

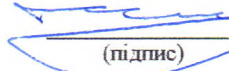
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет електроніки
(повна назва інституту/факультету)

Кафедра акустичних та мультимедійних електронних систем
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

 С.А. Найда
(підпис) (ініціали, прізвище)

“01” червня 2020

Дипломна робота
на здобуття ступеня бакалавра

зі спеціальності (спеціалізації) 171 Електроніка (Електронні та інформаційні системи і технології телебачення, кінематографії)
(код та назва спеціальності)

на тему: "Система світло-кольорового супроводу звуковідтворення музичних фонограм"

Виконав: студент IV курсу, групи ДВ-61
(шифр групи)

Сущук Олексій Олександрович
(прізвище, ім'я, по батькові)



(підпис)

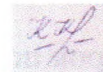
Керівник асистент, к.т.н., Філіпова Н. Ю.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)



(підпис)

Консультант _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент Клен К. С.
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)



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

Студент



(підпис)

Київ – 2020 року

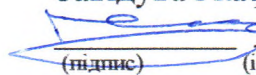
**Національний технічний університет України
"Київський політехнічний інститут імені І. Сікорського"**

Інститут (факультет) _____ Факультет електроніки
(повна назва)

Кафедра _____ Акустичних та мультимедійних електронних систем
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність (спеціалізація) 171 Електроніка (Електронні та інформаційні системи і технології телебачення, кінематографії)
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
 Є.А. Найда
(підпис) (ініціали, прізвище)
"25" травня 2020

**ЗАВДАННЯ
на дипломну роботу студенту**

_____ Сущуку Олексію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Система світло-кольорового супроводу звуковідтворення музичних фонограм

керівник роботи Філіпова Наталія Юріївна, асистент, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "25" травня 2020 р. №1196-с

2. Строк подання студентом роботи _____ 01 червня 2020

3. Вихідні дані до роботи Створення системи світло-кольорового супроводу звуковідтворення музичних фонограм

4. Зміст роботи Аналіз системи та компонентів для створення приладу, розгляд готових аналогів та розробка пристрою

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) презентація з наведеними результатами розробки системи

6. Консультанти розділів роботи*

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

7. Дата видачі завдання 13 квітня 2020

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Написання першого розділу	15 травня 2020	виконано
2	Написання другого розділу	22 травня 2020	виконано
3	Написання третього розділу	29 травня 2020	виконано
4	Підготовка матеріалів до друку та оформлення пояснювальної записки	1 червня 2020	виконано
5	Підготовка та оформлення презентації для доповіді	5 червня 2020	виконано

Студент


 (підпис)
О.О. Суцук

(ініціали, прізвище)

Керівник роботи


 (підпис)
Н.Ю. Філіпова

(ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломної роботи.

УДК 004.35

РЕФЕРАТ

Дипломна робота: 60 с., 7 табл., 1 дод., 20 рис., 20 джерел.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СВІТЛОМУЗИКА, МІКРОКОНТРОЛЕРИ, ARDUINO, RGB СВІТЛОДІОДИ.

Об'єктом дипломної роботи є дослідження та розробка пристрою та програмного забезпечення для створення світломузики, застосування її у різних галузях.

Предметом дослідження є плата Arduino, світлодіодна стрічка, програмні засоби програмування мікроконтролерів.

Мета роботи полягає у створенні приладу та програмних засобів, для реалізації світломузики.

Методами роботи є аналіз та дослідження апаратно-програмних засобів для розробки системи світло-кольорового супроводу звуковідтворення музичних фонограм, застосовуючи фізико-технічні, апаратні та алгоритмічно-програмні методи побудови схем в Arduino.

В результаті виконання дипломної роботи створено системи світло-кольорового супроводу звуковідтворення музичних фонограм, її характеристики, переваги та недоліки, приклади застосування у житті та особливості поєднання з реальним світом.

ABSTRACT

Graduate work: 60 pages, 7 tables, 20 figures, 20 sources.

SOFTWARE, LIGHT MUSIC, MICROCONTROLLERS, ARDUINO, RGB LEDS.

The object of the thesis is the research and development of devices and software for the creation of light music, its application in various fields.

The subject of research is the analysis of modern control systems of world effects, and their reproduction, namely control and reproduction of light effects on the basis of LED strips.

The purpose of the work is to create a device and software for the implementation of light music.

The research method is a theoretical analysis and a practical result of creating a system of light-color accompaniment of sound reproduction of musical phonograms.

The relevance of the study of the system of control of light effects is the spread of entertainment facilities and various techniques of video editing using light effects. The quality of the final content directly depends on the convenience and efficiency of using light effect control systems.

As a result of the thesis work created systems of light-color accompaniment of sound reproduction of musical phonograms, its characteristics, advantages and disadvantages, examples of application in life and features of combination with the real world.

ЗМІСТ

Вступ.....	7
1 Загальні відомості про платформу та програмування Arduino	9
1.1 Загальні відомості про мікроконтролери.....	9
1.2 Плати Arduino	13
1.3 Робота з цифровими та аналоговими сигналам	18
1.4 Програмування Arduino	22
Висновки до розділу.....	34
2 Аналіз роботи Arduino з світло діодами	35
2.1 Підключення RGB світлодіодів.....	35
2.2 Управління світлодіодними стрічками на Arduino.....	38
Висновки до розділу.....	42
3 Практична реалізація світломузики через модуль Arduino	43
3.1 Схема створеного пристрою.....	45
3.2 Опис елементів системи пристрою.....	46
3.2.1 Характеристики Arduino Nano	47
3.2.2 Характеристики адресної світлодіодної стрічки	48
3.2.3 Характеристики пасивних елементів системи	49
3.3 Опис алгоритму роботи коду	50
3.4 Лістинг програми	51
Висновки до розділу.....	62
Висновки	63
Перелік джерел посилання	64
Додаток А.....	66

ВСТУП

Актуальність дослідження апаратно-програмних засобів для розробки системи світло-кольорового супроводу звуковідтворення музичних фонограм є велике збільшення популярності використання світлових ефектів в розважальних закладах та при зйомках різного контенту, як на відкритих майданчиках, так і в студіях. Тому полягає потреба в використанні портативних, зручних у використанні та багатофункціональних пристроях управління світловими ефектами, а саме для створення світломузики.

Предметом дослідження є аналіз сучасних систем керування світовими ефектами, та їх відтворення, а саме управління та відтворення світлових ефектів на базі світлодіодних стрічок.

Метою роботи є дослідження та розробка системи світло-кольорового супроводу звуковідтворення музичних фонограм.

Об'єктом дослідження є платформа Arduino та системи освітлення.

Результатом дипломної роботи є створений проект системи світло-кольорового супроводу звуковідтворення музичних фонограм.

Практичним значенням результату є можливість використання створеного приладу для більшої зручності та ефективності використання світломузики.

Існують готові пристрої для супроводу світловими ефектами музичних фонограм. Для аналізу розглянуто два прикладу.

RadioKit K165

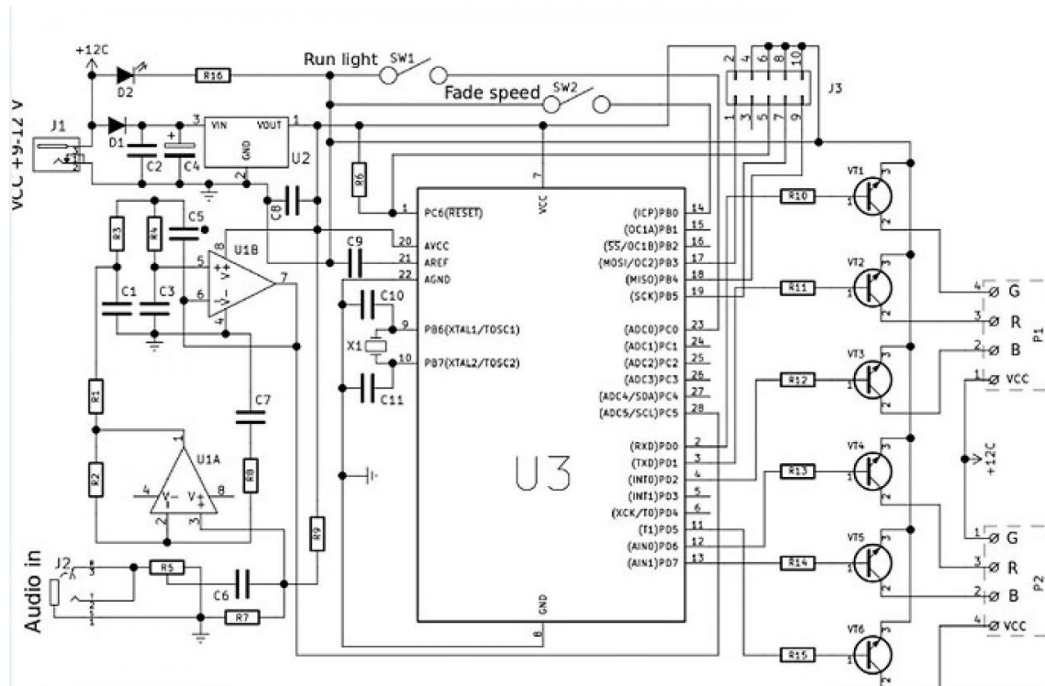


Рисунок – Принципова електрична схема пристрою RadioKit K165

До роз'єму J1 підключається вилка DC живлення 9-24 В, відповідно до обраної RGB стрічкою, світлодіод D2 сигналізує подачу живлення. До роз'єму J2 підключається штекер Jack 3.5мм який необхідно підключити до будь-якого звуковідтворюючого пристрою, можна і на вихід підсилювача низької частоти. До виходів P1, P2 підключається світлодіодна RGB стрічка 12/24 В, як показано на схемах, або зіставити на свій розсуд колірні канали (СЧ, НЧ, ВЧ). За допомогою підлаштований резистора R5 задається рівень вхідного звукового сигналу, який визначає яскравість світіння світлодіодної стрічки. Кнопка SW2 "Fadespeed" Одиночне натискання змінює швидкість згасання каналів при відсутності більш сильного піку [2].

Залежно від характеру музики може знадобиться різна швидкість загасання для кращого візуального сприйняття. Утримання кнопки SW2 довше 3 секунд буде перемикає режим роботи (стандартний, агресивний, агресивний x2). Кнопка SW1 Runlight одиночне натискання змінює режим пристрою в стані спокою (біжать вогні, плавна підсвічування, вимкнено). За замовчуванням, при першому включенні пристрою, встановлений режим біжать вогнів. Утримання кнопки SW1

довше 3 секунд зберігає поточні налаштування (швидкість загасання, режим поведінки в тиші, режим роботи). Утримання одночасно двох кнопок "Runlight" і "Fadespeed" довше 3 секунд призведе до скидання на початкові налаштування [2].

Приклад існуючої безконтактної світломузики для RGB-стрічки

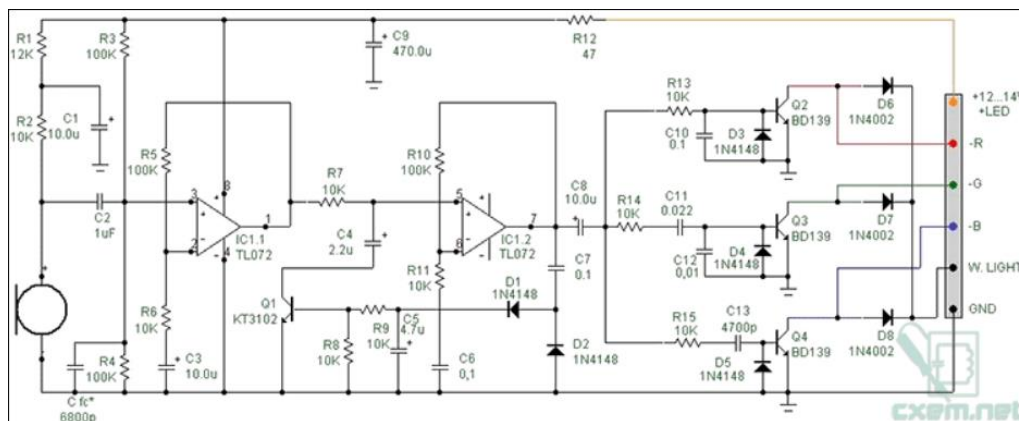


Рисунок – Принципова електрична схема безконтактної світломузики для RGB-стрічки

Цей пристрій призначений для супроводу світловими ефектами музичних фонограм. Особливістю девайса є відсутність його електричного підключення до джерела звукового сигналу. Розроблено пристрій було для застосування спільно зі світлодіодним RGB стрічкою на 12 В [3].

Сигнал з мікрофона дуже слабкий, тому його необхідно спочатку посилити для подальшої роботи з ним. На принциповій схемі підсилювачем мікрофона служить IC1.1. Амплітуда напруги мікрофона посилюється в 11 разів. Для налаштування рівня сигналу потрібен «стабілізатор» рівня сигналу, щоб незалежно від гучності фонограми, амплітуда сигналу залишалася незмінною. Цей «стабілізатор» сигналу називається автоматичний регулятор рівня, або компресор. На схемі IC1.2 і Q1. Друга половинка мікросхеми знову служить підсилювачем в 11 разів, але її вихідний сигнал випрямляється (D1 і D2), і подається у вигляді зсуву на базу Q1. Якщо сигнал занадто великий, Q1 відкривається, опір його колекторного переходу зменшується, тим самим транзистор шунтує надмірний сигнал на вході IC1.2, який надходить з R7. Далі потрібно розкласти стандартизований сигнал на три частотні складові, щоб ударним і бас-

інструментам відповідали червоні спалахи; вокалу, ритм і соло гітар - зелені; ударним і високих тонів - сині. Червоний сигнал пропускається ФНЧ (R13, C10); зелений - ФСЧ (R14, C11, C12); синій - ФВЧ (R15, C13). Далі стоять підсилювачі струму і напруги для кожного каналу, тому що енергії цих відфільтрованих сигналів не вистачить для запалювання відповідних каналів RGB стрічки [3].

