Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему::

«Комп'ютеризована система управління будинком»

ПОЯСНЮВАЛЬНА ЗАПИСКА

08-23.БДР.003.00.000 ПЗ

Виконав студент 4 курсу, групи 1КІ–18б
спеціальності 123 — Комп'ютерна інженерія
Василевський Д.А.

Керівник роботи к.т.н., проф. каф. ОТ
Колесник І.С.
"13" 06 2022 р.

Опонент к.т.н., доц. каф. ЗІ
Куперштейн Л.М.
"14" 06 2022 р.

Допущено до захисту
д.т.н., проф. Азаров О.Д.
"15" червня 2022 р.

ВНТУ 2022

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень — бакалавр
Спеціальність — 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри обчислювальної техніки
проф., д.т.н. О.Д.Азаров
«08» 02 2022 р.

З А В Д А Н Н Я
НА БАКАЛАВСЬКУ ДИПЛОМНУ РОБОТУ

студенту □ **Василевському Денису Анатолійвичу**

1 Тема бакалаврської дипломної роботи «Комп'ютеризована система управління будинком», керівник роботи Колесник Ірина Сергіївна, к.т.н., доцент, затверджено наказом вищого навчального закладу від 24.03.2022 року № 66

2 Строк подання студентом роботи 13.06.2022

3 Вихідні дані до роботи: Технічні параметри систем управління, середовище розробки Arduino IDE, модуль Ethernet Enc28j60.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) вступ, огляд і аналіз сучасного стану галузі, проектування системи управління будинком, програмне забезпечення системи управління будинком, висновки перелік джерел посилання.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): технічне завдання, Код бібліотеки EtherCard, Код бібліотеки EtherEncLib, Код бібліотеки UIPEthernet

6 Дата видачі завдання 10.02.2022.

7 Календарний план виконання МКР приведений в таблиці 1.

Таблиця 1 — Календарний план

з/п	Назва етапів виконання бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задач роботи	14.02.22	ВІК
2	Огляд інформаційних джерел	15.02-28.02.22	ВІК
3	Огляд і аналіз сучасного стану галузі	01.03-14.03.22	ВІК
4	Проектування системи управління будинком	15.03-28.03.22	ВІК
5	Програмне забезпечення системи управління будинком	29.03-11.04.22	ВІК
6	Оформлення пояснювальної записки та ілюстративного матеріалу	12.04-25.04.22	ВІК
7	Аналіз виконання роботи. Висновки. Додатки	26.04-09.05.22	ВІК
8	Перевірка якості виконання бакалаврської роботи та усунення недоліків	17.05.22	ВІК

Студент Василевський Д.А.

Керівник роботи Колесник І.С.

АНОТАЦІЯ

Бакалаврська дипломна робота присвячена розробці комп’ютеризованої системи управління будинком. Для реалізації поставленої мети у якості програмної частини було використано середовище розробки Arduino IDE, а для реалізації апаратної керуючу плату-мікроконтролер Ardunio Leonrdo, різні датчики та 3 додаткові пристрої – модуль Ethernet Enc28j60, сервопривід та реле.

ANNOTATION

The bachelor's thesis is devoted to the development of a computerized home management system. The Arduino IDE development environment was used as a software part to achieve this goal, and the Ardunio Leonrdo microcontroller control board, various sensors and 3 additional devices - Ethernet Enc28j60 module, servo drive and relay - were used as software.

ЗМІСТ

					08-23.БДР.003.00.000 ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Василевський Д.			Комп'ютеризована система управління будинком Пояснювальна записка	Літ.	Арк. 6 / Аркушів 72
Перевір.		Колесник І.С					
Реценз.		Куперштейн Л.М.				ВНТУ, гр. 1КІ – 18б	
Н. Контр.		Швець С. І.					
Затверд.		Азаров О.Д.					

ВСТУП

У сучасному світі будь-який будинок або квартира складається з певного набору підсистем, які відповідають за виконання різних функцій. Залежно від типу будівлі, кількість та складність підсистем змінюється. У міру їх ускладнення контроль над ними стає все складнішим, витрати на обслуговування зростають. Щоб уникнути цього починають впроваджувати систему автоматизації будинку (Розумний будинок, Smart Home).

Донедавна автоматизований центральний контроль систем всієї будівлі застосовувався тільки в комерційних будівлях більшого розміру та дорогих будинках. Як правило, включаючи лише освітлення, опалення та системи кондиціонування, а управління було доступне лише у певних контрольних пунктах у межах самої будівлі. Сьогодні ситуація змінюється і системи автоматизації будинку виходять на загальнодоступний ринок. Система домашньої автоматизації дозволяє керувати багатьма інженерними вузлами та приладами з центрального пристрою як локально, так і віддалено. Як керуючий пристрій може бути комп'ютер, планшет або смартфон. Дана технологія може здаватися новою і певною мірою унікальною, але насправді система автоматизації використовує та поєднує вже існуючі технології.

Метою бакалаврської роботи є розробка системи автоматизації будинку, яка надасть користувачеві нові можливості з управління та контролю його житла.

Мета бакалаврської роботи досягається вирішенням кількох **завдань**:

— аналіз комерційних рішень на ринку домашньої автоматизації;

— вибір засобів та середовища розробки;

— створення макету пристрою;

— створення панелі керування системою автоматизації будинку.

Результатом бакалаврської роботи є програмно-апаратний пристрій, що дозволяє керувати електроприладами, контролювати пориви води, контролювати доступ до приміщення та відображати температуру всередині та зовні кімнати.

Об'єкт дослідження — комп’ютеризовані системи управління будинком.

Предмет дослідження — методи та засоби проектування комп’ютеризованих систем управління будинком.

1 ОГЛЯД І АНАЛІЗ СУЧАСНОГО СТАНУ ГАЛУЗІ

1.1 Система автоматизації будинку

Системи автоматизації будинку відомі також під іншими назвами — «Розумні будинки», «Smart home», інтелектуальні будинки та адаптивні будинки.

Саме поняття Smart home почало широко використовуватися в США. У 70-х роках минулого століття компанії Leviton та X10 розробили технологію управління побутовими електроприладами. Дана технологія працювала при частоті 60 Гц і напруга мережі 110 В, через це її недоцільно було застосовувати.

Принцип системи автоматизації будівлі передбачає спільне використання програмно-апаратних засобів, за рахунок яких значно зростає ефективність роботи та надійність управління всіх систем та пристроїв, які мають датчики або підключені безпосередньо до головного пристрою.

Важливою особливістю систем автоматизації є те, що людина може однією командою змінювати навколишню обстановку в будинку, а автоматика відповідно до зовнішніх і внутрішніх умов задає і відстежує режими роботи всіх контрольованих систем і електроприладів. Це дає можливість відмовитися від використання дистанційних пультів керування, множини вимикачів, окремих блоків керування кліматичними установками або відеоспостереження тощо.

1.2 Можливості інтелектуальних будинків

Концепція інтелектуального будинку надає безмежні можливості його власника, розглянемо основні їх.

Легкість керування

Одна з головних переваг системи автоматизації будинку — це зручність і легкість управління різним обладнанням та системами, об'єднаними в єдине ціле.

Для моніторингу за системою та управління нею, може використовуватися як центральний головний пристрій, який знаходиться безпосередньо в будинку,

або портативний пристрій з виходом в Інтернет, за допомогою якого можна віддалено контролювати роботу «Розумного будинку».

Панель контролю (локальна або дистанційна), що управляє системою «Розумний Дім», має широкі можливості управління, так як в одну систему автоматизації зв'язуються різні прилади, такі як системи освітлення, електрична мережа, кліматичні та обігрівальні установки, системи охорони, це дозволяє контролювати та змінювати умови комплексно. Ці установки визначаються користувачем, виходячи з можливих життєвих ситуацій, і можуть бути налаштовані в будь-який момент.

Для підвищення зручності використання системи автоматизації будинку можуть включати набір сценаріїв автоматизованої роботи з фіксованими налаштуваннями.

Ефективне використання ресурсів та енергії

Системи автоматизації можуть допомогти заощадити ресурси та гроші. Важлива міра енергозбереження – централізація управління освітленням з використанням спеціально розроблених графіків увімкнення та вимкнення світла. Велику економію коштів та ресурсів можна отримати за рахунок максимального використання природного світла усередині приміщення. Штори або жалюзі можуть бути обладнані сервоприводами, що дозволяє ефективно використовувати природне освітлення. Крім цього, великий ефект дає використання енергозберігаючих ламп, але навіть найекономічніша лампа, що працює в порожньому приміщенні, стає безглуздим споживачем електроенергії.

Найкращого енергозбереження можна досягти й іншим способом — використанням інфрачервоних датчиків та датчиків освітлення. Інфрачервоні датчики забезпечують автоматичне увімкнення та вимкнення, залежно від того знаходиться людина в кімнаті чи ні. Датчики освітлення вимірюють рівень освітленості в приміщенні та при досягненні певного значення, так само, як і інфрачервоні датчики включають або вимикають світло. В основі системи енергозбереження лежить контроль температури. За аналогією зі світлом, регулюється і температура приміщення.

Кліматичний контроль

Завдяки використанню систем інтелектуального будинку можна оптимально використовувати системи опалення, роблячи великий внесок у збереження електроенергії та водних ресурсів. Споживання енергії зберігається на мінімальному рівні, необхідному задля забезпечення найвищого комфорту. Інтелектуальне керування дозволяє кожній кімнаті мати власну температуру, незалежну від зовнішніх умов. Система також може контролювати оптимальну вологість повітря та свіжість повітря. Можна в будь-який час побачити або змінити поточну температуру у вибраному приміщенні. Якщо конкретна частина будинку або кімната не заселена, то немає жодної необхідності для підтримання комфортної температури, і якщо потрібно, то можна віддалено вимкнути опалювальні системи. Контроль температури може працювати спільно з датчиками рухами та камерами,

Функціональна сумісність

Донедавна системи автоматизації будинку характеризувалися різнорідністю з багатьма пристроями, які повинні взаємодіяти та виконувати ефективно завдання. Причиною цього є природа «Розумних будинків». Це розподілена архітектура, якій потрібна певна міра сумісності та інтероперабельності для управління різнорідними системами, що включають різні платформи. Дані системи розроблялися в ізоляції та складалися з несумісного програмного забезпечення та різноманітної апаратної платформи.

Особливістю сучасних систем автоматизації є здатність пов'язувати різні електронні пристрої, щоб вони могли працювати як єдина система. Організація спільної роботи цих пристроїв може бути простою або складною, все залежить від «відкритості» системи автоматизації. Чим відкритіша система, тим легше буде для електронного обладнання та датчиків «спілкуватися» один з одним. Для підтримки сумісності між декількома електронними пристроями, виробники систем домашньої автоматизації та побутової техніки часто утворюють партнерські зв'язки, або впроваджують універсальні протоколи передачі даних

Wi-fi або Bluetooth, а також надають відкритий доступ до керуючих елементів пристрою.

Управління освітленням

Система автоматизації будинку дозволяє налаштовувати та керувати колірною гамою на розсуд користувача. Інфрачервоні датчики забезпечують автоматичне перемикання світла. Для забезпечення комфорту в будинку, кожна кімната має бути добре освітлена. Без системи «Розумний Дім» для цього знадобиться встановлення великої кількості різних вимикачів, контролерів та реле. Система автоматизації позбавить цієї необхідності, т.к. управління світловими приладами може відбуватися з панелі керування, смартфона або напівавтоматичному режимі, завдяки інфрачервоним датчикам або датчикам звуку. Крім основних завдань для освітлення є можливість регулювати рівень яскравості освітлювальних приладів. Система в автоматичному режимі може вирішити, яке освітлення необхідно встановити залежно від погоди на вулиці або обстановки в будинку. Це досягається, використовуючи встановлені сценарії. "Розумний Дім" може імітувати ефект присутності людей, що дає додатковий захист, коли вдома нікого немає.

Безпека

Система автоматизації будинку — це єдина система управління та контролю комфортом та безпекою будинку та його мешканців. Вона контролює не тільки цілісність інженерних систем, а й збереже будинок від візиту непроханих гостей. Системи безпеки можуть виконувати такі функції:

— запобігання надзвичайним ситуаціям, що загрожують матеріальному майну та здоров'ю людини: витоку води та газу, загоряння, пробої в електропроводці тощо;

— контроль цілісності периметра;

— імітацію присутності;

— контроль доступу до приміщення;

— відеоспостереження за будинком та прилеглою територією, отримання картинки з будь-якої камери відеоспостереження через Інтернет або панель контролю;

— можливість виклику позавідомчої охорони;

— отримання зображення з будь-якої камери відеоспостереження через Інтернет.

Причиною серйозних матеріальних втрат можуть стати спалахи, несправності в системах водопостачання та неналежна охорона. Система автоматизації будинку призначені для забезпечення безпеки та запобігання надзвичайним ситуаціям.

Дротові і бездротові датчики контролюють стан зон, що охороняються. Як такі датчики можуть виступати:

— магнітоконтактні датчики (геркони) — використовується для контролю проникнення у приміщення через двері чи вікна;

— датчики руху — пристрої, основне завдання яких полягає у своєчасній фіксації факту руху для автоматичного виконання будь-якої дії;

— датчики задимленості, вогню, температури — призначені для виявлення загорянь, що супроводжуються появою диму, світлових спалахів або підвищеної температури у приміщеннях;

— датчик рівня та наявності витоків води — з їх допомогою вимірюють рівень та наявність води. Вони можуть працювати на механічному,гідростатичному, електричному, магнітному чи оптичному принципі.

— Датчик витоку газу призначений для попередження про наявність у повітрі побутового газу, а також деяких інших горючих газів.

Крім вищезгаданих датчиків, можуть використовуватися й інші, залежно від потреб замовника та специфіки будинку чи квартири.

Залежно від типу сигналу датчики викликають відповідну реакцію керуючої системи. Наприклад, якщо датчик зафіксував загрозу виникнення пожежі, система

сповістить всіх присутніх у приміщенні і передасть інформацію на пожежну станцію. Також будуть включені системи пожежі гасіння, звичайно якщо ними обладнано будинок, припинено доступ свіжого повітря, що сприяє утворенню полум'я, перекрито газ, за необхідності відключено електрику. У разі несанкціонованого доступу до приміщення система передає сигнал на пульт позавідомчої охорони та повідомлення господарю, включає звукову та світлову сигналізацію.

Протікання води здатні завдати матеріальної шкоди не лише господарям, а й сусідам. Для запобігання надзвичайним ситуаціям, пов'язаним з витоками води, використовуються системи контролю витоків, які включають датчики рівня та наявності води. Дані датчики встановлюються у місцях з'єднання водопровідних труб, сантехніки, побутових приладів. Щоб автоматизована система могла регулювати подачу води, між ручними вентилями встановлюються магнітні клапани або сервоприводи. У разі виявлення вологи та певного рівня води на підлозі, датчики подають інформацію головному пристрою, який у свою чергу перекриває подачу води, запобігаючи затопленню приміщення.

Недоліки розумних будинків так само важливі як переваги. Першим і найважливішим недоліком є ціна обладнання та його установки. Цей аспект багаторазово перекриває витратами весь ефект від комфорту та економії ресурсів, зазначених у перевагах. Якщо порахувати економію від використання системи автоматизації, термін окупності виходить просто фантастичним. Також варто відзначити, що при виході з ладу частини системи, витрати на відновлення працездатності системи можуть виявитися недоцільними.

Другий недолік виникає при монтажі та встановленні системи «Розумний Дім». Для функціонування всіх підсистем, пристроїв та датчиків необхідне прокладання електропроводки для їх з'єднання. Звичайно, можна використовувати датчики, які мають керування по радіоканалу, але при цьому вартість системи багаторазово збільшиться. Якщо система впроваджується в старий будинок, необхідно повністю переробити системи водопостачання, опалення, кондиціювання та вентиляції, а також замінити всю електропроводку і встановити

все необхідне обладнання, можливо, доопрацювати вікна, двері, жалюзі або штори, встановивши на них електропривод. Фактично потрібно зруйнувати свій будинок, щоб збудувати його наново.

Третім недоліком є відведення окремого місця під обладнання. Для коректної роботи всієї системи необхідно використання стабілізаторів напруги для контролю від стрибків напруги та короткого замикання у мережі. Крім цього, для автономного забезпечення цілісності інженерних систем та охоронних функцій слід використання резервного джерела живлення, такого як акумуляторні батареї або генератор, які також потребують відведення певного місця.

1.3 Комерційні рішення та технології управління домашньою автоматизацією

Комерційні системи домашньої автоматизації стали застосовуватися приблизно з 1975 року у США. Новатором у цій галузі стала компанія Pico Electronics, яка створила протокол зв'язку X10. Надалі зарубіжний ринок поповнився та іншими компаніями їхніми комунікаційними протоколами. Нижче буде розглянуто найзначніші.

