Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Схемотехника

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

МИКРОПРОЦЕССОРНОЕ УСТРОЙСТВО ИЗМЕРЕНИЯ ПАРАМЕТРОВ ЗВУКОВОГО СИГНАЛА

БГУИР КП 1-40 02 01 119 ПЗ

Студент: группы 950501,
Поддубный Д. П.

Руководитель: доцент каф. ЭВМ
Селезнев И. Л.

Минск 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Обработка сигналов является обширной областью радиотехники, в которой осуществляются задачи по восстановлению, разделению информационных потоков, подавлению шумов, сжатию данных и усилению сигналов. Для осуществления данных операций требуется большое количество математичеких методов, которые описывают преобразования сигналов при помощи технических устройств.

Сигналы деляться на аналоговые и цифровые. В данном курсовом проекте при помощи устройства на базе микроконтроллера будет анализироваться звук, который является аналоговым сигналом.

У звука существует множество параметров таких как высота, длительность, громкость, спектральный состав, интенсивность, тембр. Однако, как правило звук является совокупностью нескольких гармонических колебаний окружающей среды, а потому часто вышеперечисленные характеристики не являются информативными для анализа данного явления.

Для комплексного анализа звука существует спектрограмма, которая позволяет охарактеризовать каждую составляющую звука по отдельности.

Существует множество вариантов спектрограмм как двумерных, так и трехмерных. Более того существуют различные представления спектрограмм, амплитуда и частота могу быть как линейными, так и логарифмическими, в зависимости от графика.

В данном курсовом проекте будет разработан спекторграф, который показывает уровень сигнала во множестве частотных полос.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Состав устройства

Так как устройство служит для построения спектрограммы звука в реальном времени, то в нем необходим звуковой вход для передачи сигналы на схему, аналогово-цифровой преобразователь для превращения аналогового сигнала в цифровой, а также микроконтроллер, который служит для анализа преобразованного сигнала, построения спектрограммы и вывода её на экран.

1.2 Микроконтроллеры

Выбор был сделан в пользу готовых плат со втроенными микроконтроллерами, так как это значительно упрощает сборку схемы. Обязательным условием было наличие возможности для подключения аналого-цифрового преобразователя, так как он является важнейшей частью схемы. В качестве вариантов рассматривались платы Raspberry Pi Zero, а также Arduino Nano v3.0. Было произведено подробное сравнение функционала этих плат на основе источников [1] и [2]. Также было произведено подробное сравнение микроконтроллеров, которые встроены в данную схему (см. таблицу 1.1) на основе источников [3] и [4]

Таблица 1.1 – Сравнение микроконтроллеров

Параметр\Микроконтроллер	Broadcom BCM2835	ATmega328
Напряжение питания, В	3,3	5
Ток потребления, мА	100	90
Тактовая частота, МГц	1000	16
Разрядность, бит	32	8
Встроенный АЦП	да	нет

Несмотря на то, что Broadcom BCM2835 является полноценным процессором и включает в себя большие возможности расширения, и поддерживает полноценное ядро Linux, выбор был сделан в пользу платы на базе ATmega328, так как она имеет встроенный аналогово цифровой преобразователь, что является крайне важным условием для данного проекта, а также имеет более низкую цену.

1.3 Датчики и модули

Датчик – это элемент измерительного, сигнального, регулирующего или управляющего устройства, преобразующий контролируемую величину (температуру, давление, частоту, силу света, электрическое напряжение, ток и

т.д.) в сигнал, удобный для измерения, передачи, хранения, обработки, регистрации, а иногда и для воздействия им на управляемые процессы. Или проще, датчик – это устройство, преобразующее входное воздействие любой физической величины в сигнал, удобный для дальнейшего использования [5].

В данном проекте основным является датчик аналогово-цифрового преобразователя встроенный в микроконтроллер ATmega328. Аналогово-цифровой преобразователь - устройство, перобразующее входной аналоговый сигнал в дискретный код (цифровой сигнал).

