

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки
Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему РОЗРОБКА КОМП'ЮТЕРНОГО МОДУЛЯ ГЕНЕРАЦІЇ ЗВУКОВИХ І
СВІТЛОВИХ СИГНАЛІВ ДЛЯ ДИТЯЧОГО ЦЕНТРУ

Виконав: студент 4 курсу, групи КНТ-527
спеціальності _____

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____

«Комп'ютерна інженерія»

Бороденко Віталій Геннадійович

(прізвище та ініціали)

Керівник Проскурін М.П.

(прізвище та ініціали)

Рецензент Куляба-Харитоновна Т.І

(прізвище та ініціали)

м. Запоріжжя
2021 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій
 кафедра «Комп'ютерні системи та мережі»
 ступінь вищої освіти (освітній ступінь) бакалаврський
 спеціальність 123 Комп'ютерна інженерія
(код і найменування)
 освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри Кудерметов Р.К.

« » 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА

Борошенко Віталію Геннадійовичу

(прізвище, ім'я, по батькові)

Тема проекту (роботи) Розробка комп'ютерного модуля генерації звукових і
тлових сигналів для дитячого центру

Рівень проекту (роботи) Прокурін Микола Петрович, к. т. н., доцент

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

Затверджені наказом вищого навчального закладу від «17» березня 2021 року № 81

Строк подання студентом проекту (роботи) 06 травня 2021 року

Вихідні дані до проекту (роботи) платформа Arduino та її модулі, середовище розробки
Arduino IDE, алгоритми реалізації зв'язку між кількома платами Arduino

Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно
 зробити) 1) Аналіз предметної області;

Вибір апаратного забезпечення та розробка схем для реалізації системи;

Програмна модель системи;

Реалізація та дослідження системи.

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

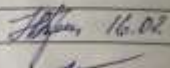
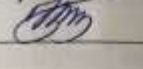
Порівняння апаратних комплектуючих модуля;

Алгоритм програми;

Електрична схема модуля;

Демонстрація роботи.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконання завдання
1-3	Проскурін М.П. к. т. н., доцент	 16.02	
Нормоконтроль	Щербак Н.В., ст.викладач		

7. Дата видачі завдання 01.03.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Прогноз
1	Аналіз технічного завдання	01.03.2021 р.	
2	Вибір апаратного забезпечення	15.03.2021 р.	
3	Розробка алгоритму роботи системи	20.03.2021 р.	
4	Реалізація системи	01.04.2021 р.	
5	Дослідження системи	10.04.2021 р.	
6	Оформлення отриманих результатів у ПЗ	15.04.2021 р.	
7	Оформлення графічного матеріалу	20.04.2021 р.	
8	Оформлення допоміжного матеріалу	01.05.2021 р.	

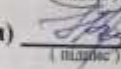
Студент



В.Г. Бороденко

(ініціали та прізвище)

Керівник проекту (роботи)



М.П. Проскурін

(ініціали та прізвище)

ПЗ: 70 с., 26

ARDUINO
МІКРОКОНТРОЛЕР
СИСТЕМА ЗВУКУОб'єкт розробки
Arduino.

Мета роботи

Пояснювальна

аналізу технічного завдання зроблено висновок про розміри, та на апаратної частини модуля частини модуля світлодіодну систему. Розглянуто основні параметри створено скетч частини. Звукова частина виконує роль Четвертий розділ комплекту, світлодіодна частина для підключення

РЕФЕРАТ

ПЗ: 70 с., 26 рис., 2 табл., 12 джерел.

ARDUINO NANO, КОМП'ЮТЕРНИЙ МОДУЛЬ, МІКРОКОМП'ЮТЕР, МІКРОКОНТРОЛЕР, ПЕРЕДАВАЧ, ПРИЙМАЧ, СВІТЛОДІОДНА СТРИЧКА, СИСТЕМА ЗВУКОВОГО КЕРУВАННЯ

Об'єкт розробки – система генерації світлових та звукових сигналів на базі Arduino.

Мета роботи – розробити комп'ютерний модуль.

Пояснювальна записка складається з 4 розділів. Перший розділ присвячено аналізу технічного завдання. Було проаналізовано кілька типів плат Arduino та зроблено висновок, що найкраще підходить саме Arduino Nano, через її невеликі розміри, та наявність програматору на її борту. Другий розділ описує вибір апаратної частини комп'ютерного модуля. При реалізації системи світлової частини модуля в якості самого об'єкту світла було використано адресну світлодіодну стрічку WS2812B. Третій розділ присвячено програмній розробці. Розглянуто основи роботи в фірмовому середовищі програмування Arduino IDE, створено скетчі (програми для завантаження на Arduino), та розглянуто їх головні частини. Звукова частина модуля реалізована як комп'ютерна програма, яка виконує роль 20 полосного аналізатора спектра і пульта управління програмами. Четвертий розділ представляє реалізацію системи. Для складання мінімального комплекту, з безпосереднім підключенням до USB ПК, потрібна лише світлодіодна стрічка з WS2812B, arduino nano, три провідника і міні USB кабель для підключення плати Arduino до одного з USB портів ПК.

ЗМІСТ

Скорочення та умовні позначки	6
Вступ.....	7
1 Аналіз технічного завдання.....	9
1.1 Аналіз вихідних даних і вимог	9
1.2. Сімейство мікрокомп'ютерів Arduino.....	11
1.3 Висновки до розділу 1	15
2 Вибір апаратного забезпечення та розробка схем для реалізації системи.....	16
2.1 Мікрокомп'ютер Arduino Nano.....	16
2.2 Світлова частина	19
2.3 Висновки до розділу 2	23
3 Програмна модель системи	24
3.1 Розробка програми	24
3.2 Програмування мікроконтролера	28
3.3 Висновки до розділу 3	31
4 Реалізація системи.....	31
4.1 Збирання схеми.....	31
4.3 Інструкція користувача.....	34
4.4 Результати перевірки системи	35
4.4 Висновки до розділу 4	36
Висновки	38
Перелік джерел посилання	40
Додаток А.....	41
Перелік графічного матеріалу:	
Плакат 1 – Порівняння апаратних комплектуючих модуля	
Плакат 2 – Електрична схема модуля	
Плакат 3 – Алгоритм програми	
Плакат 4 - Демонстрація роботи	

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

LED	-	Light-emitting diode
RGB	-	Red-Green-Blue
SMD	-	Surface Mounted Device
IC	-	інтегральна схема
ПЗ	-	програмне забезпечення
ПК	-	персональний комп'ютер
ШИМ	-	широтно-імпульсна модуляція

ВСТУП

Arduino - це електронний конструктор і зручна платформа швидкої розробки електронних пристроїв [1]. Можна сказати, що Arduino - сполучна ланка між віртуальним комп'ютерним світом і навколишнього фізичним середовищем. При написанні програми на звичайному комп'ютері з Arduino з'являється можливість управляти не віртуальними об'єктами, а реальними двигунами, датчиками і світлодіодами.

Творцем Arduino прийнято вважати викладача інституту IDI (Interaction Design Institute Ivrea) італійського р Івреї Массімо Банц. Платформа виникла в 2005 році у зв'язку з завданням навчити студентів швидко і якісно створювати електронні проекти. Вона стала зручним і простим інструментом для навчання студентів програмуванню і дуже скоро - причиною міжнародної революції в світі електроніки. Массімо Банц подарував всім людям можливість створювати свої електронні пристрої, при цьому не витрачаючи багато часу на вивчення теоретичного матеріалу [2].

До появи Arduino існували й інші мікроконтролери (наприклад, мікроконтролер BASIC Stamp, на який спирався Массімо Банц при створенні Arduino). Але у всіх мікроконтролерів того часу був істотний недолік - дуже висока вартість. Через це використовувати мікроконтролери в навчанні було практично неможливо [3].

Платформа Arduino отримала велику популярність у всьому світі завдяки своїй доступності, зручності використання, простоті мови програмування, відкритій архітектурі і програмного коду. Електронні пристрої, створені за допомогою Arduino, дають можливість вашому комп'ютеру звертатися до фізичним світом. Ці пристрої за допомогою різних датчиків можуть отримувати інформацію про навколишнє середовище. Наприклад, до плати Arduino можна підключати датчики тиску, температури, механічного напруги, інфрачервоний датчик, світлодіод, графічний дисплей і інші пристрої.

Сьогодні на основі платформи Arduino можна створювати різні системи домашньої автоматизації, світлодіодні куби, музичні інструменти, різноманітні дисплеї, роботизовані маніпулятори і багато іншого.

Метою бакалаврської роботи є розробка комп'ютерного модуля генерації звукових і світлових сигналів для дитячого центру. Для досягнення мети необхідно виконати наступні завдання:

- аналіз технічного завдання;
- вибір апаратного забезпечення комп'ютерного модуля;
- розробка схеми комп'ютерного модуля;
- опис алгоритмів функціонування;
- програмування мікроконтролера.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Аналіз вихідних даних і вимог

Для розробки проекту можна вибрати три платформи: Arduino, BeagleBone, RaspberryPi. Ці три моделі були обрані для порівняння не випадково: всі вони реально доступні, виконані в компактному форм-факторі і можуть використовуватися для створення різних цифрових пристроїв. Платформу Arduino можна вважати основним компонентом в товаристві радіоаматорів. Зараз плати Arduino доступні в різних форм-факторах, з різним набором периферії. Велика їх частина виконана на 8-розрядному мікроконтролері компанії Atmel. Головний плюс плат Arduino – порівняно просте проектування та збірка макетів. Програмування мікроконтролера здійснюється через спеціальну консоль, яка одночасно є середовищем розробки і терміналом послідовного порту. Завантаживши програмний код в пам'ять мікроконтролера, спеціальною кнопкою на платі (або командою «логічна одиниця») на обраний цифровий порт вводу / виводу, запускається програмний код на циклічне виконання.



Рисунок 1.1 – Платформа Arduino Nano

Плата Raspberry Pi (рис. 1.2) є новачком на ринку - це одноплатний комп'ютер, що позиціонується на ринку як дешеве рішення для розробників вбудованих систем. Незважаючи на скромний вигляд і низьку вартість, ця платформа є окремим комп'ютером, для розробки безлічі проектів. Використовує SD - карту для зберігання прикладного та системного програмного забезпечення. Є можливість підключення широкого асортименту інтерфейсів, зокрема, мереж

Ethernet, 3G, різного роду дисплеїв. Використовує мікропроцесор ARM1176JZF-S з тактовою частотою 700 мегагерц.



Рисунок 1.2 – Одноплатний комп'ютер Raspberry Pi

BeagleBone (рис. 1.3) це потужний Linux-комп'ютер компактних розмірів, який підтримує роботу з ОС Android і Ubuntu . Загалом, ця плата подібна до Raspberry Pi, однак відрізняється більшою обчислювальною потужністю і можливостями щодо розширення. Система побудована на обчислювальному процесорі TI Sitara AM335x з ядром A8 Cortex, якого вистачає навіть для роботи достатньо потужних планшетів.

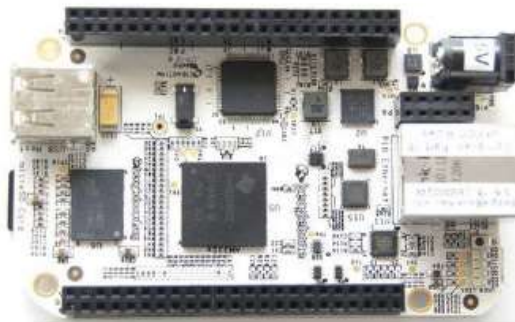


Рисунок 1.3– Одноплатний комп'ютер BeagleBone

Arduino - це досить функціональна і гнучка платформа розробки, вбудованих додатків з величезними можливостями для взаємодії із зовнішнім світом. Вона прекрасно, підійде для вивчення мікроконтролерів і може служити основою для невеликих проєктів. Raspberry Pi буде оптимальним вибором для додатків, що вимагають наявності дисплея, графічного призначеного для

користувача інтерфейсу і підключення до Інтернет, взаємодії з різного роду датчиками. У платформі BeagleBone ідеально поєднана гнучкість Arduino, продуктивність процесора плати Raspberry Pi і ОС Linux. Маючи достатню кількість входів / виходів, BeagleBone забезпечує просте підключення до мережі і дозволяє реалізувати web-сервер [1]. Для вирішення поставлених задач ми обираємо сімейство мікрокомп'ютерів Arduino за просте проектування та налаштування проектів, доступність та низький ціновий рівень.

1.2 Сімейство мікрокомп'ютерів Arduino

Arduino (Ардуіно) — апаратна обчислювальна платформа для аматорського конструювання, основними компонентами якої є плата мікроконтролера з елементами вводу/виводу та середовище розробки Processing/Wiring на мові програмування, що є підмножиною C/C++. Arduino може використовуватися як для створення автономних інтерактивних об'єктів, так і підключатися до програмного забезпечення, яке виконується на комп'ютері (наприклад: Processing, Adobe Flash, Max/MSP, Pure Data, SuperCollider). Інформація про плату (рисунок друкованої плати, специфікації елементів, програмне забезпечення) знаходяться у відкритому доступі і можуть бути використані тими, хто воліє створювати плати власноруч [4].

Плата Arduino складається з мікроконтролера Atmel AVR, а також елементів обв'язки для програмування та інтеграції з іншими пристроями. На багатьох платах наявний лінійний стабілізатор напруги +5В або +3,3В. Тактування здійснюється на частоті 16 або 8 МГц кварцовим резонатором. У мікроконтролер записаний завантажувач (bootloader), тому зовнішній програматор не потрібен.

На концептуальному рівні усі плати програмуються через RS-232 (послідовне з'єднання), але реалізація даного способу відрізняється від версії до версії. Новіші плати програмуються через USB, що можливо завдяки мікросхемі

конвертера USB-to-Serial FTDI FT232R. У версії платформи Arduino Uno як конвертер використовується контролер Atmega8 у SMD-корпусі. Дане рішення дозволяє програмувати конвертер таким чином, щоб платформа відразу розпізнавалася як миша, джойстик чи інший пристрій за вибором розробника зі всіма необхідними додатковими сигналами керування. У деяких варіантах, таких як Arduino Mini або неофіційній Boarduino, для програмування потрібно підключити до контролера окрему плату USB-to-Serial або кабель [5].

Плати Arduino дозволяють використовувати значну кількість виводів мікроконтролера як вхідні/вихідні контакти у зовнішніх схемах. Наприклад, у платі Decimila доступно 14 цифрових входів/виходів, 6 із яких можуть генерувати ШІМ сигнал, і 6 аналогових входів. Ці сигнали доступні на платі через контактні площадки або штирові роз'єми. Також існує багато різних зовнішніх плат розширення, які називаються «shields» («щити»), які приєднуються до плати Arduino через штирові роз'єми.

