МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра комп'ютерної інженерії

ДО ЗАХИСТУ ДОПУЩЕНИЙ

Зав. кафедри _____

д.е.н., проф. Переверзєв А.В.

ВИПУСКНА РОБОТА

РОЗРОБКА ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ ВІДОБРАЖЕННЯ 3D-ГОЛОГРАМИ «КУБ»

Виконав
ст. гр. КІ-118к9 _____ Попов М.А.

Керівник
викладач _____ Суха К.С.

Запоріжжя
2022

ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Зав. Кафедри
д.е.н., проф.
Переверзєв А.В.
_____
20.06.2022 р.

З А В Д А Н Н Я

НА ВИПУСКНУ РОБОТУ МОЛОДШОГО СПЕЦІАЛІСТА

Студенту гр. *КІ118к9*, спеціальності *123 «Комп'ютерна інженерія»*

Попову Миколі Андрійовичу

(прізвище ім'я по батькові)

1. Тема: Розробка програмно-апаратного комплексу відображення 3D-голограми «Куб»

затверджена наказом по інституту 09.2-17 від 17 січня 2022 р.

2. Термін здачі студентом закінченої роботи 20 червня 2022 р.

3. Перелік питань, що підлягають розробці

1. *Вивчення літератури, присвяченої темі розробки*

2. Детальне знайомство з особливостями роботи голограм

3. Розробка пристрою який буде виводити об'ємне 3D зображення

4. Вибір схемотехнічного рішення пристрою, розрахунок електричних параметрів та підбір електричних компонентів

5. Складання та налагодження приладу

6 Створення програмного забезпечення для роботи з приладом

7. Тестування приладу

8. Оформлення звіту

Дата видачі завдання лютого 22 р.

Керівник випускної роботи _____ Суха К.С.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Попов М.А.
(підпис студента) (прізвище та ініціали)

ЗАТВЕРДЖУЮ

Зав.кафедрою_____

К А Л Е Н Д А Р Н И Й Г Р А Ф І К

підготовки випускної роботи молодшого спеціаліста

здобувачами освіти коледжу ЗІЕІТ *денної(заочної)* **форми навчання**

гр. _____ П.І.Б._____

2021-2022 навчальний рік

№ етапу	Зміст	Термін виконання	Готовність по графіку %, підпис керівника	Підпис керівника про повну готовність етапу, дата
1.	Формулювання (корегування) теми випускної роботи молодшого спеціаліста та збір практичного матеріалу за темою випускної роботи	**17.01.22-26.02.22**		
2.	**І атестація** розділ випускної роботи молодшого спеціаліста	**28.03.22-02.04.22**		
3.	**ІІ атестація** розділ випускної роботи молодшого спеціаліста	**10.05.22-04.06.22**		
4.	**ІІІ атестація** ІІІ розділ випускної роботи молодшого с спеціаліста, висновки та рекомендації, додатки, реферат	**30.05.22-04.06.22**		
5.	Перевірка випускної роботи програмою «Антиплагіат»	**30.05.22-18.06.22**		
6.	Доопрацювання випускної роботи молодшого спеціаліста, підготовки презентації, отримання відгуку керівника і рецензії	**06.06.22-11.06.22**		
7.	**Попередній захист випускної роботи молодшого спеціаліста**	**14.06.22-18.06.22**		
8.	Подача випускної роботи на кафедру	**за 3 дні до захисту**		
9.	Захист випускної роботи молодшого спеціаліста	**20.06.22-25.06.22**		

Керівник_____(П.І.Б) «____»_____2022р.

Студент_____(П.І.Б.) «____»_____2022р.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Слово / словосполучення	Скорочення
А	
Аналого-цифровий перетворювач	АЦП
К	
Коефіцієнт корисної дії	ККД
П	
Прилад із зарядовим зв'язком	ПЗЗ
Ш	
Широтно-імпульсна модуляція (ШІМ — англ. pulse-width modulation, PWM)	ШІМ
L	
Світлодіод (англ. Light-emitting diode)	LED
Т	
Транзисторно-транзисторна логіка (ТТЛ, англ. transistor-transistor logic)	TTL
S	
(англ. Serial Peripheral Interface, SPI bus — послідовний периферійний інтерфейс, шина SPI)	SPI
Т	
двонаправлена двопровідна послідовна шина TWI (англ. Two Wire Interface)	TWI
U	
вузол обчислювальних пристроїв, призначений для організації зв'язку з іншими цифровими пристроями (англ. Universal Asynchronous Receiver-Transmitter, UART)	UART
U	
універсальна послідовна шина (англ. Universal Serial Bus,)	USB
А	
опорна напруга аналого-цифровий перетворювач (англ. Analog Reference)	AREF
С	
Послідовний порт (англ. serial port)	COM
G	
Земля (англ. GROUND)	GND
V	
Живлення (англ. voltage in)	Vin

E	
EEPROM (англ. Electrically Erasable rogrammable Read-Only Memory) — постійний апам'ятовувач	EEPROM
I	
Послідовна шина даних для зв'язку інтегральних схем (англ. Inter-Integrated Circuit)	I2C
I	
Порядок обслуговування переривання (англ. Interrupt Service Routine)	ISR
G	
Графічний процесор	GPU
I	
Внутрішньосхе́мне програмува́ння (англ. In System Programming и In Circuit Serial Programming)	ICSP

РЕФЕРАТ

Випускна робота молодшого спеціаліста: 87 сторінок, 47 рисунків, 1 таблиця, 3 додатоки, 30 першоджерел.

Об’єкт дослідження: розробка програмно-апаратного комплексу відображення 3D-голограми «Куб».

Мета роботи: створення пристрою який буде виводити тривимірне зображення. Було обрано апаратну частину проекту, яка складається з мікроконтролеру Arduino Uno, світлодіодної матриці 8x8 MAX7219, датчику обертів, кнопки і електродвигуна с редуктором. Створено програмний код для управління мікроконтролером. Створено робочий макет проекту Розробка програмно-апаратного комплексу відображення 3D-голограми «Куб», що може бути використано для подальшого застосування на практиці.

Завдання, які треба вирішити в роботі: вивчити моделі роботи приладів для проектування голограми, зібрати матеріали як із інтернет ресурсів, так і з фізичних, проаналізувати та підібрати інструменти для розробки програмної частини, розробити технічну частину, протестувати прилад, написати пояснювальну записку до дипломної роботи, написати висновки як для всієї роботи, так і для кожного розділу, сформувати додатки до пояснювальної записки, підготувати презентацію до захисту.

ARDUINO, ARDUINO IDE, C++, ATMEGA328, FRITZING

ЗМІСТ

ВСТУП

Сучасні інформаційні технології та комп’ютерна техніка не перестають вражати різноманітністю можливостей обробки великої кількості інформації.

Вони дають нам можливість побачити об'ємні фігури предметів на приклад.

Голограми в повітрі – це голографічні моделі і 3D-піраміди. На презентаціях, конференціях, виставках та інших заходах різного рівня все частіше використовуються просторові голограми, які формуються за допомогою голографічних проекторів. Розробка 3D голограми актуальна для вивчення, тому що її просування зможе змінити світ на краще.

Сьогоднішні моделі проекторів здатні створювати безліч 3D-ефектів. У тому числі голографічні відеопроекції, створювані завдяки застосуванню прозорих плівок зворотної відеопроекції. Відеопотік, минаючи через них, робить зображення, що буквально «парить» у повітрі. У ряді нових технологій передачі інформації – відеоконференції та інтерактивна голографія, що створює ефект прозорої поверхні, що висить у повітрі.

Потенціал голографічних проекторів у міру розвитку нинішніх технологій безперервно розширюються, а якість зображень покращується. Вони стають доступнішими і компактнішими. Сьогодні на вечірках і в нічних клубах можна зустріти лазерні голографічні міні-проектори, що створюють важкі лазерні малюнки, які поєднуються з димовими ефектами.

РОЗДІЛ 1
ОПИС ТА ОГЛЯД ПРЕДМЕТНОЇ ЧАСТИНИ ДИПЛОМНОЇ РОБОТИ

1.1 Голограма

Голограма означає деякий проектований вид об'єкта або істоти, що застосовується в комунікаційних цілях. засновником голографії (від ін. ὅλος - повний і γράφω - пишу) - напрямки для розробки точного запису, трансляції та зміни хвильових полів оптичного електромагнітного випромінювання, що створюють тривимірні зображення, - неодноразово називають угорського фізика Д. Габора у 1947 р. і ввів термін «голограма» у науковий обіг. один із російських засновників голографії Ю. Денисюк зазначав, що його спонукала до вивчення об'ємних зображень у рамках розробки тривимірної голографії фантастична історія російського палеонтолога І. Єфремова «Зоряні кораблі», написана в 1946 р. Денісюк у своїх дослідженнях врахував принцип, що людина бачить не самі об'єкти, а їхні світлові образи. У нього з'явилася ідея записати на фотопластинці саме світлове поле, а після цього плоску світлову хвилю націлювати на платівку, щоб відновити вигляд об'єкта, що бракує. Запис інтерференційної зображення цього об'єкта у двох вимірах, а й у глибині з часом стала назватися методом Денисюка чи способом тривимірної фіксації. крім того застосування лазерів, вчених до створення повноцінної об'ємної голограми, наблизив дифракційний ефект Брегга, коли виникають інтенсивно засвічені площини, де і з'являється голограма. Голландський фізик Герардт Хоофт загалом висунув гіпотезу голографічного Всесвіту. Цю ідею цілком підтвердити поки що не вдалося.

1.2 Види голографії

1.2.1 Цифрова голографія

Цифрова голографія є інтерферометричною технікою візуалізації. потрібні дві хвилі для створення голограми та отримання амплітудних, а також кількісних фазових зображень одночасно. Ці дві хвилі називаються «об’єктною хвилею» та «опорною хвилею» відповідно. В цифрової голографії зазвичай використовуються два типи репрезентативних конфігурацій: позаосьовий і фазовий[1]. На малюнку 1.1 та 1.2 схематично показані ці методики. Дві лазерні світлові хвилі генеруються шляхом поділу лазерного проміння за допомогою розділювача, а один промінь лазерного світла освітлює об’єкт. Світло, що відбивається від об’єкта, є об’єктною хвилею, яка освітлює датчик зображення.

Інший промінь світла, який служить опорною хвилею, безпосередньо надходить до датчика зображення. Ці хвилі заважають одна одній, і на площині датчика зображення формується інтерференційна смуга. На рис. 1.1 напрямки оптичних осі двох хвиль відрізняються один від одного. Різниця кутів падіння між двома хвилями призводить до тонкого візерунка, і датчик зображення записує одну голограму поза віссю. На рис. 1.2 дві хвилі освітлення надходять з одного напрямку, і датчик зображення записує декілька голограм, змінюючи фазу опорної хвилі за допомогою фазового модулятора. Рисунок 1.3 схематично ілюструє процедуру реконструкції зображення цифрової голографії за допомогою методу перетворення Фур’є, який часто використовується для позаосьовой цифрової голографії[3].

Двовимірне перетворення Фур’є локалізує інформацію про хвилю об’єкта, і велика різниця кутів відокремлює об’єктну хвилю від небажаних компонентів зображення голограми в просторовій частотній області νx−νy. Завдяки фільтрації в просторово-частотній області отримується лише інформація про об'єктну хвилю. Після розрахунку 2D зворотного перетворення Фур’є видалення просторової несучої, утвореної нахилом опорної хвилі, і обчислення інтегралів дифракції,

сфокусована інтенсивність і кількісні фазові зображення отримують з однієї голограми. Для прискорення розрахунку використовується двовимірне швидке перетворення Фур'є[2]. Хоча просторові характеристики, такі як поле зору та роздільна здатність, частково приносяться в жертву для отримання інтенсивних і фазових зображень об'єкта з голограми, можна досягти однокадрового голографічного зображення. На рисунку 1.4 показаний приклад експериментальних результатів, отриманих з однієї позаосьової голограми. З однієї голограми полістиролових кульок, діаметр яких становив 5 мкм, було отримано прозорі та фазові зображення, як показано на рисунку 1.4 a і b, а 3D-відображення, показане на рис. 1.4 c, було отримано з фазового зображення.

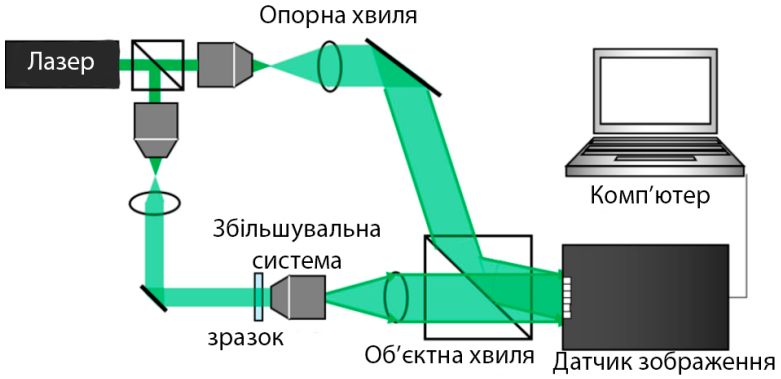


Рисунок 1.1 — Принципова схема позаосьової системи цифрової голографії

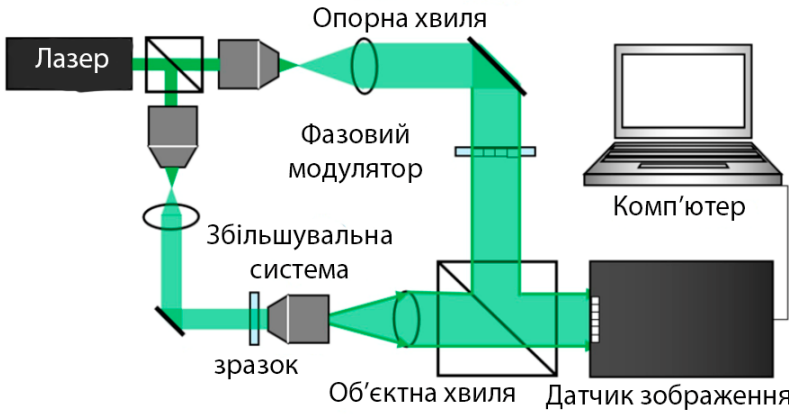


Рисунок 1.2 — Система цифрової голографії з фазозсувною інтерферометрією

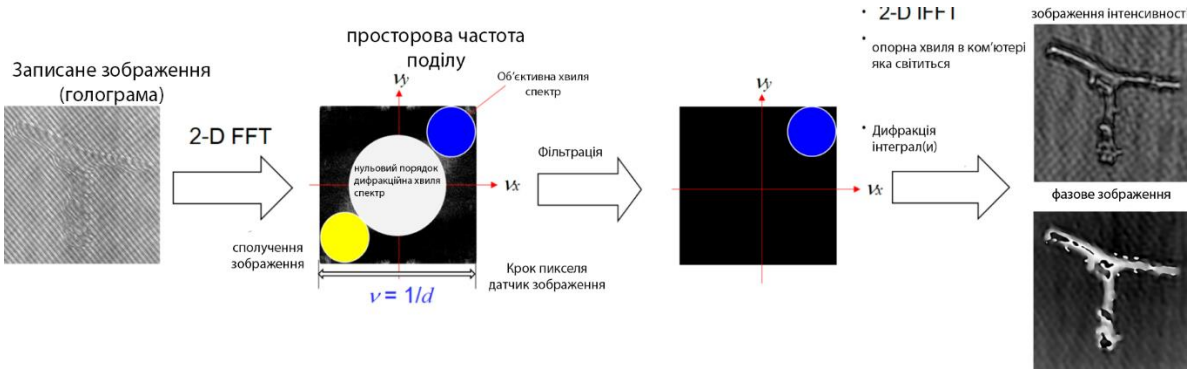


Рисунок 1.3 — Процедура реконструкції зображення в позаосьовій цифровій голографії методом перетворення Фур'є

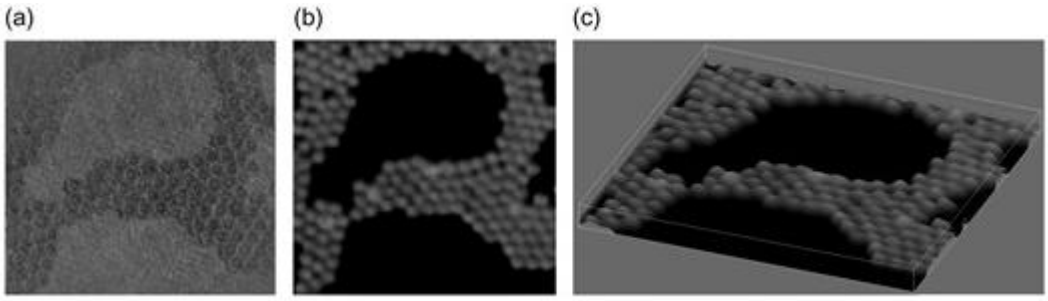


Рисунок 1.4 — (a) Інтенсивність і (б) фазові зображення полістиролових кульок, діаметр яких становить 5 мкм, які отримані за допомогою голограми на довжині хвилі 532 нм. (c) 3D-зображення, отримане з (b).

