СОДЕРЖАНИЕ

ВВЕДЕНИЕ……..…..3
1 ОБЗОР ЛИТЕРАТУРЫ……..…4
1.1 Микроконтроллеры……..…4
1.2 Тактовые кнопки……...……6
1.3 Осевой переменный резистор……..……8
1.4 Аккумулятор……...……..…9
1.5 Модуль беспроводной передачи данных……..10
2 РАЗРАБОТКА СТРУКТУРЫ УСТРОЙСТВА УДАЛЕННОГО УПРАВЛЕНИЯ КОМПЬЮТЕРА…… ..12
3 ОБОСНОВАНИЕ ВЫБОРА УЗЛОВ, ЭЛЕМЕНТОВ ФУНКЦИОНАЛЬНОЙ СХЕМЫ УСТРОЙСТВА ……...13
4 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ УСТРОЙСТВА……... ..16
4.1 Расчет потребляемой мощности устройства……....17
4.2 Микроконтроллер...……....17
4.3 Модуль регулирования звука.……..…..17
4.4 Тактовые кнопки…….……..……..17
4.5 Модуль повышения напряжения……...……....17
4.6 Модуль контроля заряда……....17
4.7 Модуль передачи данных……..……....18
4.8 Светодиоды……..……..…..18
5 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ……..19
5.1 Требования к разработке программного обеспечения……....19
5.2 Блок-схема алгоритма.……..……..19
5.3 Исходный код программы……..20
5.4 Исходный код драйвера устройства……..20
ЗАКЛЮЧЕНИЕ……... ..21
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……..…..22
ПРИЛОЖЕНИЕ А Структурная схема……....23
ПРИЛОЖЕНИЕ Б Функциональная схема……... ..24
ПРИЛОЖЕНИЕ В Принципиальная схема…...……..25
ПРИЛОЖЕНИЕ Г Исходный код программы……....26
ПРИЛОЖЕНИЕ Д Исходный код драйвера устройства……....28
ПРИЛОЖЕНИЕ Е Блок-схема алгоритма……..39
ПРИЛОЖЕНИЕ Ж Ведомость документов…..……....40
ПРИЛОЖЕНИЕ З Ведомость документов…..……..41

ВВЕДЕНИЕ

В настоящее время, многим активным пользователям компьютеров и ноутбуков не хватает удобства использования своих устройств. Каждый задумывается, как быстрее выполнить какие-нибудь действия или дополнить это новыми функциями, которые обеспечат комфорт использования современных устройств.

Одним из главных достоинств для современного пользователя можно считать параметр удаленности, чтобы выполнять необходимые функции и действия, не подходя к своим устройствам, тем более, если такое устройство является удобным и портативным.

Целью курсового проекта является проектирование и разработка портативного микропроцессорного устройства для удаленного управления компьютера. Функционал данного устройства включает в себя управление громкостью системы, а также полного отключения ее звуков, отдельное выключение устройств вывода звука, возможность выключение и перехода устройства в спящий режим, а также переключение доступных устройств вывода звука и экстренного вызова диспетчера задач с принудительным сворачиванием всех запущенных приложений.

Дополнительным функционалом является возможность установки быстрого доступа к приложениям или файлам любого формата.

Таким образом, грамотно спроектированное устройство удаленного управления компьютером обеспечит комфорт всем пользователям, особенно имеющим профессиональную необходимость в этом: стримерам разных платформ, создателям музыкального и развлекательного контента.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Микроконтроллеры

Для реализации работоспособного устройства необходимо серьезно отнестись к выбору элементной базы, ведь качественно подобранные составные элементы напрямую влияют на результат проекта.

В линейке контроллеров различных производителей есть множество плат, причем каждая из них отличается своими размерами и возможностями, поскольку имеет свое назначение.

Рассмотрим контроллеры Raspberry Pi [1].

Raspberry Pi – это миниатюрный одноплатный компьютер, который несмотря на свои скромные размеры, имеет высокую производительность, что позволяет ему выйти на один уровень со стационарными ПК. Изначально Raspberry Pi была разработана, как учебное пособие по информатике. Но сама идея оказалась настолько удачной, что за несколько лет мини-компьютер стал популярен в очень широких кругах. С течением времени Raspberry Pi пережила несколько модификаций, каждая из которых отличалась от предшественника каким-либо параметром. Такой подход позволил регулировать стоимость изделия в зависимости от потребностей пользователя, что также положительно сказалось на популярности устройства. Вся линейка Raspberry Pi применяет процессоры с ARM-архитектурой, которая зарекомендовала себя с лучшей стороны. На рисунке 1.1 представлены самая младшая модель в линейке Raspberry Pi и самая последняя модификация контроллера.

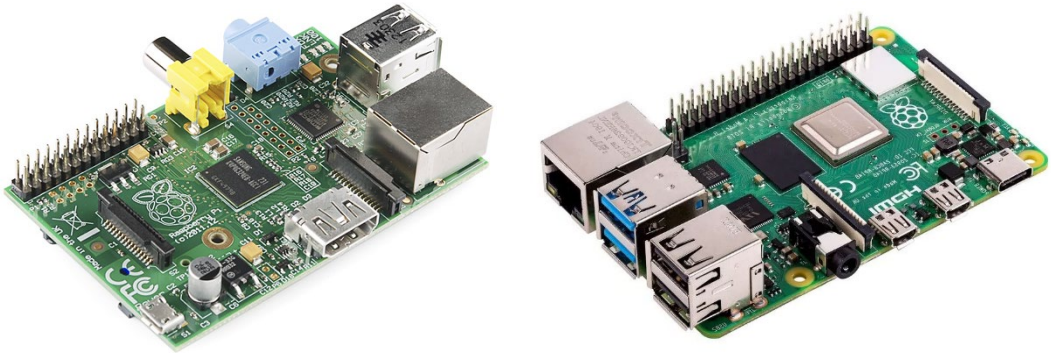

Рисунок 1.1 – Вид плат Raspberry Pi B и Raspberry Pi 4B

На сегодняшний день существует 11 разновидностей Raspberry Pi. Последние версии оснащены беспроводными WiFi и Bluetooth модулями, расширяющими границы применения контроллеров в области Ethernet-технологий.

В таблице 1.1 приведена сравнительная таблица, в которой отражены особенности каждой модификации с указанием некоторых технических данных.

Таблица 1.1 – Сравнительная характеристика контроллеров Raspberry Pi

Модификация	Процессор	Тактовая частота	Объем ОЗУ	Wi-Fi	Поддержка Bluetooth
B	ARM1176JZ-F	700 МГц	512 Мб	–	–
A	ARM1176JZ-F	700 МГц	256 Мб	–	–
B+	ARM1176JZ-F	700 МГц	512 Мб	–	–
A+	ARM1176JZ-F	700 МГц	256 Мб	–	–
2B	ARM Cortex-A7	900 МГц	1 Гб	–	–
Zero	ARM1176JZ-F	1 ГГц	512 Мб	–	–
3B	Cortex-A53	1,2 ГГц	1 Гб	+	4.1
Zero W	ARM1176JZ-F	1 ГГц	512 Мб	+	4.0
3B+	Cortex-A53	1,4 ГГц	1 Гб	+	4.2
3A+	Cortex-A53	1,4 ГГц	512 Мб	+	4.2
4B	Cortex-A72	1,5 ГГц	1–4 Гб	+	5.0

Хоть Raspberry Pi внешне может напомнить Arduino, он всё-таки использует кардинально другой метод функционирования. Данная плата, как и обычный ПК, работает под управлением одной из специализированных операционных систем.

