

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

Навчально-науковий інститут математики та інформаційних технологій

Кафедра інформаційних технологій та систем

**Плотнікова Людмила Петрівна**

**АВТОМАТИЗОВАНА СИСТЕМА КЕРУВАННЯ МОДУЛЯМИ  
РОЗУМНОГО БУДИНКУ НА БАЗІ ARDUINO**

**кваліфікаційна робота**

**здобувача вищої освіти другого (магістерського) рівня**

**освітньої програми «Мультимедійні системи»**

**за спеціальністю 121 Інженерія програмного забезпечення**

Особистий підпис \_\_\_\_\_ Людмила ПЛОТНІКОВА

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

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

Полтава – 2025

## АНОТАЦІЯ

**Плотнікова Л. П.**

**Тема:** Автоматизована система керування модулями розумного будинку на базі ARDUINO.

**Спеціальність:** 121 «Інженерія програмного забезпечення».

**Установа:** ЛНУ імені Тараса Шевченка, 2025р.

**Магістерська робота містить:** 70 с., 30 рис., 2 табл., 3 додатки, 29 джерел.

**Об'єкт дослідження:** системи автоматизації житлового приміщення на основі мікроконтролерів, зокрема системи «розумний будинок».

**Предмет дослідження:** процес розробки, реалізації та оцінки ефективності системи "розумний будинок" на базі мікроконтролера Arduino, включаючи створення клієнт-серверної архітектури для управління системою.

**Мета дослідження:** аналіз та розробка автоматизована система керування модулями розумного будинку на базі ARDUINO, а також створення клієнт-серверної архітектури, у якій завдання або мережеве навантаження розподілені між постачальниками послуг, серверами, та замовниками послуг – клієнтами.

**Результати роботи.** В ході даного дослідження було розглянуто різні аспекти проектування та розробки системи "розумний будинок", починаючи з огляду різних типів датчиків та мікроконтролерів, що можуть бути використані, і закінчуючи описом роботи готової системи. Дослідження різних типів датчиків та мікроконтролерів дозволило визначити, які компоненти найкраще підходять для задач "розумного будинку", враховуючи їх специфікації та можливості. Серед них, Arduino UNO, PIR-датчик та DHT11 були вибрані як ключові елементи для створення системи. Було також розглянуто різні середовища розробки, зокрема Arduino IDE, Proteus 8 Professional, Visual Studio та Virtual Null Modem. Використання цих інструментів дозволило ефективно розробляти, моделювати та тестувати програмне забезпечення для системи. Ключовим етапом стала розробка програмного забезпечення для Arduino IDE, а також клієнтської та серверної програм. Розробка коду для кожної частини системи була ретельно виконана, що дозволило забезпечити надійну роботу системи. Система "розумний будинок" дозволяє автоматизувати та контролювати різні аспекти домогосподарства, включаючи освітлення, температуру, безпеку та вологість.

**Ключові слова:** МІКРОКОНТРОЛЕР, АВТОМАТИЗОВАНА СИСТЕМА, ТЕМПЕРАТУРНИЙ КОНТРОЛЬ, ДИСТАНЦІЙНЕ КЕРУВАННЯ, МАКЕТ ПРИСТРОЮ, ДАТЧИКИ, ARDUINO UNO, IoT, СИСТЕМА РОЗУМНИЙ БУДИНОК.

## ANNOTATION

**Plotnikova Liudmyla**

**Theme:** Автоматизована система керування модулями розумного будинку на базі ARDUINO.

**Speciality:** 121 "Software Engineering".

**Institution:** Luhansk Taras Shevchenko National University (LTSNU), 2025 year.

**Bachelor work of:** 70 p., 30 im, 2 table, 3 ap., 29 sources.

**A research object of:** microcontroller-based home automation systems, in particular smart home systems..

**The article of research-** the process of developing, implementing and evaluating the effectiveness of a "smart home" system based on the Arduino microcontroller, including the creation of a client-server architecture for system management.

**An aim of research is** - analysis and development of an automated smart home module control system based on ARDUINO, as well as the creation of a client-server architecture in which tasks or network load are distributed between service providers, servers, and service customers - clients.

**Job performanes.-** This study examined various aspects of designing and developing a smart home system, starting with an overview of the different types of sensors and microcontrollers that can be used and ending with a description of the operation of the finished system. The study of different types of sensors and microcontrollers allowed us to determine which components are best suited for the tasks of a smart home, taking into account their specifications and capabilities. Among them, Arduino UNO, PIR sensor and DHT11 were selected as key elements for creating the system. Various development environments were also considered, including Arduino IDE, Proteus 8 Professional, Visual Studio and Virtual Null Modem. The use of these tools allowed us to effectively develop, model and test the software for the system. A key stage was the development of the software for Arduino IDE, as well as client and server applications. The development of the code for each part of the system was carefully carried out, which allowed us to ensure reliable operation of the system. The smart home system allows us to automate and control various aspects of the household, including lighting, temperature, security and humidity.

**Keywords:** MICROCONTROLLER, AUTOMATED SYSTEM, TEMPERATURE CONTROL, REMOTE CONTROL, DEVICE DESIGN, SENSORS, ARDUINO UNO, IoT, SMART HOME SYSTEM.

## ЗМІСТ

<b>ВСТУП .....</b>	<b>6</b>
<b>РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ СИСТЕМ «РОЗУМНИЙ БУДИНОК» .....</b>	<b>8</b>
1.1. Огляд різних "систем розумний будинок" .....	8
1.2. Мікроконтролери в системах "розумний будинок": огляд та застосування .....	14
1.3. Огляд датчиків інтелектуальних систем управління будинком .....	20
Висновки до розділу .....	24
<b>РОЗДІЛ 2. ВИБІР ІНСТРУМЕНТІВ ДЛЯ "СИСТЕМИ РОЗУМНИЙ БУДИНОК" НА ОСНОВІ МІКРОКОНТРОЛЕРА ARDUINO.....</b>	<b>25</b>
2.1. Огляд основних компонентів системи .....	25
2.1.1. Огляд Arduino UNO .....	25
2.1.2. Опис послідовного порту .....	30
2.1.3. Опис PIR-датчика .....	32
2.1.4. Опис DHT11 .....	34
2.2. Огляд середовищ розробки для програмування мікроконтролерів .....	36
2.2.1. Огляд Arduino IDE .....	36
2.2.2. Огляд Proteus 8 Professional .....	41
2.2.3. Огляд Visual Studio .....	43
2.2.4. Огляд Virtual Null Morem.....	45
Висновки до розділу .....	46
<b>РОЗДІЛ 3. МОДЕЛЮВАННЯ СИСТЕМИ "РОЗУМНИЙ ДІМ" НА ПЛАТФОРМІ ARDUINO ЗА ДОПОМОГОЮ PROTEUS.....</b>	<b>48</b>
3.1. Моделювання системи на основі мікроконтролера Arduino .....	48
3.2. Розробка програмного забезпечення для Arduino IDE .....	49
3.3. Опис програмного коду Arduino UNO .....	52
3.4. Опис коду клієнтської програми.....	55
3.5 Опис коду серверної програми.....	58
3.6. Робота системи розумний будинок.....	62

Висновки до розділу .....	65
<b>ВИСНОВКИ.....</b>	<b>67</b>
<b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....</b>	<b>69</b>
<b>ДОДАТОК А .....</b>	<b>72</b>
<b>ДОДАТОК Б.....</b>	<b>74</b>
<b>ДОДАТОК В.....</b>	<b>79</b>

## ВСТУП

Сучасний спосіб життя характеризується зростаючою залежністю від технологій, що висуває нові вимоги до комфорту та безпеки житлових приміщень. Існуючі системи домашньої автоматизації часто мають обмежений функціонал або є складними в налаштуванні.

Розробка системи "розумний будинок" на базі мікроконтролера Arduino є актуальним напрямом дослідження з огляду на:

- об'єднання різних систем домашнього управління в єдину, інтуїтивно зрозумілу платформу.
- оптимізація споживання енергії за рахунок автоматизації процесів опалення, освітлення та інших систем.
- підвищення рівня безпеки житла завдяки функціям контролю доступу, моніторингу стану приміщення тощо.
- створення комфортного житлового середовища, яке адаптується до потреб мешканців.
- впровадження нових технологій та інноваційних рішень в галузі автоматизації житла.

**Об'єкт дослідження:** системи автоматизації житлового приміщення на основі мікроконтролерів, зокрема системи "розумний будинок".

**Предмет дослідження:** процес розробки, реалізації та оцінки ефективності системи "розумний будинок" на базі мікроконтролера Arduino, включаючи створення клієнт-серверної архітектури для управління системою.

**Мета дослідження:** аналіз та розробка автоматизована система керування модулями розумного будинку на базі ARDUINO, а також створення клієнт-серверної архітектури, у якій завдання або мережеве навантаження розподілені між постачальниками послуг, серверами, та замовниками послуг – клієнтами.

Досягнення зазначеної мети передбачає вирішення таких основних завдань:

1. Провести аналіз потенційні системи розумний будинок на основі різних мікроконтролерів.

2. Обґрунтувати вибір та побудувати ефективну систему розумний будинок на основі мікроконтролера Arduino.
3. Розробити програмне забезпечення для мікроконтролера Arduino.
4. Розробити та реалізувати серверну та клієнтську системи обробки.
5. Розробити практичні рекомендації по вдосконаленню та розширенню функціональностей системи розумного будинка на основі мікроконтролера Arduino на майбутнє.

**Методи дослідження:** теоретичний аналіз літературних джерел, моделювання роботи системи, експериментальні дослідження, аналіз даних та узагальнення результатів.

Результати проведеного дослідження мають значний потенціал для практичного застосування. Отримані дані та розроблені алгоритми можуть бути використані для підвищення функціональності, надійності та енергоефективності існуючих систем "розумний будинок". Розроблені в рамках дослідження рішення можуть слугувати основою для створення нових, більш інтелектуальних та адаптивних систем автоматизації житлових приміщень. Результати дослідження можуть сприяти популяризації технологій "розумного будинку" серед широкого кола користувачів, підвищуючи обізнаність про переваги та можливості таких систем. Розроблені заходи безпеки можуть бути використані для підвищення рівня захисту систем "розумний будинок" від несанкціонованого доступу та кібератак. Реалізація інтуїтивно зрозумілих інтерфейсів та адаптивних алгоритмів управління може значно підвищити комфорт користування системами "розумний будинок".

# РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ СИСТЕМ «РОЗУМНИЙ БУДИНОК»

## 1.1. Огляд різних "систем розумний будинок"

Сучасні технології створюють нові можливості для оптимізації життєвого простору. Одним із яскравих прикладів є системи "розумний будинок", які об'єднують різноманітні пристрої та системи в єдину інтегровану платформу, забезпечуючи автоматизацію та управління різними функціями в житловому приміщенні.

Залежно від рівня інтеграції та набору функцій, системи "розумний будинок" можна поділити на такі категорії:

*Базові системи:* забезпечують управління окремими пристроями (освітлення, опалення) та моніторинг основних параметрів (температура, вологість).

*Системи середнього рівня:* об'єднують різні підсистеми (освітлення, безпека, мультимедіа) в єдину платформу, забезпечуючи більш складні сценарії автоматизації.

*Інтелектуальні системи:* здатні до самонавчання та адаптації до змін умов навколишнього середовища та поведінки користувачів.

Сучасні системи "розумний будинок" пропонують широкий спектр функцій, які значно підвищують комфорт та безпеку проживання:

1. Управління мікрокліматом: автоматичне регулювання температури, вологості, освітлення залежно від часу доби, погоди та особистих переваг користувачів.
2. Безпека: системи відеоспостереження, датчики руху, диму, витоку газу, контролю доступу забезпечують захист житла та його мешканців.
3. Енергоефективність: оптимізація споживання енергії за рахунок автоматичного вимкнення приладів у відсутність людей, регулювання освітлення та опалення.



4. Комфорт: створення комфортного житлового середовища за допомогою інтеграції музичних систем, систем "розумний дім" можуть керувати музикою, телебаченням та іншими розважальними системами.
5. Доступність: можливість віддаленого управління системою за допомогою смартфонів, планшетів та інших мобільних пристроїв.

Реалізація функціоналу систем "розумний будинок" стала можливою завдяки розвитку таких технологій:

*Мікроконтролери:* Arduino, Raspberry Pi та інші платформи, які слугують основою для створення розумних пристроїв.

*Датчики:* температури, вологості, руху, світла, якості повітря та інші, які збирають інформацію про стан навколишнього середовища.

*Виконавчі механізми:* електроприводи, реле, клапани, які керують роботою різних пристроїв у будинку.

*Мережі:* Wi-Fi, Bluetooth, Zigbee, Ethernet, які забезпечують обмін даними між пристроями системи.

*Програмне забезпечення:* спеціалізовані програмні платформи та мови програмування для розробки та конфігурування систем "розумний будинок".

Системи "розумний будинок" відкривають нові можливості для створення комфортного, безпечного та енергоефективного житла. Постійний розвиток технологій сприяє розширенню функціоналу та зниженню вартості таких систем, що робить їх доступними для широкого кола споживачів.

Система Ajax є одним із популярних рішень на ринку систем розумного будинку. Її основні переваги полягають у:

- високій надійності. Завдяки використанню двостороннього радіозв'язку Jeweller забезпечується стабільна та захищена передача даних між пристроями системи.
- простоті установки. Система легко встановлюється без необхідності проведення складних монтажних робіт.

- гнучкості. Можливість підключення великої кількості різних датчиків та виконавчих механізмів дозволяє адаптувати систему до індивідуальних потреб користувача.
- автономності. Наявність резервного живлення у хабі забезпечує безперебійну роботу системи навіть при відключенні електроенергії.
- мобільності. Керування системою здійснюється за допомогою мобільного додатка, що дозволяє контролювати свій будинок з будь-якої точки світу.

Однак, система Ajax, як і будь-яка інша, має свої особливості:

1. Працездатність датчиків безпосередньо залежить від роботи хаба, що може обмежувати можливості системи в разі виходу з ладу центрального вузла.
2. Для реалізації функцій відеоспостереження необхідно додатково підключати IP-камери.
3. Хоча управління системою через мобільний додаток є зручним, деякі користувачі можуть віддавати перевагу традиційним способам управління.



Рис. 1.1. Розумний будинок Ajax

Заснована на передових технологіях, система Ajax пропонує користувачам широкий спектр функцій для забезпечення безпеки та комфорту в домі. Її компактний дизайн та інтуїтивно зрозумілий інтерфейс роблять систему зручною у використанні. Завдяки цим якостям, Ajax є одним з найпопулярніших рішень на ринку систем "розумний будинок" [2].

Система BroadLink представляє собою універсальне рішення для автоматизації житлового простору. На відміну від спеціалізованих систем, таких як Ajax, орієнтованих на безпеку, BroadLink пропонує ширший спектр можливостей для управління різноманітними приладами та системами в будинку.

Ключові переваги системи BroadLink:

1. Гнучкість. Система підтримує широкий спектр протоколів зв'язку (IR, RF, Wi-Fi), що дозволяє інтегрувати в неї практично будь-який побутовий прилад.
2. Автономна робота датчиків. Кожен датчик працює незалежно, що забезпечує високу надійність системи.
3. Простота установки та налаштування. Інтуїтивно зрозумілий інтерфейс мобільного додатку дозволяє легко налаштувати систему без спеціальних знань.
4. Доступна вартість. Система BroadLink є одним з більш бюджетних варіантів на ринку систем "розумний будинок".

Однак, система має і певні обмеження. Сигнал системи може діяти на відстані до 50 метрів, що може бути недостатньо для великих приміщень. Відсутність хаба може ускладнювати управління системою в цілому. Система BroadLink в основному орієнтована на автоматизацію, а не на забезпечення високого рівня безпеки.

Займаючи впевнене друге місце в рейтингу систем "розумний будинок", BroadLink є одним з найбільш популярних рішень на ринку. Незважаючи на деякі недоліки, її висока функціональність, доступність та простота використання роблять її гідним конкурентом більш дорогим системам (рис. 1.2).



Рис. 1.2. Розумний будинок BroadLink

Система "Розумний будинок" Fibaro - це високотехнологічне рішення для автоматизації та безпеки житлових приміщень, яке вимагає професійної інсталяції. Завдяки широкому спектру функцій та можливостей налаштування, система Fibaro дозволяє створити індивідуальне та комфортабельне житлове середовище (рис. 1.3).



Рис. 1.3. Розумний будинок Fibaro

Система "Розумний дім" Fibaro пропонує широкий спектр можливостей для автоматизації житлового простору. Завдяки підтримці протоколу Z-Wave і різноманітним датчикам, вона дозволяє створити індивідуальне та комфортабельне середовище. Однак, висока вартість обладнання та необхідність професійного монтажу роблять її менш доступною для широкого кола користувачів [4].