Аналіз існуючих систем світло-кольорового супроводу звуковідтворення музичних фонограм доводить актуальність розробки системи на базі Arduino для спрощення та збільшення практичності пристрою.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ПЛАТФОРМУ ТА ПРОГРАМУВАННЯ ARDUINO

1.1 Загальні відомості про мікроконтролери

Перед тим як почати роботу з платформою Arduino, важливо отримати загальні відомості про мікроконтролери. Мікроконтролери застосовуються, перш за все, для автоматизації в метрології, техніці управління і автоматичного регулювання. Перевага мікроконтролерів є в тому, що можна ефективно з малими витратами, практично вимірювати і інтерпретувати фізичні величини, щоб потім приймати необхідні рішення і виконувати необхідні дії. Область можливих застосувань мікроконтролерів надзвичайно широка: від приватного домогосподарства (наприклад, для управління теплицею або освітленням) до промислового виробництва, де можуть обслуговуватися і експлуатуватися комплексні пристрої, керовані системами мікроконтролерів [4].

Контролер · представляє собою, по суті, мікрокомп'ютер і містить всі властиві йому основні модулі (рис. 1.2). Стандартні елементи мікроконтролера - це центральний процесор (CPU), пам'ять програм (Flash-пам'ять), оперативна пам'ять (RAM) і зовнішні пристрої [4].

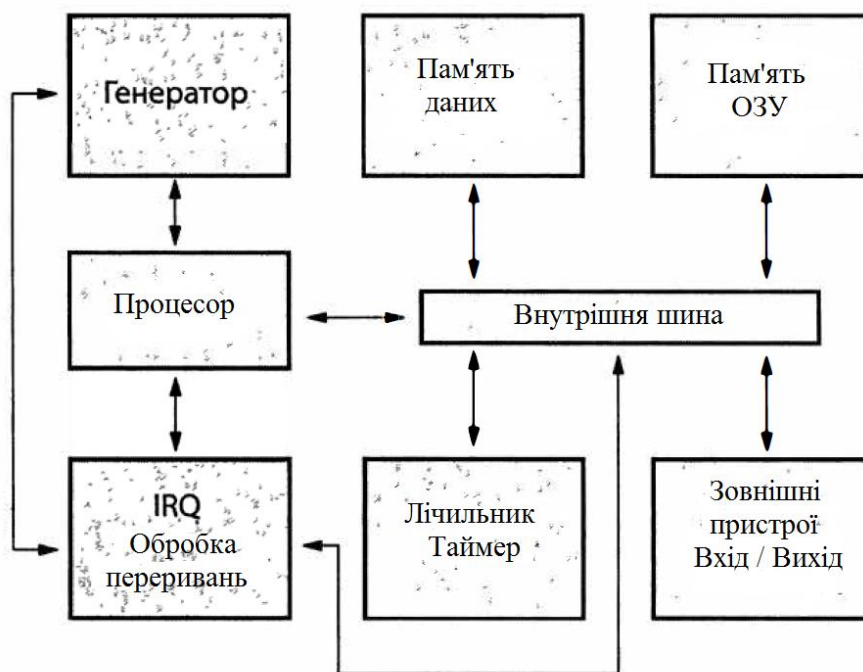


Рисунок 1.1 – Спрощена блок-схема мікроконтролера

Основний функціональний пристрій мікроконтролера – центральний процесор (CPU – Central Processing Unit). Його можна порівняти з "мозком" мікроконтролера. Сигнали представлені в цифровій формі над якими виконуються логічні і арифметичні операції. Оперативна пам'ять і пам'ять програм традиційно розглядаються окремо. Програма користувача, зберігається в незалежній Flash-пам'яті програм. В залежності від типу контролера пам'ять програм може займати об'єм від кБ до МБ. В деяких обчислювальних системах підключивши зовнішні Flash-накопичувачі можна збільшити пам'ять програм [4].

Оперативна пам'ять служить для тимчасового зберігання різних проміжних даних. Тут зберігаються результати обчислень, отримані під час виконання програми. Призначення оперативної пам'яті ОЗУ (RAM - Random Access Memory) – можливість швидкого звернення до обмеженої кількості даних. Її обсяг, як правило, значно менше, а швидкодія набагато більше, ніж Flash-пам'яті. Значення записуються і зберігаються в ОЗУ під час виконання і енергозалежні на відміну від Flash-пам'яті, тобто після перезавантаження контролера вміст ОЗУ повністю

стирається. На рис. 1.2 зображена структура ОЗУ мікроконтролера ATmega168PA[4].

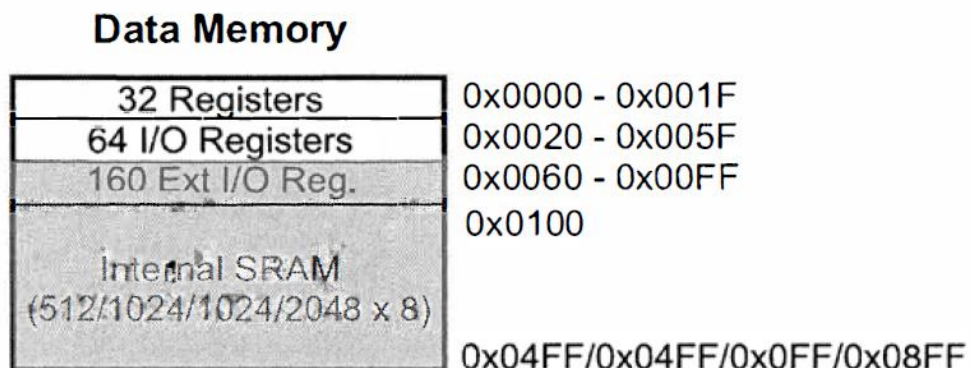


Рисунок 1.2 – Структура ОЗУ мікроконтролера ATmega168PA.

Зовнішніми (периферійними) пристроями часто називають всі компоненти мікроконтролера, крім центрального процесора і пам'яті. Зокрема, до них відносяться зовнішні інтерфейси, наприклад, цифрові входи і виходи (Input / Output - скорочено I/O). Більшість мікроконтролерів забезпечені також різними цифровими і аналоговими входами і виходами [4].

Будь-який алгоритм є послідовність деяких дій, записаних у вигляді набору послідовно виконуваних команд (інструкцій, операторів). При цьому серед таких команд можуть зустрічатися команди переходу, які в деяких випадках порушують вихідну послідовність виконання операторів строго один за одним. Серед інших повинні бути також команди введення і виведення даних, а також команди виконання арифметичних і логічних операцій. Команди повинні десь зберігатися, тому невід'ємною частиною всієї системи має бути пристрій пам'яті програм. Десь треба складати дані, як вихідні, так і результати роботи програми, тому має бути пристрій пам'яті даних. Так як команди і дані, в кінцевому рахунку, все одно є числа, то пам'ять може бути загальна, тільки треба вміти відрізняти, де саме команди, а де - дані. Це є один з принципів фон Неймана, хоча і в мікроконтролерах, традиційно використовують НЕ фон-неймановську, а так звану гарвардську архітектуру, коли пам'яті даних і програм розділені (цей поділ, втім, може в певних межах порушуватися) [4].

Принцип фон Неймана

Склад основних пристроїв ЕОМ: універсальна обчислювальна машина повинна включати в себе пристрій пам'яті, арифметико-логічний пристрій (АЛП), пристрій управління (УУ), пристрій вводу / виводу. Принцип двійкового кодування: дані і програми зберігаються в пам'яті у формі двійкового коду. Принцип збереженої програми: під час обробки, дані і програми знаходяться в загальній пам'яті ЕОМ. Принцип адресується пам'яті: пам'ять складається з двійкових розрядів – бітів, які об'єднуються в осередки, кожна з яких має адресу. Адреса комірки пам'яті - це її порядковий номер. Принцип програмного управління: робота машини відбувається автоматично під керуванням програми, яка поміщається в оперативну пам'ять [5].

Гарвардська архітектура - архітектура ЕОМ, відмітною ознакою якої є роздільне зберігання і обробка команд і даних. Архітектура була розроблена Говардом Ейкен в кінці 1930-х років в Гарвардському університеті.

У 30-х роках уряд США доручило Гарвардському і Прінстонському університетам розробити архітектуру ЕОМ для військово-морської артилерії.

Типові операції (додавання і множення) вимагають від будь-якого обчислювального пристрою декількох дій: вибірку двох операндів, вибір інструкції і її виконання, і, нарешті, збереження результату. Ідея, реалізована Ейкен, полягала у фізичному поділі ліній передачі команд і даних. У першому комп'ютері Ейкена «Марк І» для зберігання інструкцій використовувалася перфорована стрічка, а для роботи з даними - електромеханічні регістри. Це дозволяло одночасно пересилати і обробляти команди і дані, завдяки чому значно підвищувався загальну швидкодію комп'ютера [6].

Процесор, побудований за фон Нейманом, більш універсальний – наприклад, він дозволяє без особливих проблем нарощувати пам'ять, будувати її ієрархічно і більш ефективно її перерозподіляти прямо по ходу роботи. так, в системі Windows завжди передбачається, що комп'ютер має практично необмежений обсяг пам'яті (вимірюваний в терабайт), а якщо її реально не вистачає, до справи підключається своп-файл (так званий файл підкачки) на жорсткому диску. У той же час

мікроконтролерів подібна гнучкість не дуже потрібна – на їх основі, як правило, будуються вузли, які виконують вузьке завдання і працюють за конкретною програмою, так що потрібну конфігурацію системи нічого не варто передбачити заздалегідь [7].

Основний цикл роботи процесора повинен бути таким: вибірка чергової команди (з пам'яті), якщо необхідно вибірка вихідних даних для неї, виконання команди, розміщення результатів в пам'яті (знову ж якщо це необхідно). Вся робота в цьому циклі має відбуватися автоматично по командам деякого пристрою управління, що містить тактовий генератор- системний годинник, за якими все синхронізується. Крім того, десь це все має відбуватися: складування даних, коду команди, виконання дій і т. п., так що процесор повинен містити якийсь набір робочих регістрів (по суті - невелику за обсягом над швидко пам'ять), певним чином пов'язаних як між собою, так і з пристроєм управління і АЛУ, яка неминуче має бути присутнім. Вирішальну роль в роботі процесора грає лічильник команд. він автоматично встановлюється на нуль на початку роботи, що відповідає першій команді, і автоматично ж збільшується на одиницю з кожної виконаної командою. Якщо по ходу справи порядок проходження команд порушується, наприклад, зустрічається команда переходу (розгалуження), то в лічильник завантажується відповідний адреса команди - її номер від початку програми. Якщо це не просто розгалуження, а виконання підпрограми, яке передбачає в подальшому повернення до основної послідовності команд (до наступної команді після виклику підпрограми), то перед переходом до виконання підпрограми поточне значення лічильника команд зберігається в спеціально відведений для цієї мети області пам'яті стеку. За командою закінчення підпрограми збережений адресу витягується з стека, і виконання основної програми триває [8].

1.2 Плати Arduino

Arduino Uno це найбільш поширена стандартна плата. Основа плати є чіп Atmel ATmega328, який має 32 КБ флеш-пам'яті, 1 Кбайт EEPROM, і 2 КБ SRAM.

Також в складі чіпу є 14 цифрових каналів вводу / виводу, 6 з яких використовуються як ШІМ виводи, ще 6 аналогові входи. Плата має кварцевий резонатор на 16 МГц, USB вхід, роз'єм для внутрішнього програмування ICSP. Це найбільш поширена плата для реалізації аматорських проектів в мікропроцесорній техніці [9].



Рисунок 1.3 – Зовнішній вигляд Arduino Uno

Платформа **Arduino Leonardo** відрізняється від інших плат тим, що в цій платі USB порт вмонтований в самий мікроконтролер ATmega32u4, за допомогою цього є непотрібним використання ще одного додаткового процесора, а також плата Leonardo при підключенні до комп'ютера може ідентифікуватися не тільки як віртуальний COM-порт, але також як мишка або клавіатура. Також через зменшення додаткового процесора вона має меншу вартість [9].

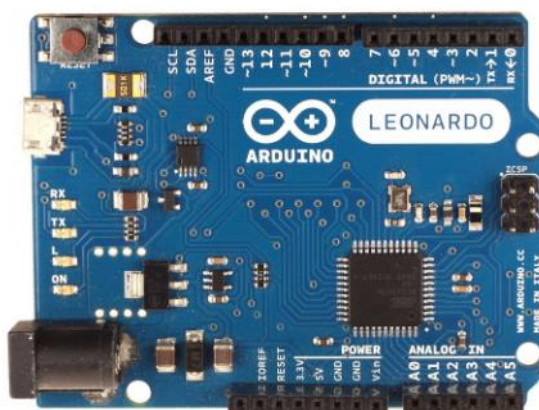


Рисунок 1.4 – Зовнішній вигляд Arduino Leonardo

Платформа **Arduino Ethernet** складається на основі ATmega328, як у

платформі Uno, основною характеристикою є функція підключення до мережі Ethernet. Так, як і в Arduino Uno, платформа Arduino Ethernet має в собі 14 дискретних входів / виходів, але для управління модулем Ethernet і для зчитування SD карт використовується 4 входи [9].

Також в Arduino Ethernet є модуль POE (Power Over Ethernet). За допомогою цієї опції Arduino Ethernet може брати живлення від мережі Ethernet і тоді не потрібно використовувати зовнішні джерела живлення, але без POE Arduino потрібно для живлення зовнішні джерела [9].