Теперь рассмотрим две платы Arduino: Uno и Nano, представленные на рисунке 1.2. Arduino – это марка микроконтроллеров для управления системами автоматики и робототехники. Плата ориентирована на начинающих пользователей [2]. Контроллер сочетает в себе функции мини-компьютера с запоминающими устройствами ОЗУ и ПЗУ. Система выполняет задачи по программированию необходимой техники.

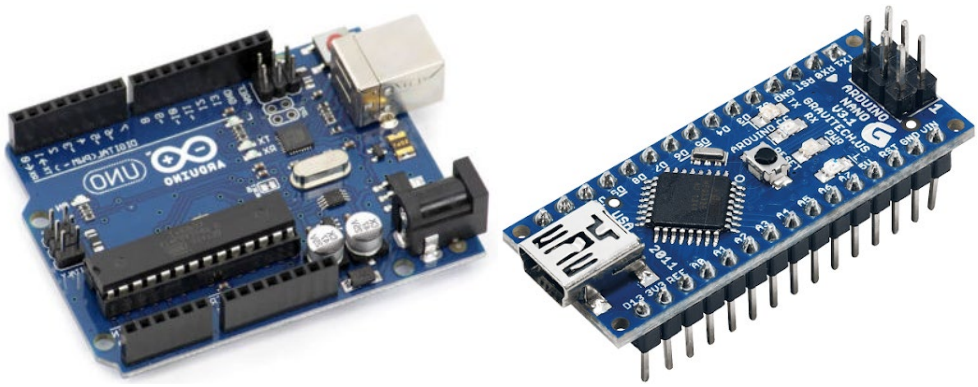

Рисунок 1.2 – Вид плат Arduino Uno и Arduino Nano

Arduino Uno является стандартной платой Arduino и наиболее распространенной. Она основана на чипе ATmega328, имеющем 32 КБ флэш-памяти.

Плата имеет 14 цифровых и 6 аналоговых каналов входа, что позволяет перекрывать большинство любительских задач в области микроконтроллерной техники. Данная плата контроллера является одной из самых дешевых и наиболее часто используемых.

Nano – одна из самых миниатюрных плат Arduino. Она является полным аналогом Arduino Uno – так же работает на чипе ATmega328. Из-за своих габаритных размеров плата часто используется в проектах, в которых важна компактность. На плате отсутствует вынесенное гнездо внешнего питания, контроллер работает через USB.

Полная сравнительная характеристика контроллеров представлена в таблице 1.2.

Таблица 1.2 – Характеристика плат Arduino Uno и Arduino Nano

Характеристика	Плата	
	Arduino Uno	Arduino Nano
Микроконтроллер	ATmega328	ATmega328
Напряжение питания	5 В	5 В
Количество цифровых входов	14	14
Количество аналоговых входов	6	8
Максимальный ток цифрового выхода	40 мА	40 мА
Флэш-память	32 Кб	32 Кб
ОЗУ	2 Кб	2 Кб
Тактовая частота	16 МГц	16 МГц
Размеры	69 х 53 мм	18 х 45 мм

Проанализировав характеристики в таблице 1.2, можно сделать вывод, что платы Arduino Uno и Arduino Nano практически аналогичны по своим показателям, за исключением габаритов.

1.2 Тактовые кнопки

Основа устройства - это тактовые кнопки, с помощью которых будет производится удаленное управление компьютером.

Рассмотрим тактовую кнопку KLS7-TS6601-17, изображенную на рисунке 1.3, системные характеристики которой представлены в таблице 1.3.

Таблица 1.3 – Характеристика тактовой кнопки KLS7-TS6601-17

Характеристика	Параметры
Размер, мм	6x6x17
Размер с учетом толкателя, мм	17
Типоразмер кнопки, мм	6x6
Ориентация отн. платы	Прямая (Вертикальная)
Способ монтажа	DIP (в отверстия)
Количество выводов	4

Рисунок 1.3 – Вид тактовой кнопки KLS7-TS6601-17

Далее рассмотрим тактовую кнопку MX1A-21NW, изображенную на рисунке 1.4

Рисунок 1.4 – Вид тактовой кнопки MX1A-21NW

Тактовую кнопку MX1A-21NW можно считать премиум сегментом, которая используется в производстве механических клавиатур. Технические характеристики представлены в таблице 1.4.

Таблица 1.4 – Характеристика тактовой кнопки MX1A-21NW

Характеристика	Параметры
Размер, мм	15.6x15.6x15.2
Размер с учетом толкателя, мм	15,2
Типоразмер кнопки, мм	15.6x15.6
Ориентация отн. платы	Прямая (Вертикальная)
Способ монтажа	DIP (в отверстия)
Количество выводов	4

1.3 Осевой переменный резистор

В создаваемом устройстве важной функцией является изменение громкости системы. Для этого необходим осевой переменный резистор. Рассмотрим их технические параметры, представленные в таблице 1.5.

Таблица 1.5 – Характеристика 16K1 F 100K и 16K2 KC 2x1K

Характеристика	Осевой переменный резистор	
	16K1 F 100K	16K2 KC 2x1K
Вид вала (поперечный срез)	Сплошной с лыской (полукруг), ВС-3	С рифлением и шлицем (звездочка), ВС-6
Диаметр оси, мм	6	6
Монтаж	в плату и на корпус	в плату и на корпус
Число оборотов	Однооборотный	Однооборотный
Длина оси, мм	16	16
Исполнение	Одинарный	Сдвоенный
Сопротивление	100 кОм	1 кОм
Мощность, Вт	0,125	0,125
Наличие выключателя	Без выключателя	Без выключателя
Характеристика сопротивления	линейная	линейная

Вид осевых переменных резисторов 16K1 F 100K и 16K2 KC 2x1K представлен на рисунке 1.5.

Рисунок 1.5 – Вид осевых переменных резисторов 16K1 F 100K и 16K2 KC 2x1K

1.4 Аккумулятор

Для поддержания удаленной работы необходим источник питания. Для удобства использования были выбраны пальчиковые аккумуляторы.

Рассмотрим пальчиковые аккумуляторы 18650 и US18650VTC4, представленные на рисунке 1.6.

Рисунок 1.6 – Вид пальчиковых аккумуляторов 18650 и US18650VTC4

Сравнительная характеристика пальчиковых аккумуляторов представлена в таблице 1.6.