Таблиця 1.1

**Порівняльна таблиця систем "розумний будинок" Ajax, BroadLink, Fibaro**

<b>Характеристика</b>	<b>Ajax</b>	<b>BroadLink</b>	<b>Fibaro</b>
Спеціалізація	Безпека, охорона, автоматизація	Універсальна автоматизація	Комплексна автоматизація будинку
Протоколи зв'язку	Jeweller (власний), Wi-Fi	IR, RF, Wi-Fi	Z-Wave
Центральний блок	Хаб Hub	Різноманітні контролери (RM Pro, Mini, etc.)	Хаб Home Center
Датчики та актуатори	Широкий асортимент датчиків (руху, відкриття дверей, витоку води тощо), сирени, реле	Різноманітні датчики та пульти дистанційного керування для управління побутовою технікою	Широкий спектр датчиків та актуаторів для управління освітленням, опаленням, безпекою тощо
Мобільний додаток	Ajax Security System (iOS, Android)	BroadLink (iOS, Android)	Fibaro Home Center (iOS, Android)
Голосові помічники	Google Assistant, Amazon Alexa	Google Assistant, Amazon Alexa	Google Assistant, Amazon Alexa
Автономність роботи	Висока завдяки резервному живленню	Залежить від моделі пристрою	Висока завдяки вбудованому акумулятору в деяких пристроях
Вартість	Середня	Доступна	Вища
Складність встановлення	Відносно проста	Проста	Вимагає професійного монтажу

Характеристика	Ajax	BroadLink	Fibaro
Функціональність	Орієнтована на безпеку та автоматизацію	Універсальна автоматизація, управління побутовою технікою	Комплексна автоматизація будинку, інтеграція з іншими системами
Масштабованість	Легко розширюється новими пристроями	Легко розширюється новими пристроями	Можливість створення складних сценаріїв автоматизації

## 1.2. Мікроконтролери в системах "розумний будинок": огляд та застосування

Мікроконтролери є серцем сучасних систем автоматизації, зокрема, систем "розумний будинок". Ці інтегральні мікросхеми, об'єднуючи процесор, пам'ять та периферійні пристрої на одному кристалі, забезпечують гнучку платформу для створення різноманітних автоматизованих функцій.

Мікроконтролер функціонує за принципом виконання програмного коду, завантаженого в його пам'ять. Основні компоненти мікроконтролера включають:

- Центральний обчислювальний блок, який виконує логічні операції та керує роботою інших компонентів.
- Пам'ять що використовується для зберігання програмного коду та даних. Розрізняють оперативну пам'ять (RAM) для тимчасового зберігання даних та постійну пам'ять (ROM, EEPROM, Flash) для зберігання програм.
- Периферійні пристрої які забезпечують взаємодію мікроконтролера з зовнішнім світом. До них належать порти введення-виведення (GPIO), аналогово-цифрові та цифро-аналогові перетворювачі (ADC, DAC), таймери, лічильники та модулі комунікації (UART, I2C, SPI, USB, Ethernet).

В системах "розумний будинок" мікроконтролери виконують широкий спектр функцій:

1. Управління освітленням. Регулювання яскравості, кольору, включення/вимкнення освітлення за розкладом або за командами користувача.
2. Контроль температури. Регулювання роботи систем опалення та кондиціювання.
3. Забезпечення безпеки. Моніторинг датчиків руху, диму, витоку газу та управління системами безпеки.
4. Автоматизація побутових приладів. Керування роботою побутової техніки (пральні машини, посудомийні машини, кондиціонери тощо).
5. Створення мультимедійних систем. Інтеграція аудіо та відео систем, управління домашнім кінотеатром.

Вибір конкретного мікроконтролера залежить від вимог проєкту, таких як:

- Обчислювальна потужність що визначається кількістю обчислень, які необхідно виконати в одиницю часу.
- Обсяг пам'яті який залежить від розміру програми та обсягу даних, які потрібно зберігати.
- Наявність периферійних пристроїв що визначається необхідністю взаємодії з різними датчиками та виконавчими механізмами.
- Споживана потужність - важливий параметр для портативних пристроїв.
- Вартість залежить від характеристик мікроконтролера та виробника.

Популярні платформи для розробки систем "розумний будинок":

Arduino відкрита платформа, яка дозволяє легко почати розробку завдяки простому середовищу програмування та великій спільноті розробників.

Raspberry Pi більш потужна плата, яка може використовуватися для більш складних проєктів, що вимагають високої обчислювальної потужності.

ESP8266/ESP32 бюджетні мікроконтролери з вбудованим Wi-Fi модулем, що дозволяють легко підключати пристрої до Інтернету.

Arduino Uno – це одна з найпопулярніших платформ для швидкого прототипування та розробки електронних пристроїв. Її відкрита архітектура, заснована на мікроконтролері ATmega328P, забезпечує гнучкість та простоту використання, що робить її ідеальним інструментом як для новачків, так і для досвідчених розробників.

Плата Arduino Uno має модульну конструкцію, що дозволяє легко розширювати її функціональність за допомогою додаткових модулів (щитів). Основні компоненти плати включають:

- Мікроконтролер ATmega328P: центральний процесор, який виконує всі обчислення та керує роботою плати.
- Цифрові входи/виходи: використовуються для підключення різноманітних датчиків, актуаторів та інших електронних компонентів.
- Аналогові входи: дозволяють зчитувати аналогові сигнали, такі як напруга з датчиків температури, світла тощо.
- USB-інтерфейс: використовується для програмування плати та підключення до комп'ютера.
- Роз'єм живлення: дозволяє підключати зовнішнє джерело живлення.

Arduino Uno використовує спрощений мову програмування, заснований на C++, що робить його доступним для вивчення навіть для тих, хто не має досвіду програмування. Інтегроване середовище розробки (IDE) забезпечує зручний інтерфейс для написання, компіляції та завантаження програмного коду на плату [6].

Arduino Uno - це універсальна платформа, яка знаходить широке застосування в різних сферах, від створення простих роботів до розробки складних систем автоматизації. Завдяки своїй доступності, простоті використання та великій спільноті розробників, Arduino стала одним з найпопулярніших інструментів для навчання та реалізації творчих ідей в галузі електроніки (рис. 1.4).



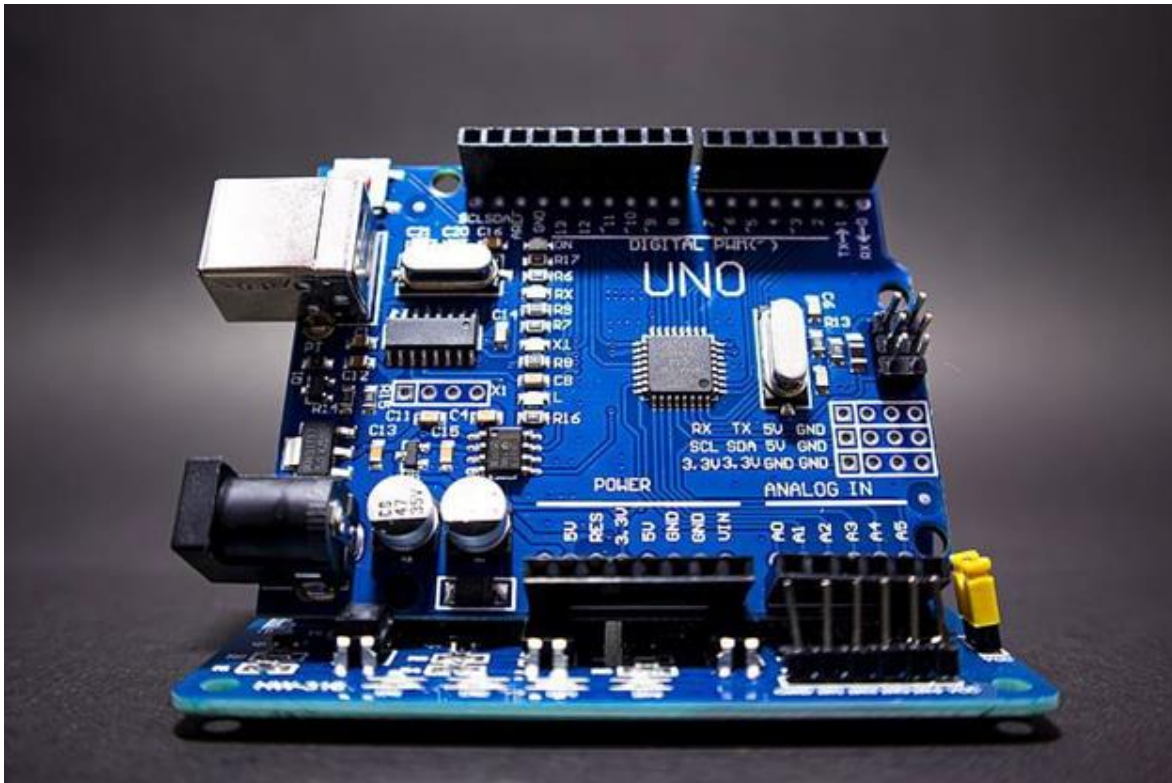


Рис. 1.4. Мікроконтролер Arduino UNO

Raspberry Pi - це універсальний міні-комп'ютер, який використовується як для навчання програмуванню, так і для створення складних систем автоматизації. Його відкрита архітектура та низька вартість роблять його доступним для широкого кола користувачів.

Raspberry Pi 4, як представник четвертого покоління популярної серії одноплатних комп'ютерів, пропонує значний потенціал для реалізації проєктів у сфері "розумного дому". Завдяки своїм технічним характеристикам та гнучкості, він здатний виконувати роль центрального контролера, об'єднуючи різноманітні пристрої та системи в єдину інтегровану платформу [7].

Ключові переваги Raspberry Pi 4 для "розумного дому". Потужний процесор та достатня оперативна пам'ять дозволяють виконувати складні обчислення та одночасно керувати великою кількістю пристроїв. Наявність Ethernet, Wi-Fi, Bluetooth та USB портів забезпечує гнучкість підключення до різних датчиків, актуаторів та інших пристроїв. Можливість встановлення різних операційних систем, таких як Raspbian, Ubuntu та Windows 10 IoT Core, дозволяє вибрати оптимальне рішення для конкретного проєкту. Наявність GPIO портів дозволяє підключати додаткові модулі та розширювати функціональність

системи. Велика та активна спільнота розробників забезпечує постійний розвиток платформи та наявність великої кількості готових рішень.

Raspberry Pi 4 перетворює ваш дім у розумний простір, надаючи безмежні можливості для автоматизації. Від управління освітленням і кліматом до забезпечення безпеки та створення розважальних систем, Raspberry Pi 4 дозволяє створити житло, яке адаптується до ваших потреб. Завдяки відкритому програмному забезпеченню та великій спільноті розробників, ви можете легко налаштувати систему під себе і розширювати її функціональність (рис. 1.5).

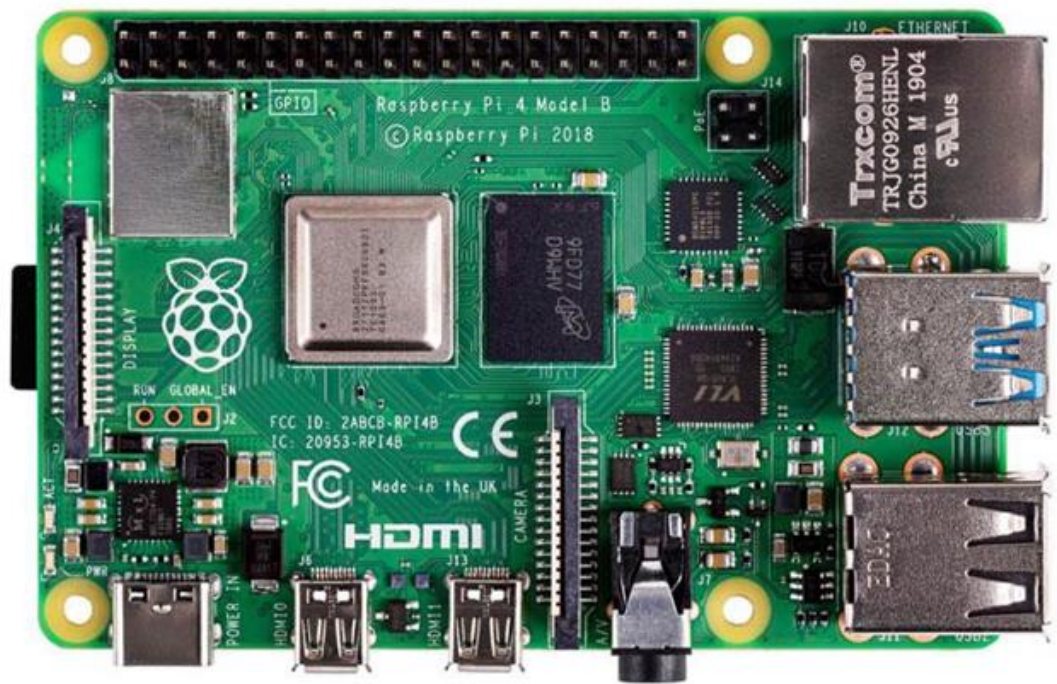


Рис. 1.5. Мікроконтролер Raspberry Pi 4

ESP8266 - це універсальний мікроконтролер з вбудованим Wi-Fi, який спрощує створення Інтернету речей [8]. Завдяки своїй гнучкості, доступності та підтримці різних платформ програмування, ESP8266 став популярним вибором для розробників, які бажають створювати інноваційні IoT-пристрої. Він може використовуватися як для додавання бездротової функціональності до існуючих пристроїв, так і для побудови самостійних IoT-систем, таких як розумні датчики, системи автоматизації будинку та багато іншого (рис. 1.6).



Рис. 1.6. Мікроконтролер ESP8266

Мікроконтролер ESP8266 відіграє ключову роль у розвитку Інтернету речей (IoT). Його компактні розміри, низька вартість та вбудований модуль Wi-Fi роблять його ідеальним вибором для широкого спектра застосувань.

ESP8266 використовується для створення різноманітних IoT-пристроїв, таких як датчики температури та вологості, системи автоматичного поливу, системи безпеки, розумні розетки та багато інших. Завдяки можливості підключення до Інтернету, ці пристрої можуть передавати дані в хмару, взаємодіяти з мобільними додатками та іншими системами.

Однією з ключових переваг ESP8266 є його гнучкість. Завдяки підтримці різних платформ розробки, таких як Arduino IDE та NodeMCU, програмування цього мікроконтролера є відносно простим, навіть для початківців. Це дозволяє розробникам швидко створювати прототипи та втілювати в життя свої ідеї.

### 1.3. Огляд датчиків інтелектуальних систем управління будинком

Сучасні системи "розумний дім" неможливі без використання широкого спектру датчиків, які збирають інформацію про стан навколишнього середовища та різних систем в будинку. Ці дані використовуються для автоматизації процесів, підвищення комфорту та забезпечення безпеки.

Датчики можуть вимірювати різноманітні параметри, включаючи температуру, вологість, освітленість, рух, рівень шуму, якість повітря, відкриття дверей та вікон, витік води та газу. Залежно від типу датчика та його призначення, він може бути підключений до централізованої системи управління за допомогою дротових або бездротових інтерфейсів. [9].

Датчики температури відіграють ключову роль у створенні комфортного та енергоефективного житлового середовища. Різноманітність типів датчиків дозволяє вибрати оптимальне рішення для конкретних завдань.

*Терморезистори (RTD)* є одними з найточніших датчиків температури. Їх принцип роботи заснований на зміні електричного опору металевого елемента залежно від температури. RTD характеризуються високою лінійністю, стабільністю та широким діапазоном робочих температур. Однак, вони можуть бути дорогими порівняно з іншими типами датчиків.

*Термістори* також змінюють свій опір залежно від температури, але на відміну від RTD використовують напівпровідникові матеріали. Термістори компактні, дешеві та мають високу чутливість, але їх характеристики можуть бути нелінійними, що ускладнює калібрування.

*Термопари* працюють на принципі термоелектричного ефекту, тобто при сполученні двох різних металів виникає електрична напруга, величина якої залежить від різниці температур між спаями. Термопари відзначаються високою надійністю, широким діапазоном вимірюваних температур та відносно низькою вартістю. Однак, вони можуть бути менш точними за RTD і вимагають додаткових електронних схем для перетворення термо-ЕРС в температуру.

*Напівпровідникові датчики* температури використовують зміну напруги на рп-переході для вимірювання температури. Вони компактні, енергоефективні та

часто інтегруються в мікросхеми. Однак, їх характеристики можуть залежати від інших параметрів, таких як напруга живлення і струм.

*Інфрачервоні датчики* температури вимірюють теплове випромінювання об'єктів, що дозволяє визначати їх температуру безконтактно. Вони широко використовуються для вимірювання температури рухомих об'єктів, а також для виявлення гарячих точок.

Вибір типу датчика температури залежить від наступних факторів:

- Діапазон вимірюваних температур. Кожен тип датчика має свої обмеження за діапазоном вимірювань.
- Точність вимірювання. Для точних наукових досліджень потрібні більш точні датчики, такі як RTD.
- Вартість. Термістори та напівпровідникові датчики зазвичай дешевші за RTD і термопари.
- Розміри. Для обмежених просторів можуть знадобитися мініатюрні датчики.
- Умови експлуатації. Важливо враховувати вплив вологості, вібрації та інших факторів на роботу датчика.

Датчики вологості - це прилади, які вимірюють вміст водяної пари в повітрі або інших середовищах. Вони широко застосовуються в різних галузях, від сільського господарства до промисловості, для контролю мікроклімату, прогнозування погоди та інших цілей. Принцип роботи датчиків вологості заснований на виявленні змін фізичних властивостей матеріалу залежно від рівня вологості [10].

Принцип роботи різних типів датчиків вологості заснований на зміні фізичних властивостей речовин під впливом вологості. Найбільш поширені типи датчиків вологості включають:

*Капацитивні датчики.* Ці датчики вимірюють зміну ємності конденсатора, утвореного двома електродами, між якими розташований діелектрик, що поглинає вологу. Зі збільшенням вологості збільшується діелектрична проникність середовища між електродами, що призводить до зміни ємності.



*Резистивні датчики.* Принцип роботи таких датчиків заснований на зміні електричного опору спеціальних матеріалів (наприклад, солей або полімерів) при зміні вологості. Збільшення вологості призводить до зміни провідності матеріалу, що фіксується датчиком.

*Термічні датчики.* Ці датчики використовують різницю в теплопровідності сухого і вологого повітря для вимірювання вологості. Чим вища вологість, тим повільніше відбувається випаровування води з поверхні нагрівального елемента датчика.