В микроконтроллере ATmega328 имеется 10 разрядный АЦП, который висит на порту C. Выходное напряжение конвентируется в 10 разрядное двоичное значение. Минимальное значение соотвествует 0, а максимальное опорному напряжению. В данном микроконтроллере имеется 4 вида опорных напряжений, которые переключаются определённым битом при настройке АЦП. Полученный результат преобразования записывается в 2 регистра ADCH и ADCL. [6]

1.4 Обработка сигнала

Для обработки преобразованного аналогового сигнала в цифрой в реальном времени используется быстрое преобразование Фурье. Данный алгоритм является оптимальным для вычисление дискретного преобразования Фурье, работает за $O(n^2)$ и позволяет в реальном времени обрабатывать сигнал и раскалдывать его на различные синусоиды. [7]

В данном проекте использовалась реалзиция алгоритмы быстрого преобразования Фурье из библиотеки ArduinoFFT. [8]

1.5 Обзор аналогов конечных устройств

Существует большое количество аналогов устройств, которые позволяют анализировать сигнал и строить на его основе спектрограмму в реальном времени. Как правило данную функцию встраивают во все профессиональные эквалайзеры, например KlarkTeknik DN60 [9] и в программы для профессиональной обработки звука, такие как Audacity и Adobe Audition.

Все аналоги также используют быстрое преобразование Фурье для построения спектрограмм.

2 РАЗРАБОТКА СТРУКТУРЫ УСТРОЙСТВА

2.1 Постановка задачи

Необходимо разработать микропроцессорное устройсво для измерения параметров звука. Функции, возлагаемые на данное устройство:

- преобразование аналогового сигнала в цифровой;
- построение спектрограммы;
- вывод спектрограммы на дисплей;
- переключение режимов работы дисплея;

Управление устройством включает себя управление режимами отображения спектрограммы на дисплей.

2.2 Определение компонентов структуры устройства

Компоненты структуры устройства выбираются исходя из функций, определенных в постановке задачи. Были определены следующие компоненты (см. приложение А):

1) Аналогово-цифровой преобразователь – компонент, который принимает на входе аналоговый сигнал, переводит его в цифровой и передает микроконтроллеру.
2) Микроконтроллер – модуль, который анализирует полученный цифровой сигнал, строит спектрограмму в реальном времени, выводит информацию на дисплей, а также контроллирует режимы работы дисплея.
3) Дисплей – модуль, который служит для отображения спектрограммы.

2.3 Взаимодействие компонентов устройства

Аналоговый сигнал поступает на аудиовход, после чего преобразовывается в цифровой сигнал и передается микроконтроллеру.

Микроконтроллер анализирует сигнал и строит спектрограмму, после чего передает её на дисплей. По нажатию кнопки могут меняться режимы работы дисплея:

1) «Режим 1» – отображает сигнал полнный сигнал на определнной частоте
2) «Режим 2» - отображает только верхнюю точку сигнала на определенной часоте
3) «Режим 3» - отображает только верхнюю и нижнюю точки сигнала на определенной частоте
4) «Режим 4» - отображает верхнюю точку, и 3 нижние точки сигнала
5) «Режим 5» - работает аналогично первому режиму, но перевернут на 180 градусов

3 ОБОСНОВАНИЕ ВЫБОРА УЗЛОВ, ЭЛЕМЕНТОВ ФУНКЦИОНАЛЬНОЙ СХЕМЫ УСТРОЙСТВА

3.1 Обоснование выбора микроконтроллера

В таблице 1.1 показаны основные различия плат на базе ATmega328 и Broadcom BCM2835. Как и сказано ранее плата на базе Broadcom BCM2835 является куда более мощной и многофункциональной, а также имеет множество модулей расширения, более того, данная плата поддерживает sd карты и имеет больше оперативной памяти, однако не имеет встроенного аналогово-цифрового преобразователя, что является наиболее важной частью проекта. Также, платы на основе ATmega328 имеют более низкую цену.

3.2 Обоснование выбора датчиков

Так как аналогово-цифровой преобразователь встроен в плату Arduino Nano на базе микроконтроллера ATmega328, которая используется в проекте, то взаимодействие с ним также происходит через Arduino. Настройка аналогово-цифрового преобразователя происходит програмно (см. приложение Д). Настройка происходит следующим образом:

1) Необходимо настроить регистр ADMUX (Регистр настройки мультиплексора АЦП).