Рисунок 1.5 – Зовнішній вигляд Arduino Ethernet

Arduino Mega 2560 має набагато більше каналів вводу / виводу ніж Arduino Uno і тому візуально вона довшого розміру. Arduino Mega 2560 має 54 цифрових каналів вводу / виводу і також 16 аналогових входів. Вона містить 256 КБ флеш-пам'яті, завдяки чому може містити об'ємні програми. А також має Вона також має 8 КБ SRAM і 4 КБ EEPROM, що є досить багато. Для комутації плати з декількома пристроями Arduino Mega має 4 UART [9].

Плати Arduino Mega призначені для використання в проектах де потрібно багато каналів входу.

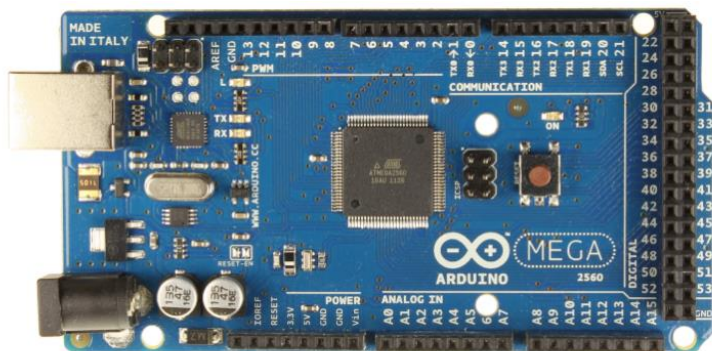


Рисунок 1.6 – Зовнішній вигляд Arduino Mega 2560

Платформа **Arduino Mini** це маленький пристрій, для використання в проектах, які є досить мініатюрні і потрібно як уможна більше економити місце. В ній міститься 14 цифрових входів / виходів і 4 аналогових входів. Пристрій не має USB-роз'єму, регулятора потужності, через те, що вони потребують відносно багато місця. Програмування плати виконується через зовнішній USB або RS232 через TTL послідовний адаптер [9].



Рисунок 1.7 – Зовнішній вигляд Arduino Mini

Контролер **Arduino Micro** одна з самих найменших платформ Arduino. В ній міститься 20 цифрових каналів вводу / виводу, досить багато, як для її розмірів, 7 з них можуть бути використані як виводи ШІМ. Також має 12 аналогових входів. Завдяки розташуванню гребінок на платі, з нею зручно працювати на макетній платі [9].



Рисунок 1.8 – Зовнішній вигляд Arduino Micro

Платформа **Arduino Due** спроектована не чіпі Atmel SAM3X8E а не на AVR. Цим вона кардинально відрізняється від інших плат Arduino. Мікроконтроллер Atmel SAM3X8E працює на 84 МГц і він є 32-розрядним пристроєм. В платі міститься 54 цифрових каналів , з яких 12 можуть бути використані для ШІМ, а також 12 аналогових входів, тобто всього має досить багато виводів. На платі таж є 4 UART, інтерфейс Twin-Wire, порт SPI, а також порт JTAG. Для живлення платформи **Arduino Due** потрібно від 3,3 В [9].



Рисунок 1.9 – Зовнішній вигляд Arduino Due

1.3 Робота з цифровими та аналоговими сигналами

Дані про навколишній світ всі пристрої отримують в аналоговому вигляді. Для управління цифровим входом є кнопка. Перемикач це цифровий пристрій, він має тільки два можливих стани: увімкнено або вимкнено, HIGH або LOW, 1 або 0, і т.д. Цифрова інформація являє собою серію бінарних (цифрових) даних. Кожний біт приймає тільки одне з двох можливих значень [10].

Аналогові сигнали не можна уявити числом станів, теоретично вони можуть мати нескінченне число значень в межах деякого діапазону. Припустимо, сонячний світло-це аналоговий сигнал, який потрібно виміряти. Природно, є розумний діапазон, в межах якого змінюється освітленість (вимірюється в люксах але очікувати значення показань між 0 люкс (для абсолютно чорного) і 130 000 люкс на прямому сонячному світлі. Якби вимірювальний прилад був абсолютно точний, то можна отримати нескінченне число значень в даному діапазоні [10].

Графіки на рис. 1.10 показують, чим один від одного відрізняються цифрові та аналогові сигнали. Зліва прямокутні імпульси, амплітуда яких бере тільки два значення: 0 і 5 вольт. Точно так, як з кнопкою, тільки HIGH АБО LOW. Справа зображений фрагмент косинусоїдального сигналу. Незважаючи на те, що його амплітуда знаходиться в тих же межах (0 і 5 вольт), аналоговий сигнал приймає нескінченне число значень між цими двома [10].

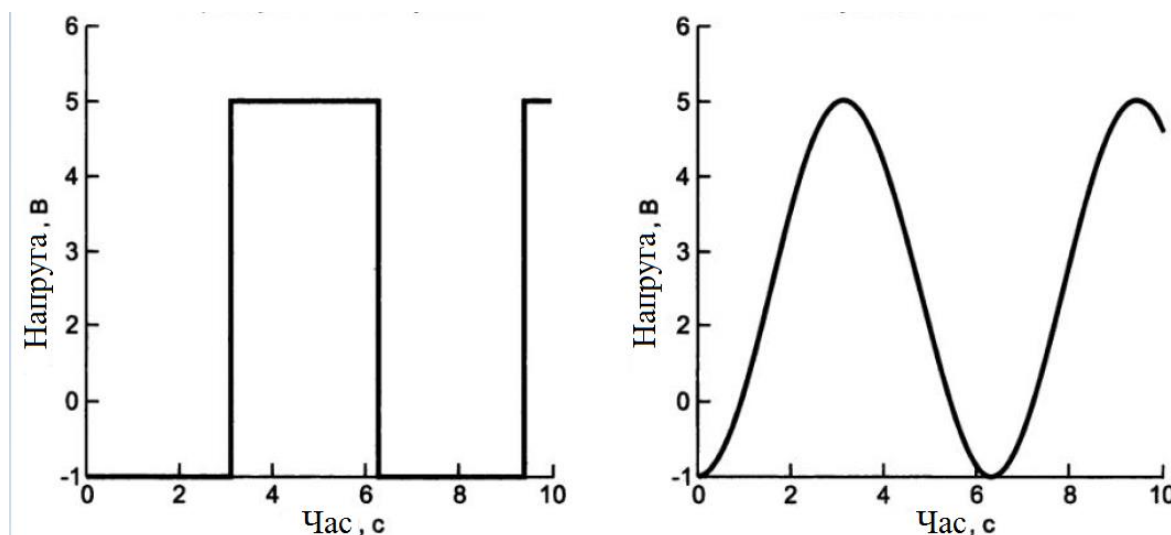


Рисунок 1.10 Аналоговий та цифровий сигнал

Комп'ютерна система ніколи не може оперувати з нескінченним числом десятичних розрядів для аналогового значення, тому що обсяг пам'яті комп'ютера обмежені. Для того, щоб з'єднати інтерфейс цифрового контролера Arduino с аналоговим реальним світом потрібен аналого-цифровий перетворювач (АЦП), який перетворює аналогові значення в цифрові з заданою точністю. Щоб виміряти освітленість в своїй кімнаті, хороший світлочутливий датчик видає вихідну напругу, яка залежить від освітленості кімнати. Коли в приміщенні абсолютно темно, пристрій видавав 0В, а при максимальній освітленості 5 В. Проміжні значення відповідають середнім освітленості. Перетворити аналогові значення напруги в числа, які може обробляти контролер, дозволяє аналого-цифровий перетворювач Arduino. Точність АЦП залежить від його розрядності. На платі Arduino Uno встановлений 10-розрядний АЦП. Це означає, що АЦП може розділити аналоговий сигнал на 2^{10} різних значень [10].

Опорна напруга визначає максимальну напругу на вході АЦП, його значення відповідає коду 1023. При нульовій вхідній напрузі АЦП видає на виході 0, при вхідній напрузі 2,5, на вході буде значення 512 (половина від 1023), при вхідній напрузі 5В вихідний код дорівнює 1023.

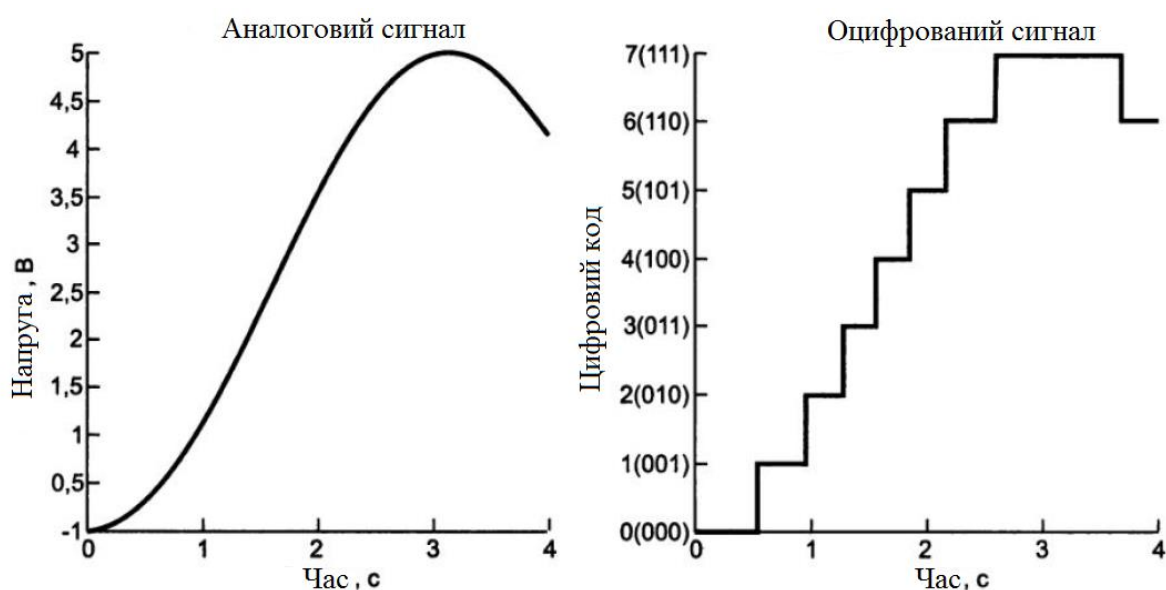


Рисунок 1.11 Трьохрозрядне аналого-цифрове перетворення

У трьохразрядного АЦП 3 біта дозволу, оскільки $2^3 = 8$, отже, у нього є 8 рівнів, від 0 до 7. Будь-якому аналоговому значенню, яке надходить на вхід такого АЦП, на виході відповідає код від 0 до 7. На рис. 1.11 показано, що рівні вхідної напруги перетворюються у вихідні дискретні цифрові коди, з якими може оперувати мікроконтролер. Чим вище розрядність, тим більше рівнів, які доступні для подання кожного значення. Як згадувалося, у Arduino Uno АЦП має 1024 ступені, а не 8, як на рис. 1.11. У контролерів Arduino до більшості контактів введення-виведення можна підключати зовнішні схеми. Всі контакти можуть служити цифровими входами і виходами. Частина контактів Arduino можуть також діяти в якості аналогових входів. Багато з контактів працюють в режимі мультиплексування і виконують додаткові функції: різні комунікаційні інтерфейси, послідовно інтерфейси, широтно-імпульсні модулятори і зовнішні переривання [10].

Виводи платформи Arduino мають змогу працювати як входи або як виходи, а аналогові входи (АТmega) можуть сконфігуруватися та працювати так, як цифрові порти вводу / виводу. Виводи Arduino налаштовані як канали вводу, тому не потрібно функцію `pinMode()`. Сконфігуровані порти введення знаходяться в високоімпедансному стані. Тобто порт введення дає дуже мале навантаження на схему, в яку він інтегрований. Для переводу порту введення з одного стану в інше потрібно маленьке значення струму. Якщо до виводу нічого не підключити, то значення братимуться випадкові величини [11].

За допомогою підтягуючи резисторів можна задати порту відомий стан, це якщо на нього не надходить сигнал, їх підключають на вхід або до живлення, або до землі. Мікроконтролер АТmega має вже запрограмовані вбудовані підтягуючи резистори 20 кОм, Програмування резисторів здійснюється таким чином:

```
pinMode (pin, INPUT) // призначити виводи порт введення
```

```
digitalWrite (pin, HIGH) // включити підтягуючий резистор
```

Виводи, які сконфігуровані як порти виводу знаходяться в низькоімпедансному стані. Виводи можуть пропускати через себе достатньо великий струм. Виводи чіпа АТmega можуть бути джерелом струму до 40 мА і це є

недостатньо для більшості реле і двигунів. Короткі замикання виводів Arduino або спроби підключити енергоємні прилади можуть пошкодити вихідні транзистори вводу або весь мікроконтролер ATmega [11].

Мікроконтролери ATmega, які використовуються в Arduino, мають шестиканальний аналого-цифровий перетворювач (АЦП). Роздільна здатність перетворювача становить 10 бітів, це дозволяє на виході перетворювача отримати значення від 0 до 1023. Аналогові входи можуть бути використані, як цифрові виводи порти вводу / виводу, також вони мають нумерацію від 14 до 19: *pinMode (14, OUTPUT); digitalWrite (14, HIGH);* Для виводу, який працював як цифровий порт виводу, команда *analogRead* буде працювати некоректно. В такому випадку рекомендовано налаштувати аналоговий вхід. Широтно-імпульсна модуляція (ШИМ) – це операція отримання змінюваного аналогового значення за допомогою цифрових пристроїв. При подачі на вихід сигналу, який складається з високих і низьких рівнів, моделюється напруга між максимальним (5 В) і мінімальним (0 В) значеннями. Тривалість включення максимального значення називається шириною імпульсу. Щоб отримати різні аналогові величини, потрібно змінити ширину імпульсу. В результаті на виході буде величина напруги, яка дорівнює площі під імпульсами. Виклик функції *analogWrite ()* з масштабом 0-255, означає, що значення *analogWrite (255)* буде 5 В (100% робочий цикл, включення 5 В), а значення *analogWrite (127)* - 2,5 В (50% робочий цикл) [11].