*Оптичні датчики.* Принцип роботи таких датчиків заснований на зміні оптичних властивостей середовища при зміні вологості. Наприклад, можна вимірювати поглинання або розсіювання світла в залежності від кількості водяної пари в повітрі.

Вибір конкретного типу датчика вологості залежить від ряду факторів, таких як:

- Точність вимірювань. Висока точність необхідна для наукових досліджень та медичних застосувань.
- Діапазон вимірювань. Різні датчики мають різні діапазони вимірювання відносної вологості.
- Швидкість відгуку. Для деяких застосувань важлива швидкість, з якою датчик реагує на зміни вологості.
- Вартість. Ціна датчиків може значно відрізнятися залежно від типу та виробника.
- Розміри. Для обмежених просторів можуть знадобитися мініатюрні датчики.
- Умови експлуатації. Важливо враховувати вплив температури, тиску, вібрації та інших факторів на роботу датчика.

Датчики руху - це електронні пристрої, які виявляють зміни в навколишньому середовищі, спричинені рухом об'єктів. Вони широко використовуються в системах безпеки, автоматизації та контролі доступу для підвищення безпеки, економії енергії та створення комфорту [11].

Датчики руху є невід'ємною частиною сучасних систем безпеки, автоматизації та контролю доступу. Вони дозволяють виявляти рух об'єктів у просторі, забезпечуючи високий рівень безпеки та комфорту. Принцип роботи різних типів датчиків руху заснований на виявленні змін фізичних параметрів навколишнього середовища, спричинених рухом.

*Піроелектричні (інфрачервоні) датчики.* Цей тип датчиків виявляє інфрачервоне випромінювання, яке випускають усі теплі тіла, включаючи людей і тварин. При русі об'єкта в зоні дії датчика змінюється інтенсивність інфрачервоного випромінювання, що фіксується датчиком. Перевагами таких датчиків є висока чутливість до руху теплокровних істот та відносно низька вартість.

*Ультразвукові датчики.* Ці датчики працюють на принципі ехолокації. Вони випромінюють ультразвукові хвилі, які відбиваються від об'єктів у зоні дії датчика. За зміною часу повернення відбитого сигналу датчик визначає наявність руху. Ультразвукові датчики менш чутливі до перешкод, таких як штори або скло, але можуть бути чутливими до перешкод від інших джерел ультразвуку.

*Мікрохвильові датчики.* Принцип роботи мікрохвильових датчиків аналогічний до ультразвукових, але вони використовують мікрохвильове випромінювання. Мікрохвильові датчики мають більшу дальність дії і можуть проникати через деякі матеріали, що робить їх придатними для використання в складних умовах.

*Датчики вібрації.* Ці датчики фіксують механічні вібрації, які виникають при русі об'єктів. Вони зазвичай використовуються для виявлення руху великих об'єктів, таких як двері або вікна.

Вибір конкретного типу датчика руху залежить від таких факторів, як:

- Дальність виявлення. Відстань, на якій датчик може виявити рух.
- Кут огляду. Розмір зони, яку датчик контролює.
- Імунітет до перешкод. Здатність датчика працювати в умовах підвищених шумів або перешкод.
- Вартість. Ціна датчика залежить від його характеристик і виробника.

- Умови експлуатації. Температура, вологість та інші фактори навколишнього середовища можуть впливати на роботу датчика.

Датчики руху широко використовуються в системах безпеки, автоматизації освітлення, контролі доступу, а також в промисловості. Завдяки своїй високій чутливості та надійності, вони дозволяють створити комфортні та безпечні умови проживання та роботи.

### **Висновки до розділу**

Проведений аналіз різних типів датчиків, мікроконтролерів та систем "розумний дім" дозволяє зробити ряд висновків.

Різноманітність датчиків, від температурних до датчиків руху, забезпечує збір детальної інформації про стан навколишнього середовища та об'єктів в будинку. Ці дані є основою для прийняття рішень системами автоматизації, що дозволяє оптимізувати процеси управління освітленням, кліматом, безпекою та іншими аспектами житлового середовища.

Мікроконтролери, як серце систем "розумний дім", забезпечують обробку даних, отриманих від датчиків, та керування виконавчими пристроями. Завдяки своїй гнучкості та масштабованості, мікроконтролери дозволяють створювати складні системи автоматизації, які відповідають індивідуальним потребам користувачів.

Системи "розумний дім" демонструють значний потенціал для підвищення комфорту, безпеки та енергоефективності житлових приміщень. Вони дозволяють автоматизувати рутинні завдання, створювати персоналізовані сценарії управління, а також інтегруватися з іншими системами "розумного міста".

Таким чином, системи "розумний дім" представляють собою перспективний напрямок розвитку технологій, який дозволяє створити більш комфортні, безпечні та енергоефективні житлові середовища.



## **РОЗДІЛ 2. ВИБІР ІНСТРУМЕНТІВ ДЛЯ "СИСТЕМИ РОЗУМНИЙ БУДИНОК" НА ОСНОВІ МІКРОКОНТРОЛЕРА ARDUINO**

### **2.1. Огляд основних компонентів системи**

#### **2.1.1. Огляд Arduino UNO**

Для реалізації проєкту "розумний дім" в рамках даної роботи було обрано мікроконтролерну плату Arduino Uno. Це рішення обумовлене низкою факторів, включаючи доступність, гнучкість та широку підтримку спільноти розробників.

Arduino Uno представляє собою універсальну платформу для швидкого прототипування та розробки електронних пристроїв. Плата базується на мікроконтролері ATmega328P, який забезпечує достатню обчислювальну потужність для виконання завдань, необхідних для управління системою "розумний дім". Наявність 14 цифрових входів/виходів, 6 аналогових входів, а також підтримка різних типів комунікацій (USB, UART, I2C, SPI) робить Arduino Uno універсальним інструментом для підключення різноманітних датчиків, актуаторів та інших периферійних пристроїв.

Однією з ключових переваг Arduino Uno є її відкритість. Плата має відкритий апаратний і програмний код, що дозволяє глибоко вивчити її роботу та модифікувати під власні потреби. Крім того, існує велика спільнота розробників, яка створює різноманітні бібліотеки, щити та інші ресурси, що спрощують процес розробки.

Простота використання Arduino Uno також є важливим фактором. Інтуїтивно зрозуміле середовище розробки Arduino IDE дозволяє швидко створювати та завантажувати програми на плату, навіть для користувачів без глибоких знань в області програмування.

Таким чином, вибір Arduino Uno в якості основної платформи для проєкту "розумний дім" є обґрунтованим з точки зору її технічних характеристик, доступності, гнучкості та широкої підтримки спільноти розробників.

Ключові характеристики Arduino Uno:

- Мікроконтролер ATmega328P
- 14 цифрових входів/виходів

- 6 аналогових входів
- USB-інтерфейс
- Підтримка різних мов програмування (C++, C)
- Відкритий апаратний і програмний код
- Велика спільнота розробників

Використання Arduino Uno дозволить реалізувати широкий спектр функцій в системі "розумний дім", включаючи управління освітленням, контролювання температури, забезпечення безпеки та багато іншого (рис. 2.1).



Рис. 2.1. Плата Arduino UNO

Плата Arduino Uno може функціонувати від двох типів джерел живлення: від порту USB комп'ютера або від зовнішнього джерела постійного струму. Вибір джерела живлення відбувається автоматично.

Опис пінів плати на базі ATMEGA328 показаний на рисунку 2.2.

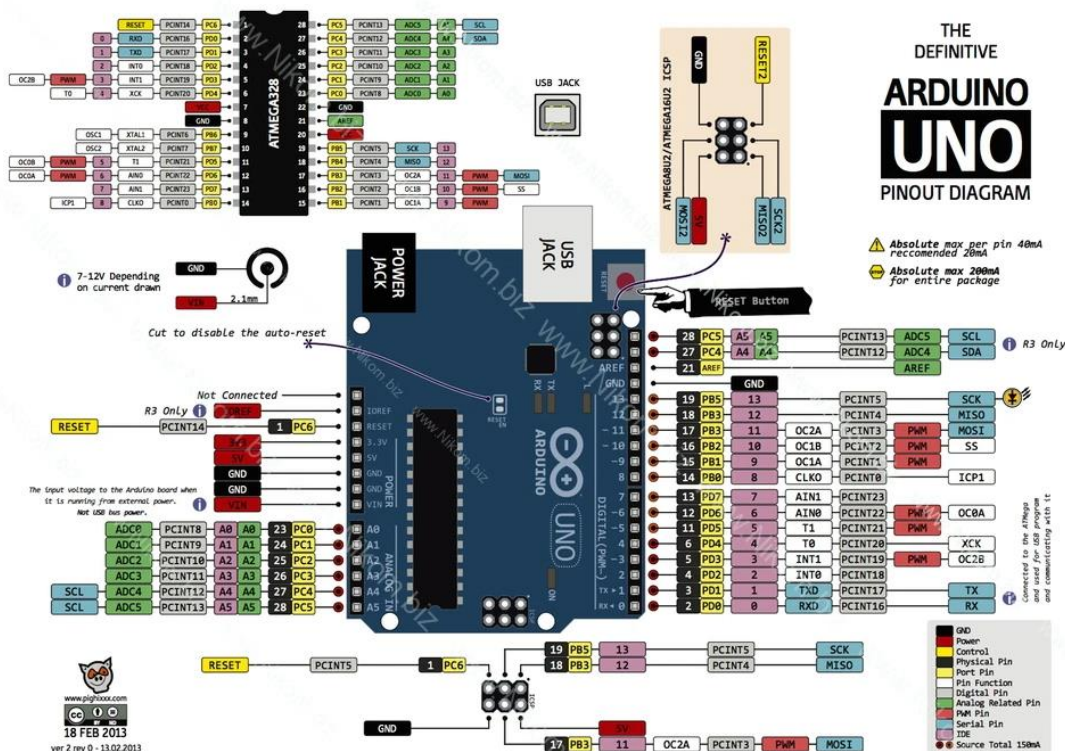


Рис. 2.2. Опис пінів плати Arduino UNO

Зовнішні джерела живлення можуть бути представлені мережевими адаптерами змінного струму або акумуляторами. Для підключення зовнішнього джерела живлення використовується спеціальний роз'єм на платі. Рекомендується використовувати адаптери з напругою в діапазоні від 7 до 12 В. Вихідна напруга адаптера 5 В подається на відповідний контакт на платі, забезпечуючи живлення всіх компонентів.

Важливо зазначити, що використання напруги живлення нижче 7 В може призвести до зниження вихідної напруги 5 В і, як наслідок, до нестабільної роботи плати. З іншого боку, надмірна напруга (понад 12 В) може призвести до перегріву стабілізатора напруги та виходу плати з ладу. Тому, для забезпечення оптимальної роботи Arduino Uno, слід дотримуватися рекомендованих параметрів живлення.

Підключення зовнішнього джерела живлення здійснюється шляхом під'єднання позитивного полюса до контакту VIN, а негативного - до контакту GND на платі. При використанні акумулятора також необхідно під'єднати його до цих контактів.

Вибір джерела живлення залежить від конкретних умов експлуатації плати Arduino Uno. Для стаціонарних застосувань можна використовувати мережевий адаптер, а для портативних пристроїв - акумулятори або батареї.

Цифрові виводи Arduino Uno, які можуть працювати як входи, так і виходи, забезпечують гнучкість при розробці електронних проєктів. Кожен вивід здатний працювати з напругою до 5 В і витримувати струм до 40 мА. Для зручності роботи з датчиками передбачена можливість підключення внутрішніх підтягуючих резисторів. Функції `pinMode()`, `digitalWrite()` та `digitalRead()` дозволяють легко керувати станом цих виводів (рис. 2.3).



Рис. 2.3. Виводи Arduino UNO

Мікроконтролер ATmega328P, який лежить в основі плати Arduino Uno, обладнаний наступними типами пам'яті:

1. Флеш-пам'ять. Обсяг флеш-пам'яті становить 32 кілобайти, з яких близько 500 байт відведено під завантажувач. Ця пам'ять використовується для зберігання програмного коду.



2. Оперативна пам'ять (SRAM). Обсяг оперативної пам'яті становить 2 кілобайти. Вона використовується для тимчасового зберігання даних під час виконання програми.
3. Пам'ять EEPROM. Обсяг пам'яті EEPROM становить 1 кілобайт. Цей тип пам'яті призначений для зберігання даних, які повинні зберігатися навіть після відключення живлення.

Для захисту мікроконтролера та підключених до нього пристроїв від перевантажень, плата Arduino Uno оснащена запобіжником. Запобіжник розрахований на максимальний струм 500 міліампер. У випадку перевищення цього значення, запобіжник перегорає, розриваючи електричний ланцюг та запобігаючи пошкодженню плати. Цей додатковий рівень захисту є важливим, оскільки навіть сучасні комп'ютери не завжди можуть забезпечити повний захист від перевантажень за лінією USB.

Принципова схема плати показана на рисунку 2.4.

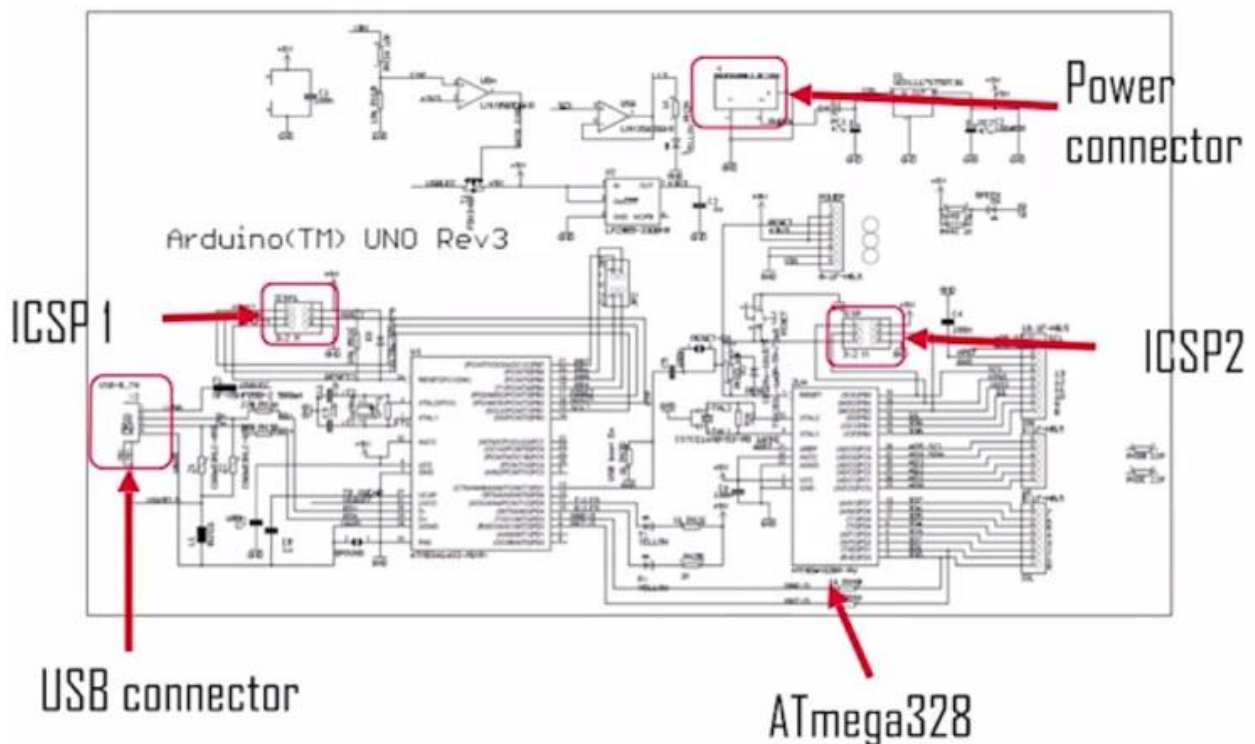


Рис. 2.4. Принципова схема плати Arduino UNO

Таким чином, конструкція плати Arduino Uno передбачає наявність необхідних ресурсів пам'яті для виконання різноманітних завдань та забезпечує

надійний захист від перевантажень, що підвищує її довговічність та безпеку використання.

### **2.1.2. Опис послідовного порту**

Послідовний порт, стандартизований як RS-232, є двонаправленим інтерфейсом, що використовується для асинхронної послідовної передачі даних. Це означає, що дані передаються без синхронізуючих імпульсів, а початок і кінець кожного байта визначаються стартовим і стоповим бітами. Швидкість передачі даних вимірюється в бодах і може варіюватися від декількох сотень до декількох десятків тисяч біт за секунду (рис. 2.5).



Рис. 2.5. COM порт або RS-232 порт

Послідовний порт є одним з найдавніших і найпростіших типів інтерфейсів для обміну даними між цифровими пристроями. Його основна особливість полягає в послідовній передачі інформації, тобто біти даних передаються по одному за лінією зв'язку. На відміну від паралельних портів, де всі біти даних передаються одночасно, послідовний порт забезпечує більш просту і надійну передачу даних на великі відстані.

Ключові характеристики послідовних портів:

1. Дуплексність. Більшість послідовних портів підтримують дуплексний режим роботи, тобто одночасну передачу і прийом даних. Це дозволяє пристроям спілкуватися в обох напрямках.

2. Швидкість передачі даних. Швидкість передачі даних, вимірювана в бодах, визначає кількість бітів, які можуть бути передані за одну секунду. Швидкість передачі залежить від типу послідовного порту та використовуваного обладнання.
3. Стандарт RS-232. Найпоширенішим стандартом для послідовних портів є RS-232. Він визначає електричні характеристики, рівні сигналів та інші параметри інтерфейсу.
4. Протоколи передачі даних. Для організації обміну даними по послідовному порту використовуються різні протоколи, такі як UART (Universal Asynchronous Receiver-Transmitter). Ці протоколи визначають формат даних, способи синхронізації та контролю помилок.

