-Перемички червоного, зеленого, синього кольорів.

-Прототипна плата.
-З’єднайте всі компоненти згідно зі схемою:
-Аноди RGB-світлодіода підключіть до цифрових портів 9, 10, 11 через резистори.
-Катод підключіть до GND.

Створення HTML-файлу

У файлі index.html створіть повзунки для керування інтенсивністю кольорів:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>RGB Controller</title>
</head>
<body>
   <h1>RGB LED Controller</h1>
   <label for="red">Red</label>
   <input id="red" type="range" min="0" max="255" value="0" />
   <label for="green">Green</label>
   <input id="green" type="range" min="0" max="255" value="0" />
   <label for="blue">Blue</label>
   <input id="blue" type="range" min="0" max="255" value="0" />
   <script src="/socket.io/socket.io.js"></script>
   <script>
      const socket = io();
      document.querySelectorAll('input').forEach(slider => {
         slider.addEventListener('input', () => {
            const colors = {
               red: document.getElementById('red').value,
               green: document.getElementById('green').value,
               blue: document.getElementById('blue').value,
            };
            socket.emit('colorChange', colors);
         });
      });
   </script>
</body>
</html>
```

Серверний код[10]
Створіть файл rgb-controller.js:

```
const five = require("johnny-five");
const express = require("express");
const http = require("http");
const socketIo = require("socket.io");

const board = new five.Board();
const app = express();
const server = http.createServer(app);
const io = socketIo(server);

app.use(express.static(__dirname));

board.on("ready", () => {
   const red = new five.Led(9);
   const green = new five.Led(10);
   const blue = new five.Led(11);

   io.on("connection", (socket) => {
      console.log("User connected");
      socket.on("colorChange", (colors) => {
         red.brightness(colors.red);
         green.brightness(colors.green);
         blue.brightness(colors.blue);
      });
   });
});

server.listen(3000, () => {
   console.log("Server running at http://localhost:3000");
});
```

Запуск програми

1. Запустіть сервер:

```
node rgb-controller.js
```

2. Відкрийте браузер за адресою http://localhost:3000.
3. Використовуйте повзунки для зміни кольору RGB-світлодіода в реальному часі.

Цей приклад демонструє, як за допомогою бібліотеки Johnny-Five можна створити простий веб-інтерфейс для керування RGB-світлодіодом.

Архітектура бібліотеки Johnny-Five[11]

Бібліотека Johnny-Five для Node.js є потужним інструментом для створення проектів у галузі робототехніки, автоматизації та взаємодії з апаратними компонентами, такими як датчики, серводвигуни, світлодіоди та багато іншого, через платформи, зокрема Arduino. Вона надає високорівневу абстракцію для взаємодії з апаратними пристроями, дозволяючи програмістам використовувати JavaScript для контролю і керування фізичними елементами.

Основні компоненти архітектури Johnny-Five

1. Платформи:

Основним елементом бібліотеки є підтримка різних платформ мікроконтролерів, таких як Arduino, Raspberry Pi, Intel Galileo та інші.
Кожна платформа реалізується окремо, що дозволяє користувачам працювати з різними типами апаратних засобів, використовуючи однаковий синтаксис.

2. Інтерфейси пристроїв:

В Johnny-Five реалізовано абстракцію для роботи з різними типами апаратних пристроїв, таких як світлодіоди, датчики, мотори, серводвигуни та інші.Для кожного пристрою (наприклад, Led, Servo, Temperature, Button) є спеціальний клас, який інкапсулює логіку взаємодії з конкретним пристроєм.

3. Інтерфейс для підключення та керування пристроєм:

Бібліотека створює зручні API для роботи з платформами. Користувач може ініціалізувати підключення до пристрою через команду new Board() і отримати доступ до інтерфейсів пристроїв через методи, які дозволяють налаштовувати та керувати ними.

Приклад ініціалізації підключення до Arduino: var five = require("johnny-five");

```
var board = new five.Board();

board.on("ready", function() {
  var led = new five.Led(13); // Створюємо об'єкт для управління світлодіодом
  led.on(); // Включаємо світлодіод
});
```

4. Обробка подій:

Як і більшість бібліотек для Node.js, Johnny-Five підтримує механізм подій. Події можуть бути використані для отримання зворотного зв'язку від пристроїв або для реагування на зміни стану.

Наприклад, при натисканні кнопки або зміні показань датчика температури бібліотека може генерувати події, які можна обробляти в коді: var button = new five.Button(2); // Кнопка підключена до піну 2

```
button.on("press", function() {
  console.log("Кнопка натиснута");
});
```

5. Модуль керування мотором:

Для роботи з різними типами моторів (DC, серводвигуни, крокові мотори) в Johnny-Five реалізовано спеціалізовані класи. Кожен клас забезпечує необхідні методи для керування швидкістю, напрямком обертання або кутом повороту.

Приклад роботи з серводвигуном: var servo = new five.Servo(10); // Підключення серводвигуна до піну 10

