НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Приладобудівний факультет

Кафедра приладів і систем орієнтації і навігації

До захисту допущено:

Завідувач кафедри

_____ Надія БУРАУ

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерно - інтегровані технології та системи навігації і керування»

спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

на тему: «Квест-кімната**»**

Виконав:
студент ІІІ курсу, групи ПГ-п81
Скиданчук Андрій Юрійович _____

Керівник:

асистент, к.т.н., Рупіч Сергій Сергійович _____

Рецензент:
асистент кафедри ПБ Н.М. Назаренко _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут
імені Ігоря Сікорського»

Інститут (факультет)_____Приладобудівний _____

(повна назва)

Кафедра_____Приладів і систем орієнтації і навігації_____

(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність __151 – Автоматизація та комп'ютерно-інтегровані технології

(код і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ __ Н.І. Бурау_

(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

_____Скиданчуку Андрію Юрійовичу _____

(прізвище, ім'я, по батькові)

1. Тема дипломної роботи_____Квест-кімната_

Науковий керівник роботи___Рупіч Сергій Сергійович, к.т.н._____ ,

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

затверджені наказом по університету від «__» _____ 201_ р. № _____

2. Термін подання студентом дисертації «08» червня 2021 р.

3. Вихідні дані до роботи Плата Arduino Nano, потенціометр, джойстик, серводвигуни, світлодіоди, макетна плата, дисплей TM74HC595, кнопки, дроти.

4. Зміст роботи Вступ 1. Огляд технології розумного будинку. 2. Загальна характеристика процесу розробки мікропроцесорних систем. 3. Розробка трьох завдань для проекту. 4. Розробка макету квест-кімнати. Висновки

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)
Схема структурна (А2), Схема підключення (А2), Корпус (А2), Схема принципова (А2), Алгоритм програми першого завдання (А3), Алгоритм програми другого завдання (А3), Алгоритм програми третього завдання (А2)

6. Консультанти розділів роботи*

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

7. Дата видачі завдання 16.04.2021

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд літератури з напрямку проектування та протипування систем по типу розумного дому	01.05.20	
2.	Розробка завдань для квест-кімнати та закупівля необхідного обладнання	10.05.20	
3.	Програмування компонентів системи	29.05.20	
4.	Оформлення графічних документів та збирання макету	04.06.20	
5.	Оформлення пояснювальної записки. Підготовка до захисту	08.06.20	

Студент _____ А. Ю. Скиданчук
(підпис) (ініціали, прізвище)

Керівник дипломної роботи _____ С. С. Рупіч
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломної роботи.

АНОТАЦІЯ

Дипломну роботу виконано на 51 аркуші, вона містить перелік посилань на використані джерела з 6 найменувань. У роботі наведено 20 рисунків.

Метою даної дипломної роботи є розробка та виготовлення масштабованої квест на базі мікроконтроллера Arduino.

У дипломній роботі розглядається актуальність електрично-програмованих систем по типу розумного будинку та прототипування таких розробок. Зазвичай такі системи основані на використанні мікропроцесорної техніки з керуючим мікропроцесором. Було наведено приклади прототипування різних систем. На основі даного підходу було поставлено завдання розробити квест-кімнати, де потрібно виконати три різні квести. Квести представляють собою взаємодію між користувачем та різними типами датчиків і пристроїв, які підключені до керуючої плати. В якості керуючого елемента було обрано Arduino Nano.

У роботі наведено теоретичні відомості та технічні характеристики про Arduino Nano та периферійні пристрої, що були використані для макету проєкту, а також проведено огляд основних етапів розробки програмної та апаратної частини.

Було побудовано макет квест кімнати відповідно до завдання та креслень, перевірено його працездатність.

SUMMARY

Thesis is performed on 51 sheets, it contains a list of references to the sources used from 6 items. The paper contains 20 figures.

The purpose of this thesis is to develop and manufacture a scalable quest based on the Arduino microcontroller.

The thesis considers the relevance of electrically programmable systems such as a smart home and prototyping of such developments. Typically, such systems are based on the use of microprocessor technology with a control microprocessor. Examples of prototyping of various systems were given. Based on this approach, the task was to develop quest rooms where you need to complete three different quests. Quests are the interaction between the user and the different types of sensors and devices that are connected to the control board. The Arduino Nano was chosen as the control.

The paper provides theoretical information and technical characteristics of the Arduino Nano and peripherals that were used for the layout of the project, as well as an overview of the main stages of software and hardware development.

The layout of the quest room was built in accordance with the task and drawings, its efficiency was checked.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ та СКОРОЧЕНЬ

АЦП	Аналого-цифровий перетворювач
МК	Мікроконтроллер
МПС	Мікропропроцесорна система
МПТ	Мікропроцесорна техніка
ОЗП	Оперативно запам'ятовуючий пристрій
ПЗП	Постійно запам'ятовуючий пристрій
ТЗ	Технічне завдання
ЦП	Центральний процесор
ШІМ	Широтно імпульсна модуляція
ICSP	In Circuit Serial Programming
RX	Receive pin
TX	Transmit pin
USB	Universal Serial Bus

ВСТУП

На теперешній час спостерігається зростання та збільшення кількості інновації у сфері інформаційних технологій. Те, що було футуристикою декілька років тому, зараз є звичайною справою. Темп зростання обсягів даних та обчислювальних потужностей надзвичайно стрімкий. Сучасні мобільні пристрої зазвичай мають технічні характеристики як у ноутбуків або персональних комп'ютерів (ПК), випущених п'ять років тому, а їхня собівартість набагато менша та дешевшає з кожним роком. За такого рівня технічного прогресу у більшості звичайних побутових пристроїв вже є власний процесор, що керує ним та дозволяє автоматизувати роботу. В результаті таких досягнень виникає питання, а чи можливо створити єдину систему контролю пристроїв, що нас оточують? Чи можна автоматизувати більшу частину побутових процесів? Відповіддю на ці питання стали розробки системи розумного будинку.

Багато відомих та не дуже компаній займаються розробкою та створенням таких систем. Основними перевагами системи розумного будинку є можливість автоматизації роботи пристроїв у домі. Це дозволяє автоматично контролювати параметри мікроклімату у приміщенні, вмикати та вимикати світло, побутову техніку та різні мультимедійні пристрої. Також корисною функцією даної системи є можливості енергозбереження. Коли ця система контролює охолоджувальні та нагрівальні установки, то вона здатна ефективно витрачати енергію, адже очевидним є той факт, що немає ніякої необхідності підтримувати комфортну температуру у приміщені за умови відсутності господаря [1].

При всіх позитивних моментах використання системи розумного будинку вона має та недоліки. Основним з них є вартість готової системи, іншим недоліком є фактична відсутність стандартизації на даний момент. Кожен з виробників реалізує концепцію розумного будинку виключно у своєму баченні. Він створює свою специфікацію, стандартизацію та модулі, які зазвичай не будуть працювати в системі від іншого виробника. Також проблемою готових рішень під ключ є

неможливість модернізації їх користувачами власноруч без достатнього рівня кваліфікації або технічної освіти.

Концепція «розумного» будинку полягає у використанні мікропроцесорної техніки, об’єднаних в єдину систему. Такий підхід за останні роки поширюється на інші прикладні завдання, такі як автоматизація освітлення/опалення, синхронізація пристроїв однієї еко системи, системи безпеки та їх взаємодія між собою.

Зазвичай, перед впровадженням та реалізацією мікропроцесорні системи використовується попереднє прототипування. Тобто, будується або створюється макет, за допомогою якого є можливість комплексно відпрацювати всі можливі недоліки. Прототип складається з подібних до реальних систем елементів та компонентів.

Наявність великої кількості пристроїв та датчиків забезпечує велику кількість сфер застосування, надає можливість зробити універсальний прилад, який може спростити певний процес. На сьогоднішній день є можливість виконувати розробки подібних систем навіть в домашніх умовах. Саме тому виникає ідея створення прототипу квест-кімнати за принципами «розумного» будинку. Метою дипломного проєкту є реалізація ігрового макету «Квест-кімната».

РОЗДІЛ 1

ОГЛЯД СТАНУ ПРОБЛЕМИ

1.1 Огляд технології розумний будинок

Розумний будинок - це сучасна будівля, всі комунікації в якій поєднуються за допомогою високотехнологічних пристроїв, щоб це було зручно для людини. Побутова техніка в розумному будинку об'єднана в універсальну домашню мережу, до якої можна отримати доступ в загальнодоступній мережі. Це поєднання багатьох систем різних виробників в єдиний керований комплекс та є головною особливістю такого будинку. Сигнали від датчиків, встановлених в кожній кімнаті, направляються на центральний комп'ютер, який обробляє отримані сигнали і, в залежності від завдання, генерує команди управління для використовуваних пристроїв [2].

За первинним задумом розумний будинок повинен бути готовим до змін, тобто будівля повинна легко пристосовуватися до потреб людини. Даній технології властива можливість змінювати конфігурацію систем, нарощуючи або видозмінюючи її. Технічні та інженерні системи такого будинку повинні бути спроєктовані так, щоб їх можна було досить просто адаптувати до можливих змін в майбутньому. Крім цього, складові таких систем повинні мати можливість інтеграції один з одним при мінімумі витрат. Їх обслуговування повинно бути організовано також оптимальним чином.

Основні функції розумного будинку включають в себе управління наступними системами:

- електропостачання та освітлення;
- інтернет, телефонний та стільниковий зв'язок, система оповіщення;
- телебачення, аудіо- та відеосистеми;
- дистанційне керування;
- водопостачання та каналізація;
- клімат-контроль, опалення та вентиляція;

- забезпечення безпеки та відеоспостереження;
- пожежна сигналізація;
- моніторинг поломок, наприклад витоків газу або протікання води;

За допомогою єдиного керуючого модуля, до якого підключений, наприклад, датчик руху або ультразвуковий далекомір можна з’єднати з системою освітлення, та при наближенні об’єкта від датчика до керуючого модуля, а від нього до електричної системи освітлення передаватиметься сигнал , який його увімкне. На зразок цього будуються аудіо системи, які взаємодіють з керуючим модулем, а той у свою чергу має бездротовий приймач, до я кого користувач може під’єднатись та вмикати мультимедійні файли.