Розпіновку COM порту RS232 зображено на рисунку 2.6.

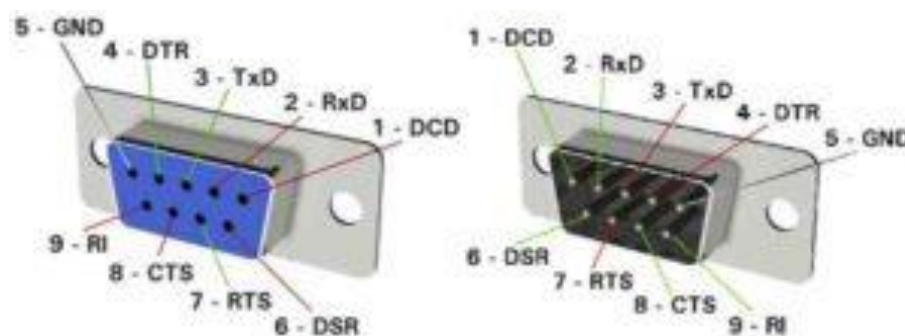


Рис. 2.6. Розпіновка COM порту RS232

Незважаючи на появу сучасніших інтерфейсів, таких як USB, послідовний порт продовжує залишатися важливим інструментом для взаємодії з мікроконтролерами, зокрема, з платами Arduino. Його простота, надійність та широка підтримка роблять його незамінним у багатьох додатках.

Arduino використовує послідовний порт для обміну даними з комп'ютером або іншими пристроями. Цей процес передбачає послідовну передачу бітів даних по одній лінії зв'язку. Для роботи з послідовним портом в середовищі Arduino використовується бібліотека Serial. Вона надає набір функцій, таких як `Serial.begin()`, `Serial.print()`, `Serial.println()`, `Serial.read()`, які дозволяють ініціалізувати порт, передавати дані та приймати їх.

Перед початком роботи з послідовним портом необхідно встановити швидкість передачі даних за допомогою функції `Serial.begin()`. Швидкість передачі вимірюється в бодах і визначає кількість бітів, які передаються за секунду. Правильне встановлення швидкості є критично важливим для успішного обміну даними.

Послідовний порт широко використовується для налагодження програмного забезпечення для Arduino. Виведення діагностичної інформації на комп'ютер через послідовний порт дозволяє відстежувати роботу програми в реальному часі, виявляти помилки та оптимізувати алгоритми.

Більшість плат Arduino мають як мінімум один послідовний порт. Наприклад, популярна модель Arduino Uno має один основний послідовний порт, підключений до USB-роз'єму. Деякі більш складні моделі, такі як Arduino Mega, можуть мати кілька послідовних портів, що дозволяє підключати до плати одночасно кілька пристроїв [12].

### **2.1.3. Опис PIR-датчика**

Піроелектричний інфрачервоний датчик (PIR-датчик) є високочутливим електронним пристроєм, призначеним для виявлення інфрачервоного випромінювання, яке випускають усі теплі тіла, включаючи людей і тварин. Принцип роботи PIR-датчика заснований на використанні піроелектричного матеріалу, який генерує електричний заряд при зміні температури [13].

Конструктивно PIR-датчик складається з лінзи Френеля, яка фокусує інфрачервоне випромінювання на піроелектричний елемент. Коли в зоні дії датчика з'являється рухомий об'єкт, що випромінює тепло, його інфрачервоне випромінювання змінює температуру піроелектричного елемента, що призводить до виникнення електричного сигналу. Цей сигнал обробляється електронною схемою датчика і може використовуватися для подальшої передачі або активації інших пристроїв (рис. 2.7).





Рис. 2.7. PIR-датчик

Піроелектричні інфрачервоні датчики (PIR-датчики) є ефективним засобом для виявлення руху в навколишньому середовищі. Принцип їх роботи заснований на реєстрації змін теплового випромінювання, яке виникає при русі об'єктів.

Однією з основних функцій PIR-датчиків є виявлення руху. Коли в зоні дії датчика з'являється об'єкт, що випромінює тепло (наприклад, людина або тварина), датчик реєструє зміну теплового потоку і генерує електричний сигнал. Цей сигнал може використовуватися для активації різних пристроїв та систем.

Застосування PIR-датчиків є досить широким і охоплює різні сфери:

- Системи безпеки. PIR-датчики використовуються для створення охоронних систем, які спрацьовують при виявленні несанкціонованого проникнення.
- Автоматизація освітлення. Датчики дозволяють автоматично вмикати та вимикати освітлення залежно від присутності людей, що сприяє економії енергії.
- Системи контролю доступу. PIR-датчики можуть використовуватися для відкривання дверей або шлагбаумів при

виявленні авторизованих осіб.

- Робототехніка. Датчики руху використовуються для створення роботів, які можуть орієнтуватися в просторі та уникати перешкод.

Ключові характеристики PIR-датчиків:

1. Чутливість: здатність датчика виявляти навіть незначні зміни теплового випромінювання.
2. Дальність дії: відстань, на якій датчик може виявити рух.
3. Кут огляду: розмір зони, яку датчик контролює.
4. Час реакції: швидкість, з якою датчик реагує на зміну теплового потоку.

Більшість сучасних PIR-датчиків дозволяють користувачеві налаштовувати такі параметри, як чутливість, час затримки реакції та кут огляду. Це дозволяє оптимізувати роботу датчика для конкретних умов застосування.

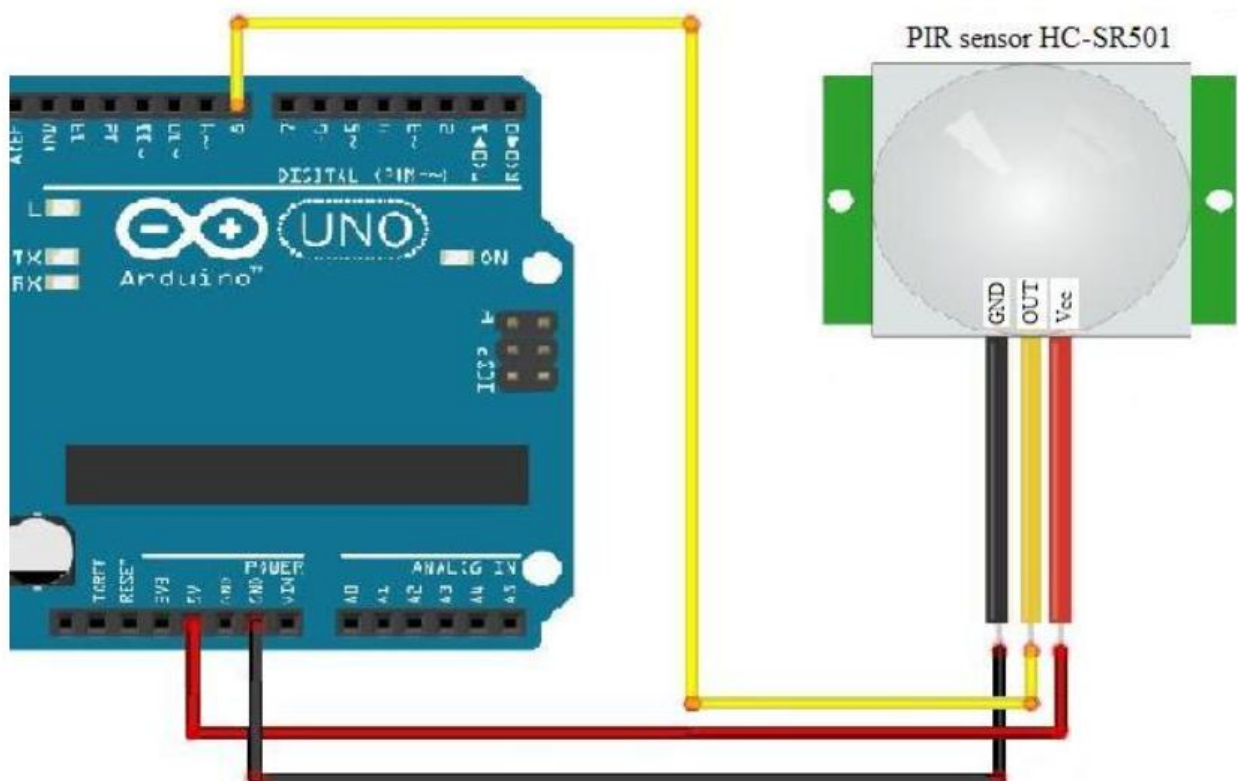


Рис. 2.8. Схема підключення датчика руху HC-SR501

#### 2.1.4. Опис DHT11

DHT11 - це компактний цифровий датчик, який дозволяє вимірювати

температуру та вологість повітря з високою точністю. Він широко застосовується в різних галузях, включаючи метеорологію, сільське господарство та системи "розумний дім" (рис. 2.9).

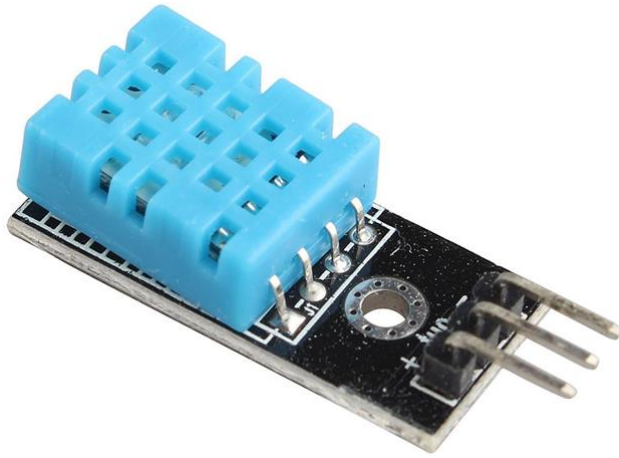


Рис. 2.9. DHT11 датчик

DHT11 складається з двох основних елементів: терморезистора для вимірювання температури та ємнісного датчика для вимірювання вологості. Принцип роботи датчика заснований на зміні електричних характеристик цих елементів залежно від температури та вологості навколишнього середовища. Отримані дані перетворюються в цифровий сигнал, який передається на мікроконтролер.

Основні характеристики DHT11:

1. Діапазон вимірювань. Датчик здатний вимірювати температуру в діапазоні від 0 до 50 градусів Цельсія з точністю  $\pm 2$  градуси, а відносну вологість - від 20% до 80% з точністю  $\pm 5\%$ .
2. Цифровий інтерфейс. Для взаємодії з мікроконтролерами DHT11 використовує простий цифровий інтерфейс, що дозволяє легко інтегрувати його в різні електронні системи.
3. Низьке енергоспоживання. Датчик має низький рівень споживання енергії, що робить його придатним для використання в автономних пристроях, що живляться від батарей.
4. Компактні розміри. Малі габарити дозволяють легко розміщувати датчик у різних пристроях та системах [14].

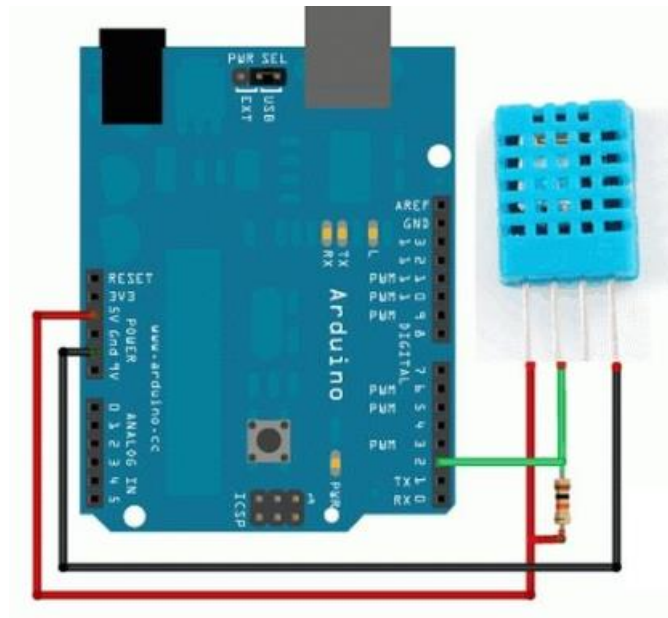


Рис. 2.10. Схема підключення датчика DHT11

## 2.2. Огляд середовищ розробки для програмування мікроконтролерів

### 2.2.1. Огляд Arduino IDE

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

Особливості Arduino IDE:

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

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

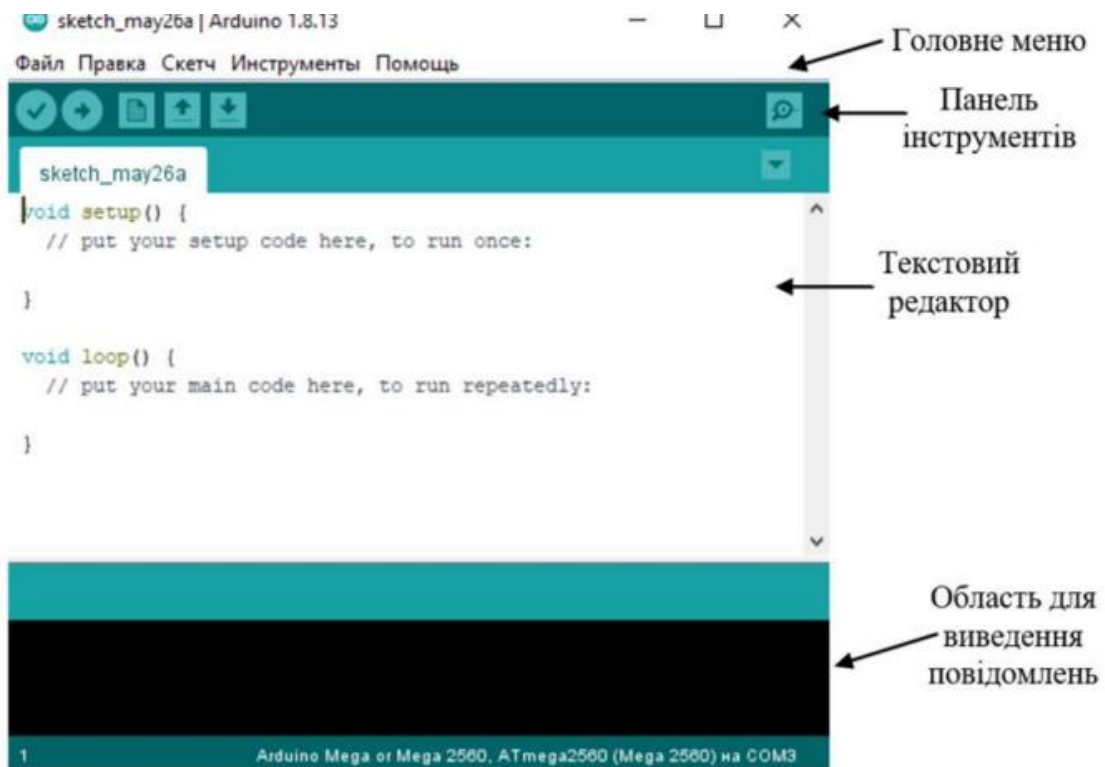


Рис. 2.11. Інтерфейс Arduino IDE

Скетчі, написані в середовищі розробки Arduino IDE, зберігаються у файлах з розширенням .ino. Редактор коду в IDE забезпечує стандартний набір функцій для роботи з текстом: копіювання, вставка, пошук і заміна. Інформація про хід виконання програми та можливі помилки відображається в області повідомлень. Вибір моделі плати Arduino та послідовного порту здійснюється в нижній частині вікна програми. Панель інструментів надає швидкий доступ до основних функцій середовища розробки. На панелі інструментів знаходиться 6 кнопок (рис. 2.12):



Рис. 2.12. Панель інструментів

### Основні функції Arduino IDE

- **Перевірити.** Дана опція дозволяє перевірити синтаксис написаного коду. У разі виявлення помилок вони відображаються в області повідомлень.

- **Завантажити.** Використовується для компіляції програми та її завантаження у мікроконтролер Arduino.

- **Створити.** Дозволяє створити новий файл для написання скетчу.

- **Відкрити.** Призначена для відкриття існуючого файлу зі скетчем.

- **Зберегти.** Дозволяє зберегти поточний скетч у файл.

- **Монітор послідовного порту.** Використовується для відкриття програми Serial Monitor, яка відображає дані, що надходять від Arduino на комп'ютер через послідовний інтерфейс. Монітор підтримує роботу як із USB-варіантами плати, так і зі звичайними версіями Arduino. Для передачі даних на зовнішній пристрій у вікні монітора вводиться текст, після чого натискається кнопка "Відправити" або клавіша Enter. Швидкість передачі даних потрібно налаштувати відповідно до параметрів, вказаних у функції `Serial.begin()` у вашому коді. Функція `Serial.print()` дозволяє виводити текст у вікно монітора.

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



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

Для завантаження скетчу необхідно вибрати відповідну плату та порт (див. рис.2.13), які використовуються для вашої операційної системи. Це здійснюється через меню Tools, де у розділі Board вибирається модель плати, а в розділі портів — відповідний COM-порт (наприклад, COM1, COM2, COM4, COM5, COM7 тощо) або USB-з'єднання.