Ардуіно і Ардуіно-сумісні плати спроектовані таким чином, щоб їх можна було при необхідності розширювати, додаючи в пристрій нові компоненти («shields»). Ці плати розширень підключаються до Ардуіно за допомогою 20 встановлених на них штирових роз'ємів. Існує ряд уніфікованих плат, що допускає конструктивно жорстке з'єднання процесорної плати та плат розширення в стопку через штирові лінійки. Крім того, випускаються плати зі зменшеним (наприклад, Nano, Lilypad) і спеціальним (для задач робототехніки) форм-фактором [6].

Сторонніми виробниками випускається велика гамма всіляких датчиків і виконавчих пристроїв, в тій чи іншій мірі сумісних між собою і з процесорними платами Ардуіно. Сторонніми виробниками також випускаються набори електромеханічних елементів, орієнтованих на роботу спільно з платами Ардуіно (як правило, через спеціальні плати-«драйвери») — двигуни, електромагніти тощо.

У концепцію Ардуіно не входить корпусних або монтажних деталей. Розробник вибирає метод установки і механічного захисту процесорних плат та

компонентів розширення самостійно [7].

Arduino Uno є стандартною платою Arduino і можливо найбільш поширеною (рис.1.4). Вона заснована на чіпі ATmega328, що має на борту 32 КБ флеш-пам'яті, 2 КБ SRAM і 1 Кбайт EEPROM пам'яті. На периферії має 14 дискретних (цифрових) каналів введення / виводу і 6 аналогових каналів введення / виводу, це дуже різнобічно-корисні девайси, що дозволяють перекривати більшість аматорських завдань в області мікроконтроллерной техніки. Дана плата контролера є однією з найдешевших і найбільш часто використовуваних. При плануванні нового проекту, якщо ви не знайомі, з платформою Arduino, радимо почати з Uno.



Рисунок 1.4 - Arduino Uno

Arduino Nano - це функціональний аналог Arduino Uno, але розміщений на мініатюрній платі (рис.1.1). Відмінність полягає у відсутності власного гнізда для зовнішнього живлення, використанням чіпа FTDI FT232RL для USB-Serial перетворення (або CH340G, потрібно встановити відповідні драйвера) і застосуванням mini-USB кабелю для взаємодії замість стандартного. В іншому, начинка і способи взаємодії збігаються з базовою моделлю. Платформа має Штиркові контакти, що дозволяє легко встановлювати її на макетну плату. Використовуйте Arduino Nano там, де важлива компактність, а можливостей Arduino Mini чи мало, або не хочеться займатися паянням.

Arduino Mega як Arduino Uno, але на базі більш потужного мікроконтролера

тієї ж архітектури (рис. 1.5). Відмінний вибір «на виріст» або якщо Arduino Uno перестала справлятися. В рази більше пам'яті: 256 КБ проти 32 КБ постійної і 8 КБ проти 2 КБ оперативної. В рази більше портів: 60 з них 16 аналогових і 15 з ШІМ. Трохи довше базової Arduino Uno: 101×53 мм проти 69×53 мм



Рисунок 1.5 - Arduino Mega

Arduino Mini та ж Arduino Uno, але в іншому форм-факторі. Компактна: всього 30×18 мм (рис. 1.6). Через форм-фактора не можна без хитрощів встановлювати плати розширення Arduino. Передбачається з'єднання з додатковими модулями проводами і / або через макетну плату. На платі немає USB-порту, тому прошивати потрібно через окремий USB-Serial адаптер.

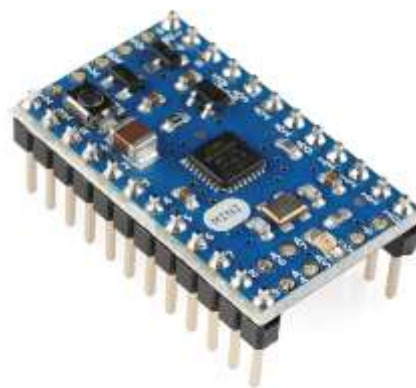


Рисунок 1.6 - Arduino Mini

Під час огляду типів плат Arduino були розглянуті технічні характеристики Uno, Nano, Pro Mini, Mega (див. таблицю 1.1).

Таблиця 1.1 – Порівняльна таблиця плат Arduino

Характеристика	Arduino Uno	Nano	Pro Mini	Mega
Мікроконтролер	ATmega328	ATmega328	ATmega328	ATmega2560
Робоча напруга	5 В	5 В	3,3 або 5В	5 В
Напруга живлення	7-12 В	7-12 В	3,35-12 або 5-12 В	7-12 В
Цифрові виходи	14	14	14	54
Канали ШІМ	6	6	6	15
Аналогові виходи	6	8	8	16
FLASH-пам'ять	32КБ	32 КБ	16 КБ	256 КБ
SRAM	2КБ	2 КБ	1 КБ	8 КБ
Тактова частота	16МГц	16 МГц	8 або 16 МГц	16 МГц
Розмір	68x53 мм	18x43 мм	18x33 мм	101x53 мм
Вбудований програматор	Так	Так	Ні	Так

1.3 Висновки до розділу 1

Arduino - це досить функціональна і гнучка платформа розробки, вбудованих додатків з величезними можливостями для взаємодії із зовнішнім світом. Вона прекрасно, підійде для вивчення мікроконтролерів і може служити основою для невеликих проектів. Raspberry Pi буде оптимальним вибором для додатків, що вимагають наявності дисплея, графічного призначеного для користувача інтерфейсу і підключення до Інтернет, взаємодії з різного роду датчиками. У платформі BeagleBone ідеально поєднана гнучкість Arduino, продуктивність процесора плати Raspberry Pi і ОС Linux. Маючи достатню кількість входів / виходів, BeagleBone забезпечує просте підключення до мережі і дозволяє реалізувати web-сервер. Для вирішення поставлених задач ми обираємо

сімейство мікрокомп'ютерів Arduino за просте проектування та налаштування проектів, доступність та низький ціновий рівень.

В ході порівняння чотирьох видів плат Arduino було зроблено висновок, що майже всі розглянуті варіанти мають однакові характеристики і найбільше відрізняються лише наявністю програматорів, розмірами та метою використання, тому для створення системи дистанційного керування найкраще підійде Arduino Nano, через її невеликий розмір (на відміну від Uno та Mega) та наявність програматору (на відміну від Pro Mini).

2 ВИБІР АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ ТА РОЗРОБКА СХЕМ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

2.1 Мікрокомп'ютер Arduino Nano

Для виконання бакалаврської роботи я буду використовувати плату Arduino на платформі Nano (рис. 2.1). Плата Arduino складається з мікроконтролера Atmel AVR (ATmega328P), а також елементів обв'язки для програмування та інтеграції з іншими схемами. На багатьох платах присутній лінійний стабілізатор напруги +5В або +3,3В. Тактування здійснюється на частоті 16 або 8 МГц 10 кварцовим резонатором (в деяких версіях керамічним резонатором). У мікро-контролер попередньо прошивається завантажувач BootLoader, тому зовнішній програматор не потрібен.

мають особливі функції:

Послідовна шина: 0 (RX) і 1 (TX). Виводи використовуються для отримання (RX) і передачі (TX) даних TTL. Дані виводи підключені до відповідних виводів мікросхеми послідовної шини FTDI USB-to-TTL.

Будь-який з виводів забезпечує ШІМ з роздільною здатністю 8 біт за допомогою функції `analogWrite ()`.

ШІМ: 3, 5, 6, 9, 10, і 11. Будь-який з виходів забезпечує ШІМ з роздільною здатністю 8 біт за допомогою функції `analogWrite ()`.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). За допомогою даних виводів здійснюється зв'язок SPI, яка, хоча і підтримується апаратною частиною, не включена в мову Arduino.

LED: 13. Вбудований світлодіод, підключений до цифрового виводу 13. Якщо значення на виведення має високий потенціал, то світлодіод горить.

На платформі Nano встановлені 8 аналогових входів, кожен дозволом в 10 біт (тобто може приймати 1024 різних значення). Стандартно висновки мають діапазон вимірювання до 5 В щодо землі, проте є можливість змінити верхню межу за допомогою функції `analogReference ()`. Деякі виводи мають додаткові функції: I2C: A4 (SDA) і A5 (SCL). За допомогою виводів здійснюється зв'язок I2C (TWI).

В текст програми (скетчу) програміст не зобов'язаний вставляти заголовки використовуваних стандартних бібліотек. Ці заголовки додасть передпроцесор Arduino відповідно до конфігурації проекту. Однак призначені для користувача бібліотеки потрібно вказувати.

Слід зазначити, що менеджер проекту Arduino IDE має нестандартний механізм додавання бібліотек. Бібліотеки у вигляді вихідних текстів на стандартному C ++ додаються в спеціальну папку в робочому каталозі IDE. При цьому назва бібліотеки буде додано до списку бібліотек в меню IDE. Програміст зазначає потрібні бібліотеки і вони вносяться до списку компіляції.

Плати Ардуіно здатні управляти світлодіодними конструкціями по заздалегідь заданими програмами. Їх бібліотеки можна скачати з офіційно сайту,

знайти в інтернеті або написати новий sketch (code) самому [6].

Ось деякі варіанти використання подібних систем.

Управління освітленням. За допомогою датчика освітлення вмикається світло в кімнаті як відразу, так і з поступовим наростанням яскравості в міру заходу сонця. Включення може також проводитися через wi-fi, з інтеграцією в систему «розумний будинок» або з'єднанням по телефону.

Включення світла на сходах або в довгому коридорі. Дуже красиво виглядає діодний підсвічування кожної сходинки в окремішність. При підключенні до плати датчика руху, його спрацювання викличе послідовне, з затримкою часу включення підсвічування ступенів або коридору, а відключення цього елемента призведе до зворотного процесу.

Світломузика. Подавши на аналогові входи звуковий сигнал через фільтри, на виході вийде цвetoмузикальна установка.

Моддінг комп'ютера. За допомогою відповідних датчиків і програм колір світлодіодів може залежати від температури або завантаження процесора або оперативної пам'яті. Працює такий пристрій по протоколу dmx 512.

Подібні установки збираються на мікросхемах WS 2811, WS 2812 WS 2812B.

2.2 Світлова частина

Світлодіодна стрічка це одна з різновидів світлодіодного світильника, виконаного у вигляді гнучкої пластикової основи з шаром, що клеїть на зворотному боці і розташованими на ній світлодіодами. Вона використовується для декоративного підсвічування інтер'єру, а також як альтернатива основного джерела освітлення. Поставляється в пластикових катушках.

Довжина в катушці - 5 метрів. Стрічка являє собою єдину мідну плату, на якій розташовані резистори і світлодіоди. Ці п'ять метрів складаються з окремих

сегментів по 5 см кожен (рис. 2.2).

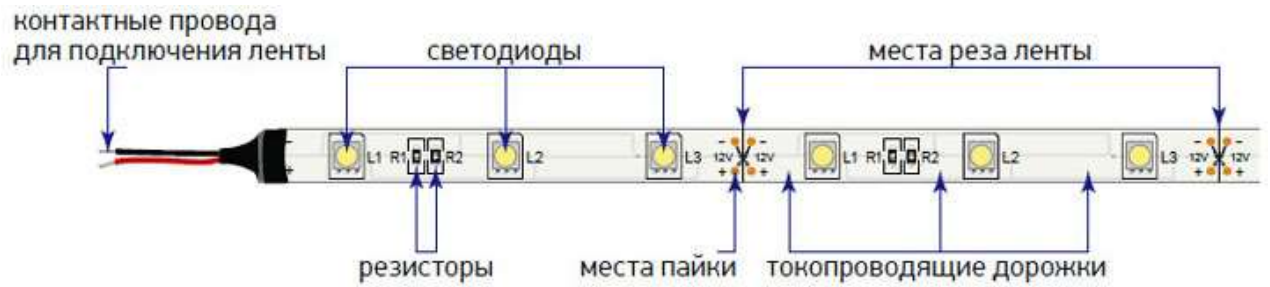


Рисунок 2.2 - Світлодіодна стрічка

Світлодіодні стрічки бувають декількох типів:

- одноколірні або монохромні (рис. 2.3а);
- кольорові або RGB = Red-Green-Blue (червоний-синій-зелений(рис. 2.3б);
- відкриті (рис. 2.3г);
- захищені від вологи (рис. 2.3д).



Рисунок 2.3 – Типи світлодіодних стрічок

Відрізнити одноколірну від RGB можна за кількістю припаяних дротів. На монохромного їх всього два, а на кольоровий чотири. При цьому для підключення одноколірної потрібен тільки блок живлення, а для RGB стрічки ще обов'язковий

контролер. Також вони діляться за розміром світлодіодів і способу їх застосування. Найпоширеніші це:

- SMD3528 (рис. 2.4)
- SMD5050.

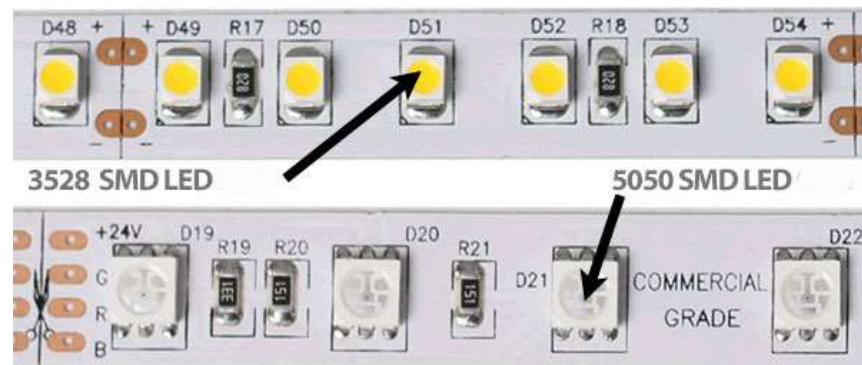


Рисунок 2.4 - SMD3528

SMD від англійського Surface Mounted Device - пристрій поверхневого монтажу. Тобто світлодіод припаяний на підкладку зверху. Цифри після SMD позначають розмір світлодіода в міліметрах - його довжина і ширина.

На відміну від звичайної світлодіодним RGB-стрічки, в якій всі світлодіоди однаково реагують на сигнал з RGB-контролера, в адресному LED-стрічці кожен світлодіод отримує індивідуальну команду управління. В результаті користувач може максимально точно підбирати відтінок для кожного світлодіода, створювати світлові ефекти і збирати матриці з 16 млн кольорів.

