МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій
(повне найменування інституту, назва факультету)

Кафедра комп'ютерної інженерії та електромеханіки
(повна назва кафедри)

ДИПЛОМНА БАКАЛАВРСЬКА РОБОТА

на тему

"СИСТЕМА РОЗУМНИЙ БУДИНОК" НА ОСНОВІ МІКРОКОНТРОЛЕРА ARDUINO

Виконав: студент групи БКІ-19

спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

Буряк Д.О.
(прізвище та ініціали)

Керівник д.т.н., проф. Злотенко Б.М.
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
Інститут інженерії та інформаційних технологій
Кафедра комп'ютерної інженерії та електромеханіки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп‟ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри КІЕМ

_____проф. Злотенко Б.М.

"_____" _____2023 року

З А В Д А Н Н Я
НА ДИПЛОМНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Буряк Дмитрій Олександрович
(прізвище, ім'я, по батькові)

1. Тема дипломної бакалаврської роботи **"Система розумний будинок" на основі мікроконтролера Arduino**

Науковий керівник роботи Злотенко Борис Миколайович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
д.т.н., професор
затверджені наказом вищого навчального закладу від 15.03.2023 № 75-уч.
2. Строк подання студентом роботи 00 червня 2023 року
3. Вихідні дані до дипломної бакалаврської роботи: технічне завдання, технічна та патентна література.
4. Зміст дипломної бакалаврської роботи (перелік питань, які потрібно розробити): 1. Аналітичний огляд існуючих систем розумний будинок на основі різних мікроконтролерів. 2. Вибір інструментів для "Система розумний будинок" на основі мікроконтролера Arduino. 3. Розробка "Система розумний будинок" на основі мікроконтролера Arduino.
5. Дата видачі завдання 10.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної бакалаврської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	06.03.2023	
2	Розділ 1. Аналітичний огляд існуючих систем розумний будинок на основі різних мікроконтролерів	17.03.2023	
3	Розділ 2. Вибір інструментів для "Система розумний будинок" на основі мікроконтролера Arduino	29.03.2023	
4	Розділ 3. Розробка "Системи розумний будинок" на основі мікроконтролера Arduino	28.04.2023	
6	Висновки	30.05.2023	
7	Оформлення дипломної бакалаврської роботи (чистовий варіант)	05.05.2023	
8	Здача дипломної бакалаврської роботи на кафедру для рецензування (за 14 днів до захисту)	08.06.2023	
9	Перевірка дипломної бакалаврської роботи на наявність співпадінь (за 10 днів до захисту)	09.06.2023	
10	Подання дипломної бакалаврської роботи на затвердження завідувачу кафедри (за 7 днів до захисту)	10.06.2023	

Студент _____ Буряк Д.О.
(підпис) (прізвище та ініціали)

Науковий керівник роботи _____ Злотенко Б.М.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Буряк Д. О. "Система розумний будинок" на основі мікроконтролера Arduino : дипломна бакалаврська робота за спеціальністю 123 Комп'ютерна інженерія / Д. О. Буряк ; наук. кер. Б. М. Злотенко. – Київ : КНУТД, 2023. – 73 с.

Дипломна бакалаврська робота за спеціальністю 123 Комп'ютерна інженерія, освітньою програмою «Комп'ютерні системи та мережі». – Київський національний університет технологій та дизайну, Київ, 2023 рік.

Робота присвячена дослідженню і розробленню Системи розумний будинок на основі мікроконтролера Arduino.

В першому розділі в повному обсязі описано, та проаналізовані систем розумний будинок на основі різних мікроконтролерів.

В другому розділі проаналізовано інструменти для виконання роботи.

В третьому розділі описана розробка проекту.

В четвертому представлений виготовлений прилад та інструкція щодо використання.

Пояснювальна записка виконана в текстовому редакторі Microsoft Word, в роботі використані програми Microsoft Visual C++, Arduino IDE, Proteus 8 Professional.

Ключові слова: система розумний будинок, мікроконтролер, arduino, аналіз система розумний будинок

ABSTRACTS.

Buryak D.O. "Smart home system" based on the Arduino microcontroller. - Manuscript.

Bachelor's thesis in the specialty 123 Computer Engineering, educational program "Computer Systems and Networks". - Kyiv National University of Technology and Design, Kyiv, 2023.

The work is devoted to the research and development of the Smart Home System based on the Arduino microcontroller.

In the first section, the smart home systems based on different microcontrollers are fully described and analysed.

The second section analyses the tools for performing the work.

The third section describes the project development.

The explanatory note was written in the Microsoft Word text editor, Microsoft Visual C++, Arduino IDE, Proteus 8 Professional, Virtual Null Modem.

Keywords: smart home system, microcontroller, arduino, analysis of the smart home system

ЗМІСТ

ВСТУП

Актуальність роботи. Тема "Система розумний будинок на основі мікроконтролера Arduino" є дуже актуальною в сучасному світі. З розвитком технологій та залежності більшості людей від комп'ютерів та гаджетів, зручність та безпека в домі стала однією з проблем у сфері домашнього життя.

Розробка та забезпечення Системи розумний будинок на основі мікроконтролера Arduino включає в себе побудову надійної систему, робота якої буде змодельована в програмі Proteus 8 Professional, яка буде давати можливість побачити дистанційно температуру повітря, вологість повітря та можливість дізнатися є хтось у кімнаті чи ні.

Отже, дипломна робота на тему "Система розумний будинок на основі мікроконтролера Arduino" буде актуальною і корисною, оскільки її результати дозволять розробити ефективну систему розумний будинок для забезпечення комфорту, зручності та безпеки домашнього життя.

Крім того, така робота буде сприяти розвитку нових технологій та методів в галузі розумних будинків, що є актуальним в контексті зростаючої необхідності людей в розумних будинках.

Метою роботи є розроблення системи розумний будинок на основі мікроконтролера Arduino, а також створення клієнт-серверної архітектури, у якій завдання або мережеве навантаження розподілені між постачальниками послуг, серверами, та замовниками послуг – клієнтами.

Отже, мета дипломної роботи полягає в тому, щоб дослідити та розробити ефективну систему розумного будинка на основі мікроконтролера Arduino що дозволить підвищити рівень захищеності дому та забезпечити комфорт та зручність. Дипломна робота також має на меті дослідити поточний стан різних систем розумних будинків. Результатом дослідження повинен бути розумний будинок на основі мікроконтролера Arduino, який буде передавати дані та зберігати через клієнт-серверну систему обробки.

Для досягнення поставленої мети в роботі вирішені такі **завдання:**

1. Аналіз потенційні системи розумний будинок на основі різних мікроконтролерів.
2. Вибір та побудова ефективної система розумний будинок на основі мікроконтролера Arduino.
3. Розробка програмного забезпечення для мікроконтролера Arduino.
4. Розробка та реалізація серверної та клієнтської системи обробки.
5. Тестування працездатності розумного будинку.
6. Розробити практичні рекомендації по вдосконаленню та розширенню функціональностей системи розумного будинка на основі мікроконтролера Arduino на майбутнє.

Результати дослідження дипломної роботи можуть бути корисні для кінцевих користувачів, які хотіли би впровадити систему "розумний будинок" у своїх домівках. Дипломна робота допоможе слугувати гідом у виборі та налаштуванні системи, а також надати їм знання про потенційні проблеми з безпекою та приватністю. Також дослідження може допомогти їм краще розуміти потенціал та обмеження мікроконтролерів Arduino в контексті систем "розумний будинок".

Об’єкт дослідження – створення системи розумного будинку.

Дипломна робота зосереджується на процесі проектування, розробки, налаштування та експлуатації системи автоматизації житлового простору, в основу якої покладено використання мікроконтролера Arduino. Цей процес включає в себе розробку алгоритмів керування домашніми пристроями, створення системи взаємодії з користувачем, а також реалізацію заходів щодо безпеки і надійності системи.

Предмет дослідження – розроблення системи розумного будинку та створити створення клієнт-серверної архітектури, у якій завдання або мережеве навантаження розподілені між постачальниками послуг, серверами, та замовниками послуг – клієнтами.

Методи досліджень. Аналіз документів та літературних джерел, експертні оцінки, тестування та моделювання.

Інформаційною базою досліджень є:

Науково-технічна література про Arduino та його можливості в контексті розробки систем "розумного будинку".

Стандарти та специфікації для пристроїв "розумного будинку", що базуються на Arduino, включаючи протоколи зв'язку, стандарти безпеки, технічні вимоги тощо.

Дослідження та статті в наукових журналах, пов'язаних з технологією розумного будинку, особливо ті, що використовують Arduino в якості основи.

Документація і технічні керівництва щодо конкретних моделей мікроконтролерів Arduino, які можуть використовуватися в розробці.

Опубліковані розробки, проекти та випадки використання систем "розумний будинок", що використовують Arduino, які можна знайти в наукових статтях, блогах, на форумах та інших джерелах.

Дані про використання та ефективність систем "розумний будинок" на базі Arduino, отримані від реальних користувачів та інженерів, які впроваджують такі системи.

Практичне значення отриманих результатів. Отримані результати дослідження можуть мати практичне значення, зокрема, щодо вдосконалення розумних будинків, забезпечення безпеки та комфорту їх власникам.

Структура та обсяг роботи. Дипломна робота бакалавра складається зі вступу, 3 розділів та висновків по них, загальних висновків, списку використаних джерел та додатків. Основний текст роботи викладений на 47 сторінках, містить 23 рисунка, список джерел з 18 найменувань. Загальний обсяг роботи, враховуючи додаток, складає 73 аркуша.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ СИСТЕМ РОЗУМНИЙ БУДИНОК

1.1. Огляд різних "систем розумний будинок"

Система розумного будинку - це інноваційна технологічна інфраструктура, яка поєднує різні пристрої та системи в будинку, щоб забезпечити автоматизацію, контроль та зручність для мешканців. Вона заснована на використанні розумних пристроїв, датчиків, мережі інтернет речей та програмного забезпечення для збору та обробки даних [1].

Розумний будинок має бути обладнаний різними системами, які взаємодіють між собою та з мешканцями, забезпечуючи різноманітні функції та послуги. Основними складовими системи розумного будинку можуть бути:

1. Управління освітленням: Завдяки автоматизації освітлення, мешканці можуть контролювати включення та вимикання світла, його яскравість і колір за допомогою додатків на смартфонах або голосових помічників. Це дозволяє створити різні настрої освітлення в будинку і економити електроенергію.

2. Управління опаленням та кондиціонуванням повітря: Система розумного будинку може автоматично регулювати температуру в приміщеннях залежно від погодних умов або графіку мешканців. Це забезпечує комфортні умови проживання та допомагає знизити енергоспоживання.

3. Безпека та системи моніторингу: Системи відеоспостереження, датчики пожежі, виявлення вторгнення та системи контролю доступу можуть бути інтегровані в розумну систему будинку. Це дозволяє мешканцям віддалено контролювати безпеку будинку, отримувати сповіщення про події та реагувати на них.

4. Автоматизація побутових пристроїв: Розумний будинок може керувати побутовими пристроями, такими як холодильники, пральні машини, посудомийні машини та інші. Ви можете ввімкнути їх, налаштувати режими роботи або отримувати повідомлення про стан пристроїв.

5. Автоматизація розеток та електроприладів: Розумні розетки дозволяють мешканцям управляти електроприладами з віддаленням. Ви можете ввімкнути або вимкнути пристрої, перевірити їх статус та встановити графік роботи для енергозбереження.

6. Розумна система бездротового зв'язку: Інтеграція бездротових технологій, таких як Wi-Fi або Bluetooth, дозволяє мешканцям контролювати систему розумного будинку з будь-якої точки в будинку або навіть віддалено через мобільний додаток.

Зручність системи розумного будинку полягає в тому, що мешканці можуть контролювати всі аспекти свого будинку за допомогою одного додатку на смартфоні або голосових помічників. Вони можуть віддалено керувати освітленням, опаленням, безпекою та побутовими пристроями, що надає їм зручність, ефективність та енергозбереження. Додатково, система розумного будинку може навчатися звичкам мешканців та автоматично адаптуватися до їхніх потреб, що робить проживання в будинку ще більш комфортним та економічним.

Система "Розумний будинок" Ajax - це високотехнологічний набір пристроїв, який дозволяє автоматизувати та контролювати різні аспекти вашого дому, працюючи на надійно зашифрованому двосторонньому радіозв'язку Jeweller. Ця технологія, розроблена компанією Ajax, гарантує безпеку та надійність системи (рис. 1.1).

Система Ajax має багато переваг:

1. Простота установки: Систему можна легко встановити без потреби в спеціальних навичках або інструментах.