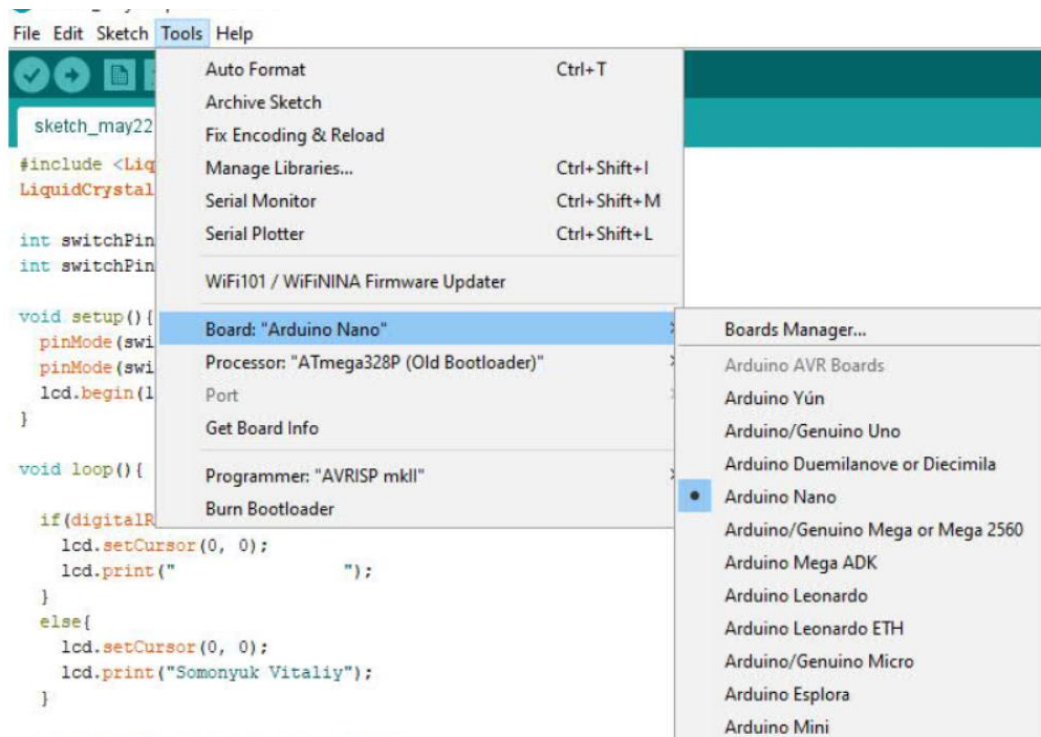


Рис. 2.13. Зображення вибору плати

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

Після вибору потрібних параметрів можна приступати безпосередньо до процесу прошивки. Для цього необхідно скопіювати написаний код. Компілятор перетворить ваш код, написаний на спрощеній версії C++, в машинний код, який розуміє мікроконтролер Arduino. Скопійований код

передається в буфер завантажувача - спеціальної програми, яка зашита в пам'ять мікроконтролера. Завантажувач, у свою чергу, переписує отриманий код в основну пам'ять мікроконтролера і запускає його на виконання.

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

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

- `setup()` виконується один раз на початку програми і використовується для ініціалізації портів введення-виведення, встановлення початкових значень змінних та підключення бібліотек.

- `loop()` виконується циклічно і містить основний алгоритм програми. Все, що має повторюватися, розміщується саме тут.

Основні функції для роботи з портами введення-виведення:

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

Arduino IDE широко використовується для розробки різноманітних електронних проектів, включаючи системи автоматизації будинку, робототехніку, інтернет речей та багато інших. Завдяки своїй простоті та універсальності, Arduino IDE стала популярним інструментом як для професійних розробників, так і для любителів електроніки.

Arduino IDE є потужним і зручним інструментом для розробки програмного забезпечення для платформи Arduino. Завдяки своїм можливостям



та доступності, Arduino IDE стала стандартом де-факто для багатьох розробників, які працюють з мікроконтролерами.

### **2.2.2. Огляд Proteus 8 Professional**

Proteus Design Suite - це потужний програмний пакет, призначений для автоматизації процесу розробки електронних схем. Пакет пропонує інтегроване середовище для моделювання, симуляції та візуалізації роботи електронних схем, включаючи мікроконтролерні системи.

Однією з ключових особливостей Proteus є можливість детального моделювання поведінки електронних компонентів, включаючи мікроконтролери, мікропроцесори та цифрові сигнальні процесори (DSP). Це дозволяє інженерам перевіряти правильність роботи розроблених схем ще на етапі проектування, виявляти потенційні проблеми та оптимізувати їх роботу.

Важливою перевагою Proteus є концепція наскрізного проектування. Це означає, що зміни, внесені в схему, автоматично відображаються у всіх пов'язаних з нею модулях, таких як трасування друкованої плати. Це значно спрощує процес розробки та зменшує ризик помилок.

Пакет Proteus включає в себе велику бібліотеку електронних компонентів, що дозволяє швидко створювати схеми різної складності. Крім того, Proteus підтримує моделювання аналогових і цифрових схем, що робить його універсальним інструментом для електронних інженерів. (рис. 2.14).

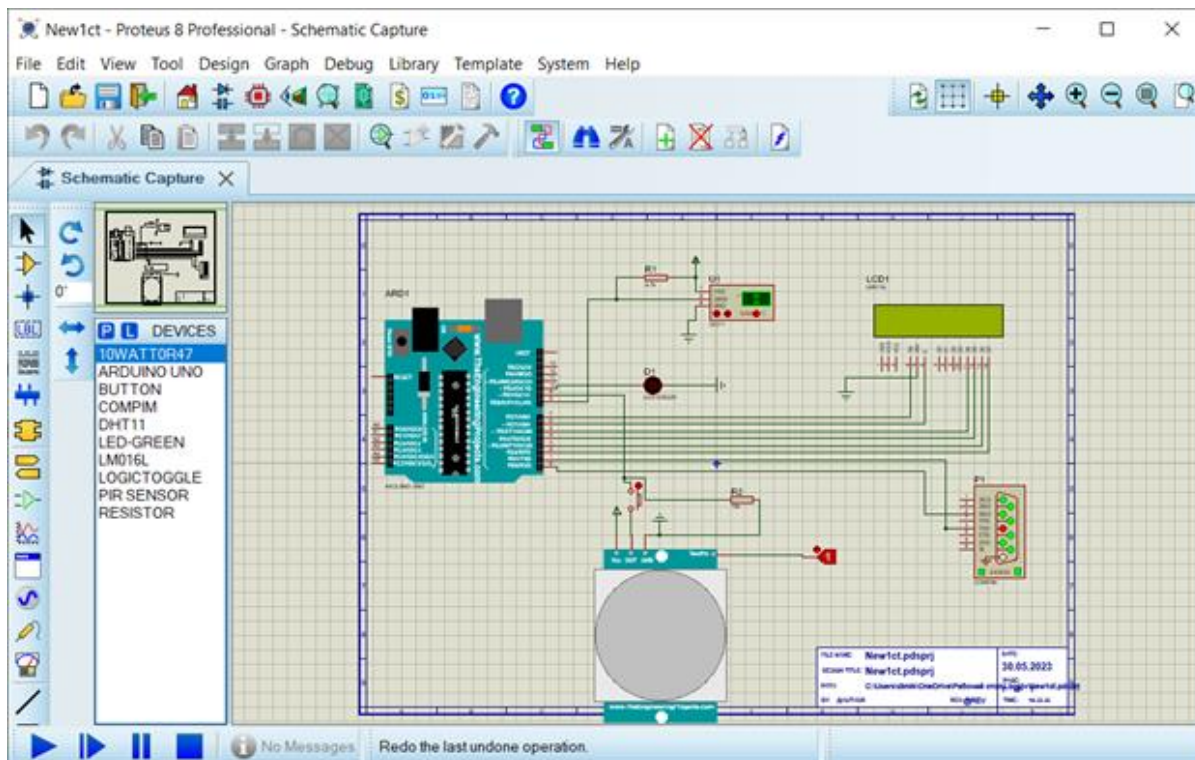


Рис. 2.14. Proteus 8 Professional

Proteus Design Suite - це потужний інструментарій, який об'єднує в собі можливості моделювання електронних схем, проектування друкованих плат та розробки програмного забезпечення для мікроконтролерів. Цей пакет дозволяє інженерам і розробникам створювати прототипи електронних пристроїв та проводити їх симуляцію ще до фізичної реалізації.

Ключові компоненти Proteus Design Suite:

- ISIS модуль призначений для створення схем, вибору компонентів з бібліотеки та моделювання їх роботи. ISIS дозволяє аналізувати як аналогові, так і цифрові схеми, а також виконувати симуляцію мікроконтролерних систем.
- ARES модуль використовується для розробки друкованих плат. На основі створеної в ISIS схеми, ARES автоматично генерує трасування доріжок і розміщення компонентів.

Однією з найважливіших особливостей Proteus є можливість детального моделювання мікроконтролерів. Для цього в схему додається модель мікроконтролера, в яку завантажується програмний код. Далі, за допомогою симулятора, можна спостерігати за роботою мікроконтролера та взаємодією з

іншими компонентами схеми. Це дозволяє виявити та виправити помилки в проекті на ранніх етапах розробки [16].

Ключові переваги Proteus Design Suite:

1. Об'єднує всі необхідні інструменти для створення, моделювання та аналізу електронних схем, а також для проектування друкованих плат.
2. Підтримує широкий спектр електронних компонентів, включаючи мікроконтролери, аналогові та цифрові інтегральні схеми, пасивні елементи.
3. Забезпечує точне моделювання роботи електронних схем, що дозволяє виявляти потенційні проблеми на ранніх етапах розробки.
4. Надає інструменти для візуалізації роботи схеми в режимі реального часу, що полегшує розуміння її функціонування.
5. Містить велику кількість готових моделей електронних компонентів, що значно прискорює процес розробки.
6. Сумісний з різними стандартами електронних схем та друкованих плат.

Proteus Design Suite є універсальним інструментом для електронних інженерів, який дозволяє ефективно розробляти, моделювати та тестувати електронні пристрої. Його широкі можливості, інтуїтивний інтерфейс та постійна підтримка роблять його незамінним помічником для професіоналів та любителів електроніки.

### **2.2.3. Огляд Visual Studio**

Microsoft Visual Studio - це комплексна платформа для розробки програмного забезпечення, що пропонує широкий спектр інструментів і функцій для створення різноманітних додатків. Цей продукт від компанії Microsoft є одним з найпопулярніших середовищ розробки у світі, що обумовлено його потужністю, універсальністю та постійним розвитком.

Visual Studio дозволяє розробникам створювати як консольні програми, так і програми з графічним інтерфейсом користувача. Підтримуються різноманітні

технології, включаючи Windows Forms, ASP.NET для веб-розробки, а також мобільні платформи. Це робить Visual Studio універсальним інструментом для розробки програмного забезпечення під різні операційні системи та пристрої [17].

Однією з ключових особливостей Visual Studio є наявність інтегрованого середовища розробки (IDE), яке об'єднує в собі редактор коду, компілятор, налагоджувач та інші необхідні інструменти. Редактор коду забезпечує інтелектуальне автодоповнення коду, підсвічування синтаксису та інші функції, що полегшують процес написання коду. Вбудований налагоджувач дозволяє ефективно виявляти та усувати помилки в програмах.

Крім того, Visual Studio включає в себе набір візуальних інструментів для створення графічних інтерфейсів, дизайну баз даних та веб-сайтів. Це дозволяє розробникам зосередитися на логіці програми, делегуючи рутинні завдання автоматизованим інструментам (рис. 2.15).

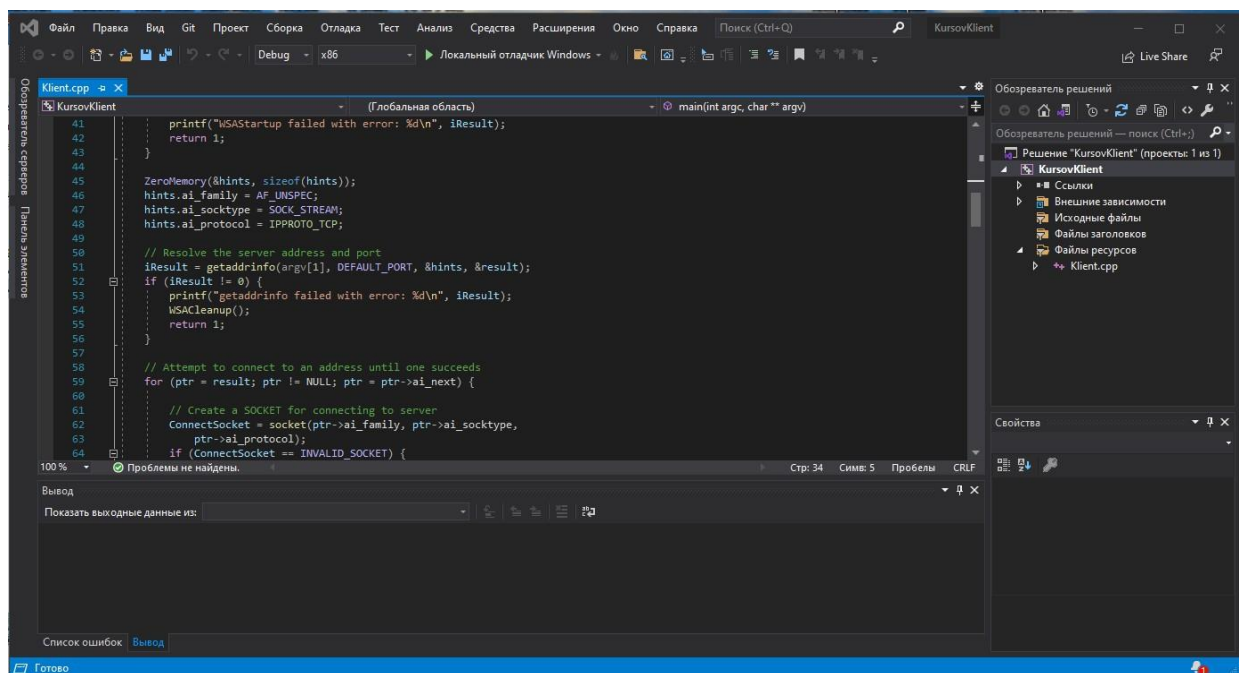


Рис. 2.15. Visual Studio

Visual Studio є одним з найпопулярніших інтегрованих середовищ розробки (IDE), що надають розробникам потужні інструменти для створення різноманітних програмних продуктів. Ця платформа підтримує широкий спектр мов програмування, включаючи C++, C#, F# та Visual Basic, що дозволяє

розробникам обирати найбільш підходящий інструмент для вирішення конкретних завдань.

За допомогою Visual Studio можна розробляти як традиційні десктопні додатки для операційної системи Windows, так і сучасні веб-додатки, мобільні програми та хмарні сервіси. Платформа пропонує широкий набір інструментів для створення користувацьких інтерфейсів, роботи з базами даних, а також для відлагодження та тестування програмного коду.

Однією з ключових переваг Visual Studio є підтримка різноманітних технологій і платформ. Це дозволяє розробникам створювати кросплатформні додатки, які можуть працювати на різних операційних системах і пристроях. Крім того, Visual Studio тісно інтегрується з іншими продуктами Microsoft, такими як Azure, що полегшує розробку хмарних додатків [18].

#### 2.2.4. Огляд Virtual Null Modem

Віртуальний нульовий модем - це програмний інструмент, який створює віртуальний канал комунікації між двома послідовними портами. Це дозволяє обмінюватися даними між різними пристроями або програмами без використання фізичного кабелю. Віртуальний нульовий модем широко використовується для тестування програмного забезпечення, що працює з послідовними портами, а також для створення тестових середовищ. (рис. 2.16).

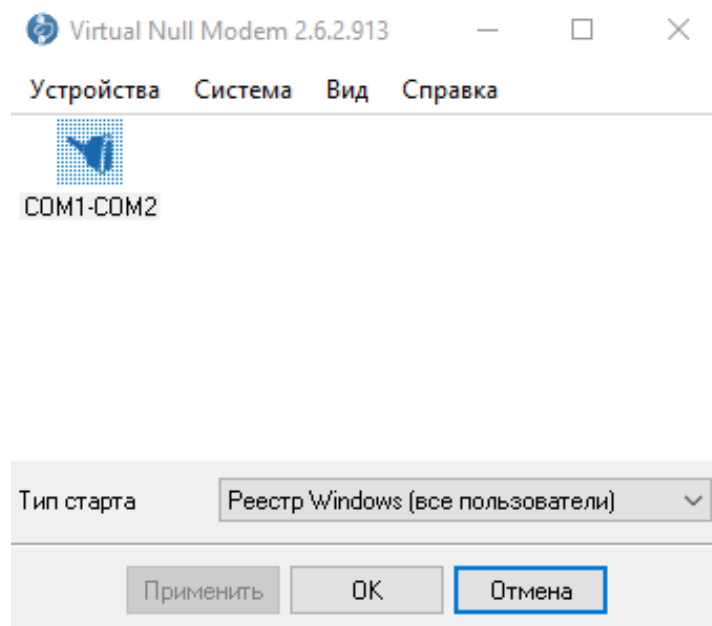


Рис. 2.16. Virtual Null Modem

Virtual Null Modem — віртуальний нульовий модем, який емулює поведінку фізичного нуль-модемного кабелю. Він дозволяє створювати будь-яку кількість віртуальних послідовних портів (наприклад, COM10, COM11, COM127 тощо) і з'єднувати їх між собою. Це означає, що програми, які працюють з одним віртуальним портом, можуть обмінюватися даними з програмами, підключеними до іншого віртуального порту, як ніби вони підключені до одного фізичного пристрою.

### **Висновки до розділу**

У другому розділі було проведено детальний аналіз основних компонентів та інструментів, необхідних для реалізації системи "розумний дім" на базі платформи Arduino.