Для таких систем з нескладними задачами, керуючим модулем, в основному, виступають мікроконтролери (МК). Існують декілька актуальних на сьогоднішній день мікроконтроллерів на базі мікропроцессорів, які використовуються для подібного виду завдань. Самі популярні з них це МК Arduino та Respberry.

Кожен із наведених видів МК має ряд переваг та недоліків. Тому важливим етапом для створення прототипу квест-кімнати є розробка мікропроцесорних систем.

1.2 Розробка мікропроцесорної системи на основі мікроконтролера

Мікропропроцесорна система (МПС) на основі МК використовується найчастіше як вбудована системи для рішення задач керування певним об'єктом. Важливою особливістю даного застосування є робота в реальному часі, тобто забезпечення реакції на зовнішні події протягом визначеного часового інтервалу. Такі пристрої одержали назву контролерів.

Технологія проєктування контролерів на базі МК цілком відповідає принципу нерозривного проєктування та відлагодження апаратних та програмних засобів, прийнятому в мікропроцесорній техніці. Це означає, що перед виробником такого роду мікропроцесорної техніки (МПТ) стоїть задача реалізації повного циклу проєктування, починаючи від розробки алгоритму функціонування,

закінчуючи комплексними іспитами в складі виробу, а, можливо, та супроводом при виробництві [3].

У технічному завданні формуються вимоги до контролера з точки зору реалізації визначеної функції керування. Технічне завдання містить у собі набір вимог що визначають дії, які користувач може подавати на контролер, та які розроблювальний прилад повинен робити. Технічне завдання може мати вид текстового опису, яке іноді налічує у собі внутрішні протиріччя.

На підставі вимог користувача складається функціональна специфікація, що визначає функції, що виконуються контролером за запитом користувача після завершення проєктування, уточнюючи тим самим, наскільки пристрій відповідає пропонованим вимогам. Вона містить у собі описи форматів даних, як на вході, так та на виході, а також зовнішні умови, що керують діями контролера.

Функціональна специфікація та вимоги користувача є критеріями оцінки функціонування контролера після завершення проєктування. Може знадобитися проведення декількох ітерацій, що включають обговорення вимог та функціональної специфікації з потенційними користувачами контролера, та відповідну корекцію вимог та специфікації. Вимоги до типу використовуваного МК формуються на даному етапі найчастіше в неявному виді.

Етап розробки алгоритму керування є найбільш відповідальним, оскільки помилки даного етапу зазвичай виявляються тільки при іспитах закінченого виробу та приводять до потреби в дорогій переробці всього пристрою. Розробка алгоритму звичайно зводиться до вибору одного з декількох можливих варіантів алгоритмів, що відрізняються співвідношенням об'єму програмного забезпечення й апаратних засобів.

При цьому необхідно виходити з того, що максимальне використання апаратних засобів спрощує розробку та забезпечує високу швидкодію контролера в цілому, але супроводжується, як правило, збільшенням вартості та споживаної потужності. Зв'язано це з тим, що збільшення частки апаратних засобів досягається або шляхом вибору більш складного МК, або шляхом використання

спеціалізованих інтерфейсних схем. Зазначені фактори спричиняють зростання вартості й енергоспоживання. Збільшення питомої ваги програмного забезпечення надає можливість скоротити кількість елементів контролера та вартість апаратних засобів, але це приводить до зниження швидкодії, збільшенню необхідного об'єму внутрішньої пам'яті МК, збільшенню термінів розробки та налагодження програмного забезпечення. Критерієм вибору надалі - є можливість максимальної реалізації заданих функцій програмними засобами при мінімальних апаратних витратах та за умови забезпечення заданих показників швидкодії та надійності в повному діапазоні умов експлуатації. Загальними вимогами є можливість захисту інформації (програмного коду) контролера, необхідність забезпечення максимальної тривалості роботи в автономному режимі й інші. У результаті виконання цього етапу остаточно формуються вимоги до параметрів використовуваного МК [3].

При виборі типу МК враховуються наступні основні характеристики:

- розрядність;
- швидкодія;
- набір команд та способів адресації;
- вимоги до джерела живлення та споживана потужність у різних режимах;
- об'єм постійно запам'ятовуючого пристрою (ПЗП) програм та оперативно запам'ятовуючого пристрою (ОЗП) даних;
- можливості розширення пам'яті програм та даних;
- наявність та можливості периферійних пристроїв, включаючи засоби підтримки роботи в реальному часі (таймери, процесори подій та т. п.);
- можливість перепрограмування в складі пристрою;
- наявність та надійність засобів захисту внутрішньої інформації;
- можливість постачання в різних варіантах конструктивного виконання;
- вартість у різних варіантах виконання;
- наявність повної документації;

- наявність та доступність ефективних засобів програмування та налагодження МК;
- кількість та доступність каналів постачання, можливість заміни виробами інших фірм.

Список цей не є вичерпним, оскільки специфіка проєктованого пристрою може перенести акцент вимог на інші параметри МК. Визначальними можуть виявитися, наприклад, вимоги до точності внутрішнього компаратора чи напруг наявність великої кількості вихідних каналів широтно імпульсної модуляції (ШІМ).

Однак, для практичної реалізації можливості вибору оптимального МК необхідне досить глибоке пророблення алгоритму керування, оцінка об'єму програми, яка виконується, та кількості ліній з'єднання з об'єктом на етапі вибору МК. Допущені на даному етапі прорахунки можуть згодом привести до необхідності зміни моделі МК та повторного розведення друкованої плати макета контролера. У таких умовах доцільно виконувати попереднє моделювання основних елементів прикладної програми з використанням програмно-логічної моделі обраного МК.

За відсутності МК, який забезпечує необхідні згідно технічного завдання (ТЗ) характеристики проєктованого контролера, необхідно повернутися до етапу розробки алгоритму керування та перегляду співвідношення між об'ємом програмного забезпечення й апаратних засобів. Відсутність придатного МК найчастіше означає, що для реалізації необхідного об'єму обчислень (алгоритмів керування) за відведений час потрібна додаткова апаратна підтримка. Негативний результат пошуку МК пов’язаний із необхідними характеристиками зв'язаний також з необхідністю обслуговування великої кількості об'єктів контролю. У цьому випадку можливе використання зовнішніх схем обробки МК.

На етапі розробки структури контролера остаточно визначається поєднання наявних та таких, які необхідно розробляти, апаратних модулів, протоколів обміну між модулями, типів роз'ємів. У частині програмного забезпечення визначаються

поєднання та зв'язки програмних модулів, мова програмування. На цьому ж етапі здійснюється вибір засобів проєктування та налагодження.

Можливість перерозподілу функцій між апаратними та програмними засобами на даному етапі існує, але вона обмежена характеристиками вже обраного МК [3]. При цьому необхідно мати на увазі, що сучасні МК випускаються, як правило, серіями (сімействами) контролерів, сумісних програмно та конструктивно, але, такі, що відрізняються за своїми можливостями (об'єм пам'яті, набір периферійних пристроїв та т. д.). Це дає можливість вибору структури контролера з метою пошуку найбільш оптимального варіанта реалізації.

1.2.1 Основні етапи розробки апаратної частини

Після розробки структури апаратних та програмних засобів подальша робота над контролером може бути розпаралелена. Розробка апаратних засобів містить у собі розробку загальної принципової схеми, розведення топології плат, монтаж макета та його автономне налагодження. Час виконання цих етапів залежить від наявного набору апробованих функціонально-топологічних модулів, досвіду та кваліфікації виробника. На етапі вводу принципової схеми та розробки топології використовуються, як правило, розповсюджені системи проєктування типу "ACCEL EDA" чи "OrCad" [3].

Автономне налагодження апаратури на основі МК із відкритою архітектурою припускають контроль стану багаторозрядних магістралей адреси та даних з метою перевірки правильності звертання до зовнішніх ресурсів пам'яті та периферійних пристроїв. Закрита архітектура МК припускає реалізацію більшості функцій розроблювального пристрою внутрішніми засобами мікроконтролера. Тому розроблювальний контролер буде мати малу кількість периферійних інформаційних систем(ІС), а обмін з ними буде відбуватись переважно за послідовними інтерфейсами. Проте, в даному випадку виникають питання узгодження за навантажувальною здатністю паралельних портів МК та налагодження алгоритмів обміну послідовними каналами [3].

1.2.2 Основні етапи розробки програмної частини

Зміст етапів розробки програмного забезпечення, його трансляції та налагодження на моделях істотно залежить від використовуваних системних засобів. В даний час ресурси 8-розрядних МК достатні для підтримки програмування на мовах високого рівня. Це дозволяє використовувати всі переваги структурного програмування, розробляти програмне забезпечення з використанням роздільно трансльованих модулів. Одночасно продовжують широко використовуватися мови низького рівня типу асемблера, особливо при необхідності забезпечення контрольованих інтервалів часу. Задачі попередньої обробки даних часто вимагають використання обчислень із плаваючою комою, трансцендентних функцій [3].

В даний час найпотужнішим засобом розробки програмного забезпечення для МК є інтегровані середовища розробки, що складаються з менеджера проєктів, текстового редактора та симулятора, а також допускають підключення компіляторів мов високого рівня типу Паскаль чи Сі. При цьому, варто зазначити, що архітектура багатьох 8-розрядних МК унаслідок малої кількості ресурсів, сторінкового розподілу пам'яті, незручної індексної адресації та деяких інших архітектурних обмежень не забезпечує компілятору можливості генерувати ефективний код. Для обходу цих обмежень, виробники ряду компіляторів змушені були перекласти на користувача самостійну оптимізацію коду програми.

Для перевірки та налагодження програмного забезпечення використовуються так звані програмні симулятори, які надають користувачу можливість виконувати розроблену програму на програмно-логічній моделі МК. Програмні симулятори поширюються, як правило, безкоштовно та сконфігуровані відразу на кілька МК одного сімейства. Вибір конкретного типу МК серед моделей сімейства забезпечує відповідна опція меню конфігурації симулятора. При цьому моделюється робота центрального процесора (ЦП), усіх портів вводу/виводу, переривань та іншої периферії. Карта пам'яті модельованого МК завантажується в симулятор автоматично, налагодження ведеться в символьних позначеннях регістрів.