На рисунку 1.2 показано кілька голограм з різними фазами, які послідовно записуються за допомогою фазового модулятора, такого як п'єзокеровано дзеркало, акустооптичний модулятор або електрооптичний модулятор. На основі інтерферометрії зі зсувом фази витягується лише інформація про об'єктну хвилю за допомогою кількох фазових голограм[4]. Після вилучення голографічне зображення об'єкта отримують шляхом обчислення інтеграла(ів) дифракції. Незважаючи на те, що потрібен запис кількох голограм, для запису об'єктної хвилі можна використовувати повний твір пропускної здатності датчика зображення, при цьому можна отримати як широке поле зору, так і високу роздільну здатність. Для досягнення високої тимчасової роздільної здатності за допомогою інтерферометрії

зі зсувом фази, Аватсуджі та Вайант приблизно в той же час запропонували паралельну фазову цифрову голографію, яка використовує мультиплексування з розподілом у просторі множинних голограм із зсувом фази[6].

Використовуючи техніку однократного зсуву фази, можна отримати більший добуток просторової пропускної здатності, який можна записувати, порівняно з позаосьовою конфігурацією за допомогою методу перетворення Фур'є[8].

Швидкість вимірювання визначається частотою кадрів датчика зображення, швидкістю передачі даних і часом обчислення для процедури відновлення зображення.

Сучасна високошвидкісна камера має частоту кадрів 1 Мф/с, і така камера використовувалася для цифрового голографічного запису з частотою кадрів 106 кадрів/с[9]. Навпаки, висока вартість розрахунку є проблемою, яку слід вирішити. Щоб обчислити інтеграл(и) дифракції за допомогою комерційно доступного комп’ютера, потрібно набагато більше часу через вимоги до розрахунку для 2D швидке перетворення Фур'є. Використання графічного процесора (GPU) та польової вентильної матриці є можливими рішеннями для досягнення голографічної реконструкції зображення з надзвичайно високою пропускною здатністю[11].

1.2.2 Рентгенівська голографія

Голографічне зображення підходить не тільки для залишкового стану як зазвичай виконується у випадку, направлена фотоемісійна електронна мікроскопія або магнітно-силова мікроскопія, але також може використовуватися в прикладних електричних і магнітних полях і демонструє великий потенціал для вивчення динаміки мультифероїкових тонких плівок[12].

Подібно до інших методів виведення фотона з введенням фотона, рентгенівська голографія дає можливість стежити за еволюцією магнітних доменних структур і вивчати внутрішні властивості матеріалів. Як доповнення до магнітно-

силової мікроскопії або Лоренца, рентгенівська голографія має особливу перевагу в тому, що вона чутлива до тривимірного профілю намагніченості[13]. Крім того, його елементна специфічність є корисною для вивчення гетерогенних систем. Нещодавня розробка в області голографії с перетворенням Фур'є зменшила обмеження на еталонний розмір, щоб надати набагато ширший діапазон можливостей. Техніка, відома як голографія з розширеним посиланням за допомогою лінійного диференціального оператора автокореляції (HERALDO), дозволяє використовувати більші об'єкти як посилання без шкоди для просторової роздільної здатності. З моменту першої експериментальної демонстрації в 2008 році звіти HERALDO розкрили його можливості безлінзового зображення у когерентному м'якому розсіювання рентгенівських променів та надшвидкому однознімному зображенні[15].

В порівнянні з іншими методами мікроскопії, такими як рентгенівський мікроскоп з повним пропусканням поля і скануючий просвічуючий рентгенівський мікроскоп, де використовується оптика зонної пластини з високою роздільною здатністю, можна знайти в Zhu et al.[17]. На відміну від стандартного перетворення Фур'є, що використовує отвори в якості опорних точок, в HERALDO референс виходить з граничних хвиль, створених гострими кутами або краями розширеного об'єкта. В принципі, найвища можлива роздільна здатність більше не обмежується розміром еталонного зображення, а радше якістю його найбільш чітких елементів. Голограма розмножується в цифровому вигляді за допомогою диференціального фільтра, і після простого перетворення Фур'є цього результату отримується реконструкція зі складним значенням.

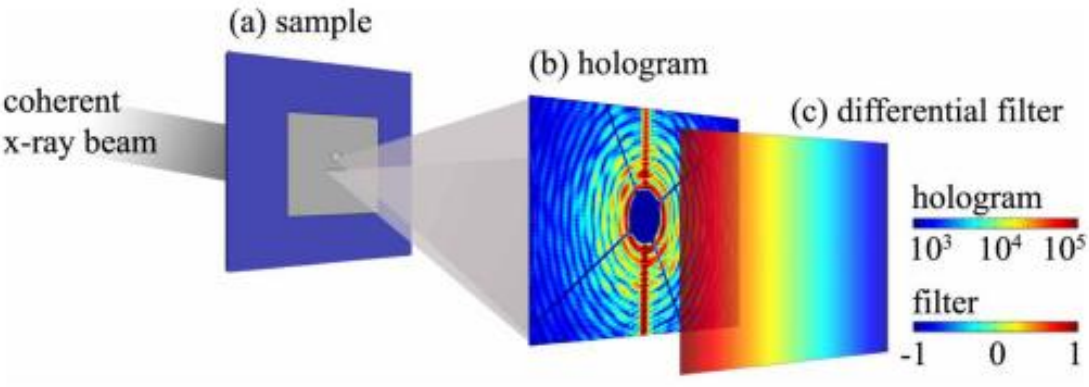


Рисунок 1.5 — Будова рентгенівської голограми

На рисунку 1.5 показано будову рентгенівської голограми (a) Об'єкт і опорна щілина освітлені когерентним рентгенівським променем. (b) Голограма, утворена інтерференцією розсіяного рентгенівського випромінювання від об'єкта та еталонної щілини, записується на ПЗЗ-камеру в дальньому полі. (c) Лінійний диференціальний фільтр, визначений похідною напрямку щілини, помножується на голограму перед виконанням перетворення Фур’є для отримання відновленого зображення.

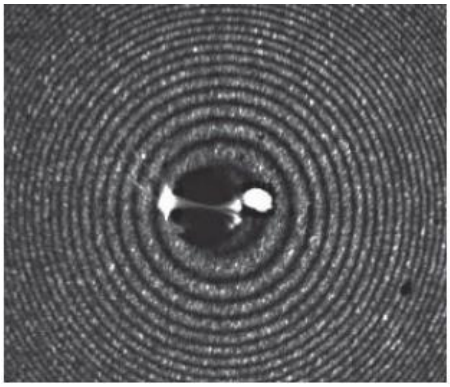


Рисунок 1.6 — Рентгенівська голограма

1.2.3 Оптична голографія

Оптична голографія — це різновид голографії, в якій заноситься світлове поле, створюване оптичним випромінюванням. Образ що отримується за допомогою

голографії, називається голограма, і вважається більш точним автостереоскопічним відтворенням зорового враження, виробленого знятими об'єктами[5]. При цьому зберігається почуття глибини простору та багаторакурсність, а зображення виглядає як вид на знятий предмет через вікно, яким служить голограма.

Принциповою відмінністю голографії від усіх інших методів реєстрації зображення є розподіляння інформації по всіх знятих предметах на всій поверхні датчика, такого, скажімо як фотопластинка. таким чином пошкодження голограми, що веде до зменшення її площі, не призводить до втрати частини зображення. кожний уламок розбитої на декілька частин фотопластинки з голограмою продовжує зберігати зображення всіх знятих об'єктів. зменшується тільки кількість доступних ракурсів, а зображення на дуже маленьких уламках втрачає стереоскопічність і чіткість[5].

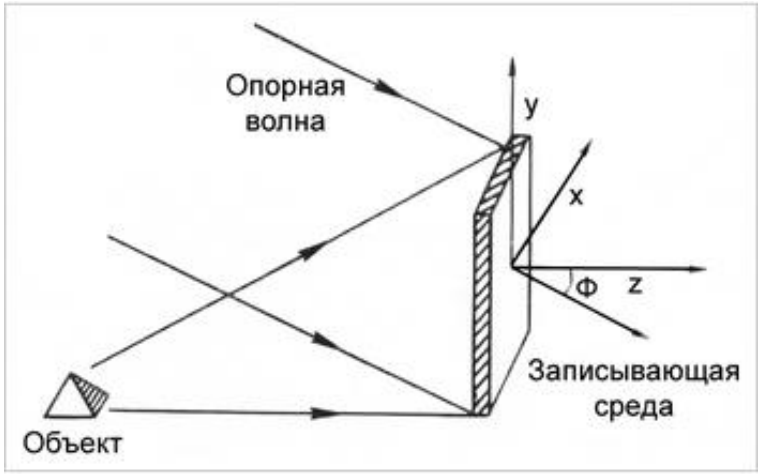


Рисунок 1.7 — Оптична голографія

Оптичні голограми поділяються на декілька типів:

2D голограма. Цей тип голограми виглядає як 2D картинка яка пофарбована чотирма основними кольорами і палітрою їх взаємних переходів. Коли відбувається зміна кута огляду змінюється її колір або відбувається зміна графічних контурів або структури.

Рисунок 1.8 — 2D голограма

2D/3D голограма. Ці голограми мають зображення яке розташовані на декількох площинах , і ці зображення розташовані одна за одною і створюють тривимірний ефект. 2D/3D-голограми найчастіше застосовується для захисту товарів, документів та цінних паперів[7].

В цих голограмах поєднуються переваги голографії такі як: вражаюча глибина; голограми, яскраві кольори; плоскі логотипи; дуже малі за розміром тексти та ін.

Тепер розглянемо голограму більш детально 2D/3D – складається з картинки яка складається з трьох пов’язаних між собою малюнків — це передній, основний та задній план. Кожен графічний малюнок це окремий плоский елемент або прозоре зображення.

Наприклад, на передньому плані може бути зображений логотип компанії; здебільшого — сам продукт; на задньому плані – ландшафт. Всі малюнки ставляться один поверх одного на відстані 5-8 мм. Така модель дає уявлення про багаторівневе зображення типової 2D/3D голограми. Нахиляючи або обертаючи таку голограму,

ви можете побачити зміну тіні, яку об'єкти кидають на більш глибокі шари зображення. Зазвичай використовують дві чи три такі площини.

2D/3D-голограми можуть бути виконані на основі кольорового малюнку чи навіть фотографії об'єкта. Малюнок повинен мати безперервні лінії та чіткі межі області кожного кольору[7].

Голографічне зображення формується оптично з допомогою лазерного зображення спеціальних голографічних установках. 2D/3D голограми добре захищаються від копіювання впроваджуючи в їх структуру спеціальні графічні елементи та оптичні ефекти. У такі голограми можна впровадити безліч захисних елементів. Створення 2D/3D-голограм вимагає особливої ретельності під час їх розробки.

Рисунок 1.9 — 2D/3D голограма

3D голограма. 3D-голограми відтворюють точну просторову копію реального об'єкта. При розгляді 3D-голограми створюється враження, що Ви бачите об'єкт через вікно, розмір якого визначається розміром голограми. Створення такої ілюзії є одним з головних завдань при розробці 3D-голограми. На голограмі об'єкт видно з

різних боків, видно реальну гру тіні і кольорів. Крім того, досягається ефект, при якому частина об'єкта як би знаходиться перед вікном - усередині приміщення. 3D-голограми записуються з реального об'єкта (масштаб 1:1) або його тривимірної моделі. Тому дуже важливо зробити точну модель та правильно виконати її зйомку[10].

3D-голограми широко використовуються в рекламних та декоративних цілях, при виконанні комплексних завдань із захисту та створення іміджу торгових марок, рідше – для захисту цінних паперів та документів. Це пов'язано з їхньою не дуже високою стійкістю до оптичного копіювання. Однак існує ціла низка технічних та художніх ефектів, що ускладнюють копіювання 3D голограм. Дані ефекти одночасно роблять їх видовищнішими.

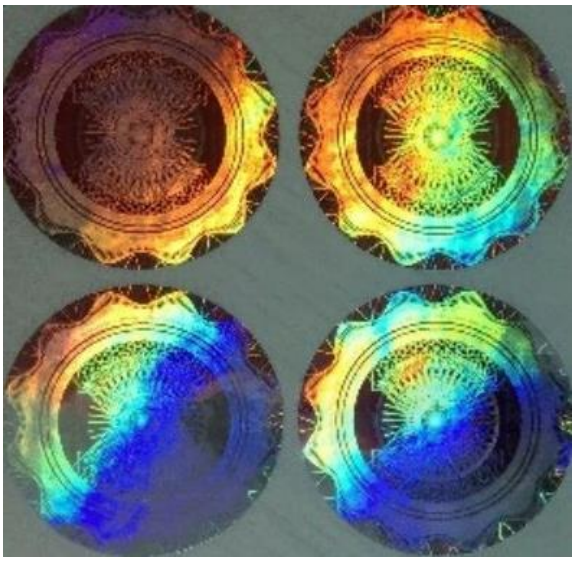


Рисунок 1.10 — 3D голограма.

1.2.4 Акустична голографія

Акустична голографія — це процес, за допомогою якого проводиться вимірювання тиску в наборі точок на поверхні в околиці акустично випромінюючий об'єкт аналізується, щоб визначити швидкість векторне поле, акустичний тиск та

інтенсивність звуку карта в тривимірному просторі. Якщо точки вимірювання є записані на поверхні, яка розташована на відстані принаймні одна довжина акустичної хвилі від активно вібруючої поверхні, потім обробка цих даних призводить до акустичного зображення, яке може вирішуйте лише джерела, розділені принаймні одним акустичним довжина хвилі. Однак, якщо точки вимірювання взято в ближнього поля джерела області, то просторова роздільна здатність може бути зменшено до частки довжини хвилі, що призводить до того, що відомо як акустична голографія ближнього поля.

Вимірювання 1130 C. I. Хайєк Поверхня називається поверхнею голограми, а площа опускання ці точки становлять отвір. Ці вимірювані тиски потім обробляються за допомогою зворотного поширення зображення поверхневий тиск структури, а також векторна швидкість та поля інтенсивності. Крім того, використовують прямі пропагатори може відобразити акустичне поле в 3D в області за межами поверхню голограми та область джерела. Потреба в ближньому полі вимірювання в межах акустичної довжини хвилі гарантує це можуть бути джерела, відокремлені менше ніж одна довжина хвилі вирішено [18].

Якщо площину вимірювальної голограми розташована за межами однієї довжини хвилі, єдині джерела, які розділені одну або кілька довжин хвиль можна розділити [20].

Вони фільтруються, оцифровуються та обробляються обчислити тиск, вектор швидкості та вектор інтенсивності поля на поверхні вібратора або в будь-якій точці тривимірного простору огородження вібраційної конструкції.