1.4 Програмування Arduino

Для програмування Arduino використовується інтегроване середовище розробки Arduino (Arduino Integrated Development Environment, IDE). Програми для Arduino пишуться на мові з простою назвою C. Ця мова існує з найперших днів розвитку обчислювальної техніки. А ось що дійсно привносить Arduino - це набір простих у використанні команд, написаних на C, які ви можете використовувати в своїх програмах [12].

В основі структури програми Ардуїно є два складника, це *setup ()* і *loop ()*.

```

void setup () {
// код виконується тільки один раз при запуску програми
}
void loop () {
// основний код програми, який зациклений
}

```

Функція `setup ()` виконується тільки один раз, це відбувається при підключенні живлення або обнулення контролера. Зазвичай в ній задаються початкові установки регістрів, змінних. Функція має бути присутня в програмі, навіть вона пуста і нічого в собі не містить. Після завершення функції `setup ()` управління програмою переходить до функції `loop ()`. Ця функція в нескінченному циклі виконує команди, які записуються в її тіло. А в свою чергу ці команди роблять всі алгоритмічні дії контролера. Вирази можуть містити як завгодно багато прогалів, переносів рядків. Ознакою завершення вираження є символ "крапка з комою"[12].

`{ }` Фігурні дужки потрібні для визначення блоків функції або виразів. Наприклад, у функціях `setup ()` і `loop ()`.

`/* ... */` блок в якому записують коментар, обов'язково потрібно його закрити.

`/* Коментар */`

`//` такий коментар не закривається і діє до кінця рядка.

`//` наступний рядок коментаря [13]

Змінні це осередок оперативної пам'яті, в яких зберігається інформація. Програма використовує змінні для зберігання проміжних даних, які використовуються для обчислень. Для обчислень можуть бути використані дані різних форматів, різної розрядності.

Таблиця 1.1 – Типи даних

Тип даних	Разрядність, біт	Діапазон чисел
Boolean	8	true, false

Char	8	-128 ... 127
unsigned char	8	0 ... 255
Byte	8	0 ... 255
Int	16	-32768 ... 32767
unsigned int	16	0 ... 65535
Word	16	0 ... 65535
Long	32	-2147483648 ... 2147483647
unsigned long	32	0 ... 4294967295
Short	16	-32768 ... 32767
Float	32	-3.4028235+38 ... 3.4028235+38
Double	32	-3.4028235+38 ... 3.4028235+38

Типи даних обирають дивлячись наскільки точні данні потрібно отримати з обчислень, також залежить від форматів даних і т.д. Не варто для програм які виконують мало обчислень використовувати тип даних long. Працювати програма буде, але в такому випадку операція займе більше пам'яті даних і програми, а також знадобиться більше часу для обробки [12].

Спочатку вказується тип даних, а потім ім'я змінної.

```
int x; // оголошується змінна з ім'ям x типу даних int
```

```
float widthBox; // оголошується змінна з ім'ям widthBox типу даних float
```

Всі змінні повинні бути оголошені та присвоєні до того як будуть використовуватися програмою.

В залежності в якій частині програми буде оголошена змінна, залежить і які блоки програми зможуть її використовувати. Змінні, які оголошені на початку

програми, до функції `void setup ()`, вважаються глобальними і доступні в будь-якому місці програми. Локальні змінні оголошуються всередині функцій або в блоках програми і можуть використовуватися тільки в оголошених блоках. Можливі кілька змінних з одним ім'ям, але різними областями видимості [13].

```
int mode; // дана змінна в доступі всім функціям
```

```
void setup () {
```

```
// порожній блок, в якому початкові установки не потрібні
```

```
}
```

```
void loop () {
```

```
long count; // змінна count доступна тільки в тілі функції loop ()
```

```
for (int i = 0; i < 8;) // змінна i в доступі тільки всередині тіла циклу
```

```
{
```

```
i ++;
```

```
}
```

```
}
```

При оголошенні змінної можна задати її початкове значення

```
int x = 5; // оголошена змінна x з початковим значенням 5
```

```
char f = 'g'; // оголошена змінна f з початковим значенням коду якої  
присвоєний символу "g"
```

При арифметичних операціях в яких різні типи даних відбувається автоматичне перетворення типів даних. Але рекомендовано краще завжди використовувати явне перетворення.

```
int x; // змінна int
```

```
char y; // змінна char
```

```
int z; // змінна int
```

```
a = b + (int) c; // змінна y явно перетворена в int [13].
```

Таблиця 1.2 – Арифметичні операції.

=	присвоєння
+	Складання
-	віднімання
*	Множення
/	Ділення
%	остача від ділення

Таблиця 1.3 – Операції відношення

==	Дорівнює
!=	не дорівнює
<	Менше
>	Більше
<=	менше або дорівнює
>=	більше або дорівнює

Таблиця 1.4 – Логичні операції

&&	логічне І
	логічне АБО
!	логічне НЕ

Таблиця 1.5 – Операції над вказівниками

*	непряма адресація
&	отримання адреси змінної

Таблиця 1.6 – Бітові операції

&	I
	АБО
^	ВИКЛЮЧНЕ АБО
~	ІНВЕРСІЯ
<<	ЗСУВ ВЛІВО
>>	ЗСУВ ВПРАВО

Таблиця 1.7 – Операції змішаного присвоєння

++	+ 1 до змінної
--	- 1 до змінної
+=	складання
-=	віднімання
*=	множення
/=	ділення
%=	остача від ділення
&=	бітове I
=	бітове АБО

Оператор IF перевіряє умову в дужках його тіла і якщо умова виконується то код переходить на наступний вираз або блок.

```
if (x == 10) // якщо x = 10, то виконується z = 6
```

```
z = 6;
```

```
if (x > 7) // якщо x > 7, то виконується блок z = 1, y = 2;
```

```
{z = 1; y = 2; }
```

IF ... ELSE дозволяє зробити вибір між двох варіантів, за деякої умови.

```
if (x > 15) // якщо x > 15, то виконується блок z = 25, y = 13;
```

```
{
```

```
    z = 25;
```

```
    y = 13;
```

```
}
```

```
else // або інакше, то виконується цей блок
```

```
{
```

```
    z = 8;
```

```
    y = 9;
```

```
}
```

ELSE IF - дозволяє зробити вибір з декількох варіантів

```
if (x > 7) // якщо x > 7, то виконується блок z = 9, y = 4;
```

```
{
```

```
    z = 9;
```

```
    y = 4;
```

```
}
```

```
else if (x > 40) // якщо x > 40, виконується цей блок
```

```
{
```

```
}
```

```
else // інакше буде виконуватися цей блок
```

```
{
```

```
    z = 5;
```

```
    y = 4;
```

```
}
```

SWITCH CASE - множинний вибір, завдяки йому можна порівняти змінну з декількома константами і виконати блок, в якому змінна дорівнює константі.

```
switch (x) {
```

```
case 8:
```

```
    // код виконується якщо x = 8
```

```

    break;
case 15:
    // код виконується якщо x = 15
    break;
default:
    // код виконується у випадку коли не підійшло ніяке значення
    break;
}

```

Цикл FOR. Конструкція цього циклу дозволяє організовувати цикли з заданим кількістю ітерацій.

```

for (дія до початку циклу;
    умова за якою продовжується цикл;
    дія в кінці кожної ітерації) {
    // код тіла циклу
}

```

Приклад циклу з 500 ітерацій.

```

for (i = 0; i < 500; i++) // початкове значення буде 0, кінцеве 499, заданий крок
дорівнює 1
{
    sum = sum + i;
}

```

Цикл WHILE. Оператор дозволяє організовувати цикли з конструкцією:

```

while (вираз)
{
    // код тіла циклу
}

```

Цикл виконується до тих пір, поки вираз, який знаходиться в дужках істинний. Приклад циклу на 11 ітерацій.

```

x = 11
while (x < 11)

```

```
{
    // код тіла циклу
    x ++;
}
```

DO WHILE - цикл з умовою на виході.

do

```
{
    // код тіла циклу
} While (вираз);
```

Цикл виконується поки вираз істинний.

BREAK - оператор виходу з циклу. Використовується для того, щоб перервати виконання циклів for, while, do while.

```
x = 1;
while (x <8)
{
    if (z > 15) break; // якщо z > 15, то вийти з циклу
    // код тіла циклу
    x ++;
}
```

GOTO - оператор безумовного переходу.

```
goto metka1; // перехід виконання на metka1
```

.....

```
metka1:
```

CONTINUE - пропускаються оператори до кінця тіла циклу.

```
x = 1;
while (x <11)
{
    // код тіла циклу
    if (z > 15) continue; // якщо z > 15, то повернутися на початок тіла циклу
    // код тіла циклу
```

```
x ++;
}
```

Масив це область пам'яті, де пронумерована послідовність величин однакового типу даних.

Оголошується масив таким чином:

```
int ages [18]; // масив з 18 змінних типу даних int
float weight [150]; // масив з 150 змінних типу даних float
```

При оголошенні масиви можна ініціалізувати:

```
int ages [5] = {21, 35, 42, 19, 85};
```

Звертаються до змінних масивів таким чином:

```
x = ages [3]; // x присвоюється значення з 3 елементу масиву.
```

```
ages [1] = 38; // 1 елементу масиву задається значення 38
```

Нумерація елементів масивів завжди починається з нуля.

За допомогою **функцій** можна виконувати одні і ті ж дії з різними даними.

Функція має:

- ім'я, по якій її викликають;
- аргументи - дані, які функція використовує для обчислення;
- тип даних, що повертається функцією.

Описується призначена для користувача функція поза функцій `setup ()` і `loop ()`.

```
void setup () {
```

```
// код виконується тільки один раз при запуску програми
```

```
}
```

```
void loop () {
```

```
// основний код програми, який виконується в циклі
```

```
}
```

```
// оголошення функції яка має ім'я functionHI
```

```
type functionHI (type argument5, type argument5, ..., type argument)
```

```
{
```

```
// тіло функції
```

```
return ();
```

```
}
```

Приклад функції, що обчислює суму квадратів двох аргументів.

```
int sumQwadr (int x, int y)
{
    return (x * x + y * y);
}
```

Виклик функції відбувається таким чином:

```
d = 5; b = 8;
a = sumQwadr (b, c); // в a буде сума квадратів змінних b і c
```

Функції бувають вбудовані, призначені для користувачів, які підключаються [14].

Так як плати Arduino мають досить обмежений обсягом пам'яті, має сенс включати в програму, яка в кінцевому підсумку виявиться в платі, тільки той код, який дійсно буде потрібно. Один із способів досягнення цього – використовувати бібліотеки. В Arduino і взагалі в мові C під бібліотекою розуміється колекція функцій. Наприклад, Arduino IDE включає бібліотеку для роботи зі світлодіодним рідкокристалічним дисплеєм. Вона займає приблизно 1,5 Кбайт пам'яті для програм. Немає ніякого сенсу підключати цю бібліотеку, якщо вона не використовується, тому такі бібліотеки підключаються тільки при необхідності. Підключення виконується додаванням директив `#include` в початок скетчу. Додати інструкції `include` для підключення будь-яких бібліотек, що поставляються в складі Arduino IDE, можна за допомогою пунктів меню Sketch --> Import Library (Скетч --> Підключити бібліотеку) [12].

До складу Arduino IDE входить велика колекція офіційних бібліотек, в тому числі:

- EEPROM - для збереження даних в електрично стираємо програмовану постійну пам'ять (Electrically Erasable Programmable Read-Only Memory);
- Ethernet - для реалізації мережеских взаємодій;
- Firmata - стандартна бібліотека для реалізації взаємодій через

послідовний порт;

- LiquidCrystal - для роботи з алфавітно-цифровими рідкокристалічними дисплеями;
- SD - для читання і запису даних на карти флеш-пам'яті;
- Servo - для управління сервоприводами;
- SPI - для реалізації взаємодій по шині послідовного інтерфейсу;
- Software Serial - для реалізації взаємодій по послідовним лініях звикористанням будь-яких цифрових виходів;
- Stepper - для управління кроковими електромоторами;
- WiFi - для доступу до бездротової мережі WiFi;
- Wire - для реалізації взаємодій по протоколу I2C.
- Деякі бібліотеки призначені для конкретних моделей плат Arduino:
- Keyboard - дозволяє платам Arduino імітувати USB-клавіатуру (Leonardo, Du i Micro);
- Mouse - дозволяє платам Arduino імітувати USB-миша (Leonardo, Due i Micro);
- Audio - утиліти для програвання звуку (тільки Due);
- Scheduler - для управління виконанням декількох потоків (тільки Due);
- USBHost - для підключення USB-периферії (тільки Due).

Існує величезна кількість бібліотек, написаних іншими користувачами Arduino, які можна завантажити з Інтернету. Деякі з них, які користуються особливою популярністю:

- OneWire - для читання даних з цифрових пристроїв з інтерфейсом 1-wire,
- що випускаються компанією Dallas Semiconductor;
- Xbee - для реалізації бездротових взаємодій;
- GFX - графічна бібліотека для роботи з різними дисплеями, що випускаються компанією Adafruit;
- Capacitive Sensing - для роботи з ємнісними датчиками;
- FFT - бібліотека частотного аналізу [12]

Висновки до розділу 1

В результаті виконання першого розділу визначено основні переваги та недоліки різних плат Arduino. Проаналізовані основні аспекти та особливості роботи мікроконтролерів. Досліджено роботу з цифровими та аналоговими сигналами. Підібрані потрібні елементи для створення проекту. Проаналізовані основні моменти роботи та програмування плат Arduino. В результаті аналізу різних плат Arduino, для проекту обрано плату Nano, так, як вона більш всього підходить, враховуючи потрібний функціонал та практичність.