Адресна світлодіодна стрічка складається з RGB-світлодіодів в SMD корпусі 5050 та мікросхем ШІМ-драйверів. В даний час найбільш популярними є адресні LED-стрічки з використанням чіпів WS2811 і WS2812B. Модифікація WS2811 є інтегральну мікросхему (IMC) в корпусі DIP-8 (9,2x6,4 мм) або SOP-8 (5,1x4,0 мм). Даний 3-канальний драйвер має наступну конфігурацію висновків: 1 - ШІМ-регульований вихід (червоний); 2 - ШІМ-регульований вихід (зелений); 3 - ШІМ-регульований вихід (синій); 4 - загальний; 5 - вихід передачі даних; 6 - вхід передачі даних; 7 - вибір режиму роботи; 8 - харчування +5 В (рис. 2.5).

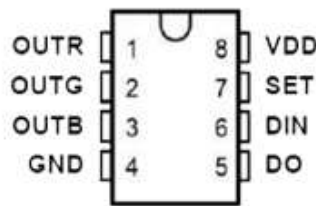


Рисунок 2.5 - Адресна світлодіодна стрічка

В адресній стрічці з використанням чіпа WS2811 і напругою 5 вольт мікросхема драйвера розташовується в безпосередній близькості перед кожним RGB-світодіодом SMD 5050, поруч з яким також встановлено токоограничующие резистори і конденсатор, що захищає від перешкод.

Але на сьогоднішній момент такі моделі застаріли і зустрічаються вкрай рідко. Сьогодні у продажу є адресні світлодіодні стрічки на чіпах WS2811 тільки з живленням від +12 В. В цьому випадку чіп WS2811 керує не одним світлодіодом, а групою з 3 штук. Подібна уніфікація дозволила значно спростити складання адресних світлодіодних стрічок і модулів, а сам WS2812B має лише 4 висновки: 1 - харчування (+ 3,5 ... +5,3 В); 2 - вихід передачі даних; 3 - загальний; 4 - вхід передачі даних. ІМС драйвера споживає не більше 1 мкА, а максимальний струм одного адресного світлодіода становить 60 мА. Діапазон робочих температур: від -25 до + 80 ° С.

Адресна стрічка WS2812B поділена на сегменти, в кожному з яких розташований світлодіод і конденсатор (для підвищення завадостійкості). Щодо напруги живлення всі вони між собою підключені паралельно, тобто +5 В буде присутній на кожному сегменті. А ось передача даних здійснюється послідовно: від попереднього сегмента до чого. Тому при виході з ладу одного з світлодіодів ланцюга все такі сегменти перестануть світитися. Управління готовими пристроями і модулями на базі WS2812B проводиться за допомогою спеціалізованого контролера, всередині якого записана програма. На радіоаматорському рівні управляти роботою адресному світлодіодної стрічки найзручніше через Arduino, використовуючи для цього невелику програму - скетч.

В таблиці 2.1 наведено порівняння технічних характеристик адресних світлодіодних стрічок.

Таблиця 2.1 - Порівняння технічних характеристик адресних світлодіодних стрічок WS2811, US1903 и WS2818

Характеристика	US1903	WS2811	WS2818
ІС	Зовнішня	Зовнішня	Зовнішня
Вольтаж	DC12v	DC12v	DC12v/DC24v
Колір	RGB	RGB	RGB
Кількість світлодіодів на метр	60	60	60
Споживана потужність	14 Вт/м	14 Вт/м	14 Вт/м
Дроти	Позитивний + від'ємний + Один рядок даних	Позитивний + від'ємний + Один рядок даних	Позитивний + від'ємний + Подвійні лінії передачі даних

2.3 Висновки до розділу 2

Для виконання бакалаврської роботи я буду використовувати плату Arduino на платформі Nano. Плата Arduino складається з мікроконтролера Atmel AVR (ATmega328P), а також елементів обв'язки для програмування та інтеграції з іншими схемами. На багатьох платах присутній лінійний стабілізатор напруги +5В або +3,3В. Тактування здійснюється на частоті 16 або 8 МГц 10 кварцовим резонатором (в деяких версіях керамічним резонатором). У мікро-контролер попередньо прошивається завантажувач BootLoader, тому зовнішній програматор не потрібен. Плати Ардуіно здатні управляти світлодіодними конструкціями по заздалегідь заданими програмами.

Розглянуті сучасні світлодіодні стрічки. Основна відмінність адресних світлодіодних стрічок:

- WS2818 може бути підключений як до DC12v, так і до DC24v, в даний час WS2811 / UCS1903 призначений тільки для піксельної підсвічування DC12v;

- IC WS2811 / UCS1903 має тільки одну лінію даних, програмований світло WS2811 має подвійні лінії даних.

Для подальшої розробки обрана адресна світлодіодна стрічка WS2818. Адресна стрічка WS2812B поділена на сегменти, в кожному з яких розташований світлодіод і конденсатор (для підвищення завадостійкості). Щодо напруги живлення всі вони між собою підключені паралельно, тобто +5 В буде присутній на кожному сегменті. А ось передача даних здійснюється послідовно: від попереднього сегмента до чого. Тому при виході з ладу одного з світлодіодів ланцюга все такі сегменти перестануть світитися. Управління готовими пристроями і модулями на базі WS2812B проводиться за допомогою спеціалізованого контролера, всередині якого записана програма. На радіоаматорському рівні управляти роботою адресною світлодіодної стрічки найзручніше через Arduino, використовуючи для цього невелику програму - скетч.

3 ПРОГРАМНА МОДЕЛЬ СИСТЕМИ

3.1 Розробка програми

У візуалізаторі за допомогою алгоритму швидкого перетворення Фур'є виділяються 8 частот (поріг чутливості на кожную частоту свій, знижується від 1 до 8), перетворюються в колір і виводяться на лінійку світлодіодів по одному з восьми алгоритмів.

Розроблена програма для персонального комп'ютера виконує роль 20 полосного аналізатора спектра і пульта управління програмами. Алгоритм програми 20-и бітного аналізатора спектра на базі Arduino наведено на рисунку 3.1.



Рисунок 3.1 – Алгоритм програми генерації звуків

Програма для персонального комп'ютера пропонується у вигляді виконуваного модуля без вихідного коду.

На вході аудіо потік з будь-якого записуючого звукового пристрою персонального комп'ютера. Наприклад: мікшер, лінійний вхід, мікрофон або будь-який віртуальний аудіо кабель. Використовуючи мікшер або віртуальний аудіо кабель можна отримати аудіо потік з будь-якого аудіо плеєра. Наприклад: ITUNES, AIMP, VLC або будь-якого іншого.

Статичні режими роботи програми.

Випадкова двоколірна схема («двотавр»): вибрані два випадкових кольору і тільки вони використовуються для відображення піків сигналу. Згодом будуть обрані нові кольори. Використовують параметр, щоб налаштувати швидкість зміни колірної схеми.

Веселка («радуга»): все піки сигналів відображаються як один і той же колір (з невеликою кількістю випадкових варіацій) і цей колір змінюється як веселка з плином часу.

Кольорові частоти («МИКС»): в цьому режимі кожен пік сигналу забарвлюється в залежності від частотної смуги де він знаходиться. Найнижча смуга червоного кольору, і далі вгору по спектру. Є 8 смуг частот: червоний, оранжевий, жовтий, зелений, блакитний, синій, фіолетовий, білий. Цей колірний режим найбільш цікавий, коли частотна характеристика налаштована на всі смуги частот.

Динамічні програми:

- 1 - theaterChaseRainbow - із прикладів до бібліотеки Adafruit_NeoPixel;
- 2 - rainbowCycle - із прикладів до бібліотеки Adafruit_NeoPixel;
- 3 - мерехтливі вогні;
- 4, 5, 6 - прості варіанти біжать вогнів.

Також на вкладці динамічні ви можете встановити колір світіння всієї стрічки вибором кольору на кольоровому полі. Динамічні програми можуть виконуватися автономно, без підключення до ПК.

Зовнішній вигляд програми наведен на рис. 3.2 – 3.6.

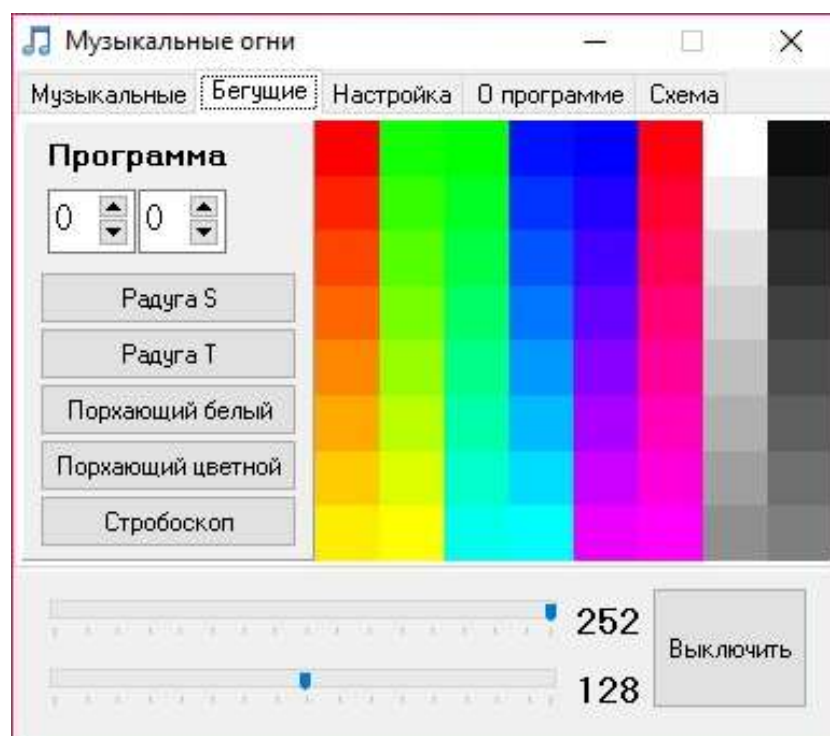


Рисунок 3.2 – Зовнішній вигляд вкладки динамічних режимів

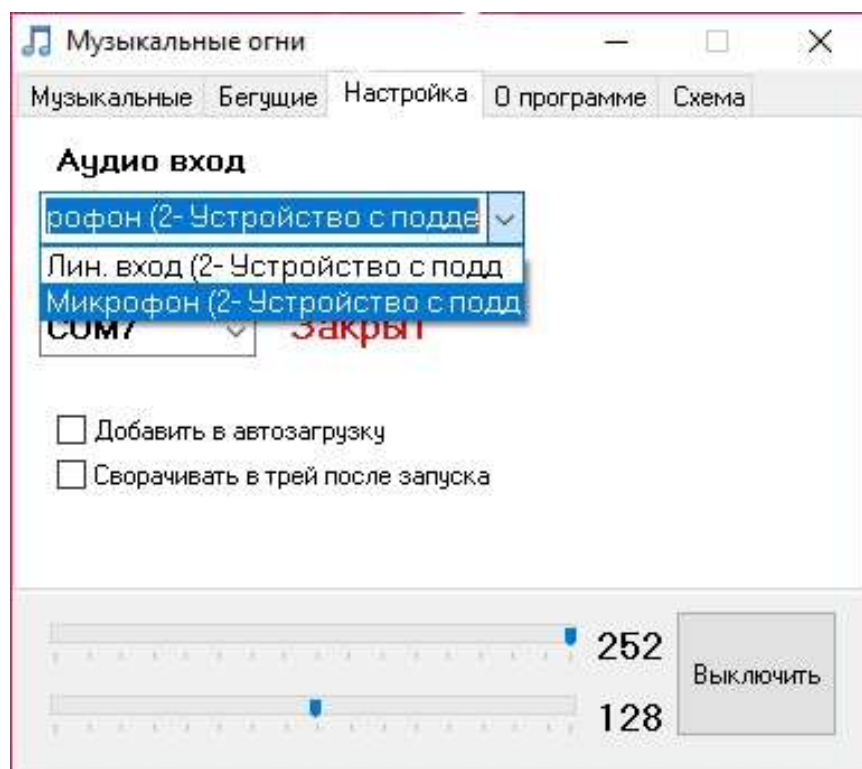


Рисунок 3.3 – Підключення до звукового виходу

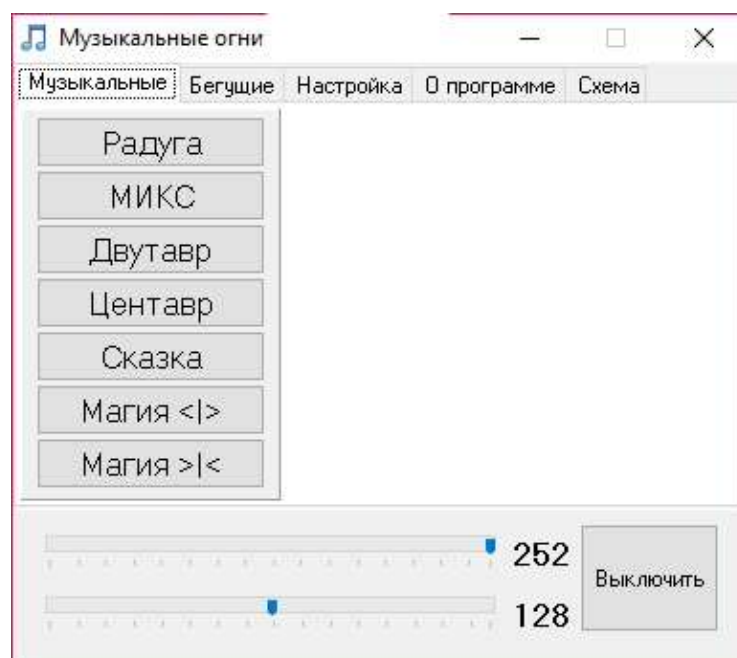


Рисунок 3.4 - Зовнішній вигляд вкладки статичних режимів

Програма розроблялась на мові Delphi. На рисунку 3.5 зображено структуру

проекту програми.

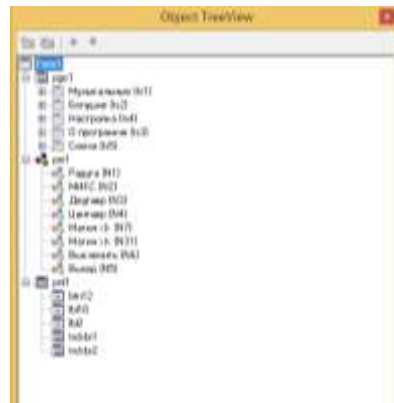


Рисунок 3.5 – Структура проекту програми

3.2 Програмування мікроконтролера

Для програмування буде використовуватись інтегрована середа розробки Arduino IDE.