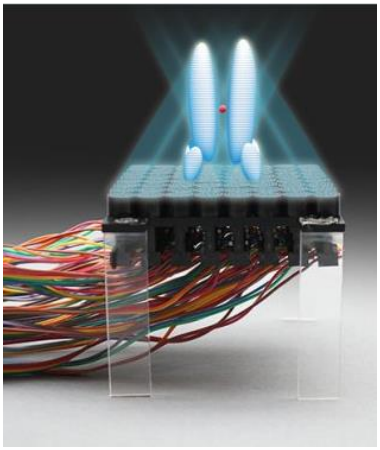


Рисунок 1.11 — Акустична голографія

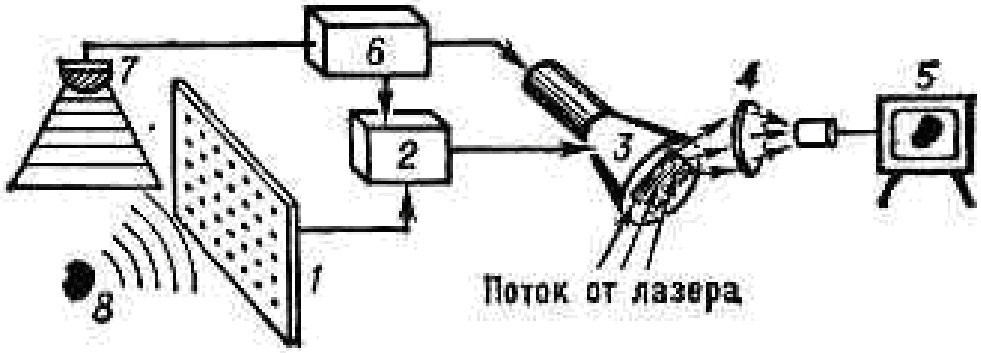


Рисунок 1.12 — Схема голографічного пристрою з матричною двомірною антеною

На рисунку 1.12 зображенні такі елементи як: 1 — антена; 2 — пристрій паралельного формування голограм; 3 - пристрій відображення голограми на трубці з мішенню з електрооптичного кристала; 4 — оптична система відновлення зображення; 5 — індикатор, що дає зображення предмета; 6 — генератор, що задає; 7 — випромінювач; 8 — предмет. Для оптичного відновлення акустичної голограми часто користуються пристроєм з приймальною антеною у вигляді двомірної матриці приймачів звуку (рис. 1.9). Електричні сигнали за допомогою комутатора модулюють силу струму електронно-променевої трубки. Електронний промінь змінює локальний коефіцієнт заломлення мішені відповідно до інтерференції картини розсіяного акустичного поля.

У подібних пристроях число приймальних елементів в антені має бути досить велике, що створює технічні проблеми за їх практичної реалізації. Описана схема використовується в основному в діапазоні звукових та низьких ультразвукових частот від 1 до 300-500 кГц. У більш високочастотний діапазон, методи реєстрації голограм ґрунтуються на носіях, які чутливі до інтенсивності звуку. Найбільшого поширення набули способи, засновані на методі поверхневого рельєфу. Звук. хвиля, що падає на поверхню води, що відображає, деформує її, формуючи рельєф, що являє собою акустичну голограму, яка при освітленні її світлом відновлює зображення (рис.1.13).

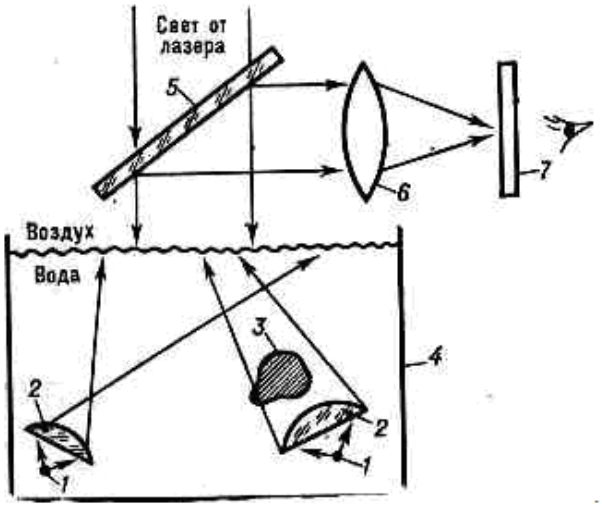


Рисунок 1.13 — Схема безлінзової ультразвукової голографії

На рисунку 1.13 зображені такі елементи як: 1 — випромінювачі; 2 —акустичні лінзи; 3 — предмет; 4 — кювета з водою; 5 — напівпрозоре дзеркало; 6 — оптична система відновлення; 7 — площина реєстрації зображення.

1.2.5 Сейсмічна голографія

Сейсмічна голографія — чисельний спосіб обробки сейсмограм, отриманих при спостереженнях, за даними розподілу амплітуд і фаз хвиль (голограм).

Відновлення сейсмічних зображень базується на спектральному аналізі сигналів та підсумовуванні гармонійних компонентів, зрушених по фазі. Використання способу доцільно при застосуванні гармонійних або періодичних джерел сейсмічних сигналів – техногенних вібрацій від гідроелектростанцій, компресорів та інших механізмів чи то з сейсмічних[14].

У сейсмології та сейсморозвідці немає необхідності вживання спец. прийомів оптичних. Сам термін, що широко застосовувався в 70-х рр. XX ст. , майже вийшов із вживання.

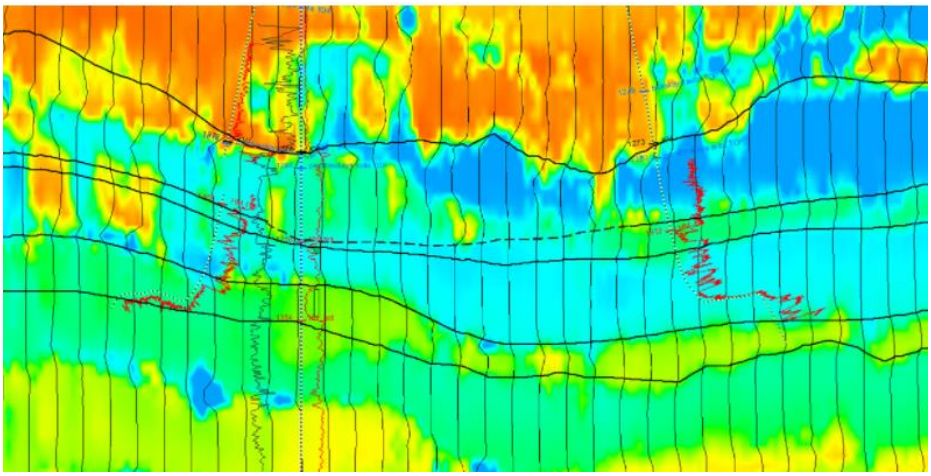


Рисунок. 1.14 — Сейсмічна голографія

1.3 Способи застосування 3D голограми

Політика. Світ сьогоднішнього політичного суб'єкта важко уявити без грамотного конструювання його віртуального образу – партії необхідно мати відомий бренд, а політичному лідеру – іміджева конструкція, орієнтована на цільову аудиторію. Те, що прийнято називати «віртуальним», рівнозначне не лише мережі Інтернет, а й голограмам.

Рисунок 1.15 — Голограма французького політика Жана-Люка Меланшона

Маркетинг та реклама. Віртуальна реальність сьогодні перестає бути чимось на межі фантастики. Не оминули 3D-технології та сферу реклами. Голографічні інсталяції як привертають увагу, а й виводять рекламу на зовсім другий рівень. Тому стандартна і нудна реклама яка не приваблює великої кількості клієнтів, буде в минулому адже сучасна людина навчилася не звертати на все це увагу, так як це його звичайне оточення.

Рисунок 1.16 — реклама годинників

Освіта. За допомогою голографії можна транслювати викладача з іншого кінця світу. для прикладу у 2015 році нобелівський лауреат та професор фізики у Стенфордському інституті Карл Віман виступив у Наньянському технологічному інституті (Сінгапур), не залишаючи США. Підготовка та настроювання голографічного монітора зайняла три тижні. А планування презентації, зокрема тестування інтернет-швидкості, — п'ять місяців.

Рисунок 1.17 — Голограма викладача

Медицина. Група медичних вчених з Інституту Калгарі витратила шість років на розробку голограми під назвою CAVEman.

За підсумками УЗІ, рентгенів, біопсій та інших медичних тестів, система здатна створювати величезні правдоподібні моделі тіла пацієнтів. 3D голограма необхідного органу дозволяє у подробицях вивчити всі особливості та уникнути лікарської помилки

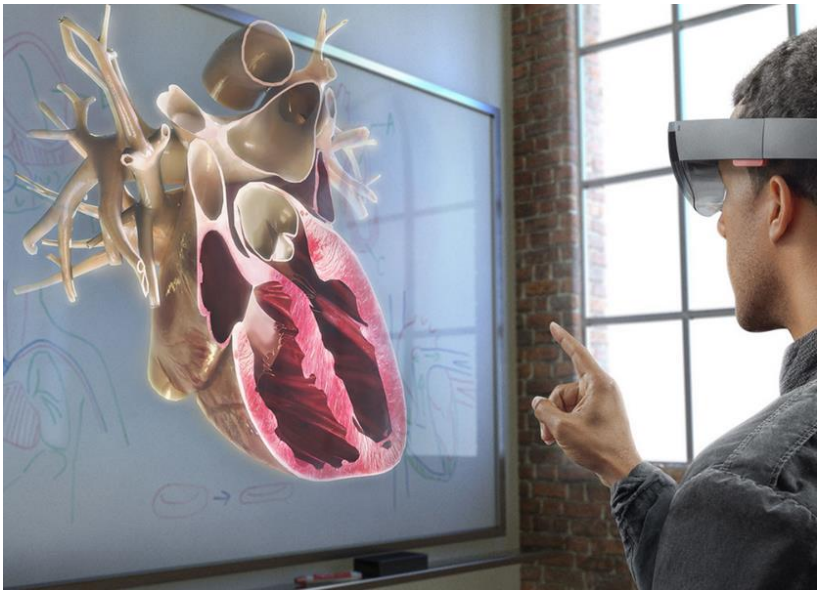


Рисунок 1.18 — Голограма серця

Розваги. 3D голограми чудовій спосіб щоб весело провести свій вільний час, наприклад як в Гонконгу для дітей було проведено голографічне шоу. На їхніх очах почали появлятися різні тварини, герої мультиків і ігор. Вони виглядали так ніби вони існують насправді що хочеться доторкнутись до них.

Рисунок 1.19 — Голограма слона

1.4 Огляд аналогів голограми

1.4.1 Голографічна піраміда

Голографічний проектор на смартфон це піраміда яка направляє промені світла які відображаються на екрані з чотирьох боків і створює одну голографічну картинку а завантажити спеціальні відеоролики для голографічної піраміди можна у мережі інтернет вони знаходяться у вільному доступі

Рис 1.20 — Голографічна піраміда для телефону

Переваги:

- Компактність;
- Виходить більш детальне зображення;
- Ціна.

Недоліки:

- Обмежений час дії;
- Маленьке зображення.

1.4.2 Голографічний проектор 3Д

Голографічний проектор це такий LED вентилятор який дуже яскравий та контрастний. На LED вентиляторі можна демонструвати у тривимірному зображенні рекламувати будь-який товар та будь-які послуги.

Голографічний вентилятор обертається з високою швидкістю. Світлодіоди, які в будовані в вентилятор, синхронізуються з частотою обертання, створюючи повноцінне тривимірне зображення протягом періодів обертання. Зображення з'являється у повітрі.

Так як у вентилятора висока швидкість обертання, ефект обертання не помітний людському оку, а супер яскраві світлодіоди формують зображення постійно переміщаючись, створюється ефект картинки, що літає в повітрі - об'єктивно псевдоголографії[16].

Рис 1.21 — Голографічний вентилятор

Переваги:

- Велике яскраве різнокольорове зображення

Недоліки:

- Ціна
- Розміри

1.5 Визначення технічного завдання

Об’єктом даної дипломної роботи є Розробка програмно-апаратного комплексу відображення 3D-голограми «Куб», який розробляється згідно заданого технічного завдання.

Дана розробка повинна мати наступні можливості:

- Вивід 3D зображення
- Керування за допомогою кнопки

1.6 Постановка задачі

Для виконання дипломної роботи на тему Розробка програмно-апаратного комплексу відображення 3D-голограми «Куб» необхідно виконати наступні задачі:

- Вивчення моделі роботи приладів для проектування голограми;
- Збір матеріалів як з інтернет ресурсів, так і з фізичних;
- Аналіз та підбір інструментів для розробки програмної частини;
- Розробка технічної частини;
- Тестування приладу;
- Написання пояснювальної записки дипломної роботи;
- Написання висновків як для всієї роботи, так і для кожного розділу;
- Формування додатків до пояснювальної записки;
- Підготовка презентації до захисту.

1.7 Висновки до першого розділу

У першому розділі було розглянуто принципи голографії та її види також було проаналізовано аналоги та сфери використання голограм у сучасному світі.

Описано структуру технологічного завдання щодо роботи донного продукту.

РОЗДІЛ 2
ВИБІР ТА ОБҐРУНТУВАННЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

2.1 Вибір мікроконтролеру та платформи

Arduino UNO виконаний на мікроконтролері ATmega328. Для роботи мікроконтролера є все необхідне для початку роботи з мікроконтролером: 14 цифрових входів/виходів (з них 6 можуть використовуватися в якості ШІМ-виходів) 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для програмування всередині схеми (ICSP) і кнопка скидання[21].

Рисунок 2.1 — Плата Arduino Uno

Щоб почати роботу з мікроконтролером нам потрібно подати живлення від комп’ютера за допомогою USB-кабелю, батарейки або AC/DC-адаптера. Кожен з 14 цифрових виводів може бути використаний в якості виходу або входу.

Рівень напруги на виводах 5 В. Гранично допустиме значення цього параметра складає 40 мА. Кожен вивід має внутрішній підтягаючий резистор опором 20-50 кОм. Резистор може бути відключений програмно.

Деякі виводи можуть виконувати додаткові функції. Послідовний інтерфейс: виводи 0 (Rx) та 1 (Tx). Використовуються для прийому (Rx) і передачі (Tx) послідовних даних логічних рівнів TTL. Також виводи під'єднані до виводів які передають данні ATmega328, яка використовуються як міст USB-UART[21]. Зовнішні переривання: виводи 2 і 3. Ці виводи їх можна використовувати як входи зовнішніх переривань.

Також їх можна програмно встановити на переривання по низькому рівню, негативному чи позитивному фронту, або на зміну рівня сигналу. ШІМ: виводи 3, 5, 6, 9, 10, 11. Які можуть працювати в режимі ШІМ модуляції з роздільною здатністю 8 розрядів. Послідовний інтерфейс SPI: виводи 11 (MOSI), 13 (SCK) 10 (SS), 12 (MISO).

Світлодіод: вивід 13. Світлодіод, підключений до виводу 13. Світиться при високому рівні сигналу на виводі. Інтерфейс TWI: SCL або SDA вивід A4 і A5. Комунікаційний інтерфейс TWI.

Arduino UNO має 6 аналогових входів, які позначаються як A0-A5. Роздільна здатність аналогового цифрового перетворення 10 розрядів. За замовчуванням, вхідна напруга вимірюється щодо землі в діапазоні 0-5 В, але може бути змінено за допомогою виведення AREF і програмних установок. Ще 2 виведення плати мають функції: AREF.

Опорна напруга АЦП мікроконтролера RESET. Низький рівень на цьому вивлду викликає скидання мікроконтролера.

Комунікаційні інтерфейси. В Arduino UNO є засоби зв'язку з іншого платою UNO з комп'ютером, або з іншими мікроконтролерами. Для цього на платі існує інтерфейс UART з логічними рівнями TTL (5 В), пов'язаний з виводами 0 (RX) і 1 (TX). ATmega328 ця мікросхема на платі пов'язує UART з USB портом комп'ютера. Коли трапляється підключення до порту комп'ютера, з'являється COM порт, він є віртуальним, через нього програми працюють з Ардуіно.

В ATmega328 прошивка використовує стандартні драйвери USB-COM і не потрібно встановлювати ніякі додаткові драйвери. Для операційної системи Windows необхідний відповідний .inf файл.

Таблиця 2.1

Технічні характеристики Arduino Uno

Основні критерії	Значення
Мікроконтролер	ATmega328
Робоча напруга	5В
Напруга живлення (рекомендований)	7-12В
Напруга живлення (граничне)	6-20В
Цифрові входи / виходи	14 (з них 6 можуть використовуватися в якості ШІМ-виходів
Аналогові входи	6
Максимальний струм одного виведення	40мА
Максимальний вихідний струм виводу 3.3V	50мА
Flash-пам'ять	32 КБ (ATmega328) з яких 0.5 КБ використовуються завантажувачем
SRAM	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактова частота	16МГц

Контролер програмується за допомогою програмного забезпечення Arduino IDE. Програмування відбувається під управлінням резидентного завантажувача по протоколу STK500. Апаратний програматор при цьому не потрібно.

Плата Arduino UNO отримує живлення від USB порту або від іншого джерела живлення. Джерелом живлення плати може виступати батарея або мережевий адаптер[21]. Підключення адаптера відбувається через роз'єм діаметром 2,1 мм.