```
servo.to(90); // Встановлюємо кут повороту 90 градусів
```

6. Інтерфейс команд:

Johnny-Five надає можливість створення складніших проектів через використання команд. Команди в Johnny-Five можуть бути пов'язані з подіями або можуть бути використані для виконання складних операцій, таких як вимірювання температури, освітленості або автоматичне регулювання параметрів через взаємодію з іншими пристроями.

7. Асинхронність і обробка затримок:

Всі операції з Johnny-Five (чтення сенсорів, включення/вимкнення пристроїв) виконуються асинхронно, що дозволяє ефективно керувати декількома пристроями одночасно без блокування потоку.

Наприклад, можна одночасно вимірювати температуру та керувати світлодіодами, при цьому обробка кожної операції не блокуватиме інші завдання.

Підтримка інтеграції з іншими бібліотеками:

Johnny-Five підтримує інтеграцію з іншими бібліотеками для розширення функціональності. Наприклад, можна використовувати бібліотеки для роботи з інтерфейсами Bluetooth або Wi-Fi, що дозволяє створювати бездротові проекти.

Модулі бібліотеки

Board: Відповідає за підключення та ініціалізацію платформи, на якій працює програма.

Led: Клас для керування світлодіодами, який дозволяє вмикати/вимикати світлодіод, змінювати його яскравість тощо.
Servo: Модуль для управління серводвигунами, який дозволяє змінювати їх кут обертання.
Button: Клас для роботи з кнопками, включаючи події натискання.
Sensor: Базовий клас для роботи з різними типами сенсорів, такими як температурні, вологостійкі, освітленості тощо.
Motor: Клас для керування різними типами моторів, включаючи DC мотори, крокові мотори та серводвигуни.
Temperature: Клас для роботи з температурними датчиками, такими як LM35 або DHT11.
Інтеграція з іншими інструментами
Бібліотека Johnny-Five також підтримує інтеграцію з іншими інструментами для створення складніших проектів. Наприклад:
Johnny-Five може бути поєднана з бібліотекою Socket.IO для створення реального часу взаємодії з веб-інтерфейсами.
Для створення більш складних автоматизованих систем Johnny-Five може працювати разом з серверами, такими як Express, що дозволяє створювати веб-додатки для управління роботами або іншими проектами на базі мікроконтролерів.[12]
Загальний підхід
Основною перевагою архітектури бібліотеки Johnny-Five є її висока абстракція, яка дозволяє користувачам швидко створювати прототипи та проекти за допомогою простого JavaScript-коду. Бібліотека забезпечує простоту використання, що робить її доступною навіть для новачків у галузі робототехніки, і водночас потужність, що дозволяє розробляти складні системи та пристрої.

Висновки до розділу 3

У цьому розділі було здійснено огляд кількох бібліотек, що забезпечують можливість керування роботами на платформі Arduino. Кожну бібліотеку було розглянуто з точки зору її функціональних можливостей, зокрема, як вона дозволяє програмно взаємодіяти з апаратними пристроями, підключеними до мікроконтролера. У рамках дослідження для кожної бібліотеки було створено програму, яка демонструє базові функції, зокрема, управління світлодіодами та іншими пристроями, відображення в консоль інформації про стан з'єднання, а також взаємодію з підключеними модулями.
Крім того, на прикладі скріншотів було продемонстровано реальні зміни в поведінці світлодіода, що підтверджує правильність налаштувань і функціонування обраних бібліотек. Ці програми дозволяють зрозуміти, як працюють основні функції кожної бібліотеки, та як вони можуть бути використані для створення складніших систем управління роботом.
Особливо хочу відзначити бібліотеку Johnny-Five, яка, на мою думку, виявилася найефективнішою серед розглянутих варіантів. Вона відзначається простим інтерфейсом і легкістю у використанні, що є важливим аспектом для розробників, які прагнуть швидко впроваджувати свої ідеї в реальність. Її перевагою є не лише активна спільнота та постійне вдосконалення коду, але й здатність працювати з широким спектром мікроконтролерів, що робить її універсальним інструментом для реалізації різноманітних проектів у робототехніці та автоматизації.
Таким чином, використання бібліотеки Johnny-Five дозволяє не лише ефективно управляти роботами на платформі Arduino, а й розширювати можливості інтеграції з іншими апаратними платформами. Це робить її оптимальним вибором для розробників, які шукають рішення для побудови гнучких та масштабованих систем.

4.Алгоритми управління роботизованими системами за допомогою JavaScript

У сучасному світі робототехніка займає важливе місце в індустрії, медицині, транспорті та побуті. Від ефективності управління роботизованими системами залежить успіх багатьох технологій, які полегшують наше життя та підвищують продуктивність різних галузей. Одним з ключових аспектів, що забезпечує ефективну роботу таких систем, є використання надійних алгоритмів і мов програмування. Серед популярних мов програмування, що активно застосовуються для створення алгоритмів управління роботами, варто відзначити JavaScript, який здобув велику популярність завдяки своїй гнучкості, простоті реалізації та широкій екосистемі бібліотек.

JavaScript забезпечує гнучкість і можливість легко реалізувати алгоритми для різних платформ — від простих сенсорів до складних роботизованих систем. Його універсальність робить його привабливим вибором для розробників, адже він дозволяє інтегрувати алгоритми управління безпосередньо з апаратним забезпеченням за допомогою спеціальних бібліотек, таких як Johnny-Five, що дозволяють працювати з мікроконтролерами.

Робототехніка є міждисциплінарною галуззю, яка поєднує механіку, електроніку, інформатику та інші науки. Розвиток цієї галузі значно сприяв зростанню доступності технологій для створення роботизованих систем, і сьогодні ця область не обмежується лише великими корпораціями. Завдяки прогресу в мікроконтролерах і розробці нових мов програмування, зокрема JavaScript, розробка роботів стала доступною для широкого кола ентузіастів, студентів і професіоналів.

JavaScript, який традиційно асоціюється з веб-розробкою, сьогодні знаходить нове застосування в робототехніці завдяки своїй здатності інтегруватися з апаратними платформами, такими як Arduino, Raspberry Pi, ESP32, і іншими. Це дозволяє створювати роботизовані системи, які можуть бути легко налаштовані і програмовані для виконання різноманітних задач: від автоматизації побутових процесів до вирішення складних інженерних задач у промисловості та медицині.

Завдяки розвитку нових технологій і зростанню популярності JavaScript в галузі робототехніки, цей інструмент дозволяє реалізовувати інноваційні рішення для різноманітних проектів. У цьому розділі ми розглянемо основи створення алгоритмів управління роботизованими системами за допомогою JavaScript, а також розглянемо переваги та

можливості, які ця мова програмування надає для розвитку сучасних робототехнічних технологій.