Завантаживши програму в симулятор, користувач має можливість запускати її в покроковому чи безупинному режимах, задавати умовні чи безумовні точки зупинки, контролювати та вільно модифікувати вміст комірок пам'яті та регістрів симульованого МК [3].

1.3 Методи та засоби налаштування апаратних та програмних засобів

Етап спільного налагодження апаратних та програмних засобів у реальному масштабі часу є самим трудомістким та вимагає використання інструментальних засобів налагодження. До числа основних інструментальних засобів налагодження відносять:

- схемні емулятори;
- плати розвитку (оціночні плати);
- монітори налагодження;
- емулятори ПЗП.

Схемний емулятор – програмно-апаратний засіб, здатний замінити емульований МК у реальній схемі. Стикування схемного емулятора з відлагоджуваною системою відбувається за допомогою кабелю з спеціальною емуляційною голівкою, яка вставляється замість МК у відлагоджувану систему. Якщо МК не можна вийняти з відлагоджуваної системи, то використання емулятора можливе, тільки, якщо цей мікроконтролер має відлагоджувальний режим, при якому всі його виводи знаходяться в третьому стані. У цьому випадку для підключення емулятора використовують спеціальний адаптер-кліпсу, яка під'єднується безпосередньо до виводів емульованого МК [3].

Схемний емулятор - це найбільш потужний й універсальний відлагоджувальний засіб, який робить процес функціонування відлагоджуваного контролера прозорим, тобто легко контрольованим, довільно керованим та таким, що легко модифікується.

Плати розвитку, чи, як прийнято їх називати в закордонній літературі, оціночні плати (Evaluation Boards), є свого роду конструкторами для макетування

електронних пристроїв. Зазвичай, це друкована плата з установленим на ній МК та всієї необхідної йому стандартної периферії. На цій платі також установлюють схеми зв'язку з зовнішнім комп'ютером. Як правило, там же є вільне поле для монтажу прикладних схем користувача. Іноді передбачено вже готове розведення для установки додаткових пристроїв, що рекомендуються фірмою. Наприклад, постійно запам’ятовуючий пристрій (ПЗП), ОЗП, рідкокристалічний дисплей, клавіатура, аналого-цифровий перетворювач (АЦП) та ін. Крім навчальної чи макетної мети, такі дороблені користувачем плати можна використовувати як контролери які складаються з однієї плати, які вбудовуються в малосерійну продукцію.

Для більшої зручності плати розвитку комплектуються ще та найпростішим засобом налагодження на базі монітора налагодження. Використовуються два типи моніторів налагодження: один для МК, що мають зовнішню шину, а другий - для МК, що не мають зовнішньої шини [3].

У першому випадку відлагоджувальний монітор поставляється у виді мікросхеми ПЗП, що вставляється в спеціальну розетку на платі живлення. Плата також має ОЗП для програм користувача та канал зв'язку з зовнішнім комп'ютером чи терміналом. В другому випадку електронна плата має вмонтовані схеми програмування внутрішнього ПЗП мікроконтроллеру, які керуються від зовнішнього комп'ютера. При цьому програма монітора просто заноситься в ПЗП МК разом із прикладними кодами користувача. Прикладна програма повинна бути спеціально підготовлена для того, щоб у потрібні місця уставити виклики відлагоджувальних підпрограм монітора. Потім здійснюється пробний прогін. Щоб внести в програму виправлення, користувачу треба стерти ПЗП та зробити повторний запис. Готову прикладну програму одержують з налагодженої шляхом видалення усіх викликів моніторних функцій та самого монітора налагодження. Можливості налагодження, надані комплектом "плата розвитку плюс монітор", не настільки універсальні, як можливості схемного емулятора, та й деяка частина ресурсів МК у процесі налагодження відбирається для роботи монітора. Проте,

наявність набору готових програмно-апаратних засобів, що дозволяють без утрати часу приступити до монтажу та налагодження проєктованої системи, у багатьох випадках є вирішальним чинником. Особливо, враховуючи, що вартість такого комплекту трохи менша, ніж вартість більш універсального емулятора.

Емулятор ПЗП - програмно-апаратний засіб, що дозволяє заміщати ПЗП на відлагоджуваній платі, та підставляє замість нього ОЗП, у яке може бути завантажена програма з комп'ютера через один зі стандартних каналів зв'язку. Цей пристрій надає можливість користувачу уникнути багаторазових циклів перепрограмування ПЗП. Емулятор ПЗП потрібен тільки для МК, які можуть звертатися до зовнішньої пам'яті програм. Цей пристрій за складністю та вартістю можна порівняти з платами розвитку та має одну велику перевагу-універсальність. Емулятор ПЗП може працювати з будь-якими типами МК.

Емульована пам'ять доступна для перегляду та модифікації, але контроль над внутрішніми керуючими регістрами МК був донедавна неможливий [3].

Останнім часом з'явилися моделі інтелектуальних емуляторів ПЗП, які надають можливість "заглядати" усередину МК на платі користувача. Інтелектуальні емулятори – це гібрид зі звичайного емулятора ПЗП, монітора налагодження та схем швидкого перемикання шини з емулятора на монітор. Це створює ефект, як якби монітор налагодження був установлений на платі користувача та при цьому він не займає в МК ніяких апаратних ресурсів, крім невеликої зони програмних кроків.

Етап спільного налагодження апаратних та програмних засобів у реальному масштабі часу завершується, коли апаратура та програмне забезпечення спільно забезпечують виконання всіх кроків алгоритму роботи системи. Наприкінці етапу налагодження програма заноситься за допомогою програматора в енергонезалежну пам'ять МК, та перевіряється робота контролера без емулятора. При цьому, використовуються лабораторні джерела живлення. Частина зовнішніх джерел сигналів може моделюватися.

Етап інтеграції розробленого контролера у виріб полягає в повторенні робіт зі спільного налагодження апаратури та керуючої програми, але при роботі в складі виробу, живленні від штатного джерела та з інформацією від штатних джерел сигналів та датчиків [3].

Поєднання й об'єм випробувань розробленого та виготовленого контролера залежить від умов його експлуатації та визначається відповідними нормативними документами. Проведення випробувань таких функціонально складних виробів, як сучасні контролери, може вимагати розробки спеціалізованих засобів контролю стану виробу під час випробувань.

1.4 Прототипування мікропроцесорних систем

Усі складні, великі за розміром електричні системи перед їх виконанням у реальному розмірі прототипуються. Для прототипуання мікропроцесорних систем в загальному вигляді постає задача поєднання всіх пристроїв в єдину систему, для вирішення якої необхідно використовувати як головний елемент мікропроцессор, як засіб для зв’язку між усіма компонентами прототипу. Для вирішення масштабного проекту робиться його макет, який у декілька разів менший у розмірі, ніж реальний об’єкт. В залежності від типу завдання, під час прототипування допускаються певні спрощення у конструкції, проте усі електричні та схеми підключення зберігають своє логічне призначення. Такі прототипи мають відповідати загальному процесу та функціональності реального об’єкту.

Зазвичай прототипування робиться для того, щоб перевірити функціональність систем перед їх реалізацією. Під час перевірки виявляються всі недоліки, що можуть вплинути на готовий продукт та можливість виготовлення такої системи.

Прикладами такого підходу є системи освітлення, опалення, поливні системи. У систем освітлення ключовими елементами для прототипування є датчики присутності та освітленості, тобто система буде спрацьовувати у певний час та на певних ділянках, щоб перевірити як це працює, достатньо зібрати макет

об’єкту де ця система буде застосовуватись та задати рухомий об’єкт. Щодо систем опалення це датчик температури та присутності, для поливу – датчик вологості ґрунту та температури [1, 4-5].

В даному дипломному проекті пропонується розробити, зібрати та запрограмувати макет квест-кімнати як прототип мікропроцесорної системи на основі принципів розумного будинку.

РОЗДІЛ 2

РОЗРОБКА КВЕСТ КІМНАТИ

2.1 Вимоги та завдання до прототипу

Макет квест-кімнати представляє собою ігровий варіант системи, в якій поєднано декілька різних типів пристроїв за допомогою МК Arduino Nano. В проєкті пропонується вирішити 3 види завдань, щоб добути перемогу. В якості трьох квестів було обрано наступні:

1. Змінюючи опір потенціометру, потрібно знайти таке значення, щоб отримати підказку до завдання з введенням коду та переходити до наступного завдання. В якості сигналізуючого елементу щодо забезпечення користувача інформації правильності/хибності проходження квесту, використовується RGB світлодіод.

2. Метою квесту є правильно визначити комбінацію з чотирьох положень. Ці положення фіксуються програмно. Кожна правильна дія відображається включенням відповідного світлодіода. Після вирішення завдання, користувач отримує підказку до завдання з введенням коду та переходить до наступного квесту.

3. Отримавши усі підказки користувач переходив до третього завдання , умовою якого є введення коду, шляхом натискання кнопок. Значення коду складається з чотирьох регістрів. Вирішення цього завдання надає користувачу приз.

2.2 Характеристика плати Arduino Nano

Arduino Nano (рис. 2.1) це плата мікроконтролера на базі ATmega328P. МК має 14 цифрових пінів вводу / виводу (з яких 6 можна використовувати як PWM виходи), 6 аналогових входів, кварцовий кристал 16 МГц, USB-з'єднання, роз'єм живлення, вивід ICSP та кнопку скидання. Плата містить все необхідне для

підтримки мікроконтролера. Для його увімкнення потрібно під’єднати до комп'ютера за допомогою USB-кабелю або за допомогою адаптера змінного струму-постійного струму або акумулятора.