Таблица 1.6 – Характеристика пальчиковых аккумуляторов 18650 и US18650VTC4

Характеристика	Пальчиковые аккумуляторы	
	18650	US18650VTC4
Типоразмер	18650	18650
Ёмкость, мАч	800	2100
Напряжение, В	3,7	3,6
Защита батареи	Без защиты	Без защиты
Химический состав	Li-Ion	Li-Ion
Вид контактов (выводов)	Без выводов	Без выводов
Метод подключения (монтажа)	В держатель, отсек	В держатель, отсек
Ток разряда (Iразр), А	30	30

1.5 Модуль беспроводной передачи данных

Для обеспечения выполнения функций, необходимо передавать данные с устройства в систему, для этого были выбраны Bluetooth модули, представленные на рисунке 1.7.

HC-05[3] – одно из лучших решений для организации двусторонней связь по Bluetooth Arduino-устройства с планшетом, ноутбуком или другим Bluetooth-устройством. Bluetooth-модуль HC-05, может работать как master (осуществлять поиск Bluetooth-устройств и инициировать установку связи), так и slave (ведомое устройство)..

HC-06 - модуль, который используется для реализации беспроводной связи с различными устройствами, такими как телефон или планшет, а так же может использоваться для обмена данными между двумя модулями Arduino.

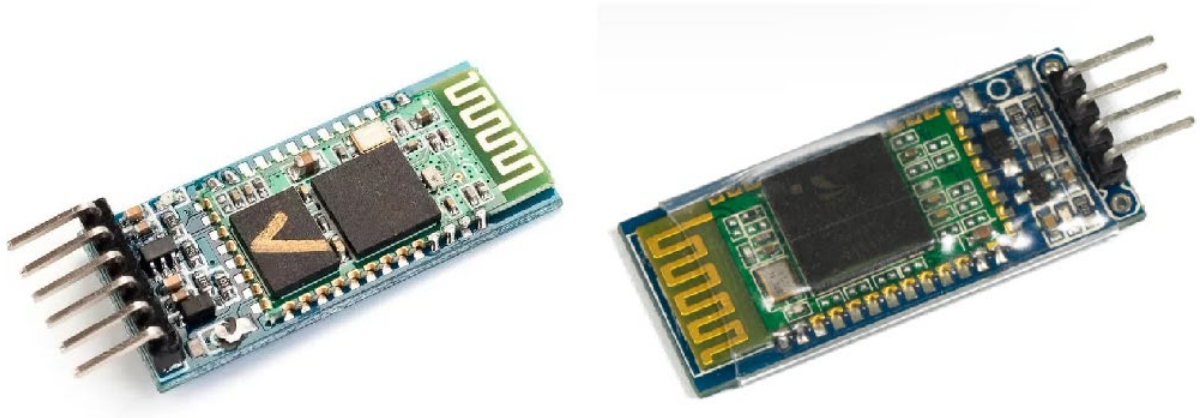

Рисунок 1.7 – Вид Bluetooth модулей HC-05 и HC-06

Сравнительная характеристика модулей передачи данных представлена в таблице 1.7.

Таблица 1.7 – Характеристика датчиков HC-05и HC-06

Характеристика	Модуль передачи данных	
	HC-05	HC-06
Питание	3,3–5 В	3,3В – 6 В
Максимальный ток	50 мА	45 мА
Скорость передачи данных	1200–1382400 бод	1200–1382400 бод
Рабочие частоты	2,4–2,48 ГГц	2,40 ГГц – 2,48ГГц
Поддержка спецификации bluetooth	версия 2.1	версия 2.1
Дальность связи	до 10 метров	до 30 м

Таким образом, рассмотрены разновидности всех основных модулей, необходимых для выполнения данного курсового проекта.

2 РАЗРАБОТКА СТРУКТУРЫ УСТРОЙСТВА УДАЛЕННОГО УПРАВЛЕНИЯ КОМПЬЮТЕРОМ

В этом разделе будет описана общая структура системы и обосновано ее построение в том виде, в котором она представлена на структурной схеме в Приложении А. Ниже будет описана структура, без учета характеристик и моделей устройств.

Для работы устройства можно выделить основные блоки: блок микроконтроллера, блок управления устройством, блок управления системой, блок питания, модуль повышения напряжения, модуль регулирования звука и модуль передачи данных.

Для реализации блока микроконтроллера используется аппаратная платформа Arduino Nano на основе микроконтроллера ATmega328p. Блок осуществляет обмен данными со всеми остальными блоками.

Блок управления устройством представляет из себя несколько кнопок, которые настраивают работу самого устройства. Блок микроконтроллера подает данные из этого блока.

Блок управления системой состоит из набора кнопок, которые определяют выполнение возложенных функций. Данные из этого блока передаются в блок микроконтроллера и после анализа входных данных передаются в блок передачи данных.

Блок питания обеспечивает работу блока микроконтроллера, а также поддерживает функцию зарядки. Этот блок соединен с блоком преобразования напряжения.

Блок модуля повышения напряжения содержит устройство, которое повышает напряжение из блока питания в блок микроконтроллера.

Блок модуля регулирования звука передает данные переменного резистора в блок микроконтроллера, после анализа полученных данных они будут переданы в блок передачи данных.

Блок передачи данных обеспечивает удаленное управление компьютером, путем передачи полученных от блока микроконтроллера данных в компьютер.

3 ОБОСНОВАНИЕ ВЫБОРА УЗЛОВ, ЭЛЕМЕНТОВ ФУНКЦИОНАЛЬНОЙ СХЕМЫ УСТРОЙСТВА

Источник бесперебойного питания должен обеспечивать круглосуточную работу любого устройства, которое подключено к нему, с сохранением выходных параметров, поэтому к нему выдвигаются жесткие требования, как к конструкции, так и к выбору элементов схемы.

Условно элементы схемы можно разделить на элементы общего применения и специальные.

Элементы общего применения являются изделиями массового производства, поэтому они достаточно широко стандартизированы. Стандартами и нормами установлены технико-экономические и качественные показатели, параметры и размеры элементов. Такие элементы называют типовыми. Выбор типовых элементов проводится по параметрам и характеристикам, которые описывают их свойства, как при нормальных условиях эксплуатации, так и при разных влияниях (климатических, механических и др.).

Основными электрическими параметрами является: номинальное значение величины, характерной для данного элемента (сопротивление резисторов, емкость конденсаторов, индуктивность катушек и т. д.) и границы допустимых отклонений; параметры, которые характеризуют электрическую прочность и способность долгосрочно выдерживать электрическую нагрузку; параметры, которые характеризуют потери, стабильность и надежность.

Основными требованиями, которыми приходилось руководствоваться при проектировании охранного устройства, являются требования по наименьшей стоимости изделия, его высокой надежности и минимальным малогабаритным показателям. Исходя из перечисленных выше критериев были выбраны наиболее лучшие элементы.

В качестве основного элемента был выбран контроллер Arduino Nano, который наилучшим образом подходил для реализации поставленной цели. Характеристики Arduino Nano были приведены в таблице 1.1. Ключевым фактором является многофункциональность и малогабаритность данного микроконтроллера.