1.3.1 Технологія X10

Цей комунікаційний протокол і стандарт є найстарішим і найпопулярнішим у сфері систем автоматизації. Був розроблений 1975 року компанією Pico Electronics для управління домашніми електроприладами. Технологія X10 заснована на передачі сигналів електропроводки будинку. Для передачі сигналів використовуються коливання частоті 120 кГц тривалістю 1 мс. Спочатку технологія працювала при частоті 60 Гц і напруга мережі 110 і використовувалася тільки на території США. X10 допомагає вирішити різні завдання домашньої автоматизації, починаючи від управління світловими приладами і закінчуючи системою безпеки.

Використання електромережі для управління електроприладами з одного боку є безперечною перевагою цієї технології. Для підключення приладу до виконавчого модуля немає необхідності прокладати нову проводку або використовувати дорогі датчики та контролери, оснащені бездротовим інтерфейсом. Управління електроприладом йде безпосередньо завдяки передавачам, які формують та відправляють команди X10 в електромережу. З іншого боку, такий підхід несе безліч недоліків, такі як:

— низька швидкість передачі;
— низька схибленість;
— проблема хибного спрацьовування;
— можливі конфлікти пристроїв X10 різних виробників;
— можливий несанкціонований доступ до пристроїв X10 електромережі.

1.3.2 Технологія Z-Wave

Z-Wave є бездротовий протокол передачі даних для домашньої автоматизації. Дана технологія орієнтована для забезпечення простого та надійного способу бездротового керування освітленням, вентиляцією, системами безпеки, домашніми кінотеатрами, автоматичними воротами та елементами управління. У 2005 році компанією Zensys було створено альянс Z-Wave, до якого увійшли різні компанії. На сьогодні існують сотні сумісних продуктів Z-Wave, що продаються під різними марками.

Технологія Z-Wave призначена для забезпечення надійної та швидкої передачі простих команд з малими затримками на швидкості до 100 кб/с. Z-Wave працює в діапазоні частот до 1 ГГц, що зумовлено малою кількістю потенційних джерел перешкод. Z-Wave використовує пористу архітектуру мережі. Пристрої можуть взаємодіяти один з одним за допомогою проміжних вузлів, що дозволяє обійти побутові перешкоди або «мертві зони». Найпростіша мережа є єдиним керованим приладом і первинним контролером. Додаткові пристрої можуть бути додані будь-коли. Мережа Z-Wave може включати до 232 пристроїв.

1.3.3 Технологія KNX

KNX є стандартним протоколом зв'язку для автоматизації будівель. Продукція KNX поширювалася під кількома товарними знаками. Найбільш відомі Instabus, ABB i-Bus, Tebis, Theben.

Всі пристрої, що працюють за технологією KNX, з'єднані між собою двопровідною шиною, що дозволяє їм обмінюватися даними. Функції окремих пристроїв визначається на етапі планування проекту, але можуть бути змінені і адаптовані в будь-який час.

До складу обладнання KNX входять такі типи пристроїв:

— датчики;

— приводи;

— системні пристрої та компоненти.

Датчики є точкою для кожної дії. Вони за фіксування тих чи інших зовнішніх подій. Настання події має викликати певну реакцію системи у відповідь. Після настання такої події сенсор посилає по мережі керуючу команду відповідного виконавчого пристрою. Виконавчі пристрої отримують пакети даних, які потім перетворюються на дії. Такими діями можуть бути управління жалюзі, вимкнення світла або управління системою опалення та кондиціювання повітря.

KNX є повністю розподіленою мережею, яка вміщує до 65 536 пристроїв у 16 біт індивідуального адресного простору. Логічна топологія дозволяє підключити до 256 пристроїв однієї лінії.

Однією із сильних сторін системи KNX є те, що будь-який продукт, розроблений різними компаніями, але при цьому, що має сертифікацію KNX, може працювати спільно. Члени асоціації мають понад 7000 KNX сертифікованих продуктів. Цей широкий асортимент продукції дозволяє:

— керувати освітленням;

— контролювати опалювальні установки та системи кондиціювання;

— аналізувати аварійні сигнали;

— керувати енергоспоживанням природними ресурсами.

Основним недоліком комерційних рішень, використовують свій протокол зв'язку є їх закритість, тобто. використовуються протоколи передачі з фірмовим кодуванням, які завжди дозволяють інтегрувати у єдину систему чуже устаткування.

1.3.4 Компанії на ринку України

На ринку існує не так багато компаній з виробництва систем автоматизації будинку, найбільш відомі це «Luxury systems», «EasySmartBox». Насправді ці підприємства є дистриб'юторами, тобто. вони займаються збутом та гарантійним обслуговування від імені фірми-виробника. Для квартири або будинку площею до 45 кв.м ціна «під ключ» починається від 50 тисяц гривень за базову комплектацію. За мінімальну суму ви отримаєте налаштовану та готову до використання систему функціями, з такими базовими функціями, як управління освітленням та кліматом, при цьому вам не доведеться розумітися на тому, як вона працює, а обслуговування лягає на продавця. Якщо вам це по кишені, такі системи — ваш вибір. Проте у роботі розглядається рішення, яке включає більшість переваг існуючих систем, враховує та виключає наявні недоліки і, головне, є дешевою та конкурентоспроможною.

2 ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ БУДИНКОМ

Аналогом дорогих і професіоналів автоматизованих систем є системи з розряду «зроби сам». Найголовніша перевага — це найнижча ціна обладнання. До мінусів можна віднести нижчу якість та надійність комплектуючих порівняно з комерційними продуктами та відсутність гарантій та технічної підтримки. Крім вище перерахованих недоліків, слід враховувати, що ви повинні мати такі навички:

— знання електрики. розуміння принципів управління різними електроприладами;

— розуміння принципів побудови систем автоматики: типи контролерів, входи та виходи контролерів, види сигналів;

— навички програмування та чітке розуміння алгоритмів роботи бажаної системи;

— гарне знання використовуваного обладнання.

Розумні будинки з розряду «зроби сам» здебільшого засновані на використанні широко відомого мікроконтролера Arduino або одноплатного комп'ютера Raspberry Pi. При створенні системи автоматизації будинку, вибір падає на дані платформи виключно через їхню відкритість і низьку ціну.

2.1 Аналіз та вибір мікронтролера

Raspberry Pi має всі риси комп'ютера, з центральним та графічним процесором, зовнішньою та оперативною пам'яттю, відеовиходом для підключення до екрану, USB-портами. А ось Arduino не має таких можливостей, і на перший погляд схожа на звичайну плату з мікросхемою. Мікрокомп'ютери сімейства Raspberry Pi використовують спеціально розроблену версію Linux, яка є багатозадачною та підтримує графічний інтерфейс. На рисунках 2.1 та 2.2 представлений одноплатний комп'ютер Raspberry Pi та плата-мікроконтролер Ardunio.

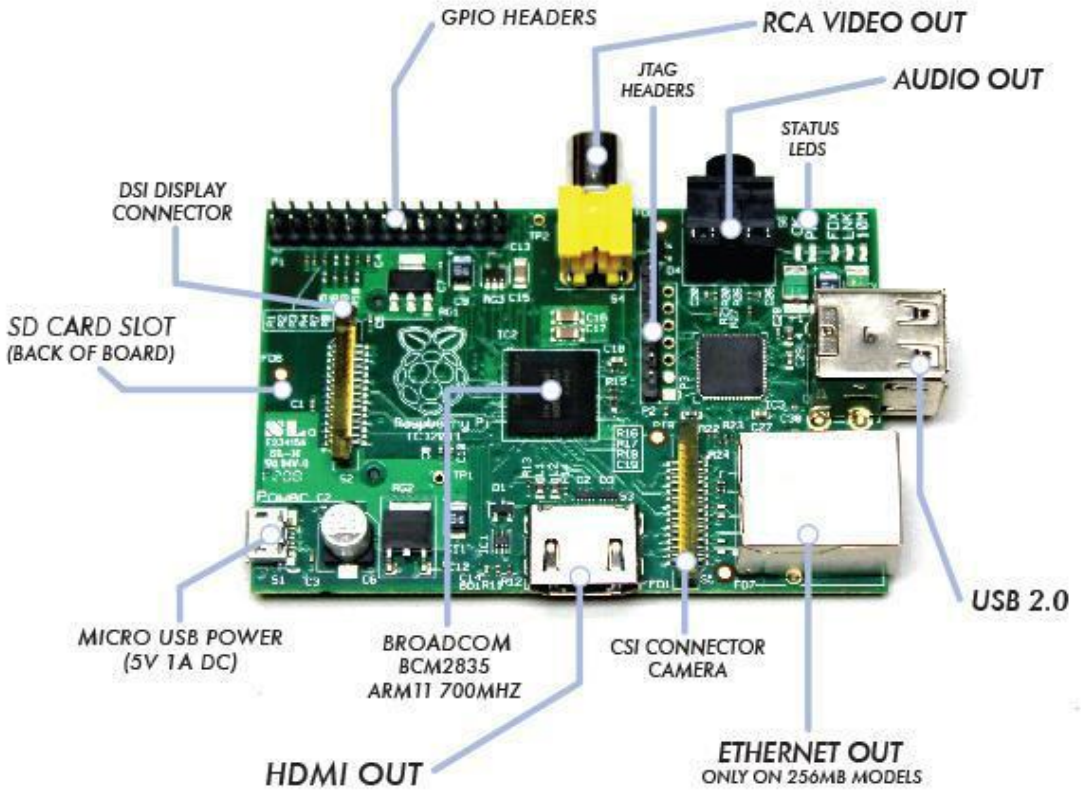


Рисунок 2.1 — Одноплатний комп'ютер Raspberry Pi

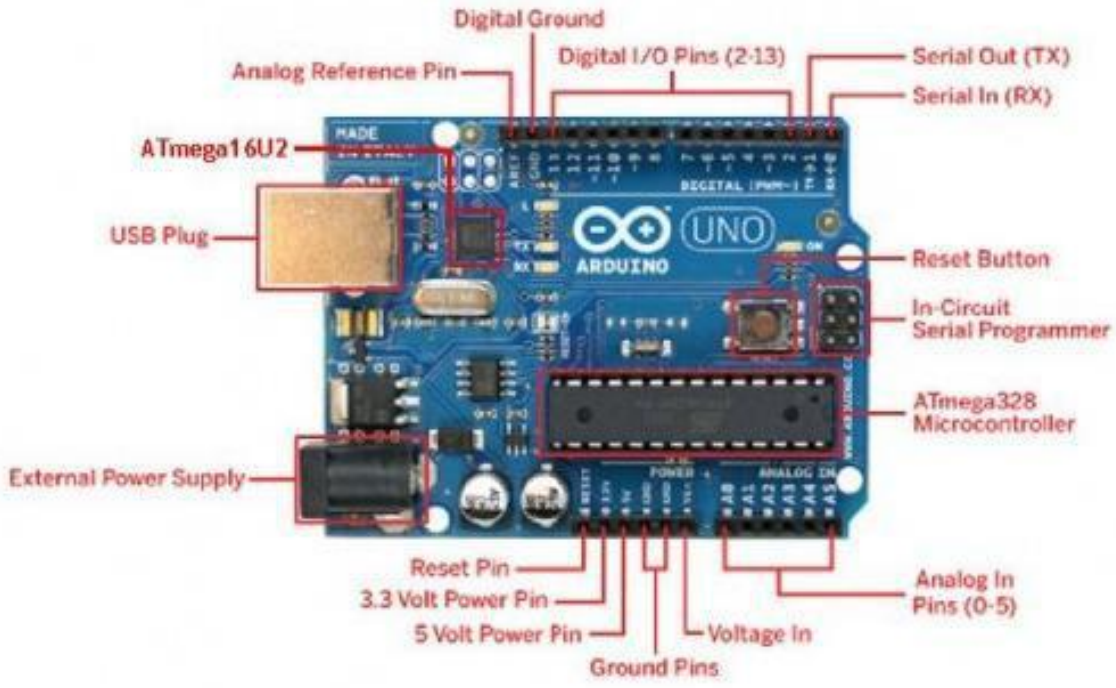


Рисунок 2.2 — Плата-мікроконтролер Ardunio

Плати Arduino є мікроконтролерами, а чи не повноцінними комп'ютерами. На даний момент існує близько 25 різновидів плат, які відрізняються розміром,

процесором, кількістю аналогових та цифрових портів. На Arduino не передбачено використання операційної системи, на відміну від Raspberry Pi, і тому лише одна програма може виконуватись в даний час. Однак це не означає, що немає можливості керувати безліччю завдань на Arduino. Просто для цього буде потрібно трохи інший підхід при написанні коду. При роботі з Arduino в розпорядженні немає жодних базових інструментів операційної системи, але, з іншого боку, відсутність операційної системи дозволяє безпомилково і швидко виконувати завантажену програму. Відсутність операційної системи Arduino дозволяє ініціалізувати програму відразу після включення пристрою. Навіть при непередбаченому відключенні струму на Arduino немає практично нікого ризику пошкодити програму, що виконується, і отримати якісь помилки. Щоб відновити роботу, варто заново підключити Arduino до джерела живлення. На Raspberry Pi робота самого пристрою завершується і відновлюється програмним процесом, як у звичайного комп'ютера, аварійне відключення може призвести до серйозних помилок в операційній системі.

Більшість плат Arduino працює від постійного струму 3.3 або 5В, але гранична напруга може досягати до 20 вольт. Для безперебійної роботи Raspberry Pi, і підключених до неї пристроїв, потрібно постійна напруга 5В, тому Raspberry Pi складно переносити з місця на місце, так просто вставити в нього дві батарейки не вдасться.

Всі сучасні комп'ютери мають доступ до Інтернету, навіть у мікрокомп'ютера Raspberry Pi він теж є. На платі встановлено Ethernet-порт, який забезпечує доступом до мережі. Rapberry Pi підтримує навіть бездротовий Інтернет, все що потрібно купити USB-адаптер для WiFi і встановити відповідний драйвер. Після всіх необхідних налаштувань, з'явиться повноцінний доступ до Інтернету. Крім серфінгу в Інтернеті, Raspberry Pi можна використовувати для створення та одночасної роботи різних веб-серверів та віртуальної мережі. Плата-мікроконтролер Arduino без додаткових модифікацій не має змоги працювати з мережею. Щоб забезпечити доступ до Інтернету, слід обладнати Arduino додатковою платою з Ethernet-портом. Це досить складна робота, яка потребує

певних навичок у схемотехніці та мережевому адмініструванні. І після всіх маніпуляцій, звичного доступу до мережі не буде, все що вдасться зробити — керувати платою-мікроконтролером через Інтернет або запустити на ній веб-сервер.

Функціонал Arduino та Raspberry Pi практично безмежний, щоб розширити його можна використати додаткові плати розширення.

Існує велика різноманітність таких плат, які призначені для вирішення специфічних завдань. Raspberry Pi та Arduino відмінно працюють разом. Arduino краще підходить для керування двигунами, прийому сигналу з датчика, реле управління і т.д. Причому Arduino виступає як керуюча плата, а Raspberry Pi виконує складні обчислювальні операції. На рисунку 2.3 наведено приклад плат розширення.