2. Бездротовий зв'язок: Всі пристрої системи підключаються бездротово, що забезпечує гнучкість установки і зменшує необхідність проводів.

3. Велика зона дії сигналу: Система може покривати велику зону до 2000 метрів.

4. Автономна робота: Завдяки хабу з резервним джерелом живлення, система може працювати до 16 годин без підключення до електромережі.

5. Мультиплатформеність: Система підтримує Wi-Fi та GSM-зв'язок, що дає змогу контролювати її через смартфон (iOS або Android).

6. Багатофункціональність: Система може підтримувати до 100 пристроїв, що робить її ідеальною для великих будинків або комерційних приміщень.

Однак, система також має деякі недоліки:

1. Відсутність автономності датчиків: Система залежить від роботи центрального контролера .

2. Відсутність власної камери відеоспостереження: Необхідно підключати стороннє обладнання.

3. Управління тільки через телефон: Незважаючи на те, що це знімає необхідність встановлювати додаткові програми на ПК, для деяких користувачів це може бути не зручно.

Рисунок 1.1 – Розумний будинок Ajax

Загалом, Ajax "Розумний будинок" є багатофункціональною, надійною, зручною і компактною системою, яка має стильний дизайн та зрозумілий інтерфейс. Вона використовує передові технології для забезпечення безпеки та комфорту в домі, і це робить її однією з найкращих систем "Розумний будинок" на ринку [2].

Система "Розумний будинок" BroadLink представляє собою комплект сучасних цифрових пристроїв, призначених для інтелектуального управління різними системами в будинку, включаючи побутову техніку, освітлення, енергетику, охорону та багато іншого. Вона надає можливість кожному компоненту працювати як самостійно, так і взаємодіяти з іншими елементами системи [3].

Ось декілька ключових переваг системи BroadLink:

1. Швидка установка, підключення та налаштування: Система BroadLink легко встановлюється та налаштовується, що дозволяє швидко почати використовувати її.

2. Широкий асортимент датчиків: Система включає датчики вологості, температури, освітлення, шуму, забруднення повітря, що дозволяє контролювати різні аспекти середовища в будинку.

3. Автономна робота датчиків: Відсутність необхідності в центральному хабі робить кожен датчик в системі автономним, що збільшує гнучкість та надійність системи.

4. Контроль через Інтернет: Система BroadLink може контролюватися по Wi-Fi через Інтернет з будь-якої точки планети, що забезпечує зручність управління.

Однак, система також має деякі недоліки:

1. Обмежена дальність дії сигналу: Сигнал системи BroadLink може діяти на відстані до 50 метрів.

2. Відсутність резервного живлення хаба: Якщо мережа відключена, система може перестати працювати.

3. Пульт працює тільки на прийом сигналів: Це може обмежити функціональність системи.

Незважаючи на ці недоліки, BroadLink "Розумний будинок" є високоякісною системою, що займає впевнене друге місце в рейтингу систем розумного будинку. Вона відрізняється широким функціоналом, якісним програмним забезпеченням, простотою в установці та користуванні, а також доступною вартістю (рис. 1.2).

Рисунок 1.2 – Розумний будинок BroadLink

Система "Розумний будинок" Fibaro є професійним обладнанням для автоматизації та безпеки будинку. Вона пропонує широкий функціонал, але

вимагає встановлення та налаштування своєї апаратури досвідченими фахівцями (рис. 1.3).

Рисунок 1.3 – Розумний будинок Fibaro

Декілька переваг системи Fibaro включають:

1. Широкий асортимент датчиків та пристроїв: Система включає в себе різноманітні датчики та пристрої, що підтримують велику кількість сценаріїв для користувача.

2. Робота на базі протоколу Z-Wave: Це дозволяє системі успішно взаємодіяти з іншим обладнанням.

3. Розумна розетка: Вона відображає рівень енергоспоживання підключених пристроїв і автоматично вимикається при скачках напруги.

4. Голосове управління через сервіс Google: Хоча ця функція доступна тільки англійською мовою.

Однак, система також має деякі недоліки:

1. Висока вартість обладнання: Ціна обладнання Fibaro починається від 600 доларів.

2. Професійний монтаж та налаштування: Система вимагає професійного монтажу та налаштування, що може додати до вартості та складності встановлення.

3. Обмежена дальність сигналу: Сигнал системи діє на відстані до 50 метрів без перешкод.

Все враховуючи, система "Розумний будинок" Fibaro є високоякісною системою з широким функціоналом, але вона вимагає професійного встановлення та налаштування, що може зробити її менш доступною для деяких користувачів [4].

1.2. Огляд різних мікроконтролерів для розумного будинка

Мікроконтролер — це інтегральна схема, яка в собі об'єднує процесор, пам'ять та периферійні пристрої на одному кристалі. Він може виконувати різноманітні функції, залежно від того, як він був програмований.

Мікроконтролери зазвичай програмуються через середовища програмування, такі як Arduino IDE, STM32Cube, MPLAB для PIC мікроконтролерів, або інші. Вони використовуються в різних областях, включаючи вбудовані системи, автоматизацію, робототехніку, автомобільну промисловість і багато інших.

Основна мета мікроконтролера - це керування або обробка даних в електронних пристроях та системах. Він використовується для виконання конкретних завдань відповідно до програми, яка була завантажена в його пам'ять. Наприклад, у розумному будинку мікроконтролер може використовуватися для контролю різних систем, таких як освітлення, температура, безпека тощо [5].

Основні компоненти мікроконтролера включають:

1. Процесор (Центральний обчислювальний блок, ЦОБ): Виконує основні обчислення і керує роботою мікроконтролера.

2. Пам'ять: Зберігає програми і дані. Є два основних типи пам'яті - оперативна (RAM) для зберігання тимчасових даних і постійна (ROM, EEPROM, Flash) для зберігання програм та постійних даних.

3. Периферійні пристрої: Це можуть бути порти вводу/виводу (GPIO), модулі комунікації (SPI, I2C, UART, USB, Ethernet), аналогово-цифрові перетворювачі (ADC), цифро-аналогові перетворювачі (DAC), таймери, контролери переривань і багато інших.

4. Шина: Це система внутрішніх взаємозв'язків, яка з'єднує всі вищезазначені частини мікроконтролера і дозволяє їм спілкуватися між собою.

Arduino Uno – це мікроконтролерна плата, що базується на чипі ATmega328P. Ця плата є однією з найпопулярніших у родині Arduino, в основному через свою доступність і простоту використання, що робить її відмінним вибором для новачків.

Архітектура Uno є відкритою, тобто її схеми та програмне забезпечення доступні для вільного використання і модифікації. Це дозволяє вам створювати власні версії плати, якщо ви маєте достатній рівень знань.

Arduino Uno оснащений 14 цифровими вводоми-виводами, які можуть бути використані як входи або виходи, з них 6 можуть бути використані для виходу широтно-імпульсної модуляції (PWM). Він також має 6 аналогових входів, 16-мегагерцовий кварцевий резонатор, USB-порт для програмування і підключення до комп'ютера, живлення, ICSP-заголовок і кнопку скидання.

Одним із ключових аспектів Arduino Uno є його простота програмування. Він використовує власний мову програмування, який заснований на Wiring, та інтегрований середовище розробки (IDE), який дозволяє вам легко писати код і завантажувати його на плату. Це середовище розробки також забезпечує багато прикладів коду, які можуть допомогти вам почати [6].

Отже, Arduino Uno широко використовується для створення різноманітних DIY проектів, включаючи роботів, автоматизацію дому, музичні інструменти, іграшки, системи безпеки, і багато інших. Його гнучкість і простота використання

роблять його відмінним інструментом для ентузіастів електроніки, вчителів, студентів і професіоналів (рис. 1.4).

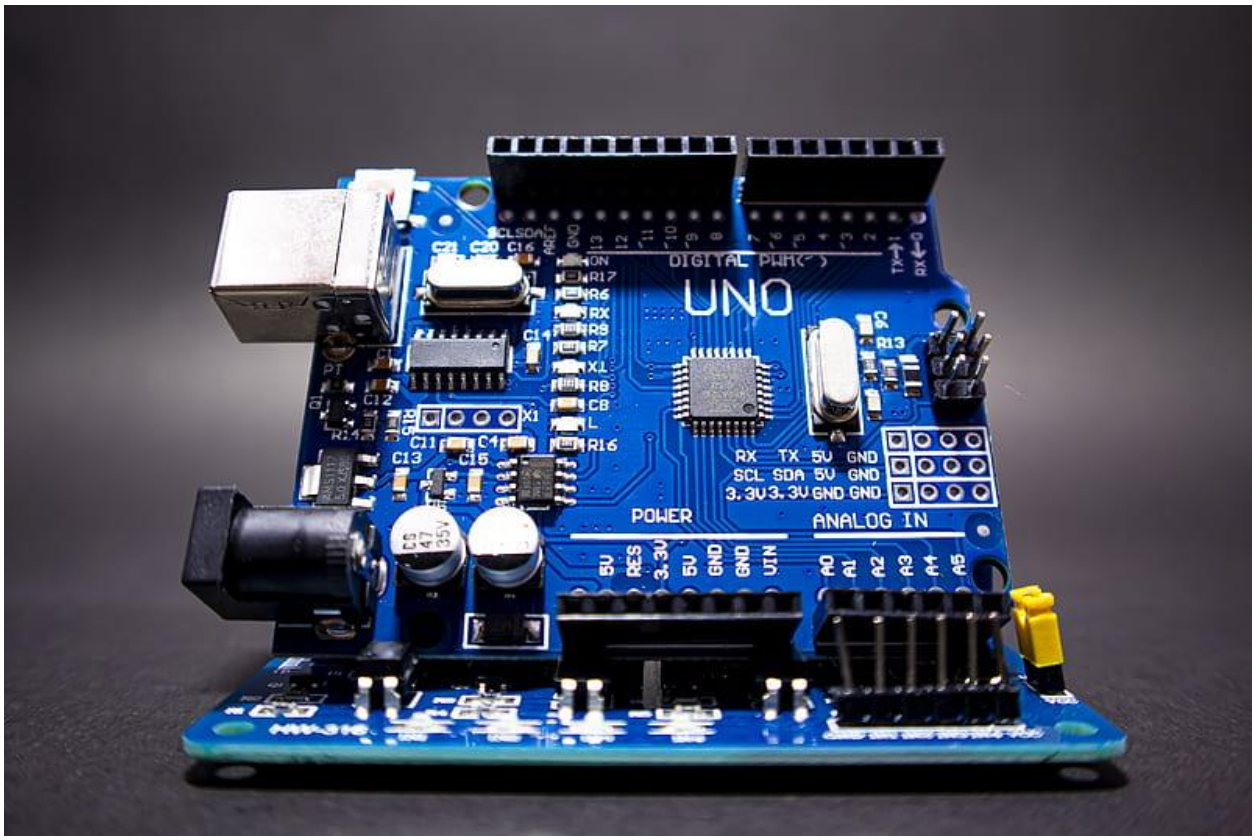

Рисунок 1.4 – Мікроконтролер Arduino UNO

Raspberry Pi - це серія одноплатових мікрокомп'ютерів, що були розроблені Raspberry Pi Foundation у Великобританії. Raspberry Pi зазвичай використовується для навчання інформатики, а також в різноманітних проектах та пристосуваннях, від робототехніки до "розумних" будинків.

Нова версія Raspberry Pi 4 - це потужний варіант в серії, який має багато особливостей, які роблять його ідеальним для розумного будинку. Воно має з'єднання Ethernet, Wi-Fi, Bluetooth, а також декілька USB-портів для підключення різноманітних пристроїв. Одна з важливих особливостей Raspberry Pi 4 є те, що він здатний працювати з різноманітними операційними системами, включаючи Raspbian, Ubuntu і навіть версії Windows 10 IoT Core. Це означає, що ви можете вибрати операційну систему, яка найкраще підходить для вашого проекту розумного будинку.

Raspberry Pi 4 може бути використаний для автоматизації різних систем у вашому будинку, включаючи освітлення, опалення, вентиляцію та кондиціонування. Ви також можете використовувати його для створення власної системи безпеки, яка може включати в себе камери відеоспостереження і датчики руху [7].

За допомогою підходящого програмного забезпечення, як-то Home Assistant або OpenHAB, Raspberry Pi 4 може служити в якості головної системи вашого розумного будинку, що забезпечує координацію і керування всіма підключеними пристроями та системами (рис. 1.5).