2. АНАЛІЗ РОБОТИ ARDUINO З СВІТЛОДІОДАМИ

1.5 Підключення та управління світлодіодами

Світлодіод – один з найпоширеніших електронних компонентів, що застосовуються в електротехнічних проектах. І природно те, що будь-які DIY проекти починаються з прикладів роботи зі світлом - миготіння, мерехтіння, перемикання. Існує величезна кількість світлодіодів різних варіантів і моделей в різних корпусах і з різними характеристиками [15].

Головне, що потрібно знати про світлодіодах (крім того, що вони світяться) - що це напівпровідниковий прилад. Світло у вигляді випускаються фотонів виникає в спеціальному шарі (pn - переході), коли через нього проходить певний струм.

При підключенні світлодіодів важливо знати два головних правила:

- У світлодіода є позитивний і негативний контакти, тому важливо дотримуватись полярності при підключенні.
- У світлодіодів є обмеження по протіканню через них струму, тому необхідно забезпечувати правильний режим електроживлення. Втім, це стосується будь-яких електронних приладів.

Перше правило легко дотримуватися, якщо знати, де у світлодіода мінус, а де плюс. Тут на допомогу приходять правила маркування. Дивлячись на ніжки, вони різного розміру. Більш довга ніжка означає плюс. Якщо немає можливості порівняти довжину або хтось вже до вас відрізав частину ніжок, обмацуємо корпус (візуально визначити буде складно) - з однієї зі сторін корпус злегка обрізаний (скошений), з цього боку знаходиться мінус [15].

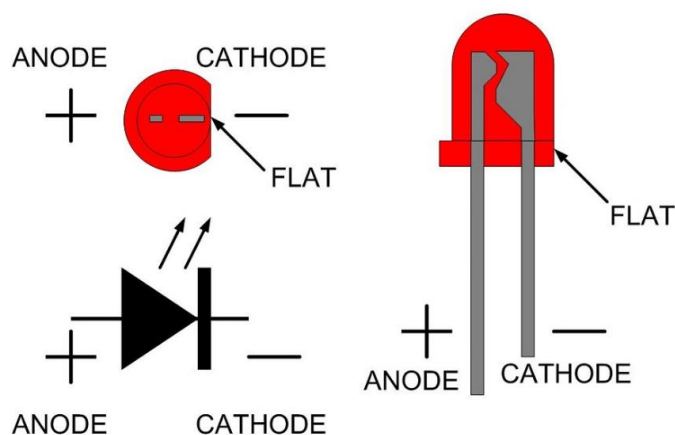


Рисунок 1.12 – Полярність світло діода

Діод, включений неправильно, просто розмикає ланцюг. Природно, до певної межі, тому що якщо подати досить великий струм, то рано чи пізно станеться пробій і пристрій перестане працювати зовсім. Правило обмеження по струму теж легко виконати. Потрібно лише отримати в специфікації до вашого світлодіода значення максимального струму для даної моделі і додати в схему опір, що забезпечує потрібний струм в ланцюзі. Все це легко розраховується за допомогою закону Ома і численних калькуляторів, доступних в інтернеті [15].

Підключення світлодіода до електричного кола - досить проста операція. Потрібно з'єднати відповідні ніжки з плюсом і мінусом. Довгу ніжку підключаємо до тієї частини ланцюга, яка веде в підсумку до плюса. Важливо не забути включити в ланцюг резистор. Щоб підключити світлодіод до Ардуіно треба зрозуміти, де на цій платі буде плюс, де мінус. Адже світлодіод – пристрій електричний, для його роботи потрібна електричний ланцюг з плюсом і мінусом. Перше, що можна зробити – це підключити живлення до виводу 5V плати Uno. Світло буде горіти, але таким пристроєм керувати за допомогою скетчу неможливо. Світло буде горіти до тих пір, поки включено живлення. Звичайно, можна вимикати живлення і відводити плату в режим сну в програмі [15].

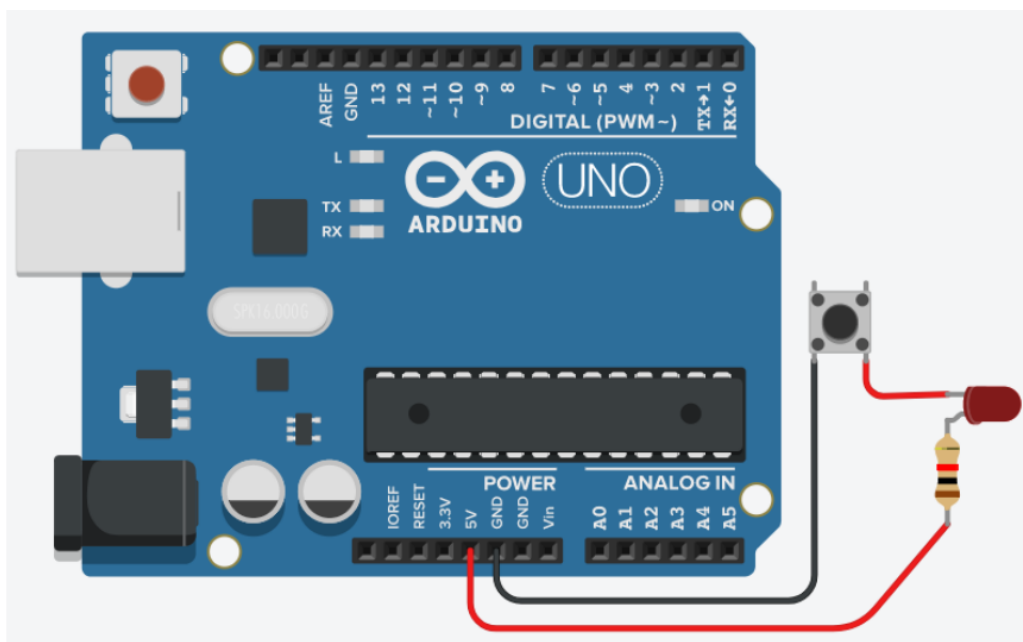


Рисунок 1.13 – Підключення світлодіода безпосередньо до піну живлення Ардуіно

Знову ж правильним варіантом підключення є використання пінів (роз'ємів) плати, на які можна подавати напругу з скетчу. Якщо уявити плату Ардуіно у вигляді батарейки з великою кількістю позитивних і негативних полюсів. Просто вмикаються або вимикаються потрібні роз'єми, подаючи на підключення до них пристрої напругу і струм [15].

Всі мінуси плати позначені одним словом - GND, скорочене від ground – земля («земля» або «корпус» – так часто називають загальну лінію для електричної схеми). На платі Arduino Uno або Nano можна знайти три таких контакту: біля 13 Піна і на іншій стороні, в зоні роз'ємів живлення [15].

Всі інші цифрові або аналогові Піни (з номерами або позначеннями A0-A5) можуть бути і плюсами і мінусами. Якщо ми вкажемо в програмі, що на потрібний порт треба подати напругу (високий рівень сигналу), то він стає плюсом. Якщо вкажемо низький рівень сигналу, то пін стане «мінусом» [15].

Підключивши світлодіод до Ардуіно, отримуємо дуже зручний інструмент для управління ним. Адже не потрібно нічого вмикати або вимикати фізично. Досить просто вказати в програмі потрібні інструкції, щоб сама плата подавала

напругу на потрібні Піни, включаючи або вимикаючи світлодіод. Одного разу завантаживши програму в пам'ять контролера, змусимо його виконувати потрібний алгоритм кожен раз, коли буде підключено живлення.

Включення світлодіода відбудеться в той момент, коли на пін, до якого він підключений, подається високий рівень сигналу (напруга). За це в Ардуіно відповідає функція *digitalWrite* з другим параметром HIGH. Наприклад, для світлодіода, підключеного до пину 12 команда буде виглядати так: *digitalWrite (12, HIGH);*

Щоб вимкнути світлодіод, використовується та ж команду, але з параметром LOW: *digitalWrite (12, LOW)*. Якщо викликати першу команду, а потім через якийсь проміжок часу другу, то у нас світлодіод спочатку загориться, а потім згасне. Таки чином реалізується мерехтіння світлодіода. Додаванням функції *pinMode* (зазвичай в блоці *setup ()*) скореговується пін, як вихід [15].

Для управління яскравістю використовується спеціальна технологія з назвою ШІМ. Підключившись до пінам контролера, що підтримує ШІМ, можна регулювати інтенсивність світла за допомогою функції *analogWrite ()*. Потрібно вказувати їй в якості параметрів номер Піна і значення з діапазону від 0 до 255. Чим більше число, тим яскравіше буде світити лампочка. Наприклад, для Піна 3 приклад буде виглядати так: *analogWrite (3, 255)* або *analogWrite (3, 100)*. У першому випадку яскравість буде максимально можлива, у другому - набагато менше [15].

2.2 Управління світлодіодними стрічками на Arduino

Світлодіодні стрічки RGB представляють собою групи RGB-світлодіодів, зібраних послідовно на гнучкій стрічці з нанесеними на ній провідниками. Світлодіодні стрічки бувають двох типів: аналогові і цифрові. В аналогових стрічках все світлодіоди включені в паралель. Отже, можна задавати колір всієї світлодіодної стрічки, але не можна встановити певний колір для конкретного світлодіода. Ці стрічки прості в підключенні і не дороги. Цифрові світлодіодні

стрічки влаштовані трохи складніше. До кожного світлодіоду додатково встановлюється мікросхема, що робить можливим керувати будь-яким з них окремо. Такі стрічки дорожче звичайних, але мають великі можливості щодо застосування, заходів з світломузикою, світлових шоу, динамічній фоновій підсвідці [16].

Як уже сказано, кожному світлодіоду цифровій RGB-стрічки додатково встановлюється мікросхема (трёхканальний ШІМ-контролер) - наприклад: LPD6803, WS2801, WS2811, WS2812. У даному проекті RGB-світлодіодна стрічка WS2812. Кожен метр стрічки складається з 30-ти SMD-світлодіодів WS2812 типорозміру 5050 [11].

Стрічка "розумна" і управляється за спеціальним цифровим протоколом. Для управління нею використовується спеціальні контролери, тобто якщо просто підключити до стрічки живлення, вона працювати не буде. Можна використовувати готові контролери, але набагато цікавіше та більш функціонально керувати стрічкою вручну, використовуючи, наприклад, платформу Ардуіно, для чого стрічку потрібно правильно підключити [12].

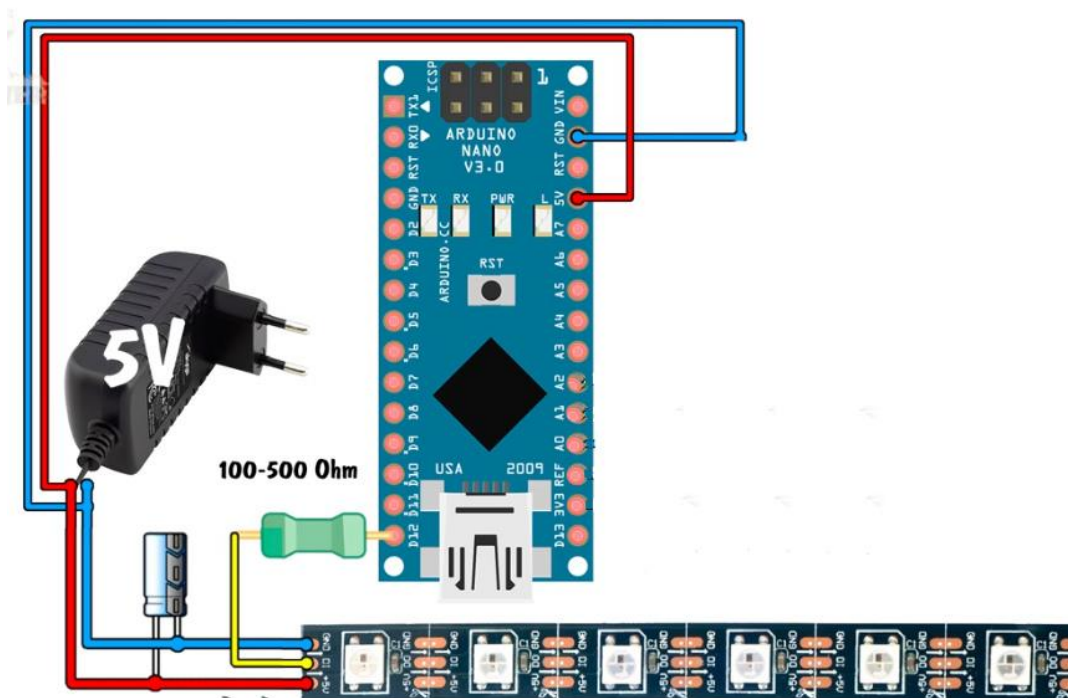


Рисунок 1.14 – Підключення світлодіодної стрічки

На стрічці всього три шини: "земля", живлення і керування. Управління працює так: протягом перших 50 мс відбувається ініціалізація стрічки шляхом заземлення управляючої шини. Після цього контролером відправляється пачка з пакетів по 24 біта (8 бітів на кожний колірний канал) містять інформацію про яскравість, кожен пакет призначається для одного діода. Дані в пачці нічим не розділяють, тобто кожен наступний пакет йде безпосередньо за попереднім. Отримавши всю пачку (довжина пачки дорівнює 24 біта на кількість діодів в стрічці), контролер першого діода бере перші 24 біта, використовує інформацію по призначенню, а решту пачку ретранслює вперед. Таким чином, до останнього діода доходить пачка з одного 24-бітного пакету [17].