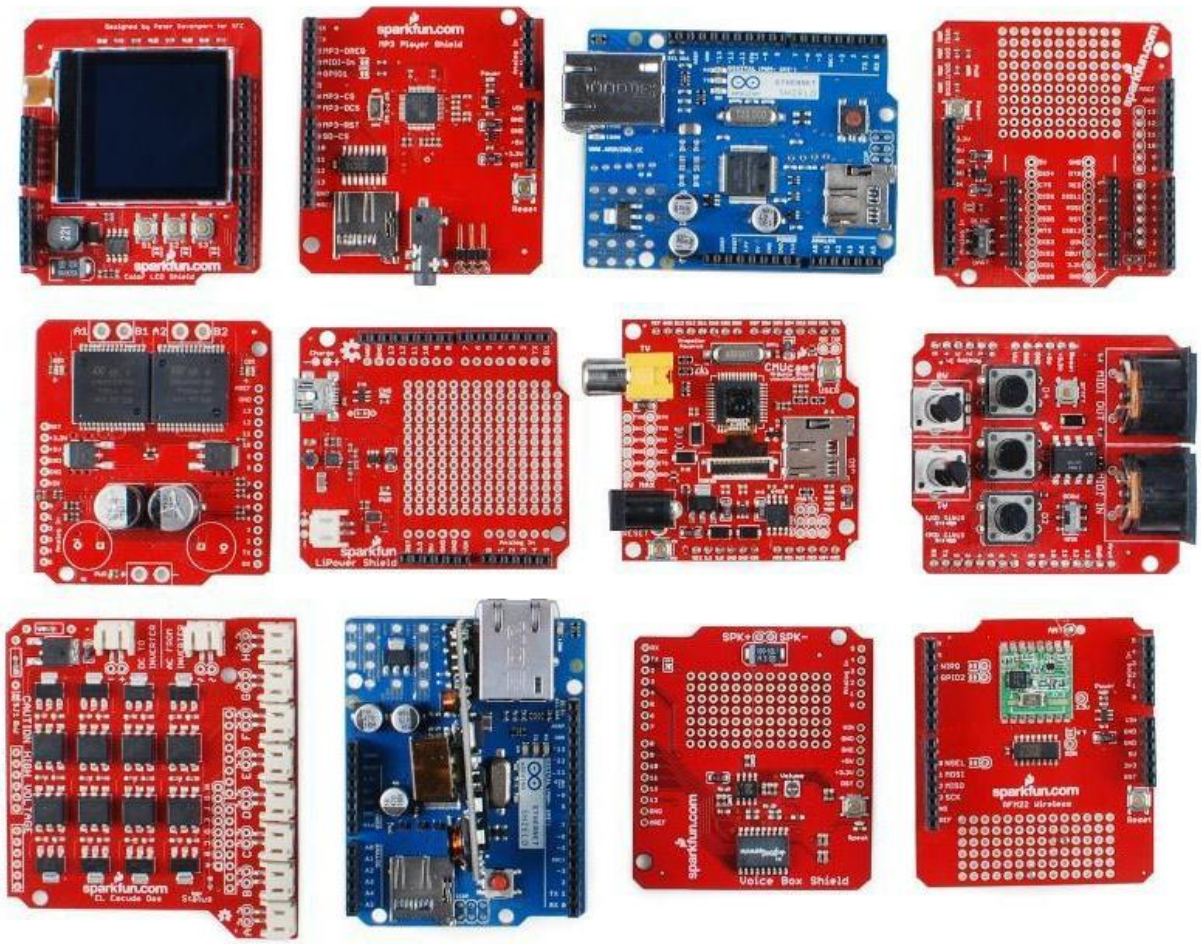


Рисунок 2.3 — Плати розширення

Для створення системи автоматизації будинку, було обрано плату мікроконтролера Arduino. Оскільки основним завданням цієї плати є взаємодія з датчиками та пристроями, Arduino відмінно підходить для апаратних проектів, де потрібно чітко та швидко реагувати на різні сигнали. В даному випадку система автоматизації будинку є таким проектом. Вище згадувалося, що Arduino має безліч різновидів, найбільш підходящою платою реалізації проекту стала Arduino Leonardo.

2.2 Опис Arduino Leonardo

Arduino Leonardo — це пристрій на базі мікроконтролера ATmega32U4. До складу пристрою входить: 20 входів/виходів, кварцовий резонатор на 16 МГц, роз'єм micro-USB, роз'єм живлення, роз'єм для внутрішньосхемного програмування ICSP та кнопка скидання. Об'єм доступної пам'яті для зберігання програми в мікроконтролері ATmega32U4 дорівнює 32 КБ, 4 КБ виділяється під потреби завантажувача. Об'єм оперативної пам'яті типу SRAM дорівнює 2.5 КБ. Крім того, мікроконтролер має 1 КБ EEPROM. Головна відмінність Leonardo від усіх попередніх плат полягає в тому, що в мікроконтроллер ATmega32U4 вбудований USB-контролер, що виключає необхідність додаткового процесора.

Arduino Leonardo як джерело живлення може використовувати micro-USB або зовнішню батарею або адаптер мережі. На рисунку 2.4 представлена плата-мікроконтролер Arduino Leonardo.

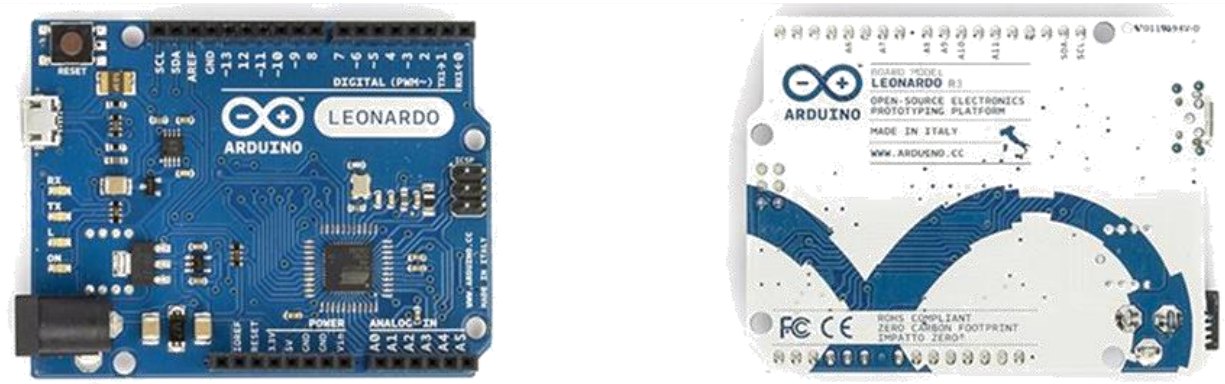


Рисунок 2.4 — Arduino Leonardo

Напруга джерела живлення може змінюватись від 6 до 20 В. Для захисту від короткого замикання та перевантаження, на платі передбачені запобіжники, що відновлюються. При перевищенні споживаного струму більше 500 мА запобіжник автоматично розірве з'єднання. Arduino Leonardo має низку можливостей для здійснення зв'язку з комп'ютером або іншими мікроконтролерами. Чіп ATmega32U4 включає в себе приймач UART (універсальний асинхронний приймач/передавач), призначений для організації зв'язку з іншими цифровими пристроями. Мікроконтролер ATmega32U4 підтримує послідовний зв'язок через USB, при підключенні до комп'ютера визначатиметься як віртуальний COM-порт. Варто відзначити, що в мікроконтролері ATmega32U4 реалізована підтримка інтерфейсу ICSP,

На рисунках 3.5, 3.6 представлені виводи (піни, порти), що знаходяться мікросхемі ATmega32U4 і на самій платі, таблиці 2.1 представлено їх докладне призначення. Дані порти необхідні для підключення аналогових та цифрових пристроїв та для забезпечення живлення. Кожен порт управляється трьома регістрами – DDR, PORT, PIN, які визначені змінними серед Arduino IDE. Регістр DDR визначає, чи є порт входом або виходом, так само він служить для визначення напрямку передачі даних порту. PORT є регістром даних порту, а PIN зчитує стан та адресу вхідних/вихідних «штирків».

Важливою особливістю Arduino Leonardo є те, що на платі є 7 виходів (на малюнку 6 відзначені зеленим), які можуть використовуватися як ШІМ (PWM). Плати Arduin про зазвичай не можуть видавати довільну напругу. Вони можуть видати напругу живлення – логічну 1 (HIGH, високий), або GND – логічний 0.

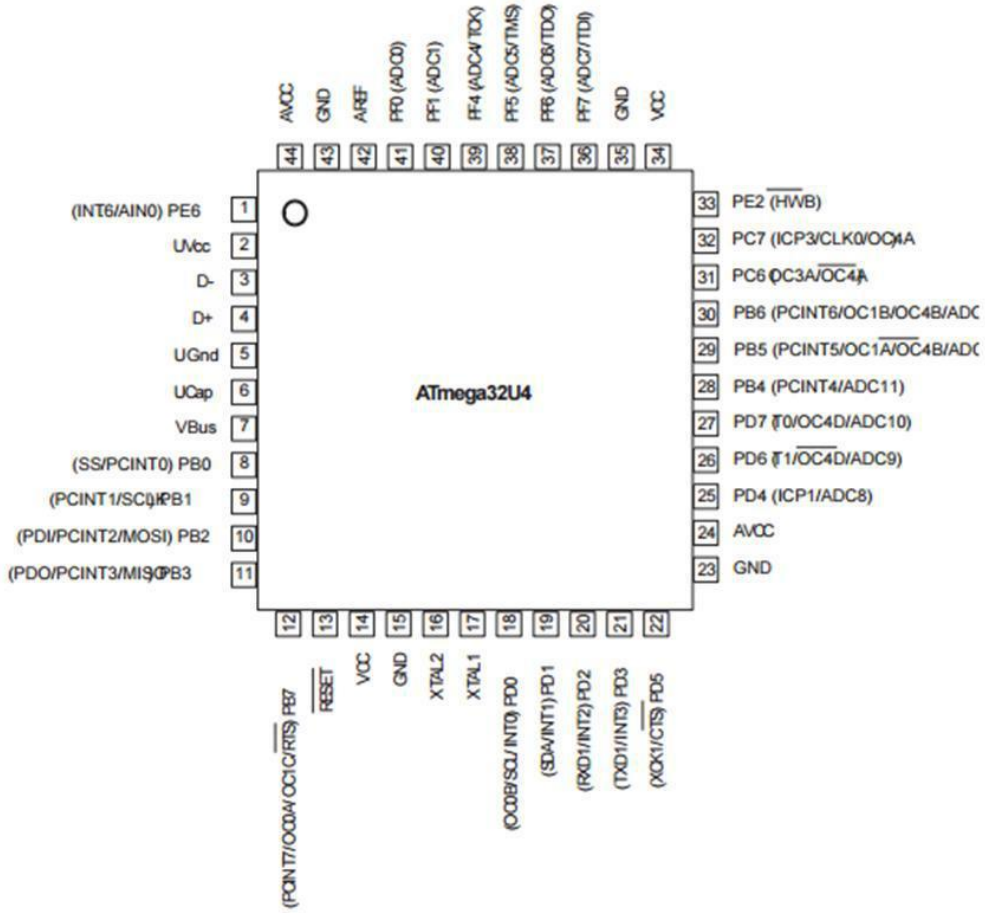


Рисунок 2.5 – Виводи на мікросхемі ATmega32U4

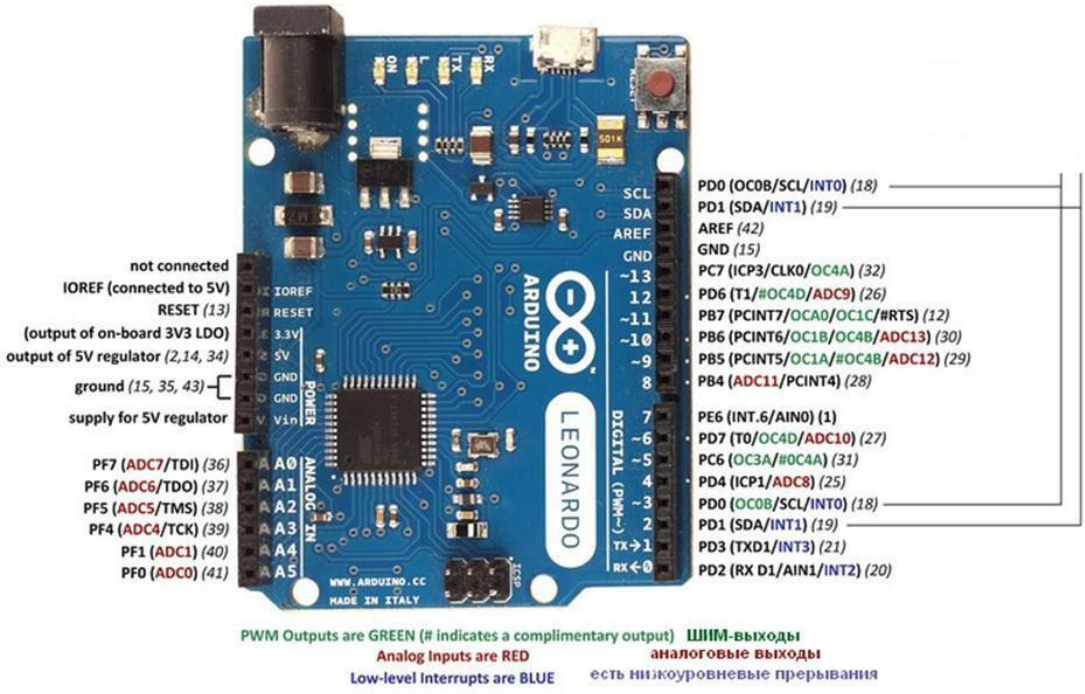


Рисунок 2.6 — Виводи на платі Arduino Leonardo

Таблиця 2.1— Призначення виводів

VCC	Напруга живлення
GND	Земля
AREF	Аналоговий опорний вхід для аналого-цифрового перетворювача
AVCC	Контакт напруги для всіх цифрових та аналогових каналів конвертера
VBUS	Контакт позитивного потенціалу джерела живлення, відповідальний за роботу USB
UGND	Контакт землі, що відповідає за роботу USB
UVCC	Внутрішній регулятор вхідної напруги живлення USB
UCAP	Внутрішній регулятор вихідної напруги джерела підключеного до зовнішнього конденсатора
XTAL1	Вхід інвертуючого підсилювача системного генератора та вхід зовнішніх системних тактуючих імпульсів
XTAL2	Вихід підсилювача системного генератора, що інвертує
D-	Контакт шини приймання даних USB
D+	Контакт шини передачі даних USB
Port B	Група висновків DIGITAL із номерами 11-8
Port C	Група висновків DIGITAL із номерами 13, 5
Port D	Група висновків DIGITAL з номерами 0, 1, 2, 3, 4, 5, 6, 12, SDA (лінія даних), SCL (лінія тактування)
Port F	Група висновків ANALOG із номерами A0-A5
RESET	Вхід скидання
ISCP	Порт, що використовується для роботи з програматором.

2.3 Можливості поточної системи автоматизації будинку

Величезна кількість різноманітних виходів на платі, дозволить підключити різні датчики та пристрої, так, як Arduino Leonardo сама по собі не зможе виконувати функції системи автоматизації будинку. Перед етапом вибору датчиків та пристроїв слід визначитися з призначенням поточної системи домашньої автоматизації. Поточний проект системи автоматизації буде призначено для кімнати площею 15 кв.м, що знаходиться в приватному будинку. Система домашньої автоматизації повинна виконувати такі завдання:

— управління освітлювальними та побутовими приладами;
— віддалене керування всією системою;
— контроль температури всередині приміщення та на вулиці;
— контроль витоків води;
— контроль несанкціонованого проникнення.

Дана система є прототипом, вона не може контролювати всі інженерні вузли, прилади, опалювальні системи та забезпечувати повноцінну безпеку житла. Це пов'язано з великими фінансовими витратами на покупку комплектуючих, впровадження системи в будинок та нестачею навичок у програмуванні та електротехніці. Надалі дана система домашньої автоматизації модернізуватиметься і з часом не поступатиметься за якістю та функціоналом повноцінним комерційним рішенням.

2.4 Вибір датчиків, додаткових пристроїв та принцип їх роботи

2.4.1 Датчик звуку

Управління освітлювальними приладами за допомогою системи домашньої автоматизації дозволяє забезпечити комфортну експлуатацію приміщення, заощадити витрати електроенергії та продовжити термін служби електроприладів. Багато систем дозволяють керувати світлом за допомогою мобільного телефону, натисканням віртуальної кнопки або навіть голосовими або звуковими

командами. У поточному проекті буде реалізована функція увімкнення та відключення світла по хлопку, для такого типу керування використовується датчик звуку.

Датчик звуку застосовується для стеження за рівнем шуму або виявлення гучних сигналів, таких як хлопок, стукіт або свист. Він має великий функціонал, але в даному випадку використовуватиметься для управління освітлювальними приладами. Для поточного завдання було обрано датчик на мікросхемі lm393, вибір обумовлений низькою ціною та простотою його роботи. На рисунку 2.7 представлений цей датчик.

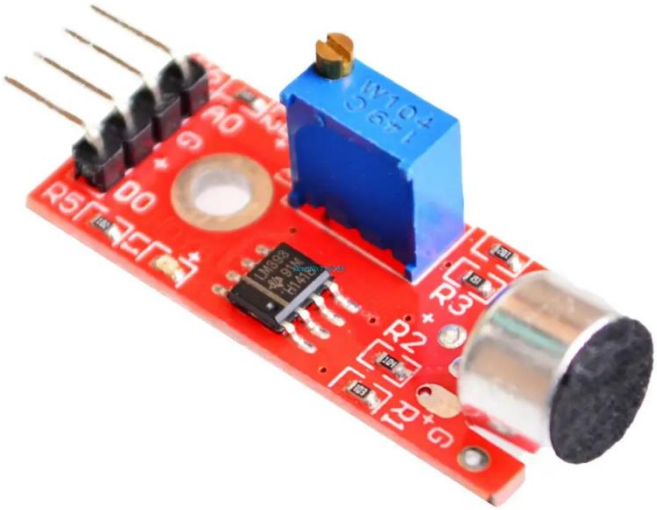


Рисунок 2.7 — Датчик звуку на мікросхемі lm393

Принцип роботи датчика дуже простий. Мікрофон перетворює звукові коливання на коливання електричного струму. Щоб позбавитися шумів і посилити сигнал, застосовується електронна обв'язка (на рисунку 3.7 вона розташована по всій поверхні плати).