Для реализации блока управления системой была выбраны тактовые кнопки KLS7-TS6601-17, характеристики которых приведены в таблице 1.3. Данный выбор обусловлен высотой кнопки, ее дешевизной и удобством подключения.

Для обеспечения питания устройства были выбраны аккумуляторы 18650, характеристики которых приведены в таблице 1.4. Для зарядки аккумуляторов был выбран модуль контроля заряда MOD-TP4056-MMUSB-P[4], его характеристики приведены в таблице 3.1. Для повышения напряжения аккумуляторов был выбран модуль MT3608[5], характеристики которого приведены в таблице 3.2. Все модули были выбраны учитывая их показатели надежности и удобства использования.

Таблица 3.1 – Характеристика модуля контроля заряда MOD-TP4056-MMUSB-P

Характеристика	Параметры
Назначение	Контроль заряда
Напряжение рабочее, В	3.7 - 4.2

Таблица 3.2 – Характеристика модуля повышения напряжения MT3608

Характеристика	Параметры
Исполнение	Встраиваемая плата
Напряжение вых., В (Vout)	4.5-28
Ток вых.макс., А	2
Регулируемый	Да
Тип	повышающий
Напряжение входное, В	2-24
Выходные контакты	Клеммник винтовой
Наличие и цвет индикатора	Без индикатора

Для передачи данных из микроконтроллера был выбран Bluetooth модуль HC-05, характеристики которого приведены в таблице 1.7. Выбор обусловлен тем, что модуль поддерживает режим передачи и приема данных, что в перспективе поможет улучшить устройство. Также его низкая стоимость среди аналогов и отличная работоспособность.

Для регулирования звука, был выбран осевой переменный резистор 16K1 F 100K, характеристики которого приведены в таблице 1.3. Учитывая удобство подключения, отличную надежность и высокое сопротивление, выбор был сделан в пользу этого компонента.

Для управления самим устройством, были выбраны кнопки L-KLS7-TS1202-11-180, характеристика которых приведена в таблице 3.3 в совокупности со светодиодами LEDF3, характеристика которых приведена в таблице 3.4. Эти модели наилучшим образом вписываются в проект, учитывая дешевизну и лучшие показатели среди аналогов.

Таблица 3.3 – Характеристика модуля контроля заряда MOD-TP4056-MMUSB-P

Характеристика	Параметры
Размер, мм	12x12x11
Размер с учетом толкателя, мм	11
Типоразмер кнопки, мм	12x12
Ориентация отн. платы	Прямая (Вертикальная)
Способ монтажа	DIP (в отверстия)
Количество выводов	4

Таблица 3.4 – Характеристика светодиодов LEDF3

Характеристика	Параметры
Цвет свечения	Белый, синий
Диаметр (размер), мм	3

Электрическая функциональная схема представлена в Приложении Б.

4 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ УСТРОЙСТВА

4.1 Расчет потребляемой мощности устройства

Принципиальная схема служит для передачи с помощью условных графических знаков связей между элементами электрического устройства. Данная схема позволяет понять, как работает устройство, как его детали соединены друг с другом.

Принципиальная схема микропроцессорного устройства контроля параметров супермаркета представлена в приложении В.

Потребляемая мощность разрабатываемого устройства равна сумме мощностей, потребляемых его элементами. Расчет мощности элементов схемы представлен в таблице 4.1.

Таблица 4.1 – Расчет мощности элементов схемы

Блок	U, В	I, мА	P, мВт
Тактовая кнопка	5	50	50
Резистор, 220 Ом	5	0.5	2.5
Резистор, 10 кОм	5	0.5	2.5
Осевой переменный резистор, 100 кОм	5	25	125
Bluetooth модуль	5	50	250
Модуль повышения напряжения	5	2	10
Модуль контроля заряда	5	1	5
Светодиод	5	30	150
Суммарная мощность, мВт			565

В реализованной схеме используются: 18 тактовых кнопок, 6 резисторов 220 Ом и 2 резистора 10 кОм, 1 осевой переменный резистор 100 кОм, 1 модуль повышения напряжения, 1 модуль контроля заряда, 6 светодиодов и 1 Bluetooth-модуль.

Таким образом потребляемая мощность будет равна:

P = 50*18 + 2.5*6 + 2.5*2 + 125 + 250 + 10 + 5 + 150*6 = 2210 мВт.

Учитывая поправочный коэффициент в 20%, максимальная потребляемая мощность составит 2652 мВт. Рассчитаем потребляемый ток:

$$I = \frac{P}{U} = \frac{2{,}652}{5} = 0{,}530 \approx 0{,}53 \text{ А}.$$

4.2 Микроконтроллер

Информация о характеристиках выбранного микроконтроллера представлена в разделе 1.

На аналоговый вход A5 поступает сигнал с переменного резистора.

На цифровые входы D12-D5 поступает сигнал с матрицы кнопок. На Вход D2-D3 подается сигнал с отдельных кнопок, которые управляют самим устройством.

На входы RSD и TXD подается сигнал и передается сигнал с модуля передачи информации. На вход 5V подаётся напряжение источника питания – 5В.

На цифровой вход D4 подается сигнал работы с светодиодами.

4.3 Модуль регулирования звука

Для регулирования звука в системе используется переменный осевой резистор 16K1 F 100K, информация из которого передается на вход микроконтроллера и после обработки передается по блоку передачи информации в систему. Согласно спецификации, на вход Vin подается подаётся напряжение источника питания – 5В. Остальные входы используются для подключения к микроконтроллеру.

4.4 Тактовые кнопки

Для выбора действий управления компьютером были выбраны тактовые кнопки KLS7-TS6601-17 и L-KLS7-TS1202-11-180 для управление устройством. Данные о нажатии на кнопки передаются на цифровые входы D12-D5 и D2-D3 микроконтроллера.

4.5 Модуль повышения напряжения

Для повышения напряжения из блока питания используется модуль MT3608, выходы +VOUT и -VOUT подключены со входами Vin и GND микроконтроллера, соответственно. Входы +VIN и -VIN подключены с модулем контроля заряда.

4.6 Модуль контроля заряда

Для зарядки блока питания и подачи напряжения на микроконтроллер используется модуль TO4056. На входы OUT+ и OUT- подается напряжение блока питания Vin и GND, соответственно. Выходы B+ и B- соединены с входами +VIN и -VIN модуля повышения напряжения.

4.7 Модуль передачи данных

В качестве модуля передачи данных используется Bluetooth HC-05, для работы модуля его входы 5V и GND подается напряжение 5V и GND с микроконтроллера соответственно. Для обмена информацией с микроконтроллером задействованы входы RX и TX, которые соединены со входами RXD и TXD микроконтроллера, соответственно.

4.8 Светодиоды

Для освещения устройства были выбраны светодиоды LEDF3. Которые получают информацию со входа D4 микроконтроллера и задействуют его вход GND.