4.1. Основи управління роботизованими системами

Алгоритми управління — це набір правил і інструкцій, які визначають, як робот аналізує інформацію, приймає рішення та виконує дії. Вони є центральною частиною програмного забезпечення роботизованої системи.

Основні функції алгоритмів:

1. Інтерпретація даних із сенсорів
   Отримані дані обробляються алгоритмами для перетворення "сирої" інформації в зрозумілі команди. Наприклад, алгоритм аналізує показники датчика відстані та визначає, чи є небезпека зіткнення.
2. Прийняття рішень
   На основі проаналізованих даних алгоритм обирає відповідну дію. Це може бути ухилення від перешкоди, зміна напрямку руху чи зупинка робота.
3. Керування механізмами робота
   Алгоритми генерують команди для приводів, які виконують фізичні дії. Наприклад, серводвигуни повертаються під певним кутом для захоплення об'єкта.
4. Адаптація до змін у середовищі
   Складні алгоритми дозволяють роботам адаптуватися до динамічного середовища, використовуючи машинне навчання або штучний інтелект. Такі системи здатні навчатися на основі попереднього досвіду, покращуючи свою продуктивність з часом.

Типи алгоритмів управління:

Прості алгоритми
Вони працюють за принципом заздалегідь визначених інструкцій, де всі можливі дії прописані заздалегідь. Наприклад, якщо сенсор виявляє перешкоду на відстані 30 см, робот зупиняється.

Складні алгоритми
Включають системи штучного інтелекту, які дозволяють роботу приймати рішення на основі аналізу великих обсягів даних, прогнозів або навіть змінювати свою поведінку відповідно до нових обставин.

Використання алгоритмів у JavaScript

JavaScript часто застосовується для розробки подієво-орієнтованих алгоритмів, які ефективно обробляють сигнали в реальному часі. Наприклад:

Взаємодія з датчиками через веб-інтерфейси.

Реакція на події, такі як надходження даних із сенсорів чи активація певного механізму.

Роботизована система — це не просто сукупність окремих компонентів, а складна інтегрована структура, яка функціонує завдяки взаємодії між апаратною та програмною складовою. Кожен елемент цієї системи, від сенсорів і виконавчих механізмів до мікроконтролерів і процесорів, виконує свою роль, однак їх ефективна робота можливе лише за умови правильно розроблених алгоритмів управління.

Алгоритми керування є ключовим елементом, який перетворює набір апаратних пристроїв у функціональний і автономний механізм. Вони визначають, як саме робот буде взаємодіяти з навколишнім середовищем, реагувати на зміни та виконувати поставлені завдання. Алгоритми контролю забезпечують обробку даних від сенсорів, прийняття рішень на основі отриманих відомостей та керування рухами або іншими функціями механізмів, таких як маніпулятори, мотори, камери тощо.

У сучасних роботизованих системах програмне забезпечення часто взаємодіє з апаратною частиною за допомогою спеціальних бібліотек та інтерфейсів. Це дозволяє розробникам створювати гнучкі та адаптивні алгоритми, які можуть бути легко змінені для адаптації до різних умов роботи та завдань. Програмні рішення можуть включати базові операції, такі як обробка даних з сенсорів, або складніші методи, такі як машинне навчання, що дозволяють роботам оптимізувати свою поведінку на основі досвіду.

Алгоритми керування роблять систему автономною, здатною діяти без постійного втручання людини. Вони дозволяють роботам реагувати на зміни в навколишньому середовищі, обробляти сенсорні дані в реальному часі, приймати оптимальні рішення та виконувати різноманітні завдання. У результаті, роботизована система стає потужним інструментом, здатним виконувати складні операції, від простих автоматизованих дій до складних, багатозадачних процесів у реальному часі.

4.2 Переваги використання JavaScript

JavaScript — це одна з найпопулярніших мов програмування, яка спершу асоціювалася виключно з веб-розробкою, але з часом знайшла застосування у багатьох інших галузях. Однією з таких галузей стала робототехніка, де JavaScript завдяки своїм унікальним властивостям та розвиненій екосистемі отримала значну популярність. Це стало можливим завдяки таким бібліотекам і фреймворкам, як Johnny-Five, Cylon.js та Nodebots, які дозволяють взаємодіяти з апаратним забезпеченням, створюючи алгоритми управління для роботів.

Однією з ключових переваг JavaScript є її платформонезалежність, тобто можливість запуску коду на широкому спектрі пристроїв. Це робить мову особливо привабливою для робототехнічних систем, які можуть працювати на різноманітних апаратних платформах, від простих мікроконтролерів до потужних комп’ютерів. Подієва модель програмування, властива JavaScript, дозволяє ефективно обробляти сигнали сенсорів і керувати приводами в реальному часі, що є критично важливим у робототехніці. Крім того, екосистема JavaScript включає величезну кількість бібліотек, модулів та інструментів, які полегшують інтеграцію з різними типами апаратного забезпечення.

JavaScript також має низький поріг входу, що дозволяє навіть новачкам швидко почати створювати проекти в галузі робототехніки. Простота навчання мови поєднується з її потужними можливостями, що робить її ідеальним вибором для швидкого прототипування. Завдяки цьому розробники можуть швидко створювати та тестувати алгоритми, витрачаючи мінімум часу на початкову підготовку. У результаті JavaScript стає не лише інструментом для професіоналів, а й відправною точкою для тих, хто тільки починає знайомство з програмуванням у сфері робототехніки.

Розробка робототехнічних систем починається з фундаментального етапу — постановки задачі. Саме цей крок визначає, наскільки успішним буде процес створення алгоритмів і роботи системи в цілому. Постановка задачі дозволяє точно сформулювати, які функціональні можливості потрібні, якими мають бути вимоги до системи та які обмеження необхідно врахувати. Наприклад, якщо робот повинен автономно пересуватися в складному середовищі, важливо врахувати можливі перешкоди, типи поверхонь, а також інші аспекти, які можуть впливати на його здатність до маневрування. Якщо завдання полягає у виконанні маніпуляцій з об'єктами, необхідно передбачити, які саме предмети робот буде переміщувати, їх вагу, форму, розміри та матеріал.

Формулювання основної мети є першим і надзвичайно важливим етапом, адже воно визначає, що саме повинен робити робот. Мета має бути сформульована чітко і лаконічно, щоб уникнути неоднозначності у подальших етапах розробки. Робот може бути призначений для виконання рутинних операцій, таких як сортування об'єктів на виробничій лінії, або складних завдань, таких як рятувальні місії чи розвідка в небезпечних умовах. Наприклад, робот може використовувати датчики і камери для аналізу навколишнього середовища, приймати рішення на основі зібраних даних і передавати цю інформацію у вигляді звітів або команд до центральної системи.