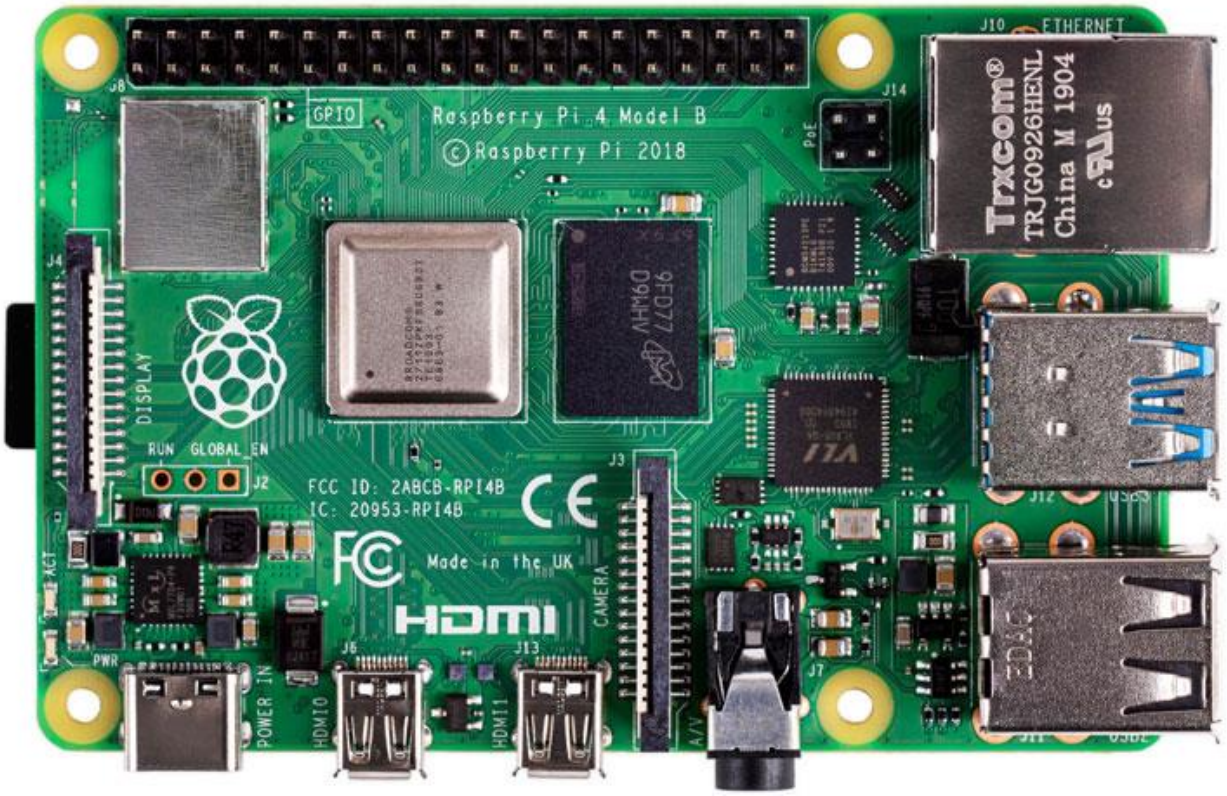

Рисунок 1.5 – Мікроконтролер Raspberry Pi 4

Також важливо згадати що Raspberry Pi - це платформа з відкритим вихідним кодом. Це означає, що існує велика спільнота розробників, які створюють та діляться програмним забезпеченням і ідеями для проектів. Це робить Raspberry Pi величезно гнучким інструментом для будь-якого проекту розумного будинку.

ESP8266 - це високоінтегрований мікроконтролер з Wi-Fi, розроблений компанією Espressif Systems. Його основна ціль - забезпечити Wi-Fi з'єднання для різних IoT (Internet of Things, Інтернет речей) пристроїв, а також для мікроконтролерів і різноманітних електронних проектів [8].

ESP8266 має вбудований 32-бітний мікроконтролер, який можна програмувати за допомогою різних інструментів, включаючи Arduino IDE та NodeMCU firmware, що базується на Lua. Це робить ESP8266 дуже гнучким інструментом для розробників. З одного боку, ESP8266 може служити в якості Wi-Fi модуля, який додає з'єднання з Інтернетом до пристроїв, які не мають вбудованої Wi-Fi підтримки. З іншого боку, воно може служити в якості повноцінного мікроконтролера з Wi-Fi з'єднанням для створення різних IoT пристроїв (рис. 1.6).

Рисунок 1.6 – Мікроконтролер ESP8266

ESP8266 використовується в широкому спектрі застосувань, включаючи розумні домашні пристрої, автоматизовані системи поливу, моніторинг погоди, моніторинг середовища і багато іншого.

Основним чином, ESP8266 є надзвичайно корисним для будь-якого проекту, що вимагає з'єднання з Інтернетом або Wi-Fi мережею. Додатково, він є доступним за ціною.

1.3. Огляд різних типів датчиків

Розумні будинки використовують широкий спектр датчиків для автоматизації і контролю різних систем у будинку. Датчики можуть вимірювати все, від температури і вологості до руху та освітлення. Датчики можуть бути підключені до централізованої системи управління розумним будинком, яка дозволяє автоматично адаптувати системи будинку до змінних умов, а також дозволяє користувачам контролювати та моніторити їх через додаток на смартфоні або інший пристрій.

Датчики температури - це важливі компоненти розумного будинку, які вимірюють температуру в різних частинах приміщення. Вони можуть використовуватися для контролю систем опалення, вентиляції та кондиціонування (HVAC), для забезпечення комфортного температурного режиму, а також для моніторингу температури для забезпечення безпеки [9].

Види датчиків температури:

Терморезистори: Використовують зміну опору металевих елементів для вимірювання температури. Вони дуже точні і надійні, але можуть бути дорожчими за інші типи датчиків.

Термістори: Термістори схожі на RTD, але використовують полупровідникові матеріали замість металів. Вони менш стабільні і менш точні, але зазвичай дешевші і менші за розміром.

Термопари: Термопари використовують термоелектричний ефект для вимірювання температури. Вони дуже надійні, можуть вимірювати великі діапазони температур, але є менш точними за RTD.

Напівпровідникові датчики температури: Ці датчики використовують зміну напруги в напівпровідникових матеріалах для вимірювання температури. Вони компактні, енергоефективні і дешеві, але можуть бути менш надійними за інші типи датчиків.

Інфрачервоні датчики температури: Ці датчики вимірюють інфрачервоне випромінювання, що випускається об'єктами, для вимірювання їх температури. Вони можуть вимірювати температуру без контакту, що робить їх корисними для вимірювання температури важкодоступних об'єктів або поверхонь.

Кожен тип датчика має свої переваги та недоліки, тому вибір датчика температури залежить від конкретних вимог до системи розумного будинку.

Датчик вологості - це пристрій, який використовується для виявлення рівня вологості в середовищі. Вони широко використовуються в різних галузях, включаючи сільське господарство, прогнозування погоди, системи вентиляції та кондиціонування, а також в різних наукових та промислових застосуваннях [10].

Ось декілька видів датчиків вологості:

Капацитивні датчики вологості: ці датчики використовують дві металеві пластини з діелектриком між ними. Коли вологість змінюється, змінюється діелектрична стала, що в свою чергу змінює ємність. Зміну ємності можна виміряти і перетворити на вимір вологості.

Резистивні датчики вологості: вони використовують матеріал, який змінює свій опір в залежності від вологості. Типові матеріали включають сіль або провідний полімер. Зміну опору можна виміряти і перетворити на вимір вологості.

Термічні датчики вологості: вони використовують властивість води поглинати тепло, щоб визначити вологість. При зміні вологості змінюється температура поверхні датчика, що можна виміряти.

Датчики вологості на основі зміни маси: ці датчики використовують тонкі пластини, які змінюють свою частоту коливань в залежності від ваги води на них. Ці зміни можна виміряти і перетворити на вимір вологості.

Оптичні датчики вологості: вони використовують властивість води змінювати індекс заломлення світла. Використовуючи світлодіод та датчик світла, можна виміряти зміни в світлі, яке проходить через зразок повітря або прямо відбивається від нього.

Датчики руху - це пристрої, які використовуються для виявлення фізичного руху (зміни положення) в оточуючому середовищі. Вони використовуються в різних сферах, включаючи системи безпеки, автоматичне освітлення, системи слідкування за активністю та інше [11].

Ось декілька видів датчиків руху:

Піроелектричні або ІЧ-датчики (інфрачервоні): ці датчики руху використовують інфрачервоне випромінювання для виявлення руху. Вони визначають зміни в тепловому випромінюванні в зоні охорони. Люди, тварини та інші об'єкти випромінюють теплове випромінювання, тому при русі в області детекції датчика він реєструє ці зміни.

Ультразвукові датчики: ці датчики використовують ультразвукові хвилі для виявлення руху. Вони випромінюють ультразвукові хвилі, а потім слухають їх ехо. Якщо об'єкт переміщується, ехо змінюється, і датчик це визначає.

Мікрохвильові датчики: ці датчики використовують мікрохвильове випромінювання для виявлення руху. Схоже на ультразвукові датчики, вони випромінюють хвилі та слухають їх ехо. Мікрохвильові датчики можуть працювати на більших відстанях і проникнути через деякі матеріали, що є їх перевагою.

Датчики вібрації: ці датчики виявляють фізичну вібрацію об'єкта або поверхні. Вони можуть використовуватися для виявлення руху на великих відстанях, наприклад, вібрацій в структурі будівлі, коли хтось ходить по ній.

Кожен з цих видів датчиків має свої особливості, а вибір конкретного типу залежить від конкретного застосування та вимог до системи.

Висновки до розділу 1

У цьому розділі були розглянуті та проаналізовані різні типи датчиків, мікроконтролерів та систем "розумний будинок". Ми розглянули різноманітність датчиків, включаючи температурні датчики, датчики освітлення, датчики руху та інші, які використовуються для створення комфорту та безпеки в розумному будинку.

Ми також розглянули різні мікроконтролери, які є центром систем розумного будинку, контролюючи різні датчики та пристрої, що мають надзвичайно важливе значення для функціонування таких систем.

Огляд систем "розумний будинок" демонструє широкий спектр можливостей, які вони можуть надати, включаючи автоматичне керування освітленням, температурою, безпекою та багатьма іншими аспектами домашнього життя.

Кожна з розглянутих систем має свої переваги та недоліки, але їх спільною метою є підвищення комфорту, ефективності енергоспоживання та безпеки домівок.

Отже, огляд цих різних технологій і компонентів є важливим для розуміння потенціалу розумного будинку та вибору найкращих рішень для конкретних потреб користувачів. Розвиток технологій продовжує відкривати нові можливості для розумних будинків, і це огляд є важливим кроком для зрозуміння та використання цих можливостей.

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 виконання дипломної роботи буде братися плата Arduino UNO. Arduino Uno — це широко використовувана плата мікроконтролерів з відкритим кодом на базі мікроконтролера ATmega328P. У його склад входить все необхідне для зручної роботи з мікроконтролером: 14 цифрових входів/виходів (з них 6 можуть використовуватися в якості ШІМ-виходів), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для програмування всередині схеми (ICSP) і кнопка скидання. Для початку роботи з пристроєм досить просто подати живлення від AC/DC-адаптера або батарейки, або підключити його до комп'ютера за допомогою USB-кабелю (рис. 2.1).

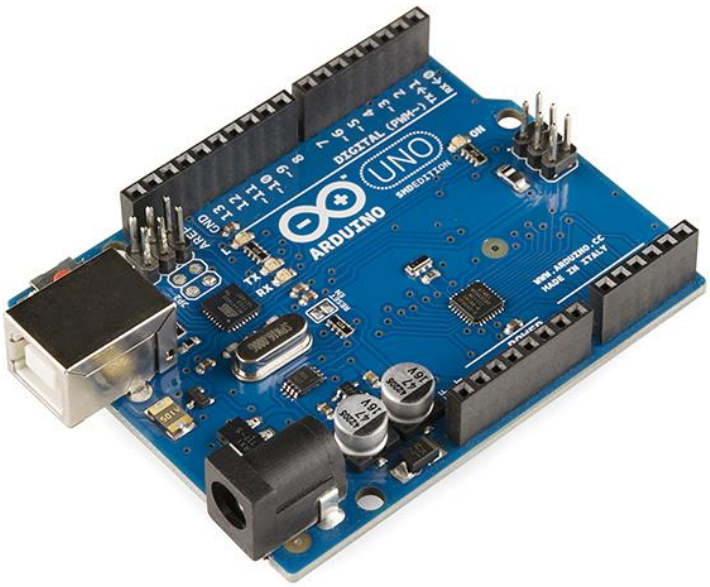

Рисунок 2.1 – Плата Arduino UNO

Arduino Uno може живитися від USB або від зовнішнього джерела живлення — тип джерела вибирається автоматично. В якості зовнішнього джерела живлення (НЕ USB) може використовуватися мережевий AC/DC-адаптер або акумулятор/батарея. Штекер адаптера (діаметр — 2.1 мм, центральний контакт — позитивний) необхідно вставити у відповідний роз'єм живлення на платі. У разі живлення від акумулятора/батареї, її дроти необхідно під'єднати до виводів Gnd і Vin роз'єму POWER. Напруга зовнішнього джерела живлення може бути в межах від 6 до 20 В. Однак, зменшення напруги живлення нижче 7 В призводить до зменшення напруги на виході 5V, що може стати причиною нестабільної роботи пристрою. Використання напруги більше 12 В може призводити до перегріву стабілізатора напруги і виходу плати з ладу. З огляду на це, рекомендується використовувати джерело живлення з напругою в діапазоні від 7 до 12 В.