5 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

5.1 Требования к разработке программного обеспечения

Разработка программного обеспечения для микроконтроллера преследует ряд важных задач, учитывая своеобразность написание кода под микроконтроллеры: код должен быть эффективным и легкочитаемым, также необходимо учитывать тот фактор, что объем памяти в микроконтроллере сильно ограничен, следовательно крайне недопустимо тратить ее попусту.

5.2 Блок-схема алгоритма

Блок A01 является входной точкой программы.

Блок B01 представлен строками 3-21. В данном участке программы происходит инициализация глобальных переменных, которые буду использоваться для корректной работы устройства.

Блок D01 представлен строкой 24. В блоке инициализируются задействованные выходы устройства.

Блок F01 Представлен строками 23, 25. Происходит инициализация порта передачи информации и начальное положение потенциометра.

Блок H01 представлен строкой 27. Блок определяет начало главного цикла устройства.

Блок J01 представлен строкой 28. Блок определяет нажатие клавиш с матрицы кнопок.

Блок L01 представлен строками 32-35. Блок обрабатывает нажатие кнопок и отправляет данные по модулю HC-05, если кнопка была нажата.

Блок N01 представлен строками 38-41 и 49-59. Блок запоминает настройку освещения устройства и переключает ее при нажатии на определенную кнопку.

Блок B03 представлен строками 43-46 и 60-65. Блок перезагружает порт модуля HC-05 при нажатии на определенную кнопку.

Блок D03 представлен строками 47 и 66-78. Блок обрабатывает значения на потенциометре.

Блок F03 представлен строками 66-78. Блок отправляет данные по HC-05 при соблюдении необходимых условий.

Блок H03 представлен строкой 48. В данном блоке происходит завершение главного цикла устройства и осуществляется переход к следующей итерации.

Блок J03 является выходной точкой программы.

Блок-схема алгоритма представлена в приложении Е.

5.3 Исходный код программы

Программное обеспечение устройства написано в соответствии с разработанной блок-схемой. Исходный код программы представлен в приложении Д.

5.4 Исходный код драйвера устройства

Программное обеспечение для компьютера, которое отвечает за подключение разработанного устройства к системе, настройкой и выполнением возложенного функционала представлено в приложении Д.

ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проекта успешно спроектировано и разработано микропроцессорное устройство для удаленного управления компьютера, в функционал которого входит: управление громкостью системы, а также полного отключения ее звуков, отдельное выключение устройств вывода звука, возможность выключение и перехода устройства в спящий режим, а также переключение доступных устройств вывода звука и экстренного вызова диспетчера задач с принудительным сворачиванием всех запущенных приложений.

Дополнительным функционалом является возможность установки быстрого доступа к приложениям или файлам любого формата.

Разработанное микропроцессорное устройство обладает следующими достоинствами: относительно низкая стоимость, простота реализации и сборки, возможность самостоятельно дополнить функционал, учитывая свободный код устройства.

В дальнейшем планируется усовершенствование данного курсового проекта. Одним из таких улучшений является оптимизация алгоритма анализа полученных данных, а также создание более дружественного интерфейса и пополнение функционала устройства и возможность его выбора.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Документация Raspberry Pi [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.raspberrypi.org/documentation/ – Дата доступа: 23.04.2022.

[2] Документация Arduino [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.arduino.cc/en/main/docs – Дата доступа: 23.04.2022.

[3] Документация HC-05 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf – Дата доступа: 23.04.2022.

[4] Документация TO4056 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://static.chipdip.ru/lib/977/DOC002977110.pdf – Дата доступа: 23.04.2022.

[5] Документация MT3608 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.alldatasheet.com/view.jsp?Searchword=Mt3608&gclid=CjwKCAjwx46TBhBhEiwArA_DjGUQd5JADFkyKSLdxKWpdwjIE72Q7d7COIYUdxNKuNitMOgtO4y9rBoCxuwQAvD_BwE– Дата доступа: 23.04.2022.

ПРИЛОЖЕНИЕ А
(обязательное)

Структурная схема

ПРИЛОЖЕНИЕ Б
(обязательное)

Функциональная схема

ПРИЛОЖЕНИЕ В

(обязательное)

Принципиальная схема

ПРИЛОЖЕНИЕ Г
(обязательное)

Исходный текст программы устройства

```
#include <Keypad.h>
#include <SoftwareSerial.h>

#define  blueth_RX 0
#define  blueth_TX 1
const int pinButtonLed = 2;
const int pinButtonReloadBlth = 3;
#define PIN_POT A4
bool boxLed = false;
int rotate;

SoftwareSerial mySerial (blueth_TX, blueth_RX);
const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
{'A', 'B', 'C', 'D'},
{'E', 'F', 'G', 'H'},
{'I', 'J', 'K', 'L'},
{'M', 'N', 'O', 'P'}
};

byte colPins[COLS] = {9, 10, 11, 12}; //к каким выводам подключаем управление строками
byte  rowPins[ROWS] = {5, 6, 7, 8}; //к каким выводам подключаем управление столбцами

Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup() {
mySerial.begin(9600);
pinMode(4, OUTPUT);
rotate = analogRead(PIN_POT);
}

void loop() {

char customKey = customKeypad.getKey();//записывем нажатый символ
int buttonStateLed = digitalRead(pinButtonLed);
int buttonStateReload = digitalRead(pinButtonReloadBlth);

//mtx
if (customKey) { //если что-то нажато
Serial.println(customKey); //выводим нажатый символ в монитор порта
mySerial.println(customKey);
}
///

////////////////////// LED
if (buttonStateLed) {
boxLed = boxButtonSettingsLed(boxLed);
delay(1000);
}
//////////////////////////blth reload
if (buttonStateReload) {
bluetoothReload();
}
///

rotate = sendRotate(analogRead(PIN_POT));
}

bool boxButtonSettingsLed(bool flag ) {
if (!flag) {
digitalWrite(4, HIGH);
flag = true;
} else {
digitalWrite(4, LOW);
flag = false;
}
delay(10);
return flag;
}

void bluetoothReload() {
mySerial.end();
delay(1000);
mySerial.begin(9600);
delay(1000);
}

int sendRotate(int rt) {
```

```
67. if (rt > (rotate + 50) ) {
68. mySerial.println('Q');
69. delay(300);
70. return rt;
71. }
72. if (rt < (rotate - 50)) {
73. mySerial.println('R');
74. delay(300);
75. return rt;
76. }
77. return rotate;
78. }
```

ПРИЛОЖЕНИЕ Д
(обязательное)