Було встановлено, що Arduino Uno є оптимальним вибором для створення прототипів завдяки своїй доступності, простоті використання та широкій підтримці спільноти розробників. Датчики, такі як PIR-датчик та DHT11, забезпечують збір необхідної інформації про навколишнє середовище, що є основою для прийняття рішень системою автоматизації.

Для розробки програмного забезпечення для Arduino було розглянуто кілька інструментів:

- Arduino IDE спеціалізоване середовище розробки, оптимізоване для роботи з платформою Arduino. Воно надає інтуїтивний інтерфейс та широкий набір бібліотек.
- Proteus Design Suite дозволяє моделювати електронні схеми та їх роботу, що допомагає виявити та виправити помилки на ранніх етапах розробки.
- Visual Studio універсальне середовище розробки, яке підтримує широкий спектр мов програмування та платформ. Воно може використовуватися для створення більш складних програмних додатків, які взаємодіють з системою "розумний дім".

- Віртуальний нульовий модем дозволяє емулювати послідовні порти, що спрощує процес відлагодження та тестування програмного забезпечення.

Вибір конкретного інструмента залежить від складності проекту, досвіду розробника та вимог до кінцевого продукту. Комбінація цих інструментів дозволяє створювати ефективні та надійні системи автоматизації.

## РОЗДІЛ 3. МОДЕЛЮВАННЯ СИСТЕМИ "РОЗУМНИЙ ДІМ" НА ПЛАТФОРМІ ARDUINO ЗА ДОПОМОГОЮ PROTEUS

### 3.1. Моделювання системи на основі мікроконтролера Arduino

Для візуалізації та перевірки функціональності розробленої системи "розумний дім" було проведено детальне моделювання в середовищі Proteus 8 Professional. Цей потужний інструмент дозволив створити віртуальний прототип системи, що включає в себе всі необхідні компоненти та взаємозв'язки між ними.

В рамках моделювання було відтворено основні елементи системи: мікроконтролер Arduino Uno, який виконує роль центрального контролера, а також датчики руху PIR і температури та вологості DHT11. За допомогою програмного забезпечення Proteus було створено віртуальну схему, що відображає фізичну структуру системи. Далі, за допомогою вбудованого симулятора, було змодельовано роботу схеми в різних умовах. Це дозволило виявити потенційні проблеми та оптимізувати структуру системи ще до її фізичної реалізації.

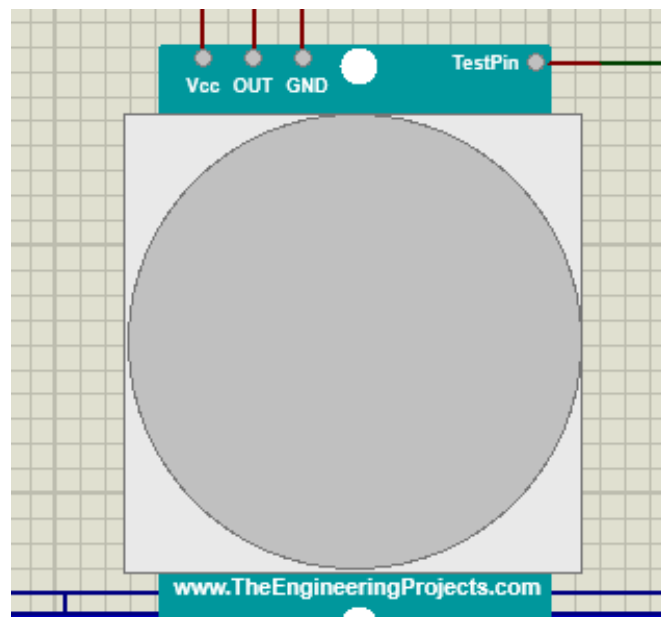


Рис. 3.1. PIR-датчик в Proteus 8 Professional

Розроблена система "розумний дім" базується на мікроконтролері Arduino Uno, який виконує роль центрального процесора, що керує роботою всіх компонентів системи. До складу системи входять такі датчики:

**PIR-датчик** використовується для виявлення руху в приміщенні. При фіксації руху датчик передає сигнал на мікроконтролер, що може бути



використано для автоматичного увімкнення освітлення або активації інших функцій системи безпеки (рис. 3.1).

**Датчик DHT11** виконує функцію вимірювання температури та відносної вологості повітря в приміщенні. Отримані дані передаються на мікроконтролер для подальшої обробки та відображення користувачеві або використання в системах автоматичного регулювання мікроклімату (рис. 3.2).

Мікроконтролер Arduino Uno обробляє дані, отримані від датчиків, та приймає рішення щодо керування виконавчими пристроями системи. Наприклад, при виявленні руху PIR-датчиком мікроконтролер може увімкнути освітлення, а при зміні температури або вологості, що виходять за задані межі, активувати систему кондиціонування або вентиляції.

Для більш розширених функцій система може бути інтегрована з хмарними сервісами. В такому випадку дані з датчиків передаються на віддалений сервер, де їх можна зберігати, аналізувати та використовувати для створення різноманітних сервісів

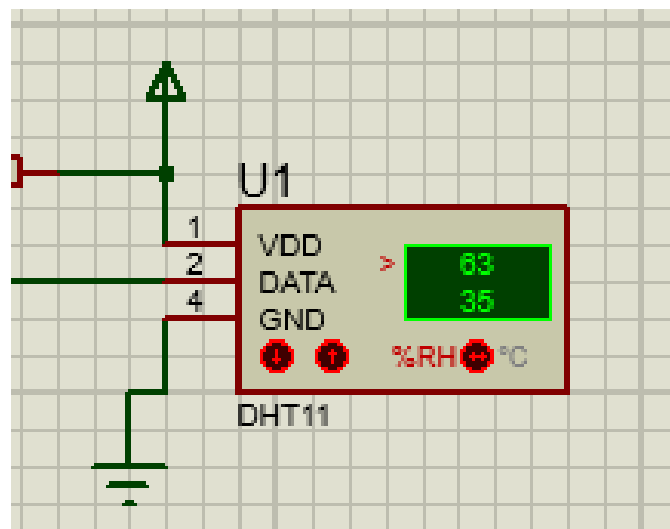


Рис. 3.2. DHT11 датчик в Proteus 8 Professional

### 3.2. Розробка програмного забезпечення для Arduino IDE

Після детального моделювання та аналізу функціональних можливостей системи "розумний будинок" було перейдено до етапу розробки програмного забезпечення. Основою програмної реалізації стала платформа Arduino та її інтегроване середовище розробки (IDE).

Arduino IDE була обрана завдяки своїй інтуїтивності, відкритості та

широкій підтримці спільноти розробників. Це середовище дозволило ефективно створити програмний код, який керує роботою всіх компонентів системи: мікроконтролера, датчиків та виконавчих пристроїв.

Процес розробки програмного забезпечення включав в себе наступні етапи:

1. Створення алгоритмів для зчитування даних з датчиків, прийняття рішень на основі отриманої інформації та керування виконавчими пристроями.
2. Написання коду на мові програмування C++ в середовищі Arduino IDE.
3. Програмний код був ретельно протестований на симуляторі Proteus, а потім завантажений на мікроконтролер для перевірки роботи в реальних умовах.

Ключові функції програмного забезпечення:

- Програма періодично зчитує дані з PIR-датчика (наявність руху) та датчика DHT11 (температура, вологість).
- На основі отриманих даних програма приймає рішення про виконання певних дій, наприклад, увімкнення освітлення, включення вентиляції тощо.
- Програма керує роботою реле, сервоприводів та інших виконавчих пристроїв для виконання необхідних дій.
- Програма може забезпечувати відображення інформації на дисплеї або передачу даних на зовнішній пристрій для віддаленого моніторингу та управління.

В результаті проведеної роботи було розроблено програмне забезпечення, яке забезпечує ефективну роботу системи "розумний дім". Програма дозволяє автоматизувати різноманітні процеси в будинку, підвищити рівень комфорту та безпеки, а також зменшити споживання енергії. (рис. 3.3).

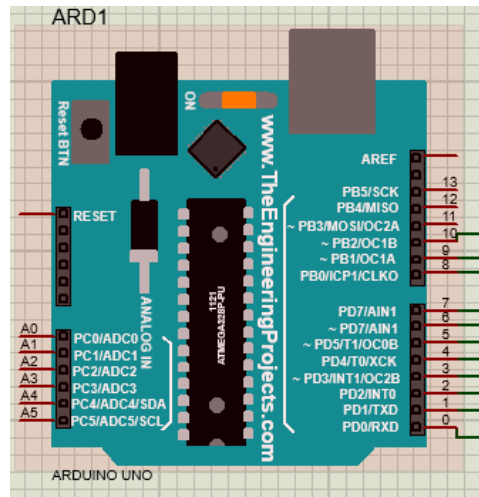


Рис. 3.3. Arduino UNO в Proteus 8 Professional

Для реалізації функціоналу системи "розумний будинок" було розроблено програмне забезпечення на основі платформи Arduino. Програмний код, написаний на мові C++, забезпечує взаємодію між мікроконтролером та підключеними до нього датчиками та виконавчими пристроями.

Для кожного типу датчика був розроблений окремий модуль програмного забезпечення. Модуль, відповідальний за обробку даних з PIR-датчика, здійснює періодичний збір інформації про рух в приміщенні. При виявленні руху, програма ініціює відповідну дію, наприклад, увімкнення освітлення. Модуль, що працює з DHT11, збирає дані про температуру та вологість повітря і передає їх на мікроконтролер для подальшої обробки та відображення.

Для передачі даних з мікроконтролера на віддалений сервер було розроблено спеціальний модуль програмного забезпечення. Цей модуль забезпечує надійне підключення до мережі та передачу даних у відповідному форматі. Були розроблені алгоритми для обробки помилок передачі даних та забезпечення стабільної роботи системи.

Розроблене програмне забезпечення було ретельно протестовано в різних умовах для виявлення та усунення можливих помилок. Було проведено ряд тестів для перевірки коректності роботи датчиків, мікроконтролера та мережевого з'єднання. В результаті тестування були внесені необхідні корективи в програмний код.

Розроблене програмне забезпечення забезпечує ефективну роботу системи "розумний будинок". Воно дозволяє збирати дані з датчиків, приймати рішення на їх основі та керувати виконавчими пристроями. Завдяки використанню мови програмування C++ та платформи Arduino, програмне забезпечення є гнучким та легко адаптується до різних конфігурацій системи. Реалізована система демонструє високий рівень надійності та може бути використана для створення більш складних систем автоматизації.

### **3.3. Опис програмного коду Arduino UNO**

Програмне забезпечення, розроблене для керування системою "розумний дім", базується на мові програмування C++ та використовує середовище розробки Arduino IDE. Код, написаний для мікроконтролера Arduino Uno, організовано у вигляді функцій `setup()` та `loop()`, що є стандартною структурою для програм Arduino.

На початку роботи програми виконується функція `setup()`. У цій функції відбувається:

- Підключаються необхідні бібліотеки `DHT.h` для роботи з датчиком DHT11 та `LiquidCrystal.h` для керування рідкокристалічним дисплеєм.
- Встановлюється режим роботи портів введення-виведення для підключених датчиків та виконавчих пристроїв.
- Ініціалізується послідовний порт для передачі даних на комп'ютер або інший пристрій.
- Налаштовується рідкокристалічний дисплей для відображення інформації.
- Виконується первинна ініціалізація датчиків DHT11 для подальшого зчитування даних про температуру та вологість.

*Основний цикл програми (функція `loop()`)*

Функція `loop()` виконується циклічно після виконання функції `setup()`. Вона містить основний алгоритм роботи програми:

1. Зчитування даних з датчиків. Регулярно зчитуються дані з датчика DHT11 (температура та вологість).
2. Обробка даних. Отримані дані обробляються та зберігаються в змінних.
3. Прийняття рішень. На основі отриманих даних приймаються рішення про виконання певних дій, наприклад, увімкнення/вимкнення освітлення, вентиляції тощо.
4. Виведення даних. Отримані дані можуть виводитися на дисплей або передаватися на зовнішній пристрій.

```
void loop()
{
  int t = dht.readTemperature();
  int h = dht.readHumidity();
  // ... решта коду для обробки даних та управління пристроями
}
```

У наведеному фрагменті коду функція `loop()` періодично зчитує значення температури (`t`) та вологості (`h`) з датчика DHT11. Далі ці дані можуть бути використані для прийняття рішень та керування іншими компонентами системи.

В цьому блоці ми зчитуємо температуру і вологість з датчика DHT.

```
if ( isnan(t)) {
  Serial.println(F("Failed to read from DHT sensor!"));
  lcd.clear();
  lcd.setCursor(5, 0);
  lcd.print("Error");
  return;
}
```

Якщо функція зчитування температури повертає значення "NaN" (Not a Number), що свідчить про помилку під час зчитування даних з датчика, то програма переходить до виконання процедури обробки помилок. Ця процедура передбачає виведення повідомлення про помилку на серійний порт для

подальшої діагностики, а також на рідкокристалічний дисплей для інформування користувача про проблему.

```
lcd.setCursor(0, 0);  
lcd.print("Temp: ");  
lcd.print(t); lcd.print("C ");  
Serial.write('t');  
delay(150);  
Serial.write((char)t);  
delay(200);
```

Для виведення інформації на рідкокристалічний дисплей використовуються функції бібліотеки LiquidCrystal, які дозволяють встановити курсор у потрібну позицію та вивести текстові рядки. Передача даних через послідовний порт здійснюється за допомогою функції Serial.write(), яка дозволяє передавати окремі символи.

```
lcd.setCursor(0, 2);  
lcd.print(" Hum: ");  
lcd.print(h);  
lcd.print("%");  
Serial.write('h');  
delay(150);  
Serial.write((char)h);  
delay(200);
```

Для виведення інформації про вологість на рідкокристалічний дисплей використовуються функції бібліотеки LiquidCrystal, які дозволяють встановити курсор у потрібну позицію та вивести текстові рядки. Передача даних через послідовний порт здійснюється за допомогою функції Serial.write(), яка дозволяє передавати окремі символи "%".

```
if(digitalRead(9)==HIGH)  
{  
digitalWrite(10,HIGH);  
if(mov==false)  
{  
Serial.write('m');
```

```

delay(500);
mov=true; notmov=true;
}

    }
    else{ digitalWrite(10,LOW);
    if(notmov==true)
    {
    Serial.write('n');
    delay(500);
    notmov=false;
    }
    mov=false;
    }
}

```

Цей блок коду здійснює перевірку стану цифрового входу, до якого підключений PIR-датчик. Якщо стан входу змінюється на високий (HIGH), то на вихідний пін 10 подається високий рівень сигналу, що може використовуватися для керування зовнішнім пристроєм. Крім того, відбувається зміна стану допоміжних змінних mov та notmov для відстеження попередніх станів системи та передачі відповідних сигналів через послідовний порт.

### 3.4. Опис коду клієнтської програми

Для забезпечення віддаленого доступу та керування датчиком ліній було розроблено клієнтську частину програмного забезпечення. Цей модуль відповідає за встановлення з'єднання з сервером, передачу команд та отримання даних. Детальний код клієнтської частини наведено в Додатку В.

Клієнтська програма, написана на мові C++, використовує бібліотеку Winsock2 для роботи з мережевими сокетом. Основними функціями клієнта є:

- Ініціалізація. На початку виконання програми відбувається ініціалізація бібліотеки Winsock2 та створення сокета для встановлення з'єднання з сервером.
- Підключення до сервера. Клієнт намагається встановити з'єднання з сервером за вказаною адресою та портом.

- Передача команд. Після встановлення з'єднання клієнт може передавати серверу різноманітні команди для керування датчиком ліній або отримання інформації про його стан.
- Отримання даних. Клієнт може отримувати від сервера дані, наприклад, результати вимірювань датчика.
- Закриття з'єднання. Після завершення роботи клієнт закриває з'єднання з сервером.

Для реалізації клієнтської частини було використано наступні бібліотеки:

Windows.h: надає доступ до функцій операційної системи Windows.

Winsock2.h: бібліотека для роботи з мережевими сокетами.

Ws2tcpip.h: додаткова бібліотека для роботи з TCP/IP.

stdlib.h, stdio.h, fstream, iostream: стандартні бібліотеки C++ для роботи з введенням/виведенням даних, файлами та маніпуляціями з пам'яттю.

Для підвищення читабельності коду та спрощення його модифікації в програмі використовуються наступні константи:

DEFAULT\_BUFLen: визначає максимальний розмір буфера для передачі даних.

DEFAULT\_PORT: визначає номер порту, на якому слухає сервер.

### ***Ініціалізація бібліотеки Winsock***

```
WSADATA wsaData;
SOCKET ConnectSocket = INVALID_SOCKET;
struct addrinfo* result = NULL, * ptr = NULL, hints;
int iResult;
int recvbuflen = DEFAULT_BUFLen;
iResult = WSASStartup(MAKEWORD(2, 2), &wsaData);
if (iResult != 0) {
    printf("WSASStartup failed with error: %d\n", iResult);
    return 1;
}
```

Цей фрагмент коду виконує ініціалізацію бібліотеки Winsock, яка необхідна для роботи з мережевими функціями в операційній системі Windows. Функція WSASStartup ініціалізує бібліотеку Winsock і встановлює версію, яка



підтримується додатком. Якщо ініціалізація не вдалася, програма виводить повідомлення про помилку та завершує свою роботу.

### ***Підготовка до підключення до сервера***

```
ZeroMemory(&hints, sizeof(hints));
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP;
iResult = getaddrinfo("127.0.0.1", DEFAULT_PORT, &hints, &result);
if (iResult != 0) {
    printf("getaddrinfo failed with error: %d\n", iResult);
    WSACleanup();
    return 1;
}
```

Перед встановленням з'єднання з сервером програма готує структуру `hints`, яка містить інформацію про тип сокета, протокол та сімейство адрес (IPv4 або IPv6). Функція `getaddrinfo` використовується для перетворення імені хоста або IP-адреси сервера та номера порту в структуру `addrinfo`, яка містить інформацію про адресу сервера, необхідну для встановлення з'єднання. Якщо функція `getaddrinfo` повертає помилку, то це означає, що не вдалося розпізнати адресу сервера або порт, і програма завершує свою роботу.