З використанням функцій pinMode (), digitalWrite () і digitalRead () кожен з 14 цифрових виводів може працювати в якості входу або виходу. Рівень напруги на виводах обмежений 5 В. Максимальний струм, який може віддавати або споживати один вивід, становить 40 мА. Всі виводи пов'язані з внутрішніми підтягуючими резисторами (за умовчанням відключеними) номіналом 20-50 кОм (рис. 2.2).

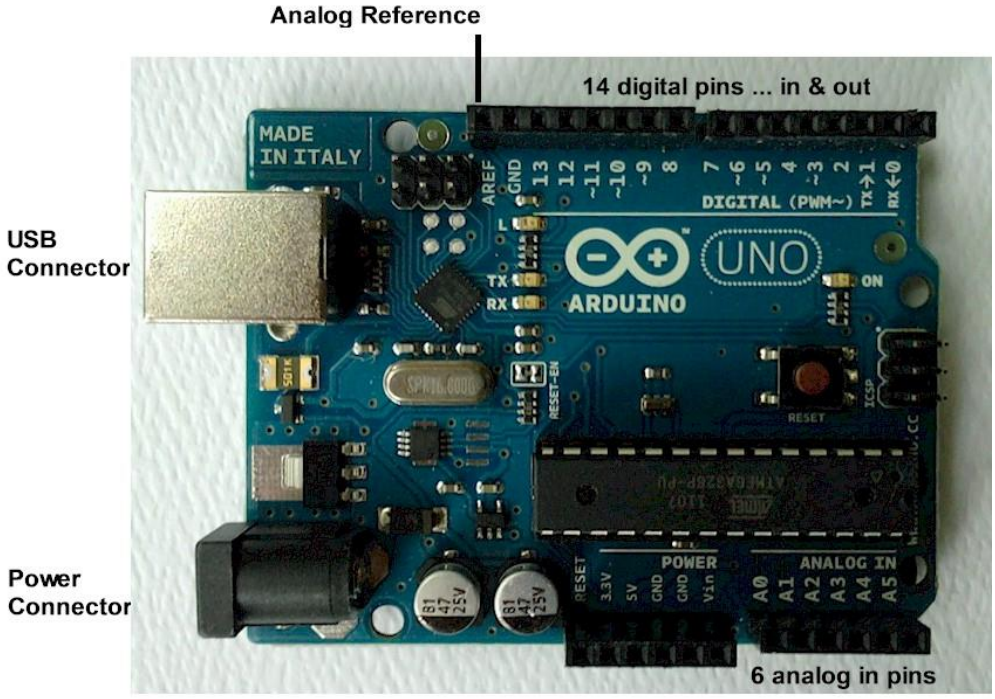

Рисунок 2.2 – Виводи Arduino UNO

Обсяг флеш-пам'яті ATmega328 становить 32 КБ (з яких 0.5 КБ використовуються завантажувачем). Мікроконтролер також має 2 КБ пам'яті SRAM і 1 КБ EEPROM (з якої можна зчитувати або записувати інформацію за допомогою бібліотеки EEPROM).

Незважаючи на те, що більшість комп'ютерів мають власний захист, такі запобіжники забезпечують додатковий рівень захисту. Якщо від USB-порту споживається струм більше 500 мА, запобіжник автоматично розірве з'єднання до усунення причин короткого замикання або перевантаження.

2.1.2. Опис послідовного порту

Послідовний порт, відомий також як COM порт або RS-232 порт, це тип інтерфейсу, який використовується для обміну даними між пристроями. Він передає дані по одному біту за раз, за допомогою послідовного протоколу (рис. 2.3).

Рисунок 2.3 – COM порт або RS-232 порт

Ключові особливості послідовного порту:

1. **Передача даних:** Послідовні порти передають дані "послідовно", що означає, що біти даних відправляються один за одним по одній лінії передачі даних.
2. **Дуплексна комунікація:** Більшість послідовних портів підтримує дуплексну комунікацію, що означає здатність одночасно передавати та отримувати дані.
3. **Швидкість передачі даних:** Швидкість передачі даних (бодрейт) в послідовному порті вимірюється в бодах (бітах на секунду).
4. **Інтерфейс:** Послідовний порт зазвичай використовує RS-232 інтерфейс для фізичного з'єднання, але також може використовувати інші стандарти, такі як RS-422 або RS-485.
5. **Використання:** Послідовні порти використовуються в різних сферах, включаючи телекомунікації, промислову автоматизацію, і, раніше, для з'єднання периферійних пристроїв, таких як миші та модеми, до комп'ютерів.

Останнім часом послідовний порт став менш популярним через появу USB, який надає більшу швидкість передачі даних та простоту використання. Проте, він все ще широко використовується в промислових і наукових застосуваннях.

Arduino використовує послідовний порт для взаємодії з комп'ютером або іншими пристроями. Це забезпечує засіб для передачі даних між пристроями біт за бітом. Передача даних Arduino може передавати дані через послідовний порт використовуючи функції Serial.print() або Serial.write(). Читання даних Arduino може читати дані, що надходять через послідовний порт, використовуючи функцію Serial.read(). Встановлення швидкості передачі даних перед використанням послідовного порту, Arduino повинен встановити швидкість передачі даних через функцію Serial.begin(). Підключення до комп'ютера Arduino підключається до комп'ютера через USB, який також слугує як віртуальний послідовний порт. Налагодження послідовний порт також використовується для налагодження програмного забезпечення Arduino. Ви можете виводити дані на комп'ютер через послідовний порт для перегляду стану програми Arduino в реальному часі.

Сумісність більшість Arduino плат, як-от Uno, Mega, та Leonardo, мають хоча б один апаратний послідовний порт. Деякі моделі, як-от Arduino Mega, мають кілька послідовних портів [12]

2.1.2. Опис PIR-датчика

PIR-датчик (Пасивний Інфрачервоний Датчик) - це електронний сенсор, який вимірює інфрачервоне світло, що випромінюється об'єктами у його полі зору [13]. Вони часто використовуються в системах безпеки і мають широке застосування в будівельній автоматиці та пристроях, що економлять енергію (рис. 2.4).

Рисунок 2.4 – PIR-датчик

Основні характеристики PIR-датчика:

1. **Виявлення руху:** При русі об'єкта (наприклад, людини) в полі зору датчика він виявляє зміну в інфрачервоному випромінюванні, що спричинена цим рухом. Ця інформація може використовуватися для активізації системи безпеки або автоматичного включення світла.

2. **Системи безпеки:** У системах безпеки PIR-датчики використовуються для виявлення небажаного проникнення. Коли датчик виявляє рух, він активізує тривогу, відправляє повідомлення на пульт охорони або викликає іншу передбачену реакцію системи.

3. **Автоматичне освітлення:** PIR-датчики також використовуються в системах освітлення, що включаються автоматично при виявленні руху. Це може бути корисно в коридорах, ванних кімнатах, гаражах або на вулиці для зручності та енергоефективності.

4. **Налаштовування:** Багато PIR-датчиків дозволяють користувачам налаштувати чутливість датчика і час затримки перед тим, як він поверне себе до стану очікування після виявлення руху. Це дозволяє більш точно налаштувати датчик для конкретного застосування.

5. **Низький споживання енергії:** PIR-датчики - це пасивні пристрої, що не випромінюють енергію, тому вони споживають відносно мало енергії, що робить їх ідеальними для застосувань, де важливо тривале живлення від батареї.

2.1.3. Опис DHT11

DHT11 - це цифровий температурно-вологісний датчик, який використовується для вимірювання температури та вологості навколишнього середовища (рис. 2.5).

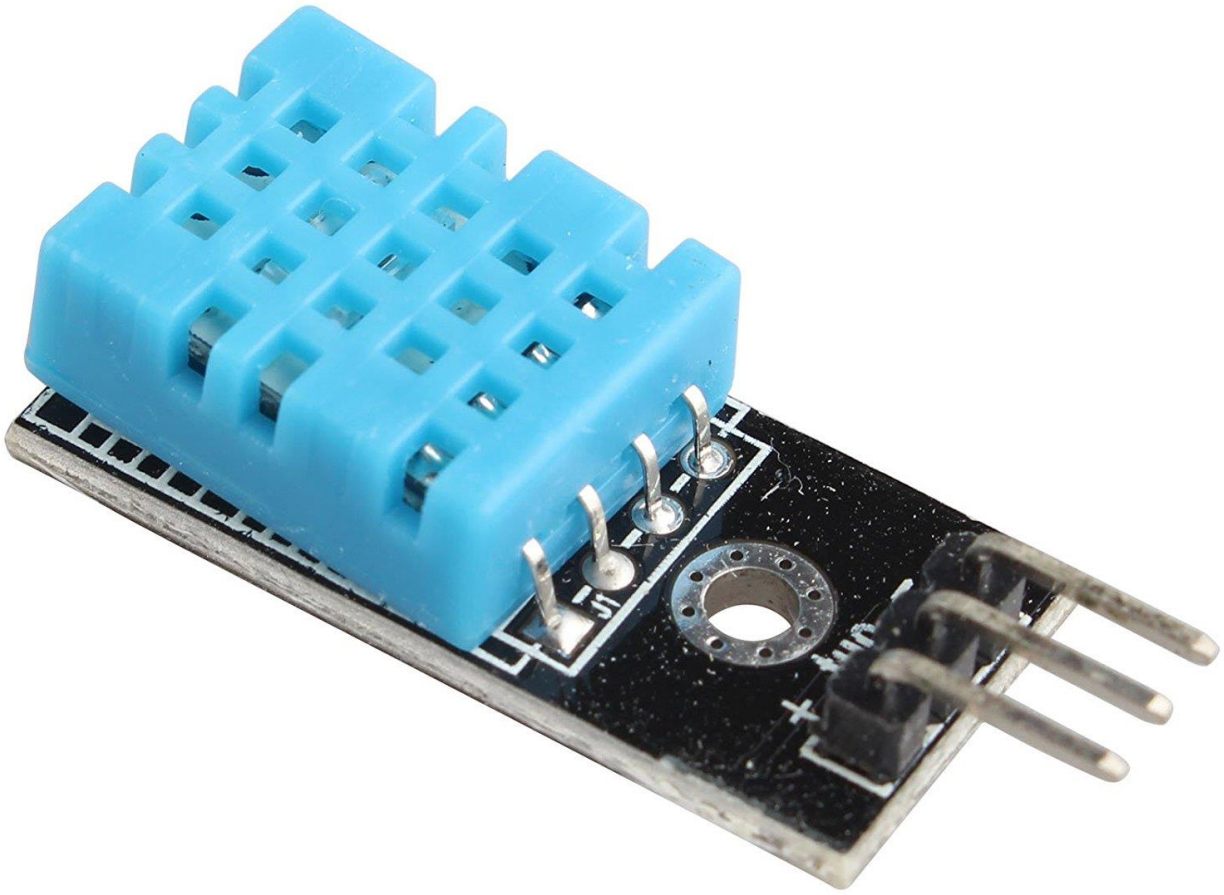

Рисунок 2.5 – DHT11 датчик

Основні характеристики DHT11:

1. **Вимірювання температури:** DHT11 може вимірювати температуру в діапазоні від 0 до 50 градусів Цельсія з точністю ±2 градуси.

2. **Вимірювання вологості:** DHT11 може вимірювати відносну вологість в діапазоні від 20% до 80% з точністю ±5%.

3. **Цифровий інтерфейс:** DHT11 використовує цифровий інтерфейс для передачі даних до мікроконтролера. Це означає, що ви можете підключити його до будь-якого цифрового порту на вашому мікроконтролері.