Батарея підключається до контактів GND і Vin роз'єму POWER. Напруга джерела живлення може бути в діапазоні 6 - 20 В. Краще не допускати зниження напруги нижче 7 Вчерез те що пристрій буде не стабільно працювати. Також краще не підвищувати напругу більш ніж 12 В, тому що може перегріється стабілізатор і вийти з ладу. Тому слід звернути свою увагу на діапазон напруги живлення 7 - 12 В цей діапазон є рекомендованим. Для того щоб підключити живлення потрібні наступні виводи:

- Vin - Живлення плати від зовнішнього джерела живлення. Чи не пов'язано з живленням 5 В від USB або виходами інших стабілізаторів. Через цей контакт можна отримувати живлення для свого пристрою, якщо плата харчується від адаптера.
- 5 V - Вихід стабілізатора напруги плати. На ньому напруга 5 В при будь-якому способі живлення. Живлення плати через цей висновок не рекомендується, тому що не використовується стабілізатор, що може привести до виходу мікроконтролера з ладу.
- 3 V 3 - Напруга 3,3 В від стабілізатора напруги на платі. Має граничний допустимий струм живлення 50 мА. GND - Загальний провід.
- IOREF - На виведення інформація про робочій напрузі плати. Плата розширення може вважати значення сигналу і переключитися на режим живлення 5 В або 3,3 В.

У мікроконтролера три типи пам'яті:

- 32 кБ флеш (FLASH); – вона використовується для зберігання програми.

Якщо контролер прошивається через USB скетчем. Тоді пам'ять Arduino очищується, слід потрібно лише завантажити порожній скетч.

- 2 кБ оперативної пам'яті (SRAM); В цій пам’яті зберігаються об'єкти та змінні, що створюються в скетчі. Оперативна пам’ять вона енергозалежна, при відключенні плати Uno, всі дані будуть видалені.
- 1 кБ незалежній пам'яті (EEPROM); В цю пам’ять можна записувати данні, які при вимкненні живлення Uno не зникнуть. Мінусом EEPROM є те що в неї обмежена кількість циклів перезапису - 100 000 разів за твердженнями виробника

Рисунок 2.2 — Структура мікроконтролера

2.2 Переривання в Arduino

В Arduino немає операційної системи, і тому стає питання про багатозадачність контролеру. Тому щоб вирішити цю проблему використовують методику переривання[24]. Це можуть бути переривання апаратні або програмні.

Під час роботи процесор виповнює певні операції, а переривання зупиняє всі ці процеси якими займався процесор і заставляє передати управління обробнику переривання (ISR, Interrupt Service Routine). Оброблювач це така функція яка пишеться вже самим користувачем який поміщає туди код який буде відреаговувати на подію.

Після того як переривання закінчується процесор відновлює свою роботу[23]. Це відбувається автоматично, і тому користувачеві потрібно лише писати обробник переривання і робити так щоб не заставляти процесор занадто часто робити переривання.

Для того щоб працювати з перериваннями потрібно знати як часто можна викликати переривання його особливості а також потрібно розуміти схеми і як працюють прилади які підключенні до плати.

Види переривань:

- Апаратні переривання це переривання які відбуваються на рівні мікропроцесорної архітектури. Під час роботи процесора відбувається якась зовнішня дія. Наприклад йде сигнал вид якогось датчику або натиснули кнопку чи ще щось і коли відбувається ця подія запускається в програмі переривання за допомогою спеціальної інструкції.
- Програмні переривання це переривання які ініціюються інструкцією в коді програми. Ці переривання зазвичай використовують для функцій вбудованого ПО.

В Arduino є 4 види переривань і це:

- Заземлений контакт й переривання працює до тих пір поки на буде поступати сигнал LOW.
- Зміна сигналу. Коли відбувається зміна сигналу на піні тоді обробник переривання починає працювати.
- Відбулась зміна сигналу на піні з низького сигналу на високий тоді відбувається переривання.
- Зміна сигналу від високого до низького буде відбуватися переривання.

Переривання потрібні в Arduino для того щоб вирішити проблему з синхронізацією[23]. Наприклад прилад який виводить якісь символи подає сигнал переривання на процесор а він в свою чергу викликає переривання який захоплює символ. І це дозволяє економити час процесора тому що без переривання витрачалося би процесорний час на те щоб перевірити статус UART.

Тому переривання займається всіма необхідними речами не зачіпаючи саму програму

Основними причинами, через які необхідно викликати переривання, є:

- Визначення зміни стану виводу;
- Переривання за таймером;
- Переривання даних з SPI, I2C, UART;
- Аналогово-цифрове перетворення;
- Готовність використовувати EEPROM, флеш-пам'ять.

Функція attachInterrupt використовується для роботи з перериваннями. Вона служить для з'єднання зовнішнього переривання з обробником. Синтаксис виклику: attachInterrupt(interrupt, function, mode)

Аргументи функції:

- interrupt – номер викликаного переривання (стандартно 0 – для 2-го піна, для плати Ардуїно Уно 1 – для 3-го піна),
- function – назва викликаної функції при перериванні (важливо – функція не повинна приймати, ні повертати будь-які значення),
- mode – умова спрацьовування переривання.

2.3 LED матриця 8x8

Світлодіодна матриця це компактний графічний модуль який побудований на базі мікросхеми MAX7219 та має просте підключення, виводить прості зображення. В цьому модулі є п’ять виводів з двох сторін. З однієї сторони данні входять а з іншої виходять в інший модуль. Це дозволяє з’єднати матрицю у ланцюг[25].

Рисунок 2.4 — Плата світлодіодної матриці

Світлодіодна матриця це модуль з мікросхемою, яка необхідна для неї та, власне, також сам матричний індикатор. Зазвичай цей індикатор не впаюється в плату, а просто вставляється у гніздо. Зроблено це так, щоб його було можна спочатку закріпити на будь-якій поверхні, аж потім вставити матрицю.

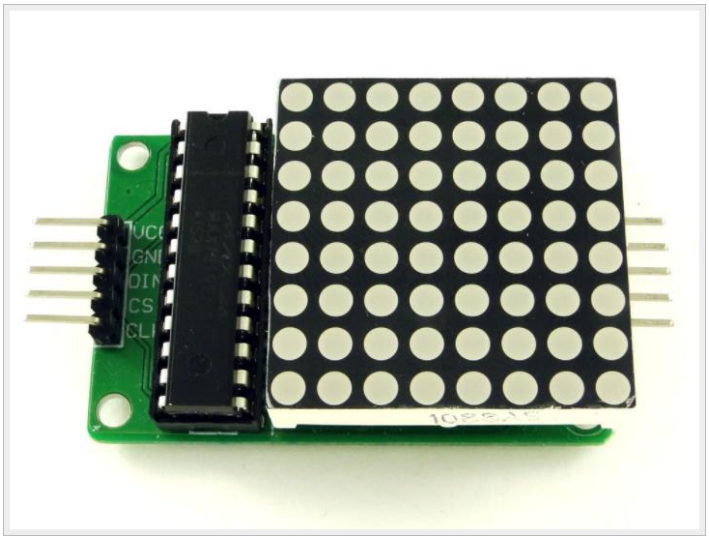


Рисунок 2.5 — Світлодіодна матриця

Характеристики світлодіодної матриці:

- Розмірність матриці: 8x8
- Колір світлодіодів: червоний
- Тип матриці: загальний катод
- Драйвер: MAX7219
- Робоча напруга: 5 В
- Розміри: 3.2 см X 3.2 см X 1.3 см
- Кріплення: 4 отвори діаметр 3 мм
- Інтерфейс: SPI

Виводи:

- VCC - напруга живлення 5 В
- GND - загальний
- DIN – вхід даних
- CS - Дозвіл завантаження даних
- CLK - стробуючі імпульси

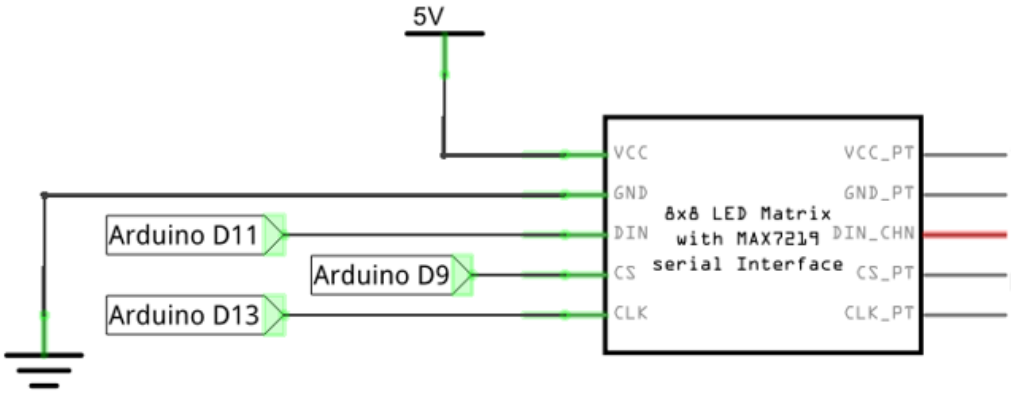


Рисунок 2.6 — Принципова схема LED матриці 8x8

2.4 Електродвигун с редуктором JGA25-370

Це прилад, який виконаний у вигляді агрегату, що складається з редуктора та електродвигуна, які з'єднані між собою проміжною муфтою або без неї. Але як елемент електроприводу, має широке застосування у багатьох галузях промисловості; його переваги це - високий ККД, простота обслуговування, компактність а ще простота монтажу[26].

Рисунок 2.7 — Електродвигун с редуктором JGA25-370

Характеристики:

- Напруга живлення: від 6 до 12В;
- Довжина валу: 9мм;
- Обертаючий момент 1200 об/хв;
- Діаметр валу: 4мм.

2.5 Датчик обертів

Датчик обертів використовується для того щоб виміряти швидкість обертання за допомогою диска з прорізами або датчиком перешкоди. Принцип його дії це визначити об'єкт який проходить між прорізами. Цей модуль також можна використати як високоточний кінцевий датчик положення (ENDSTOP)[27]. На платі розташований пороговий компаратор напруги для отримання вихідного сигналу.

Він відрізняється від інших аналогічних датчиків своїми невеликими розмірами.

Рисунок 2.8 — Датчик обертів

Виводи:

- VCC: + харчування;
- GND: Загальний;
- DO: Цифровий вихід;
- AT: Аналоговий вихід.

Характеристики:

- Напруга живлення: 3,3 - 5В;
- Ширина паза датчика: 5 мм; ;
- Тип виходу: аналоговий та цифровий;
- Використовуваний компаратор: LM393.

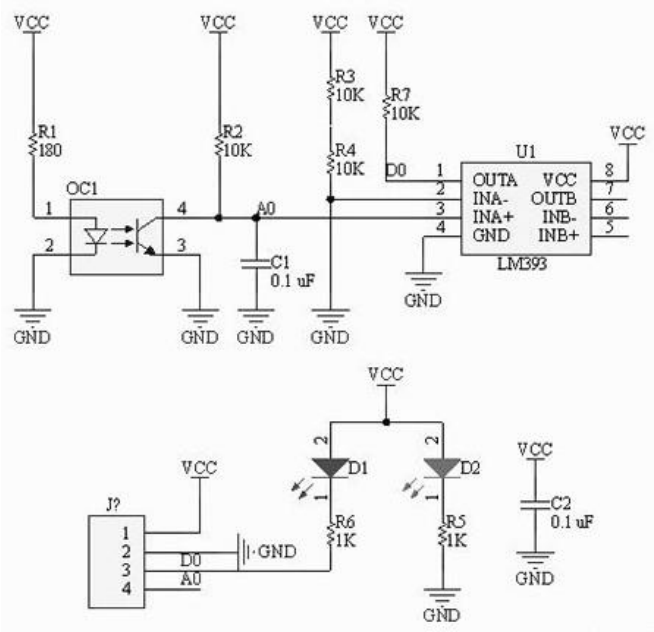


Рисунок 2.9 — Принципова схема датчика обертів

2.6 Блок живлення

Рисунок 2.10 — Блок живлення YN 207

Блок живлення це напівпровідниковий прилад який потрібний для створення необхідних параметрів напруги для правильної роботи приладу.

Сам блок не є основним джерелом живлення приладу а всього на всього вторинним джерелом тому що він не може виробляти електроенергію з іншого виду енергії а всього лише змінює параметри поступаючиго в нього струму.

Найчастіше побутові блоки живлення які знижують напругу з 220В до 5В, 9В, 12В та 24В.

Характеристики:

- Вхід 220V 50/60Hz;
- Вихід від 3В до 12В.

2.7 Кнопка

Майже усі радіотехнічні прилади мають компоненти які контролюють подачу живлення або його зупиняє. Це зазвичай тумблери, кнопки або ракетні перемикачі.

В багатьох приладах використовують кнопку. Кнопка представляє собою невеликий прилад якого тільки одна роль це подача живлення або його зупинення до окремих елементів або усього пристрою.

Рисунок 2.11 — Кнопка

Характеристики:

- Максимальний ток 1.5 А;
- Максимальна напруга 250В;
- Механічна порочність 10 000 натискань і вище;
- Контактний спротив <=0.03 ом;
- Робоча температура: -25-85 ºС.

2.8 Засоби проектування та модулювання

Fritzing — це безкоштовне програмне забезпечення має ліцензією GPL 3.0 з відкритим вихідним кодом та двійковими файлами для розробки систем автоматизованого проектування. Програмне забезпечення було розроблено в Університеті прикладних наук Потсдама[28].

На розвиток Fritzing вплинули мова програмування Processing та система Arduino[29]. В програмі Fritzing можна перетворити прототип на основі Arduino на топологію друкованої плати для серійного виготовлення. На офіційному сайті Fritzing можна не лише скачати програму а ще ділитися своїми проектами та обговорювати їх та обмінюватися досвідом с іншими користувачами.

Програма сильно спрощує створення та налагодження схем за допомогою макетних плат. Ця програма була написана на мові C++ із застосуванням графічного стеку Qt.

Рисунок 2.12 — Меню елементів Fritzing

У цьому меню показана велика кількість компонентів які можна використовувати для ваших майбутніх або існуючих проектів. Також існують бібліотеки для модулів.

Бібліотека Arduino має 22 різні дошки та щити Arduino, які можна вибрати ту яка вам потрібна[28]. Щоб скористуватися модулями потрібно лише перетягнути ці модулі на вигляд макета, а потім почати підключення вхідних і вихідних штифтів до компонентів на макеті[29]. Після того, як ви під'єднали усі елементи схеми у вас з'явився ланцюг, підключений до макета, і тоді можна вже переглянути готову схему. Є в цій програмі великий мінус і це при роботі над проектом ви зіткнетесь зі скупченням елементів на розстановку яких витрачається багато часу. Після того як ви провили обробку, перевірку та можливо модифікацію вашого проекту, потім можна відправити свою схему в організацію Fritzing та зробити друковану плату. Ціни звісно не маленькі, але вони можуть працювати безпосередньо з файлу Fritzing, створеного програмою.

Веб-сайт Fritzing пропонує ряд послуг, щоб допомогти вам почати роботу та дає вам можливість поділитися своїми дизайнами. Ці послуги включають в себе

серію покрокових навчальних посібників, які допомагають користувачам дізнатись, як використовувати інструменти, а якщо у вас з’явилися якісь питання, ви можете звернутися на форуми, де ви можете задавати питання та отримати потрібну вам допомогу[28]. Fritzing має деякі обмеження, але це хороший інструмент для документування та створення прототипів простих схем, особливо тих, що використовуються в Arduino.

В програмі Fritzing ви можете побачити 3 види відображення макетування, схема та третій вид це вид плати. Ви можете намалювати свою схему як макетній платі так і на принциповій схемі. Друга схема буде будуватися автоматично але її потрібно буде редагувати тому що програма автоматично розставить елементи як попало і на ваша схема буде виглядати заплутано.