Деталізація задачі дозволяє розбити складне завдання на менші підзадачі, що спрощує розробку системи. Наприклад, якщо робот виконує навігацію, слід враховувати, чи пересувається він у закритому приміщенні, чи на відкритій місцевості, а також чи є у середовищі сходи, вузькі проходи або інші перешкоди. Якщо робот має працювати з об'єктами, слід проаналізувати, як саме він буде їх захоплювати, переносити і встановлювати. У випадку, коли робот повинен спілкуватися з людиною, слід визначити, який тип комунікації є оптимальним — текстовий, голосовий або візуальний.

Важливим кроком є визначення вхідних даних, які потрібні для роботи алгоритмів. Наприклад, робот може отримувати інформацію з ультразвукових або інфрачервоних сенсорів для вимірювання відстані до об'єктів, дані з камер для аналізу форми та розмірів об'єктів, сигнали GPS для визначення місця розташування або біометричні показники для медичних застосувань. Ці дані проходять обробку в реальному часі, щоб забезпечити швидку і точну реакцію системи на зміни в середовищі.

Результатом роботи алгоритмів є вихідні дії, які робот повинен виконувати. Це може бути активація моторів для зміни напрямку руху, запуск маніпуляторів для виконання певних операцій, передача даних до зовнішніх систем або видача сигналів для спілкування з людиною. Вихідні дії мають бути чітко визначені, щоб оцінити їх ефективність і відповідність початковим цілям.

Розробка робототехнічних систем також потребує врахування зовнішніх умов і обмежень. Це можуть бути складні умови середовища, такі як наявність пилу, води або перепадів температур, які можуть впливати на функціонування сенсорів і механізмів. Важливо враховувати енергоспоживання, щоб забезпечити тривалу автономну роботу робота, а також фінансові обмеження, які визначають доступність технологій та матеріалів.

Моделювання задачі дозволяє створити віртуальну або фізичну модель середовища, в якому буде працювати робот. Це допомагає протестувати всі можливі сценарії роботи, визначити потенційні проблеми та внести необхідні корективи до алгоритму ще до запуску системи в реальних умовах.

Останнім, але не менш важливим кроком, є визначення критеріїв успішності. Це можуть бути показники, що оцінюють, наскільки ефективно робот виконує завдання, наприклад, точність, швидкість або надійність. Завдяки цьому можна об’єктивно оцінити, чи відповідає робот очікуванням і вимогам.

Ретельне виконання всіх цих етапів є запорукою створення ефективної та надійної робототехнічної системи. Це не лише дозволяє уникнути помилок, але й забезпечує відповідність кінцевого продукту початковим цілям і вимогам.

4.3 Вибір обладнання

JavaScript є потужним інструментом для розробки програмного забезпечення, яке взаємодіє з апаратним забезпеченням, особливо через використання Node.js. Це дозволяє створювати інтерактивні проекти, що використовують різноманітні датчики, актуатори та інші компоненти, що управляються за допомогою популярних мікроконтролерів та платформ. Одна з найпоширеніших платформ для використання з JavaScript – це Arduino. Arduino — це відкрита апаратна платформа, яка здебільшого використовується для створення прототипів електронних пристроїв. Щоб взаємодіяти з Arduino через JavaScript, зазвичай використовують бібліотеку Firmata, яка дозволяє підключати Arduino до комп'ютера і використовувати його через програмне забезпечення. Завдяки такій інтеграції можна з легкістю отримати доступ до різноманітних датчиків, сенсорів, сервоприводів та інших компонентів, що дозволяє створювати інтерактивні пристрої і системи.

Іншою популярною платформою для роботи з JavaScript є Raspberry Pi, який є маленьким, але потужним комп'ютером, здатним виконувати повноцінну операційну систему, зазвичай на базі Linux. Raspberry Pi може бути використаний як автономний обчислювальний блок для виконання складних обчислювальних задач і навіть для управління різноманітними пристроями через GPIO (General Purpose Input/Output) порти. Оскільки Raspberry Pi підтримує сучасні мови програмування, в тому числі JavaScript, це робить його відмінним вибором для проектів, що потребують потужних обчислювальних ресурсів, зокрема для створення серверів, мультимедійних рішень чи навіть IoT-платформ.

Ще однією дуже популярною платформою для IoT-проектів є ESP8266/ESP32. Ці мікроконтролери є відмінним вибором для проектів, які вимагають бездротових з'єднань, таких як Wi-Fi або Bluetooth. ESP8266 та його вдосконалена версія ESP32 дають змогу створювати малі енергоефективні пристрої, які можуть обмінюватися даними через Інтернет. Це дозволяє використовувати ці плати для розробки рішень у сфері Інтернету Речей (IoT), автоматизації будинків та створення розумних пристроїв. ESP32 має ще більші можливості, зокрема підтримку Bluetooth, що розширює його застосування в проектах, що вимагають додаткових типів з'єднань.

Всі ці платформи дозволяють легко інтегрувати JavaScript у процес розробки апаратних проектів. Це надає розробникам можливість створювати складні та інноваційні рішення, що поєднують апаратне та програмне забезпечення для виконання різноманітних завдань.

4.4 Інсталяція необхідного середовища

Для ефективної роботи з JavaScript у галузі робототехніки важливо налаштувати правильне програмне середовище, яке дозволить зручно взаємодіяти з апаратними компонентами та реалізовувати різноманітні функції. Зазвичай для цього використовуються кілька основних інструментів, які забезпечують інтеграцію між програмним забезпеченням та апаратним забезпеченням.

Першим кроком є встановлення Node.js — серверного середовища виконання JavaScript. Node.js дозволяє запускати JavaScript не тільки в браузері, а й на сервері або локальній машині, що дає змогу створювати потужні та гнучкі програми для робототехніки. Node.js включає у себе всі необхідні інструменти для взаємодії з апаратними частинами, а також підтримує велику кількість бібліотек, що робить його ідеальним вибором для таких проектів.

Наступним важливим елементом є бібліотека Johnny-Five, яка забезпечує високорівневий інтерфейс для роботи з різними типами апаратного забезпечення. Вона дозволяє програмістам писати JavaScript-код для управління такими компонентами, як серводвигуни, датчики, світлодіоди та багато іншого. Завдяки Johnny-Five можна легко керувати такими популярними платформами, як Arduino, Raspberry Pi, ESP8266 та іншими.

Для успішної інтеграції JavaScript з апаратним забезпеченням також необхідні драйвери та прошивки. Вони дозволяють зв'язувати мікроконтролери та інші пристрої з вашим комп'ютером або сервером, де працює JavaScript. Наприклад, для Arduino зазвичай використовуються спеціальні драйвери, а для підключення через Johnny-Five — бібліотека Firmata, що забезпечує зв'язок між програмним кодом і апаратними компонентами.

Щоб встановити Johnny-Five, можна скористатися менеджером пакетів npm, використовуючи наступну команду:

```
npm install johnny-five
```

Ця команда завантажить і встановить бібліотеку, що дозволить почати роботу з апаратними компонентами через JavaScript. У процесі налаштування середовища також може знадобитись встановлення додаткових бібліотек або драйверів для конкретних платформ або мікроконтролерів, з якими ви працюєте.

4.5 Структура алгоритму

Базова структура алгоритму для роботи з апаратними компонентами в контексті робототехніки зазвичай складається з кількох основних етапів, що дозволяють ефективно зібрати, обробити та використати дані з сенсорів, а також керувати виконавчими механізмами. Кожен з цих етапів є важливим для забезпечення належної роботи системи, тому їх слід враховувати при розробці будь-якого проекту.

Перший етап — ініціалізація плати та підключення сенсорів. Це початковий крок, на якому налаштовуються всі компоненти для подальшої роботи. Підключення сенсорів до мікроконтролера або плати дозволяє здійснити з'єднання між апаратними компонентами та програмним забезпеченням. У цей момент відбувається також ініціалізація всіх необхідних бібліотек і драйверів, а також перевірка працездатності пристроїв.

Наступним етапом є зчитування даних із сенсорів. Після підключення та ініціалізації сенсори починають передавати дані на мікроконтролер або сервер. Цей етап передбачає отримання даних про стан середовища (температура, вологість, освітленість, рух і т. д.), які згодом будуть використовуватися для прийняття рішень.

Після цього йде етап обробки даних і прийняття рішень. Отримані від сенсорів дані потребують аналізу, щоб зрозуміти, що вони означають і як повинна діяти система. Наприклад, якщо температура в кімнаті перевищує певний поріг, система може прийняти рішення включити охолоджуючий пристрій. Тут можуть використовуватись алгоритми для фільтрації, порівняння значень або застосування складних логічних умов.

Останнім етапом є управління виконавчими механізмами. Після обробки даних система повинна діяти відповідно до прийнятих рішень. Це може включати в себе вмикання або вимикання пристроїв (моторів, світлодіодів, сервоприводів), зміну положення механізмів, активацію звукових чи візуальних сигналів тощо. Цей етап забезпечує фізичне реагування системи на змінені умови навколишнього середовища або команди від користувача.

Ці чотири етапи формують базову структуру алгоритму, який можна адаптувати під конкретні вимоги проекту, залежно від апаратних засобів та цілей, що ставляться перед системою.

4.6 Реалізація алгоритму

Приклад 1: Зчитування даних із сенсора
Нижче наведено приклад коду для зчитування даних з аналогового сенсора за допомогою бібліотеки Johnny-Five в середовищі Node.js. У цьому прикладі сенсор підключено до аналогового порту A0 на платі, і дані з сенсора постійно зчитуються та виводяться в консоль.