4. **Низький рівень споживання енергії:** DHT11 має дуже низьке споживання енергії, що робить його ідеальним для батарейних або енергоефективних проектів.

5. **Широке використання:** DHT11 широко використовується в різноманітних проектах, що базуються на мікроконтролерах, таких як Arduino, для моніторингу умов навколишнього середовища.

6. **Простота використання:** DHT11 дуже простий у використанні. Ви маєте лише підключити його до живлення, заземлення та цифрового входу вашого мікроконтролера, а потім використати відповідну бібліотеку для читання даних з датчика [14].

2.2 Програмування мікроконтролерів огляд середовища розробки

2.2.1. Огляд Arduino IDE

Arduino IDE (Integrated Development Environment) - це програмне забезпечення, яке використовується для написання та завантаження програмного коду до мікроконтролерів Arduino. Це надзвичайно популярна платформа, яка об'єднує в собі редактор коду, компілятор, інструменти для завантаження коду на плату [15].

Основні особливості Arduino IDE:

1. Простота використання: Arduino IDE створена з метою спростити процес розробки програмного забезпечення для вбудованих систем. Вона надає дружній до користувача інтерфейс, що допомагає новачкам швидко осягнути основи.

2. Мова програмування: Arduino IDE використовує спрощену версію C++, що робить код більш доступним для людей, які тільки починають вчити програмування.

3. Бібліотеки: Arduino IDE включає велику кількість вбудованих бібліотек, що дозволяють розробникам легко взаємодіяти з різноманітними сенсорами та актуаторами.

4. Підтримка багатьох плат: Arduino IDE підтримує різні плати Arduino, включаючи Arduino Uno, Mega, Nano та інші.

5. Серійний монітор: Arduino IDE має вбудований серійний монітор, який дозволяє розробникам відстежувати та відлагоджувати дані в реальному часі.

6. Open Source: Arduino IDE - відкрите програмне забезпечення, що означає, що ви можете змінювати його відповідно до ваших потреб і розділяти зі спільнотою.

Таким чином, Arduino IDE є потужним інструментом для розробки програмного забезпечення для проектів на базі Arduino, включаючи розумні будинки. Вона дозволяє вам швидко і ефективно писати код, використовуючи різноманітні бібліотеки та інструменти, які призначені спеціально для цієї платформи (рис. 2.6).

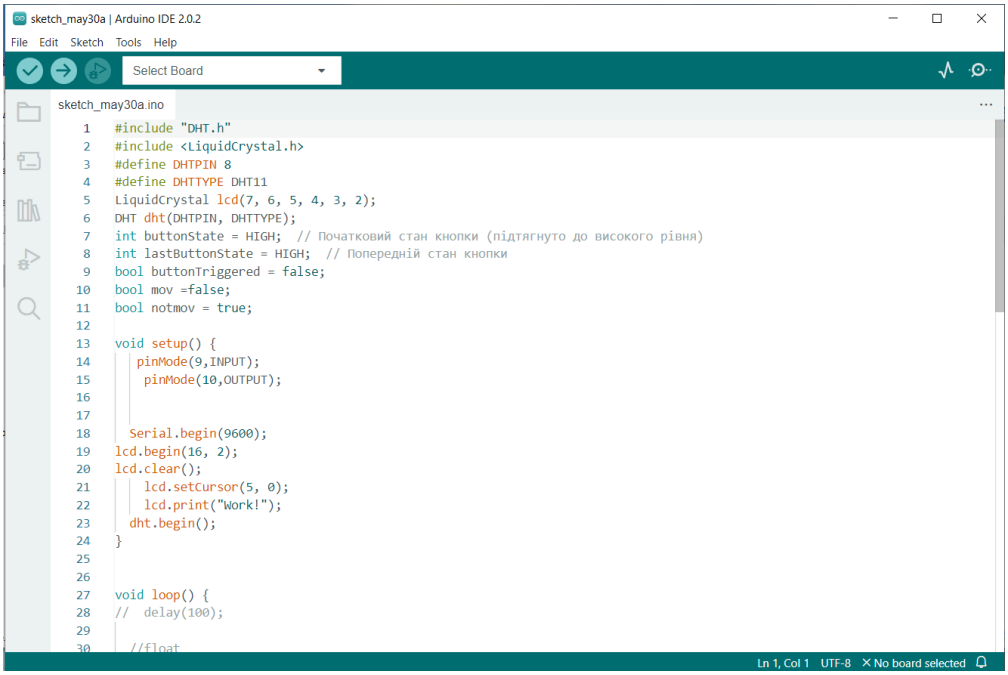

Рисунок 2.6 – Arduino IDE

2.2.2. Огляд Proteus 8 Professional

Proteus Design Suite — це запатентований набір програмних засобів, який використовується переважно для автоматизації електронного проектування. Пакет є системою моделювання, що базується на основі моделей електронних компонентів, прийнятих в PSpice . Відмінною рисою пакету PROTEUS VSM є можливість моделювання роботи програмованих пристроїв: мікроконтролерів , мікропроцесорів, DSP та ін.

В Proteus 8 Professional повністю реалізована концепція наскрізного проектування, коли інженер змінює щось у логіці роботи схемотехніки і програмний пакет тут же «підхоплює» дані зміни в системі трасування. Бібліотека компонентів містить довідкові дані (рис. 2.7).

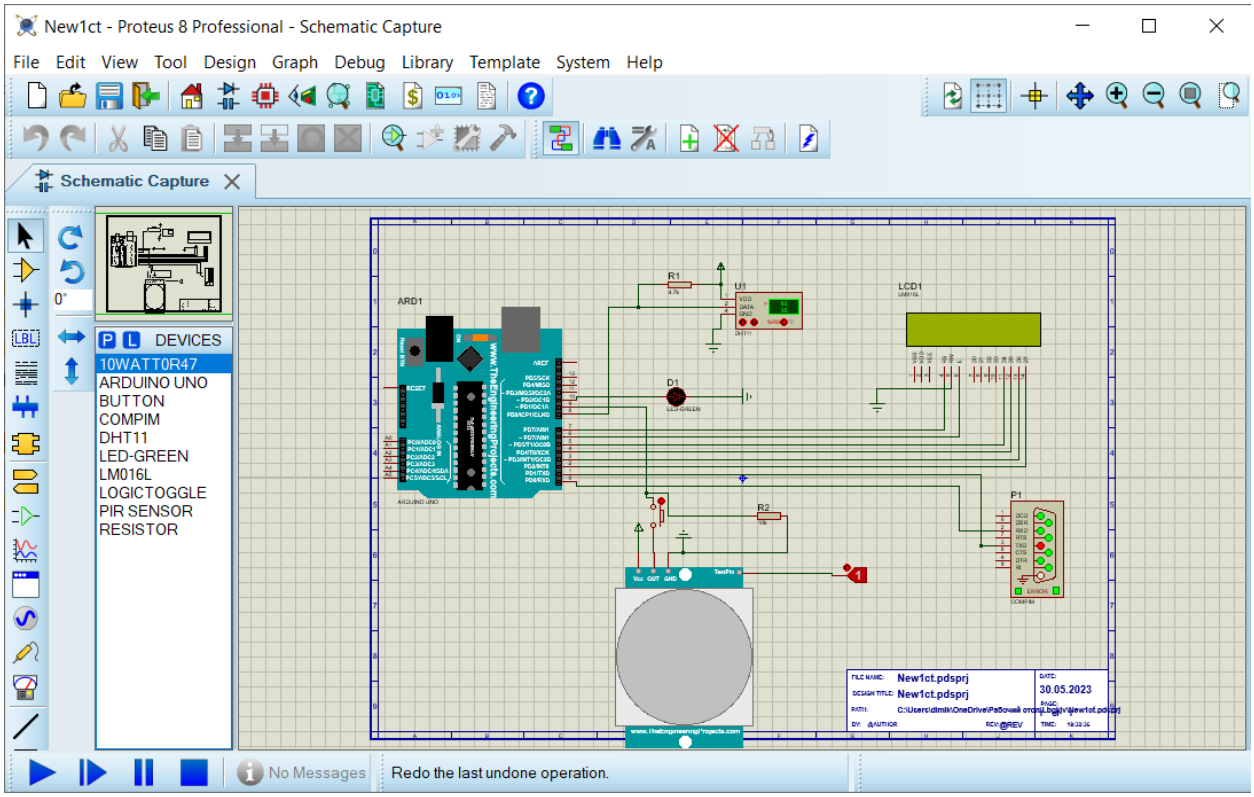

Рисунок 2.7 – Proteus 8 Professional

Додатково до пакету PROTEUS VSM входить система проектування друкованих плат . Пакет Proteus складається з двох частин, двох підпрограм: ISIS –

програма синтезута моделювання безпосередньо електронних схем та ARES — програма розробки друкованих плат.

Симуляція мікроконтролера в Proteus працює шляхом застосування шістнадцяткового файлу або файлу налагодження до частини мікроконтролера на схемі. Потім він моделюється разом із будь-якою аналоговою та цифровою електронікою, підключеною до нього. Це дозволяє використовувати його в широкому спектрі створення прототипів проектів у таких сферах, як керування двигуном, контроль температури та дизайн інтерфейсу користувача [16].

Він також знаходить застосування в загальній спільноті любителів і, оскільки не вимагає апаратного забезпечення, його зручно використовувати як навчальний або навчальний інструмент.

2.2.3. Огляд Visual Studio

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів.

Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms, а також вебсайти, вебзастосунки, вебслужби як у рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight [17].

Visual Studio включає редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторингу коду. Вбудований налагоджувач може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Інші вбудовані інструменти включають редактор форм для спрощення створення графічного інтерфейсу програми, веб-редактор, дизайнер класів і дизайнер схеми бази даних (рис. 2.8).

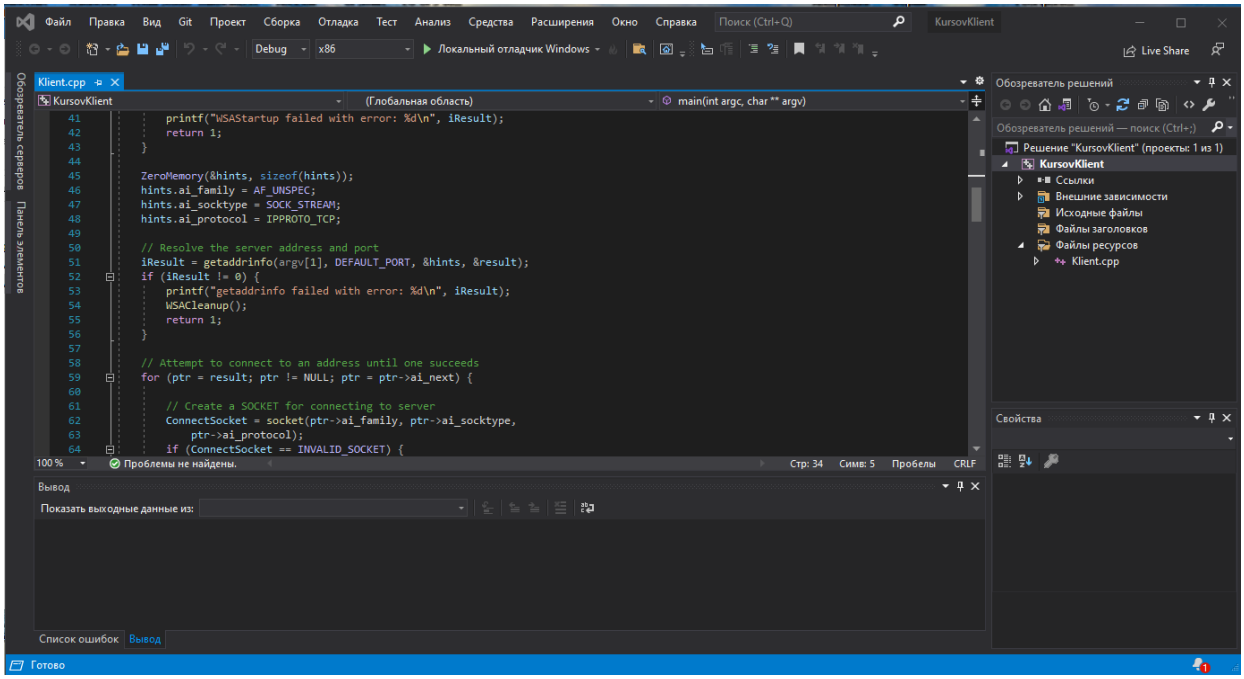

Рисунок 2.8 – Visual Studio

Засоби розробки мовами C++, C та асемблера, а також бібліотеки доступні у складі Visual Studio у Windows. Можна використовувати C++ в Visual Studio для створення будь-яких рішень, від простих додатків до класичних додатків для Windows, від драйверів пристроїв і компонентів операційних систем до кросплатформатних ігор для мобільних пристроїв, від систем для невеликих пристроїв Інтернету речей до багатосерверних обчислювальних платформ в хмарі Azure.