2) Настройки регистра ADCSRA (Регистр статуса и контроля А)

3) При необходимости, настройки ADCSRB (Регистр статуса и контроля В)

4) Чтение результата преобразования

3.3 Проектирование питания

Перед выбором источника питания был проведен расчет потребляемой мощности каждого модуля (см. таблицу 3.1).

Таблица 3.1 – Расчет потребляемой мощности

Имя устройства	Напряжение питания, В	Максимальный ток потребления, мА	Мощность, мВт
ATmega328	5	90	450
Каскадный дисплей	5	20	100
Суммарная мощность, Вт			0.55

Для схемы, разрабатываемой в курсовом проекте, необходимо обеспечить 1 уровня входного напряжения 5 В.

Питание будет подаваться на плату Arduino Nano, которая, в свою очередь будет запитывать матрицу дисплей, а также подавать питание

Рассматривалось несколько вариантов подачи питания через Arduino Nano:

1) При питании платы от USB – максимальный ток 500 мА
2) При питании платы в Vin – максимальный ток 2 А при Vin 7V, 500 мА при Vin 12V
3) При питании платы в 5V – максимальный ток зависит от блока питания

Был выбран вариант питания через USB в качестве самого универсального. Это позволяет питать схему почти от любого современного устройства.

3.4 Взаимодействие узлов

В приложении Б приведена функциональная схема устройства. Все модули подключены к микроконтроллеру с помощью определенных интерфейсов.

Аналогово цифровой преобразователь интегрирован в ATmega328p, AREF подается с пина 3V3.

Питание дисплея подключается через пины Arduino Nano 5V, а данные передаются через пины D10, D11, D13.

4 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ УСТРОЙСТВА

4.1 Разработка схемы питания

Остальные элементы берут питание от платы Arduino через соответсвующие пины. Одним из важнейших элементов устройства является светодиодная матрица, которая управляется встроенным в её плату микроконтроллером MAX7219. Данная матрица требует напряжение в 5 вольт, которое берет с пина Arduino Nano 5V.

4.2 Работа с Аналогово-цифровым преобразователем

Аналогово-цифровой преобразователь является устройством, которое преобразует входной аналоговый сигнал в цифровой. Каждый аналогово-цифровой преобразователь хараектризуется разрешением . Разрешение аналогово-цифрового преобразователя – это минимальное изменение величины аналогового сигнала, которое может быть преобразовано данным устройством и определяется разрядностью самого аналогово-цифрового преобразователя. В случае платы Arduino Nano, в нее уже встроен данный преобразователь, как было упомянуто выше. Он имеет разрядность 10 бит, что означает, что полученный сигнал может быть преобразован в числовое значение от 0 до 1023, которое позже будет обработано микропроцессором.

Также частота дискретизации аналогово-цифрового преобразователя является крайне важной характеристикой. Аналоговый сигнал является непрерывной функцией времени, который преобразуется в последовательность числовых значений. Частота, с которой аналогово-цифровой преобразователь генерирует числовые значения на основе аналогового сигнала. Диапазон слышемого звука доходит до 20кГц, значит для дискретизации такого аналогового сигнала, согласно Теореме Найквиста-Шеннона-Котельникова, по которой любой аналоговый сигнал может быть восстановлен по своим дискретным отсчетам, взятыми с частотой $f > 2x$, где x – максимальная частота, которая ограничена спектром реального сигнала. В микропроцессоре ATmega 328P встроен аналогово-цифровой преобразователь с частотой дискретизации до 38.64 КГц, что позволяет обрабатывать аналоговый сигнал с частотой до 19.32 КГц, что почти полностью покрывает спектр слышимого звука.

По сути аналово-цифровой преобразователь является аналоговым компаратором, а потому ему нужен эталонный сигнал, на основе которого он будет производить дискретизацию. Данный сигнал подается от стабилизированного источника питания 3,3 В с пина 3V3 платы Arduino Nano через резистор 5 кОм на пин AREF, который АЦП и использует как эталонный сигнал для сравнения.