Исходный текст программы драйвера устройства

MainWindow.cs

```
using System;
using System.Collections.Generic;
using System.IO.Ports;
using System.Reflection;
using System.Text.RegularExpressions;
using System.Windows;
using System.Windows.Controls;

namespace ButtonBox
{

    public partial class MainWindow : Window
    {
        public SerialPort serialPort = new SerialPort();
        public bool init = false;
        public string mess;
        QuickLaunch quickLaunch;


        public MainWindow()
        {
            InitializeComponent();
            loadPorts();
            loadFastOpenComboBox();
            checkBox.IsChecked = FileSystem.parseAutorunSettings();
        }

        private void loadPorts()
        {
            string[] ports = SerialPort.GetPortNames();

            foreach (string port in ports)
            {
                comboBoxPortsList.Items.Add(port);
            }

        }

        private void loadFastOpenComboBox()
        {
            char signal = 'I';
            for (int i = 9; i < 17; i++)
            {
                comboBoxFastOpenAppList.Items.Add("Button " + i + " (" + signal + ")");
                signal++;
            }

        }
        public bool InitializePort(string portName)
        {
            try
            {
                serialPort.PortName = portName;
                serialPort.BaudRate = 9600;
                if (!serialPort.IsOpen)
                    serialPort.Open();

                serialPort.DataReceived += new SerialDataReceivedEventHandler(DataReceived);

                return true;
            }
            catch
            {
                serialPort.Close();
                return false;
            }

        }

        private void DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
            try
            {

                SerialPort sp = (SerialPort)sender;
                Dispatcher.Invoke((System.Action)(() =>
                {
```

```
                mess = sp.ReadLine();
                mess = mess.Replace("\r", "");
                Console.WriteLine(mess);
                ButtonControl.userChoice(mess);

            }));


        }
        catch (Exception ex)
        {
            Console.WriteLine("Received failed" + ex.Message);
        }
    }

    private void ButtonInitPort_Click(object sender, RoutedEventArgs e)
    {
        //ButtonControl.userChoice("I"); // debug
        if (init == false)
        {
            try
            {
                if (InitializePort(comboBoxPortsList.Text) == true)
                {
                    init = true;
                    Console.WriteLine("init true");
                    labelPortChoice.Content = "Port initialized successfully";
                }
                else
                {
                    //some
                }
            }
            catch
            {
                Console.WriteLine("init failed");
                labelPortChoice.Content = "Port failed to initialize";
            }
        }
    }

    private void buttonMuteMicro_Click(object sender, RoutedEventArgs e)
    {
        ButtonControl.userChoice("A");
    }

    private void buttonCloseWOpenTM_Click(object sender, RoutedEventArgs e)
    {
        ButtonControl.userChoice("D");
    }

    private void buttonMuteSystem_Click(object sender, RoutedEventArgs e)
    {
        ButtonControl.userChoice("B");
    }

    private void buttonChangeOutput_Click(object sender, RoutedEventArgs e)
    {
        ButtonControl.userChoice("C");
    }

    private void buttonDoSleepMode_Click(object sender, RoutedEventArgs e)
    {
        ButtonControl.userChoice("G");
    }

    private void buttonOffPc_Click(object sender, RoutedEventArgs e)
    {
        ButtonControl.userChoice("H");
    }

    private void buttonBindFastOpen_Click(object sender, RoutedEventArgs e)
    {

        try
        {
            string selectItem = Regex.Match(input: comboBoxFastOpenAppList.SelectedItem.ToString(),
@"\(([^)]*)\)").Groups[1].Value;

            string path = textBoxpathFastOpen.Text;

            if (!selectItem.Equals("") && !path.Equals(""))
            {

                if (!System.IO.File.Exists(path))
                {
                    MessageBox.Show("File not found");
                    return;
                }
```

```
                    QuickLaunch quickLaunch = new QuickLaunch(selectItem, path);
                    FileSystem.setBindToFile(quickLaunch);
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error buttonBindFastOpen_Click()" + ex.Message);
            }
        }

        private void comboBoxFastOpenAppList_SelectionChanged(object sender, System.Windows.Controls.Selection-
ChangedEventArgs e)
        {
            ComboBox comboBox = (ComboBox)sender;
            string selectItem = Regex.Match(input: comboBox.SelectedItem.ToString(),
@"\(([^)]*)\)").Groups[1].Value;
            List<QuickLaunch> list = FileSystem.parsBindFile();

            foreach (var item in list)
            {
                if (item.Signal.Equals(selectItem))
                {
                    labelBindPath.Visibility = Visibility.Visible;

                    labelBindPath.Content = "Path: " + item.Filepath.ToString();
                    return;
                }
                else
                    labelBindPath.Content = "Free to Bind";
            }

        }

        private void checkBox_Checked(object sender, RoutedEventArgs e)
        {
            FileSystem.setFileSettings(true);
            try
            {
                Microsoft.Win32.RegistryKey Key =
                Microsoft.Win32.Registry.LocalMachine.OpenSubKey(
                "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\", true);


                Key.SetValue(Title, Assembly.GetExecutingAssembly().Location);
                Key.Close();
            }catch(Exception ex)
            {
                Console.WriteLine("Error checkBox_Checked()" + ex.Message);
            }
        }

        private void checkBox_Unchecked(object sender, RoutedEventArgs e)
        {
            FileSystem.setFileSettings(false);
            try
            {
                Microsoft.Win32.RegistryKey key =
                Microsoft.Win32.Registry.LocalMachine.OpenSubKey(
               "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true);
                key.DeleteValue(Title, false);
                key.Close();
            }catch(Exception ex)
            {
                Console.WriteLine("Error checkBox_Unchecked()" + ex.Message);
            }
        }
    }
}
```

ButtonControl.cs

```
using Microsoft.Toolkit.Uwp.Notifications;
using System;
using System.Collections.Generic;

namespace ButtonBox
{
    class ButtonControl
    {

        static public void userChoice(string choice)
        {
            switch (choice)
            {
                case "A":
                    muteControlMicro();
                    break;
```

```
            case "B":
                muteSystem();
                break;

            case "C":

                break;
            case "D":
                closeWindowsOpenTskMng();
                break;
            case "E":

                break;
            case "F":
                //downVolume();
                outputDeviceChange();
                break;
            case "G":
                sleepMode();
                break;
            case "H":
                shutdownMode();
                break;
            case "I":
                quickStart(choice);
                break;
            case "J":
                quickStart(choice);
                break;
            case "K":
                quickStart(choice);
                break;
            case "L":
                quickStart(choice);
                break;
            case "M":
                quickStart(choice);
                break;
            case "N":
                quickStart(choice);
                break;
            case "O":
                quickStart(choice);
                break;
            case "P":
                quickStart(choice);
                break;
            case "Q":
                upVolume();
                break;
            case "R":
                downVolume();
                break;

            default:
                {
                    Console.WriteLine("User Choice is " + choice);
                    return;
                }

        }

    }
    static public void muteControlMicro()
    {
        try
        {

            if (WindowsSound.muteMicro())
            {
                notification("Micro is off");
            }
            else
            {
                notification("Micro is on");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("muteControlMicro()" + ex.Message);
        }
    }


    static public void notification(string notifText)
    {
        try
        {
            new ToastContentBuilder()
```

```
                .AddArgument("action", "viewConversation")
                .AddArgument("conversationId", 9813)
                .AddText("ButtonBox")
                .AddText(notifText)
                .Show(toast =>
                {
                    toast.ExpirationTime = DateTime.Now.AddSeconds(1);
                });
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error notification()" + ex.Message);
    }

}

static public void muteSystem()
{
    try
    {
        WindowsVolume.Mute();
        muteControlMicro();
        notification("System Volume and micro - off");
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error muteSystem()" + ex.Message);
    }

}

static public void outputDeviceChange()
{
    try
    {
        WindowsSound.outputDevice();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error outputDeviceChange()" + ex.Message);
    }

}


static public void closeWindowsOpenTskMng()
{
    try
    {
        WindowManager.closeWindowsOpenTM(true);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error closeWindowsOpenTskMng()" + ex.Message);
    }

}

static public void sleepMode()
{
    try
    {
        WindowsPower.doStandby();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error sleepMode()" + ex.Message);
    }
}

static public void shutdownMode()
{
    try
    {
        WindowsPower.doShutdown();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error shutdownMode()" + ex.Message);
    }

}

static public void quickStart(string quickChoice)
{
    try
    {
        List<QuickLaunch> quickLaunchList = FileSystem.parsBindFile();
        string path = null;
```

```
                foreach (var item in quickLaunchList)
                {
                    if (item.Signal.Equals(quickChoice))
                        path = item.Filepath;
                }

                System.Diagnostics.Process.Start(path);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error quickStart()" + ex.Message);
            }
        }

        static public void upVolume()
        {
            try
            {
                for (int i = 0; i < 2; i++)
                {
                    WindowsVolume.VolumeUp();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error upVolume()" + ex.Message);
            }
        }
        static public void downVolume()
        {
            try
            {
                for (int i = 0; i < 2; i++)
                {
                    WindowsVolume.VolumeDown();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error downVolume()" + ex.Message);
            }
        }
    }

}
```