### ***Встановлення з'єднання з сервером***

```
// ... (попередній код)
for (ptr = result; ptr != NULL; ptr = ptr->ai_next) {
    // ... (код встановлення з'єднання)
}
// ... (решта коду)
```

Цей цикл проходить по списку адрес, отриманих за допомогою функції `getaddrinfo`, і намагається встановити з'єднання з сервером за кожною з цих адрес. Як тільки з'єднання встановлено, цикл переривається. Такий підхід забезпечує гнучкість і дозволяє програмі підключатися до сервера навіть якщо його IP-адреса змінилася.

### ***Обробка команд та передача даних***

```
while (menu) {
```

```
// ... (код обробки команд)
}
```

Цей блок коду реалізує основний цикл програми, в якому відбувається обробка команд користувача. Програма постійно очікує введення команди від користувача і виконує відповідні дії. Наприклад, якщо користувач вводить команду "1", програма передає запит на отримання файлу на сервер.

### ***Закриття з'єднання та очищення ресурсів***

```
iResult = shutdown(ConnectSocket, SD_RECEIVE);
if (iResult == SOCKET_ERROR) {
    // ... (обробка помилки)
}
closesocket(ConnectSocket);
WSACleanup();
```

Після завершення роботи програма виконує очищення ресурсів. Функція `shutdown` закриває з'єднання з сервером для приймання даних, а функція `closesocket` повністю закриває сокет. На завершення викликається функція `WSACleanup` для звільнення ресурсів бібліотеки `Winsock`.

## **3.5 Опис коду серверної програми**

Серверна частина системи відіграє центральну роль, забезпечуючи зберігання даних, обробку запитів клієнтів та управління ресурсами. Вона працює за принципом сервісу, надаючи певні функції іншим програмам (клієнтам), що підключаються до неї. Зв'язок між клієнтом і сервером здійснюється за допомогою мережевих протоколів, що дозволяє їм взаємодіяти на відстані.

Розвиток серверної частини є критичним етапом створення будь-якої клієнт-серверної системи. Адже саме сервер визначає надійність, масштабованість та функціональність всієї системи. Тому детальний аналіз та розробка серверної частини є ключовими завданнями при створенні будь-якого програмного забезпечення.

Серверна частина, як правило, складається з кількох основних компонентів:

- Слухач сокетів: цей компонент постійно очікує на вхідні з'єднання від клієнтів.
- Потоки обробки запитів: коли клієнт підключається, створюється новий потік, який обробляє запит клієнта.
- База даних: для зберігання даних сервер часто використовує базу даних.
- Логіка бізнес-процесів: ця частина реалізує основну функціональність сервера, наприклад, обробку запитів, автентифікацію користувачів, виконання операцій з даними.

Для реалізації серверної частини в даному проекті використовується мова програмування C++ та бібліотека Winsock. Бібліотека Winsock надає набір функцій для створення мережових додатків в операційній системі Windows.

Основні етапи розробки серверної частини:

1. Ініціалізація бібліотеки Winsock. На початку виконання програми відбувається ініціалізація бібліотеки Winsock для забезпечення роботи з мережевими функціями.
2. Створення сокета. Створюється сокет, який слухає на вказаному порту в очікуванні з'єднань від клієнтів.
3. Прийняття з'єднань. Коли клієнт підключається, сервер приймає з'єднання і створює новий потік для обробки цього клієнта.
4. Обробка запитів. У кожному потоці відбувається обробка запитів, отриманих від клієнта. Залежно від типу запиту, сервер виконує відповідні дії, наприклад, читає дані з бази даних, виконує обчислення або керує виконавчими пристроями.
5. Відправка відповідей. Після обробки запиту сервер відправляє відповідь клієнту.
6. Закриття з'єднання. Після завершення обробки запиту з'єднання з клієнтом закривається.

На початку виконання програми здійснюється ініціалізація необхідних для роботи з мережею структур даних. Зокрема, ініціалізується структура WSADATA

для взаємодії з бібліотекою Winsock, створюється дескриптор сокета SOCKET для прослуховування вхідних з'єднань, а структура addrinfo заповнюється інформацією про адресу і порт сервера. Ці дії є підготовчим етапом перед встановленням з'єднань з клієнтами і подальшою обробкою їх запитів.

```
extern HANDLE hCOMPort; int key = 0;
using namespace std; bool check = true;
int cdecl main(void) { WSADATA wsaData; int iResult;
int con = 1;
SOCKET ListenSocket = INVALID_SOCKET; struct addrinfo* result = NULL;
struct addrinfo hints; conCom();
cout << "\n"; bool menu = true;
iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);
if (iResult != 0)
{
    printf("WSAStartup failed with error: %d\n", iResult); return 1;
}
int sizehints = sizeof(hints);
ZeroMemory(&hints, sizeof(hints));
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP; hints.ai_flags = AI_PASSIVE;
std::cout << "Server started \n";
//запуск потоків запису та читання COM порта COMPortStartThreads();
```

Для забезпечення взаємодії з клієнтами сервер створює сокет і асоціює його з конкретним мережевим інтерфейсом та номером порту. Цей процес називається прив'язкою сокета. Після цього сервер переходить у режим очікування вхідних з'єднань від клієнтів.

```
while (con == 1)
{
    iResult = getaddrinfo("127.0.0.1", DEFAULT_PORT, &hints, &result); if (iResult != 0)
    {
        printf("getaddrinfo failed with error: %d\n", iResult);
        WSACleanup();
        return 1;
    }
}
```

```

}
ListenSocket = socket(result->ai_family, result->ai_socktype, result->ai_protocol);
if (ListenSocket == INVALID_SOCKET)
{
printf("socket failed with error: %ld\n", WSAGetLastError()); freeaddrinfo(result);
WSACleanup(); return 1;
}
iResult = bind(ListenSocket, result->ai_addr, (int)result->ai_addrlen);
if (iResult == SOCKET_ERROR)
{
printf("bind failed with error: %d\n", WSAGetLastError());
freeaddrinfo(result);
closesocket(ListenSocket);
WSACleanup();
return 1;
}
freeaddrinfo(result);
SOCKET ClientSocket = INVALID_SOCKET;
iResult = listen(ListenSocket, SOMAXCONN);

```

Сервер працює за принципом файл-сервера, надаючи клієнтам доступ до необхідних файлів. Після встановлення з'єднання клієнт відправляє серверу запит на передачу файлу, а сервер, в свою чергу, передає цей файл клієнту. Такий підхід забезпечує зручний доступ до файлів з різних пристроїв.

```

ClientSocket = accept(ListenSocket, NULL, NULL); if (ClientSocket ==
INVALID_SOCKET)
{
printf("accept failed with error: %d\n", WSAGetLastError()); closesocket(ListenSocket);
WSACleanup();
return 1;
}
else
{
send(ClientSocket, (char*)&check, sizeof(check), 0);
}
closesocket(ListenSocket);

```

```

menu = true;
while (menu==true) {
recv(ClientSocket, (char*)&key, sizeof(key), 0);
int k = 0;
const int BUFFER_SIZE = 1024;
int byRecv;
std::ofstream file; while (key == 0xC1)
{
Завершення роботи сервера. if (key == 4)
{
key = 0; menu = false;
}
}
void COMTerminate(void); CloseHandle(hCOMPort);
return 0; }

```

### 3.6. Робота системи розумний будинок

Для детального аналізу функціонування системи було проведено моделювання її роботи в програмному середовищі Proteus 8 Professional. Цей етап дозволив візуалізувати взаємодію всіх компонентів системи та перевірити коректність розроблених алгоритмів.

На першому етапі в середовищі Proteus було створено детальну схему, яка відображає фізичне підключення всіх компонентів системи: мікроконтролера Arduino Uno, датчиків DHT11 та PIR, а також порту COM. Ця схема дозволила перевірити правильність розподілу сигналів та взаємозв'язків між компонентами.

Паралельно з розробкою схеми в середовищі Proteus було розроблене програмне забезпечення для мікроконтролера Arduino Uno. Програмний код, написаний на мові C++ в середовищі Arduino IDE, забезпечує збір даних з датчиків, обробку цих даних та керування виконавчими пристроями. Скомпільований код був завантажений в мікроконтролер для подальшої перевірки його роботи в моделі.

В моделі Proteus було налаштовано взаємодію між віртуальними компонентами, що дозволило симулювати роботу реальної системи. Датчики DHT11 та PIR генерували тестові дані, які передавалися на мікроконтролер

Arduino Uno. Мікроконтролер обробляв ці дані відповідно до закладених алгоритмів і керував роботою інших компонентів системи (рис.3.4).

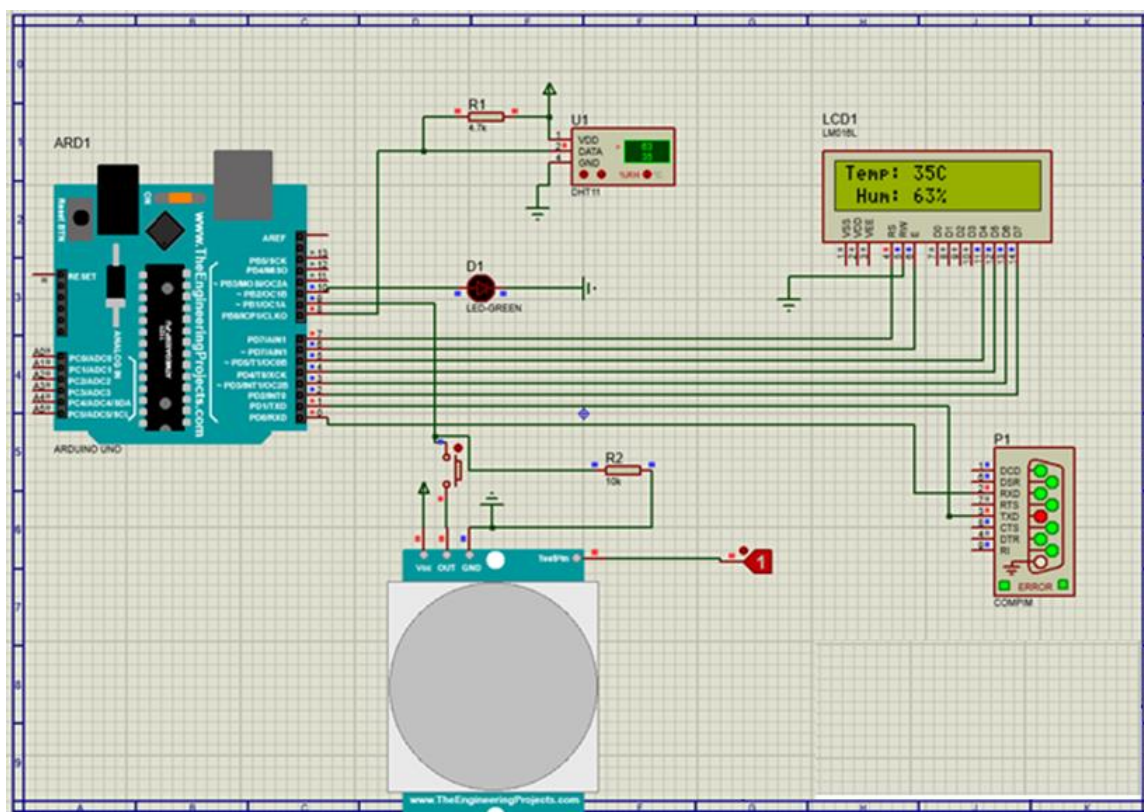


Рис. 3.4. Виконання симуляції Arduino UNO в Proteus 8 Professional

Для забезпечення взаємодії між різними компонентами системи ми використовуємо віртуальні COM-порти, створені за допомогою програми Virtual Null. Ця програма дозволяє нам симулювати роботу реальних послідовних портів і записувати передані дані у файл для подальшого аналізу. Незважаючи на деякі проблеми з візуалізацією даних у реальному часі, програма надійно записує всю необхідну інформацію (рис.3.5).

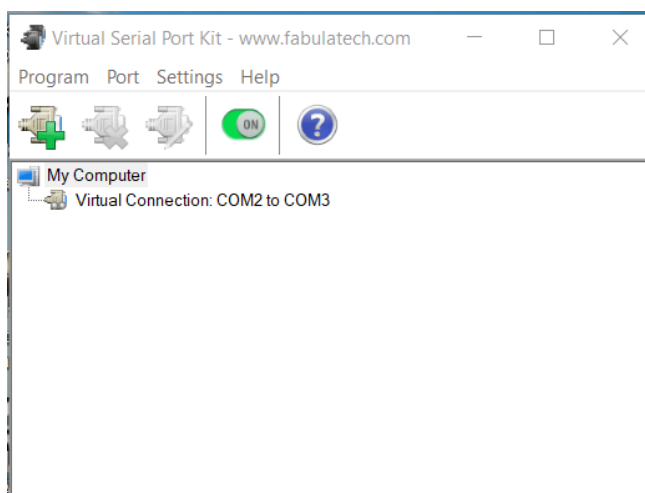


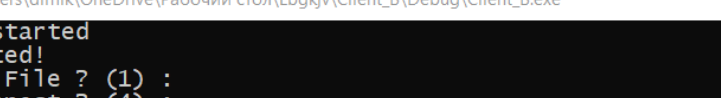
Рис. 3.5. Виконання програми Virtual Null

Для забезпечення коректної роботи системи ми налаштовуємо програмне середовище. У програмі Proteus конфігуруємо схему згідно з рисунком 3.4, а в програмі Virtual Null створюємо віртуальні послідовні порти для перехоплення та запису даних. Це дозволяє нам створити контрольоване середовище для розробки та тестування програмного забезпечення, що взаємодіє з фізичними пристроями (рис.3.6).

[illegible]

Рис. 3.6. Виконання програми серверної частини

Сервер працює в режимі очікування запитів від клієнтів. Після отримання запиту на передачу файлу, сервер зчитує необхідні дані з датчиків і відправляє їх клієнту. Після завершення передачі даних з'єднання з клієнтом закривається, і сервер повертається в режим очікування нових запитів. (рис.3.7).



```
C:\Users\dimik\OneDrive\Рабочий стол\Lbgkj\Client_B\Debug\Client_B.exe
User started
Connected!
Take File ? (1) :
Disconnect ? (4) :
1
File taking...
File received: Datatem.txt
Take File ? (1) :
Disconnect ? (4) :
4
Connection closed
Для продолжения нажмите любую клавишу . . .
```

Рис. 3.7. Виконання програми клієнтської частини

Файл виконує функцію сховища для даних, отриманих від датчиків, забезпечуючи їх збереження для подальшого аналізу та обробки (рис.3.8).





Проведена ретельна перевірка функціональності системи в різних режимах експлуатації. Були виявлені та усунені всі виявлені недоліки.

Результати проведеного дослідження демонструють, що платформа Arduino є ефективним інструментом для створення систем автоматизації житлових приміщень. Розроблена система "розумний будинок" відкриває нові можливості для підвищення комфорту та якості життя.

## ВИСНОВКИ

У рамках магістерського дослідження було розроблено автоматизовану систему керування модулями розумного будинку на базі ARDUINO.

Для проведення комплексного дослідження та розробки системи "розумний будинок" на основі мікроконтролера Arduino було проаналізовано сучасні наукові публікації, присвячені можливостям платформи Arduino в контексті створення систем автоматизації житлових приміщень. Особлива увага приділялася дослідженням, що висвітлюють архітектуру, програмне забезпечення та застосування Arduino в системах "розумний будинок". Вивчено діючі стандарти та технічні вимоги до пристроїв "розумного будинку", що базуються на Arduino, включаючи протоколи зв'язку (Wi-Fi, Bluetooth, Zigbee тощо), стандарти безпеки (IoT, кібербезпека) та вимоги до сумісності з іншими системами автоматизації. Проведено аналіз наукових статей, опублікованих у провідних фахових журналах, що стосуються технологій "розумного будинку" з використанням Arduino. Особливий інтерес представляли дослідження, в яких представлено результати експериментальних робіт, порівняльний аналіз різних рішень та оцінку ефективності систем. Вивчено технічну документацію на використовувані моделі мікроконтролерів Arduino, включаючи описи апаратних засобів, програмних інтерфейсів та прикладів застосування. Проаналізовано опубліковані проекти, випадки використання та відгуки користувачів систем "розумний будинок" на основі Arduino, які були знайдені в наукових статтях, блогах, на форумах та інших відкритих джерелах. Зібрано та проаналізовано дані про використання та ефективність систем "розумний будинок" на базі Arduino, отримані від реальних користувачів та інженерів, що займаються впровадженням таких систем.

Проведене дослідження дозволило оптимізувати вибір компонентів та розробити ефективне програмне забезпечення. Результатом роботи стала функціональна система, здатна дистанційно збирати дані з датчиків та керувати різними пристроями в будинку.

На основі проведеного аналізу було обрано оптимальний набір компонентів, включаючи платформу Arduino Uno, датчики DHT11 та PIR.

Було також розглянуто різні середовища розробки, зокрема Arduino IDE, Proteus 8 Professional, Visual Studio та Virtual Null Modem. Використання цих інструментів дозволило ефективно розробляти, моделювати та тестувати програмне забезпечення для системи. Ключовим етапом стала розробка програмного забезпечення для Arduino IDE, а також клієнтської та серверної програм. Розробка коду для кожної частини системи була ретельно виконана, що дозволило забезпечити надійну роботу системи.