Потенціометр (підстроювальний резистор) є резистором з рухомим контактом, що служить для регулювання чутливості мікрофона. Потенціометр у цьому датчику має 3 виводи. На останні висновки подається +5V і GND, між цими контактами знаходиться спеціальна речовина, яка є «резистивним», по ньому якраз і рухається повзунок, з'єднаний із середнім контактом. Змінюючи положення ротора, що знаходиться всередині підстроювального резистора,

змінюється і вихідна напруга. Наприклад, переміщуючи ротор з лівого крайнього становища в крайнє праве, збільшується опір, а водночас і зменшуємо напругу. Потенціометр застосовується, коли необхідно змінити умови спрацьовування датчика звуку без зміни його програмної частини. Це зручно, так, як датчик підключається до керуючої плати через 3 контакти:

— +5V;

— GND;

— OUT – зчитування сигналу датчика мікроконтролером.

2.4.2 Температурний датчик

Наш повсякденний побут тісно пов'язаний із таким важливим параметром як температура. Контроль температури дуже важливий для забезпечення комфорту, а також для раціонального використання електроенергії та природних ресурсів. Сучасні «Розумні будинки» можуть автоматично контролювати мікроклімат як усього будинку, так і окремих приміщень. У поточній системі автоматизації, контроль температури, нестиме більше інформаційний характер. Всередині кімнати та зовні будуть встановлені два датчики температури. Використання цих датчиків дозволить у майбутньому створити повноцінну систему домашньої автоматизації з можливістю автоматичного контролю мікроклімату приміщення.

Як датчик температури був обраний мініатюрний DS18B20. Спочатку передбачалося використання більш функціонального датчика DHT21, який має, наприклад, додаткову функцію — вимірювання вологості повітря, але для поточних цілей використання DHT21 було б не доцільним, через високу ціну та надмірну функціональність.

Безперечною перевагою є те, що цей датчик цифровий, це дозволяє зчитувати і передавати на Arduino Leonardo кожну зміну температури з високою швидкістю і точністю. Крім цього, можна підключити паралельно до 127 таких датчиків, використовуючи монтажну плату.

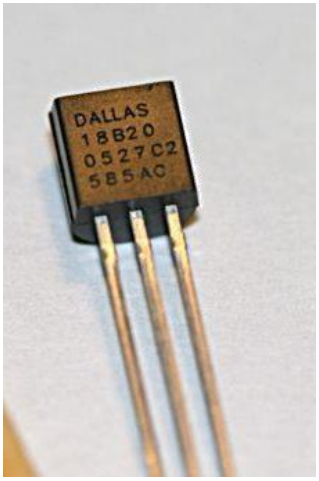


Рисунок 2.8 — Датчик температури DS18B20

У поточному проекті буде використано 2 DS18B20. Один призначений контролю температури всередині приміщення, інший зовні. Для підключення до плати-мікроконтролера, використовується 3 стандартні контакти: GND, +5V, DATA. Але для коректного функціонування датчика слід використовувати резистор, що підтягує, на 4.7 кОм. Підтягуючий резистор утворює ланцюг між DATA і +5V, що забезпечує підтяжку сигналу між провідником, по якому поширюється електричний сигнал, і живленням.

2.4.3 Датчик руху та геркон

Датчик руху (або детектор руху) та геркон є основою системи безпеки, тому що ці пристрої першими визначають несанкціоноване проникнення до будинку. Детектори руху використовують різні технології виявлення руху. Найбільш підходящим для цього проекту є PIR HC-SR501, даний датчик представлений на рисунку 3.9.

PIR (пасивні інфрачервоні датчики) детектори мають малі габарити, низьку ціну, споживають мало енергії, легкі в експлуатації, практично не схильні до зносу. Детектори руху складаються з чутливого піроелектричного сенсора, який вимірює рівень інфрачервоного випромінювання. Кожен об'єкт у кімнаті має певний рівень випромінювання. При зміні температури змінюється і рівень випромінювання. Датчик HC-SR501 використовує чіп BISS0001, який сприймає

зовнішнє джерело випромінювання та проводить мінімальну обробку сигналу для його перетворення з аналогового в цифровий вигляд.

Рисунок 2.9 — Датчик руху PIR HC-SR501

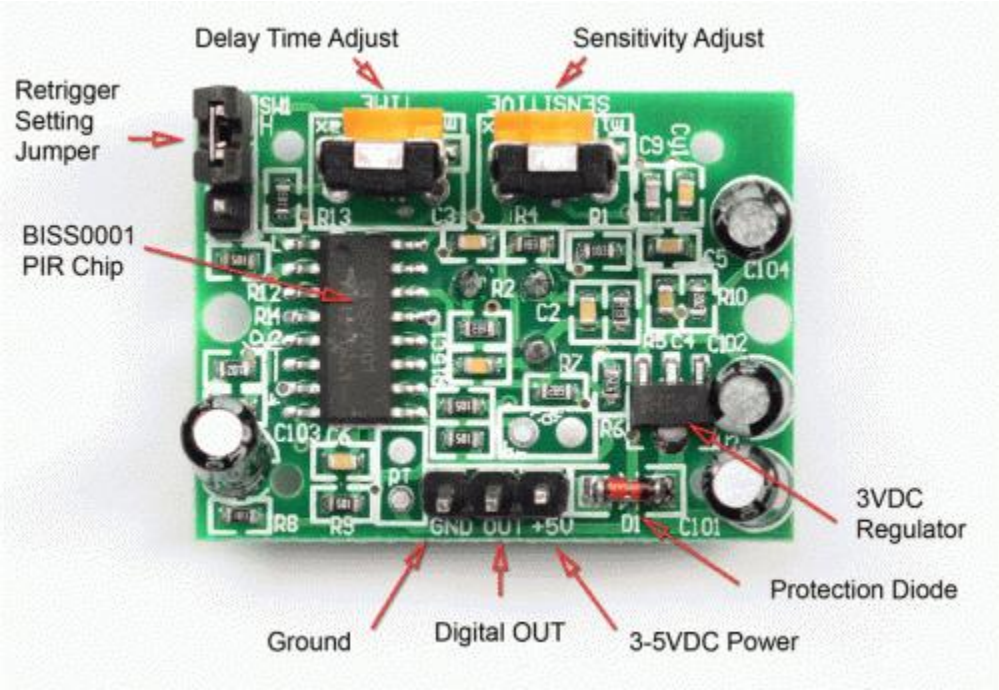


Рисунок 2.10 — Система керування та «тонкого» налаштування

HC-SR501 має два режими роботи H (повторюване захоплення) і L (одинкове захоплення). Якщо перемичка встановлена на H, то при спрацьовуванні сенсора кілька разів поспіль на виході OUT залишається високий логічний рівень. Якщо L, то на виході з'являється окремий імпульс. Для виявлення несанкціонованого руху, в поточній системі домашньої автоматизації використовується звичайний режим L. Для коректної роботи детектора потрібне його налаштування, не тільки через програмні засоби, але і через 2 підстроювальні резистори, розташованих на друкованій платі. Це робиться для того, щоб унеможливити помилкові спрацьовування датчика. Варто відзначити, що функціонал PIR HC-SR501 не обмежується лише виявленням руху. Даний датчик може застосовуватись і для виявлення вогню.

Геркон (герконовий датчик) є спеціальним електромеханічним пристроєм з магнітокерованими феромагнітними контактами. Назва «геркон» походить від словосполучення «герметичний контакт». Це пояснює пристрій датчика.

Рисунок 3.11 – Геркон

По суті, геркон — це два контакти, що у вакуумній колбі. Контакти можуть бути у двох станах — замкненому і розімкнутому, свій стан вони змінюють з допомогою на них магнітного поля. Геркони дуже популярні датчики, що використовуються в основному для контролю відкриття вікон та дверей. У поточній системі цей датчик використовуватиме контролю відкриття вікна.

2.4.4 Датчик витоків води та сервопривід

Будь-яка людина, яка хоч раз стикалася з такою ситуацією, як, порив труби в будинку або крана, знає, наскільки важливо вчасно виявити проблему. Навіть невеликі протікання води, які здаються незначними, В результаті можуть призвести до великих матеріальних витрат. В даний час існує безліч комерційних рішень у сфері контролю та оповіщення при аварійних ситуаціях, пов'язаних із поривом води. Розрізняються вони ціною, функціоналом та відкритістю. як детектор води була обрана недорога плата-сенсор з компаратором LM393.

Рисунок 2.12 — Датчик витоку води

Даний датчик призначений в першу чергу для виявлення води, але при певному налаштуванні підстроювального резистора та зміненого програмного коду може використовуватися і для контролю рівня води. Це робиться для того, щоб позбутися помилкових спрацьовувань, особливо коли датчик знаходиться в приміщенні з високою вологістю або при попаданні на нього незначної кількості води. Звичайно, такий тип датчика від мокрих підлог не врятує, але все ж таки зможе запобігти великим витокам.

Датчик складається з друкованої плати, якою знаходяться струмопровідні контакти та плати з компаратором LM393. Як контакти, призначені для передачі даних, використовується аналоговий, або цифровий контакт. Плата-

мікроконтролер Arduino Leonardo може зчитувати цифрове значення датчика від 0 до 1023 або аналогове показання в діапазоні напруг від 0 до 5 В. Якщо сенсорна плата має сухий стан, на аналоговий вихід палати подається напруга 5 В, при використанні цифрового значення 1023. У випадку, якщо на сенсор потрапляє волога, то напруга на аналоговому виході змінюється від 5 В до 0 В, на цифровому від 1023 до 0. Значення змінюються в залежності від кількості або рівня води на сенсорі.

Датчик витоку води має як попереджати про пориви, а й боротися із нею, тобтоперекривати воду. Для цього може використовуватися електромагнітний клапан (соленоїд), або сервопривід (серводвигун).

Рисунок 2.13 — Електромагнітний клапан

Рисунок 2.14 — Сервопривід

При виявленні пориву сигнал передається від датчика, до керуючої плати Arduino, потім від Arduino до модуля реле і від реле до сервоприводу або електромагнітного клапана, які в свою чергу і перекривають воду. У системі використовується сервопривід.

Сервопривід є поворотним приводом або лінійним приводом, який дозволяє точно контролювати кутове або лінійне положення, а також швидкість і прискорення. Привід є сервомеханізм із замкнутим контуром, який використовує зворотний зв'язок за положенням, для контролю своєї позиції та кінцевого положення. Привід має кілька складових елементів. Найголовніша — це електромотор. Супутнім пристроєм електромотора є редуктор, який знижує надмірну швидкість обертання електромотора. Для контролю за положенням електромотора використовується потенціометр. При повороті бігунка потенціометра відбувається зміна опору, пропорційне куту повороту. Крім цього, в сервопривод є електронна начинка, яка відповідає за прийом зовнішнього сигналу, зчитування значень з потенціометра, їх порівняння.

2.4.5 Реле

Реле є електричним перемикачем. Такі пристрої використовують електромагніт для механічної роботи перемикача. Реле використовуються там, де необхідно контролювати електричний ланцюг, через сигнал малої потужності, або де кілька схем повинні контролюватись одним сигналом.

Пристрій реле можна розділити на дві частини: вхід та вихід. Секція входу має котушку, яка генерує магнітне поле, коли невелика напруга від електронної схеми впливає на неї. Ця напруга називається робочою. Зазвичай реле доступні в різній конфігурації робочих напруг, як 5В, 9В, 12В, 24В і т.д. Вихідна секція складається з контакторів (супрекосателей), які включаються та відключаються механічно. У базовій схемі реле є три типи контакторів: нормально відкритий NO, нормально закритий NC та загальний COM. При без вхідного стану, COM підключений до NC. При подачі робочої напруги котушки реле збуджується і COM змінює контакт NO.

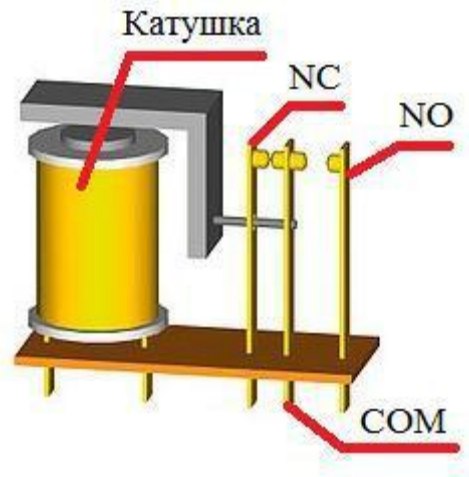


Рисунок 2.15 — Стандартна схема реле

Роль реле у системі домашньої автоматизації велика. Реле є зв'язуючим пристроєм між платою-мікроконтролером та електричним приладом, що працює від мережі 220 В. Хоч і реле, складається всього лише з електромагнітної котушки та найпростішої електронної обв'язки, без нього не вдалося б зв'язати плату-мікроконтролер і побутовий або освітлювальний прилад, який постійно потребує увімкнення або вимкнення з мережі. Працюючий прилад від мережі 220 В, підключений до Arduino без реле, вивів би його з плату-мікроконтролер, без можливості ремонту.

Для поточного проекту було обрано модуль SRD-05VDC-SL-C із двома реле. Модуль буде керувати одним побутовим приладом та сервоприводом.

Рисунок 2.16 — Модуль реле SRD-05VDC-SL-C

2.4.6 Адаптер Ethernet ENC28J60

Більшість систем домашньої автоматизації працюють в автоматичному режимі та мають встановлені сценарії. Однак завжди існує інформація, яку потрібно повідомити користувачеві або яка була б у принципі корисна. Крім цього, часом необхідно мати можливість дистанційно керувати, світлом і побутовими приладами. Так давно «Розумні будинки» було неможливо надати користувачеві віддалений контролю над інженерними вузлами і побутовими приладами, т.к. не мали доступ в Інтернет. Усі системи та прилади, що входять до «Розумного дому» могли контролюватись лише централізовано, тобто. всередині приміщення був пульт або контрольна панель. Зараз ситуація повністю змінилася – за бажанням користувача, всі системи домашньої автоматизації можуть мати віддалений та централізований контроль.

В поточному проекті буде реалізовано управління системою автоматизації будинку через локальну мережу. Так, як вся система керується платою-мікроконтролером Arduino Leonardo, потрібно вирішити проблему доступу Інтернет. Для надання доступу до мережі, чудово підходить адаптер ethernet ENC28J60.

ENC28J60 одна з найцікавіших плат розширення, призначена для управління платою Arduino через локальну мережу або Інтернет. Адаптер має специфікацію IEEE 802.3i, для підключення до маршрутизатора використовується інтерфейс 10BASE-T, який використовує кручену пару. Широкі апаратні можливості ENC28J60 дозволяють створювати різні веб-сервери. Адаптер пов'язує плату-мікроконтролер із мережею через інтерфейс SPI (на Arduino Leonardo використовується ICSP). ENC28J60 дозволяє віддалено керувати різними приладами, причому керування може відбуватися одночасно або по черзі з кількох комп'ютерів чи смартфонів.

Веб сервер за допомогою ENC28J60 працює наступним чином. Як клієнт виступає браузер комп'ютера або смартфона, за допомогою якого відбувається

підключення до сервера. Основним завданням клієнта є надсилання різних запитів серверу.