Для того щоб запрограмувати Arduino Nano, спочатку необхідно її підключити до комп'ютера, за допомогою USB кабелю. Так як вона вже включає в собі програматор, додаткових маніпуляцій для підключення не треба. Після вдалого підключення через диспетчер пристроїв необхідно переглянути, яким портом Arduino ідентифікувалася комп'ютером. Далі в вікні Arduino IDE необхідно перейти в Інструменти-Порт та вибрати порт (див. рис. 3.6), який був вказаний в диспетчері пристроїв.

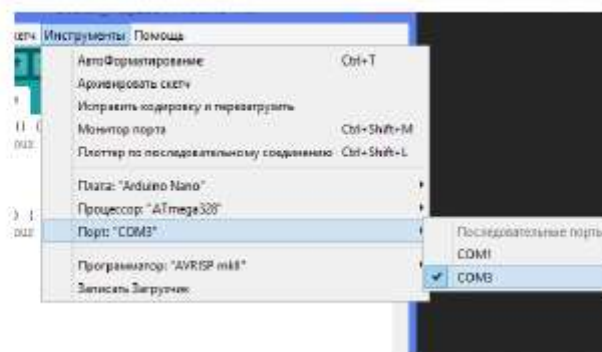


Рисунок 3.6 – Вибір порту

Наступним кроком треба обрати плату та процесор, який на ній встановлено (якщо треба). В нашому випадку це Arduino Nano на процесорі ATmega328 (див. рис. 3.7 – 3.8).

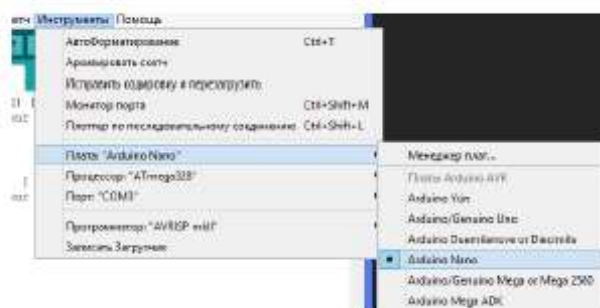


Рисунок 3.7 – Вибір плати

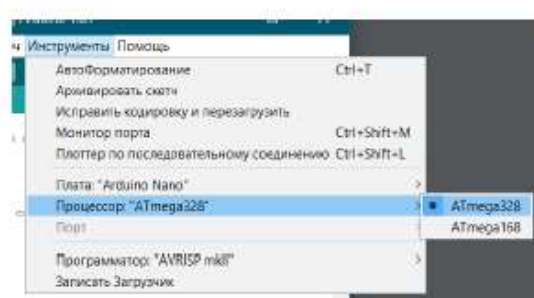


Рисунок 3.8 – Вибір процесору 54

Після написання коду програми у вікні Arduino IDE його необхідно перевірити відповідною кнопкою на панелі і якщо не буде знайдено помилок в

кодi, то можна записувати скетч на плату відповідною кнопкою. Якщо в налаштуванні самого IDE було правильно вказано порт, плату і процесор, то завантажувач повідомить про вдале завантаження коду на плату та кількість зайнятої пам'яті.

Так як кожен елемент WS2812B фактично складається з 3 світлодіодів (синього, червоного, зеленого), то для управління його світінням потрібно 3 байта (по одному на кожен колір). У свою чергу, кожен байт може приймати значення від 0 до 255, в результаті чого можна задати більш 16,5 млн відтінків. Розмір скетчу буде дорівнює кількості світлодіодних сегментів, помноженому на 3. Передача даних відбувається наступним чином: ШІМ-драйвер WS2812B першого сегмента забирає з посилки перші 3 байти, пропускаючи інші дані на вихід DO. Далі слід пауза довжиною до 50 мкс, що означає, що другий за рахунком драйвер повинен вжити таких 3 байта. І так далі. Тривалість паузи більше 50 мкс означає кінець передачі і повторення циклу. Для роботи з адресними стрічками і модулями найпростіше використовувати бібліотеки FastLED і Adafruit NeoPixel. У середині кожної бібліотеки є готові скетчі, на основі яких нескладно навчитися самостійно створювати нові світлові ефекти. Щоб скетч заробив з першого разу, необхідно в заголовку правильно вказати кількість світлодіодів в стрічці (NUM_LEDS) і номер порту для передачі даних (PIN).

Для роботи контролера Arduino потрібно виконати наступні дії.

Встановити в Arduino IDE бібліотеки: Adafruit_Neopixels, IRremote, RF24 (якщо використовується бездротовий варіант). Встановити в скетчі:

- кількість світлодіодів у стрічці:

```
#define stripLeds 120
```

- номери виходів до яких підключена стрічка і ІК приймач ДУ:

```
#define stripPin 2 // вихід управління світлодіодним стрічкою
#define irPin 3 // вхід IR
```

- номери контактів для управління радіо модулем (якщо використовується):

```
RF24 radio (9, 10); // Об'єкт radio для роботи з nRF24, номери
висновків nRF24L01 + (CE, CSN)
```

На даний момент в скетчі реалізовані 7 цвetoмузикальних програм і понад 70 варіантів зміни вогнів (додаток А).

3.3 Висновки до розділу 3

У візуалізаторі за допомогою алгоритму швидкого перетворення Фур'є виділяються 8 частот (порог чутливості на кожен частоту свій, знижується від 1 до 8), перетворюються в колір і виводяться на лінійку світлодіодів по одному з восьми алгоритмів. Розроблена програма для персонального комп'ютера виконує роль 20 полосного аналізатора спектра і пульта управління програмами. Програма для персонального комп'ютера пропонується у вигляді виконуваного модуля без вихідного коду. На вході аудіо потік з будь-якого записуючого звукового пристрою персонального комп'ютера. Наприклад: мікшер, лінійний вхід, мікрофон або будь-який віртуальний аудіо кабель. Використовуючи мікшер або віртуальний аудіо кабель можна отримати аудіо потік з будь-якого аудіо плеєра. Наприклад: ITUNES, AIMP, VLC або будь-якого іншого. Програма розроблялась на мові Delphi.

Для програмування мікроконтролера використовувалась інтегрована середовище розробки Arduino IDE. На даний момент в скетчі реалізовані 7 цвetoмузикальних програм і понад 70 варіантів зміни вогнів.

4 РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Збирання схеми

У кожній адресній стрічці є початок і кінець, які не можна міняти місцями в процесі побудови схеми. Щоб не заплутатися, виробники використовують умовні позначення, наприклад, стрілки, що вказують напрямок сигналу. Підключення адресній світлодіодній стрічці WS2812B до Arduino проводиться

за трьома проводам, як показано на рисунку 4.1. Контакти харчування + 5V і GND з'єднують з відповідними висновками на платі Arduino. Якщо приєднується відрізок налічує понад 13-ти світлодіодів, то необхідно використовувати виносний блок живлення. При цьому загальний провід (GND) Arduino і «мінус» блоку живлення повинні бути з'єднані між собою. Контакт DIN (digital input) призначений для прийому даних від контролера і електрично з'єднується з будь-яким з його цифрових портів. З іншого боку адресному стрічці (і кожного сегмента теж) розміщено 3 контакту: + 5V, DO (digital output) і GND, до яких можна підключити ще кілька відрізків різної довжини.

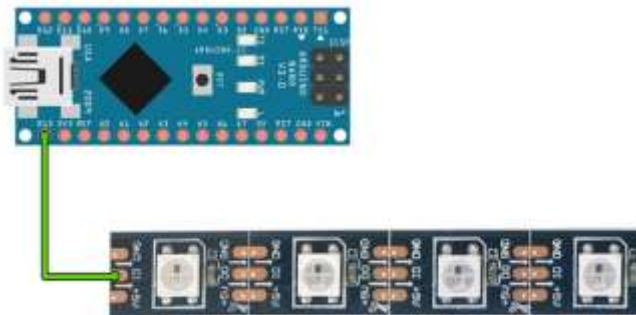


Рисунок 4.1 – Підключення світлодіодної стрічки

Перший варіант, безпосереднє підключення комп'ютерного модуля до USB ПК, скетч COMtoLed (додаток А) (рис.4.2).

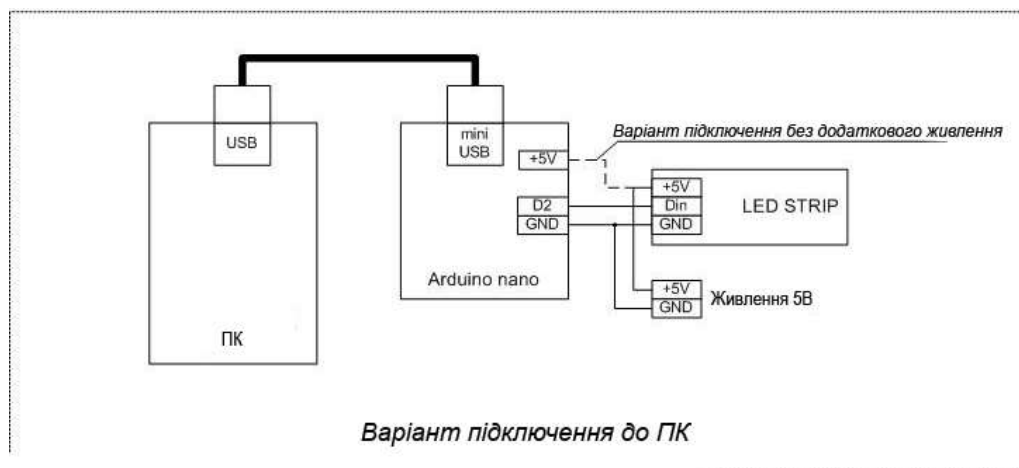


Рисунок 4.2 – Підключення модуля до ПК

Другий варіант, підключення комп'ютерного модуля через радіоміст побудований на nRF24L01, скетчі COMtoRF (додаток А.2) і RFtoLed (рис.4.3).

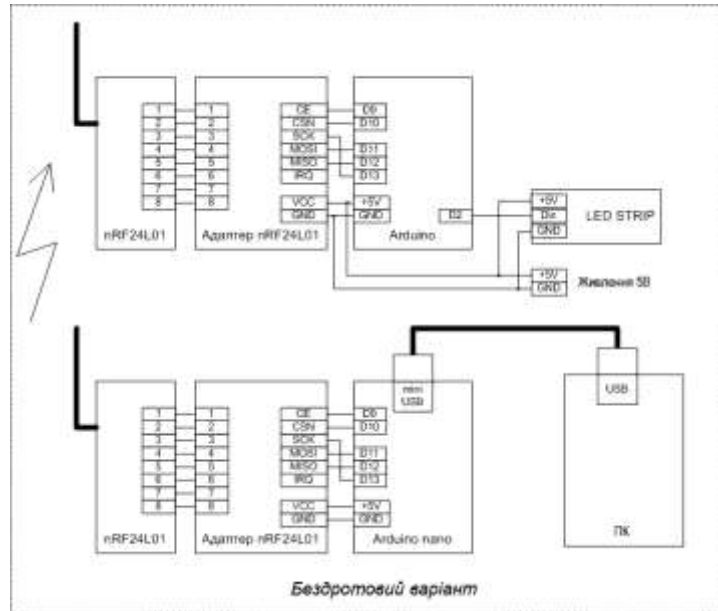


Рисунок 4.3 – Бездротове під'єднання модуля до ПК

Для складання мінімального комплекту, з безпосереднім підключенням до USB ПК, потрібна лише світлодіодна стрічка з WS2812B, arduino nano, три провідника і міні USB кабель для підключення плати Arduino до одного з USB портів ПК. В даному варіанті кількість активних світлодіодів обмежена 20 по одному на кожен колірну смугу програми ЦМУ. Обмеження введено для запобігання перевантаження USB порту.

Кількість світлодіодів при даній схемі включення обмежена потужністю вашого джерела живлення, але не може перевищувати 256. Для зняття обмеження в 256 світлодіодів вам потрібно поміняти в скетчах типи змінних відповідають за кількість світлодіодів з `uint8_t` на `uint16_t`.

Для складання бездротового варіанту, додатково буде потрібно ще одна плата Arduino, два радіо модуля nRF24L01 з платами адаптерів, блок живлення 5V на потужність відповідну вашій стрічці. Обидва модуля nRF24L01 підключаються до плат Arduino ідентично. Підключення здійснюється за типовою схемою до

цифрових висновків D9, D10, D11, D12, D13 або іншим при відповідній зміні заголовків конфігурації в скетчах. Вхід управління світлодіодним стрічкою Din (Din на стрічці) заведений на цифровий вихід D2 плати arduino і може бути змінений при внесенні відповідних змін до скетч.

4.3 Інструкція користувача

Для запуску комп'ютерного модуля потрібно виконати наступний алгоритм.

Запустити виконачу програму, яка не вимагає спеціальної установки. Створіть папку з будь-яким ім'ям яке у вас асоціюється з кольоромузичне програмою і скопіюйте в неї файл "PROG.EXE". Програма готова до запуску.

Встановіть середу Arduino IDE. Скачайте і встановіть бібліотеки Adafruit_Neopixel, RF24 і IRremote.

Внесіть відповідні зміни в скетчіз парметрами стрічки та підключення.

Встановіть кількість світлодіодів:

```
#define stripLed 120 // кількість світлодіодів в стрічці
```

Встановіть номер виведення до якого підключена світлодіодна стрічка:

```
#define stripPin 2 // вихід управління світлодіодним стрічкою
```

Встановіть номери виводів до яких підключений модуль nRF24L01:

```
RF24 radio (9, 10); // створюємо об'єкт radio для роботи з бібліотекою RF24, вказуючи номери висновків nRF24L01 + (CE, CSN)
```

Збираємо проект і заливаємо в плату або плати в залежності від варіанту. Підключаємо стрічку, запускаємо програму.

Налаштування програми:

- встановіть послідовний COM порт до якого підключений модуль Arduino;
- виберіть аудіо пристрій з виходу якого аудіо потік буде подаватися на вхід кольоромузичне програми;
- виберіть одну з програм кольоромузичне або динамічної візуалізації.

4.4 Результати перевірки системи

На рисунках 4.4 – 4.7 наведені фотографії роботи комп'ютерного модуль для дитячого центру.