За допомогою Visual Studio та .NET можна розробляти класичні програми, веб-програми, мобільні програми, ігри та рішення для Інтернету речей. Програми .NET можна створювати мовою C#, F# або Visual Basic.[18]

2.2.3. Огляд Virtual Null Morem

Virtual Null Modem — це, по суті, кабель для з’єднання двох послідовних портів один з одним. Зазвичай для цього потрібні два комп’ютери або два фізичні порти на одному комп’ютері (рис. 2.9).

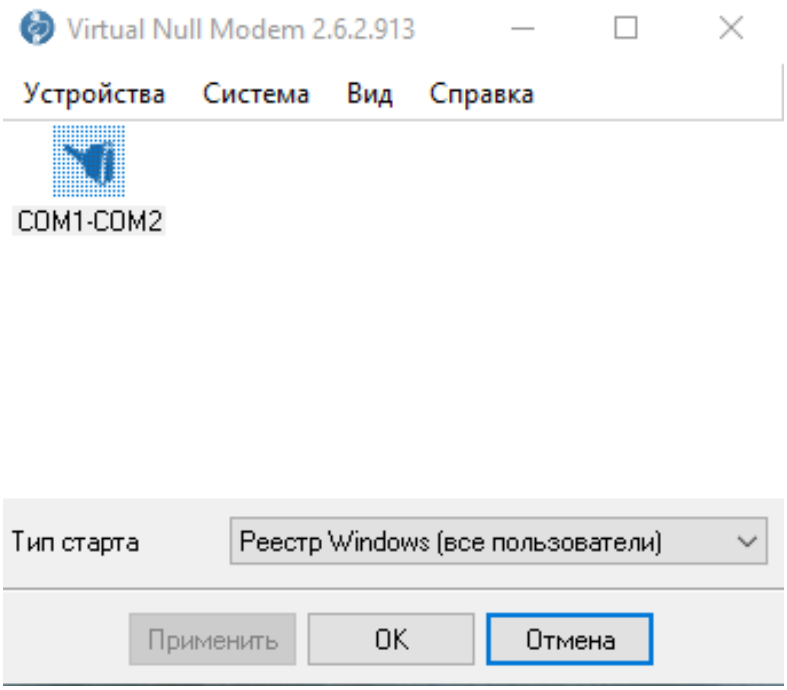

Рисунок 2.9 – Virtual Null Modem

Virtual Null Modem — це утиліта, метою якої є емуляція одного або кількох послідовних портів RS232, підключених через віртуальний нуль-модемний кабель. В іншому випадку ви можете створити будь-яку кількість чистих віртуальних послідовних портів у вашій системі (COM10, COM11, COM127 тощо), які будуть з’єднані один з одним через віртуальний нуль-модемний кабель.

Висновки до розділу 2

Основними компонентами системи, які були розглянуті в даному випадку, є Arduino UNO, послідовний порт, PIR-датчик та DHT11. Arduino UNO - це відкрита платформа для розробки електроніки, яка базується на простому мікроконтролері і доступному програмному забезпеченні. Послідовний порт використовується для зв'язку між пристроями, а PIR-датчик і DHT11 є датчиками руху та вологості/температури відповідно.

Програмування мікроконтролерів включає в себе використання різних середовищ розробки. Arduino IDE, Proteus 8 Professional, Visual Studio і Virtual Null Modem. Arduino IDE - це середовище розробки, яке прямо спрямоване на роботу з Arduino. Proteus 8 Professional - це потужний інструмент для моделювання і перевірки електронних схем. Visual Studio - це багатофункціональне середовище розробки, яке підтримує багато мов програмування. Врешті, Virtual Null Modem емулює роботу порту для встановлення з'єднання між двома програмами.

Узагальнюючи, розуміння роботи основних компонентів системи та володіння потрібними інструментами розробки є важливими аспектами при розробці мікроконтролерних систем. Різноманітні середовища розробки, які були розглянуті, надають потужні інструменти для ефективного програмування та тестування системи розумний будинок.

РОЗДІЛ 3. РОЗРОБКА "СИСТЕМИ РОЗУМНИЙ БУДИНОК" НА ОСНОВІ МІКРОКОНТРОЛЕРА ARDUINO

3.1 Моделювання "системи розумний будинок" на основі мікроконтролера arduino

Моделювання системи "розумний будинок" було здійснено в програмному середовищі Proteus 8 Professional, яке дозволяє створювати, перевіряти та відлагоджувати різноманітні електронні схеми та системи.

На даному етапі було створено модель "розумного будинку", яка включає в себе мікроконтролер Arduino Uno та два основних компонента - PIR-датчик та датчик DHT11.

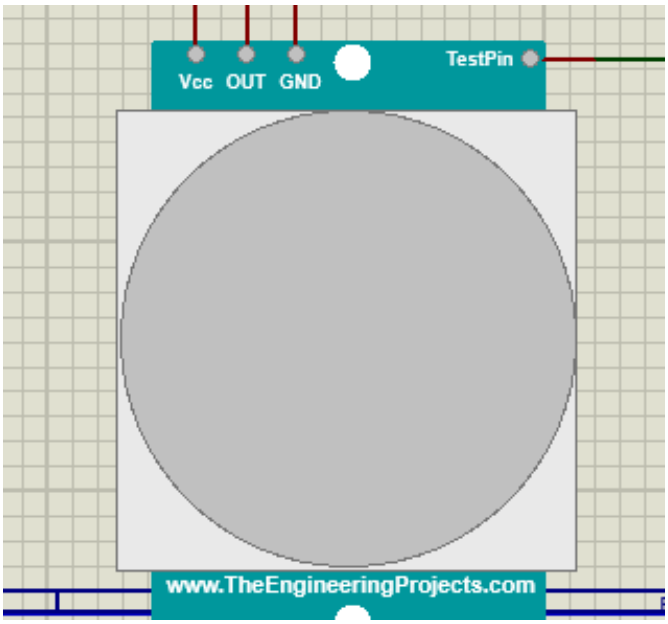

Рисунок 3.1 – PIR-датчик в Proteus 8 Professional

PIR-датчик використовується для забезпечення безпеки, а також для автоматичного включення світла при руху (рис. 3.1). Цей датчик відправляє сигнал на мікроконтролер Arduino, коли відчуває присутність або рух людини. DHT11 - це датчик вологості та температури, що відправляє дані про показники вологості та температури в приміщенні до мікроконтролера (рис. 3.2).

Мікроконтролер Arduino Uno відповідає за обробку інформації від датчиків та контроль роботи системи. Інформація з датчиків передається через цифрові входи мікроконтролера.

З підключенням до сервера, Arduino обробляє дані з датчиків та передає цю інформацію на сервер через мережеве з'єднання. Сервер, в свою чергу, передає ці дані клієнтському серверу, де вони можуть бути відображені для кінцевого користувача або використані для подальшого аналізу та впровадження в автоматичні процеси системи "розумний будинок".

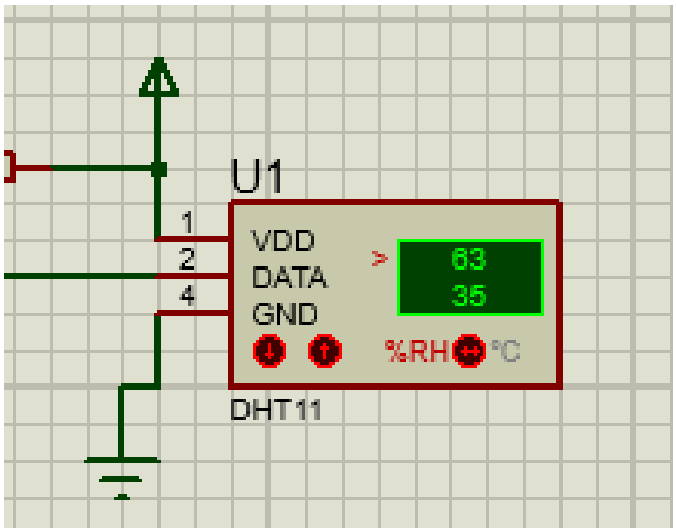

Рисунок 3.2 – DHT11 датчик в Proteus 8 Professional

3.2 Розробка програмного забезпечення для Arduino IDE

Схема системи "розумний будинок" була розроблена з урахуванням необхідності забезпечення функціональної безпеки, зручності користувача та максимальної ефективності роботи системи.

Усі деталі системи були детально перевірені та відлагоджені в ході моделювання, що дозволило забезпечити їх безперебійну роботу в реальних умовах експлуатації.

Розробка програмного забезпечення для проекту "розумний будинок" була здійснена з використанням Arduino IDE - відкритого середовища розробки, яке дозволяє створювати програми для мікроконтролера Arduino (рис. 3.3).

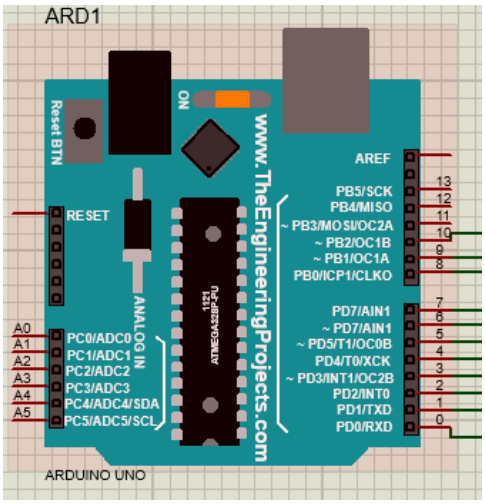

Рисунок 3.3 – Arduino UNO в Proteus 8 Professional

Для PIR-датчика було написано програмний код, який зчитує сигнали вхідного порту, до якого підключений датчик. При виявленні руху, система включає світло та надсилає сигнал на сервер.

Скрипт для DHT11 датчика було розроблено для зчитування температури та вологості в приміщенні. Дані передаються до Arduino, який, в свою чергу, надсилає їх на сервер. Всю відповідальність за комунікацію з сервером несе мікроконтролер Arduino. Було розроблено програмний код, що забезпечує передачу даних з мікроконтролера на сервер через мережеве з'єднання.

Також була виконана робота над кодом для опрацювання можливих помилок у процесі передачі даних та реагування на нестандартні ситуації.

Програмне забезпечення було протестовано та відлагоджено, щоб гарантувати надійність його роботи. Усі програми було написано з використанням мови програмування C++, що є стандартом для роботи з Arduino IDE (рис. 3.4).

Таким чином, в результаті розробки програмного забезпечення, було отримано готову до реального використання систему "розумний будинок", яка включає в себе різноманітні можливості для забезпечення комфорту та безпеки користувачів.

3.3 Опис програмного коду Arduino UNO

Код написано на мові програмування C++, який використовується для створення проектів на базі Arduino - відкритої платформи для вбудованих систем. Ось що робить кожен блок коду:

```
#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;
int lastButtonState = HIGH;
bool mov =false;
bool notmov = true;
```

Цей блок підключає потрібні бібліотеки (DHT.h для роботи з датчиком DHT11, LiquidCrystal.h для роботи з LCD дисплеєм) та визначає змінні та константи для подальшої роботи. DHT11 - це датчик вологості та температури.

```
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();}
```

Setup() виконується один раз при запуску програми. Воно налаштовує піни, серійний порт та дисплей.

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

В цьому блоку 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 зчитування датчика не вдається (тобто, якщо температура - NaN), цей блок виведе повідомлення про помилку на серійний порт і LCD дисплей.