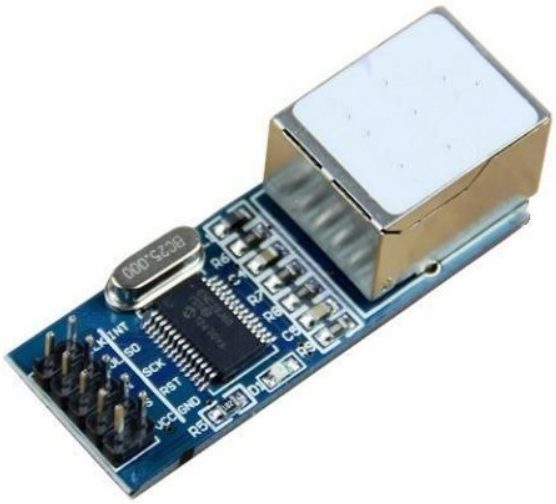


Рисунок 2.17 — Ethernet адаптер ENC28J60

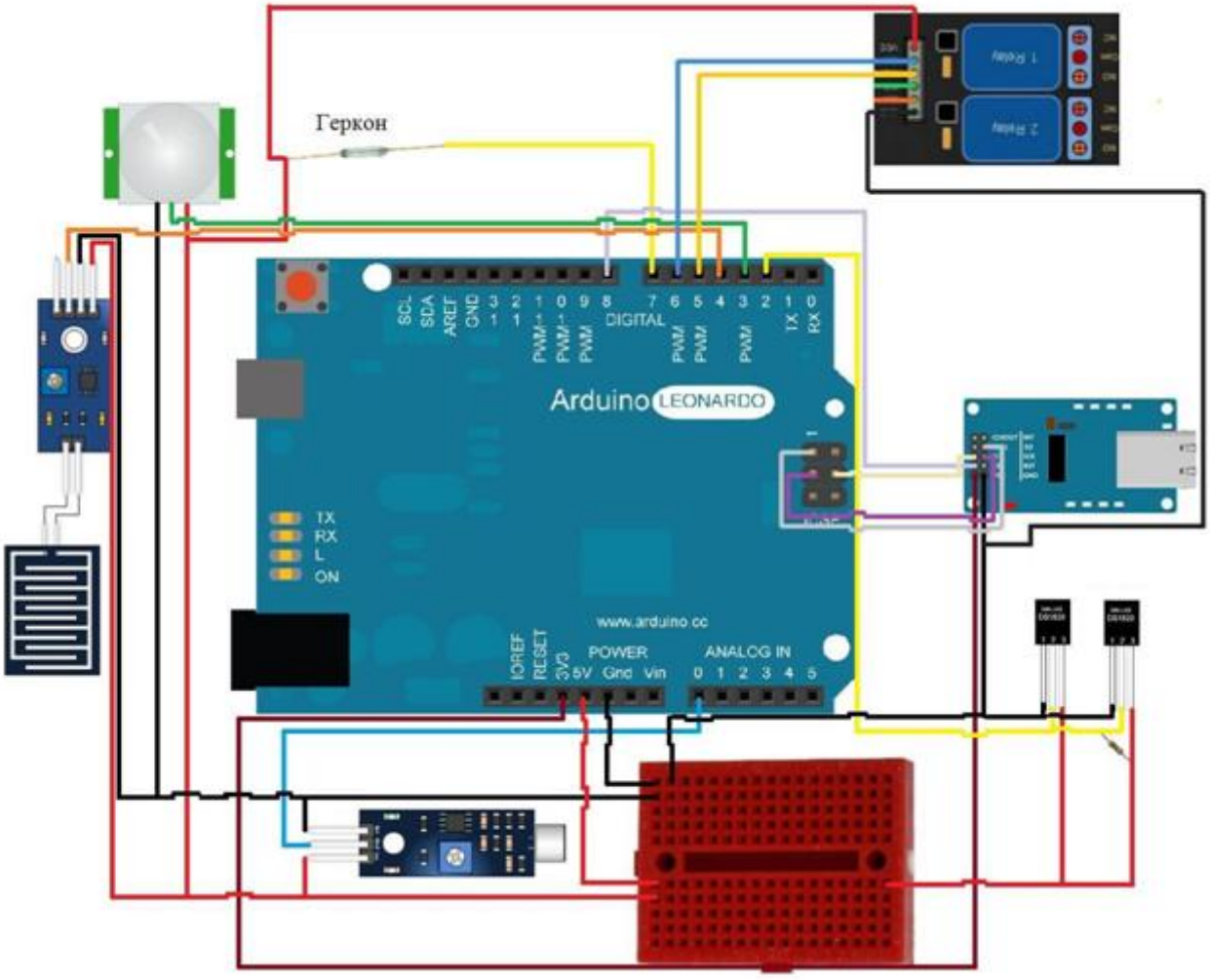
Використовується два види запитів GET та POST. GET — запит на відображення інформації, POST — запит про передачу будь-яких даних. Переходячи на IP адресу сервера, який заданий у коді, надсилається запит GET і сервер видає заздалегідь сформовану сторінку HTML з необхідною інформацією. Щоб надіслати POST запит, слід лише натиснути на віртуальну кнопку або заповнити відповідну форму. Після цього, змінені дані пересилаються на сервер і обробляються у відповідність до зазначеного алгоритму. Як сервер виступає сам адаптер ENC28J60, що працює відповідно до HTTP протоколу. ENC28J60 повинен мати унікальну IP та фізичну адресу. Дані параметри задаються програмним кодом, що зберігається на самому чіпі плати Arduino Leonardo.

Варто зазначити, що керування системою може відбуватися тільки тоді, коли смартфон або комп'ютер перебуває в одній підмережі разом з усіма компонентами системи автоматизації будинку. Це пов'язано з тим, що Інтернет з'єднання забезпечує провайдер Yota. Yota не надає за промовчанням так званий «білий» IP. Провайдер резервують під себе одну або кілька адрес, і всі, хто до нього підключений виходять в мережу тільки під цією адресою. Припустимо, провайдер має адресу «192.168.1.10», тоді у всіх, хто підключений до нього буде

така сама IP-адреса в Інтернеті «192.168.1.10». При цьому у внутрішній (локальній) мережі у кожного клієнта буде своя власна внутрішня адреса, але в Інтернеті – у всіх спільна («сірий» IP).

2.5 Об'єднання компонентів у єдину систему

На рисунку 2.18 відображено схему всієї системи, без споживачів (електроприладів). Для зручності підключення була використана монтажна плата, на яку було виведено +5V та GND. Як видно, на Arduino Leonardo більше половини пінів вільно, це дозволить у майбутньому підключити додаткові датчики та пристрої, які дозволять удосконалити систему автоматизації будинку.



у

Рисунок 2.18 — Схема системи управління будинком

Даний набір компонентів призначений тільки для однієї кімнати, щоб система автоматизації могла керувати всіма інженерними вузлами та забезпечувати повну безпеку будинку, необхідно витратити близько 1000-1500 гривень на додаткові датчики та пристрої, а також на монтаж проводки. Ця ціна набагато нижча за комерційні рішення. Але при самостійному створенні «Розумного дому» потрібно більше часу на використання готового продукту, а також розробка системи потребує певних навичок у програмуванні та електротехніці.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ УПРАВЛІННЯ БУДИНКОМ

3.1 Налаштування мікропроцесорної платформи

Найчастіше мікроконтролерам потрібний доступ до Інтернету, локальної мережі. Це може стати в нагоді, наприклад, для побудови розумного будинку, встановлення простого веб-сервера та багато іншого. Все це може забезпечити Ethernet модуль ENC28J60.

Необхідно забезпечитипро підключення ENC28J60 до Arduino. Цей модуль дозволяє Arduino виходити в Інтернет або локальну мережу.

Модуль підключається до Arduino за допомогою SPI інтерфейсу. Тактова частота інтерфейсу може досягати 20 МГц. Для підключення до мережі TCP/IP використовується роз'єм RJ-45. Модуль гальванічно розв'язаний із ним.

Контролер модуля має максимальну тактову частоту 25 МГц. Усі функції роботи з мережними протоколами покладено нього, Arduino залишається лише відправляти пакети даних і приймати отримані.

Модуль працює при напрузі 3,3В завдяки чому сумісний і з мікроконтролерами STM32 або STM8. Максимальний струм може становити 250 мА, номінальний - 170 мА.

Схему підключення ENC28J60 до Arduino показано на рисунку 3.1.

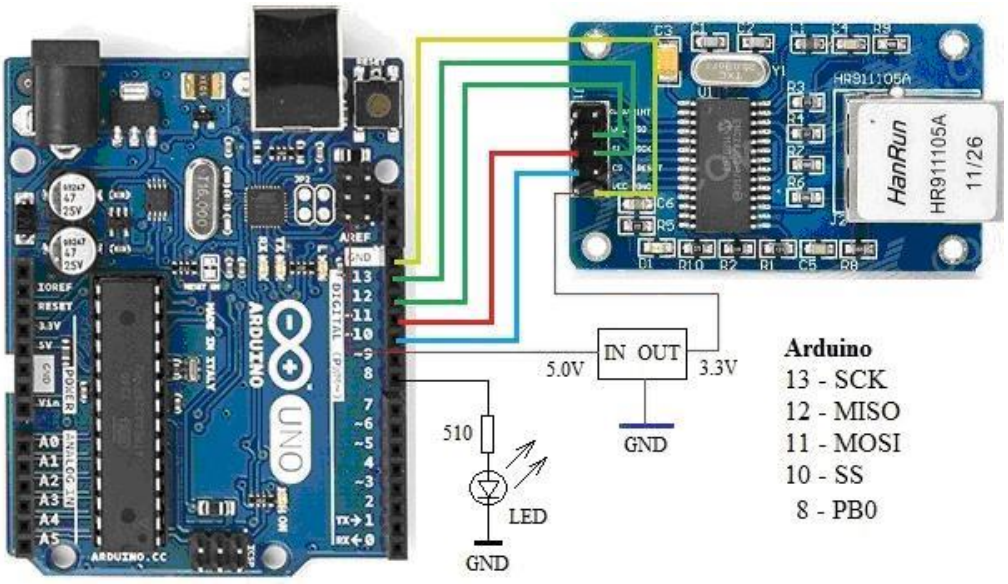


Рисунок 3.1 — Схему підключення ENC28J60 до Arduino

Номер піна CS можна змінити за допомогою функції ether.begin().

Зверніть увагу на те, що модуль споживає досить багато енергії і струму, що видається портом 3,3 В, може не вистачити, особливо на китайських клонах. У цьому випадку потрібно буде використовувати зовнішнє джерело живлення.

Тепер перейдемо до програмування та підключення ENC28J60 до Arduino.

Серед стандартних бібліотек Arduino IDE немає бібліотеки для нашого модуля, тому будемо використовувати бібліотеку EtherCard, яка підтримує наступні моделі Arduino: UNO, Mega, Leonardo, Nano, Pro Mini, LilyPad, Duemilanove та інші, що базуються на AVR мікроконтролерах. Плати з архітектурою ARM, такі як 101, Zero, Due їй не підтримуються. Ця бібліотека не єдина, є ще EtherEncLib та UIPEthernet. Спочатку ініціалізація бібліотек, які наведено у додатках

Далі вказуються піни для підключення модуля з контролером ENC28j60. Також нам необхідно вказати параметри нашого мережевого пристрою. Для цього вказуємо MAC адресу — пам'ятайте, він не повинен збігатися з MAC адресою Ваших мережевих пристроїв. Так само й IP Адреса — має бути індивідуальною — але перебувати у Вашій підмережі.

Наприклад, у Вас роутер(192.168.0.1), Ваш ПК(192.168.0.5) Ваш пристрій може бути(192,168,0,100).

Наприклад, у Вас роутер(192.168.4.1), Ваш ПК(192.168.4.10) Ваш пристрій може бути(192.168.4.100).

Далі потрібно вказати порт. За замовчуванням 80 — оскільки веб-браузери за замовчуванням опитують саме його (дивіться тест зміни порту у відео нижче).

Далі ETHER_28J60 ethernet; — Вказуємо на ім'я об'єкта для звернення (ethernet), нижче в програмі ми звертатимемося по цьому імені.

Нам необхідна ініціалізація мережевого контролера — застосовуємо всі установки адрес та портів.

Спробуємо вивести в послідовний порт комп'ютера IP адресу пристрою, маску підмережі, стандартний шлюз і адресу DNS сервера.MAC-адреса тут

представлена в шістнадцятковому вигляді. Слідкуйте за тим, щоб він не збігався з жодною з вже наявних адрес в мережі, інакше можливі проблеми.

```
static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };

byte Ethernet::buffer[700];

void setup () {
Serial.begin(57600);
Serial.println(F("n[testDHCP]"));

Serial.print("MAC: ");
for (byte i = 0; i < 6; ++i) {
Serial.print(mymac[i], HEX);
if (i < 5)
Serial.print(':');
}
Serial.println();

if (ether.begin(sizeof Ethernet::buffer, mymac) == 0)
Serial.println(F("Failed to access Ethernet controller"));

Serial.println(F("Setting up DHCP"));
if (!ether.dhcpSetup())
Serial.println(F("DHCP failed"));

ether.printIp("My IP: ", ether.myip);
ether.printIp("Netmask: ", ether.netmask);
ether.printIp("GW IP: ", ether.gwip);
ether.printIp("DNS IP: ", ether.dnsip);
}

void loop () {}
```

Рисунок 3.2 — Лістинг коду підключення

3.2 Веб-інтерфейс панелі керування

Інтелектуальний будинок складний в облаштуванні, адже в ньому присутня маса датчиків, контролерів і сенсорів, які завжди передають інформацію на пристрій, що надалі після переробки інформації передає кінцеві команди приладам. Системи домашньої автоматизації можуть працювати автоматично.

Але завжди є дані, які потрібно повідомити користувачеві. Для цього якраз використовують панель керування або контрольну панель. Крім цього, не завжди можливо запрограмувати всі дії користувача, які будуть виконуватись автоматично. До таких дій можна віднести незаплановане включення світла або електроприладів. Використання панелі керування дозволить користувачеві вручну керувати різними приладами.

При проектуванні системи мають бути враховані такі рекомендації та положення:

— інтерфейс системи має бути максимально простим та зрозумілим користувачеві;

— дизайн web-ресурсу має бути достатньо інформативним;

— реєстраційне вікно не створюється та не використовується;

— для роботи web-ресурсу потрібен доступ до Інтернету, весь процес роботи та управління системою повинен здійснюватися в онлайн-режимі;

— має бути передбачена можливість додавання до програмно-апаратного комплексу нових елементів та пристроїв;

— додавання нових елементів має виконуватися без порушення загальної структури ресурсу.

Проектований web-ресурс має стати основою з мінімальною функціональністюнеобхідною та достатньою для простого використання. Передбачається, що можливості системи можуть бути розширені.

У розроблюваної системі має бути реалізований наступний інтерфейсний функціонал та компоненти,як показано на рисунку 3.3.

Програмний комплекс повинен бути працездатним на пристрої, що відповідає наступним вимогам:

— будь-яка операційна система у тому числі мобільна;

— не менше 256 Мб оперативної пам'яті;

— доступ до Інтернету зі швидкістю не менше 256 кб/с;

будь-який браузер.

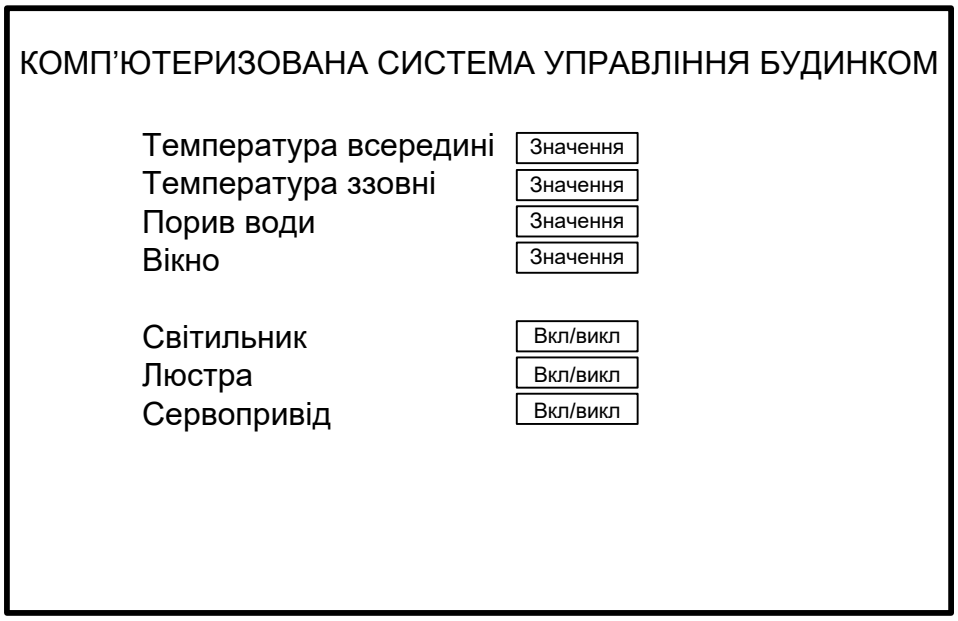


Рисунок 3.3 — Макет головної стариці web-ресурсу

Контрольною панеллю для поточної системи є стандартна HTML сторінка, запущена з допомогою Ethernet модуля ENC28J60. Доступ до панелі на даний момент може відбуватися, коли смартфон або комп'ютер знаходяться в одній мережі із системою автоматизації будинку. Щоб зайти на панель керування, необхідно набрати адресний рядок браузера 192.168.4.100/all. Панель керування відображає інформацію про температуру зовні та всередині кімнати, поточний стан вікна та інформацію про пориви. У разі критичних ситуацій на контрольній панелі відображається відповідна інформація. Крім інформації, на контрольній панелі представлені 3 кнопки ручного включення та відключення світильника, люстри та сервоприводу, що перекриває подачу води. Кнопка керування світильником не задіяна, так, як використовується 2-х модульне реле, але завдяки реле можна підключити будь-який інший прилад, при цьому не змінюючи програмну частину, що відповідає за дію при натисканні кнопки. Крім ручного керування світло може включатися і по бавовні, а при виявленні пориву сервопривід в автоматичному режимі перекриває подачу води.