```
const five = require("johnny-five");
const board = new five.Board();

board.on("ready", () => {
  const sensor = new five.Sensor("A0"); // Сенсор підключено до аналогового порту A0

  sensor.on("data", () => {
    console.log(`Sensor value: ${sensor.value}`);
  });
});
```

У цьому коді:

1. require("johnny-five") – підключення бібліотеки Johnny-Five, яка дозволяє працювати з різними апаратними компонентами, включаючи сенсори.
2. const board = new five.Board(); – створюється об'єкт плати, що ініціалізує взаємодію з фізичним обладнанням.
3. board.on("ready", () => { ... }); – ця частина коду виконується після того, як плата готова до роботи, тобто після того, як вона підключена та налаштована.
4. const sensor = new five.Sensor("A0"); – ініціалізація сенсора, підключеного до аналогового порту A0 на платі.
5. sensor.on("data", () => { ... }); – встановлюється обробник події, який активується кожного разу, коли сенсор зчитує нові дані. Дані сенсора виводяться в консоль за допомогою console.log(Sensor value: ${sensor.value});.

Цей код дозволяє постійно відслідковувати значення з сенсора, що підключений до аналогового порту A0, і виводити його в консоль. Це дуже корисно для тестування та моніторингу сенсорних даних у реальному часі.

Приклад 2: Управління двигуном

```
const motor = new five.Motor({
```

```
  pins: {
    pwm: 9,
    dir: 8,
  },
});

board.on("ready", () => {
  motor.forward(255); // Робот рухається вперед на максимальній швидкості

  setTimeout(() => {
    motor.stop(); // Зупинка через 5 секунд
  }, 5000);
});
```

5.3. Приклад 3: Уникнення перешкод