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

Цей блок виводить зчитану температуру на LCD дисплей і серійний порт. Він виводить слово "Temp:", потім значення температури, і потім символ "C".

```
   lcd.setCursor(0, 2);
```

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

Аналогічно блоку 3, цей блок виводить зчитану вологість на LCD дисплей і серійний порт. Він виводить слово "Hum:", потім значення вологості, і потім символ "%".

```
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 читає стан PIR-датчика. Якщо стан високий (HIGH), він встановлює вихідний пін 10 в стан високий (HIGH). Крім того, якщо mov було встановлено в false, воно встановлює mov в true і виводить 'm' на серійний порт. Якщо стан вхідного піна 9 - не HIGH, воно встановлює вихідний пін 10 в стан низький (LOW) і якщо notmov було встановлено в true, воно встановлює notmov в false і виводить 'n' на серійний порт.

3.4. Опис коду клієнтської програми

Для того щоб забезпечити роботу датчика ліній на відстані використовується клієнт-серверна взаємодія, зокрема клієнтська частина. В даному підрозділі ми розглянемо розробку клієнтської частини. Ознайомимося з її роботою та функціоналом. Весь код буде знаходитися в додатку В.

Цей код пише TCP клієнт на C++, який використовує Winsock2 для встановлення з'єднання з сервером, передає йому деякі команди, а також може отримувати файли від сервера.

Основні Частини:

1. **Імпорт бібліотек:** На початку програма імпортує необхідні бібліотеки, що забезпечують здатність роботи з мережевими з'єднаннями та файлами.

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdlib.h>
#include <stdio.h>
#include <fstream>
#include <iostream>
#pragma comment (lib, "Ws2_32.lib")
#pragma comment (lib, "Mswsock.lib")
#pragma comment (lib, "AdvApi32.lib")
```

```
#define DEFAULT_BUFLEN 512
#define DEFAULT_PORT "27015"
```

2. **Ініціалізація Winsock:** Здійснює ініціалізацію бібліотеки Winsock, яка забезпечує роботу з мережевими функціями в Windows.

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

3. **Підготовка до підключення:** Перед встановленням з'єднання з сервером, програма спочатку визначає параметри з'єднання (адреса сервера, порт, тип сокета, протокол).

```
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;
```

```
}
```

4. **Встановлення з'єднання:** Програма встановлює з'єднання з сервером.

```
for (ptr = result; ptr != NULL; ptr = ptr->ai_next) {
   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;
   }
   iResult = connect(ConnectSocket, ptr->ai_addr, (int)ptr->ai_addrlen);
   if (iResult == SOCKET_ERROR) {
      closesocket(ConnectSocket);
      ConnectSocket = INVALID_SOCKET;
      continue;
   }
   break;
}
```

5. **Обробка команд:** Програма отримує команди від користувача та передає їх на сервер, також обробляє команду отримання файлу.

```
while (menu)
{
   std::cin >> key;
   while (key == 1)
   {
      if (key == 1) {
         std::cout << "File taking...\n";
```

```
            key = 0xC1;
        }}
    if (key == 4) {
        send(ConnectSocket, (char*)&key, sizeof(key), 0);
        printf("Connection closed\n");
        menu = false;
    }
}
```

6. **Закриття з'єднання і очистка ресурсів:** Програма завершує з'єднання та очищує всі ресурси, які використовувались під час роботи.

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

3.5 Опис коду серверної програми

Сервер — програма, що надає деякі послуги іншим програмам. Зв'язок між клієнтом і сервером зазвичай здійснюється за допомогою передачі повідомлень, часто через мережу, і використовує певний протокол для кодування запитів клієнта і відповідей сервера. Серверні програми можуть бути встановлені як на

серверному, так і на персональному комп'ютері, щоразу вони забезпечують виконання певних служб (наприклад, сервер баз даних чи вебсервер).

Після того як ми розібралися що таке сервер. Ми зрозуміли що він потрібен для того щоб зберігати наші дані. І для того щоб кожен користувач котрий має доступ міг їм користуватися.

Розробка серверної частини є ще більш важливішою, ніж розробка клієнтської частини, від це фундамент всього. Тому ознайомимося з її роботою та функціоналом. Весь код буде знаходитися в додатку Г.

Підключення бібліотек і визначення констант. Тут включаються бібліотеки для роботи з Windows API, Winsock2 для роботи з мережею, і т.д. Крім того, визначаються деякі константи для подальшого використання у коді.

```
#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"
```

Ініціалізація змінних і WSADATA здійснює ініціалізацію основних змінних і структур, таких як WSADATA (використовується для ініціалізації Winsock), SOCKET (слухає вхідні підключення) і addrinfo (для адреси і порту сервера).

```
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();
```

Створення сокета сервера і прослуховування вхідних підключень. Тут сервер створює сокет і прив'язує його до конкретної адреси та порту. Він також починає прослуховувати вхідні підключення.

```
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);
```