Істотним недоліком такого типу панелі управління є відсутність повідомлень, тобто доводиться тримати веб-сторінку постійно відкритою. У моєму випадку використовується віджет веб-сторінки, розташований на робочому

столі смартфона та комп'ютера. Надалі планується створення окремої програми для смартфона, а також покращення графічного інтерфейсу панелі управління.

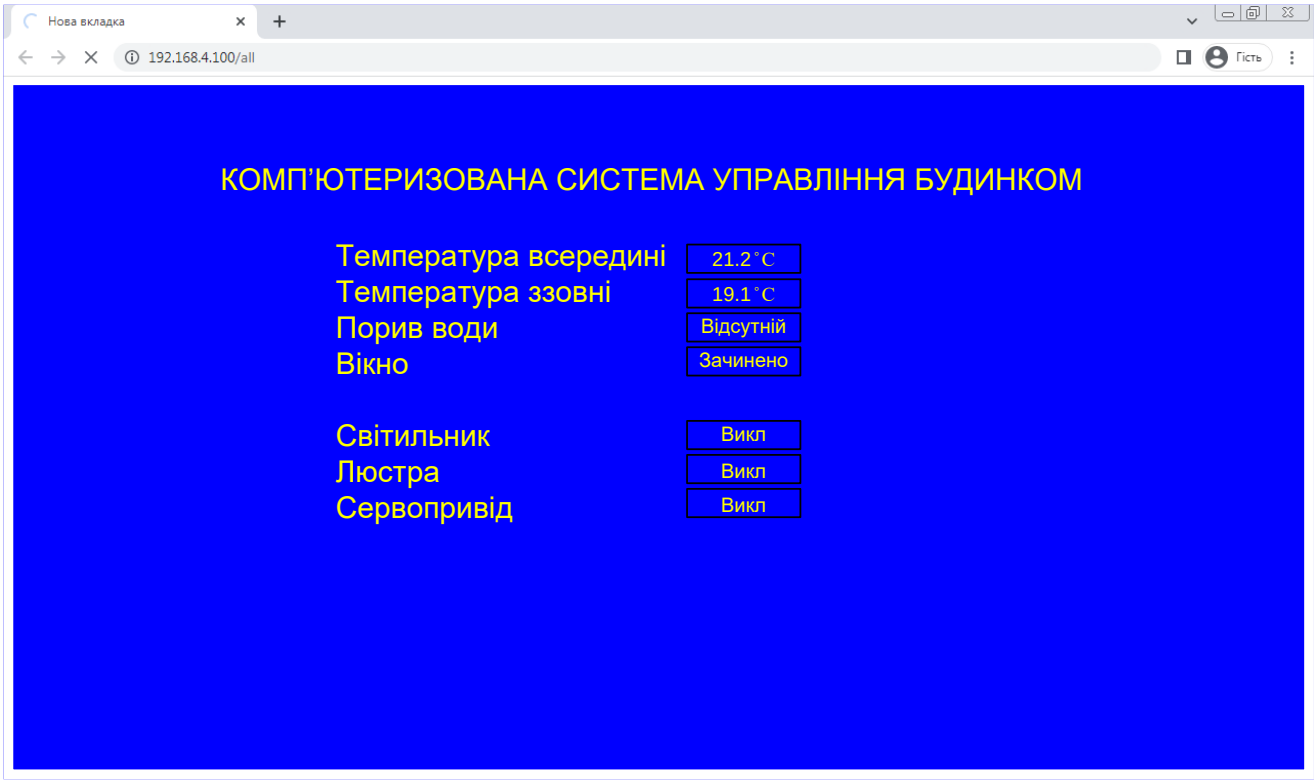


Рисунок 3.4 — Загальний вигляд панелі керування

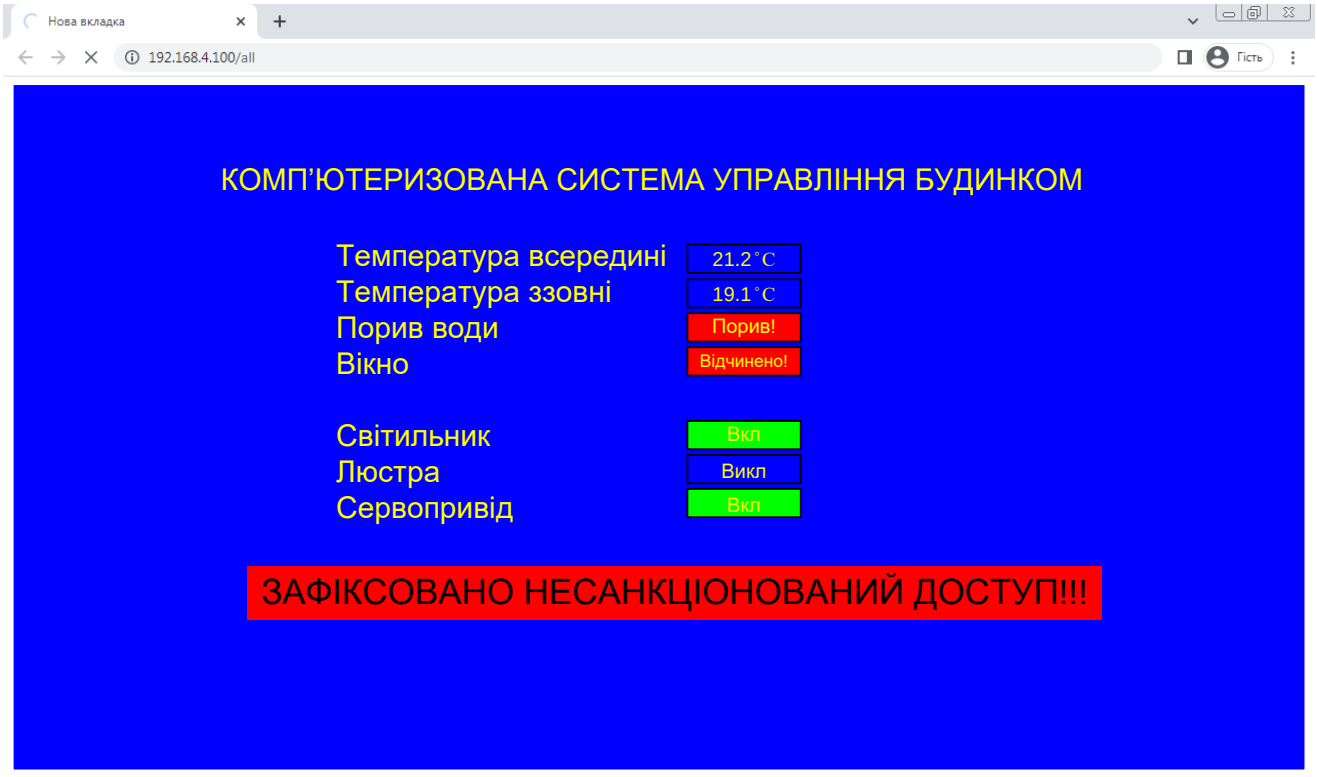


Рисунок 3.5 — Панель керування при виникненні порушень

ВИСНОВКИ

В рамках даної роботи було виконано розробку комп’ютеризованої системи управління будинком, яка дозволяє керувати електроприладами, контролювати доступ до приміщення, запобігати поривам води, а також надавати віддалений доступ до панелі управління.

Під час написання бакалаврської роботи було проведено аналіз ринку та технологій систем домашньої автоматизації. Сформовано основні цілі та завдання, які має виконувати поточна система. Зроблено вибір засобів розробки.

Результатом бакалаврської роботи є програмно-апаратний продукт — комп’ютеризована система управління будинку, має необхідний набір функціоналу на вирішення поставлених перед нею завдань. Програмна частина реалізована за допомогою середовища розробки Arduino IDE, а апаратна включає керуючу плату-мікроконтролер Ardunio Leonrdo, різні датчики та 3 додаткові пристрої – модуль Ethernet Enc28j60, сервопривід та реле.

Ця система є прототипом або основою по-справжньому «Розумного будинку». Вона має такі очевидні мінуси, як відсутність повноцінного графічного інтерфейсу контрольної панелі, система не надає повного контролю доступу до приміщення, не забезпечує пожежну безпеку та не повідомляє користувача про критичні ситуації. Система автоматизації будинку розроблялася для власної кімнати у приватному будинку та надалі модернізуватиметься в результаті не поступатиметься за якістю та функціоналом дорогим комерційним рішенням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Веб-сервер [Електронний ресурс] // Довідкова інформаційна система «Вікіпедія», 2016. – Режим доступу:https://ua.wikipedia.org/wiki/Веб-сервер

2. Геркон [Електронний ресурс] // Довідкова інформаційна система «Вікіпедія», 2016. – Режим доступу:https://ru.wikipedia.org/wiki/Геркон

3. Датчик руху [Електронний ресурс] // Довідкова інформаційна система "Вікіпедія", 2016. - Режим доступу:https://ua.wikipedia.org/wiki/Датчик_руху

4. Датчик шуму [Електронний ресурс] // Інформаційний ресурс «Амперка»,2015 року. - Режим доступу: http://wiki.amperka.ru/продукти:troyka:sound-loudness-sensor

5. Дубровін Д.А. РОЗРОБКА СИСТЕМИ "Розумний будинок": наукова стаття/Д.А. Дубровін - Москва: Фінансовий університет при Уряді Російської Федерації, 2015. - 5 с.

6. Як уберегти квартиру від затоплення? Огляд детекторів витоку води [Електронний ресурс] // Журнал споживчої електроніці Ferra. - Режим доступу: http://www.ferra.ru/ru/digihome/review/SmartHome-

7. Недоліки системи "Розумний дім" [Електронний ресурс] // Сонес. - Режим доступу:http://sones.ru/stati/nedostatki-sistemy-umnyi-dom.html

8. Основа для розумної кімнати, або як Arduino у гуртожитку живе [Електронний ресурс] // Geektimes, 2014 року. – Режим доступу: https://geektimes.ru/post/258598/

9. Піроелектричний інфрачервоний (PIR) датчик руху та Arduino [Електронний ресурс]// Arduino-diy. – Режим доступу: http://arduino-diy.com/arduino-piroelektricheskiy-infrakrasnyy-PIR-datchik-dvizheniya

10. Підключення мікроконтролера до локальної мережі: працюємо з ENC28J60[Електронний ресурс] // Спільнота Easyelectronics.– Режим доступу: http://we.easyelectronics.ru/electro-and-pc/podklyuchenie-mikrokontrollera-k-lokalnoy-seti-rabotaem-s-enc28j60.html

11. Підтягуючий резистор [Електронний ресурс] // Довідкова інформаційна система «Вікіпедія», 2015. – Режим доступу: https://ua.wikipedia.org/wiki/Підтягуючий_резистор

12. Протокол X10 [Електронний ресурс] // IQ home, 2015. – Режим доступу:http://www.iq-home.ru/tech/x10.html

13. Реле модуль підключення до Arduino [Електронний ресурс] // Співтовариство Zelectro, 2013. – Режим доступу:http://zelectro.cc/relayModule

14. Сервоприводи [Електронний ресурс] // Інформаційний ресурс «Амперка», 2013 року. – Режим доступу: http://wiki.amperka.ru/робототехніка:сервоприводи

15. Системи безпеки «розумного дому» [Електронний ресурс]// BIPA-АРТБУД. – Режим доступу:

http://www.eremont.ru/enc/engineer/clever/security-system-smart-home.html

16. Скільки коштує розумний будинок [Електронний ресурс] // Home Sapiens. -

Режим доступу:http://home-sapiens.ru/skolko-stoit-umnyiy-dom/

17. СТО 4.2-07-2014 Система менеджменту якості. Загальні вимоги до побудови, викладу та оформлення документів навчальної діяльності. - Введ. 30-12-2013. - Красноярськ: СФУ, 2014. - 60 с.

18. Тесля О.В. «Розумний будинок» своїми руками. Будуємо інтелектуальну цифрову систему у своїй квартирі: книга / Є.В. Тесля –Санкт-Петербург: ЛітРес, 2011. - 370 с.

19. Розумний будинок власноруч [Електронний ресурс] // Dom-electro. - Режим доступу: http://www.dom-electro.ru/розумний-дім-своїми-руками/

20. Що таке сервопривід [Електронний ресурс] // Спільнота Zelectro, 2013. – Режим доступу:http://zelectro.cc/what_is_servo/

ДОДАТОК А

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ ВНТУ

д.т.н., проф.

_____О. Д. Азаров

“__” _____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи

«Комп’ютеризована система управління будинком»
08-23.БДР.003.00.000 ТЗ

Науковий керівник к.т.н., доц. каф. ОТ

_____ Колесник І.С.

Студент групи 1КІ-18б

_____ Василевський Д.А.

Вінниця 2022

1. Найменування та область застосування

Комп’ютеризована система управління будинком. Системи із застосуванням сучасних досягнень в мікропроцесорній техніці для автоматизації та управління будинком.

2. Основи для розробки

Потреба в управлінні, контролі, автоматизації та захисті приватної будинкової власності.

3. Мета та призначення розробки

Розробка системи автоматизації будинку, яка надасть користувачеві нові можливості з управління та контролю його житла.

4. Етапи БДР та очікувані результати

Робота виконується в п’ять етапів, що наведені в таблиці 4.1.

Таблиця 4.1 – Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз завдання. Вступ	14.02.20	14.02.20	Вступ
2	Огляд і аналіз сучасного стану галузі	15.02.20	28.02.20	Розділ 1
3	Проектування системи управління будинком	01.03.20	14.03.20	Розділ 2
4	програмне забезпечення системи управління будинком	15.03.20	28.4.20	Розділ 3
5	Оформлення пояснювальної записки	29.04.20	17.05.20	ПЗ,презентація

5. Матеріали, що подаються до захисту БДР

Пояснювальна записка БДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відзив наукового керівника, рецензія

опонента, протоколи складання державних екзаменів, анотації до БДР українською та іноземною мовами, довідка про відповідність оформлення БДР діючим вимогам.

6. Порядок контролю виконання та захисту БДР

Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

7. Вимоги до оформлення БДР

Вимоги до БДР наведені нище:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Едина система конструкторської документації. Основні написи»;

— Документами на які посилаються у вище вказаних.

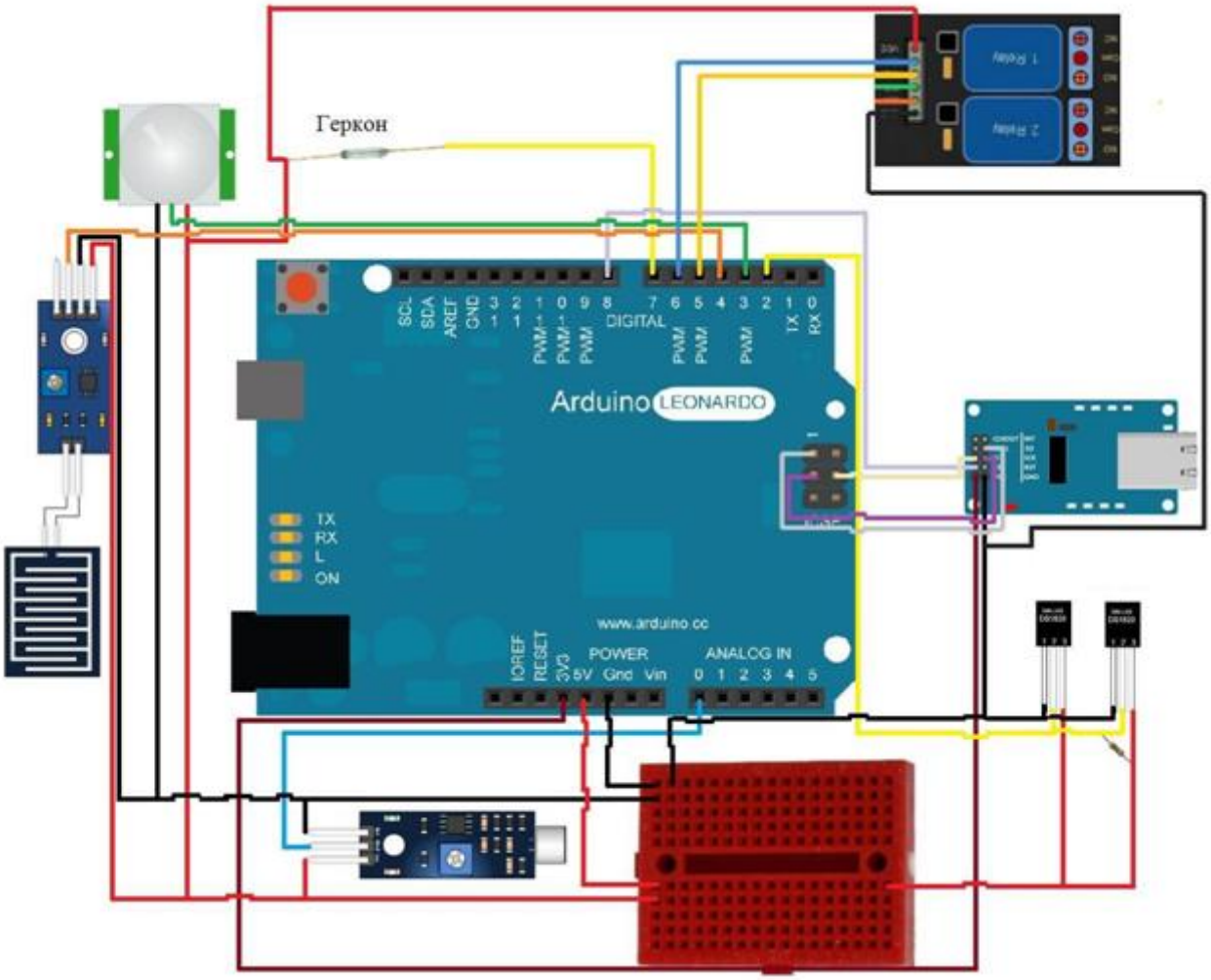
Вимоги щодо технічного захисту інформації в БДР з обмеженим доступом

Відсутні.