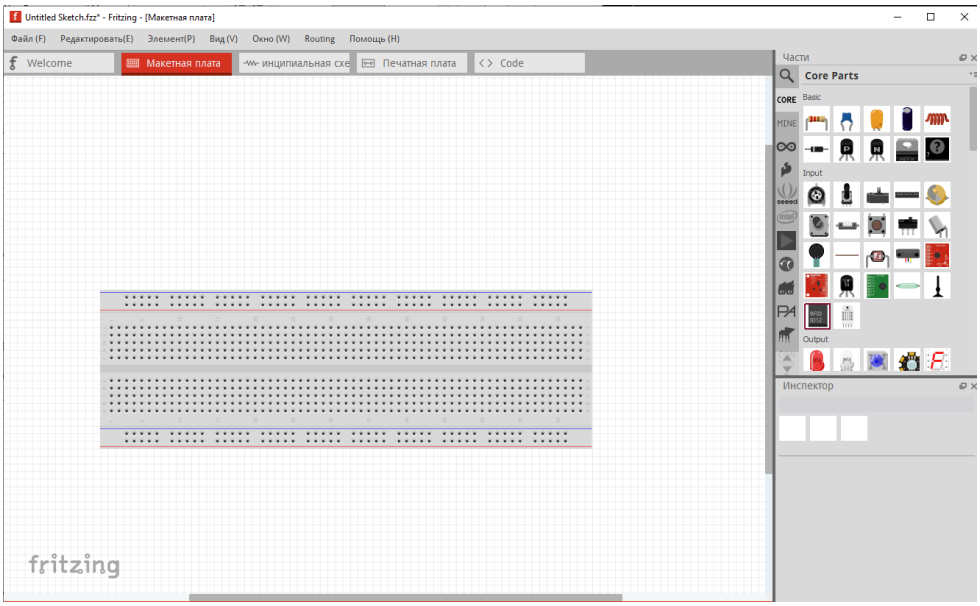


Рисунок 2.13 — Макетна плата

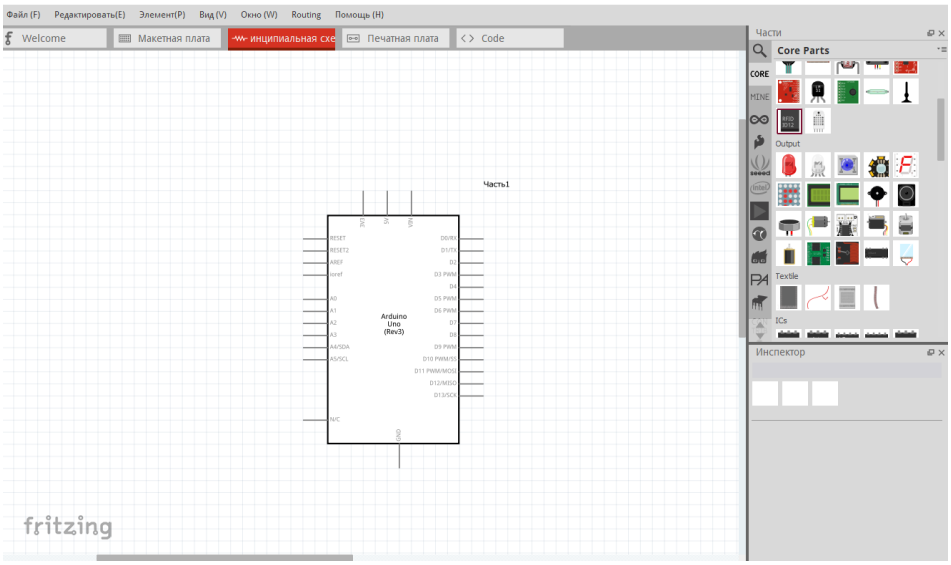


Рисунок 2.14 — Принципова схема

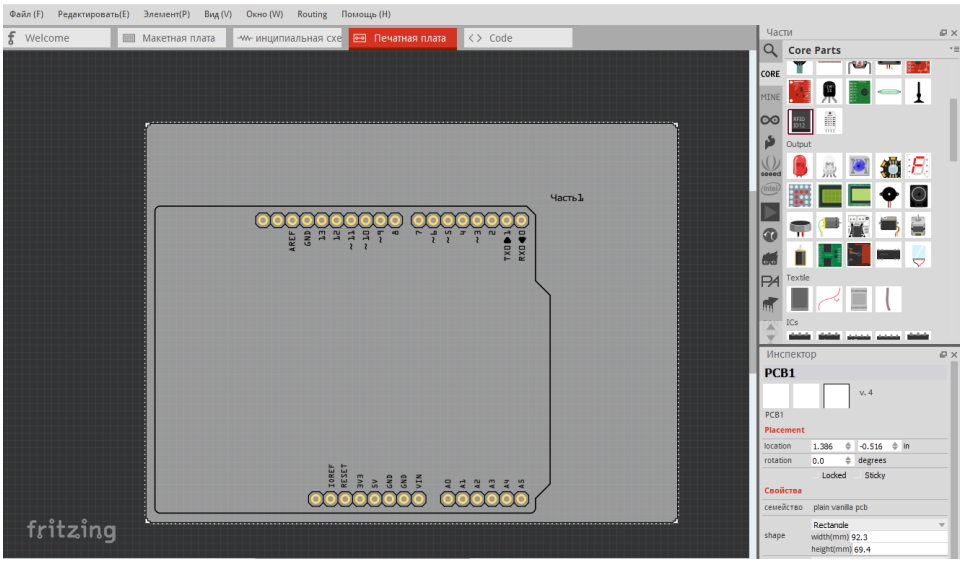


Рисунок 2.15 — Печатна плата

2.9 Вибір середовища розробки та мови програмування

Дуже важливим є питання що до вирішення та використання програмних засобів для того щоб спроектувати систему та написати відповідні набори команд для прошивки платформи. Тому розглянемо основні програмні компоненти нижче.

2.9.1 Середовище розробки

Arduino IDE — це платформа яка дає змогу новачкам і професіоналам швидко розробити програму для електронних пристроїв. Arduino має велику популярність тому що у неї проста та зручна мова програмування, а також її архітектура і програмний код знаходиться у вільному доступі в мережі Інтернет.

Arduino дає змогу комп'ютеру взаємодіяти з фізичним світом. Пристрої на базі Arduino отримують інформацію про навколишнє середовище за допомогою різних датчиків, а також управляють різними виконавчими пристроями[30].

Микроконтроллер програмується за допомогою мови Arduino (заснований на мові Wiring) і інтегрованого середовища розробки Arduino IDE (заснована на середовищі Processing)[30]. Проекти які були засновані на Arduino, взаємодіють з програмним забезпеченням на комп'ютері (наприклад, Processing, MaxMSP, Flash) або можуть працювати самостійно. Плата може бути зібрана самостійно або куплена вже у повному зборі.

В склад Arduino IDE входить:

- Вбудований текстовий редактор програмного коду;
- Компілятор;
- Відладчик (підтримує тільки плати Arduino на базі SAMD і Mbed);
- Модуль передачі прошивки в плату;
- Області повідомлень;
- Вікна виведення тексту (консолі);
- Панелі інструментів з кнопками часто використовуваних команд і декількох меню.

Для того щоб завантажити програму, в середовище розробки потрібно під’їдатися до Arduino. Скетч це програма, яка пишеться в текстовому редакторі, що має інструменти вирізки / вставки, пошуку заміни тексту і написана в середовищі

Arduino IDE. Коли проект зберігається і експортується в області повідомлень можуть з'являтися пояснення або помилки.

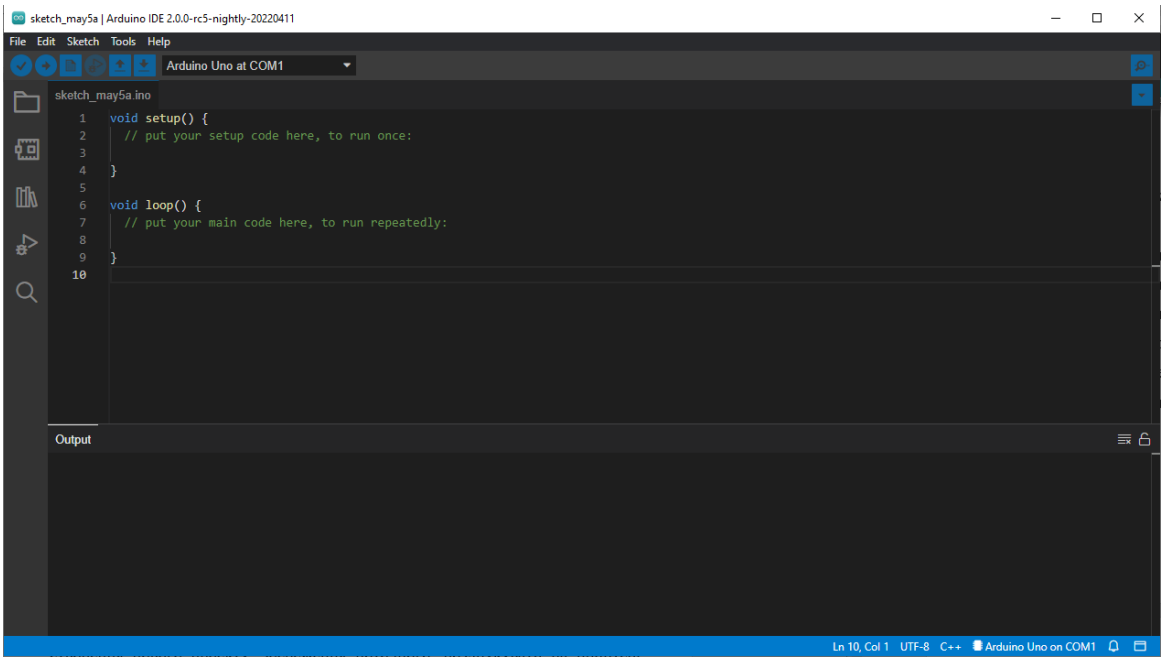


Рисунок 2.16 — Інтерфейс Arduino IDE

На початку роботи з Arduino IDE можна знайти такі елементи як:

- Меню програми;
- Панель швидкого доступу до найбільш важливих функцій;
- Редактор (для розміщення коду програми);
- Панель повідомлень і статусу програми.

В меню програми ми управляємо проектом, можна зберегти або створити новий проект також можна роздрукувати код.

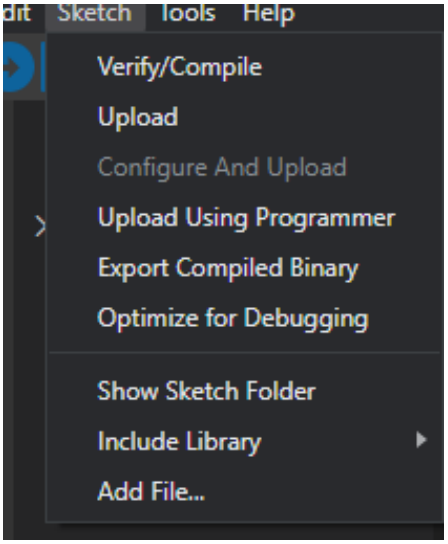


Рисунок 2.17 — Меню «Скетч»

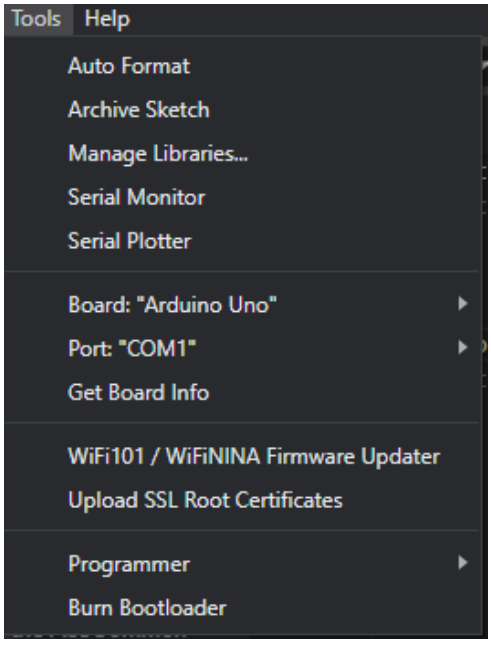


Рисунок 2.18 — Меню «Інструменти»

Важливий елемент в програмі є меню «Інструменти» вона дає можливість вибрати потрібну вам плату. У представленому вам списку знаходяться всі офіційні версії Arduino, також можна вибрати порт, до якого підключений Arduino. Зазвичай Arduino IDE сам визначає порт, але іноді потрібно вручну вибрати порт. Але в новій версії Arduino IDE 2.0.0 можна і не заходити меню «Інструменти» щоб вибрати плату

і порт до якого вона підключена, тепер це можна зробити у головному меню програми[30].

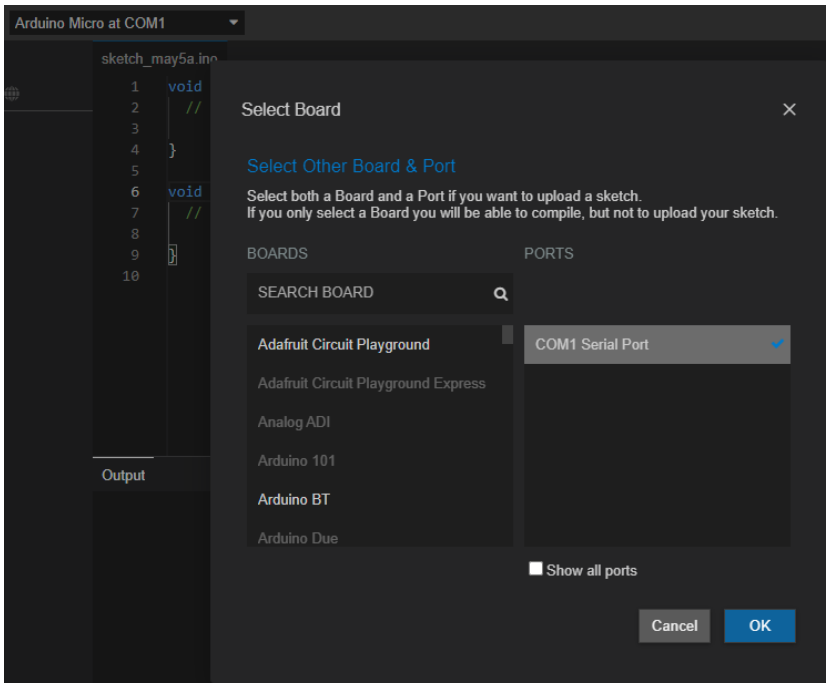


Рисунок 2.19 — Вибір плати та порту

Для того щоб нормально працювати в Arduino IDE потрібно використовувати панель швидкого доступу, вона оснащена важливими кнопками.

Це робить роботу у програмі значно легше, і надає доступ до всіх необхідних параметрів для написання і тестування програми.

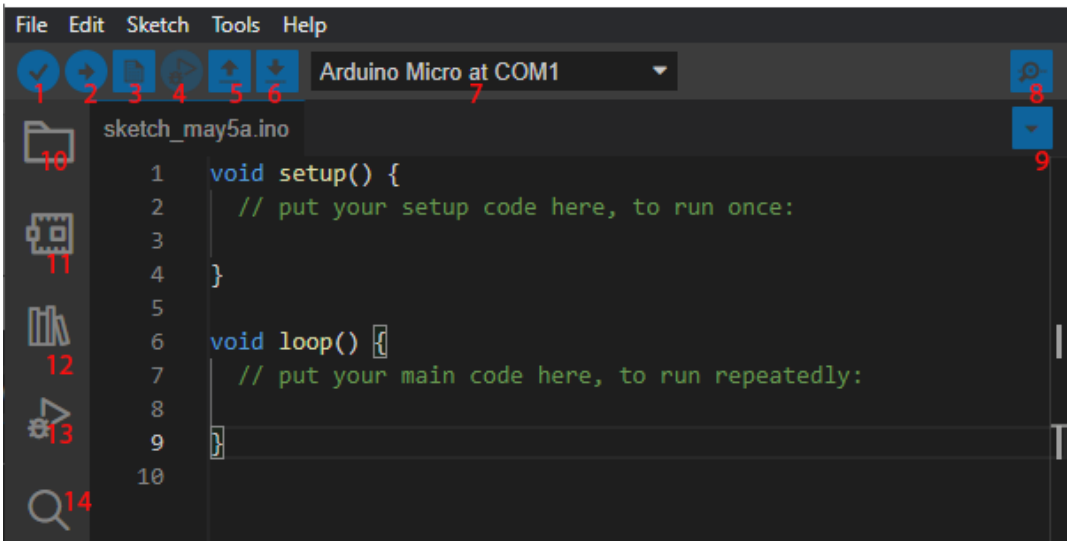


Рисунок 2.20 — Панель швидкого доступу

Панель швидкого доступу дозволяє скористуватися такими функціями як:

- Компілятор;
- Завантажувач програми в мікроконтролер;
- Новий проект;
- Відладчик;
- Відкрити існуючий проект;
- Зберегти проект;
- Вибір плати та порту;
- Монітор послідовного порту;
- Вікно повідомлень і статусу;
- Скетчбук;
- Менеджер плат;
- Менеджер бібліотек;
- Відладчик з детальною інформацією;
- Пошук.