4.3 Обработка аналогового сигнала с входа

Получаемый аналоговый колеблется выше и ниже нулевого уровня напряжения, поэтому необходимо смещение постоянного тока, что будет гарантировать, что выход аналово-цифрового преобразователя не будет ограничивать циклы входного сигнала, поэтому стабилизированное напряжение 3,3В (такое же подается на AREF пин АЦП) делиться двумя резисторами на 100кОм и затем подается на аналоговый вход для смещения постоянного тока. Напряжение, которое подается на аналоговый вход вычисляется по следующей формуле:

$$U\text{вход} = U\frac{R_1}{R_1 + R_2}$$

Получаем:

$$U\text{вход} = 3{,}3\ \text{В} * \frac{100\ \text{кОм}}{100\ \text{кОм} + 100\ \text{кОМ}} = 1.65\ \text{В}$$

Благодаря данному смещению на входе будет подаваться 512 бит даже при отсутвии сигнала, поэтому они будут вычитаться программною

Также на каждом канале аудиовхода стоят разделительные конденсаторы, которые служат для того, чтобы запретить протекание постоянного тока, между определенными точками цепи и обеспечить свободное прохождение переменного тока.

4.4 Подключение модулей

Обработанный аналоговый сигнал поступает на пин A0 платы Arduino Nano, после чего обрабатывается, при помощи сравнения с эталлоным сигналом, который поступает на пин AREF.

Светодиодная матрица получает сигналы CS, DIN и CLK с пинов D10, D11 и D13 соответсвенно платы Arduino Nano.

5 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

5.1 Требования к программе

Разрабатываемая программа должна позволять выполнять все функции, указанные в п. 2.1. Для этого программа должна реализовывать следующий функционал:

- Инициализация аналогово-цифрового преобразователя.
- Дискретизация преобразованного сигналаю
- Отрисовка полученной спектрограммы
- Обновление дисплея
- Изменение режима дисплея

Так как для реализации проекта была выбрана плата Arduino Nano на базе микроконтроллера ATmega328P, то алгоритм работы программы и её библиотеки написаны на языке программирования C в спцеильной среде разработки ArduinoIDE

5.2 Блок-схема программы

В приложении Г приведена схема программы.

Программа состоит из двух основных частей:

- инициализация;
- цикл обработки сигнала;

В начале программы происходит инциализация аналогово-цифрового преобразователя и объявление массивов чисел, которые определяют режимы работы дисплея, после чего запускается цикл, в котором аналоговый сигнал, который поступает на плату, преобразуется и по нему строиться спектрограмма при помощи библиотеки ArduinoFFT, которая является наиболее эффективной, быстрой и точной реализацией алгоритма быстрого преобразования Фурье для платы Arduino Nano.

5.3 Управление работой устройства

При нажатии на кнопку на устройстве сигнал подается на пин D5 платы Arduino Nano и переключает режим работы дисплея, который описан программно (см. Приложение- Д)

ЗАКЛЮЧЕНИЕ

В процессе разработки были получены и закреплены навыки в схемотехнике и проектировании устройств.
В ходе данного курсового проекта было разработано устройство, которое позволяет в реальном времени обрабатывать аналоговый сигнал в диапазоне частот слышимого звука и выводить данные на светодиодную матрицу. Устройство преобразует получаемый сигнал для корректной обработки. Данное устройство имеет довольно низкую стоимость и энергопотребление из-за грамотно подобранных компонентов. Недостатоком является низкое качество сборки, так как для данного устройство оптимальным является вариант с распайкой всех компонентов на одной плате. В дальнейшем планируется сделать данное устройство частью процессора для обработки звука вместе с ламповым усилителем. В процессе разработки данного устройства были получены навыки проектирования устройств для обработки звука, а также изучено и усвоено большое количество материала по обработке аналогового сигнала и преобразованию его в цифровой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1]. Руководство пользователя Arduino Nano v2.3. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf – Дата доступа: 16.09.2021

[2]. Документация Raspberry Pi. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html – Дата доступа: 16.09.2021