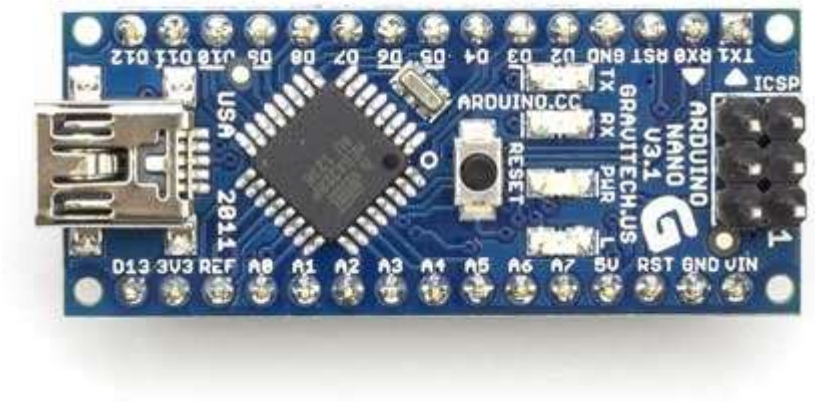

Рис. 2.1 Arduino Nano

Характеристики Arduino Nano:

- Мікроконтролер: ATmega328
- Робоча напруга: 5В
- Напруга живлення (рекомендована): 7-12В
- Напруга живлення (гранична): 6-20В
- Цифрові входи/виходи: 14 (з них 6 можуть використовуватися в якості ШІМ-виходів)
- Аналогові входи: 6
- Максимальний струм одного виводу: 40 мА
- Максимальний вихідний струм виводу: 3.3V 50 мА
- Flash-пам'ять: 32 КБ (ATmega328) з яких 0.5 КБ використовуються завантажувачем
- SRAM: 2 КБ (ATmega328)
- EEPROM: 1 КБ (ATmega328)

- Тактова частота: 16 МГц[3]

2.2.1 Опис виводів Arduino Nano

Плата Arduino Nano (рис. 2.2) налічує 14 цифрових виводів які за попередніх налаштувань працюють на вхід або вихід сигналу . Напруга на входах/виходах 5В. Значення струм, який може віддавати або споживати один вивід, становить 40 мА.

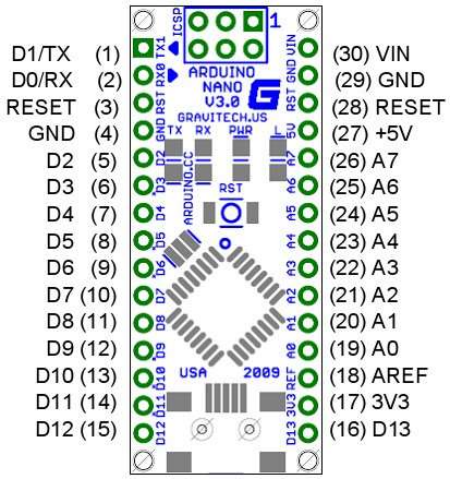

Рис. 2.2 Опис виводів

Всі виводи пов'язані з внутрішніми налічують резистори (за замовчуванням-відключеними) номіналом 20-50 кОм. Крім цього, деякі виводи Arduino можуть виконувати додаткові функції:

- Послідовний інтерфейс: виводи 0 (RX) та 1 (TX). Використовуються для отримання (RX) та передачі (TX) даних по послідовному інтерфейсу. Ці виводи з'єднані з відповідними виводами мікросхеми ATmega8U2, яка виконує роль перетворювача USB-UART.
- Піни 2 та 3 можуть служити джерелами переривань, що виникають при фронті, спаді або при низькому рівні сигналу на цих виходах.
- ШІМ: виводи 3, 5, 6, 9, 10 та 11 можуть виводити 8-бітові аналогові значення в вигляді ШІМ-сигналу.
- Інтерфейс SPI: виводи 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Із застосуванням бібліотеки SPI дані виводи можуть здійснювати зв'язок по інтерфейсу SPI.

- Світлодіод: 13. Вбудований світлодіод, приєднаний до виходу 13. При відправці значення HIGH світлодіод включається, при відправці LOW - вимикається.
- В Arduino Nano є 6 аналогових входів (A0 - A5), кожен з яких може утворити аналогову напругу у вигляді 10-бітного числа (1024 різних значення). За замовчуванням, вимір напруги здійснюється щодо діапазону від 0 до 5 В. Проте, верхню межу цього діапазону можна змінити, використовуючи вивід AREF .Крім цього, деякі з аналогових входів мають додаткові функції:
- TWI: вивід A4 або SDA та вивід A5 або SCL. Дані виводи можуть здійснювати зв'язок по інтерфейсу TWI.

Крім перерахованих на платі існують два піна:

- AREF. Вхід для живлення 7-12В.
- Reset. Подача логічого нуля (LOW) на цей піін перезавантажить мікроконтролер. Цей вивід використовують для кнопки скидання налаштувань на платах розширення [6].

2.3 Периферійні пристрої

Відовідно до завдання, для його реалізаціїї та коректної роботи було підібрано потрібні електронні компоненти. Для виконання 1го пункту було використано потенціометр з ручкою та для світлової індикації було узято 1 RGB світлодіод. Для виконання другого, задіяли джойстик та 4 світлодіоди. У третьому завданні реалізували 7-ми сегментний індикатор та 4 кнопки, також для відкривання дверцят після виконання кожного завдання використали сервоприводи.

2.3.1 Опис джойстику

Модуль джойстика (рис. 2.3) на Arduino аналогічний тим, які використовуються в ігрових приставках. Це зроблено шляхом установки двох

потенціометрів під кутом 90 градусів. Потенціометри з'єднані з короткою палицею, центрованої пружинами.

Цей модуль виробляє на виході близько 2,5 В від X та Y, коли він знаходиться в положення спокою. Переміщення джойстика приведе до зміни вихідного сигналу від 0 В до 5 В, в залежності від його напряму. Якщо ви підключите цей модуль до мікроконтролеру, ви можете очікувати, що значення буде близько 512 в положенні спокою.

Коли ви переміщаєте джойстик, ви можете побачити, що значення змінюються від 0 до 1023, в залежності від його положення [6].

Рис. 2.3 Джойстик

Характеристики:

- Напруга живлення: номінальна 3.0…5,5 В;
- Вихідний сигнал: цифровий (кнопка) и аналоговий (осі X и Y);
- Розміри: 26 мм x 40 мм x 22 мм.

2.3.2 Опис екрану TM74HC595 Display

Модуль семисегментного 4-значного дисплея 0,36 дюйма червоного кольору світіння з керуванням на драйвері 74HC595 (рис. 2.4). Модуль управляється як від контролера Arduino так та від Raspberry Pi або будь-якого іншого міні-комп'ютера.

Характеристики:

- Мікросхеми регістрів: 74HC595
- Напруга живлення: від 3.3 до 5 В
- Індикатор: 4-х символьний 0.36 дюйма з загальним анодом

- Розмір: 46 x 22 x 10 мм [6].

Рис. 2.4 TM74HC595 Display

2.3.3 Опис сервоприводу

Сервопривод (рис. 2.5) - це механізм з електромотором з керуванням. За допомогою налаштувань механічний привід повертається на заданий кут із заданою швидкістю або зусиллям. Найбільш популярні сервоприводи, які утримують заданий кут та сервоприводи, що підтримують задану швидкість обертання. Сервоприводи мають кілька складових частин. Привід - електромотор з редуктором. Найчастіше швидкість обертання мотора буває занадто велика для практичного використання. Для зниження швидкості використовується редуктор, механізм з шестернею, передає та перетворює крутний момент.

Включаючи та вимикаючи електромотор, можна обертати вихідний вал - кінцеву шестерню сервоприводу, до якої можна під’єднати – важіль у формі кола, хрестовини або перекладинки для передачі крутного руху на робочий елемент. Для контролю положення використовується датчик зворотного зв'язку - енкодер, який буде перетворювати кут повороту назад в електричний сигнал. Для цього часто використовується потенціометр. При повороті бігунка потенціометра відбувається

зміна його опору, пропорційне куту повороту. Таким чином, з його допомогою можна встановити поточний стан механізму [6].

Крім електромотора, редуктора та потенціометра в сервоприводі є електронна начинка, яка відповідає за прийом зовнішніх параметрів, зчитування значень з потенціометра, їх порівняння та включення / вимикання двигуна. Вона відповідає за підтримання негативного зворотного зв'язку.

Сервопривод має три контакти, за допомогою яких відбувається підключення до системи. Два з них відповідають живленню мотора, третій передає керуючий сигнал, який використовується для виставлення положення пристрою.

Рис. 2.5 Сервопривод

Характеристики:

- Робоча напруга - 4.8-7.2 В
- Кут повороту 120 градусів
- Крутний момент - 8,5 кг / см (при 4.8 В), 10 кг / см (при 6 В)
- Швидкість - 0,20 сек / 60 ° (при 4,8 В), 0,16 сек / 60 ° (при 6 В)
- Матеріал шестерні - метал
- Вага - 55 г
- Розмір - 40x20x42 мм[3]

2.4 Розробка схем

2.4.1 Розробка структурної схеми

Для реалізації поставленого завдання та візуалізації проєкту потрібно розробити структурну схему (рис. 2.6), яка буде включати наступні блок: Arduino Nano, блок живлення, джойстик, дисплей, LED, LED RGB, сервоприводи.

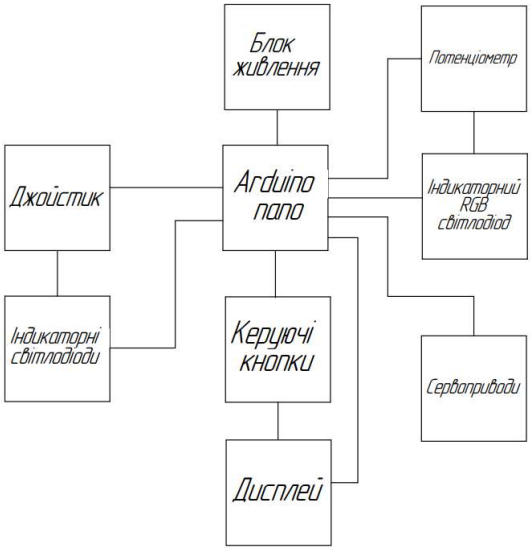

Рис. 2.6 Структурна схема квест-кімнати

Цей пристрій побудовано на платі Arduino Nano, яка використовує мікроконтролер ATmega328 - саме він є центральним елементом, який визначає всю роботу системи. Сама схема логічно розбита на 1 вхід живлення, 1 вихід сервопривід, 3 інших блоки (завдання), отримують та віддають данні по колу.