Кожен RGB-світлодіод споживає до 60 мА при максимальній яскравості білого (червоний + зелений + синій). В реальних умовах струм буде менше. Але в будь-якому випадку необхідно використовувати зовнішній блок живлення. Перед підключенням стрічки до джерела струму слід підключити конденсатор (1000 μ F, 6,3 В), між виводом Arduino і введенням даних на першому діоді бажано додати резистор від 300 до 500 Ом це захистить секції від початкового сплеску струму.

Команди в стрічці передаються від одного діода до іншого по чергові. Стрічка має початок і кінець, напрямок рух команд на деяких моделях зазначено стрілочками. У випадку даного проекту, а саме з використанням стрічки ws2812b, яка має три контакти. Два з яких подаються на живлення, а третій сигнальний, який на початку стрічки називається DI (digital input), а в кінці - DO (digital output). Стрічка подає команди в контакт DI. Контакт DO потрібен для підключення додаткових секцій стрічки або з'єднання декількох стрічок в матрицю [17].

Якщо між стрічкою і контролером (Arduino) під'єднані довгими дротами (довше 10-15 см), то сигнальний провід і землю потрібно скрутити між собою для захисту від завад, так як протокол зв'язку у стрічки досить швидкісний (800 кГц), на нього сильно впливають зовнішні завади, екранування скруткою допоможе цього уникнути. Без цього може стрічка може не працювати, поки не доторкнеться до сигнального проводу [12].

Дуже важливим є те, що цифровий сигнал проходить по двох дротах, тому для його передачі одного дроту від Ардуіно замало. Контакт стрічки GND і пін Ардуіно GND (будь-який з наявних) повинні бути з'єднані [12].

Продовжуючи тему живлення, треба відзначити те, що дуже важливим є якість пайки, а саме на силових точках (підключення дроту до стрічки, підключення цього ж дроту до БЖ), а також товщину дротів. Для підключення стрічки потрібно брати дріт перетином мінімум 1.5 квадрата, якщо потрібна повна яскравість. Приклад: на дроті 0.75 кв.мм. на довжині 1,5 метра при струмі 2А падає 0.8В, що критично для 5В живлення. Якщо програмно білий колір світить не білим, а жовтим або червоним, то це є ознакою падіння напруги. Чим червоніше, тим сильніше впала напруга! [12].

Миготлива стрічка створює перешкоди на лінію живлення, а у випадку коли стрічка і контролер під'єднані до одного джерела - перешкоди йдуть на мікроконтролер і можуть стати причиною нестабільної роботи, глюків і навіть перезавантаження (якщо БП слабкий). Для згладжування таких перешкод рекомендується ставити електролітичний конденсатор 6.3В ємністю 470 мкФ (ставити більш ємний немає сенсу) по харчуванню мікроконтролера, а також більш "жирний" конденсатор (1000 або 2200 мкФ) на харчування стрічки. Ставити їх необов'язково, але дуже бажано. Якщо ви помітите зависання і глюки в роботі системи (Ардуіно + стрічка + інше залізо), то причиною в 50% є якраз живлення [18].

Шар міді на стрічці не дуже товстий, тому від точки підключення живлення уздовж стрічки напруга починає падати, чим більше яскравість, тим більше просадка. Якщо потрібно зробити великий і яскравий шматок стрічки, то живлення потрібно дублювати мідним дротом 1.5 (або більше, треба експериментувати) квадрата через кожен метр [12].

Висновки до розділу 2

В результаті виконання другого розділу досліджено роботу Arduino з світлодіодами та світлодіодними стрічками. В результаті аналізу та дослідження обрано для даного проекту RGB-світлодіодну стрічку WS2812. В стрічці WS2812 до кожного світлодіоду додатково встановлюється мікросхема, що робить можливим керувати будь-яким з них окремо. Такі стрічки дорожче звичайних, але мають великі можливості щодо застосування, заходів з світломузикою, світлових шоу, динамічній фоновій підсвідці. Кожен метр стрічки складається з 30-ти SMD-світлодіодів WS2812 типу, розміру 50*50.

Проаналізовано можливості реалізації управління світлодіодною стрічкою на Arduino, а також дуже важливий аспект, це правильність живлення, підключення та управління стрічкою.

З ПРАКТИЧНА РЕАЛІЗАЦІЯ СВІТЛОМУЗИКИ ЧЕРЕЗ МОДУЛЬ ARDUINO

Незважаючи на різноманіття видів світлових ефектів в області кіно-, відео зйомки, в різних розважальних закладах та при створенні різного типу контенту при різних умовах зйомки використовують світломузику. Часто виникає необхідність в використанні компактної, зручної та ефективної системи на локації. Враховуючи поширення розважальних закладів, студій, поширенню використання світлових ефектів у кіно, було б доцільно мати портативний прилад за допомогою якого можна зробити веселу та розважальну обстановку майже на будь-якій локації масових заходів.

Для вирішення цієї задачі необхідно створити прилад, за допомогою якого, при наявності мобільного телефону, комп'ютера, зовнішнього приладу на якому є музичні композиції, зробити веселу, розважальну обстановку. А саме реалізувати світломузику.

3.1 Схема створеного пристрою

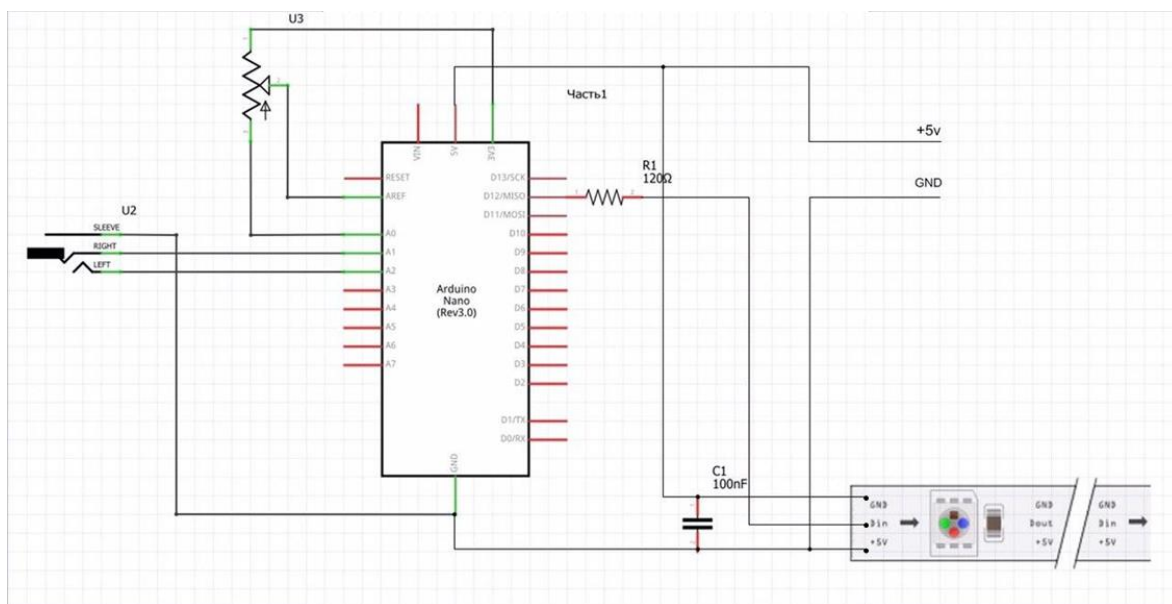


Рисунок 3.1 – Схема принципова електрична

Пристрій складається з наступних елементів:

- Плата Arduino Nano.
- Адресна світлодіодна стрічка ws2812b.
- Імпульсний блок живлення 5В.
- Аудіо гніздо 3.5 мм.
- Потенціометр від 10 до 50кОм.
- Конденсатор 1000 мкФ 6.3 В.
- Резистор номіналом від 100 до 500 Ом

Адресна світлодіодна стрічка ws2812b, яка працює від 5В, має три контакти, два під живлення і один сигнальний. Аудіо кабель який йде від комп'ютера або мобільного телефону, підключається до ардуіно таким чином: центральний канал на пін GND, правий та лівий канали на аналогові піни. Зробити це можна за допомогою звичайного аудіо гнізда 3.5мм.

Для того, щоб знизити опорну напругу на ардуіно для обробки аудіо сигналу, потрібно підключити дільник напруги з резисторів. Це потрібно тому, що сигнал з аудіо штекера має дуже низьку напругу, всього біля 1В, а ардуіно обробляє до 5В. Тому подавши на вхід для опорної напруги ардуіно REF низьку напругу, повідомляємо платі що будемо працювати з низькою напругою. Номіналами резисторів можна налаштувати поріг обробляючої напруги.

Але можна все спростити, замінивши купу резисторів одним потенціометром, як зробили в нашому випадку. Номінал не важливий, він має бути від 10 до 50кОм. Обертаючи потенціометр, ми відповідно можемо підлаштуватися на любую гучність, це все крім того, що налаштування будуть автоматичними.

Підключення світлодіодної стрічки відбувається через резистор номіналом від 100 до 500 Ом на логічний вхід. Стрічка має напівнаправлення. Підключатися потрібно саме до пін DI. Зовнішнє живлення на 5 В, чим потужніше – тим краще, залежить від довжини стрічки. Для захисту від завад на звук, встановлено конденсатор 1000 мкФ 6.3 В.

3.2 Опис елементів системи пристрою

3.2.1 Характеристики Arduino Nano

Nano - одна з найменших плат Arduino. Вона як і Arduino Uno, так само працює на чіпі ATmega328P, але з меншим форм-фактором. Arduino Nano через те, що має габаритні розміри, часто використовується в проектах, де дуже важлива компактність приладу. На платі відсутнє винесене гніздо для зовнішнього живлення, Ардуіно живиться та працює через USB (miniUSB або microUSB). Інші параметри збігаються з моделлю Arduino Uno [19].



Рисунок 3.2 – Зовнішній вигляд Arduino Nano

- Напруга живлення 5В;
- Вхідне живлення 7-12В (рекомендоване);
- Кількість цифрових пінів - 14, з яких 6 можуть використовуватися як виходи ШІМ;
- 8 аналогових входів;
- Максимальний струм цифрового виводу 40 мА;
- Флеш пам'ять 16 Кб або 32 Кб, в залежності від чіпу;
- ОЗУ 1 Кб або 2 Кб, залежить від чіпу;
- EEPROM 512 байт або 1 Кб;
- Частота 16 МГц;
- Розміри 19 x 42 мм;
- Вага 7 м

3.2.2 Характеристики адресної світлодіодної стрічки.

Світлодіоди послідовно ланцюгом з'єднані з трьома мідними контактними доріжками: загальний, живлення та передача даних. Стрічка підключається тільки з одного боку (напрямок вказано біля кожного світлодіода). За допомогою даної led стрічки можна реалізувати різнокольорову керовану підсвітку. Світлодіодна стрічка RGB часто застосовується в якості програмованої повноколірного декоративного підсвічування [20].

На зворотній стороні стрічки завдано клейкий скотч, який дозволяє прикріпити стрічку практично до будь-якої плоскої рівної поверхні. Світлодіодна стрічка WS2812B RGB IC маю адресне управління, тобто кожен світлодіод може управлятися окремо певною програмою за допомогою RGB контроллера. Світлодіодні стрічки часто застосовують для підсвічування банерів для реклами, рекламних щитків, в ландшафтному дизайні і т.д. [20].



Рисунок 3.3 – Зовнішній вигляд адресної світлодіодної стрічки.

- Споживає при максимальному навантаженні 18Вт на метр
- Живлення 4-5В
- Потужність діода 0.3W / LED
- Щільність: 60 / 30 LED / м
- Кут світіння: 120 град
- Ресурс: до 50,000 годин
- Управління по 1 загального проводу, послідовно з'єднує всі світлодіоди. (Або по SPI у APA102, WS2801)
- Стрічка як і окремі WS2812, APA104 тощо. Працюють і з бібліотекою FastLED і Adafruit neopixel.
- Стрічка боїться переполюсовки і перевищення живлення. З Ардуіно починають працювати при рівні напруги 3.5-3.8В

3.2.3 Характеристики пасивних елементів системи

Потенціометр RM063



Рисунок 3.4 – Зовнішній вигляд потенціометра

Технічні характеристики:

- Номінальна потужність: 0,1Вт
- Допуск опору: 20%
- Максимальне напруження: 50В постійного струму
- Предельний ток підвіжного контакту: 20мА
- Опір ізоляції: > 1Гом
- Механічна виносність: 100 циклів
- Діапазон температур: -25 .. + 70 ° С [16]

Резистор

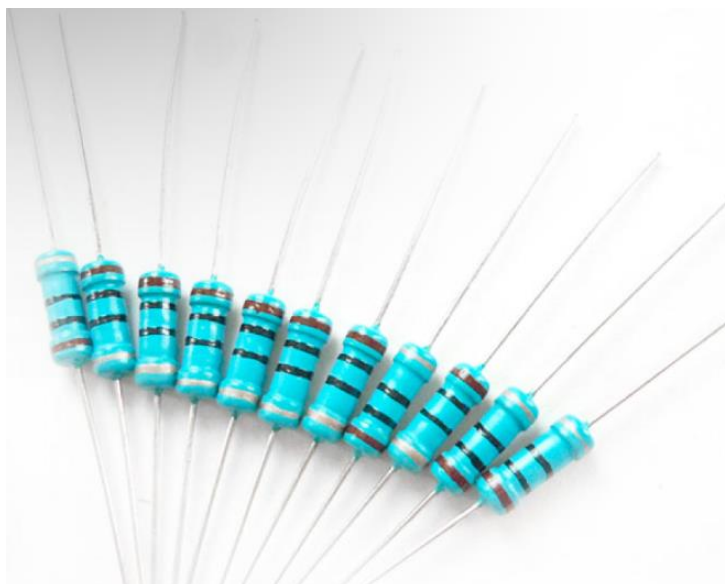


Рисунок 3.5 – Зовнішній вигляд резистора

Технічні характеристики:

- Максимальна розсіювальна потужність: 1 Вт;
- Опір: 100 Ом;
- Відхилення опору: 5%;

Конденсатор



Рисунок 3.5 – Зовнішній вигляд конденсатора

Технічні характеристики:

- Номінальна ємність: 1000мкФ
- Номінальна напруга: 6,3 В
- Робоча температура: 85 ° С, 105 ° С
- Допустиме відхилення ємності $\pm 20\%$
- Тангенс кута втрат, $\text{tg}\delta$ 0,1 - 0,24
- Напрацювання на відмову, щонайменше 2000 год

3.3. Опис алгоритму роботи коду програми

Для початку написання коду програми потрібно розробити алгоритм її роботи. Перед реалізацією передачі звукового сигналу на світлодіодну стрічку

потрібно розробити алгоритм обробки сигналу гучності а потім алгоритм передачі та режиму роботи світлодіодної стрічки.