[3]. Документация Atmel ATmega328P [Электронный ресурс] – Электронные данные. – Режим доступа: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf – Дата доступа: 16.09.2021

[4]. Документация BCM2835 [Электронный ресурс] – Электронные данные. – Режим доступа: https://www.raspberrypi.org/app/uploads/2012/02/BCM2835-ARM-Peripherals.pdf – Дата доступа: 16.09.2021

[5]. Электротехническая энциклопедия [Электронный ресурс] – Электронные данные. – Режим доступа: http://www.electrolibrary.info/subscribe/sub_16_datchiki.htm – Дата доступа: 16.09.2021

[6]. Описание АЦП в ATmega328p [Электронный ресурс] – Электронные данные. – Режим доступа: https://sites.google.com/site/100voltsamper/mikrokontroller-razbor-poleetov/acp-v-atmega328p-pod-kakim-sousom-ego-neobhodimo-podavat – Дата доступа: 16.09.2021

[7]. Принцип построения алгоритмов быстрого преобразования Фурье [Электронный ресурс] – Электронные данные. – Режим доступа: https://ru.dsplib.org/content/fft_introduction/fft_introduction.html – Дата доступа: 16.09.2021

[8]. ArduinoFFT [Электронный ресурс] – Электронные данные. – Режим доступа: https://github.com/kosme/arduinoFFT – Дата доступа: 16.09.2021

[9]. Klarkteknik DN60 [Электронный ресурс] – Электронные данные. – Режим доступа: https://www.klarkteknik.com/product.html?modelCode=P0BPH – Дата доступа: 16.09.2021

ПРИЛОЖЕНИЕ А
(обязательное)

Схема структурная

ПРИЛОЖЕНИЕ Б
(обязательное)

Схема функциональная

ПРИЛОЖЕНИЕ В

(обязательное)

Схема принципиальная

ПРИЛОЖЕНИЕ Г
(обязательное)

Схема программы

ПРИЛОЖЕНИЕ Д
(обязательное)