2.4.2 Розробка та опис принципової схеми

Відповідно до структурної схеми (рис. 2.6), було створено принципову схему (рис. 2.7), яка визначає повний состав елементів та зв'язки між ними й дає детальне уявлення про принципи роботи виробу. На ній зображені усі елементи та

пристрої, необхідні для здійснення й контролю у виробі електричних процесів, усі електричні зв'язки між ними, а також елементи(з'єднувачі, затискачі та т.д.), якими закінчуються вхідні й вихідні ланцюги, та включає наступні елементи:

- ArduinoNano;
- Джойстик;
- TM74HC595 Display;
- LED RGB;
- LED 4 шт.;
- Резистор 200Ом 4шт.;
- Потенціометр10 кОм;
- 4 кнопки:

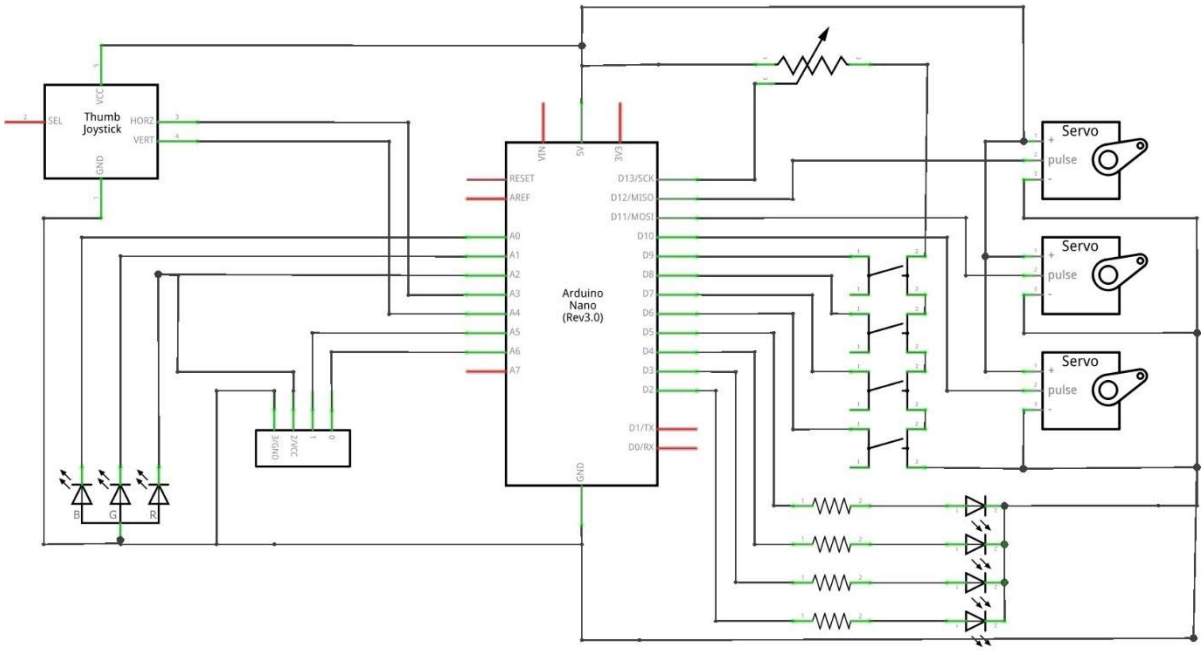

Рис.2.7 Принципова електрична схема

Призначення елементів:

- Arduino Nano - є ключовим елементом керування усіма елементами.
- Джойстик, 4 LED, сервопривід у взаємодії з платою є одним із «завдань», яке необхідно виконати користувачу , а саме комбінацією на джойстику відкрити двері з сервоприводом;

- Потенціометр, LED RGB, сервопривід у взаємодії з платою є одним із «завдань», яке необхідно виконати користувачу , а саме задати поворотом ручки значення опору/кольору, яке відкриє двері з сервоприводом;
- TM74HC595 Display, 4 кнопки, сервопривід у взаємодії з платою є одним із «завдань», яке необхідно виконати користувачу, а саме ввести кодовий ключ на дисплей за допомогою кнопок, підказка до якого лежить у двох попередніх відкритих дверцях. Вирішення останнього завдання забезпечує отримання головного «призу».

2.4.4 Розробка та опис схеми підключення

На рис. 2.8 наведено схему підключення всіх компонентів, відповідно до принципової (рис. 2.7), та структурної (рис. 2.6) схем .

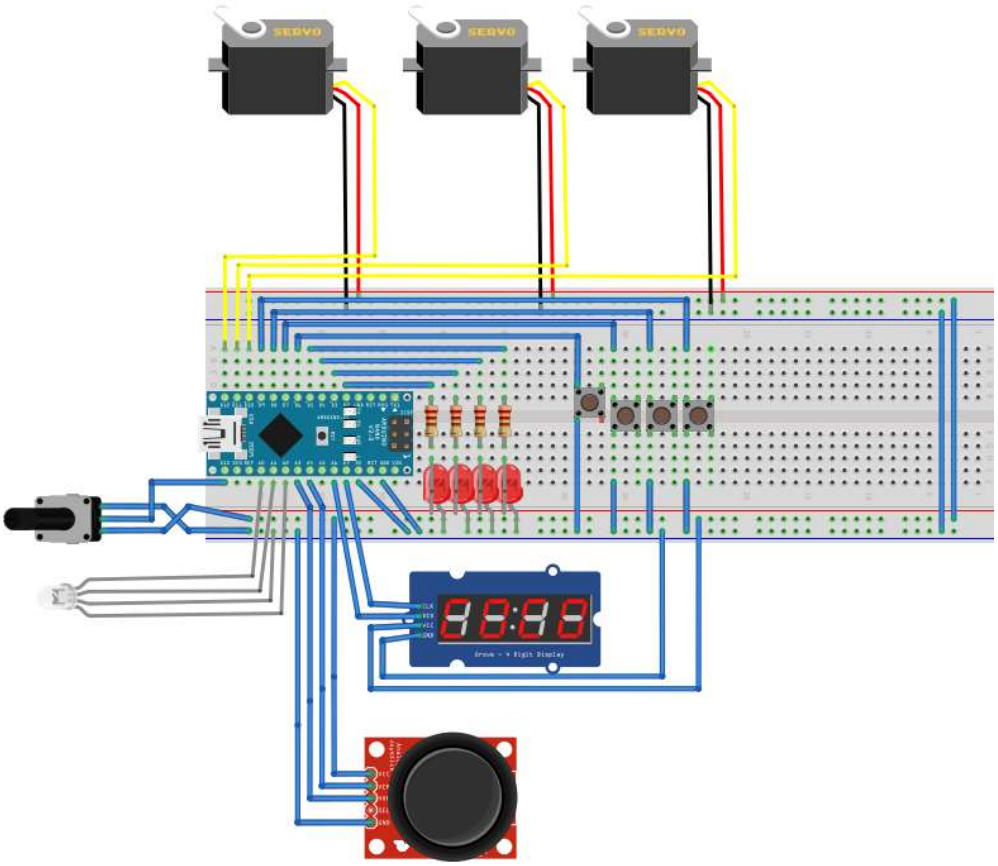

Рис.2.8 Схема підключення

Схема підключення макету квест кімнати включає наступні елементи:

- Джойстик, порти A2, A3;

- TM74HC595 Display, порти A4-A5;
- Потенціометр10 кОм, порт A6;
- LED RGB, порт A0-A1,13;
- LED 4 шт. + Резистор 200Ом 4шт, порти 12-9;
- Сервопривод 3шт., порти 8-6;
- 4 кнопки, порти 5-2.

2.5 Розробка програмного забезпечення

Для написання коду програми пристрою було використано середовище розробки Arduino IDE (рис.2.9). Воно складається з консолі в яку можна вписати код програми, відладчика, області повідомлень, панелі інструментів з доступним інтерфейсом , та також є декілька меню. Програма підключається до апаратної частини Arduino , при підключенні плати необхідно обрати у панелі інструментів необхідний COM порт та завантажити програму [7].

Скетч це програма написана в середовищі Arduino. Він пишеться в текстовому редакторі, що має інструменти редагування , пошуку та заміни тексту. Під час збереження та експорту проєкту в області повідомлень з'являються інформаційне повідомлення в нижній частині , під програмою в окремому вікні. Також там виводяться усі зауваження, які знайдуться при превірці програми. Вікно виведення тексту (консоль) показує повідомлення Arduino, що включають повні звіти про помилки та іншу інформацію. Кнопки панелі інструментів дозволяють перевірити та записати програму, створити, відкрити та зберегти скетч, відкрити моніторинг послідовної шини.

Рис. 2.9 Середовище розробки ArduinoIDE

В програмі повинні бути дві обов'язкові функції: *setup*() та *loop*(). Функція *setup*() запускається один раз, після кожного включення живлення або скидання плати Arduino (рис. 2.10). У тілі даної функції пишеться код для ініціалізації змінних, установки режиму роботи цифрових виходів. В нескінченному циклі послідовно виконує команди функція *loop*. Тобто після завершення функції знову відбудеться її виклик [8].