```
const distanceSensor = new five.Proximity({
  controller: "HCSR04",
  pin: 7,
});

board.on("ready", () => {
  const motor = new five.Motor({
    pins: {
      pwm: 9,
      dir: 8,
    },
  });

  distanceSensor.on("data", () => {
    if (distanceSensor.cm < 10) {
      console.log("Obstacle detected! Reversing...");
      motor.stop();
      setTimeout(() => {
        motor.reverse(255);
      }, 1000); // Реверс через 1 секунду
    } else {
      motor.forward(255); // Якщо перешкод немає, рухаємось вперед
    }
  });
});
```

4.7. Тестування та оптимізація

Тестування алгоритму в реальних умовах є критичним етапом, оскільки дозволяє виявити можливі проблеми, які можуть виникнути лише в реальному середовищі. Для цього слід виконати кілька важливих кроків:

1. Випробування в реальних умовах

Тестування в реальних умовах передбачає запуск алгоритму на фізичному обладнанні, що дозволяє побачити, як система працює при реальних значеннях сенсорних даних та взаємодії з іншими пристроями.

Налаштування тестового середовища: Переконайтеся, що всі сенсори та виконавчі механізми підключені та налаштовані належним чином.

Тестування при різних умовах: Використовуйте різні сценарії для перевірки алгоритму: наприклад, зміни умов навколишнього середовища (температура, вологість, освітленість), перевірка на втрати сигналу або некоректні дані від сенсорів.

Моніторинг системи: Важливо спостерігати за тим, як алгоритм реагує на реальні умови. Для цього можна використовувати системи моніторингу, які дозволяють записувати стан системи та виводити його на екран.

2. Логування для відстеження помилок

Логування є важливим інструментом для виявлення помилок та відстеження роботи алгоритму в реальному часі. Це дозволяє відслідковувати виконання кожного етапу алгоритму, виявляти помилки в обробці даних або в роботі сенсорів. Логування може допомогти швидко виявити проблеми та усунути їх.

Загальне логування: Використовуйте функції логування, наприклад console.log(), для запису важливих подій, таких як зчитування даних, обробка значень або активація виконавчих механізмів.

```
console.log("Sensor data received:", sensor.value);
```

Логування помилок: При виникненні помилок важливо записувати їх у спеціальний журнал. Наприклад, можна додати обробку винятків у коді для відслідковування непередбачених ситуацій, таких як помилки з підключенням до сенсорів або перевантаження системи.

```
try {
  // Операція, яка може викликати помилку
} catch (error) {
  console.error("Error occurred:", error);
}
```

Зберігання логів: Логування можна зберігати у файли для подальшого аналізу або для віддаленого моніторингу системи.

```
const fs = require("fs");
fs.appendFileSync("log.txt", `Sensor value: ${sensor.value}\n`);
```

Аналіз логів: Після тестування важливо проаналізувати зібрані логи для виявлення потенційних проблем або патернів, що можуть вказувати на необхідність оптимізації алгоритму.

3. Перевірка стабільності роботи

Протягом тестування важливо звертати увагу на стабільність роботи системи. Алгоритм повинен функціонувати без зависань, некоректних результатів або помилок при тривалій роботі. Для цього можна використовувати наступні підходи:

Тестування на тривалість: Залиште систему працювати на кілька годин або навіть днів, щоб перевірити її стабільність у тривалому режимі.

Стрес-тестування: Створіть умови, що можуть призвести до надмірного навантаження на систему, наприклад, підключивши додаткові сенсори або пристрої, щоб перевірити, чи витримує система високі навантаження.

Після завершення тестування проаналізуйте зібрані дані, лог-файли і результати, щоб визначити, чи відповідає алгоритм вимогам, а також чи потрібні додаткові налаштування або коригування для покращення його роботи.

Оптимізація алгоритму є важливим кроком для забезпечення ефективної і стабільної роботи системи. Це дозволяє зменшити час затримки, адаптувати систему до змінних умов і покращити точність обробки даних. Ось кілька напрямків для оптимізації:

1. Зменшення затримок у передачі даних

Затримки у передачі даних можуть суттєво впливати на швидкість реакції системи, особливо в реальних часах. Щоб зменшити ці затримки, можна застосувати кілька методів:

Асинхронне зчитування даних: Використовуйте асинхронні операції для обробки даних від сенсорів, щоб уникнути блокування програми під час виконання операцій зчитування.

```
sensor.on("data", async () => {
  await processData(sensor.value);
  console.log("Sensor data processed");
});
```

Буферизація даних: Для зменшення навантаження на систему можна використовувати буфери, які зберігають дані перед їх передачею, дозволяючи зменшити затримки при їх обробці.

Оптимізація мережевих запитів: Якщо дані передаються через мережу (наприклад, до сервера), оптимізуйте мережеві запити (зменшення кількості запитів, стиснення даних).

2. Адаптація алгоритмів до нових умов середовища

Для забезпечення стабільної роботи алгоритму в змінних умовах середовища, алгоритми повинні мати здатність до адаптації:

Динамічне калібрування сенсорів: Сенсори можуть з часом змінювати свої характеристики або стати менш точними. Впровадження механізмів автоматичного калібрування дозволить забезпечити точніші показники в умовах змінної середовища.

Наприклад, можна зчитувати середнє значення на певний період часу для згладжування випадкових коливань:

```
let sensorReadings = [];
const calibrationPeriod = 1000; // Калібрування за 1 секунду

setInterval(() => {
    let averageValue = sensorReadings.reduce((a, b) => a + b, 0) / sensorReadings.length;
  console.log("Calibrated Sensor Value:", averageValue);
  sensorReadings = []; // Очищення масиву після калібрування
}, calibrationPeriod);
```

Використання адаптивних алгоритмів: Алгоритми, що можуть змінювати свої параметри на основі умов навколишнього середовища (наприклад, адаптивні фільтри для зменшення шуму або динамічне коригування порогових значень).

3. Покращення точності обробки сенсорних даних

Точність обробки сенсорних даних має важливе значення для правильного функціонування системи. Ось кілька способів покращити точність:

Фільтрація шуму: Використовуйте фільтри для усунення випадкових або непотрібних коливань в даних. Найпоширеніші методи — це фільтри середнього значення або фільтри Калмана.