Рисунок 4.4 – З'єднання мікроконтролера та світлодіодної стрічки



Рисунок 4.5 – Статичний режим «Веселка»

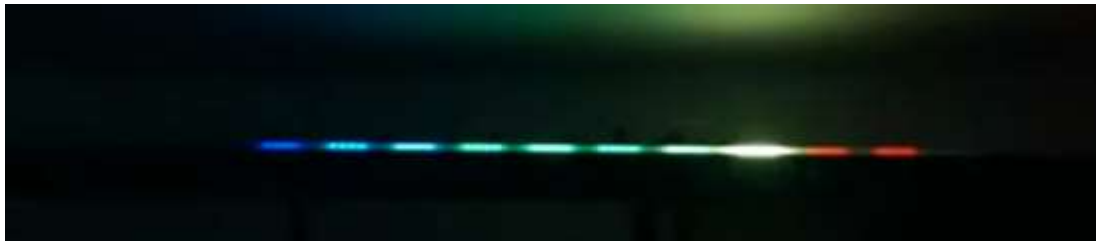


Рисунок 4.6 – Динамічний режим «Веселка»



Рисунок 4.7 – Статичний режим «МІКС»

4.4 Висновки до розділу 4

Для складання мінімального комплексу, з безпосереднім підключенням до USB ПК, потрібна лише світлодіодна стрічка з WS2812B, arduino nano, три провідника і міні USB кабель для підключення плати Arduino до одного з USB портів ПК. В даному варіанті кількість активних світлодіодів обмежена 20 по одному на кожен кольоровий смугу програми ЦМУ. Обмеження введено для запобігання перевантаження USB порту.

Основні принципи роботи комп'ютерного модуля:

- смугова фільтрація звуку і управління вибором світлого режиму виконується програмою "PROG.EXE" для ПК;
- формування світлого режиму виконується модулями Arduino;
- управління світлодіодною стрічкою здійснює модуль Arduino відповідно до обраної програми;
- для відтворення музики може використовуватися будь-який програмний

плеєр, наприклад WinAmp, AIMP, ITUNES, Windows Media і т.п. або апаратний плеєр має лінійний вихід.

ВИСНОВКИ

В ході бакалаврської роботи розглянуто поняття звукового та світлового комп'ютерного модуля для ПК та покроково розглянуто приклад створення такої системи на базі платформи Arduino.

Було проаналізовано кілька типів плат Arduino та зроблено висновок, що найкраще підходить саме Arduino Nano, через її невеликі розміри, та наявність програматору на її борту. Розглянуто основи роботи в фірмовому середовищі програмування Arduino IDE, створено скетчі (програми для завантаження на Arduino), та розглянуто їх головні частини.

При реалізації системи світлової частини модуля в якості самого об'єкту світла було використано адресну світлодіодну стрічку WS2812B. Звукова частина модуля реалізована як комп'ютерна програма, яка виконує роль 20-полосного аналізатора спектра і пульта управління програмами.

Основні принципи роботи комп'ютерного модуля:

- смугова фільтрація звуку і управління вибором цвetoмузикальній програм виконується програмою "PROG.EXE" для ПК.
- формування цвetoмузикальній програм виконується модулями Arduino.
- управління світлодіодною стрічкою здійснює модуль Arduino відповідно до обраної програми.
- для відтворення музики може використовуватися будь-який програмний плеєр, наприклад WinAmp, AIMP, ITUNES, Windows Media і т.п. або апаратний плеєр має лінійний вихід.

В результаті розробки комп'ютерного модуля були вирішені наступні завдання:

- проведено аналіз апаратного забезпечення світловою та звуковою частин комп'ютерного модуля;
- розроблена схеми комп'ютерного модуля;
- опис алгоритмів функціонування звуковою частини модуля;

- проведено програмування мікроконтролера Arduino;
- проведено тестування комп'ютерного модуля.

В результаті виконаних робіт можна зробити висновок, що дана система на основі Arduino повністю задовольняє вимогам технічного завдання, має хороші технічні показники, має декілька режимів світлового відображення, проста у виготовленні та експлуатації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Офіційний сайт платформи Arduino [Электронный ресурс]. – Режим доступа: <http://arduino.cc>
2. Ключев А.О. Аппаратные и программные средства встраиваемых систем / Д.Р. Ковязина, П.В. Кустарев, А.Е. Платунов. - СПб, 2010, 290 с.
3. Bianzi M. Getting Started with Arduino, 2008, -128 p.
4. Блум Дж. Вивчаємо Arduino [пер. з англ. Петін В.] - СПб.: БХВ-Петербург - 2015. 336 с.
5. Голіцина О.Л. Інформаційні технології: Підручник / О.Л. Голіцина, Н.В. Максимов, Т.Л. Партика, І.І. Попов. - М .: Форум, ИНФРА-М, 2013. - 608с.
6. Еванс Б. Arduino. Блокнот програміста. [пер. з англ. Голобов В.] - - СПб.: БХВ-Петербург 2007. 40с.
7. Gerz E. di Justo P. Atmospheric Monitoring with Arduino - O'Reilly Media, Inc. 2013 89с.
8. Монк С. Програмируем Arduino. Профессиональная работа со скетчами. — СПб.: Питер, 2017. — 272 с.
9. Петин В.А., Биняковский А.А. Практическая энциклопедия Arduino / В.А. Петин, А.А. Биняковский. — М.: ДМК Пресс, 2017. — 152 с.
10. Голиков Д.В. Scratch и Arduino. 18 игровых проектов для юных программистов микроконтроллеров. - СПб.: BHV, 2018. - 160 с.
11. Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino. - СПб.: BHV, 2016. - 256 с.
12. Уилмсхерст Т. Разработка встроенных систем с помощью PIC-микроконтроллеров. - СПб.: Корона-Век, 2015. - 544 с.

ДОДАТОК А

Скетч комп'ютерного модуля

Лістинг А.1 – Керування світлодіодною стрічкою

```

#include <nRF24L01.h>                                // Підключаем файл настроек
из библиотеки RF24
#include <SPI.h>                                     // Підключаем библиотеку для
работы с шиной SPI
#include <RF24.h>                                    // Підключаем библиотеку для
работы с nRF24L01+
#include <Adafruit_NeoPixel.h>

#define stripPin 2                                  // выход управления светодиодной лентой

#define stripLed 120                                // количество светодиодов в ленте
#define bandPass 20                                 // полос (групп светодиодов)
#define LedtoColor stripLed/bandPass
#define fallspeed 15

// Parameter 1 = number of pixels in strip
// Parameter 2 = Arduino pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products
w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels,
WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most
NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA
pixels, not v2)
Adafruit_NeoPixel strip = Adafruit_NeoPixel(stripLed, stripPin,
NEO_GRB + NEO_KHZ800);

RF24      radio(9, 10);                            // Создаём объект radio для работы
с библиотекой RF24, указывая номера выводов nRF24L01+ (CE, CSN)

typedef union{
    struct {
        uint8_t b,g,r,w;
    };
    uint32_t dw;
} TColor;

typedef union{

```

```

    struct {
        uint8_t b0,b1;
    };
    uint16_t w;
} TWord;

typedef union{
    struct { uint8_t b0,b1,b2,b3; };
    uint32_t dw;
} TDWord;

const uint32_t PROGMEM
    colorTab[]={

0xFF0000,0xFF1100,0xFF2200,0xFF3300,0xFF4400,0xFF5500,0xFF6600,0xFF
7700,0xFF8800,0xFF9900,0xFFAA00,0xFFBB00,0xFFCC00,0xFFDD00,0xFFEE00
,0xFFFF00,    //красный - жёлтый

0xFFFF00,0xEEFF00,0xDDFF00,0xCCFF00,0xBBFF00,0xA AFF00,0x99FF00,0x88
FF00,0x77FF00,0x66FF00,0x55FF00,0x44FF00,0x33FF00,0x22FF00,0x11FF00
,0x00FF00,    //жёлтый - зелёный

0x00FF00,0x00FF11,0x00FF22,0x00FF33,0x00FF44,0x00FF55,0x00FF66,0x00
FF77,0x00FF88,0x00FF99,0x00FFAA,0x00FFBB,0x00FFCC,0x00FFDD,0x00FFEE
,0x00FFFF,    //зелёный - циан (голубой)

0x00FFFF,0x00EEFF,0x00DDFF,0x00CCFF,0x00BBFF,0x00A AFF,0x0099FF,0x00
88FF,0x0077FF,0x0066FF,0x0055FF,0x0044FF,0x0033FF,0x0022FF,0x0011FF
,0x0000FF,    //голубой - синий

0x0000FF,0x1100FF,0x2200FF,0x3300FF,0x4400FF,0x5500FF,0x6600FF,0x77
00FF,0x8800FF,0x9900FF,0xAA00FF,0xBB00FF,0xCC00FF,0xDD00FF,0xEE00FF
,0xFF00FF,    //синий - пурпур (маджента)

0xFF00FF,0xFF00EE,0xFF00DD,0xFF00CC,0xFF00BB,0xFF00AA,0xFF0099,0xFF
0088,0xFF0077,0xFF0066,0xFF0055,0xFF0044,0xFF0033,0xFF0022,0xFF0011
,0xFF0000,    //маджента - красный

0xFFFFFFFF,0xEFEFEF,0xDFDFDF,0xCF CF CF,0xBF BF BF,0xAFAFAF,0x9F9F9F,0x8F
8F8F,0x7F7F7F,0x6F6F6F,0x5F5F5F,0x4F4F4F,0x3F3F3F,0x2F2F2F,0x1F1F1F
,0x0F0F0F,0X000000}; // белый - чёрный

    const uint8_t PROGMEM rotateInterval[] =
{224,224,224,224,224,224,224,224,224,224,224,224,224,224,224,224,22
4,224,224,224,224,224,224,224,224,224,224,224,224,224,224};

    const TDWord PROGMEM progColor[] = {
        { 0,16,64,80},    // Цвета первой субпрограммы 1,2,3,4 цвета
соответственно (для всех программ)
        {16,32,64,48},    // Цвета второй субпрограммы

```

```
{32,64,16,48},  
{48,0,80,16},  
{64,16,32,80},  
{80,16,0,48},  
{96,64,48,16},  
{96,0,32,64} // Цвета восьмой субпрограммы  
};  
  
/*  
0 - красный  
16 - желтый  
32 - зеленый  
48 - аква  
64 - синий  
80 - фиолетовый  
96 - белый  
*/  
  
uint8_t paramTabl[32] =  
{20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,  
20,20,20,20,20,20,20,20,20,20};  
TColor clN[4] =  
{{0,0,255,0},{0,255,0,0},{0,0,255,0},{0,255,255,0}};  
//0xFF0000,0x00FF00,0x0000FF,0xFFFF00  
uint8_t magn[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};  
uint8_t threshold[20] =  
{192,192,192,192,192,160,160,160,128,128,128,112,112,112,96,96,96,80,80,80};  
uint8_t stepN[4] = {0,0,0,0};  
uint8_t waitN[4] = {3,3,4,4};  
uint8_t waitNc[4] = {3,3,4,4};  
uint8_t readData[24]; // Буфер команды  
uint8_t prog = 29;  
uint8_t subprog = 4;  
uint8_t subcolor = 0;  
uint8_t progStep = 0;  
uint8_t progSubStep = 0;  
uint8_t progDir = 0;  
uint8_t param = 10;  
uint8_t rotate = 1;  
uint8_t rotateStep = 0;  
uint8_t brightness = 255;  
uint32_t clBlack;  
uint32_t clWhite;  
uint32_t rndNumber1 = 0;  
uint32_t rndNumber2 = 0;  
uint8_t nPhase = 0;  
uint8_t bpData[20];  
  
void setup(){  
    // initialize serial:
```

```

Serial.begin(115200);
strip.begin();
strip.show(); // Initialize all pixels to 'off'

clWhite=strip.Color(255, 255, 255);
clBlack=strip.Color(0, 0, 0);

    radio.begin(); // Иницилируем
работу nRF24L01+
    //    radio.setAutoAck(false);

    radio.setChannel(5); // Указываем
канал приёма данных (от 0 до 127), 5 - значит приём данных
осуществляется на частоте 2,405 ГГц (на одном канале может быть
только 1 приёмник и до 6 передатчиков)
    radio.setDataRate      (RF24_1MBPS); // Указываем
скорость передачи данных (RF24_250KBPS, RF24_1MBPS, RF24_2MBPS),
RF24_1MBPS - 1Мбит/сек
    radio.setPALevel      (RF24_PA_HIGH); // Указываем
мощность передатчика (RF24_PA_MIN=-18dBm, RF24_PA_LOW=-12dBm,
RF24_PA_HIGH=-6dBm, RF24_PA_MAX=0dBm)
    radio.openReadingPipe (1, 0x1234567890LL); // Открываем 1
трубу с идентификатором 0x1234567890 для приема данных (на ожном
канале может быть открыто до 6 разных труб, которые должны
отличаться только последним байтом идентификатора)

    radio.startListening (); // Включаем
приемник, начинаем прослушивать открытые трубы
}

TColor clOne(uint8_t spr) {
    TColor cl;

    cl.dw=pgm_read_dword(&colorTab[spr * 16]);
    return cl;
}

void sub12() {

    sub12_1(clN[0]);
    strip.show();
    if (++progStep>38) {
        progStep=0;
        progDir=!progDir;
        if ((subprog==7)and(progDir!=0)) {

```

```

        clN[0]=clOne(subcolor);
        if (++subcolor>6) subcolor=0;
    }
}

void sub12_1(TColor color) {
    TColor cl;
    TWord akk;
    uint8_t d;
    int16_t r;

    for(uint8_t i=stripLed-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));
    if (progDir==0) d=progStep*6;
        else d=(38-progStep)*6;
    r=color.r-d;
    if (r<0) r=0;
    akk.w=r*r;
    cl.r=akk.b1;
    r=color.g-d;
    if (r<0) r=0;
    akk.w=r*r;
    cl.g=akk.b1;
    r=color.b-d;
    if (r<0) r=0;
    akk.w=r*r;
    cl.b=akk.b1;
    cl=mulBrightness(cl);
    strip.setPixelColor(0, cl.dw);
}

void sub11() {
    TColor clF,clD;
    uint8_t i;

    clF=mulBrightness(clN[0]);
    clD=mulBrightness(clN[1]);
    for(i=0; i<progStep*4; i++) strip.setPixelColor(i,clD.dw);
    for(i=stripLed-progStep*4; i<stripLed; i++)
strip.setPixelColor(i,clD.dw);
    for(i=progStep*4; i<stripLed-progStep*4; i++)
strip.setPixelColor(i,clF.dw);

    for(i=progSubStep; i<progSubStep+4; i++)
strip.setPixelColor(i,clD.dw);
    for(i=stripLed-progSubStep-4; i<stripLed-progSubStep; i++)
strip.setPixelColor(i,clD.dw);
}