Технічне завдання до виконання отримав _____ Василевський Д.А.

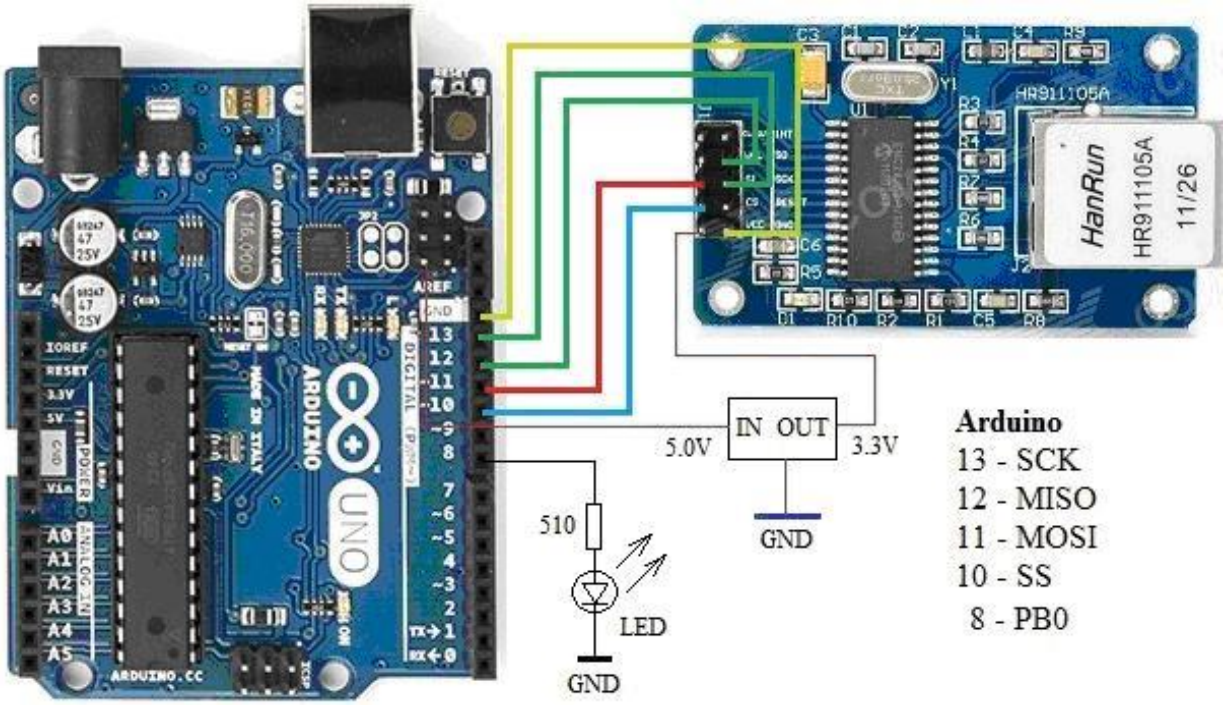
ДОДАТОК Б

Схема системи управління будинком



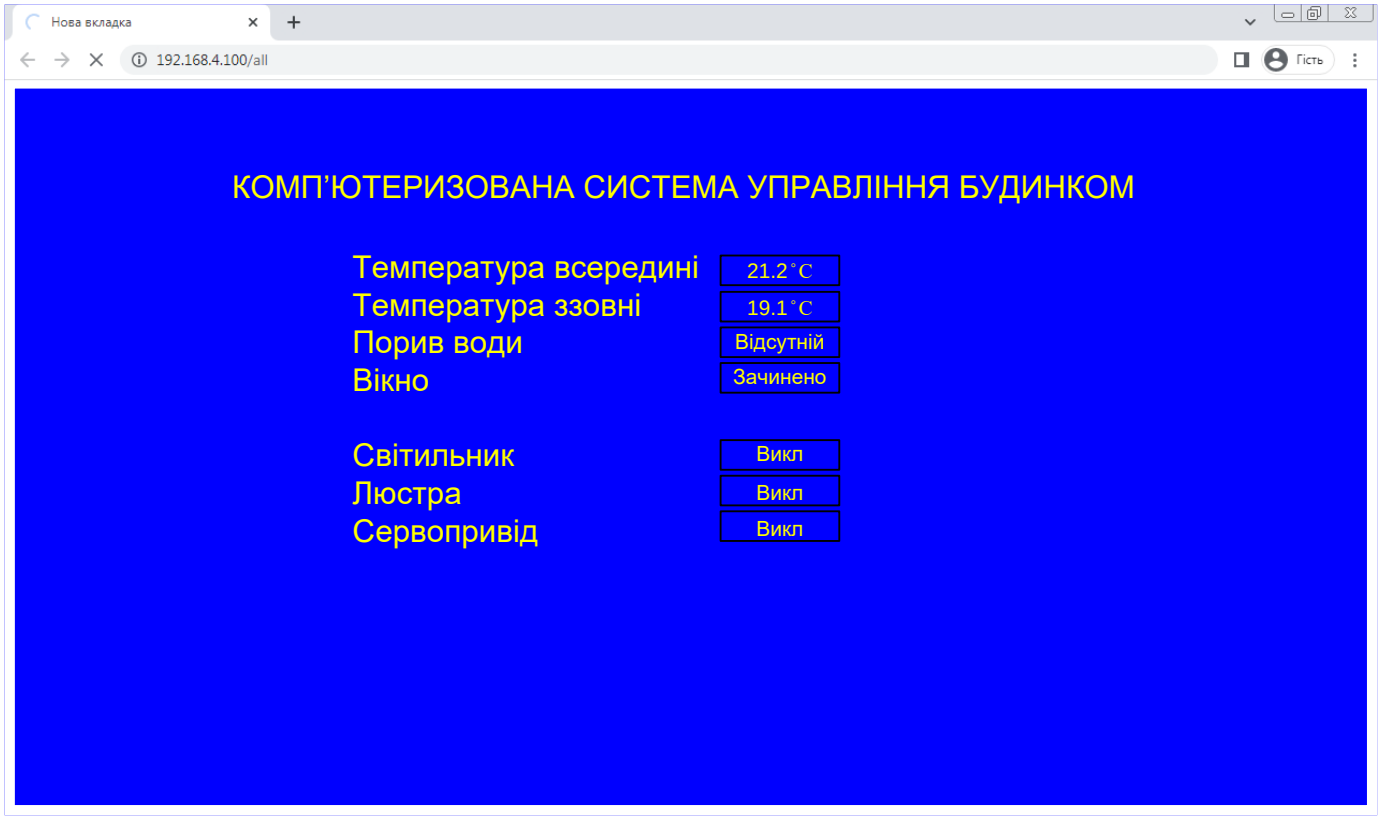
ДОДАТОК В

Схема підключення ENC28J60 до Arduino



ДОДАТОК Г

Загальний вигляд панелі керування



ДОДАТОК Д

Код бібліотеки EtherCard

```
#ifndef EtherCard_h
#define EtherCard_h
#ifndef __PROG_TYPES_COMPAT__
  #define __PROG_TYPES_COMPAT__
#endif

#if ARDUINO >= 100
#include <Arduino.h> // Arduino 1.0
#define WRITE_RESULT size_t
#define WRITE_RETURN return 1;
#else
#include <WProgram.h> // Arduino 0022
#define WRITE_RESULT void
#define WRITE_RETURN
#endif

#include <avr/pgmspace.h>
#include "bufferfiller.h"
#include "enc28j60.h"
#include "net.h"
#include "stash.h"

/** Enable DHCP.
*   Setting this to zero disables the use of DHCP; if a program uses DHCP it will
*   still compile but the program will not work. Saves about 60 bytes SRAM and
*   1550 bytes flash.
*/
#define ETHERCARD_DHCP 1

/** Enable client connections.
* Setting this to zero means that the program cannot issue TCP client requests
* anymore. Compilation will still work but the request will never be
* issued. Saves 4 bytes SRAM and 550 byte flash.
*/
#define ETHERCARD_TCPCLIENT 1
```

```
/** Enable TCP server functionality.
*   Setting this to zero means that the program will not accept TCP client
*   requests. Saves 2 bytes SRAM and 250 bytes flash.
*/
#define ETHERCARD_TCPSERVER 1

/** Enable UDP server functionality.
*   If zero UDP server is disabled. It is
*   still possible to register callbacks but these will never be called. Saves
*   about 40 bytes SRAM and 200 bytes flash. If building with -flto this does not
*   seem to save anything; maybe the linker is then smart enough to optimize the
*   call away.
*/
#define ETHERCARD_UDPSERVER 1

/** Enable automatic reply to pings.
*   Setting to zero means that the program will not automatically answer to
*   PINGs anymore. Also the callback that can be registered to answer incoming
*   pings will not be called. Saves 2 bytes SRAM and 230 bytes flash.
*/
#define ETHERCARD_ICMP 1

/** Enable use of stash.
*   Setting this to zero means that the stash mechanism cannot be used. Again
*   compilation will still work but the program may behave very unexpectedly.
*   Saves 30 bytes SRAM and 80 bytes flash.
*/
#define ETHERCARD_STASH 1


/** This type definition defines the structure of a UDP server event handler callback
function */
typedef void (*UdpServerCallback)(
    uint16_t dest_port,    ///< Port the packet was sent to
    uint8_t src_ip[IP_LEN],    ///< IP address of the sender
    uint16_t src_port,    ///< Port the packet was sent from
    const char *data,   ///< UDP payload data
    uint16_t len);        ///< Length of the payload data
```

```
/** This type definition defines the structure of a DHCP Option callback function */
typedef void (*DhcpOptionCallback)(
    uint8_t option,     ///< The option number
    const byte* data,   ///< DHCP option data
    uint8_t len);       ///< Length of the DHCP option data



/** This class provides the main interface to a ENC28J60 based network interface card and
is the class most users will use.
*   @note   All TCP/IP client (outgoing) connections are made from source port in range
2816-3071. Do not use these source ports for other purposes.
*/
class EtherCard : public Ethernet {
public:
    static uint8_t mymac[ETH_LEN];  ///< MAC address
    static uint8_t myip[IP_LEN];    ///< IP address
    static uint8_t netmask[IP_LEN]; ///< Netmask
    static uint8_t broadcastip[IP_LEN]; ///< Subnet broadcast address
    static uint8_t gwip[IP_LEN];   ///< Gateway
    static uint8_t dhcpip[IP_LEN]; ///< DHCP server IP address
    static uint8_t dnsip[IP_LEN];  ///< DNS server IP address
    static uint8_t hisip[IP_LEN];  ///< DNS lookup result
    static uint16_t hisport;  ///< TCP port to connect to (default 80)
    static bool using_dhcp;   ///< True if using DHCP
    static bool persist_tcp_connection; ///< False to break connections on first packet
received
    static uint16_t delaycnt; ///< Counts number of cycles of packetLoop when no packet
received - used to trigger periodic gateway ARP request

    // EtherCard.cpp
    /**   @brief  Initialise the network interface
    *     @param  size Size of data buffer
    *     @param  macaddr Hardware address to assign to the network interface (6 bytes)
    *     @param  csPin Arduino pin number connected to chip select. Default = 8
    *     @return <i>uint8_t</i> Firmware version or zero on failure.
    */
    static uint8_t begin (const uint16_t size, const uint8_t* macaddr,
                          uint8_t csPin = SS);
```

```
    /**   @brief  Configure network interface with static IP
    *     @param  my_ip IP address (4 bytes). 0 for no change.
    *     @param  gw_ip Gateway address (4 bytes). 0 for no change. Default = 0
    *     @param  dns_ip DNS address (4 bytes). 0 for no change. Default = 0
    *     @param  mask Subnet mask (4 bytes). 0 for no change. Default = 0
    *     @return <i>bool</i> Returns true on success - actually always true
    */
    static bool staticSetup (const uint8_t* my_ip,
                             const uint8_t* gw_ip = 0,
                             const uint8_t* dns_ip = 0,
                             const uint8_t* mask = 0);

    // tcpip.cpp
    /**   @brief  Sends a UDP packet to the IP address of last processed received packet
    *     @param  data Pointer to data payload
    *     @param  len Size of data payload (max 220)
    *     @param  port Source IP port
    */
    static void makeUdpReply (const char *data, uint8_t len, uint16_t port);

    /**   @brief  Parse received data
    *     @param  plen Size of data to parse (e.g. return value of packetReceive()).
    *     @return <i>uint16_t</i> Offset of TCP payload data in data buffer or zero if packet
processed
    *     @note   Data buffer is shared by receive and transmit functions
    *     @note   Only handles ARP and IP
    */
    static uint16_t packetLoop (uint16_t plen);

    /**   @brief  Accept a TCP/IP connection
    *     @param  port IP port to accept on - do nothing if wrong port
    *     @param  plen Number of bytes in packet
    *     @return <i>uint16_t</i> Offset within packet of TCP payload. Zero for no data.
    */
    static uint16_t accept (uint16_t port, uint16_t plen);

    /**   @brief  Send a response to a HTTP request
    *     @param  dlen Size of the HTTP (TCP) payload
    */
    static void httpServerReply (uint16_t dlen);
```

```
/**   @brief  Send a response to a HTTP request
*     @param  dlen Size of the HTTP (TCP) payload
*     @param  flags TCP flags
*/
static void httpServerReply_with_flags (uint16_t dlen , uint8_t flags);

/**   @brief  Acknowledge TCP message
*     @todo   Is this / should this be private?
*/
static void httpServerReplyAck ();

/**   @brief  Set the gateway address
*     @param  gwipaddr Gateway address (4 bytes)
*/
static void setGwIp (const uint8_t *gwipaddr);

/**   @brief  Updates the broadcast address based on current IP address and subnet mask
*/
static void updateBroadcastAddress();

/**   @brief  Check if got gateway hardware address (ARP lookup)
*     @return <i>unit8_t</i> True if gateway found
*/
static uint8_t clientWaitingGw ();

/**   @brief  Check if got gateway DNS address (ARP lookup)
*     @return <i>unit8_t</i> True if DNS found
*/
static uint8_t clientWaitingDns ();

/**   @brief  Prepare a TCP request
*     @param  result_cb Pointer to callback function that handles TCP result
*     @param  datafill_cb Pointer to callback function that handles TCP data payload
*     @param  port Remote TCP/IP port to connect to
*     @return <i>unit8_t</i> ID of TCP/IP session (0-7)
*     @note   Return value provides id of the request to allow up to 7 concurrent requests
*/
static uint8_t clientTcpReq (uint8_t (*result_cb)(uint8_t,uint8_t,uint16_t,uint16_t),
                             uint16_t (*datafill_cb)(uint8_t),uint16_t port);
```

```
/**   @brief  Prepare HTTP request
*     @param  urlbuf Pointer to c-string URL folder
*     @param  urlbuf_varpart Pointer to c-string URL file
*     @param  hoststr Pointer to c-string hostname
*     @param  additionalheaderline Pointer to c-string with additional HTTP header info
*     @param  callback Pointer to callback function to handle response
*     @note   Request sent in main packetloop
*/
static void browseUrl (const char *urlbuf, const char *urlbuf_varpart,
                       const char *hoststr, const char *additionalheaderline,
                       void (*callback)(uint8_t,uint16_t,uint16_t));

/**   @brief  Prepare HTTP request
*     @param  urlbuf Pointer to c-string URL folder
*     @param  urlbuf_varpart Pointer to c-string URL file
*     @param  hoststr Pointer to c-string hostname
*     @param  callback Pointer to callback function to handle response
*     @note   Request sent in main packetloop
*/
static void browseUrl (const char *urlbuf, const char *urlbuf_varpart,
                       const char *hoststr,
                       void (*callback)(uint8_t,uint16_t,uint16_t));

/**   @brief  Prepare HTTP post message
*     @param  urlbuf Pointer to c-string URL folder
*     @param  hoststr Pointer to c-string hostname
*     @param  additionalheaderline Pointer to c-string with additional HTTP header info
*     @param  postval Pointer to c-string HTML Post value
*     @param  callback Pointer to callback function to handle response
*     @note   Request sent in main packetloop
*/
static void httpPost (const char *urlbuf, const char *hoststr,
                      const char *additionalheaderline, const char *postval,
                      void (*callback)(uint8_t,uint16_t,uint16_t));

/**   @brief  Send NTP request
*     @param  ntpip IP address of NTP server
*     @param  srcport IP port to send from
*/
```

```
   static void ntpRequest (uint8_t *ntpip,uint8_t srcport);

   /**   @brief  Process network time protocol response
   *     @param  time Pointer to integer to hold result
   *     @param  dstport_l Destination port to expect response. Set to zero to accept on any
port
   *     @return <i>uint8_t</i> True (1) on success
   */
   static uint8_t ntpProcessAnswer (uint32_t *time, uint8_t dstport_l);

   /**   @brief  Prepare a UDP message for transmission
   *     @param  sport Source port
   *     @param  dip Pointer to 4 byte destination IP address
   *     @param  dport Destination port
   */
   static void udpPrepare (uint16_t sport, const uint8_t *dip, uint16_t dport);

   /**   @brief  Transmit UDP packet
   *     @param  len Size of payload
   */
   static void udpTransmit (uint16_t len);

   /**   @brief  Sends a UDP packet
   *     @param  data Pointer to data
   *     @param  len Size of payload (maximum 220 octets / bytes)
   *     @param  sport Source port
   *     @param  dip Pointer to 4 byte destination IP address
   *     @param  dport Destination port
   */
   static void sendUdp (const char *data, uint8_t len, uint16_t sport,
                        const uint8_t *dip, uint16_t dport);

   /**   @brief  Resister the function to handle ping events
   *     @param  cb Pointer to function
   */
   static void registerPingCallback (void (*cb)(uint8_t*));

   /**   @brief  Send ping
   *     @param  destip Pointer to 4 byte destination IP address
   */
```

```
    static void clientIcmpRequest (const uint8_t *destip);

    /**   @brief  Check for ping response
    *     @param  ip_monitoredhost Pointer to 4 byte IP address of host to check
    *     @return <i>uint8_t</i> True (1) if ping response from specified host
    */
    static uint8_t packetLoopIcmpCheckReply (const uint8_t *ip_monitoredhost);

    /**   @brief  Send a wake on lan message
    *     @param  wolmac Pointer to 6 byte hardware (MAC) address of host to send
message to
    */
    static void sendWol (uint8_t *wolmac);

    // new stash-based API
    /**   @brief  Send TCP request
    */
    static uint8_t tcpSend ();

    /**   @brief  Get TCP reply
    *     @return <i>char*</i> Pointer to TCP reply payload. NULL if no data.
    */
    static const char* tcpReply (uint8_t fd);

    /**   @brief  Configure TCP connections to be persistent or not
    *     @param  persist True to maintain TCP connection. False to finish TCP connection
after first packet.
    */
    static void persistTcpConnection(bool persist);

    //udpserver.cpp
    /**   @brief  Register function to handle incoming UDP events
    *     @param  callback Function to handle event
    *     @param  port Port to listen on