```
let smoothedValue = (sensor.value + previousValue) / 2; // Простий фільтр середнього значення
previousValue = smoothedValue;
console.log("Smoothed Sensor Value:", smoothedValue);
```

Інтерполяція даних: Для зменшення помилок, що виникають через нестабільні сенсори, можна використовувати методи інтерполяції, щоб отримати більш стабільні і точні результати.

4. Впровадження систем самонавчання для підвищення ефективності роботи

Впровадження систем самонавчання може значно підвищити ефективність роботи, дозволяючи системі адаптуватися до змінних умов і вчитися на основі минулих даних.

Машинне навчання: Впровадження моделей машинного навчання для прогнозування результатів на основі історичних даних може значно підвищити ефективність роботи. Наприклад, система може навчатися на основі отриманих даних для автоматичної корекції своїх дій.

Для простого прикладу: можна використовувати алгоритми регресії для передбачення значень сенсорів або класифікації ситуацій на основі наборів даних.

Адаптивні алгоритми: Алгоритми, що змінюються на основі аналізу попередніх помилок або реакцій на зміни середовища. Це може включати в себе налаштування порогових значень для сенсорів, зважування важливості окремих сенсорів або покращення алгоритмів прийняття рішень.

Оптимізація алгоритмів є безперервним процесом, що потребує постійної уваги до деталей. Зменшення затримок, покращення точності та адаптація до нових умов середовища є ключовими аспектами для досягнення ефективності та стабільної роботи системи. Впровадження технологій машинного навчання та самонавчання дозволяє не тільки автоматизувати процеси, але й зробити їх більш гнучкими та точними, що підвищує загальну продуктивність системи.

Висновок розділу 4

JavaScript є потужним інструментом для створення алгоритмів управління роботизованими системами. Завдяки наявності доступних бібліотек, таких як Johnny-Five, та широкій екосистемі, розробники можуть швидко створювати ефективні рішення для різних платформ, включаючи Arduino, Raspberry Pi та ESP8266/ESP32. Це дозволяє значно скоротити час розробки та забезпечити високий рівень гнучкості при проектуванні робототехнічних систем.

Серед ключових переваг JavaScript у робототехніці — його асинхронна природа, що дозволяє обробляти множину запитів та задач одночасно, не блокуючи основний потік виконання. Це особливо важливо для систем, які працюють у реальному часі та потребують швидкої реакції на зміни в навколишньому середовищі.

У майбутньому роль JavaScript у робототехніці продовжуватиме зростати завдяки її гнучкості, універсальності та здатності інтегруватися з іншими технологіями, такими як інтернет речей (IoT), машинне навчання та штучний інтелект. Це відкриває нові можливості для автоматизації, інновацій і вирішення складних задач у реальному часі, таких як адаптивне управління роботами в мінливих умовах або оптимізація роботи в умовах високих навантажень.

Таким чином, JavaScript є не лише популярним інструментом серед розробників, але й важливим елементом еволюції робототехнічних систем, забезпечуючи їх ефективність, адаптивність і масштабованість.

Висновки

У процесі дослідження були детально розглянуті три основні бібліотеки для роботи з Arduino: Serialport, Babbler_h та Johnny-Five. Кожна з них демонструє унікальні особливості, що дозволяють обирати найбільш підходяще рішення залежно від специфіки завдань, які необхідно реалізувати.

Бібліотека Serialport є базовим інструментом для роботи з послідовними портами. Вона забезпечує простий та інтуїтивний синтаксис, що дозволяє легко передавати дані між Arduino та комп'ютером. Її основною перевагою є простота використання та якісна документація, що робить її доступною навіть для новачків. Однак функціональні можливості цієї бібліотеки обмежені, особливо у випадках, коли потрібно реалізовувати складну логіку або проводити обчислення. Serialport є ідеальним вибором для задач початкового рівня, таких як тестування або базове управління пристроями.

Бібліотека Babbler_h пропонує більш розширені можливості, зокрема стабільність у роботі та автоматичну обробку помилок. Вона підтримує формат JSON для обміну даними, що полегшує інтеграцію Arduino з сучасними веб-додатками. Особливою перевагою є автоматичне перепідключення пристрою у разі розриву з'єднання. Проте ця бібліотека має певні вимоги, такі як необхідність наявності команд ping та help у прошивці пристрою, що вимагає додаткових налаштувань з боку користувача. Завдяки своїй надійності Babbler_h найкраще підходить для задач, де важлива стабільність з'єднання та швидка обробка даних.

Johnny-Five — це найгнучкіший інструмент серед розглянутих бібліотек. Вона підтримує не лише Arduino, але й інші апаратні платформи, надаючи широкі можливості для розробки робототехнічних та інтерактивних проєктів. Однією з найбільш значущих переваг є можливість створення веб-інтерфейсів для керування пристроями, що відкриває шлях до інтеграції з сучасними технологіями. Водночас ця бібліотека вимагає встановлення спеціальної прошивки (Firmata), що може бути складним для новачків. Завдяки своїй багатофункціональності Johnny-Five є оптимальним вибором для складних і творчих проєктів.

У ході практичної роботи кожна з бібліотек була протестована на різних завданнях. Serialport продемонструвала свою ефективність у задачах базового рівня, таких як виведення повідомлень та керування світлодіодом. Babbler_h відзначилася у складніших сценаріях, де була реалізована функція автоматичного перепідключення та обробки помилок. Johnny-Five стала незамінною для розробки інтерактивних проєктів, зокрема веб-інтерфейсу для керування RGB-світлодіодом.

Загалом вибір бібліотеки значною мірою залежить від вимог проєкту. Для простих задач найкраще підходить Serialport завдяки її простоті та доступності. Якщо проєкт вимагає високої надійності з'єднання, доцільно використовувати Babbler_h. Johnny-Five є ідеальним інструментом для складних і творчих застосувань, де потрібна інтеграція з веб-технологіями та підтримка широкого спектра апаратних платформ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. javascript.org.ua - JS Communities [Electronic resource] URL:https://javascript.org.ua/agentic-ai-rozuminnya-avtonomnogo-intelektu-arhitekturi-ta-realnih-zastosuvan/
2.Introduction to JavaScript Robotics - Suz Hinton [Electronic resource] URL:https://www.youtube.com/watch?v=Dv248eC9y3s&ab_channel=Nearform
3.Андрій Креневич – Алгоритми і структури даних. Джерело: Підручник, присвячений вивченню алгоритмів і структур даних у програмуванні.
4.JavaScript [Electronic resource] URL:https://uk.wikipedia.org/wiki/
5.ЕЛЕКТРОННИЙ НАВЧАЛЬНИЙ ПОСІБНИК [Electronic resource] URL:https://gamehub-cbhe.deusto.es/wp-content/uploads/2018/10/book_1.pdf
6.Мова програмування JavaScript https [Electronic resource] URL://uk.javascript.info/date
7.Java [Electronic resource] URL:https://uk.wikipedia.org/wiki/JavaScript
8. Що таке JavaScript [Electronic resource] URL:https://dou.ua/lenta/articles/how-to-learn-javascript/
9. Мова програмування JavaScript [Electronic resource] URL:https://uk.javascript.info/
10.Run JavaScript Everywhere [Electronic resource] URL:https://nodejs.org/en
11. Редактори коду [Electronic resource] URL:https://nodejs.org/en/download/current
12. r_d courses [Electronic resource] URL:https://robotdreams.cc/uk/course/
13. Node.js Introduction [Electronic resource] URL:https://www.w3schools.com/nodejs/nodejs_intro.asp

Демонстраційні матеріали

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО - КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Алгоритм управління роботизованими системами на мові JavaScript

Виконав: студент групи ІСДМ-61 Фадієнко В.С.

Науковий керівник: Данильченко В. М.

Київ - 2024

Мета та методи дослідження

Об'єкт дослідження - бібліотеки Serialport, Babbler_h та Johnny-Five, які використовуються для роботи з Arduino та іншими апаратними платформами.

Мета роботи - аналіз трьох основних бібліотек для роботи з Arduino (Serialport, Babbler_h та Johnny-Five) з точки зору їх функціональності, зручності використання та відповідності до різних категорій задач. Це дозволить визначити, яка з бібліотек найкраще підходить для різних типів проєктів.

Методи дослідження - аналіз документації для вивчення особливостей бібліотек, експериментальний підхід для тестування їх у реальних проєктах і порівняльний аналіз для визначення переваг та недоліків кожної бібліотеки.

Актуальність роботи

Робота є актуальною, оскільки Arduino та сумісні платформи стають все більш популярними для створення проєктів у галузях освіти, розробки прототипів і IoT. Різноманіття доступних бібліотек створює потребу у чіткому розумінні їх функціональності та обмежень, щоб обрати оптимальний інструмент для реалізації конкретних задач. Аналіз та порівняння бібліотек дозволяє заощадити час розробників і підвищити ефективність створення проєктів.

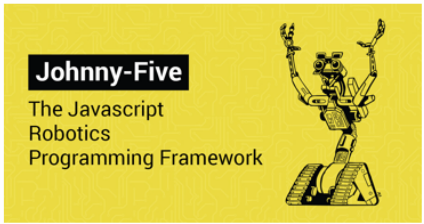
Основні бібліотеки

Node.js

Serialport

Babbler_h

Johnny-Five



5

Типи плат Arduino

1. **Arduino Uno**

- Найпопулярніша модель.
- Базується на мікроконтролері ATmega328P.
- 14 цифрових входів/виходів (6 з них — PWM).
- Простота використання, ідеальна для початківців.

2. **Arduino Nano**

- Компактна версія для проєктів з обмеженим простором.
- Має функціональність, схожу з Arduino Uno.

3. **Arduino Mega**

- 54 цифрових входи/виходи, 16 аналогових входів.
- Ідеально підходить для роботи з великими даними чи кількома пристроями.

4. **Arduino Leonardo**

- Підтримує емуляцію комп'ютерних пристроїв (клавіатури, миші).
- Базується на мікроконтролері ATmega32u4.

6

Основні компоненти

1. Arduino (Апартна частина)
   - Мікроконтролерна плата для управління робототехнічними пристроями.
   - Популярні моделі: Arduino Uno, Arduino Mega.
2. JavaScript та Node.js
   - Програмна платформа для асинхронного управління апаратними компонентами.
   - Використання бібліотек, таких як Johnny-Five
3. Апаратні компоненти робота
   - Сенсори (датчики температури, ультразвукові, світлові тощо).
   - Виконавчі механізми (моторчики, серводвигуни).
   - Елементи живлення (батареї, адаптери).
4. Програмне забезпечення
   - Node.js для програмування логіки роботи.
5. Комунікація між Arduino та ПК
   - Протоколи: Serial (послідовний порт)
- Підключення через USB

Основні компоненти

1. Arduino (Апартна частина)
   - Мікроконтролерна плата для управління робототехнічними пристроями.
   - Популярні моделі: Arduino Uno, Arduino Mega.
2. JavaScript та Node.js
   - Програмна платформа для асинхронного управління апаратними компонентами.
   - Використання бібліотек, таких як Johnny-Five
3. Апаратні компоненти робота
   - Сенсори (датчики температури, ультразвукові, світлові тощо).
   - Виконавчі механізми (моторчики, серводвигуни).
   - Елементи живлення (батареї, адаптери).
4. Програмне забезпечення
   - Node.js для програмування логіки роботи.
5. Комунікація між Arduino та ПК
   - Протоколи: Serial (послідовний порт)
- Підключення через USB

Висновки

Вибір бібліотеки для роботи з Arduino та Node.js залежить від завдань проєкту. Serialport ідеально підходить для простих рішень завдяки своїй доступності та легкості використання. Babbler_h забезпечує надійність, автоматичне перепідключення та інтеграцію з веб-додатками, що робить її оптимальною для складніших завдань. Johnny-Five є універсальним інструментом для розробки інтерактивних і складних робототехнічних проєктів із підтримкою веб-технологій

Дякую за увагу!