```

```

strip.show();
progSubStep--;
if (progSubStep==progStep*4) {
    progStep++;
    if (progStep>stripLed/8-1) {
        progStep=0;
        clF=clN[0];
        clN[0]=clN[1];
        if (subprog==7) {
            clN[1]=clOne(subcolor++);
            if (subcolor>6) subcolor=0;
        } else clN[1]=clF;
    }
    progSubStep=stripLed-1;
}
}

void sub10() {
    TColor clF,clD;
    uint8_t i;

    clF=mulBrightness(clN[0]);
    clD=mulBrightness(clN[1]);
    for(i=0; i<progStep*4; i++) strip.setPixelColor(i,clD.dw);
    for(i=progStep*4; i<stripLed; i++)
strip.setPixelColor(i,clF.dw);
    for(i=progSubStep; i<progSubStep+4; i++)
strip.setPixelColor(i,clD.dw);
    strip.show();
    progSubStep--;
    if (progSubStep==progStep*4) {
        progStep++;
        if (progStep>stripLed/4-1) {
            progStep=0;
            clF=clN[0];
            clN[0]=clN[1];
            if (subprog==7) {
                clN[1]=clOne(subcolor++);
                if (subcolor>6) subcolor=0;
            } else clN[1]=clF;
        }
        progSubStep=stripLed-1;
    }
}

void sub9() {
    TColor cl;

    for(uint8_t i=0; i<4; i++) {
        waitN[i]--;
        if (waitN[i]==0){

```

```

        waitN[i]=waitNc[i];
        stepN[i]++;
        if (stepN[i]>stripLed-1) stepN[i]=0;
        cl=mulBrightness(clN[i]);
        if (i%2==0) strip.setPixelColor(stepN[i], cl.dw);
            else strip.setPixelColor(stripLed-1-stepN[i],
cl.dw);
    }
}
strip.show();
progStep++;
}

void sub8() {
    TColor cl;

    for(uint8_t i=stripLed-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));
    if (progStep>15) cl=mulBrightness(clN[0]);
        else cl=mulBrightness(clN[1]);
    strip.setPixelColor(0 ,cl.dw);
    strip.show();
    if (++progStep>19) progStep=0;
}

void sub7() {
    TColor cl;

    for(uint8_t i=stripLed-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));
    if (progStep>17) cl=mulBrightness(clN[0]);
        else cl=mulBrightness(clN[1]);
    strip.setPixelColor(0 ,cl.dw);
    strip.show();
    if (++progStep>19) progStep=0;
}

void sub6() {
    TColor cl;

    if (progDir==0) cl=mulBrightness(clN[0]);
        else cl=mulBrightness(clN[1]);
    strip.setPixelColor(progStep ,cl.dw);
    strip.show();
    if (++progStep>stripLed-1) {
        progStep=0;
        progDir=!progDir;
    }
}

```

```

void sub5() {
    TColor cl;
    uint8_t i;

    for(i=stripLeds-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));
    if (progDir==0) cl=multsBrightness(clN[0]);
        else cl=multsBrightness(clN[1]);
    strip.setPixelColor(0 ,cl.dw);
    strip.show();
    if (++progStep>9) {
        progStep=0;
        progDir=!progDir;
    }
}

void sub4() {

    sub4_1(clN[0]);
    strip.show();
    if (++progStep>19) {
        progStep=0;
        if (subprog==7) {
            clN[0]=clOne(subcolor++);
            if (subcolor>6) subcolor=0;
        }
    }
}

void sub4_1(TColor color) {
    TColor cl;
    uint8_t i;

    for(i=stripLeds-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));
    if (progStep<10) cl.dw = strip.Color(color.r >> progStep,
color.g >> progStep, color.b >> progStep);
        else cl.dw = strip.Color(color.r >> 19-progStep,
color.g >> 19-progStep, color.b >> 19-progStep);
    cl=multsBrightness(cl);
    strip.setPixelColor(0, cl.dw);
}

void sub3() {

```

```

    sub3_1(clN[0]);
    strip.show();
    if (++progStep>23) {
        progStep=0;
        if (subprog==7) {
            clN[0]=clOne(subcolor++);
            if (subcolor>6) subcolor=0;
        }
    }
}

void sub3_1(TColor color) {
    TColor cl;
    TWord akk;
    uint8_t i;

    for(i=stripLed-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));
    if (progStep<10) {
        if (progStep<2) {
            if (progStep==0) cl.dw = strip.Color(color.r >> 2, color.g
>> 2, color.b >> 2);
            else cl.dw = strip.Color(color.r, color.g,
color.b);
        } else cl.dw = strip.Color(color.r >> (progStep-2), color.g
>> (progStep-2), color.b >> (progStep-2));
        } else cl.dw = clBlack;
    cl=mulBrightness(cl);
    strip.setPixelColor(0, cl.dw);
}

void sub2() {

    sub2_1(clN[0]);
    strip.show();
    if (++progStep>7) {
        progStep=0;
        if (subprog==7) {
            clN[0]=clOne(subcolor++);
            if (subcolor>6) subcolor=0;
        }
    }
}

void sub2_1(TColor cl) {
    uint8_t i;

```

```

        for(i=stripLed-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));
        if (progStep<4) {
            cl=mulBrightness(cl);
        } else cl.dw=clBlack;
        strip.setPixelColor(0, cl.dw);
    }

void sub1() {

    sub1_1(clN[0]);
    strip.show();
    if (++progStep>16) {
        progStep=0;
        if (subprog==7) {
            clN[0]=clOne(subcolor++);
            if (subcolor>6) subcolor=0;
        }
    }
}

void sub1_1(TColor cl) {
    TColor clC;
    uint8_t i;

    for(i=stripLed-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));
    if (progStep<8) {
        if (progStep<2) clC.dw=clWhite;
        else clC=cl;
    }
    else {
        if (progStep<10) clC.dw=clWhite;
        else clC.dw=clBlack;
    }
    cl=mulBrightness(clC);
    strip.setPixelColor(0, cl.dw);
}

void sub00() {
    uint8_t i;
    TColor cl;
    TWord akk;

    for(i=stripLed-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));
    if (progStep==0) {
        cl.dw = pgm_read_dword(&colorTab[random(96)]);
    }
}

```

```

        cl=mulsBrightness(cl);
    } else cl.dw = clBlack;
    strip.setPixelColor(0, cl.dw);
    strip.show();
    if (++progStep>subprog) progStep=0;
}

void sub01() {
    uint8_t i;
    TColor cl;

    for(i=stripLed/2-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));
    for(i=stripLed/2; i<stripLed; i++)
strip.setPixelColor(i,strip.getPixelColor(i+1));
    if (progStep==0) {
        cl.dw = pgm_read_dword(&colorTab[random(96)]);
        cl=mulsBrightness(cl);
    } else cl.dw = clBlack;
    strip.setPixelColor(0, cl.dw);
    strip.setPixelColor(stripLed-1, cl.dw);
    strip.show();
    if (++progStep>subprog-3) progStep=0;
}

void sub06() {
    uint8_t i;
    TColor cl;

    for(i=stripLed-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));
    if (progStep==0) {
        if (++subcolor>6) subcolor=0;
        // cl.dw=clOne(subcolor);
        cl=mulsBrightness(clOne(subcolor));
        strip.setPixelColor(0, cl.dw);
    }
    strip.show();
    if (++progStep>8) progStep=0;
}

void sub07() {
    uint8_t i;
    TColor cl;

    for(i=stripLed/2-1; i>0; i--)
strip.setPixelColor(i,strip.getPixelColor(i-1));

```

```

        for(i=stripLed/2; i<stripLed-1; i++)
strip.setPixelColor(i,strip.getPixelColor(i+1));
        if (progStep==0) {
            if (++subcolor>6) subcolor=0;
            //    cl.dw=clOne(subcolor);
            cl=mulBrightness(clOne(subcolor));
            strip.setPixelColor(0, cl.dw);
            strip.setPixelColor(stripLed-1, cl.dw);
        }
        strip.show();
        if (++progStep>8) progStep=0;
    }

void zmul() {
    TColor cl;
    TWord akk;
    uint8_t i,k,n,gain;

    for(i=0; i<bandPass; i++) {
        akk.w=readData[i]*brightness;
        gain=akk.b1;
        if (magn[i]<fallspeed) magn[i] = 0;
            else magn[i] -= fallspeed;
        if (magn[i]<gain) magn[i]=gain;
            else gain=magn[i];
        cl.dw = pgm_read_dword(&colorTab[96*i/bandPass]);
        cl=mulGain(cl,gain);
        cl.dw = strip.Color(cl.r, cl.g, cl.b);
        n=i*LedtoColor;
        for(k=0; k<LedtoColor; k++) strip.setPixelColor(n+k,
cl.dw);
    }
    strip.show();
}

void zmu2() {
    TColor cl;
    TWord akk;
    uint8_t i,k,gain;

    for(i=0; i<bandPass; i++) {
        akk.w=readData[i]*brightness;
        gain=akk.b1;
        if (magn[i]<fallspeed) magn[i] = 0;
            else magn[i] -= fallspeed;
        if (magn[i]<gain) magn[i]=gain;
            else gain=magn[i];
        cl.dw = pgm_read_dword(&colorTab[96*i/bandPass]);
        cl=mulGain(cl,gain);

```

```

        cl.dw = strip.Color(cl.r, cl.g, cl.b);
        for(k=0; k<LedtoColor; k++)
strip.setPixelColor(i+k*bandPass, cl.dw);
    }
    strip.show();
}

void zmu3() {
    TColor cl;
    TWord akk;
    uint8_t n,i,k,gain;

    for(i=0; i<bandPass; i++) {
        if (magn[i]<fallspeed) magn[i] = 0;
                                else magn[i] -= fallspeed;
        akk.w=readData[i]*brightness;
        gain=akk.b1;
        if (magn[i]<gain) magn[i]=gain;
                                else gain=magn[i];
        cl.dw = pgm_read_dword(&colorTab[96*i/bandPass]);
        cl=mulGain(cl,gain);
        cl.dw = strip.Color(cl.r, cl.g, cl.b);
        n=i*LedtoColor/2;
        for(k=0; k<LedtoColor/2; k++) {
            strip.setPixelColor(n+k, cl.dw);
            strip.setPixelColor(stripLed-(n+k)-1, cl.dw);
        }
    }
    strip.show();
}

void zmu4() {
    TColor cl;
    TWord akk;
    uint8_t n,i,k,gain;

    for(i=0; i<bandPass; i++) {
        if (magn[i]<fallspeed) magn[i] = 0;
                                else magn[i] -= fallspeed;
        akk.w=readData[i]*brightness;
        gain=akk.b1;
        if (magn[i]<gain) magn[i]=gain;
                                else gain=magn[i];
        cl.dw = pgm_read_dword(&colorTab[96*i/bandPass]);
        cl=mulGain(cl,gain);
        cl.dw = strip.Color(cl.r, cl.g, cl.b);
        n=i*LedtoColor/2;
        for(k=0; k<LedtoColor/2; k++) {
            strip.setPixelColor(stripLed/2+n+k, cl.dw);

```

```

        strip.setPixelColor(stripLed/2-(n+k)-1, cl.dw);
    }
}
strip.show();
}

void fromCenterShift() {
    uint8_t i;
    uint32_t c;

    for(i=0; i<stripLed/2; i++) { c=strip.getPixelColor(i+1);
strip.setPixelColor(i,c); }
    for(i=stripLed-1; i>stripLed/2; i--) {
c=strip.getPixelColor(i-1); strip.setPixelColor(i,c); }
}

void toCenterShift() {
    uint8_t i;
    uint32_t c;

    for(i=stripLed/2; i<stripLed-1; i++) {
c=strip.getPixelColor(i+1); strip.setPixelColor(i,c); }
    for(i=stripLed/2-1; i>0; i--) { c=strip.getPixelColor(i-1);
strip.setPixelColor(i,c); }
}

void zmu5() {
    TColor cl;
    uint32_t clW;
    uint8_t i,k,j;

    fromCenterShift();
    k=0;
    j=readData[0];
    for(i=1; i<bandPass; i++) {
        if (j<readData[i]) {
            j=readData[i];
            k=i;
        }
    }
    if (readData[k]<12) clW = strip.Color(32, 32, 32);
    else {
        cl.dw = pgm_read_dword(&colorTab[96*k/bandPass]);
        cl=mulBrightness(cl);
        clW = strip.Color(cl.r, cl.g, cl.b);
    }
    strip.setPixelColor(stripLed/2, clW);
    strip.setPixelColor(stripLed/2-1, clW);
    strip.show();
}

```

```

}

void zmu6() {
    TColor cl;
    uint32_t clW;
    uint8_t i,k,j;

    toCenterShift();
    k=0;
    j=readData[0];
    for(i=1; i<bandPass; i++) {
        if (j<readData[i]) {
            j=readData[i];
            k=i;
        }
    }
    if (readData[k]<12) clW = strip.Color(32, 32, 32);
    else {
        cl.dw = pgm_read_dword(&colorTab[96*k/bandPass]);
        cl=mulBrightness(cl);
        clW = strip.Color(cl.r, cl.g, cl.b);
    }
    strip.setPixelColor(0, clW);
    strip.setPixelColor(stripLed-1, clW);
    strip.show();
}

void zmu7() {
    TColor cl;
    uint32_t clW;
    uint8_t i,k,m;

    fromCenterShift();
    m=0;
    for(i=0; i<bandPass; i++) {
        if (readData[i]-magn[i]>0) {
            if (m<readData[i]) {
                m=readData[i];
                k=i;
            }
        }
        magn[i]=readData[i];
    }
    if (m<6) clW = strip.Color(32, 32, 32);
    else {
        cl.dw = pgm_read_dword(&colorTab[96*k/bandPass]);
        cl=mulBrightness(cl);
        clW = strip.Color(cl.r, cl.g, cl.b);
    }
}

```

```

    strip.setPixelColor(stripLed/2, clW);
    strip.setPixelColor(stripLed/2-1, clW);
    strip.show();
}

void zmu8() {
    TColor cl;
    uint32_t clW;
    uint8_t i,k,m;

    toCenterShift();
    m=0;
    for(i=0; i<bandPass; i++) {
        if (readData[i]-magn[i]>0) {
            if (m<readData[i]) {
                m=readData[i];
                k=i;
            }
        }
        magn[i]=readData[i];
    }
    if (readData[k]<6) clW = strip.Color(32, 32, 32);
    else {
        cl.dw = pgm_read_dword(&colorTab[96*k/bandPass]);
        cl=mulBrightness(cl);
        clW = strip.Color(cl.r, cl.g, cl.b);
    }
    strip.setPixelColor(0, clW);
    strip.setPixelColor(stripLed-1, clW);
    strip.show();
}

void zmu9() {
    TColor cl;
    uint8_t i,k,j;
    uint16_t summ = 0;
    uint16_t n;

    for(i=0; i<bandPass; i++) {
        if (magn[i]<fallspeed) magn[i] = 0;
        else magn[i] -= fallspeed;
        if (magn[i]<readData[i]) magn[i]=readData[i];
        summ += magn[i];
    }
    if (summ>16) {
        k=0;
        for(i=0; i<bandPass; i++) {
            cl.dw = pgm_read_dword(&colorTab[96*i/bandPass]);
            cl.dw = strip.Color(cl.r, cl.g, cl.b);