FileSystem.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace ButtonBox
{
    class FileSystem
    {
        const string path = "BindList.bin";
        const string settingsPath = "Settings.bin";

        static public void setBindToFile(QuickLaunch quickLaunch)
        {

            try
            {
                if (!System.IO.File.Exists(path))
                {
                    FileStream fstream = new FileStream(path, FileMode.Create);
                    fstream.Close();
                }


                string item = quickLaunch.Signal + "|" + quickLaunch.Filepath;

                var lines = File.ReadAllLines(path).ToList();
                int index = lines.FindIndex(s => s.Contains(quickLaunch.Signal + "|"));

                if (index.Equals(-1))
                {
                    index = 0;
                    lines.Insert(index, item);
                }
                else
                {
                    // добавить месс бокс с вопросом о замене
```

```
                    var result = MessageBox.Show("Button is binded. Do you want to replace the bind?", "But-
tonBox Binds", MessageBoxButton.YesNo, MessageBoxImage.Question, MessageBoxResult.No);
                    switch (result)
                    {
                        case MessageBoxResult.Yes:
                            index = lines.Count - 1;
                            lines.RemoveAt(index);
                            lines.Insert(index, item);
                            break;
                        case MessageBoxResult.No:
                            break;
                    }
                    return;
                }

                File.WriteAllLines(path, lines);
                parsBindFile();
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.Message);
            }
            finally
            {
                Console.WriteLine("Executing finally block.");
            }

        }

        static public List<QuickLaunch> parsBindFile()
        {
            List<QuickLaunch> quickLaunchesList = new List<QuickLaunch> { };
            try
            {
                if (!System.IO.File.Exists(path))
                {
                    FileStream fstream = new FileStream(path, FileMode.Create);
                    fstream.Close();
                }


                var lines = File.ReadAllLines(path).ToList();

                foreach (string line in lines)
                {
                    string[] temp = line.Split(new char[] { '|' });
                    QuickLaunch quickLaunch = new QuickLaunch(temp[0], temp[1]);
                    quickLaunchesList.Add(quickLaunch);
                }
            }
            catch (Exception ex)
            {

            }
            return quickLaunchesList;
        }

        static public void setFileSettings(bool autorun)
        {
            try
            {
                StreamWriter settingsFile = new StreamWriter(settingsPath);
                if (autorun.Equals(true))
                    settingsFile.WriteLine("true");
                else
                    settingsFile.WriteLine("false");

                settingsFile.Close();
            }
            catch (Exception ex) { Console.WriteLine("setFileSettings()" + ex.Message); }
        }

        static public bool parseAutorunSettings()
        {
            if (!System.IO.File.Exists("Settings.bin"))
            {
                setFileSettings(false);
            }

            try
            {
                StreamReader streamReader = new StreamReader(settingsPath);

                if (streamReader.ReadLine().Equals("true"))
                    return true;
                else
                    return false;
            }
            catch (Exception ex)
```

```
            {
                Console.WriteLine("Error parseAutorunSettings()" + ex.Message);
            }
            return false;

        }
    }
}
```

QuickLaunch.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ButtonBox
{
    class QuickLaunch
    {
        private string signal;
        private string filepath;


        public QuickLaunch( string signal, string filepath)
        {
            this.filepath = filepath;
            this.signal = signal;
        }

        public string Filepath { get => filepath; set => filepath = value; }
        public string Signal { get => signal; set => signal = value; }
    }
}
```

WindowManager.cs

```
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace ButtonBox
{
    class WindowManager
    {
        [DllImport("user32.dll", EntryPoint = "FindWindow", SetLastError = true)]
        static extern IntPtr FindWindow(string lpClassName, string lpWindowName);
        [DllImport("user32.dll", EntryPoint = "SendMessage", SetLastError = true)]
        static extern IntPtr SendMessage(IntPtr hWnd, Int32 Msg, IntPtr wParam, IntPtr lParam);

        const int WM_COMMAND = 0x111;
        const int MIN_ALL = 419;
        const int MIN_ALL_UNDO = 416;

        public static void closeWindowsOpenTM(bool taskmng)
        {
            IntPtr lHwnd = FindWindow("Shell_TrayWnd", null);
            // свернуть
            SendMessage(lHwnd, WM_COMMAND, (IntPtr)MIN_ALL, IntPtr.Zero);
            //System.Threading.Thread.Sleep(2000);
            // развернуть
            // SendMessage(lHwnd, WM_COMMAND, (IntPtr)MIN_ALL_UNDO, IntPtr.Zero);
            if (taskmng)
                Process.Start("taskmgr");
        }
    }
}
```