Для алгоритму режиму гучності спочатку робиться 100 вимірів напруги на АЦП, з них шукається максимальне, фільтрується по нижньому порозі шумів. Для більшої чіткості анімації на стрічці зводиться в степінь, після чого фільтрується «рухомим середнім». Шукається середня гучність за декілька секунд, та максимальна гучність шкали. Наступним кроком переводиться сигнал в кількість запалених світлодіодів та вмикаються світлодіоди згідно режиму.

Алгоритм для реалізації світломузики. Спочатку сигнал розбивається на спектри частот за допомогою перетворення Хартлі, потім фільтрується по нижньому порозі шумів. Шукається максимальні значення в трьох діапазонах (низьке, середнє, високе), та розраховується середня гучність. Якщо сигнал більше середнього то вмикаються світлодіоди. Робиться масив colorMusicFlash в якому три ячейки, 1 або 0, вкл. або викл., та вмикаються відрізки світло діодів згідно масиву.

3.4 Лістинг програми

```
// стрічка
#define NUM_LEDS 60    // кількість світлодіодів
#define BRIGHTNESS 230 // яскравість (0 - 255)

// піни
#define SOUND_R A2      // аналоговий пін вхід аудіо, правий канал
#define SOUND_L A1      // аналоговий пін вхід аудіо, лівий канал
#define SOUND_R_FREQ A3 // аналоговий пін вхід аудіо для режиму с частотами
#define BTN_PIN 3       // кнопка переключення режимів (PIN --- КНОПКА --- GND)
#define LED_PIN 12      // пін DI світлодіодної стрічки
#define POT_GND A0      // пін земля для потенціометра

// налаштування радуги
#define RAINBOW_SPEED 6 // швидкість руху радуги (чим менше число, тим швидше радуга)
```

```

#define RAINBOW_STEP 6    // крок змінення кольору радуги

// відтворення
#define MODE 0            // режим при старті
#define MAIN_LOOP 5       // період основного цикла відтворення (за умовчуванням 5)
#define SMOOTH 0.5        // коефіцієнт плавності анімації VU (за умовчуванням 0.5)
#define SMOOTH_FREQ 0.8   // коефіцієнт плавності анімації частот (за умовчуванням
0.8)
#define MAX_COEF 1.8      // коефіцієнт гучності (максимальне рівне середньому * коеф)
(за умовчуванням 1.8)
#define MAX_COEF_FREQ 1.2 // коефіцієнт порога для "вспішки" світломузики (за
умовчуванням 1.5)

// сигнал
#define MONO 1            // 1 - тільки один канал , 0 - два канали
#define EXP 1.4          // степінь посилення сигналу (за замовчуванням 1.4)
#define POTENT 1          // 1 - використовуємо потенціометр, 0 - використовується
внутрішнє джерело опорної напруги 1.1 В

// нижній поріг шумів
int LOW_PASS = 100;      // нижній поріг шумів режим VU, ручне налаштування
int SPEKTR_LOW_PASS = 40; // нижній поріг шумів режим спектра, ручне налаштування
#define AUTO_LOW_PASS 0   // дозволити налаштування нижнього порогу шумів при
запуску (0)
#define EEPROM_LOW_PASS 1 // поріг шумів зберігається в енергозалежній пам'яті (1)
#define LOW_PASS_ADD 13   // "добавочна" величина до нижнього порогу, для
надійності (режим VU)
#define LOW_PASS_FREQ_ADD 3 // "добавочна" величина до нижнього порогу, для
надійності (режим частот)

// режим світломузики
#define LOW_COLOR RED     // колір низких частот
#define MID_COLOR GREEN   // колір середніх
#define HIGH_COLOR YELLOW // колір високих
// кольори

```

```

#define BLUE    0x0000FF
#define RED     0xFF0000
#define GREEN   0x00ff00
#define CYAN    0x00FFFF
#define MAGENTA 0xFF00FF
#define YELLOW  0xFFFF00
#define WHITE   0xFFFFFFFF
#define BLACK   0x000000

#define STRIPE NUM_LEDS / 5

#define FHT_N 64      // ширина спектра x2
#define LOG_OUT 1
#include <FHT.h>        // Хартлі
#include <EEPROMex.h>

#include "FastLED.h"
CRGB leds[NUM_LEDS];

#include "GyverButton.h"
GButton butt1(BTN_PIN);

// градієнт-палітра від зеленого до червоного
DEFINE_GRADIENT_PALETTE(soundlevel_gp) {
    0,  0,  255, 0, // green
    100, 255, 255, 0, // yellow
    150, 255, 100, 0, // orange
    200, 255, 50,  0, // red
    255, 255, 0,   0 // red
};
CRGBPalette32 myPal = soundlevel_gp;

byte Rlenght, Llenght;
float RsoundLevel, RsoundLevel_f;
float LsoundLevel, LsoundLevel_f;

```

```

float averageLevel = 50;
int maxLevel = 100;
byte MAX_CH = NUM_LEDS / 2;
int hue;
unsigned long main_timer, hue_timer;
float averK = 0.006, k = SMOOTH, k_freq = SMOOTH_FREQ;
byte count;
float index = (float)255 / MAX_CH; // коефіцієнт переведа для палітри
boolean lowFlag;
byte low_pass;
int RcurrentLevel, LcurrentLevel;
int colorMusic[3];
float colorMusic_f[3], colorMusic_aver[3];
boolean colorMusicFlash[3];
byte this_mode = MODE;

#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))

void setup() {
  Serial.begin(9600);
  FastLED.addLeds<WS2811, LED_PIN, GRB>(leds, NUM_LEDS).setCorrection(
TypicalLEDStrip );
  FastLED.setBrightness(BRIGHTNESS);

  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);

  pinMode(POT_GND, OUTPUT);
  digitalWrite(POT_GND, LOW);
  butt1.setTimeout(900);

  // для збільшення точності зменшуємо опорну напругу,
  // виставив EXTERNAL і підключивши Aref до виходу 3.3V на платі через дільник

```

```

// GND ---[10-20 кОм] --- REF --- [10 кОм] --- 3V3
// в данній схемі GND береться з A0 для зручності підключення
if (POTENT) analogReference(EXTERNAL);
else analogReference(INTERNAL);

// змінюємо частоту оцифровки до 18 кГц
sbi(ADCSRA, ADPS2);
cbi(ADCSRA, ADPS1);
sbi(ADCSRA, ADPS0);

if (AUTO_LOW_PASS && !EEPROM_LOW_PASS) {           // якщо дозволена
автоналаштування нижнього порогу шумів
    autoLowPass();
}
if (EEPROM_LOW_PASS) {           // відновити значення шумів з пам'яті
    LOW_PASS = EEPROM.readInt(0);
    SPEKTR_LOW_PASS = EEPROM.readInt(2);
}
}

void loop() {
    butt1.tick(); // обов'язкова функція відпрацювання
    if (butt1.isSingle())           // якщо одиничне натискання
        if (++this_mode > 4) this_mode = 0; // змінюється режим

    if (butt1.isHoled()) { // кнопка зажата
        digitalWrite(13, HIGH); // ввімкнути світлодіод 13 пін
        FastLED.setBrightness(0); // вимкнути стрічку
        FastLED.clear();           // очистити масив пікселів
        FastLED.show();           // відправити значення на стрічку
        delay(500);
        autoLowPass();           // вимряти шум
        delay(500);
        FastLED.setBrightness(BRIGHTNESS); // повернути яскравість
        digitalWrite(13, LOW); // вимкнути світлодіод
    }
}

```

```

}

// кільцеве вимірювання положення райдуги по таймеру
if (millis() - hue_timer > RAINBOW_SPEED) {
    if (++hue >= 255) hue = 0;
    hue_timer = millis();
}

// головний цикл відображення
if (millis() - main_timer > MAIN_LOOP) {
    // сбрасываем значения
    RsoundLevel = 0;
    LsoundLevel = 0;

    // перші два режима - гучність (VU meter)
    if (this_mode == 0 || this_mode == 1) {
        for (byte i = 0; i < 100; i++) {
            // робимо 100 вимірів
            RcurrentLevel = analogRead(SOUND_R); // з правого
            if (!MONO) LcurrentLevel = analogRead(SOUND_L); // і лівого каналів

            if (RsoundLevel < RcurrentLevel) RsoundLevel = RcurrentLevel; // шукаємо
максимальне
            if (!MONO) if (LsoundLevel < LcurrentLevel) LsoundLevel = LcurrentLevel;
        }

        // фільтруємо по нижньому порозі шумів
        RsoundLevel = map(RsoundLevel, LOW_PASS, 1023, 0, 500);
        if (!MONO) LsoundLevel = map(LsoundLevel, LOW_PASS, 1023, 0, 500);

        // обмежуємо діапазон
        RsoundLevel = constrain(RsoundLevel, 0, 500);
        if (!MONO) LsoundLevel = constrain(LsoundLevel, 0, 500);

        // вводим в степінь
        RsoundLevel = pow(RsoundLevel, EXP);
    }
}

```

```

if (!MONO)LsoundLevel = pow(LsoundLevel, EXP);

// фільтр
RsoundLevel_f = RsoundLevel * k + RsoundLevel_f * (1 - k);
if (!MONO)LsoundLevel_f = LsoundLevel * k + LsoundLevel_f * (1 - k);

if (MONO) LsoundLevel_f = RsoundLevel_f; // якщо моно, то лівий = правому

// якщо значення вище порогу
if (RsoundLevel_f > 15 && LsoundLevel_f > 15) {

    // розрахунок спільної середньої гучності з двох каналів, фільтрація.
    averageLevel = (float)(RsoundLevel_f + LsoundLevel_f) / 2 * averK + averageLevel * (1 -
averK);

    // приймаємо максимальну гучність шкали як середню, помножену на деякий
коефіцієнт MAX_COEF
    maxLevel = (float)averageLevel * MAX_COEF;

    // переводимо сигнал в довжину стрічки (де MAX_CH половина кількості
світлодіодів)
    Rlenght = map(RsoundLevel_f, 0, maxLevel, 0, MAX_CH);
    Llenght = map(LsoundLevel_f, 0, maxLevel, 0, MAX_CH);

    // обмежуємо до максимальної кількості світлодіодів
    Rlenght = constrain(Rlenght, 0, MAX_CH);
    Llenght = constrain(Llenght, 0, MAX_CH);

    animation();
}
}

// 3-5 режим - світломузика
if (this_mode == 2 || this_mode == 3 || this_mode == 4) {
    analyzeAudio();
}

```

```

colorMusic[0] = 0;
colorMusic[1] = 0;
colorMusic[2] = 0;
// низькі частоти
for (byte i = 3; i < 6; i++) {
  if (fht_log_out[i] > SPEKTR_LOW_PASS) {
    if (fht_log_out[i] > colorMusic[0]) colorMusic[0] = fht_log_out[i];
  }
}
// середні частоти, вибірка з 6 по 10 тон
for (byte i = 6; i < 11; i++) {
  if (fht_log_out[i] > SPEKTR_LOW_PASS) {
    if (fht_log_out[i] > colorMusic[1]) colorMusic[1] = fht_log_out[i];
  }
}
// високі частоти, вибірка з 11 по 30 тон
for (byte i = 11; i < 31; i++) {
  if (fht_log_out[i] > SPEKTR_LOW_PASS) {
    if (fht_log_out[i] > colorMusic[2]) colorMusic[2] = fht_log_out[i];
  }
}
for (byte i = 0; i < 3; i++) {
  colorMusic_aver[i] = colorMusic[i] * averK + colorMusic_aver[i] * (1 - averK); //
фільтрація
  colorMusic_f[i] = colorMusic[i] * k_freq + colorMusic_f[i] * (1 - k_freq); // локальна
фільтрація
  if (colorMusic_f[i] > ((float)colorMusic_aver[i] * MAX_COEF_FREQ))
    colorMusicFlash[i] = 1;
  else
    colorMusicFlash[i] = 0;
}
animation();
}

FastLED.show(); // відправляємо значення на стрічку