Листинг кода

```
#include <arduinoFFT.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

#define SAMPLES 64            //Must be a power of 2
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW   // Set display type  so that  MD_MAX72xx library treets it properly
#define MAX_DEVICES  4   // Total number display modules
#define CLK_PIN   13  // Clock pin to communicate with display
#define DATA_PIN  11  // Data pin to communicate with display
#define CS_PIN    10  // Control pin to communicate with display
 #define  xres 32      // Total number of  columns in the display, must be <= SAMPLES/2
 #define  yres 8       // Total number of  rows in the display


 int MY_ARRAY[]={0, 128, 192, 224, 240, 248, 252, 254, 255}; // default = standard pattern
 int MY_MODE_1[]={0, 128, 192, 224, 240, 248, 252, 254, 255}; // standard pattern
 int MY_MODE_2[]={0, 128, 64, 32, 16, 8, 4, 2, 1}; // only peak pattern
 int MY_MODE_3[]={0, 128, 192, 160, 144, 136, 132, 130, 129}; // only peak +  bottom point
 int MY_MODE_4[]={0, 128, 192, 160, 208, 232, 244, 250, 253}; // one gap in the top , 3rd light onwards
 int MY_MODE_5[]={0, 1, 3, 7, 15, 31, 63, 127, 255}; // standard pattern, mirrored vertically


 double vReal[SAMPLES];
 double vImag[SAMPLES];
 char data_avgs[xres];

 int yvalue;
 int displaycolumn , displayvalue;
 int peaks[xres];
 const int buttonPin = 5;    // the number of the pushbutton pin
 int state = HIGH;             // the current reading from the input pin
 int previousState = LOW;   // the previous reading from the input pin
 int displaymode = 1;
 unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled
 unsigned long debounceDelay = 50;    // the debounce time; increase if the output flickers


 MD_MAX72XX mx = MD_MAX72XX(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);   // display object
 arduinoFFT FFT = arduinoFFT();                                    // FFT object



 void setup() {
```

```
33.  ADCSRA = 0b11100101;      // set ADC to free running mode and set pre-
   scalar to 32 (0xe5)
34.  ADMUX = 0b00000000;       // use pin A0 and external voltage reference
35.  pinMode(buttonPin, INPUT);
36.  mx.begin();           // initialize display
37.  delay(50);            // wait to get reference voltage stabilized
38.  }

39.  void loop() {
40.  // ++ Sampling
41.  for(int i=0; i<SAMPLES; i++)
42.  {
43.  while(!(ADCSRA & 0x10));        // wait for ADC to complete current
   conversion ie ADIF bit set
44.  ADCSRA = 0b11110101 ;               // clear ADIF bit so that ADC can
   do next operation (0xf5)
45.  int value = ADC - 512 ;                 // Read from ADC and subtract
   DC offset caused value
46.  vReal[i]= value/8;                     // Copy to bins after
   compressing
47.  vImag[i] = 0;
48.  }
49.  // -- Sampling


50.  // ++ FFT
51.  FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
52.  FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);
53.  FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);
54.  // -- FFT


55.  // ++ re-arrange FFT result to match with no. of columns on display (
   xres )
56.  int step = (SAMPLES/2)/xres;
57.  int c=0;
58.  for(int i=0; i<(SAMPLES/2); i+=step)
59.  {
60.  data_avgs[c] = 0;
61.  for (int k=0 ; k< step ; k++) {
          i.  data_avgs[c] = data_avgs[c] + vReal[i+k];
62.  }
63.  data_avgs[c] = data_avgs[c]/step;
64.  c++;
65.  }
66.  // -- re-arrange FFT result to match with no. of columns on display (
   xres )


67.  // ++ send to display according measured value
68.  for(int i=0; i<xres; i++)
69.  {
70.  data_avgs[i] = constrain(data_avgs[i],0,80);            // set max &
   min values for buckets
71.  data_avgs[i] = map(data_avgs[i], 0, 80, 0, yres);        // remap
   averaged values to yres
72.  yvalue=data_avgs[i];

73.  peaks[i] = peaks[i]-1;    // decay by one light
74.  if (yvalue > peaks[i])
75.         peaks[i] = yvalue ;
```

```
76.  yvalue = peaks[i];
77.  displayvalue=MY_ARRAY[yvalue];
78.  displaycolumn=31-i;
79.  mx.setColumn(displaycolumn, displayvalue);              // for left to
   right
80.  }
81.  // -- send to display according measured value

82.  displayModeChange ();         // check if button pressed to change
   display mode
83.  }

84.  void displayModeChange() {
85.  int reading = digitalRead(buttonPin);
86.  if (reading == HIGH && previousState == LOW && millis() -
   lastDebounceTime > debounceDelay) // works only when pressed

87.  {

88.  switch (displaymode) {
89.  case 1:    //       move from mode 1 to 2
90.  displaymode = 2;
91.  for (int i=0 ; i<=8 ; i++ )
92.          {MY_ARRAY[i]=MY_MODE_2[i];
93.  }
94.  break;
95.  case 2:    //       move from mode 2 to 3
96.  displaymode = 3;
97.  for (int i=0 ; i<=8 ; i++ ) {
      a. MY_ARRAY[i]=MY_MODE_3[i];
98.  }
99.  break;
100. case 3:    //     move from mode 3 to 4
101. displaymode = 4;
102. for (int i=0 ; i<=8 ; i++ ) {
103.         MY_ARRAY[i]=MY_MODE_4[i];
104. }
105. break;
106. case 4:    //     move from mode 4 to 5
107. displaymode = 5;
108. for (int i=0 ; i<=8 ; i++ )
109. {
110.         MY_ARRAY[i]=MY_MODE_5[i];
111. }
112. break;
113. case 5:    //      move from mode 5 to 1
114. displaymode = 1;
115. for (int i=0 ; i<=8 ; i++ ) {
116.         MY_ARRAY[i]=MY_MODE_1[i];
117. }
118. break;
119. }

120. lastDebounceTime = millis();
121. }
122. previousState = reading;
123. }
```

ПРИЛОЖЕНИЕ Е
(обязательное)

Ведомость документов

ПРИЛОЖЕНИЕ Ж
(обязательное)

Перечень элементов