WindowsPower.cs

```
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Windows;

namespace ButtonBox
{
    class WindowsPower
    {
        [DllImport("Powrprof.dll", CharSet = CharSet.Auto, ExactSpelling = true)]
        public static extern bool SetSuspendState(bool hiberate, bool forceCritical, bool disableWakeEvent);
        // Hibernate
        static public void doHibernate()
        {
            SetSuspendState(true, true, true);
        }
        static public void doStandby()
        {
            WindowManager.closeWindowsOpenTM(false);
            MessageBoxResult result = MessageBox.Show("Sleep mode?", "ButtonBox", MessageBoxButton.YesNo);
            switch (result)
            {
                case MessageBoxResult.Yes:
```

```
                    SetSuspendState(false, true, true);
                    break;
                case MessageBoxResult.No:
                    break;
            }
        }

        static public void doShutdown()
        {
            WindowManager.closeWindowsOpenTM(false);
            MessageBoxResult result = MessageBox.Show("Shutdown PC?", "ButtonBox", MessageBoxButton.YesNo);
            switch (result)
            {
                case MessageBoxResult.Yes:
                    var psi = new ProcessStartInfo("shutdown", "/s /t 0");
                    psi.CreateNoWindow = true;
                    psi.UseShellExecute = false;
                    Process.Start(psi);

                    break;
                case MessageBoxResult.No:
                    break;
            }
        }
    }
}
```

WindowsSound.cs

```
using NAudio.CoreAudioApi;
using NAudio.Mixer;
using NAudio.Wave;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Management.Automation.Runspaces;
using System.Reflection;

namespace ButtonBox
{
    class WindowsSound
    {

        static public bool muteMicro()
        {
            try
            {
                for (int deviceIndex = 0; deviceIndex < NAudio.Wave.WaveIn.DeviceCount; deviceIndex++)
                {
                    var device = WaveIn.GetCapabilities(deviceIndex);
                }

                var waveInEvent = new NAudio.Wave.WaveInEvent();

                //получить миксер аудио устройства ввода по умолчанию
                var mixer = waveInEvent.GetMixerLine();
                var muteControl = mixer.Controls.FirstOrDefault(x => x.ControlType == NAudio.Mixer.MixerCon-
trolType.Mute) as BooleanMixerControl;
                // вкл/выкл
                if (muteControl.Value.Equals(true))
                {
                    muteControl.Value = false;
                    return false;
                }
                else
                {
                    muteControl.Value = true;
                    return true;
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error muteControlMicro()" + ex.Message);
            }
            return false;
        }

        static public void OutputDevicetoToChangeToFile()
        {
            try
            {
                MMDeviceEnumerator _DeviceEnumerator = new MMDeviceEnumerator();
                var enumerator = new MMDeviceEnumerator();
                var devices = _DeviceEnumerator.EnumerateAudioEndPoints(DataFlow.Render, DeviceState.Active);
```

```
                var AudioOutDefaultDevice = _DeviceEnumerator.GetDefaultAudioEndpoint(DataFlow.Render,
Role.Multimedia);
                List<string> names = new List<string>();

                StreamWriter fileDevList = new StreamWriter("OutputDeviceList.bin");
                foreach (var device in devices)
                {
                    fileDevList.WriteLine(device.ID.ToString());

                }
                fileDevList.Close();
                StreamWriter currDev = new StreamWriter("CurrOutputDevice.bin");
                currDev.WriteLine(AudioOutDefaultDevice.ID.ToString());
                currDev.Close();

                _DeviceEnumerator.Dispose();
                enumerator.Dispose();
                AudioOutDefaultDevice.Dispose();
            }
            catch (Exception e)
            {
                Console.WriteLine("Error OutputDevicetoToChangeToFile() " + e.Message);
            }

        }

        static public void outputDevice()
        {
            OutputDevicetoToChangeToFile();
            try
            {
                string[] devList = File.ReadAllLines("OutputDeviceList.bin");

                StreamReader fileCurrDev = new StreamReader("CurrOutputDevice.bin");
                string currDev = fileCurrDev.ReadLine();
                fileCurrDev.Close();

                int next = 0;

                for (int i = 0; i < devList.Length; i++)
                {
                    if (devList[i].Equals(currDev))
                    {
                        next = i + 1;
                        if (next >= devList.Length)
                        {
                            next = 0;
                            break;
                        }
                        break;
                    }
                }

                StreamWriter fileCurrDevWR = new StreamWriter("CurrOutputDevice.bin");
                fileCurrDevWR.WriteLine(devList[next]);
                fileCurrDevWR.Close();

                InitialSessionState iss = InitialSessionState.CreateDefault();
                iss.ImportPSModule(new string[]
                {
                     Path.Combine(Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location), "Audi-
oDeviceCmdlets.dll")
                });

                Runspace runspace = RunspaceFactory.CreateRunspace(iss);
                runspace.Open();

                Pipeline pipeline = runspace.CreatePipeline();

                Command command_set = new Command("Set-AudioDevice");

                CommandParameter param_set = new CommandParameter("ID", devList[next]);
                command_set.Parameters.Add(param_set);
                pipeline.Commands.Add(command_set);
                var results = pipeline.Invoke();

            }
            catch (Exception e)
            {
                Console.WriteLine("Error outputDeviceChange() " + e.Message);
            }
        }
    }
}
```

WindowsVolume.cs

```
using System;
using System.Runtime.InteropServices;
```

```
namespace ButtonBox
{
    class WindowsVolume
    {
        private const byte VK_VOLUME_MUTE = 0xAD;
        private const byte VK_VOLUME_DOWN = 0xAE;
        private const byte VK_VOLUME_UP = 0xAF;
        private const UInt32 KEYEVENTF_EXTENDEDKEY = 0x0001;
        private const UInt32 KEYEVENTF_KEYUP = 0x0002;

        [DllImport("user32.dll")]
        static extern void keybd_event(byte bVk, byte bScan, UInt32 dwFlags, UInt32 dwExtraInfo);

        [DllImport("user32.dll")]
        static extern Byte MapVirtualKey(UInt32 uCode, UInt32 uMapType);
        public static void VolumeUp()
        {
            keybd_event(VK_VOLUME_UP, MapVirtualKey(VK_VOLUME_UP, 0), KEYEVENTF_EXTENDEDKEY, 0);
            keybd_event(VK_VOLUME_UP, MapVirtualKey(VK_VOLUME_UP, 0), KEYEVENTF_EXTENDEDKEY | KEYEVENTF_KEYUP,
0);
        }

        public static void VolumeDown()
        {
            keybd_event(VK_VOLUME_DOWN, MapVirtualKey(VK_VOLUME_DOWN, 0), KEYEVENTF_EXTENDEDKEY, 0);
            keybd_event(VK_VOLUME_DOWN, MapVirtualKey(VK_VOLUME_DOWN, 0), KEYEVENTF_EXTENDEDKEY |
KEYEVENTF_KEYUP, 0);
        }

        public static void Mute()
        {
            keybd_event(VK_VOLUME_MUTE, MapVirtualKey(VK_VOLUME_MUTE, 0), KEYEVENTF_EXTENDEDKEY, 0);
            keybd_event(VK_VOLUME_MUTE, MapVirtualKey(VK_VOLUME_MUTE, 0), KEYEVENTF_EXTENDEDKEY |
KEYEVENTF_KEYUP, 0);
        }
    }
}
```

ПРИЛОЖЕНИЕ Е
(обязательное)

Блок-схема алгоритма

ПРИЛОЖЕНИЕ Ж
(обязательное)

Перечень элементов

ПРИЛОЖЕНИЕ З
(обязательное)

Ведомость документов