    */
    static void udpServerListenOnPort(UdpServerCallback callback, uint16_t port);

    /**   @brief  Pause listing on UDP port
    *     @brief  port Port to pause
    */
```

```
    static void udpServerPauseListenOnPort(uint16_t port);

    /**   @brief  Resume listing on UDP port
    *     @brief  port Port to pause
    */
    static void udpServerResumeListenOnPort(uint16_t port);

    /**   @brief  Check if UDP server is listening on any ports
    *     @return <i>bool</i> True if listening on any ports
    */
    static bool udpServerListening();                    //called by tcpip, in packetLoop

    /**   @brief  Passes packet to UDP Server
    *     @param  len Not used
    *     @return <i>bool</i> True if packet processed
    */
    static bool udpServerHasProcessedPacket(uint16_t len);    //called by tcpip, in
packetLoop

    // dhcp.cpp
    /**   @brief  Update DHCP state
    *     @param  len Length of received data packet
    */
    static void DhcpStateMachine(uint16_t len);

    /**   @brief  Configure network interface with DHCP
    *     @param  hname The hostname to pass to the DHCP server
    *     @param  fromRam Set true to indicate whether hname is in RAM or in program
space. Default = false
    *     @return <i>bool</i> True if DHCP successful
    *     @note   Blocks until DHCP complete or timeout after 60 seconds
    */
    static bool dhcpSetup (const char *hname = NULL, bool fromRam =false);

    /**   @brief  Register a callback for a specific DHCP option number
    *     @param  option The option number to request from the DHCP server
    *     @param  callback The function to be call when the option is received
    */
    static void dhcpAddOptionCallback(uint8_t option, DhcpOptionCallback callback);
```

```
   /**   @brief  Register a callback for multiple DHCP option numbers
   *     @param  optionlist pointer to null terminate list of DHCP option numbers (must be
static)
   *     @param  callback The function to be call when the option is received
   */
   static void dhcpAddOptionCallback(uint8_t* optionlist, DhcpOptionCallback callback);

   // dns.cpp
   /**   @brief  Perform DNS lookup
   *     @param  name Host name to lookup
   *     @param  fromRam Set true to indicate whether name is in RAM or in program
space. Default = false
   *     @return <i>bool</i> True on success.
   *     @note   Result is stored in <i>hisip</i> member
   */
   static bool dnsLookup (const char* name, bool fromRam =false);

   // webutil.cpp
   /**   @brief  Copies an IP address
   *     @param  dst Pointer to the 4 byte destination
   *     @param  src Pointer to the 4 byte source
   *     @note   There is no check of source or destination size. Ensure both are 4 bytes
   */
   static void copyIp (uint8_t *dst, const uint8_t *src);

   /**   @brief  Copies a hardware address
   *     @param  dst Pointer to the 6 byte destination
   *     @param  src Pointer to the 6 byte destination
   *     @note   There is no check of source or destination size. Ensure both are 6 bytes
   */
   static void copyMac (uint8_t *dst, const uint8_t *src);

   /**   @brief  Output to serial port in dotted decimal IP format
   *     @param  buf Pointer to 4 byte IP address
   *     @note   There is no check of source or destination size. Ensure both are 4 bytes
   */
   static void printIp (const uint8_t *buf);

   /**   @brief  Output message and IP address to serial port in dotted decimal IP format
   *     @param  msg Pointer to null terminated string
```

```
    *     @param  buf Pointer to 4 byte IP address
    *     @note   There is no check of source or destination size. Ensure both are 4 bytes
    */
    static void printIp (const char* msg, const uint8_t *buf);

    /**   @brief  Output Flash String Helper and IP address to serial port in dotted decimal
IP format
    *     @param  ifsh Pointer to Flash String Helper
    *     @param  buf Pointer to 4 byte IP address
    *     @note   There is no check of source or destination size. Ensure both are 4 bytes
    *     @todo   What is a FlashStringHelper?
    */
    static void printIp (const __FlashStringHelper *ifsh, const uint8_t *buf);

    /**   @brief  Search for a string of the form key=value in a string that looks like
q?xyz=abc&uvw=defgh HTTP/1.1\\r\\n
    *     @param  str Pointer to the null terminated string to search
    *     @param  strbuf Pointer to buffer to hold null terminated result string
    *     @param  maxlen Maximum length of result
    *     @param  key Pointer to null terminated string holding the key to search for
    *     @return <i>unit_t</i> Length of the value. 0 if not found
    *     @note   Ensure strbuf has memory allocated of at least maxlen + 1 (to accommodate
result plus terminating null)
    */
    static uint8_t findKeyVal(const char *str,char *strbuf,
                        uint8_t maxlen, const char *key);

    /**   @brief  Decode a URL string e.g "hello%20joe" or "hello+joe" becomes "hello
joe"
    *     @param  urlbuf Pointer to the null terminated URL
    *     @note   urlbuf is modified
    */
    static void urlDecode(char *urlbuf);

    /**   @brief  Encode a URL, replacing illegal characters like ' '
    *     @param  str Pointer to the null terminated string to encode
    *     @param  urlbuf Pointer to a buffer to contain the null terminated encoded URL
    *     @note   There must be enough space in urlbuf. In the worst case that is 3 times the
length of str
    */
```

```
    static  void urlEncode(char *str,char *urlbuf);

    /**   @brief  Convert an IP address from dotted decimal formated string to 4 bytes
    *     @param  bytestr Pointer to the 4 byte array to store IP address
    *     @param  str Pointer to string to parse
    *     @return <i>uint8_t</i> 0 on success
    */
    static uint8_t parseIp(uint8_t *bytestr, const char *str);

    /**   @brief  Convert a byte array to a human readable display string
    *     @param  resultstr Pointer to a buffer to hold the resulting null terminated string
    *     @param  bytestr Pointer to the byte array containing the address to convert
    *     @param  len Length of the array (4 for IP address, 6 for hardware (MAC) address)
    *     @param  separator Delimiter character (typically '.' for IP address and ':' for
hardware (MAC) address)
    *     @param  base Base for numerical representation (typically 10 for IP address and 16
for hardware (MAC) address
    */
    static void makeNetStr(char *resultstr,uint8_t *bytestr,uint8_t len,
                           char separator,uint8_t base);

    /**   @brief  Convert a 16-bit integer into a string
    *     @param  value The number to convert
    *     @param  ptr The string location to write to
    */
    char* wtoa(uint16_t value, char* ptr);

    /**   @brief  Return the sequence number of the current TCP package
    */
    static uint32_t getSequenceNumber();

    /**   @brief  Return the payload length of the current Tcp package
    */
    static uint16_t getTcpPayloadLength();
};

extern EtherCard ether; //!< Global presentation of EtherCard class

#endif
```

ДОДАТОК Е

Код бібліотеки EtherEncLib

```
#include <Arduino.h>
#include "TcpStack.h"

#ifndef ETHERENCLIB_H
#define ETHERENCLIB_H

// by Renato Aloi May 2015
// Trying to implement SocketTX avoiding RAM buffers
// New rule of 3 buffers
// 1. Buffer for incomming TCP Headers (58 bytes) -- TcpStack->m_tcpData[]
// 2. Buffer for outcomming TCP Headers (58 bytes) -- TcpStack->m_sendData[]
// 3. Buffer for Http parameters (50 bytes) -- EtherEncLib->m_httpData[]
// Total RAM from Buffers: 166 bytes + 20 bytes = 186 bytes (from 2x ip and 2x mac addresses: (2 * 4 bytes) + (2 * 6 bytes))

#define BUFFER_PARAMS_LEN  50 // buffering only parameters and tcp header to conserve RAM
                              // TCP Headers Buffers = 116 bytes + Params Buffer = 50
                              // Total = 166 bytes total RAM consumption
                              // by Renato Aloi (May 2015)
#define DEBUGLIB          0

class EtherEncLib
{

  private:
        unsigned int m_port;
        TcpStack m_stack;
        char  m_httpData[BUFFER_PARAMS_LEN];

        int freeRam () {
          extern int __heap_start, *__brkval;
          int v;
          return (int) &v - (__brkval == 0 ? (int) &__heap_start : (int) __brkval);
        };
        unsigned char analize    (void); // by SKA
```

```
  public:
    EtherEncLib(unsigned int port) : m_port(port), m_stack(TcpStack()) {
                for (unsigned i = 0; i < BUFFER_PARAMS_LEN; i++)  m_httpData[i] = 0;
        };

    void         begin      (unsigned char *ip, unsigned char *mac);
    unsigned char available  (void);
    void         print      (char c);
    void         print      (char *rd);
    void         print      (unsigned int val);
    void         print      (int val);
    void          print      (char *, unsigned char); // made by SKA
    void         close      (void);
    char          read       (void);
    char         *getParams (void);
//--- made by SKA ---

    unsigned char isGet;
    unsigned char isPost;
    unsigned char isIndexHtml;

//--- made by SKA ---
};

#endif
```

ДОДАТОК Є

Код бібліотеки UIPEthernet

```
#ifndef UIPETHERNET_H
#define UIPETHERNET_H

//#define UIPETHERNET_DEBUG
//#define UIPETHERNET_DEBUG_CHKSUM
//#define UIPETHERNET_DEBUG_UDP
//#define UIPETHERNET_DEBUG_CLIENT

#include "ethernet_comp.h"
#include <Arduino.h>
#include "Dhcp.h"
#include "IPAddress.h"
#include "utility/Enc28J60Network.h"
#include "UIPClient.h"
#include "UIPServer.h"
#include "UIPUdp.h"

extern "C"
{
#include "utility/uip_timer.h"
#include "utility/uip.h"
}

#define UIPETHERNET_FREEPACKET 1
#define UIPETHERNET_SENDPACKET 2
#define UIPETHERNET_BUFFERREAD 4

#define uip_ip_addr(addr, ip) do { \
                  ((u16_t *)(addr))[0] = HTONS(((ip[0]) << 8) | (ip[1])); \
                  ((u16_t *)(addr))[1] = HTONS(((ip[2]) << 8) | (ip[3])); \
                } while(0)

#define ip_addr_uip(a) IPAddress(a[0] & 0xFF, a[0] >> 8 , a[1] & 0xFF, a[1] >> 8)
//TODO this is not IPV6 capable

#define uip_seteth_addr(eaddr) do {uip_ethaddr.addr[0] = eaddr[0]; \
                        uip_ethaddr.addr[1] = eaddr[1];\
```

```
                    uip_ethaddr.addr[2] = eaddr[2];\
                    uip_ethaddr.addr[3] = eaddr[3];\
                    uip_ethaddr.addr[4] = eaddr[4];\
                    uip_ethaddr.addr[5] = eaddr[5];} while(0)

#define BUF ((struct uip_tcpip_hdr *)&uip_buf[UIP_LLH_LEN])

class UIPEthernetClass
{
public:
  UIPEthernetClass();

  int begin(const uint8_t* mac);
  void begin(const uint8_t* mac, IPAddress ip);
  void begin(const uint8_t* mac, IPAddress ip, IPAddress dns);
  void begin(const uint8_t* mac, IPAddress ip, IPAddress dns, IPAddress gateway);
  void begin(const uint8_t* mac, IPAddress ip, IPAddress dns, IPAddress gateway,
IPAddress subnet);

  // maintain() must be called at regular intervals to process the incoming serial
  // data and issue IP events to the sketch.  It does not return until all IP
  // events have been processed. Renews dhcp-lease if required.
  int maintain();

  IPAddress localIP();
  IPAddress subnetMask();
  IPAddress gatewayIP();
  IPAddress dnsServerIP();

private:
  static memhandle in_packet;
  static memhandle uip_packet;
  static uint8_t uip_hdrlen;
  static uint8_t packetstate;

  static IPAddress _dnsServerAddress;
  static DhcpClass* _dhcp;

  static unsigned long periodic_timer;
```

```
  static void init(const uint8_t* mac);
  static void configure(IPAddress ip, IPAddress dns, IPAddress gateway, IPAddress
subnet);

  static void tick();

  static boolean network_send();

  friend class UIPServer;

  friend class UIPClient;

  friend class UIPUDP;

  static uint16_t chksum(uint16_t sum, const uint8_t* data, uint16_t len);
  static uint16_t ipchksum(void);
#if UIP_UDP
  static uint16_t upper_layer_chksum(uint8_t proto);
#endif
  friend uint16_t uip_ipchksum(void);
  friend uint16_t uip_tcpchksum(void);
  friend uint16_t uip_udpchksum(void);

  friend void uipclient_appcall(void);
  friend void uipudp_appcall(void);

#if UIP_CONF_IPV6
  uint16_t uip_icmp6chksum(void);
#endif /* UIP_CONF_IPV6 */
};

extern UIPEthernetClass UIPEthernet;

#endif
```

ДОДАТОК Ж
ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА
НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Комп'ютеризована система управління будинком

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 69% Схожість 31%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Василевський Д. А.
(підпис)

Керівник роботи _____ Колесник І. С.
(підпис)