Всі вище перелічені опції, вони розташовані на панелі швидкого доступу, та продубльовані в меню програми.

Arduino IDE в порівняні з її аналогами такими як WinAVR, AVR Studio, добре підходить для новачків, які тільки починають розбиратися в програмуванні та схемотехніці. В самій IDE є дуже багато різних бібліотек розташованих на вершині AVR LibC також має розумний інтерфейс і значно спрощує створення/завантаження.

Платформа Arduino в основному спрощує те, що ви хочете зробити зі своєю схемою, а не займатися технічними тонкощами використання самого мікропроцесора.

2.9.2 Вибір мови програмування

Щоб писати команди для мікроконтролера AVR можна скористуватися Assembler та GCC C/C++. Але коли використовуєш IDE то потрібно використовувати мову Processing яка має компілятор GCC C, який інтегровано до відповідного середовища.

Мова програмування яка використовується для пристроїв Arduino заснована на C/C++ і скомпонована з бібліотекою AVR Libc яка дозволяє використовувати будь-які її функції.

2.9.3 Необхідні блоки скетча

`void setup()` виконується лише один раз і вона приводить систему у робочий стан, сюди зазвичай записують функції ініціалізації портів і периферійних пристрої.

`void loop()` це цикл який повторюється знов і знов і так до нескінченності, це основний цикл програми, в якому виконується основна частина.

Функція − це одиниця програми, вона має ім'я і містить в собі деяку послідовність дій. Багато команд в коді закінчуються ";", виключенням є оператори

циклів, вибору, умов і опис прототипу функції. На початку програми, перед функцією `setup`, оголошують змінні і визначення. При включені плати перша функція яка виконується це `setup`. Ця функція виконується один раз і в ній не започатковано режими роботи портів: порти, до яких підключені різні датчики, встановлюються як входи, а порти з виконавчими пристроями як виходи.

2.10 Висновок за розділом

У другому розділі дипломної роботи була обрана мова програмування C++ і були детально описані програми за допомогою яких був написаний код програми і була змодельована технічна частина дипломного проекту. А саме були використані такі програми як: Arduino IDE, Fritzing.

Також була розглянута технічна частина в якій детально були описані наступні деталі які були використані в технічній частині дипломного проекту: Мікроконтролер Arduino UNO, Світлодіодна матриця, Мотор с редуктором, Блок живлення 3-12В, Датчик обертів, Кнопка.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Апаратна частина проекту

Збірку проекту було розпочато з корпусу на якому будуть розміщатися необхідні модулі, які потрібні для роботи проекту.

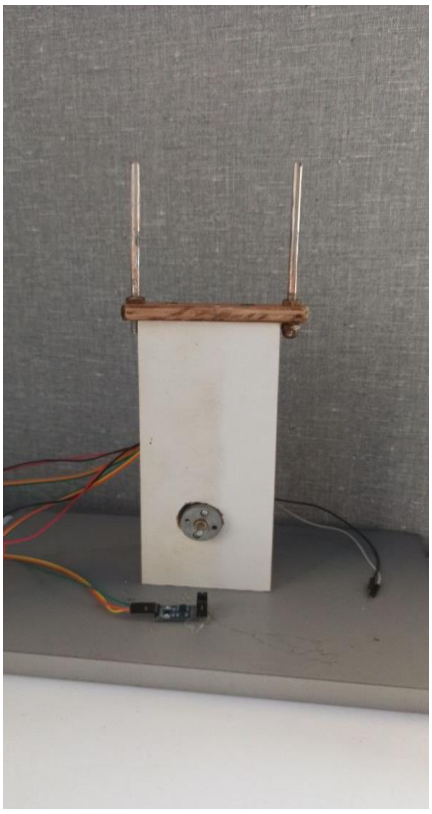


Рисунок 3.1 — Передня частина корпусу

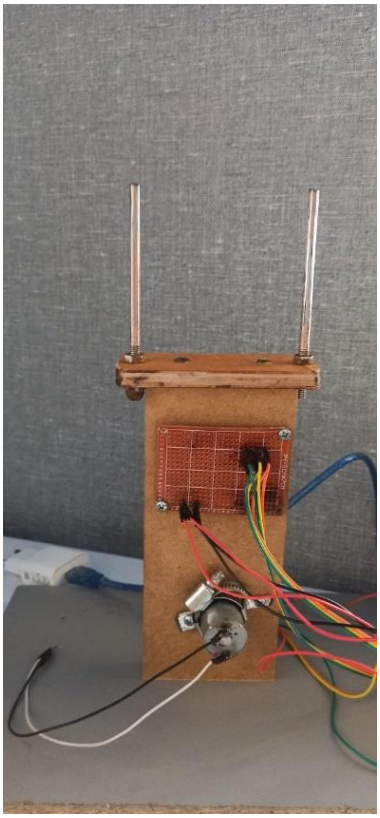


Рисунок 3.2 — Задня частина корпусу

А також потрібно звернути увагу на механічну частину проекту, а точніше на кривошип, який штовхає дванадцяти вольтовий моторчик, а той в свою чергу рухає в гору та низ світлодіодну матрицю яка виводить зображення. Таким чином при швидкому русі матриці в верх та низ зображення розмивається та становиться об’ємним.

Іншими словами це зображення яке виходить із двовимірного об’єкту в тривимірний за рахунок руху.

Рисунок 3.3 — Кривошип із матрицею

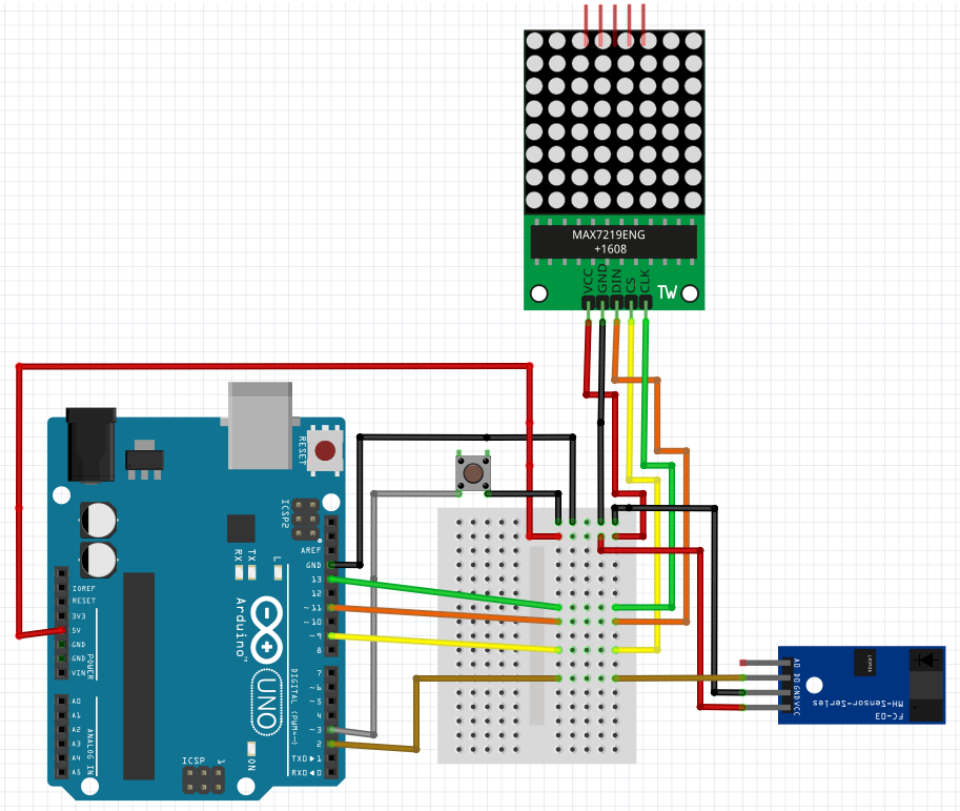


Рисунок 3.4 — Схема підключення модулів до Ардуіно

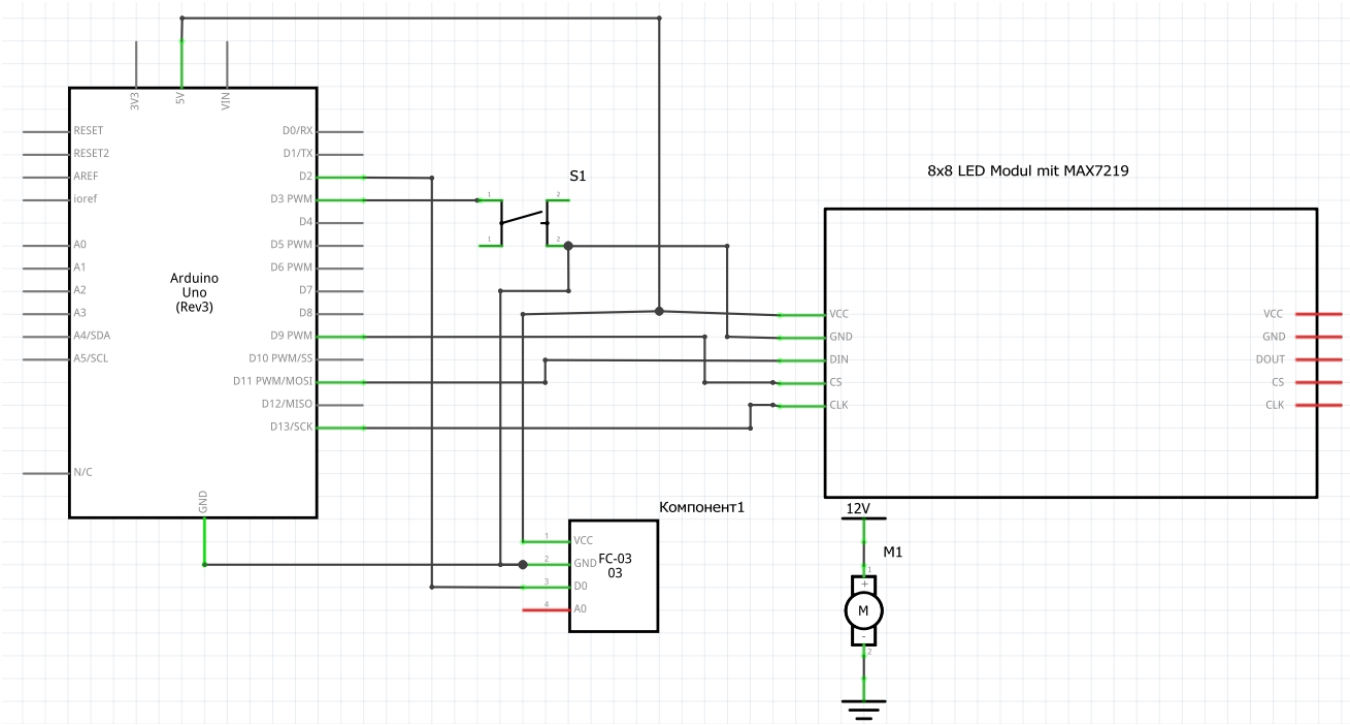


Рисунок 3.5 — Принципова схема проекту

На принциповій схемі показано підключення модулів до плати Arduino Uno за наступним розведенням входів/виходів:

- D2 – під’єднаний до D0 оптичного датчику обертів;
- D3 – під’єднаний до входу кнопки;
- D9 – під’єднаний до входу CS світлодіодної матриці;
- D11 – під’єднаний до входу DIN світлодіодної матриці;
- D13 – під’єднаний до входу CLK світлодіодної матриці;
- Мотор під’єднаний окремо до блоку живлення.

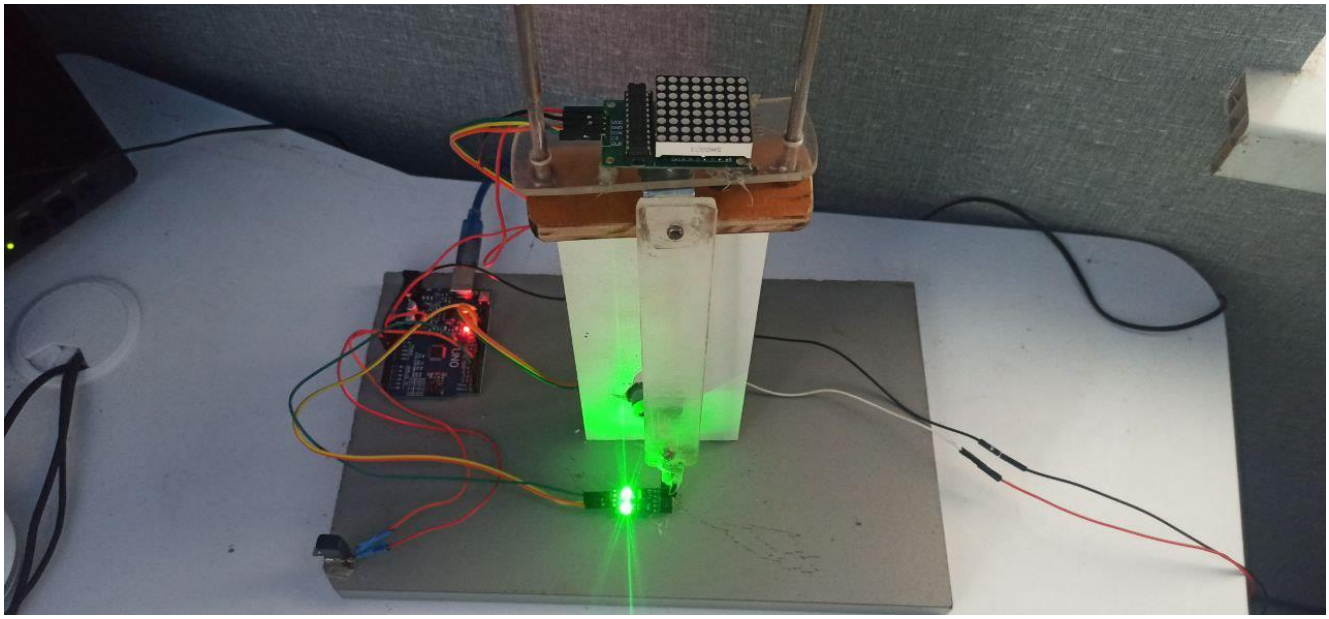


Рисунок 3.5 — Зібраний проект у дії

3.2 Програмна частина

Лістинг 3.1 – алгоритм керування матрицею

```
//функция для работы с матрицей через шину SPI
void sendCMD(uint8_t address, uint8_t value) {
  //Подает LOW значение на цифровой вход
  digitalWrite(CS_PIN, LOW);
  //трансфер байтов по шине SPI
  SPI.transfer(address);
  SPI.transfer(value);
  SPI.transfer(address);
  SPI.transfer(value);
  //Подает HIGH значение на цифровой вход
  digitalWrite(CS_PIN, HIGH);
}
//установка яркости
void setBright(byte value) {
  sendCMD(0x0a, value);  // яркость 0-15
}
void sendDATA(uint8_t address, uint8_t value) {
  digitalWrite(CS_PIN, LOW);
  SPI.transfer(++address);
```

```
  SPI.transfer(value);
  digitalWrite(CS_PIN, HIGH);
}
//очистка матрицы
void clearMatrix() {
  for (byte i = 0; i < 8; i++) {
    sendDATA(i, 0);
  }
}
```

3.3 Висновок за розділом

В третьому розділі була проведена робота з механізмом який виводить трьох вимірне зображення за допомогою поступальних рухів, та була проведена робота з коректуванням всіх помилок коду та механізму виведення зображення

ВИСНОВОКИ

Під час роботи над дипломною роботою було досягнуто основної мети а саме, був створений пристрій який виводить тривимірне зображення.

Під час виконання роботи були поставлені наступні задачі які були успішно виконані: було вивчено моделі роботи приладів для проектування голограми, зібрано матеріали як з інтернет ресурсів, так і з фізичних, проаналізовано та підібрано інструменти для розробки програмної частини, був написаний код на мові C++, розроблено технічну частину, проведено тестування приладу, була написана пояснювальна записка дипломної роботи, написані висновки як для всієї роботи, так і для кожного розділу.