Блок схему алгоритму програми 1-го завдання наведено на рис. 2.10. Користувач, змінюючи положення ручки потенціометра, повинен залишити правильне положення, для того щоб спрацював механізм відчинення дверей та отримати підказку для завдання з введенням коду. Зміна положення потенціометра призводить до зміни кольору RGB світлодіода за умови правильного обертання. Код реалізації завдання наведено на рис. 2.11

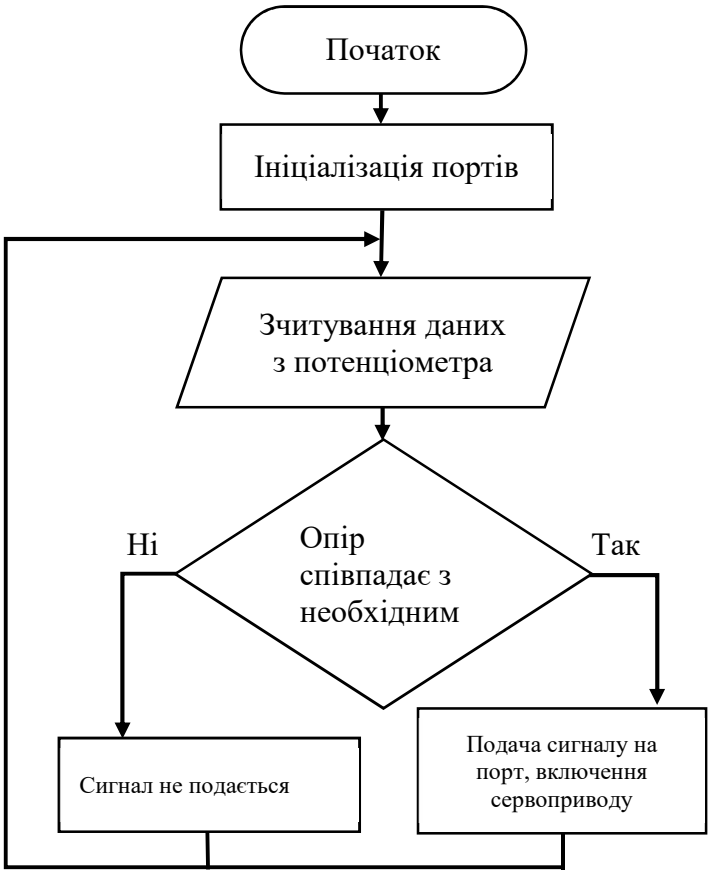

Рис. 2.10 Алгоритм програми першого завдання

```
freeMode();
logic();
val = analogRead (A0);

newVal=map(val, 0, 1023, 0, 1530);

if (newVal<=255){//красный
    rgb (newVal,0,0);
} else if ((newVal>255)&&(newVal<=510)){//ж
    rgb (255,newVal-255,0);
} else if ((newVal>510)&&(newVal<=765)) {//зел
    rgb (765-newVal,255,0);
}else if ((newVal>765)&&(newVal<=1020)) {//гол
    rgb (0,255,newVal-765);
}else if ((newVal>1020)&&(newVal<=1275)) {//с
    rgb (0,1275-newVal,255);
}else if ((newVal>1275)&&(newVal<=1530)) {//с
    rgb (newVal-1275,0,255);

}
Serial.println (newVal);

if ((newVal>1000)&&(newVal<=1530)){
    delay(500);
   servol.write(0);
   delay(5000);
   servol.write(90);
  delay(500);
   }
```

Рис.2.11 Код для реалізації першого завдання

Для виконання другого завдання було використано так звані флаги, для того щоб задати декілька умов послідовного виконання, а саме за допомогою джойстику ввести правильну послідовність його дій. Кожний правильний крок запалює світлодіод та виконує умову *krok*. При виконанні послідовності із чотирьох положень подається сигнал до сервоприводу, який відчиняє двері з підказкою до останнього завдання. Нижче зображені блок схема алгоритмів програми 1-го завдання (рис. 2.12). Та частина коду, (рис. 2.13) яка відповідає за коректне виконання цього завдання.

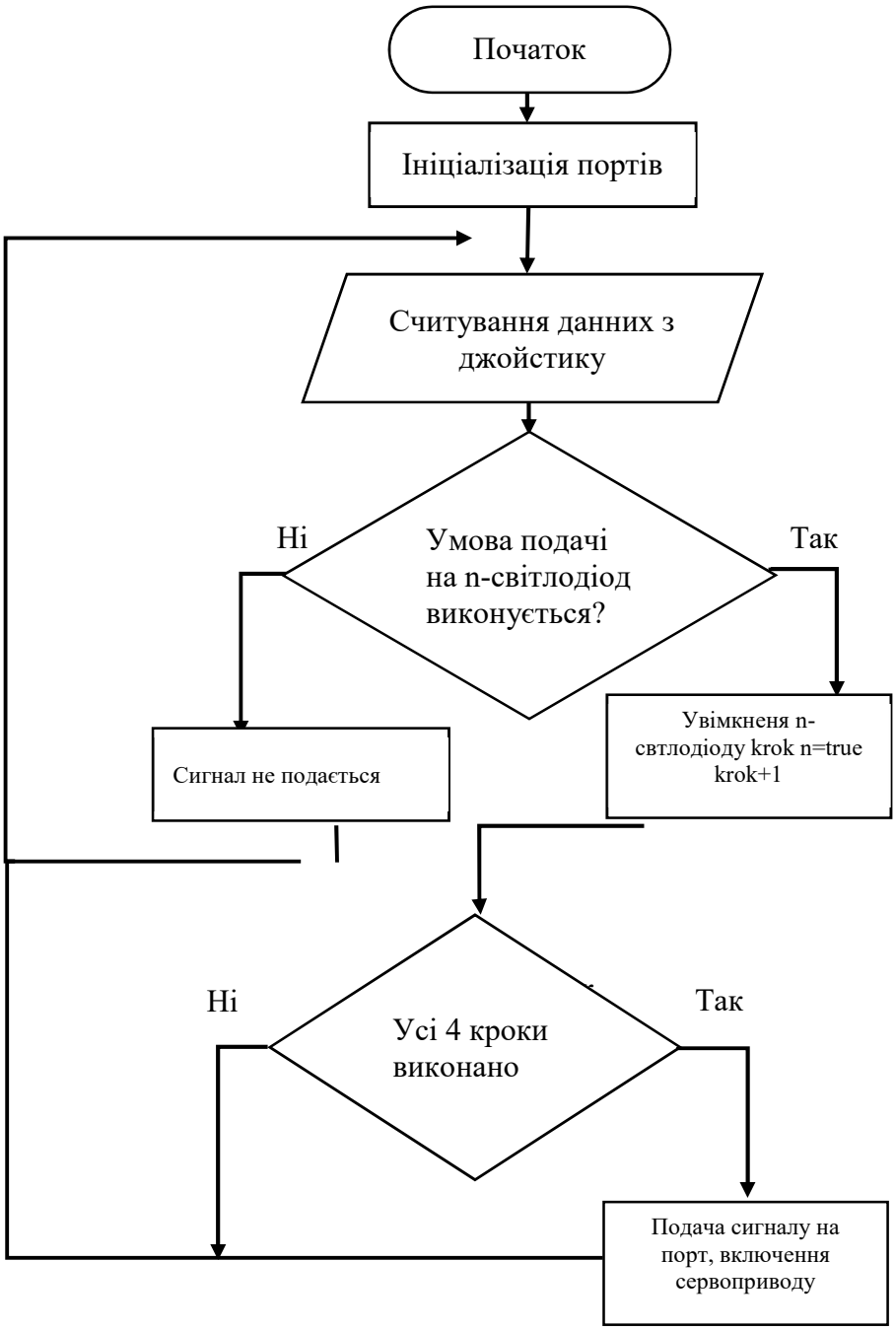

Рис. 2.12 Алгоритм програми другого завдання

```
diplom3
 void freeMode(){
   int X = analogRead(pinX);
   int Y = analogRead(pinY);
   int  Yd= map(Y, 700, 1023,    0, 128);//zele
   int Yv = map(Y, 300, 0,    0, 128);//zelt2
    int Xd = map(X, 700, 1023,  0, 128);//jeltll
    int Xv = map(X, 300, 0,  0, 128);//krasn

if ((Xd>120)&&(krok1==false)&&(krok2==false)&&(krok3==false)){
 digitalWrite(pin3,HIGH);
 krok=true;
 delay(300);
              }

if ((Yd>120)&&(krok==true)&&(X>300)&&(X<700)){
            digitalWrite(pin5,HIGH);
             krok1=true;
              delay(300);
     }
if ((Xv<20)&&(krok1==true)&&(Y<300)&&(Y<700)){
         digitalWrite(pin2,HIGH);
        krok2=true;
         delay(300);
          }
if  ((Yv<20)&&(krok2==true&&(X<300)&&(X<700))){
         digitalWrite(pin4,HIGH);
        krok3=true;
         delay(300);
         }
        }
void logic() {

if ((krok==true)&&(krok1==true)&&(krok2==true)&&(krok3==true)) {
    servo2.write(0);
    delay(5000);
    servo2.write(90);
    delay(2000);
    krok=false;
    krok1=false;
    krok2=false;
    krok3=false;
     digitalWrite(pin5,LOW);
    delay(100);
    digitalWrite(pin4,LOW);
    delay(100);
    digitalWrite(pin3,LOW);
    delay(100);
    digitalWrite(pin2,LOW);
}}
```

Рис.2.13 Код для реалізації другого завдання

Для виконання третього завдання було використано умову *if*, для того щоб вводити числа у одному із чотирьох регістрах та перемикатись між ними за допомогою натискань кнопок. Підказки до коду треба отримати із двох попередніх завдань. На рис. 2.14 наведено блок-схему алгоритму третього завдання, а на рис. 2.15 представлено запрограмовану реалізацію цього алгоритму