```

```

FastLED.clear();      // очищуємо масив
  main_timer = millis(); // зкидаємо таймер
}
}

void animation() {
  // згідно режиму
  switch (this_mode) {
  case 0:
    count = 0;
    for (int i = (MAX_CH - 1); i > ((MAX_CH - 1) - Rlenght); i--) {
      leds[i] = ColorFromPalette(myPal, (count * index)); // заливка по палітрі " від зеленого
до червоного"
      count++;
    }
    count = 0;
    for (int i = (MAX_CH); i < (MAX_CH + Llenght); i++) {
      leds[i] = ColorFromPalette(myPal, (count * index)); // заливка по палітрі " від зеленого
до червоного"
      count++;
    }
    break;
  case 1:
    count = 0;
    for (int i = (MAX_CH - 1); i > ((MAX_CH - 1) - Rlenght); i--) {
      leds[i] = ColorFromPalette(RainbowColors_p, (count * index) / 2 - hue); // заливка по
палітрі райдуга
      count++;
    }
    count = 0;
    for (int i = (MAX_CH); i < (MAX_CH + Llenght); i++) {
      leds[i] = ColorFromPalette(RainbowColors_p, (count * index) / 2 - hue); // заливка по
палітрі райдуга
      count++;
    }
    break;

```

case 2:

```

for (int i = 0; i < NUM_LEDS; i++) {
if (i < STRIPE)      leds[i] = colorMusicFlash[2] * HIGH_COLOR;
    else if (i < STRIPE * 2) leds[i] = colorMusicFlash[1] * MID_COLOR;
    else if (i < STRIPE * 3) leds[i] = colorMusicFlash[0] * LOW_COLOR;
    else if (i < STRIPE * 4) leds[i] = colorMusicFlash[1] * MID_COLOR;
    else if (i < STRIPE * 5) leds[i] = colorMusicFlash[2] * HIGH_COLOR;
}
break;

```

case 3:

```

for (int i = 0; i < NUM_LEDS; i++) {
    if (i < NUM_LEDS / 3)      leds[i] = colorMusicFlash[2] * HIGH_COLOR;
    else if (i < NUM_LEDS * 2 / 3) leds[i] = colorMusicFlash[1] * MID_COLOR;
    else if (i < NUM_LEDS)      leds[i] = colorMusicFlash[0] * LOW_COLOR;
}
break;

```

case 4:

```

uint32_t this_color;
if (colorMusicFlash[2]) this_color = HIGH_COLOR;
else if (colorMusicFlash[1]) this_color = MID_COLOR;
else if (colorMusicFlash[0]) this_color = LOW_COLOR;
else this_color = BLACK;
for (int i = 0; i < NUM_LEDS; i++) {
    leds[i] = this_color;
}
break;
}
}

```

void autoLowPass() {

 // для режиму VU

 delay(10); // чекаємо ініціалізації АЦП

 int thisMax = 0; // максимум

 int thisLevel;

 for (byte i = 0; i < 200; i++) {

 thisLevel = analogRead(SOUND_R); // робимо 200 вимірів

```

    if (thisLevel > thisMax)          // шукаємо максимум
        thisMax = thisLevel;          // запам'ятовуємо
    delay(4);                          // чекаємо 4мс
}

LOW_PASS = thisMax + LOW_PASS_ADD;    // нижній поріг як максимум тиші + деяка
величина

// для режиму спектра
thisMax = 0;
for (byte i = 0; i < 100; i++) {      // робимо 100 вимірів
    analyzeAudio();                   // разбити в спектр
    for (byte j = 2; j < 32; j++) {    // перші 2 канала
        thisLevel = fht_log_out[j];
        if (thisLevel > thisMax)      // шукаємо максимум
            thisMax = thisLevel;      // запам'ятовуємо
    }
    delay(4);                          // чекаємо 4мс
}

SPEKTR_LOW_PASS = thisMax + LOW_PASS_FREQ_ADD; // нижній поріг як
максимум тиші

if (EEPROM_LOW_PASS && !AUTO_LOW_PASS) {
    EEPROM.updateInt(0, LOW_PASS);
    EEPROM.updateInt(2, SPEKTR_LOW_PASS);
}
}

void analyzeAudio() {
    for (int i = 0 ; i < FHT_N ; i++) {
        int sample = analogRead(SOUND_R_FREQ);
        fht_input[i] = sample; // put real data into bins
    }

    fht_window(); // window the data for better frequency response
    fht_reorder(); // reorder the data before doing the fht
    fht_run();    // process the data in the fht
    fht_mag_log(); // take the output of the fht
}

```

Висновки до розділу 3

В результаті виконання третього розділу розроблено схему для створення проекту. Проаналізовані та підібрані потрібні по характеристикам елементи. Розроблений алгоритм роботи програмного забезпечення проекту. Написаний лістинг програми з реалізацією декількох режимів роботи. Розроблено та реалізовано проект по системі світло-кольорового супроводу звуковідтворення музичних фонограм.

ВИСНОВКИ

Під час виконання дипломної роботи досліджено роботу, функціонал та основні характеристики різних плат Arduino, їхні можливості, та можливості реалізації системи світло-кольорового супроводу звуковідтворення музичних фонограм на цих платформах.

У третьому розділі було створено програмно-апаратний комплекс системи світло-кольорового супроводу звуковідтворення музичних фонограм. Для проектування системи було обрано елементну базу для створення дослідного зразку. Потім було створено алгоритм роботи мікропроцесорного модулю та необхідне програмне забезпечення. В результаті створено систему світло-кольорового супроводу звуковідтворення музичних фонограм.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. В.В. Макаренко, К.О. Трапезон, А.М. Чермянін. Основні вимоги до оформлення атестаційних робіт, дипломних та курсових проектів: методичні рекомендації для студентів усіх форм навчання факультету електроніки. – К.: ФЕЛ НТУУ “КПІ”, 2006. – 112 с.
2. Радиоконструктор RadioKit K165 // RadioKit K165 – URL: <https://electronoff.ua/good/radiokonstruktor-k165-6-ti-kanalnaya-cvetomuzyka-dlya-rgb-lenty-12-24v.php>
3. Бесконтактная цветомузыка для RGB-ленты // Цветомузыка – URL: <https://cxem.net/sound/light/light55.php>
4. Соммер У. Программирование микроконтроллерных плат Arduino/Freduino. - СПб.: БХВ- Петербург, 2012. - 256 с. ил - (Электроника)
5. Принципы фон Неймана // Фон Нейман – URL: <http://informatics-lesson.ru/ware/fon-neumann.php>
6. Гарвардская архитектура – URL: <https://studfile.net/preview/1043400/page:4/>
7. Francis Perea. Arduino Essentials. Published by Packt Publishing Ltd. Livery Place 35. Livery Street. Birmingham B3 2PB, UK.
8. Ревич Ю. В. Занимательная электроника. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2015. — 576 с.: ил. (Электроника)
9. Огляд плат Arduino // плати Arduino – URL: http://geekmatic.in.ua/ua/the_different_arduinos
10. Блум Джереми Изучаем Arduino: инструменты и методы технического волшебства: Пер. с англ. — СП.: БХВ-Петербург, 2015. .336 с.: ил. (Электроника)
11. Петин В. А. Проекты с использованием контроллера Arduino.-2-е изд, перераб. и доп.-СПб.: БХВ-Петербург, 2015.-464 с.: ил. -(Электроника)
12. С. Монк. Програмуємо Arduino. Професійна робота со скетчами . — СПб.: Питер, 2017.
13. Максим Иванов (aka e-maxx). Сборник алгоритмов на C++ // Алгоритмы

на C++ – URL: <http://e-maxx.ru/algo/>

14. Основы программирования Ардуино на языке C // Программирование Ардуино – URL: <http://mypractic.ru/urok-4-osnovy-programmirovaniya-arduino-na-yazyke-c.html#3https://developers.google.com/ar/>

15. Подключение светодиода к Ардуино – URL: <https://arduinomaster.ru/uroki-arduino/podklyuchenie-svetodioda-k-arduino/>

16. ЦВЕТОМУЗЫКА НА ARDUINO – URL: <https://alexgyver.ru/colormusic/>

17. АДРЕСНАЯ СВЕТОДИОДНАЯ ЛЕНТА // WS2812 – URL: https://alexgyver.ru/ws2812_guide/

18. Электрик Инфо. Ардуино для чайников. Иллюстрированное практическое руководство. 2018

19. Плата Arduino Nano v 3.0 : распиновка, схемы, драйвер // Arduino Nano – URL: <https://arduinomaster.ru/platy-arduino/plata-arduino-nano/>

20. Адресная светодиодная лента и её подключение к Arduino – URL: <https://ledjournal.info/spravochnik/adresnaja-svetodiodnaja-lenta.html>

21. Подстроечные резисторы // Резисторы – URL: <https://roboshop.spb.ru/RM063-103>

22. Том Иго. Arduino, датчики и сети для связи устройств. БХВ-Петербург 2015.

Додаток А
SUMMARY

SUMMARY

The relevance of the study of the system of control of light effects is the spread of entertainment facilities and various techniques of video editing using light effects. Convenience and efficiency of using light effects control systems directly depends on the quality of the final content.

As a result of the thesis work created systems of light-color accompaniment of sound reproduction of musical phonograms, its characteristics, advantages and disadvantages, examples of application in life and features of combination with the real world.

LED is one of the most common electronic components used in electrical projects. And naturally, any DIY projects start with examples of working with light - flickering, flickering, switching. There is a huge number of LEDs of different variants and models in different cases and with different characteristics. RGB LED strips are groups of RGB LEDs assembled in series on a flexible strip with conductors applied to it. LED strips are of two types: analog and digital. In analog tapes, all LEDs are connected in parallel. Therefore, you can set the color of the entire LED strip, but you cannot set a specific color for a specific LED. These tapes are easy to connect and not expensive. Digital LED strips are a little more complicated. A chip is additionally installed for each LED, which makes it possible to control any of them separately.

As already mentioned, each LED of the digital RGB-tape is additionally installed chip (three-channel PWM controller) - for example: LPD6803, WS2801, WS2811, WS2812. In this project RGB-LED strip WS2812. Each meter of tape consists of 30 SMD-LEDs WS2812 size 5050.

The tape is "smart" and is controlled by a special digital protocol. This means that if you just plug in the power tape, nothing will happen, that is, you can not check the tape without a control controller. Ready-made controllers are used to control the tape, but it is much more interesting to control the tape manually, using, for example, the Arduino platform, for which the tape must be connected correctly

WS2812 is three LEDs (red, blue, green) and an 8-bit PWM driver for them in one

housing. Three wires are used to connect the tape. + 5V contact to which plus is supplied from the power supply, GND - minus the power supply. The control is performed by applying a 24-bit signal for each LED (8 bits per color). The signal from the Arduino is fed to the Din pin of the first LED, which, in turn, writes the first 24 bits to its temporary memory and according to them, sets the glow color, cuts off these first 24 bits from the signal and sends the signal through the Do pin next to the next LEDs. Therefore, the tape has a direction. It is indicated on the tape by an arrow, or from Din to Do contact.

The more LEDs, the more current flows. WS2812 is very voracious and one diode consumes (at a maximum brightness of all three colors) 0.06 A. Therefore, the + 5V and GND power wires must be selected for the maximum current flowing through them. I used a copper flexible wire with a cross section of 0.5 mm (with a margin and to further increase the length of the tape). The wire for the signal contacts is not so demanding on the cross section, the current flowing through it is small, so we take a thin flexible wire.

Color music works as follows: divides the number of LEDs in half, the music volume is indicated by the diode lighting from the middle to the ends of the entire tape. Therefore, if you want two bands working like on my video, both tapes are connected in parallel (the wire from the Din contacts of both tapes to one Arduino pin, D12). And then we indicate the number of LEDs in the sketch equal to the sum of the diodes of both tapes. If you want the tape to work from the middle, connect two strips in series and also indicate the total number of LEDs.

The choice of power supply must be approached thoroughly. We need 5 volt stabilized, we select the current strength depending on the number of LEDs, it is better to take with a margin. I recommend, for example, such a 3 A.

A potentiometer is needed to set the reference voltage to measure the level of the audio signal. Contact A0 is used as GND (for Arduino Nano it is more convenient to connect). You can do without a potentiometer by turning on the use of the internal reference voltage in the sketch.

Contacts A1, A2 are used to connect the audio signal, do not forget to connect the Arduino GND contacts and the audio signal wires.

Button, used to switch modes, connects to pin D3 Arduino.

The connection of the socket and contact A3 through the capacitor is necessary for some modes to work. So that we can listen to music, and our color music work, we need to separate the audio signal. You can do this using a tee.

You can also connect speakers to the audio output on the motherboard, and color music to the headphone output. The main thing is that you can't connect the color music to the output after the amplifier, there is too much signal level for the Arduino. First you need to download the latest version of the Arduino IDE development environment.

After starting the color music, proceed as follows: first you need to configure the lower limit of the sound level. To do this, pause the music, hold and hold the button until the diode on the Arduino board lights up. Then we turn on the music and see if all the tape lights up when the music is working. If the tape only works at the very beginning, or vice versa, the potentiometer always turns on.

The sound stream coming to the PC gets to the device by default, that is, to the input of the “virtual audio cable”, and its output serves as a sound source in the color music program.

If you use another music player and / or you do not have an amplifier or an audio receiver with airplay, you will need a “virtual audio cable” that can switch the audio stream to several devices. You must supply one of the outputs to your output sound device (output of a sound card or PC speakers), and the other should be indicated as a sound source in the color music program.

The third option will require the availability or manufacture of an audio cable. You can use the line input of your sound card as a sound source in the CMU program. In this case, you can use any external sound source (audio player, MP3 player, etc.) with a linear output.

You will have to deal with the “virtual audio cable” with the splitter and the cable manufacturing option yourself, I did not need these options. Perhaps there are other options for organizing the sound source for the CMU program.

The profile for installing the LED strip or another option for attaching the LED strip is selected from your desires.