Обробка вхідних підключень і пересилання файлів. Обробляє вхідні підключення, приймає їх і пересилає файл клієнту. Це здійснюється в циклі, доки сервер не отримає конкретний сигнал від клієнта.

```
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

Розглянемо як працює наша система. Так як вона розроблена за допомогою віртуальної емуляції, то буде наведено поетапне підключення та взаємодія всіх використаних програм.

Для початку розробка проекту починається з проектування схем та підключення елементів. Тому починаємо з програми Proteus 8 Professional. В даній програмі було розроблено реалізацію підключення Arduino UNO, датчик DHT11, PIR-датчик та порту COMPIM. В плату Arduino UNO був завантажений скомпільований код програми. Який був розроблений у середовищі Arduino IDE. В коді було реалізовано підключення та передача даних з датчиків на плату (рис.3.4).

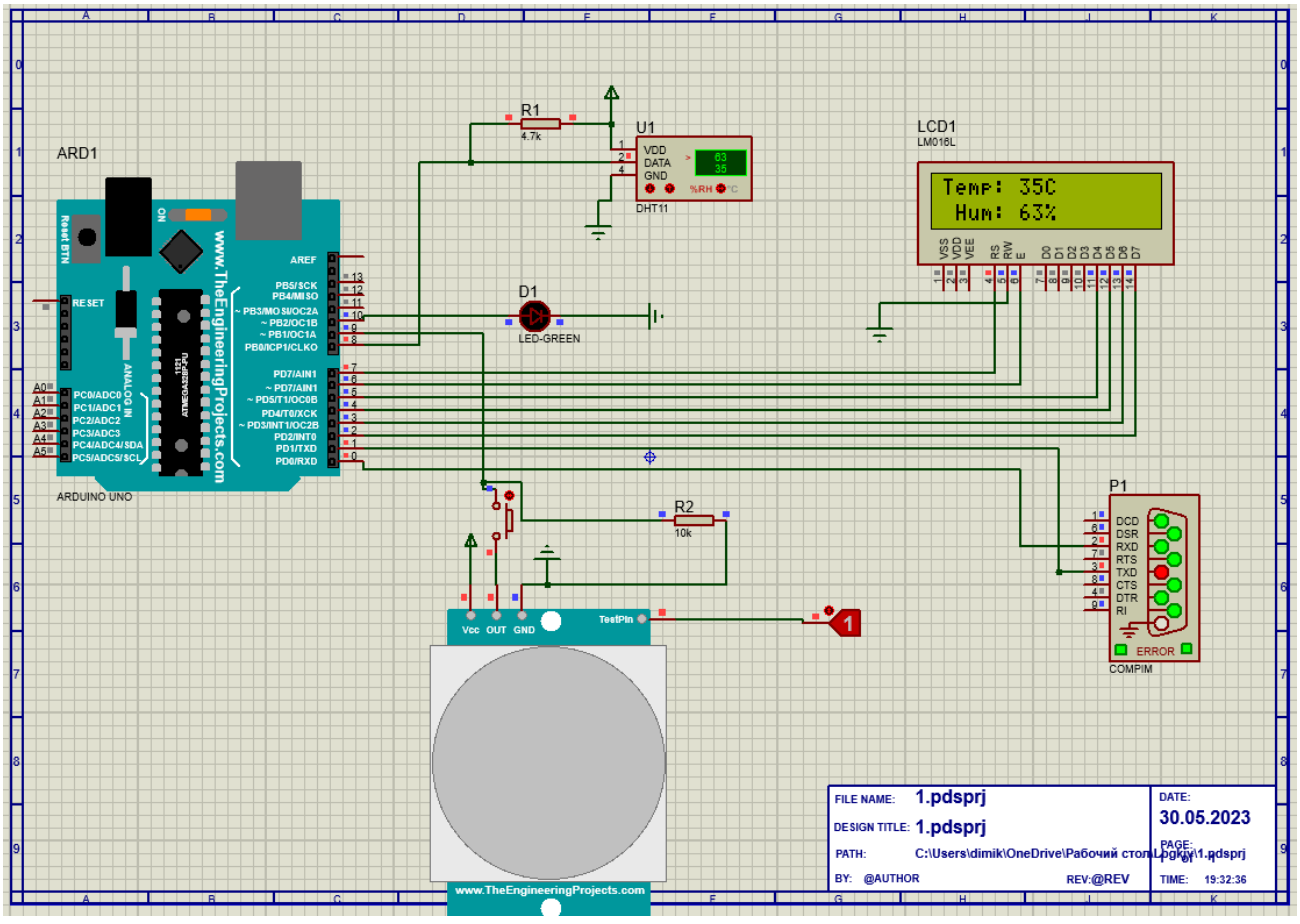

Рисунок 3.4 – Виконання симуляції Arduino UNO в Proteus 8 Professional

Після цього налаштуємо COM-порти. Як було написано вище. Virtual Null створює 2 віртуальні порти com2 та com3, які між собою з’єднані. Після цього підключаємо програму Virtual Null яка перехвачує дані між цими портами і записує у файл. У програмі Virtual Null вказано налаштування запису файлу та час прийому даних. Virtual Null перехвачує дані між портами і записує у файл. Програма не завжди правильно відображає інформацію, але запис файла робить завжди правильно (рис.3.5).

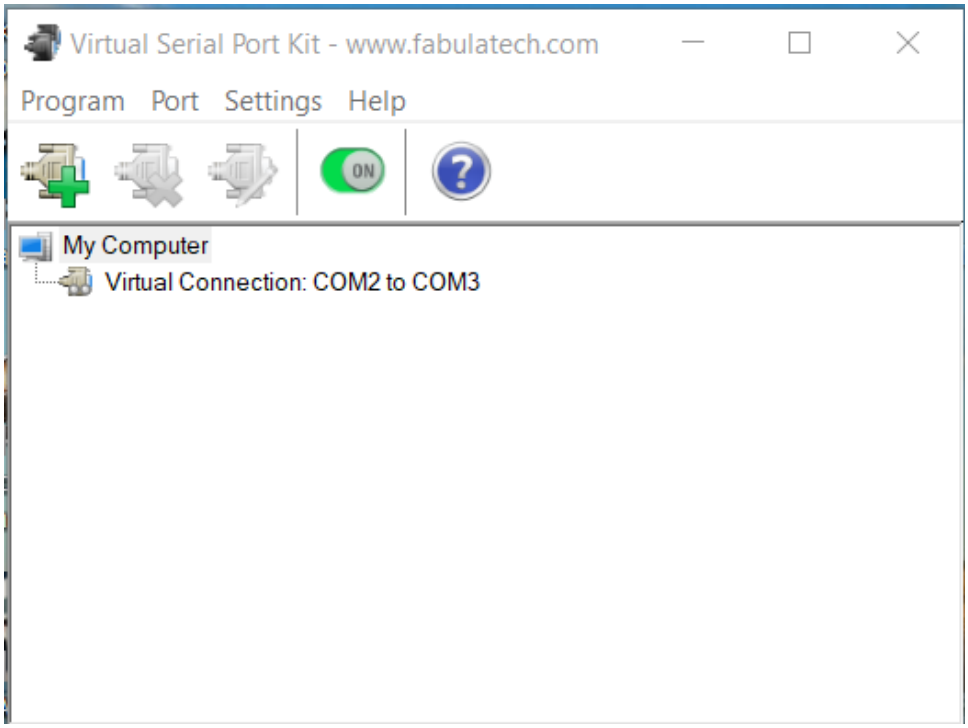

Рисунок 3.5 – Виконання програми Virtual Null

В програмі Proteus правильно налаштовуємо послідовний порт, як це показано на рисунку 3.4, а в програмі Virtual Null вказуємо порт, який ми перехвачуємо. Та який файл та куди зберігаємо.

Наступний етап це розробка клієнт-серверної частини за допомогою C++ та Visual Studio. Запускаємо код серверної частини (рис.3.6).

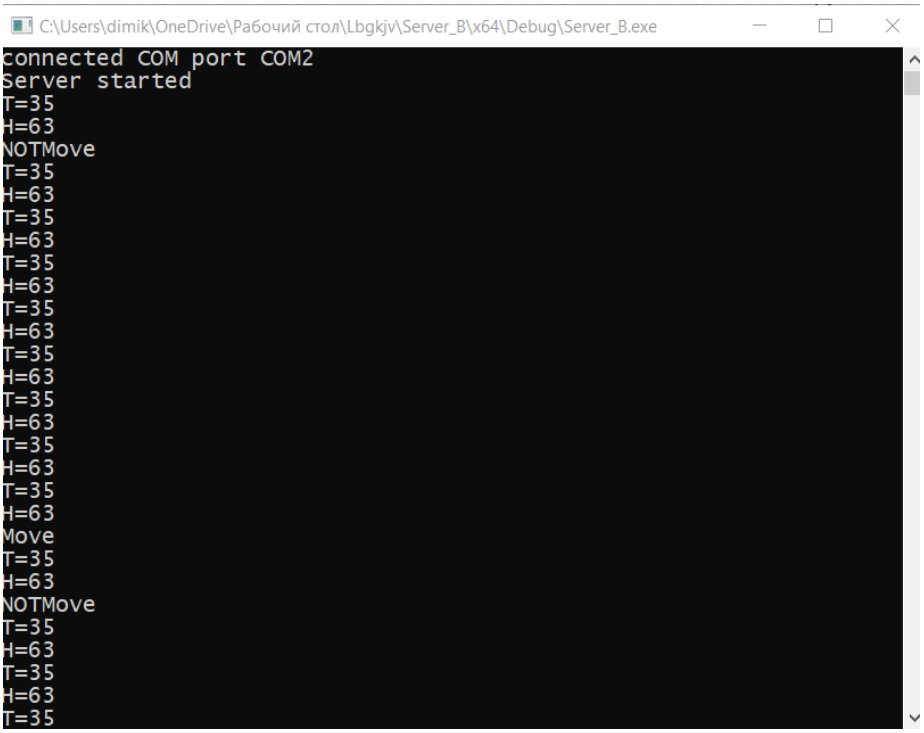

Рисунок 3.6 – Виконання програми серверної частини

Сервер постійно зчитую показники датчиків. Після цього потрібно запустити клієнтську частину. Вибираємо пункт отримати файл і файл відправляється до клієнта. Можна відключатися (рис.3.7).

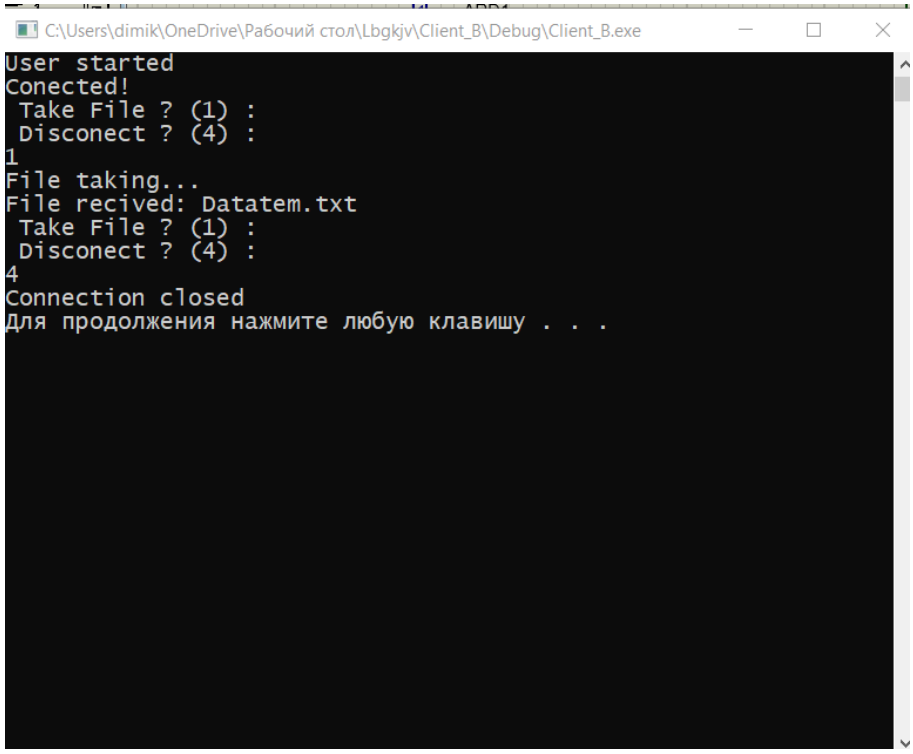

Рисунок 3.7 – Виконання програми клієнтської частини

Файл зберігає всі показники датчиків (рис.3.8).

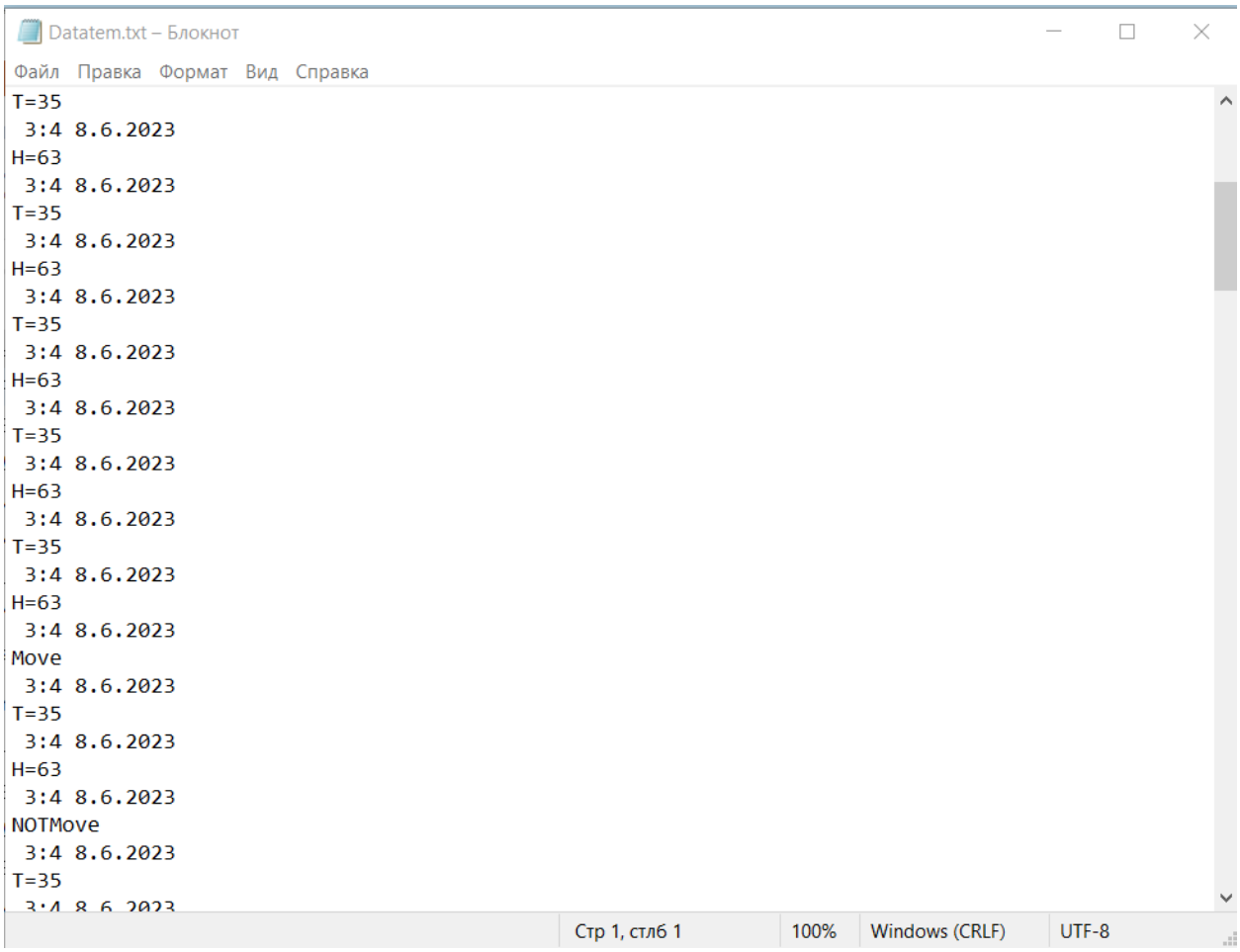

Рисунок 3.8 – Отриманий файл клієнтом

Висновки до розділу 3

В даному розділі була розроблена система "розумний будинок" на основі мікроконтролера Arduino. Початковий етап включав моделювання системи, де було враховано основні компоненти та їх взаємодію, що допомогло створити ефективний та цілісний план реалізації.

Для реалізації проекту було розроблено програмне забезпечення в Arduino IDE. Специфічний код Arduino UNO було створено для управління апаратними компонентами системи, враховуючи всі специфічні вимоги "розумного будинку". Додатково, було написано код для клієнтської та серверної програм, які забезпечують взаємодію між різними частинами системи і дозволяють користувачам керувати та моніторити свої домашні системи.

В результаті роботи системи "розумний будинок", користувачі мають можливість управляти основними функціями свого будинку через централізований інтерфейс, зокрема освітленням, температурою, безпекою та іншими параметрами.

У підсумку, розробка "розумного будинку" на основі мікроконтролера Arduino вимагає глибокого розуміння програмування, роботи з апаратним забезпеченням та принципів взаємодії між різними компонентами системи. Завдяки цьому, можна створити ефективну та інтуїтивну систему, яка значно покращує комфорт та безпеку життя користувачів.

ЗАГАЛЬНІ ВИСНОВКИ

Під час виконання дипломної роботи було створено "розумний будинок" на мікроконтролері Arduino. Даний розумний будинок може дистанційно показувати показники датчиків, таких як температура, вологість та рух в кімнаті.

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. Це забезпечує більш комфортне та ефективне використання домашнього простору.

Отже, під час виконання дипломної роботи було розроблено систему "розумний будинок" на мікроконтролері Arduino UNO. Результатом дослідження є змодельований розумний будинок, та клієнт-серверний сервер.

Розумний будинок може передавати дані з датчиків дистанційно що дає змогу слідити за температурою, вологістю та рухом в кімнаті. Моделювання цього розумного будинку дозволить всім бажаючим дізнатися та створювати власні розумні будинки, що є важливою частиною в сучасному та технологічному житті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розумний дім [Електронний ресурс] URL: https://stylus.ua/uk/articles/528.html (дата звернення: 04.04.2024)
2. Розумний будинок AJAX [Електронний ресурс] URL: https://ajax.systems/ua/ (дата звернення: 08.05.2023)
3. Розумний будинок Broadlink [Електронний ресурс] URL: https://broadlink.com.ua/ (дата звернення: 09.05.2023)
4. Розумний будинок Fibaro [Електронний ресурс] URL: https://www.fibaro.com/ru/ (дата звернення: 10.05.2022)
5. Мікроконтролер [Електронний ресурс] URL: https://elprivod.nmu.org.ua/ua/interesting/what_is_mp_mc_plc.php (дата звернення: 12.05.2023)
6. Arduino UNO [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/Arduino_Uno (дата звернення: 12.04.2023)
7. Raspberry Pi [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/Raspberry_Pi (дата звернення: 12.05.2023)
8. ESP8266 [Електронний ресурс] URL: https://alexgyver.ru/lessons/esp8266/ (дата звернення: 12.05.2023)
9. Датчик температури [Електронний ресурс] URL: https://ao-tera.com/ua/technology/types-of-temperature-sensors (дата звернення: 14.05.2023)
10. Датчики вологості [Електронний ресурс] URL: https://sitemasters.com.ua/elektroobladnannja/datchiki-vologosti-i-temperaturi-dlja/ (дата звернення: 16.05.2023)
11. Датчики руху [Електронний ресурс] URL: https://www.brille.ua/ua/datchiki-dvizheniya-vidy/ (дата звернення: 20.05.2023)
12. Послідовний порт [Електронний ресурс] URL: http://ni.biz.ua/3/3_5/3_55535_posledovatelniy-port.html (дата звернення: 21.05.2023)
13. PIR-датчик [Електронний ресурс] URL: https://www.mini-tech.com.ua/datchik-dvizheniya-infrakrasniy-pir-sensor-hc-sr501 (дата звернення: 22.05.2023)

14. Датчик вологості та температури DHT11 [Електронний ресурс] URL: https://arduino.ua/prod185-datchik-vlajnosti-i-temperatyri-dht11 (дата звернення: 22.05.2023)
15. Arduino IDE [Електронний ресурс] URL: https://www.arduino.cc/en/software (дата звернення: 24.05.2023)
16. Proteus 8 Professional [Електронний ресурс] URL: https://ru.taiwebs.com /windows/download-proteus-professional-2164.html (дата звернення: 24.04.2023)
17. Microsoft Visual Studio [Електронний ресурс] URL: https://visualstudio.microsoft.com/ru/vs/getting-started/ (дата звернення: 20.04.2023)
18. Visual Studio та .NET [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio (дата звернення: 21.04.2023)

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 = WSAStartup(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 fileRequestedsize = 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*)&fileRequestedsize, 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 < fileRequestedsize);
                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 = 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();

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;
}
```