В результаті проведеного дослідження була розроблена функціональна система "розумний будинок" на основі платформи Arduino. Система забезпечує автоматизацію та дистанційний контроль за різними параметрами домашнього середовища, такими як температура, вологість та освітлення. Розроблений прототип може слугувати основою для створення більш складних систем автоматизації житлових приміщень.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Томас М. Сіггу, Джеймс Ф. Келлер. Digital Transformation: Survive and Thrive in an Era of Mass Extinction. 2019 рік. 256 с.
2. Розумний дім [Електронний ресурс] URL: <https://stylus.ua/uk/articles/528.html> (дата звернення: 04.11.2024)
3. Розумний будинок Fibaro [Електронний ресурс] URL: <https://www.fibaro.com/ru/> (дата звернення: 10.11.2024)
4. Розумний будинок Broadlink [Електронний ресурс] URL: <https://broadlink.com.ua/> (дата звернення: 09.11.2024)
5. Розумний будинок AJAX [Електронний ресурс] URL: <https://ajax.systems/ua/> (дата звернення: 08.11.2024)
6. Послідовний порт [Електронний ресурс] URL: [http://ni.biz.ua/3/3\\_5/3\\_55535\\_posledovatelniy-port.html](http://ni.biz.ua/3/3_5/3_55535_posledovatelniy-port.html) (дата звернення: 21.11.2024)
7. Петин В.В. Створення розумного будинку на базі Arduino. ДМК ПРЕСС, 2018. 449 с.
8. П. Шантану, В. Кумар Robotic Process Automation: A Primer. 2018 рік. 200 с.
9. Мікроконтролер [Електронний ресурс] URL: [https://elprivod.nmu.org.ua/ua/interesting/what\\_is\\_mp\\_mc\\_plc.php](https://elprivod.nmu.org.ua/ua/interesting/what_is_mp_mc_plc.php) (дата звернення: 12.11.2024)
10. Мартін Форд. The Rise of Robots: Technology and the Threat of a Jobless Future. 2020 рік. 368 с.
11. Датчики руху [Електронний ресурс] URL: <https://www.brille.ua/ua/datchiki-dvizheniya-vidy/> (дата звернення: 20.11.2024)
12. Датчики вологості [Електронний ресурс] URL: <https://sitemasters.com.ua/elektroobladnannja/datchiki-vologosti-i-temperaturi-dlja/> (дата звернення: 16.11.2024)

13. Датчик температури [Електронний ресурс] URL: <https://ao-tera.com/ua/technology/types-of-temperature-sensors> (дата звернення: 14.11.2024)
14. Датчик вологості та температури DHT11 [Електронний ресурс] URL: <https://arduino.ua/prod185-datchik-vlajnosti-i-temperatyri-dht11> (дата звернення: 22.11.2024)
15. Visual Studio та .NET [Електронний ресурс] URL: [https://uk.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio) (дата звернення: 21.11.2024)
16. Robert Faludi. Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing. 2019. 322с.
17. Raspberry Pi [Електронний ресурс] URL: [https://uk.wikipedia.org/wiki/Raspberry\\_Pi](https://uk.wikipedia.org/wiki/Raspberry_Pi) (дата звернення: 12.11.2024)
18. Proteus 8 Professional [Електронний ресурс] URL: <https://ru.taiwebs.com/windows/download-proteus-professional-2164.html> (дата звернення: 24.11.2024)
19. PIR-датчик [Електронний ресурс] URL: <https://www.mini-tech.com.ua/datchik-dvizheniya-infrakrasniy-pir-sensor-hc-sr501> (дата звернення: 22.11.2024)
20. Mikell P. Groover Automation, Production Systems, and Computer-Integrated Manufacturing. 2019. IV. 815 с.
21. Microsoft Visual Studio [Електронний ресурс] URL: <https://visualstudio.microsoft.com/ru/vs/getting-started/> (дата звернення: 20.11.2024)
22. Marco Schwartz. Smart Home Automation with Arduino. 2018. 102с.
23. Marco Schwartz. Home Automation with Arduino: Automate your Home using OpenSource Hardware. 2021. 176 с 12. DATASHEET ARDUINO MEGA 2560. 2020. URL: <https://octopart.com/datasheet/arduino+mega+2560+rev3-arduino-29408153> (дата звернення: 01.11.2024).

24. Marco Schwartz. Building Smart Homes with Raspberry Pi Zero. 2018. 196 с.
25. Marco Schwartz. Arduino: Building a Smart Home with Arduino and Android. 2019 рік. 201 с.
26. ESP8266 [Електронний ресурс] URL: <https://alexgyver.ru/lessons/esp8266/> (дата звернення: 12.11.2024)
27. Arduino UNO [Електронний ресурс] URL: [https://uk.wikipedia.org/wiki/Arduino\\_Uno](https://uk.wikipedia.org/wiki/Arduino_Uno) (дата звернення: 12.11.2024)
28. Arduino IDE [Електронний ресурс] URL: <https://www.arduino.cc/en/software> (дата звернення: 24.11.2024)
29. Alasdair Allan. Make: Bluetooth: Bluetooth LE Projects with Arduino, Raspberry Pi, and Smartphones. 2015. 256 с

## ДОДАТОК А

### Код програми для завантаження на плату Arduino UNO

```
include "DHT.h"
include <LiquidCrystal.h>
define DHTPIN 8
define DHTTYPE DHT11 LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

DHT dht(DHTPIN, DHTTYPE);

int buttonState = HIGH; // Початковий стан кнопки (підтягнуто до високого рівня) int
lastButtonState = HIGH; // Попередній стан кнопки

bool buttonTriggered = false; bool mov =false;

bool notmov = true;

void setup() {
pinMode(9,INPUT);
pinMode(10,OUTPUT)
; Serial.begin(9600);
lcd.begin(16, 2);

lcd.clear();
lcd.setCursor(5, 0);
lcd.print("Work!");
dht.begin();
}

void loop() {
int t =
dht.readTemperature();

int h = dht.readHumidity();

if ( isnan(t)) {
Serial.println(F("Failed to read from DHT sensor!"));
```



```

    lcd.clear();

    lcd.setCursor(5, 0);
    lcd.print("Error");
    return;

}

lcd.setCursor(0, 0); lcd.print("Temp: "); lcd.print(t); lcd.print("C ");

Serial.write('t'); delay(150); Serial.write((char)t);

delay(200); lcd.setCursor(0, 2); lcd.print(" Hum: "); lcd.print(h); lcd.print("%");

Serial.write('h'); delay(150); Serial.write((char)h); delay(200); if(digitalRead(9)==HIGH){
digitalWrite(10,HIGH); if(mov==false){

    Serial.write('m'); delay(500); mov=true; notmov=true;

}
}
else{
    digitalWrite(10,LOW)
    ; if(notmov==true){
    Serial.write('n');
    delay(500);
    notmov=false;

    }

    mov=false;
}
}
}

```

## ДОДАТОК Б

### Код програми клієнтської частини

```
define WIN32_LEAN_AND_MEAN
include <windows.h>
include <winsock2.h>
include <ws2tcpip.h>
include <stdlib.h>
include <stdio.h>
include <fstream>
include <iostream>
// Need to link with Ws2_32.lib, Mswsock.lib, and
Advapi32.lib
#pragma comment (lib, "Ws2_32.lib")
#pragma comment (lib, "Mswsock.lib")
#pragma comment (lib, "AdvApi32.lib")
define DEFAULT_BUFLen 512
define DEFAULT_PORT "27015"

int cdecl main(int argc, char** argv)
{

    WSADATA wsaData;
    SOCKET ConnectSocket = INVALID_SOCKET;
    struct addrinfo* result = NULL,
    * ptr = NULL, hints;

    //char recvbuf[DEFAULT_BUFLen]; int iResult;
    int recvbuflen = DEFAULT_BUFLen;

    std::cout << "User started \n";
    // Initialize Winsock
    iResult = WSASStartup(MAKEWORD(2, 2), &wsaData); if (iResult != 0) {
    printf("WSAStartup failed with error: %d\n", iResult); return 1;
    }

    ZeroMemory(&hints, sizeof(hints));
    hints.ai_family = AF_UNSPEC; hints.ai_socktype = SOCK_STREAM; hints.ai_protocol =
    IPPROTO_TCP;

    // Resolve the server address and port
    iResult = getaddrinfo("127.0.0.1", DEFAULT_PORT, &hints, &result); if (iResult != 0)
    {
        printf("getaddrinfo failed with error: %d\n", iResult);
        WSACleanup();
        return 1;
    }
}
```

```

}

// Attempt to connect to an address until one succeeds for (ptr = result; ptr != NULL; ptr = ptr-
>ai_next)
{

    // Create a SOCKET for connecting to server ConnectSocket = socket(ptr->ai_family, ptr-
    >ai_socktype,
        ptr->ai_protocol);
    if (ConnectSocket == INVALID_SOCKET) {
        printf("socket failed with error: %ld\n", WSAGetLastError()); WSACleanup();
        return 1;
    }

    // Connect to server.
    iResult = connect(ConnectSocket, ptr->ai_addr, (int)ptr->ai_addrlen); if (iResult ==
    SOCKET_ERROR)
    {
        closesocket(ConnectSocket); ConnectSocket = INVALID_SOCKET; continue;
    }
    break;
}

freeaddrinfo(result);

bool s;

bool stop;
if (ConnectSocket == INVALID_SOCKET)
{
    printf("Unable to connect to server!\n");
    WSACleanup();
    return 1;
}
else {
    printf("Conected!\n");
    recv(ConnectSocket, (char*)&s, sizeof(s), 0);
    stop = !s;
}
bool menu = true;
int key = 0;
// = false;

```

```

int t;
//form pac stop
= false; while
(menu)
{
    std::cout << " Take File ? (1) : " <<
    std::endl; std::cout << " Disconect ? (4) :
    " << std::endl; std::cin >> key;

    int k = 0;
    const int BUFFER_SIZE =
    1024; char
    bufferFile[BUFFER_SIZE];
    char
    fileRequested[FILENAME_MAX];
    std::ofstream file;
    while (key == 1)
    {
        if (key == 1) {
            std::cout << "File taking...\n";
            key = 0xC1;
        }
        if (k == 0) send(ConnectSocket, (char*)&key, sizeof(key), 0);

        bool clientClose = false;
        int codeAvailable =
        404;
        const int fileNotFound =
        404; long fileRequestedsized
        = 0;

        do {
            int fileDownloaded = 0;
            memset(fileRequested, 0,
            FILENAME_MAX);
            int byRecv = recv(ConnectSocket, (char*)&fileRequested, FILENAME_MAX, 0);

            if (byRecv == 0 || byRecv == -1) { clientClose = true;
            }

```

```

byRecv = recv(ConnectSocket, (char*)&codeAvailable, sizeof(int), 0);
if (byRecv == 0 || byRecv == -1)
{
    clientClose = true; break;
}
if (codeAvailable == 200) {
    byRecv = recv(ConnectSocket, (char*)&fileRequestedsized, sizeof(long), 0);
    if (byRecv == 0 || byRecv == -1)
    {
        clientClose = true; break;
    }
    file.open(fileRequested, std::ios::binary | std::ios::trunc);
    do
    {
        memset(bufferFile, 0, BUFFER_SIZE);
        byRecv = recv(ConnectSocket, (char*)&bufferFile, BUFFER_SIZE, 0); if (byRecv == 0 ||
byRecv == -1) {
            clientClose = true; break;
        }
        file.write(bufferFile,
            byRecv); fileDownloaded
            += byRecv;
    } while (fileDownloaded <
fileRequestedsized); file.close();
    std::cout << "File recived: " << fileRequested <<
    std::endl; clientClose = true;

}
else if (codeAvailable == 404) {
    std::cout << "Can't open file or file not found!" <<
    std::endl; key = 0;
    clientClose = true;
    break;
}

} while (!clientClose);
key = 0;
k++;

```

```

    }

    if (key == 4)
    {
        send(ConnectSocket, (char*)&key, sizeof(key), 0);
        printf("Connection closed\n");
        menu = false;
    }
}
iResult = shutdown(ConnectSocket, SD_RECEIVE);

if (iResult == SOCKET_ERROR) {
    printf("shutdown failed with error: %d\n",
        WSAGetLastError()); closesocket(ConnectSocket);
    WSACleanup();
    return 1;
}
// cleanup
closesocket(ConnectSocket);
WSACleanup();
system("pause");
return 0;
}

```

## ДОДАТОК В

### Код програми розробки серверної частини

```
undef UNICODE

define WIN32_LEAN_AND_MEAN

include <windows.h>
include <winsock2.h>
include <ws2tcpip.h>
include <stdlib.h>
include <stdio.h>
include <stdio.h>
include <fstream>

include <iostream>
include "ard.h"

// Need to link with Ws2_32.lib
#pragma comment (lib, "Ws2_32.lib")
// pragma comment (lib, "Mswsock.lib")

define DEFAULT_BUFLen 512
define DEFAULT_PORT "27015"

extern HANDLE hCOMPort;

int key = 0;

using namespace
std; bool check =
true;

int cdecl main(void)
{
    WSADATA
    wsaData; int iResult;

    int con = 1;

    SOCKET ListenSocket = INVALID_SOCKET;
```

```

struct addrinfo* result =
NULL; struct addrinfo
hints; conCom();

cout << "\n";

bool menu = true;

iResult = WSASStartup(MAKEWORD(2, 2),
&wsaData); if (iResult != 0)
{
    printf("WSASStartup failed with error: %d\n", iResult); return 1;
}
int sizehints = sizeof(hints);
ZeroMemory(&hints, sizeof(hints)); hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP; hints.ai_flags = AI_PASSIVE;
std::cout << "Server started \n";

//запуск потоків запису та читання COM порта COMPortStartThreads();

while (con == 1)
{
    // Resolve the server address and port
    iResult = getaddrinfo("127.0.0.1", DEFAULT_PORT, &hints, &result); if (iResult != 0)
    {
        printf("getaddrinfo failed with error: %d\n", iResult);
        WSACleanup();

        return 1;
    }
    // Create a SOCKET for the server to listen for client connections.
    ListenSocket = socket(result->ai_family, result->ai_socktype, result-
>ai_protocol); if (ListenSocket == INVALID_SOCKET)

```



```

{

    printf("socket failed with error: %ld\n", WSAGetLastError());
    freeaddrinfo(result);

    WSACleanup();
    return 1;
}

// Setup the TCP listening socket
iResult = bind(ListenSocket, result->ai_addr, (int)result->ai_addrlen); if (iResult ==
SOCKET_ERROR)

{

    printf("bind failed with error: %d\n", WSAGetLastError());
    freeaddrinfo(result);
    closesocket(ListenSocket); WSACleanup();

    return 1;
}
freeaddrinfo(result);

SOCKET ClientSocket = INVALID_SOCKET; iResult = listen(ListenSocket, SOMAXCONN);
if (iResult == SOCKET_ERROR)

{

    printf("listen failed with error: %d\n", WSAGetLastError());
    closesocket(ListenSocket);
    WSACleanup();
    return 1;
}

// Accept a client socket
ClientSocket = accept(ListenSocket, NULL,
NULL); if (ClientSocket ==
INVALID_SOCKET) {

    printf("accept failed with error: %d\n",
WSAGetLastError()); closesocket(ListenSocket);

```

```

WSACleanup();
return 1;

}
else { send(ClientSocket, (char*)&check,
sizeof(check), 0); } closesocket(ListenSocket);

menu = true;
while (menu==true)
{
    recv(ClientSocket, (char*)&key, sizeof(key), 0); int k = 0;

    const int BUFFER_SIZE = 1024; int byRecv;

    std::ofstream file; while (key == 0xC1)
    {

        bool clientClose = false;

        char fileRequested[FILENAME_MAX]; const int fileAvailable = 200;

        const int fileNotFound = 404; const int BUFFER_SIZE = 1024; char
        bufferFile[BUFFER_SIZE];

    std::ifstream file; do {
        int fileDownloaded = 0; memset(fileRequested, 0, FILENAME_MAX);
        if (strcpy_s(fileRequested, FILENAME_MAX, "Datatem.txt") != 0) { cout << "Errrr";
        }

        byRecv = send(ClientSocket,(char*)&fileRequested, FILENAME_MAX, 0);
        if (byRecv == 0 || byRecv == -1)
        {

            clientClose = true; break;

        }

    file.open(fileRequested, std::ios::binary); if (file.is_open())
    {
        int bySendinfo = send(ClientSocket, (char*)&fileAvailable, sizeof(int), 0); if (bySendinfo == 0 ||
        bySendinfo == -1) {
            clientClose = true;

```

```

    }
    file.seekg(0, std::ios::end); long fileSize = file.tellg();
    bySendinfo = send(ClientSocket, (char*)&fileSize, sizeof(long), 0); if (bySendinfo == 0 ||
    bySendinfo == -1) {
        clientClose = true;
    }
    file.seekg(0, std::ios::beg); do {
        file.read(bufferFile, BUFFER_SIZE); if (file.gcount() > 0)
            bySendinfo = send(ClientSocket, (char*)&bufferFile, file.gcount(), 0); if (bySendinfo == 0 ||
            bySendinfo == -1) {
                clientClose = true; break;
            }
    } while (file.gcount() > 0);

    file.close();
    std::cout << "File sended!!!\n"; clientClose = true;
    key = 0;

    }
    else
    {

        int bySendCode = send(ClientSocket, (char*)&fileNotFound, sizeof(int), 0); if (bySendCode == 0 ||
        bySendCode == -1) {
            clientClose = true; break;
        }
        } while (!clientClose);

    }

    if (key == 4) { key = 0; menu = false;
    }
    }

}

void COMTerminate(void); CloseHandle(hCOMPort);
return 0;
}

```