```

```

        cl=mulBrightness(cl);
        n=stripLed*magn[i]/summ/2;
        for(j=0; j<n; j++) {
            strip.setPixelColor(stripLed/2+k, cl.dw);
            strip.setPixelColor(stripLed/2-k-1, cl.dw);
            k++;
        }
    }
    if (k<stripLed/2) {
        for(j=k; j<stripLed/2; j++) {
            strip.setPixelColor(stripLed/2+k, clBlack);
            strip.setPixelColor(stripLed/2-k-1, clBlack);
            k++;
        }
    }
}
strip.show();
}

void zmul0() {
    TColor cl;
    TWord akk;
    uint8_t i,k,j;
    uint8_t m=0;
    uint16_t n;

    for(i=0; i<bandPass; i++) {
        if (magn[i]<fallspeed*2) magn[i] = 0;
        else magn[i] -= fallspeed;
        if (magn[i]<readData[i]) magn[i]=readData[i];
        if (m<magn[i]) m=magn[i];
    }
    if (m>16) {
        k=0;
        for(i=0; i<bandPass; i++) {
            cl.dw = pgm_read_dword(&colorTab[96*i/bandPass]);
            cl.dw = strip.Color(cl.r, cl.g, cl.b);
            akk.w = cl.r * brightness; cl.r = akk.b1;
            akk.w = cl.g * brightness; cl.g = akk.b1;
            akk.w = cl.b * brightness; cl.b = akk.b1;
            akk.w=LedtoColor*magn[i];
            n=akk.b1;
            for(j=0; j<n; j++) {
                strip.setPixelColor(stripLed/2+k, cl.dw);
                strip.setPixelColor(stripLed/2-k-1, cl.dw);
                k++;
            }
        }
        if (k<stripLed/2) {
            for(j=k; j<stripLed/2; j++) {
                strip.setPixelColor(stripLed/2+k, clBlack);
            }
        }
    }
}

```

```

        strip.setPixelColor(stripLed/2-k-1, clBlack);
        k++;
    }
}
strip.show();
}

```

```

void skazka() {
    TColor cl;
    TWord akk;
    uint8_t i;
    uint16_t thsumm = 0;
    uint8_t th;

    for (i=0;i<bandPass;i++) thsumm += readData[i];
    thsumm /= bandPass;

    for (i=0;i<bandPass;i++) {
        th= (thsumm * threshold[i])>>7;
        if (readData[i]>th) magn[i]=250;
        else {
            if (magn[i]>9) magn[i]-=10;
        }
    }
    for (i=0;i<bandPass;i++) {
        akk.w=magn[i]*brightness;
        uint8_t gain=akk.b1;
        cl.dw = pgm_read_dword(&colorTab[96*i/bandPass]);
        cl=mulGain(cl,gain);
        cl.dw=strip.Color(cl.r,cl.g,cl.b);
        uint8_t n=i*LedtoColor/2;
        for(uint8_t k=0; k<LedtoColor/2; k++) {
            strip.setPixelColor(stripLed/2+n+k, cl.dw);
            strip.setPixelColor(stripLed/2-(n+k)-1, cl.dw);
        }
    }
    strip.show();
}

```

```

void stroboscope(uint8_t wait) {
    TColor cl;
    uint8_t i,j;

    cl.dw = strip.Color(255, 255, 255);
    for(i=0; i<stripLed-1; i++) strip.setPixelColor(i, cl.dw);
    strip.show();
    delay(wait);
    cl.dw = strip.Color(0, 0, 0);
}

```

```

    for(i=0; i<stripLed-1; i++) strip.setPixelColor(i, cl.dw);
    strip.show();
    delay(param);
}

void flashRandom() {
    TColor cl;

    rndNumber1 = random(stripLed-1);
    strip.setPixelColor(rndNumber1, clWhite);
    rndNumber2 = random(stripLed-1);
    strip.setPixelColor(rndNumber2, clWhite);
    strip.show();
    delay(10);
    strip.setPixelColor(rndNumber1, clBlack);
    strip.setPixelColor(rndNumber2, clBlack);
    strip.show();
    delay(param);
}

void flashRandomColor() {
    rndNumber1 = random(stripLed-1);
    strip.setPixelColor(rndNumber1,
pgm_read_dword(&colorTab[random(96)]));
    rndNumber2 = random(stripLed-1);
    strip.setPixelColor(rndNumber2,
pgm_read_dword(&colorTab[random(96)]));
    strip.show();
    delay(10);
    strip.setPixelColor(rndNumber1, clBlack);
    strip.setPixelColor(rndNumber2, clBlack);
    strip.show();
    delay(param);
}

void flashColor1() {
    TColor cl;

    strip.setPixelColor(rndNumber1, clBlack);
    strip.setPixelColor(rndNumber2, clBlack);
    cl.dw = pgm_read_dword(&colorTab[random(96)]);
    cl.r >>= 2; cl.g >>= 2; cl.b >>= 2;
    strip.setPixelColor(random(stripLed), cl.dw);
    cl.dw = pgm_read_dword(&colorTab[random(96)]);
    cl.r >>= 2; cl.g >>= 2; cl.b >>= 2;
    strip.setPixelColor(random(stripLed), cl.dw);
    rndNumber1 = random(stripLed);
    strip.setPixelColor(rndNumber1, clWhite);
    rndNumber2 = random(stripLed);

```

```

    strip.setPixelColor(rndNumber2, clWhite);
    strip.show();
    delay(50);
}

void flashColor2() {
    TColor cl;

    cl.dw = pgm_read_dword(&colorTab[random(96)]);
    cl.r >>= 2; cl.g >>= 2; cl.b >>= 2;
    strip.setPixelColor(rndNumber1, cl.dw);
    rndNumber1 = random(stripLed);
    strip.setPixelColor(rndNumber1, clWhite);
    strip.show();
    delay(50);
}

void flashColor3() {
    TColor cl;

    strip.setPixelColor(rndNumber1, clBlack);
    strip.setPixelColor(rndNumber2, clBlack);
    if (nPhase++ > 128) nPhase=0;
    if (nPhase > 96) {
        cl.dw = pgm_read_dword(&colorTab[random(96)]);
        cl.r >>= 2; cl.g >>= 2; cl.b >>= 2;
        strip.setPixelColor(random(stripLed), cl.dw);
        cl.dw = pgm_read_dword(&colorTab[random(96)]);
        cl.r >>= 2; cl.g >>= 2; cl.b >>= 2;
        strip.setPixelColor(random(stripLed), cl.dw);
        cl.dw = pgm_read_dword(&colorTab[random(96)]);
        cl.r >>= 2; cl.g >>= 2; cl.b >>= 2;
        strip.setPixelColor(random(stripLed), cl.dw);
        cl.dw = pgm_read_dword(&colorTab[random(96)]);
        cl.r >>= 2; cl.g >>= 2; cl.b >>= 2;
        strip.setPixelColor(random(stripLed), cl.dw);
    }
    rndNumber1 = random(stripLed);
    strip.setPixelColor(rndNumber1, clWhite);
    rndNumber2 = random(stripLed);
    strip.setPixelColor(rndNumber2, clWhite);
    strip.show();
    delay(50);
}

void ColorWhite(uint8_t colorNumber) {
    uint8_t i;
    uint16_t br;
    TColor cl;
    if (colorNumber<48) {

```

```

        colorNumber *= 2;
        if (colorNumber>95) colorNumber=95;
        cl.dw = pgm_read_dword(&colorTab[colorNumber]);
        cl.dw = strip.Color(cl.r, cl.g, cl.b);
        for(i=0; i<stripLed; i++) strip.setPixelColor(i, cl.dw);
    }
    else {
        br = 63 - colorNumber;
        br *= 16;
        if (br>239) colorNumber = 255;
        cl.dw=strip.Color(br, br, br);
        for(i=0; i<stripLed; i++) strip.setPixelColor(i, cl.dw);
    }
    strip.show();
}

// Slightly different, this makes the rainbow equally
distributed throughout
void rainbowCycle() {
    uint16_t i;

    for(i=0; i< strip.numPixels(); i++) strip.setPixelColor(i,
Wheel(((i * 256 / strip.numPixels()) + progStep) & 255));
    strip.show();
    delay(param);
}

//Theatre-style crawling lights with rainbow effect
void theaterChaseRainbow() {
    for (int q=0; q < 3; q++) {
        for (int i=0; i < strip.numPixels(); i=i+3)
strip.setPixelColor(i+q, Wheel( (i+progStep) % 255));    //turn
every third pixel on
        strip.show();
        delay(param);
        for (int i=0; i < strip.numPixels(); i=i+3)
strip.setPixelColor(i+q, 0);    //turn every third pixel off
    }
}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
    WheelPos = 255 - WheelPos;
    if(WheelPos < 85) {
        return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
    }
    if(WheelPos < 170) {
        WheelPos -= 85;

```

```

        return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
    }
    WheelPos -= 170;
    return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
}

void setMode() {
    if (readData[20]<240) {
        prog = readData[20]>>3;
        subprog = readData[20] & 7;
    }
    else {
        prog = 30;
        subprog = readData[20]-240;
    }
    param = readData[21];
    brightness = readData[22];
    rotate = readData[23];
    paramTab1[prog]=param;
}

void clSet(uint8_t spr) {
    TDWord k;
    /*
    switch (spr) {
        case 0: { clN[0].dw=pgm_read_dword(&colorTab[0]);
clN[1].dw=pgm_read_dword(&colorTab[16]);
clN[2].dw=pgm_read_dword(&colorTab[64]);
clN[3].dw=pgm_read_dword(&colorTab[80]); break; }
        case 1: { clN[0].dw=pgm_read_dword(&colorTab[16]);
clN[1].dw=pgm_read_dword(&colorTab[32]);
clN[2].dw=pgm_read_dword(&colorTab[64]);
clN[3].dw=pgm_read_dword(&colorTab[48]); break; }
        case 2: { clN[0].dw=pgm_read_dword(&colorTab[32]);
clN[1].dw=pgm_read_dword(&colorTab[64]);
clN[2].dw=pgm_read_dword(&colorTab[16]);
clN[3].dw=pgm_read_dword(&colorTab[48]); break; }
        case 3: { clN[0].dw=pgm_read_dword(&colorTab[48]);
clN[1].dw=pgm_read_dword(&colorTab[0]);
clN[2].dw=pgm_read_dword(&colorTab[80]);
clN[3].dw=pgm_read_dword(&colorTab[16]); break; }
        case 4: { clN[0].dw=pgm_read_dword(&colorTab[64]);
clN[1].dw=pgm_read_dword(&colorTab[16]);
clN[2].dw=pgm_read_dword(&colorTab[32]);
clN[3].dw=pgm_read_dword(&colorTab[80]); break; }
        case 5: { clN[0].dw=pgm_read_dword(&colorTab[80]);
clN[1].dw=pgm_read_dword(&colorTab[16]);
clN[2].dw=pgm_read_dword(&colorTab[0]);
clN[3].dw=pgm_read_dword(&colorTab[48]); break; }
        case 6: { clN[0].dw=pgm_read_dword(&colorTab[96]);
clN[1].dw=pgm_read_dword(&colorTab[64]);

```

```

clN[2].dw=pgm_read_dword(&colorTab[48]);
clN[3].dw=pgm_read_dword(&colorTab[16]); break; }
    case 7: { clN[0].dw=pgm_read_dword(&colorTab[96]);
clN[1].dw=pgm_read_dword(&colorTab[0]);
clN[2].dw=pgm_read_dword(&colorTab[32]);
clN[3].dw=pgm_read_dword(&colorTab[64]); break; }
    }
    */
    k.dw=pgm_read_dword(&progColor[spr]);
    clN[0].dw=pgm_read_dword(&colorTab[k.b0]);
    clN[1].dw=pgm_read_dword(&colorTab[k.b1]);
    clN[2].dw=pgm_read_dword(&colorTab[k.b2]);
    clN[3].dw=pgm_read_dword(&colorTab[k.b3]);
}

```

```

TColor multsBrightness(TColor cl) {
    TWord akk;

    akk.w = cl.r * brightness; cl.r = akk.b1;
    akk.w = cl.g * brightness; cl.g = akk.b1;
    akk.w = cl.b * brightness; cl.b = akk.b1;
    return cl;
}

```

```

TColor multsGain(TColor cl, uint8_t br) {
    TWord akk;

    akk.w = cl.r * br; akk.w = akk.b1 * akk.b1; cl.r = akk.b1;
    akk.w = cl.g * br; akk.w = akk.b1 * akk.b1; cl.g = akk.b1;
    akk.w = cl.b * br; akk.w = akk.b1 * akk.b1; cl.b = akk.b1;
    return cl;
}

```

```

void cmdMusical() {
    if (prog==30) {
        switch (subprog) {
            case 0: { zmu1(); break; } // 240
            case 1: { zmu2(); break; }
            case 2: { zmu3(); break; }
            case 3: { zmu4(); break; }
            case 4: { zmu5(); break; }
            case 5: { zmu6(); break; }
            case 6: { zmu7(); break; }
            case 7: { zmu8(); break; } // 247
            case 8: { zmu9(); break; } // 248
            case 9: { zmu10(); break; } // 249
            case 10: { skazka(); break; } // 250
        }
    }
}

```

```

void cmdRunning(uint8_t t) {

    progStep=0;
    progSubStep=0;
    if (prog==0) {
        switch (subprog) {
            case 0: case 1: case 2: { sub00(); break; }
            case 3: case 4: case 5: { sub01(); break; }
            case 6: { sub06(); break; }
            case 7: { sub07(); break; }
        }
    }
    else {
        clSet(subprog);
        switch (prog) {
            case 1: { sub1(); break; };
            case 2: { sub2(); break; };
            case 3: { sub3(); break; };
            case 4: { sub4(); break; };
            case 5: { sub5(); break; };
            case 6: { sub6(); break; };
            case 7: { sub7(); break; };
            case 8: { sub8(); break; };
            case 9: { sub9(); break; };
            case 10: { progSubStep=stripLed-1; sub10(); break; };
            case 11: { progSubStep=stripLed-1; sub11(); break; };
            case 12: { progSubStep=stripLed-1; sub12(); break; };
            case 29: {
                switch (subprog) {
                    case 0: { theaterChaseRainbow(); break; }
                    case 1: { rainbowCycle(); break; }
                    case 2: { strip.clear(); flashRandom(); break; }
                    case 3: { strip.clear(); flashRandomColor(); break; }

                    case 4: { flashColor1(); break; }
                    case 5: { flashColor2(); break; }
                    case 6: { flashColor3(); break; }
                    case 7: { stroboscope(t); break; }
                }
                break;
            }
        }
    }
}

void loop(){