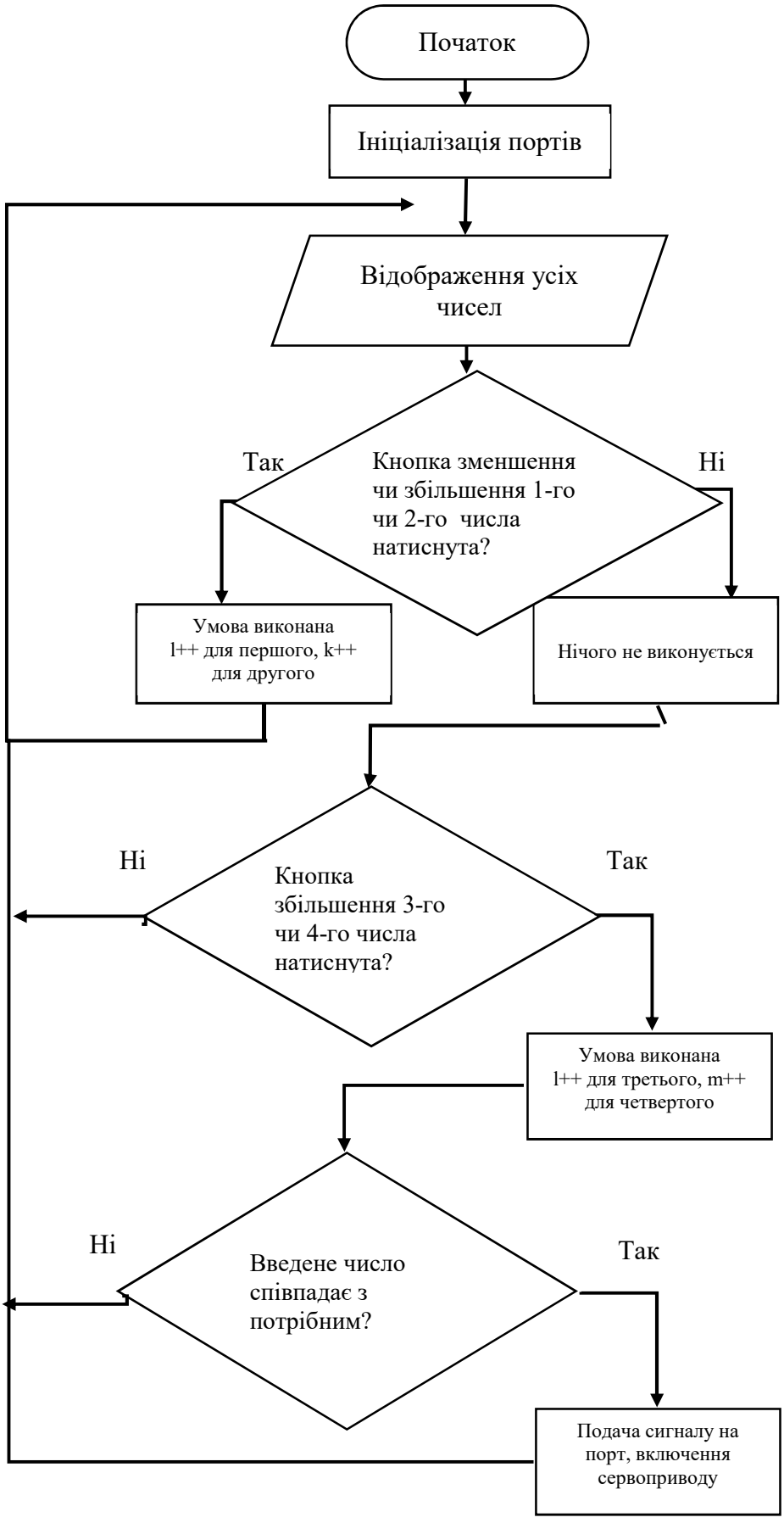

Рис. 2.14 Алгоритм програми третього завдання

```
void showDisplay(){

  const byte digit[10] = {  // маска для 7 сигментного индикатора
      0b11000000, // 0
      0b11111001, // 1
      0b10100100, // 2
      0b10110000, // 3
      0b10011001, // 4
      0b10010010, // 5
      0b10000010, // 6
      0b11111000, // 7
      0b10000000, // 8
      0b10010000, // 9
  };

  const byte chr[4] = { // маска для разряда
      0b00001000,
      0b00000100,
      0b00000010,
      0b00000001
  };

  // отправляем в цикле по два байта в сдвиговые регистры

     digitalWrite(RCLK, LOW); // открываем защелку
      shiftOut(DIO, SCLK, MSBFIRST, digit[digitBuffer[0]]);  // отправляем байт с "числом"
      shiftOut(DIO, SCLK, MSBFIRST, chr[0]);   // включаем разряд
       digitalWrite(RCLK, HIGH);
         digitalWrite(RCLK, LOW); // открываем защелку
      shiftOut(DIO, SCLK, MSBFIRST, digit[digitBuffer[8]]);  // отправляем байт с "числом"
      shiftOut(DIO, SCLK, MSBFIRST, chr[1]);   // включаем разряд
       digitalWrite(RCLK, HIGH);
         digitalWrite(RCLK, LOW); // открываем защелку
      shiftOut(DIO, SCLK, MSBFIRST, digit[digitBuffer[0]]);  // отправляем байт с "числом"
      shiftOut(DIO, SCLK, MSBFIRST, chr[2]);   // включаем разряд
       digitalWrite(RCLK, HIGH);
         digitalWrite(RCLK, LOW); // открываем защелку
      shiftOut(DIO, SCLK, MSBFIRST, digit[digitBuffer[j]]);  // отправляем байт с "числом"
      shiftOut(DIO, SCLK, MSBFIRST, chr[3]);   // включаем разряд
       digitalWrite(RCLK, HIGH);


    if (j==9) {
    delay(1); // ждем немного перед отправкой следующего "числа"
    servo3.write(0);
    delay(5000);
    servo3.write(90);
    delay(2000);
    j=0;}
```

Рис.2.15 Код для реалізації Третього завдання

Повний код програми приведений в Додатку А.

2.6 Складання, програмування та перевірка макету реально діючого пристрою

2.6.1 Складання макету

Відповідно до завдання, квест-кімната – це масштабований макет системи, в якому поєднані всі елементи електроніки в єдиний пристрій. Крім того, прототип повинен бути компактним за розміром і виконувати весь логічний функціонал. На рис. 2.16 наведено залізну коробку, яку було обрано для зовнішнього вигляду квест-кімнати. Ця коробка має розміри 18,5x13,5x9 см.

().

Рис.2.16 Вигляд коробки до поверхневої обробки

Для покращення зовнішнього вигляду була проведена обробка її поверхні. Спочатку було проведено шліфування , для зняття старого шару фарби. Далі було проведено полірування щоб прибрати подряпин з корпусу від шліфувального паперу. Після цього на корпус було нанесено 2 шари чорної фарби та 1 шар лаку (рис. 2.16).

Рис.2.16 Вигляд коробки після поверхневої обробки

Після завершення обробки корпусу квест-кімнати, всі компоненти були під’єднані всередині. Останнім етапом була перевірка працездатності всього макету.

2.6.2 Алгоритм роботи із електронним пристроєм «Квест-кімната»

Для використання даного пристрою користувачу у заданому порядку необхідно:

1. Ввімкнути кабель під’єднаний до блоку живлення у мережу 220В.

2. Перевірити чи засвітився 4-х значний дисплей на передній панелі приладу.

3. Покрутити ручку потенціометра для зміни кольору RGB світлодіоду.

4. Змінити положення джойстику у різні сторони для загорання хоча б одного світлового індикатору.

5. Після цього, пристрій перевірено і він може працювати.

6. Перевірити спрацьовування системи можна повернувши ручку потенціометра на панелі справа- при правильному положенні пристрій відкриє дверцята із підказкою.

Для повного використання приладу необхідно продовжити виконання завдання:

1. Після виконання завдання із потенціометром необхідно перейти до завдання з джойстиком у правій панелі, змінюючи положення джойстику засвітити 4 світлодіодні індикатори які знаходяться над ним

2. Забрати підказку яка знаходиться у ніші.

3. Вирішити загадки які написані на листках із ніш та отримати 4 цифрові символи.

4. Задати за допомогою кнопок отримане числове значення на дисплей розташований на передній панелі.

5. Після введення правильного числового значення, відкриється заслонка на задній панелі, отримати звідти «приз».

2.6.3 Перевірка працездатності приладу

На рис. 2.18 наведено результат перевірки першого квесту. Видно, що зміна положення ручки потенціометра призвела до зміни кольору світлодіоду. Неправильне значення відповідає червоному кольору, правильне – синьому.

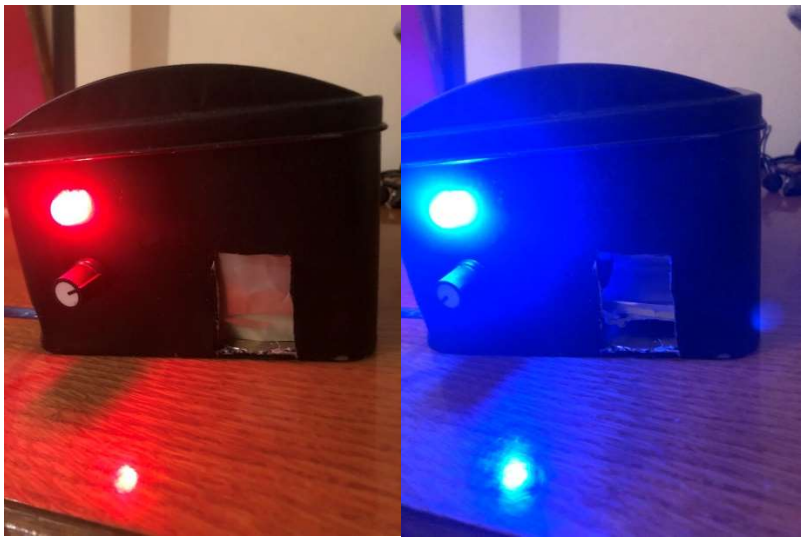

Рис. 2.18 Дія приладу

На рис. 2.19 наведено результат перевірки другого квесту. Для виконання завдання необхідно ввести комбінацію униз-вправо-ліво-уверх.

Рис. 2.19 Дія приладу

На рис. 2.20 наведено результат перевірки третього квесту. Для виконання завдання користувач повинен ввести код 1919.

Рисунок 2.20 Дія приладу

ВИСНОВОК

У проєкті проведено огляд технології «розумний» будинок і розробки програмної та апаратної частин. На сьогоднішній день це актуальна задача з можливостями застосування у багатьох сферах, тому було використано розглянутий принцип на основі мікропроцесорної техніки для розробки електронного пристрою "квест-кімната".

Наведено основні теоретичні відомості про МК Arduino Nano, логічний принцип дії та основні його характеристики. Розглянуто основні технології розробки апаратного та програмного забезпечення.

Було зібрано готовий пристрій який логічно був розділений на 3 завдання, послідовне виконання перших двох давало підказку до третього, який, у свою чергу, дає користувачу приз.