Для технічної частини проекту були розглянуті і описані наступні деталі які були використані в дипломному проекті: Мікроконтролер Arduino UNO, світлодіодна матриця, мотор с редуктором, блок живлення 3-12В, датчик обертів, кнопка. В третьому розділі була проведена робота над проектом а саме була розроблена принципова схема проекту, сам проект і також була створена програмна частина.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гудман Дж. В. та Лоуренс Р. В. (1967) Формування цифрового зображення з електронно виявлених голограм [Текст] / М. Гудман Дж. В. та Лоуренс Р. В. (1967). 77-79 с.

2. Poon T C, та Liu J P (2014) Введення у сучасну цифрову голографію за допомогою
MATLAB (видавництво Кембриджського університету, Нью-Йорк). [Текст] / Poon T C, та Liu J P письма по електроніки (2014). 179с.

3. Takeda M, Ina H, і Kobayashi S (1982) Метод перетворення Фур'є аналіз шаблонів
для комп'ютерної топографії та інтерферометрії. J. [Текст] / Takeda M, Ina H, and Kobayashi S (1982) 156–160с.

4. Bruning J H, Herriott D R, Gallagher J E, Rosenfeld D P, White A D і Brangaccio D J (1974) Цифровий інтерферометр для вимірювання хвильового фронту для тестування оптичних поверхонь і лінз. [Текст] / Brangaccio D J (1974) 2693–2703с.

5. Оптична голографія [Електронний ресурс] / Режим доступу: www. URL: https://en.wikipedia.org/wiki/Physics_of_optical_holography

6. Wyant J C (2003) Динамічна інтерферометрія. [Текст] / Wyant J C (2003) 36–41с.

7. 2D/3D голограми [Електронний ресурс] / Режим доступу: www. URL: https://www.hiraholovision.com/pdf/td-td-holograms.pdf

8. Tahara T, Awatsuji Y, Nishio K, Ura S, Kubota T, та Matoba O (2010) Порівняльний аналіз та кількісна оцінка поля зору та зони огляду однокадрової фазової цифрової голографії з використанням мультиплексування з просторовим розподілом. [Текст] / Tahara T, Awatsuji Y, Nishio K, Ura S, Kubota T, та Matoba O (2010) 519–524с.

9. Xia P, Awatsuji Y, Nishio K, and Matoba O (2014) Цифрова голографія з одним мільйоном кадрів в секунду. [Текст] / Xia P, Awatsuji Y, Nishio K, and Matoba O (2014) 1693–1695c.

10. 3D голограми [Електронний ресурс] / Режим доступу: www. URL: https://magic-holo.com/en/what-is-a-3d-hologram/

11. Shimobaba T, Kakue T, and Ito T (2016) Огляд швидких алгоритмів та апаратних реалізацій на комп'ютерній голографії. [Текст] / Shimobaba T, Kakue T, and Ito T (2016) 1611–1622c.

12. С. Айзебітт, Й. Люнінг, В.Ф. Шлоттер, М. Льорген, О. Хеллвіг, В. Еберхардт і Й. Штер, «Безлінзове зображення магнітних наноструктур за допомогою рентгенівської спектроголографії» (2004).

13. Г. Ботьє, А. Марті, Ф. Лівет, Г. ван дер Лаан, С. Станеску та П. Бенкок, "М'яке когерентне розсіювання рентгенівських променів: інструмент і методи на ESRF ID08"(2007). [Текст] / Г. Ботьє, А. Марті, Ф. Лівет, Г. ван дер Лаан, С. Станеску та П. Бенкок, (2007). 432c.

14. Сейсмічні голограми [Електронний ресурс] / Режим доступу: www. URL: https://www.researchgate.net/publication/228908209_Principles_of_Seismic_Holography_for_Diagnostics_of_the_Shallow_Subphotosphere

15. M. Guizar-Sicairos та J. R. Fienup, "Голографія з розширеним посиланням шляхом автокореляційної лінійної диференціальної операції"(2007). [Текст] / M. Guizar-Sicairos та J. R. Fienup, (2007).

16. 3D вентилятор [Електронний ресурс] / Режим доступу: www. URL: https://produkt-ooo.prom.ua/p1288473598-proektor-gologramma .html

17. D. Zhu, M. Guizar-Sicairos, B. Wu, A. Scherz, Y. Acremann, T. Tyliszczak, P. Fischer, N. Friedenberger, K. Ollefs, M. Farle, J. R. Fienup, and J. Stöhr , "Висока роздільна здатність x- безлінзове зображення за допомогою диференціального голографічного кодування» (2010). [Текст] / D. Zhu, M. Guizar-Sicairos, B. Wu, A.

Scherz, Y. Acremann, T. Tyliszczak, P. Fischer, N. Friedenberger, K. Ollefs, M. Farle, J. R. Fienup, and J. Stöhr (2010) 105c.

18. Д.Л. Ван Рой, «Реконструкція цифрового позазвукового хвильового фронту в ближньому полі», Публікація IBM № 320.2402 (19 травня 1971 р.)

19. Є.Г. Вільямс, Х.Д. Дарді та Р.Г. Фінк, «Техніка вимірювання структурної інтенсивності в пластинах»

20. Н.М. Сміт, Принципи голографії, Wiley-Interscience, Нью-Йорк (1969)

21. Мікроконтролер Ардуіно Уно [Електронний ресурс] / Режим доступу: www. URL: https://uk.wikipedia.org/wiki/Arduino_Uno –

22. Світлодіодна матриця 8x8 на базі MAX7219 [Електронний ресурс] / Режим доступу: www. URL:https://robotclass.ru/tutorials/arduino-matrix-8x8-max7219/

23. Переривання в Ардуїно [Електронний ресурс] / Режим доступу: www. URL:https://arduinomaster.ru/program/preryvaniya-arduino-attachinterrupt/

24. Переривання в Ардуїно [Електронний ресурс] / Режим доступу: www. URL:https://ru.wikipedia.org/wiki/Прерывание

25. Електродвигун с редуктором [Електронний ресурс] / Режим доступу: www. URL:https://uk.wikipedia.org/wiki/Мотор-редуктор

26. Датчик обертів [Електронний ресурс] / Режим доступу: www. URL: https://beegreen.com.ua/datchik-shvidkosti-pidrahunku-impulsiv-fc-03- arduino-10356

27. Програма для моделювання Fritzing [Електронний ресурс] / Режим доступу: www. URL:https://ru.wikipedia.org/wiki/Fritzing

28. Програма для моделювання Fritzing [Електронний ресурс] / Режим доступу: www. URL:https://voltiq.ru/fritzing-review/

29. Arduino IDE [Електронний ресурс] / Режим доступу: www. URL:https://ru.wikipedia.org/wiki/Arduino_IDE

30. Середа розробки Arduino IDE [Електронний ресурс] / Режим доступу: www. URL:http://arduino.ru/Arduino_environment