```

```

        if (radio.available()) { // Если в буфере имеются
принятые данные, то получаем номер трубы, по которой они пришли, по
ссылке на переменную pipe
            radio.read(&readData, 24); // Приём команды
            if (readData[20] < 240) { setMode(); cmdRunning(param); }
            else {
                switch (readData[20]) {
                    case 251: { setMode(); ColorWhite(param);
strip.show(); break; } // Установка цвета ленты
                    case 252: {
// Изменение параметров
                        param = readData[21];
                        brightness = readData[22];
                        rotate = readData[23];
                        if (rotate > 0) {
                            if (prog > 12) prog = 0;
                            rotateStep = pgm_read_byte(&rotateInterval[prog]);
                        }
                        break;
                    }
                    default: { setMode(); cmdMusical(); }
                }
            }
        }
    }
    else {
        if (rotate != 0) {
            if (prog < 30) {
                if (progStep == 0) {
                    if (++rotateStep == 0) {
                        prog = random(72);
                        subprog = prog & 7;
                        prog >>= 3;
                        param = paramTabl[prog];
                        clSet(subprog);
                        rotateStep = pgm_read_byte(&rotateInterval[prog]);
                    }
                }
            }
        }
        if (prog < 29) delay(param);
        switch (prog) {
            case 0: {
                switch (subprog) {
                    case 0: case 1: case 2: { sub00(); break; }
                    case 3: case 4: case 5: { sub01(); break; }
                    case 6: { sub06(); break; }
                    case 7: { sub07(); break; }
                }
                break;
            }
            case 1: { sub1(); break; }
            case 2: { sub2(); break; }
            case 3: { sub3(); break; }

```

```

case 4: { sub4(); break; }
case 5: { sub5(); break; }
case 6: { sub6(); break; }
case 7: { sub7(); break; }
case 8: { sub8(); break; }
case 9: { sub9(); break; }
case 10: { sub10(); break; }
case 11: { sub11(); break; }
case 12: { sub12(); break; }
case 29: {
    switch (subprog) {
        case 0: { theaterChaseRainbow(); break; }
        case 1: { rainbowCycle(); break; }
        case 2: { flashRandom(); break; }
        case 3: { flashRandomColor(); break; }
        case 4: { flashColor1(); break; }
        case 5: { flashColor2(); break; }
        case 6: { flashColor3(); break; }
        case 7: { stroboscope(param); break; }
    }
    progStep++;
    break;
}
}
}

```

Аналіз апаратних складових модуля

Таблиця 1 – Порівняльна таблиця плат Arduino

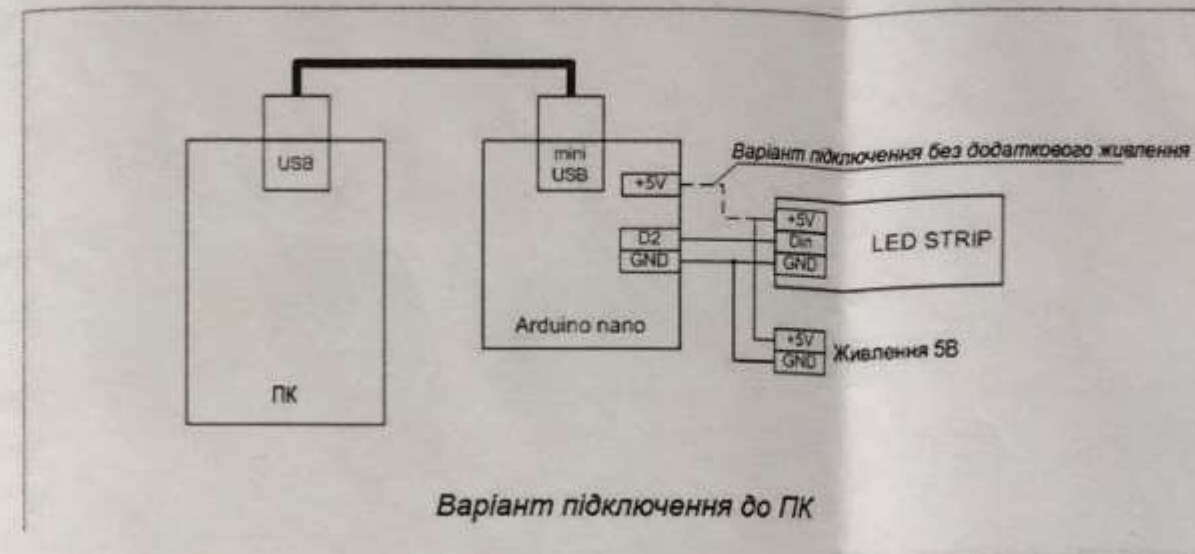
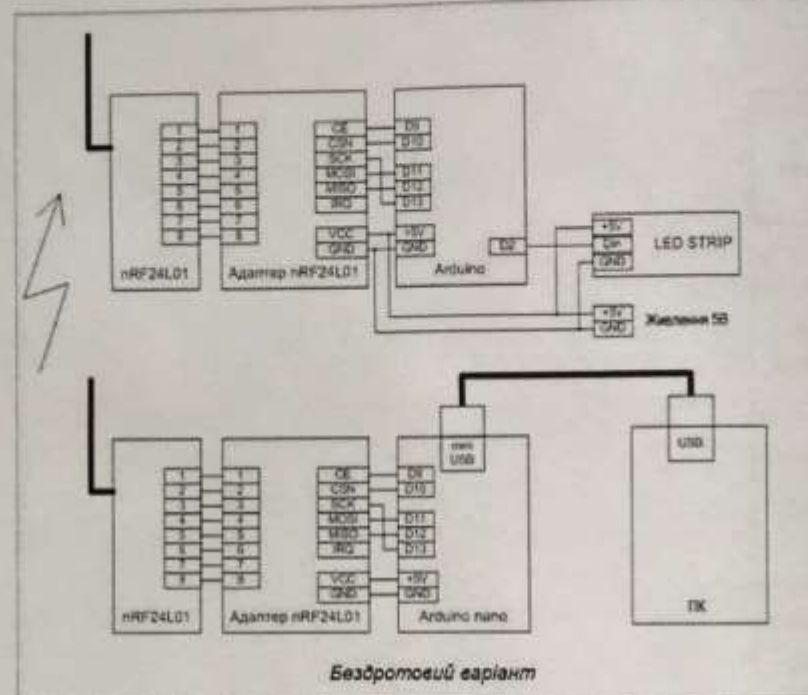
Характеристика	Arduino Uno	Nano	Pro Mini	Mega
Мікроконтролер	ATmega328	ATmega328	ATmega328	ATmega2560
Робоча напруга	5 В	5 В	3,3 або 5В	5 В
Напруга живлення	7-12 В	7-12 В	3,3-12 або 5-12 В	7-12 В
Цифрові виходи	14	14	14	54
Канали ШІМ	6	6	6	15
Аналогові виходи	6	8	8	16
FLASH-пам'ять	32КБ	32 КБ	16 КБ	256 КБ
SRAM	2КБ	2 КБ	1 КБ	8 КБ
Тактова частота	16МГц	16 МГц	8 або 16 МГц	16 МГц
Розмір	68x53 мм	18x43 мм	18x33 мм	101x53 мм
Вбудований програматор	Так	Так	Ні	Так

Таблиця 2 - порівняння технічних характеристик адресних світлодіодних стрічок WS2811, US1903 и WS2818

Характеристика	US1903	WS2811	WS2818
ІС	Зовнішня	Зовнішня	WS2818
Вольтаж	DC12v	DC12v	Зовнішня
Колір	RGB	RGB	DC12v/DC24v
Кількість світлодіодів на метр	60	60	RGB
Споживана потужність	14 Вт/м	14 Вт/м	60
Дроти	Позитивний + від'ємний + Один рядок даних	Позитивний + від'ємний + Один рядок даних	Позитивний + від'ємний + Подвійні лінії передачі даних

13.02070849.00024 ПЛІ				
Зем.	Лист.	№ докум.	Підп.	Дат.
Розроб.	Борошенко В.Г.			
Перш.	Прокопів М.П.			
Г. контр.				
Н. контр.	Щербак Н.В.			
Затв.	Кудерський Р.А.			
Розробка конструкторного модуля генерації звукових і світлових сигналів для дитячого центру Аналіз апаратних складових модуля				
Лист 1				
Листів 1				
НУ «Запорізька політехніка»				
КНТ-527				

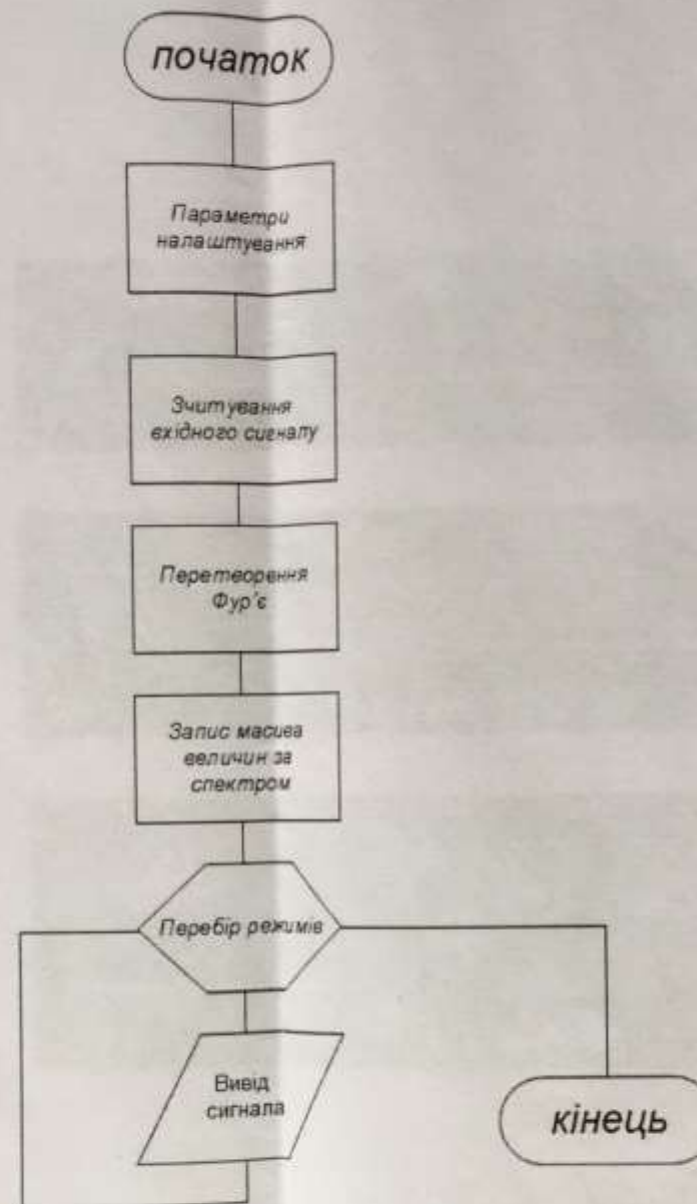
Електрична схема комп'ютерного модуля



Для складання мінімального комплекту, з безпосереднім підключенням до USB ПК, потрібна лише світлодіодна стрічка з WS2812B, arduino nano, три провідника і міні USB кабель для підключення плати Arduino до одного з USB портів ПК.

					13.02070849.00024 ПЛ2			
Зам.	Лист.	№ докум.	Підп.	Датум	Розробка комп'ютерного модуля генерації звукових і світлових сигналів для бітничого центру Електрична схема комп'ютерного модуля	Лист	Маса	Масштаб
Розроб.	Борошенко В.Г.							
Перев.	Проскурін М.П.							
І.контр.						Лист 1	Лист 1	
Н.контр.	Щербак Н.В.				НУ «Запорізька політехніка» КНТ-527			
Затв.	Кудрицька Р.К.							

Алгоритм роботи програми

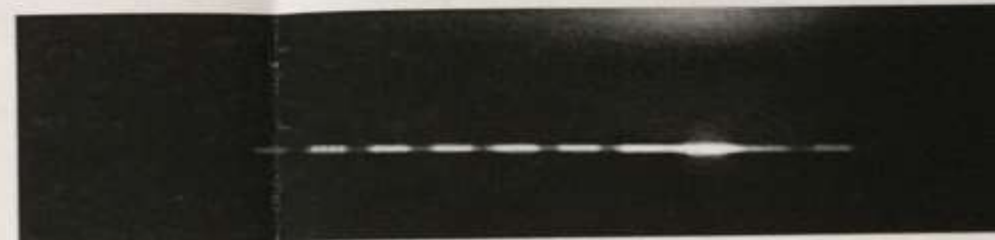


					13.02070849.00024 ПЛЗ		
					Розробка комп'ютерного модуля генерації звукових і світлових сигналів для дитячого центру		
					Алгоритм роботи програми		
Зам.	Лист.	№ докум.	Підп.	Дат.			
Розроб.		Борошенко В.Г.	<i>[Signature]</i>				
Перев.		Проскурін М.П.	<i>[Signature]</i>				
Т. контр.							
Н. контр.		Щербак Н.В.	<i>[Signature]</i>				
Затв.		Кудерський Р.К.	<i>[Signature]</i>				
					Лист 1		
					Лист 1		
					НУ «Запорізька політехніка»		
					КНТ-527		

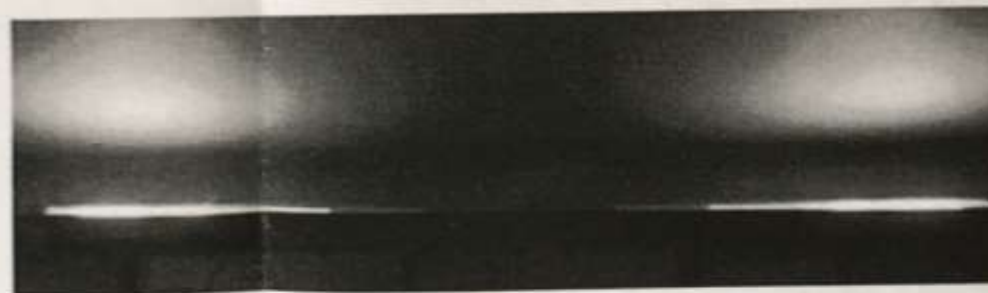
Демонстрація роботи комп'ютерного модуля



Статичний режим
«Веселка»



Динамічний режим
«Веселка»



Статичний режим
«Мікс»

					13.02070849.00024 ПЛ4		
Зам.	Лист	М. докум.	Підп.	Заче	Розробка комп'ютерного модуля генерації звукових і світлових сигналів для дитячого центру		
Розроб.					Демонстрація роботи комп'ютерного модуля		
Перев.							
Г. контр.							
Н. контр.							
Затв.							
					Лист 1	Місця	Місця
					Лист 1		
					НУ «Запорізька політехніка»		
					КНТ-527		