У першому завданні було використано головні властивості потенціометру - змінювати свій опір та RGB світлодіоду - змінювати колір у трьох-кольоровому діапазоні, завдяки цьому, користувач задавши ручкою отримував підказку до завдання з введенням коду та переходив далі.

У другому завданні використовувався джойстик, положення якого фіксувалось чисельним значенням вихідного сигнали, при правильному положенні виконували умови, загорялись світлодіоди та при виконанні 4-х, умов, користувач отримував підказку до завдання з введенням коду та переходив далі.

Отримавши усі підказки користувач переходив до третього завдання , де натискаючи кнопки він вводив значення у кожен із 4-х регістрів та при правильно введенному числі отримував «приз».

СПИСОК ЛІТЕРАТУРИ

[1] Пасічник Р.М. АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ ОСВІТЛЕННЯМ «РОЗУМНОГО ДОМУ» [Електронний ресурс] / Пасічник Р.М, Гнатієвич О.В.. – 2017. – Режим доступу до ресурсу: http://dspace.wunu.edu.ua/bitstream/316497/20532/1/%D0%9F%D0%B0%D1%81%D1%96%D1%87%D0%BD%D0%B8%D0%BA%20%D0%A0.%D0%9C.%20%D0%93%D0%BD%D0%B0%D1%82%D1%96%D1%94%D0%B2%D0%B8%D1%87%20%D0%9E.%D0%92..pdf.

[2] Тесля Е. «Умный дом» своими руками / Елена Тесля. – Санкт-Петербург: Издательство «Питер», 2008. – 523 с. – (Издательство «Питер»).

[3] Возняк О. Курс електроніки [Електронний ресурс] / Олег Возняк. – 2018. – Режим доступу до ресурсу: http://vozom.ho.ua/index.html.

[4] Пасічник Б.В. Магістерська дисертація [Електронний ресурс] / Пасічник Б.В.. – 2018. – Режим доступу до ресурсу: https://ela.kpi.ua/bitstream/123456789/28906/1/Pasichnyk_magistr.pdf.

[5] Харабет Р. І. Магістерська дисертація "Програмно-апаратний комплекс автоматизації ведення домашнього господарства" [Електронний ресурс] / Родіон Ігорович Харабет. – 2019. – Режим доступу до ресурсу: https://ela.kpi.ua/bitstream/123456789/32280/1/Kharabet_magistr.pdf.

[6] В.А. Петин, А.А. Биняковский - Практическая энциклопедия Arduino / ДМК Пресс, 2020.-166с

[7] Software Arduino IDE [Електронний ресурс] Режим доступу до ресурсу: https://www.arduino.cc/en/software

[8] Иго Т. Arduino, датчики и сети для связи устройств: Пер. с англ. — 2-е изд. — СПб.: БХВ-Петербург, 2015. — 544 с.: ил.

ДОДАТОК А

Код програми пристрою

```
#include <Servo.h>
#define A0 0
#define R9 A3
#define G10 A4
#define B8 A5
#define SCLK 9  // display
#define RCLK 6
#define DIO 5
Servo servo1;
Servo servo2;
Servo servo3;
byte digitBuffer[10];
const int pin5 = 2;
const int pin4 = 4;
const int pin3 = 7;
const int pin2 = 8;
const int pinX = A1;
const int pinY = A2;
int val;
int newVal;
  int j=0;
  int k=0;
boolean butt_flag = 0;
boolean butt;
boolean butt_flag1 = 0;
boolean butt1;
```

```
boolean krok=false;
boolean krok1=false;
boolean krok2=false;
boolean krok3=false;
void setup() {
  servo1.attach(11);
  servo2.attach(3);
  servo3.attach(10);
  // put your setup code here, to run once:
  pinMode(RCLK, OUTPUT);
  pinMode(SCLK, OUTPUT);
  pinMode(DIO, OUTPUT);
   pinMode(R9, OUTPUT);
  pinMode(G10, OUTPUT);
  pinMode(B8, OUTPUT);
  pinMode(pin5, OUTPUT);
  pinMode(pin4, OUTPUT);
  pinMode(pin3, OUTPUT);
  pinMode(pin2, OUTPUT);
  pinMode(pinX, INPUT);
  pinMode(pinY, INPUT);
  pinMode(12, INPUT_PULLUP);
  pinMode(A6, INPUT_PULLUP);
  Serial.begin(9600);
}
void rgb (int red, int green, int blue){
    analogWrite (R9, red);
  analogWrite (G10, green);
```

```
  analogWrite (B8, blue);
}
void loop() {
 digitBuffer[0] = 1;
  digitBuffer[1] = 2;
  digitBuffer[2] = 3;
  digitBuffer[3] = 4;
  digitBuffer[4] = 5;
  digitBuffer[6] = 6;
  digitBuffer[6] = 7;
  digitBuffer[7] = 8;
  digitBuffer[8] = 9;
  digitBuffer[9] = 0;

  showDisplay();

  // put your main code here, to run repeatedly:
  freeMode();
  logic();
  val = analogRead (A0);

  newVal=map(val, 0, 1023, 0, 1530);

  if (newVal<=255){
     rgb (newVal,0,0);
  } else if ((newVal>255)&&(newVal<=510)){
     rgb (255,newVal-255,0);
  } else if ((newVal>510)&&(newVal<=765)) {
```

```
    rgb (765-newVal,255,0);
 }else if ((newVal>765)&&(newVal<=1020)) {
    rgb (0,255,newVal-765);
 }else if ((newVal>1020)&&(newVal<=1275)) {
    rgb (0,1275-newVal,255);
 }else if ((newVal>1275)&&(newVal<=1530)) {
    rgb (newVal-1275,0,255);
      }
 Serial.println (newVal);

if ((newVal>1000)&&(newVal<=1530)){
   delay(500);
  servo1.write(360);
  delay(5000);
  servo1.write(90);
 delay(500);
  }
///кнопкилииш
 butt = !digitalRead(12);
 if (j==10) {j=0;};
 if (butt == 1 && butt_flag == 0 ) {
  butt_flag = 1;
  Serial.println("Button pressed");

 }

 if (butt == 0 && butt_flag == 1) {
  butt_flag = 0;
```

```
j++;
 }

 butt1 = !digitalRead(A6);
 if (k==5) {k=0;};
 if (butt1 == 1 && butt_flag1 == 0 ) {
   butt_flag1 = 1;
   Serial.println("Button pressed");

 }

 if (butt1 == 0 && butt_flag1 == 1) {
   butt_flag1 = 0;
 k++;
 }
  if ((newVal>1000)&&(newVal<=1530)){
    delay(500);
   servo1.write(360);
   delay(5000);
   servo1.write(90);
  delay(500);
   }

 }

 void freeMode(){
   int X = analogRead(pinX);
   int Y = analogRead(pinY);
```

```
  int  Yd= map(Y, 700, 1023,    0, 128);//zele
  int Yv = map(Y, 300, 0,   0, 128);//zelt2
   int Xd = map(X, 700, 1023,  0, 128);//krasniy
   int Xv = map(X, 300, 0,  0, 128);//jelt11

if ((Xd>120)&&(krok1==false)&&(krok2==false)&&(krok3==false)){
 digitalWrite(pin3,HIGH);
 krok=true;
 delay(300);
            }

if ((Yd>120)&&(krok==true)&&(X<300)&&(X<700)){
           digitalWrite(pin5,HIGH);
            krok1=true;
             delay(300);
    }
if ((Xv<20)&&(krok1==true)&&(Y<300)&&(Y<700)){
        digitalWrite(pin2,HIGH);
       krok2=true;
        delay(300);
         }
if  ((Yv<20)&&(krok2==true&&(X<300)&&(X<700))){
        digitalWrite(pin4,HIGH);
       krok3=true;
        delay(300);
        }
       }
void logic() {
```

```
if ((krok==true)&&(krok1==true)&&(krok2==true)&&(krok3==true)) {

   servo2.write(0);
   delay(5000);
   servo2.write(90);
   delay(2000);
   krok=false;
   krok1=false;
   krok2=false;
   krok3=false;
    digitalWrite(pin5,LOW);
   delay(100);
   digitalWrite(pin4,LOW);
   delay(100);
   digitalWrite(pin3,LOW);
   delay(100);
   digitalWrite(pin2,LOW);



}}
void showDisplay(){

  const byte digit[10] = {
     0b11000000, // 0
     0b11111001, // 1
     0b10100100, // 2
```

```
    0b10110000, // 3
    0b10011001, // 4
    0b10010010, // 5
    0b10000010, // 6
    0b11111000, // 7
    0b10000000, // 8
    0b10010000, // 9
};

const byte chr[4] = {
    0b00001000,
    0b00000100,
    0b00000010,
    0b00000001
};

   digitalWrite(RCLK, LOW);
    shiftOut(DIO, SCLK, MSBFIRST, digit[digitBuffer[j]]);
    shiftOut(DIO, SCLK, MSBFIRST, chr[0]);
     digitalWrite(RCLK, HIGH);
       digitalWrite(RCLK, LOW);
    shiftOut(DIO, SCLK, MSBFIRST, digit[digitBuffer[k]]);
    shiftOut(DIO, SCLK, MSBFIRST, chr[1]);
     digitalWrite(RCLK, HIGH);
       digitalWrite(RCLK, LOW);
    shiftOut(DIO, SCLK, MSBFIRST, digit[digitBuffer[l]]);
    shiftOut(DIO, SCLK, MSBFIRST, chr[2]);
     digitalWrite(RCLK, HIGH);
```

```
    digitalWrite(RCLK, LOW);
  shiftOut(DIO, SCLK, MSBFIRST, digit[digitBuffer[m]]);
  shiftOut(DIO, SCLK, MSBFIRST, chr[3]);
   digitalWrite(RCLK, HIGH);


if (j==9) {
delay(1);
servo3.write(0);
delay(5000);
servo3.write(90);
delay(2000);
j=0;}
if (k==1) {
delay(1);
servo3.write(0);
delay(5000);
servo3.write(90);
delay(2000);
j=0;}
if (l==1) {
delay(1);
servo3.write(0);
delay(5000);
servo3.write(90);
delay(2000);
j=0;}
```