Додаток А

Код проекту

```
#define OFFSET 10 // позволяет скомпенсировать неправильно
выставленную длину кривошипа
#define START_EFFECT 5  // эффект при старте
// также эффекты можно переключать кнопкой
#define CS_PIN 9
#include <SPI.h>
#include <GyverTimers.h>
#include <FastLED.h>
#include "buttonMinim.h"
#include "fonts.h"
buttonMinim butt(3);

// двухмерный масив байтов
byte buffer[8][8];
#define drawDot(x, y, z) bitSet(buffer[(x)][(z)], (y))
#define clearDot(x, y, z) bitClear(buffer[(x)][(z)], (y))
// макросы
//============================
#define EVERY_MS(x) \
  static uint32_t tmr;\
  bool flag = millis() - tmr >= (x);\
  if (flag) tmr = millis();\
  if (flag)
#define FOR_i(from, to) for(int i = (from); i < (to); i++)
#define FOR_j(from, to) for(int j = (from); j < (to); j++)
#define FOR_k(from, to) for(int k = (from); k < (to); k++)

#define EFF_AMOUNT 6
bool loading = true;
byte effNum = START_EFFECT;
const byte sinDivider[] = {215, 115, 89, 82, 82, 89, 115,
215};
volatile uint32_t cycle;
volatile uint32_t debounce;
volatile int8_t counter = 0;
volatile byte state = 0;
```

```
bool flag = false;
int cycleF;

//запуск приложения, то есть точка входа
void setup() {
  //устанавливает режим работы заданного выхода
  pinMode(CS_PIN, OUTPUT);
  //устанавливает порядок вывода даннах из шины SPI
  SPI.setBitOrder(MSBFIRST);
  //Инициализирует шину SPI
  SPI.begin();
  //устанавливем настройки окружения
  maxInit();
  //очищаем матрицу
  clearMatrix();
  //устанавливаем яркость
  setBright(15);
  //Инициируем последовательное соединение и задаём
скорость передачи данных. 9600 битов в секунду
  Serial.begin(9600);
  //задаём обработчика внешних прерываний - isr. В данном
случае обработчик будет вызыватся когда состояние вывода
изменится с высокого уровня на низкий
  attachInterrupt(0, isr, FALLING);
  //установка Timer на одну секунду
  Timer1.setPeriod(1000);
  Timer1.enableISR(CHANNEL_A);
}

//зацикленная функция для отслеживания тиков
void loop() {
  systemTick();
  effectsTick();
  controlTick();
}


//контроль над тиками - периодами
void controlTick() {
```

```
  if (butt.clicked()) {
    if (++effNum >= EFF_AMOUNT) effNum = 0;
    loading = true;
  }
}

//обработка периодов и обработка вращения
void systemTick() {
  EVERY_MS(100) {
    // фильтруем период
    cycleF = (12 * cycleF + 4 * cycle) >> 4;

    // проверить  есть ли вращение, если нет то очистить
матрицу
    if (millis() - debounce > 1000) {
      if (flag) {
        flag = false;
        clearMatrix();
      }
    } else {
      if (!flag) {
        flag = true;
      }
    }
  }
}

//установка основных параметров окружения
void maxInit() {
  sendCMD(0x0f, 0x00);  // отключить режим теста
  sendCMD(0x09, 0x00);  // выключить декодирование
  sendCMD(0x0a, 0x00);  // яркость
  sendCMD(0x0b, 0x0f);  // отображаем всё
  sendCMD(0x0C, 0x01);  // включить
}

//функция для работы с матрицей через шину SPI
void sendCMD(uint8_t address, uint8_t value) {
  //Подает LOW значение на цифровой вход
```

```
  digitalWrite(CS_PIN, LOW);
  //трансфер байтов по шине SPI
  SPI.transfer(address);
  SPI.transfer(value);
  SPI.transfer(address);
  SPI.transfer(value);
  //Подает HIGH значение на цифровой вход
  digitalWrite(CS_PIN, HIGH);
}

//установка яркости
void setBright(byte value) {
  sendCMD(0x0a, value);  // яркость 0-15
}

void sendDATA(uint8_t address, uint8_t value) {
  digitalWrite(CS_PIN, LOW);
  SPI.transfer(++address);
  SPI.transfer(value);
  digitalWrite(CS_PIN, HIGH);
}

//очистка матрицы
void clearMatrix() {
  for (byte i = 0; i < 8; i++) {
    sendDATA(i, 0);
  }
}

//класс для работы с кнопкой
#pragma pack(push,1)
typedef struct {
  bool holdedFlag: 1;
  bool btnFlag: 1;
  bool pressF: 1;
  bool clickF: 1;
  bool holdF: 1;
} buttonMinimFlags;
#pragma pack(pop)
```

```
class buttonMinim {
  public:
    buttonMinim(uint8_t pin);
    boolean pressed();
    boolean clicked();
    boolean holding();
    boolean holded();
  private:
    buttonMinimFlags flags;
    void tick();
    uint32_t _btnTimer;
    byte _pin;
};

buttonMinim::buttonMinim(uint8_t pin) {
  pinMode(pin, INPUT_PULLUP);
  _pin = pin;
}

void buttonMinim::tick() {
  boolean btnState = digitalRead(_pin);
  if (!btnState && !flags.btnFlag && ((uint32_t)millis() -
_btnTimer > 90)) {
    flags.btnFlag = true;
    _btnTimer = millis();
    flags.pressF = true;
    flags.holdedFlag = true;
  }
  if (btnState && flags.btnFlag && ((uint32_t)millis() -
_btnTimer < 350)) {
    flags.btnFlag = false;
    _btnTimer = millis();
    flags.clickF = true;
    flags.holdF = false;
  }
  if (flags.btnFlag && ((uint32_t)millis() - _btnTimer >
600)) {
    if (!btnState) {
```

```
      flags.holdF = true;
    } else {
      flags.btnFlag = false;
      flags.holdF = false;
      _btnTimer = millis();
    }
  }
}

boolean buttonMinim::pressed() {
  buttonMinim::tick();
  if (flags.pressF) {
    flags.pressF = false;
    return true;
  }
  else return false;
}

boolean buttonMinim::clicked() {
  buttonMinim::tick();
  if (flags.clickF) {
    flags.clickF = false;
    return true;
  }
  else return false;
}

boolean buttonMinim::holding() {
  buttonMinim::tick();
  if (flags.holdF) {
    return true;
  }
  else return false;
}

boolean buttonMinim::holded() {
  buttonMinim::tick();
  if (flags.holdF && flags.holdedFlag) {
    flags.holdedFlag = false;
```

```
    return true;
  }
  else return false;
}


#define BALL_SPEED 100
#define RAIN_SPEED 100
#define CUBE_SPEED 200
#define NOISE_SPEED 100
#define TEXT_SPEED 50

// битмапа из программы (эффект №2).
const byte bitmap[8][8] = {
  {0xFF, 0x81, 0x81, 0x81, 0x81, 0x81, 0x81, 0xFF},
  {0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x81},
  {0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x81},
  {0x81, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x81},
  {0x81, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x81},
  {0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x81},
  {0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x81},
  {0xFF, 0x81, 0x81, 0x81, 0x81, 0x81, 0x81, 0xFF},
};

int effSpeed = 100;

//установка эффектов
void effectsTick() {
  EVERY_MS(effSpeed) {
    switch (effNum) {
      case 0: ball();
        break;
      case 1: rain();
        break;
      case 2: drawBitmap();
        break;
      case 3: cube();
        break;
      case 4: noise();
```

```
        break;
      case 5: text("KI-118");
        break;
    }
  }
}


// рисование мячика
void ball() {
  static int8_t coord[3];
  static int8_t speed[3];
  if (loading) {
    effSpeed = BALL_SPEED;
    loading = false;
    FOR_i(0, 3) {
      coord[i] = random(80);
      speed[i] = random(5, 15);
    }
  }
  clearBuffer();
  FOR_i(0, 3) {
    int newCoord = coord[i] + speed[i];
    if (newCoord < 0 || newCoord > 70) speed[i] = -
speed[i];
    else coord[i] = newCoord;
  }
  byte newCoord[3];
  FOR_i(0, 3) newCoord[i] = coord[i] / 10;

  drawDot(newCoord[0], newCoord[1], newCoord[2]);
  drawDot(newCoord[0] + 1, newCoord[1], newCoord[2]);
  drawDot(newCoord[0], newCoord[1] + 1, newCoord[2]);
  drawDot(newCoord[0], newCoord[1], newCoord[2] + 1);
  drawDot(newCoord[0] + 1, newCoord[1] + 1, newCoord[2]);
  drawDot(newCoord[0] + 1, newCoord[1], newCoord[2] + 1);
  drawDot(newCoord[0], newCoord[1] + 1, newCoord[2] + 1);
  drawDot(newCoord[0] + 1, newCoord[1] + 1, newCoord[2] +
1);
```

```
}

// ===== дождь =====
#define RAIN_PROB 10  // вероятность дождя
void rain() {
  if (loading) {
    effSpeed = RAIN_SPEED;
    loading = false;
    clearBuffer();
  }

  FOR_i(0, 8) {
    FOR_j(0, 8) {
      clearDot(i, j, 7);
      if (random8() < RAIN_PROB) drawDot(i, j, 7);
    }
  }
  shiftDown();
}

void shiftDown() {
  FOR_i(0, 7) {
    FOR_j(0, 8) {
      buffer[j][i] = buffer[j][i + 1];
    }
  }
}

// рисование по битмапе
void drawBitmap() {
  if (loading) {
    loading = false;
    clearBuffer();
    FOR_i(0, 8) {
      FOR_j(0, 8) {
        buffer[i][j] = bitmap[i][j];
      }
    }
  }
```

```
}

// ===== куб =====
void cube() {
  static int8_t counter = 0;
  static bool dir = true;
  if (loading) {
    loading = false;
    clearBuffer();
    effSpeed = CUBE_SPEED;
  }
  clearBuffer();
  drawCube(counter, counter, counter, 8 - counter * 2);
  counter += dir ? 1 : -1;
  if (dir && counter > 4) {
    dir = false;
  }
  if (!dir && counter < 0) {
    dir = true;
    counter = 0;
  }
}

// рисовать куб
void drawCube(byte x, byte y, byte z, byte size) {
  size -= 1;
  if (x + size + 1 > 8 || y + size + 1 > 8 || z + size + 1
> 8) return;
  FOR_i(x, x + size + 1) {
    drawDot(i, y, z);
    drawDot(i, y + size, z);
    drawDot(i, y, z + size);
    drawDot(i, y + size, z + size);
  }
  FOR_i(y, y + size + 1) {
    drawDot(x, i, z);
    drawDot(x + size, i, z);
    drawDot(x, i, z + size);
    drawDot(x + size, i, z + size);
```

```
  }
  FOR_i(z, z + size + 1) {
    drawDot(x, y, i);
    drawDot(x + size, y + size, i);
    drawDot(x, y + size, i);
    drawDot(x + size, y, i);
  }
}
// ===== шум =====
#define STEP 50
void noise() {
  static int value = 0;
  if (loading) {
    loading = false;
    effSpeed = NOISE_SPEED;
  }
  value += 30;
  clearBuffer();
  FOR_i(0, 8) {
    FOR_j(0, 8) {
      drawDot(i, j, inoise8(i * STEP, j * STEP, value) /
32);
    }
  }
}

// ===== текст =====
void text(char* thisText) {
  static int8_t charPos = 0;
  static byte currentChar = 0;
  if (loading) {
    loading = false;
    effSpeed = TEXT_SPEED;
    charPos = -1;
    currentChar = 0;
  }
  if (charPos == -1) {
    loadChar(thisText[currentChar]);
    charPos++;
```

```
  }
  if (charPos > 0) shiftForward();
  if (charPos == 1) loadChar(thisText[currentChar]);
  if (charPos > 1) loadChar(32);
  charPos++;

  if (charPos > 7) {
    charPos = -1;
    currentChar++;
    if (thisText[currentChar] == '\0') currentChar = 0;
  }
}

void loadChar(char thisChar) {
  FOR_i(0, 8) {
    byte thisFont = getFont(thisChar, i); // 32 - пусто!
    byte mirrorFont = 0;
    FOR_j(0, 7) {
      if (thisFont & (1 << 6 - j)) mirrorFont |= (1 << j);
      else mirrorFont |= (0 << j);
    }
    buffer[0][7 - i] = mirrorFont;
  }
}

void shiftForward() {
  FOR_i(0, 8) {
    FOR_j(0, 7) {
      buffer[7 - j][i] = buffer[7 - j - 1][i];
    }
  }
}




const uint8_t fontHEX[][8] PROGMEM = {
  {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
```

```
{0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x00, 0x02},
// !
{0x05, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
// "
{0x28, 0x28, 0x7E, 0x14, 0x14, 0x3F, 0x0A, 0x0A},
// #
{0x04, 0x1E, 0x05, 0x05, 0x0E, 0x14, 0x14, 0x0F},
// $
{0x46, 0x49, 0x29, 0x26, 0x90, 0x50, 0x48, 0x88},
// %
{0x06, 0x09, 0x09, 0x26, 0x29, 0x11, 0x31, 0x4E},
// &
{0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
// '
{0x02, 0x02, 0x01, 0x01, 0x01, 0x01, 0x01, 0x02},
// (
{0x02, 0x02, 0x04, 0x04, 0x04, 0x04, 0x04, 0x02},
// )
{0x15, 0x0E, 0x15, 0x04, 0x00, 0x00, 0x00, 0x00},
// *
{0x00, 0x08, 0x08, 0x08, 0x7F, 0x08, 0x08, 0x08},
// +
{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x02},
// ,
{0x00, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00},
// -
{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x02},
// .
{0x04, 0x04, 0x02, 0x02, 0x02, 0x02, 0x02, 0x01},
// /
{0x0E, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0E},
// 0
{0x04, 0x06, 0x04, 0x04, 0x04, 0x04, 0x04, 0x0E},
// 1
{0x0E, 0x11, 0x10, 0x08, 0x04, 0x02, 0x01, 0x1F},
// 2
{0x0E, 0x11, 0x10, 0x0C, 0x10, 0x10, 0x11, 0x0E},
// 3
{0x08, 0x0C, 0x0A, 0x09, 0x1F, 0x08, 0x08, 0x08},
```

```
 // 4
 {0x1F, 0x01, 0x01, 0x0F, 0x10, 0x10, 0x11, 0x0E},
 // 5
 {0x0C, 0x02, 0x01, 0x0F, 0x11, 0x11, 0x11, 0x0E},
 // 6
 {0x1F, 0x10, 0x08, 0x08, 0x04, 0x04, 0x02, 0x02},
 // 7
 {0x0E, 0x11, 0x11, 0x0E, 0x11, 0x11, 0x11, 0x0E},
 // 8
 {0x0E, 0x11, 0x11, 0x11, 0x1E, 0x10, 0x08, 0x06},
 // 9
 {0x00, 0x00, 0x02, 0x02, 0x00, 0x00, 0x02, 0x02},

 {0x18, 0x18, 0x24, 0x24, 0x24, 0x7E, 0x42, 0x42},
 // A
 {0x1E, 0x22, 0x22, 0x1E, 0x22, 0x22, 0x22, 0x1E},
 // B
 {0x78, 0x04, 0x02, 0x02, 0x02, 0x02, 0x04, 0x78},
 // C
 {0x1E, 0x22, 0x42, 0x42, 0x42, 0x42, 0x22, 0x1E},
 // D
 {0x3E, 0x02, 0x02, 0x1E, 0x02, 0x02, 0x02, 0x3E},
 // E
 {0x3E, 0x02, 0x02, 0x3E, 0x02, 0x02, 0x02, 0x02},
// F
 {0x78, 0x04, 0x02, 0x02, 0x72, 0x42, 0x44, 0x78},
// G
{0x42, 0x42, 0x42, 0x7E, 0x42, 0x42, 0x42, 0x42},
// H
 {0x1C, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x1C},
// I
 {0x38, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x1C},
// J
 {0x22, 0x12, 0x0A, 0x06, 0x06, 0x0A, 0x12, 0x22},
// K
 {0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x3C},
// L
 {0x63, 0x63, 0x55, 0x55, 0x49, 0x49, 0x41, 0x41},
// M
```

```
 {0x46, 0x46, 0x4A, 0x4A, 0x52, 0x52, 0x62, 0x62},
// N
 {0x1C, 0x22, 0x41, 0x41, 0x41, 0x41, 0x22, 0x1C},
// O
 {0x1E, 0x22, 0x22, 0x22, 0x1E, 0x02, 0x02, 0x02},
// P
 {0x1C, 0x22, 0x41, 0x41, 0x41, 0x41, 0x22, 0x1C},
// Q
 {0x1E, 0x22, 0x22, 0x22, 0x1E, 0x12, 0x22, 0x42},
// R
 {0x3C, 0x02, 0x02, 0x1C, 0x20, 0x20, 0x20, 0x1E},
// S
 {0x3E, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08},
// T
 {0x42, 0x42, 0x42, 0x42, 0x42, 0x42, 0x42, 0x3C},
// U
 {0x22, 0x22, 0x22, 0x14, 0x14, 0x14, 0x08, 0x08},
// V
 {0x91, 0x91, 0x91, 0xAA, 0xAA, 0xAA, 0x44, 0x44},
// W
 {0x22, 0x22, 0x14, 0x08, 0x08, 0x14, 0x22, 0x22},
// X
 {0x22, 0x22, 0x14, 0x14, 0x08, 0x08, 0x08, 0x08},
// Y
 {0x3E, 0x20, 0x10, 0x08, 0x08, 0x04, 0x02, 0x3E},
// Z
 {0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10},
// [
 {0x01, 0x01, 0x02, 0x02, 0x02, 0x02, 0x02, 0x04},
// \
 {0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04},
// ]
 {0x08, 0x14, 0x22, 0x41, 0x00, 0x00, 0x00, 0x00},
// ^
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
// _
 {0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
// `
 {0x00, 0x00, 0x1C, 0x20, 0x3C, 0x22, 0x22, 0x3C},
```

```
// a
 {0x02, 0x02, 0x1E, 0x22, 0x22, 0x22, 0x22, 0x1E},
// b
 {0x00, 0x00, 0x38, 0x04, 0x04, 0x04, 0x04, 0x38},
// c
 {0x20, 0x20, 0x3C, 0x22, 0x22, 0x22, 0x22, 0x3C},
// d
 {0x00, 0x00, 0x1C, 0x22, 0x3E, 0x02, 0x22, 0x1C},
// e
 {0x30, 0x08, 0x08, 0x1C, 0x08, 0x08, 0x08, 0x08},
// f
 {0x3C, 0x22, 0x22, 0x22, 0x3C, 0x20, 0x20, 0x1C},
// g
 {0x02, 0x02, 0x02, 0x1E, 0x22, 0x22, 0x22, 0x22},
// h
 {0x10, 0x00, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10},
// i
 {0x10, 0x00, 0x18, 0x10, 0x10, 0x10, 0x10, 0x0C},
// j
 {0x02, 0x02, 0x12, 0x0A, 0x06, 0x0A, 0x12, 0x22},
// k
 {0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08},
// l
 {0x00, 0x00, 0x37, 0x49, 0x49, 0x49, 0x49, 0x49},
// m
 {0x00, 0x00, 0x1E, 0x22, 0x22, 0x22, 0x22, 0x22},
// n
 {0x00, 0x00, 0x1C, 0x22, 0x22, 0x22, 0x22, 0x1C},
// o
 {0x1E, 0x22, 0x22, 0x22, 0x1E, 0x02, 0x02, 0x02},
// p
 {0x3C, 0x22, 0x22, 0x22, 0x3C, 0x20, 0x20, 0x20},
// q
 {0x00, 0x00, 0x34, 0x0C, 0x04, 0x04, 0x04, 0x04},
// r
 {0x00, 0x00, 0x38, 0x04, 0x0C, 0x30, 0x20, 0x1C},
// s
 {0x08, 0x08, 0x3C, 0x08, 0x08, 0x08, 0x08, 0x30},
// t
```

```
  {0x00, 0x00, 0x22, 0x22, 0x22, 0x22, 0x22, 0x3C},
 // u
  {0x00, 0x00, 0x22, 0x22, 0x14, 0x14, 0x08, 0x08},
 // v
  {0x00, 0x00, 0x49, 0x49, 0x55, 0x55, 0x22, 0x22},
 // w
  {0x00, 0x00, 0x22, 0x14, 0x08, 0x08, 0x14, 0x22},
 // x
  {0x00, 0x00, 0x12, 0x12, 0x12, 0x1C, 0x10, 0x0C},
 // y
  {0x00, 0x00, 0x3C, 0x20, 0x10, 0x08, 0x04, 0x3C},
 // z
  {0x04, 0x04, 0x04, 0x04, 0x03, 0x04, 0x04, 0x04},
};
// интерпретатор кода символа по ASCII в его номер в
массиве fontHEX
uint8_t getFont(uint8_t font, uint8_t row) {
  font = font - '0' + 16;   // перевод код символа из
таблицы ASCII в номер согласно нумерации массива
  if (font < 126) return
pgm_read_byte(&(fontHEX[font][row]));   // для английских
букв и символов
  else return fontHEX[font - 65][row];          // для
русских букв и символов
}
//обработчик прерывания. Служит для реагирования на событие
прерывания в attachInterrupt
void isr() {
  if (millis() - debounce > 5) {
    cycle = millis() - debounce;
    debounce = millis();
    counter = 0;
    state = 0;
    drawBuffer(counter);
    //Timer1.setPeriod(cycleF / 2 / 8 * 1000);
    Timer1.setPeriod(cycleF / 2 * (sinDivider[counter] +
OFFSET));
  }
}
```

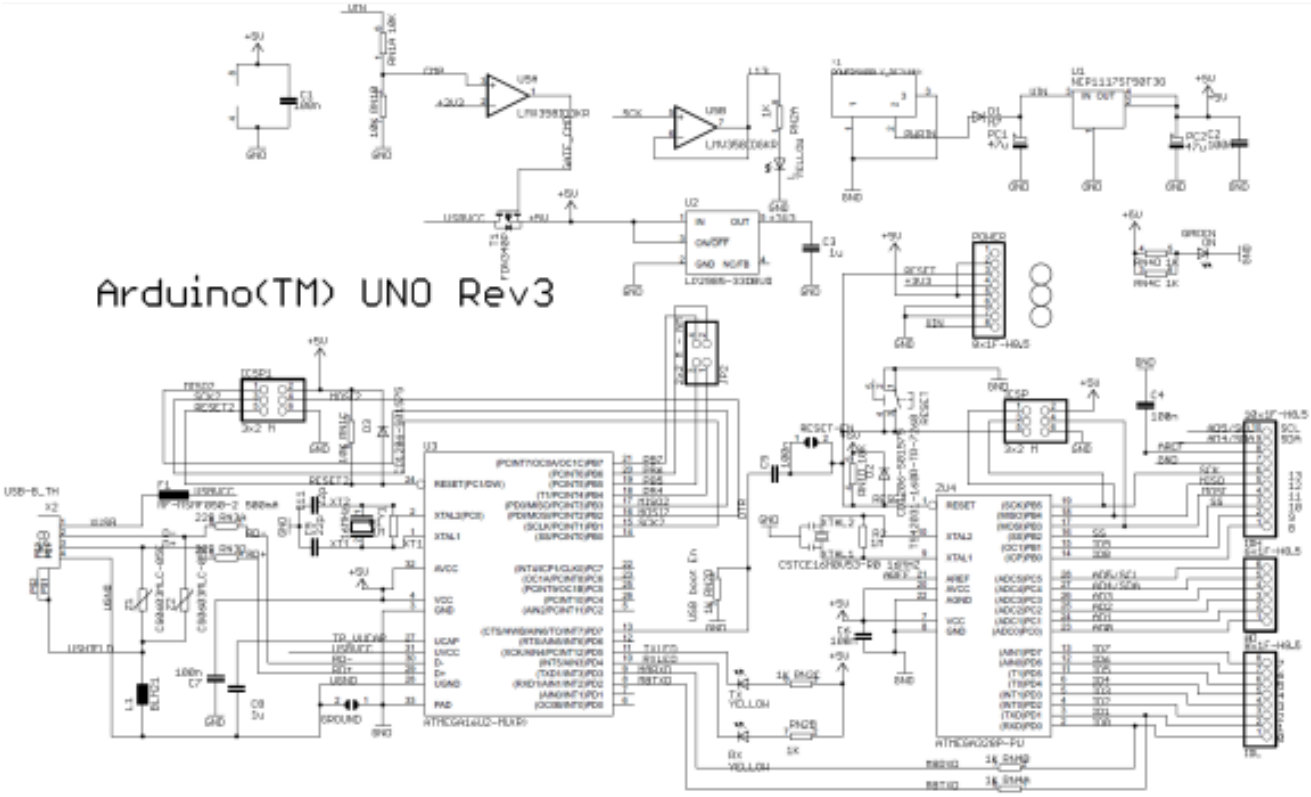
```
ISR(TIMER1_A) {
  // рисуем и вверх и вниз. В зависимости от state меняем
напрявление рисования
  switch (state) {
    case 0:
      counter++;
      if (counter > 7) {
        counter = 7;
        state = 1;
        break;
      }
      drawBuffer(counter);
      Timer1.setPeriod(cycleF / 2 * (sinDivider[counter] +
OFFSET));
      break;
    case 1:
      counter--;
      drawBuffer(counter);
      Timer1.setPeriod(cycleF / 2 * (sinDivider[counter] +
OFFSET));
      if (counter < 1) {
        state = 2;
        Timer1.stop();
      }
      break;
  }
}
//запись в буфер
void drawBuffer(byte layer) {
  for (byte i = 0; i < 8; i++) {
    sendDATA(i, buffer[i][layer]);
  }
}

//очистка буффера
void clearBuffer() {
  for (byte i = 0; i < 8; i++) {
    for (byte j = 0; j < 8; j++) {
```

```
        buffer[i][j] = 0;
      }
    }
  }
```

Додаток Б

Принципова схема ArduinoUno



Додаток В

Принципова схема проекту